
🗒️ Checkmarx for DevOps – Comprehensive Setup & Integration Guide

---

🧠 1. Introduction

🔍 What is Checkmarx?

Checkmarx is a leading **Application Security Testing (AST)** platform designed for DevSecOps.
It enables **developers and DevOps teams** to identify and fix vulnerabilities in **source code**, **open-source libraries**, and **running applications** during the development lifecycle.

Checkmarx supports multiple scanning types:

Checkmarx Component	Description
CxSAST (Static Application Security Testing)	Scans source code for security vulnerabilities before compilation
CxSCA (Software Composition Analysis)	Detects vulnerabilities in open-source dependencies (like Snyk)
CxIAST (Interactive Application Security Testing)	Monitors applications during runtime testing
CxCodebashing	Developer training platform that teaches how to fix vulnerabilities

---

💡 2. Why Checkmarx in DevOps?

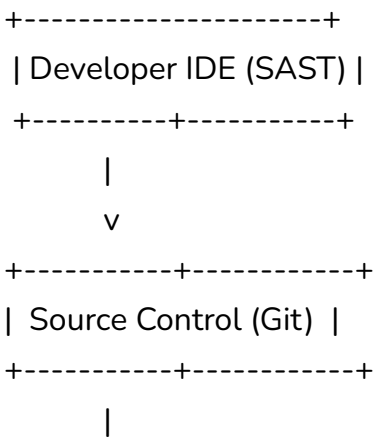
Traditional security testing occurs late in development, but **Checkmarx** integrates **security scanning early** ("shift-left security") within:

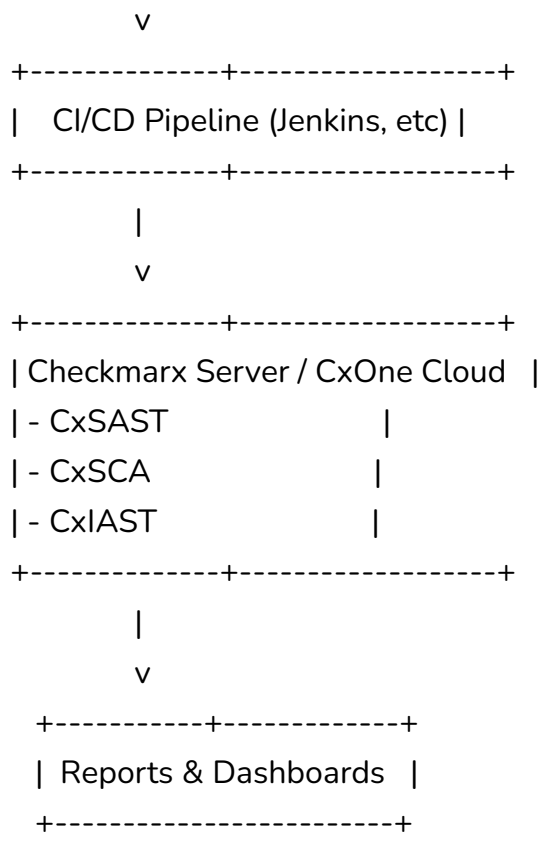
- **Source control** (GitHub, GitLab, Bitbucket)
- **Build pipelines** (Jenkins, Azure DevOps, GitHub Actions)
- **IDE tools** (Eclipse, IntelliJ, VS Code)

✅ **Benefits:**

- Early detection of code vulnerabilities
- Seamless CI/CD integration
- Actionable remediation advice
- Support for 30+ programming languages
- Governance, compliance, and reporting dashboards

🧱 3. Checkmarx Architecture Overview





🧰 4. Prerequisites

Requirement	Description
Checkmarx Account	Cloud or on-prem deployment
Admin Access	To configure projects and APIs
CI/CD Tool	Jenkins / GitHub Actions / GitLab CI
Language	Java, Node.js, Python, .NET supported

Network Access	Internet or internal access to Cx server
API Token	For CLI / automation authentication

🌐 5. Step-by-Step Setup (Checkmarx Cloud / On-Prem)

🧩 Step 1 – Access Checkmarx Portal

- Navigate to:
  👉 https://ast.checkmarx.net (for CxOne Cloud)
  or
  👉 **https:///CxWebClient/** (on-prem)

- Login with your organization credentials.

🧩 Step 2 – Create a Project

1. In the Checkmarx dashboard → Click **Projects** → **New Project**
2. Enter:
   - Project Name: my-java-app
   - Repository URL: https://github.com/ajacs/my-java-app.git
   - Branch: main
3. Choose **Scan Type:**

- Static Code Analysis (SAST)
- Software Composition Analysis (SCA)
- Optionally add IAST or SCA + SAST

🧩 **Step 3 – Configure SCM (Source Control Management)**

- Go to **Repositories** → **Add Repository**
- Select type (GitHub, Bitbucket, GitLab)
- Provide access token or credentials
- Grant **Read** permission for Checkmarx to clone the code

🧩 **Step 4 – Configure Scanning Settings**

1. Select programming language (e.g., Java, JavaScript)
2. Set **Scan Preset**:
   - "Default" (recommended)
   - "OWASP Top 10" or "CWE/SANS Top 25"
3. Set **Scan Trigger**:
   - On every push / pull request

- Scheduled (daily, weekly)

🧩 Step 5 – Start a Manual Scan

1. In the Project Dashboard → Click **Start Scan**
2. Wait for scan to complete (depends on project size)
3. Review scan report:
    - Vulnerabilities categorized by **severity** (High, Medium, Low)
    - CWE references and fix recommendations

🧩 6. Integrate Checkmarx with Jenkins (DevOps)

Step 1: Install Jenkins Plugin

1. Navigate to Manage Jenkins → Plugin Manager
2. Search for **Checkmarx Plugin**
3. Install and restart Jenkins

Step 2: Configure Checkmarx in Jenkins

Go to:

Manage Jenkins → Configure System → Checkmarx Configuration

Add:

- **Server URL**: https://ast.checkmarx.net
- **Username/Token**: API token from Checkmarx
- **Project Settings**: Default preset and team

Step 3: Create Jenkins Pipeline Job

Use a **Declarative Pipeline** with Checkmarx steps.

Example for a **Java App (Maven)**:

```
pipeline {
  agent any
  stages {
    stage('Checkout') {
      steps {
        git 'https://github.com/ajacs/my-java-app.git'
      }
    }
    stage('Build') {
      steps {
        sh 'mvn clean package'
      }
    }
    stage('Checkmarx Scan') {
      steps {
        checkmarxScan config: [
```

```
        serverUrl: 'https://ast.checkmarx.net',
        credentialsId: 'CHECKMARX_TOKEN',
        projectName: 'my-java-app',
        preset: 'Default',
        incremental: true
      ]
    }
  }
 }
}
```

🧩 7. Integrate Checkmarx with GitHub Actions

Create .github/workflows/checkmarx.yml:

```
name: Checkmarx Scan

on:
  push:
  pull_request:

jobs:
  checkmarx:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
      - name: Checkmarx Scan
        uses: CheckmarxDev/checkmarx-github-action@v2
        env:
          CX_SERVER: https://ast.checkmarx.net
          CX_TENANT: my-org
          CX_CLIENT_ID: ${{ secrets.CX_CLIENT_ID }}
          CX_CLIENT_SECRET: ${{ secrets.CX_CLIENT_SECRET }}
```

CX_PROJECT: my-java-app
CX_BRANCH: main

This triggers a **SAST + SCA scan** on every push or pull request.
Results appear both in **Checkmarx GUI** and **GitHub PR comments**.

🧩 8. Reviewing Scan Results

From the Checkmarx Dashboard:

- Navigate to your project
- Open **Scan Results**
- Review:
    - Vulnerability type (SQL Injection, XSS, Path Traversal, etc.)
    - Severity level (Critical, High, Medium, Low)
    - Source → Sink trace (where the issue starts and ends)
    - Recommended fix (code snippet or upgrade)

You can export reports as:

- **PDF**, **CSV**, **XML**, or **JSON**

🧩 9. Developer IDE Integration

Checkmarx supports:

- **IntelliJ IDEA**
- **Eclipse**
- **VS Code**

Example (IntelliJ):

1. Go to **Plugins** → **Marketplace** → **Checkmarx Plugin**
2. Install and restart IDE
3. Authenticate using your API token
4. Click **Run SAST Scan**
5. Vulnerabilities appear directly in your code editor, with fix suggestions

---

🔄 10. Continuous Monitoring

Checkmarx continuously:

- Monitors codebases for new vulnerabilities
- Alerts via **email**, **Slack**, or **Jira**
- Re-scans when:
    - Code changes (commit/push)

- New CVEs are published
- New Checkmarx rules are added

📊 11. Reporting and Governance

You can create:

- **Executive Reports**: summarize risk posture across all projects
- **Developer Reports**: show issues by severity or CWE
- **Compliance Reports**: OWASP, PCI DSS, ISO 27001 mappings

Reports can be automated and emailed weekly.

🧠 12. Best Practices for Checkmarx in DevOps

Area	Best Practice
Scan Frequency	Run SAST on every commit; SCA on release
Incremental Scans	Use incremental mode to speed up re-scans
Policy Enforcement	Fail builds if severity ≥ High
Training	Use Codebashing to train developers
Automation	Integrate with CI/CD and PR checks

False Positives	Mark verified safe code as “Not Exploitable”
Version Control	Store .checkmarx configuration files in repo

✅ 13. Validation Checklist

Step	Description	Status
Checkmarx Account Created	GUI accessible	✅
Project Added	Repo linked	✅
Jenkins or GitHub Integration	Configured	✅
Scan Triggered	Results visible	✅
Alerts Configured	Email or Slack	✅
Reports Generated	PDF / CSV	✅

🗒️ 14. Summary

- **Checkmarx** provides enterprise-grade static, dynamic, and dependency scanning for DevSecOps.
- It integrates seamlessly with **CI/CD pipelines**, **SCM**, and **IDEs**.

- Offers deep **code-to-vulnerability traceability** for faster remediation.
- Enables organizations to maintain **secure SDLC** compliance with **OWASP** and **CWE** standards.
- Core components include:
    - **CxSAST** – Static Code Scanning
    - **CxSCA** – Open-source Dependency Analysis
    - **CxIAST** – Runtime Testing
    - **CxCodebashing** – Developer Training
