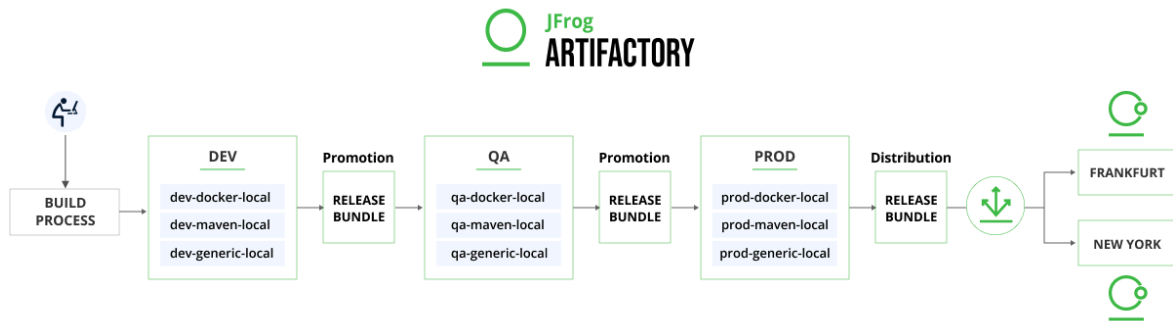


JFTD-113 JFrog Release LifeCycle Management

Hands-on Labs Walk-Through



Reference Materials

This section is a link to documentation and examples that are used both in the lab and also post SwampUp

- [JFrog Help Center](#)
- [Repository Management](#)
- [Jfrog Environments](#)
- [Release Lifecycle Management](#)
- [JFrog CLI Download](#)
 - [JFrog CLI Wiki](#)

Lab 1: UI - Release Bundle Creation / Distribution / Xray Quality Gate (Prod)

Purpose: The purpose of this lab is to show how using the JFrog Platform UI, how to create and distribute a Release Bundle from 3 different builds being stored in Artifactory that act as a single release.

Prerequisites:

- Access to the JFrog Platform
- Understanding the Repositories from which the Binaries for the Release Bundles are coming from
- Coveration
 - Generation with BuildInfo
 - Indexing for Release Bundle - Attached to Prod
 - Policy for release blocking
 - Second example: add evidence

Explanation:

- We will be creating a Release Bundle with three elements inside from three different repo types
 - Docker
 - dev-docker-local
 - Helm
 - dev-helm-local
 - Maven
 - dev-Maven-local
- We will then promote the Release Bundle from one environment to another
 - Dev > QA
 - QA > Staging
 - Staging > Production
 - Xray Gate for Scanning pre-Production
-

Process(Step by Step UI walk-through):

- Validate that the required Environments are available for promoting our Release Bundle
 - Dev
 - QA
 - STAGING
 - PROD

Environments

Global Environments (4)

DEV	QA	STAGING	PROD
Used In 3 Repositories 0 Roles	Used In 0 Repositories 0 Roles	Used In 0 Repositories 0 Roles	Used In 0 Repositories 0 Roles

- Validate that you have the Repositories in your instances that are required for this lab - these will be these repositories and will have their corresponding Environments associated with the Builds and the location where the builds we are using for the Release bundles are located.

Local Repositories

- Docker
 - dev-docker-local
 - prod-docker-local
 - staging-docker-local
 - prod-docker-local

Repositories

Virtual

Local

Remote

Federated

+ Assign Repository

13 Repositories

Repository Key ↑	Type	Project	Environment	Replications	Shared With
jftd113-dev-docker-local	 Docker	JFTD-113	DEV	0	0
jftd113-prod-docker-local	 Docker	JFTD-113	PROD	0	0
jftd113-qa-docker-local	 Docker	JFTD-113	QA	0	0
jftd113-stg-docker-local	 Docker	JFTD-113	STAGING	0	0

- Helm
 - dev-helm-local
 - qa-helm-local
 - staging-helm-local
 - prod-helm-loc

Repositories

[+ Assign Repositories](#)

Virtual **Local** Remote Federated

13 Repositories

Repository Key ↑	Type	Project	Environment	Replications	Shared With
jftd113-dev-helm-local	 HelmOCI	JFTD-113	DEV	0	0
jftd113-prod-helm-local	 HelmOCI	JFTD-113	PROD	0	0
jftd113-qa-helm-local	 HelmOCI	JFTD-113	QA	0	0
jftd113-stg-helm-local	 HelmOCI	JFTD-113	STAGING	0	0

○ Maven

- dev-Maven-local
- qa-Maven-local
- staging-Maven-local
- prod-Maven-local

Repositories

[+ Assign Repositories](#)

Virtual **Local** Remote Federated

13 Repositories

Repository Key ↑	Type	Project	Environment	Replications	Shared With
jftd113-dev-maven-local	 Maven	JFTD-113	DEV	0	0
jftd113-prod-maven-local	 Maven	JFTD-113	PROD	0	0
jftd113-qa-maven-local	 Maven	JFTD-113	QA	0	0
jftd113-stg-maven-local	 Maven	JFTD-113	STAGING	0	0

Remote Repositories

These are utilities for the transitive dependencies for the builds

Repositories

+ Assign Repositories

Set Me Up

Create a Repository

Virtual Local Remote Federated

3 Repositories

Search

Repository Key ↑	Type	Project	Environment	URL	Replications	Shared With	Actions
jftd113-dev-docker-remote	 Docker	JFTD-113	DEV	https://registry-1.docker.io/	0	0	...
jftd113-dev-helm-remote	 HelmOCI	JFTD-113	DEV	https://registry-1.docker.io/	0	0	...
jftd113-dev-maven-remote	 Maven	JFTD-113	DEV	https://repo1.maven.org/...	0	0	...

Virtual Repositories

These are combination of both the Local and Remote repositories that are being used for the builds

Repositories

+ Assign Repositories

Set Me Up

Virtual Local Remote Federated




3 Repositories

Repository Key ↑	Type	Project	Environment	Selected Repositories	Shared With	Actions
jftd113-dev-docker-virt	 Docker	JFTD-113	DEV	2 jftd113-dev-docke...	0	...
jftd113-dev-helm-virt	 HelmOCI	JFTD-113	DEV	2 jftd113-dev-helm-l...	0	...
jftd113-dev-maven-virt	 Maven	JFTD-113	DEV	2 jftd113-dev-mave...	0	...

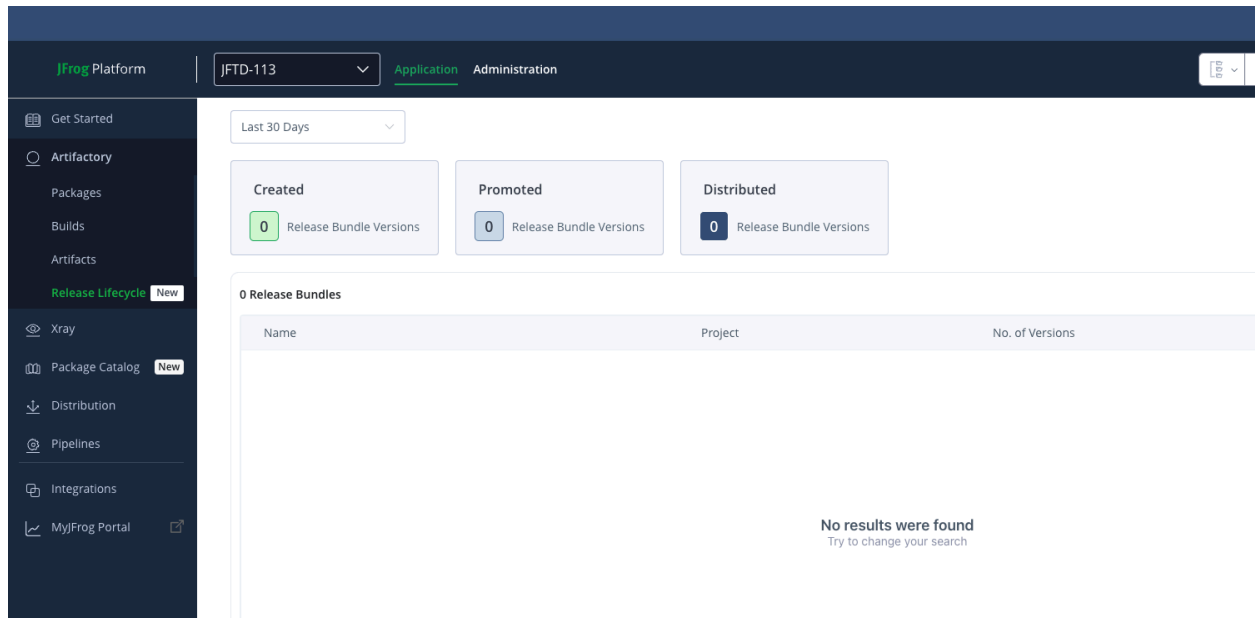
- Let's all validate that we have artifacts in the repositories that we are using for the Release Bundle, we will have 2 Artifacts (Maven = Jar, Helm = Template) and 1 Build (Docker)
 - Helm, Maven and Docker build instructions are located in the git project in the helm, maven, and docker folders and view the README

JFrog Platform | puser0 | Application Administration | after:2024-06-01 | Select a Plan | Request a Demo

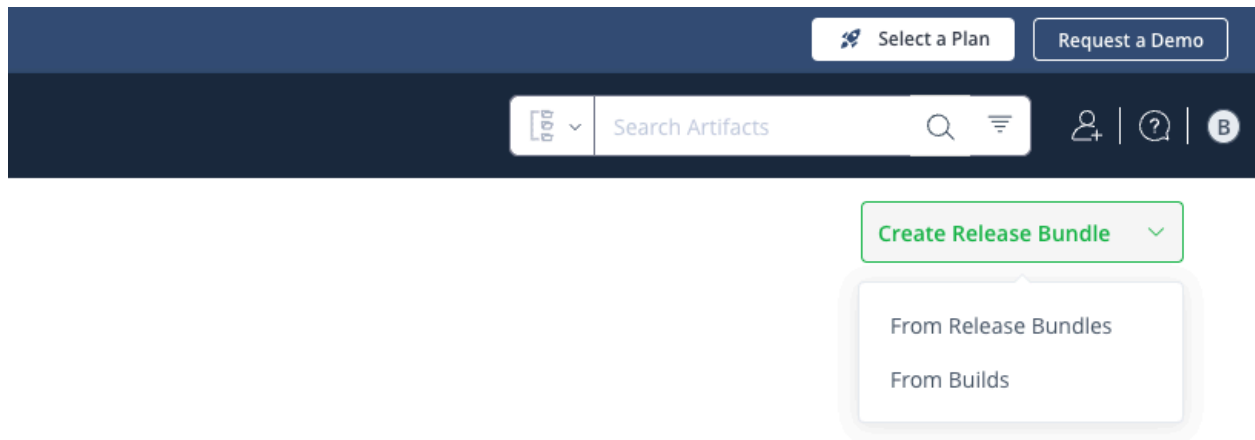
Artifactory > Builds

Build Name	Build Repository	Last Build ID	Last Build Time ↓	
<input type="checkbox"/> sample-maven-build	puser0-build-info	1.0.1	06-09-24 15:32:29 -0700	
<input type="checkbox"/> sample-docker-build	puser0-build-info	1.0.1	06-09-24 14:26:20 -0700	
<input type="checkbox"/> sample-helm-build	puser0-build-info	1.0.1	06-09-24 09:45:11 -0700	

- Next we will navigate to Release Lifecycle located at Application > Artifactory > Release Lifecycle



- Now on the far right hand side of the panel, click the “Create Release Bundle” and select “From Builds”



- You will be presented with a pop out where we will create the Release Bundle from the builds

New Release Bundle

1. Release Bundle Details 2. Builds Selection

* Release Bundle Name

* Release Bundle Version

* Signing Key ⓘ
rk-promotion-distribution-key - GPG

Cancel Next

Background Interface:

JFTD-113 Application Administration

Last 30 Days

Created **Promoted**

1 Release Bundle Versions 1 Release Bundle Versions

1 Release Bundle

Name	Project
RLM-Test-JFTD113	JFTD-113

- We will fill in the following fields
 - Release Bundle Name - This can be any but it can not have space
 - Release Bundle Version - This can be any designator you desire

This is WRONG

*** Release Bundle Name**

test bundle

⊗ Must begin with [a-z A-Z _ 0-9] and consist of [a-z A-Z _ . - 0-9]

*** Release Bundle Version**

⊗ Must begin with [a-z A-Z _ 0-9] and consist of [a-z A-Z _ . - 0-9]

*** Signing Key** ?

rk-promotion-distribution-key - GPG

This is CORRECT

*** Release Bundle Name**

test-bundle

*** Release Bundle Version**

1.0.1

We will also need to select a [Signing Key](#) that we wish to create the Release Bundle with, Release Bundles are immutable

Note - you can [change a Signing Key](#) when we get the promotion phase

*** Signing Key** ?

rk-promotion-distribution-key - GPG

- Now press the “Next” button on the bottom right

New Release Bundle



1. Release Bundle Details

2. Builds Selection

Builds Selection

Build Name

Build Version

+ Add

☐ Include Build Dependencies

- You will be presented with area where you can select the Builds and Versions that you wish to include in the Release Bundle - We will choose the 3 Builds we have in Artifactory - Maven, Helm, and Docker

1. Release Bundle Details

2. Builds Selection

Builds Selection

Build Name

sample-helm-build

Build Version

Select a version

+ Add

☐ Include Build Dependencies

1.0.1

sample-maven-build - 1.0.1 ×

sample-docker - 1.0.1 ×

- When you are satisfied with your Build selection, press the “Next” button on the lower right hand side.

Last 30 Days ▼ Create Release Bundle ▼

Created
2 Release Bundle Versions

Promoted
1 Release Bundle Versions

Distributed
0 Release Bundle Versions

2 Release Bundles

Name
test-bundle
RLM-Test-JFTD113

× JFTD-113 / test-bundle Actions ▼

Promotions Distributions

New DEV QA STAGING PROD

1.0.1
Sep 05, 2024

- You are presented with with the Kanban view of Release Bundle(s) and the various stage of your Release Lifecycle that you are defined in your environments
- Now you will [Promote](#) the Release Bundle to your target Environment
- Simple drag the Release Bundle to the desired Environment

New Promotion | Release Bundle Test-Bundle - 1.0.1



Promotion Environment

Target Repositories

Define a target environment for this promotion

Release Bundle Name

test-bundle

Release Bundle Version

1.0.1

* Signing Key

rk-promotion-distribution-key - GPG

* Target Environment

DEV

Cancel

Next

- You will be presented with a popout where you can now perform the Promotion

Define a target environment for this promotion

Release Bundle Name

test-bundle

Release Bundle Version

1.0.1

* Signing Key

rk-promotion-distribution-key - GPG

* Target Environment

DEV

DEV

QA

STAGING

PROD

- Select the Environment that you want to Promote to and click “Next” on the bottom right hand side

Review and confirm the target repositories for this promotion

Package Type	Target Repositories
 Docker	<input type="text" value="puser0-dev-docker-local"/>
 Helm	<input type="text" value="puser0-dev-helm-local"/> + 1
 Maven	<input type="text" value="puser0-dev-maven-local"/>

- You will be presented with the Target Repository associated with the Environment
- Then you will click “Promote” on the bottom right hand corner

Last 30 Days

Create Release Bundle

Created
2 Release Bundle Versions

Promoted
2 Release Bundle Versions

Distributed
0 Release Bundle Versions

2 Release Bundles

test-bundle

RLM-Test-JFTD113

JFTD-113 / test-bundle

Promotions

Distributions

New

DEV

QA

STAGING

PROD

1.0.1

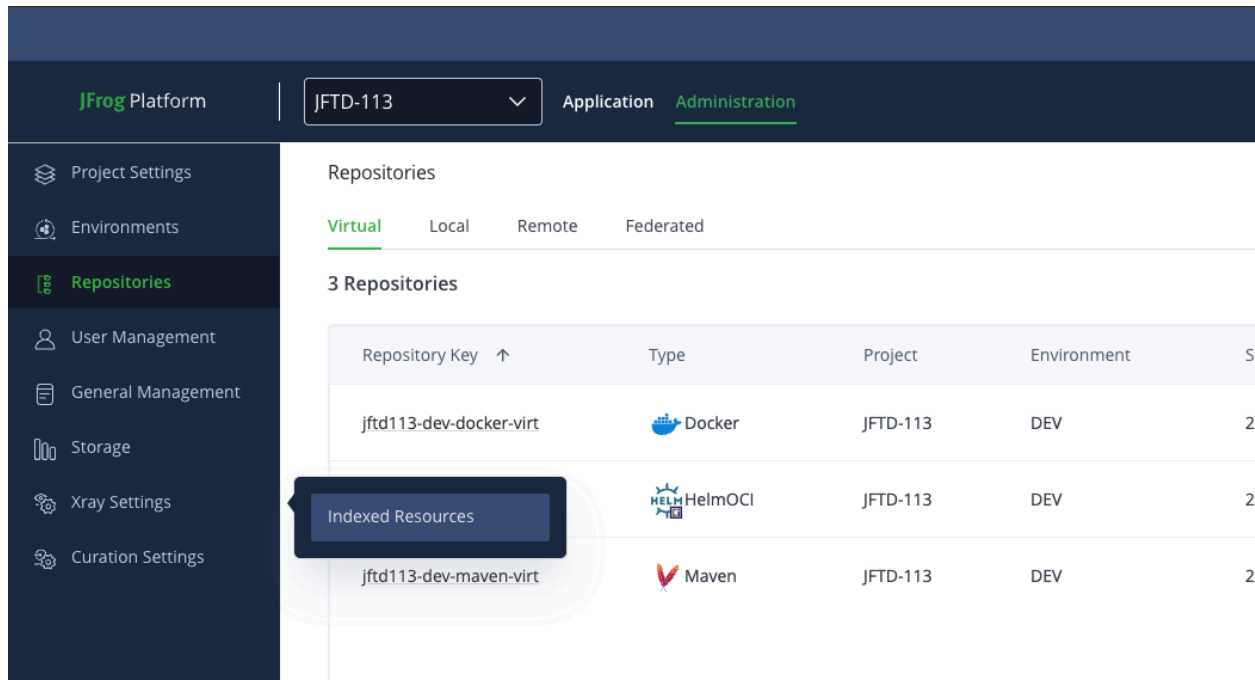
Sep 05, 2024

- The Release Bundle has now been Promoted to the next Environment

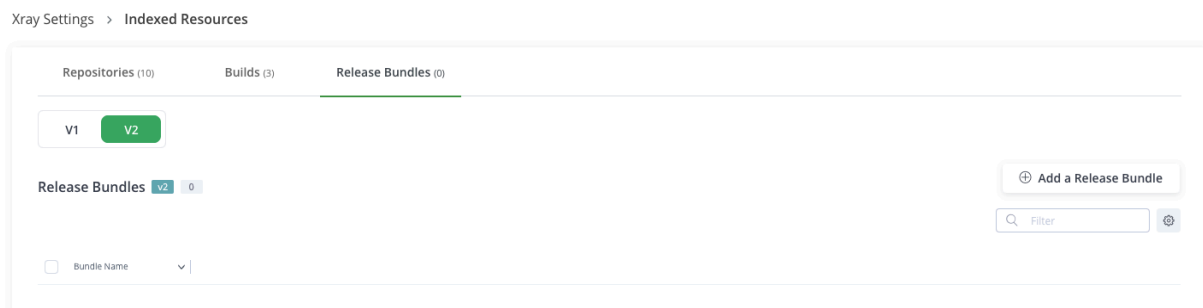
JFrog Xray as a Quality Gate for Promotion / Distribution

Now let's create a Xray Policy that will [Scan a Release Bundle](#) and act a gate between promotion steps

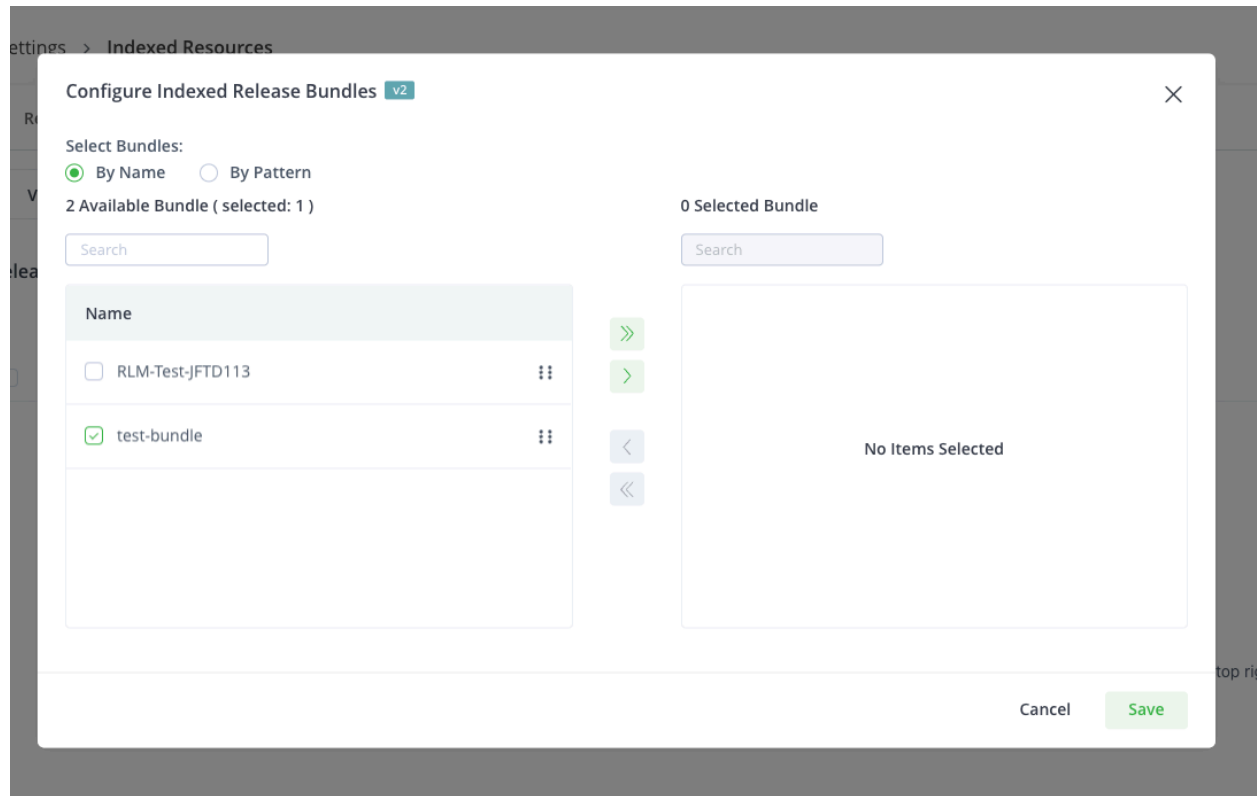
- First navigate Administration > Xray Settings > Indexed Resources



- Select 'V2' from Release Bundle type and then press the “Add a Release Bundle” button

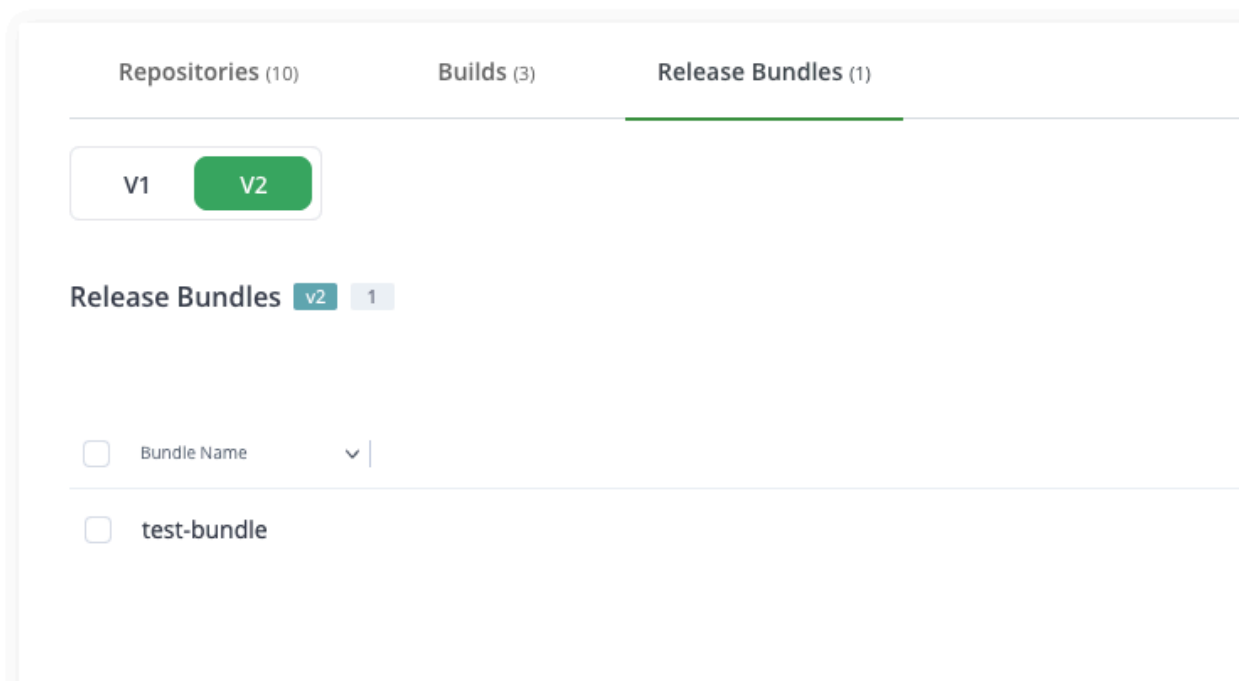


- Select the Release Bundle you want to apply Xray Policies to and click “Save”

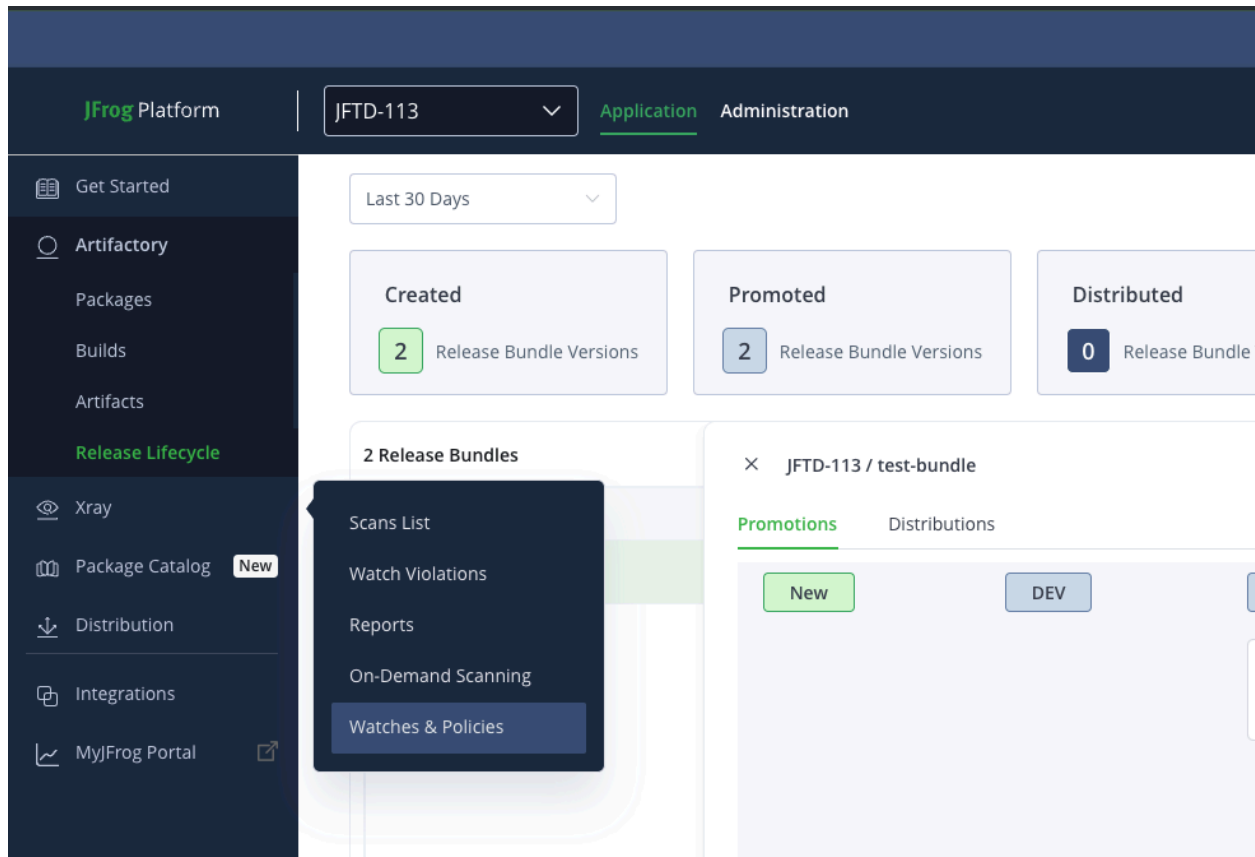


- The Release Bundles is now ready to create Xray Policies against

Xray Settings > Indexed Resources

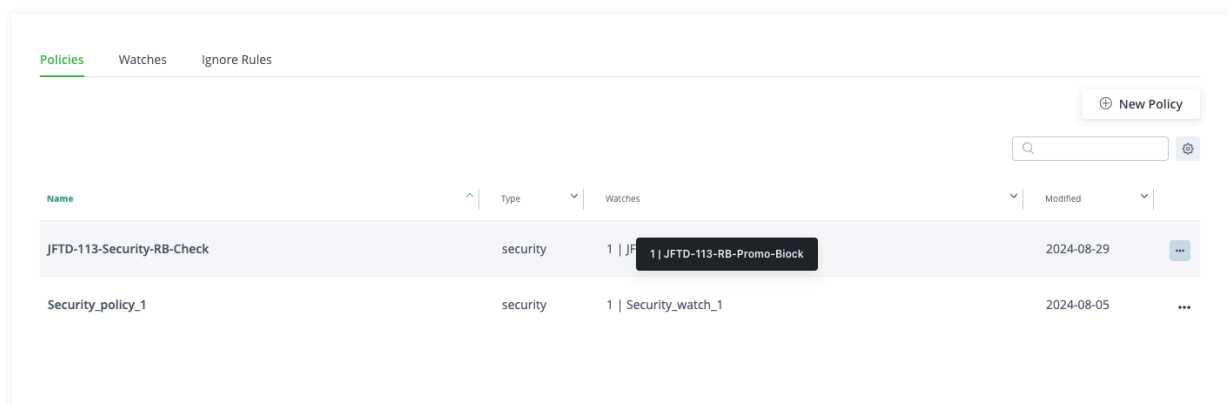


- Now Navigate to Application > Xray > Watches and Policies

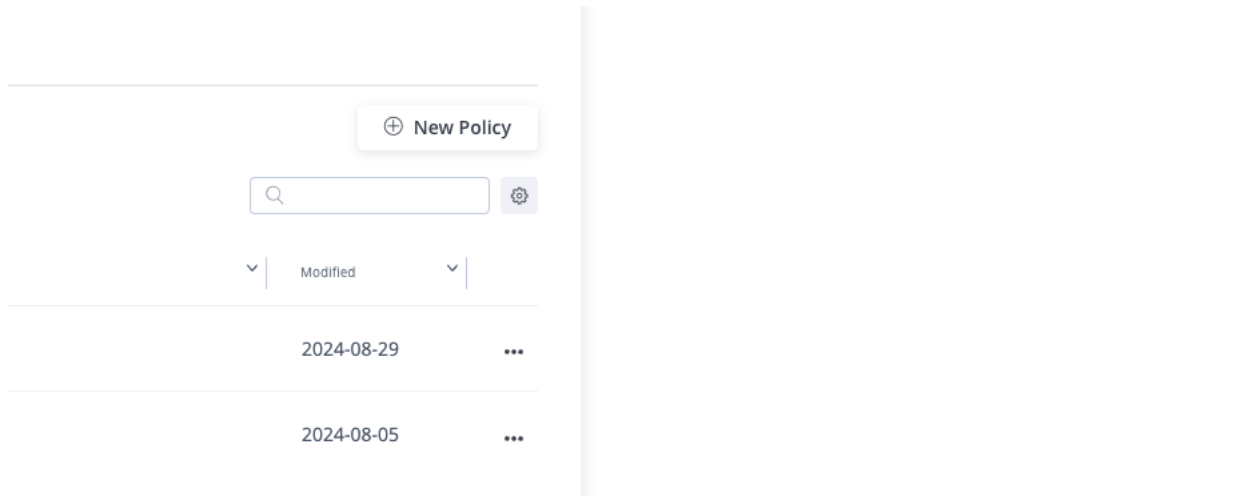


- You are going to create your [Xray Policy](#) and [Xray Watch](#) (for more detailed information please follow the links)

Xray > Watches & Policies



- Let's create 2 Policies first - these can be any of you choosing
 - One Security Policy
 - We will stop the Release Bundle Promotion on Critical CVEs
 - One License Policy
 - We will stop the Release Bundle Promotion if the "" License is found
- Start by clicking "New Policy" on the right handside



- You will be presented with the Create New Policy popup
 - Follow the instruction for this here: [Create an Xray Policy](#)

Xray > Watches & Policies > New Policy

1

Policy Details

* Policy Name

0/255

+ Add Description

(Optional)

Select Policy Type

☒ Security
 ☐ License
 ☐ Operational Risk

Next >

2

Policy Rules List

3

Apply on Scope

No selected watches

- One important part once you have defined your Policy Rules list based on type (either [Security](#), [License](#) or [Operational Risk](#)) you need define [Policy Violation Automatic Actions](#)

Create New Policy Rule

* Rule Name

Security-Critical

17/50

If The following condition is met

Rule type

CVEs

Rule category

☒ Minimal Severity

☐ CVSS Score

☐ CVE IDs

Select minimal severity

☒ Critical

Recommended severity - High

☐ Except if a Fix Version is not available

☐ Skip not applicable CVEs

Then Do the following actions

☒ Generate violation

Notify

☐ Trigger webhook

☐ Create Jira ticket

☐ Notify watch recipients

☐ Notify deployer

☐ Notify email

Block

☐ Fail Build

☐ Block download

☒ Block release bundle promotion

☒ Block release bundle distribution

Cancel

Save Rule

- We will focusing on the two following Automatic Actions

Block

☐ Fail Build

☐ Block download

☒ Block release bundle promotion

☒ Block release bundle distribution

- These will prevent either the Promotion or Distribution of a Release Bundle
- Once you have completed defining your Xray Policies, you need to apply them with [Xray Watch](#)
- Navigate back to Xray > Watches & Policies > Watches

Xray > Watches & Policies

Policies

Watches

Ignore Rules

+ New Watch

- Click “New Watch”

Name*

Description

Watch with no selected Policy/Policies will be automatically disabled

☒ Enabled

Watch Recipients

Jira Tickets

Profile Name Advanced Settings

All tickets will be sent to this selected profile

Manage Resources

Add Repositories

Add Builds

Add Bundles

Assigned Policies

- Follow the information for [Create a Watch](#)
- We are going to focus apply the Policy to a Specific Release Bundle

Configure Selected Release Bundles

V1 V2

Select By

☐ Any Bundle ☒ By Name ☐ By Pattern

1 Available Bundle

Name
<input type="checkbox"/> test-bundle

0 Selected Bundle

No Items Selected

Cancel Save By Name

- Select the Release Bundle you wish to apply to and click “Save by Name”

- If you don't see your Release Bundle make sure you select V2
- Now you should see your Release Bundle in the Watch list and the scroll up slightly

Xray > Watches & Policies > Create New Watch

- Now we are going to apply the Policy or Policies that were created earlier
 - Click the Manage Policies button

- You will see the Policies you created earlier and drag or use the arrows the ones you have selected from Left to Right and click “Save”

Choose Jira Profile
Advanced Settings

All tickets will be sent to this selected profile

Manage Resources

+
Add Repositories

+
Add Builds

Bundles

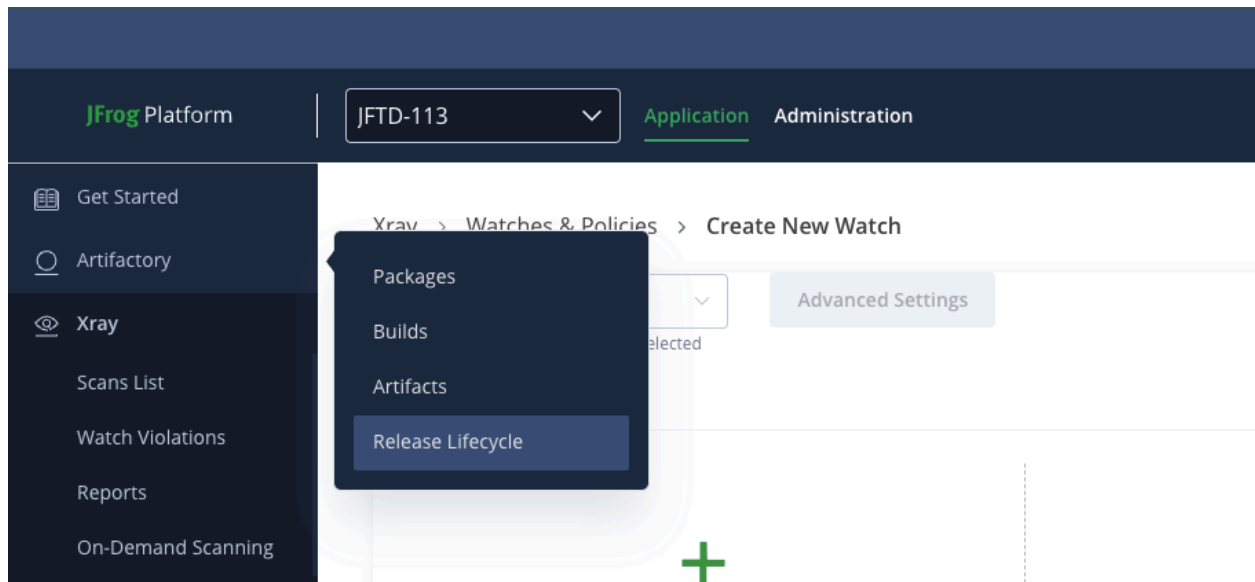
Bundles V2 (1)

Assigned Policies

Manage Policies

Name	Type	Author	Modified
Security_policy_1	security	trialadmin	2024-08-06T17:33:35.83Z

- Now click “Save” and we are ready to test
- Now navigate back to Artifactory > Release Lifecycle



JFrog Platform
JFTD-113
Application
Administration

Select a Plan
Request a Demo

Search Artifacts

User Icon
Help Icon
B

Last 30 Days
Create Release Bundle

Created
2 Release Bundle Versions

Promoted
2 Release Bundle Versions

Distributed
0 Release Bundle Versions

2 Release Bundles

Name	Project	No. of Versions	Latest Version
test-bundle	JFTD-113	1	1.0.1
RLM-Test-JFTD113	JFTD-113	1	1.0.1

- Select the Release Bundle that you created the Watches and Policies for and we are going to do the [Promote](#) step again but this time from wherever Environment it is currently located to a different Environment

New Promotion | Release Bundle Test-Bundle - 1.0.1

Promotion Environment | **Target Repositories**

Review and confirm the target repositories for this promotion

Package Type	Target Repositories
Maven	jftd113-stg-maven-local

Cancel Back Promote

- If your Xray Watches and Policies are triggered you will receive a notification that it can't promote but if it is promoted you will receive a notification that it was successful

Last 30 Days Create Release Bundle

Created 2 Release Bundle Versions Promoted 2 Release Bundle Versions Distributed 0 Release Bundle Versions

2 Release Bundles

Name
test-bundle
RLM-Test-JFTD113

JFTD-113 / test-bundle

Promotions Distributions

New DEV QA STAGING PROD

1.0.1
Sep 05, 2024

- Additionally, you can also navigate to Xray > Scan List > Release Bundles

Xray > Scans List

Repositories Builds **Release Bundles** Packages Add/Remove to Xray

V1 **V2**

Release Bundles **v2** 1

Release Bundle Name	Release Bundle Repo...	No. of Versions	Latest Version	Last Release Bundle Time	Created On	Created By	Config
[jftd113-release-b...	jftd113-release-b...	1	1.0.1	05 Sep 2024 21:36 (GMT-0700)	05 Sep 2024 21:36 (GMT-0700)	billm	

- Select the Release Bundle you wish to see the Xray scan results for and see all the information collected on an CVEs or other security information was associated with it

Xray > Scans List > [jftd113-release-bundles-v2]/test-bundle > 1.0.1

Scan Name: [jftd113-release-bundles-v...

Overview

- Policy Violations 0
- SBOM 2
- Security Issues 0
- Vulnerabilities 0**
- Malicious Packages 0
- Descendants
- Ancestors

1.0.1

Bundle Name	Created by	Last Scan
test-bundle	billm	05 Sep 2024 21:36 (GMT-0700)

Malicious Packages

Great News!
No Malicious Packages were found

Vulnerabilities

No Vulnerabilities were found

Policy Violations

No Violations were found

Software Components

View All

Most Common Types	Most Common Licenses	Components with most vulnerabilities
maven 1	Unknown 2	
releaseBundleV2 1		

Outcome:

- You have now done the following:
 - Have Repositories and Environments for your Release Bundle
 - Created a Release Bundle from 3 builds
 - Promoted a Repository from one Environment to another
 - Created a Xray Policy to act as a quality gate between Staging and Production

Lab 2: Evidence - JFrog CLI Examples

Purpose: The purpose of this lab is to write, using the JFrog CLI to add Evidence to a Release Bundle

Prerequisites:

- [The JFrog CLI](#) - we will be using this for our lab
- A Private Key which you will need to generate

Explanation:

- The JFrog CLI enables the creation of custom evidence, which is then deployed to Artifactory.
- JFrog CLI uses the following syntax for evidence:

```
jf evd create --key PRIVATE_KEY --key-alias CI-RSA-KEY --release-bundle  
BUNDLE_NAME --release-bundle-version VERSION --project PROJECT --predicate  
./policy.json --predicate-type https://jfrog.com/evidence/approval/v1
```

Process:

- First make sure that your [JFrog CLI](#) is configured for the JFrog Platform
 - [JFrog CLI Configuration](#)
- Find the Release Bundle you wish to attach the Evidence in the JFrog Platform UI
 - Navigate to Artifactory > Release Lifecycle
- Now open a terminal and run the command above and fill in the information