DevSecOps stands for **Development, Security, and Operations** — it's a practice that **integrates security into every stage** of the software development and delivery process

1. **Core Idea**
Security is everyone's responsibility — developers, testers, and operations teams all work together to build secure software from the start.

2. **Key Principles**
Shift Left: Test for security early in the SDLC.
Automation: Integrate security tools into CI/CD pipelines.
Continuous Monitoring: Keep checking for new vulnerabilities even after deployment.
Collaboration: Break silos between Dev, Sec, and Ops teams.

3. Common Tools

SAST (Static Analysis): Scans source code for vulnerabilities (e.g., SonarQube, Checkmarx).

DAST (Dynamic Analysis): Tests running apps for flaws (e.g., OWASP ZAP, Burp Suite).

SCA (Software Composition Analysis): Detects vulnerable dependencies (e.g., Snyk, Dependabot).

Secrets Scanning: Finds hardcoded secrets (e.g., TruffleHog, GitGuardian).

4. Goals

Find and fix security issues early.

Automate security checks in pipelines.

Ensure compliance and reduce risks.

Static Application Security Testing

SAST analyzes source code, bytecode, or binaries without executing them to find security vulnerabilities early in the development phase. It's called "static" because it scans code at rest — not running applications.

It Runs early in the SDLC, typically:

While writing code (IDE plugins)

During build time in CI/CD pipelines

It Detects

Injection flaws (SQL, Command, LDAP, etc.)

Hardcoded credentials

Insecure API use

Cross-site scripting (XSS)

Unvalidated inputs / unsafe deserialization

Weak cryptography usage

Dynamic Application Security Testing

DAST tests an application while it’s running — like an external hacker would. It doesn’t look at source code; instead, it sends simulated attacks to find security issues in real time. It's a kind of penetration testing. (OWASP ZAP, Burp Suite)

In a CI/CD pipeline:

Code → SAST → Build → Deploy to test → DAST → Deploy to prod

It Detects:

SQL Injection

Cross-Site Scripting (XSS)

Authentication / Session issues

Server misconfigurations

Insecure headers / redirects

Broken access controls

Its benefits are:

Tests real-world behavior of your app.

Finds runtime vulnerabilities missed by SAST.

Doesn’t need source code (good for black-box testing).

Set up OWASP ZAP

You can use it in headless/CLI mode via Docker — easiest for CI/CD.

docker pull owasp/zap2docker-stable

Add a DAST Stage in the Pipeline

After your app is built and deployed to a test environment, add a ZAP scan stage.

Example flow:

Build → Deploy to test → Run ZAP scan → Report results

Run ZAP in Docker

docker run --rm -v $(pwd):/zap/wrk:rw owasp/zap2docker-stable \ zap-baseline.py -t https://your-test-app-url.co -r zap_report.html

Automate Thresholds

You can use the -J option (JSON output) and parse results to:

Fail the build if risk level > threshold.