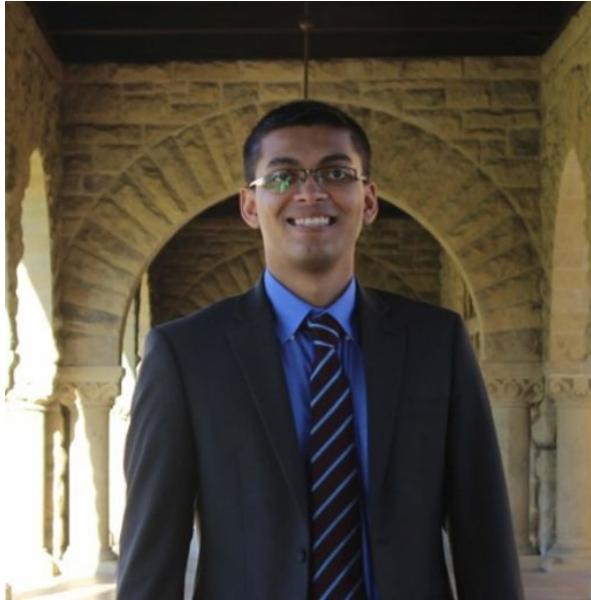




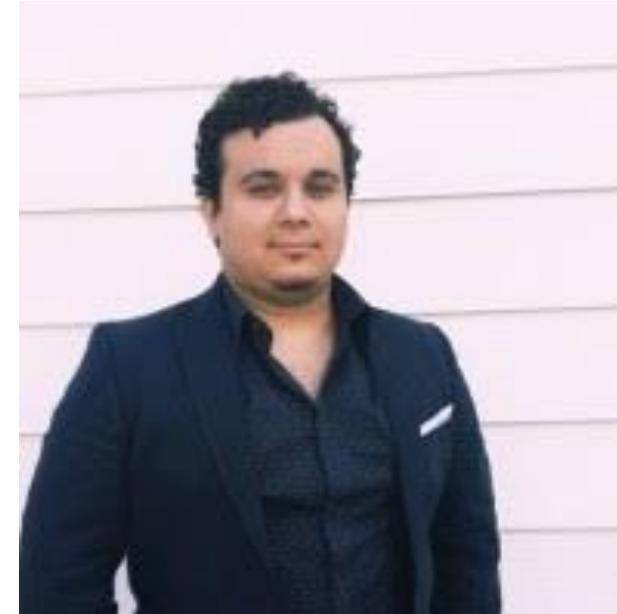
Introduction to DevSecOps with JFrog Xray

Maharshi Patel, Solutions Engineer, JFrog

JFROG TEAM



Maharshi Patel
Solutions
Engineer



Eric Rosa
Solutions
Engineer

AGENDA

01:05-02:45

DevOps & DevSecOps Overview

Introduction To JFrog Xray

Policies and Watches

Impact Analysis

02:45-3:00

Break

03:00-04:50

Shift Left Strategy

CI / CD Integration

Scan Reports

New Features

04:45-05:05

Summary, Quiz, Survey & Q/A



CLASS LOGISTICS

- Coffee Break
- Restrooms
- Course Materials
- Hands-on labs on your computer
- Survey
- Quiz
- Q&A

CLASS PREREQUISITES

- Personal
 - Basic understanding of the Software Development Lifecycle (SDLC)
 - Basic understanding of JFROG Artifactory
- Labs
 - Since a significant part of the training involves hands-on exercises. So a laptop with a browser and IDE or Terminal client.
 - Git Client installed (<https://git-scm.com/downloads>)
 - Clone Github repository from <https://github.com/jfrog/SwampUp2022>
 - We will be using **SU003-Intro-to-DevSecOps-with-JFrog-Xray** for today
 - Install JFrog CLI (<https://jfrog.com/getcli/>)
 - Docker daemon + client (<https://docs.docker.com/get-docker>)
 - Maven client (<https://maven.apache.org/install.html>)
 - NPM client (<https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>)

LAB SETUP INFORMATION

- A hands-on lab environment has already been provisioned for you to use in this class, based on your Last Name and First Initial:
- Login Information:
 - URL: <https://sup009epsuYY.jfrog.io/>
Username: fabienlsup009epsuYY@jfrog.com
Password: SwampUp2022!
 - YY = Student number, for example 03 or 05 or 32.
- Each user in the class will be given a student number.

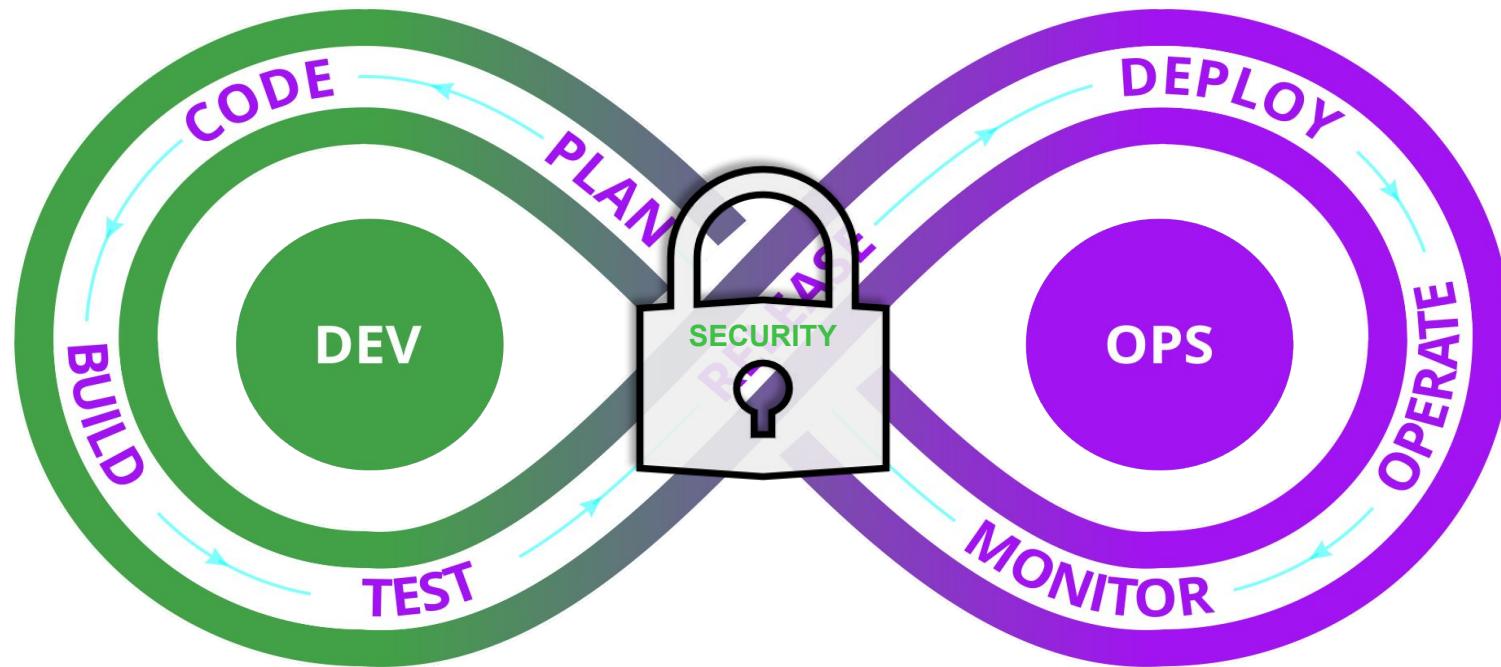


DevSecOps



The Liquid Software Company

WHAT IS DEVOPS?



The philosophy of integrating security practices within the DevOps process.
#SecurityFirst culture!

How? Introducing security earlier in the life cycle of application development

WHY DEVSECOPS?

Developer Options:

- Write from scratch
- Leverage OSS/3rd party libraries



RISKS with OSS:

- Security vulnerabilities
- License compliance issues
- Performance issues

Example:

- [Equifax](#) - usage of Apache Struts
- [Cisco](#) - usage of GPL license (General Public License)
- [Log4Shell](#)
- [Spring4Shell](#)
- [many more...](#)

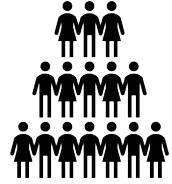


JFROG



The Liquid Software Company

JFROG AT A GLANCE



1,000+
EMPLOYEES



10
OFFICES
WORLDWIDE



3M+ ENTERPRISE
DEVELOPERS
DAILY



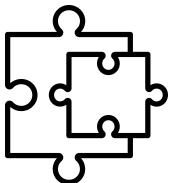
85%
OF THE
FORTUNE 100⁽¹⁾



~7,000
CUSTOMERS
GLOBALLY



>300
COMMUNITY
EVENTS ANNUALLY



DEVELOPER ECOSYSTEM
JAVA/ JAVASCRIPT, C++, .NET,
VISUAL STUDIO, PYTHON ...



\$207M
ANNUAL
REVENUE



THE JFROG APPROACH

END-TO-END DEVOPS PLATFORM

Unify, Accelerate & Secure Your Software Delivery, From Development To Distribution



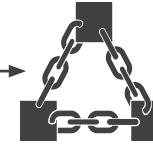
ULTIMATE SCALABILITY

Flexible, Open And Expandable By Design, That Grows With Your Scaling Needs



CONTINUOUS SECURITY

Integrated Security Across Your Pipeline to Achieve Security and Compliance at DevOps Speed



CLOUD, MULTI-CLOUD & HYBRID

Deployment Flexibility Supports Any Development Topology Including Cloud Native



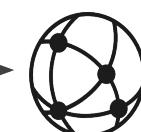
UNIVERSAL TECHNOLOGY

Supports 30+ Package Types and Any Tool in Your DevOps Ecosystem



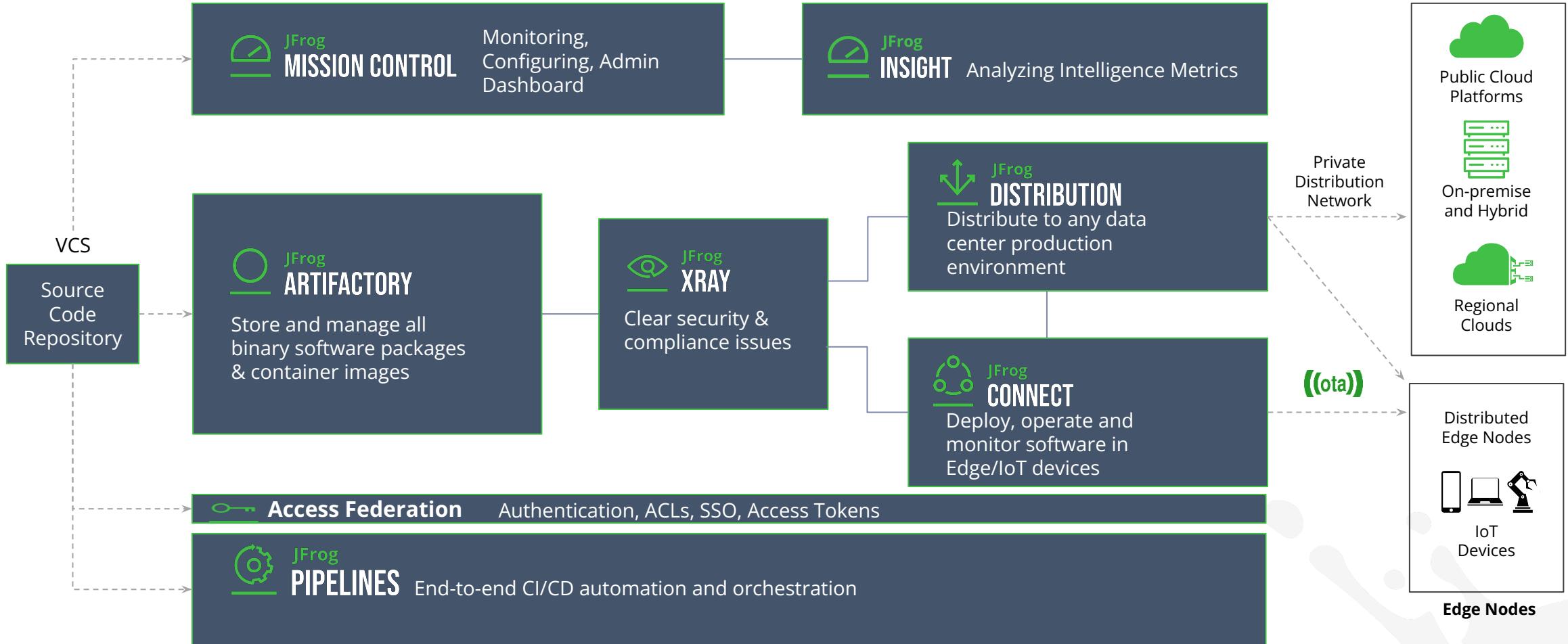
INTEGRATED ECOSYSTEM

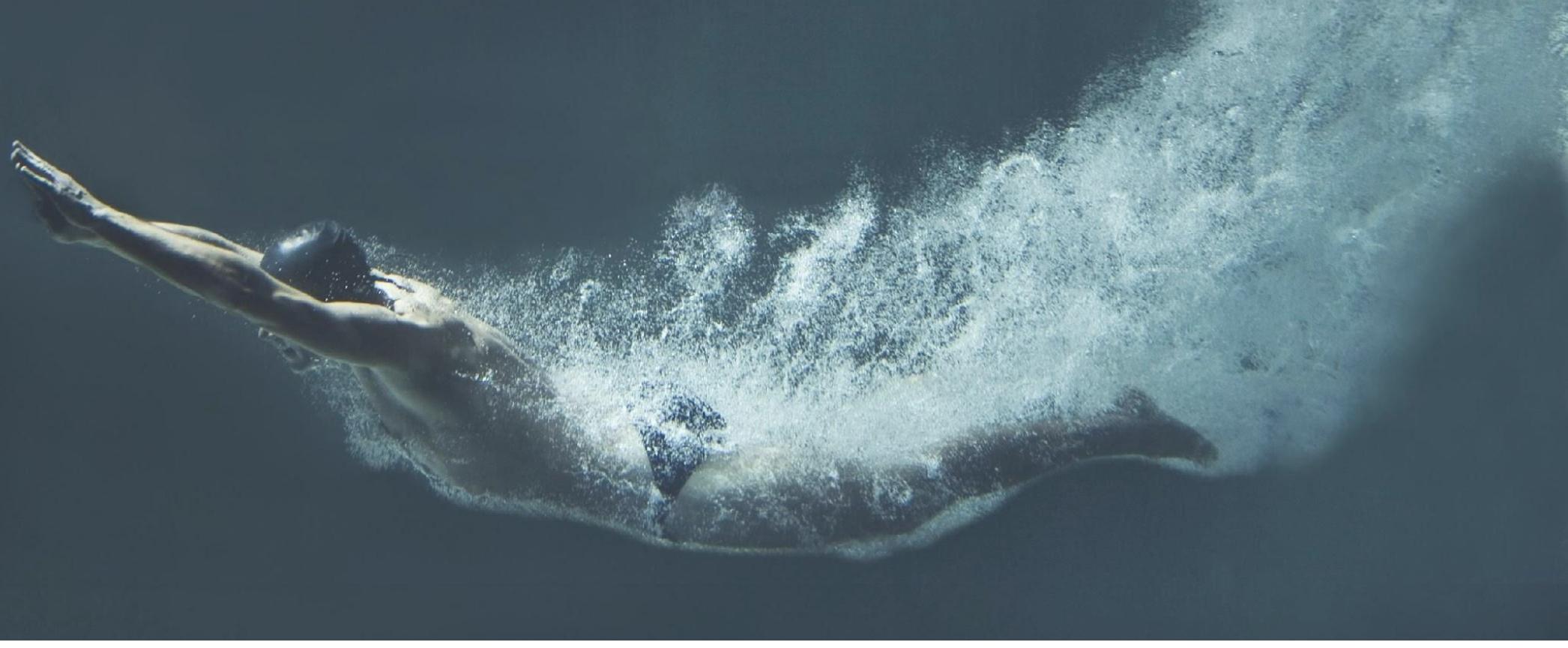
Extensive Technology, ISV and Cloud Partner Integrations, Out-of-the-Box



THE JFROG DEVOPS PLATFORM

Unify, accelerate & secure your software delivery, from development to distribution





JFROG XRAY

Universal Software Composition Analysis
(SCA)

JFROG XRAY | AUTOMATED SECURITY & COMPLIANCE

SCAN ANY SOFTWARE

Many supported package types
From single binaries to applications
Containers



FIX WHAT MATTERS, FAST

Impact analysis



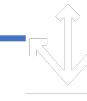
INTEGRATE IN YOUR ENVIRONMENT

Rich integrations, REST API, CLI
Automated security in CI/CD
From development to runtime



GAIN FULL VISIBILITY

Supply chain full visibility / SBOM
Vulnerabilities in all dependencies



CONTROL YOUR SECURITY POSTURE

Context-based policy
Platform management & security



DEPLOY ANYWHERE

Self-hosted (on-prem / cloud)
SaaS
Multi-Cloud
Hybrid



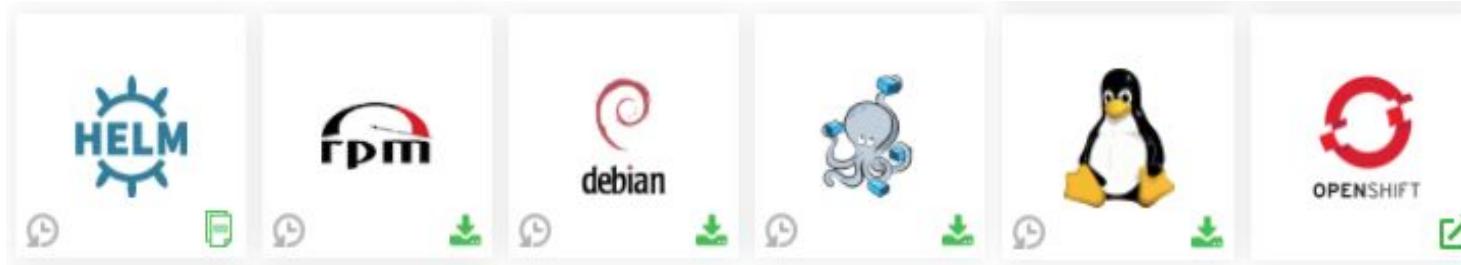
XRAY SUPPORTED PACKAGE FORMATS



Google Distroless Images

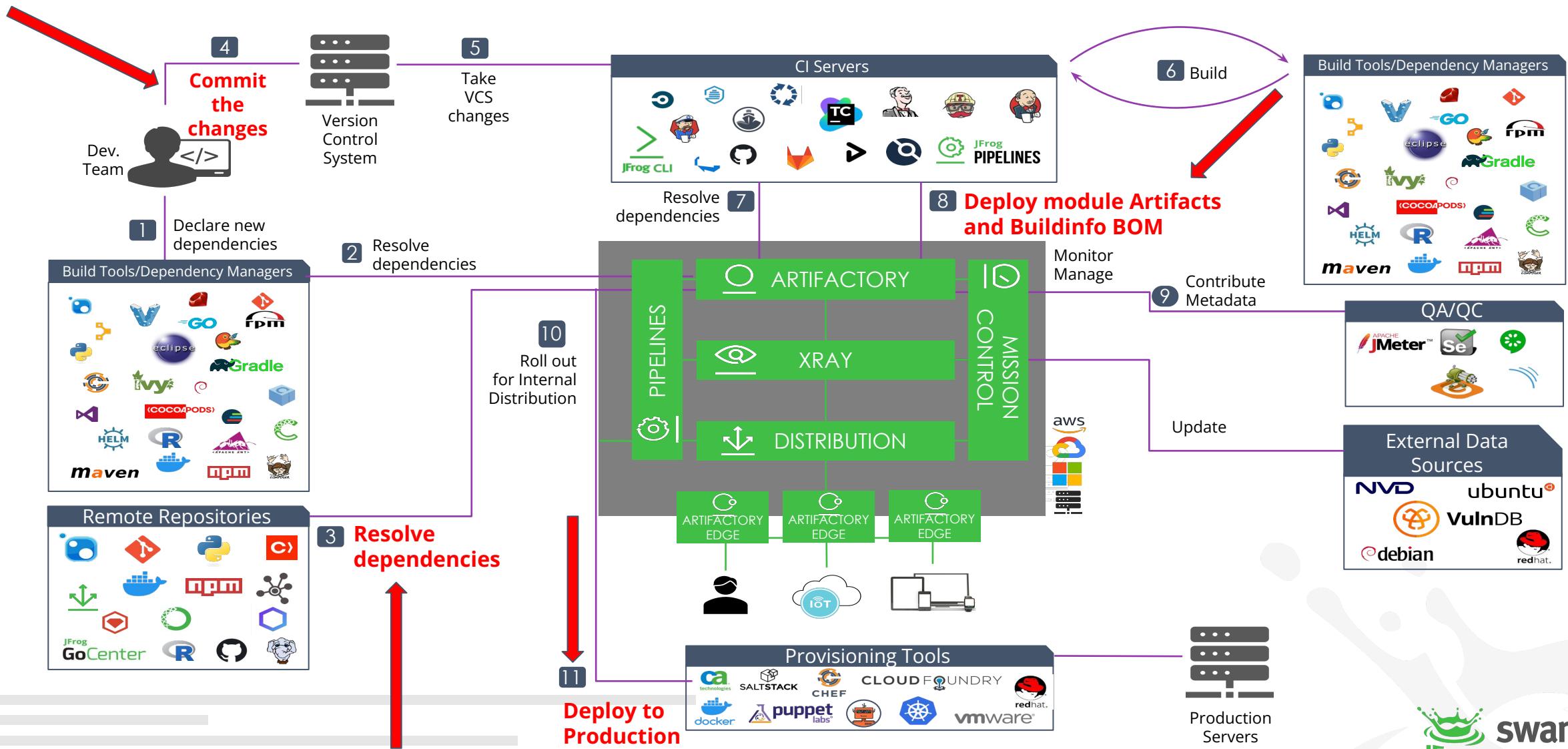
XRAY SETUP

- [Download JFrog Xray](#)



NOTE: Applicable for Self-Managed only

WHERE DOES XRAY FIT?





XRAY ARCHITECTURE



The Liquid Software Company

JFROG XRAY TERMINOLOGY

INDEXING

The process of Analyzing the Artifact and listing the components and Artifact layers

RESOURCES

Repositories, Builds and Release bundles

POLICIES

Contextless set of rules with set of automatic actions

RULES

Consists of Security, Licensing and Operation Risks criteria with corresponding Automatic actions

WATCHES

Define the scope of resources and enforce the policies assigned to them

JFROG PLATFORM RESOURCES

Repositories

payment-mvn-dev-local

payment-mvn-test-local

payment-mvn-prod-local

dashboard-npm-dev-local

dashboard-npm-stage-local

dashboard-npm-prod-local

npm-remote

maven-remote

helm-local

Builds

npm-build

mvn-build

Release Bundles

my-rb

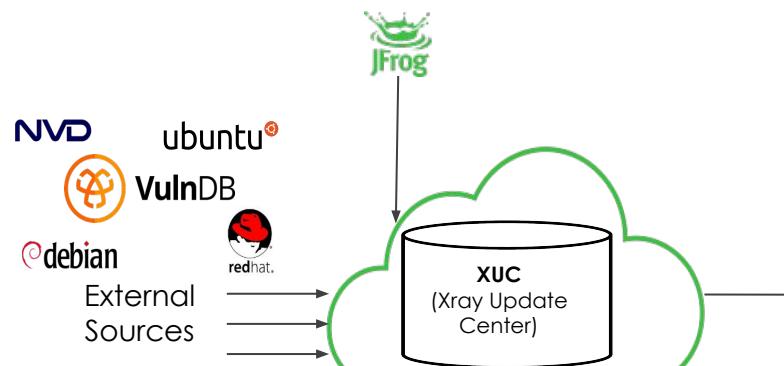
npm-build

mvn-build

helm-chart-1

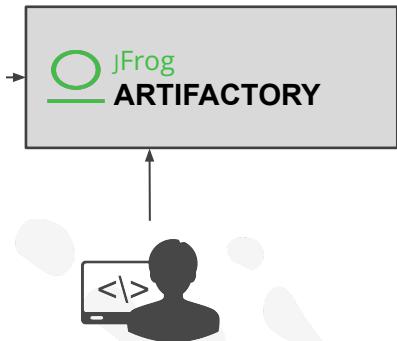
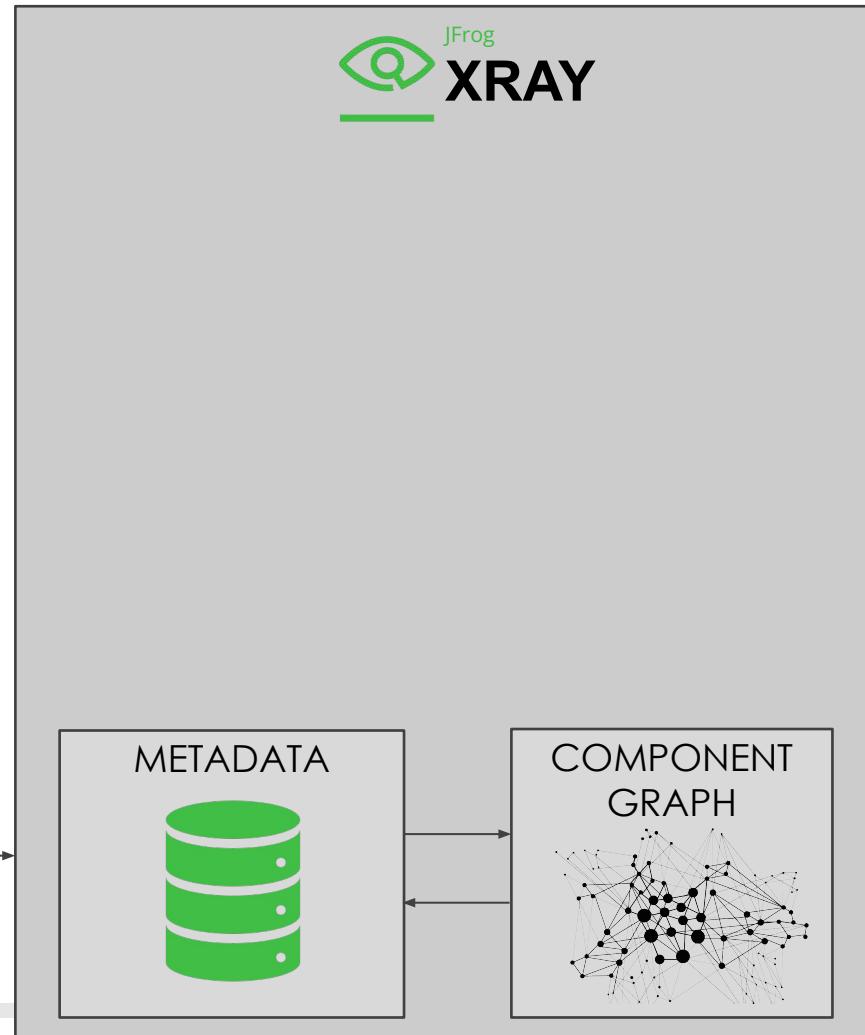
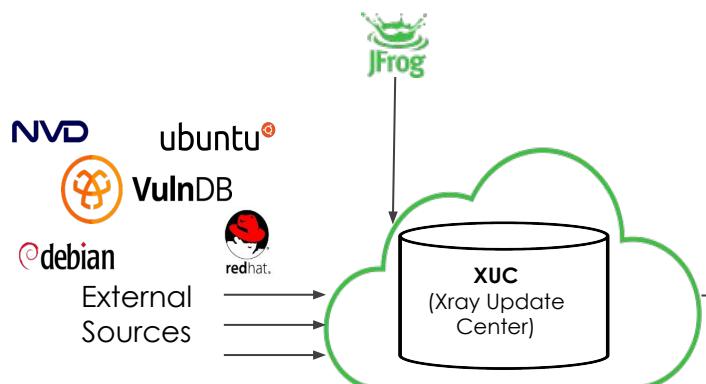
XRAY OVERVIEW

JFrog Authorized as a CNA
(CVE numbering Authorities)



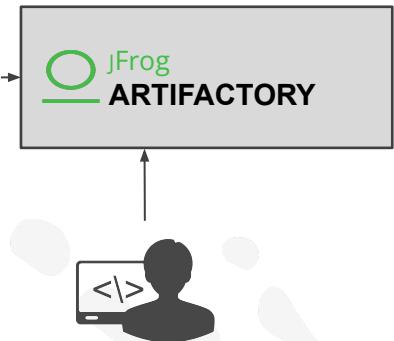
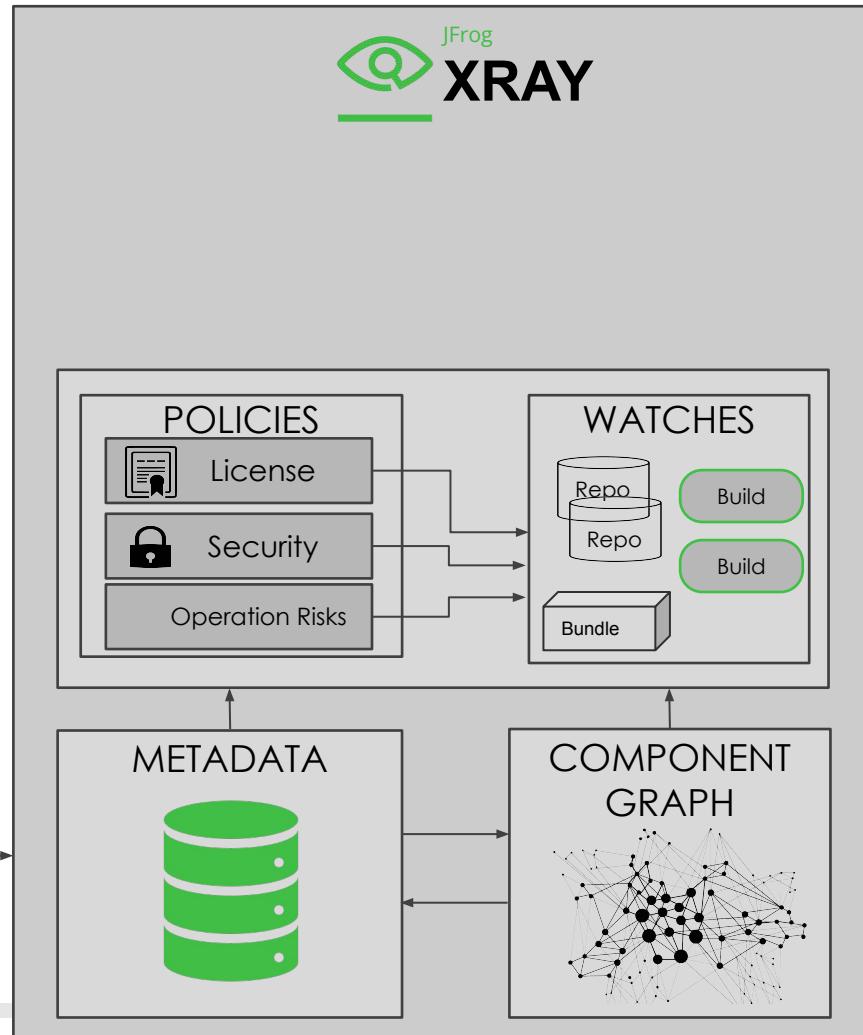
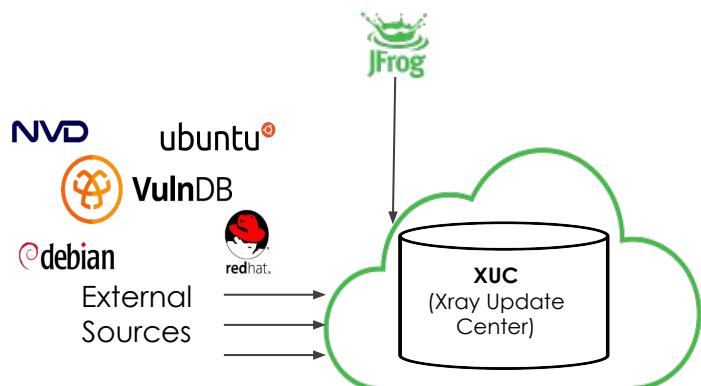
XRAY OVERVIEW

JFrog Authorized as a CNA
(CVE numbering Authorities)



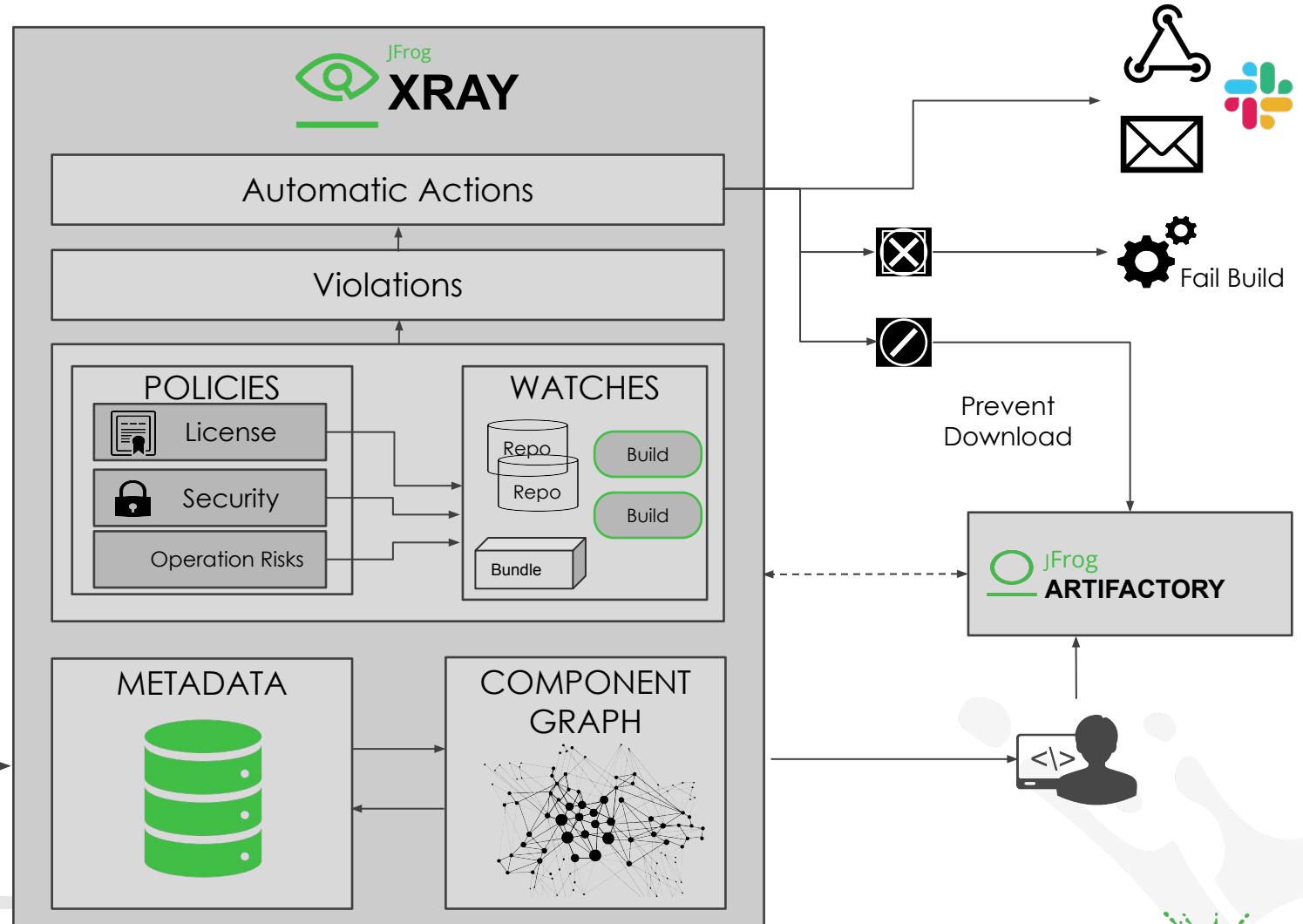
XRAY OVERVIEW

[JFrog Authorized as a CNA
\(CVE numbering Authorities\)](#)

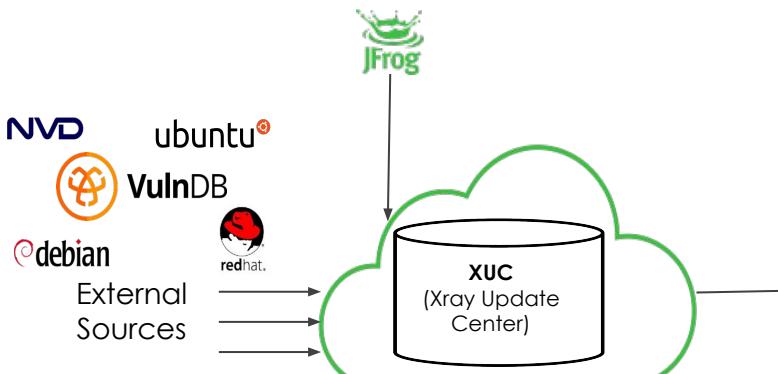


XRAY OVERVIEW

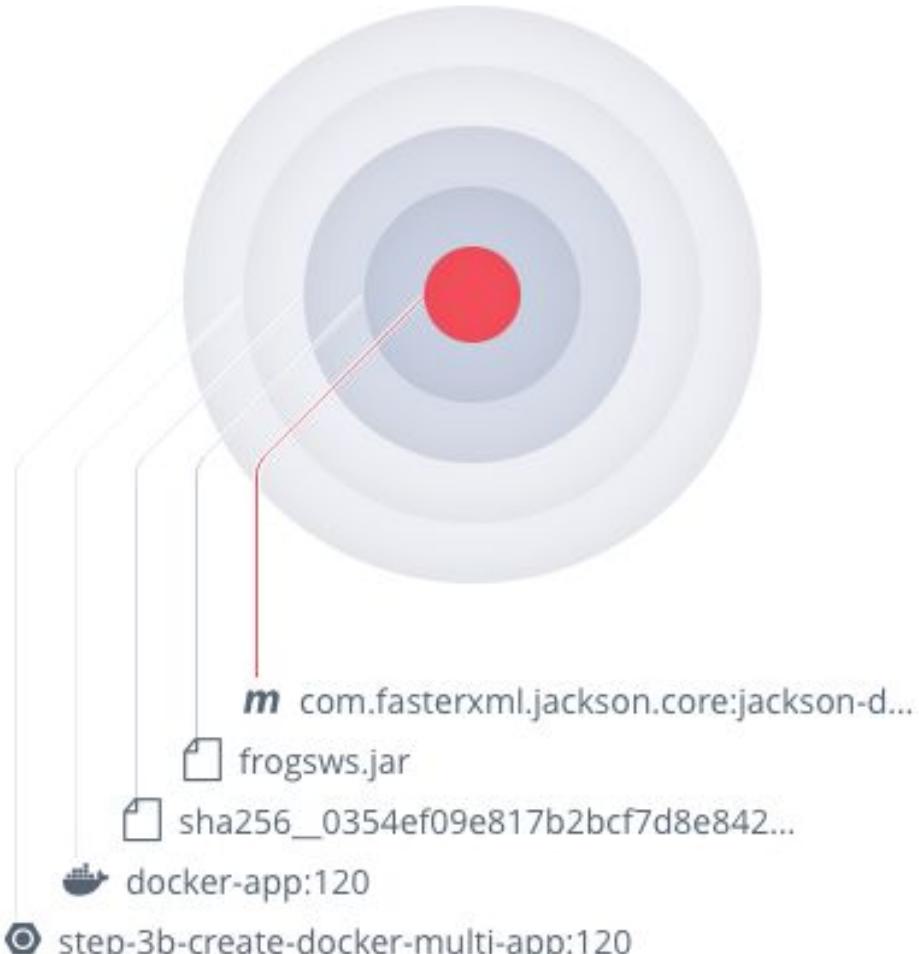
Web Hooks, Slack, Emails



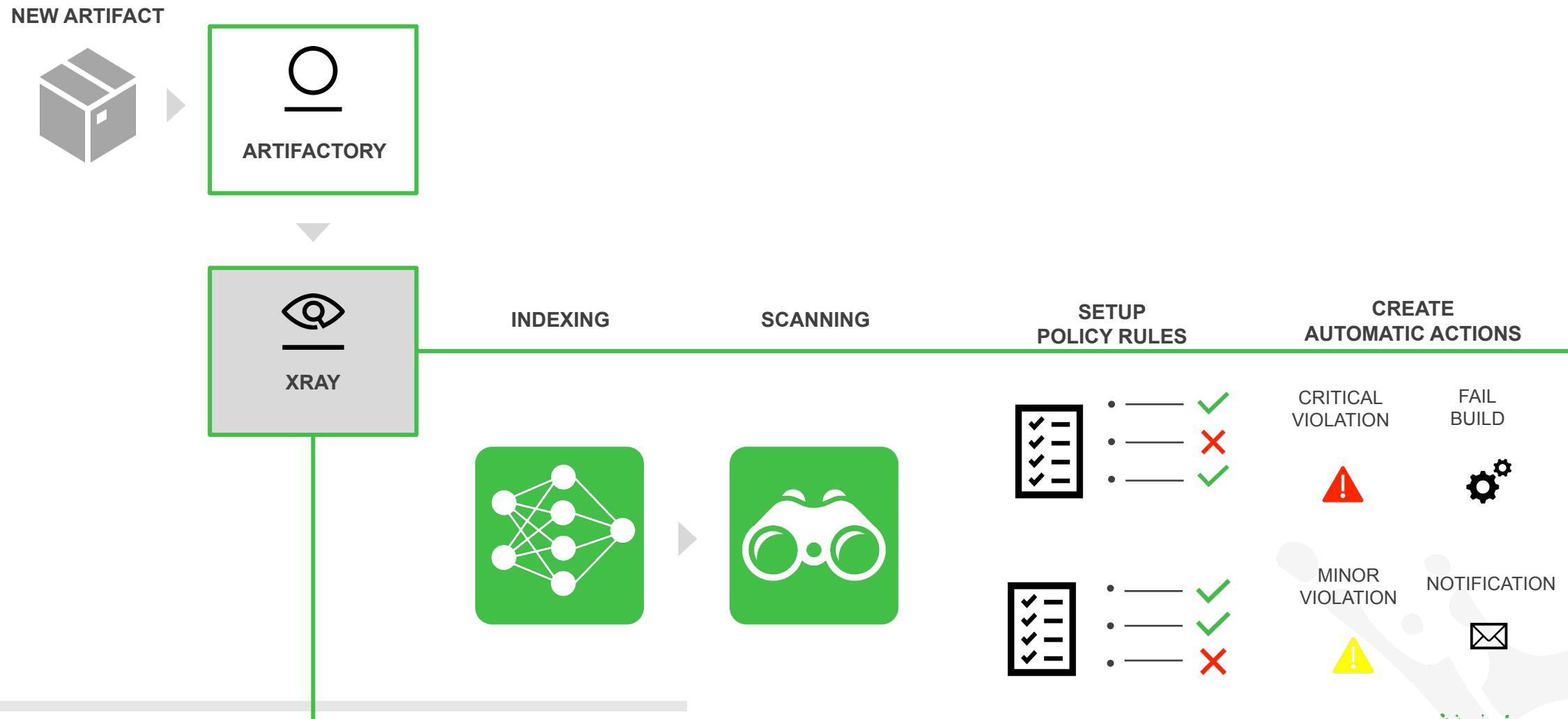
[JFrog Authorized as a CNA](#)
(CVE numbering Authorities)



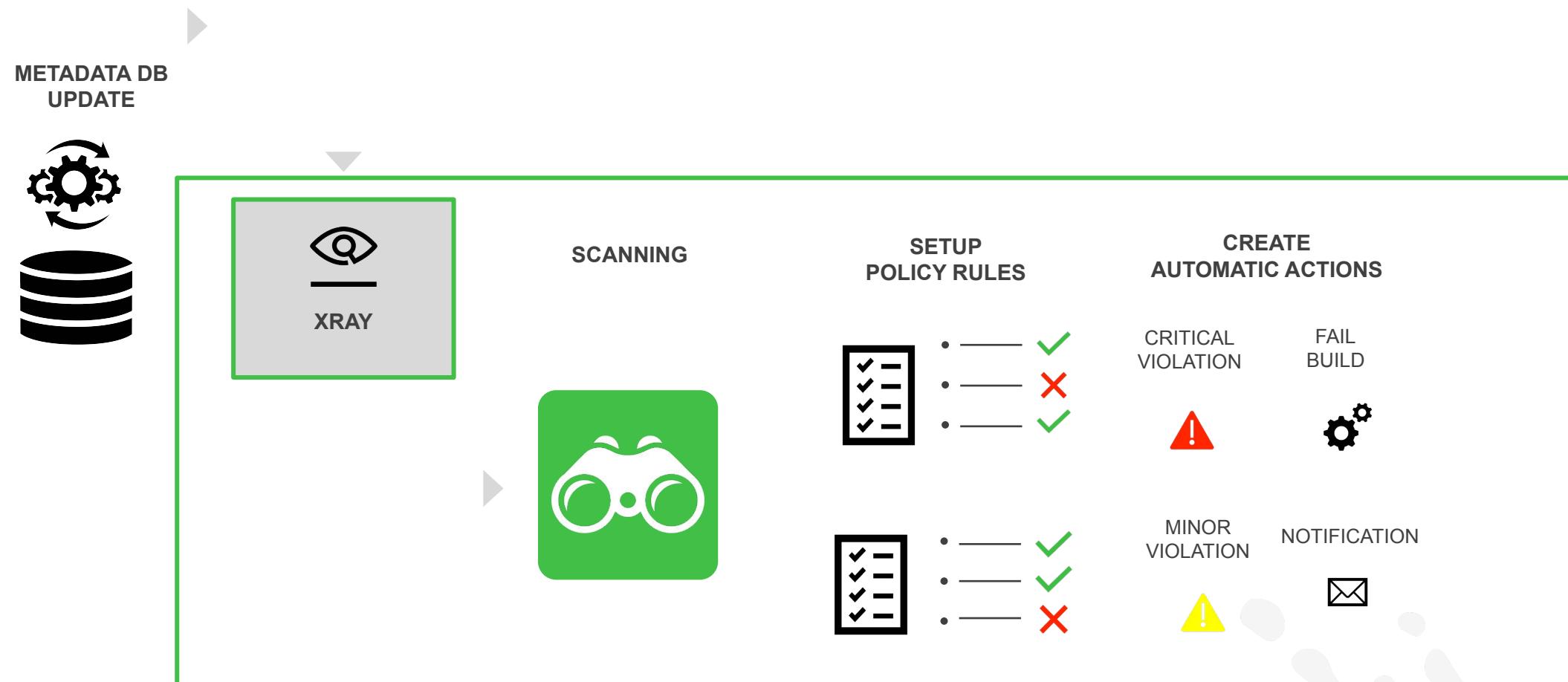
DEEP RECURSIVE SCAN



CONTINUOUS ANALYSIS ON NEW ARTIFACTS

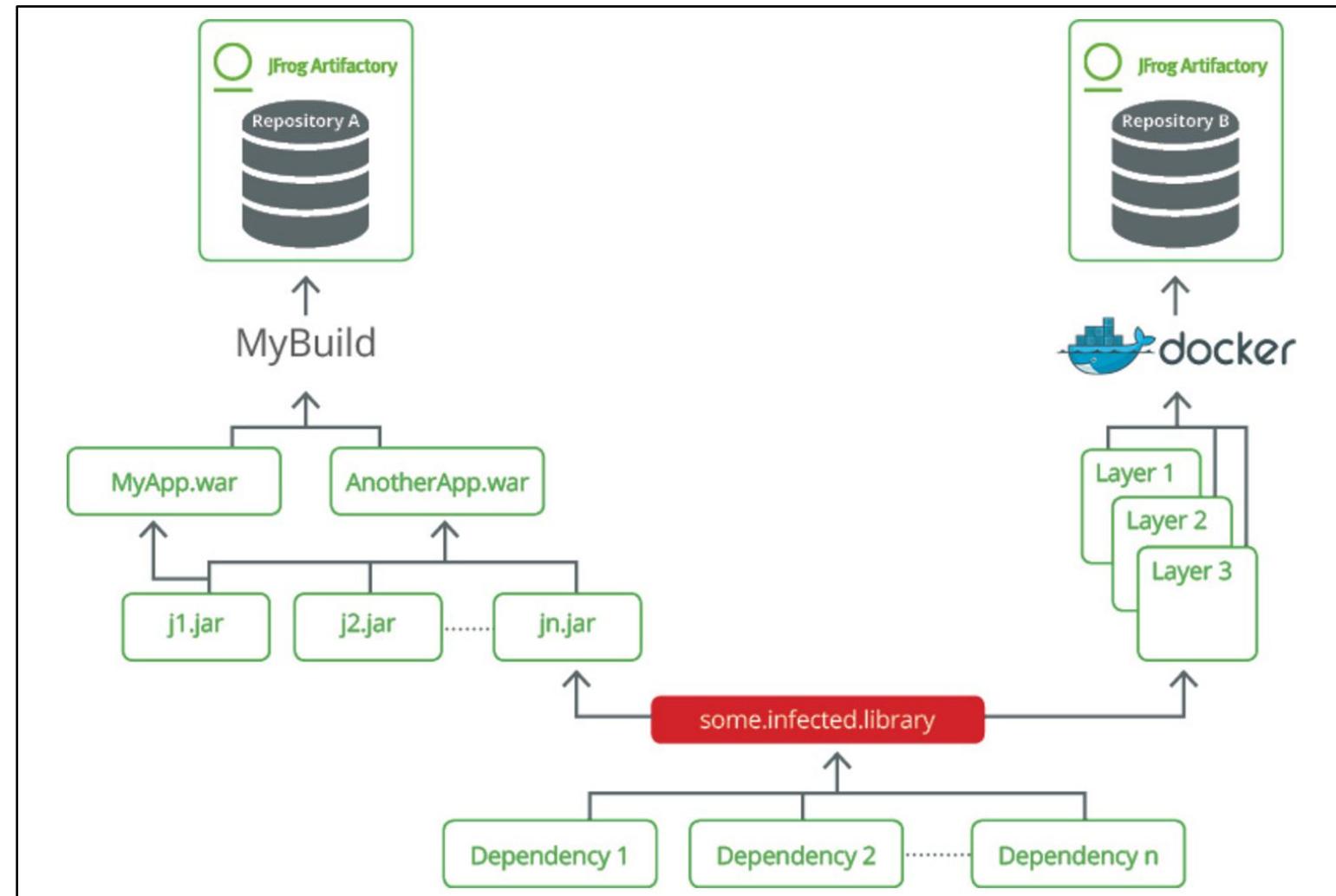


CONTINUOUS ANALYSIS ON METADATA UPDATE



IMPACT ANALYSIS

- Relationships between all the components
- Once a component has a vulnerability, Xray understands the impact it has on other components
- Xray displays a graph showing the relationship between the vulnerable component and all others that are affected



DEMO

POLICY

- A policy is :
 - contextless, which means that it only defines what to enforce and not what to enforce it on
 - composed of at least 1 rule
- Each rule is defined by :
 - a license/security/operational risk criteria
 - a set of automatic actions according to your needs
 - Generate Violations, Notify by Email, Notify Deployer, Notify Watch Recipients, Create JIRA ticket, Trigger Webhook, Block Download, Fail Build

The Rule priority list sets the order in which the rules are processed.

Once a rule is met, the subsequent rules in the list will not be applied.



XRAY POLICIES

NEW

Security Policy

Rule 1

Min Severity Level

Automatic Actions

Rule 2

Min Severity Level

Automatic Actions

License Policy

Rule 1

Allow/ Banned licenses

Automatic Actions

Rule 2

Allow/ Banned licenses

Automatic Actions

Operational Risk Policy

Rule 1

EOL/ Version Age/ Num of new version

Automatic Actions

Rule 2

EOL/ Version Age/ Num of new version

Automatic Actions

COMPONENTS OPERATIONAL RISK

- Makes you aware of the risks (EOL/Version Age/Number of New Versions/Health of Open Source Project) of using outdated or inactive OSS components in your project.
- Tracks the following:
 - EOL: OSS components in use is obsolete or declared EOL by author.
 - Version Age: Age of OSS component's age in use in months.
 - Number of new versions: Number of new versions released after current version
 - Health of OSS projects: boolean value, depending on the following:
 - Release cadence/year
 - Number of commits/year
 - Number of committers/year

ORGANIZE POLICIES

- Configure at
 - Global level
 - Project level
- Generally managed by the security team / project admin
 - requires the “Manage Policies” roles set at the user or group level
 - provides create / update / delete actions on ALL policies !!
- Global to the organization ⇔ can be referenced in any watch
- For dedicated policies per project, different solutions :
 - grant the “Manage Policies” to the watch owners
 - automate the policies creation process using Xray Rest API

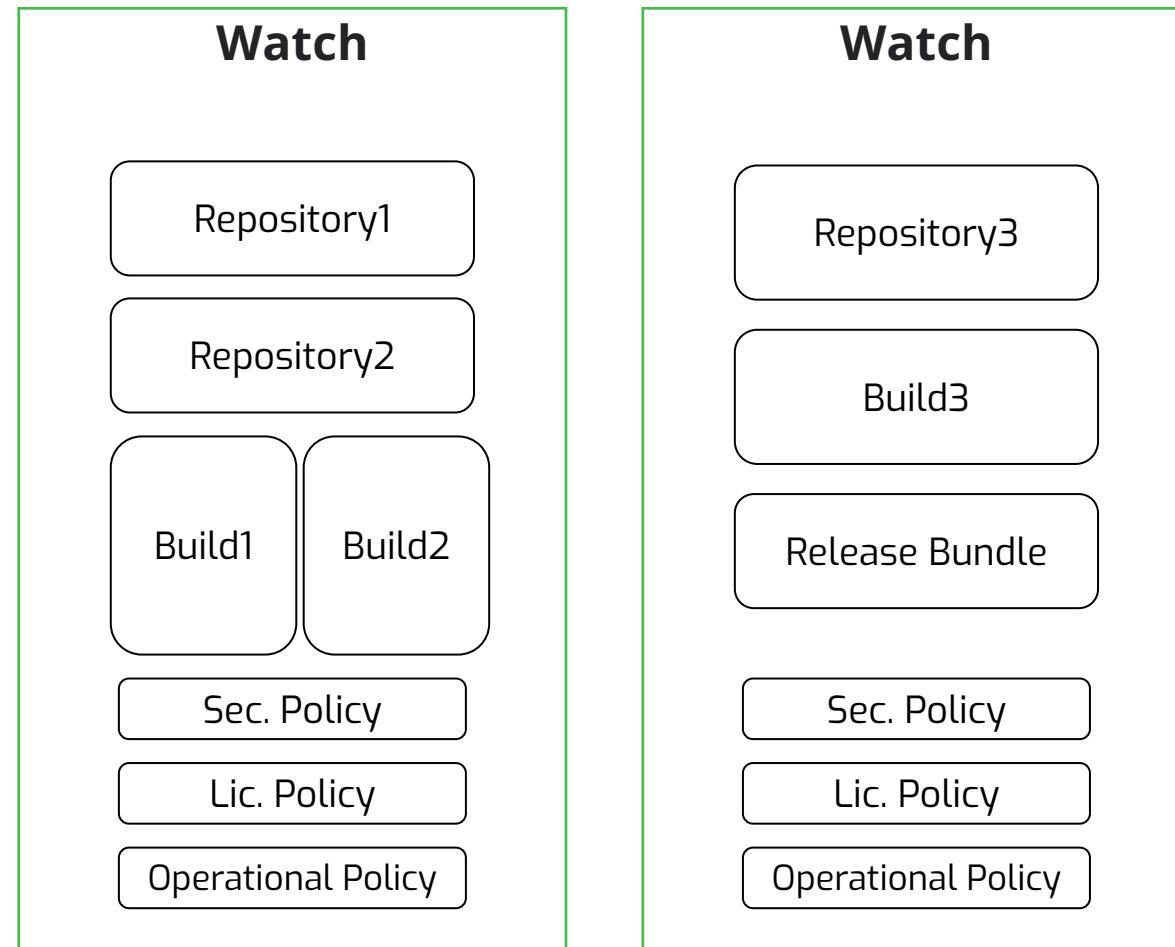
LAB 1

[SwampUp2022/SUP003-Intro to DevSecOps with JFrog Xray/lab-1](#)

WATCHES

- Define the **scope of the resources** you want to watch - repositories, builds, release bundles.
- Enforce the policies assigned to them on the artifacts by **generating violations**.
- **Filters** in a watch determine on which binaries the policies will be applied.
- Even if a rule in one policy is applied, the other policies will be applied as well. Only inside a policy we stop at the first rule.

XRAY WATCHES



ORGANIZE WATCHES

- Configure at
 - Global level
 - Project level
- Managed by a group of “trusted” people such as Release managers, Project managers, Technical leads, ...
 - requires the “Manage Watches” role set at the user or group level
 - provides create / update / delete actions on ALL watches !!

VIOLATION CONCEPT

- A violation is raised when a criteria, set in a license or security or operational risk policy rule, matches the license of a component and respectively the vulnerability level in the scanned artifact.
- A violation can be ignored temporary or permanently
 - the scope is tied to :
 - watch, build, component, vulnerability

DEMO

LAB 2

[SwampUp2022/SUP003-Intro to DevSecOps with JFrog Xray/lab-2](#)

COFFEE BREAK



AGENDA

01:05-02:45

DevOps & DevSecOps Overview

Introduction To JFrog Xray

Policies and Watches

Impact Analysis

02:45-3:00

Break

03:00-04:50

Shift Left Strategy

CI / CD Integration

Scan Reports

New Features

04:45-05:05

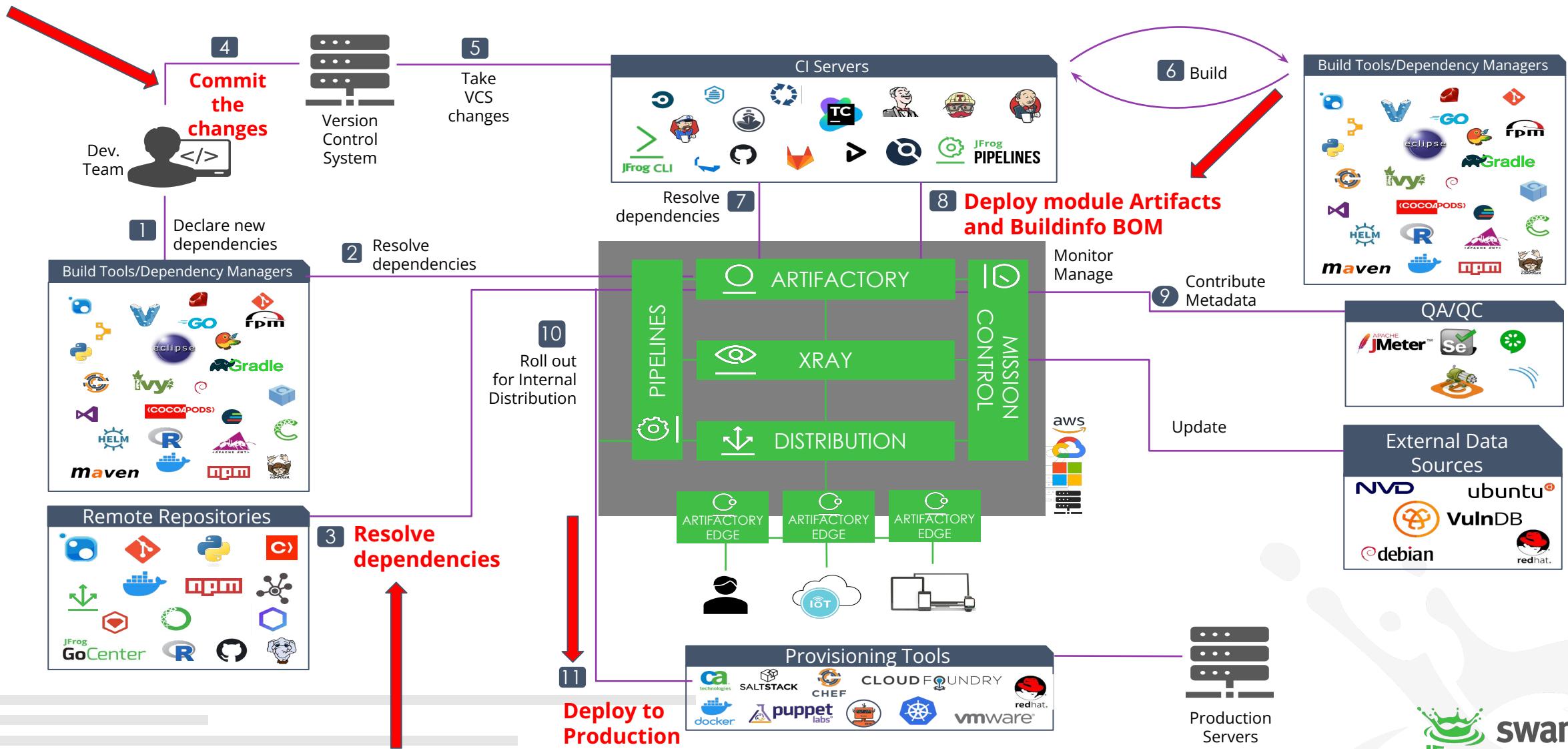
Summary, Quiz, Survey & Q/A



POLL

01:00

WHERE DOES XRAY FIT?



INTERACT WITH XRAY

- Shift Left Strategy
 - IDE Integration
 - Dependency Scanning
 - On Demand Binary Scan
 - Docker Desktop Extension
 - Frogbot
- JFrog CLI
- CI Plugins
- REST API
- GUI



IDE INTEGRATIONS



The Liquid Software Company

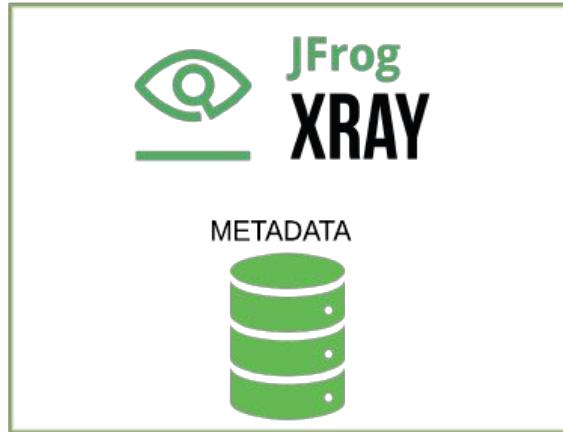
SHIFT LEFT STRATEGY AT IDE

- Cost of **remediating a vulnerability** is akin to the cost of fixing a bug
- Earlier you remediate a vulnerability in the release cycle, the **lower the cost**
- Bringing Xray's issue discovery one step earlier, to **development time**
- Flags components when vulnerabilities are discovered
- Developers will access the **component tree, vulnerability data and license details**

SECURING AT DEV (SHIFT LEFT STRATEGY)



Daily updates Sync



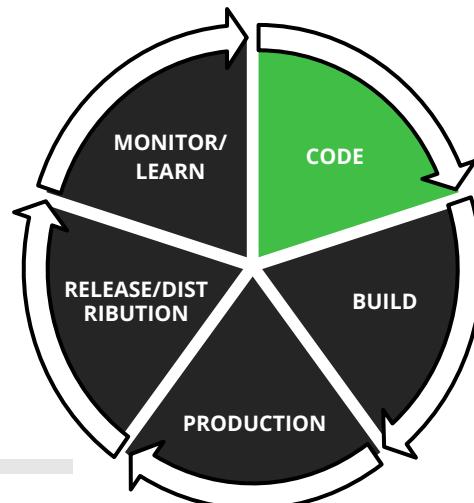
Update
Security &
License data



Visual Studio
& VSCode



Eclipse



SUPPORTED TECHNOLOGIES BY IDE PLATFORMS

Project Dependencies	Visual Studio Code	IntelliJ	WebStorm	PyCharm	GoLand	Android Studio	Visual Studio	Eclipse	Theia
Maven	✓	✓						✓	✓
Gradle		✓				✓		✓	
Python	✓	✓		✓					✓
Go	✓	✓			✓				✓
NPM	✓	✓	✓					✓	✓
NuGet	✓						✓		✓



DEPENDENCY SCANNING

- Scan your source dependencies for security or license violations using JFrog CLI
- Scan can be done **before a developer checks-in the code.**
- Run on source directory, ANYWHERE/ANYTIME, without the need to compile, test or deploy to Artifactory.
- **Supported** for Maven, Gradle, NPM, Pip, Pipenv, GO, NuGet, .NET Core packages, yarn2
- JFrog Cli 2.40 and above for [Dependency Scanning](#).

- **jf audit**

or

- **jf aud**

NOTE:

--watches - A comma separated list of Xray watches, to enable Xray to determine violations accordingly.

--licenses - List of licenses to be displayed if it's set to true.

--format - Defines the output format. Table or JSON

--repo-path - Artifactory repository path to enable Xray to determine violations accordingly.

[**Command Options**](#)



DEPENDENCY SCANNING

Security Violations

SEVERITY	IMPACTED PACKAGE	IMPACTED PACKAGE VERSION	TYPE	FIXED VERSIONS	COMPONENT	COMPONENT VERSION	CVE	CVSS V2	CVSS V3	ISSUE ID
Critical	commons-io:commons-io	2.6	Maven	[2.7]	commons-io:commons-io	2.6		5.0		XRAY-78200
High	commons-io:commons-io	2.6	Maven	[2.8.0]	commons-io:commons-io	2.6		7.1	7.5	XRAY-125253
Medium	org.apache.httpcomponents:HttpClient	4.5.6	Maven	[4.5.13]	org.apache.httpcomponents:HttpClient	4.5.6	CVE-2020-13956	5.0	5.3	XRAY-129495
Medium	commons-io:commons-io	2.6	Maven	[2.7]	commons-io:commons-io	2.6	CVE-2021-29425	5.0	5.3	XRAY-172728
Medium	commons-codec:commons-codec	1.10	Maven	[1.13]	commons-codec:commons-codec	1.10		5.0		XRAY-87486
Medium	org.postgresql:postgresql	42.2.0	Maven	[42.2.13]	org.postgresql:postgresql	42.2.0	CVE-2020-13692	6.8	7.7	XRAY-146611
Medium	org.postgresql:postgresql	42.2.0	Maven	[42.2.5]	org.postgresql:postgresql	42.2.0	CVE-2018-10936	6.8	8.1	XRAY-77885
Medium	log4j:log4j	1.2.17	Maven		log4j:log4j	1.2.17	CVE-2020-9488	4.3	3.7	XRAY-96751
Low	log4j:log4j	1.2.17	Maven		log4j:log4j	1.2.17		6.9		XRAY-87321

License Compliance Violations

LICENSE	SEVERITY	IMPACTED PACKAGE	IMPACTED PACKAGE VERSION	TYPE	COMPONENT	COMPONENT VERSION
BSD 2-Clause	High	com.ongres.scram:common	1.0.0-beta.2	Maven	com.ongres.scram:common	1.0.0-beta.2
BSD 2-Clause	High	org.postgresql:postgresql	42.2.0	Maven	org.postgresql:postgresql	42.2.0



ON DEMAND BINARY SCAN

- Build software securely during development, without trying to find and fix vulnerabilities after your code is compiled.
- Use JFrog CLI:
 - run ad-hoc scans for security purposes **without uploading to Artifactory first**.
 - point to a binary in your local file system and receive a report that contains a list of vulnerabilities and licenses for that binary.
- Get the results in Tabular or JSON format.

- `jf scan "{{file_path}}"` /
or
- `jf s "{{file_path}}"`

`jf docker scan {{image_name:tag}}`

[Command Options](#)



ON DEMAND BINARY SCAN

Security Violations										
Severity	Impacted Package	Impacted Package Version	Type	Fixed Versions	Component	Component Version	CVE	CVSS V2	CVSS V3	Issue ID
Critical	commons-io:commons-io	2.6	Maven	[2.7]	commons-io:commons-io	2.6		5.0		XRAY-78200
High	commons-io:commons-io	2.6	Maven	[2.8.0]	commons-io:commons-io	2.6		7.1	7.5	XRAY-125253
Medium	org.apache.httpcomponents:HttpClient	4.5.6	Maven	[4.5.13]	org.apache.httpcomponents:HttpClient	4.5.6	CVE-2020-13956	5.0	5.3	XRAY-129495
Medium	commons-io:commons-io	2.6	Maven	[2.7]	commons-io:commons-io	2.6	CVE-2021-29425	5.0	5.3	XRAY-172728
Medium	commons-codec:commons-codec	1.10	Maven	[1.13]	commons-codec:commons-codec	1.10		5.0		XRAY-87486
Medium	org.postgresql:postgresql	42.2.0	Maven	[42.2.13]	org.postgresql:postgresql	42.2.0	CVE-2020-13692	6.8	7.7	XRAY-146611
Medium	org.postgresql:postgresql	42.2.0	Maven	[42.2.5]	org.postgresql:postgresql	42.2.0	CVE-2018-10936	6.8	8.1	XRAY-77885
Medium	log4j:log4j	1.2.17	Maven		log4j:log4j	1.2.17	CVE-2020-9488	4.3	3.7	XRAY-96751
Low	log4j:log4j	1.2.17	Maven		log4j:log4j	1.2.17		6.9		XRAY-87321
License Compliance Violations										
License	Severity	Impacted Package	Impacted Package Version	Type	Component	Component Version				
BSD 2-Clause	High	com.ongres.scram:common	1.0.0-beta.2	Maven	com.ongres.scram:common	1.0.0-beta.2				
BSD 2-Clause	High	org.postgresql:postgresql	42.2.0	Maven	org.postgresql:postgresql	42.2.0				



DOCKER DESKTOP JFROG XRAY EXTENSION

JFrog Xray Scan

Select local image for scanning: centos:latest

Scan

Vulnerabilities: 190

Severity	Impacted Package	Version	Type	Fix Versions	CVE	CVSS 3.0	CVSS 2.0
High	pip	9.0.3	Python	N/A	CVE-2018-20225	7.8	6.8
High	pip	9.0.3	Python	[19.2]	CVE-2019-20916	7.5	5.0
High	python3-libs	0:3.6.8-37.el8	RPM	N/A	CVE-2019-9674	7.5	5.0

Summary:
Lib zipfile.py in Python through 3.7.2 allows remote attackers to cause a denial of service (resource consumption) via a ZIP bomb.

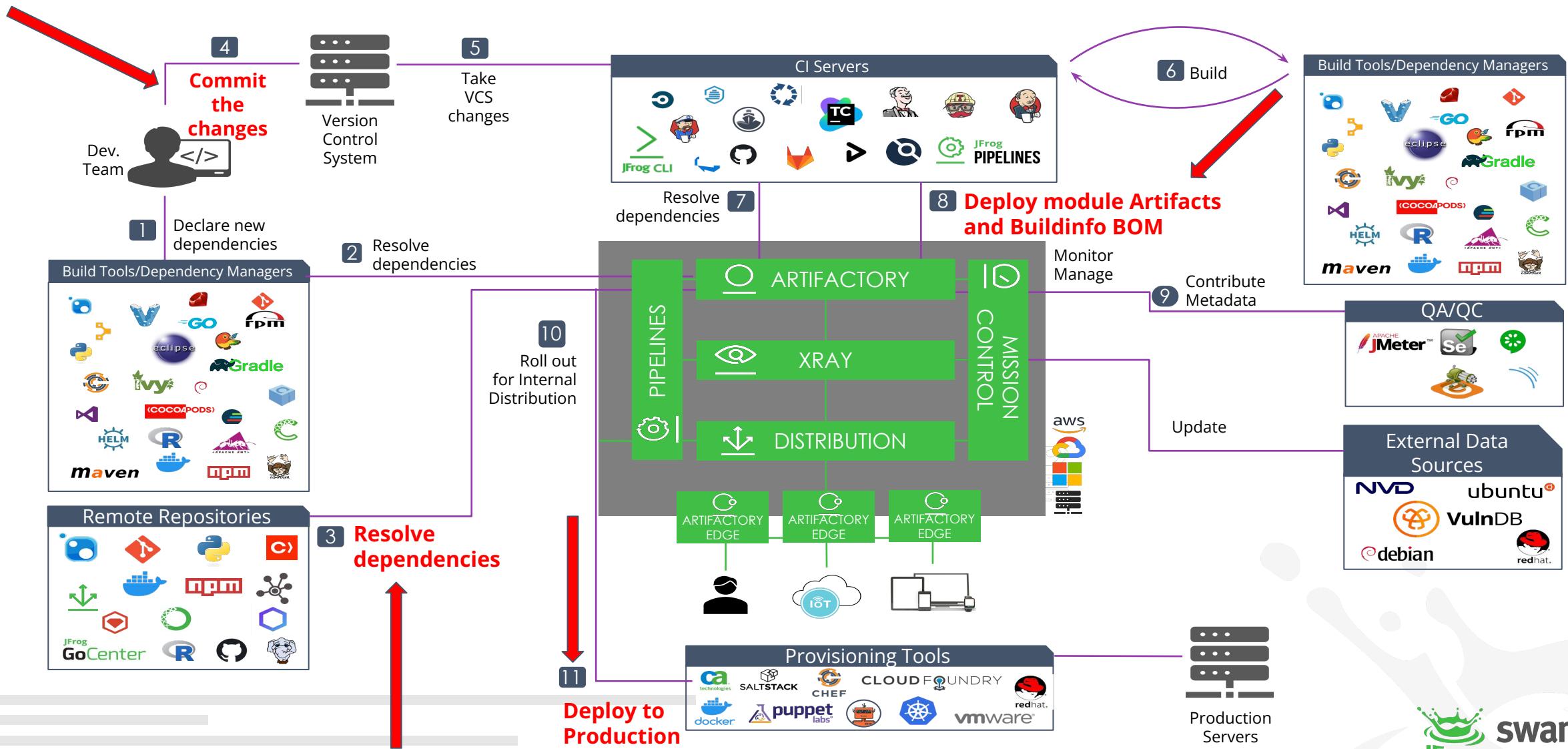
References:
<https://bugs.python.org/issue36260>
<https://bugs.python.org/issue36462>
<https://github.com/python/cpython/blob/master/Lib/zipfile.py>
<https://python-security.readthedocs.io/security.html#archives-and-zip-bomb>
<https://www.python.org/news/security/>

High bind-export-libs 32:9.11.26-3.el8 RPM [32:9.11.26-4.el8_4.] CVE-2021-25215 7.5 5.0

Diagram illustrating dependencies: centos → python3-libs → sha256_74ddd0ec08fa43d09f326

DEMO

WHERE DOES XRAY FIT?





FROGBOT

- [FrogBot](#) is available for FREE
- Protect your git projects as they are being developed.
- **Scans every Pull Request** created, for security vulnerabilities with JFrog Xray.
 - This is to ensure that new PR's don't add new security vulnerabilities to your code base.
- **Automatically** opens pull requests for upgrading vulnerable dependencies to a version with a fix.
- **What's supported**
 - Github, Gitlab and Bitbucket are supported.
 - NPM, Maven, Gradle, Go, Pip, Pipenv, NuGet, .Net, Yarn2

<< FROGBOT

A commit 2 #2
talarian1 wants to merge 2 commits into `main` from `test-branch`

[Open](#) | [Lock conversation](#)

talarian1 force-pushed the `test-branch` branch from `39c371a` to `3981060` 2 months ago

[Compare](#)

github-actions bot commented on Feb 13 • edited by talarian1

We scanned this pull request and found the issues below

SEVERITY	IMPACTED PACKAGE	VERSION	FIXED VERSIONS	COMPONENT	COMPONENT VERSION
High	github.com/nats-io/nats-streaming-server	v0.21.0	[0.24.1]	github.com/nats-io/nats-streaming-server	v0.21.0
High	github.com/mholt/archiver/v3	v3.5.1		github.com/mholt/archiver/v3	v3.5.1
Medium	github.com/nats-io/nats-streaming-server	v0.21.0	[0.24.3]	github.com/nats-io/nats-streaming-server	v0.21.0

Merged | Update Frogbot images #41
talarian1 merged 2 commits into `jfrog:dev` from `talarian1:dev` 30 minutes ago

[Update Frogbot images](#)

talarian1 added the `frogbot scan` label 1 hour ago

github-actions bot removed the `frogbot scan` label 1 hour ago

github-actions bot commented 1 hour ago • edited by talarian1

We scanned this pull request and found that it did not add vulnerable dependencies

What is Frogbot?

- > Update images size

yahavi approved these changes 32 minutes ago

talarian1 merged commit `278527f` into `jfrog:dev` 30 minutes ago

View details | Revert

DEMO

LAB 3

[SwampUp2022/SUP003-Intro to DevSecOps with JFrog Xray/lab-3](#)

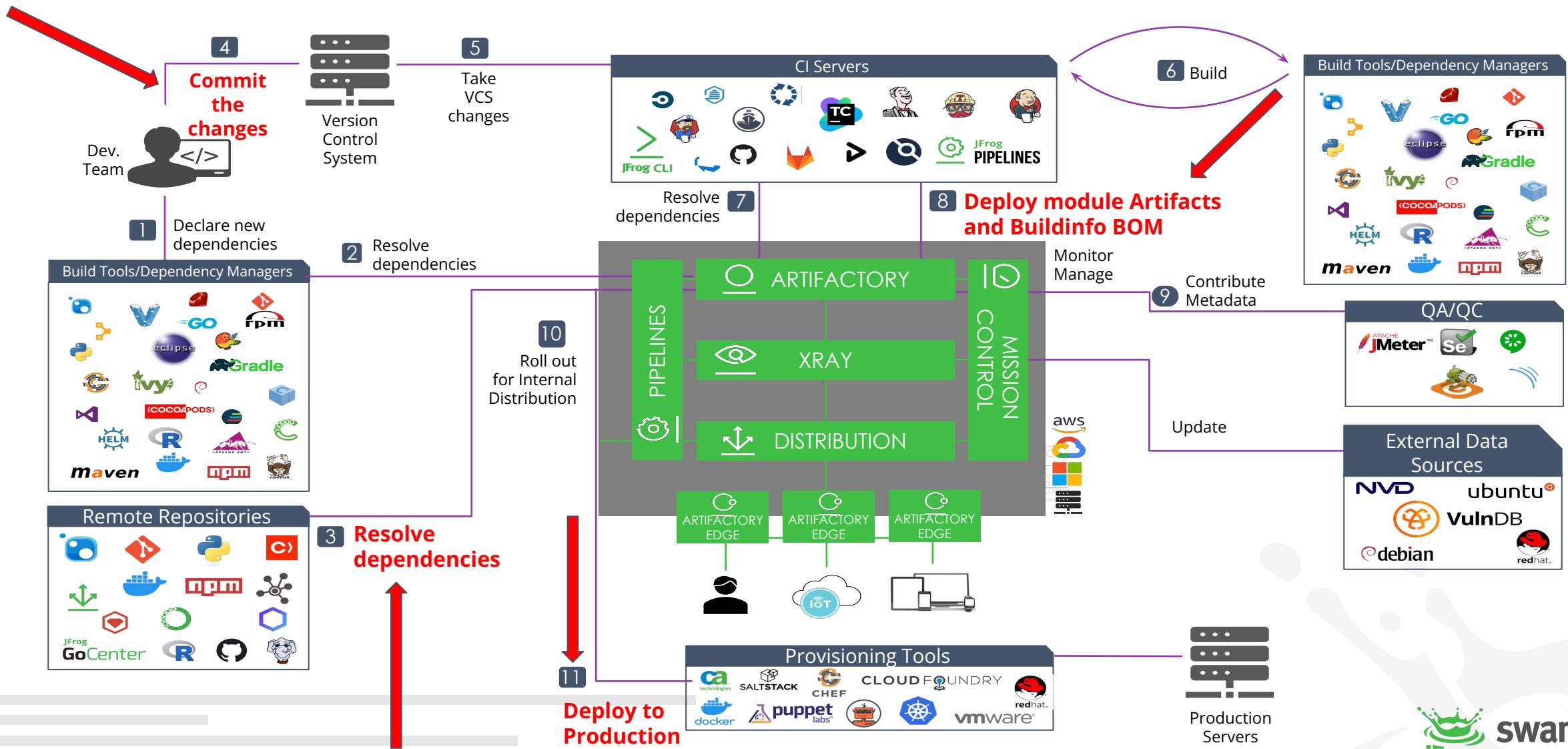


CI/CD INTEGRATION



The Liquid Software Company

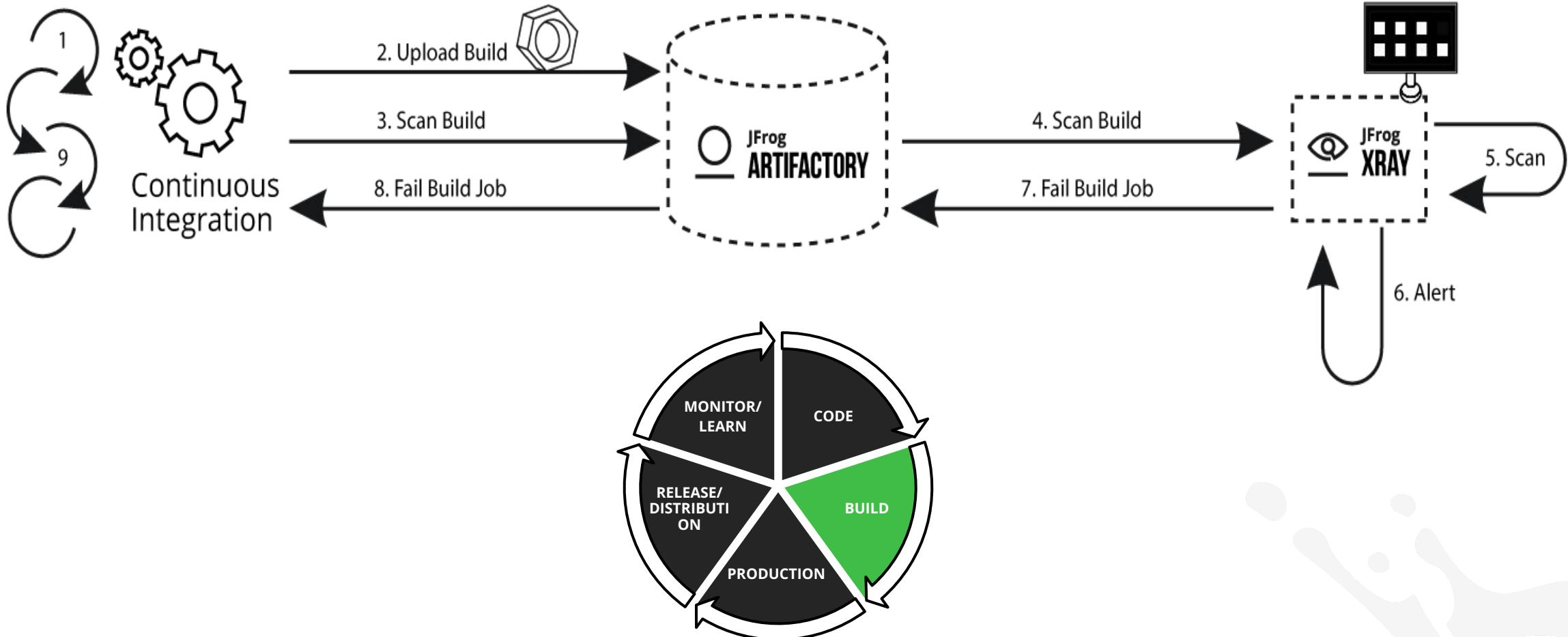
WHERE DOES XRAY FIT?



BUILD INFO

- System record for fully reproducible builds
- Dependency traceability per build
- Build info promotion
- System & Environment Variables
- Modules produced by Builds including Artifacts and Dependencies
- Build Project details - Produced by & Used By
- Difference between the Builds

SECURING AT BUILD TIME-CI

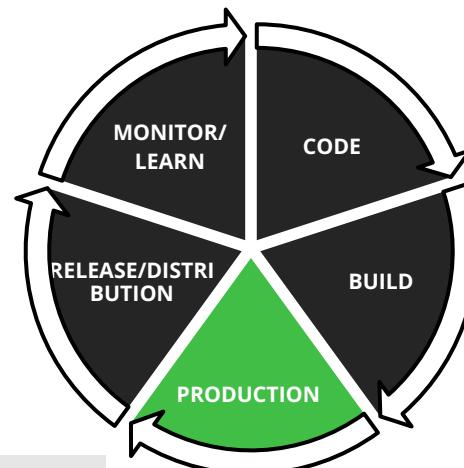


JFROG BUILD OFFERING BY PACKAGE TYPE

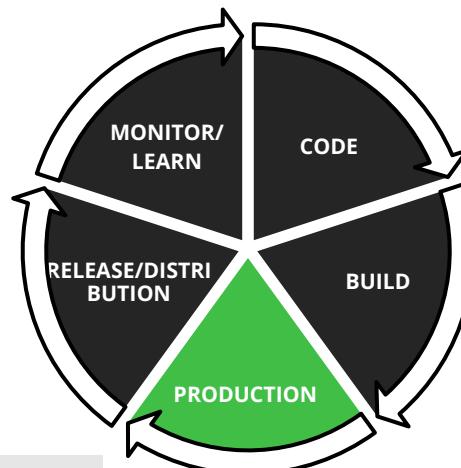
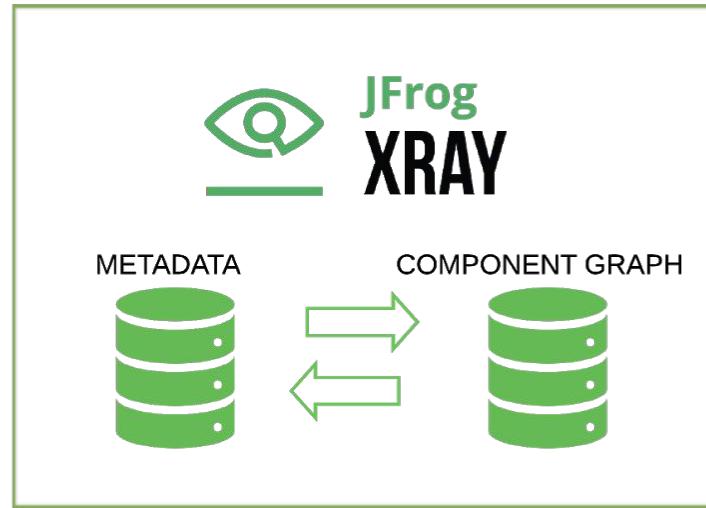
CI Servers								
Package Type	JFrog Pipelines	JFrog CLI	Jenkins	Bamboo	TeamCity	Azure DevOps	GitHub Actions	Bitbucket Pipelines
Generic	✓	✓	✓	✓	✓	✓	✓	✓
Conan			✓			✓		
Docker	✓	✓	✓			✓	✓	✓
Go	✓	✓	✓			✓	✓	✓
Gradle	✓	✓	✓	✓	✓		✓	✓
Ivy			✓	✓	✓			
Maven	✓	✓	✓	✓	✓	✓	✓	✓
npm	✓	✓	✓	✓		✓	✓	✓
NuGet	✓	✓				✓	✓	✓
Python		✓					✓	✓
Terraform		✓					✓	
Podman		✓						
Kaniko		✓	✓					

DEMO

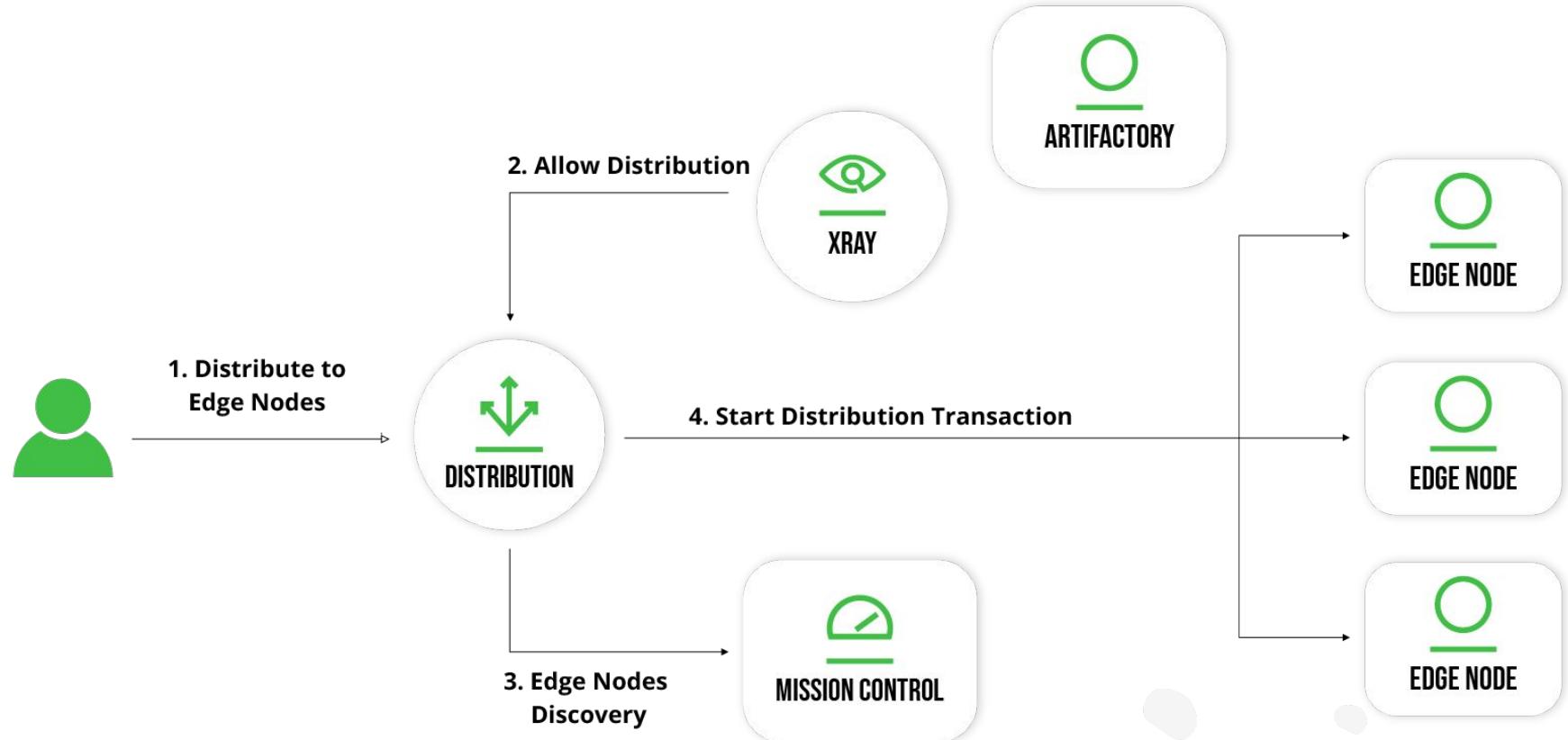
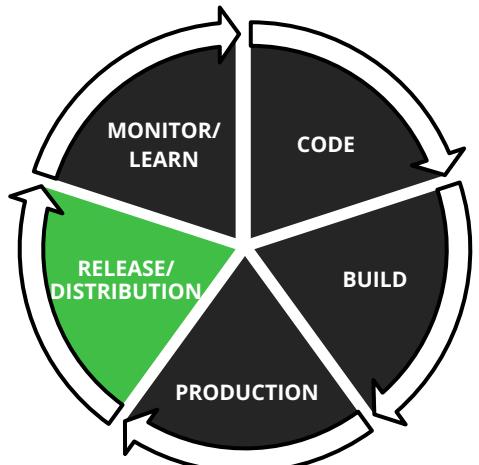
SECURING AT PRODUCTION



CONTINUOUS ANALYSIS AT PRODUCTION



SECURING DISTRIBUTION RELEASE BUNDLES



DEMO



REPORTS



The Liquid Software Company

REPORT TYPES

- Vulnerabilities Report
 - Vulnerabilities in Artifacts, Builds, Release Bundles, JFrog Project
- Licenses Due Diligence Report
 - List of Components and Artifacts and their relevant Licenses such as unknown licenses and unrecognized licenses found in component
- Operational Risk Reports
 - Data on OSS components that will help you gain insight into risk level of the components in use such as EOL, version Age, and so on.
- Violations Report
 - Security and license violations for each component in the selected scope such as type of violation, impacted artifacts, and severity

SOFTWARE BILLS OF MATERIALS (SBOM) REPORT

- Has become a critical DevSecOps piece.
- SBOM is a readable inventory of software components and dependencies which make up a piece of software.
- Supported SBOM formats:
 - SPDX & CycloneDX

Sec. 4. Enhancing Software Supply Chain Security
(vii) providing a purchaser a Software Bill of Materials (SBOM) for each product directly or by publishing it on a public website;



BRIEFING ROOM

Executive Order on Improving the Nation's Cybersecurity

MAY 12, 2021 • PRESIDENTIAL ACTIONS

By the authority vested in me as President by the Constitution and the laws of the United States of America, it is hereby ordered as follows:

Section 1. Policy. The United States faces persistent and increasingly sophisticated malicious cyber campaigns that threaten the public sector, the private sector, and ultimately the American people's security and privacy. The Federal Government must improve its efforts to identify, deter, protect against, detect, and respond to these actions and actors. The Federal Government must also carefully examine what occurred during any major cyber incident and apply lessons learned. But cybersecurity requires more

LAB 4

[SwampUp2022/SUP003-Intro to DevSecOps with JFrog Xray/lab-4](#)



RECENTLY INTRODUCED FEATURES



The Liquid Software Company

NEW FEATURES SUMMARY

- Security CVE Research and Enrichment of data
- Contextual Analysis / Applicability Scanner
- Secrets Detection
- Application Exposures
- Service Exposures
- IaC Security Analysis

RESEARCH AND ENRICHMENT OF CVE DATA

 CVE-2021-22924

Xray ID	XRAY-180050 
Severity	 Medium Source: Red Hat Security Advisory
JFrog Research Severity	 Medium
CVSS Score	 4.3 (v2)  3.1 (v3)
Artifact	centos:7 
Affected Components	2 Components
Watch	test-watch
Policy	sec-policy (Rule: sec-rule)

Remediation | Fix versions: ≥ 7.78.0

Development upgrade

- Upgrade the component to any of the suggested fixed versions.

Deployment mitigation

When using any of the following arguments:

`issuercert, cafile, carootdir, clientcert, randomfile, randomsock`

Make sure to use paths which are inaccessible to attackers (ex. non-world-writable directories)

CONTEXTUAL ANALYSIS / APPLICABILITY SCANNER

- Applicability of a CSV to your application, **through analysis of reachable paths and configurations at the binary level.**

The screenshot shows the JFrog Xray interface for a scan named "sharan-docker-test" on the "juice-shop/latest" target. The main view displays 57 Vulnerabilities. A detailed view is open for CVE-2020-8203, which is marked as Not Applicable due to contextual analysis. The detailed view includes information such as ID (XRAY-114089), JFrog Severity (Critical), Component (lodash), Version (2.4.2), Upgrade to (4.17.20), CVSS Score (5.8 (V2) / 7.4 (V3)), CWE (CWE-1321), and Jfrog Research. The "Contextual Analysis" tab is selected, showing a breakdown for npm://lodash:2.4.2. It explains that the scanner checks whether the vulnerable function `zipObjectDeep` is called with external input to its 1st (`props`) and 2nd (`values`) arguments. It also notes that the vulnerable function `zipObjectDeep` is never called with external input.

Severity	ID	Contextual Analysis	Component	Affected
High	CVE-2020-8203	Not Applicable	lodash	tar
Medium	CVE-2015-9235	---	jsonwebtoken	tar
Medium	CVE-2019-10744	Not Applicable	lodash	tar
Medium	XRAY-91651	---	marsdb	tar
Low	CVE-2021-32804	Not Applicable	lodash	tar
Low	CVE-2021-32803	---	tar	tar
Low	CVE-2021-37713	---	tar	tar
Low	CVE-2021-37712	Not Applicable	tar	tar

SECRETS DETECTION

- Discover secrets such as **passwords, access tokens, API keys** that can leaked through config files, IaC modules and binaries.

The screenshot shows the JFrog Xray application interface. On the left is a dark sidebar with a green header 'Application' containing various navigation items like Dashboard, Artifactory, Xray, Scans List, Watch Violations, Reports, On-Demand Scanning, Distribution, Pipelines, Integrations, Learning Center, and a JFrog logo at the bottom. The main content area has a light background. It shows a breadcrumb path: Xray > Scans List > sharan-docker-test > juice-shop/latest. Below this is a section titled '4 Secrets Issues' with a table:

Status	JFrog Severity	ID	Description
TO FIX	!	EXP-1520-00001	Hardcoded random b
TO FIX	!	EXP-1520-00002	Hardcoded random b
OK	H	EXP-1235-00001	Plaintext API keys fo
OK	!	EXP-1519-00001	Hardcoded random b

To the right of the table is a detailed view for the third issue (Plaintext API Keys Found):

H Plaintext API Keys Found

Status	OK
ID	EXP-1235-00001
CWE	CWE-256 []
Abbreviation	REQ.KEY.API.ENCRYPT
Fix Cost	Medium

Findings Outcomes

What was found?

What does it mean?

API keys are commonly used by embedded devices for access to SaaS APIs in the cloud. More often than not, the same key is shared between all devices.

What can happen?

Storing an API key in the image could lead to several risks. If the key is associated with a wide scope of privileges, attackers could extract it from a single image or firmware and use it maliciously to attack many targets. For example, if the embedded key allows querying/modifying data for all cloud user accounts, without per-user authentication, the attackers who extract it would gain access to systemwide data. If the cloud/SaaS provider bills by key usage - for example, every million queries cost the key's owner a fixed sum of money - attackers could use the keys for their own purposes (or just as a form of vandalism), incurring a large cost to the legitimate user or operator.

What should I do?

Use narrow scopes for stored API keys. As much as possible, API keys should be unique per host and require additional authentication with the user's individual credentials for any sensitive actions. Avoid placing keys whose use incurs costs directly in the image. Store the key with any software or hardware protection available on the host for key storage (such as operating system key-stores).

APPLICATION EXPOSURE

- find **configuration issues, security malpractices** and **insecure usage of popular libraries** related to your application framework.

The screenshot shows the JFrog Xray application interface. On the left is a navigation sidebar with sections like Application, Dashboard, Artifactory, Xray, Scans List, Watch Violations, Reports, On-Demand Scanning, Distribution, Pipelines, Integrations, and Learning Center. The main area shows a scan named "sharan-docker-test" for "juice-shop/latest". A table lists "8 Applications Issues" with columns: Status, JFrog Severity, ID, and Description. Most issues are marked as "TO FIX" with a severity of "H" or "M". One issue is marked as "OK" with a severity of "M". The right side provides a detailed view of a specific finding:

Node.js Application Does Not Enforce TLS On All Web Communications

Status	ID	Description
TO FIX	EXP-1058-00001	Node.js application does not enforce TLS on all web communications
TO FIX	EXP-1056-00001	Passwords stored in plain text
TO FIX	EXP-1056-00002	Passwords stored in plain text
TO FIX	EXP-1056-00003	Passwords stored in plain text
TO FIX	EXP-1056-00004	Passwords stored in plain text
TO FIX	EXP-1059-00001	Node.js application does not enforce TLS on all web communications
OK	EXP-1061-00001	Node.js application does not enforce TLS on all web communications
OK	EXP-1091-00001	Node.js executes untrusted code

Findings Outcomes

What was found?
https.createServer is not used, Express is used

Path	Evidence	Line Number
/juice-shop/build/server.js	express()	75

What does it mean?
By default, Node.js serves content over HTTP. Generate a server certificate file and certificate chain and configure the server to use TLS.

What can happen?
Communicating online without applying authentication, encryption, and integrity protections can result in spoofing, information disclosure, or tampering - in other words, communications could be read, modified, and even injected.

SERVICE EXPOSURE

- detect **configuration issues** and **security malpractices** for specific services and daemons included in your artifacts.

The screenshot shows the JFrog Xray interface with the following details:

Left Sidebar:

- Artifactory
- Xray (selected)
- Scans List
- Watch Violations
- Reports
- On-Demand Scanning
- Distribution
- Pipelines

Central View:

Xray > Scans List > containers > ics/latest

Scan Name: containers

14 Services Issues

Status	JFrog Severity	ID	Description
TO FIX	H	EXP-1545-00...	Envoy upstream connections do no
TO FIX	H	EXP-1546-00...	Envoy upstream connections do no
TO FIX	H	EXP-1548-00...	Envoy does not enable TLS for dow
TO FIX	H	EXP-1551-00...	Envoy admin interface is externally
TO FIX	H	EXP-1552-00...	etcd client connections do not use
TO FIX	H	EXP-1553-00...	etcd peer connections do not use T
TO FIX	H	EXP-1554-00...	Prometheus basic authentication is
TO FIX	H	EXP-1555-00...	Prometheus TLS encryption is disa

Right Panel - Envoy Upstream Connections Do Not Verify TLS Peer Certificates:

Status: TO FIX

ID: EXP-1545-00001

CWE: CWE-295 []

Abbreviation: REQ_SW_ENVOY_TLS_UPSTREAM_ENABLE

Fix Cost: Medium

Findings: trusted_ca was not detected

Evidence: /etc/envoy/req.sw.envoy.tls... validation_context: 111

What does it mean? Enable TLS for Envoy upstream connections to ensure that eavesdroppers and hackers are unable to see the data transmitted between two Envoy instances.

What can happen? Connecting to an "upstream" TLS service is conversely done by adding an `UpstreamTLSContext` to the `transport_socket` of a cluster -

IaC SECURITY ANALYSIS

- Scan your **Terraform** state and module files in Artifactory for **configuration issues** such as **insecure use of credentials**, and **insufficient access restriction** to services.

The screenshot shows the JFrog Xray interface. On the left, the navigation bar includes Application, Dashboard, Artifactory, Xray (selected), Scans List, Watch Violations, Reports, On-Demand Scanning, Distribution, Pipelines, Integrations, Learning Center, and JFrog Cloud. The main area displays a scan named "sharan-terraform-be-test" with a file named "state.latest.json". The results show 15 IaC Issues:

Status	JFrog Severity	ID	Description
OK	High	EXP-1527-00001	Authorization is not explicitly defined for the ECS service.
OK	High	EXP-1528-00001	Public policies are not being used to restrict access to the ECS service.
OK	High	EXP-1529-00001	The Root IAM user has administrative permissions.
OK	High	EXP-1530-00001	ECS services are using IAM users with admin privileges.
OK	High	EXP-1534-00001	Found IAM users with admin privileges.
OK	High	EXP-1535-00001	Found publicly-accessible IAM users.
OK	High	EXP-1536-00001	Found RDS DB instances using IAM users with admin privileges.
OK	High	EXP-1537-00001	Found unencrypted data at rest.
OK	High	EXP-1539-00001	Found batch jobs running under IAM users with admin privileges.
OK	High	EXP-1542-00001	Found ECR repositories using IAM users with admin privileges.

A detailed finding for "ECS Services Are Using Admin Roles" is expanded:

- What does it mean?**: Amazon Elastic Container Service (Amazon ECS) is a highly scalable and fast container management service. It can be used to run, stop, and manage containers on a cluster. With Amazon ECS, containers are defined in a task definition that is used to run an individual task or task within a service. A policy granting administrative access to Amazon ECS resources can be dangerous. Using this policy allows access to all of Amazon ECS features that are available in the AWS Management Console. It can lead to unwanted behavior if an attacker gets access to an ECS service with admin privileges.
- What can happen?**: An attacker getting access to the ECS service, using this policy allows access to all of Amazon ECS features that are available in the AWS Management Console.
- What should I do?**:
 - Terraform**: An `aws_ecs_service` resource must not set `iam_role = "admin"`.
Vulnerable example -

```
resource "aws_ecs_service" "secure_service" {
    name      = "myservice"
    ...
    iam_role  = "admin"
    ...
}
```
 - Secure example -**

```
resource "aws_ecs_service" "secure_service" {
    name      = "myservice"
    ...
    iam_role  = "some_role"
    ...
}
```

DEMO

ADDITIONAL FEATURES IN ROADMAP

- Support for Conda Packages scan
- Support for OCI Containers scan
- Archive file scan support : AAR and XZ files
- SPDX and CycloneDX, SBOM has CPE string for identifying software artifact
 - cpe:/<part>:<vendor>:<product>:<version>:<update>:<edition>:<language>
- Policy framework for Adv. Security Features
- IaC scan
 - For k8s
 - Scan in IDE

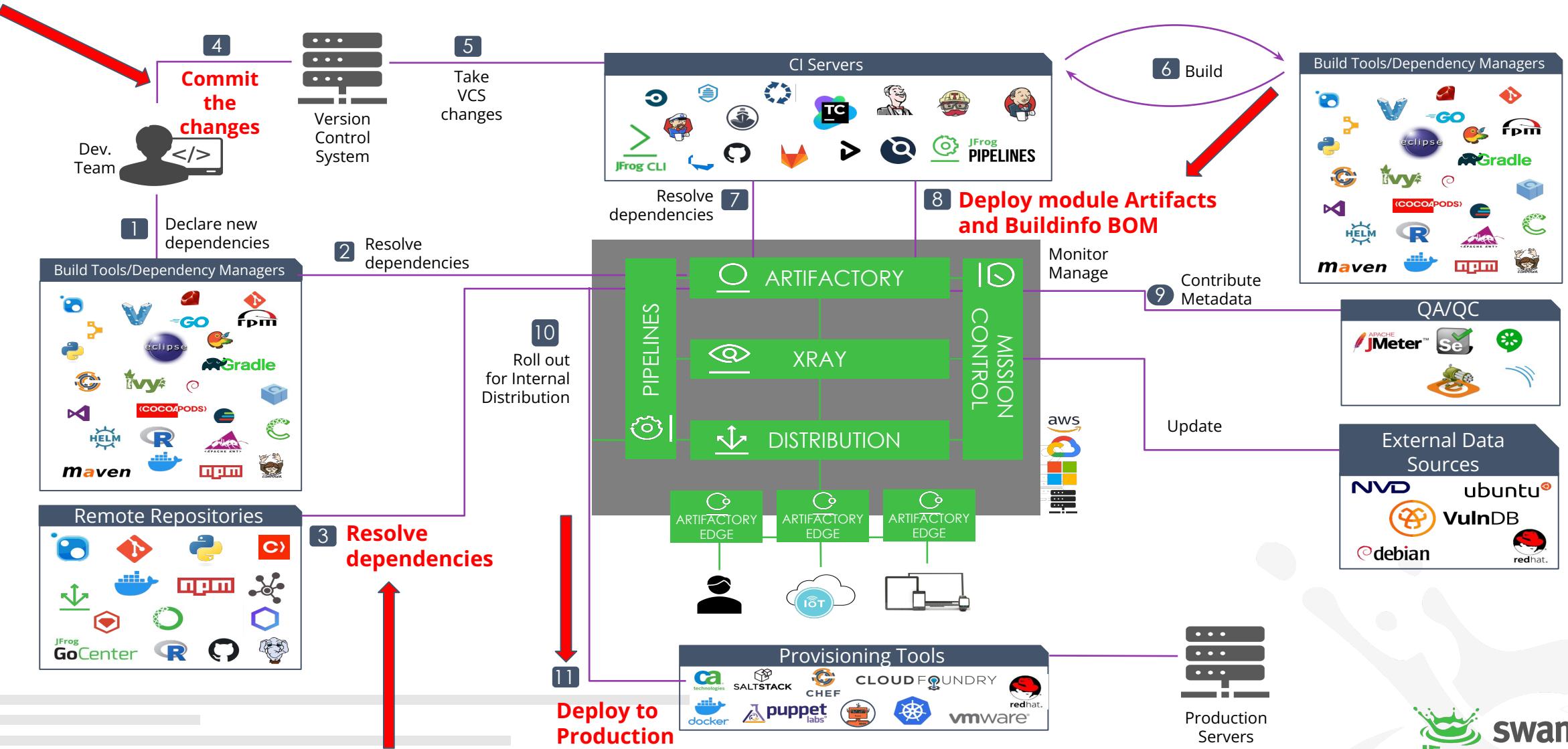


CONCLUSION



The Liquid Software Company

WHERE DOES XRAY FIT?



AGENDA

09:00-10:25

DevOps & DevSecOps Overview
Introduction To JFrog Xray
Policies and Watches
Impact Analysis

10:25-10:40

Break

10:40-11:50

IDE Plugin Integration
CI / CD Integration
Scan Reports
New additions/features

11:50-12:00

Summary, Quiz, Survey & Q/A





CLASS SURVEY



<https://www.surveymonkey.com/r/swampup22NYCtraining>



The Liquid Software Company



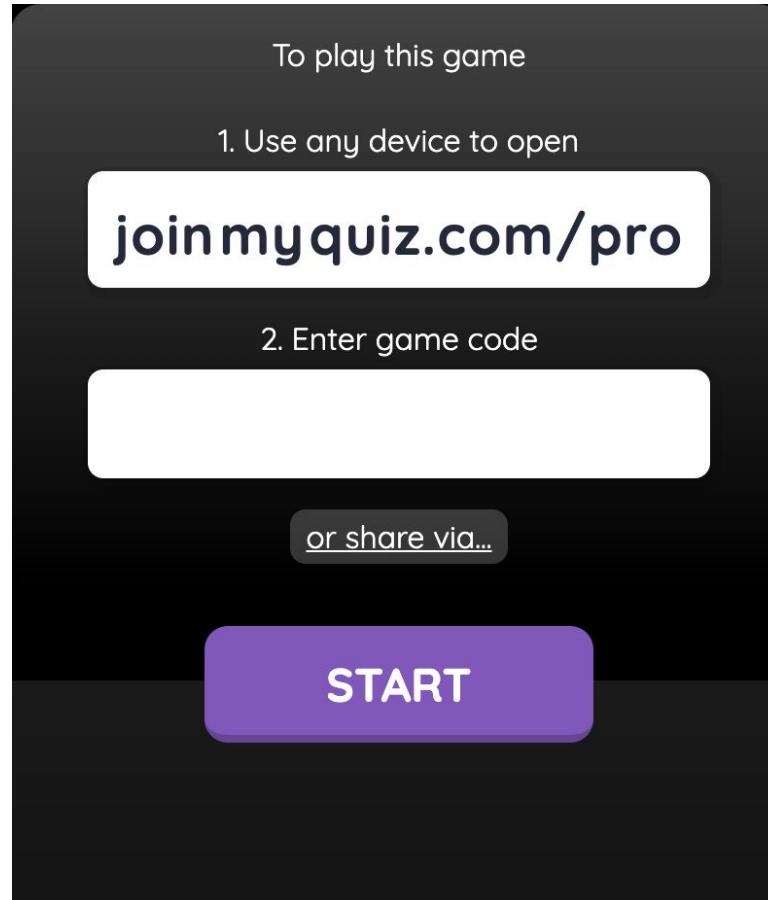
CLASS QUIZ

Please join our quiz!

Game URL will be shared in Class

Enter your Game Code to begin

<https://quizizz.com/pro/join?gc=456733>



COURSE RESOURCES

- Course material: [GitHub](#)
- JFrog Platform [Overview](#)
- JFrog Xray [Developer Guide](#)
- JFrog Xray [REST APIs](#)
- JFrog Xray [IDE Integrations](#)
- JFrog Xray [CLI](#)
- JFrog [Frogbot](#)
- JFrog Security [Research](#)
- New [Advanced Security](#) Features - [wiki](#)
- JFrog Academy: [Online Training](#)
- JFrog Free Tier: [Start Free](#)
- Adv Security Features [Test Drive](#)



Q & A



THANK YOU!

