
🧱 Lab Guide: Install and Configure JFrog Artifactory for a Java Application

---

🎯 Lab Objective

By the end of this lab, you will:
✅ Install JFrog Artifactory on a Linux VM (Ubuntu 22.04)
✅ Configure repositories (Local, Remote, Virtual)
✅ Build a Java application using Maven
✅ Upload and manage artifacts in JFrog
✅ Integrate JFrog with Jenkins for automated CI/CD

⚙️ Lab Prerequisites

Component	Requirement
OS	Ubuntu 22.04 LTS (2 vCPU, 4GB RAM minimum)
Java	JDK 11 or later
Maven	Installed on build machine
Jenkins (optional)	Installed and configured
JFrog Artifactory	Open Source (OSS) or Pro version
Network Access	Outbound internet for package download

🧩 Step 1: Update the System

sudo apt update && sudo apt upgrade -y
sudo apt install -y curl unzip wget vim net-tools

🧱 Step 2: Install JFrog Artifactory (Open Source)

Option 1 — Using Debian Package

wget
https://releases.jfrog.io/artifactory/artifactory-pro/org/artifactory/pro/jfrog-artifactory-pro/7.77.6/jfrog-artifactory-pro-7.77.6-linux.tar.gz

sudo dpkg -i jfrog-artifactory-oss-7.77.6-debian-x64.deb

Option 2 — Using Docker (Preferred for Labs)

sudo apt install -y docker.io
sudo systemctl enable docker --now

```
docker run --name artifactory -d \
  -p 8081:8081 \
  -p 8082:8082 \
  docker.bintray.io/jfrog/artifactory-oss:latest
```

🧭 Step 3: Access JFrog Artifactory UI

Once installation completes, open:

http://<server-ip>:8082/ui/

🔑 Default credentials:

Username: admin

Password: password

It will prompt to:

- Change the password
- Set base URL (use http://<server-ip>:8082/artifactory)
- Skip license setup if using OSS

🧩 Step 4: Configure Repositories

🧱 Create Local Repository (for Java Artifacts)

1. Go to **Administration → Repositories → Local → New Local Repository**
2. Select **Maven** as the package type
3. Name it: libs-release-local
4. Click **Create**

🌍 Create Remote Repository (for Maven Central)

1. Go to **Repositories → Remote → New Remote Repository**
2. Choose **Maven**
3. Name: maven-central

4. URL: https://repo1.maven.org/maven2
5. Click **Create**

🧩 Create Virtual Repository

1. Go to **Repositories** → **Virtual** → **New Virtual Repository**
2. Package Type: **Maven**
3. Name: maven-all
4. Under “Selected Repositories,” add:
    - libs-release-local
    - maven-central

Save and copy the virtual repo URL — e.g.:

http://<server-ip>:8082/artifactory/maven-all

5.

🧰 Step 5: Configure Maven to Use JFrog

Edit your Maven settings file:

vim ~/.m2/settings.xml

Add this inside <mirrors>:

```
<mirror>
  <id>artifactory</id>
  <mirrorOf>*</mirrorOf>
  <url>http://<server-ip>:8082/artifactory/maven-all</url>
</mirror>
```

And inside <servers> (for authentication):

```
<server>
  <id>artifactory</id>
  <username>admin</username>
  <password>your-password</password>
</server>
```

📦 Step 6: Build a Sample Java Project

Create or clone a sample Java app:

```
git clone https://github.com/spring-projects/spring-petclinic.git
cd spring-petclinic
```

Build the artifact:

```
mvn clean package
```

You'll get:

```
target/spring-petclinic-2.4.2.jar
```

🚀 Step 7: Upload Artifact to JFrog

You can upload manually or via CLI.

Option 1 — Using Web UI:

1. Navigate to **Artifacts** → **libs-release-local**
2. Click **Deploy**
3. Choose file: spring-petclinic-2.4.2.jar
4. Path: /com/demo/petclinic/2.4.2
5. Deploy

Option 2 — Using cURL:

curl -u admin:password -T target/spring-petclinic-2.4.2.jar \

"http://<server-ip>:8082/artifactory/libs-release-local/com/demo/petclinic/2.4.2/spring-petclinic-2.4.2.jar"

⚙️ Step 8: Verify Upload

Go to:

http://<server-ip>:8082/ui/repos/tree/General/libs-release-local/com/demo/petclinic/2.4.2/

You should see your JAR file with metadata and checksum.

🧪 Step 9: Integrate JFrog with Jenkins (Optional)

Install the **JFrog Artifactory Plugin**:

1. In Jenkins → **Manage Plugins**
2. Search “Artifactory” → Install
3. Add Artifactory Server under **Manage Jenkins → Configure System**

Example Jenkinsfile

```
pipeline {
  agent any
  environment {
    ARTIFACTORY_SERVER = 'artifactory-server'
    REPO = 'libs-release-local'
  }
  stages {
    stage('Build') {
      steps {
        sh 'mvn clean package'
      }
    }
    stage('Upload to JFrog') {
      steps {
        rtUpload(
          serverId: "${ARTIFACTORY_SERVER}",
          spec: '''{
            "files": [
              {"pattern": "target/*.jar", "target": "libs-release-local/com/demo/"}
            ]
          }'''
```

```
            )
        }
    }
  }
}
```

🧠 Step 10: Verify CI/CD Integration

When Jenkins runs:

- It builds your Java code
- Uploads the JAR into JFrog automatically
- Artifact metadata (build number, commit, timestamp) will appear in JFrog

---

🧩 Step 11: Security & Cleanup (Optional)

Enable Anonymous Access (for labs only)

Go to:

Admin → Security → Users → anonymous → Enable

Cleanup Old Builds

Go to:

Admin → Repositories → Maintenance → Clean Up Policies

Set a policy to delete old artifacts periodically.

✅ Step 12: Validate Everything

Checklist ✅

Task	Status
Artifactory Installed	✅
Maven Configured	✅
Local/Remote/Virtual Repos Created	✅
Java App Built	✅
Artifact Uploaded	✅
Jenkins Integration (Optional)	✅

🗒️ Summary

Component	Purpose
JFrog Artifactory	Centralized artifact repository
Local Repo	Stores internal builds
Remote Repo	Proxies external Maven dependencies
Virtual Repo	Unified endpoint for builds
Maven + JFrog	Builds + pushes JARs securely

Jenkins + JFrog	Automates artifact management in CI/CD

💡 Bonus

To explore deeper:

- Enable **JFrog Xray** → for vulnerability scanning
- Configure **Artifactory HA** → for reliability
- Automate promotion (Dev → QA → Prod repos) using Jenkins or CLI