
🧾 What is Snyk?

🔍 1. Introduction to Snyk

Snyk (pronounced “*sneak*”) is a **developer-focused security platform** that helps teams **find and fix vulnerabilities** in their:

- Application **source code**
- **Open-source dependencies**
- **Container images**
- **Infrastructure-as-Code (IaC)** configurations like Terraform or Kubernetes

💡 Core Philosophy:

> "Empower developers to secure their applications from the first line of code to production."

Unlike traditional security scanners that operate at the end of the DevOps pipeline, **Snyk shifts security left** — integrating into the **development phase** itself.

🧱 2. Why Snyk is Needed

Modern applications rely heavily on **third-party libraries and packages** — for example:

- A Node.js app might use 100+ npm packages.
- A Java app might include dozens of Maven dependencies.

⚠️ The challenge:
These open-source packages can contain **known vulnerabilities** that attackers exploit.

Traditional tools:

- Run late in the pipeline (e.g., after deployment)
- Require security expertise
- Produce long reports developers ignore

✅ **Snyk's solution:**

- Easy CLI and IDE integration
- Developer-friendly reports
- Direct remediation guidance
- Continuous scanning (even after deployment)

⚙️ 3. Snyk Platform Components

Component	Description
🧑‍💻 **Snyk Open Source (Snyk OSS)**	Scans dependencies (npm, Maven, pip, etc.) for known vulnerabilities using Snyk's vulnerability database.

🐳 **Snyk Container**	Analyzes Docker and OCI container images for OS-level vulnerabilities.
🏗️ **Snyk Infrastructure as Code (IaC)**	Scans Terraform, CloudFormation, and Kubernetes manifests for misconfigurations and security risks.
🧠 **Snyk Code**	Performs static application security testing (SAST) directly in your IDE or CI/CD.
🔄 **Snyk CI/CD Integrations**	Integrates with Jenkins, GitHub Actions, GitLab CI, Azure DevOps, etc., to automate scans.

🧰 4. How Snyk Works (Architecture Overview)

Step-by-Step Flow:

1. **Project Import**
   - Snyk connects to your source repository (e.g., GitHub, GitLab).
   - Reads dependency files (package.json, pom.xml, requirements.txt, etc.)
2. **Dependency Tree Generation**
   - Snyk builds a dependency graph showing direct and transitive dependencies.
3. **Vulnerability Detection**
   - It compares each dependency version against the **Snyk Vulnerability Database** (updated hourly).
4. **Reporting & Fixing**

  - Snyk generates a report with severity levels (Low, Medium, High, Critical).

  - Provides recommended versions or patches.

5. **Continuous Monitoring**

  - Snyk continuously scans your project and alerts you if a new CVE affects your dependencies.

🧩 Architecture Diagram (conceptually):

Developer → GitHub Repo → Snyk CLI/API → Snyk Cloud Platform → Vulnerability DB → Fix Suggestions → IDE / CI/CD Reports

---

💻 5. Snyk Features in Detail

🧩 1. Open Source Dependency Scanning

- Detects vulnerabilities in third-party libraries.

- Works for **npm, pip, Maven, Gradle, RubyGems, Go modules**, etc.

Example:

```
snyk test
```

- Reports known CVEs and severity.

🧩 2. Snyk Code (SAST)

- Scans your **own source code** (custom logic) for security flaws.
- Identifies SQL injection, XSS, hardcoded secrets, etc.

Runs inside VS Code, IntelliJ, or CLI.

```
snyk code test
```

-

🧩 3. Snyk Container

- Scans container images like node:18-alpine or custom-built images.

Detects OS-level vulnerabilities in layers.

```
snyk container test node:18-alpine
```

-

🧩 4. Snyk IaC (Infrastructure as Code)

- Checks for insecure cloud configuration.

Example: Public S3 buckets, open security groups, unencrypted EBS volumes.

```
snyk iac test
```

-

🧩 5. Continuous Monitoring

- snyk monitor uploads dependency snapshot to the Snyk dashboard.
- Sends email or Slack alerts if new vulnerabilities are discovered later.
- Keeps project security **up to date** over time.

🧠 6. How Snyk Detects Vulnerabilities

Snyk uses multiple data sources:

1. **National Vulnerability Database (NVD)**
2. **GitHub Security Advisories**
3. **Vendor Security Advisories**
4. **Snyk's proprietary research database**

Each vulnerability is assigned:

- **CVE ID (Common Vulnerabilities and Exposures)**
- **CVSS Score (Severity)**
- **Remediation Path**

Example:

```
✗ High severity vulnerability found in lodash
  CVE-2020-8203 | Prototype Pollution
```

Fixed in: 4.17.21
Recommendation: Upgrade lodash to 4.17.21

🚀 7. Integrations

Platform	Integration Type	Example
🧑‍💻 IDE	Developer IDE plugins	VS Code, JetBrains
☁️ Git	SCM Integration	GitHub, GitLab, Bitbucket
⚙️ CI/CD	Pipeline Integration	Jenkins, GitHub Actions
📦 Containers	Image Scanning	Docker Hub, Amazon ECR
☁️ Cloud	IaC and K8s scanning	Terraform, Helm, K8s manifests

🔧 8. Common Snyk CLI Commands

Command	Description
`snyk auth`	Authenticate with Snyk account
`snyk test`	Scan current project for vulnerabilities
`snyk monitor`	Upload project snapshot for continuous monitoring
`snyk wizard`	Interactive fixing tool
`snyk code test`	Analyze source code for security issues

snyk container test <image>	Scan Docker image
snyk iac test	Scan infrastructure-as-code files
snyk help	Display all available commands

📦 9. Example Workflow (Node.js Project)

```
# Install Snyk
npm install -g snyk

# Authenticate
snyk auth

# Run test
snyk test

# Automatically fix vulnerabilities
snyk wizard

# Monitor continuously
snyk monitor
```

🧩 10. Example Report Output

```
Testing /my-node-app...

✗ High severity vulnerability found in lodash
  Description: Prototype Pollution
  Info: https://snyk.io/vuln/SNYK-JS-LODASH-567746
  Introduced through: lodash@4.17.15
  Remediation: Upgrade to lodash@4.17.21
```

```
✔ No vulnerable paths for low severity vulnerabilities
Organization: ajacs-devops
Tested 42 dependencies for known issues, found 1 issue.
```

🛡️ 11. Snyk vs Other Tools

Feature	Snyk	SonarQube	Trivy	OWASP Dependency-Check
Focus	DevSecOps (dependencies, IaC, code)	Code quality + security	Containers & IaC	Dependency scanning
Scans Dependencies	✅	❌	✅	✅
Static Code Analysis	✅ (Snyk Code)	✅	❌	❌
Container Scanning	✅	❌	✅	❌
Continuous Monitoring	✅	❌	❌	❌
CI/CD Integration	✅	✅	✅	✅
Developer Friendly	✅	⚙️ Moderate	✅	❌

🟩 **Summary:**
Snyk is more **developer-centric and integrated** across the full software supply chain — from code to cloud.

📊 12. Key Benefits of Using Snyk

✅ **Shift Left Security:** Identify issues early in development.
✅ **Automated Fixes:** Auto-suggests version upgrades or patches.
✅ **Continuous Monitoring:** Alerts you even after code is merged.
✅ **Integration Everywhere:** Works with IDEs, SCM, CI/CD, and containers.
✅ **Detailed Insights:** Rich vulnerability details and remediation steps.
✅ **Developer Adoption:** Easy CLI, minimal configuration.

🚀 13. Typical Use Cases

- Scanning npm or Maven dependencies before deployment
- Validating Docker images in Jenkins pipelines
- Securing Terraform configurations for AWS/Azure
- Integrating security checks in GitHub pull requests
- Enforcing organizational compliance for open-source use

⚙️ 14. Limitations

Limitation	Description
Internet Required	CLI depends on Snyk API and DB
Free Plan Limits	Limited number of monitored projects

Enterprise Features	Snyk Code & Snyk Container advanced features need paid tiers
No Runtime Protection	Only pre-deployment scanning (not runtime defense)

🧠 15. Summary Notes

📌 **Snyk = Security for Developers.**
It bridges the gap between development speed and security assurance.

In Short:

- Detects → Prioritizes → Fixes vulnerabilities.
- Covers code, dependencies, containers, and cloud.
- Integrates across your SDLC (IDE → Git → CI/CD → Cloud).

🪄 Quick Recap Notes for Trainers

Topic	Key Point
Tool Type	Developer Security Platform
Usage	Find and fix vulnerabilities early
Components	OSS, Code, Container, IaC
Commands	`snyk test`, `snyk monitor`, `snyk code test`
Integration	GitHub, Jenkins, VS Code

Benefit	Continuous, developer-friendly security
Alternative	SonarQube (code quality focus)