



## Snyk GUI Setup & Scanning Java Application (Step-by-Step Guide)

---

### 1. What is Snyk? (Quick Recap)

Snyk is a **developer-first security platform** that helps identify and fix:

- Vulnerabilities in **open-source dependencies**
- Issues in **container images**
- Misconfigurations in **Infrastructure as Code (IaC)**
- Security flaws in **custom source code**

It integrates with:

- GitHub / GitLab / Bitbucket
  - CI/CD tools (Jenkins, GitHub Actions, etc.)
  - IDEs (IntelliJ, VS Code, Eclipse)
- 

### 2. Prerequisites

Requirement	Description
OS	Ubuntu / Windows / macOS

Java	JDK 8 or higher
Maven or Gradle	For Java project builds
Internet	Required for connecting to Snyk Cloud
Web Browser	For GUI access
Snyk Account	<a href="https://snyk.io">https://snyk.io</a> (Free signup)

---

### 3. Step 1 – Create a Snyk Account (GUI Setup)

1. Go to  <https://snyk.io>

2. Click **Sign up**

3. Choose a method:

- GitHub
- GitLab
- Bitbucket
- Email & Password

Once signed in, you'll be redirected to the **Snyk Dashboard**:

<https://app.snyk.io/>

4.

---

## 4. Step 2 – Connect Your Code Repository (GUI)

In the **Snyk Dashboard**, click on:

Projects → Add Projects

- 1.
2. Choose your **Source Control Platform** (e.g., GitHub)
3. Authorize Snyk to access your repositories
4. Select the **Java project repository**
5. Click **Add Selected Repositories**

 Snyk will:

- Clone your repo
- Detect `pom.xml` (Maven) or `build.gradle`
- Analyze dependencies and transitive libraries
- Display a vulnerability report

---

## 5. Step 3 – Scan Java Project Locally (CLI Integration)

Although the GUI does automatic scans, it's common to also scan locally or in CI/CD.

### Install Snyk CLI

```
npm install -g snyk
```

Verify:

```
snyk --version
```

---

### Authenticate the CLI with GUI

Run:

```
snyk auth
```

This opens your browser → log into your Snyk account → the CLI is now linked with your GUI dashboard.

---

### Test a Java Project Locally

Assuming your project has a `pom.xml` (Maven-based):

```
cd my-java-app  
snyk test
```

Sample output:

```
Testing /my-java-app...
```

- ✗ Medium severity vulnerability found in org.springframework:spring-core@5.2.0
  - Description: Denial of Service (DoS)
  - Info: <https://snyk.io/vuln/maven:org.springframework:spring-core:20191111>
  - Fix: Upgrade to version 5.2.3 or later

Tested 120 dependencies for known issues, found 4 issues.

---

## Monitor the Project (Send Report to GUI)

After testing locally, upload results to your dashboard for **continuous monitoring**:

```
snkyk monitor
```

You'll see the project appear in your GUI under:

Projects → Imported via CLI

---

## 6. Step 4 – Interpreting Results in Snyk GUI

In the web console:

- **Dashboard Overview:**

Shows all projects with vulnerability counts.

- **Project View:**

Displays:

- Vulnerability name and severity (Critical, High, Medium, Low)
- Dependency path (e.g., spring-core → commons-logging)
- Fix available (version upgrade or patch)
- CVSS score and CWE info

**Example:**

 Prototype Pollution (High)

Package: com.fasterxml.jackson.core:jackson-databind

Introduced through: spring-boot-starter-json@2.5.2

Fixed in: 2.13.4

- 
- 

## 7. Step 5 – Automatically Fix Vulnerabilities

You can use **Snyk Wizard** to automatically upgrade or patch dependencies:

`snyk wizard`

It:

- Interactively suggests dependency upgrades
  - Applies `.snyk` policy file for ignored issues
- 

## 8. Step 6 – Continuous Scanning and Alerts

Snyk continuously scans your projects:

- Whenever a new vulnerability is disclosed
- You'll receive email and dashboard alerts
- GitHub PRs will show **security warnings** if new issues are found

You can view:

- Trend graphs of vulnerability count over time

- Severity distributions
  - Auto-fix pull requests (if enabled)
- 

## 9. Optional: Snyk IDE Plugin (Java)

For real-time feedback while coding:

IntelliJ IDEA:

1. Open IntelliJ → [Settings](#) → Plugins
  2. Search: **Snyk Vulnerability Scanner**
  3. Install & restart IDE
  4. Log in to Snyk from IDE
  5. It highlights vulnerabilities in your dependencies live as you code.
- 

## 10. Optional: Snyk Integration with Jenkins (for Java build)

Example Jenkins pipeline snippet:

```
pipeline {  
    agent any  
    stages {  
        stage('Build') {  
            steps {  
                sh 'mvn clean install'  
            }  
        }  
    }  
}
```

```

}

stage('Security Scan') {
    steps {
        withCredentials([string(credentialsId: 'SNYK_TOKEN', variable: 'SNYK_TOKEN')]) {
            sh 'snyk auth $SNYK_TOKEN'
            sh 'snyk test --all-projects'
            sh 'snyk monitor'
        }
    }
}

```

---

## 11. Best Practices Notes

Area	Best Practice
<b>Authentication</b>	Always link CLI and GUI using <code>snyk auth</code>
<b>Scanning Frequency</b>	Run scans daily or before releases
<b>Policies</b>	Use <code>.snyk</code> file to ignore or patch selectively
<b>Remediation</b>	Prefer upgrades over ignoring
<b>Team Setup</b>	Assign projects to teams via Snyk Organizations
<b>Alerts</b>	Configure Slack or email notifications
<b>CI/CD</b>	Enforce scan results as build gates

---

## 12. Validation Checklist

Step	Description	Status
Snyk Account Created	GUI accessible	✓
Repository Linked	GitHub or local	✓
Java Project Detected	pom.xml found	✓
CLI Authenticated	snyk auth successful	✓
Scan Performed	snyk test output visible	✓
Dashboard Updated	Project appears in GUI	✓
Alerts Enabled	Email notifications configured	✓

---

### ✓ 13. Summary

- Snyk GUI offers **real-time monitoring** of vulnerabilities in your Java apps.
  - The **CLI tool** provides local and CI/CD scanning capabilities.
  - You can integrate Snyk with **GitHub, Jenkins, IntelliJ, and Maven** builds.
  - It's a **must-have tool** for DevSecOps pipelines ensuring security compliance before deployment.
-