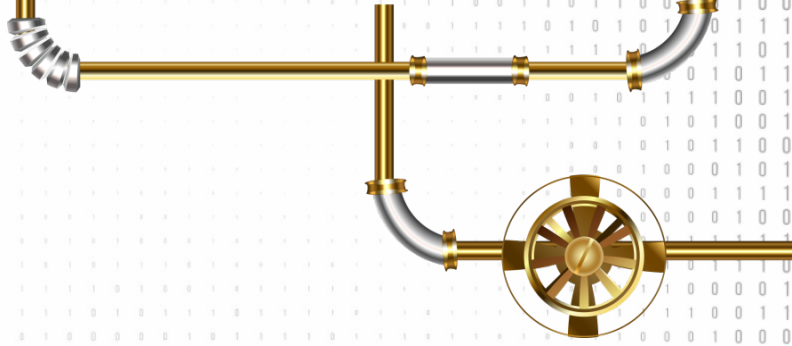


ADVANCE DEVOPS AUTOMATION WITH ARTIFACTORY

Stanley Fong, Solution Developer

SU-116 Advance Dev Ops Automation

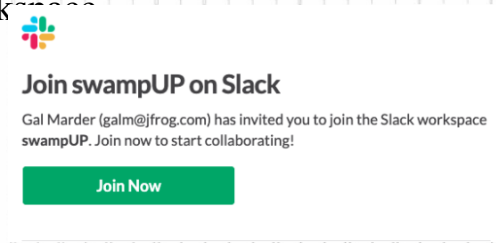
Setup Instructions

- Log into https://jfrog.orbitera.com/c2m/trial/869 and click create an account adding company and contact information
- Click “Create” then “Continue” and accept both agreements
- Click on “Launch Test Drive”
- Wait 9-10 minutes for the JFrog products to start
- You should see the environment log at the bottom
- The instructor will tell you which URL to click on the environment log to start your session
- All logins use admin/<password provided>
- Update your /etc/hosts – jfrog.local to the IP address from the Orbitera environment log (“C:\Windows\System32\Drivers\etc\hosts” on Windows)

```
------ Outputs ------
Username:
admin

Artifactory Edge (CN) URL:
http://jfrog.local:8090/artifactory/webapp

Artifactory (IN) URL:
http://jfrog.local:8092/artifactory/webapp

Xray URL:
http://jfrog.local:8000/

Jenkins URL:
http://jfrog.local:8086/

IP Address Mapping:
35.239.109.47

Artifactory (US) URL:
http://jfrog.local/artifactory/webapp

Mission Control URL:
http://jfrog.local:8080/

Wetty URL:
http://jfrog.local:3000/wetty

Artifactory Edge (EU) URL:
http://jfrog.local:8091/artifactory/webapp

Password:
m02OK3PBbv

Distribution URL:
http://jfrog.local:8083/
--------------------
>
```

slack Setup for Course Collaboration

- Check your email for an invitation to join a Slack workspace
- Click on the ‘Join Now’ icon
- Verify your name
- Create a password and click ‘Create Account’
- Agree to the ‘Slack Terms of Service’
- Workspace URL: swampup.slack.com



JFROG IN A NUTSHELL

2008
Founded

5,000
Customers

400
Employees

FORTUNE 100 COMPANY
Clients include
70%
of the
Fortune 100

$230M
Raised to date

All Hybrid
From OSS to Multicloud

6 Products
from Git to K8S

9 locations
7 countries

TECHNOLOGY LEADERSHIP

Forbes CLOUD 100 LIST

Deloitte 2018 Technology Fast 500 Winners

The 2018 SD Times 100 Award

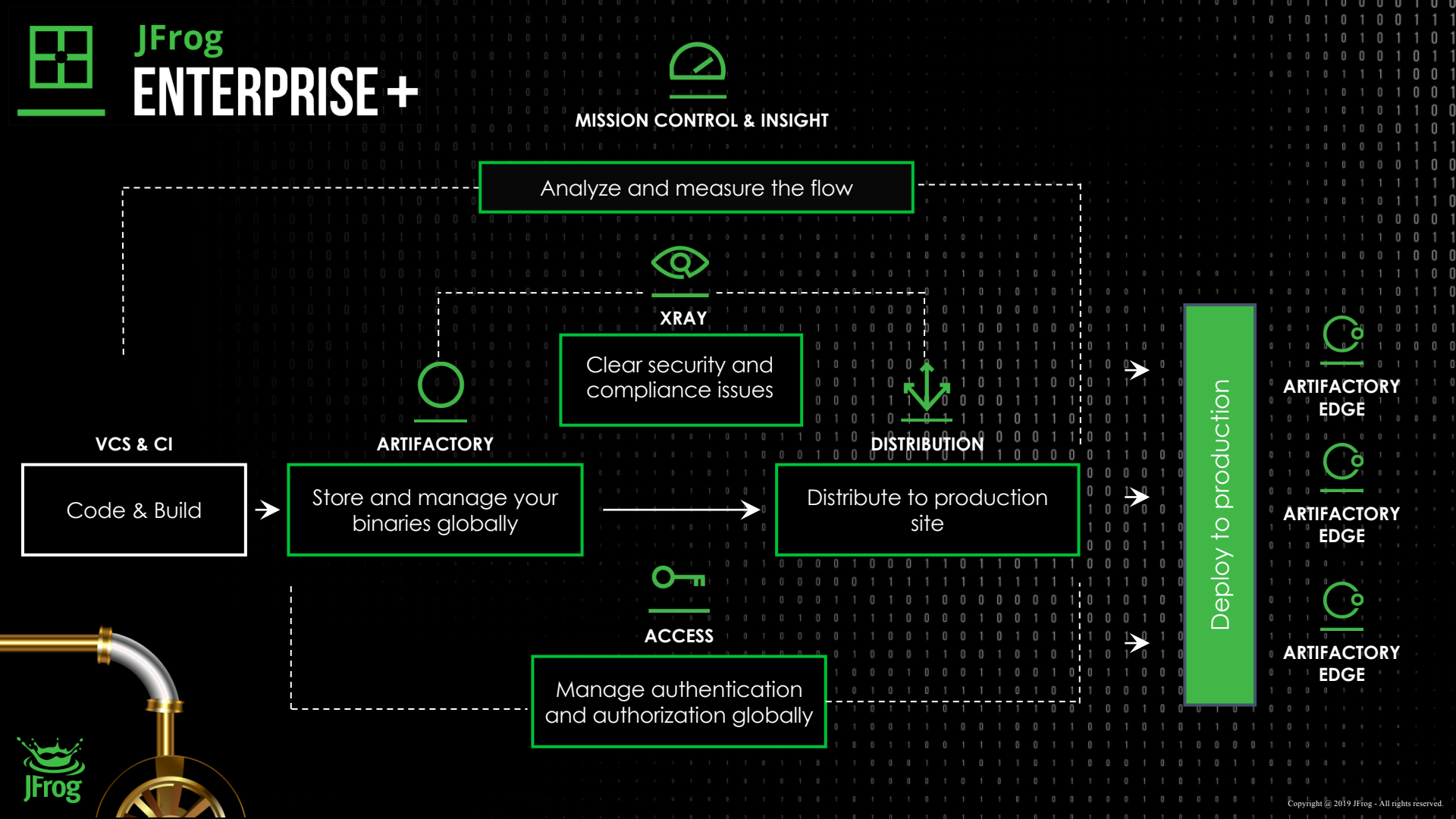
THE FROG PHILOSOPHY

HONORED TO LEAD

100 NEW LOGOS A MONTH

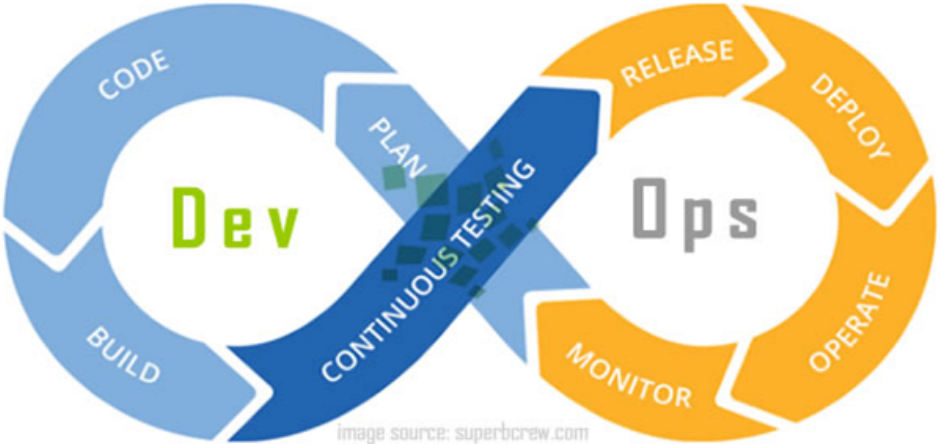
Internet & Software	Google, vmware, amazon, IBM, salesforce, ORACLE, YAHOO!, slack
Technology & Electronics	LinkedIn, intel, cisco, EMC², CHEF, box, ERICSSON
Banking & Finance	BARCLAYS, Fidelity INVESTMENTS, CHASE, citi, Bank of America, CREDIT SUISSE, WELLS FARGO
Engineering & Aerospace	ANSYS, GM, NASA, AIRBUS DEFENCE & SPACE, BOSCH, HALLIBURTON, GE, BOEING
Retail & Consumer	COSTCO WHOLESALE, NETFLIX, GAP OUTLET, VOLVO, macy's, Riot GAMES
Education & Research	Yale, OHIO UNIVERSITY, Stanford University, University of San Diego, OICR, MIT

JFrog



COURSE OBJECTIVE

Tools Introduction – Use Case: Automate delivery of an App from build to deployment ready to Kubernetes Environment



WHAT WILL LEARN

Main Focus: Automation Tools

- DevOps concept
- Intro to Artifactory Automation Tools
- Artifacts metadata - heart of DevOps Automation
- Automate N-tier application from build to deployment
- CI for official build + brief overview of Jenkins Artifactory Plugin
- Artifactory User Plugin

AGENDA

Modules

- Introduction – Liquid Software
- Design: DevOps Workflow Flow
- Module 3 - Tools + Exercises
- Lunch
- Module 4 - Automate Build to Deployment Ready
- Module 5 – Jenkins Pipeline
- Module 6 – Artifactory User Plugin
- Q&A

Teaching Assistants Introduction

Shay Bagants

Jason Gloege

Dan Dafna

HELLO

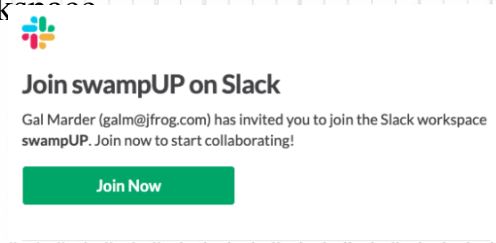
1. My name is …..
2. Company …..
3. How are you using Artifactory…
4. Take Away from this class ….

MODULE 1 - INTRODUCTION

SETUP YOUR ENVIRONMENT

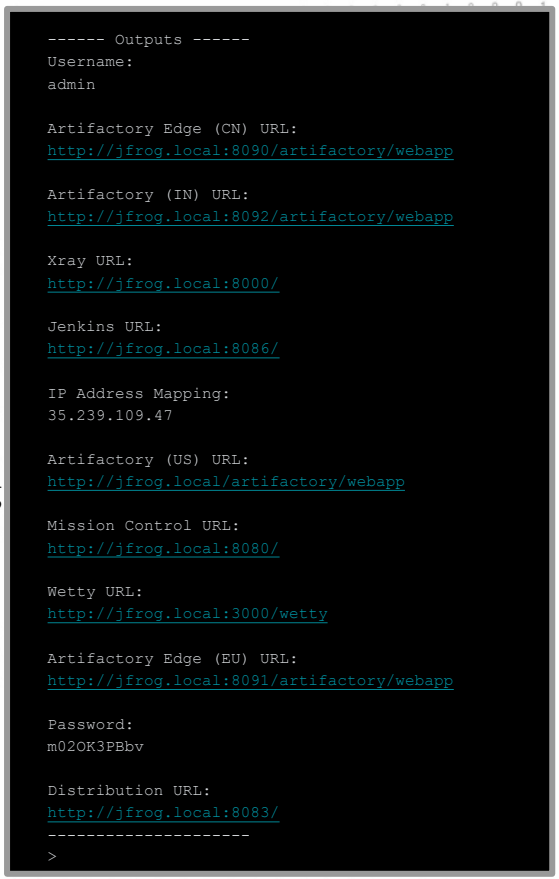
slack Setup for Course Collaboration

- Check your email for an invitation to join a Slack workspace
- Click on the ‘Join Now’ icon
- Verify your name
- Create a password and click ‘Create Account’
- Agree to the ‘Slack Terms of Service’
- Workspace URL: swampup.slack.com



Setup Instructions

- Log into https://jfrog.orbitera.com/c2m/trial/869 and click create an account adding company and contact information
- Click “Create” then “Continue” and accept both agreements
- Click on “Launch Test Drive”
- Wait 9-10 minutes for the JFrog products to start
- You should see the environment log at the bottom
- The instructor will tell you which URL to click on the environment log to start your session
- All logins use admin/<password provided>
- Update your /etc/hosts – jfrog.local to the IP address from the Orbitera environment log (“C:\Windows\System32\Drivers\etc\hosts” on Windows)



SU-116 Advance Automation Setup

- See instructions on the second monitor to setup your Orbitera Dev Environment for your exercises.
- Update /etc/hosts with jfrog.local and ip address.
- Check that you can ping the IP address and ssh there from your laptop
- If no SSH then launch http://jfrog.local:3000/wetty to get to SSH (Username: swampup | pwd: EkJ4NhJS1vjuFcj2fs)
- Clone the exercise to your local directory – if using wetty – it will be /home/swampup – git clone https://github.com/jfrogtraining/swampup2019.git
- cd su-116-advance-automation/
- Sign up for class slack channel
- You are all set …….

Understanding the Environment Details

How are my instances running?

- Each Application is running as a Docker container
- View all containers: “docker ps -a”
- View a specific container log: “docker logs ${CONTAINER_ID} -f”
- Stop an application (for example, restart Artifactory):

  > “docker stop ${CONTAINER_ID}” and to start: “docker start ${CONTAINER_ID}”
- View all aggregated logs from all Artifactory cluster containers:

  > “docker-compose -f /root/docker-compose.yml logs -f”

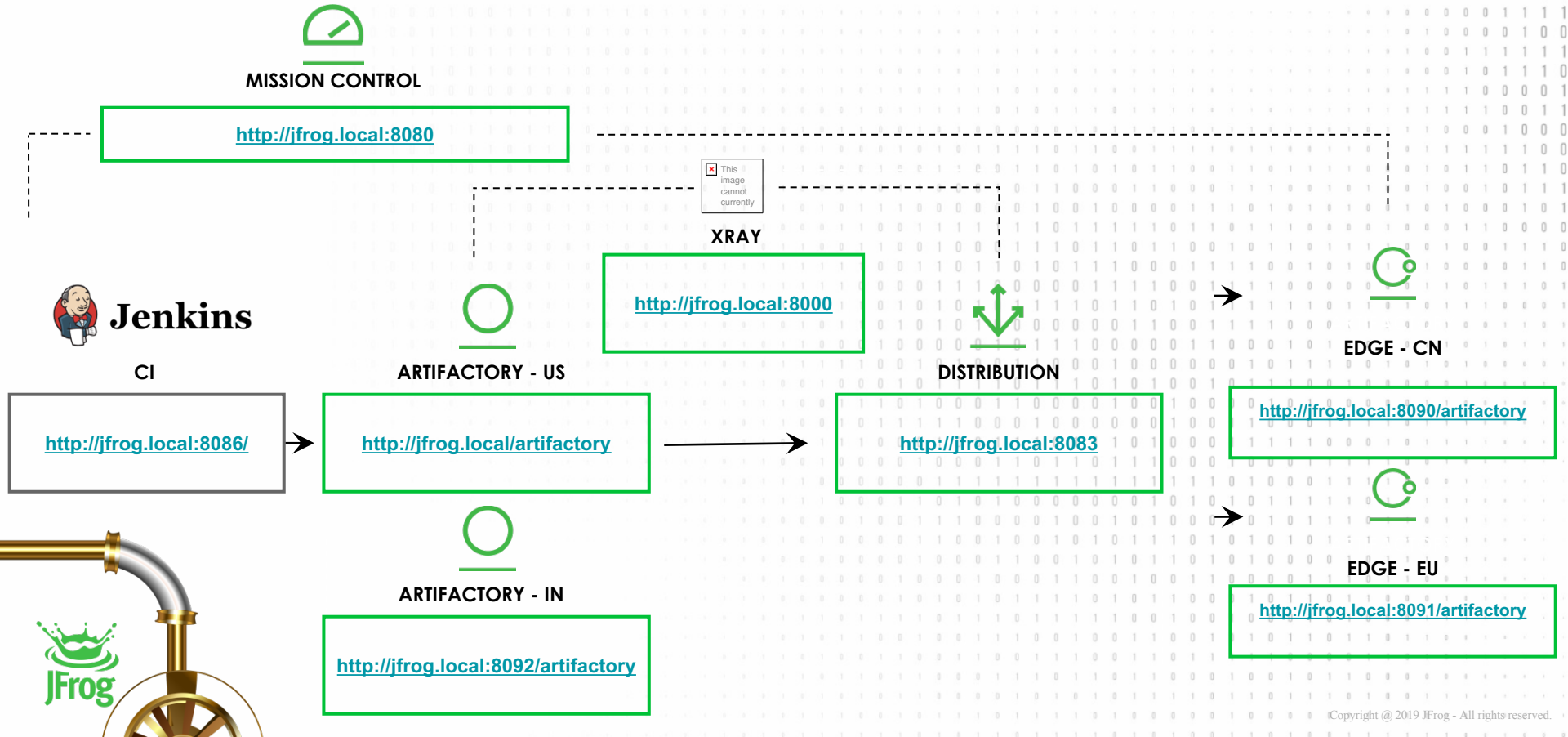
Where is my data?

- All Artifactory instances data: /jfrog/data/artifactory

How can I ssh this environment?

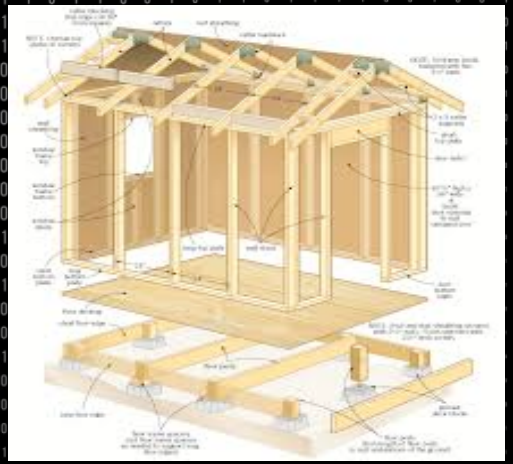
- Wetty ssh URL: http://jfrog.local:3000/wetty
- Username: swampup | pwd: EkJ4NhJS1vjuFcj2fs

The Environment Components



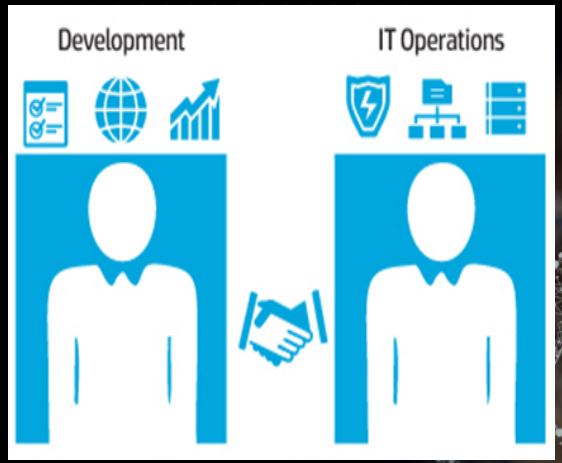
MODULE 2 – Design

DevOps Concept



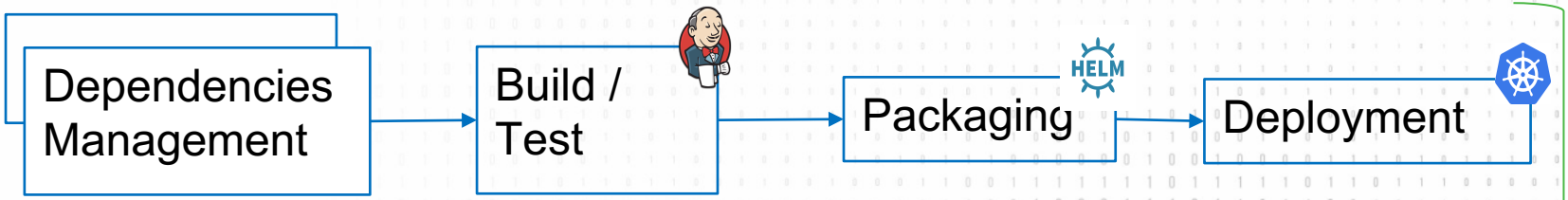
WHAT IS DEVOPS?

Collaboration between teams from development to deploying software.



CI / CD FLOW DISCUSSION

DevOps Design – Fill in the Requirements



RELEASE FAST

Software Developers

JFrog Artifactory

DevOps Design – My thoughts on Requirements

Dependencies Management

- No CVE
- Availability
- Maturity SDLC
- Immutability

Build / Test

- Immutable
- Repeatable
- Maturity SDLC
- Build Manifest File
- Promotion
- Stats
- Security Vulnerabilities
- Notification
- Dashboards

Packaging

- Multiple Targets
- Approved Artifacts.
- Packing Manifest

Deployment

RELEASE FAST

Software Developers

- My own dev env with dependencies
- I did not make any changes

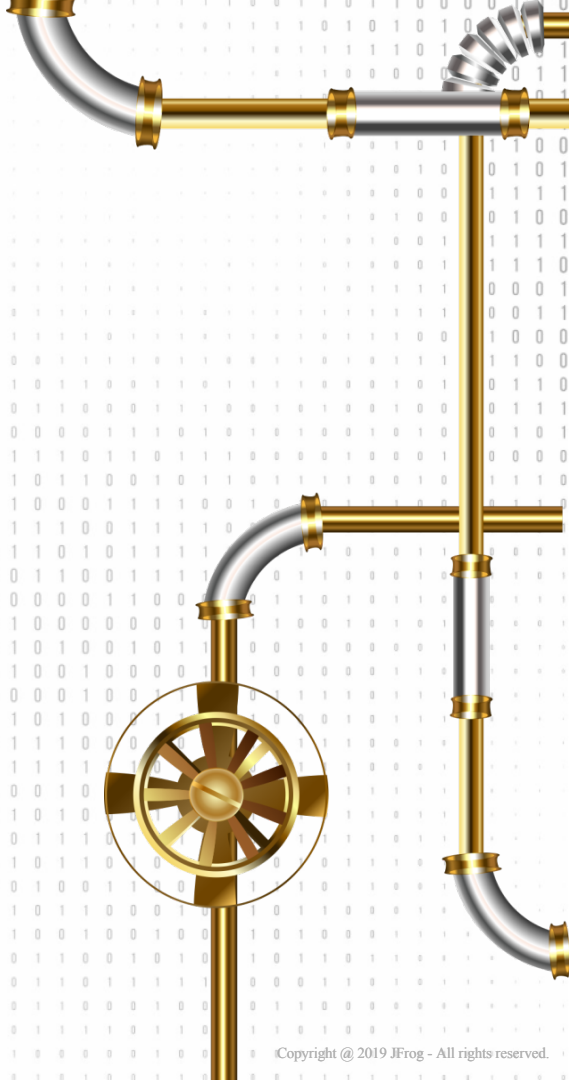
JFrog Artifactory

JFrog

PRIORITIZE DEVOPS AUTOMATION

DevOps Automation = Continuous Monitoring and Improvements

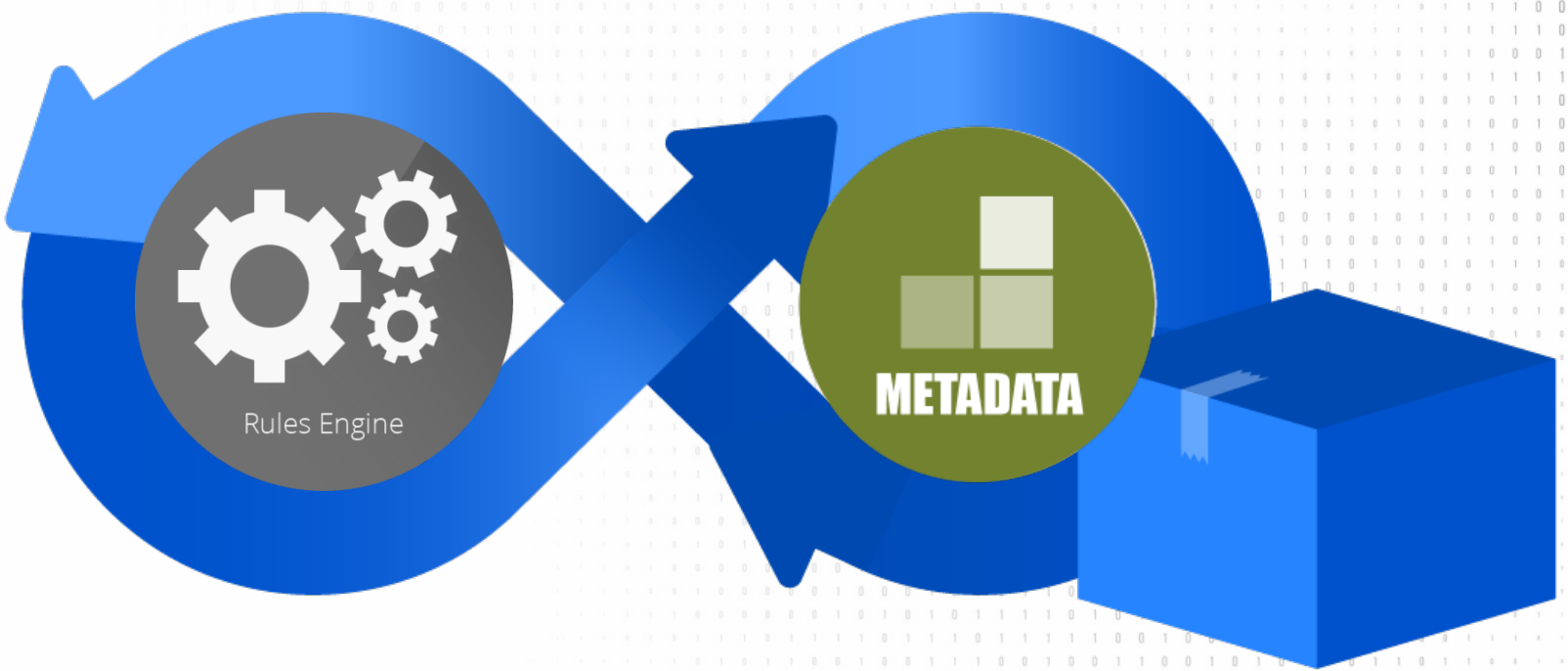
- Let the needs drive automation
- Identify your bottlenecks
- What is your ROI
- Continuously Assess and Reevaluate
- Identify your stakeholders
- **Dashboard** is your Friend and a very POWERFUL Tool.



JFrog

Our Dashboard

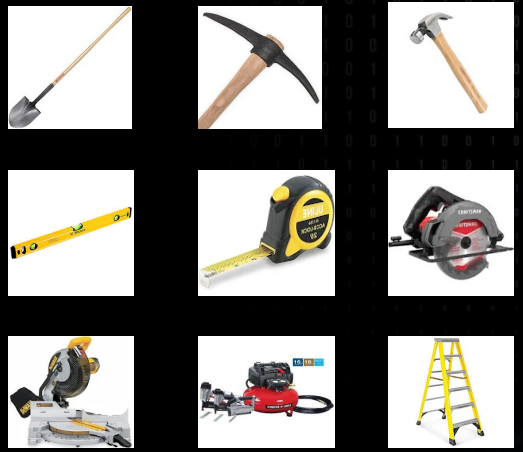
AUTOMATION IS ITERATIVE



MODULE 3 – Foundation

Tools

Foundation – Use of Tools



Artifactory & Reporting Automation Tools

Dependencies Management
Upload & Download

Build / Test & Packaging
Vulnerability Scan & Deployment

SDLC Dashboard & Notification

Administrative

Module 3a - Administrative Artifactory Repository

Exercise:
Introduction to Artifactory REST API and to create user, API Keys and repositories needed for the rest of the course

ARTIFACTORY REST API RESOURCES

Implements all Artifactory UI functionalities

- Builds
- Artifacts & Storage
- Searches
- Security
- Repositories
- System & Configuration
- Plugins
- Import & Export
- Support

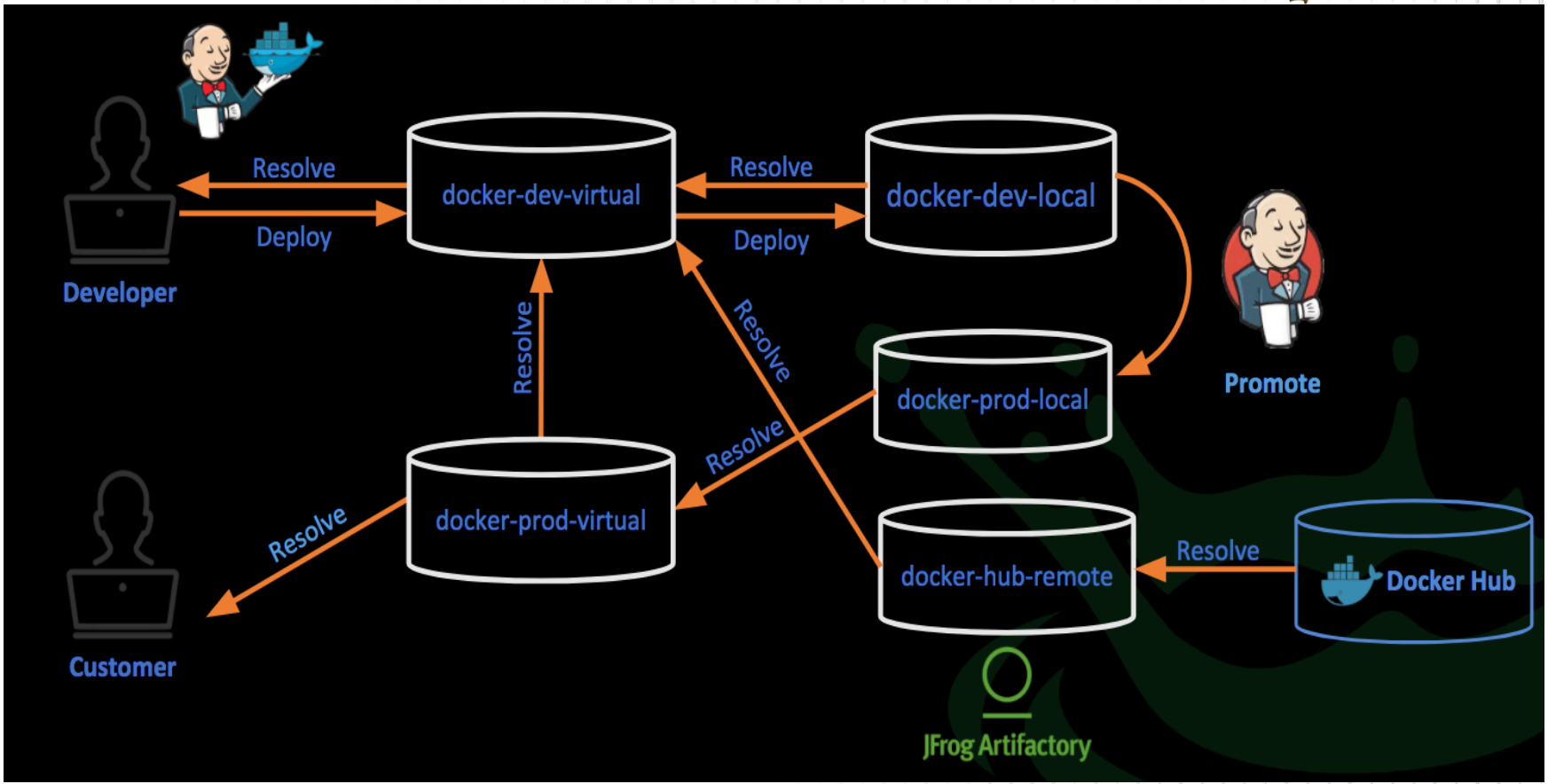
Artifactory API References

1. Artifactory Java Client Library - https://github.com/jfrog/artifactory-client-java

2. Artifactory REST API - https://www.jfrog.com/confluence/display/RTF/Artifactory+REST+API

3. Artifactory Public API 6.10.3 - http://repo.jfrog.org/artifactory/oss-releases-local/org/artifactory/artifactory-papi/%5BRELEASE%5D/artifactory-papi-%5BRELEASE%5D-javadoc.jar!/overview-summary.html

ARTIFACTORY VIRTUAL REPOSITORIES



MODULE 3a – REST API EXERCISE

Exposure to Artifactory REST API to:

- Create Gradle, Maven, Helm, NPM, and Docker Local and Virtual for dev, staging and release
- Retrieve your API Key
- Create users
- **Add jcenter as remote repository to the repo.yaml file. It is used later**

We will visit both bash version and Artifactory Java Client .version.

Reference URLs to be posted on su-116 slack channel

What did you notice about the API Keys from both Artifactory Instances?

Module 3a - Lab

- cd swampup2019/su-116-advance-automation/module3/bash/firstexercisecopyme.sh
  - Copy firstexercisecopyme.sh and make changes to create users and repositories on your local Orbitera artifactory US.
  - Create users, APIKeys and Repositories.
  - Verify on Artifactory US.

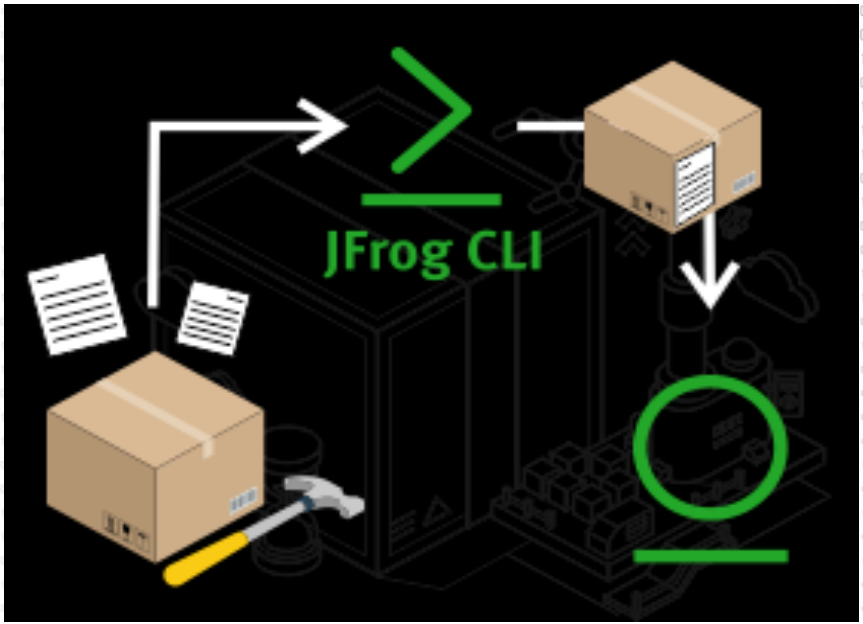
- cd swampup2019/su-116-advance-automation/module3/groovy
  - Change ArtifactoryLib.groovy library to create users, APIKeys and Repositories.
  - Change build.gradle with your Orbitera Environment.
  - Change repo.config to location of the repo.yml file.
  - Run ./gradlew to execute code

MODULE 3b – Dependencies Management

Introduction to JFROG CLI to download and upload binaries

JFROG COMMAND LINE RESOURCES

- Parallel Upload and Download
- Checksum Optimization
- Wildcard & Regular Expression
- Upload Dry Run for upload
- Build Integration
- FileSpec Support
- Metadata Support
- Artifactory REST API



JFROG CLI

Download, Upload Commands etc.

- >jfrog rt c – configure Artifactories
- **>jfrog rt curl – general curl command (new)**
- >jfrog rt ping
- >jfrog rt dl – download
- >jfrog rt u – upload
- >jfrog rt cp – copy
- >jfrog rt mv – move
- >jfrog rt del – delete
- >jfrog rt s – search
- >jfrog rt sp – set properties
- >jfrog rt delp – delete properties

Common Options:

- **--server-id**
- **--spec**
- **--flat**
- **--props**
- **--dry-run**

Reference: https://www.jfrog.com/confluence/display/CLI/CLI+for+JFrog+Artifactory

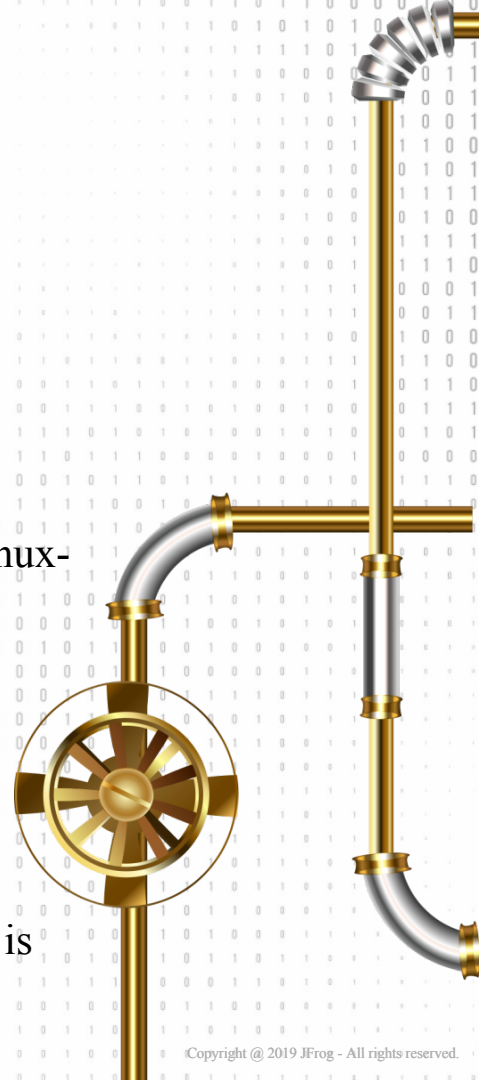
Module 3b – JFrog CLI Download exercise

Download from IN Artifactory (jfrog.local:8092) to your local workspace –

Requirements – this will be used in later builds. Script out in bash

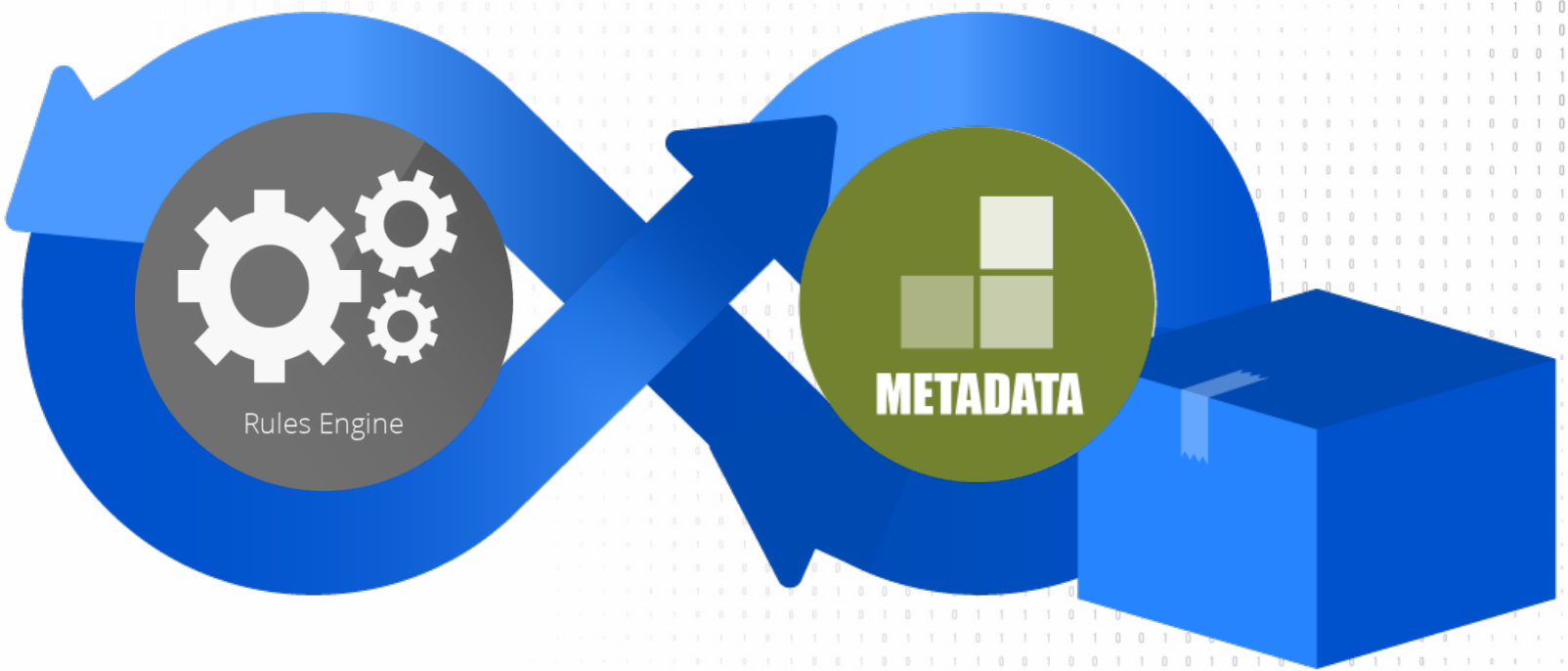
- tomcat-local/org/apache/apache-tomcat/apache-tomcat to **local** ./tomcat/apache-tomcat-8.tar.gz
- tomcat-local/java/jdk-8u91-linux-x64.tar.gz to **local** ./jdk/jdk-8-linux-x64.tar.gz directory
- Download only if properties: swampup2019: ready exits.
- Script: merge **m3b-jfrogcli.sh** to your main script from 3a.

- Do this manually first. Then automate using bash script.
- If you run the above jfrog cli command twice what happens? And what is the API Key for both US and IN?



FILESPEC, METADATA

Objective: Introduction to defining your rules and metadata



Module 3c - FILESPEC

Objective: Getting files from here to there

- Download Dependencies resolutions
- Download artifacts based on software life
- cycle maturity (properties)
- Upload Third Party software to Artifactory
- Separation of File Specification from code
- Integrates with AQL
- Wildcard or regular expression with placeholders

FILESPEC FORMAT

JSON format

```
{
 "files": [
   {
     "pattern" or "aql": "[Mandatory]",
     "target": "[Optional, Default: ./]",
     "props": "[Optional]",
     "recursive": "[Optional, Default: true]",
     "flat": "[Optional, Default: false]",
     "build": "[Optional]",
     "explode": "[Optional, Default: false]",
     "excludePatterns": ["[Optional]"]
   }
 ]
}
```

Clients:

- JFROG CLI
- Jenkins, Bamboo, TeamCity Artifactory Plugins

Reference

- https://www.jfrog.com/confluence/display/RTF/Using+File+Specs

WHY METADATA?

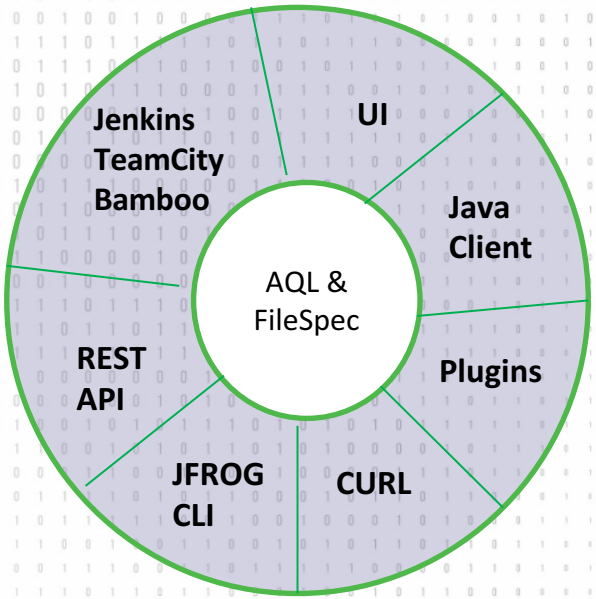
ARTIFACTORY PROPERTIES

Metadata

7 Properties

Filter by Property

Property	Value(s)
build.name	maven-example
build.number	100
build.timestamp	1523164791918
qa-team	platform,ui
release-bundle	true
unit-test	pass
vcs.revision	a7d8959aed02b2ff0722ca7fd56a941b841516f5



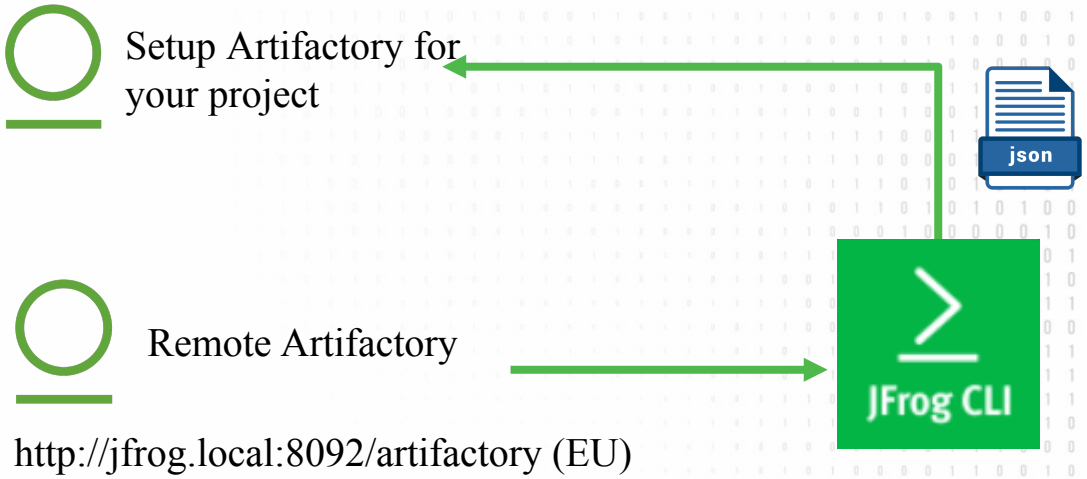
Module 3c – FILESPEC Exercise

Objective: Getting files from here to there

- Automate to use Jfrog CLI + File Spec to upload files to Artifactory Orbitera and set the properties with your id while you are at it; the property "**submitter=<your name>**"
- We will be using FileSpec for Dependencies Management in the afternoon exercises.
- Use Bash for now.

MODULE 3c – EXERCISE FILESPEC-UPLOAD ARTIFACTS

Objective: FileSpec to upload binaries to local Artifactory



Connect to Remote Artifactory & Download Tomcat, Apache, Helm for local development,

- **upload to your artifactory US instance, repo: tomcat-local using FileSpec**
- **Properties added when uploading: (source=**es-site**; approved=** pending**; submitter=**your name**)**

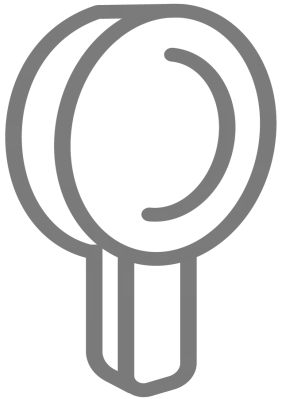
Script: (1) merge m3c-filespec.sh to your main script from 3b. (2) Change resources/swampupfilespecUpload.json on with properties – verify artifacts uploaded to artifactory US

MODULE 3d – AQL

Introduction to JFROG CLI to download and upload binaries with complex queries

Module 3d - AQL DESIGNED TO:

- Very powerful, fast and simple way to formulate complex queries that specifies any number of search criteria, filters, sorting options and output parameters.
- Runs on local, remote caches and virtual repositories.
- AQL is executed from Artifactory REST API and clients supporting Filespec.

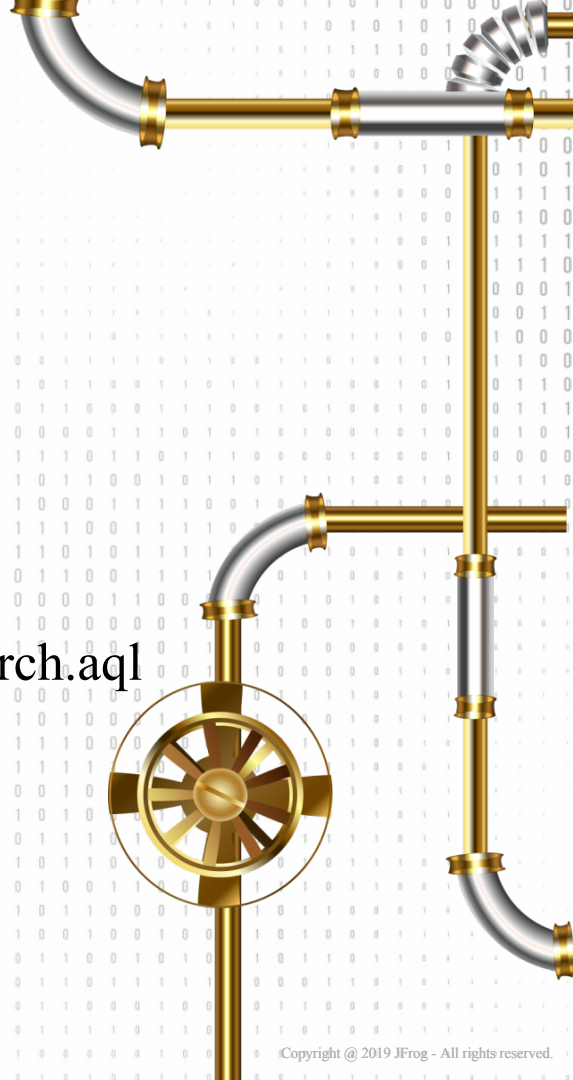


AQL GETTING STARTED

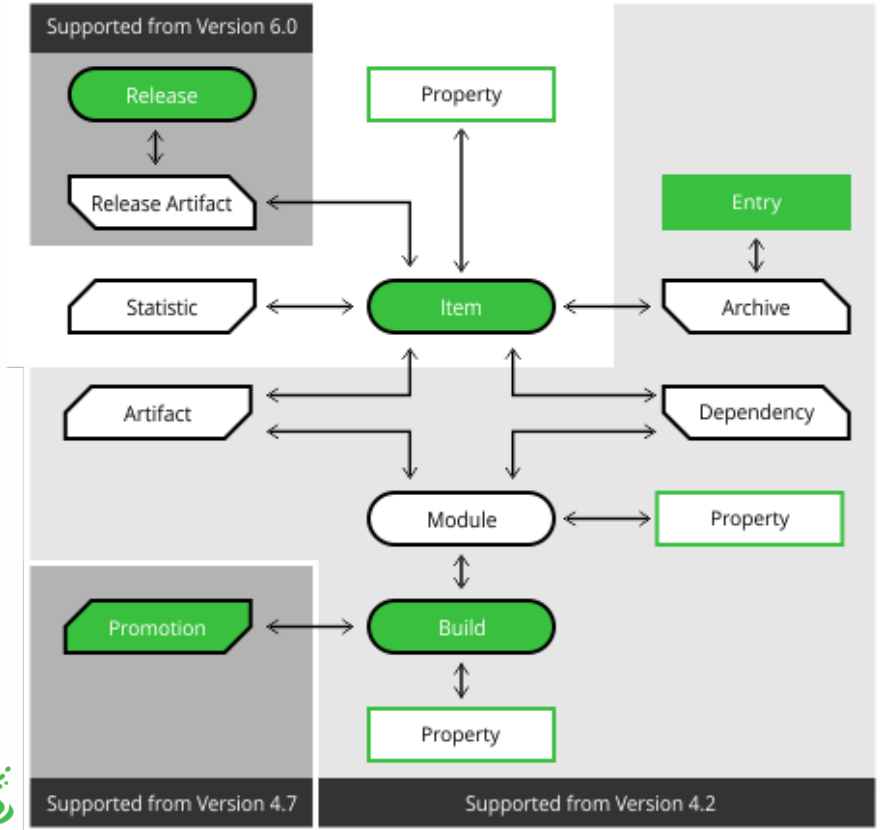
It takes time to develop the expertise – learn by example

- Start small and incrementally add criteria
- Create an query file and use curl to fine tune AQL Query
- Use Artifactory UI Search to auto generate AQL
- Use JFROG CLI to auto generate AQL

curl …. –X POST <artifactory url>/api/search/aql –T search.aql



AQL ARCHITECTURE:



Method Chaining

<domain>.**find**(<criteria>)
.**include**(<fields>)
.**sort**(<order and fields>)
.**limit**(<number of records>)
.**offset**(<offset records>)

Accessing Property

{"@<property_key>":
{"operator":"<proper
ty_value>"}}

Domain :

1. **Items**
2. Items.Property
3. Statistic
4. Build
5. Build.Property
6. **archive.entries**
7. Build.promotion
8. Artifact
9. Dependency
10. Module
11. Release

PROPERTY CRITERIA

{"@<property_key>":{"operator":"<property_value>"}} $msp (match on single property)

Task	AQL
Find all items approved by QA	items.find({"@qa_approved" : {"$eq" : "true"}})
Find all items build on a Linux machine.	items.find({"artifact.module.build.@os" : {"$match" : "linux*"}})
Two items A with AGPL-V3 and items B with both LGPL-2.1 and LGPL2.2	items.find({ "@license":{"$match": "*GPL*"}, "@license":{"$nmatch": "LGPL-2.1*"} }) Returns A items.find({ "$msp": ["@license":{"$match": "*GPL*"}, "@license":{"$nmatch": "LGPL-2.1*"}]}). Returns A & B

Search Artifacts

AQL – AUTO-GENERATE FROM ARTIFACTORY

Search Type

Package

docker-app

latest

Add search criteria...

Clear

Search

Search Results - 2 Items

View Clear

1 Items

Filter by Image

AQL

Delete

Page 1 of 1

```
curl -H 'Content-Type:text/plain' -H 'X-JFrog-Art-Api: AKCp5aTbbdw7gCfd7hWFp8fUCd9U7Bey3EJafmQS9CSd2ZfW3YZUAzsTauJhkTiDeuq7WQKW4' -X POST https://artifactory-solutions-us.jfrogbeta.com:443/artifactory/api/search/aql -d '
items.find(
  {
    "$and":[
      {"$or":[
        {"$and":[
          {"name":{"$eq":"manifest.json"}},
          {"$or":[
            {"@docker.repoName":{"$match":"docker-app*"}},
            {"@docker.repoName":{"$match":"library/docker-app*"}}
          ]}
        ]}
```

JFROG CLI - AUTO GENERATE AQL

```
>set JFROG_CLI_LOG_LEVEL=DEBUG
>jfrog rt s generic-local/*helm*
items.find({"repo": "generic-local","$or": [{"$and":[{"path": {"$match":
"."},"name": {"$match": "*helm*"}}]},{"$and":[{"path": {"$match":
"*"},"name": {"$match": "*helm*"}}]},{"$and":[{"path": {"$match":
"*helm*"},"name": {"$match":
"*"}}]}]}).include("name","repo","path","actual_md5","actual_sha1","siz
e","type","property")
```

AQL Example:1 – Find where deployed

Where is **latest** docker-app deployed – What is field names are required to make this happen? What is the domain to use?

Method Chaining

```
<domain>.find(<criteria>)
 .include(<fields>)
 .sort(<order and fields>)
 .limit(<number of records>)
 .offset(<offset records>)
```

Field Names
• repo
• path
• name
• created
• modified
• updated
• created_by
• modified_by
• type
• depth
• original_md5
• actual_md5
• original_sha1
• actual_sha1
• sha256
• size
• virtual_repos

AQL Example – Find where deployed

Where is latest docker-app deployed – What is required to make this happen? What is the domain to use?

```
items.find (
        {"repo":"docker-prod-local",
          "@deployed" :{"$eq":"*"}
        })
        .include("created", "@deployed")
        .sort ({"$desc":["created"]})
        .limit(1)
```

Method Chaining

```
<domain>.find(<criteria>)
 .include(<fields>)
 .sort(<order and fields>)
 .limit(<number of records>)
 .offset(<offset records>)
```

Field Names
• repo
• path
• name
• created
• modified
• updated
• created_by
• modified_by
• type
• depth
• original_md5
• actual_md5
• original_sha1
• actual_sha1
• sha256
• size
• virtual_repos

AQL Example:2 – Find latest

Find the latest webservices.war file from build “gradle-example” – Useful for dependency management.

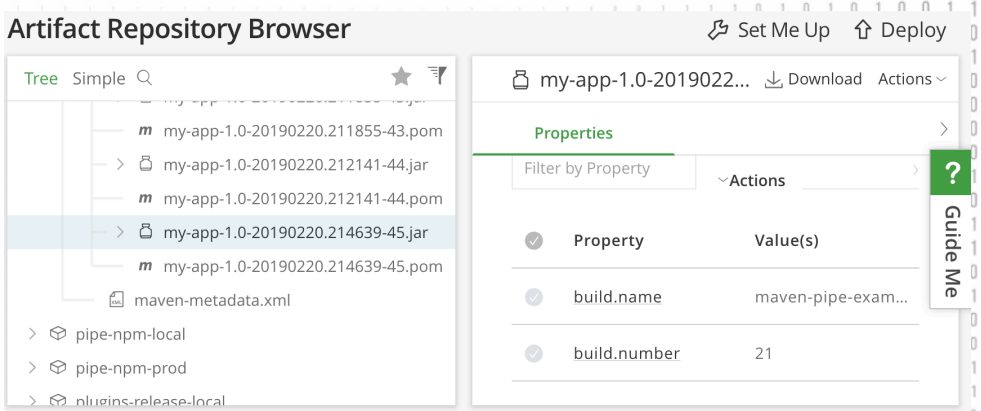
What is the domain name?

How to get to the build name – “gradle-example”

Field Names
url
name
number
created
created_by
modified
modified_by

AQL Example – Find latest

Find the latest webservices.war file from build “gradle-example” – Useful for dependency management

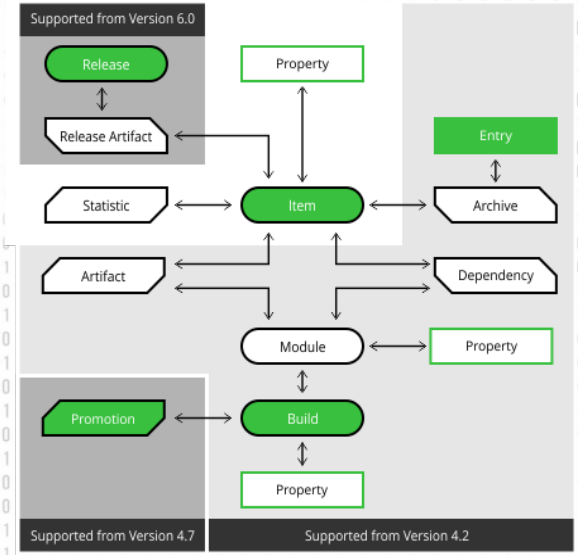



```
items.find ({
        "repo":{"$eq":"gradle-release-local"},
        "name":{"$match":"webservice-*.war"},
        "@build.name":"gradle-example”})
.include("created","path","name")
.sort({"$desc" : ["created"]})
.limit(1)
```

Field Names
url
name
number
created
created_by
modified
modified_by

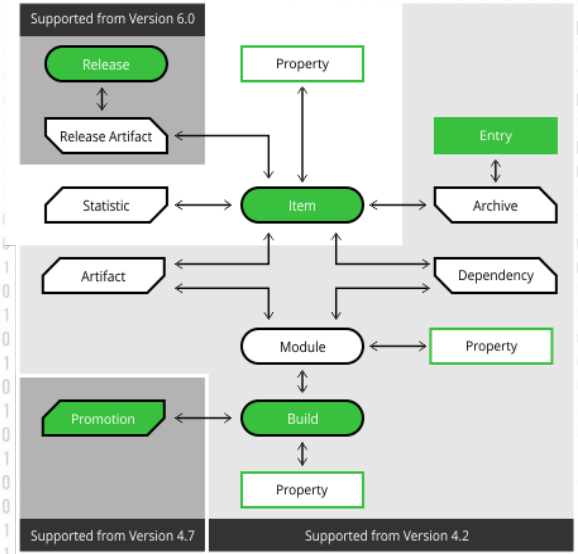
AQL Examples:3 – Where is “.class”?

Return all entries of any archive named ”common-io-2.4.jar” with ”FalseFileFilter.class” from any build named ”maven-example-build" with build number 521.



AQL Examples – Where is “.class”?

Return all entries of any archive named ”common-io-2.4.jar” with ”FalseFileFilter.class” from any build named ”maven-example-build" with build number 521. What is the domain?




```
archive.entries.find( {
  "archive.item.name":{"$eq":”commons-io-2.4.jar"},
  ”archive.entry.name”:{“$eq”:”FalseFileFilter.class”},
  "archive.item.artifact.module.build.name":{"$eq":”maven-example-build"},
  "archive.item.artifact.module.build.number":{"$eq":"521"}})
```

COMPOUND CRITERIA

<criterion>={<"$and"|"$or">:[{<criterion>},{<criterion>}] default is $and

Task	AQL
Find all items in gradle-local repo that have LGPL license.	items.find({"repo" : ”gradle-local"},{"@artifactory.licenses" : {"$match" : "*LGPL*"}})
Find all *.deb from my_debian_build or *.rpm from my_yum_build	items.find({"$or":[{"$and”: [{"artifact.module.build.name" : "my_debian_build"} , {"name" : {"$match" : "*.deb"}}] }, {"$and": [{"artifact.module.build.name" : "my_yum_build"} , {"name" : {"$match" : "*.rpm"}}]}]})

Exercise 3d-a: Find by properties.

File: resource/submitter.aql

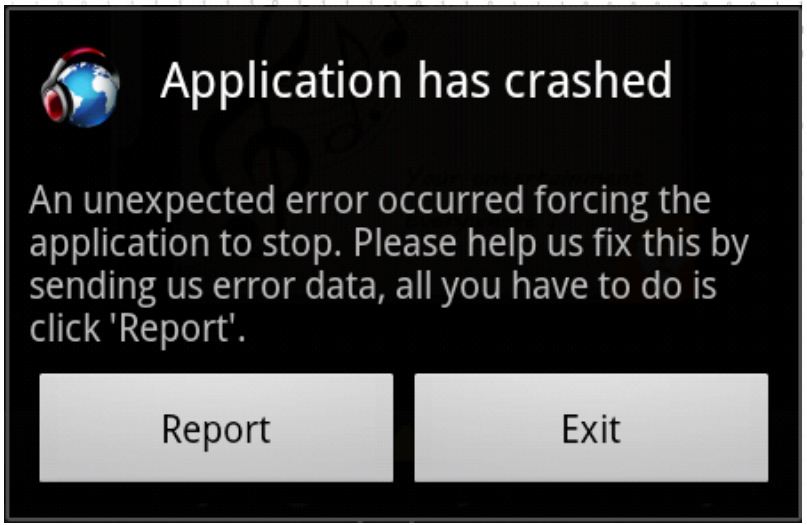
Find your artifacts that you have uploaded with your name – property is (**submitter**=your name**) from exercise module3c**

- Merge m3d-aql.sh to your script.
- Modify resource/submitter.aql to search for your name.
- Verify the output – should be artifacts that you uploaded previously.

EXERCISE 3d-b: IMPLEMENT AQL QUERY

File: resource/junitfilter.aql

You have just been informed that “junit-4.11.jar” class library has a serious null pointer exception issue. You must immediately find all products that have this issue and have development build with latest remediation.



EXERCISE 3d-c:
FIND LATEST DOCKER APP BUILD

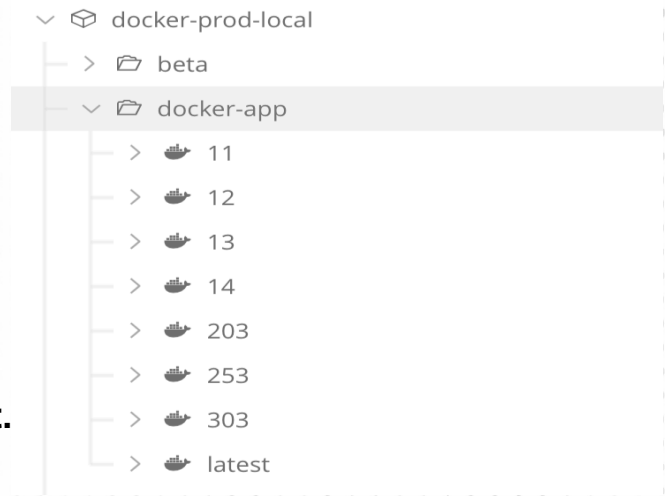
This method will work for any package type

Write a function to print the latest docker-app tag. i.e. Latest Tag: 303; Do not include ”latest” because it is not immutable

```
<domain>.find(<criteria>)
        .include(<fields>)
        .sort(<order and fields>)
        .limit(<number of records>)
```

- **Copy m3d-c-dockerlatest.sh to your main script.**

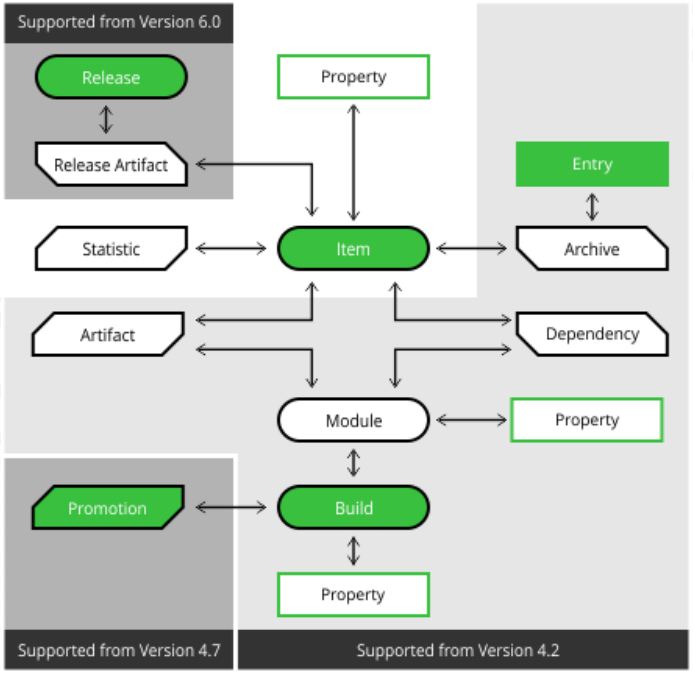
- ”repo” : ”docker-prod-local”
- “type” : ”folder”
- Docker image : docker-app



Exercise 3d-d: CLEAN UP EXAMPLE

File: cleanup.aql

- Find top 100 files with the following attributes and **list the build name and build number** that produces them.
  - Size greater than 3M
  - Created by Jenkins
  - Less than 10 downloads
  - Extensions is either jar or war files.



- **Modify resource/cleanup.aql**
- **Enable your main script to run this AQL query.**

DOWNLOAD & UPLOAD TOOLS

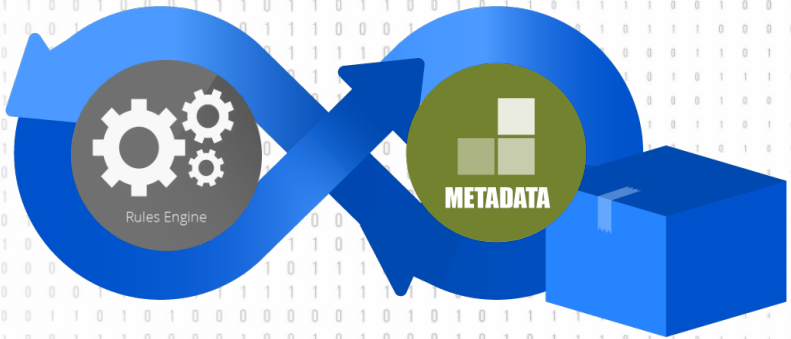
Choose as appropriate …..

Method	Application
REST API	Full flexibility with custom code. Recommend to use if any of the below does not suffice
Jfrog cli	can achieve query with one CLI command
FileSpec	Aggregate multiple CLI commands in one FileSpec
AQL	Formulate complex queries that specify any number of search criteria, filters, sorting options and output parameters
Artifactory Java Client	Integrates with Java and Groovy language. Full flexibility with custom code. Recommend to use if shell access to JFrog CLI is not available.

RECAP

Very High Level Overview – Opportunities are unlimited

- Using REST API and YAML to provision Artifactory for new projects
- JFROG CLI to upload and download build dependencies
- FileSpec to separate logic from file specifications
- AQL for complex queries
- Metadata to continuously improve your software development life cycle



MODULE 4 – BRINGING IT ALL TOGETHER

Build and Deploy multi-tier web application CI server agnostic using JFrog CLI, FileSpec, AQL and metadata

NEXT STEPS

Lets build an application using **JFROG CLI, Jenkins Artifactory Plugin**

BUILD INTEGRATION

Build Information deployed to Artifactory (Buildinfo)

- Published artifacts and corresponding dependencies
- Build Environment
- Supports bulk operations – move, copy based on buildinfo
- Bidirectional links between builds and artifacts.
- Fully reproducible builds through visibility of artifacts deployed, dependencies and information on the build environment

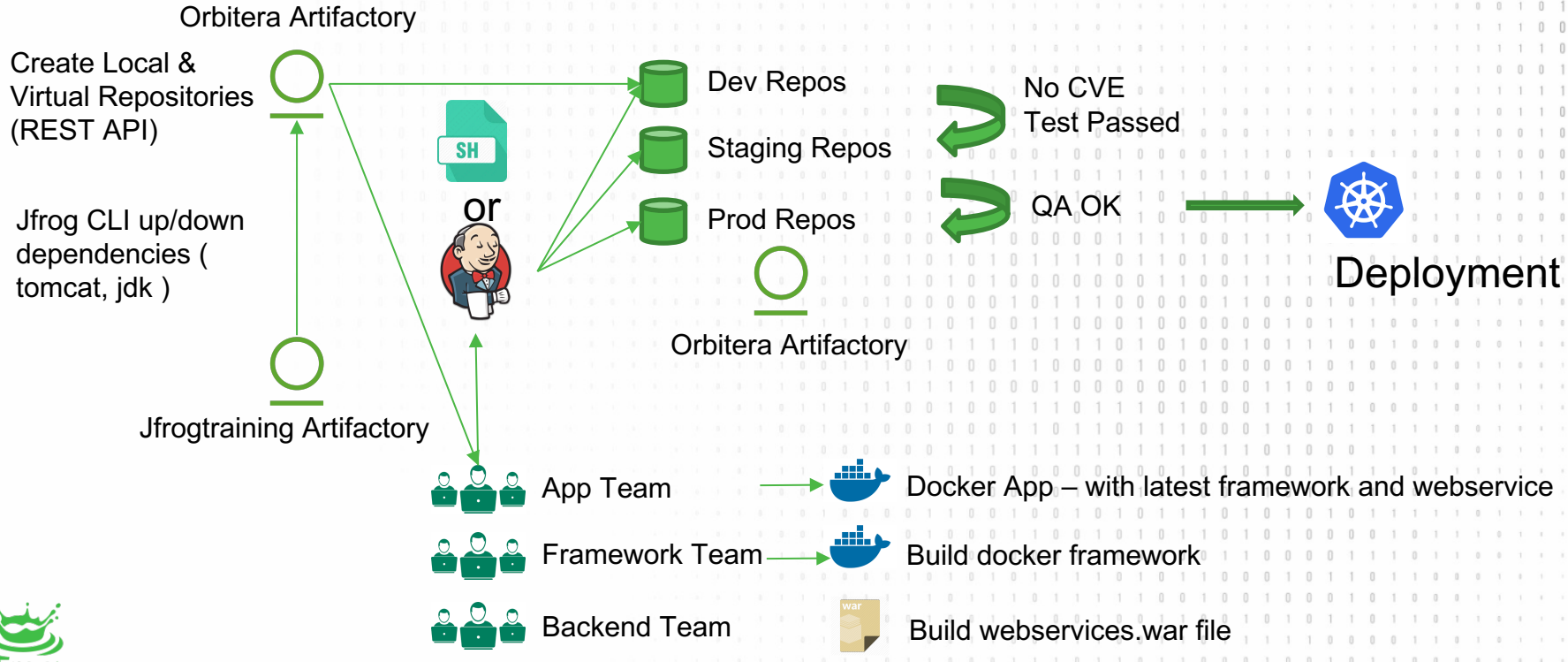
JFrog CLI – Build Integration Flow

STEP	DESCRIPTION	JFROG CLI COMMAND
1	Collect Build Name and Number	--build-name; --build-number
2	Collect Dependencies, Properties	Jfrog rt dl; Jfrog rt u; Jfrog rt bad
3	Collect Build Artifacts	Jfrog rt u –build-name –build-number
4	Collect Environment Variable	Jfrog rt bce
5	Collect Git info (i.e. Jira tickets)	Jfrog rt bag
6	Build	See next slide
6	Scan Build for vulnerabilities	Jfrog rt bs
7	Publish Build	Jfrog rt bp
8	Promote Build	Jfrog rt bpr
9	Distribute a Build	Jfrog rt bd
10	Clean Build & Discarding	Jfrog rt bc; Jfrog rt bdi

JFrog CLI – Build Integration

DESCRIPTION	JFROG CLI COMMAND
Maven Build	Jfrog rt mvnc; Jfrog rt mvn
Gradle Build	Jfrog rt gradlec; Jfrog rt gradle
MS Build	Jfrog rt u –build-name –build-number
Build Docker Images	Jfrog rt dpl; Jfrog rt dp; docker build
NPM Build	Jfrog rt npmi, Jfrog rt npmp;
Go Module Build	Jfrog rt go; Jfrog rt gp; Jfrog rt grp
Nuget Build	Jfrog rt nuget,

SU-116 Advance Automation Design – Initial



SU-116 Advance Automation Design - Reality

Orbitera Artifactory

Create Local & Virtual Repositories (REST API)

Jfrog CLI up/down dependencies (tomcat, jdk)

Jfrogtraining Artifactory

SH & 

~~Dev Repos~~

Staging Repos

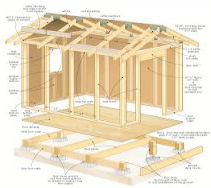
~~Prod Repos~~

Orbitera Artifactory

No CVE
Test Passed

QA OK

~~Deployment~~



App Team → Latest Released Framework + Backend (Step 3)

Framework Team → Build docker-framework with jdk, apache (Step 2)

Backend Team → Build webservices.war file (Step 1)

HOW YOU ARE YOU BUILDING IT

- Two independent development team (step1 and step2)
- Operations – packages development deliverables and deploy to Minikube
- Ability to migrate scripts to your CI server

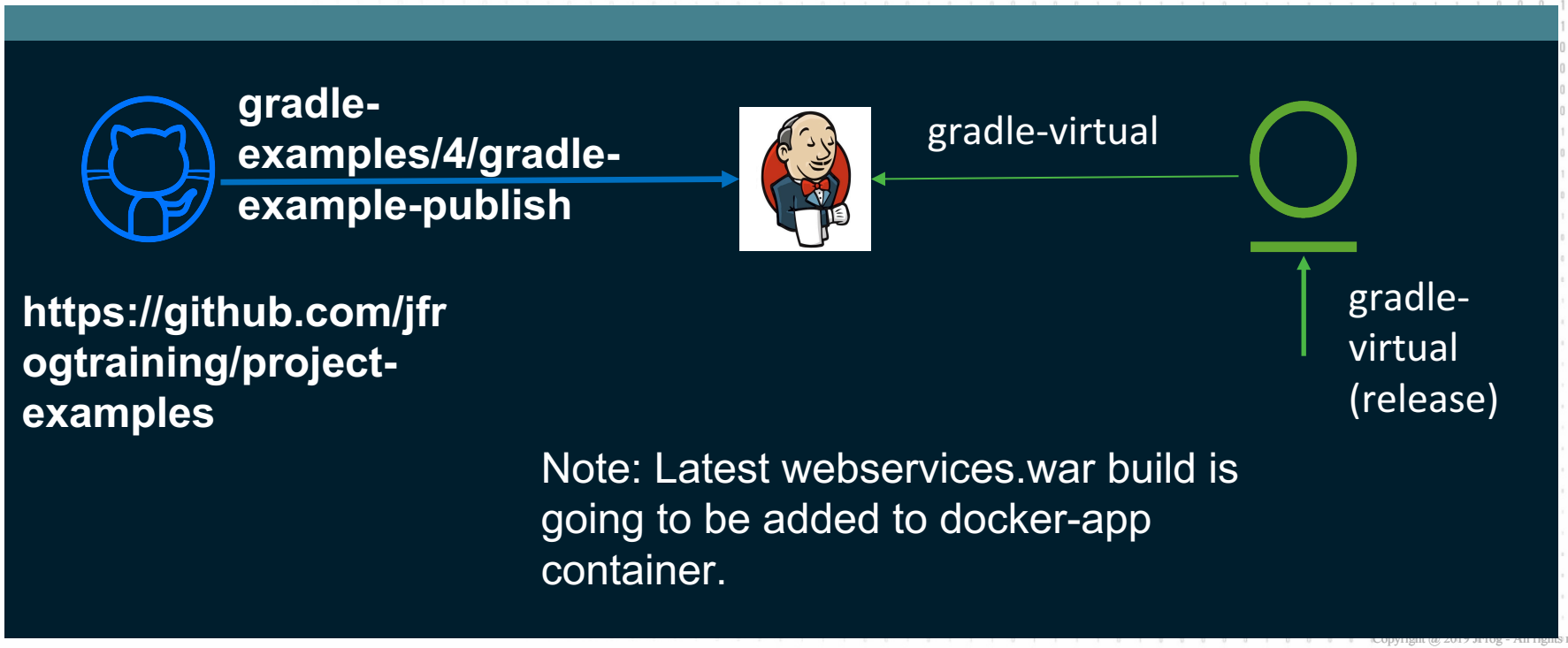
  - step1-create-application-war-file – gradle builds a simple hello world jar file
  - step2-create-docker-image-template – base docker with Apache, Tomcat and JDK
  - step3-create-docker-image-product – docker app with latest base docker and jar file

Notes:

- Docker registry is jfrog.local:5001 – points to “docker” virtual repository
- 2 Artifactory used - Artifactory (IN) act as remote repositories for jdk, apache and your local artifactory – already done
- **Manually verify gradle-release virtual repo has only grade-release-local and jcenter. Remove gradle-prod-local**

STEP1-CREATE-APPLICATION-WAR-FILE

Move your script to module4 folder. Copy step1-create-application-war-file to your script. Follow the instructions for answering the prompts



STEP1-CREATE-APPLICATION-WAR-FILE

1. Copy your main script to module4 folder.
2. Copy module4/step1-create-application-war-file to your main script.
3. Make the necessary changes.
4. When script is first executed, JFrog CLI will one time prompt you for info:
    - # Is the Gradle Artifactory Plugin already applied in the build script (y/n) [n]? **Y**
    - # Use Gradle wrapper (y/n) [n]? Y# Resolve dependencies from Artifactory (y/n) [y]? **Y**
    - # Set Artifactory server ID (press Tab for options) [jfrogtraining]: **us-site**
    - # Set repository for dependencies resolution (press Tab for options): **gradle-release**
    - # Deploy artifacts to Artifactory (y/n) [y]? **Y**
    - # Set Artifactory server ID (press Tab for options) [jfrogtraining]: **us-site**
    - # Set repository for artifacts deployment (press Tab for options): **gradle-release**
    - **#** Deploy Maven descriptor (y/n) [n]? Y# Deploy Ivy descriptor (y/n) [n]? **N**
5. Run script.

STEP2-CREATE-DOCKER-IMAGE-TEMPLATE

Docker container with jdk and apache. Same process.

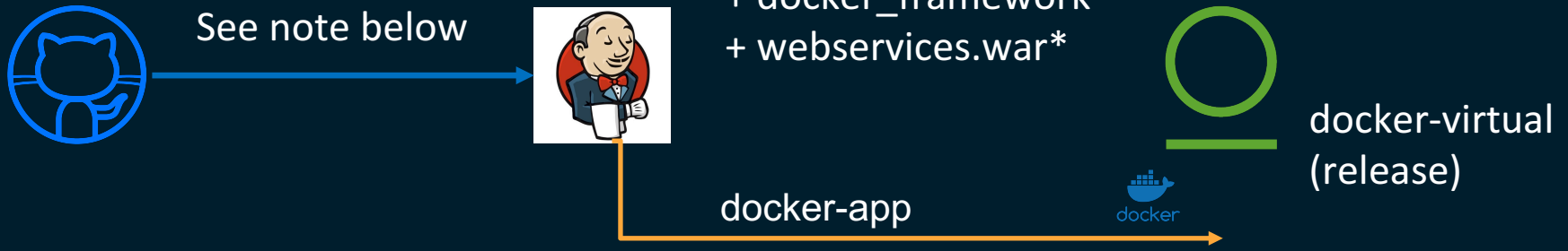
Note: Latest framework in a docker container will be used as a base for the docker-app build in step 3.

Note:

- JFrogCLI – use swampup2019/su-116-devopsautomation/module4/**step2-dockerframework**/

Step3- Create Docker App

Additional– update Docker latest tag – the script does not do that.



Note:

- JFrogCLI – use swampup2019/su-116-devopsautomation/exercises/**step3-dockerframework**/
- Jenkins Pipeline – use swampup2019/project-examples/step3_dockerproduct

RECAP

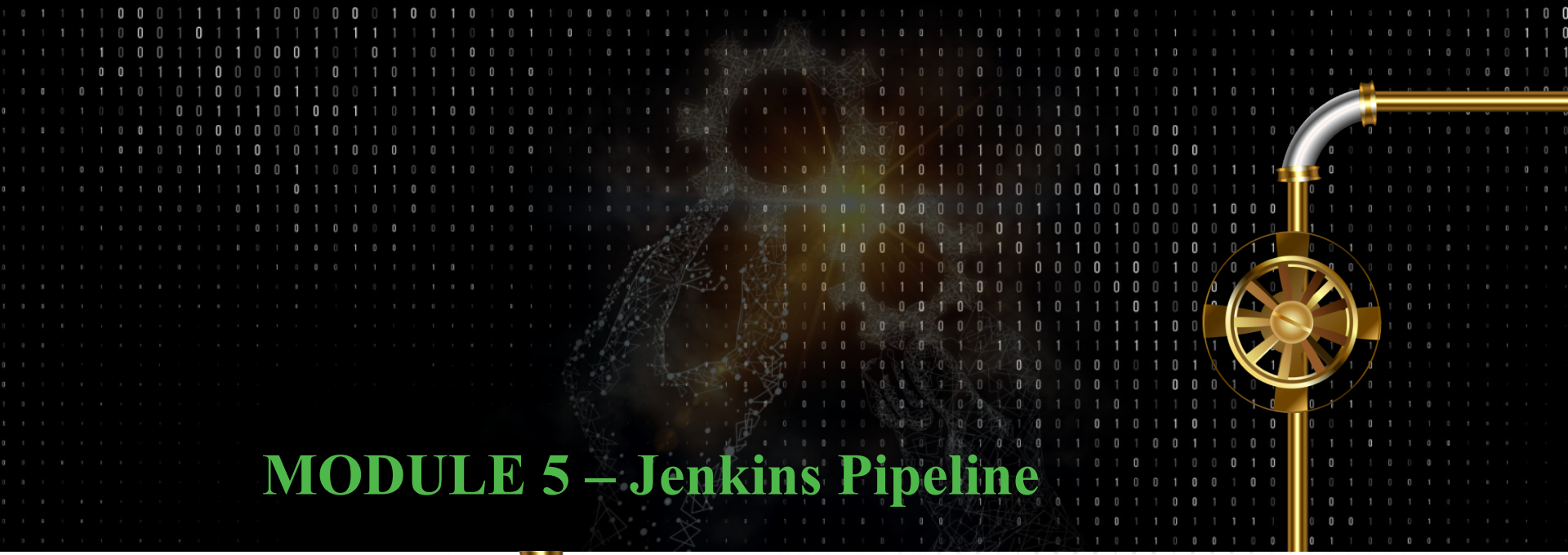
Overview – Using JFROG CLI to build sample web application - CI Server agnostic

circleci

Bitbucket

GitLab

aws-codesta



MODULE 5 – Jenkins Pipeline

Introduction to Jenkins Pipeline

Artifactory Builds with Jenkins Pipeline

Supports both Jenkins Declarative (v3.0.0) and Pipeline

STEP	DESCRIPTION	Groovy Method
1	Artifactory and Buildinfo Instance	def server = Artifactory.server 'my-server-id' def buildInfo = Artifactory.newBuildInfo()
2	Collect Dependencies, Properties	server.setProps; server.deleteProps buildinfo = server.download spec: filespec buildinfo = server.upload spec: filespec
3	Collect Build Artifacts	buidinfo.name; buildinfo.number
4	Collect Environment Variable	buildInfo.env.capture = true buildInfo.env.collect
5	Build	See following slide

Artifactory Builds with Jenkins Pipeline

STEP	DESCRIPTION	Groovy Method
7	Scan Build for vulnerabilities	scanResult = server.xrayScan scanConfig
8	Publish Build	server.publishBuildInfo buildInfo
9	Promote Build	server.promote promotionConfig
10	Distribute a Build	server.distribute distributionConfig
11	Clean Build & Discarding	Jenkins Pipeline method cleanWs()

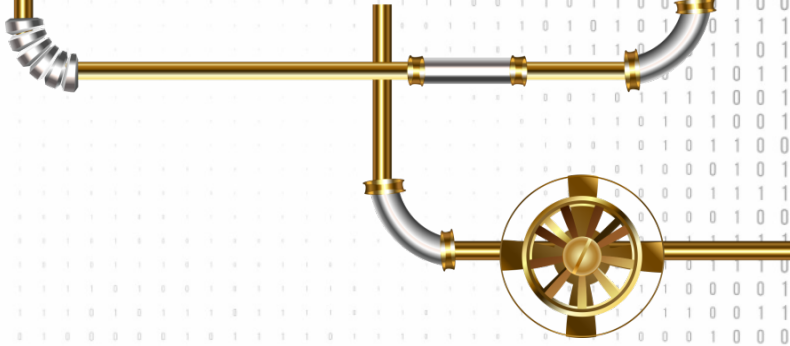
Jenkins Pipeline Labs

- Step1-createWebApplication – FILE: step1-create-application-war-file
- Step2-createWebFramework – FILE: step2-dockerframework
- Step3-createDockerApp – FILE: step3-dockerapp – **Change the script to get the latest webservices.war file.**

Use the Jenkins Pipeline code provided; cut and paste to respective Jenkins Pipeline job.

Go ahead and Launch the Jenkins URL.

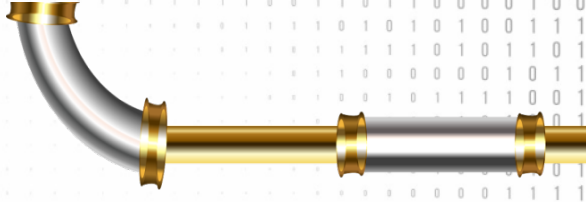
Artifactory User Plugins



JFrog 2019 swampUP

Artifactory User Plugins

Shay Bagants



Artifactory User Plugins

Extend your Artifactory features writing your own plugins

Artifactory can be easily extended by writing a Groovy user plugin that add/changes functionalities to Artifactory.

Execution points available within a User Plugin:

- Upload - Handling upload events
- Download - Handling download events
- Storage - Handling storage events
- Jobs - Defining scheduled jobs
- Executions - Defining new REST-API endpoints
- Realms - Adding custom authentication mechanism
- Build - Handling build events
- Promotions - Defining REST executable build promotion
- Staging - Defining REST retrievable build staging strategy
- Replication - Handling and filtering replication events

Artifactory Public API

Using Artifactory API and global variables

Artifactory expose Java public API for developing a user plugins:

http://repo.jfrog.org/artifactory/oss-releases-local/org/artifactory/artifactory-papi/%5BRELEASE%5D/artifactory-papi-%5BRELEASE%5D-javadoc.jar!/index.html

Within a plugin context, 5 global variables are available:

- repositories - org.artifactory.repo.Repositories
- Searches org.artifactory.search.Searches
- builds org.artifactory.build.Builds
- security org.artifactory.security.Security
- log org.slf4j.Logger

Deploying plugins

Deploy and maintain your plugin on production environment

A User Plugin can be deployed by placing it under the following directory:

`$ARTIFACTORY_HOME/etc/plugins/`

Another way to deploy a plugin is by using a REST-API endpoint:

`PUT api/plugins/{pluginName}`

Enable automatic plugin reload by adding the following system property

`artifactory.plugin.scripts.refreshIntervalSecs=10`

Apply plugin logging by changing the default logging level from WARN to INFO. To change a plugin log level, modify the `${ARTIFACTORY_HOME}/etc/logback.xml` file and add the following logger:

```
<logger name="plugin-name">
    <level value="info"
</logger>
```

Plugin examples

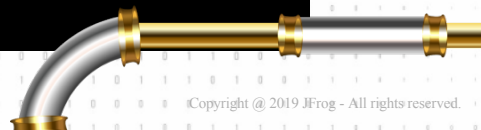
Modify download responses

```
download {
   /**
    * Modify the download response content and status in case that the artifact is tagged with a
    * 'deprecated=true' property
    */
   altResponse { request, responseRepoPath ->
      def deprecated = repositories.getProperties(responseRepoPath).getFirst('deprecated')
      if (deprecated && deprecated.toBoolean()) {
         status = 403
         message = 'This artifact was deprecated, please use some alternative.'
         log.warn "Request was made for deprecated artifact: $responseRepoPath.";
      }
   }
}
```

Plugin examples

Scheduled artifact cleanup job

```
import org.artifactory.repo.RepoPath
import org.artifactory.repo.RepoPathFactory
import org.artifactory.search.aql.AqlResult

jobs {
  cleanupJob(interval: 10000, delay: 100) {
    log.info("Deprecated artifacts cleanup started")
    String sqlQuery = "items.find({\"repo\":\"example-repo-local\"},{\"@deprecated\":\"true\"})";
    searches.aql(aqlQuery) { AqlResult result ->
      result.each { Map item ->
        String itemPath = constructPath(item)
        log.info("Deleting artifact: $itemPath")
        RepoPath repoPath = RepoPathFactory.create(itemPath)
        repositories.delete(repoPath)
      }
    }
    log.info("Deprecated artifacts cleanup completed")
  }
}

private String constructPath(Map item) {
  if (item.path.toString().equals(".")) {
    return item.repo + "/" + item.name
  }
  return item.repo + "/" + item.path + "/" + item.name
}
```



Discover more

User Plugin examples:
https://github.com/jfrog/artifactory-user-plugins

User Plugin documentation:
https://www.jfrog.com/confluence/display/RTF/User+Plugins

JFrog Certification

[Certification FAQs]

[Study Guide]

JFrog Certification

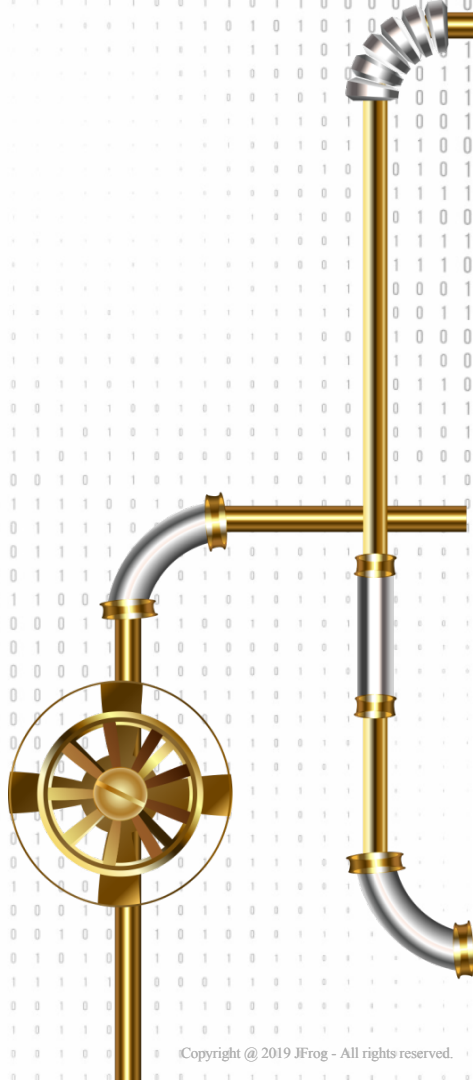
Ready, Cert, Go!

Everyone gets a voucher by email

90 minutes proctored exam - requires a webcam

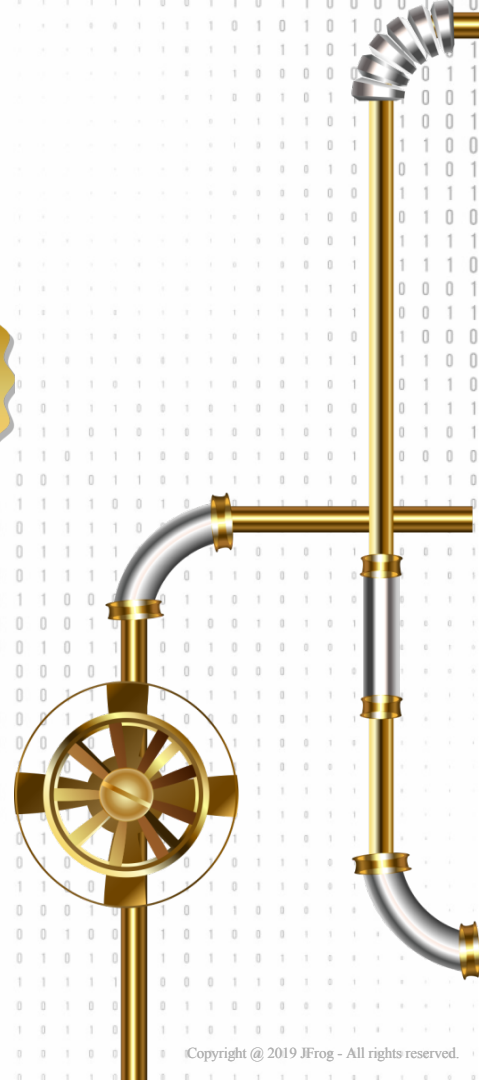
Practical Knowledge

For more information please contact certification@jfrog.com



JFrog Certification

Ready, Cert, Go!



QUESTIONS AND ANSWERS

THANK YOU!