**Artifactory Installation and Configuration**

1. **System Requirements**
   1. **Supported Platforms**

Artifactory has been tested and verified on Linux, Windows (Vista and higher), Solaris and Mac OS X

* 1. **Pre-requisites**

Java – JDK 8 and above – preferred JDK 8 update 45 and above

Set JAVA\_HOME and JRE\_HOME

JVM Memory Allocation – Not strict but recommended to reserve at least 512MB for Artifactory

Notes: This is done by modifying the JAVA\_OPTIONS variable in artifactory.bat, for a [manual installation](https://www.jfrog.com/confluence/display/RTF/Installing+on+Windows#InstallingonWindows-ManualInstallation), or the JOPTSvariable in installService.bat when running Artifactory as a [service](https://www.jfrog.com/confluence/display/RTF/Installing+on+Windows#InstallingonWindows-ServiceInstallation).

Recommended values are:

  -server -Xms512m -Xmx2g -Xss256k -XX:+UseG1GC

* 1. **Recommended Hardware**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number of developers** | **OS/JVM** | **Processor** | **\*Memory (RAM) for JVM Heap** | **Storage** |
| 1 - 20 | 64 bit | 4 cores | 4GB | Fast disk with free space that is at least 3 times the total size of stored artifacts |
| 20 - 100 | 64 bit | 4 cores | 8GB | Fast disk with free space that is at least 3 times the total size of stored artifacts |

1. **Installation on Windows**
   1. **Manual Installation**

Unzip the Artifactory download file to a location on your file system.

This will be your *%ARTIFACTORY\_HOME%* location.

Define this location as an environment variable called ARTIFACTORY\_HOME

Browse to *%ARTIFACTORY\_HOME%\bin* and execute the file *artifactory.bat*. This script searches for the Java executable and runs Artifactory's main class.

* 1. **Service Installation**

To run Artifactory as a Windows service, browse to %ARTIFACTORY\_HOME%\bin, and execute the file InstallService.bat.

By editing InstallService.bat, you can modify default properties such as JOPTS and the log directory.

For your changes to take effect you need to stop the currently running Artifactory service and run InstallService.bat again once you have completed your modifications.

D:\Softwares\artifactory-pro-5.2.0\tomcat\conf – to update the connector port in server.xml file

* 1. **Configuring Database**

Artifactory comes with a built-in embedded Derby database that can be reliably used to store data and currently support the following databases derby, MySQl, oracle and postgres.

* 1. **Running Artifactory**

After installing Artifactory you need to start the service.

To start or stop Artifactory as a service you can use the following command in a **Command Prompt** window

Starting and stopping the Artifactory service

|  |
| --- |
| sc start|stop Artifactory |

* 1. **Accessing Artifactory**

Artifactory can be accessed using the following URL:

[http://SERVER\_DOMAIN:8081/artifactory](http://server_domain:8081/artifactory)

1. **Industry Standard binary Management**

**Configuring Repositories:**

**3.1 Types of repositories**: Artifactory has 3 types of repositories - Local, Remote and Virtual. Local and remote repositories are true physical repositories, while a virtual repository is actually an aggregation of them used to create controlled domains for search and resolution of artifacts.

**3.2 Single Package Type:** When creating any repository, you must specify its package type; this is a fundamental characteristic of the repository and cannot be changed later. Once the repository type is set, Artifactory will index artifacts and calculate the corresponding metadata for every package uploaded which optimizes performance when resolving artifacts.

**3.3 Generic Repositories:** You may define a repository as **Generic** in which case it has no particular type, and you may upload packages of any type

**3.4 Local Repositories:** Local repositories are physical, locally-managed repositories into which you can deploy artifacts. Artifacts in a local repository can be accessed directly using the following URL:  
http://<host>:<port>/artifactory/<local-repository-name>/<artifact-path>

**Reference:** <https://www.jfrog.com/confluence/display/RTF/Local+Repositories>

**3.5 Remote Repositories:** A remote repository serves as a caching proxy for a repository managed at a remote URL (which may itself be another Artifactory remote repository).

Artifacts are stored and updated in remote repositories according to various configuration parameters that control the caching and proxying behavior. You can remove artifacts from a remote repository cache but you cannot actually deploy a new artifact into a remote repository.

Artifacts in a remote repository can be accessed directly using the following URL:

http://<host>:<port>/artifactory/<remote-repository-name>/<artifact-path>

**Reference**: <https://www.jfrog.com/confluence/display/RTF/Remote+Repositories>

**3.6 Virtual Repositories:** A virtual repository (or "repository group") aggregates several repositories with the same package type under a common URL. The repository is virtual in that you can resolve and retrieve artifacts from it but you cannot deploy artifacts to it.

**3.7 General Resolution Order:** We can set the order in which repositories of each type (local, remote and virtual) are searched and resolved by simply ordering them accordingly within the corresponding section of the **Configure Repositories** page. To set the order you need to add the repositories to the list of selected repositories in the order in which they should be searched to resolve artifacts.

**4. Best Practices:**

**4.1 Repository grouping** – To pare down large number of repositories list, the repository manager can define virtual repositories (or group of repositories) where every request to that group is served from any one of the repositories in the group.

**4.2 Build and release schedule**: Nightly, snapshot and continuous

Distinguish which binaries require nightly, continuous, or snapshot builds and release.

**4.3 Common build tools with respective binary repos**

|  |  |
| --- | --- |
| Platform | Build tool/binary repo |
| JVM (Java) | Maven, Ant, Gradle, Ivy |
| .Net | Nuget |
| Linux | apt-get, Yum |

**4.3 Artifact Promotion:**



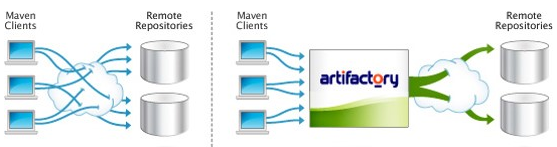
**4.4 Purging Policy**

Although artifacts usually must be kept for a long time (the same as any other product or distribution), there are some cases when we can benefit from purging repository contents.

Snapshot repositories need to be purged from time to time to ensure a reasonable disk usage—especially when using continuous integration heavily, since CI can easily generate several builds per day. Usually, snapshots can be purged when a new version is released, but that may be changed to just keep the last n snapshots. Some binary repository managers offer automatic purging procedures based on defined policies (e.g. number of snapshots to retain).

**5 Setup Artifactory as a Proxy**

Below diagram depicts that the Artifactory can be configured to fetch (resolve) dependent artifacts from Internet not from maven central, it can be (should be) used as a proxy



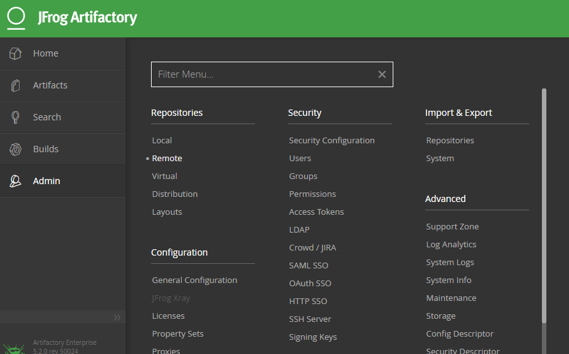
**Steps to setupMaven to resolves it dependencies by fetching the artifacts from Artifactory not from Internet (Maven Central)**

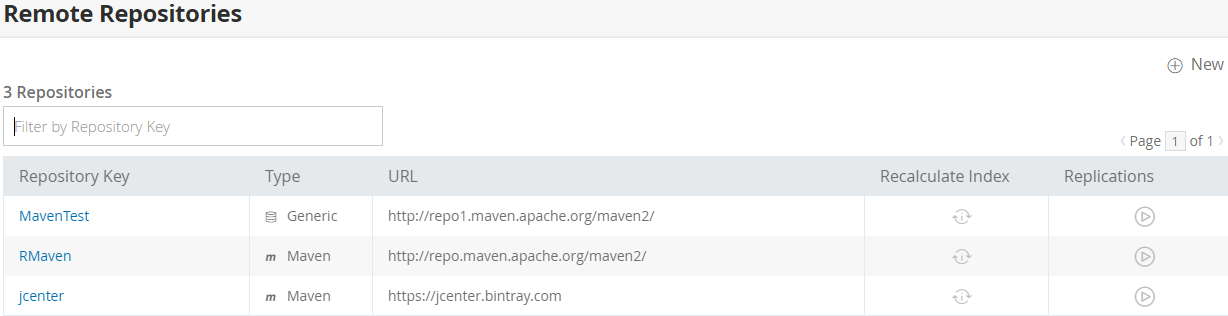
1. Force your local build to use Artifactory as a repository for all Artifacts.

Update Pom.xml (Project file) and setting.xml accordingly.

1. Configure Maven central as a Remote repository in Artifactory.

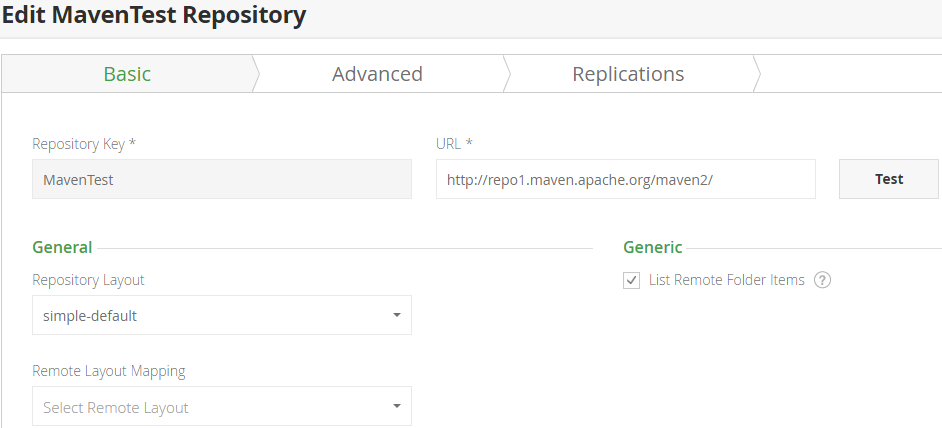
Under Admin 🡪Repositories🡪Remote



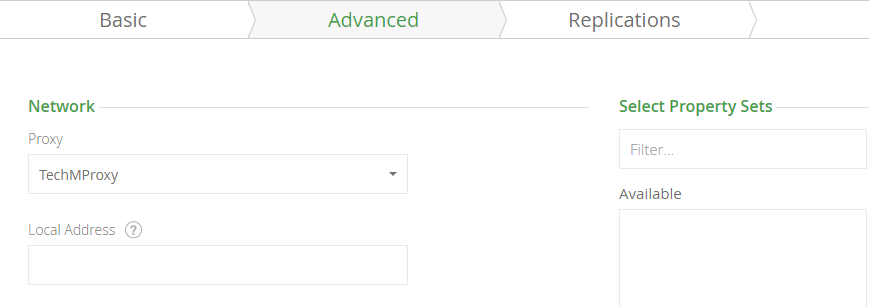


Click New -> select type of Package

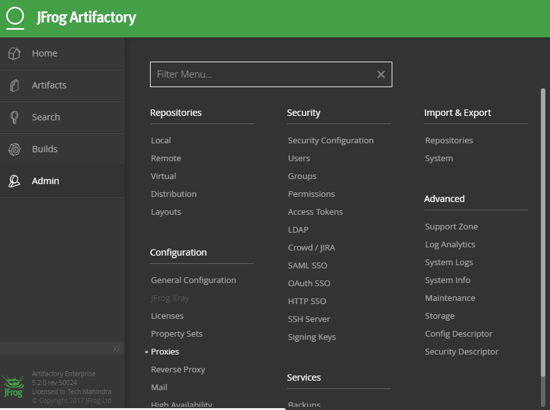
Next Step provide Repository Key (Name), URL and configure other settings as required.



Set the proxy (Which is configured in Step 3) in advanced Tab if we get any connection time out error.



1. Configure the Proxy if required (Organization using any proxy to connect to Internet)



1. Test the URL.

Notes:

A [remote repository](http://www.jfrog.com/confluence/display/RTF/Configuring+Repositories#ConfiguringRepositories-RemoteRepositories) in Artifactory serves as a caching proxy for remote artifacts (jcenter is an example of such repository). This means that it downloads artifacts from the remote URL and cache them in Artifactory,

When your build is resolving artifacts from Artifactory it is possibly resolving artifacts from remote repositories. This can be done if you are directly referencing remote repositories form your build or indirectly if you are referencing [virtual repositories](http://www.jfrog.com/confluence/display/RTF/Configuring+Repositories#ConfiguringRepositories-VirtualRepositories) which aggregates remote repositories.

In Simple terms Artifactory communicates with the maven central and downloads all the artifacts to remote repository thus acting as proxy when a request from build is made the artifacts as resolved from Artifactory.

Order of Artifact resolution:

Local🡪Cache 🡪Remote repository

If the problem is that artifacts are not resolved at all from the remote repository, try the following:

Make sure the repository is configured correctly in Artifactory. Use the "Test" button to make sure that the URL is correct and you can reach the remote URL.

Check that your build tool is properly configured to use the repository you configured. One way of checking this, is by monitoring the Artifactory [request log](http://www.jfrog.com/confluence/display/RTF3X/Artifactory+Log+Files#ArtifactoryLogFiles-RequestLog) looking for requests from your build tool.