

DevSecOps

Introduction

Name

Total Experience

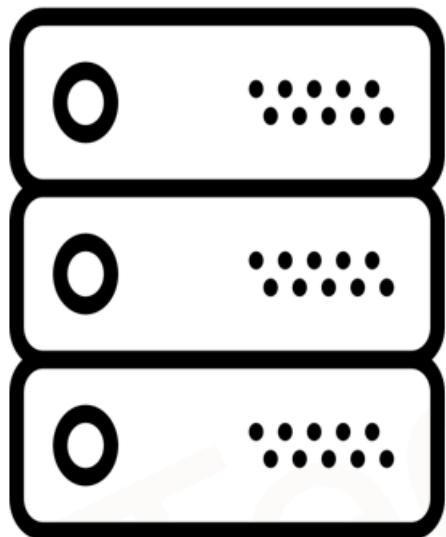
Background – Development / Infrastructure / Database / Network

Experience on monitoring tools

Your Expectations from this training

Module 1: Cloud Fundamentals & DevSecOps

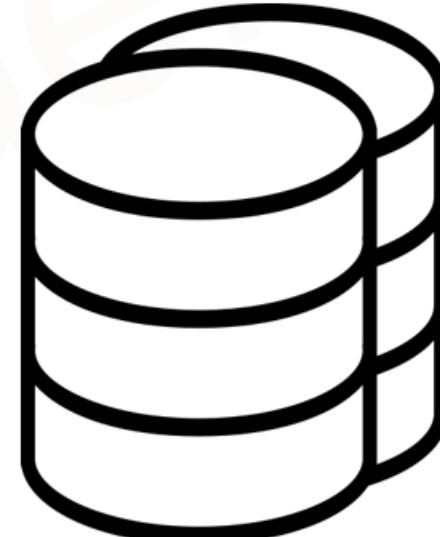
Traditional Datacenters



Servers

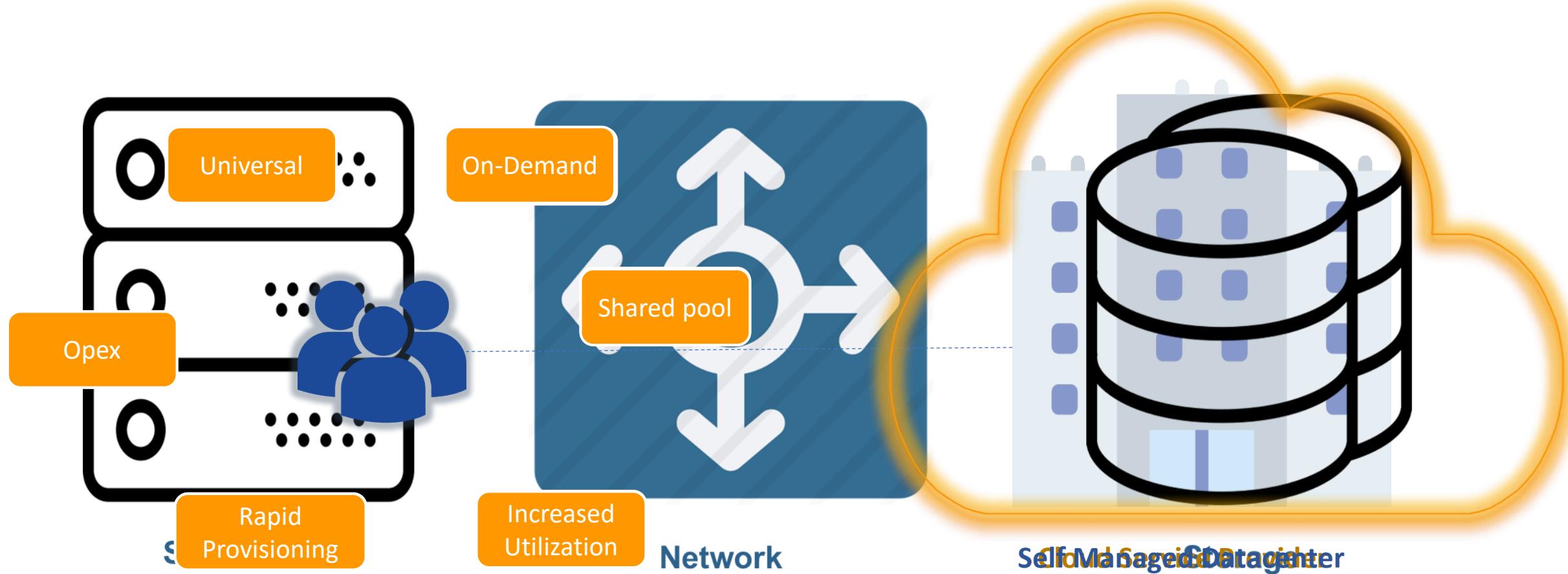


Network



Storage

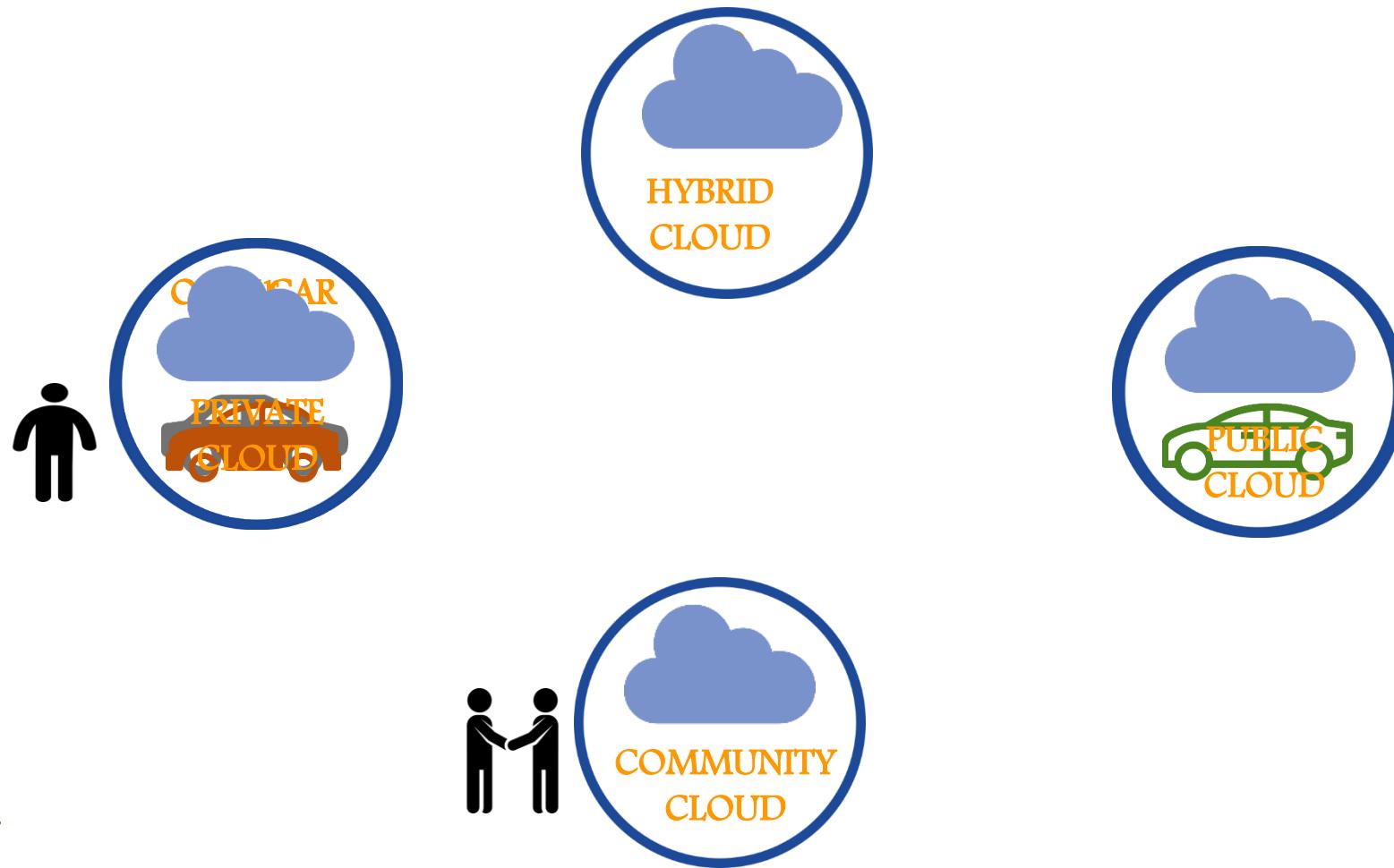
Traditional Datacenters



Characteristics of Cloud



Cloud Deployment Types



Service Models



Car Owned



Car Leased



Car Hired

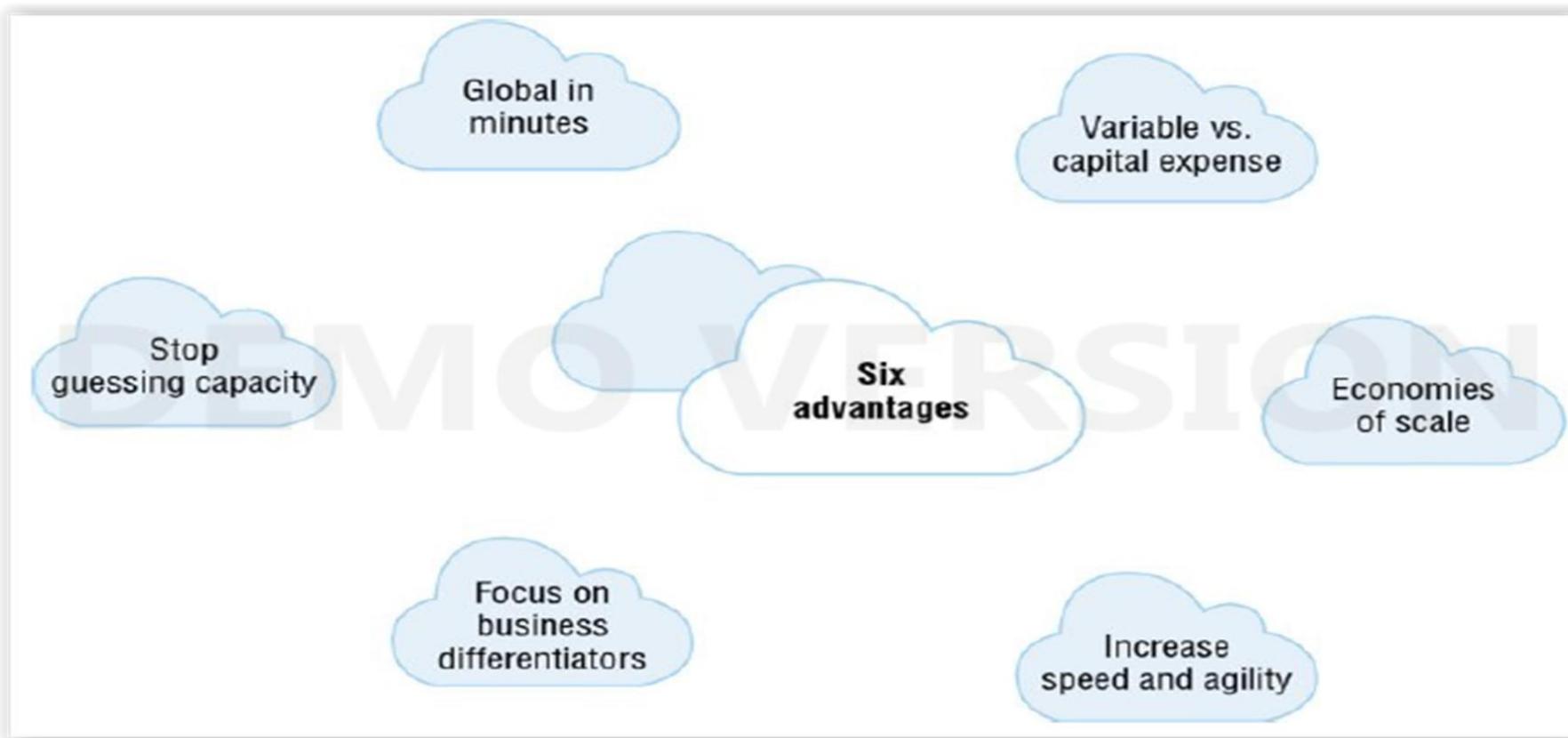


Taxi

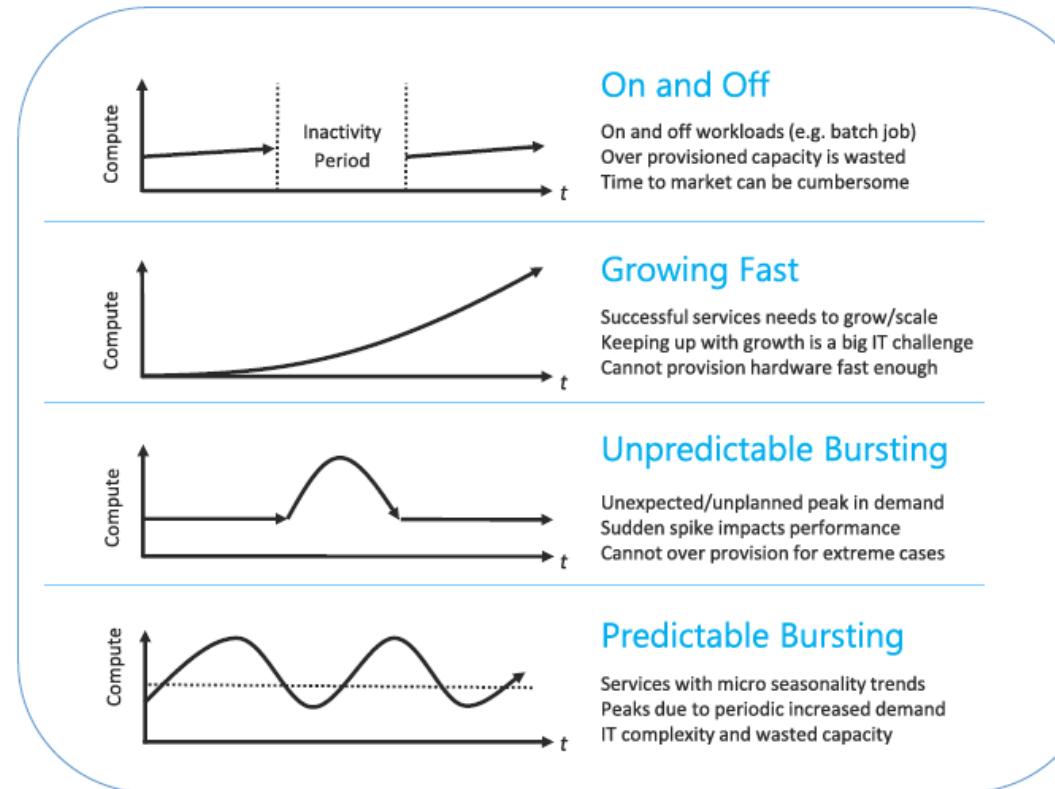
Service Models



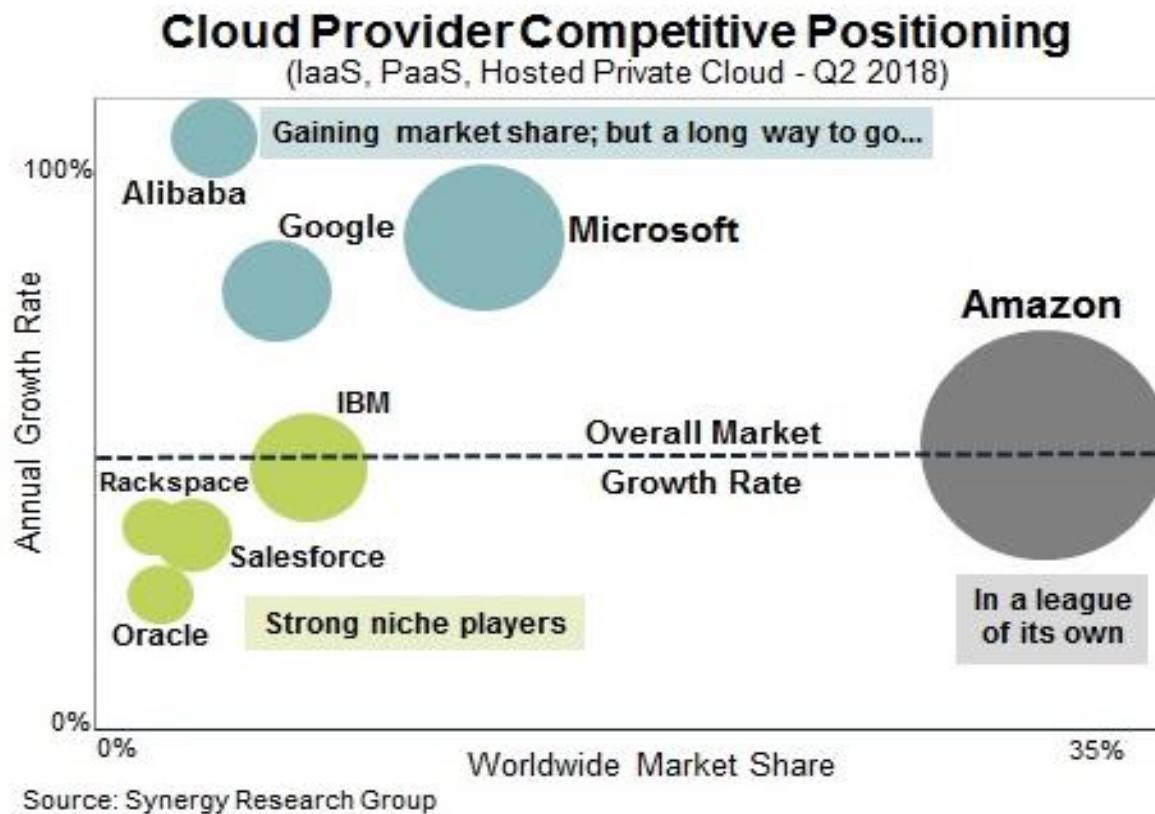
Cloud Benefits



Cloud Major Use Cases



Cloud Players



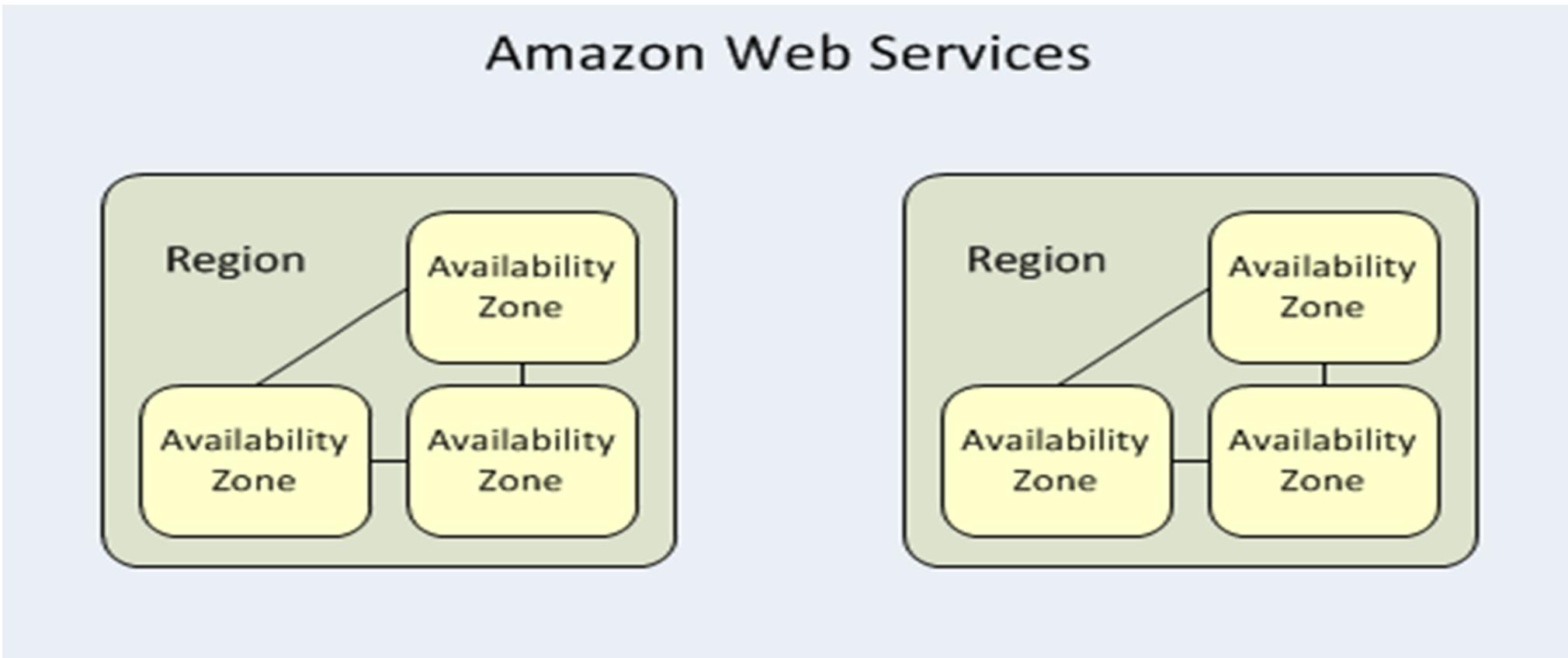
Amazon Web Services

- AWS (Amazon Web Services) is a group of web services (also known as cloud services) being provided by Amazon since 2006.
- AWS provides huge list of services starting from basic IT infrastructure like CPU, Storage as a service, to advance services like Database as a service, Serverless applications, IOT, Machine Learning services etc..
- Hundreds of instances can be build and use in few minutes as and when required, which saves ample amount of hardware cost for any organizations and make them efficient to focus on their core business areas.
- Currently AWS is present and providing cloud services in more than 190 countries.
- Well-known for IaaS, but now growing fast in PaaS and SaaS.

Why AWS

- **Low Cost:** AWS offers, pay as you go pricing. AWS models are usually cheapest among other service providers in the market.
- **Instant Elasticity:** You need 1 server or 1000's of servers, AWS has a massive infrastructure at backend to serve almost any kind of infrastructure demands, with pay for what you use policy.
- **Scalability:** Facing some resource issues, no problem within seconds you can scale up the resources and improve your application performance. This cannot be compared with traditional IT datacenters.
- **Multiple OS's:** Choice and use any supported Operating systems.
- **Multiple Storage Options:** Choice of high I/O storage, low cost storage. All is available in AWS, use and pay what you want to use with almost any scalability.
- **Secure:** AWS is PCI DSS Level1, ISO 27001, FISMA Moderate, HIPAA, SAS 70 Type II passed. In-fact systems based on AWS are usually more secure than in-house IT infrastructure systems.

AWS Regions and Availability Zones



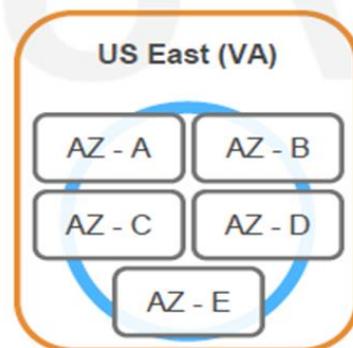
AWS Regions and Availability Zones

At least 2 AZs per region.

Examples:

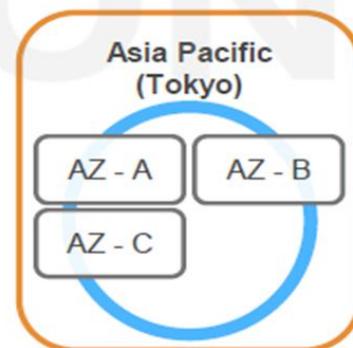
➤ US East (N. Virginia)

- us-east-1a
- us-east-1b
- us-east-1c
- us-east-1d
- us-east-1e



➤ Asia Pacific (Tokyo)

- ap-northeast-1a
- ap-northeast-1b
- ap-northeast-1c



Note: Conceptual drawing only. The number of Availability Zones (AZ) may vary.

AWS Regions and Availability Zones

- **AWS Regions:**
 - Geographic Locations
 - Consists of at least two Availability Zones(AZs)
 - All of the regions are completely independent of each other with separate Power Sources, Cooling and Internet connectivity.
- **AWS Availability Zones**
 - AZ is a distinct location within a region
 - Each Availability Zone is isolated, but the Availability Zones in a Region are connected through low-latency links.
 - Each Region has minimum two AZ's
 - Most of the services/resources are replicated across AZs for HA/DR purpose.

AWS Regions and Availability Zones

Current:

22 AWS Regions

69 AZs

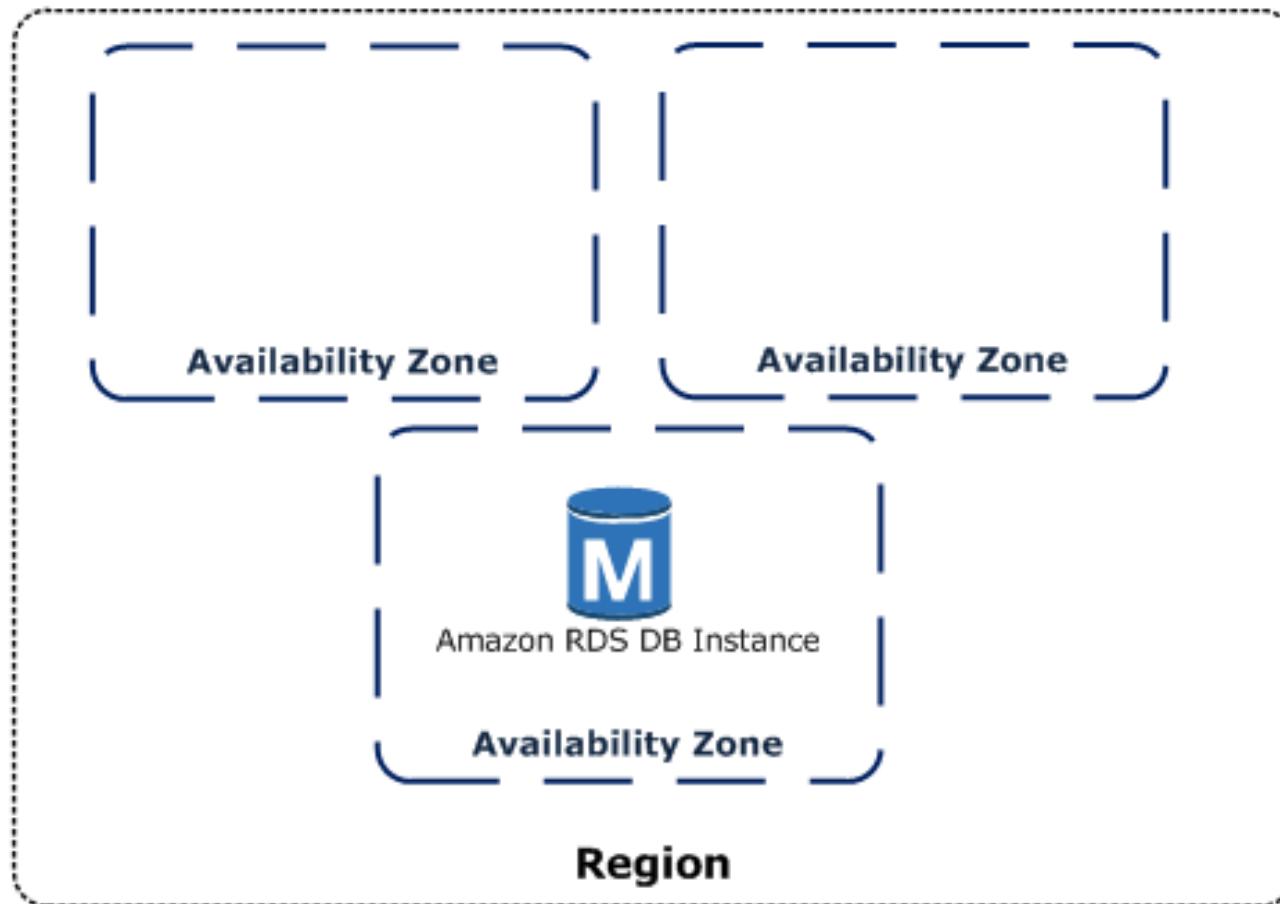
Upcoming:

4 Regions

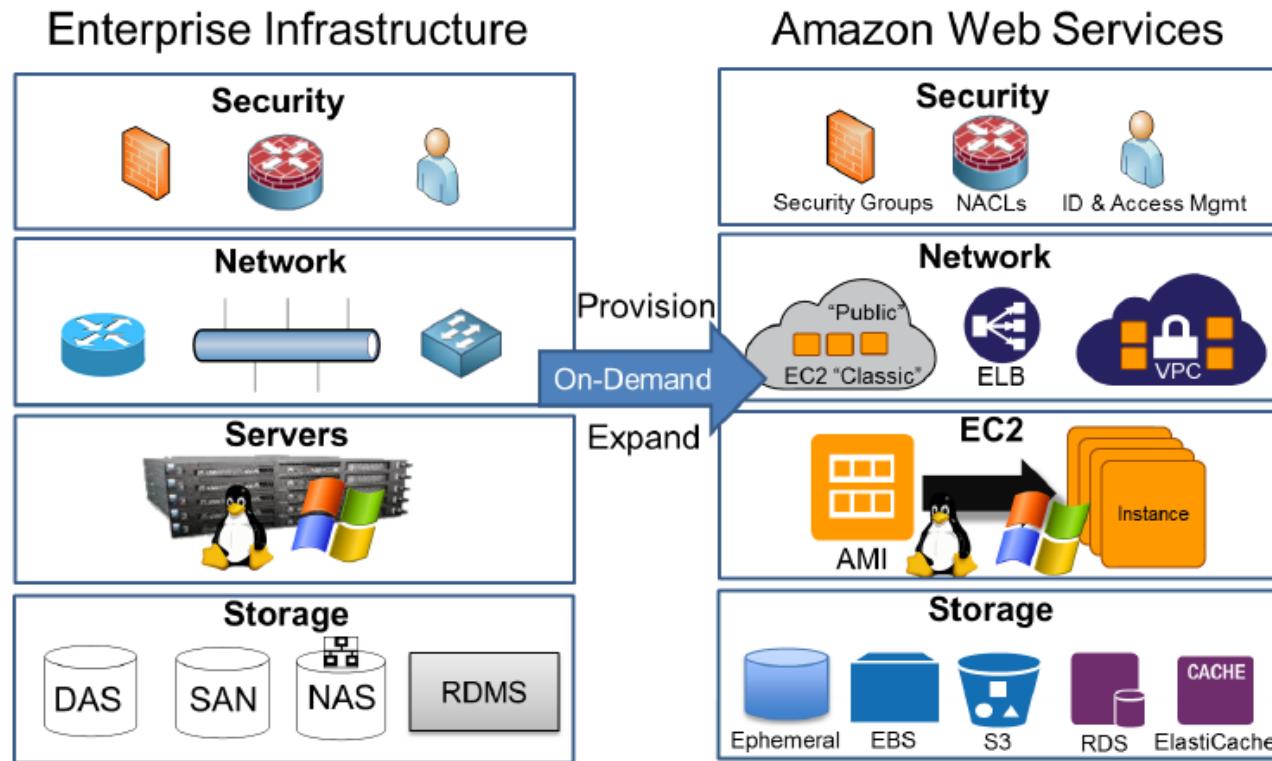
13 AZs



AWS Regions and Availability Zones



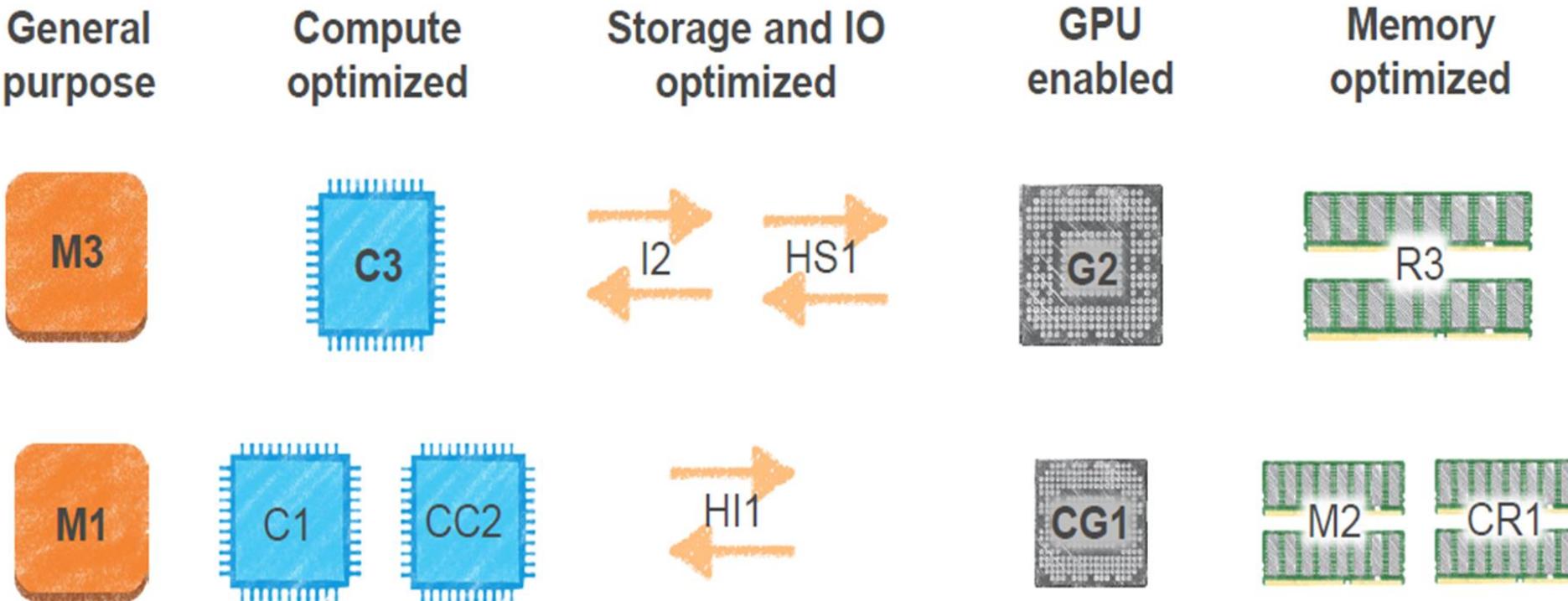
AWS



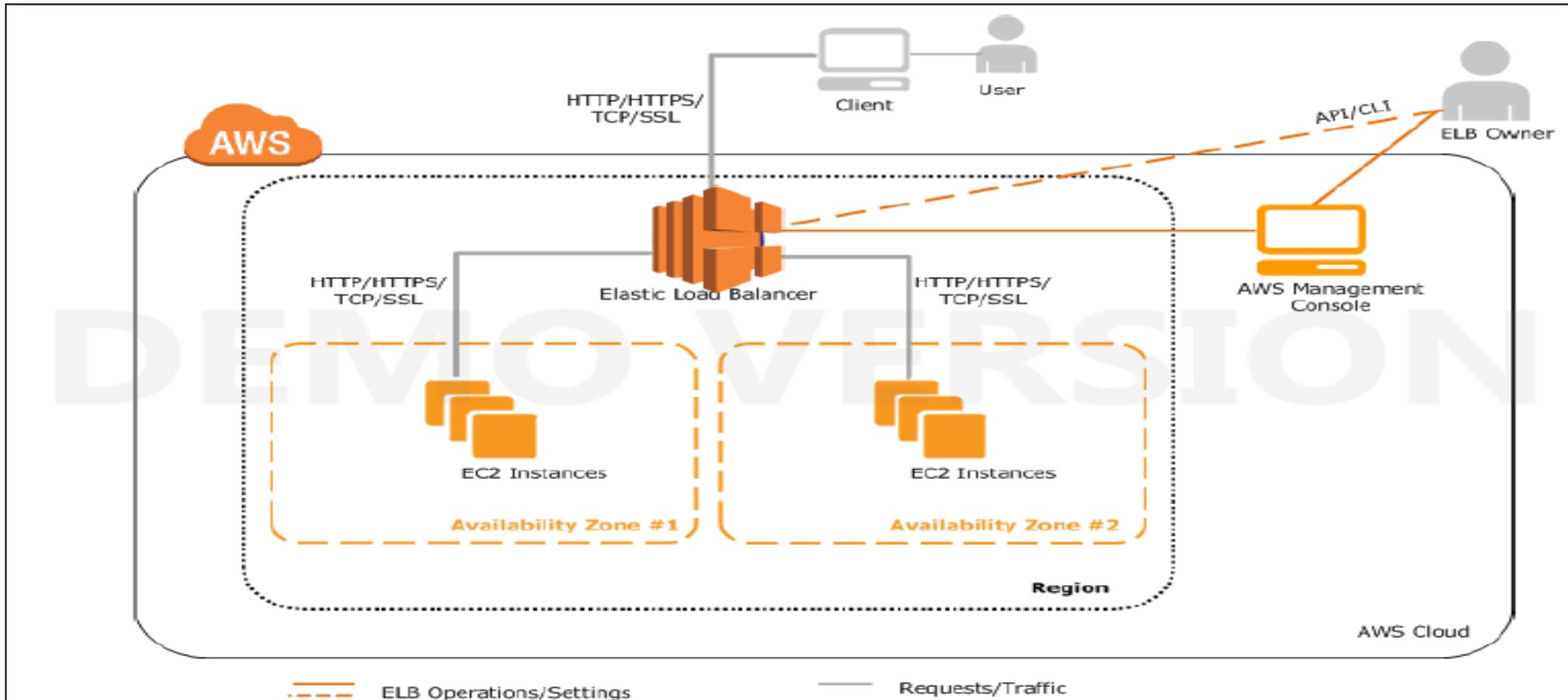
AWS Compute Services

- Amazon EC2 stands for Elastic Compute Cloud, and is the Primary AWS web service.
- Provides Resizable compute capacity
- Reduces the time required to obtain and boot new server instances to minutes
- There are two key concepts to Launch instances in AWS:
 - Instance Type
 - AMI
- EC2 Facts:
 - Scale capacity as your computing requirements change
 - Pay only for capacity that you actually use
 - Choose Linux or Windows OS as per need. You have to Manage the OS and Security of same.
 - Deploy across AWS Regions and Availability Zones for reliability/HA

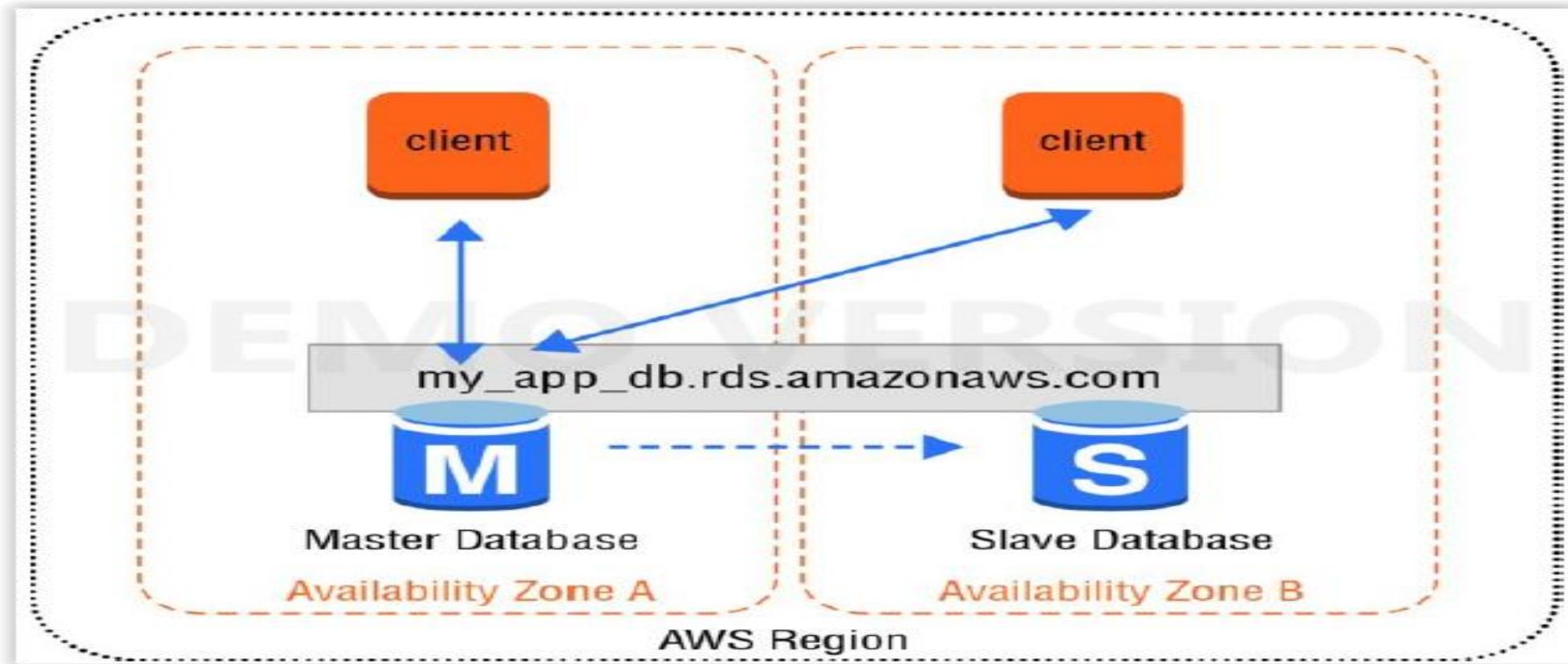
AWS EC2



ELB



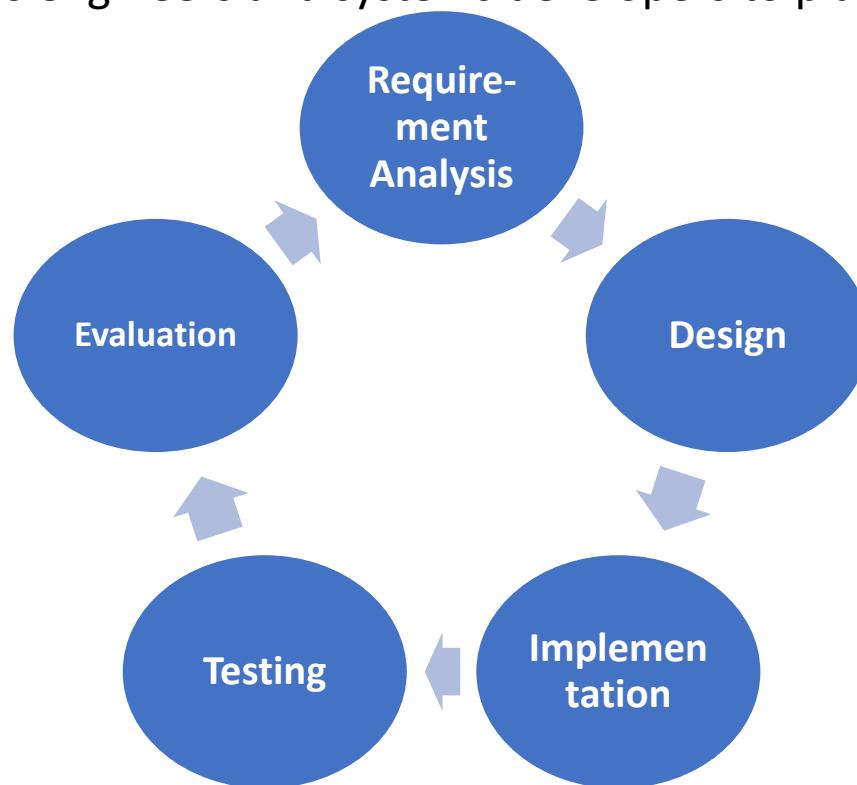
PAAS



What is DevOps

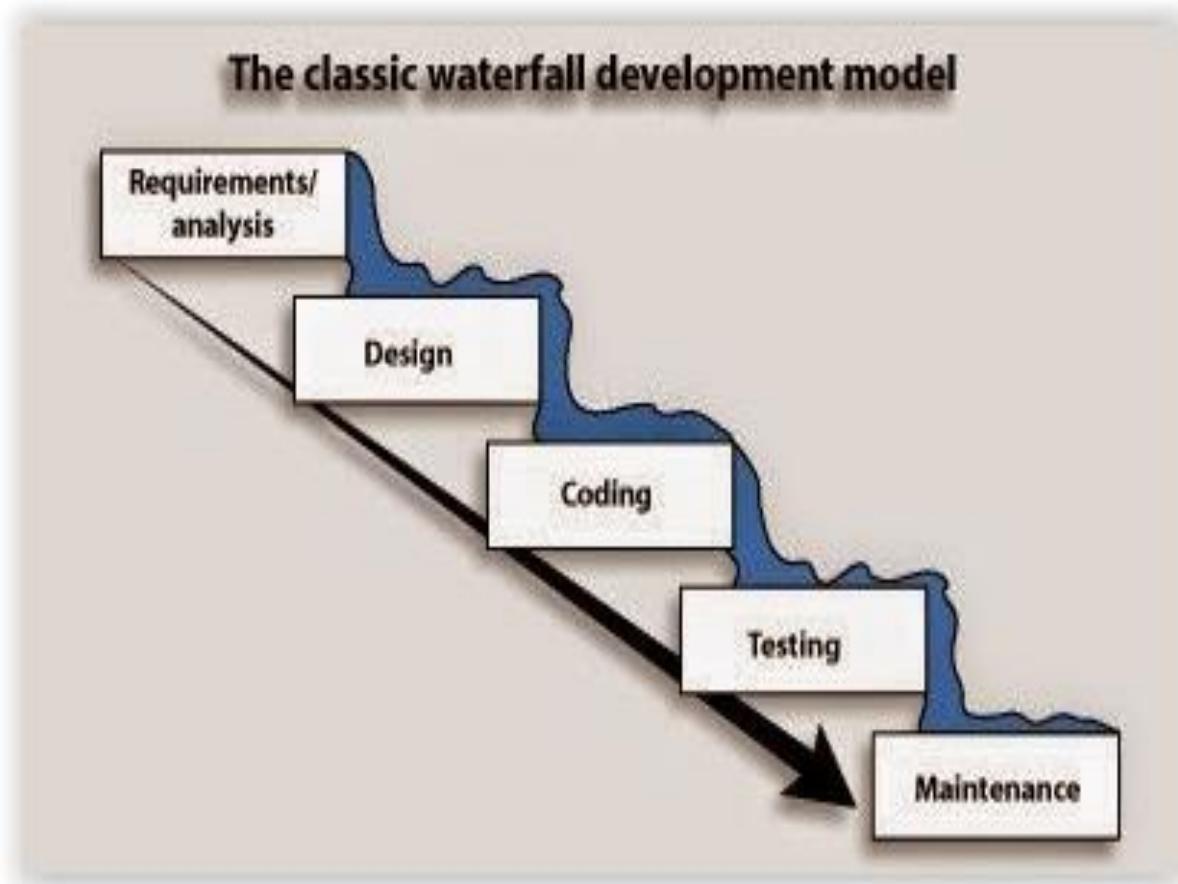
SDLC Model

- A systems development life cycle is composed of **several clearly defined and distinct work phases** which are used by systems engineers and systems developers to plan for, design, build, test, and deliver information systems



Waterfall Model

1. Determine the Requirements
2. Complete the design
3. Do the coding and testing (unit tests)
4. Perform other tests (functional tests, non-functional tests, Performance testing, bug fixes etc.)
5. At last deploy and maintain



Agile

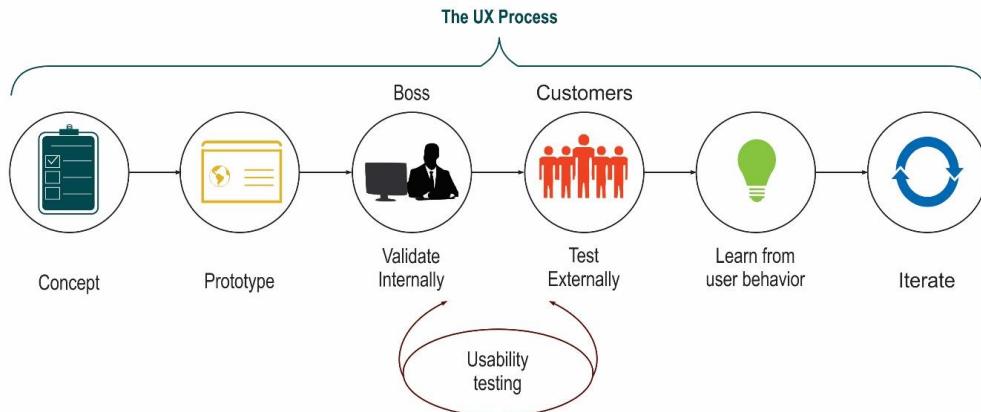
- - Shorter release cycle
- - Small batch sizes (M\
- - Cross-functional team
- - Incredibly agile

Agile Methodology



Lean Development

Lean Development (LD)



Not like this...



...instead like this!



- Suddenly ops was the bottleneck (more release less people), again WIP is more!

Challenges

Some of the challenges with the traditional teams of Development and Operations are:



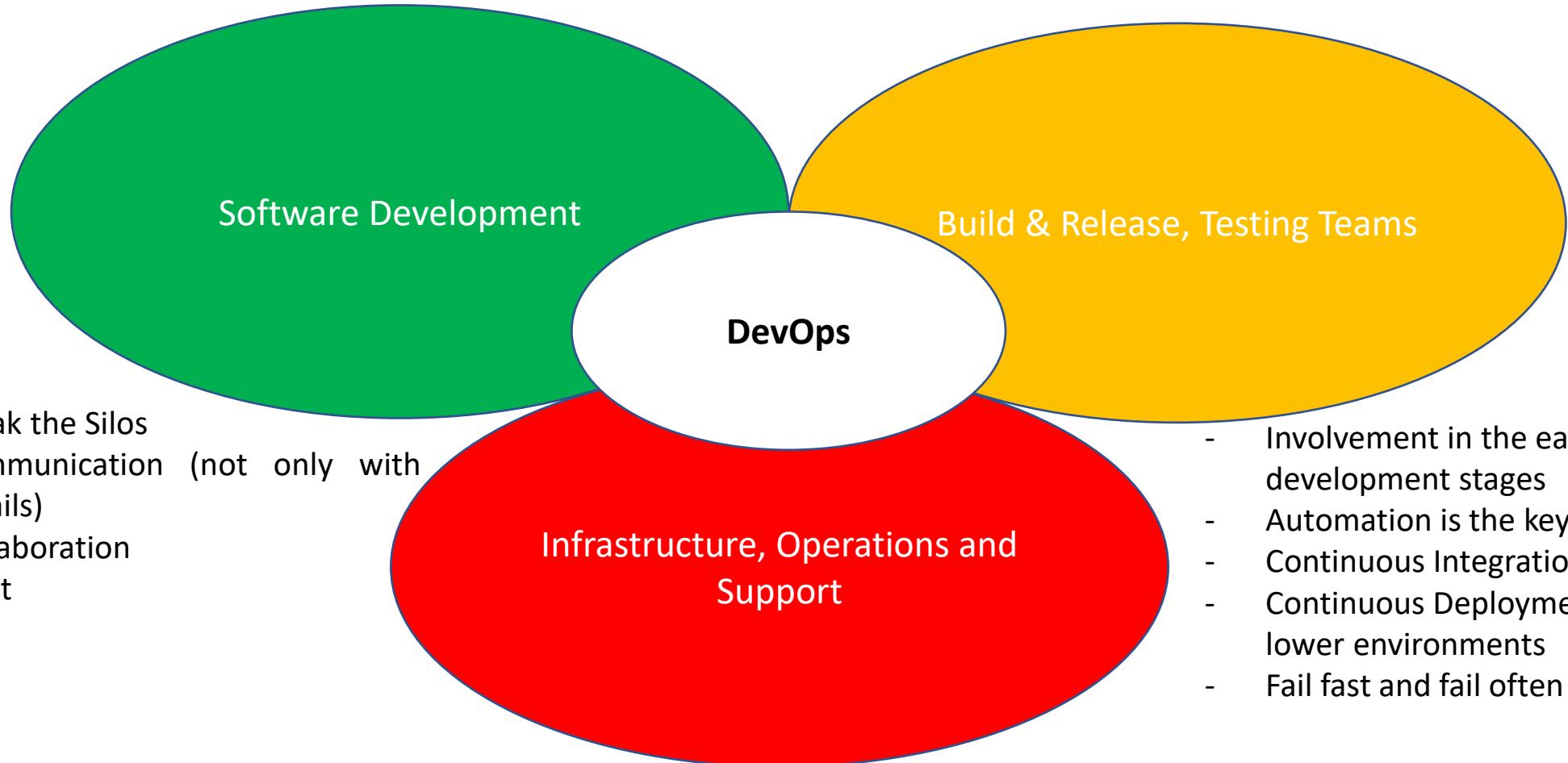
A Typical Case Study

- **Development Team:**
 - Monday Morning, the writing of code done, unit tests completed, code delivered to the Integration teams to get the code included in CI builds.
 - To get the services tested, a ticket is opened for QA teams
- **Build/Release/Testing/Integration Team:**
 - Tuesday Morning, ticket accepted, a tester put an email to the developer asking deployment instructions. There is not automated deployments, developer updated to the tester, lets come online and we will deploy the services to the QA environment together.
 - Call started, developer identified the “test environment” is not compatible.
 - Tuesday afternoon, a ticket raised in Ops Team with new specifications.
- **Ops Team:**
 - Wednesday morning, ticket accepted, specifications checked , a new port open request was identified.
 - Ticket raised for Security team, ticket accepted, change approved, port opened, email received by the Ops team the work is done.

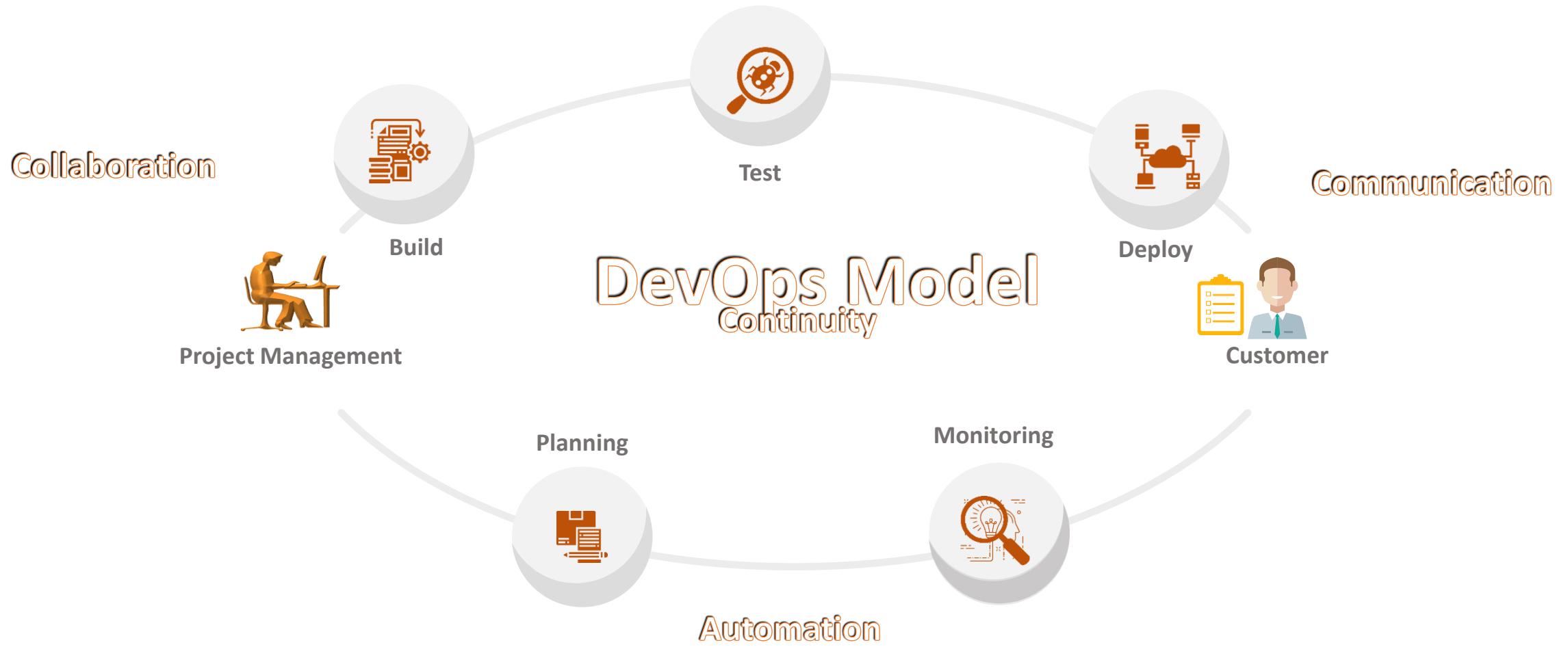
A Typical Case Study

- **Ops Team:**
 - Identified the provisioning requirements again and started work on building the environment.
- **Build/Release/Testing/Integration Team:**
 - Thursday Morning, updates received - the environment is ready. Developer and Tester again on call to deploy new services. Services deployed; tester is running test scripts. Next phase is to run regression test cases. Again a new ticket is raised for new test data with production teams and day ends.
- **Ops Team:**
 - Its Friday and the work is not on full swing, ticket accepted but not worked as production team has to complete rest of the works. Somehow the test data is gathered by Friday Evening.
- **Build/Release/Testing/Integration Team:**
 - Monday morning, tester gets the data, regression tests run, a defect found, and ticket returned to the development team.

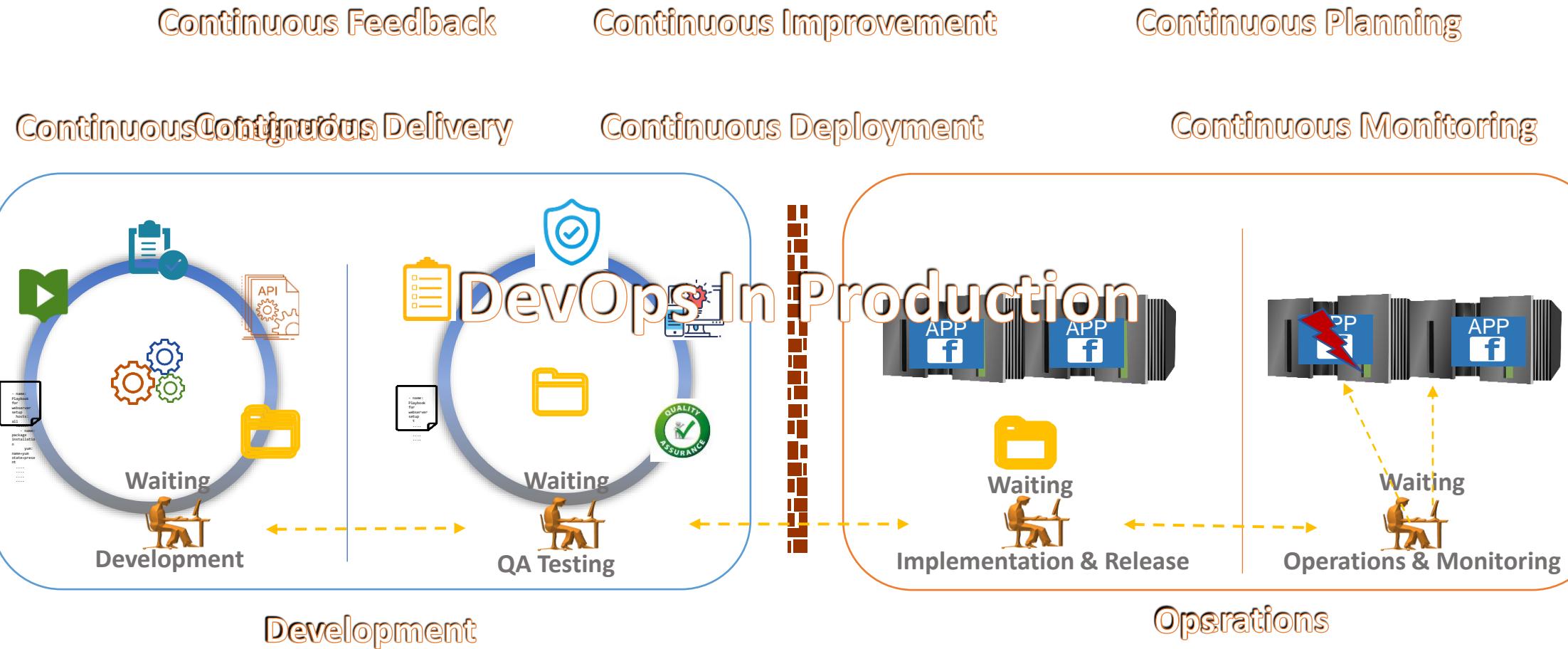
DevOps



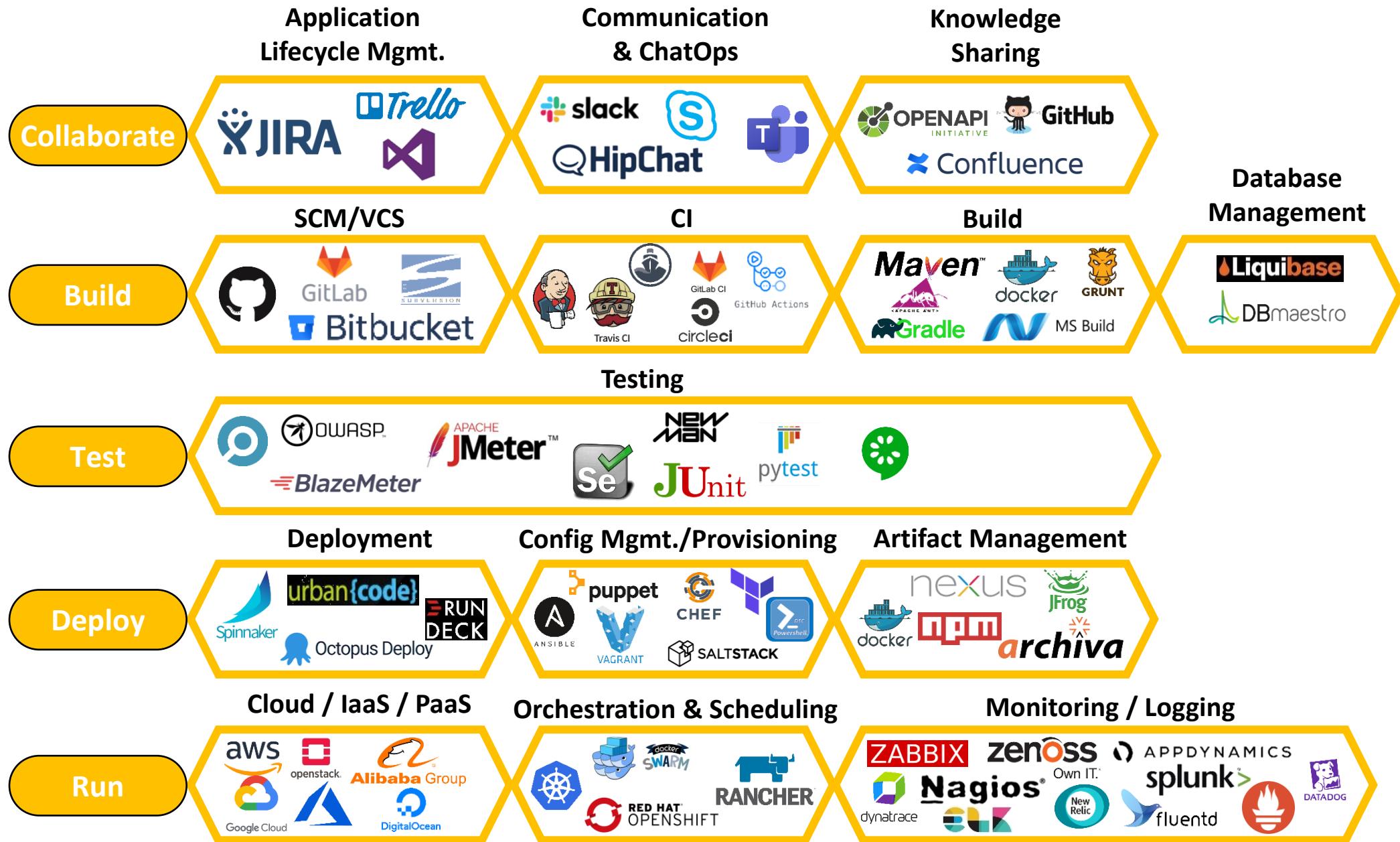
DevOps



DevOps in Action

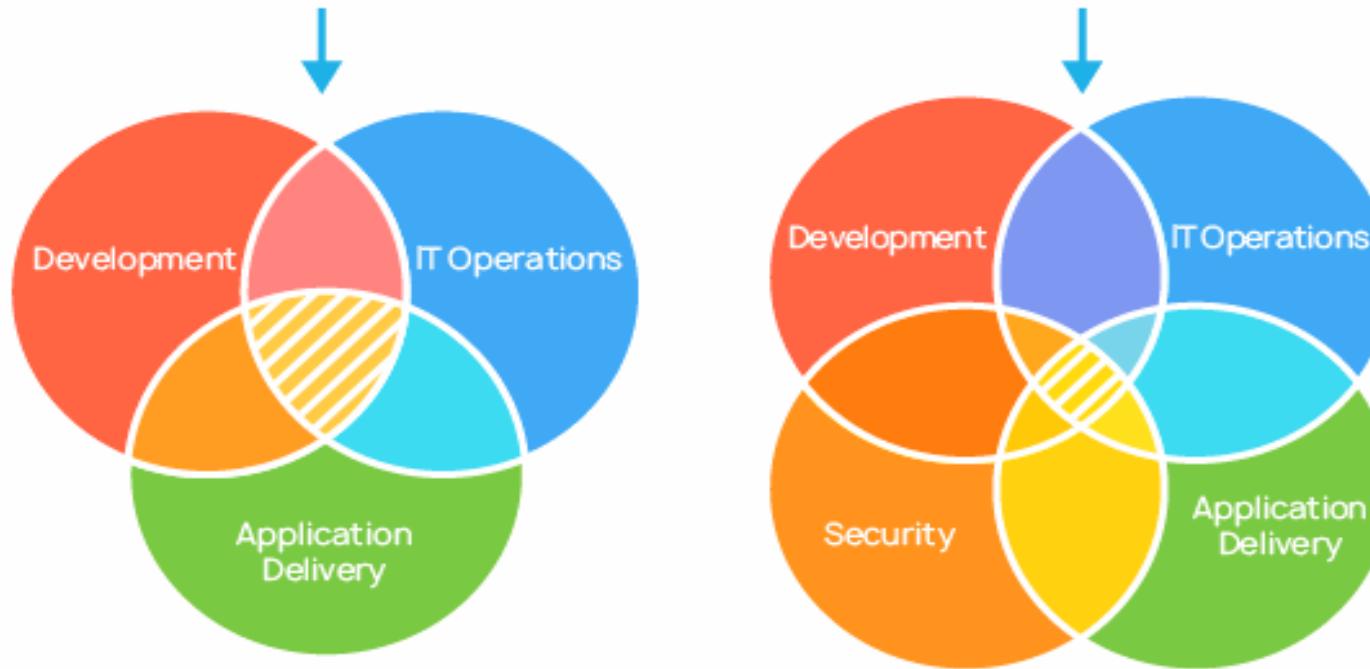


DevOps Toolsets



DevOps vs DevSecOps

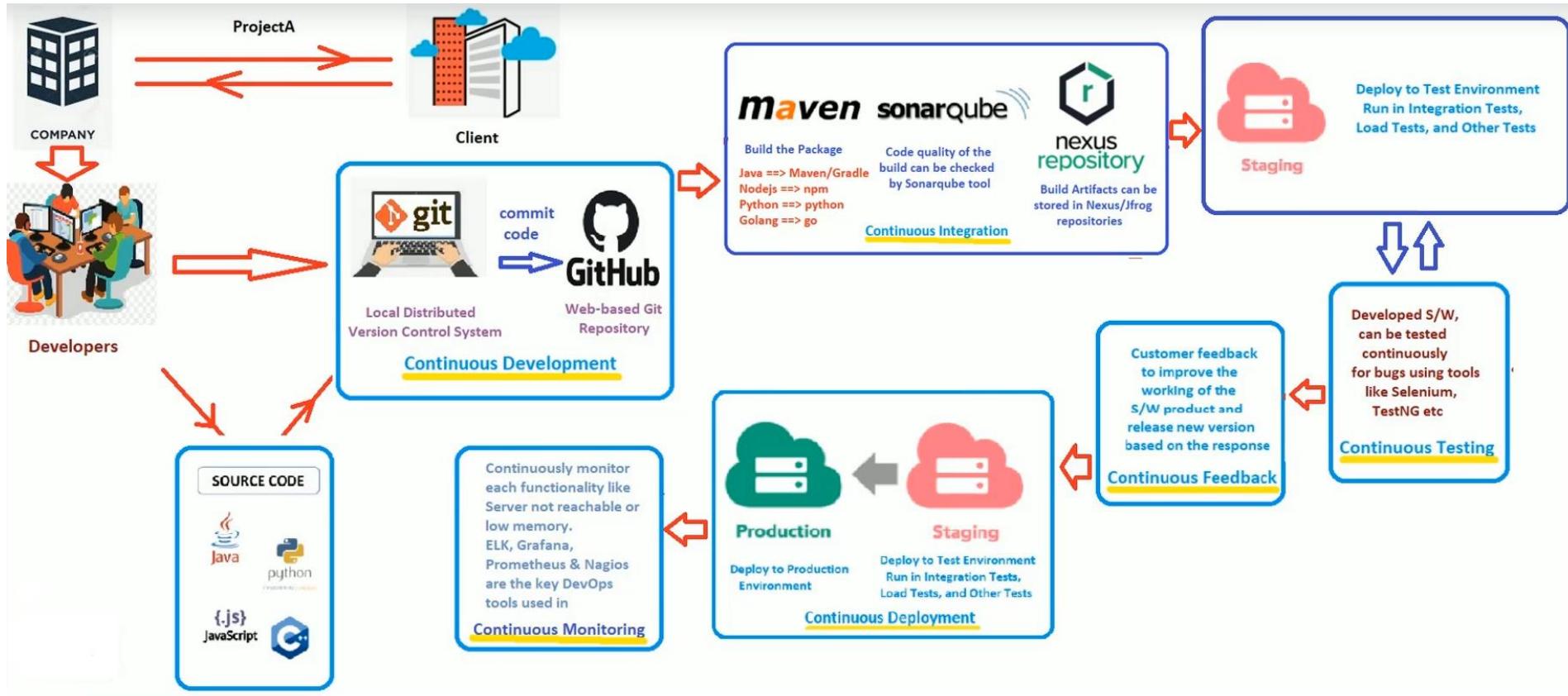
DevOps VS DevSecOps



What is DevOps?

- It is combining software development and IT operations with the aim of shortening software release cycles
- In reality it is all about automating as much as you can (e.g. infrastructure via pipelines). This enables a process for the software to be quickly released. This can involve:
 - Creating/maintaining CI/CD pipelines for development to leverage to release their code
 - Creating/maintaining environments/infrastructure so SDLC can develop/test etc
 - Managing various tech stacks such a version control (Git, SVN), CI/CD (Jenkins, GitLab, GitHub, Azure DevOps), infrastructure (Docker, VMs, Vagrant, Terraform), Cloud (Azure, AWS, GCP) and configuration management (Ansible, Chef)

DevOps Lifecycle

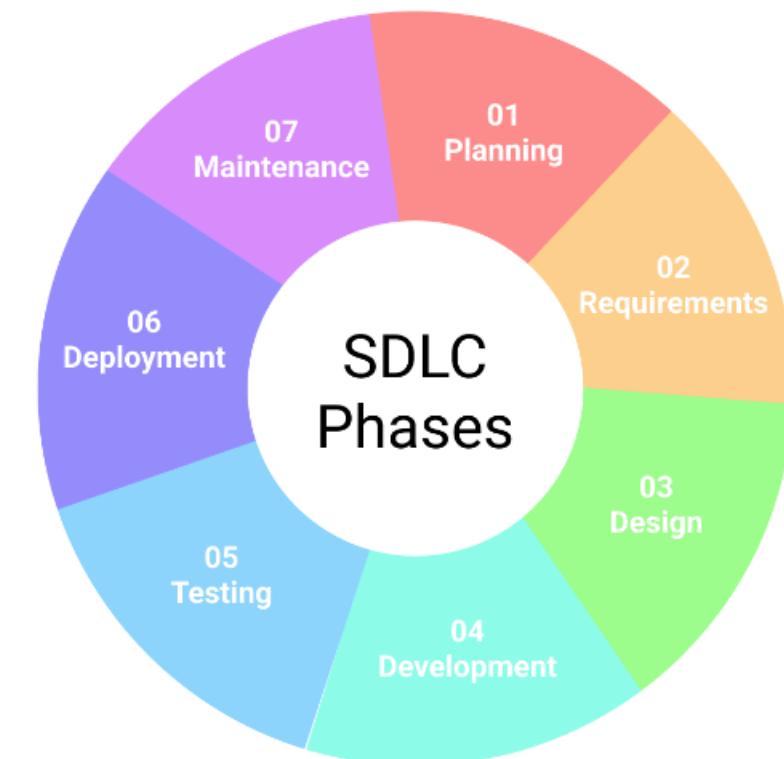


Application Security

- According to a report by Web Application Security Statistics, fixing any vulnerability can take **146 days** on average. Are you willing to wait that long?
- Maybe you are, but the hacker will not wait. So in the meantime, you can apply some initial security measures on your application to prevent hackers from exploiting any weakness.

DevSecOps

- It is essentially adding security testing to DevOps workflows
- DevSecOps uses Shift Left concept means moving the security testing further “left” as possible
- Identify issues as early as possible
- Cost to find an issue identified in development is ten times cheaper than finding in testing and 100 times cheaper than finding in production.
- So earlier we identify the issue, the cheaper it is.



DevSecOps

We can aid shift left via the implementation of:

- Creation of dedicated security pipelines that development with commit into, this could include:
 - SAST (static application security testing)
 - DAST (dynamic application security testing)
 - SCA (Software composition analysis)
- Ensuring best practices are followed and issues are prioritised and understood
- Push DevSecOps mindset – training development/ product staff in secure design/ principles/ how to develop securely

Traditional Security Tools

- Traditional security tools are software programs designed to identify and mitigate security risk within organisation's IT infrastructure.
- These tools are often used by dedicated security teams and focus on securing the network servers and endpoints.
- **For Example:**
- Firewalls
- Antivirus software
- Intrusion detection systems
- Vulnerability scanners

Limitations of Traditional Security Tools

- They tend to be reactive rather than proactive, they are assigned to detect and respond to security threats after they have occurred rather than preventing them from happening in the first place.
- They can create silos between different teams. Security teams are often seen as separate from development and operation teams which can lead to a lack of collaboration and communication.

Advantages of DevSecOps

- They allow security to be incorporated into every aspect of the software development lifecycle. This means that security becomes the responsibility of everyone involved in the process not just a specialized team.
- They encourage collaboration and communication between different teams within an organization. It is a shared responsibility rather than a sole responsibility.

Integration vs Separation

Traditional security tools are often seen as separate from the development process. They are used by dedicated security teams and are focused on securing the network, servers, and endpoints. [DevSecOps tools](#), on the other hand, are integrated into every aspect of the software development lifecycle. They are used by developers, operations teams, and security teams, and focus on securing the application itself.

Automation vs Manual

Traditional security tools often require manual processes, such as running scans or conducting audits. This can be time-consuming and can slow down the development process.

DevSecOps tools, however, automate many of these processes, making it faster and easier for developers to identify and fix security issues.

Proactive vs Reactive

As mentioned earlier, traditional security tools tend to be reactive, meaning they detect and respond to security threats after they have occurred. DevSecOps tools, on the other hand, are proactive, allowing developers to catch and fix security issues before they become a problem.

Key Principles of DevSecOps

- In order to begin implementing DevSecOps at your organisation you need six key principles to follow.
 - deliver small, frequent releases using agile methodologies
 - wherever possible, make use of automated testing
 - empower developers to influence security changes
 - ensure you are in a continuous state of compliance
 - be prepared for threats, always invest in advanced training for your engineers

Security as a Code

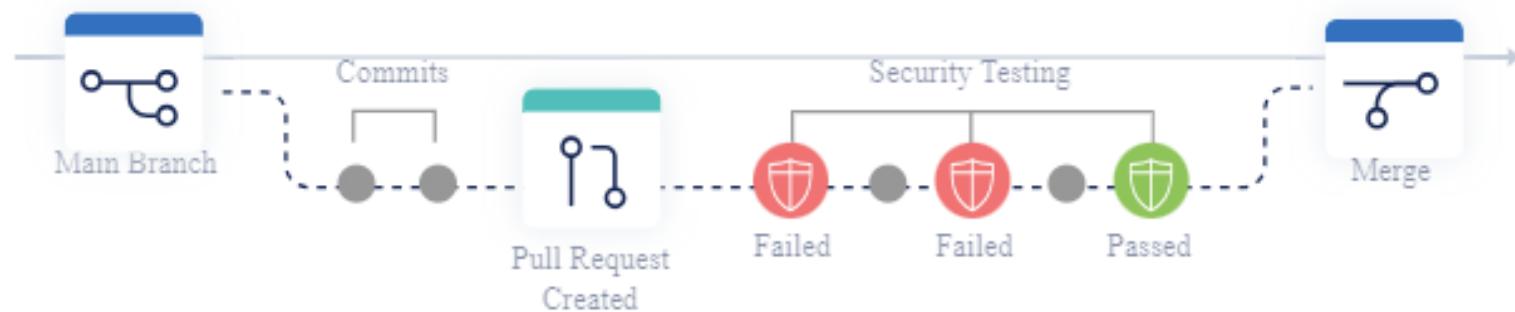
- Security as Code is the methodology of codifying security and policy decisions and socializing them with other teams.
- Security testing and scans are implemented into your CI/CD pipeline to automatically and continuously detect vulnerabilities and security bugs.
- Access policy decisions are codified into source code allowing everyone across the organization to see exactly who has access to what resources.
- Adopting Security as Code tightly couples application development with security management, while simultaneously allowing your developers to focus on core features and functionality, and simplifying configuration and authorization management for security teams.
- This improves collaboration between Development and Security teams and helps nurture a culture of security across the organization.

Implementing Security as a Code

- Security as Code generally comes in three different forms:
 - security testing
 - vulnerability scanning
 - access policies

Implementing Security as a Code

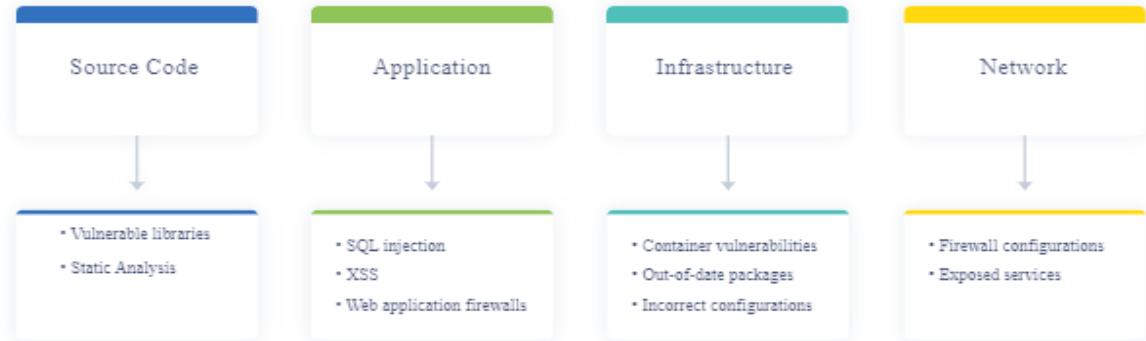
- **1. Security testing**
- It expands on best in class coding practices to add to the standard suite of tests to not only include functional and integration testing but also security focused testing.
- Static analysis for security vulnerabilities can be implemented on each commit or pull request.
- Permission boundaries can be checked to verify they cannot be crossed.
- APIs can be tested to ensure they're meeting authentication and authorization requirements.
- Security testing meets your developers where they already are, providing them immediate feedback on each and every commit.



Implementing Security as a Code

- **2. Vulnerability scanning**

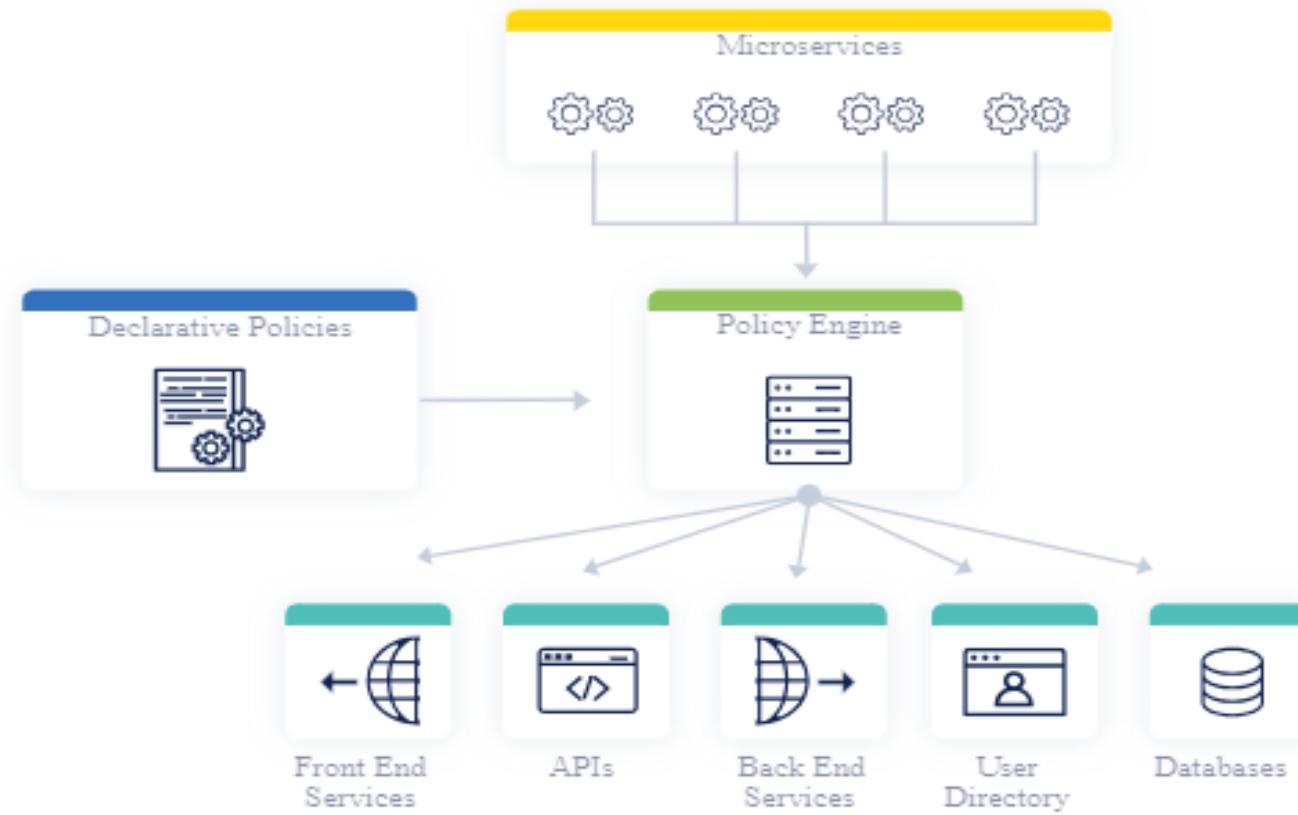
- It is applied at every level of your architecture. Across your pipeline you can verify that each of your application and deployment is secure against known vulnerabilities.
 - Source code can be scanned for vulnerable libraries.
 - For example, applications can be scanned for susceptibility to XSS and SQL injection.
 - Containers can be scanned for vulnerabilities in individual packages and for adherence to best practices. Full scanning of test, staging and production environments can be done continuously and automatically.
 - Scan early and scan continuously to verify your expected security controls are in place and so that you can find issues sooner rather than later.



Implementing Security as a Code

- **3. User and data access policies:**
- It codify governance decisions that can then be reviewed by anyone in your organization.
- These policies can be standardized, reducing the toil necessary to constantly monitor and maintain one off requests.
- Authorization can be offloaded to external libraries allowing your Dev teams to focus on core features.
- Security teams now have a central repository to work directly with developers to monitor and review authorization, allowing the entire company to move faster without breaking core security and compliance requirements.

Implementing Security as a Code

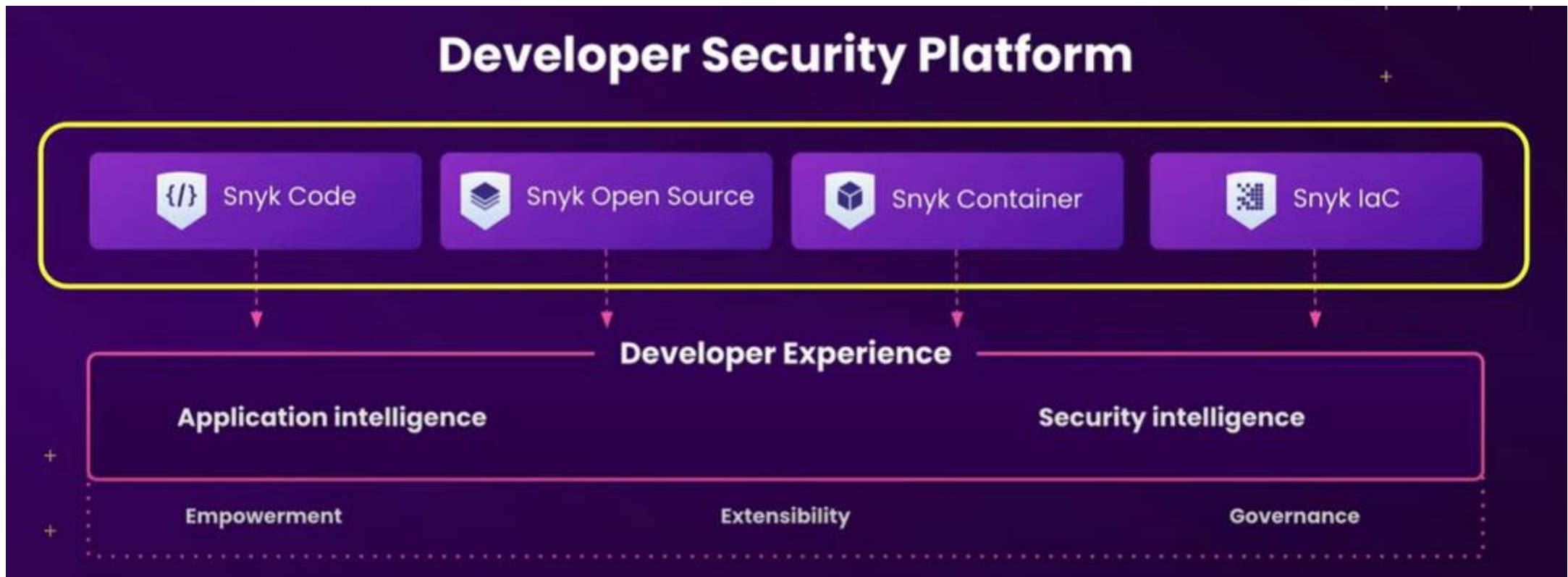


What is Snyk?

Snyk

- Snyk is a developer security platform that helps organizations find and fix vulnerabilities across their entire software development lifecycle, from code to containers and infrastructure. It's like a security superhero for developers, providing protection at every stage of the software creation process.
- **Features:**
- **Vulnerability Scanning:** Scans code, open-source dependencies, container images, and infrastructure as code for known vulnerabilities.
- **Priority and Remediation:** Provides actionable insights to prioritize and fix vulnerabilities effectively.
- **Automation and Integration:** Integrates seamlessly with developer workflows, CI/CD pipelines, and IDEs for continuous security.
- **Compliance and Reporting:** Helps meet compliance requirements and generate comprehensive security reports.

Snyk Architecture



Snyk Features

- **1. Architecture:**
- **Cloud-based platform:** Snyk operates as a cloud-based service, meaning your data and analysis are hosted securely in their infrastructure. This ensures scalability and accessibility for users anywhere.
- **Distributed scanning engines:** Snyk utilizes a network of distributed scanning engines to analyze code, dependencies, containers, and infrastructure as code. This parallel processing ensures fast and efficient vulnerability detection.
- **Vulnerability database:** Snyk maintains a constantly updated database of known vulnerabilities, encompassing various sources like **MITRE ATT&CK**, **NIST National Vulnerability Database**, and **Open Source Security Foundation**.
- **API-driven communication:** The platform relies on APIs to integrate seamlessly with developer workflows, CI/CD pipelines, and issue trackers. This enables smooth data exchange and automated security checks.

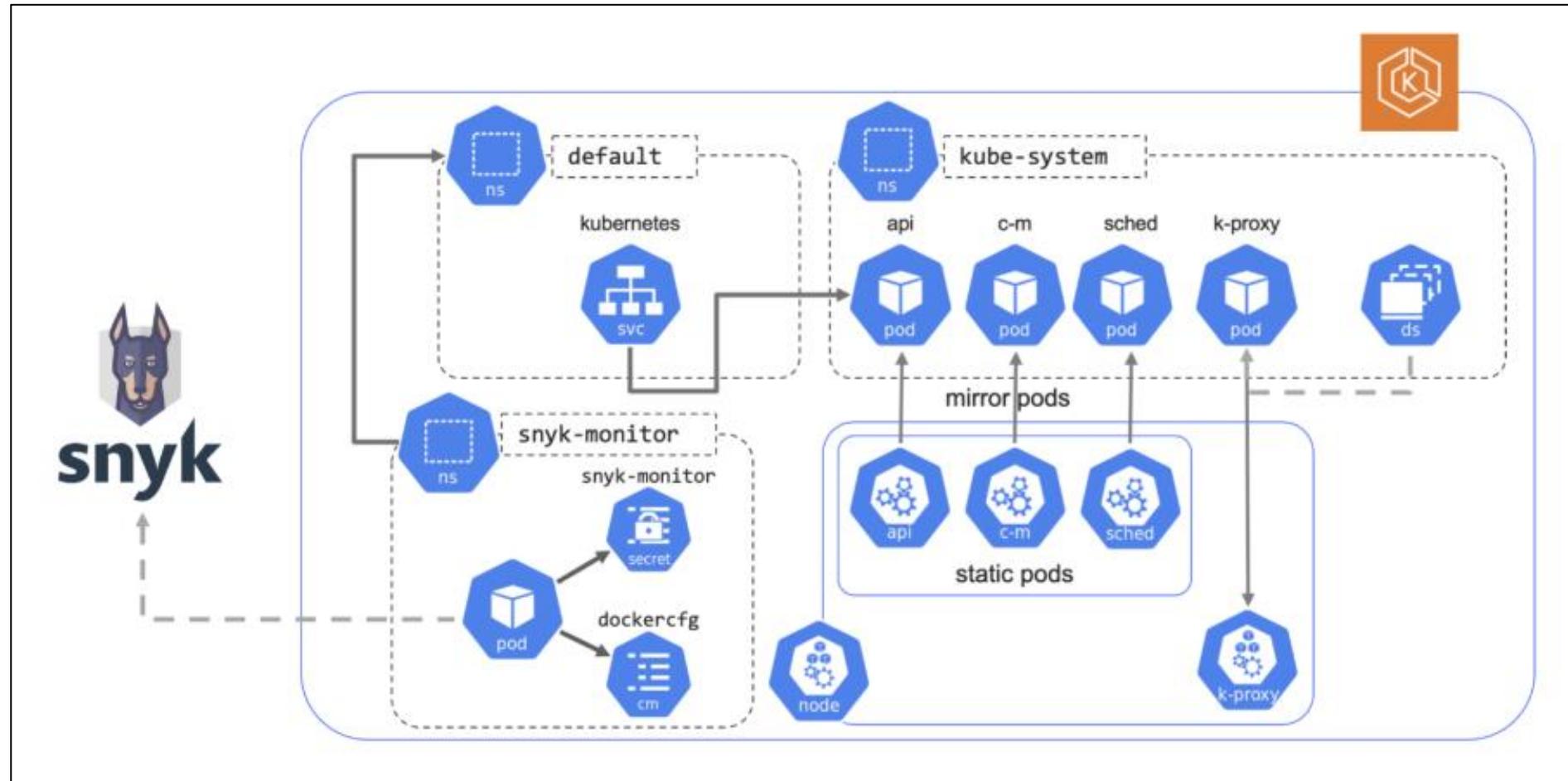
Snyk Features

- **2. Workflow:**
- **Integration:** Snyk integrates with your existing development process through plugins, SDKs, and API connections. This ensures security checks become part of your familiar workflow.
- **Scanning:** Based on your configuration, Snyk scans your code, dependencies, containers, or IaC templates for vulnerabilities.
- **Analysis and Prioritization:** The platform analyzes the identified vulnerabilities, considering severity, exploitability, and potential impact. This helps prioritize which vulnerabilities require immediate attention.
- **Remediation:** Snyk provides actionable insights and recommendations for fixing vulnerabilities. These include suggestions for patches, dependency upgrades, and configuration changes.
- **Reporting and Monitoring:** The platform generates comprehensive reports on identified vulnerabilities, remediation progress, and overall security posture. This data allows you to track progress and monitor your security posture.

Snyk Features

- **3. Key Aspects:**
- **Automation:** Snyk focuses on automating vulnerability scanning and reporting wherever possible. This frees up developers' time and ensures continuous security checks throughout the development lifecycle.
- **Collaboration:** Snyk facilitates collaboration between development and security teams by providing shared visibility into vulnerabilities and progress. This fosters a culture of DevSecOps where security is everyone's responsibility.
- **Customization:** Snyk offers customizable policies and scanning configurations to adapt to your specific needs and project requirements.
- **Continuous Improvement:** The platform constantly evolves with new features and updated vulnerability databases. This ensures your security protection stays ahead of emerging threats.

Snyk Architecture



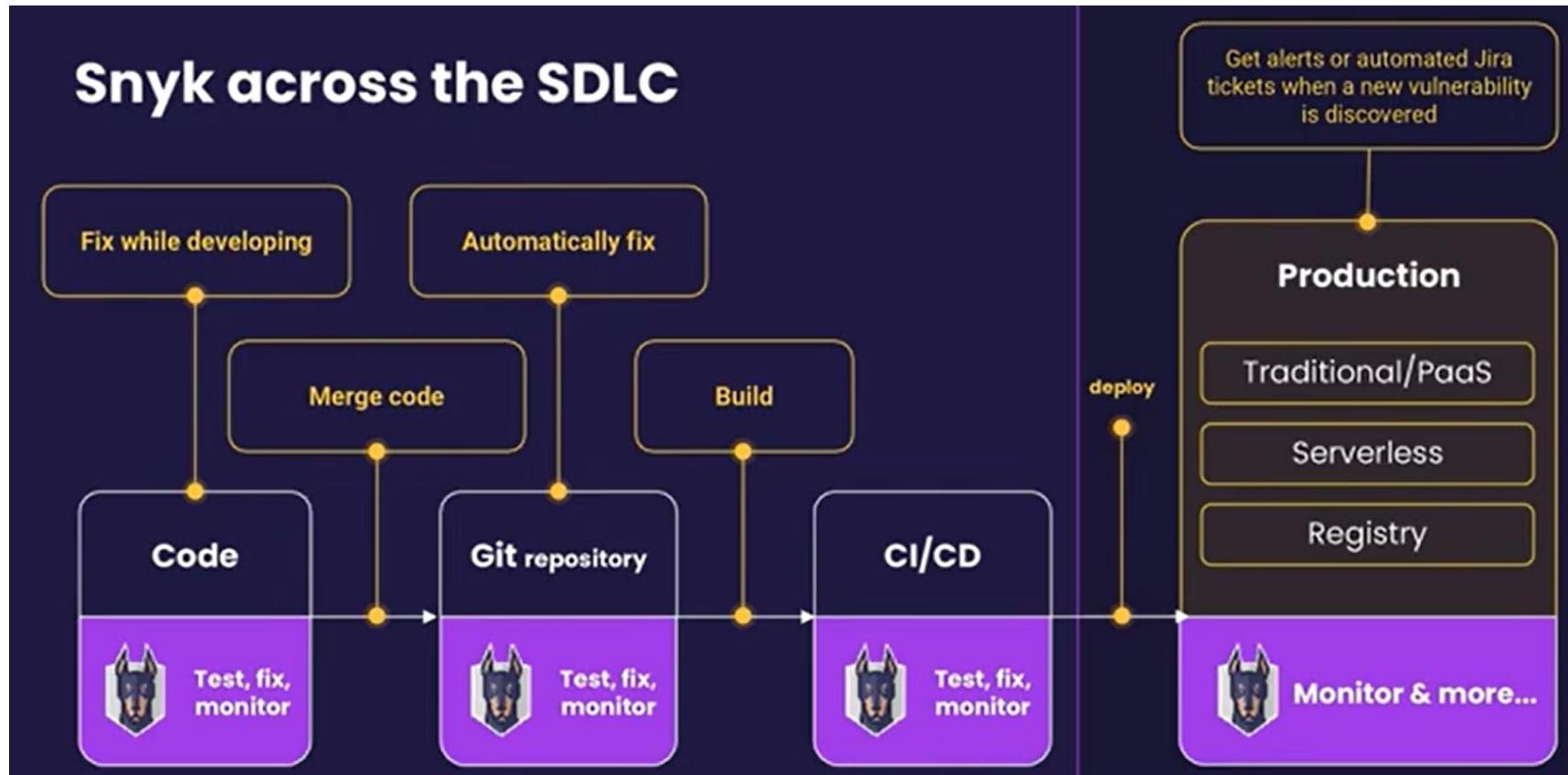
Architecture Layer

- **Frontend:** Provides user interface for interacting with scan results, reports, and configurations.
- **API Gateway:** Manages API interactions and routes requests to appropriate backend services.
- **Scanning Layer:** Utilizes diverse engines for different scan types (SAST, SCA, DAST, IaC)
 - **Code Analyzers:** Analyze source code for vulnerabilities using static analysis techniques.
 - **Dependency Scanners:** Identify vulnerabilities within open-source and proprietary dependencies.
 - **Container Scanners:** Analyze container images for vulnerabilities and misconfigurations.
 - **IaC Scanners:** Evaluate infrastructure as code templates for security weaknesses.
- **Vulnerability Database:** Houses a continually updated database of known vulnerabilities from various sources.
- **Analysis and Prioritization Engine:** Analyzes vulnerabilities based on severity, exploitability, potential impact, and context.
- **Recommendations Engine:** Suggests appropriate remediation actions for discovered vulnerabilities.
- **Reporting and Monitoring System:** Generates comprehensive reports and visualizes security posture for monitoring and tracking progress.

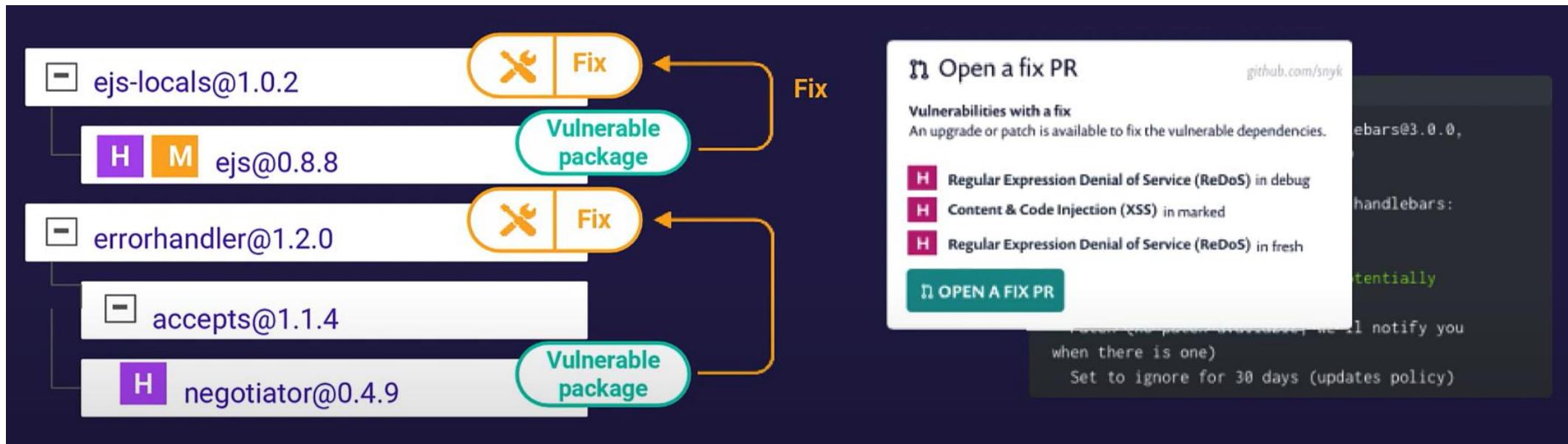
Workflow

- **Integration:** Snyk integrates with your development environment through plugins, SDKs, or APIs.
- **Triggering Scans:** Scans can be triggered manually, automatically upon code changes, or integrated into CI/CD pipelines.
- **Scanning and Analysis:** The relevant engine(s) based on scan type (code, dependencies, container, IaC) analyze the target resources.
- **Vulnerability Detection:** Snyk identifies vulnerabilities by cross-referencing scan results with its extensive vulnerability database.
- **Prioritization and Analysis:** Vulnerabilities are prioritized based on factors like severity, exploitability, and potential impact. Contextual information like affected files and lines of code is extracted.
- **Remediation Recommendations:** The platform suggests feasible remediation options like patches, dependency upgrades, and configuration changes.
- **Reporting and Monitoring:** Comprehensive reports detailing identified vulnerabilities, suggested actions, and overall security posture are generated. Security metrics are visualized for continuous monitoring.

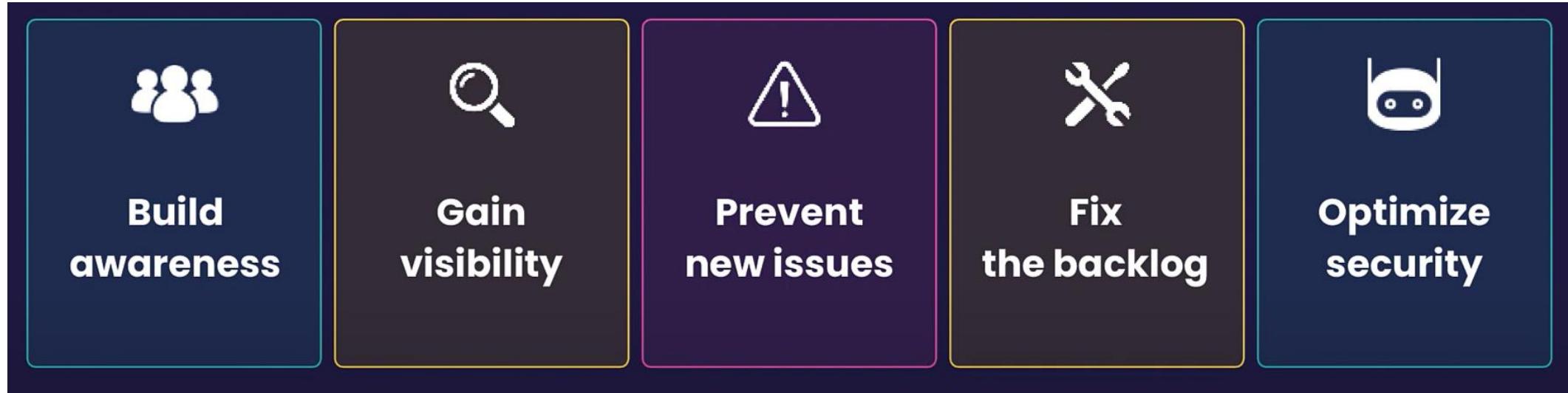
Snyk across the SDLC



Discovery and Fix issues



Managing your security with Snyk



Synk SaaS Account

Snyk Account

SaaS (Cloud-Based Solution):

- Snyk is primarily offered as a Software-as-a-Service (SaaS) solution, meaning it runs in the cloud and doesn't require installation on local infrastructure.
- Users interact with the platform via the web interface or APIs, and the scanning of projects (e.g., GitHub repositories, containers, etc.) is performed remotely.

Integration Steps: Github

1. Choose integration method (GitHub)
2. Set Access Permission
3. Configure automation settings and authenticate
4. Import and Scan the code

Snyk Account

The screenshot shows the Snyk dashboard interface. On the left is a dark sidebar with the Snyk logo and navigation links: Organization (sandy.2087), Dashboard (which is highlighted in purple), Projects, Integrations, Members, and Settings.

The main content area has a header showing the organization name "sandy.2087 > Dashboard".

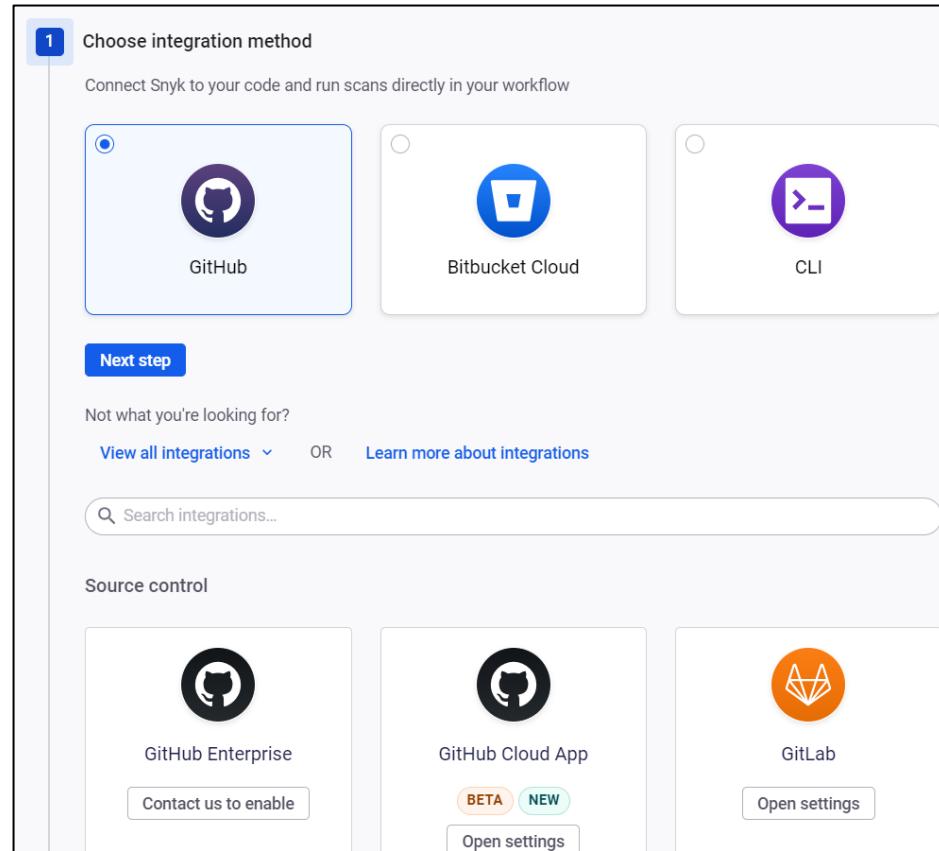
The first section is titled "Start securing your code" and contains two options:

- Connect your code**: A link to "Choose integration".
- Add and scan your first project**: A link to "Import your code to see how Snyk surfaces issues, problematic dependencies, and vulnerabilities."

The second section is titled "Invite team members" with the sub-instruction "Collaborate on projects and build secure applications together".

The third section is titled "Use Snyk in the command line" with the sub-instruction "Learn how to install our command line tool to scan your code locally".

Snyk Integration



Repository Access

2 Set access permissions

Private and public repositories
Grant Snyk access to all repository types under your Github account whether private or public.

Public repositories only
Grant Snyk access to repositories marked public under your Github account.

Once authenticated, Snyk:

- ✓ Scans the directory trees of selected repos and automatically represents them as projects
- ✓ Generates security reports that enable you to explore issues in your repositories and assist you with fixing them
- ✓ Continuously checks imported projects for vulnerabilities. When new vulnerabilities are found, you'll be notified

[Next step](#) [Previous](#)

Automation Setting

3 Configure automation settings & authenticate

Enabled features:

- Pull Request Checks
Test your pull requests for new issues and vulnerabilities
- New Fix Pull Requests
Automatically create pull requests for newly discovered open source issues and vulnerabilities
- Dependency Upgrade Pull Requests
Keep your packages up to date with automatic dependency upgrade pull requests
- Snyk Code
Analyze your source code for issues and vulnerabilities [?](#)

Import and scan the code

Import and scan your first project

Import your code to see how Snyk surfaces issues, problematic dependencies, and vulnerabilities.

Choose from your most active repositories or [view all your repositories](#)

- CloudSihmar/**php-laravel-project-latest**
- CloudSihmar/**devops-documents**
- CloudSihmar/**aws-training**
- CloudSihmar/**pet**
- CloudSihmar/**php-mariadb-lara**

Import and scan [Follow an open-source repo](#)

Scan Result

CloudSihmar/pet		Imported	Tested	Issues ↓
<input type="checkbox"/>	pom.xml	a few seconds ago	a few seconds ago	3 C 30 H 49 M 6 L ...
<input type="checkbox"/>	Dockerfile.v3	a few seconds ago	a few seconds ago	1 C 0 H 0 M 33 L ...
<input type="checkbox"/>	Dockerfile.v1	a few seconds ago	a few seconds ago	1 C 0 H 0 M 33 L ...
<input type="checkbox"/>	Code analysis	a few seconds ago	a few seconds ago	0 C 0 H 5 M 0 L ...
<input type="checkbox"/>	nginx.yaml	a few seconds ago	a few seconds ago	0 C 0 H 4 M 6 L ...
<input type="checkbox"/>	terraform/main.tf	a few seconds ago	a few seconds ago	0 C 0 H 3 M 4 L ...
<input type="checkbox"/>	service.yaml	a few seconds ago	a few seconds ago	0 C 0 H 1 M 0 L ...
<input type="checkbox"/>	Dockerfile	a few seconds ago	a few seconds ago	0 C 0 H 0 M 0 L ...

Group level overview

Company Name / Line of Business > Reports

Issues Detail

Issue Status: Open + Add filter

Issues by Severity

Critical	High	Medium	Low
--	--	--	--

TOTAL ISSUES 28K **UNIQUE VULNS** 1,556

Issue details

Modify Columns Download CSV

SCORE	ISSUE	CVE	CWE	PROJECT	EXPLOIT MATURITY	AUTO FIXABLE	INTRODUCED	SNYK PRODUCT
C 899	Vulnerability Arbitrary File Write Archive Extract	CVE-2018-1002204	CWE-29	snyk/npm7-goof:package.json	Mature	Upgrade	Jan 13, 2023	Snyk Open Source
C 899	Vulnerability Arbitrary File Write Archive Extract	CVE-2018-1002204	CWE-29	snyk/berry-goof:package.json	Mature	Upgrade	Jan 13, 2023	Snyk Open Source
H 825	Vulnerability NoSQL Injection		CWE-89	snyk/goof-read-only			Jan 13, 2023	Snyk Code
H 825	Vulnerability NoSQL Injection		CWE-89	snyk/goof-read-only			Jan 13, 2023	Snyk Code
H 825	Vulnerability NoSQL Injection		CWE-89	snyk/goof-read-only			Jan 13, 2023	Snyk Code
H 820	Vulnerability NoSQL Injection		CWE-89	snyk/snyk-goof			Jan 13, 2023	Snyk Code
H 820	Vulnerability NoSQL Injection		CWE-89	snyk/berry-goof			Jan 13, 2023	Snyk Code
H 820	Vulnerability NoSQL Injection		CWE-89	snyk/snyk-goof			Jan 13, 2023	Snyk Code
H 820	Vulnerability NoSQL Injection		CWE-89	snyk/github-apps-goof			Jan 13, 2023	Snyk Code
	Vulnerability		CWE-89	snyk/berry-goof			Jan 13, 2023	Snyk Code

Help Michele Wiedemer https://app.snyk.io/group/224147d2-2298-428c-a093-0aff4632ada5/policies



What is a Project?

Snyk can scan multiple types of files for issues, including:



Manifest files



Source code files



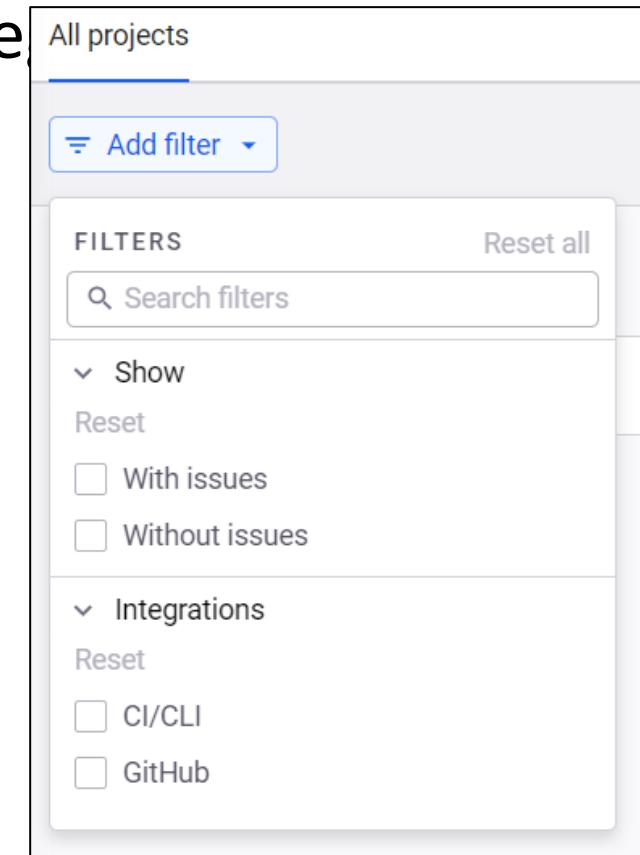
Container images



Infrastructure as code files

Project

- Filtering can be done based on issues and integrations



Vulnerability

C **org.springframework:spring-beans** - Remote Code Execution ⚡ SCORE **919**

VULNERABILITY | ***

 **Insights:** Current public exploits for the Spring4Shell vulnerability require the following conditions:

- Built with Java Runtime Environment (JRE) version 9 or above
- Deployed on either Apache Tomcat, Payara or Glassfish
- Dependent on spring-webmvc or spring-webflux

If your application configuration applies to these conditions, we advise prioritizing remediation of this vulnerability. However, given it is technically possible for additional exploit conditions to exist we do recommend attempting upgrading to a fixed versions for all vulnerable instances.

Introduced through	org.springframework.boot:spring-boot-starter-cache@2.3.1.RELEASE, org.springframework.boot:spring-boot-starter-data-jpa@2.3.1.RELEASE and others	Exploit maturity	MATURE
Fixed in	org.springframework:spring-beans@5.2.20, @5.3.18		

Show more detail ▾

 Learn about this type of vulnerability ↗

Vulnerability Database

- Snyk uses its proprietary vulnerability database, which is one of the most comprehensive and up-to-date in the industry. The Snyk vulnerability database aggregates information from multiple trusted sources and community-driven security databases, including:
- **National Vulnerability Database (NVD):** A U.S. government repository of standardized vulnerabilities (CVEs).
- **MITRE CVE:** The Common Vulnerabilities and Exposures (CVE) system, maintained by MITRE, is widely used to identify and track vulnerabilities.
- **Security Advisories:** Snyk pulls vulnerability information from security advisories and issue trackers from platforms like GitHub, npm, RubyGems, PyPI, and other package managers.
- **Community Contributions:** Snyk also includes vulnerabilities reported by the open-source community, security researchers, and its users, providing a broader set of data.
- **Public Bug Bounty Programs:** Information from bug bounty programs where vulnerabilities are disclosed and tracked.

Vulnerability Database

C org.springframework:spring-beans - Remote Code Execution ⚙ SCORE **919**

VULNERABILITY | [...](#)

Insights: Current state: [CWE-94](#) [CVE-2022-22965](#) [CVSS 9.8](#) [SNYK-JAVA-ORGSPRINGFRAMEWORK-2436751](#)

- Built with [Spring Framework](#)
- Deployed to [Maven Central](#)
- Dependent on [spring-webmvc](#) or [spring-webflux](#)

If your application configuration applies to these conditions, we advise prioritizing remediation of this vulnerability. However, given it is technically possible for additional exploit conditions to exist we do recommend attempting upgrading to a fixed versions for all vulnerable instances.

Introduced through [org.springframework.boot:spring-boot-starter-cache@2.3.1.RELEASE, org.springframework.boot:spring-boot-starter-data-jpa@2.3.1.RELEASE and others](#) Exploit maturity MATURE

Fixed in [org.springframework:spring-beans@5.2.20, @5.3.18](#)

[Show more detail ▾](#)

 [Learn about this type of vulnerability](#) ⚙

Snyk Score

- The Snyk score provides a measure of the risk associated with a vulnerability and helps prioritize issues for remediation. It combines several factors to give a meaningful indication of the urgency and severity of a vulnerability.
- **Out of 1000 score:**
- 500 points are the severity base
- 500 from contextual details

Snyk Priority Score Factors



Snyk Score

Severity:

- This refers to how serious the vulnerability is based on its impact on the system. It's often determined using the **Common Vulnerability Scoring System (CVSS)**, which rates vulnerabilities on a scale from 0 (low) to 10 (critical).
- High-severity vulnerabilities can have a significant impact on the confidentiality, integrity, or availability of an application, while lower-severity vulnerabilities might pose a lesser threat.
- Severity is one of the most critical factors in deciding whether a vulnerability should be addressed immediately or can be postponed.

Snyk Score

Exploit Maturity:

Snyk considers how mature an exploit is for a given vulnerability:

- **Proof of Concept (PoC) available:** Exploits are published and easy to replicate.
 - Example: Imagine a vulnerability is discovered in a popular web framework. A security researcher publishes a Proof of Concept showing how an attacker could exploit a flaw in the framework to bypass authentication. While this proof exists, no major attacks have yet occurred, but the risk is real because attackers could use the PoC to exploit vulnerable systems.
- **Mature Exploit:** The vulnerability is well-known and has been widely exploited.
 - Example: A critical vulnerability is found in a widely used content management system (CMS). Hackers have been exploiting this vulnerability on thousands of websites. Security companies observe this flaw being used by ransomware attackers, and as a result, it is classified as a Mature Exploit.
- This helps assess whether a vulnerability is actively being exploited in the wild or is only theoretically vulnerable, influencing the priority for fixing it.

Snyk Score

Reachability:

- This factor assesses whether the vulnerability is reachable or triggerable in your code. If a vulnerable piece of code (library or component) is not used in a way that can be exploited, it's considered less reachable.
- Snyk performs reachability analysis to determine if the vulnerable code is actually called by your application, making the vulnerability a higher priority if it is reachable.
- **Example:** You are using a web framework that includes a vulnerable library for processing file uploads. If your application allows users to upload files (and this functionality uses the vulnerable library), Snyk's reachability analysis would determine that the vulnerable code is actually called by your application. Therefore, this vulnerability is reachable and should be prioritized for fixing.

Snky Score

Fixability:

- Fixability looks at whether a solution is available to patch the vulnerability. If a fix (such as an updated package, library, or patch) is available, this factor pushes the vulnerability higher in the priority list.
- On the other hand, if no fix is available or the remediation effort is complex, the issue may be deprioritized or marked for later resolution.

Snyk Score

Time:

- Time since disclosure considers how long the vulnerability has been known. Newly discovered vulnerabilities, especially those that haven't been patched or have not been fully analyzed, often carry greater risk because attackers may exploit them before organizations are aware of them.
- Older vulnerabilities may also pose a risk if they haven't been addressed yet, but newer, more pressing vulnerabilities often take priority.

Snky Score

Social Trends:

- Social trends involve looking at real-world activity around a particular vulnerability. If there's increased chatter or discussion about a specific vulnerability across developer communities, security forums, or social media, it indicates that attackers or other organizations may be paying attention to it.
- For example, when a vulnerability starts trending (like Log4Shell), its priority increases, as attackers might be actively looking to exploit it.

Snyk Score

- **Malicious Packages:**
- **Malicious packages** refer to intentionally harmful code or libraries introduced into open-source ecosystems. These are different from accidental vulnerabilities in legitimate libraries because they are intentionally designed to cause harm.
- Snyk scans for packages that are **malicious** in nature, rather than just vulnerable, and prioritizes them higher because they represent a direct and deliberate threat.

Why score is important?

Prioritization:

By combining factors like severity, exploitability, and fix availability, the Snyk score helps organizations decide which vulnerabilities to address first, improving the efficiency of remediation efforts.

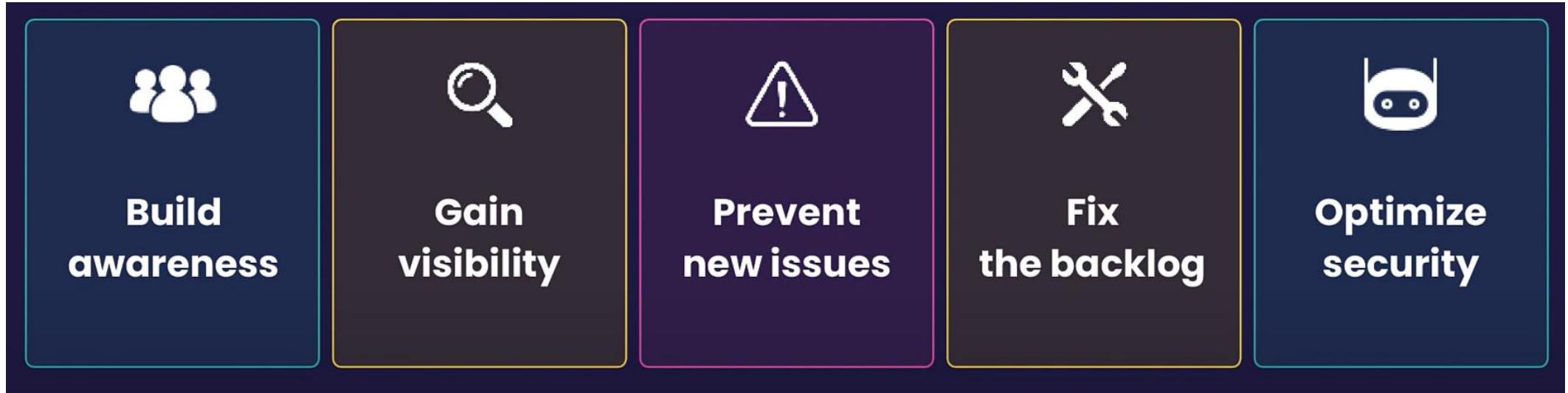
Risk Awareness:

It highlights the level of risk a vulnerability poses in the current context, enabling teams to better assess potential impact on their applications.

Resource Allocation:

Helps teams focus their time and resources on the most critical vulnerabilities, ensuring that high-risk issues are resolved first.

Managing your security with Snyk : Phases



Managing your security with Snyk: Phases

- **Snyk Developer Adoption model:**

- Gain visibility
- Prevent new issues
- Fix the backlog
- Optimize security

Maturing your security with Snyk



- Disable automations
 - Default Snyk test for pull requests
 - Automatic fix pull requests
 - Automatic dependency upgrade pull requests
 - Notification emails
- Decide whether to include patches for manual fixes

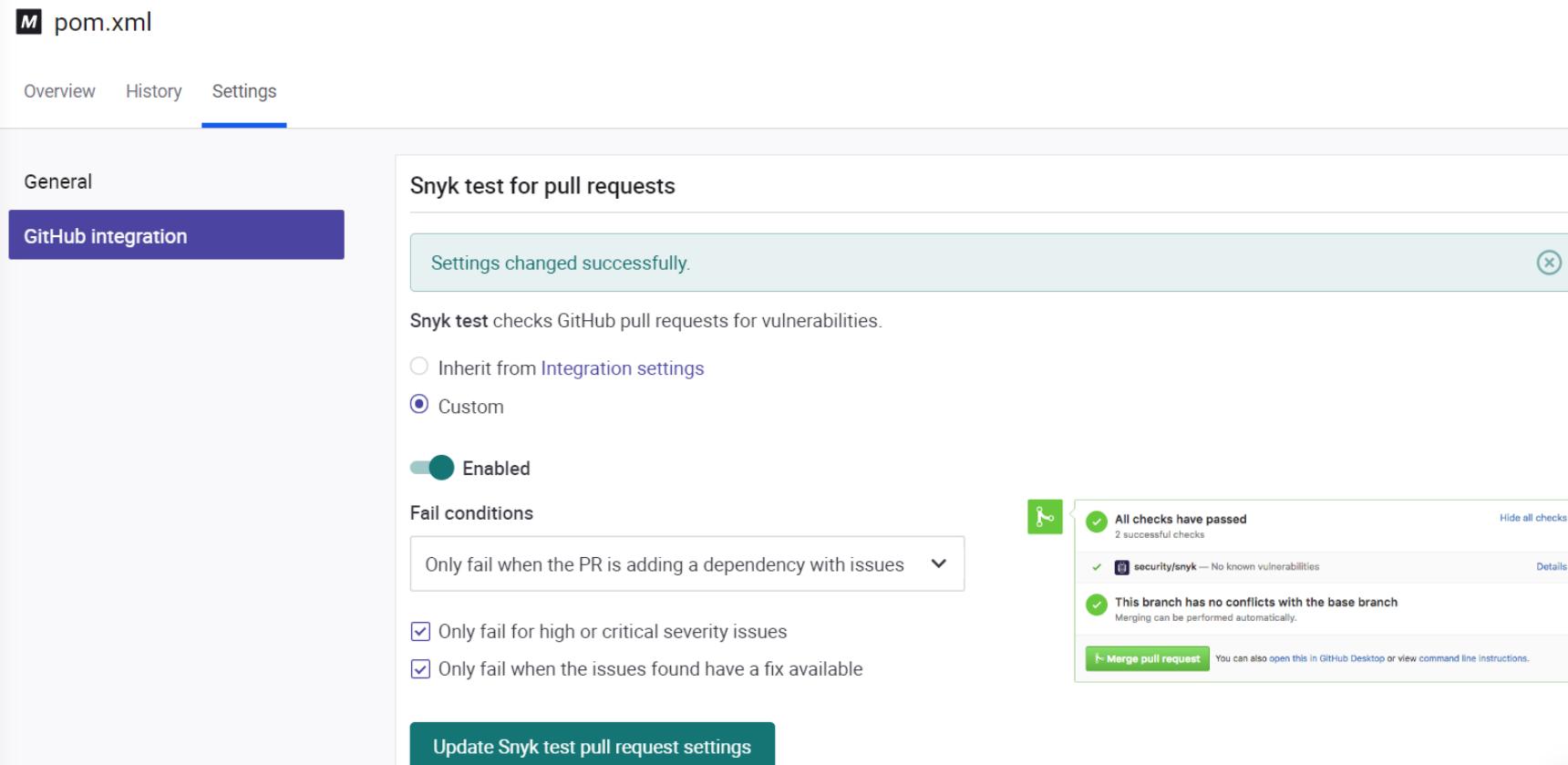
Maturing your security with Snyk



**Prevent
new issues**

- Configure Snyk to block PR/builds for specific project
- Set a triage process on blocked actions
- Fix backlog of issues

Maturing your security with Snyk: Project Level



The screenshot shows the Snyk project settings interface for a project named "pom.xml". The "Settings" tab is selected. Under the "General" section, the "GitHub integration" tab is active. A success message "Settings changed successfully." is displayed. The "Snyk test for pull requests" section is configured with "Custom" settings, "Enabled", and "Fail conditions" set to "Only fail when the PR is adding a dependency with issues". Below this, two additional fail conditions are checked: "Only fail for high or critical severity issues" and "Only fail when the issues found have a fix available". A green sidebar on the right displays "All checks have passed" (2 successful checks), "security/snyk — No known vulnerabilities", and "This branch has no conflicts with the base branch". A "Merge pull request" button is present at the bottom.

M pom.xml

Overview History Settings

General GitHub integration

Snyk test for pull requests

Settings changed successfully.

Snyk test checks GitHub pull requests for vulnerabilities.

Inherit from Integration settings

Custom

Enabled

Fail conditions

Only fail when the PR is adding a dependency with issues

Only fail for high or critical severity issues

Only fail when the issues found have a fix available

Update Snyk test pull request settings

All checks have passed
2 successful checks

security/snyk — No known vulnerabilities

This branch has no conflicts with the base branch

Merge pull request You can also open this in GitHub Desktop or view command line instructions.

Maturing your security with Snyk: Organization Level

Snyk >> Setting >> Github

Pull request status checks

Open Source security and licenses

Configure Snyk Open Source security and licenses PR Checks on your GitHub repositories.

[More info](#)

Enable

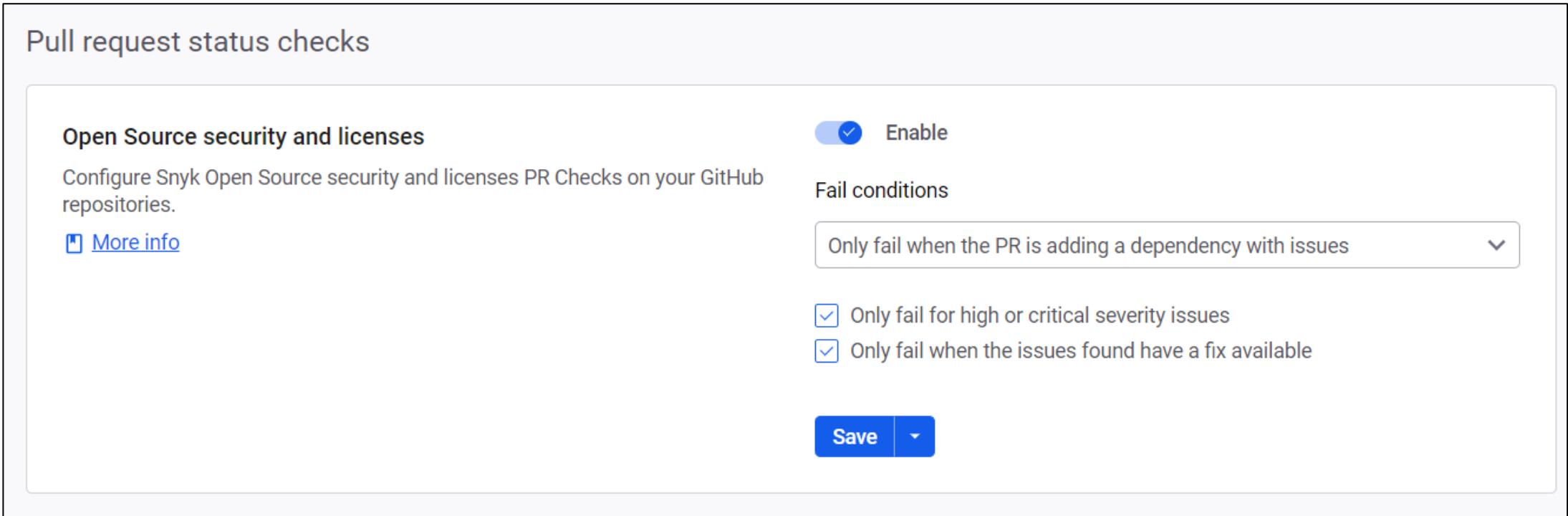
Fail conditions

Only fail when the PR is adding a dependency with issues

Only fail for high or critical severity issues

Only fail when the issues found have a fix available

Save | **▼**



Snyk test for pull requests

It is a process where Snyk automatically scans the code changes in a pull request (PR) to detect potential security vulnerabilities, outdated dependencies, and other issues before they are merged into the main codebase. This integration provides early feedback to developers about any security risks introduced by the new code or dependency changes.

Key concepts:

- **Automated Security Checks on PRs:**

- When a developer submits a pull request, Snyk scans the PR to detect vulnerabilities in the changed code or dependencies.
- It evaluates both the application's source code and the dependencies for security issues.
- The test results are typically displayed directly in the pull request, providing feedback to the developer and the reviewer.

Snyk test for pull requests

- **Shift-Left Security:**
 - This practice promotes catching vulnerabilities early in the development process, specifically during the code review stage. This is part of the "shift-left" approach in DevSecOps, where security is integrated earlier in the software development lifecycle.
- **Detailed Vulnerability Report:**
 - Snyk analyzes the code changes and reports any vulnerabilities or licensing issues found in the PR.
 - For each identified issue, Snyk provides a description, severity level (low, medium, high, critical), and possible fixes or recommendations.

Snyk test for pull requests

- **Failing PR Based on Security Threshold:**
 - Based on the severity of vulnerabilities or organizational policies, the pull request can be configured to pass or fail.
 - For example, if a high-severity vulnerability is found, Snyk can block the PR from being merged until the issue is addressed.
- **Integration with CI/CD Pipelines:**
 - Snyk can be integrated with continuous integration systems (e.g., Jenkins, GitHub Actions, GitLab CI) to automate the testing of PRs.
 - It ensures that no vulnerable code is introduced at any stage of the pipeline.

Benefits of Snyk Test for Pull Requests

- **Prevent Vulnerable Code:** Identifying issues before merging ensures that no known vulnerabilities make it to the main branch or production.
- **Faster Remediation:** Early detection of vulnerabilities means they can be addressed as soon as they are introduced, leading to quicker fixes.
- **Security Visibility:** Developers and security teams get visibility into the security posture of each code change, fostering a culture of secure coding.
- **Automated Feedback Loop:** Developers get immediate feedback about vulnerabilities, allowing them to make informed decisions on whether to proceed or address the issues.

How it works?

- Developer submits a pull request to add a new feature or update dependencies.
- Snyk automatically runs a security scan on the changes in the PR.
- If vulnerabilities are found, Snyk adds a comment to the PR with the findings.
- Depending on the severity of the issues, the PR can be marked as "failed" or "passed."
- The developer can then fix the issues and resubmit the PR, or the team can review and decide how to handle the vulnerability.

Snyk Test for Pull Requests

Snyk provides actionable fix advice for vulnerabilities in your Open Source libraries through the following:

- **Automated Snyk PRs:** automatic pull and merge requests
- **Manual Snyk PRs:** manual pull and merge requests

Snyk Test for Pull Requests

Manual Snyk PRs:

Follow these steps to generate a PR or MR directly from your Project in the Snyk Web UI:

- Navigate to your Project from the Project list
- Select the Project.
- Select Open a Fix PR/MR or Fix this vulnerability. A preview screen appears, showing you what fixes will be applied.
- Click Open a Fix PR on this screen to generate the pull request.

Snyk Test for Pull Requests: Manual

Manual Snyk PRs:

Follow these steps to generate a PR or MR directly from your Project in the Snyk Web UI:

- Navigate to your Project from the Project list
- Select the Project.
- Select Open a Fix PR/MR or Fix this vulnerability. A preview screen appears, showing you what fixes will be applied.
- Click Open a Fix PR on this screen to generate the pull request.

Snyk Test for Pull Requests: Manual

The screenshot shows the Snyk interface for a project named "CloudSihmar/pet-snyk" in the "master" branch. The "pom.xml" file is selected. The top navigation bar includes "Overview", "History", and "Settings". Below this, tabs for "Issues" (76), "Fixes" (selected), and "Dependencies" (87) are visible.

A central message reads: "Choose how to fix these vulnerabilities and open a pull request." A prominent green button labeled "Open a fix PR" is present.

The "Fix Advice" section is expanded, showing "Upgradable issues (11)":

- > Upgrade `org.webjars.jquery@2.2.4` to `org.webjars.jquery@3.5.0`
- > Upgrade `com.h2database:h2@1.4.200` to `com.h2database:h2@2.2.220`
- > Upgrade `mysql:mysql-connector-java@8.0.20` to `mysql:mysql-connector-java@8.0.28`

Snky Test for Pull Requests: Automatic

- **Automatic Pull Request:**
- **1. Global Level:**
 - Navigate to settings > Integrations.
 - Select an SCM integration, for example, GitHub.
 - Enable New vulnerabilities and Save.
- **2. Project Level:**
 - Under Projects, select a Project and select Settings.
 - Select an SCM integration, for example, GitHub.
 - In the Automatic fix pull requests section:
 - Select Customize for only this project
 - Enable New vulnerabilities
 - Select Save changes

Maturing your security with Snyk



**Fix
the backlog**

- Dedicate developer sprint time to fix issues
- Enable autofix PRs on projects

Maturing your security with Snyk

- **Dedicate Developer Time to Fixing Issues in Each Sprint**
- Starting from the next sprint, we want allocate a certain amount of developer time specifically to fixing security vulnerabilities and issues identified by Snyk.
- We want to ensure to factor in this time when planning your sprint tasks.
- **Introducing Snyk Autofix Pull Requests**
- We want to enable Snyk's Autofix Pull Requests (PRs) feature. This will automatically generate PRs for fixes where possible, allowing us to address certain issues faster.
- **Developer Responsibility:** While the autofix feature will automate some aspects, it is important that developers review these PRs and ensure the fixes are appropriate and do not introduce regressions or conflicts. Manual intervention will still be needed for more complex issues.

Maturing your security with Snyk

Automatic fix PRs

Specify which issues Snyk should open scheduled automatic PRs for.

 [More info](#)

New vulnerabilities

Known vulnerabilities (backlog)

Fix all vulnerabilities for the same dependency in a single PR

This may cause larger jumps between dependency versions. Default behavior is one PR per vulnerability

Save



Maturing your security with Snyk



**Optimize
security**

- Integrate testing results to other business applications
- Gamify security fixes
- Communicate performance to executives
- Plan security as part of new products or features

Notifications

Issue alert emails

By default, we send you an email when we find a new vulnerability, license issue, or remediation for an issue in your projects the same day we find it.

Tailor your notifications by selecting the types you wish to receive for each organization and project, and whether you want to be alerted the same day or that we find an issue, or if you'd prefer them grouped as a digest once a week.

These are default settings for all projects in this organization, which can be overridden by all users in their [account settings](#).

ON/OFF	ALERT TYPE	SEVERITY LEVEL
<input checked="" type="checkbox"/> On	<input checked="" type="checkbox"/>  Vulnerabilities <input type="checkbox"/>  License violations	All severities 

Weekly report

Weekly report RECOMMENDED

Get a weekly summary of your vulnerability status across all of the projects and organizations you're in.

Usage alerts

Approaching test limit RECOMMENDED

Get warnings about your usage, such as when are about to reach your monthly test limit.

Report status

Report ready to view 

Integrating Snyk

- **Snyk across the SDLC**
- **Snyk tests and monitors your projects for issues:**
- Snyk recommends different approaches, depending on your situation and the languages of the projects you want to test.
- You can choose to implement Snyk at different points in your software development life cycle.
- **Code:**
- To shift left and fix while developing, encourage developers to check for vulnerabilities and license issues as part of the development process. Developers can use the CLI, an IDE plug-in, or the Snyk web application.
- **Merge:**
- Set up Snyk to check for vulnerabilities and license issues when anyone opens a new pull request in the repository. Use policies to prevent pull requests. Create Jira tickets for new vulnerabilities.

Integrating Snyk

- **Snyk across the SDLC**
- **Repository**
- Use automations to create PRs with the required upgrade or patch based on your requirements.
- No changes are made to code without human approval.
- **Build**
- Integrate Snyk with CI/CD tools. Use automated Snyk tests or create policies to help prevent vulnerabilities from passing through build processes.
- **Production**
- Monitor the runtime environment with continuous tests to verify that there's no exposure to existing vulnerabilities and monitor for newly disclosed vulnerabilities.

Snyk Integration Points

Coding tools

A sampling of our coding integrations:

- Snyk CLI
- VS Code
- IntelliJ
- PhpStorm
- GoLand
- Eclipse
- WebStorm
- PyCharm
- Visual Studio 2019,2022
- Docker

Source control integrations

A sampling of our source control manager integrations:

- GitHub
- Bitbucket
- GitLab
- Azure Repos

CI/CD integrations

A sampling of our CI/CD integrations:

- CircleCI
- BitBucket Pipelines
- AWS Code Pipeline
- Azure Pipelines
- TeamCity
- Jenkins
- SnykAPI

Container Orchestrators

A sampling of our runtime integrations:

- Kubernetes

Container Registries

A sampling of our registry integrations:

- Docker Hub
- Azure Container Registry (ACR)
- Amazon ECR
- Artifactory
- Google Container Registry
- Harbor
- Quay
- and many more

Package Repositories

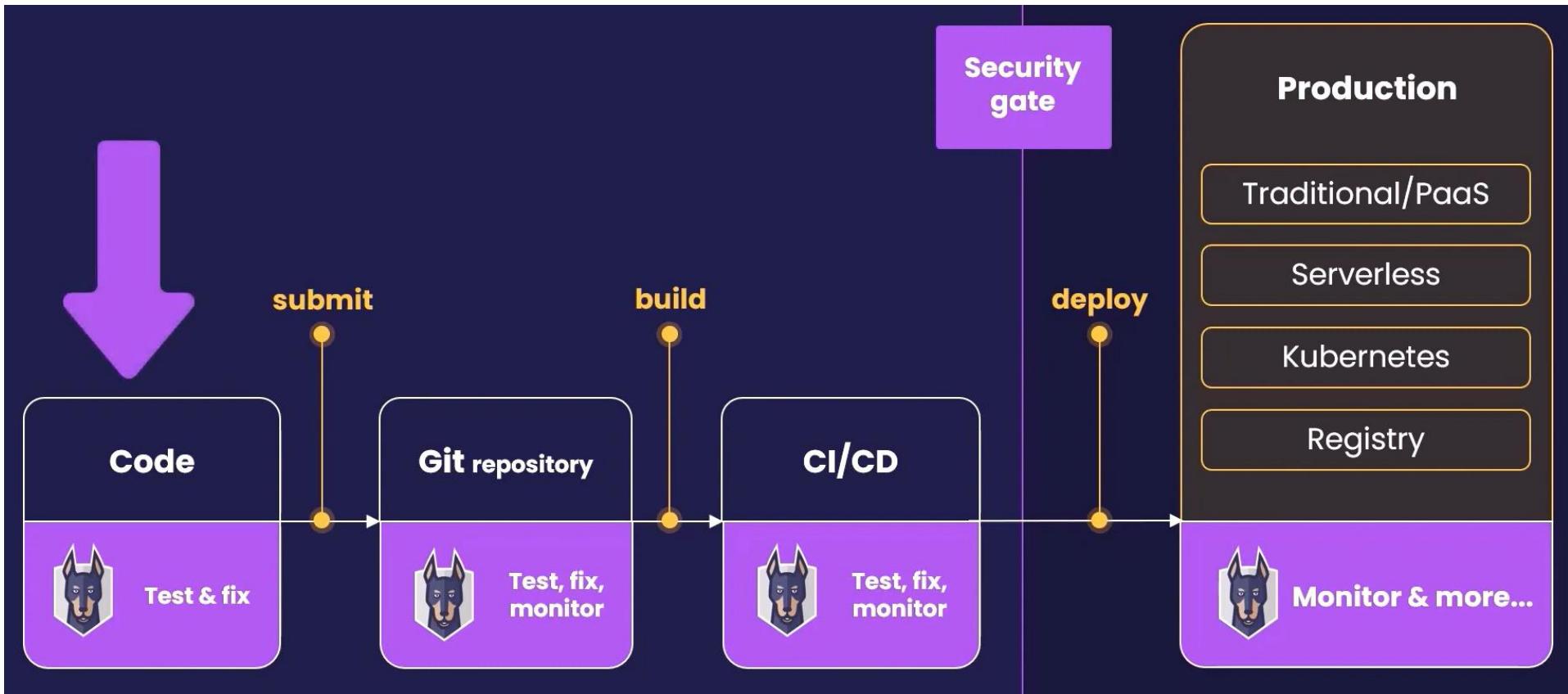
- Nexus
- Artifactory
- NPM

Issue management integrations

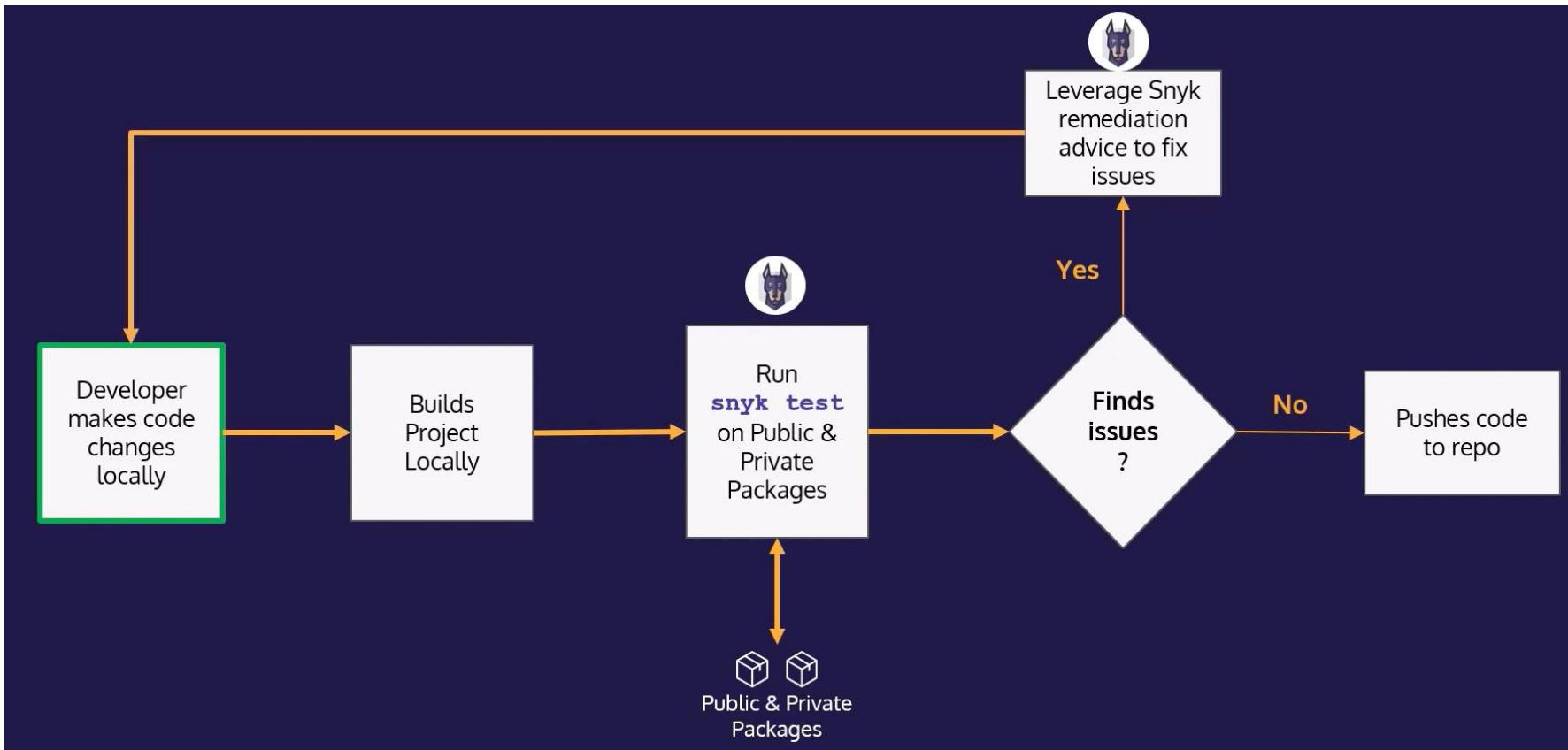
A sampling of our issue management tool integrations:

- Slack
- Jira
- Snyk API

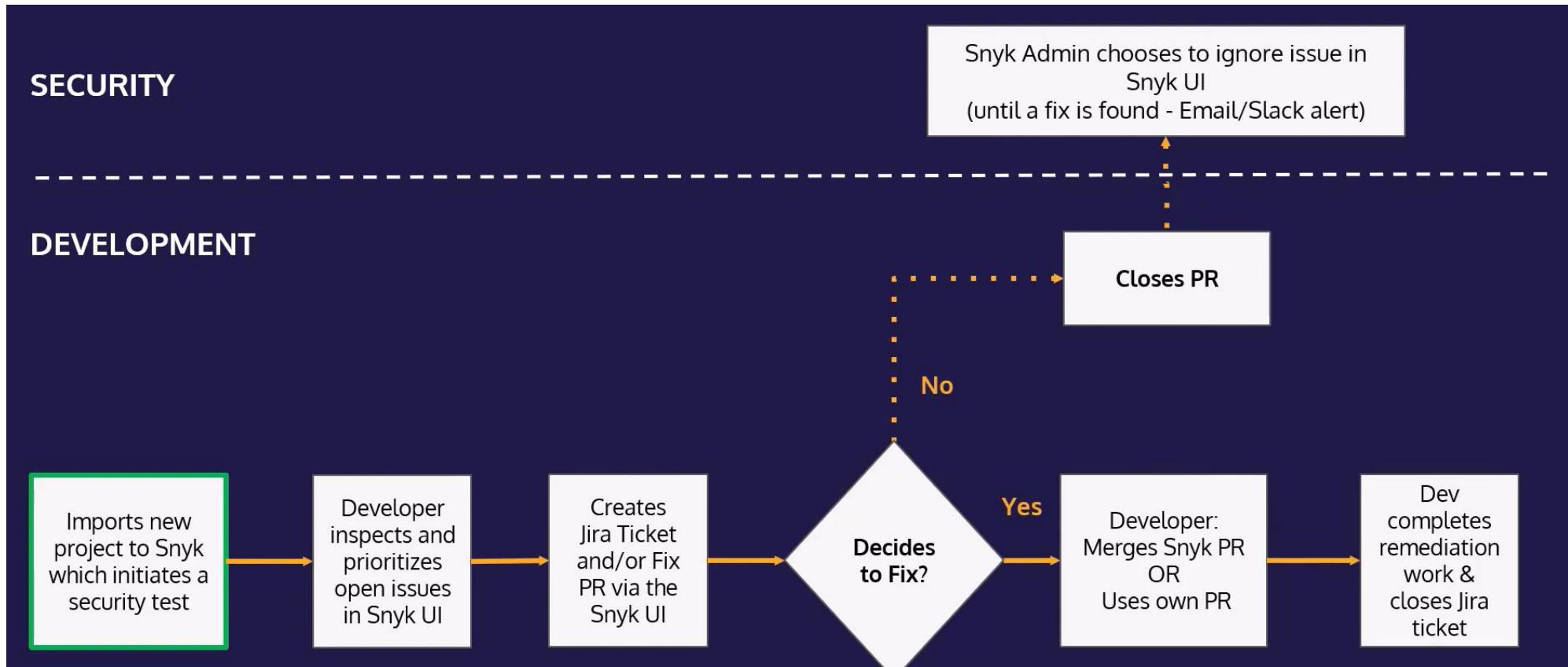
Synk test in CLI/IDE



Snyk test in CLI/IDE



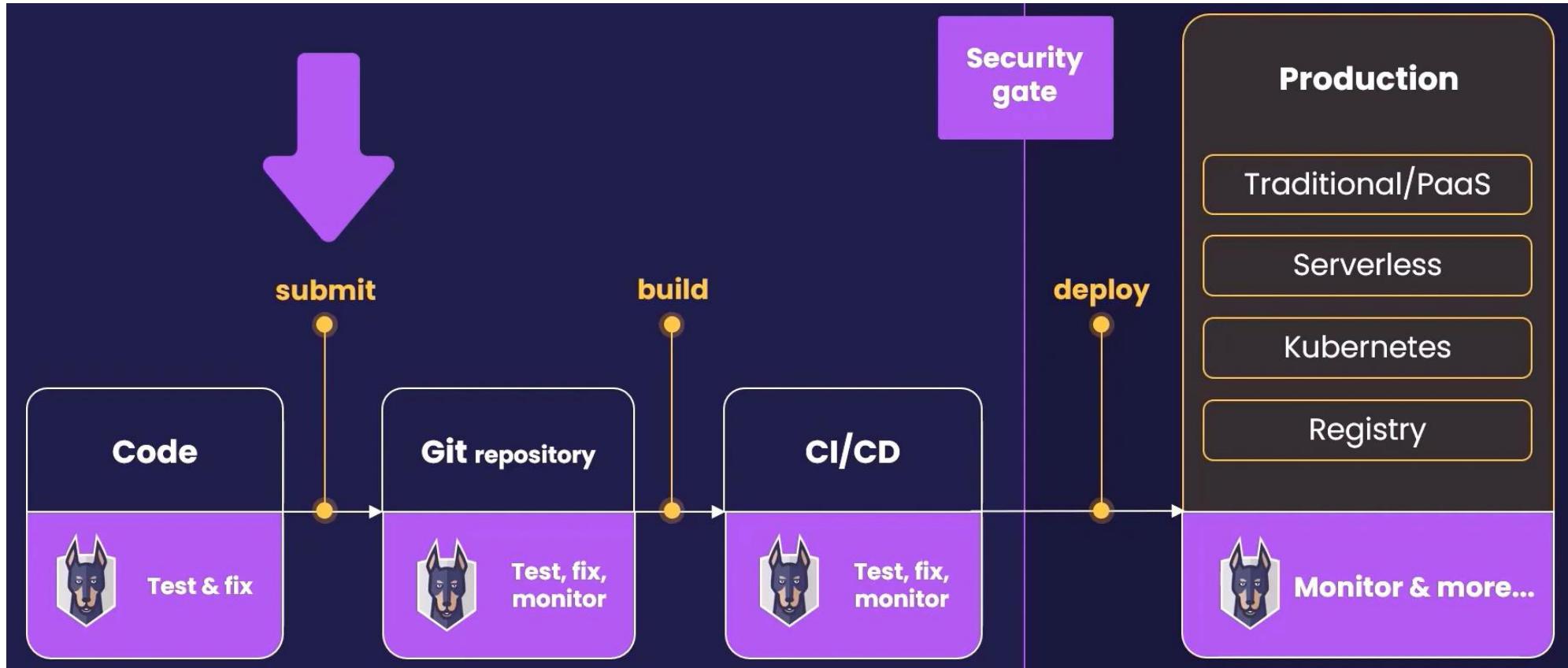
Synk Manual Fix PR Workflow



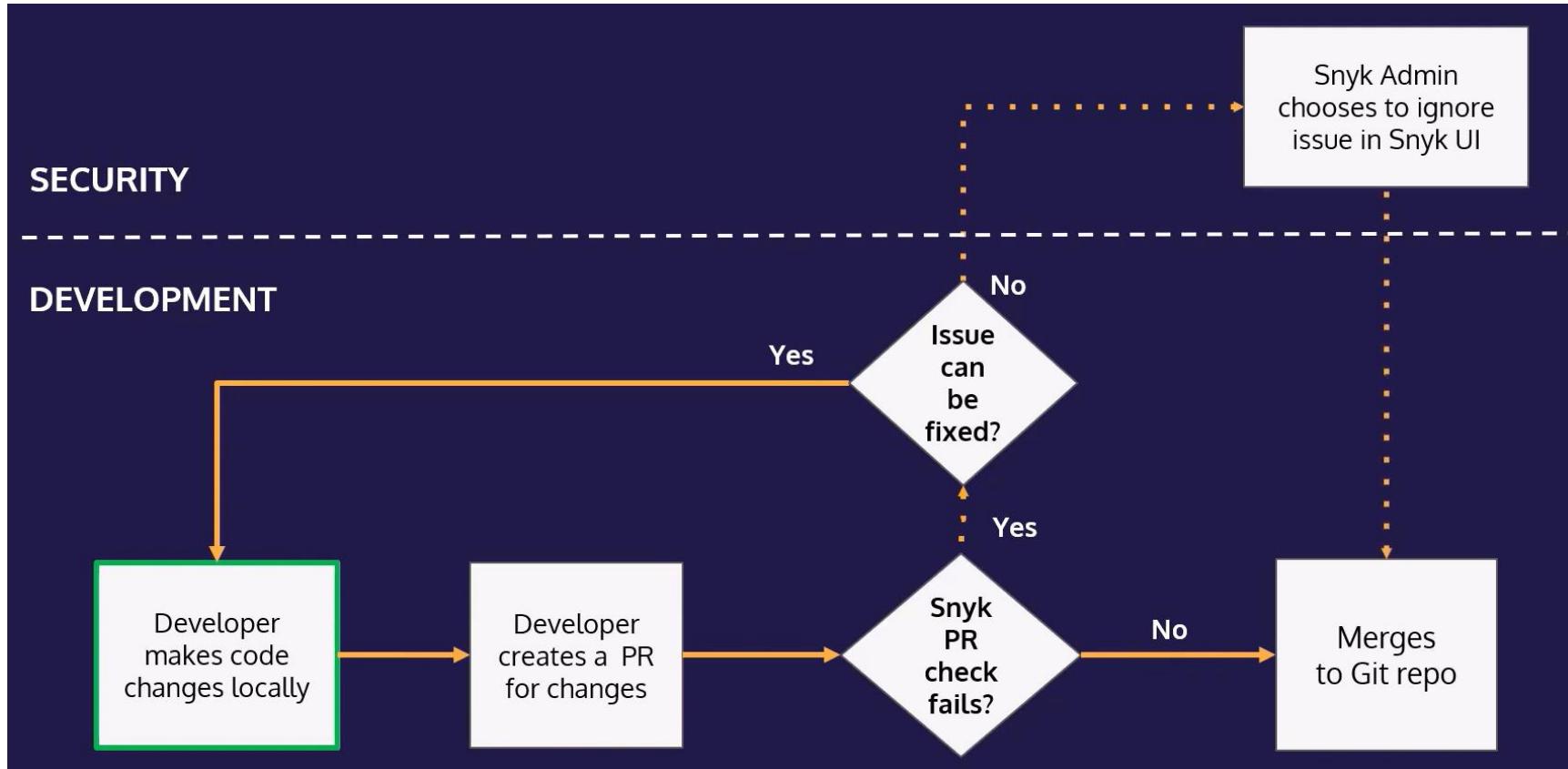
Git integration workflows

- Creating a mandatory security check before code is added to the repository
- Creating pull requests automatically if an issue Snyk identifies has a fix available
- Creating pull requests automatically if a dependency in use has an available upgrade

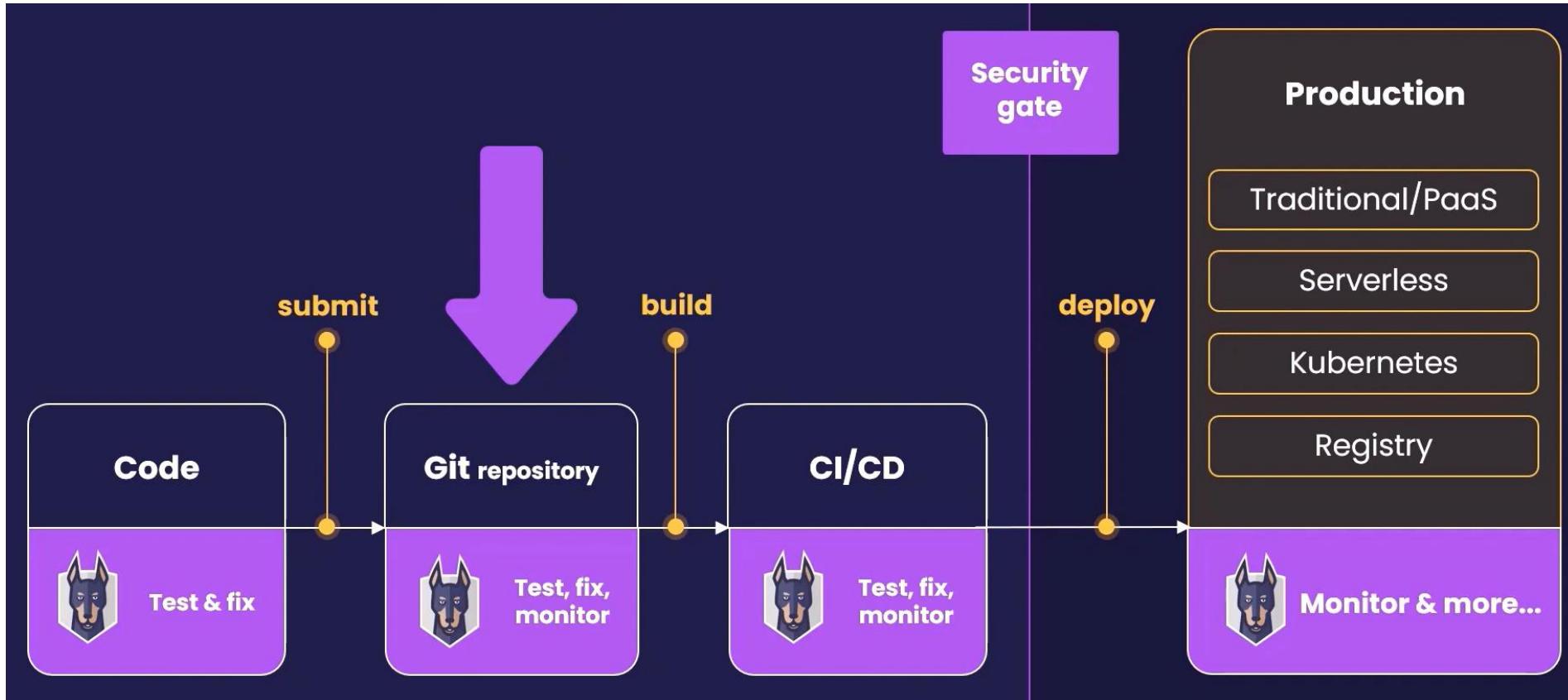
Git integration workflows



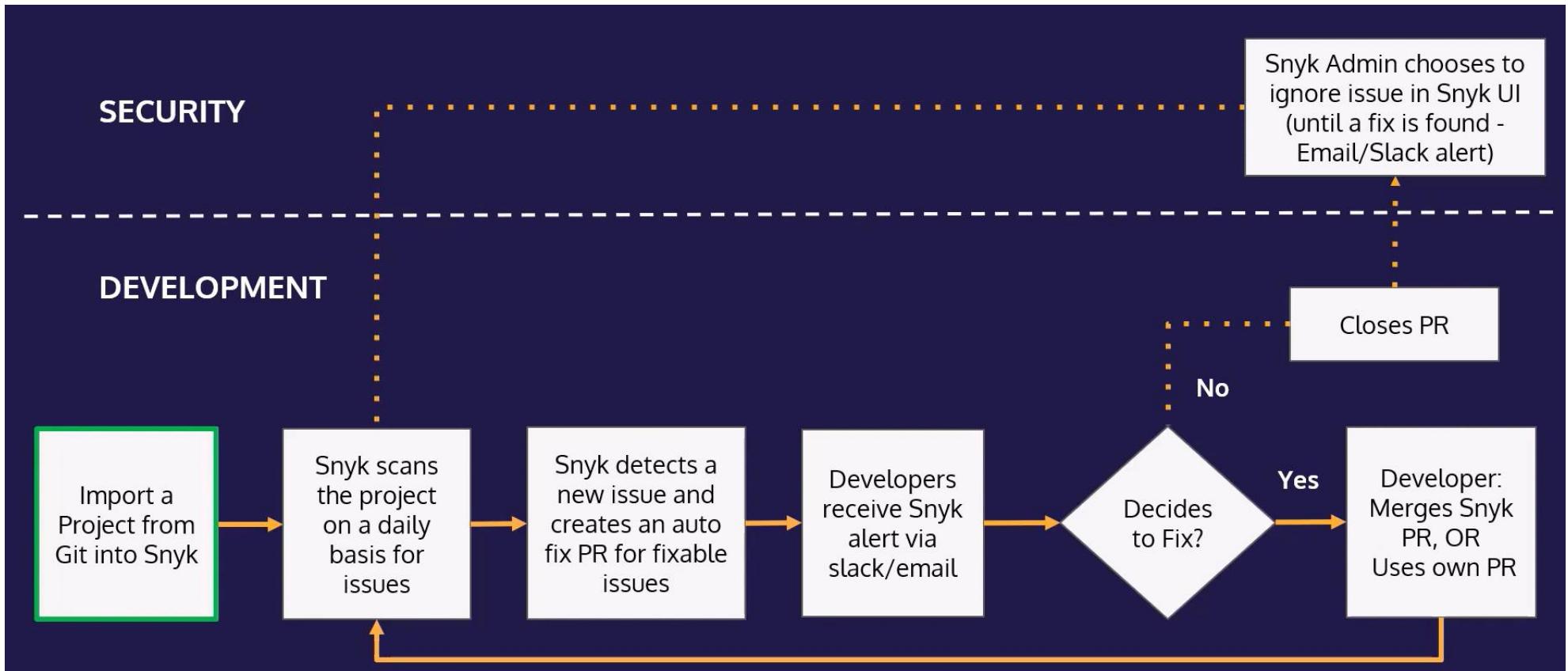
Default Snyk Test for Pull Request



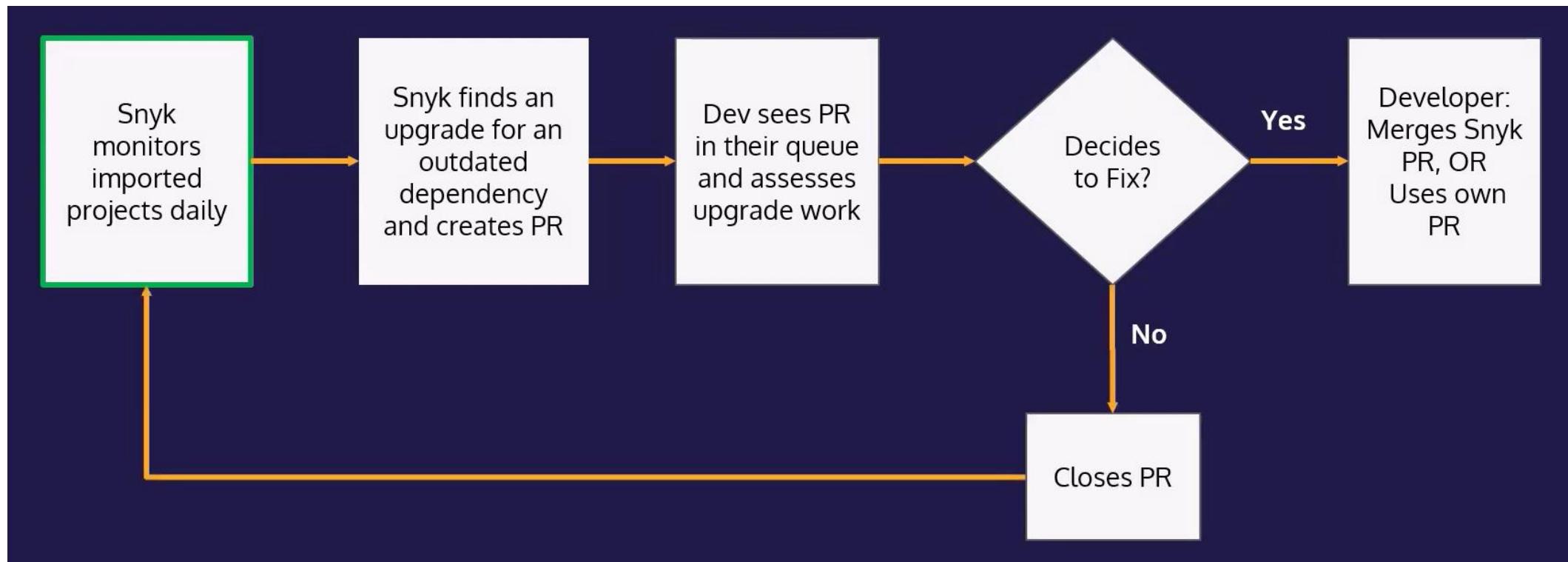
Automatic Snyk Test for Pull Request



Automatic Snyk Test for Pull Request



Automatic Dependency Upgrade Pull Request



Snyk integration with Visual Studio Code

- Snyk integrates with Visual Studio Code (VS Code) to help developers find and fix vulnerabilities in their code and open-source dependencies during the development process.
- **Steps to Integrate Snyk with VS Code:**
- **1. Install the Snyk Extension for Visual Studio Code**
- **2. Authenticate with Snyk:** Once the Snyk extension is installed, you need to authenticate with your Snyk account.
- **3. Analyze Code and Dependencies:**
 - Snyk Open Source: This checks your dependencies for vulnerabilities.
 - Snyk Code: This performs static analysis (SAST) to identify security issues in your code itself.
- **4. Review and Fix Issues:** Vulnerabilities found in the code or dependencies will be displayed in the Snyk view
- **5. Infrastructure as Code (IaC) and Containers:** The Snyk extension can also check Infrastructure as Code files (such as Terraform, Kubernetes YAML, etc.) for misconfigurations and vulnerabilities.
- **6. Automatic Scans on File Changes:** The Snyk extension continuously scans the code and dependencies in real time as you develop, providing feedback on new vulnerabilities as soon as they are introduced.

Benefits of Integrating Snyk with VS Code

- **Early Detection:**
 - Developers can find and fix security vulnerabilities early in the development process, reducing risk and avoiding issues later in the pipeline.
- **In-Context Fixes:**
 - Snyk provides recommendations and fixes directly within the code editor, making it easier to resolve security issues.
- **Comprehensive Coverage:**
 - Snyk scans your code, dependencies, container configurations, and infrastructure-as-code files all within VS Code.

Snyk CLI installation

1. MacOS

```
curl https://static.snyk.io/cli/latest/snyk-macos -o snyk  
chmod +x ./snyk  
mv ./snyk /usr/local/bin/
```

2. Windows

```
curl https://static.snyk.io/cli/latest/snyk-win.exe -o snyk.exe
```

3. Linux

```
curl https://static.snyk.io/cli/latest/snyk-linux -o snyk  
chmod +x ./snyk  
mv ./snyk /usr/local/bin/
```

Snyk CLI Commands

- **Scan for security issues:**
- Authenticate with Snyk

snyk auth or snyk auth 1b74bb97-5e51-4af4-a950-c3b065d8600d (API token)

Open Source Packages
1. Scan your projects locally
snyk test --all-projects --org=b1ecba5a-e67d-4bfa-8b27-4332218c99e4
2. Continuously monitor your projects and view the latest snapshots in the Web UI
snyk monitor --all-projects --org=b1ecba5a-e67d-4bfa-8b27-4332218c99e4

Snyk CLI Commands

- **Scan for security issues:**

Source code

To scan your **source code** for vulnerabilities

```
snyk code test --org=b1ecba5a-e67d-4bfa-8b27-4332218c99e4
```

Snyk CLI Commands

- **SAST scanning:**
- Steps:
 - 1. Clone the code
 - 2. Run SAST scanning
- snyk code test
- snyk code test -d
- snyk code test /root/p

```
root@ip-172-31-5-112:~/pet-snyk# snyk code test

Testing /root/pet-snyk ...

✗ [Medium] Path Traversal
Path: .mvn/wrapper/MavenWrapperDownloader.java, line 57
Info: Unsanitized input from a command line argument flows into exists, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to bypass the logic of the application in the conditional expression.

✗ [Medium] Path Traversal
Path: .mvn/wrapper/MavenWrapperDownloader.java, line 79
Info: Unsanitized input from a command line argument flows into exists, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to bypass the logic of the application in the conditional expression.

✗ [Medium] Path Traversal
Path: .mvn/wrapper/MavenWrapperDownloader.java, line 60
Info: Unsanitized input from a command line argument flows into java.io.FileInputStream, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to read arbitrary files.

✗ [Medium] Path Traversal
Path: .mvn/wrapper/MavenWrapperDownloader.java, line 80
Info: Unsanitized input from a command line argument flows into mkdirs, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to manipulate arbitrary files.

✗ [Medium] Path Traversal
Path: .mvn/wrapper/MavenWrapperDownloader.java, line 111
Info: Unsanitized input from a command line argument flows into java.io.FileOutputStream, where it is used as a path. This may result in a Path Traversal vulnerability and allow an attacker to write to arbitrary files.

▼ Test completed

Organization: sandy.2087
Test type: Static code analysis
Project path: /root/pet-snyk

Summary:
5 Code issues found
5 [Medium]
```

Snyk CLI Commands

- SAST scanning using Threshold

- Threshold:

- High
- Medium
- Low

- Steps:

- 1. Clone the code
 - 2. Run SAST scanning
- snyk code test --severity-threshold=high

```
root@ip-172-31-5-112:~/pet-snyk# snyk code test --severity-threshold=high
Testing /root/pet-snyk ...

✔ Test completed

organization:      sandy.2087
Test type:        Static code analysis
Project path:    /root/pet-snyk

Summary:

✔ Awesome! No issues were found.
```

Snyk CLI Commands

- **SAST scanning output in Json:**
- Steps:
 - 1. Clone the code
 - 2. Run SAST scanning
 - `snyk code test --json > sast.json`

Snyk CLI Commands

- **SAST scanning result and ignore issues**
- Steps:
 - 1. Clone the code
 - 2. Run SAST scanning
- `snyk code test --report --project-name=<PROJECT_NAME> --target-name=<TARGET_NAME>`
- If a Snyk Code Project created using the CLI does not yet exist for the value provided in the --project-name option, the Snyk CLI creates a new Project. If a Project created by using the CLI exists, a new snapshot is created under the same Project and under a target named <TARGET_NAME> and grouped by the "\$git branch --show-current)" branch name.
- Note: This feature is available in closed Beta.

Snyk CLI Commands

- **SAST scanning and exclude directories and files**
- The snyk ignore --file-path command does not ignore specific vulnerability issues. It excludes only directories and files from the CLI test.
- A file .snyk will be created in the current directory mentioning the file ignored
- `snyk ignore --id=<ISSUE_ID> [--expiry=] [--reason=]`
- `[--policy-path=<PATH_TO_POLICY_FILE>] [--path=<PATH_TO_RESOURCE>]`
- **Steps:**
- 1. Clone the code
- 2. ignore directory or a file
- `snyk ignore --file-path=src/main/java/org/t246osslab/easybuggy/vulnerabilities/NullByteInjectionServlet.java`
- 3. scan the code
- `snyk code test`

Snyk CLI Commands

- **SCA scanning and Ignore specific vulnerability**
- **Steps:**
 - 1. Clone the code
 - 2. Get the vulnerability ID
 - snyk code test --json > snyk-sast.json
 - 2. ignore directory or a file
 - `snyk ignore --id='SNYK-JAVA-LOG4J-2342645' --expiry='2025-01-10' --reason='This input is sanitized elsewhere'`
 - 3. scan the code
 - snyk test

Note: Ignoring issues or vulnerabilities using the .snyk file is not supported for Snyk Code

Snyk CLI Commands

- **SCA and SAST scanning and exclude directories and files**
- .snyk File

```
# Snyk (https://snyk.io) policy file, patches or ignores known vulnerabilities.
version: v1.25.0
# ignores vulnerabilities until expiry date; change duration by modifying expiry date
ignore:
  java/Sqli:
    - '*':
        reason: This input is sanitized elsewhere
        expires: 2025-01-10T00:00:00.000Z
        created: 2024-09-28T11:07:35.420Z
patch: {}
exclude:
  global:
    - >-
      src/main/java/org/t246osslab/easybuggy/vulnerabilities/NullByteInjectionServlet.java
```

Snyk CLI Commands

- Scan for security issues:

Containers

1. Scan your projects locally

```
snyk container test <repository>:<tag> --org=b1ecba5a-e67d-4bfa-8b27-4332218c99e4
```

2. Continuously monitor your projects and view the latest snapshots in the Web UI

```
snyk container monitor <repository>:<tag> --org=b1ecba5a-e67d-4bfa-8b27-4332218c99e4
```

Snyk CLI Commands

- Scan for security issues:

IAC

Scan your **Infrastructure as Code (IaC)** files for vulnerabilities

```
snyk iac test --report --org=b1ecba5a-e67d-4bfa-8b27-4332218c99e4
```

Scan your **Infrastructure as Code (IaC)** files for vulnerabilities and create a project on web UI

```
snyk iac test --report --org=b1ecba5a-e67d-4bfa-8b27-4332218c99e4 --target-name=pet-terraform
```

Snyk CLI Tools

- Snyk-delta
- Snyk-filter
- Snyk-to-html
- Snyk-scm-contributor-count

Snyk CLI Tools: Snyk-delta

Snyk-delta:

Snyk-delta is a tool that helps track changes in vulnerabilities, license issues, and dependencies between two snapshots taken at different points in time. It provides insight into how the state of security has changed between two runs of a Snyk scan.

How It Works:

- Baseline Snapshot:
 - This is the result of a previous Snyk scan that serves as a baseline for comparison.
- New Snapshot:
 - This is the result of the current Snyk scan, which will be compared to the baseline.
- Delta Calculation:
 - snyk-delta compares the vulnerabilities, license issues, and dependencies between these two snapshots, providing a report of the differences.

Snyk CLI Tools: Snyk-delta

Snyk-delta:

- Prerequisites
 - Snyk Enterprise plan (requires API)
 - Your project to be monitored

Commands

1. Get the output of snyk test into snyk-delta

```
snyk test --json --print-deps | snyk-delta
```

2. To point to a specific baseline snapshot, use the --baselineOrg and --baselineProject flags:

```
snyk test --json --print-deps | snyk-delta --baselineOrg xxx --baselineProject xxx --setPassIfNoBaseline true
```

Snyk CLI Tools: Snyk-filter

Snyk-filter:

The snyk-filter tool provides custom filtering for Snyk CLI output. snyk-filter takes the JSON-formatted output from the Snyk CLI, for example, snyk test --json and applies custom filtering of the results, as well as options to fail your build.

Key Features of snyk-filter:

- **Filtering based on severity:**
 - You can filter out vulnerabilities based on severity levels (low, medium, high, critical), allowing you to focus on the most critical issues.
- **Filtering based on issue types:**
 - It allows you to focus on specific types of issues, such as security vulnerabilities, license issues, or specific types of vulnerabilities (e.g., SQL injection, XSS).
- **Custom filters:**
 - You can define your custom filtering rules to narrow down the scope of the scan results based on your project's specific needs.
- **Integration with Snyk CLI:**
 - snyk-filter is commonly used alongside Snyk CLI tools, making it easier to integrate into CI/CD pipelines and automate the process of filtering vulnerabilities.
- **Filtering for specific dependencies:**
 - You can filter the results based on particular dependencies or packages, which is useful when you are concerned about certain libraries or modules in your project.

Snyk CLI Tools: Snyk-filter

Snyk-filter

Installation

```
git clone https://github.com/snyk-labs/snyk-filter.git  
cd snyk-filter  
apt update -y && apt install npm -y  
  
# install jq ahead of time (ubuntu example)  
sudo apt-get install -y jq  
  
# tell node-jq to skip trying to install it on its own  
export NODE_JQ_SKIP_INSTALL_BINARY=true  
  
# tell node-jq where the existing jq binary is  
export JQ_PATH=$(which jq)  
  
# finally, install snyk-filter (does not work with node version > 12)  
sudo npm install -g snyk-filter  
npm list -g snyk-filter
```

Snyk CLI Tools: Snyk-filter

Snyk-filter:

```
/root/snyk-filter/sample-filters/example-cvss-9-or-above.yml
```

```
version: 2
customFilters:
  filter: ".vulnerabilities |= map(if .cvssScore >= 9 then . else empty end)"
  pass: "[.vulnerabilities[] | select(.cvssScore >= 9)] | length"
  msg: "Vulnerabilities with CVSS Score of 9+ found"
```

Snyk CLI Tools: Snyk-filter

Snyk-filter:

```
/root/snyk-filter/sample-filters/example-vulns-only.yml
```

```
version: 2
customFilters:
  filter: ".vulnerabilities |= map(if .type != \"license\" then . else empty end)"
  pass: "[.vulnerabilities[] | select(.type != \"license\")] | length"
  msg: "Vulnerabilities found"
```

Snyk CLI Tools: Snyk-filter

Snyk-filter:

```
/root/snyk-filter/sample-filters/example-more-than-n-vulns
```

```
#This filter fails if there are: >2 high, upgradable vulns OR >0 critical, upgradeable vulns
```

```
version: 2
```

```
customFilters:
```

```
    filter: "if ([.vulnerabilities[] | select(.isUpgradable == true and .severity == \"high\")] | length > 2) or  
([.vulnerabilities[] | select(.isUpgradable == true and .severity == \"critical\")] | length > 0) then  
.vulnerabilities |= map(if .isUpgradable == true and (.severity == \"high\" or .severity == \"critical\") then .  
else empty end) else .vulnerabilities |= map(empty) end"
```

```
    pass: "if ([.vulnerabilities[] | select(.isUpgradable == true and .severity == \"high\")] | length > 2) or  
([.vulnerabilities[] | select(.isUpgradable == true and .severity == \"critical\")] | length > 0) then 1 else 0  
end"
```

```
    msg: "High and critical severity & upgradeable vulns found. Please review upgrade steps"
```

Snyk CLI Tools: Snyk-filter

Snyk-filter:

1. Install Maven

```
apt install maven -y
```

2. Snyk authentication

```
snyk auth 1b74bb97-5e51-4af4-a950-c3b065d8600d
```

3. Run snyk-filter

```
snyk test --json | snyk-filter -f /root/snyk-filter/sample-filters/example-cvss-9-or-above.yml
```

4. Run for IAC

```
snyk iac test --json | snyk-filter -f /root/snyk-filter/sample-filters/example-iac-high-or-above-issues.yml
```

Snyk CLI Tools: Snyk-filter

Snyk-filter:

Result:

```
root@ip-172-31-1-132:~/devsecops-jenkins-k8s-tf-sast-sca-sonarcloud-snyk-repo# snyk test --json | snyk-filter -f /root/snyk-filter/sample-filters/example-cvss-9-or-above.yml

Testing snyk-filter...
X Critical severity vulnerability found on commons-fileupload:commons-fileupload@1.3.1
- desc: Arbitrary Code Execution
- info: https://snyk.io/vuln/SNYK-JAVA-COMMONSFILEUPLOAD-30401
- from: org.t246osslab.easybuggy:easybuggy@1-SNAPSHOT > org.owasp.esapi:esapi@2.1.0.1 > commons-fileupload:commons-fileupload@1.3.1
Upgrade direct dependency org.owasp.esapi:esapi@2.1.0.1 to org.owasp.esapi:esapi@2.2.0.0 (triggers upgrades to commons-fileupload:commons-fileupload@1.3.3)

X Critical severity vulnerability found on log4j:log4j@1.2.13
- desc: Deserialization of Untrusted Data
- info: https://snyk.io/vuln/SNYK-JAVA-LOG4J-572732
- from: org.t246osslab.easybuggy:easybuggy@1-SNAPSHOT > org.slf4j:slf4j-log4j12@1.5.0 > log4j:log4j@1.2.13
Upgrade direct dependency org.slf4j:slf4j-log4j12@1.5.0 to org.slf4j:slf4j-log4j12@1.7.34

X Critical severity vulnerability found on xalan:xalan@2.7.0
- desc: Arbitrary Code Execution
- info: https://snyk.io/vuln/SNYK-JAVA-XALAN-2953385
- from: org.t246osslab.easybuggy:easybuggy@1-SNAPSHOT > org.owasp.esapi:esapi@2.1.0.1 > xom:xom@1.2.5 > xalan:xalan@2.7.0
Upgrade direct dependency org.owasp.esapi:esapi@2.1.0.1 to org.owasp.esapi:esapi@2.2.0.0

Organisation: sandy.2087
Licenses enabled

Tested 35 dependencies for known issues
org.t246osslab.easybuggy:easybuggy - Vulnerabilities with CVSS Score of 9+ found
Snyk Test Failed
```

Snyk CLI Tools: Snyk-to-html

Snyk-to-html:

The snyk-to-html tool is useful for converting Snyk's JSON output into human-readable HTML reports. This allows you to generate

Installation

```
npm install snyk-to-html -g
```

Snyk CLI Tools: Snyk-to-html

Snyk-to-html:

Commands

1. Snyk Open Source command

```
snyk test --json | snyk-to-html -o results-opensource.html
```

2. Snyk Container command

```
snyk container test [image] --json | snyk-to-html -o results-container.html
```

3. Snyk Code command

```
snyk code test --json | snyk-to-html -o results-code.html
```

4. Snyk IaC command

```
snyk iac test --json | snyk-to-html -o results-iac.html
```

Snyk CLI Tools: Snyk-to-html

Snyk-to-html:

Commands

- snyk-to-html -i results-opensource.json -o results-opensource.html
- snyk-to-html -i results-code.json -o results-code.html
- snyk-to-html -i results-container.json -o results-container.html
- snyk-to-html -i results-iac.json -o results-iac.html

Snyk CLI Tools: Snyk-to-html

Snyk-to-html:

Short	Long	Description	Default
-i	--input	The input path of the JSON or SARIF file that contains the test results. SARIF format is not supported for open source scan results	stdin
-o	--output	Precedes the name of the output file of the HTML results. Example: -o results.html	stdout
-t	--template	Template location for generating the HTML.	template/test-report.hbs
-s	--summary	Generates an HTML file with only the summary instead of the details report.	Details vulnerability report
-a	--actionable-remediation	Display actionable remediation info if available.	Not applicable
-d	--debug	Run the command in debug mode.	Not applicable

Snyk CLI Tools: Snyk-to-html

Snyk-to-html:

Command	Description
snyk-to-html -i results.json -o results.html -s	Generates a simple HTML report with only the summary of vulnerabilities.
snyk-to-html -i results.json -o results.html -a	Generates an HTML report including actionable remediation steps for the vulnerabilities.

Snyk CLI Tools: Snyk-to-html

Snyk-to-html:

Capture the exit code

You can capture the exit code of the Snyk command in a variable, preventing it from directly impacting the overall process.

```
#!/bin/bash

# Run Snyk test and output the results to a JSON file
snyk test --json > result-opensource.json
exit_code=$?

# Check if the exit code indicates success or failure
if [ $exit_code -ne 0 ]; then
  echo "Snyk test failed with exit code $exit_code. Review the results before proceeding."
  exit 1
else
  # Convert the JSON results into an HTML report if the test was successful
  snyk-to-html result-opensource.json -o snyk-report.html
  echo "Snyk test passed. HTML report generated at snyk-report.html"
fi
```

Snyk CLI Tools: Snyk-to-html

Snyk-to-html:

Use || true

Using || true allows the script to ignore the exit code of the command that came before it. This is a simple workaround but be cautious, as it will suppress all errors, potentially leading to problems later in your pipeline.

```
snyk test --json > result-opensource.json || true
```

```
# Continue with the next steps without terminating the script  
snyk-to-html result-opensource.json -o snyk-report.html
```

Snyk CLI Tools: snyk-scm-contributors-count

snyk-scm-contributors-count:

The snyk-scm-contributors-count tool is designed to provide insights into the activity and engagement within your software development projects by counting the number of contributors over the last 90 days. It supports various Source Code Management (SCM) systems, enabling you to analyze contributions from different platforms.

Supported SCMs:

- Azure DevOps
- Bitbucket Cloud
- Bitbucket Server
- GitHub
- GitHub Enterprise
- GitLab
- GitLab Server

Snyk CLI Tools: snyk-scm-contributors-count

snyk-scm-contributors-count:

Key Features:

1. Contributor Count:

- The tool retrieves and displays the number of unique contributors who have made commits to the repository in the last 90 days.

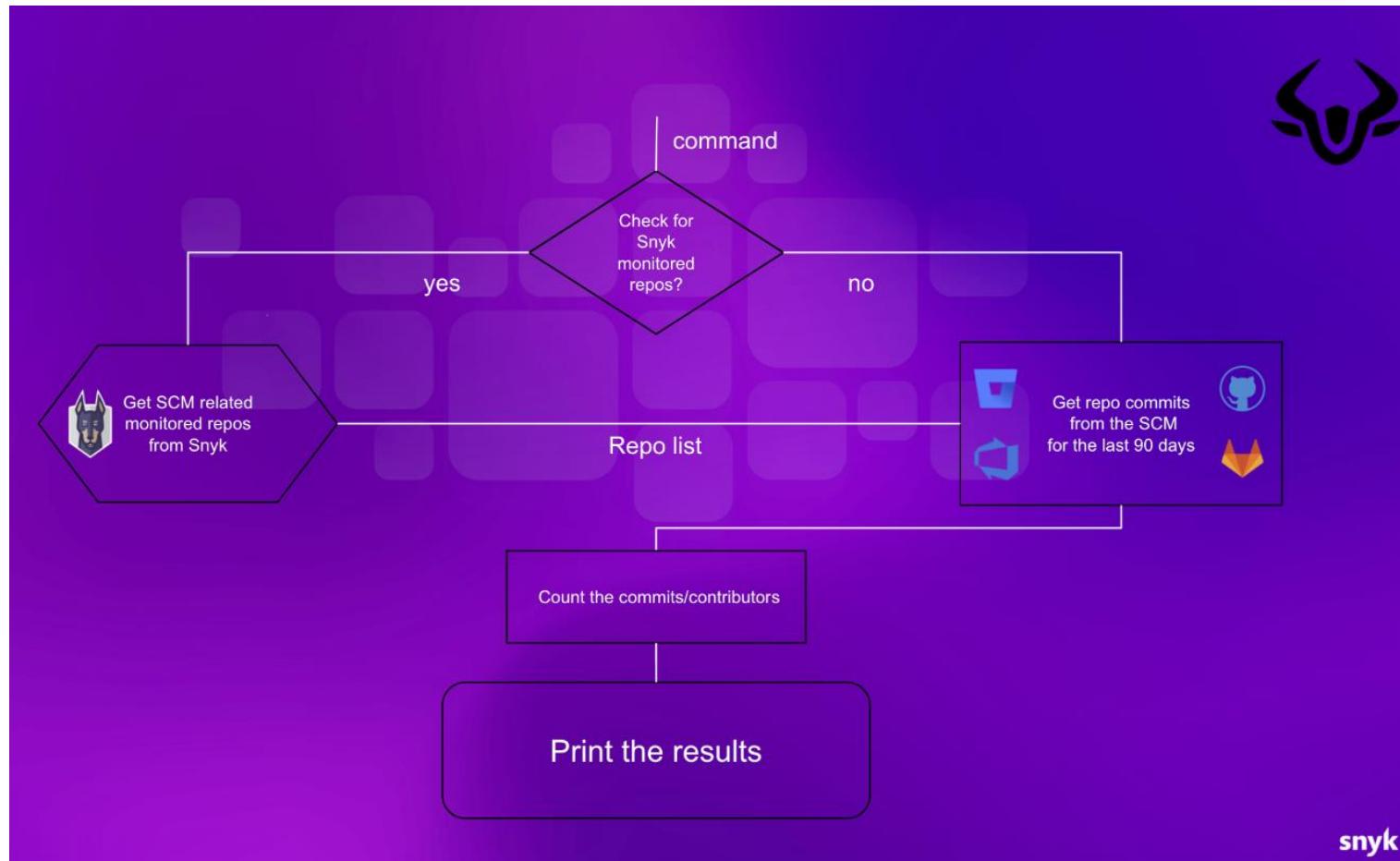
2. Summary Report:

- It prints a summary that highlights contributor activity, which can be useful for assessing team engagement and project health.

3. Cross-SCM Compatibility:

- It can work with multiple popular SCM systems, allowing you to track contributions across different repositories and services.

Snyk CLI Tools: snyk-scm-contributors-count



Snyk CLI for snyk container

- The snyk container test and snyk container monitor commands are part of the Snyk Container CLI, and they serve different purposes

1. snyk container test <repository>:<tag>

- Purpose:**

- This command scans a container image for vulnerabilities at the time you run the command. It analyzes the dependencies and libraries within the specified image.

- Use Case:**

- It's useful for testing an image before deployment. You would typically run this command during your CI/CD pipeline or manually to ensure that the image doesn't have any known vulnerabilities.

- Output:**

- The command outputs a detailed report of any vulnerabilities found, including their severity, paths, and suggested fixes. It provides a snapshot of vulnerabilities at that moment.

Snyk CLI for snyk container

2. snyk container monitor <repository>:<tag>

- **Purpose:**
 - This command monitors a container image for vulnerabilities over time. It sends the specified image to Snyk's dashboard for ongoing monitoring.
- **Use Case:**
 - After deploying an image, you can use this command to keep track of new vulnerabilities that may be discovered after the initial scan. It allows for continuous monitoring of your images.
- **Output:**
 - The command does not output vulnerabilities immediately but rather sets up monitoring. You'll receive notifications via the Snyk dashboard or through configured alerts if any new vulnerabilities are discovered in the monitored image.

Snyk CLI for snyk container

- **Scan archives with Snyk:**
- Snyk allows users to scan and monitor Docker or OCI (Open Container Initiative) archives directly, providing a versatile approach to vulnerability management without needing to load images into a

Commands

```
snyk container test docker-archive:archive.tar  
snyk container test oci-archive:archive.tar
```

-

Snyk CLI for snyk container

- **Testing Multi-Platform Images with Snyk:**
- The Snyk CLI allows users to scan multi-manifest container images explicitly for specific platforms. This functionality is essential for identifying vulnerabilities across various operating systems and architectures, ensuring comprehensive security coverage.
- **Multi-Manifest Repositories:**
 - Some repositories contain multiple manifests pointing to different images based on the operating system and architecture (e.g., Linux, ARM, etc.).
 - This enables support for diverse environments.
- **Explicit Platform Scanning:**
 - Users can specify the platform using the --platform option in the Snyk CLI.
 - `snyk container test --platform=linux/arm64 debian`

Snyk CLI for snyk container

- **Testing Multi-Platform Images with Snyk:**
- **Supported Platforms:**
 - The --platform option accepts the following values:
 - linux/amd64
 - linux/arm64
 - linux/riscv64
 - linux/ppc64le
 - linux/s390x
 - linux/386
 - linux/arm/v7
 - linux/arm/v

Snyk CLI for snyk container

- **Testing Multi-Platform Images with Snyk:**

Authentication to Remote Container Registries:

- With Docker: Utilizes pre-configured registry authentication.
- Without Docker: Pass credentials via command line or environment variables.
 - Environment Variables:
 - SNYK_REGISTRY_USERNAME
 - SNYK_REGISTRY_PASSWORD
 - Command Line Authentication
 - `snyk container test <repository>:<tag> --username=<your_username> --password=<your_password>`

Snyk CLI for snyk container

Enhanced Functionalities:

- **--json**: Outputs results in JSON format for integration with other tools.
- **--sarif**: Outputs results in SARIF format (only available with container test).
- **--exclude-base-image-vulns**: Excludes vulnerabilities from the base image (only available with container test).
- **--severity-threshold**: Sets the severity level to filter results (only available with container test).
- **--exclude-app-vulns**: Excludes application vulnerabilities.
- **--nested-jars-depth**: Specifies the depth for scanning nested JAR files.
- **--fail-on**: Sets conditions for failing the scan (only available with container test).
 - all | upgradable | patchable

Snyk CLI for snyk container

Commands

```
snyk container test cloudsihmar/pet:5 --file=Dockerfile --exclude-base-image-vulns
```

```
snyk container test --fail-on=all cloudsihmar/pet:5 --file=Dockerfile
```

```
snyk container test --fail-on=patchable cloudsihmar/pet:5 --file=Dockerfile
```

```
snyk container test --fail-on=upgradable cloudsihmar/pet:5 --file=Dockerfile
```

```
snyk container test --exclude-app-vulns cloudsihmar/pet:5
```

Snyk CLI for IAC

Snyk Infrastructure as Code (IaC) enables users to test their configuration files for vulnerabilities and best practices through the command line interface (CLI). This ensures that infrastructure setups are secure from the outset.

- Overview of Snyk IaC Testing:
 - Allows users to validate and secure configuration files for various cloud environments.
 - Ensures compliance with security best practices early in the development lifecycle.
- Using the CLI:
 - `snyk iac test`
- Share CLI result with Web UI
 - `snyk iac test myproject --report`

Snyk CLI for IAC

Supported Configuration Files:

Snyk can test a wide range of IaC files, including:

- **Terraform Files:** Validate infrastructure configurations written in Terraform.
- **CloudFormation Files:** Test AWS CloudFormation templates for security issues.
- **AWS CDK Files:** Assess configurations created with the AWS Cloud Development Kit.
- **Kubernetes Files:** Analyze Kubernetes resource definitions for vulnerabilities.
- **ARM Files:** Evaluate Azure Resource Manager templates.
- **Kustomize Files:** Check configurations managed with Kustomize.
- **Helm Charts:** Test Helm charts used for deploying applications on Kubernetes.
- **Serverless Files:** Validate configurations for serverless applications.

Snyk CLI for IAC

Functionality	Command
Test for all issues in the current directory	<code>snyk iac test</code>
Test for specific files	<code>snyk iac test file-1.tf dir/file-2.tf</code>
Test for issues in a specific directory	<code>snyk iac test my-folder</code>
Limit directory traversal depth	<code>snyk iac test --detection-depth=3</code>
Output results in JSON format	<code>snyk iac test --json</code>
Output results in SARIF format	Output results in SARIF format
Save SARIF output to a file	<code>snyk iac test main.tf --sarif-file-output=snyk.sarif</code>
Display issues above a specific severity level	<code>snyk iac test --severity-threshold=medium</code>
Target a specific Snyk Organization	<code>snyk iac test --org=infrastructure</code>
Set default Organization in Snyk config	<code>snyk config set org=infrastructure</code>

Snyk CLI for IAC

Functionality	Command	Description
Ignore Issues	- Use Snyk Web UI - Create a .snyk policy file	Issues can be ignored via the Snyk Web UI or by creating a .snyk policy file with your project during scanning.
Project Tags	snyk iac test myproject --report --project-tags=department=platform,team=persistence	Attaches tags to the scanned project; valid only when used with --report.
Project Business Criticality	snyk iac test myproject --report --project-business-criticality=high	Sets business criticality for the scanned project; values can be critical, high, medium, low.
Project Environment	snyk iac test myproject --report --project-environment=frontend,internal	Specifies the project environment; values can be frontend, backend, internal, external, mobile, saas, onprem, hosted, distributed.
Project Lifecycle	snyk iac test myproject --report --project-lifecycle=development	Sets the project lifecycle; values can be production, development, sandbox.
Set Target Reference	snyk iac test myproject --report --target-reference="\$(git branch -show-current)"	Sets the target reference for the scanned project to the current Git branch name; valid only when used with --report.

Snyk CLI for IAC

- **Ignoring Issues with the .snyk Policy File**
- You can ignore non-relevant issues in your IaC configuration files by using the .snyk policy file during the snyk iac test command.
- Store and version the .snyk file in the root of your working directory with IaC configuration files.
- The .snyk file can be created using the snyk ignore command.

Snyk CLI for IAC

- Ignore rule syntax

.snyk file

```
version: v1.19.0
ignore:
  <VULNERABILITY-ID>:
    - '*':
      reason: None Given
      expires: YYYY-MM-DDTHH:MM:SS.sssZ
      created: YYYY-MM-DDTHH:MM:SS.sssZ
```

Snyk CLI for IAC

- **Ignore specific file**

.snyk file

```
version: v1.19.0
ignore:
  SNYK-CC-K8S-1:
    - 'staging/deployment.yaml > *':
        reason: None Given
        expires: 2021-08-26T08:40:35.249Z
        created: 2021-07-27T08:40:35.251Z
    - 'staging/cronjob.yaml > *':
        reason: None Given
        expires: 2021-08-26T08:40:35.249Z
        created: 2021-07-27T08:40:35.251Z
```

Snyk CLI for IAC

- **Ignoring Individual Instances of a Vulnerability**

Command

```
snyk ignore --id=SNYK-CC-K8S-1 --path='production/deployment.yaml > [DocId:1] > spec > template > spec > containers[web] > securityContext > privileged'
```

Failing of builds in Snyk CLI

Option	Description	command
--severity-threshold=low medium high critical	Set the minimum severity level for vulnerabilities that will cause the build to fail.	snyk test --severity-threshold=high
--fail-on=all	Fail the build if there is at least one vulnerability that can be upgraded or patched.	snyk test --fail-on=all
--fail-on=upgradable	Fail the build if there is at least one vulnerability that can be upgraded.	snyk test --fail-on=upgradable
--fail-on=patchable	Fail the build if there is at least one vulnerability that can be patched or upgraded.	snyk --fail-on=patchable
Snyk delta	Fail current build only if new vulnerabilities are being introduced	snyk test --json --print-deps snyk-delta --baselineOrg xxx --baselineProject xxx
Snyk-filter	Fail build for CVSS score higher than a specified score with	snyk test --json snyk-filter -f /path/to/example-cvss-9-or-above.yml

Scanning Python package

1. Clone the code

<https://github.com/CloudSihmar/vul-python.git>

2. Install the packages

pip install -r requirements.txt

3. Run the scan

snyk test --file=requirements.txt --command=python3

snyk monitor --file=requirements.txt --command=python3

Automatic fixing with snyk-fix

- The snyk fix CLI command automatically applies the recommended updates for supported ecosystems. Ensure you use the latest version of CLI (v1.715.0 or later) to use snyk fix
- The command is supported only for Python.
- The snyk fix command is in Closed Beta and available only for Enterprise plans.
- To enable snyk fix during the beta period, click on Settings > Snyk Preview.

Usage examples

- `snyk fix --file=requirements.txt`
- `snyk fix --file=base.txt --package-manager=pip`
- `snyk fix --all-projects`

Automatic fixing with snyk-fix

The snyk fix command supports all the snyk test command options and has the following additional options:

- **--quiet** - Suppress all output to the command line.
- **--dry-run** - Run almost all the logic and display output, but do not make the final changes to the relevant files. Show a preview of the changes.
- **--sequential** - Install each dependency update separately one at a time (the default is to install all at once). The default is much slower, but helps increase the number of successful updates by allowing some updates to fail and the process to continue.

Ignore vulnerabilities using the Snyk CLI

OPTION	DESCRIPTION	DEFAULT	REQUIRED
--id	The Snyk ID for the issue to ignore. Found by running snyk test and grabbing the last segment of the URL for a given vulnerability. Example: For the vulnerability found at https://security.snyk.io/vuln/SNYK-DEBIAN10-NODETOUGHCOOKIE-5759362 , the Snyk ID is: SNYK-DEBIAN10-NODETOUGHCOOKIE-5759362 .	None	Yes
--expiry	Expiry date in YYYY-MM-DD format (RFC2822 and ISO 8601 are supported). Example: --expiry=2017-04-30.	30 days	No
--reason	Human-readable <REASON> to ignore this issue. Example: reason='Not currently exploitable'.	None	No
--policy-path=<PATH_TO_POLICY_FILE>	Path to a .snyk policy file to pass manually.	None	No
--path	Path to resource for which to ignore the issue. Example: path='tough-cookie@2.15.8'	All	No

Ignore vulnerabilities using the Snyk CLI

- **1. Snyk Open Source:**
- Default Behavior: The snyk ignore functionality works out of the box for open-source projects. You can ignore vulnerabilities using the .snyk policy file or through the Snyk Web UI.
- No special configuration is needed. Once an issue is ignored in the .snyk file or through the UI, future tests will respect these ignore rules without additional options.
- **2. Snyk Container:**
- Additional Policy Path Requirement: When you register an ignore (either in the .snyk file or through the Snyk UI), Snyk requires you to explicitly point to the .snyk policy file for the ignore rules to take effect in container tests.
- `snyk container test node --policy-path=.snyk`
- This command ensures that the .snyk policy file with your ignore rules is applied during container testing. Without specifying the --policy-path, the ignores won't be considered.

Ignore vulnerabilities using the Snyk CLI

- **3. Snyk Infrastructure as Code (IaC):**
- Snyk Ignore for IaC Projects: Ignore rules for IaC issues are handled via the `.snyk` policy file.
- For IaC, you can use the `snyk ignore` command to generate ignore rules and add them to the `.snyk` policy file. Once these are defined, Snyk will automatically apply them during IaC scans, but there is no need to use a `--policy-path` option (unlike in Snyk Container).
- **4. Snyk Code:**
- Excluding Files and Directories: For Snyk Code, the ignore functionality is slightly different. Instead of ignoring vulnerabilities like in Open Source or Container projects, you may exclude entire directories or files from Snyk Code analysis by defining exclusions.
- This exclusion behavior is defined in the CLI settings, and `.snyk` ignore rules are not directly applicable here in the same way they are for other product types.

Ignore vulnerabilities using the Snyk CLI

Command

```
snyk ignore --id=<ISSUE_ID> [--expiry=<EXPIRY>] [--reason=<REASON>] [--policy-path=<PATH_TO_POLICY_FILE>] [<OPTIONS>]
```

Scan all unmanaged Jar files

- The Snyk CLI can scan unmanaged JAR files in Java applications to identify vulnerabilities in the packages contained within them, but it can only identify packages if:
 - The package is available in Maven Central.
 - The JAR file hash matches the one in Maven Central.
- **Prerequisite:**
 - You must have a supported version of Maven installed to perform this scan.

Scan all unmanaged Jar files

- **To Scan All JAR Files in a Folder Individually:**
- You can use the following command to scan each unmanaged JAR file individually:
`snyk test --scan-unmanaged --file=/path/to/file`
- File Scanning: This command scans each JAR file in the specified path individually.
- Snyk Web UI: After scanning, the name of each scanned JAR file appears in the Snyk web UI.
- WAR Files:
 - If you have WAR files that are published in Maven Central, you can scan them directly.
 - To scan WAR or JAR files that contain other JAR files, you must extract (unzip) them first before scanning the open-source dependencies within them.
- By running the command, you can detect vulnerabilities in unmanaged JARs that are part of your Java project but not managed by a dependency manager like Maven or Gradle.

CICD adoption and deployment

Stages in adopting CI/CD integration

- Developer teams typically adopt Snyk in the following stages:
- Expose vulnerabilities (snyk monitor)
- Use Snyk as a gatekeeper (snyk test)
- Continuous monitoring (snyk test and snyk monitor)

Stages in adoptin CI/CD integration

- Developer teams typically adopt Snyk in the following stages:
- Expose vulnerabilities (snyk monitor)
- Use Snyk as a gatekeeper (snyk test)
- Continuous monitoring (snyk test and snyk monitor)

Stage 1: Expose Vulnerabilities

- **Objective:**
 - Introduce visibility of vulnerabilities during development without disrupting workflows.
- **Why Use Snyk Monitor?**
 - **Visibility:** Identifies and exposes vulnerabilities to the team early in the development cycle.
 - **No Disruption:** Initial use of snyk monitor helps avoid build failures while still highlighting issues.

Key Benefits:

- **Transparency:** Team members are informed about security issues without impacting build success.
- **Prevents Overwhelm:** Allows teams to gradually address vulnerabilities without overwhelming them with immediate failures.

Stage 1: Expose Vulnerabilities

- **Initial Setup Recommendation:**
 - Use **Snyk monitor** only—avoid failing builds in the early stages.
 - Configure CI plugins to not fail builds until the team is ready to handle vulnerabilities systematically.
- **Long-term Goal:**
 - As the team becomes comfortable managing vulnerabilities, start integrating stricter rules to fail builds once security measures are improved.

Stage 2: Use Snyk as a Gatekeeper

- **Objective:**
 - Prevent the introduction of new vulnerabilities by using Snyk as a security gatekeeper.
- **Why Use Snyk as a Gatekeeper?**
 - **Stop the Bleeding:** Prevents new vulnerabilities from entering your codebase.
 - **Proactive Protection:** Ensures security is built into your pipeline, reducing future risks.
- **Key Benefits:**
 - **Fail Builds on Vulnerabilities:** Automatically fails builds when new vulnerabilities are detected, ensuring secure code.
 - **Informed Decision-Making:** Results are provided in the console, allowing teams to decide whether to stop or continue the build.

Stage 2: Use Snyk as a Gatekeeper

- **When to Implement:**
 - Once teams are familiar with managing and addressing vulnerabilities.
 - After establishing a process to fix vulnerabilities early in development.
- **How to Set Up:**
 - Add **snyk test** to your build pipeline.
 - Enable the fail functionality to block builds with new vulnerabilities.
- **Long-Term Goal:**
 - Continuously enforce high security standards and integrate Snyk as a critical step in the development lifecycle.

Stage 3: Continuous Monitoring

- **Objective:**
 - Continuously monitor your applications for newly discovered vulnerabilities even after a successful build.
- **Why Continuous Monitoring?**
 - **Detect New Vulnerabilities:** Vulnerabilities can emerge over time due to changes in dependencies or the discovery of new threats.
 - **Ongoing Protection:** Keep your application secure, even after initial vulnerability checks.
- **Key Benefits:**
 - **Automated Monitoring:** After a successful build, Snyk automatically monitors your projects for new vulnerabilities.
 - **Proactive Alerts:** Receive alerts for vulnerabilities detected post-deployment, allowing you to react quickly.

Stage 3: Continuous Monitoring

- **How to Set Up:**
 - **Combine Snyk Test and Snyk Monitor:**
 - Use snyk test to scan for vulnerabilities during the build process.
 - If the build passes (no vulnerabilities detected), use snyk monitor to take a snapshot of the build.
 - **Snapshot for Ongoing Monitoring:**
 - Snyk monitor tracks successful builds, ensuring that future issues are identified and communicated.
- **Monitoring Workflow:**
 - Run snyk test to scan for vulnerabilities during the pipeline.
 - If no vulnerabilities are found, trigger snyk monitor to continuously monitor successful builds.
- **Long-Term Goal:**
 - Maintain security hygiene by automatically tracking vulnerabilities and applying fixes over time, ensuring your application remains secure.

CICD Deployment Methods

- **Configuring snyk in your pipeline**
- Synk native plugin
- Deploy Snyk using npm
- Deploy Snyc CLI Binary Version
- Deploy a Snyk Container

CICD Deployment Methods

- **1. Snyk Native Plugins**
- **Overview:** Easiest method to set up Snyk with common CI/CD tools.
- **Advantages:**
 - Simple and quick setup.
 - UI-based configuration for common parameters.
 - Native integration with popular CI/CD platforms (Jenkins, GitHub Actions, etc.).

CICD Deployment Methods

- **2. Deploy Snyk CLI Using npm**
- **Overview:** Install and run Snyk CLI using the npm package manager.
- **Advantages:**
 - Aligns with the CLI experience for easy troubleshooting.
 - Useful for developers familiar with npm commands.
 - Complete control over CLI configurations in the pipeline script.
- **Requirement:** Ability to run npm commands in your pipeline.

CICD Deployment Methods

- **3. Deploy Snyk CLI Binary Version**
- **Overview:** Use Snyk CLI as a binary package for integration.
- **Advantages:**
 - No dependencies on the local environment (npm not required).
 - Ideal for environments where you can't run npm commands.
 - Available for Linux, Windows, and other systems.
- **Source:** CLI binaries can be downloaded from the Snyk CLI GitHub repository.

CICD Deployment Methods

- **Deploy a Snyk Container**
- **Overview:** Use a pre-built Snyk Docker container from DockerHub.
- **Advantages:**
 - Simplifies setup for containerized environments.
 - Self-contained with minimal local dependencies.
 - Perfect for teams already working with Docker.

CICD Setup

- Define the organization using **--org** option
- Snyk Authentication
 - `export SNYK_TOKEN=<SNYK_API_TOKEN>`
 - `Snyk auth <SNYK_API_TOKEN>`
- Setup snyk using plugin, CLI and Snyk API
- The Snyk CLI can be configured to:
 - Return non-zero error codes only when certain criteria are met, for example, exit with an error code only if vulnerabilities of high severity are present.
 - Output all of its data into JSON for more flexibility.
- Setup automatic Monitoring

Exit Codes in Snyk Commands

1. Snyk Test Command (Synchronous)

- **Purpose:** Used to detect vulnerabilities and can end with an exit code.
- **Exit Codes:**
 - Determines if the build should pass or fail.
 - Example: An exit code of 0 indicates no vulnerabilities, while non-zero exit codes indicate issues.
- **Integration:** Build systems rely on these exit codes to decide whether to proceed or halt the build.

Exit Codes in Snyk Commands

2. Snyk Monitor Command (Asynchronous)

Purpose: Posts a snapshot of your project's dependency tree to your Snyk account for continuous monitoring.

Exit Codes:

Does not signal vulnerability status.

Indicates whether the snapshot was successfully created or not.

Silencing Exit Codes and Failing Builds

Scenario: To avoid failing the build when vulnerabilities are found during snyk test.

Solution:

Use || true at the end of the command: snyk test || true.

Impact: Forces the exit code to 0, allowing the build to continue even if vulnerabilities are detected.

Configuring options for failing builds

You can add options to the snyk test command to refine parameters that can result in a failed build:

--severity-threshold=high: Fail the build only for high-severity issues

--fail-on-upgradeable: fail the build only for issues that are upgradeable (can be fixed with Snyk fix advice)

Jenkins plugin integration with Snyk

Follow the steps in each section of this document to use the Snyk Jenkins plugin:

- 1) Install the Snyk Security Jenkins Plugin.
- 2) Configure a Snyk installation.
- 3) Configure a Snyk API token credential.
- 4) Add Snyk Security to your project.
- 5) Run a build and view your Snyk report.

Manage Risk

Manage Risk

Snyk has several features to help you manage Application Security (AppSec) risk.

- Snyk AppRisk
- Enterprise Analytics
- Reports
- Risk Score

Manage Risk: Snyk AppRisk

Snyk AppRisk enables Application Security teams to run a full developer security program, covering use cases under Application Security Posture Management (ASPM).

Snyk AppRisk provides:

- **Automated app asset discovery:** Continually discover application assets and classify them by business context, ensuring security is in sync with development.
- **Tailored security controls:** Define and manage appropriate security and compliance requirements, and verify the correct controls are in place.
- **Risk-based prioritization:** Assess risk for each app based on application context and best-in-class security analysis and fix guidance to focus developer remediation efforts on issues that matter most to the business.

Manage Risk: Enterprise Analytics

Enterprise Analytics shows the performance of a company's Application Security program.

Metrics are reported for the pillars:

- Exposure
- Manage
- Prevention
- Coverage

The breakdown of open issues over time is shown in these categories: Baseline, Preventable, Non-preventable, and Other new.

Customers can find the most serious problems that they must resolve, address those, and look for opportunities to proactively improve in each pillar.

Manage Risk: Reports

Use Snyk Reporting to track issues and their resolution.

The following reports are available:

- Issues Detail
- Issues Summary
- Vulnerabilities Detail
- Cloud Compliance Issues
- Developer IDE and CLI usage

Manage Risk: Risk Score

Snyk Risk Score can help you prioritize your approach to security issues, by providing you with an accurate and holistic understanding of the risk posed by a given security issue.

Manage Risk: Prioritize issues for fixing

You can find prioritization within Snyk under several names and with different customizations depending on your Snyk plan. The following list presents an overview of all the places where you can find and use prioritization. The outputs can vary depending on the type of priority you choose to use.

- Prioritization at the Project level
- Prioritization within Reporting
- Prioritization based on risk
- Prioritization based on coverage and security program

Prioritize issues for fixing

Prioritization at the Project level

Prioritize the Projects from your repositories by the type of Projects you want to focus on, for example, prioritize your Projects depending on their severity.

With this type of prioritization, you can group, filter, and sort your Projects.

Prioritize Projects based on these filters:

- With issues: displays only Projects with issues.
- Without issues: displays only Projects without issues.
- Integrations: displays the integrated Git repositories imported to Snyk.

Prioritize issues for fixing

Prioritize the issues within a Project

Prioritize the issues within a Project by using any of the following custom filters:

- Issue type
- Severity
- Fixability
- Exploit Maturity
- Status

Prioritization within Reporting

Priority Score and Risk Score prioritization

You can use both **Priority** and **Risk Scores** as a prioritization tool when you are using the Reporting option from the Snyk Web UI and you select the desired Projects from the filters list.

The Priority Score focuses on urgency and ranks security issues based on various criteria, such as **CVSS score**, **trending vulnerabilities**, **reachability**, **availability** of exploits, and other factors.

The Risk Score assesses the **overall risk** associated with vulnerabilities by considering both their severity and the context of the application, such as the likelihood of exploitation and the potential impact on the system. While the Priority Score assists teams in addressing the most urgent threats, the Risk Score offers a comprehensive view of the security posture.

Prioritization within Reporting

Priority Score:

- Focuses on urgency.

Ranks security issues based on:

- CVSS score
- Trending vulnerabilities
- Reachability
- Exploit availability
- Other key factors.

Helps teams address the most urgent threats quickly.

Prioritization within Reporting

Risk Score:

Focuses on overall risk.

Considers both:

- Severity of vulnerabilities
- Context of the application.

Evaluates:

- Likelihood of exploitation
- Potential impact on the system.

Provides a holistic view of the security posture.

Prioritization based on Risk

Snyk AppRisk:

Snyk AppRisk leverages holistic application intelligence to identify and prioritize vulnerabilities.

Focuses on risks across:

- Containers
- Code
- Open Source dependencies

Prioritization is based on asset classification defined in Snyk AppRisk policies.

Prioritization based on Risk

Prioritization with Insights:

Enables you to prioritize issues based on the risk they pose to the application.

Applies to:

- Container issues
- Code issues

Open Source vulnerabilities

Prioritization based on Risk

Key Risk Factors:

Deployed: Whether the vulnerable component is actively deployed in the environment.

Loaded Package: If the vulnerable package is loaded at runtime.

OS Condition: The operating system state and its exposure to vulnerabilities.

Public Facing: Whether the application is exposed to the public, increasing the likelihood of exploitation.

Prioritization based on coverage and security program

Snyk Enterprise and Application Analytics

Enterprise Analytics:

- Provides a comprehensive overview of your application security program at the tenant level.

Offers insights into:

- Strengths
- Weaknesses
- Opportunities for improvement.

Helps you understand overall performance and areas that need investment in your security practices.

Prioritization based on coverage and security program

Application Analytics:

- Focuses on the security and performance of individual applications.
- Offers a narrower analysis at the Snyk Group or Organization level.
- Helps pinpoint specific app vulnerabilities and inefficiencies for targeted improvements.

Priority Score vs Risk Score

- The Snyk Risk score and Priority score are keys to security management. Both types of score help Organizations handle current threats and prepare for future vulnerabilities, leading to a more robust security framework.
- Both scores provide a number between 1 and 1000, where 1 means low severity and 1000 means high severity.
- By default, the Priority score is enabled for Snyk Container, Snyk Code, Snyk IaC, and Snyk Open Source issues. If the user enables the Risk score from Snyk Preview, then the scores are applied like this:
 - Risk score is applied to **Snyk Container** and **Snyk Open Source** issues.
 - Priority score is applied to **Snyk Code** and **Snyk IaC** issues.

Priority Score

- The Snyk Priority Score is a single value assigned to an issue, to help you quickly and easily decide which issues are most important to fix.
- Scores range from 0 to 1,000; the higher the score, the more important it is to fix that issue.
- The Snyk Priority Score is determined based on a number of industry-standard criteria, including CVSS score, trending vulnerabilities, reachability, availability of exploits, and other factors.

Calculation of Priority Score:

- Severity Levels
- Exploit maturity
- Reachability
- Fixability
- Time
- Social Trends
- Malicious Packages

When to use Priority Score

Use it to first address the critical vulnerabilities and if you want to prioritize the Snyk Code issues

- **Time-Sensitive Projects** - When you are working on Projects with tight deadlines, it is important to address security concerns right away.
- **Initial Triage** - Priority score quickly identifies and mitigates immediate threats from multiple vulnerabilities.
- **Resource Allocation** - Teams can allocate security resources to promptly address urgent risks.

Priority Score Use Case

Scenario: Prioritizing Vulnerabilities in a Popular Web Application

Context:

- Your organization uses a widely popular web application platform.
- Several security vulnerabilities have been identified.
- Due to limited resources, it's not feasible to address all issues at once.

Prioritization Strategy:

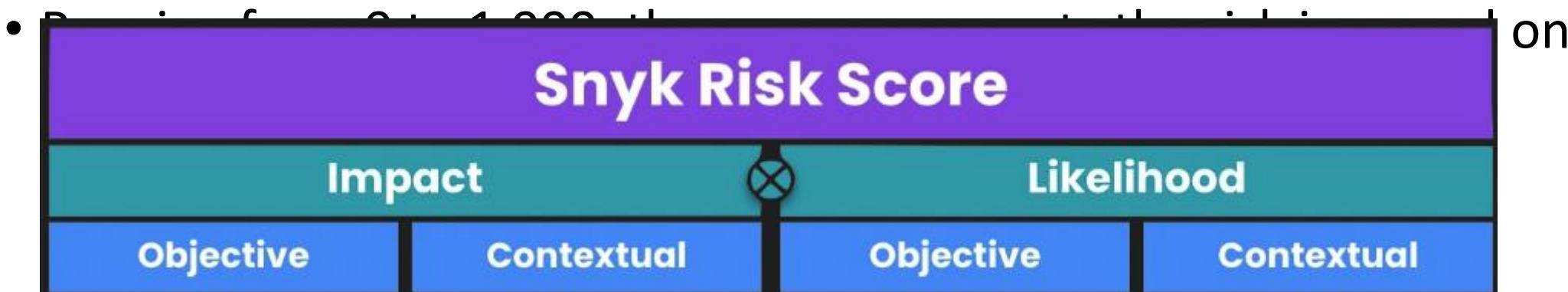
- Focus on patching critical vulnerabilities to mitigate the most significant threats to your organization's security.
- Utilize the Priority Score to strategically prioritize vulnerabilities based on their impact and ease of remediation.

Snyk Tools for Prioritization:

- Snyk Code: Scan the source code to identify and prioritize vulnerabilities at the code level.
- Snyk Open Source: Detect vulnerabilities in open-source libraries.
- Snyk Container: Identify vulnerabilities within containers.

Risk Score

- The Snyk Risk Score is a single value assigned to an issue, applied by automatic risk analysis for all vulnerability-type issues.
- License issues are not currently supported.
- Risk Score is based on the potential impact and likelihood of exploitability.



Risk Score

225 Risk Score More info

6 Impact

Objective	
Confidentiality	NONE ▼
Integrity	HIGH ▲
Availability	NONE ▼
Scope	UNCHANGED -
Contextual	
Business Criticality	HIGH -

3.7 Likelihood

Objective	
Exploit Maturity	PROOF OF CONCEPT -
User Interaction (UI)	NONE ▲
Privileges Required (PR)	NONE ▲
Attack Complexity	LOW ▲
Attack Vector	NETWORK ▲
EPSS	0.00043 ▼
Social Trends	NOT TRENDING -
Days since published	169 ▲
Malicious Code	FALSE -
Package Popularity	HIGH -

Contextual

Reachability	NO PATH FOUND -
Transitive dependency	FALSE -

Risk Score: Use Case

Scenario: Integrating a Vulnerable Third-Party Library

Context:

- You are integrating a new third-party library into an existing application.
- A scan reveals several vulnerabilities in the library.

Risk Score Prioritization:

- Use the Risk Score to filter and prioritize vulnerabilities that pose the greatest threat.
- Risk Score is available for Snyk Open Source and Snyk Container (must be enabled from the Snyk Preview screen).

Result:

- You will have a refined list of critical vulnerabilities that require immediate remediation before integrating the library.
- Fixing these issues will:
- Prevent a potential security breach.
- Safeguard the integrity of your application.

False Positives/Negatives in Vulnerability Scanning

False Positives:

A false positive occurs when a vulnerability scan identifies a security issue that is not actually present or exploitable.

Impact:

- Wastes time and resources.
- Diverts attention from real vulnerabilities.

Addressing False Positives:

- Use manual review to verify flagged vulnerabilities.
- Leverage advanced tools with better context-aware analysis (like Snyk's Risk Score).
- Regularly tune your vulnerability scanners to reduce noise.

Handling False Positive

1. Mark Issues as False Positives:

- If Snyk identifies a vulnerability that is a false positive, you can mark it to exclude it from future reports.
- Navigate to the specific issue in the Snyk Web UI and choose the option to ignore or mark it as a false positive.

2. Ignore Rules for False Positives:

- You can create ignore rules to silence specific vulnerabilities across projects or for a specified time period.
- Go to the vulnerability details and set an ignore rule with custom parameters:
 - **Duration:** Choose whether to ignore the issue permanently or temporarily.
 - **Justification:** Add a comment explaining why this issue is marked as a false positive.

Handling False Positive

3. Exclude Vulnerabilities via Policy:

You can implement Snyk policies to automatically exclude known false positives based on specific criteria such as:

- CVSS score
- Vulnerability type
- Version of the library.

4. Use Risk Score for Prioritization:

- Use Snyk's Risk Score to filter out vulnerabilities that have minimal impact or are not exploitable, reducing the chance of focusing on false positives.
- This helps your team focus on high-risk vulnerabilities and avoid wasting time on irrelevant issues.

5. Manual Review and Remediation:

- For complex projects, perform a manual review of flagged vulnerabilities to validate if the issue is a false positive.
- Once confirmed, document the reasons and update the project in Snyk accordingly.

False Positives/Negatives in Vulnerability Scanning

False Negatives:

A false negative happens when a vulnerability scan fails to identify a genuine security risk.

Impact:

- Undetected vulnerabilities pose a serious risk to application security.
- May lead to a security breach if left unresolved.

Addressing False Negatives:

- Use multiple scanning tools for cross-verification.
- Regularly update scanners to ensure they recognize new threats.
- Perform manual penetration testing to catch missed vulnerabilities.

Handling False Negatives

1. Use Multiple Scanning Tools:

- False negatives occur when vulnerabilities are missed. To minimize them, it's a good practice to use multiple security tools alongside Snyk (such as traditional static code analyzers, DAST tools, or manual penetration testing).
- This provides comprehensive coverage, catching vulnerabilities that may not be detected by a single tool.

2. Keep Dependencies Updated:

- Ensure that your third-party libraries and dependencies are up-to-date. Vulnerability databases used by Snyk are continuously updated, but old or unmaintained libraries may lack proper vulnerability tracking.
- Regular updates improve the chances of discovering newly found vulnerabilities.

3. Review Vulnerability Databases:

- Snyk relies on vulnerability databases for identifying security risks. Make sure that the vulnerability database Snyk uses is continuously updated by verifying if new issues are being added.
- Regularly check the databases for vulnerabilities that may have been missed in previous scans.

Handling False Negatives

4. Manual Code Audits:

- Conduct manual code reviews and audits, especially on critical code paths. This helps to identify vulnerabilities that automated tools may miss due to complexity or unique business logic.

5. Use Snyk's Advanced Features:

- Leverage Snyk Code for deep code analysis, particularly for proprietary or custom-built codebases. It can help identify hard-to-detect vulnerabilities that may result in false negatives.
- Use Snyk Container to catch vulnerabilities in Docker containers and their underlying infrastructure, which might not be caught by traditional scans.

6. Enable Continuous Monitoring:

- Turn on continuous monitoring in Snyk to receive notifications as new vulnerabilities are discovered. This can help catch vulnerabilities that might have been missed in earlier scans due to incomplete vulnerability data.

Handling False Negatives

7. Regularly Rescan Projects:

- Periodically rescan your projects in Snyk. As new vulnerabilities are discovered and added to the database, issues that were initially missed may now be detected.
- Set up automatic rescans or schedule them at regular intervals for key projects.

8. Perform Penetration Testing:

- Complement your Snyk scans with manual penetration testing to uncover vulnerabilities that might have gone undetected by automated tools.
- Penetration tests are particularly useful for identifying complex, contextual, or business logic vulnerabilities.

Security Policy Management

Security Policy Management

- Security policies provide an automated way to identify certain issues or types of issues and apply actions like changing the severity or ignoring the issue based on your conditions.
- A Group Administrator can change the default security policy or create new security policies for the entire group.

To open your group security policies, complete the following steps.

- Select the organization drop down.
- Select your Group Overview.
- Select the Policies tab.

Security Policy Condition

- A policy can be applied to an organization or a project
- When updating the default policy or creating a new policy, the IF statement section of the policy rule defines the conditions in which the policy applies to issues.
- The condition is based on below criteria:
 - Exploit Maturity
 - CWE
 - CVE
 - Synk ID

Security Policy Condition

Name
External Applications

Choose whether this policy applies to organizations or project attributes:
Organizations or Project attributes

Description
Exploitable and Injection Flaw Policy

In the case of a conflict between rules, the order the rules are in will determine precedence. Rules are applied top down.

Rule 1

If
Exploit Maturity ▾ Includes ▾
Mature +
Select... ▾

Rule 2

If
CWE ▾ Includes ▾
CWE-89 × CWE-77 × CWE-78 × CWE-88 × CWE-90 ×
e.g. 502 (or paste a comma separated string)

+

Security Policy Action

- When updating the default policy or creating a new policy, the THEN statement section of the policy rule defines the action that is performed on issues that meet the conditions of the rule.
- **Action:**
 - Change severity level to
 - Ignore current and future instance

Choose whether this policy applies to organizations or project attributes:

Organizations or Project attributes

Description
Exploitable and Injection Flaw Policy

In the case of a conflict between rules, the order the rules are in will determine precedence. Rules are applied top down.

Rule 1

If
Exploit Maturity ▾ Includes
Mature +

Then ▾
Change severity to ▾ Critical ▾

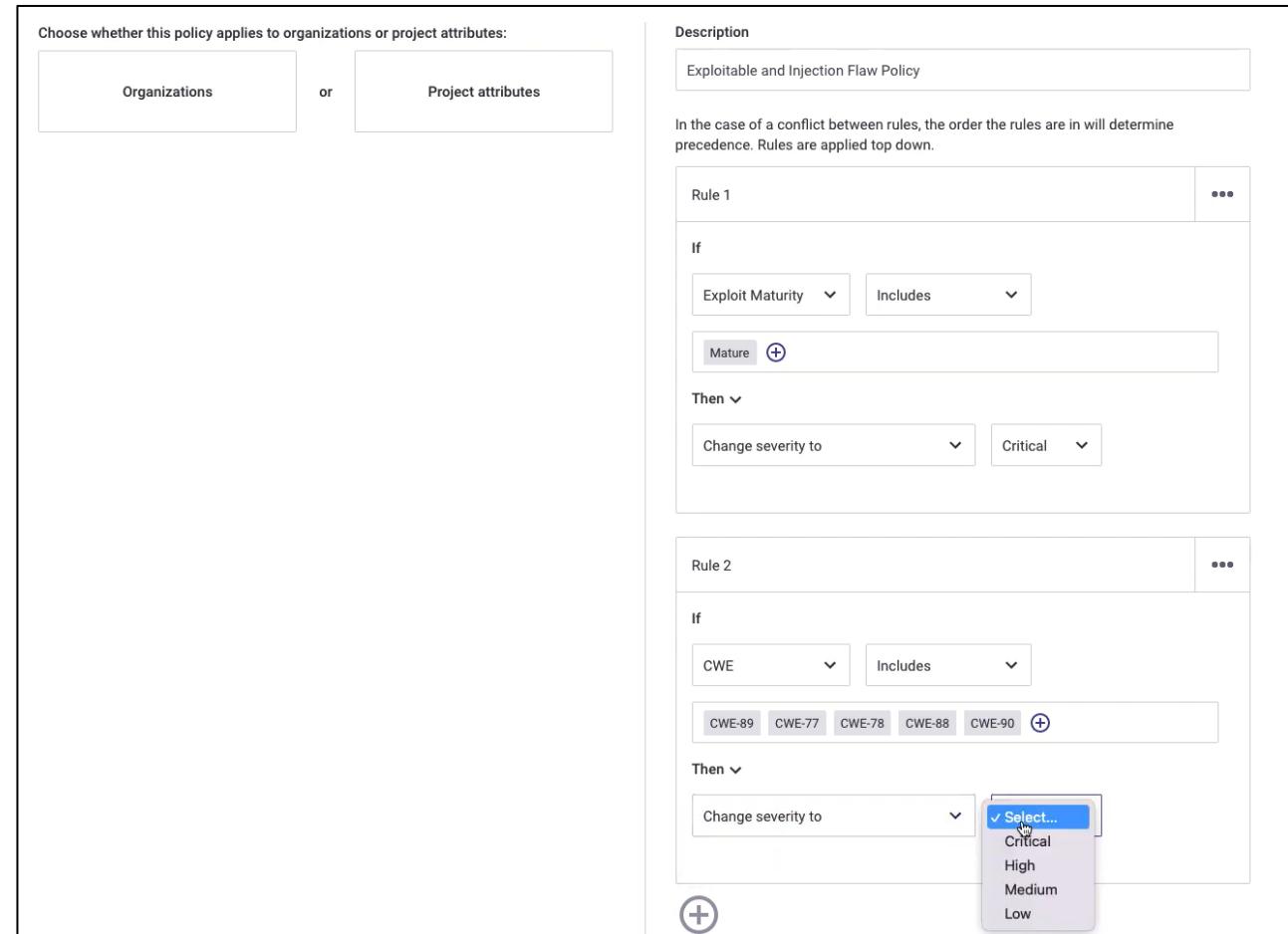
Rule 2

If
CWE ▾ Includes ▾

CWE-89 CWE-77 CWE-78 CWE-88 CWE-90 +

Then ▾
Change severity to ▾

+ Select... Critical High Medium Low



Security Policy Action

Name
Internal Low Criticality Projects (Attributes)

This policy will apply to the following project attributes
[Apply to organizations instead](#)

Business Criticality
 Critical
 High
 Medium
 Low

Environment
 Frontend
 Backend
 Internal
 External
 Mobile
 SaaS
 On-Prem
 Hosted
 Distributed

Lifecycle
 Production
 Development

Description
Internal Policy for low criticality applications

In the case of a conflict between rules, the order the rules are in will determine precedence. Rules are applied top down.

Rule 1
If
CWE ▾ Includes ▾
CWE-79 +

Then ▾
Ignore current and future instances ▾ Won't Fix ▾
Internal Projects policy

+

Security Policy Action

- We can check if the issue has been ignored manually or by policy.

The screenshot shows the Snyk security dashboard with 78 issues identified. On the left, a sidebar provides filtering options for Issue Type, Severity, Priority Score, Fixability, Exploit Maturity, and Status. The 'Status' section shows counts for Open (77), Patched (0), and Ignored (1). A callout bubble with the text "Identifies ignored issues" points to the 'Ignored' row. The main list displays a specific vulnerability for 'org.hibernate:hibernate-validator' with a medium CVSS score of 6.5. This entry is marked as 'IGNORED'. Below the list, there's a summary of ignore settings: 'Ignored a day ago by Security Policy', 'Type Ignored permanently', 'Reason Internal Projects policy', 'Ignored path *', 'Expires Never', and buttons for 'Unignore', 'Edit Ignore', and 'Fix this vulnerability'.

Issues 78 Fixes Dependencies 54 Runtime

ISSUE TYPE

SEVERITY

PRIORITY SCORE

FIXABILITY

EXPLOIT MATURITY

STATUS

Open 77

Patched 0

Ignored 1

Did you know... You can setup automatic assignment of Snyk pull requests to maximize the likelihood of vulnerability fixes being applied - enable for your GitHub integration.

M org.hibernate:hibernate-validator - Cross-site Scripting (XSS)

VULNERABILITY | CVSS 6.5 MEDIUM | SNYK-JAVA-ORGHIBERNATE-569100 | IGNORED

Introduced through com.github.derjust:spring-data-dynamodb@4.3.1 Exploit maturity NO KNOWN EXPLOIT

Fixed in org.hibernate:hibernate-validator@6.0.18.Final, @6.1.0.Final

Show more detail ▾

Learn about this type of vulnerability

Ignored a day ago by Security Policy

Type Ignored permanently

Reason Internal Projects policy

Ignored path *

Expires Never

Unignore Edit Ignore Fix this vulnerability

Identifies ignored issues

Security policy application and precedence

- Policies can be applied either to all the projects in one or more organizations, or according to attributes that you have applied to projects.
 - Environment
 - Business criticality
 - Lifecycle
 - Tags

The screenshot shows a web-based interface for creating a security policy. At the top, there's a 'Name' field containing 'External Applications'. Below it, a note says 'This policy will apply to the following project attributes' and 'Apply to organizations instead'. On the left, there are three sections: 'Business Criticality' (with options for Critical, High, Medium, and Low), 'Environment' (with options for Frontend, Backend, Internal, External, Mobile, SaaS, On-Prem, Hosted, and Distributed, where 'External' is checked), and 'Lifecycle' (with options for Production and Development). To the right, there's a 'Description' field with 'Exploitable and Injection Flaw Policy' and a note about rule precedence. The main area contains two rules. Rule 1 has an 'If' condition 'Exploit Maturity' set to 'Includes' and a value 'Mature'. The 'Then' part changes severity to 'Critical'. Rule 2 has an 'If' condition 'CWE' set to 'Includes' and values 'CWE-89', 'CWE-77', 'CWE-78', 'CWE-88', and 'CWE-90'.

Name
External Applications

This policy will apply to the following project attributes
Apply to organizations instead

Business Criticality

Critical
 High
 Medium
 Low

Environment

Frontend
 Backend
 Internal
 External
 Mobile
 SaaS
 On-Prem
 Hosted
 Distributed

Lifecycle

Production
 Development

Description
Exploitable and Injection Flaw Policy

In the case of a conflict between rules, the order the rules are in will determine precedence. Rules are applied top down.

Rule 1

If
Exploit Maturity

Then
Change severity to

Rule 2

If
CWE

Then

Security policy application and precedence

Policy manager

Search for policies...

> License policies (7) ⓘ Learn which policies take precedence + Add new policy

▼ Security policies (8) ⓘ Learn which policies take precedence + Add new policy

Project Attributes

External Application Policy (Attributes) Exploitable Flaw Policy	External	...
Internal Low Criticality Projects (Attributes) Internal Policy for low criticality applications	Low, Internal	...

Organization

Exploit Maturity Ignores Ignore Severity Issue	Applied to 0 orgs	...
JBK test ignore 1 issue	Applied to 1 org	...
CWE-94 critical CWE-94 critical	Applied to 2 orgs	...
Internal Application Policy XSS Rule	Applied to 1 org	...
External Applications Exploitable and Injection Flaw Policy	Applied to 1 org	...
Snyk Default Security Policy DEFAULT	Applied to 15 orgs	...

Snyk API

Snyk API

- The Snyk API allows Enterprise customers to integrate programmatically with Snyk.
- The Snyk API is available only for Enterprise plans.
- Snyk extensibility and the Snyk API enable developers to tune Snyk security automation to their specific workflows, ensuring consistency in both developer experience and platform governance.

Snyk REST API

- The Snyk REST API is designed according to OpenAPI and JSON standards, providing a structured and evolutionary approach to API development. Each endpoint within the API is versioned, allowing users to maintain compatibility and control over their integration.

When to Use the API:

The Snyk REST API is ideal for scenarios where you need to:

- Customize: Tailor Snyk's security features to fit your specific needs.
- Integrate: Embed Snyk's security capabilities into your applications and workflows.
- Automate: Streamline your security processes by automating tasks related to vulnerability management and remediation.

Differences Between API, CLI, and Integrations

API Limitations:

- The API may not always provide the same level of accuracy as the Snyk CLI, especially for certain package managers.
- The API infers a snapshot of your dependencies, which may lead to less accurate results compared to the actual deployed code tested by the CLI.

CLI Advantages:

- The Snyk CLI runs directly against your codebase, testing the actual deployed code and providing a more accurate snapshot of the dependency versions in use.
- It outputs results in a machine-readable JSON format using the command `snyk test --json`, which can be useful for automated processing and integration into CI/CD pipelines.

Integrations:

- Integrating Snyk with your source code management (SCM) tool allows for real-time monitoring of code changes and automatic suggestions for fixes.
- Using a Broker can enhance security and auditability, ensuring that Snyk's monitoring capabilities fit seamlessly into your development flow.
- Snyk can monitor new pull requests, facilitating immediate feedback on security vulnerabilities and enabling rapid remediation.

API Call

```
curl --request GET \
--url "https://api.snyk.io/rest/orgs/{orgId}/projects?version=2024-
06-10" \
--header "Content-Type: application/vnd.api+json" \
--header "Authorization: token API_TOKEN"
```

Note: Replace the orgID and API_TOKEN

Customizing Vulnerability Scanning with the Snyk API

The Snyk REST API allows developers and security teams to customize and automate vulnerability scanning processes according to their specific needs. This flexibility can enhance security practices and ensure that applications are monitored effectively for vulnerabilities.

- Endpoint Customization:
- Automating Scans:
- Custom Reports:
- Integration with Development Workflows:
- Webhook Notifications:

Customizing Vulnerability Scanning with the Snyk API

Example 1: Scan a Specific Project for Vulnerabilities

```
POST https://snyk.io/api/v1/test
Content-Type: application/json
Authorization: token YOUR_API_TOKEN
{
  "target": {
    "project_id": "your_project_id",
    "files": [
      {
        "path": "/path/to/your/project/package.json",
        "type": "npm"
      }
    ]
  }
}
```

Customizing Vulnerability Scanning with the Snyk API

Example 2: Filter Vulnerabilities by Severity

```
GET https://snyk.io/api/v1/project/your_project_id/vulnerabilities?severity=critical
```

```
Authorization: token YOUR_API_TOKEN
```

Customizing Vulnerability Scanning with the Snyk API

Example 3: Automate Scans in CI/CD Pipeline

```
#!/bin/bash

# Trigger a scan for the current project
curl -X POST https://snyk.io/api/v1/test \
-H "Authorization: token YOUR_API_TOKEN" \
-H "Content-Type: application/json" \
-d '{
  "target": {
    "project_id": "your_project_id",
    "files": [{"path": "./package.json", "type": "npm"}]
  }
}'
```

Customizing Vulnerability Scanning with the Snyk API

Example 4: Generate a Custom Report

```
GET https://snyk.io/api/v1/report/your_project_id
```

```
Authorization: token YOUR_API_TOKEN
```

Customizing Vulnerability Scanning with the Snyk API

Example 5: Set Up Webhook Notifications

```
{  
  "url": "https://yourapp.com/webhook",  
  "events": [  
    "vulnerability.detected"  
  ]  
}
```

Reporting

Reporting

Snyk Reports: Overview and Benefits

1. Visibility and Analytics

Data-Driven Conversations: Snyk Reports empower development and security teams to engage in informed discussions based on concrete data. By providing visibility into security issues, these reports foster collaboration and understanding between teams.

2. Access and Analysis

User-Friendly Interface: Security professionals can easily access and analyze Snyk security data through an intuitive interface. This accessibility enables teams to quickly identify vulnerabilities and risks within their applications.

3. Prioritization of Remediation

Identifying Risks: By leveraging analytics, Snyk Reports help organizations pinpoint where the greatest risks lie within their application portfolio. This insight allows teams to prioritize remediation efforts effectively, addressing the most critical vulnerabilities first.

4. Understanding Risk Posture

Comprehensive View: Snyk Reports provide a holistic understanding of the overall risk posture across the organization's application portfolio. This perspective is crucial for informed decision-making regarding security strategies and resource allocation.

Reporting

Features of Snyk Reports

1. Detailed and Aggregated Data

Project-Level Insights: Snyk Reports display both detailed and aggregated data about projects and security issues. This dual-level analysis allows organizations to drill down into specific projects while also understanding broader trends across their portfolio.

2. Scope and Customization

Scoped Data: Each report is scoped to the Snyk Organization or Group, ensuring that the data presented is relevant to the context in which users are working. This tailored approach enhances the relevance of the information provided.

3. Tracking Developer Adoption and Success

Monitoring Progress: Snyk Reports enable organizations to track developer adoption of security practices and the success of remediation efforts over time. This tracking is essential for fostering a culture of security within development teams.

Available Snyk Reports

The following reports are available:

- Issues Detail report
- Issues Summary report
- Vulnerabilities Detail report
- Featured Zero-Day report
- SLA Management report
- OWASP TOP 10 report
- CWE TOP 25 report
- CWE TOP 10 KEV report
- Cloud Compliance Issues report
- Developer IDE and CLI usage report

How Snyk can help with cloud compliance

Snyk IaC uses a unified policy as code engine to help teams develop, deploy, and operate safely in the cloud. Snyk IaC provides:

- Guardrails for security and compliance across major cloud providers, to help teams adhere to policy at all times.
- Continuous evaluation of your cloud environment, and infrastructure as code, for compliance with regulatory and internal security policies using real-time and historical reporting, packaged for security engineers and GRC teams.
- Comprehensive, best-in-class cloud compliance right out of the box.
- Coverage across major compliance standards: SOC 2, HIPAA, GDPR, PCI (or PCI DSS), NIST 800-53, ISO 27001, CIS Benchmarks (for AWS, Azure, Google, Kubernetes, and Docker), and CSA Cloud Controls Matrix.