Jfrog Artifactory

By - Kavan Dalwadi

**-Dev Jadeja**

**- Suhana Mansuri**

# Contents -

What is Jfrog Artifactory

Jfrog Pricing

Jfrog Saas vs Self-hosted

Jfrog Architecture

Installation

Packages Supported

Best Practices

Integrating JFrog Artifactory with Jenkins

# What is Jfrog Artifactory?

JFrog Artifactory is a universal DevOps solution providing end-to-end automation and management of binaries and artifacts through the application delivery process that improves productivity across your development ecosystem. Artifactory is Kubernetes ready supporting containers, Docker, Helm Charts, and is your Kubernetes and Docker registry and comes with full CLI and REST APIs customizable to your ecosystem.

## Features and Functionality -

#### Hybrid and Multi-Cloud Environments

You can host Artifactory on your own infrastructure, in the Cloud or use the SaaS solution providing maximum flexibility and choice.

#### Universal Binary Repository Manager

Artifactory offers a universal solution supporting all major package formats including Alpine, Maven, Docker, Debian, Go, Helm, NPM, Terraform and more.

#### Artifactory as Your Kubernetes Registry

Artifactory allows you to deploy containerized microservices to the Kubernetes cluster as it serves as a universal repository manager for all your CI/CD needs, regardless of where they are running in your organization.

#### High Availability

Full active/active HA solution with live failover and non-disruptive production upgrades.

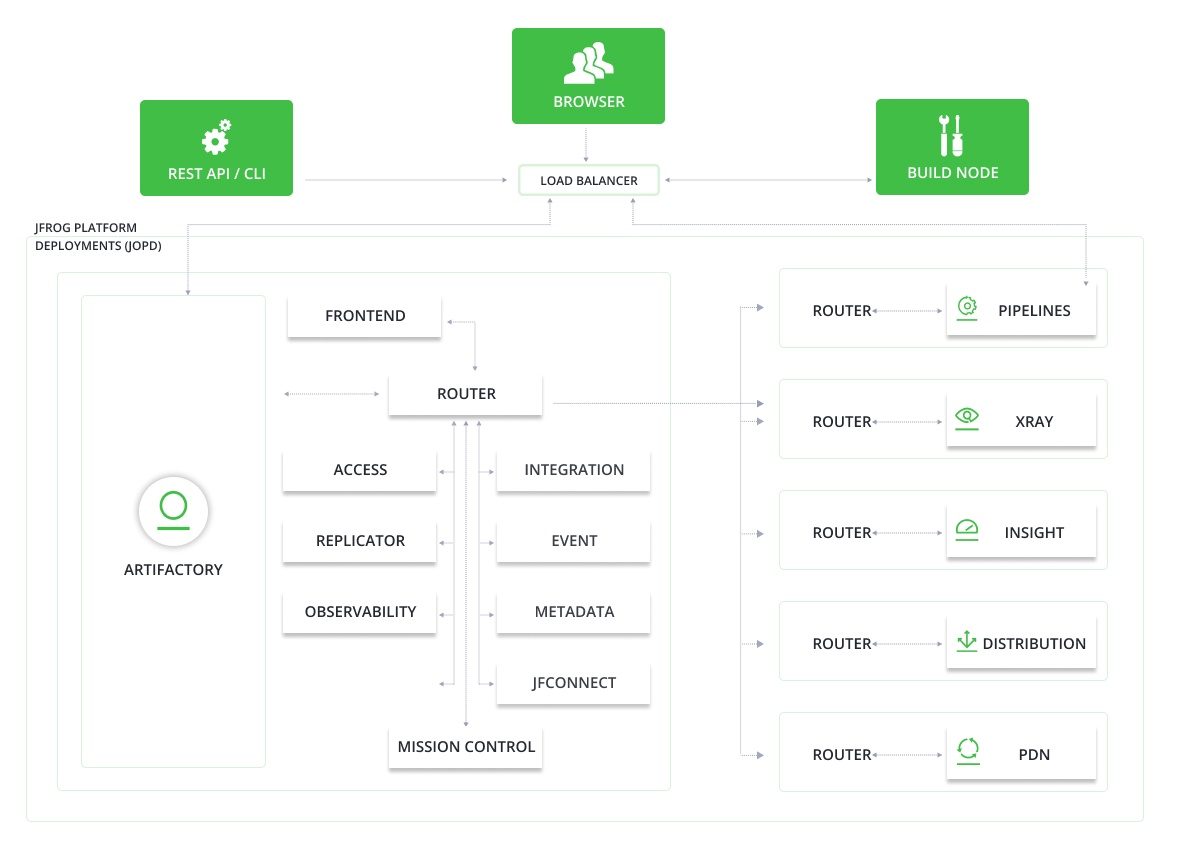
#### Artifactory Cloud with CDN Distribution

JFrog Artifactory Cloud with Amazon's CloudFront CDN solution allows Enterprise users to manage, control, and distribute high volumes of software distribution across multiple locations.

# Jfrog pricing

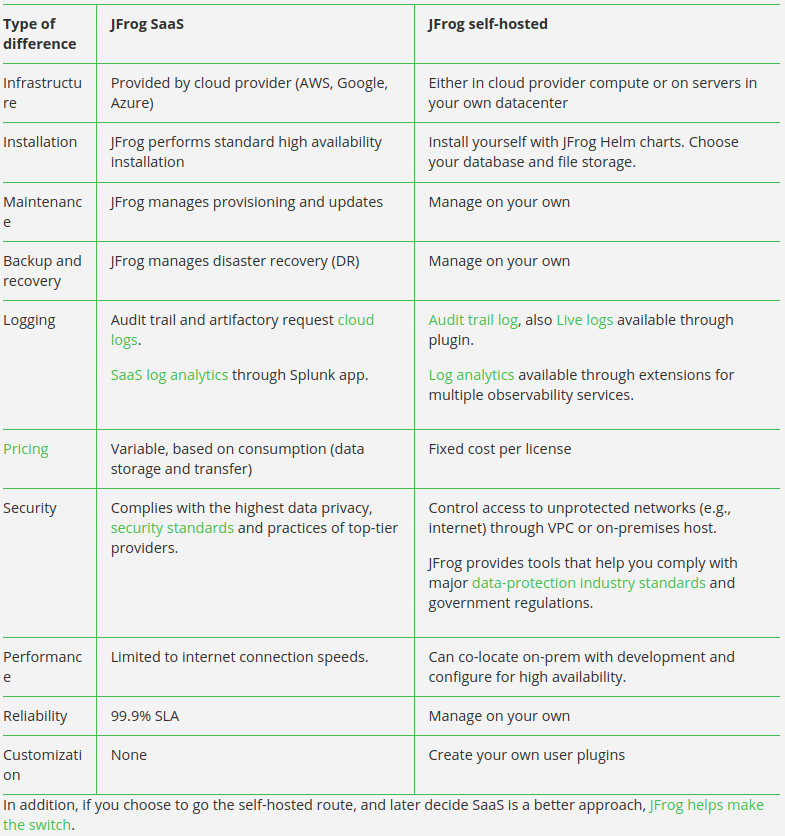
<https://jfrog.com/pricing/>

# Jfrog Architecture



<https://www.jfrog.com/confluence/display/JFROG/System+Architecture>

# Jfrog SaaS vs Self-hosted

****

# Installation

<https://www.jfrog.com/confluence/display/JFROG/Installing+the+JFrog+Platform>

# Packages supported

<https://www.jfrog.com/confluence/display/JFROG/Package+Management>

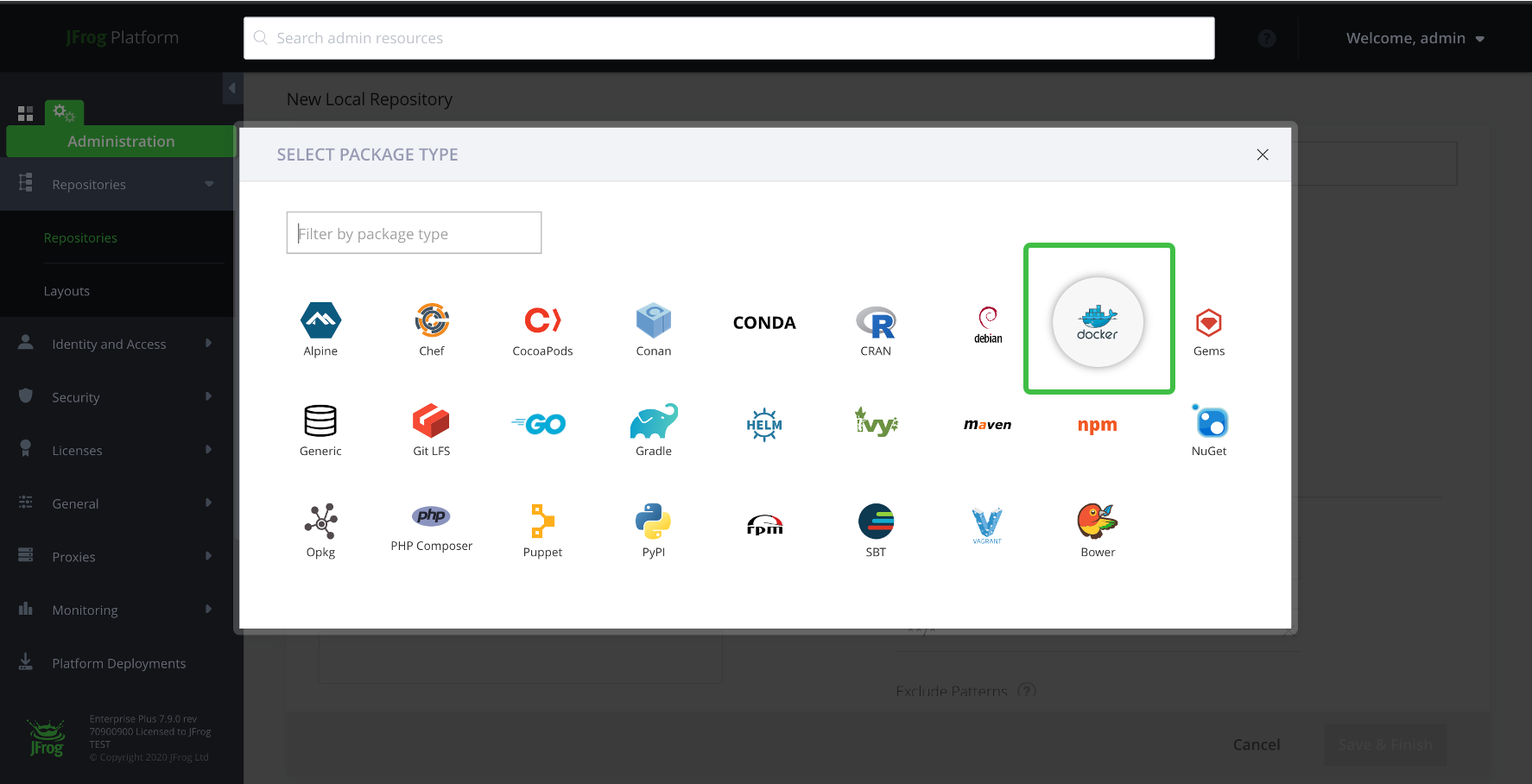
# 

# Best Practices

### 1. Repository design characteristics

**Package Type Mirrors Repository Type**

Within the JFrog Platform there are over 25 different package types available to integrate into your development environment. Each repository is constructed for one defined package type. For example, for Maven package integration, create a Maven repository structure. When adding another package type, for example Docker, then create a separate Docker set of repositories to mirror your development lifecycle, and repeat to support all the development environments across your organization. The one exception is the package type Generic, which can accept any file format.



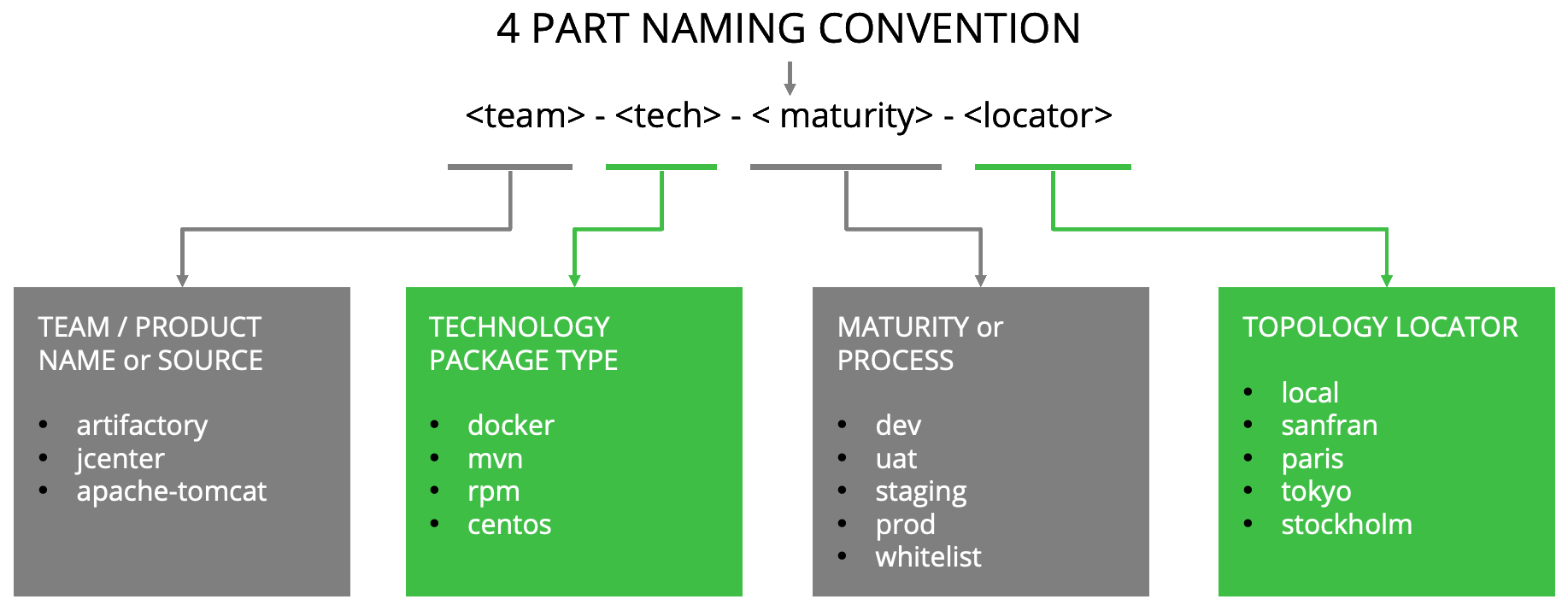
### 2. Naming conventions and structure

JFrog recommends a 4 part naming convention to scale and manage your global repositories. As shown, this includes:

**<team>-<technology>-<maturity>-<locator>**

* A product or team name as *the primary identifier* of the project.
* The technology, tool or *package type* being used.
* The package *maturity level*, such as the development, staging and release stages.
* The locator, the *physical topology* of your artifacts.

Creating Artifactory repositories per team, or folders inside repositories, enables each team to handle their specific vulnerabilities.



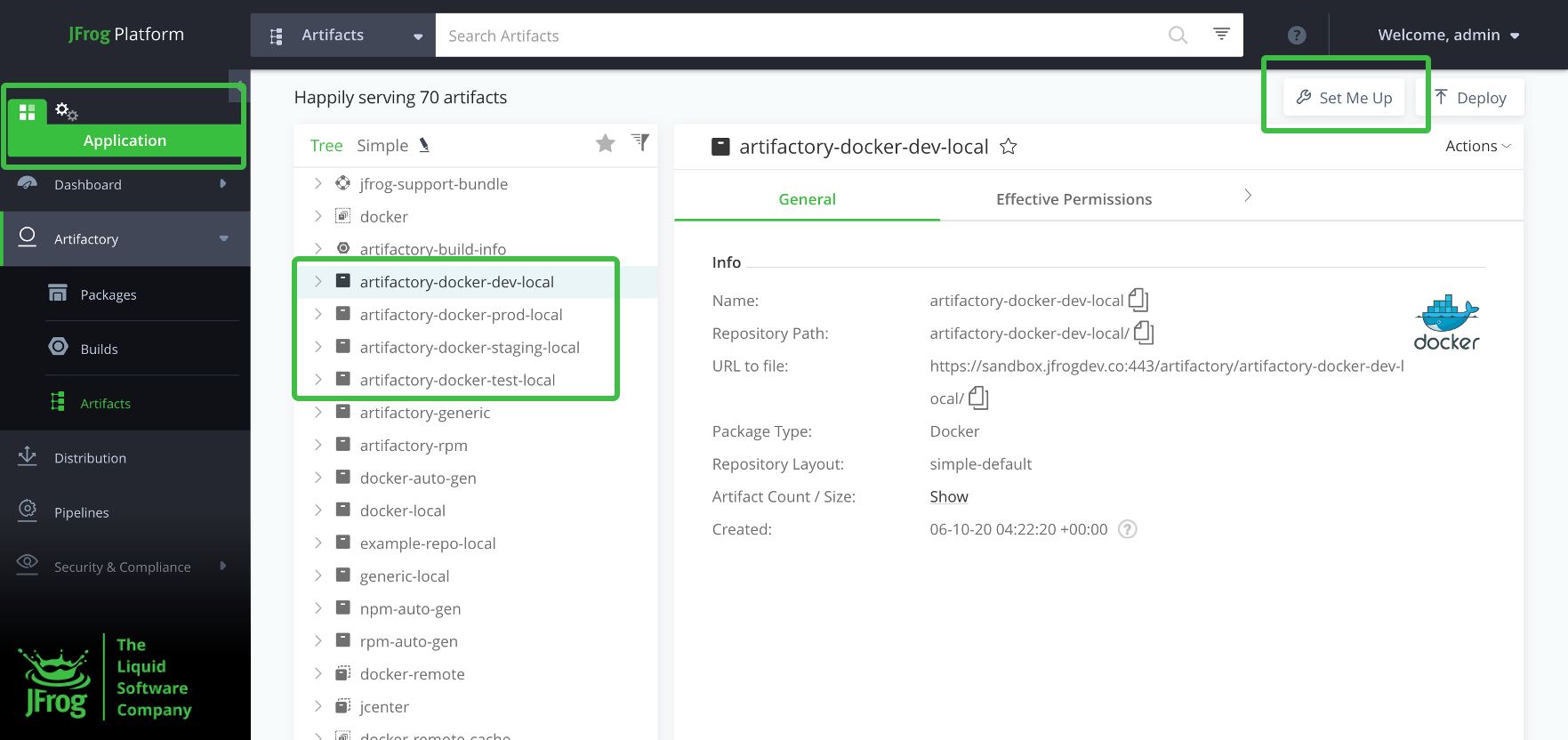
See the complete whitepaper, [Best Practices for Structuring and Naming Artifactory Repositories](https://jfrog.com/whitepaper/best-practices-structuring-naming-artifactory-repositories/), for further descriptions of the use cases and configurations to optimize your development environment.

### 3. Create a repository structure that mirrors your development life cycle.

**Repository Structure**

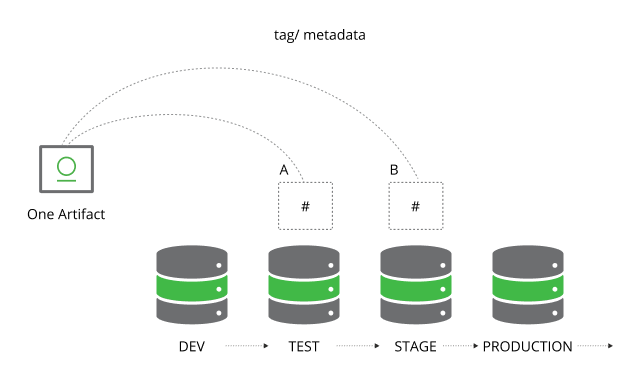
Create a repository structure that mirrors your development life cycle in order to support an intuitive promotion of the artifact across different stages in the CI. A classic 4 stage development life cycle such as; development, test, staging and production would have 4 repositories:

* yourcompany-docker-***dev***-local
* yourcompany-docker-***test***-local
* yourcompany-docker-***stage***-local
* yourcompany-docker-***prod***-local



Publish Build Artifacts with Tags/Metadata to Each Stage of the Development Lifecycle

Instead of deploying artifacts into a quarantined area, JFrog Artifactory simply tags artifacts with metadata.



With promotion properties in place, you can set permissions that apply to artifacts tagged with properties that represent different stages. We can automate this using dev tool integrations as we deploy from one stage to another.

Then the artifacts will be published to a repository that grants access to users with roles relevant to that stage, like testers for example.

Using the REST API, you can automate many of the tasks involved in build promotions.

Integrating JFrog Artifactory with Jenkins

First of all we will need :

1. An Artifactory Server

2. A Jenkins Server

**Artifactory Setup**

## Pre-requisites:

1. An AWS T2.small EC2 instance (Linux)
2. Open port 8081 and 8082 in the security group

## Installation Steps:

1. Login to instance as a root user and install Java  
    yum install java-1.8\* -y

2.Download Artifactory packages onto /opt/  
 For Latest version of Artifactory OSS [download it from here](https://jfrog.com/open-source/) For Older version of Artifactory OSS [download it from here](https://jfrog.bintray.com/artifactory/) For Latest version of Artifactory Pro [download it from here](https://jfrog.com/artifactory/)

cd /opt

Wget https://jfrog.bintray.com/artifactory/jfrog-artifactory-oss-6.9.6.zip

3.extract artifactory tar.gz file  
 unzip jfrog-artifactory-oss-6.9.6.zip

4.Go inside to bin directory and start the services  
 cd /opt/jfrog-artifactory-oss-6.9.6/bin

./artifactory.sh start

5.access artifactory from browser  
 http://<PUBLIC\_IP\_Address>:8081

6.Provide credentials  
 username: admin

password: passwrod

Now Integrating Artifactory with Jenkins

1. **Login to Jenkins to integrate Artifactory with Jenkins**
2. **Install "Artifactory" plug-in**

* Manage Jenkins -> Jenkins Plugins -> available -> artifactory
* Configure Artifactory server credentials
* Manage Jenkins -> Configure System -> Artifactory
  + Artifactory Servers
    - Server ID : Artifactory-Server
    - URL : Artifactory Server URL
    - Username : admin
    - Password : `admin@123

### Create a Freestyle Project:

-Create a new job

Job Name : artifactory-project

-Source code management

Git: <https://github.com/profdj07/jfrog-demo>

1. **Build Environment :**

Maven3-Artifactory Integration : `<provide Artifactory server and repository details

Build --> Invoke Artifactory Maven3

Goals: clean install

-Execute job

### 5.Create a Maven Project:

1.Create a new job:

Job Name : artifactory-project

Source code management

Git : <https://github.com/profdj07/jfrog-demo>

2. Build Environment:

Resolve artifacts from Artifactory : <provide Artifactory server and repository details>

3. Build - Goals: clean install

4.Post-build Actions :

Deploy Artifacts to Artifactory : <provide Artifactory server and repository details>

5. Execute Job