DevSecOps

Introduction

Name

Total Experience

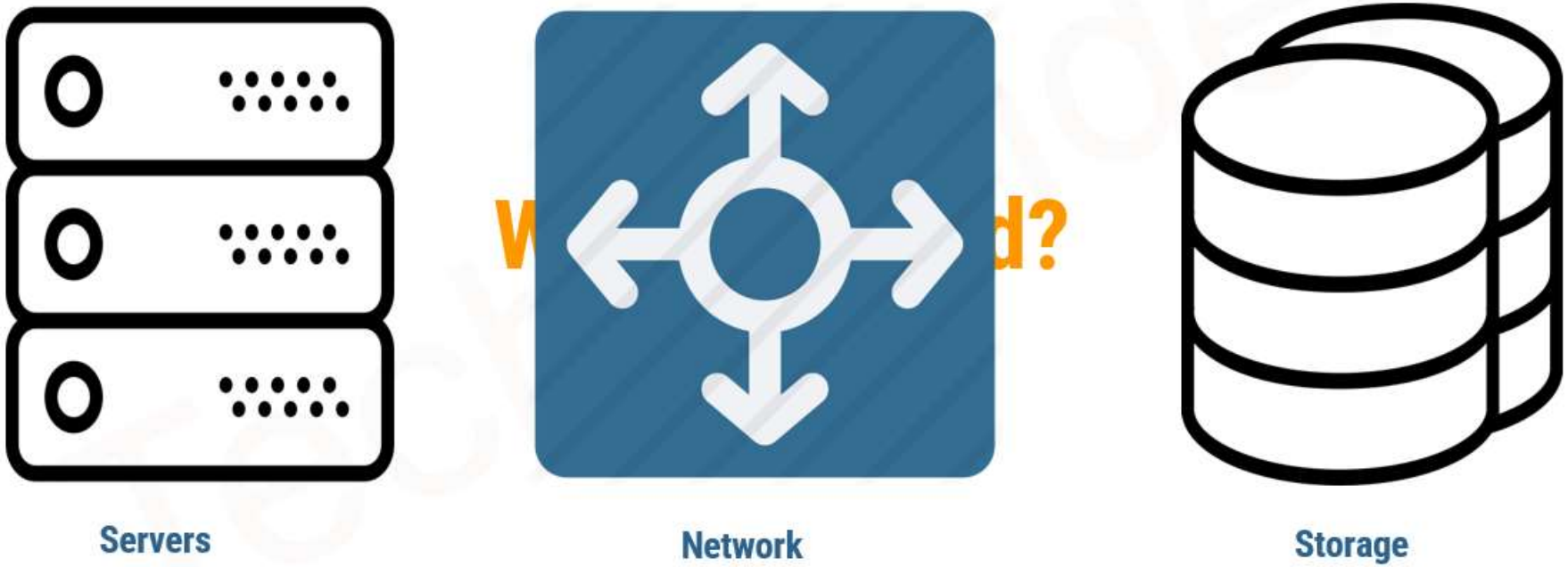
Background – Development / Infrastructure / Database / Network

Experience on monitoring tools

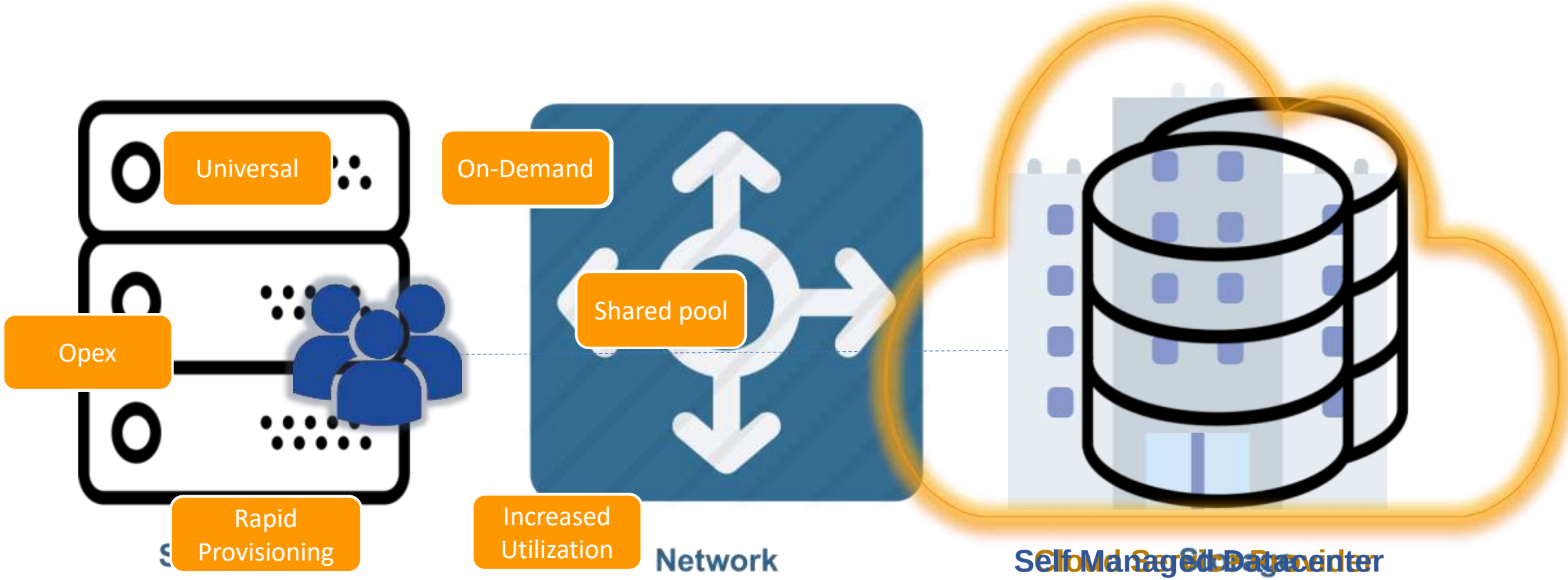
Your Expectations from this training

Module 1:
Cloud Fundamentals & DevSecOps

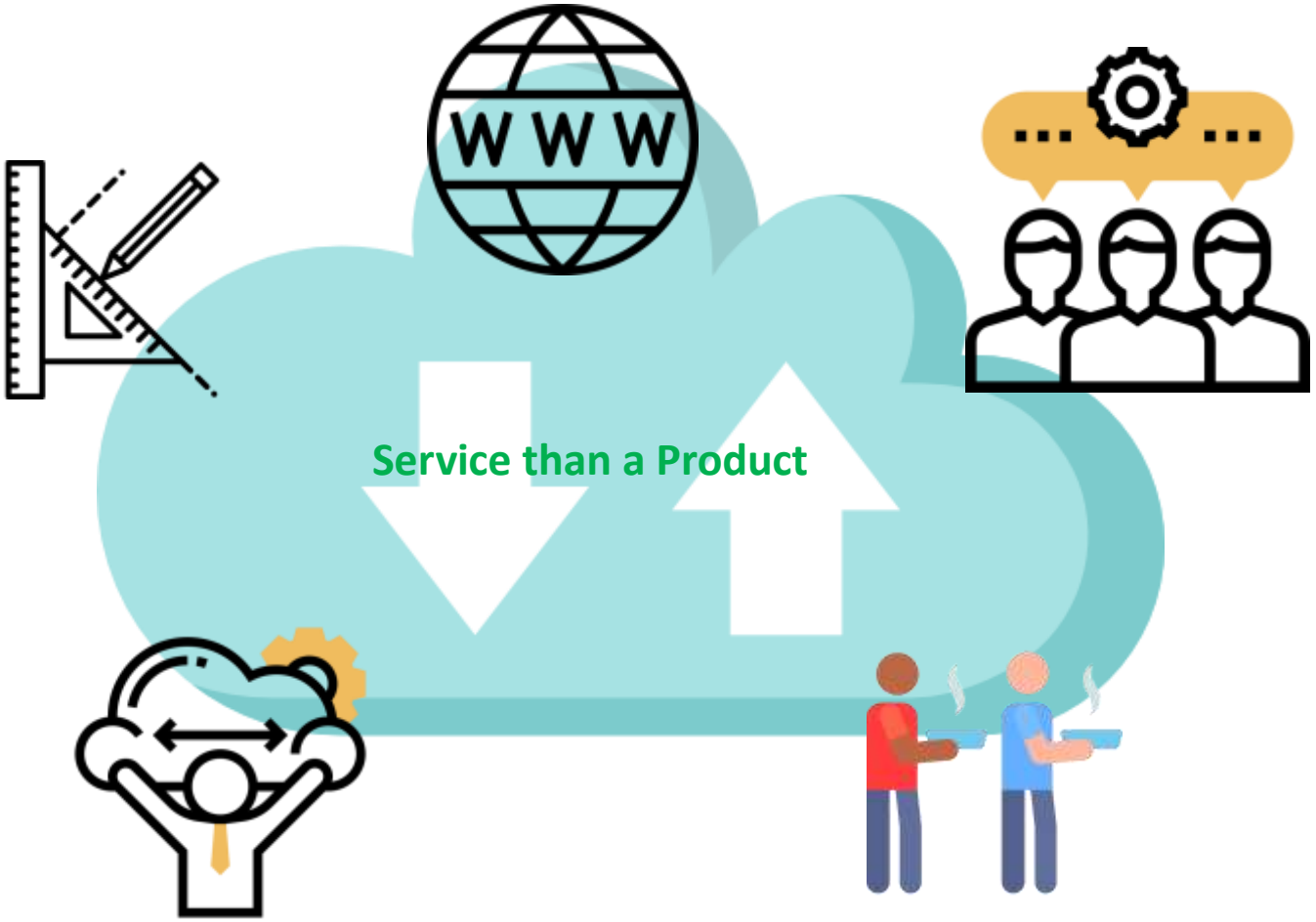
Traditional Datacenters



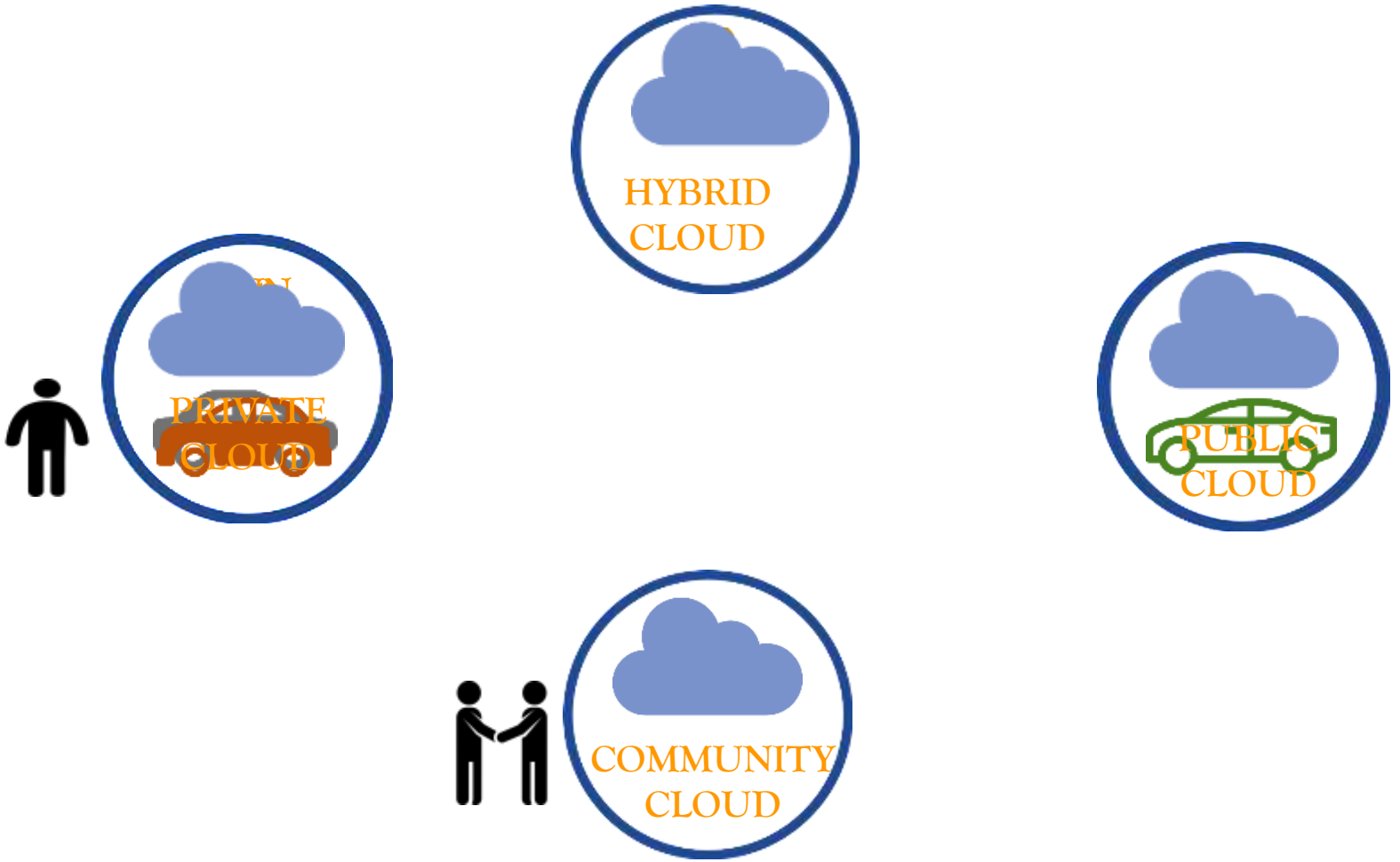
Traditional vs Cloud Datacenters



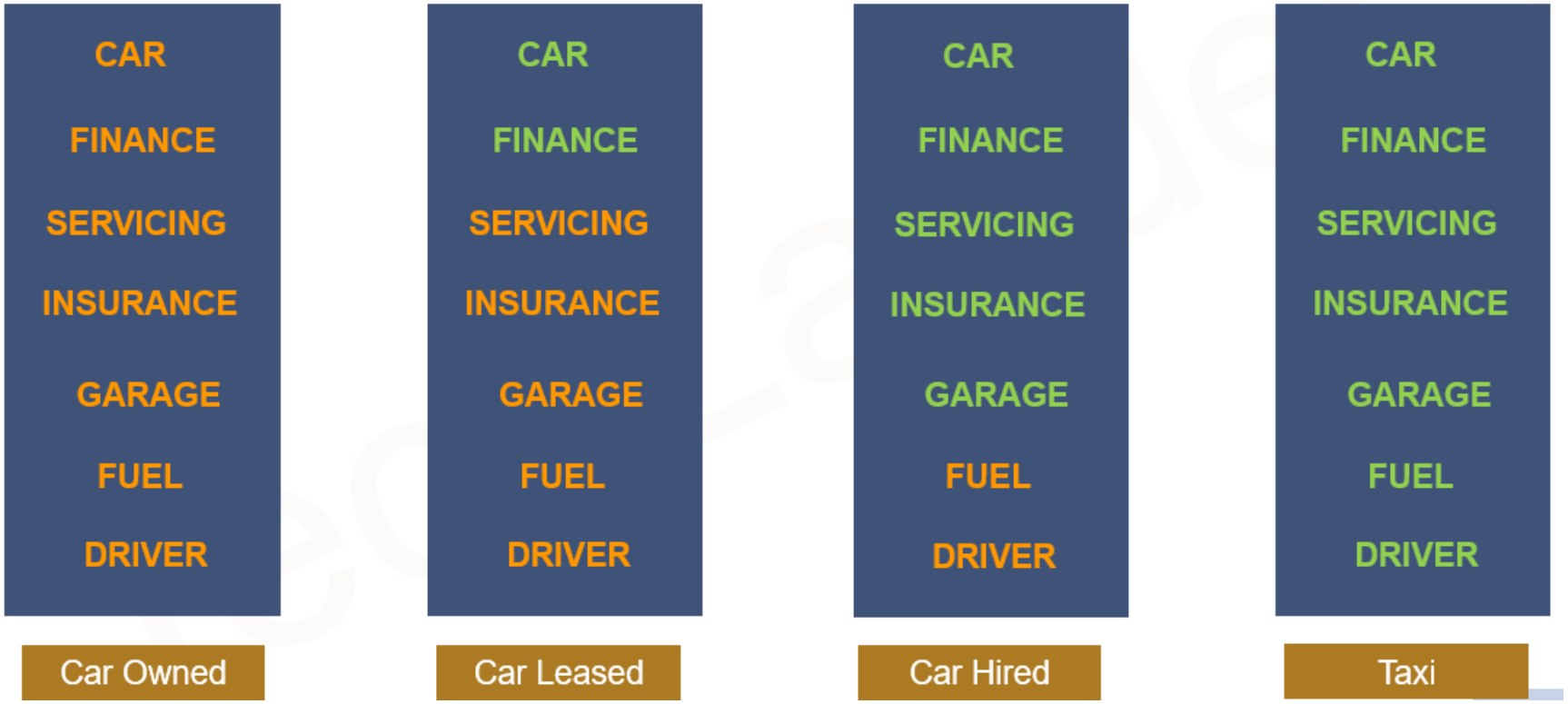
Characteristics of Cloud



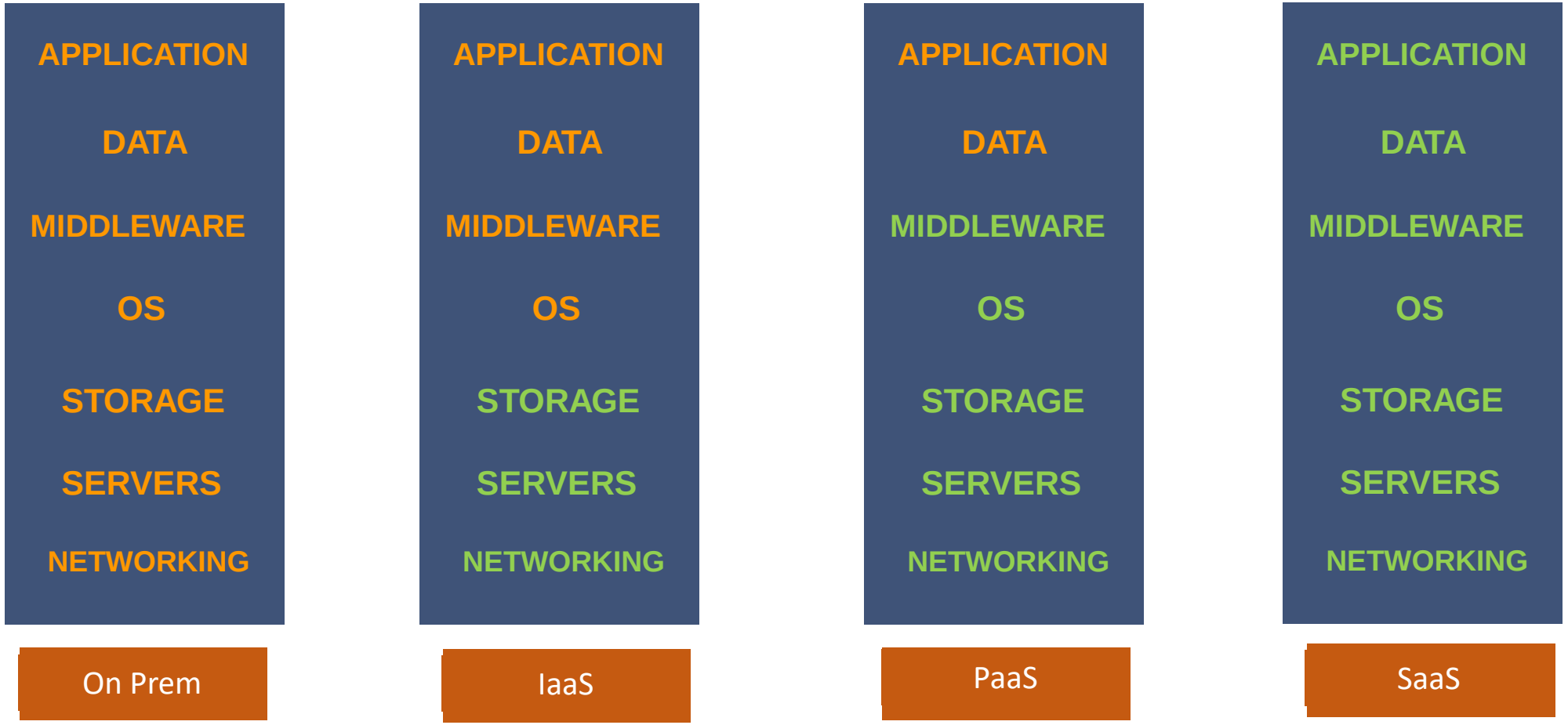
Cloud Deployment Types



Service Models

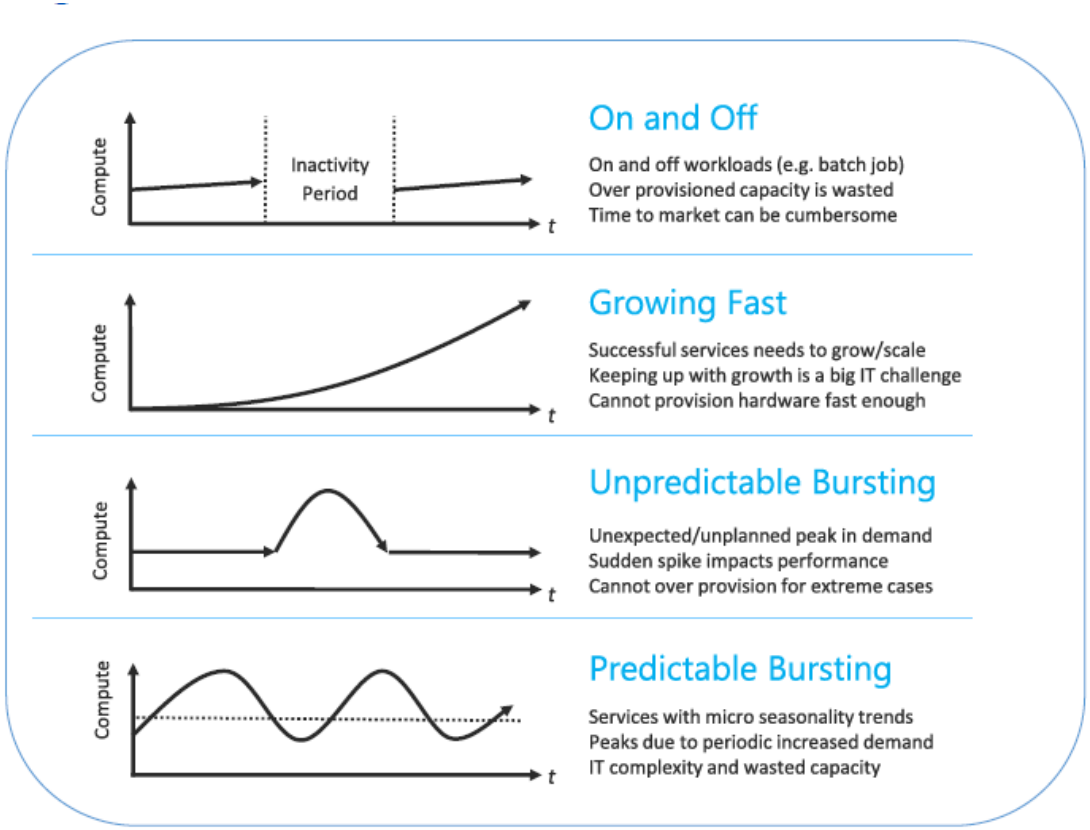


Service Models

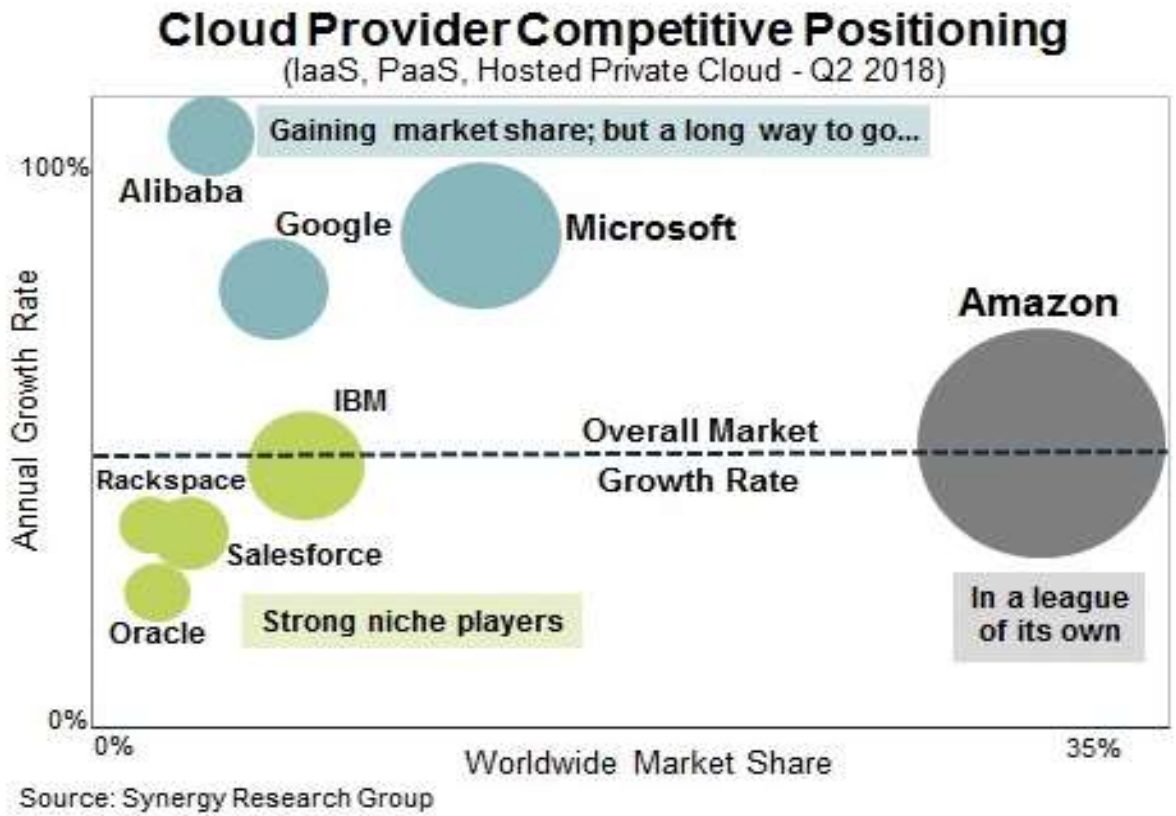


Cloud Benefits

Cloud Major Use Cases



Cloud Players



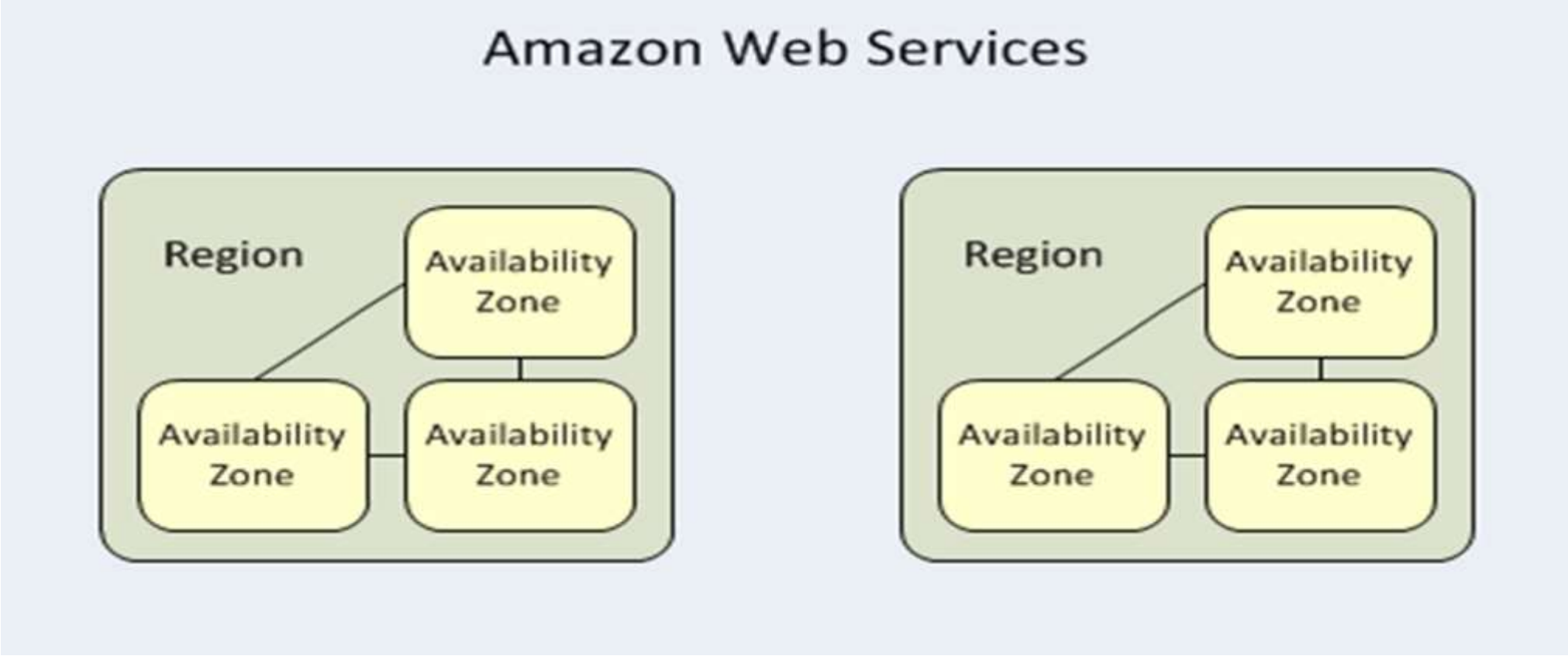
Amazon Web Services

- 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.

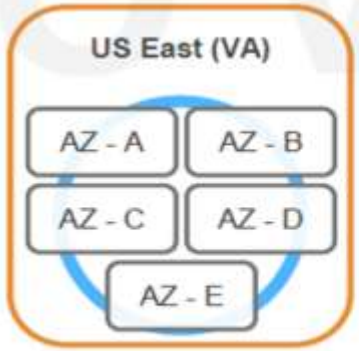
AWS Regions and Availability Zones

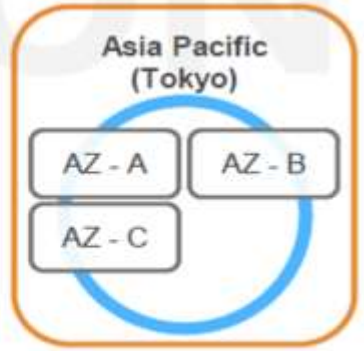


AWS Regions and Availability Zones

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

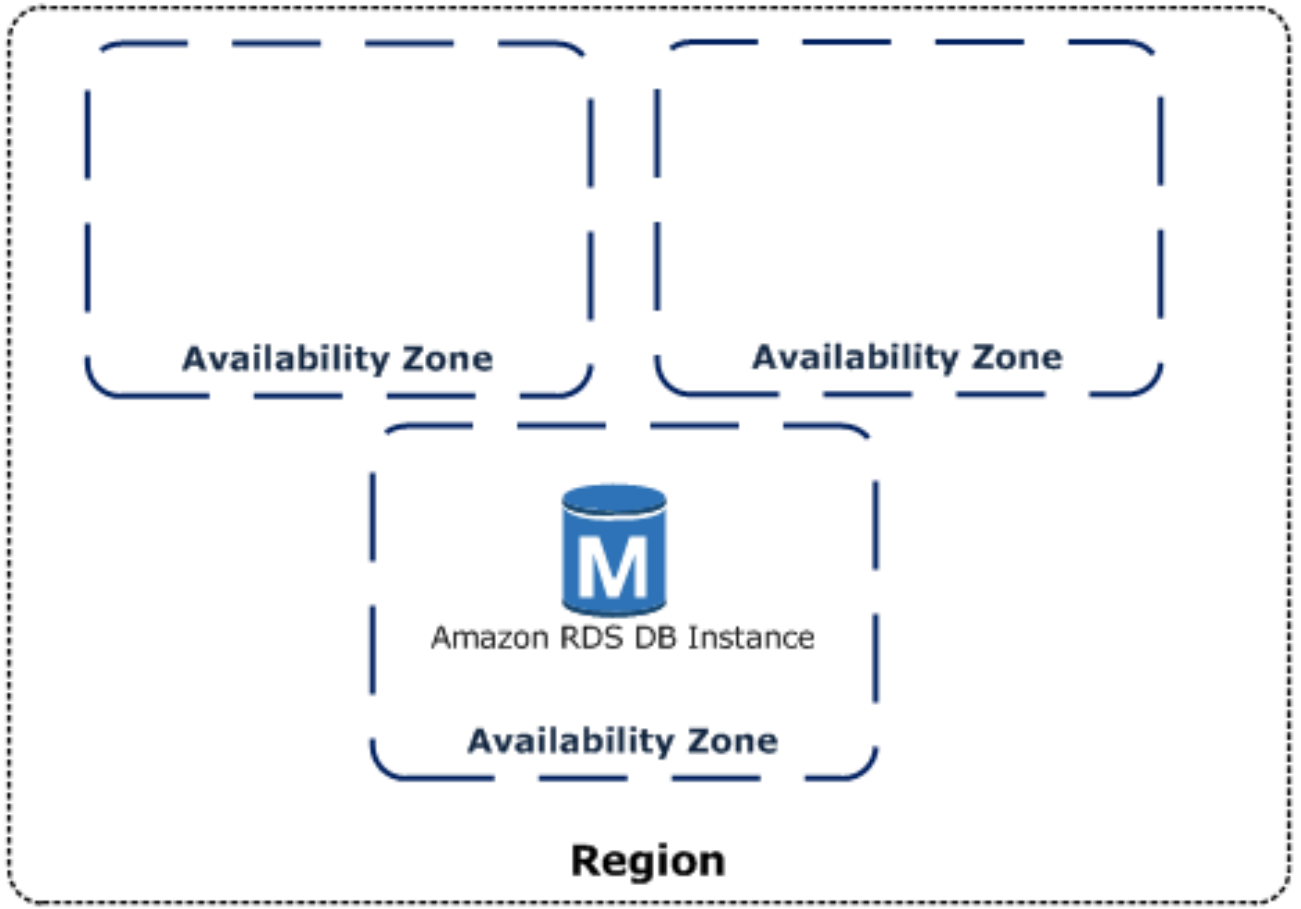
69 AZs

Upcoming:

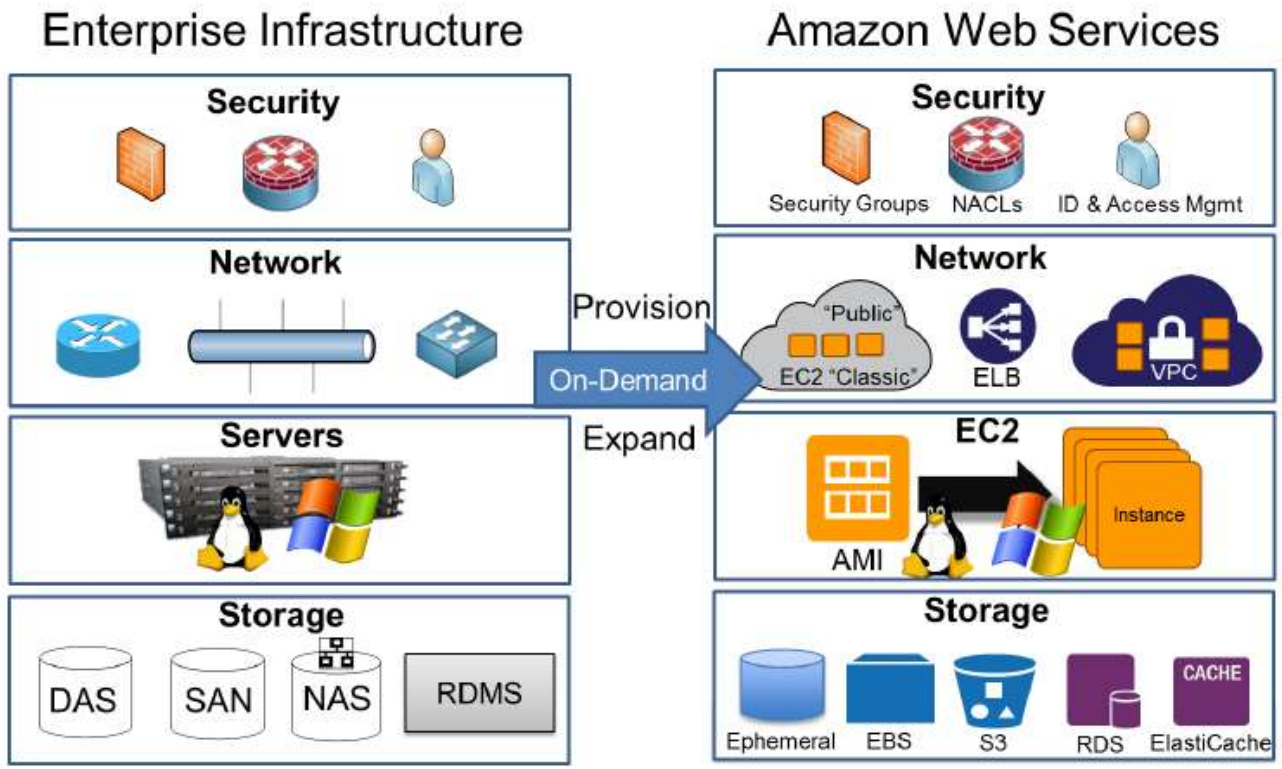
4 Regions

13 AZs

AWS Regions and Availability Zones



AWS

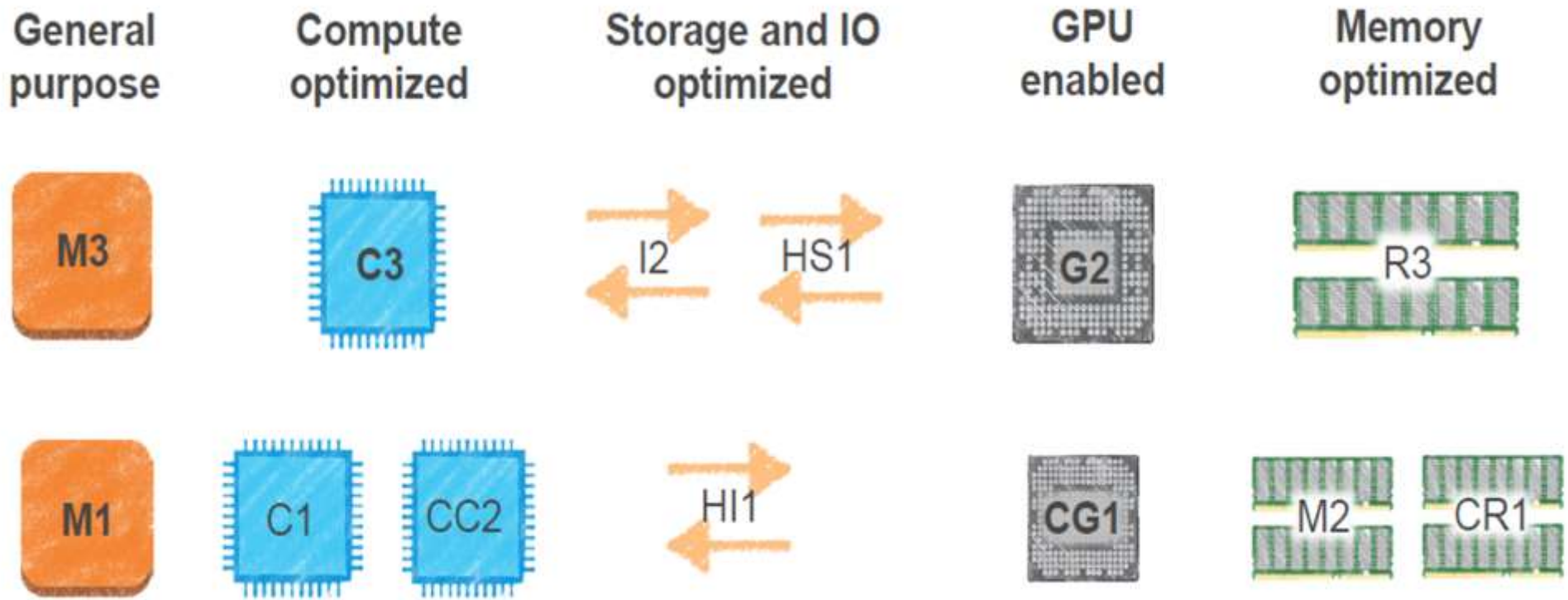

Enterprise Infrastructure
Security
Network
Servers
Storage
DAS
SAN
NAS
RDMS
Provision
On-Demand
Expand
Amazon Web Services
Security
Security Groups
NACLs
ID & Access Mgmt
Network
"Public"
EC2 "Classic"
ELB
VPC
EC2
AMI
Instance
Storage
Ephemeral
EBS
S3
RDS
CACHE
ElastiCache

AWS Compute Services

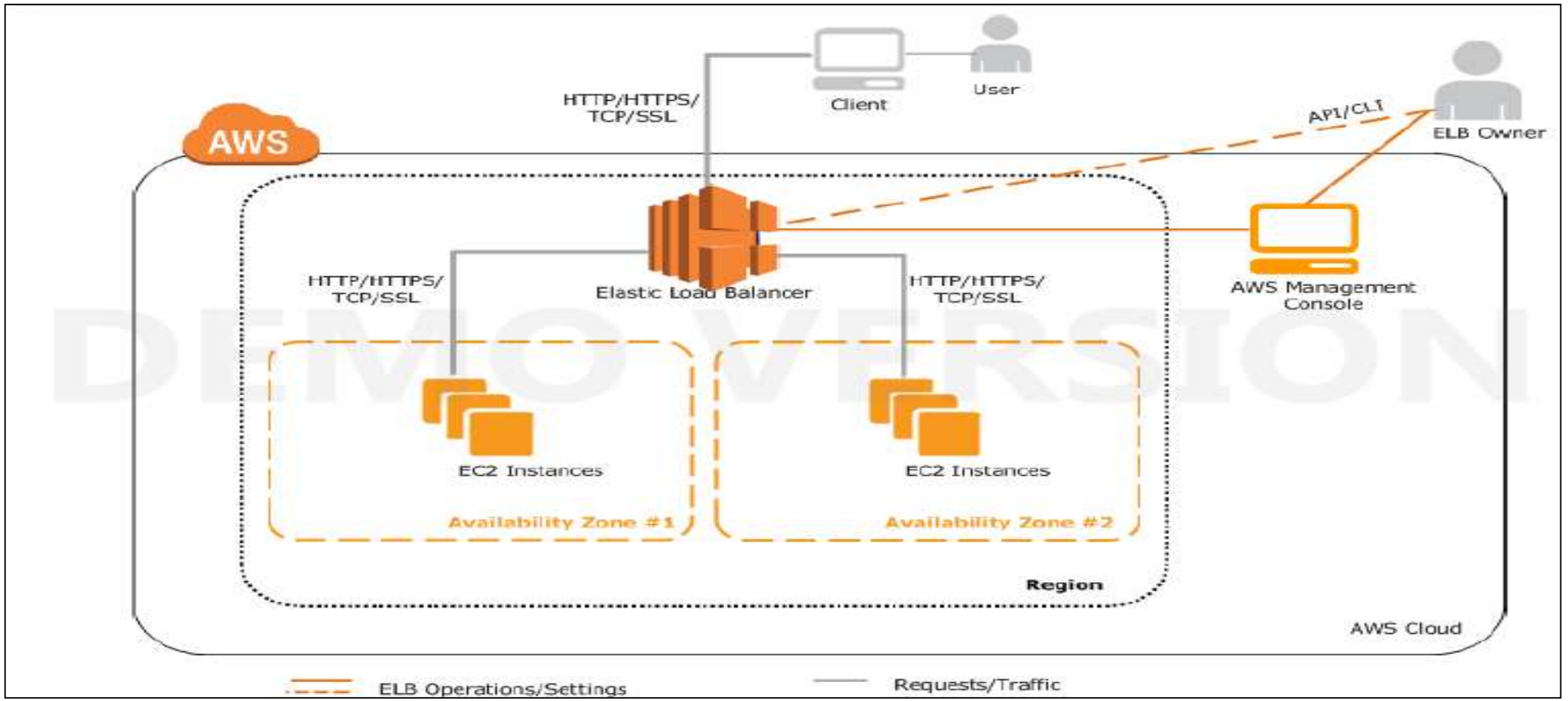
- Amazon EC2 stands for Elastic Compute Cloud, and is the Primary AWS web service.
- Provides Resizable compute capacity
- Reduces the time required to obtain and boot new server instances to minutes
- There are two key concepts to Launch instances in AWS:
  - Instance Type
  - AMI

- 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

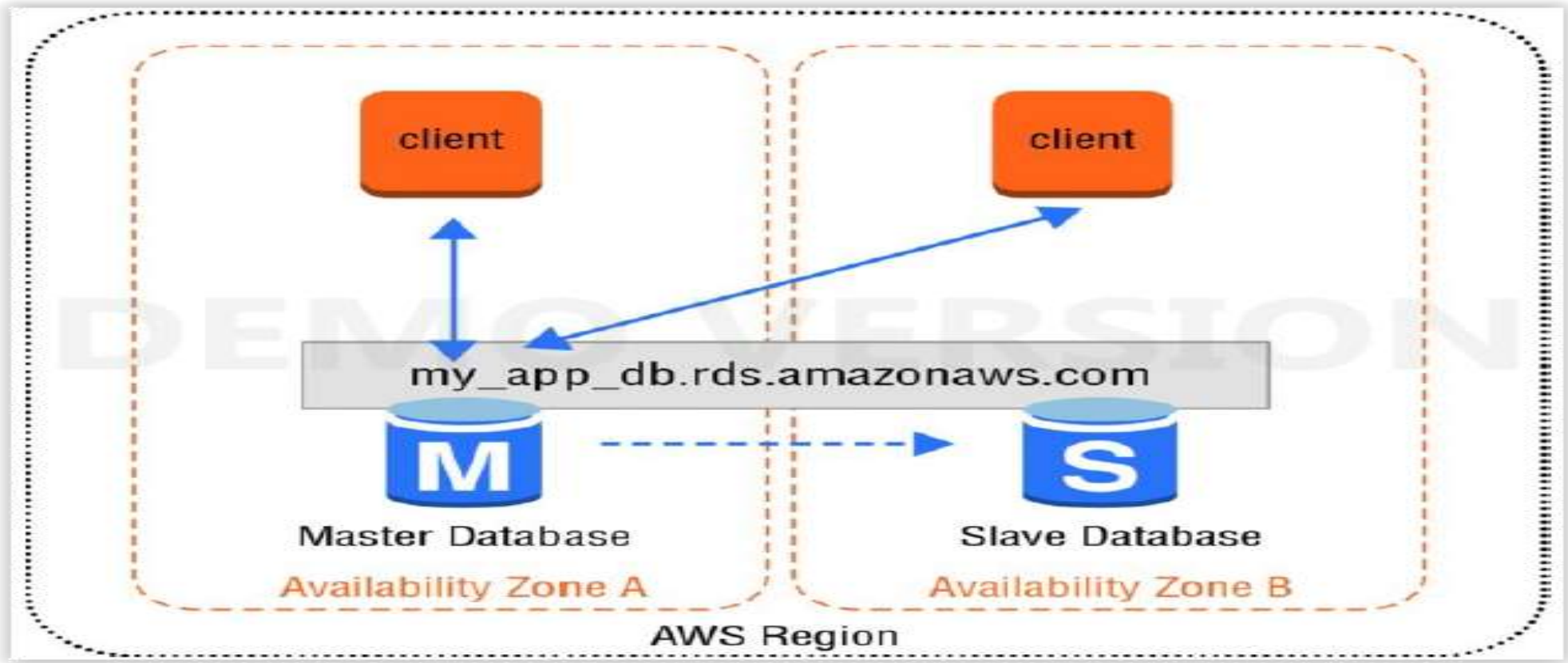
AWS EC2


General purpose
Compute optimized
Storage and IO optimized
GPU enabled
Memory optimized
M3
C3
I2
HS1
G2
R3
M1
C1
CC2
HI1
CG1
M2
CR1

ELB


AWS
HTTP/HTTPS/
TCP/SSL
Client
User
API/CLI
ELB Owner
Elastic Load Balancer
HTTP/HTTPS/
TCP/SSL
HTTP/HTTPS/
TCP/SSL
AWS Management
Console
EC2 Instances
EC2 Instances
Availability Zone #1
Availability Zone #2
Region
AWS Cloud
ELB Operations/Settings
Requests/Traffic

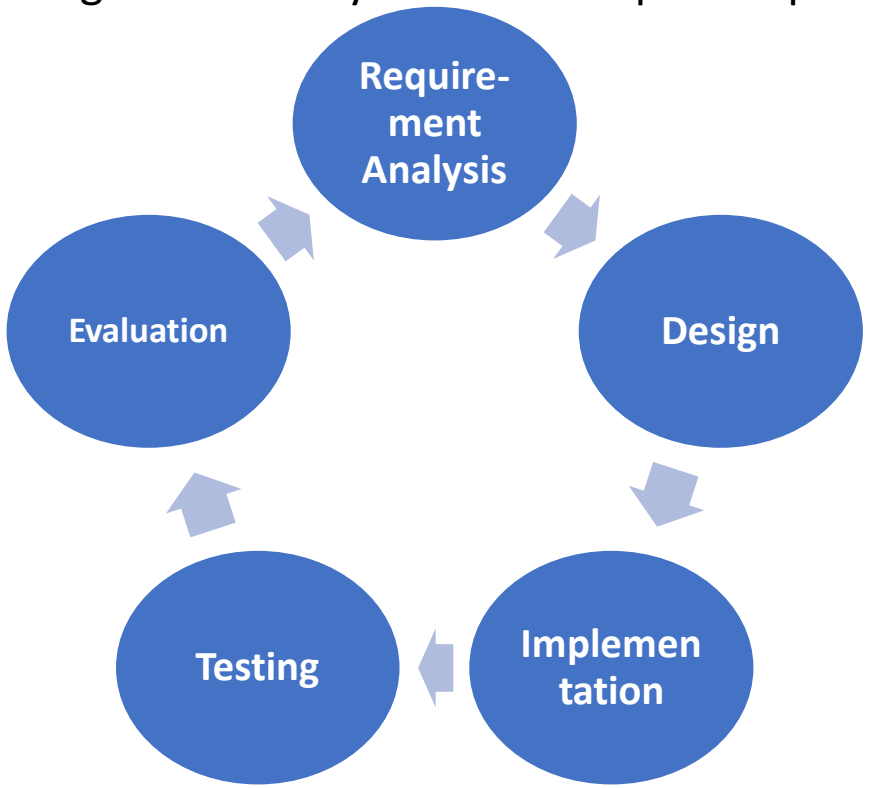
PAAS


client
client
my_app_db.rds.amazonaws.com
M
S
Master Database
Slave Database
Availability Zone A
Availability Zone B
AWS Region

What is DevOps

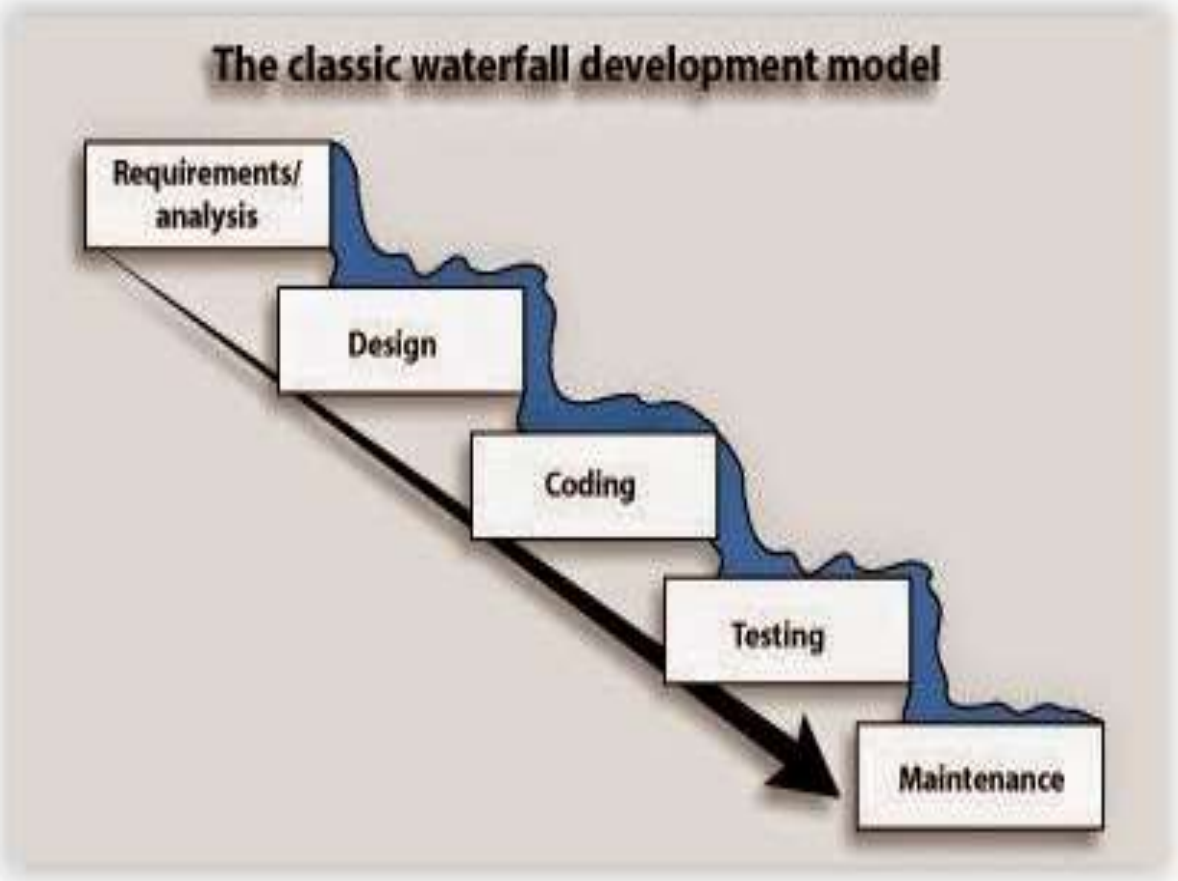
SDLC Model

- 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



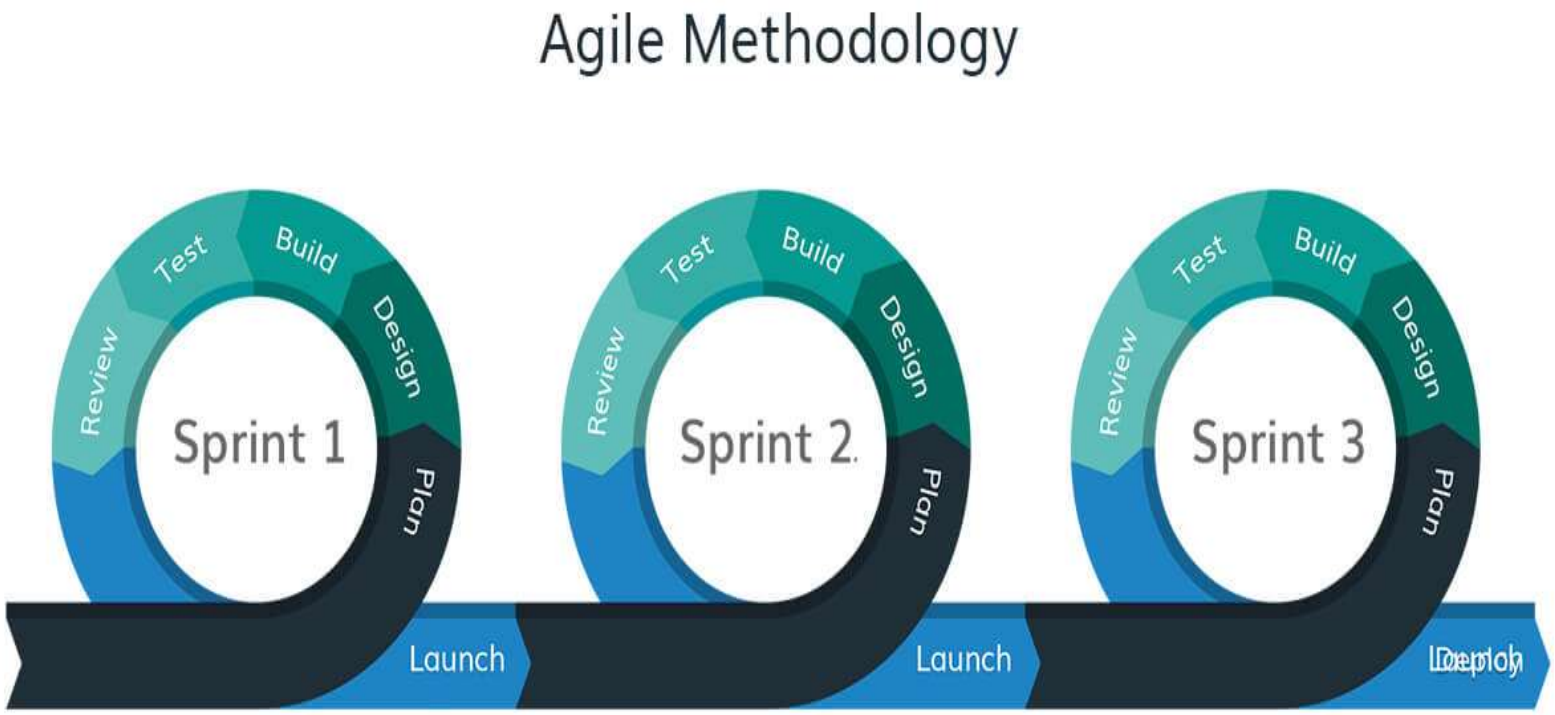
Waterfall Model

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

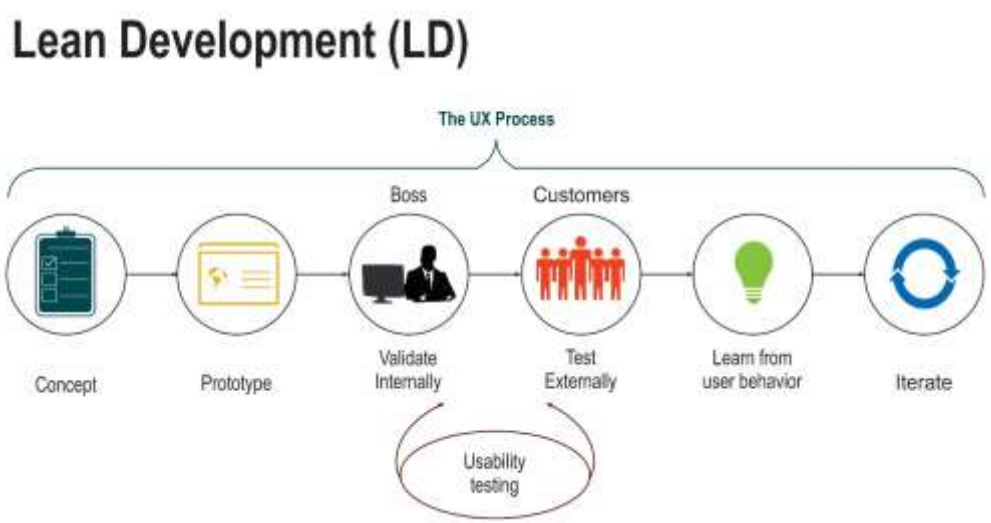


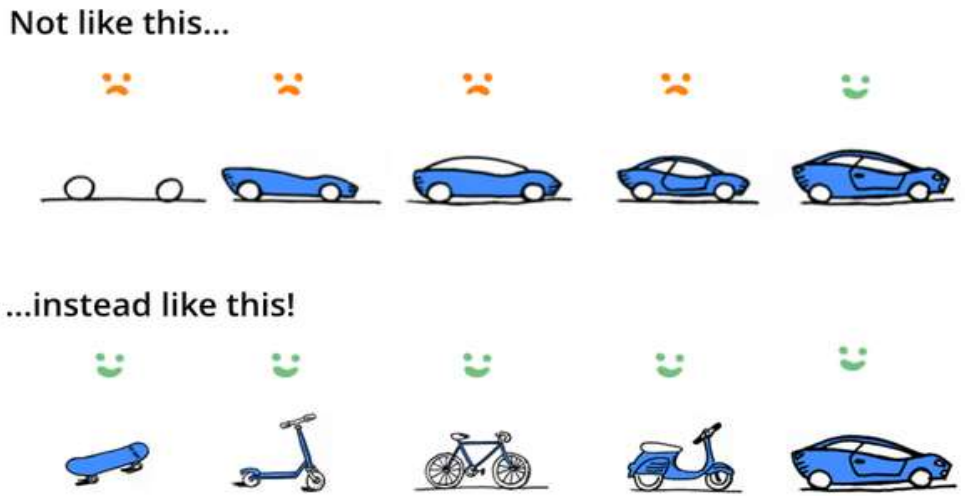
Agile

- - Shorter release cycle
- - Small batch sizes (MV
- - Cross-functional tear
- - Incredibly agile



Lean Development

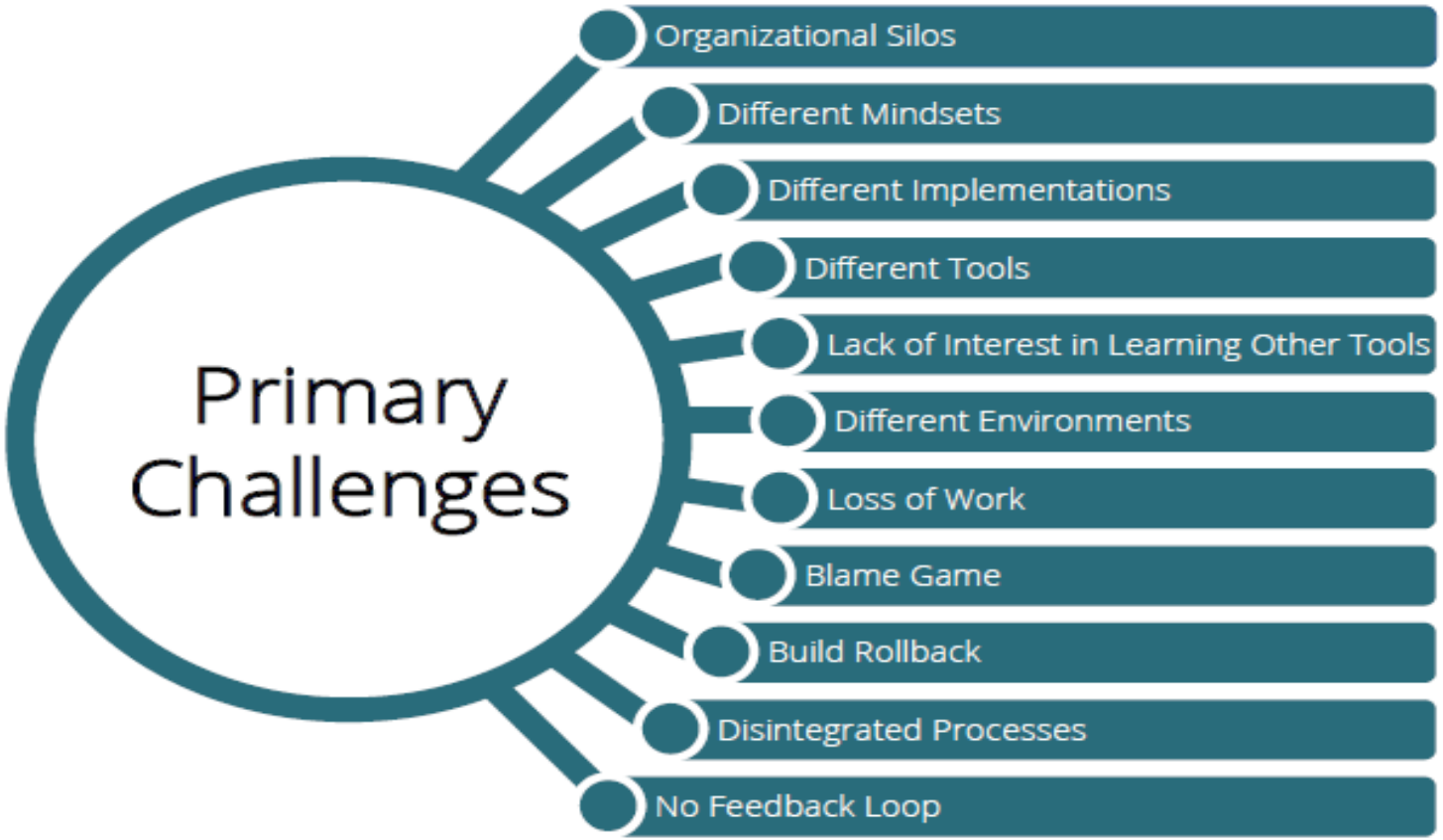




- Suddenly ops was the bottleneck (more release less people), again WIP is more!

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.

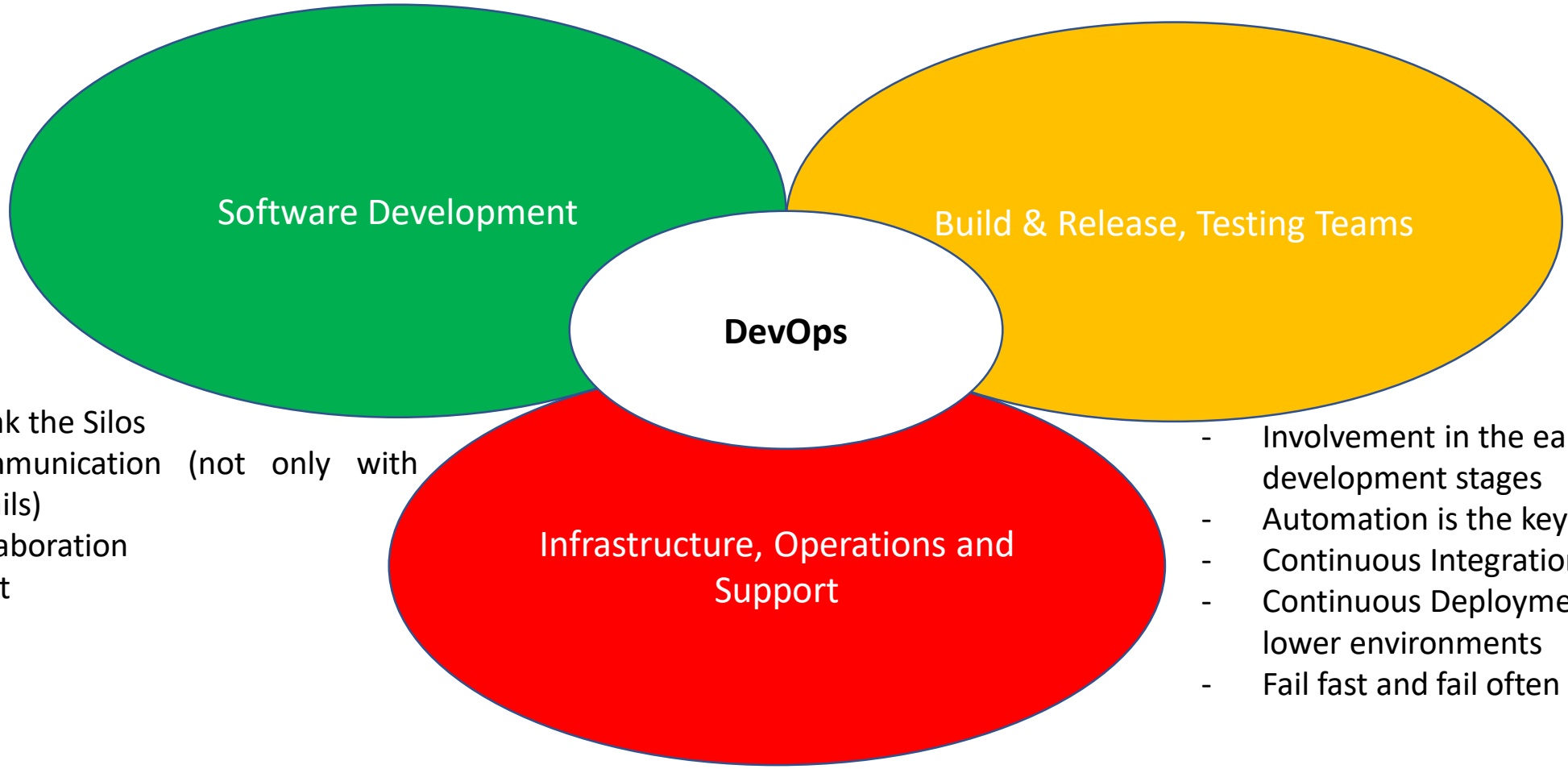
- **Ops Team:**

- 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.

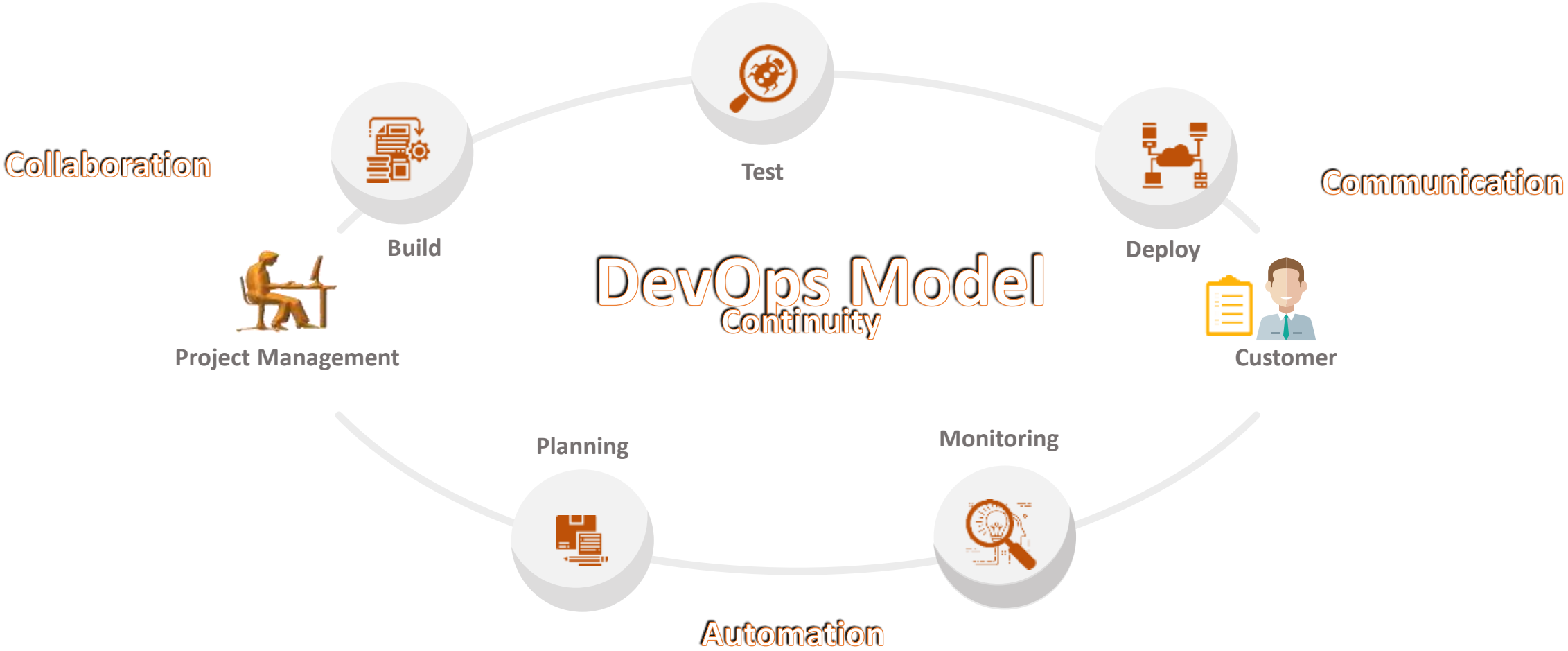
DevOps



- Break the Silos
- Communication (not only with emails)
- Collaboration
- Trust

- Involvement in the early development stages
- Automation is the key
- Continuous Integration
- Continuous Deployments in the lower environments
- Fail fast and fail often

DevOps


Collaboration
Build
Test
Deploy
Communication
Project Management
DevOps Model
Continuity
Customer
Planning
Monitoring
Automation

DevOps in Action

Continuous Feedback

Continuous Improvement

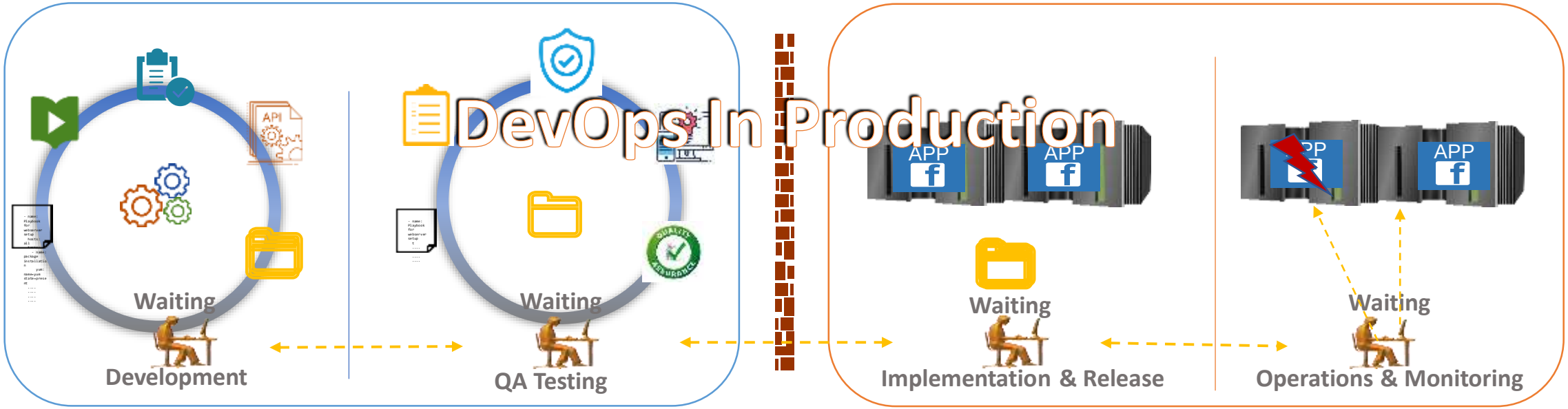
Continuous Planning

Continuous Integration

Continuous Delivery

Continuous Deployment

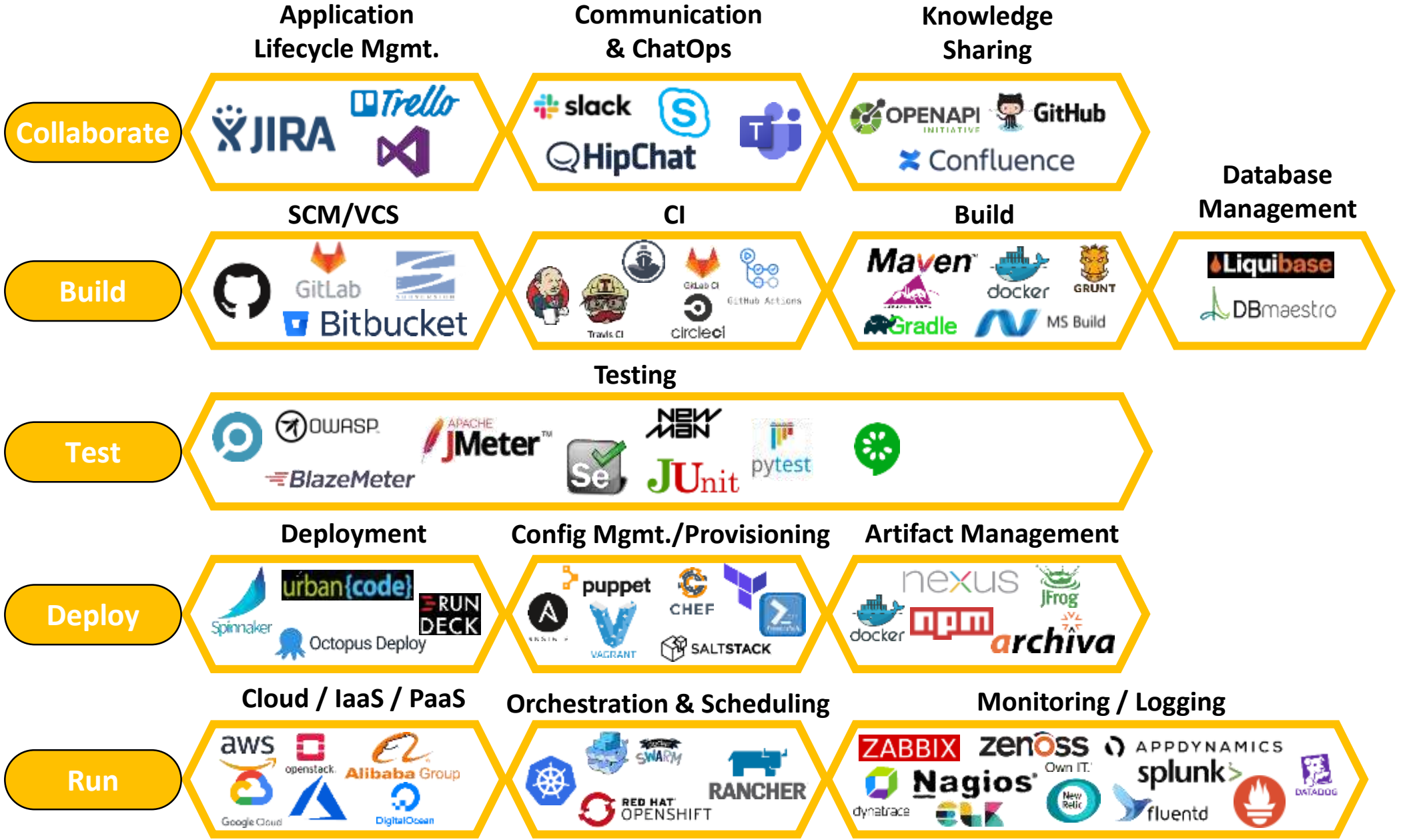
Continuous Monitoring



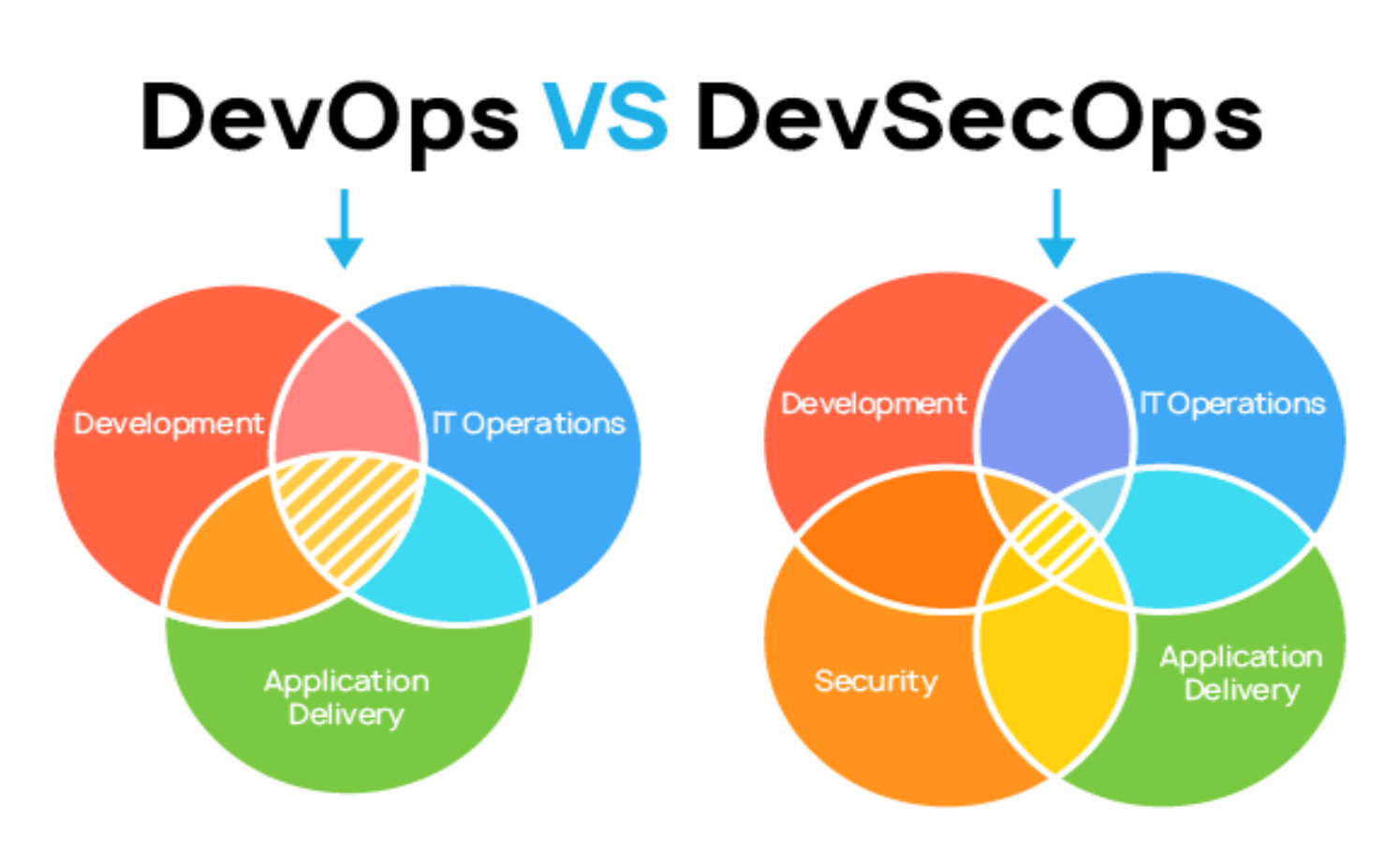
Development

Opsrations

DevOps Toolsets



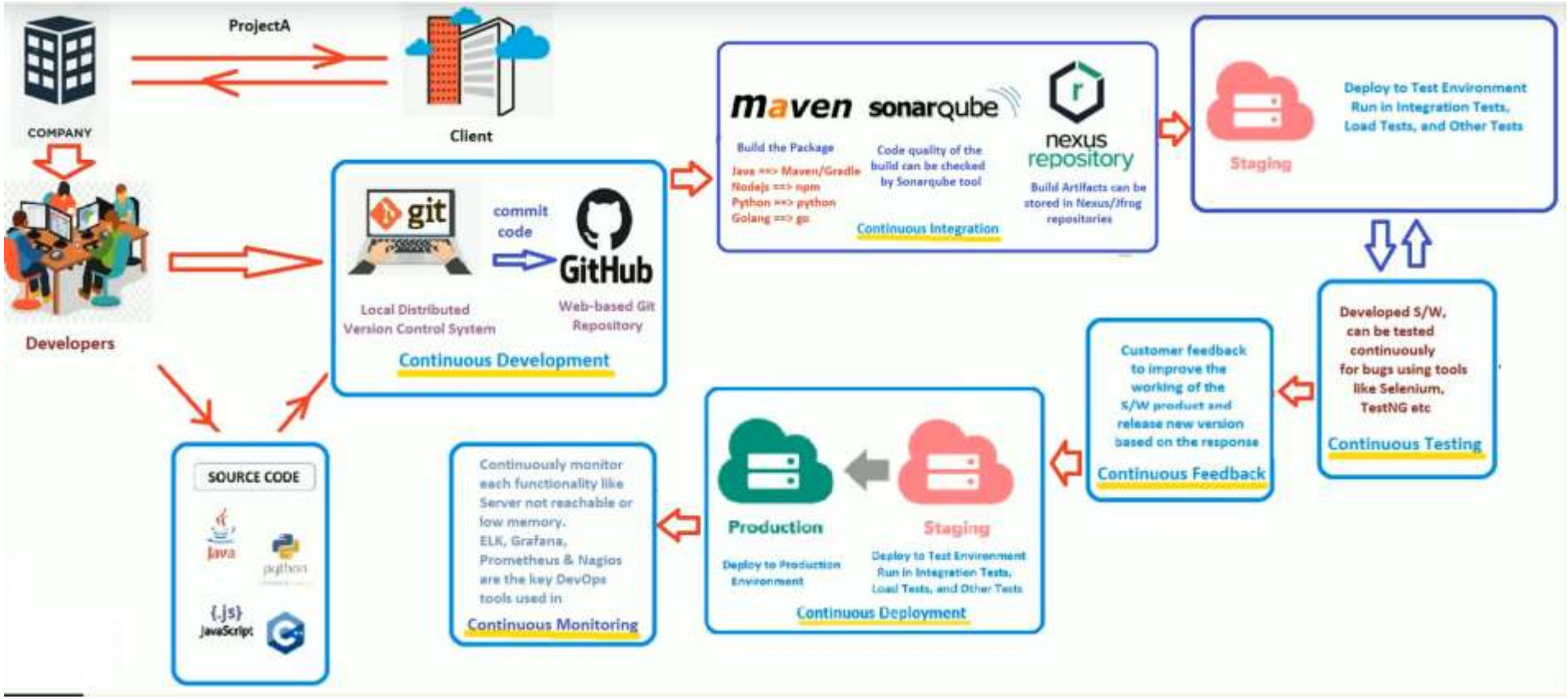
DevOps vs DevSecOps



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 archit

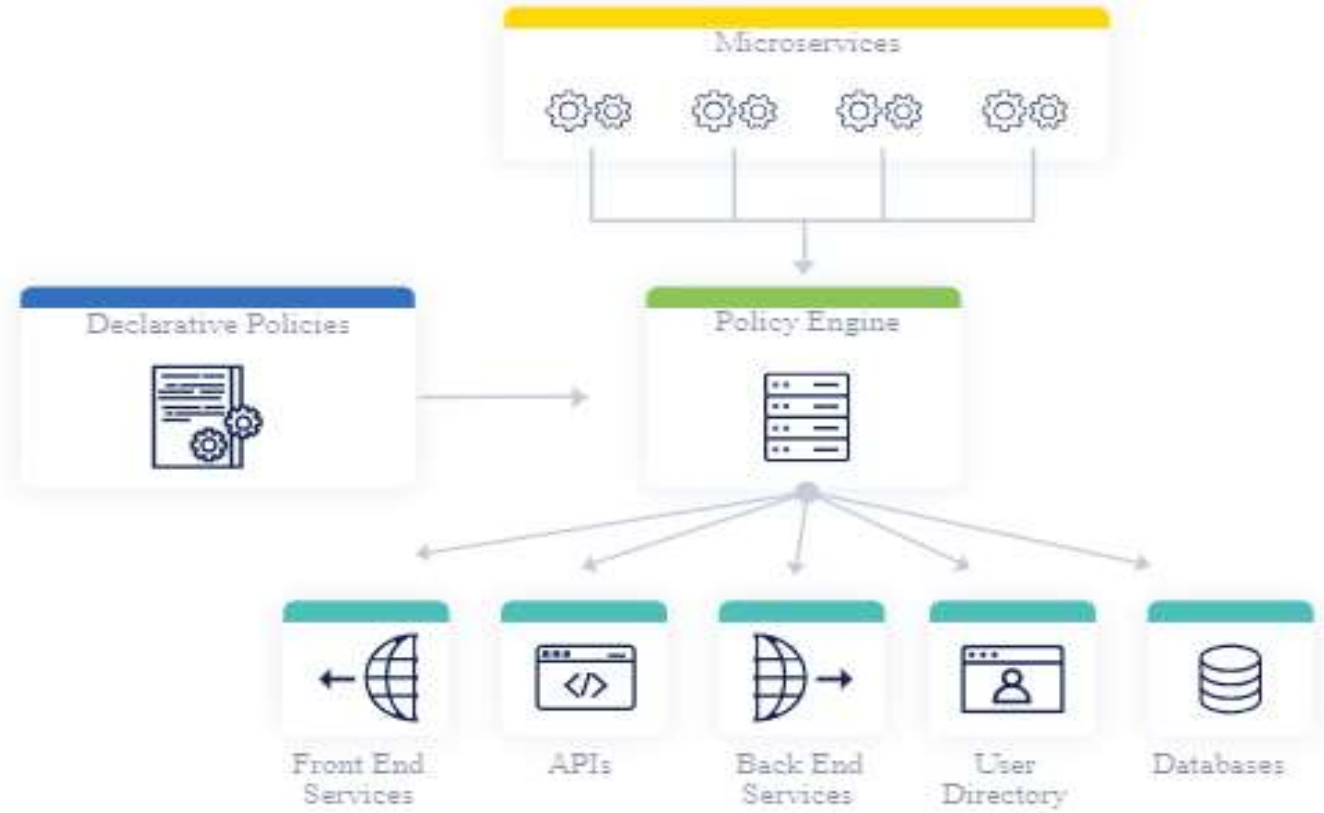
across your pipeline can verify that each
of your application and deployment is se
against known vulnerabilities.

- 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 in class 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

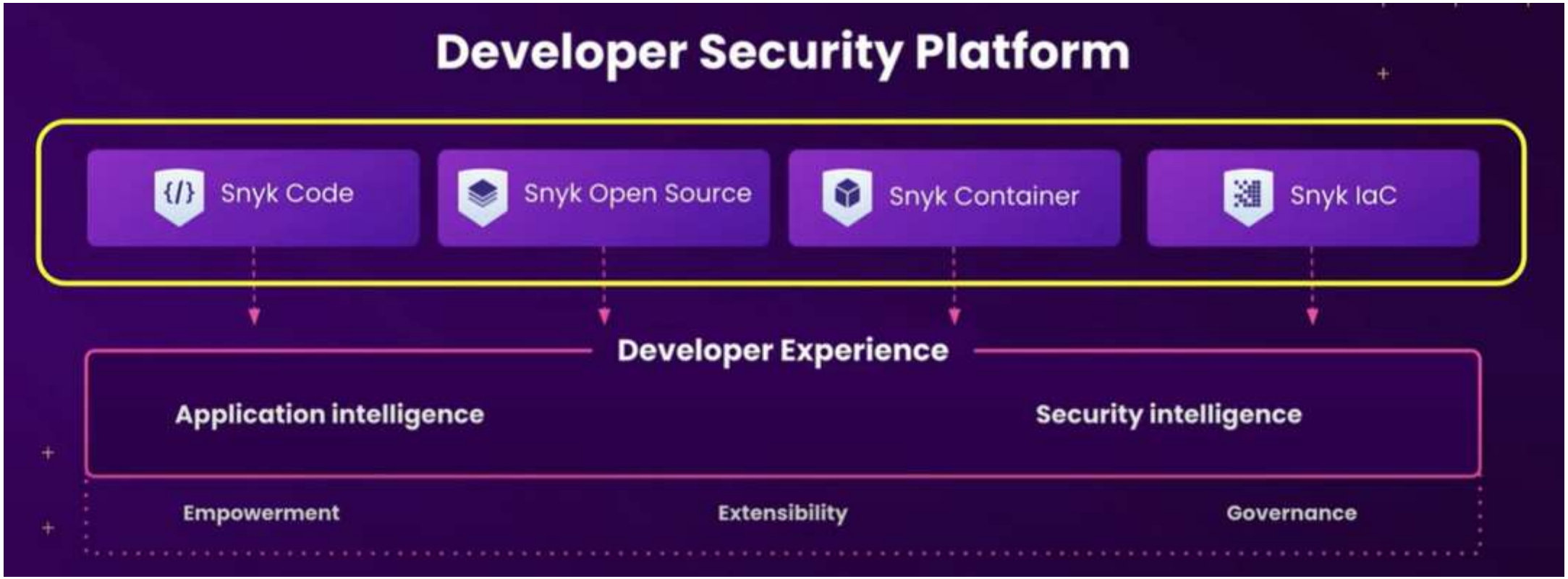


What is Snyk?

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. <u>Architecture:</u>**
- **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.

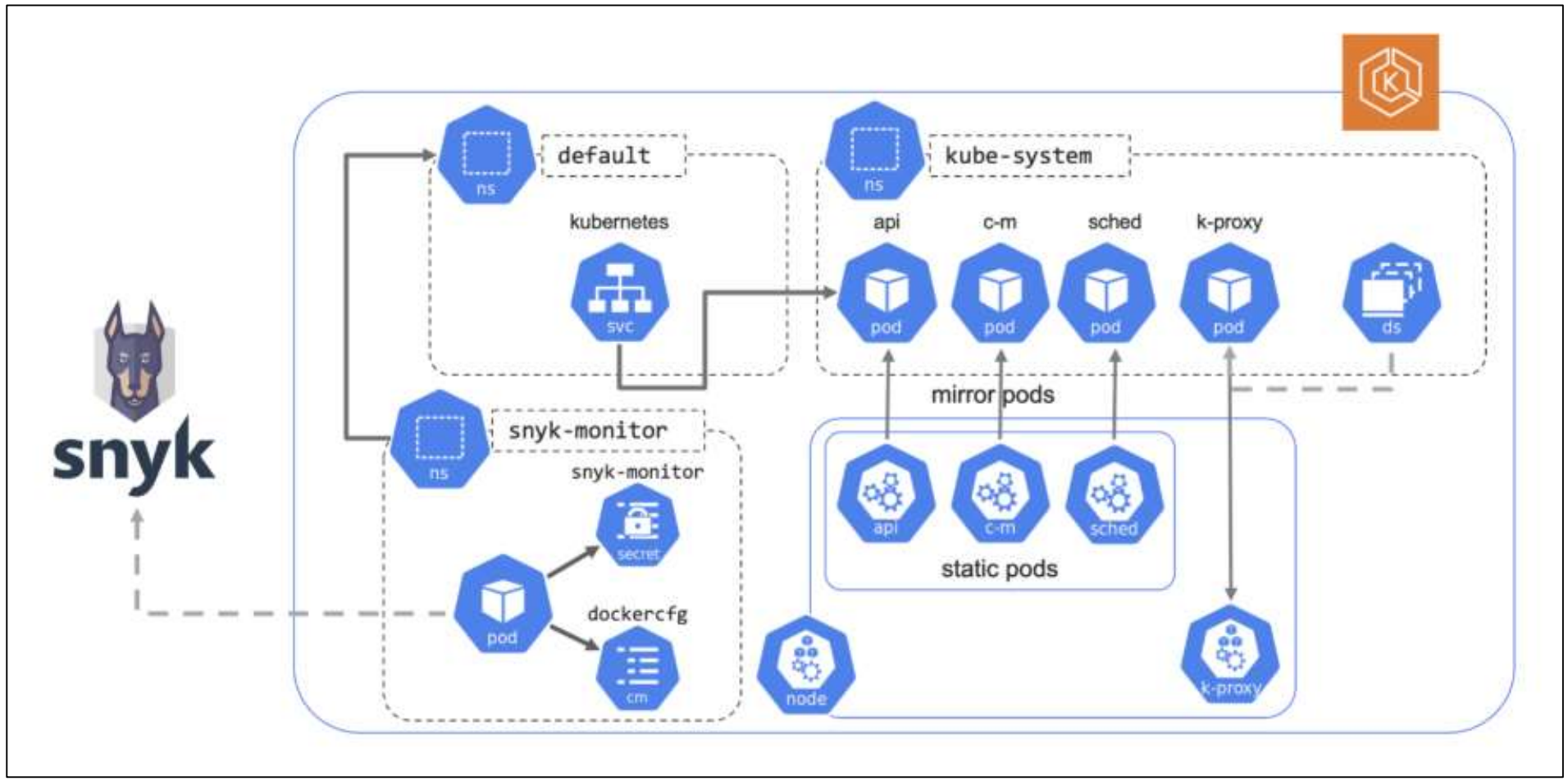
Snyk Features

- **2. <u>Workflow:</u>**
- **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. <u>Key Aspects:</u>**
- **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



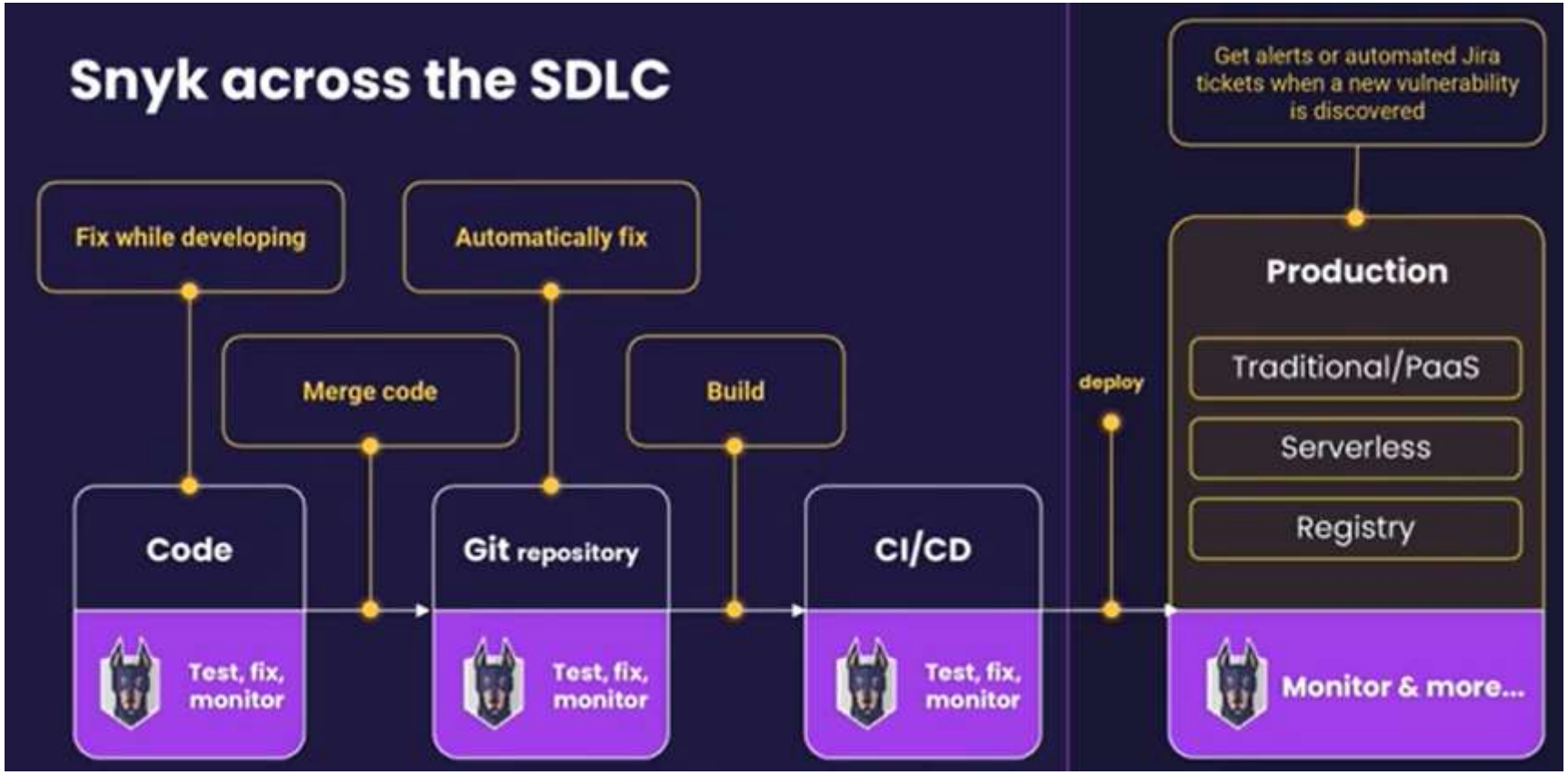
Architecture Layer

- **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.

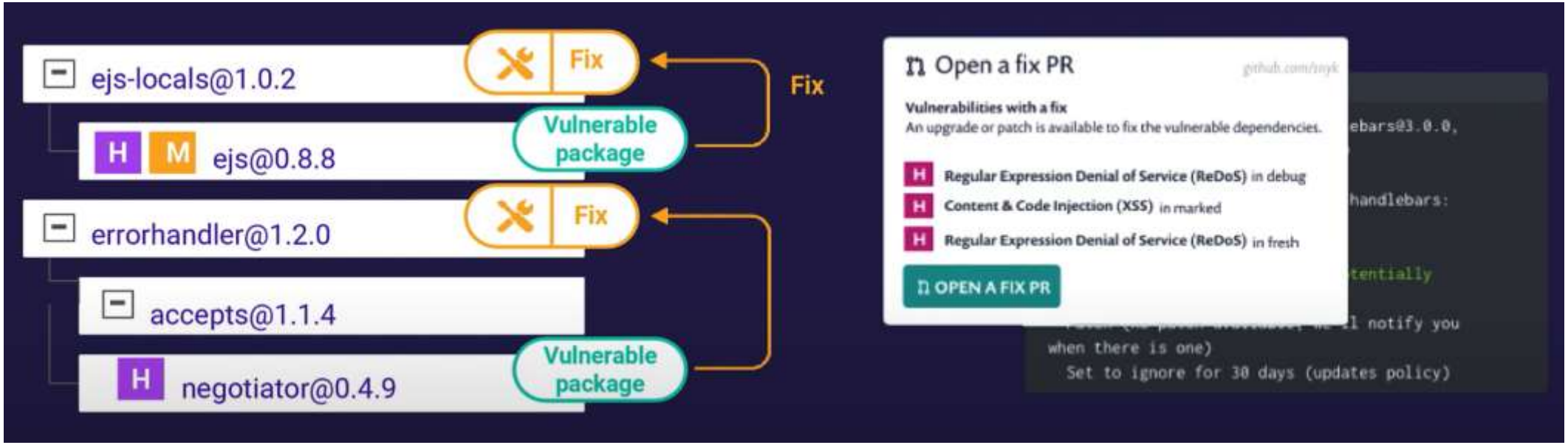
Workflow

- **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.

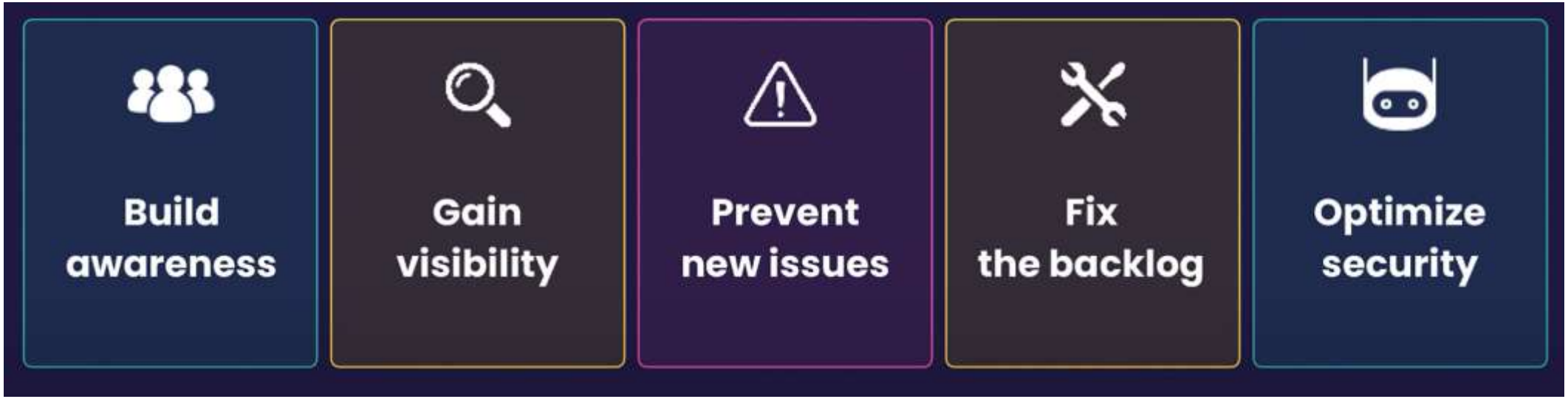
Snyk across the SDLC



Discovery and Fix issues



Managing your security with Snyk



Synk SaaS Account

Snyk Account

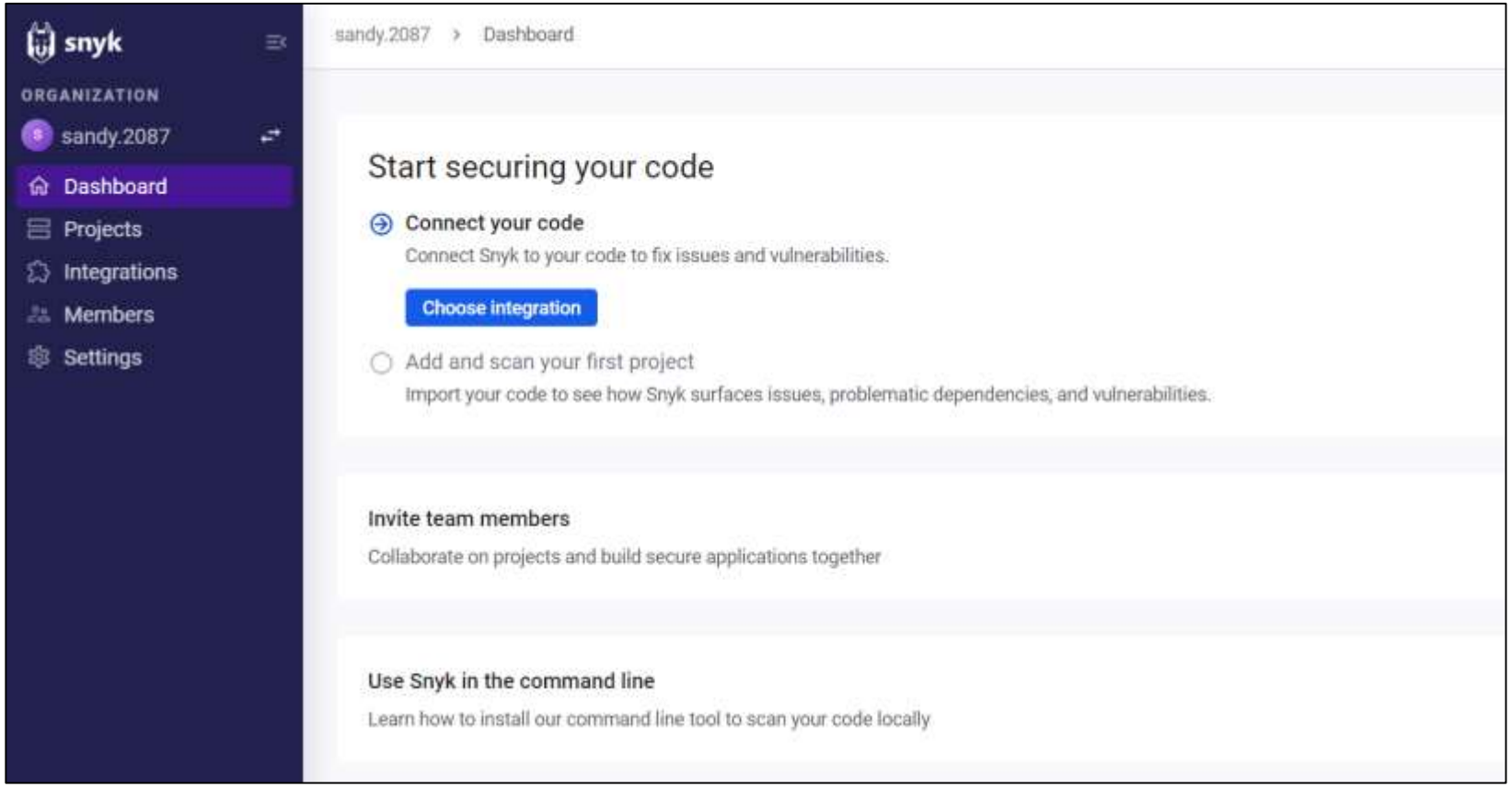
SaaS (Cloud-Based Solution):

- 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.

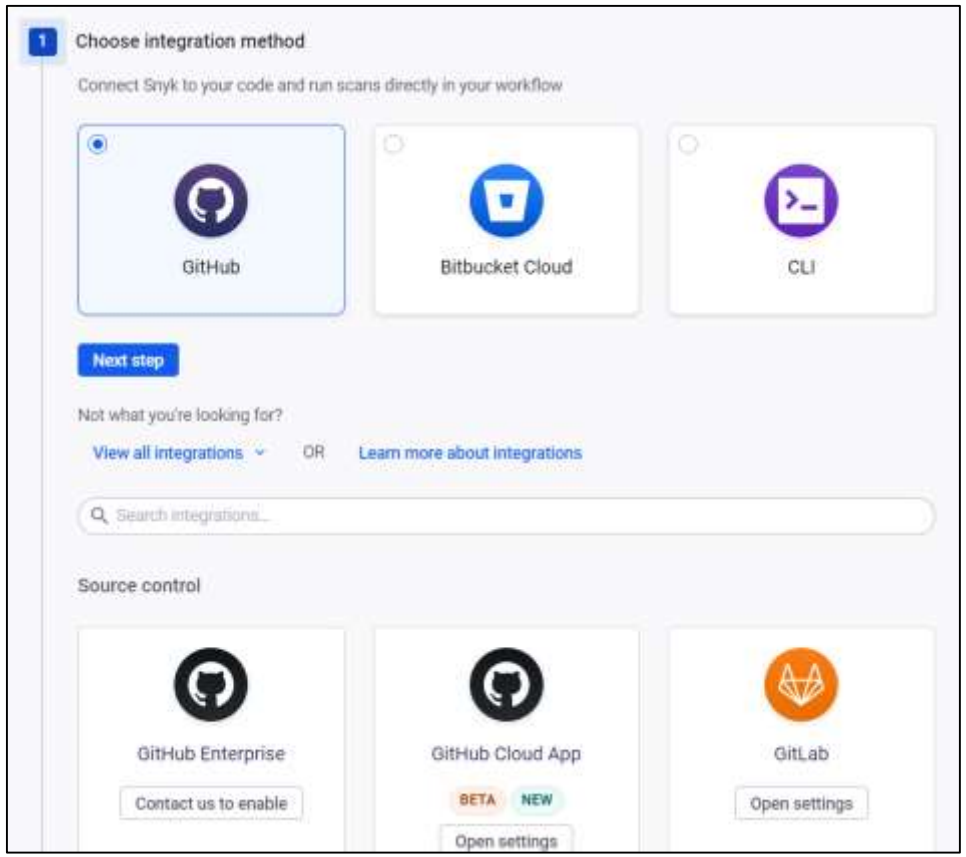
Integration Steps: Github

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

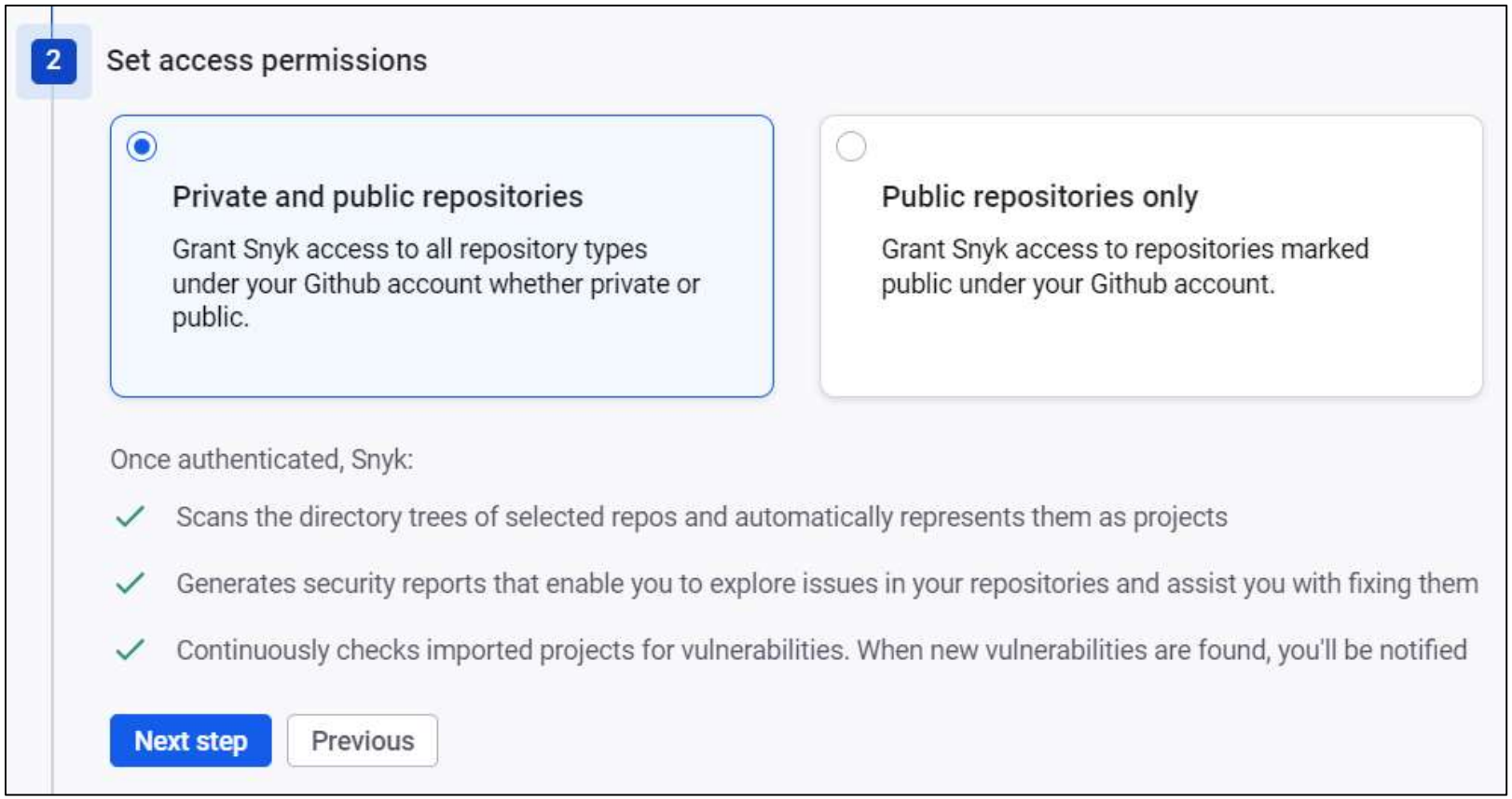
Snyk Account



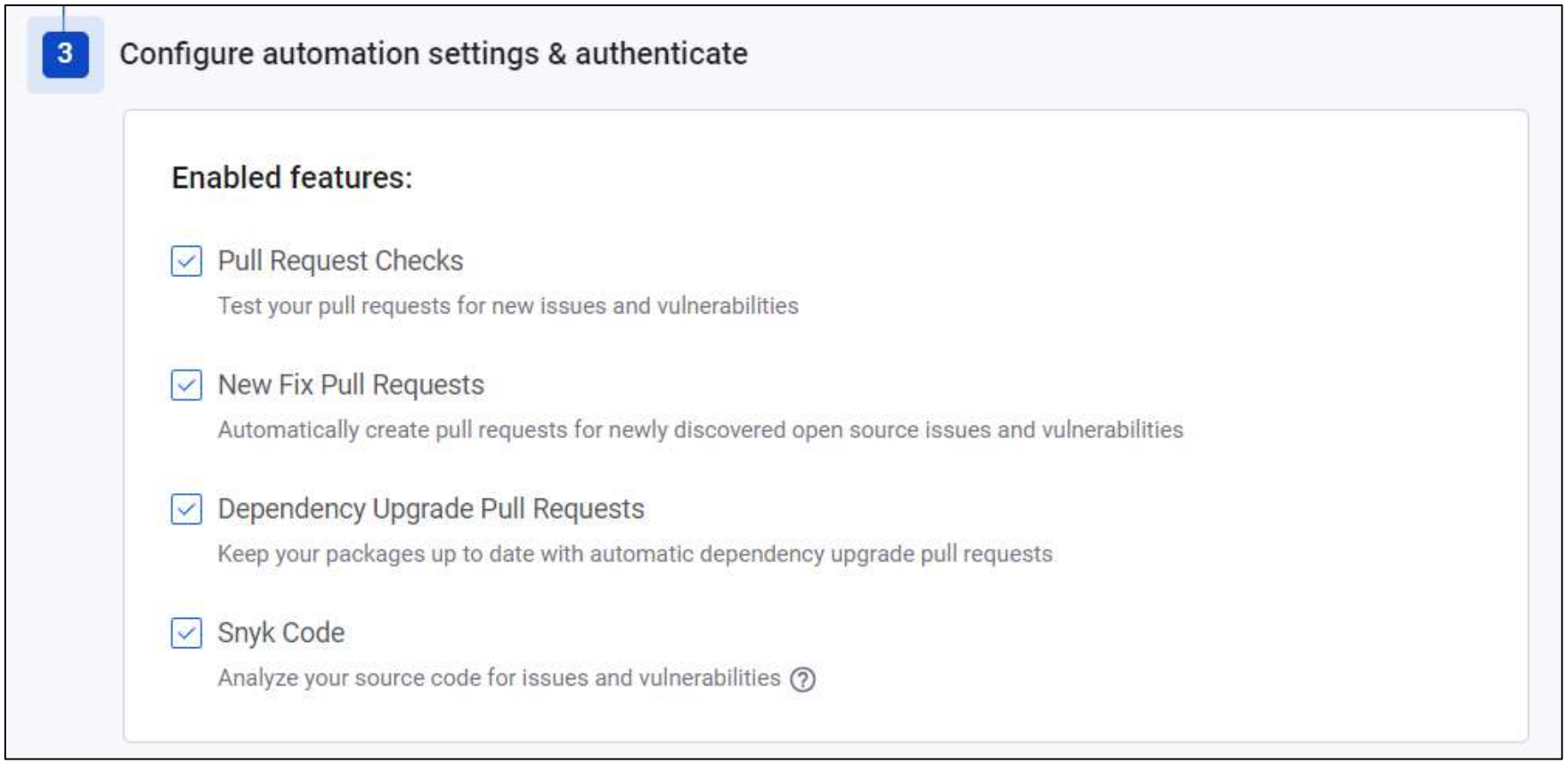
Snyk Integration



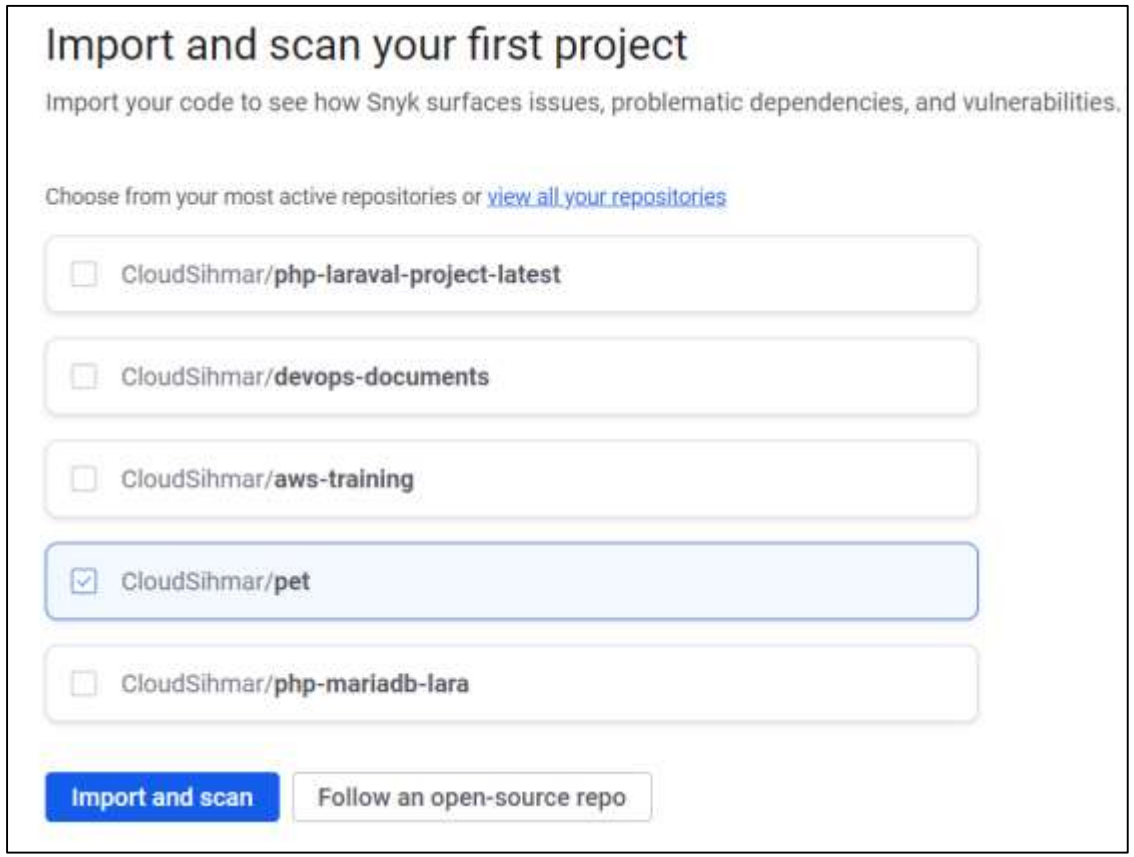
Repository Access



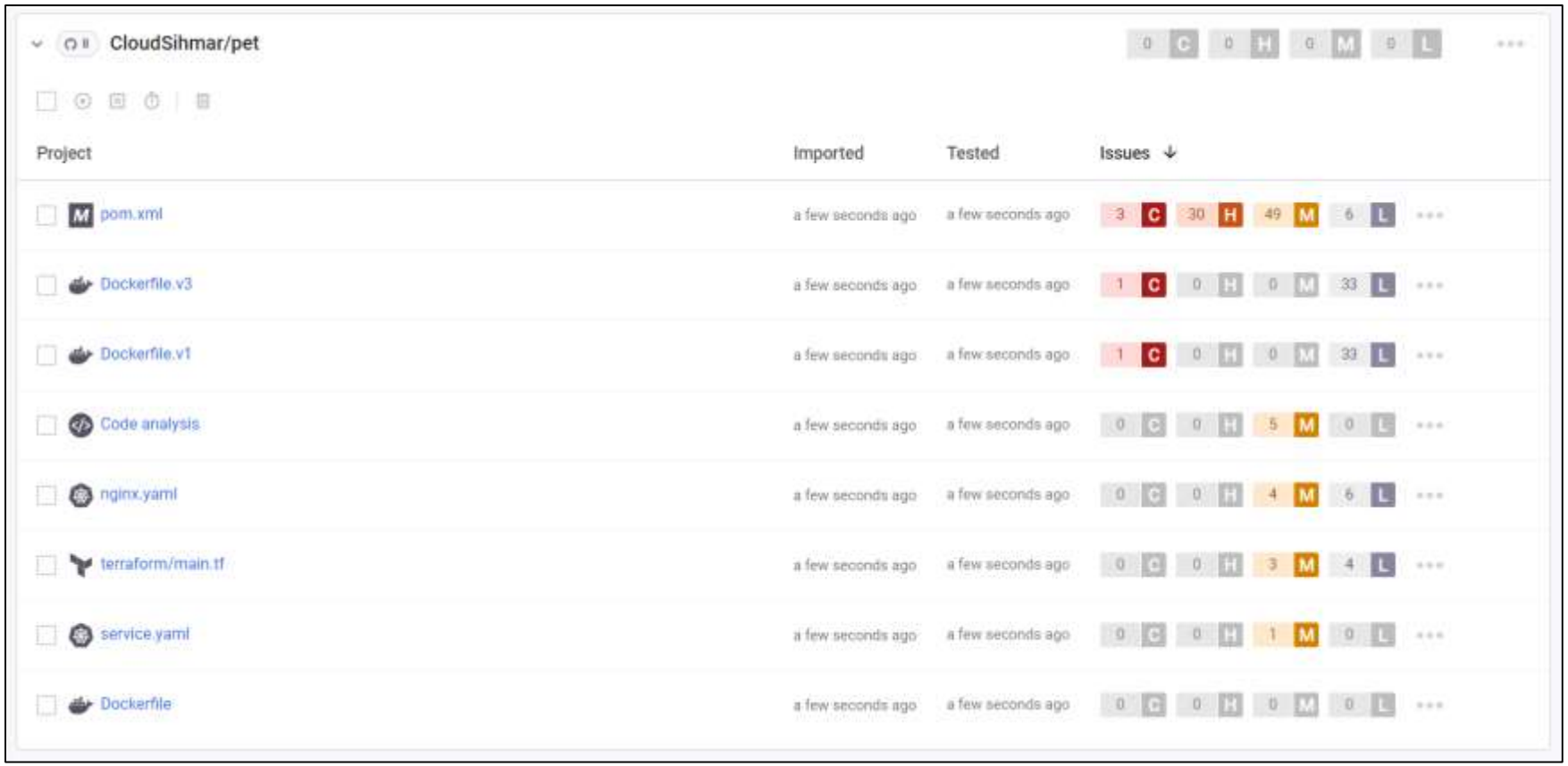
Automation Setting



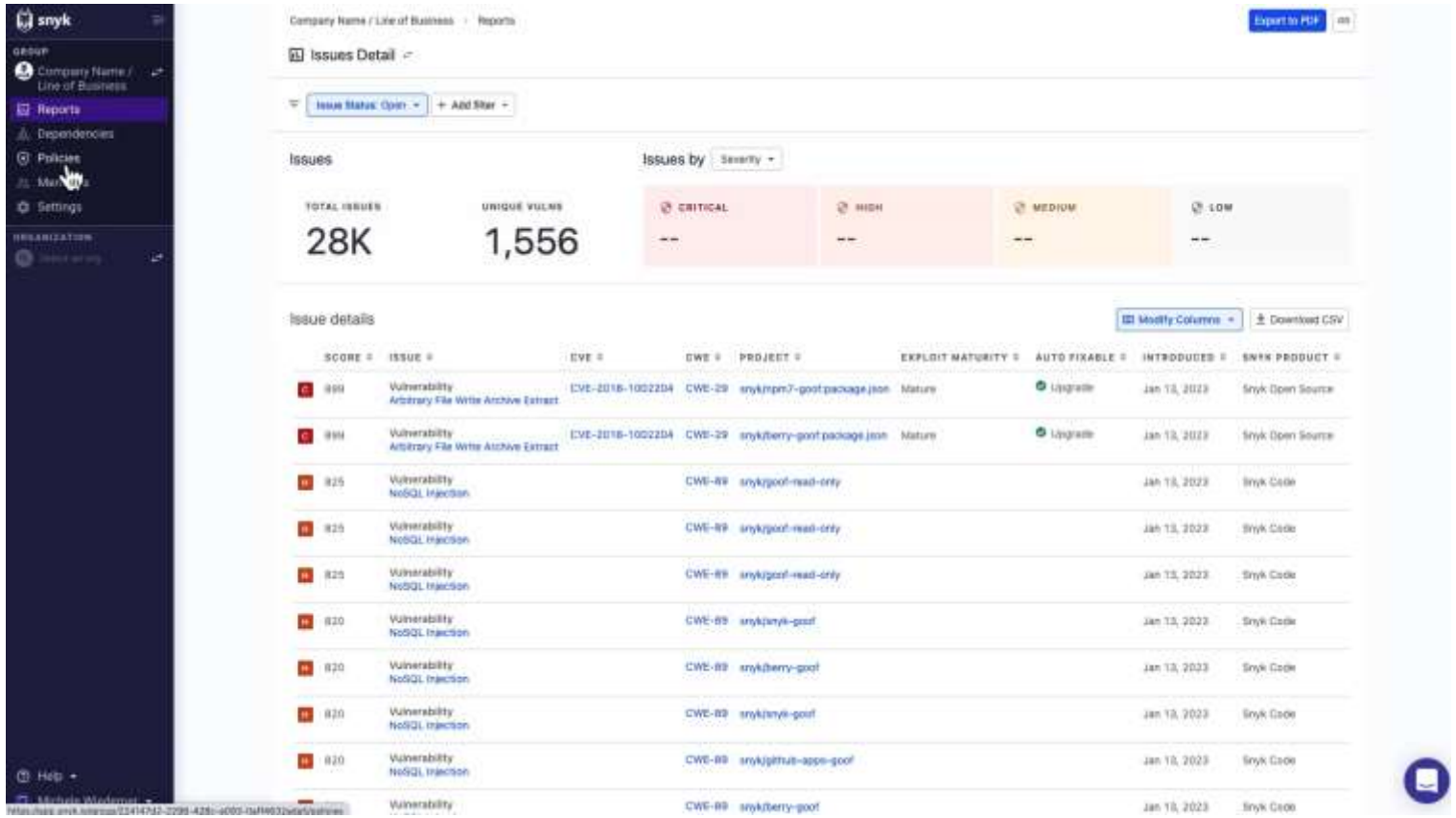
Import and scan the code



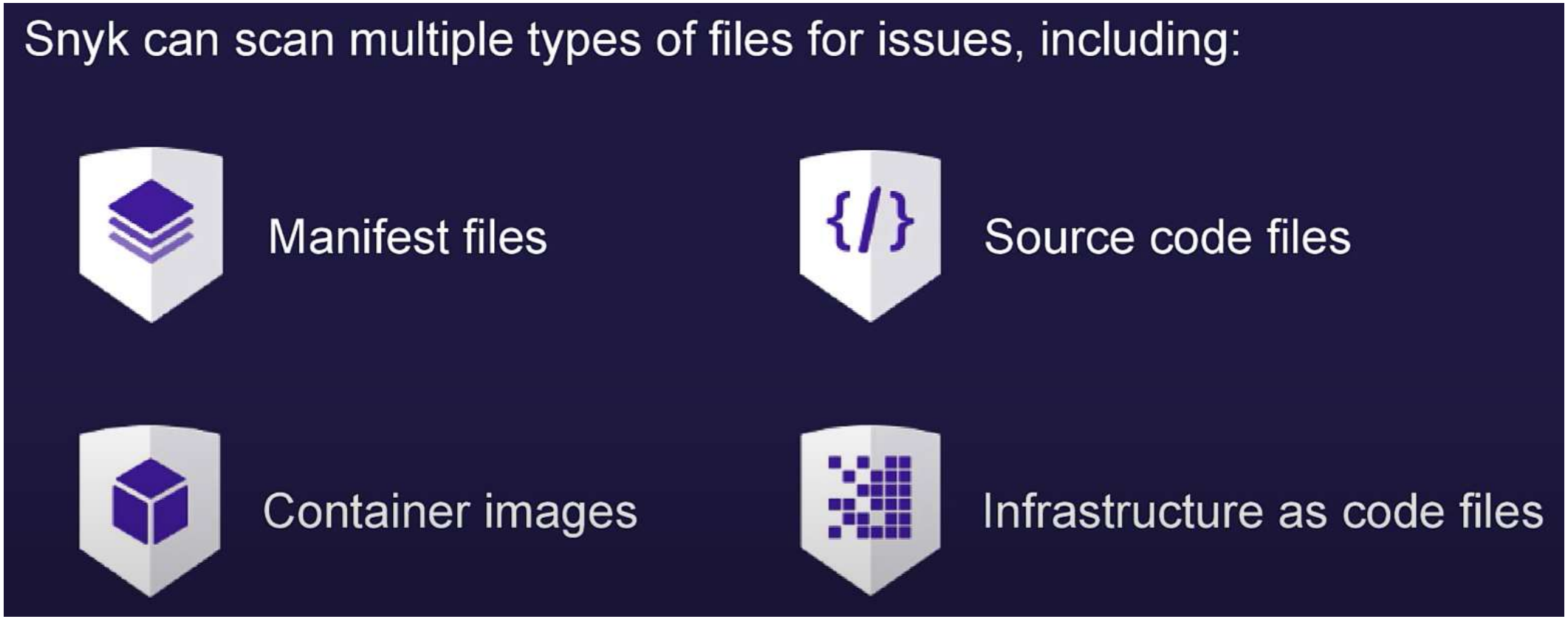
Scan Result


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

Group level overview

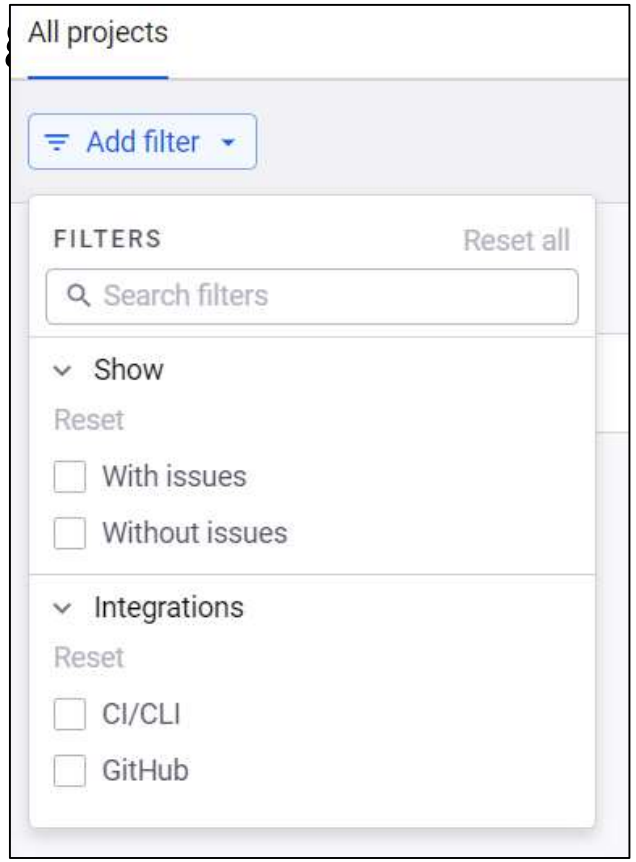


What is a Project?

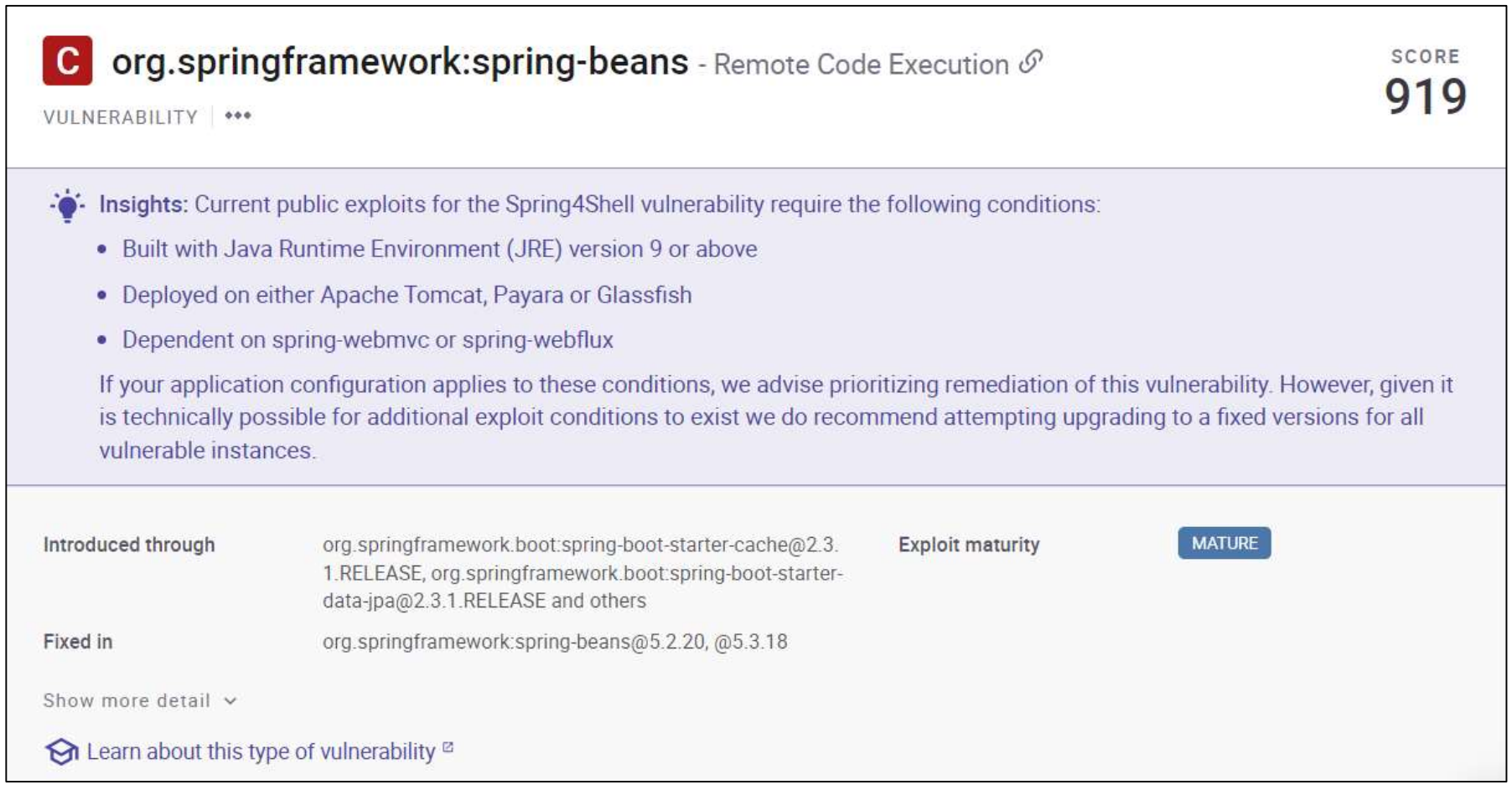


Project

- Filtering can be done based on issues and inte



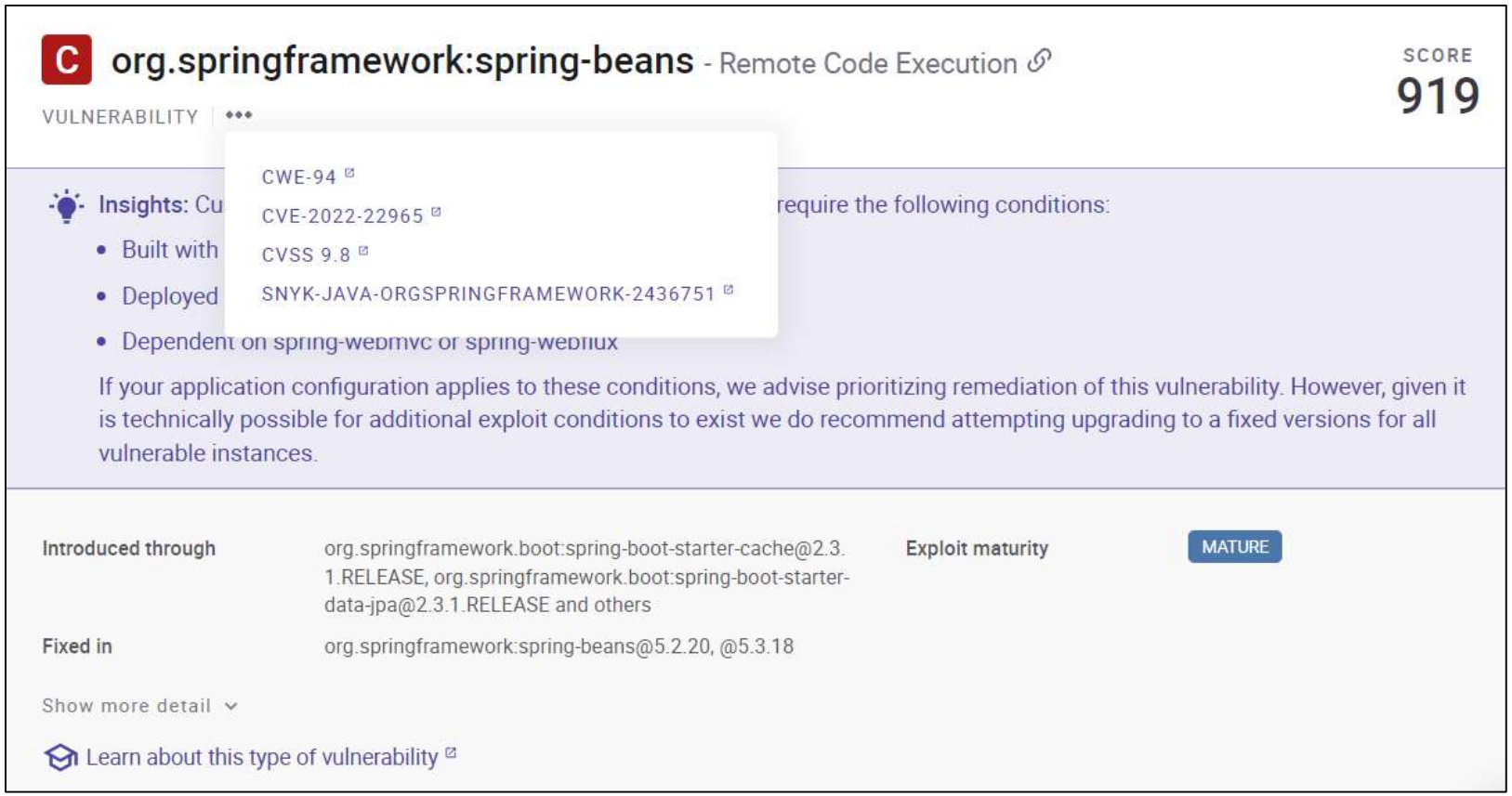
Vulnerability



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



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.

Snyk 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.

Snyk 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.

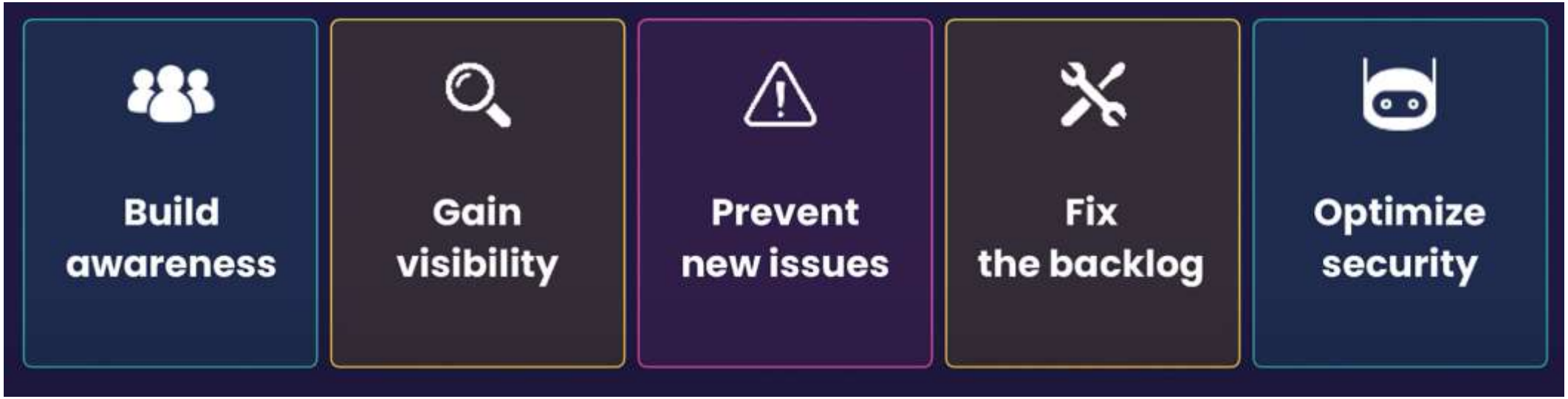
Risk Awareness:

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.

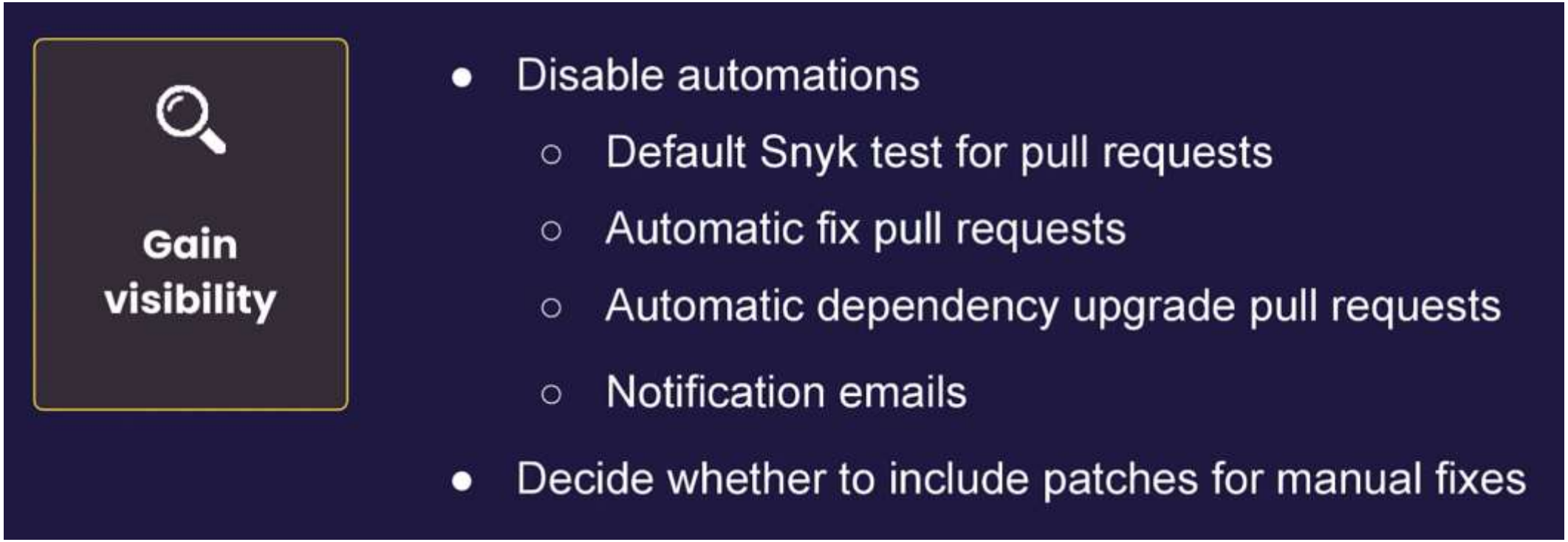
Managing your security with Snyk : Phases



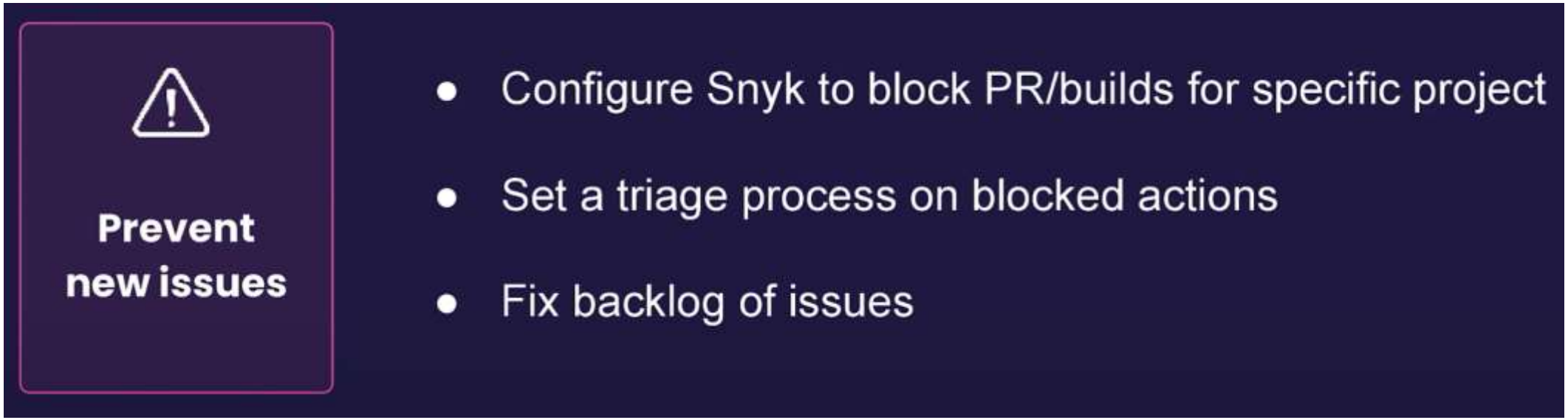
Managing your security with Snyk: Phases

- **Snyk Developer Adoption model:**
- Gain visibility
- Prevent new issues
- Fix the backlog
- Optimize security

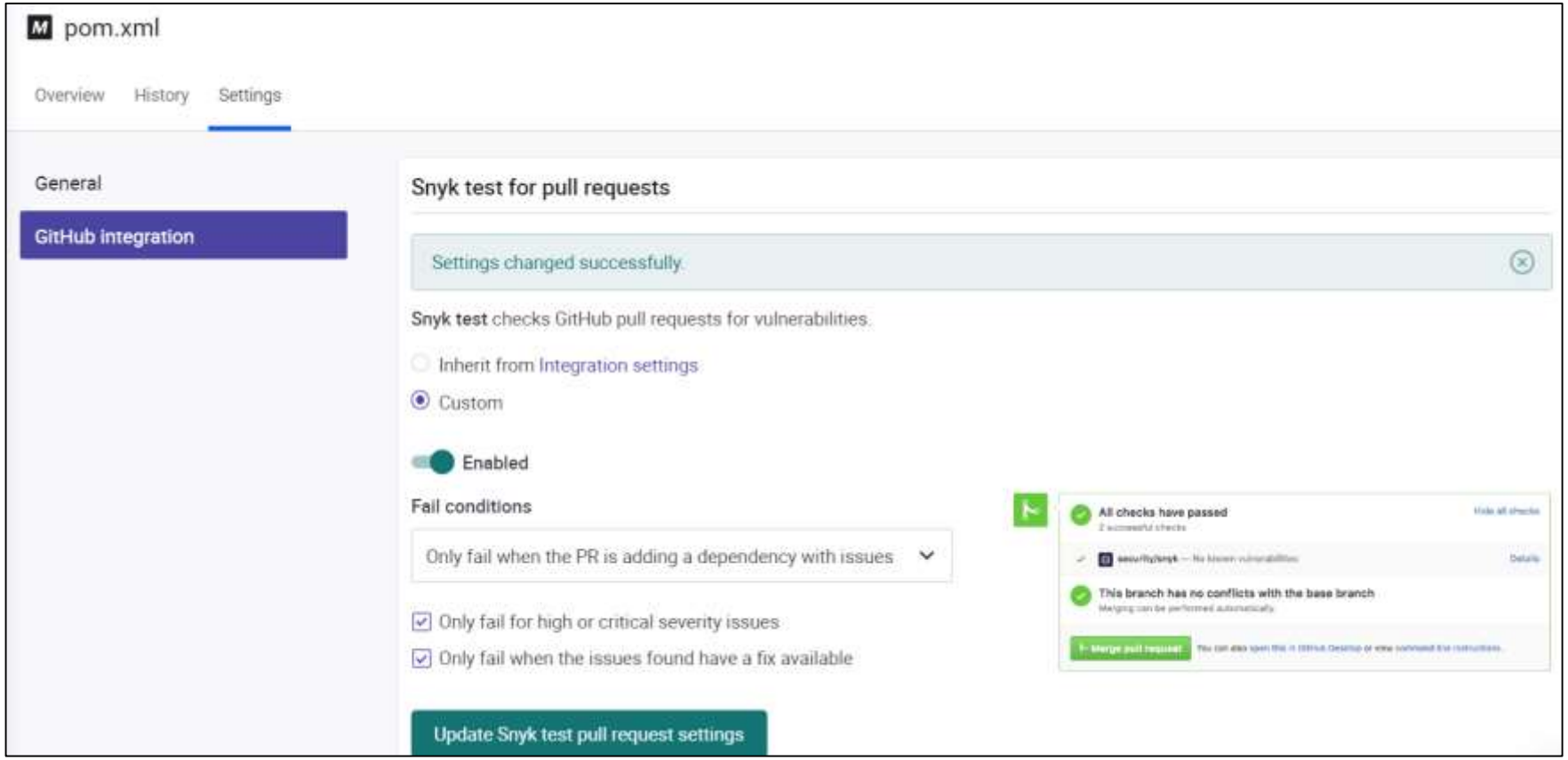
Maturing your security with Snyk



Maturing your security with Snyk

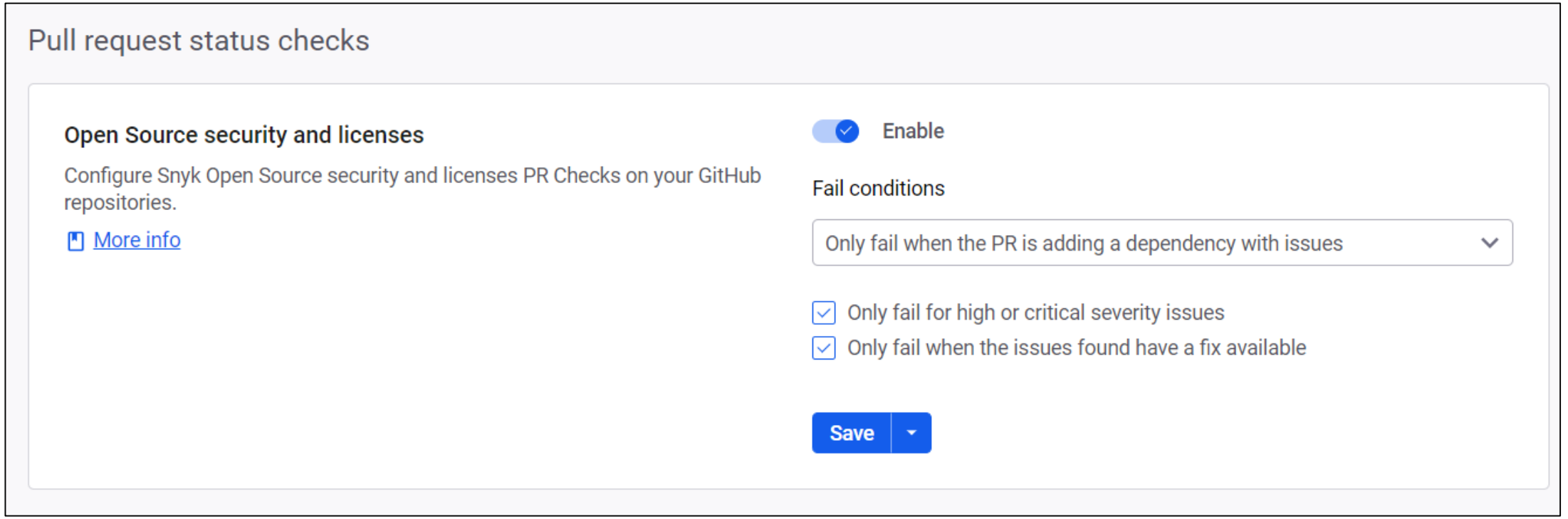


Maturing your security with Snyk: Project Level



Maturing your security with Snyk: Organization Level

Snyk >> Setting >> Github



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.

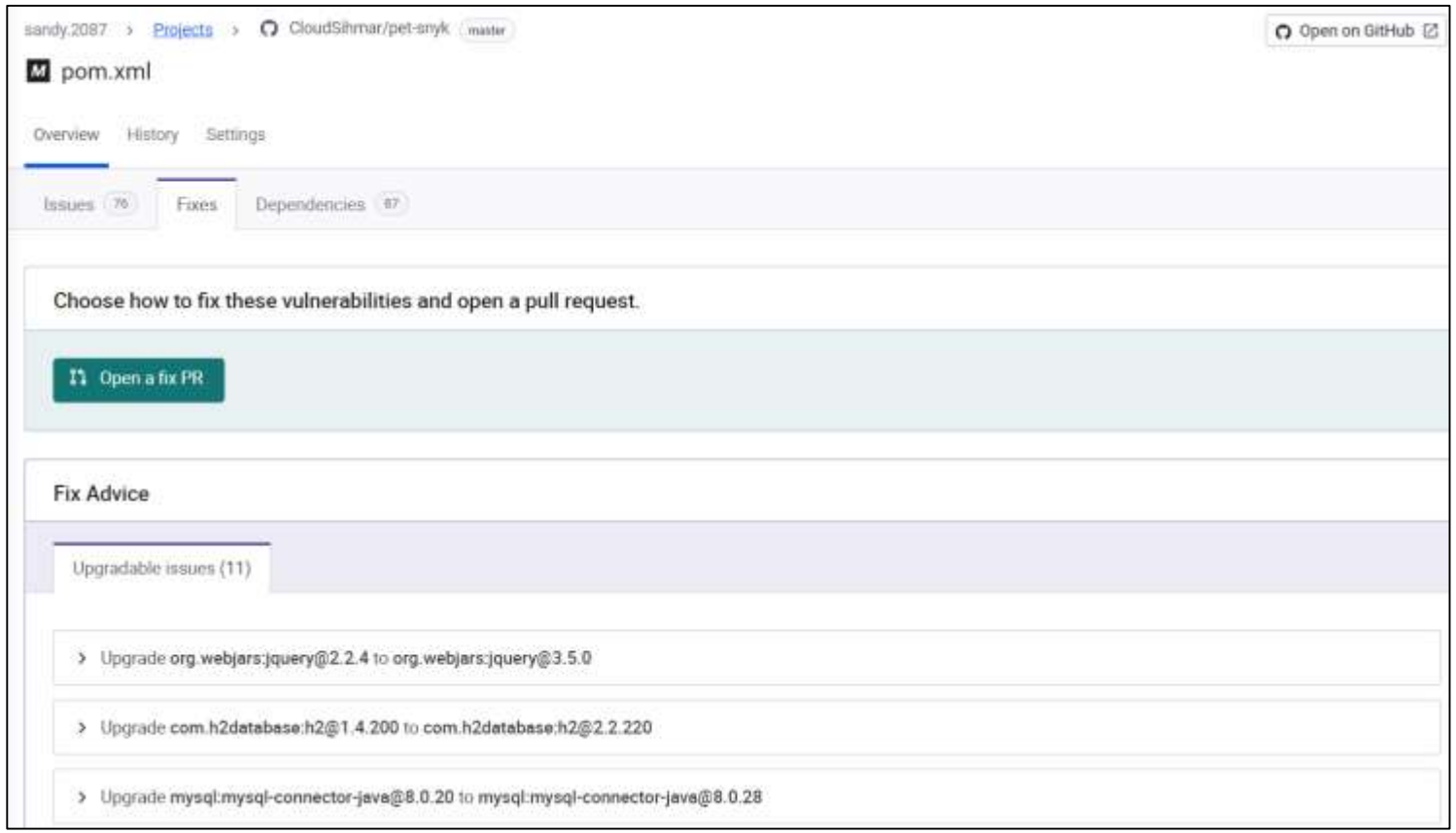
Snyk Test for Pull Requests: Manual

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.

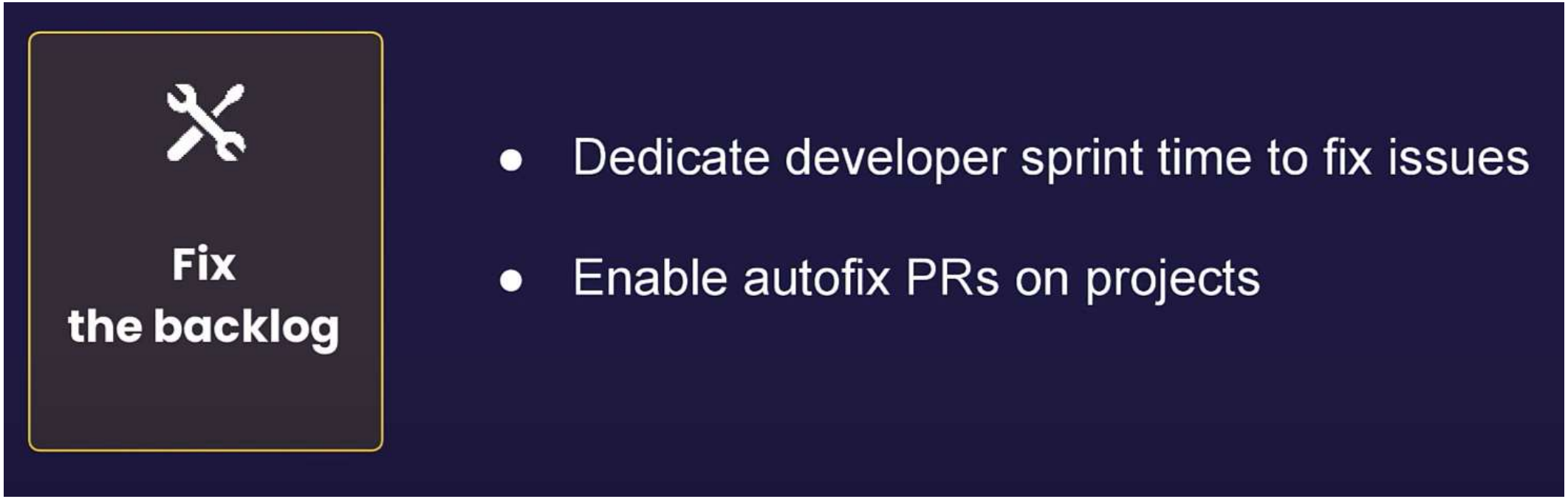
Snyk Test for Pull Requests: Manual



Snyk Test for Pull Requests: Automatic

- **<u>Automatic Pull Request:</u>**
- **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

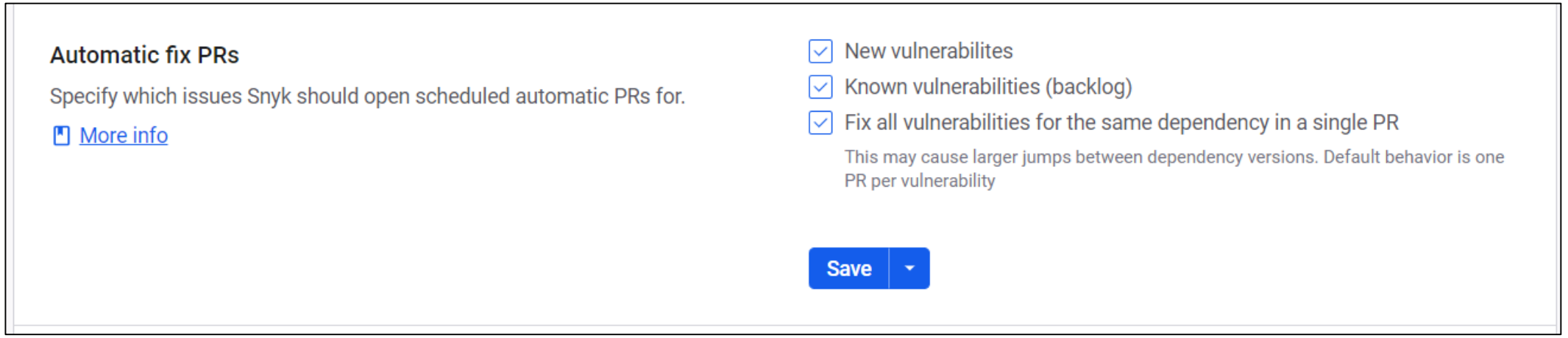
Maturing your security with Snyk



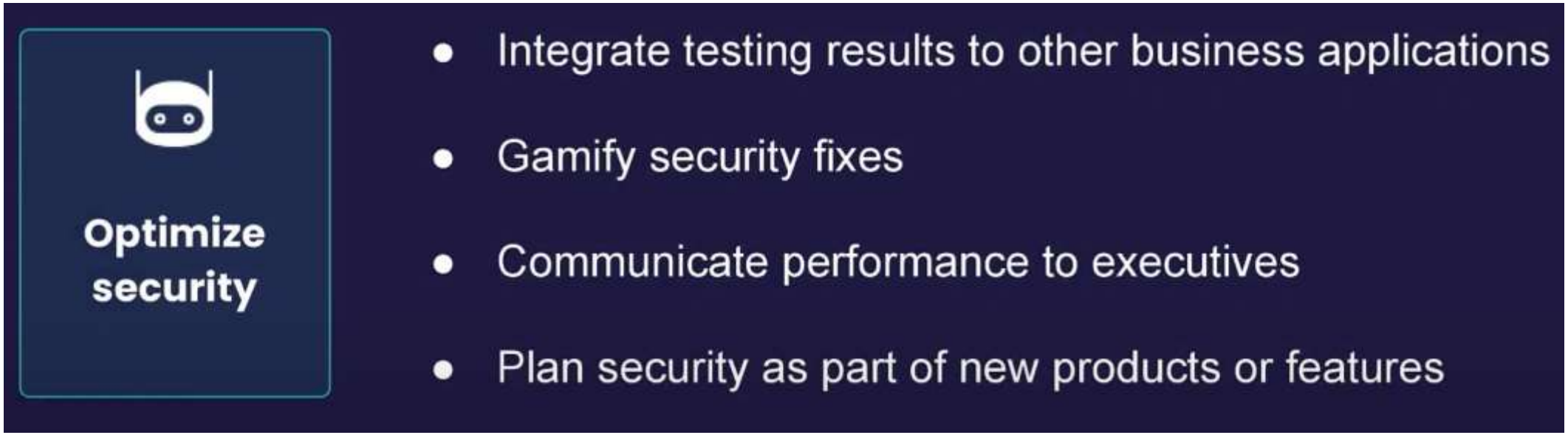
Maturing your security with Snyk

- **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.

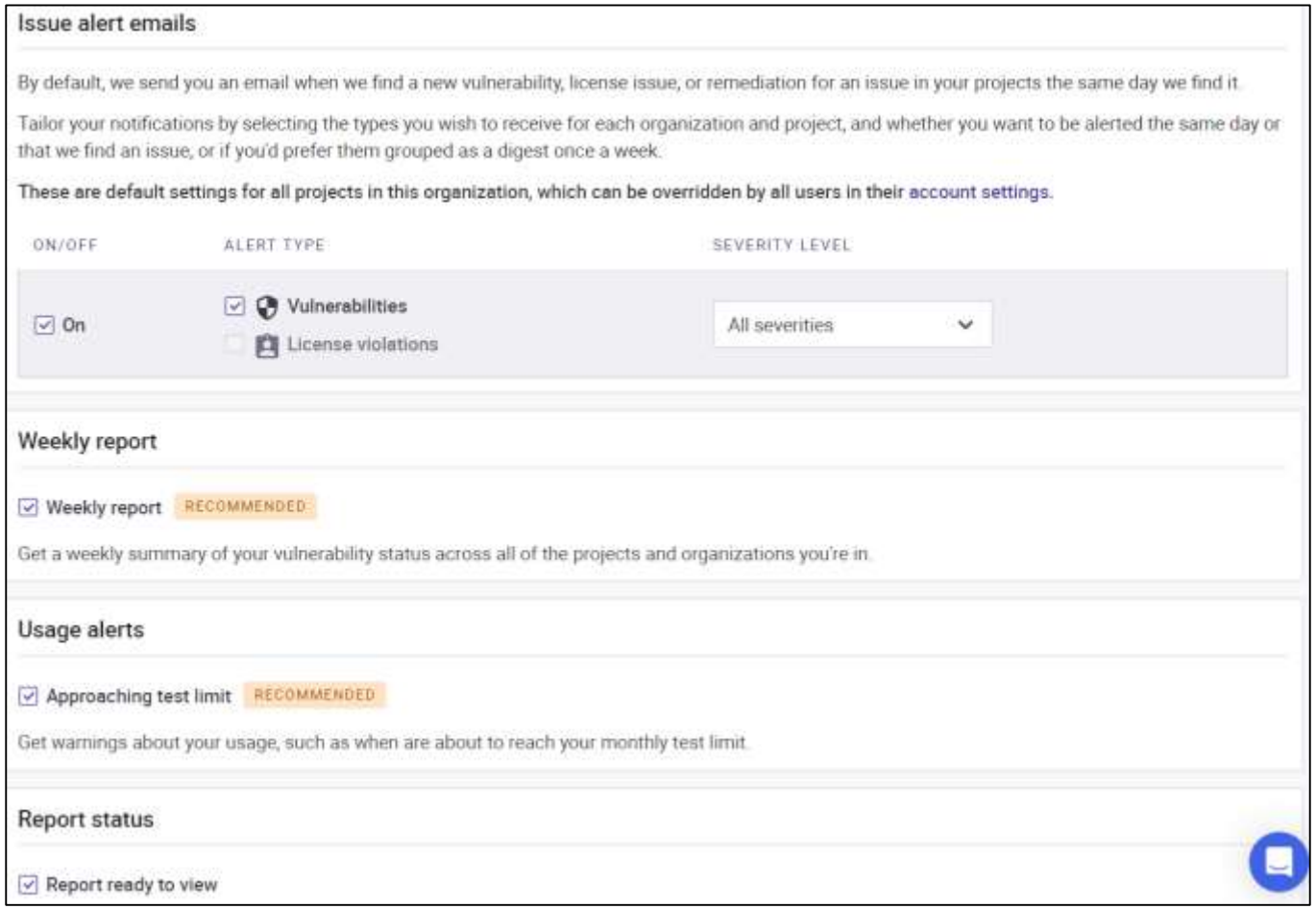
Maturing your security with Snyk



Maturing your security with Snyk



Notifications



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

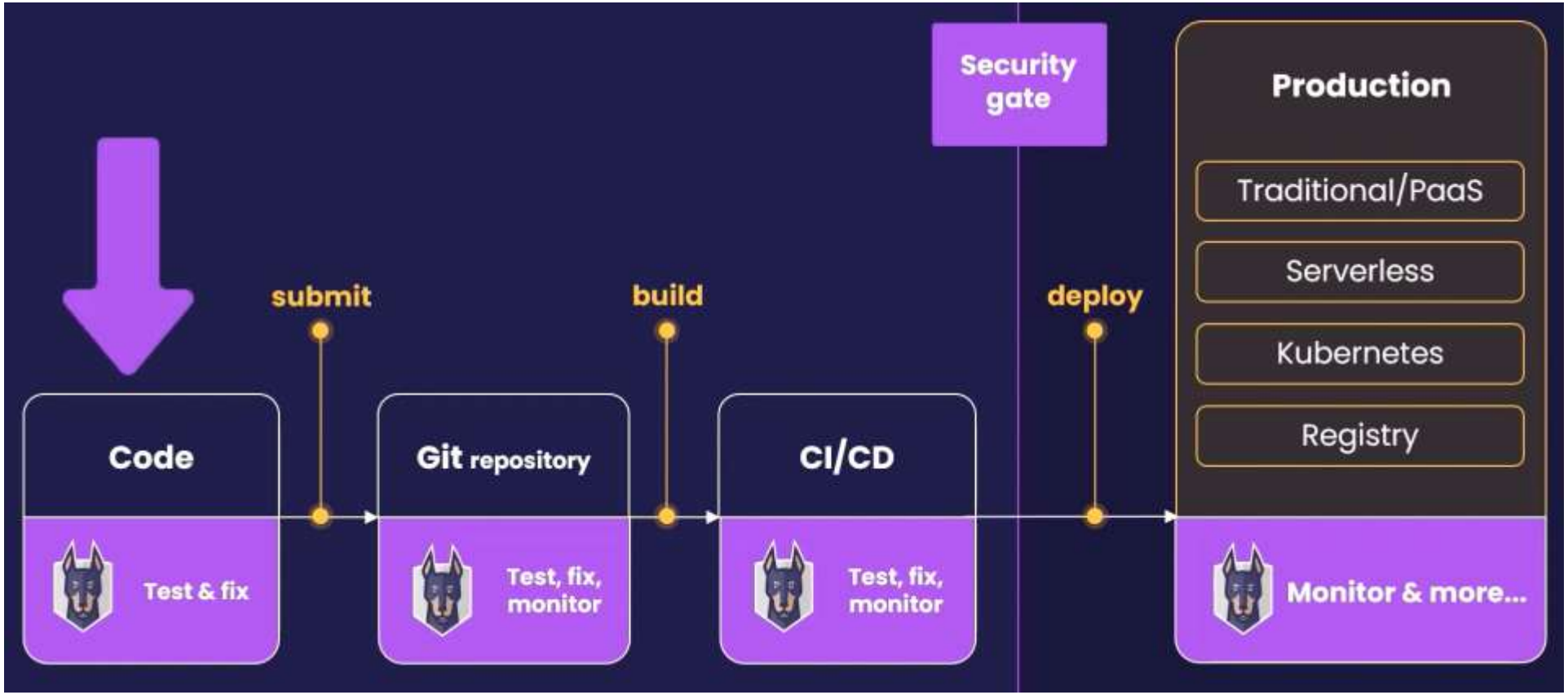
- Nexus
- Artifactory
- NPM

Issue management integrations

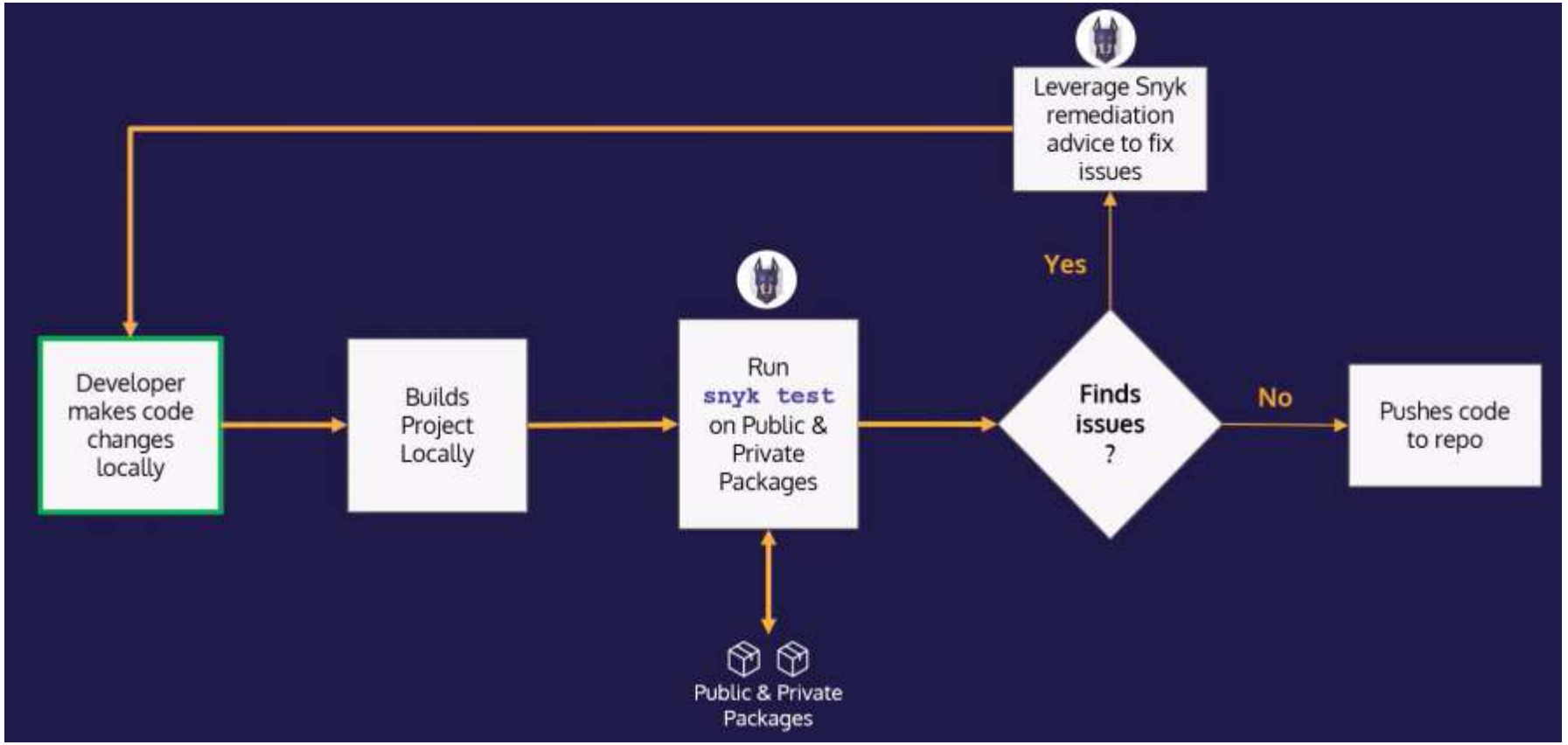
A sampling of our issue management tool integrations:

- Slack
- Jira
- Snyk API

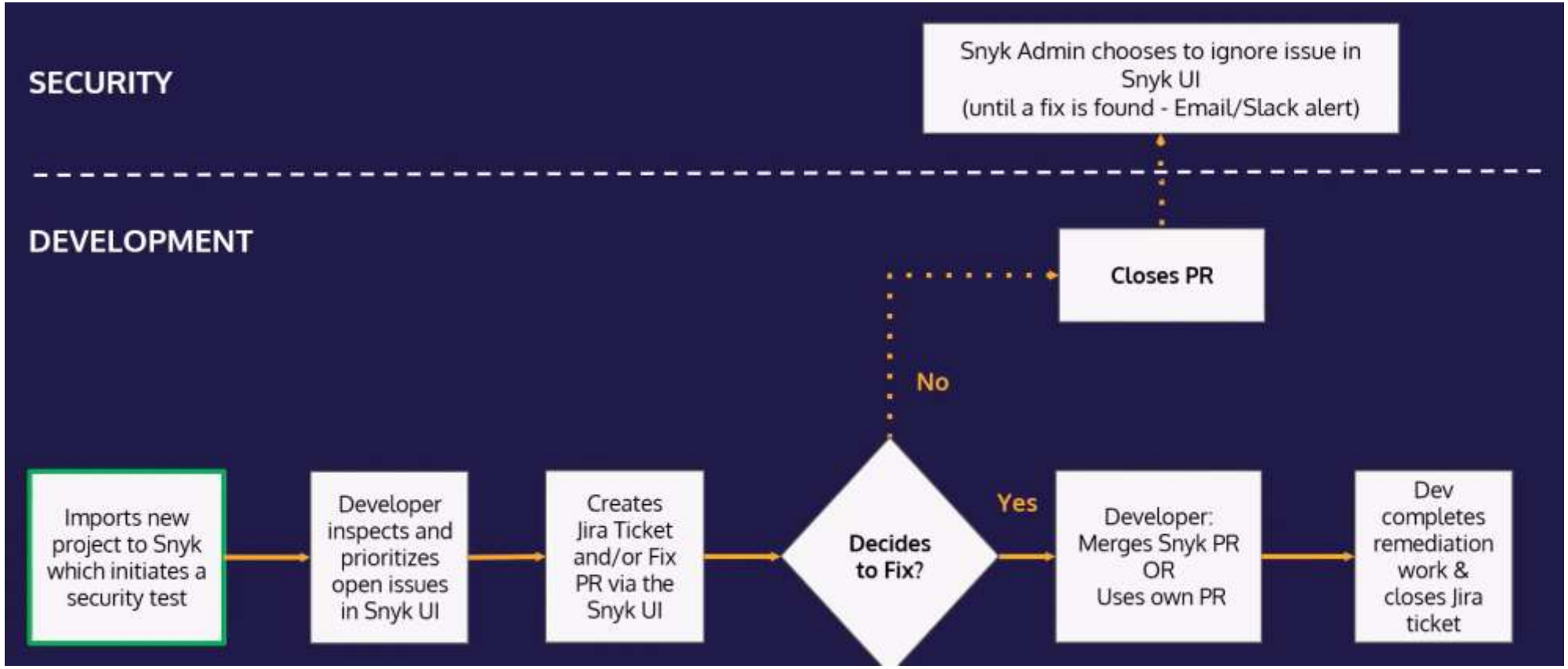
Synk test in CLI/IDE



Synk test in CLI/IDE



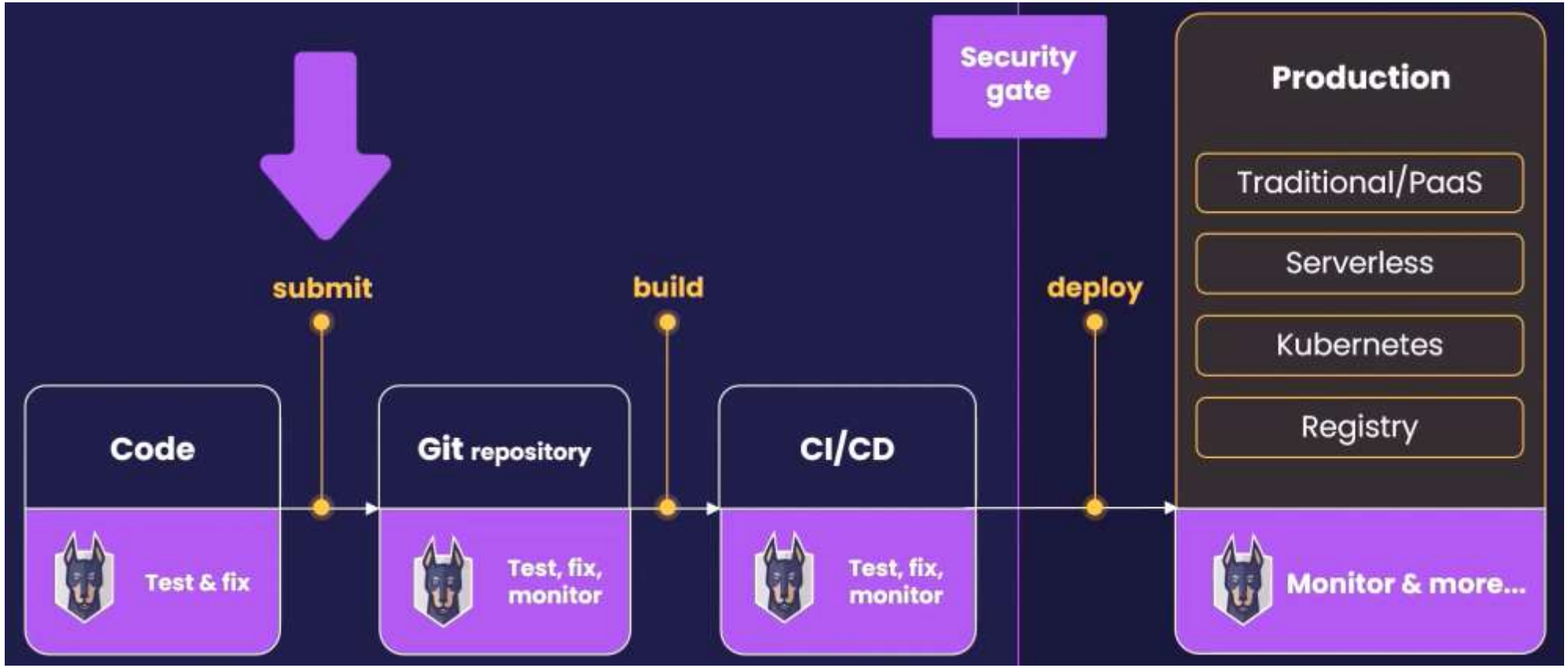
Synk Manual Fix PR Workflow



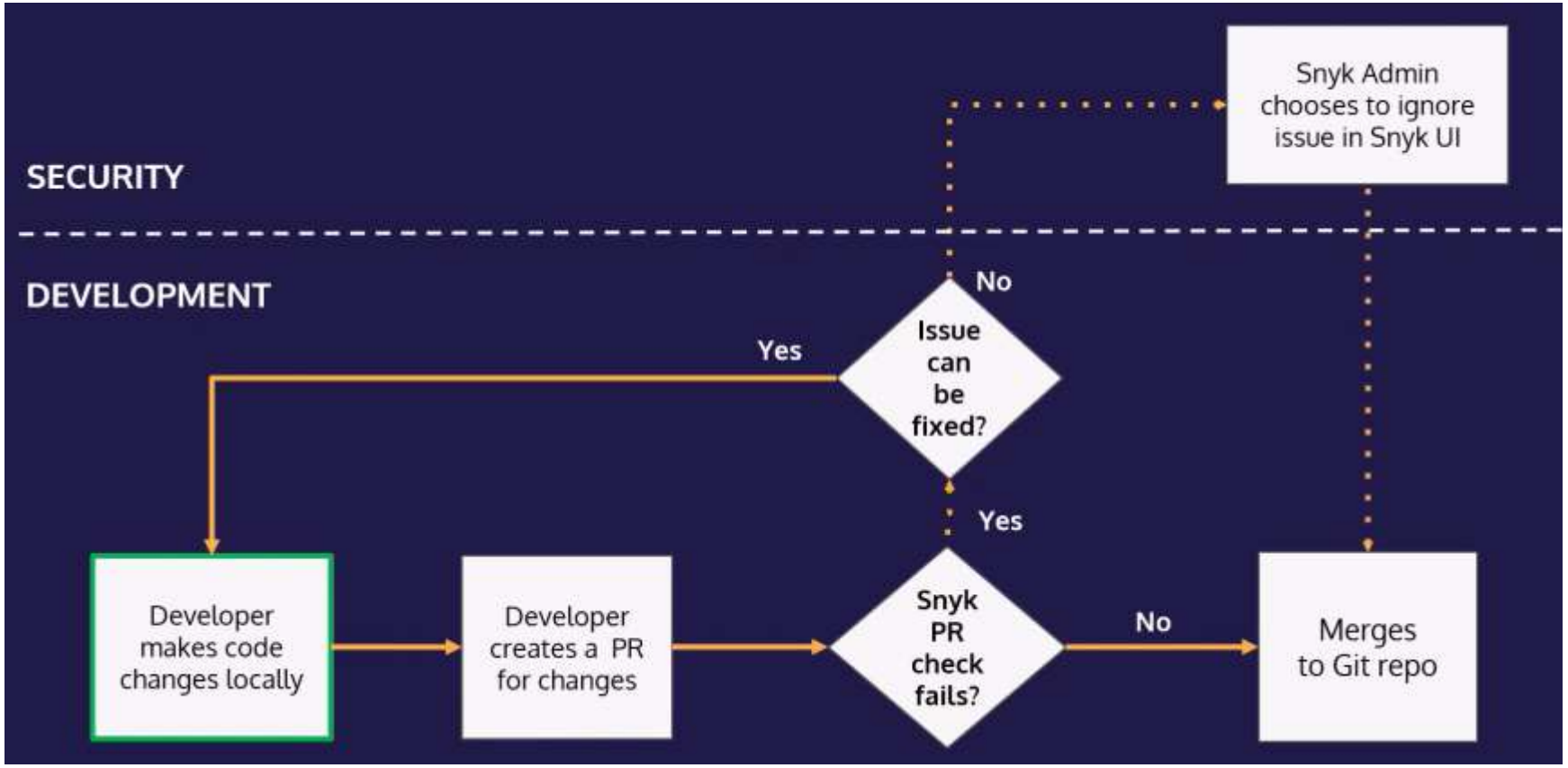
Git integration workflows

- 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

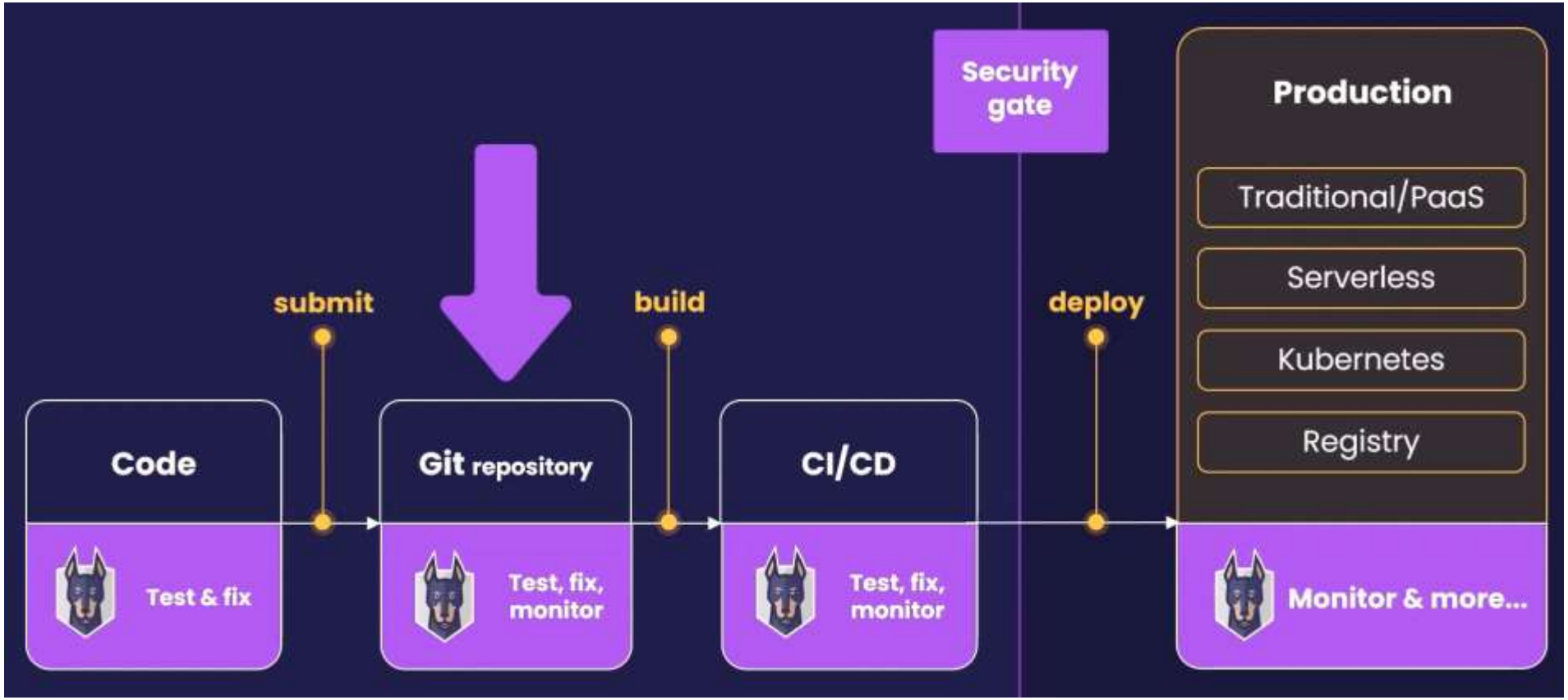
Git integration workflows



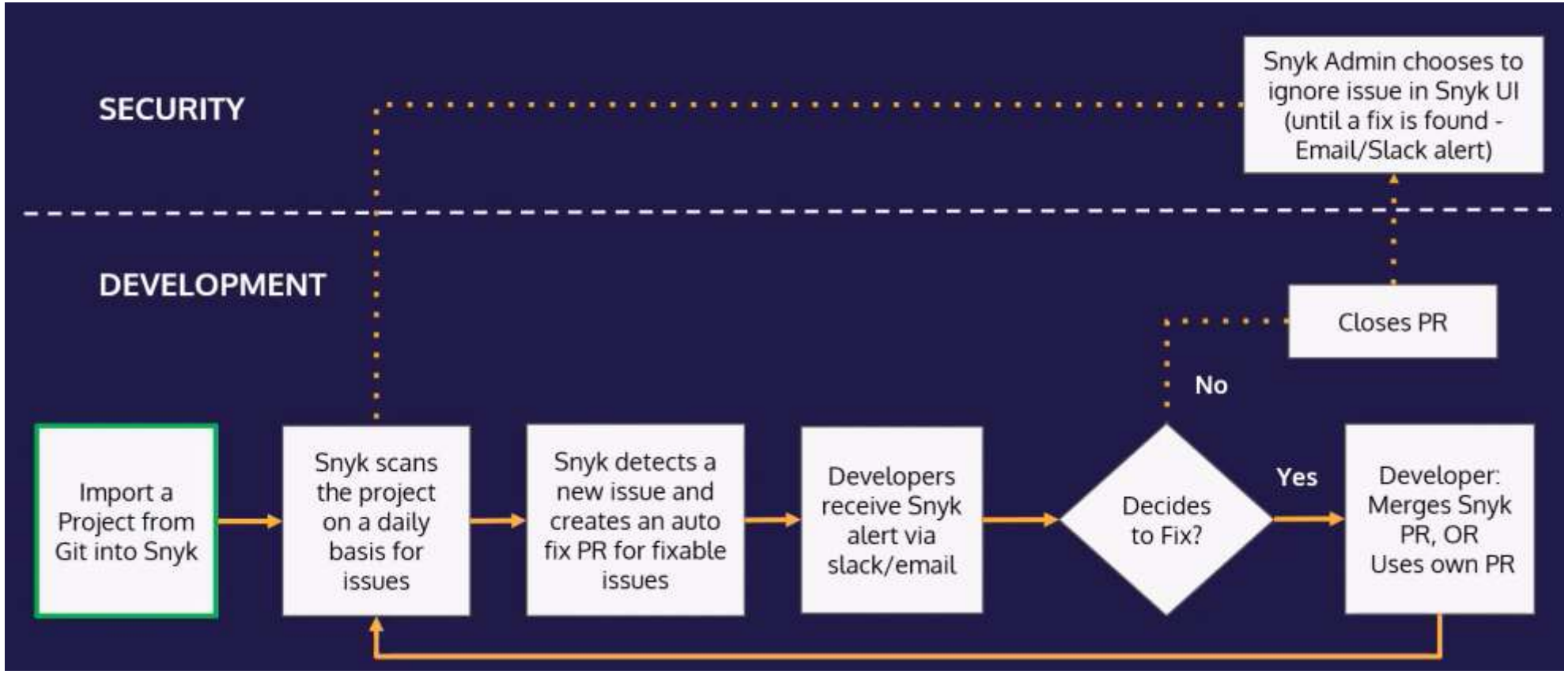
Default Snyk Test for Pull Request



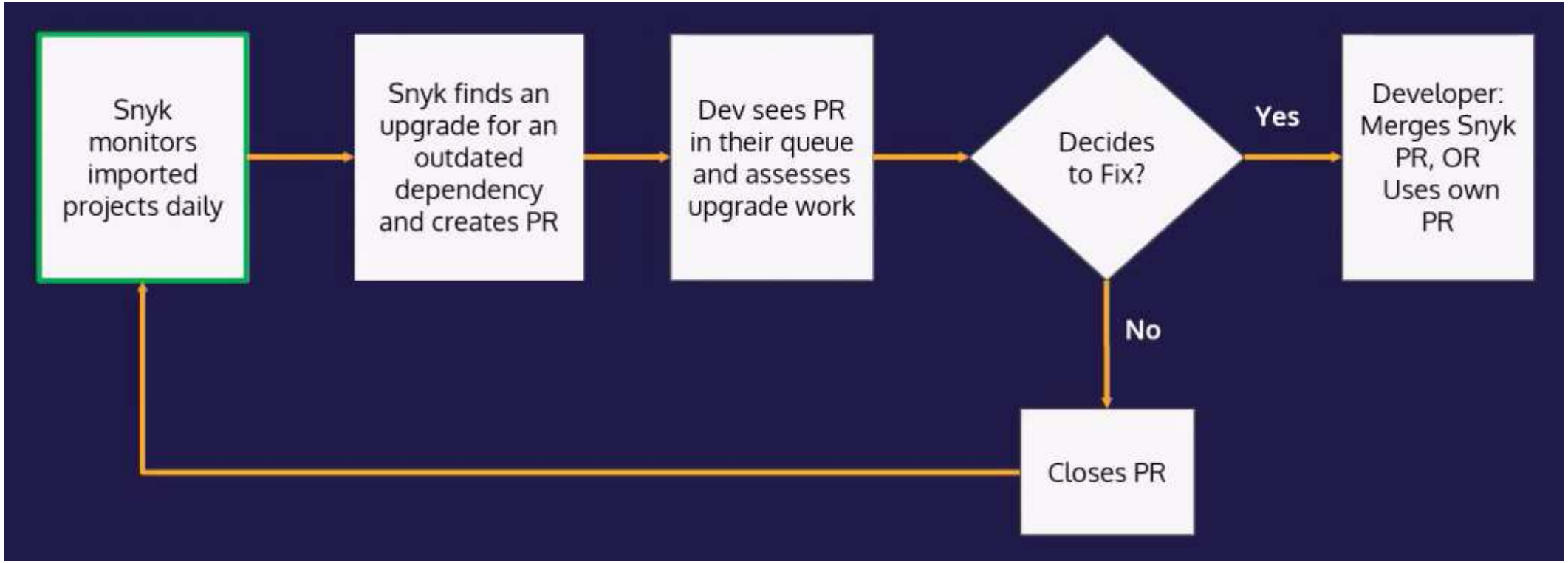
Automatic Snyk Test for Pull Request



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

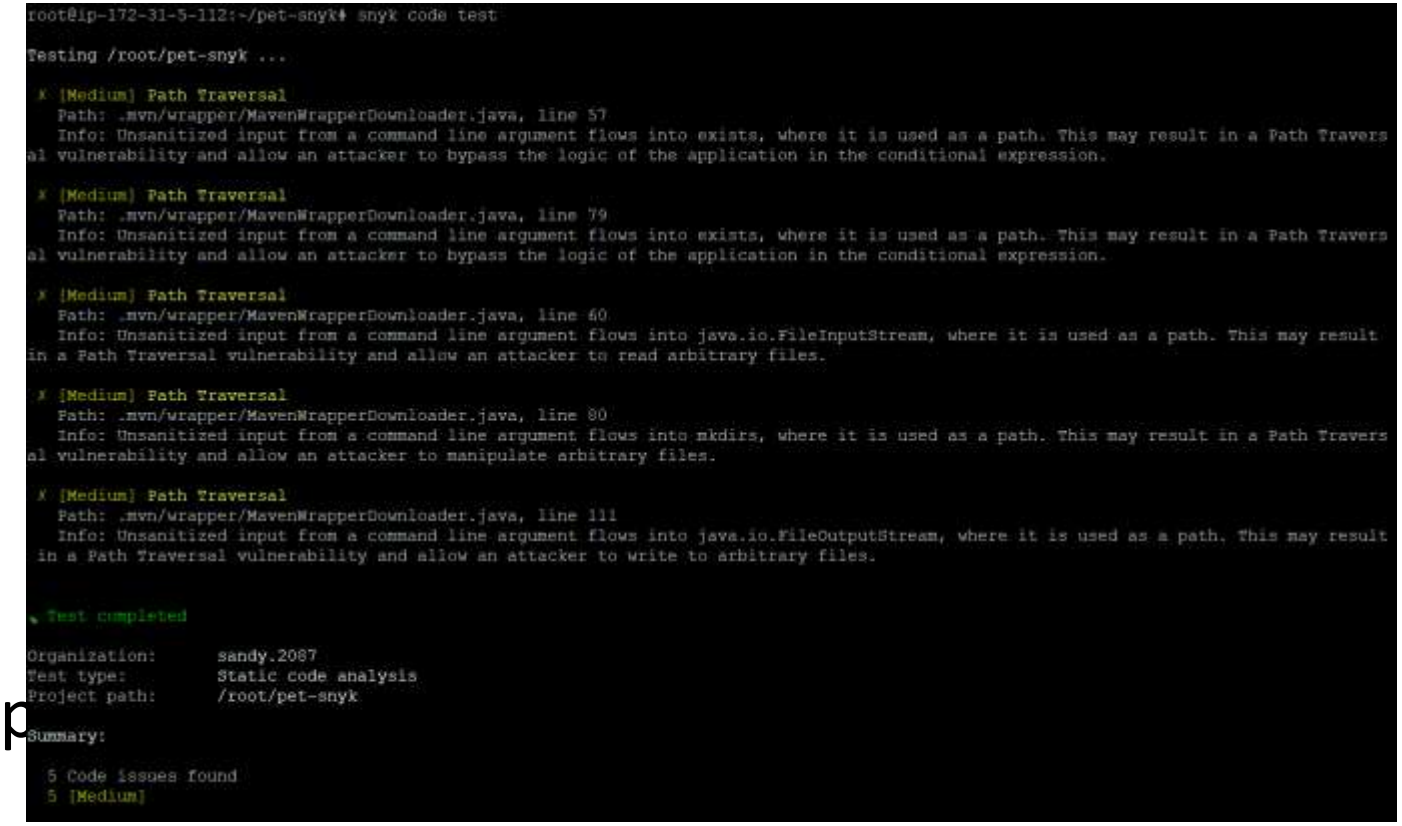
Snyk CLI Commands

- **Scan for security issues:**

Source code
To scan your **source code** for vulnerabilities
snyk code test --org=b1ecba5a-e67d-4bfa-8b27-4332218c99e4

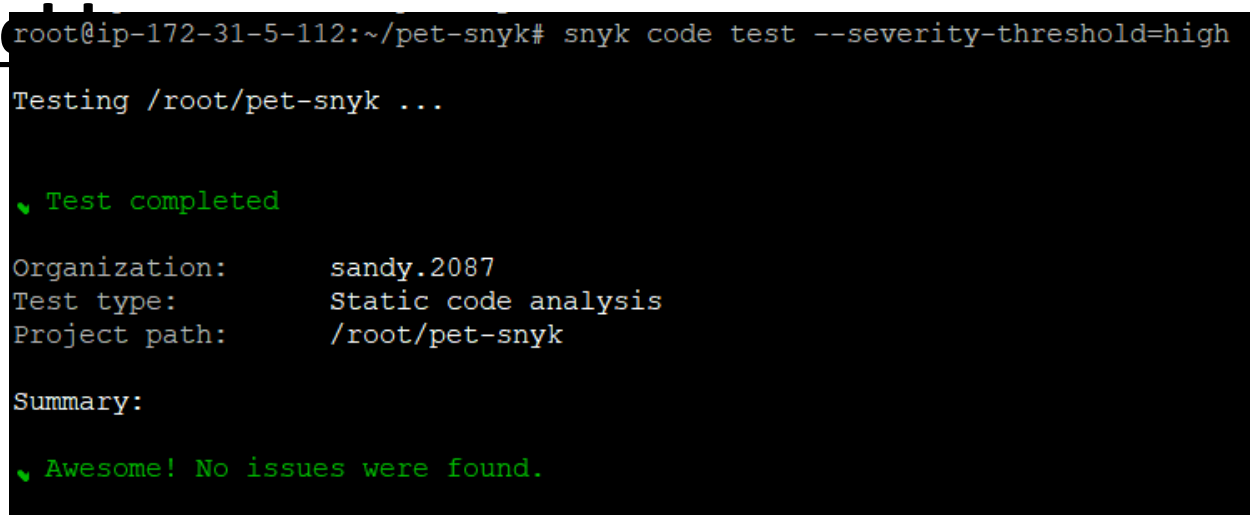
Snyk CLI Commands

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



Snyk CLI Commands

- **SAST scanning using Thresh**
- **Threshold:**
- High
- Medium
- Low
- **Steps:**
- 1. Clone the code
- 2. Run SAST scanning
- snyk code test --severity-threshold=high



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-filt

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, upgradleable 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. Synk 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...
✗ 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)

✗ 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

✗ 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 more accessible reports from your vulnerability scans.

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	--actionalable-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

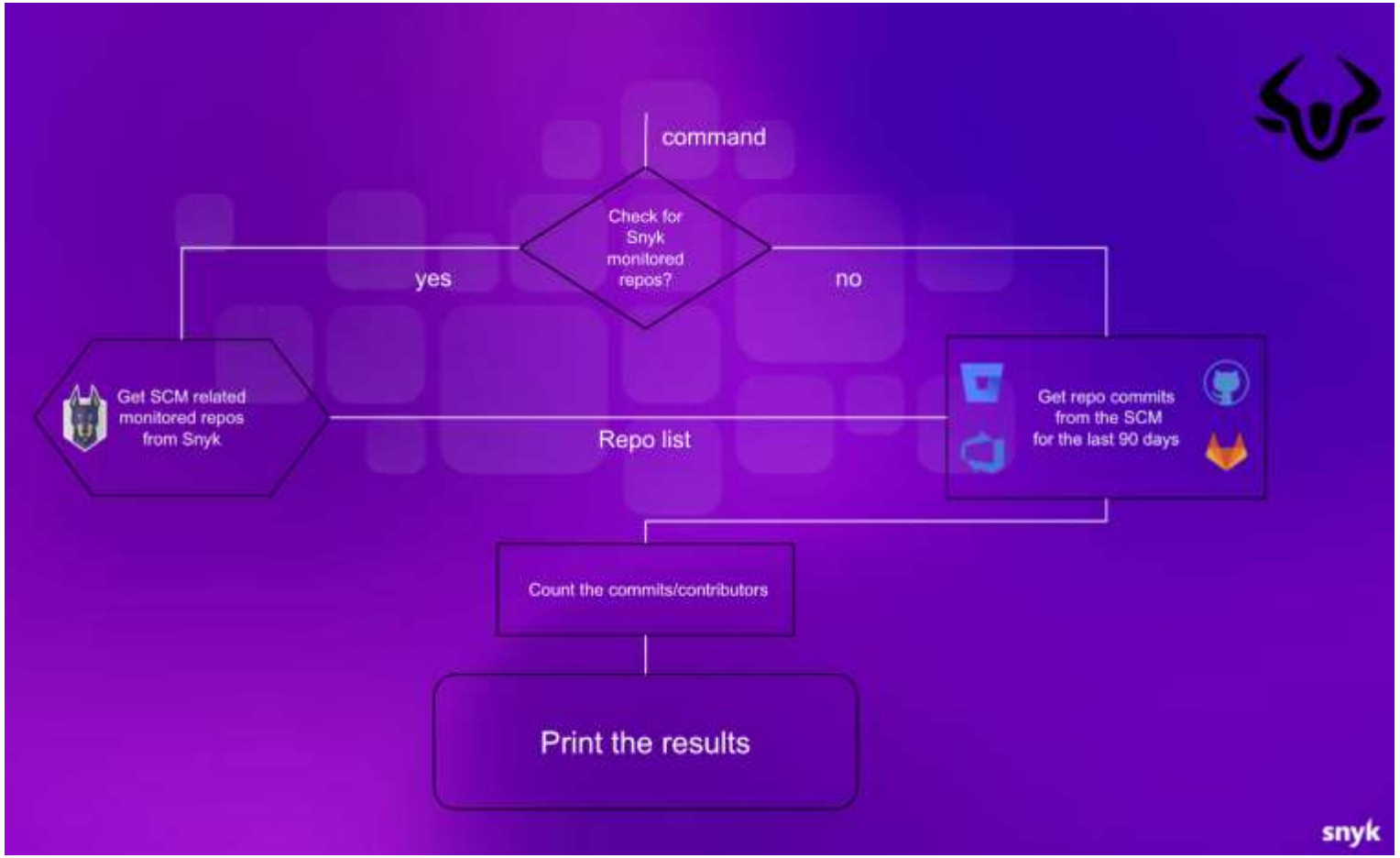
Snyk CLI Tools: snyk-scm-contributors-count

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

- **<u>Testing Multi-Platform Images with Snyk:</u>**
- 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	snyk iac test
Test for specific files	snyk iac test file-1.tf dir/file-2.tf
Test for issues in a specific directory	snyk iac test my-folder
Limit directory traversal depth	snyk iac test --detection-depth=3
Output results in JSON format	snyk iac test --json
Output results in SARIF format	Output results in SARIF format
Save SARIF output to a file	snyk iac test main.tf --sarif-file-output=snyk.sarif
Display issues above a specific severity level	snyk iac test --severity-threshold=medium
Target a specific Snyk Organization	snyk iac test --org=infrastructure
Set default Organization in Snyk config	snyk config set org=infrastructure

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.