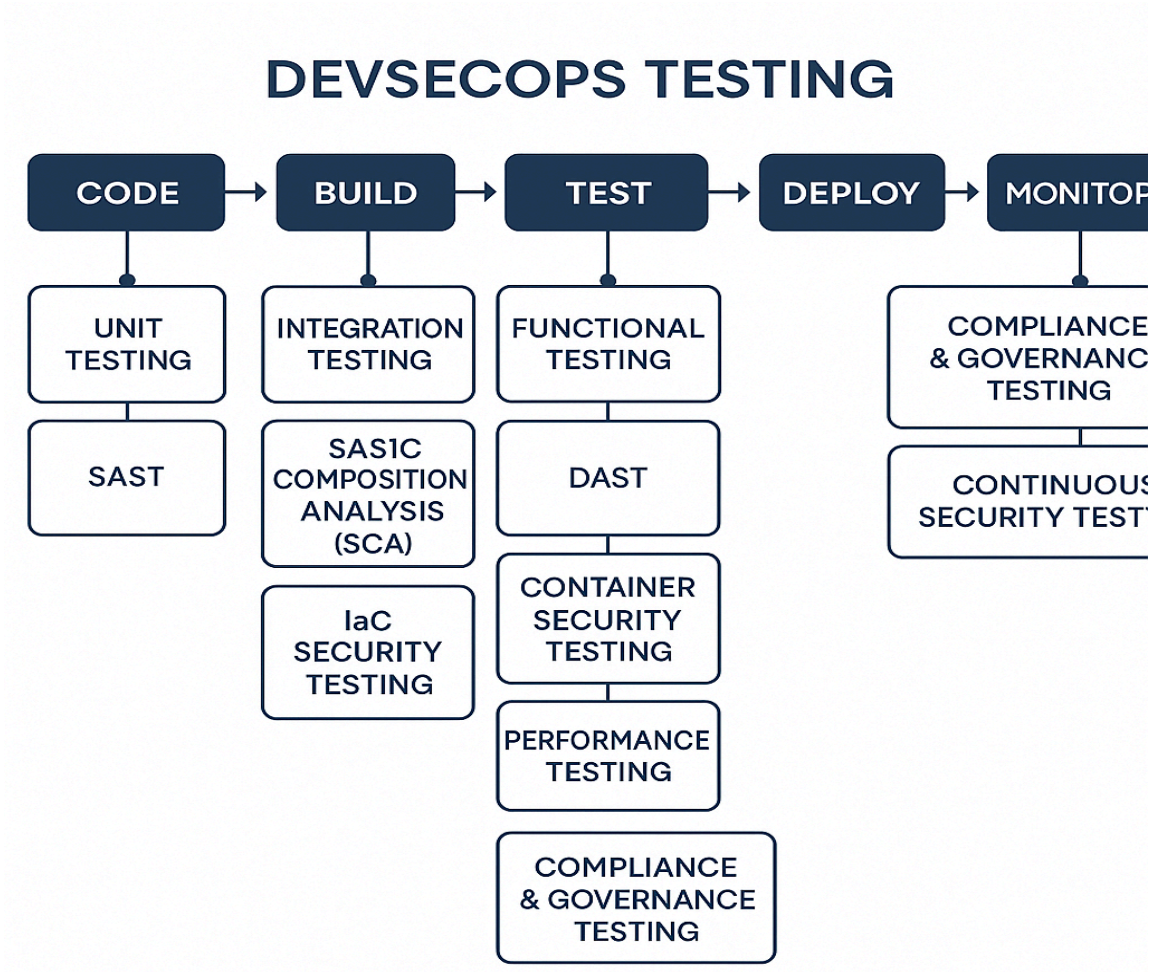

Types of Testing

In **DevSecOps**, testing isn't just about functionality — it includes **security and compliance** from the start.
Testing occurs **continuously** across the **CI/CD pipeline**, integrating both **automated** and **manual** validation at every phase (from code commit to production).



🧩 **1. Unit Testing**

Purpose: Validate individual units or components of code (functions, classes, modules).
Goal: Ensure each part works as intended in isolation.

Tools:

- JUnit (Java)
- NUnit (.NET)
- pytest (Python)
- Mocha (Node.js)

Stage: Developer stage — integrated into CI pipelines.

🧱 2. Integration Testing

Purpose: Verify that different modules or services work together correctly.
Goal: Detect interface defects between integrated components.

Tools:

- Postman / Newman (API integration)
- JUnit/TestNG (integration tests)
- Docker Compose for service-level integration

Stage: After successful unit testing, during CI.

🌐 3. Functional Testing

Purpose: Validate end-to-end functionality of the application against business requirements.
Goal: Ensure the system behaves as expected.

Tools:

- Selenium / Cypress
- Robot Framework
- TestComplete

Stage: Automated in CI/CD; often before staging or UAT deployments.

🧭 4. Regression Testing

Purpose: Verify that new code changes don't break existing features.
Goal: Maintain stability after updates.

Tools:

- Selenium Grid
- Jenkins (automated job triggers)
- Katalon / TestNG

Stage: Triggered automatically on every new build or deployment.

🧪 5. Performance Testing

Purpose: Evaluate system behavior under specific workloads (speed, stability, scalability).
Goal: Detect performance bottlenecks.

Types:

- Load Testing – normal load behavior
- Stress Testing – behavior under extreme conditions
- Soak Testing – stability over time

Tools:

- JMeter
- Gatling
- Locust

Stage: Pre-production or staging environments.

🔐 6. Static Application Security Testing (SAST)

Purpose: Analyze **source code** for security vulnerabilities **before execution**.
Goal: Detect issues like hardcoded secrets, SQL injection, and insecure dependencies early.

Tools:

- SonarQube

- Snyk Code
- Checkmarx
- Fortify

Stage: Code commit / build stage (shift-left security).

🧰 7. Dynamic Application Security Testing (DAST)

Purpose: Test the running application for security vulnerabilities in real-time.
Goal: Detect runtime issues like authentication flaws or injection attacks.

Tools:

- OWASP ZAP // Log4j
- Burp Suite
- Netsparker

Stage: After deployment to a test or staging environment.

📦 8. Software Composition Analysis (SCA)

Purpose: Identify vulnerabilities in open-source dependencies or third-party libraries.
Goal: Ensure all components comply with licensing and security policies.

Tools:

- Snyk

- OWASP Dependency-Check
- WhiteSource
- Black Duck

Stage: Build stage or early CI.

🧑‍💻 9. Container Security Testing

Purpose: Scan Docker/Kubernetes images for vulnerabilities or misconfigurations.
Goal: Secure container images before pushing to registry.

Tools:

- Trivy
- Clair
- Aqua Security
- Anchore

Stage: Pre-deployment (CI/CD and registry scan).

☁️ 10. Infrastructure as Code (IaC) Security Testing

Purpose: Validate Terraform, CloudFormation, or Ansible scripts for security compliance.
Goal: Prevent misconfigurations (open ports, public buckets, etc.).

Tools:

- Checkov
- Terrascan
- TFSec

Stage: Before provisioning infrastructure in CD.

🔄 11. Penetration Testing

Purpose: Simulate real-world attacks to identify exploitable weaknesses.
Goal: Evaluate system resilience against attackers.

Tools:

- Metasploit
- Burp Suite Pro
- Kali Linux tools

Stage: Periodically in pre-prod or post-deployment phases.

🧠 12. Compliance & Governance Testing

Purpose: Validate adherence to industry standards like **ISO 27001, PCI-DSS, HIPAA**.
Goal: Automate compliance checks and audits.

Tools:

- OpenSCAP
- Chef InSpec
- AWS Config Rules

Stage: Continuous monitoring and audit phase.

🧭 13. User Acceptance Testing (UAT)

Purpose: Ensure that the final product meets **business and user expectations**.
Goal: Final validation before production.

Tools:

- Manual testing / BrowserStack / LambdaTest
- Often integrated with test case management tools (e.g., TestRail)

Stage: Pre-production / staging environment.

🧩 14. Continuous Security Testing (DevSecOps Automation Layer)

Purpose: Continuously monitor and test code, dependencies, and configurations throughout the pipeline.
Goal: Enforce "**Shift Left + Shift Everywhere**" security.

Tools Integration Example:

- **Pre-commit:** SAST, Secrets scanning (e.g., Gitleaks)

- **Build:** SCA, IaC scan
- **Deploy:** DAST, container scan
- **Runtime:** SIEM tools (Splunk, ELK), monitoring with Falco or Prometheus

---

✅ **Summary Table**

Testing Type	Focus Area	Example Tools	Stage
Unit	Code correctness	JUnit, Pytest	Dev / CI
Integration	Module interaction	Postman, TestNG	CI
Functional	Business logic	Selenium, Cypress	CI/CD
Regression	Stability	Jenkins, TestNG	CI/CD
Performance	Load, Stress	JMeter, Gatling	Pre-prod
SAST	Code security	SonarQube, Snyk	CI
DAST	Runtime security	OWASP ZAP, Burp	Staging
SCA	Library security	Dependency-Check	CI
Container	Image security	Trivy, Clair	CD
IaC	Infra config	Checkov, Terrascan	CI/CD
Penetration	Exploit testing	Metasploit	Post-prod
Compliance	Policy adherence	Chef InSpec	Continuous

UAT	Business validation	Manual, BrowserStack	Staging