**Artifact Repository:**

An Artifact Repository Manager is a software tool that helps manage and organize the storage and retrieval of software artifacts. It serves as a centralized location where developers can store, share, and manage software components such as libraries, frameworks, binaries, and other build artifacts.

Key features of an Artifact Repository Manager include:

1. **Artifact Storage**: It provides a repository to store software artifacts in a structured manner. This can include libraries, binaries, documentation, container images, and other components needed for software development and deployment.
2. **Dependency Management**: It helps manage dependencies between software components. Developers can define and retrieve dependencies from the repository, ensuring consistent and reliable access to required components.
3. **Version Control**: An Artifact Repository Manager maintains versioning information for artifacts, allowing developers to retrieve specific versions of components and track changes over time.
4. **Access Control**: It provides authentication and authorization mechanisms to control access to artifacts. This ensures that only authorized individuals or systems can retrieve and publish artifacts.
5. **Proxying and Caching**: Artifact Repository Managers can act as a proxy server for external repositories, reducing external dependencies and improving build speed by caching artifacts locally.

**Where it is used?**

* If Organizations wants to share libraries (like JAR files) among multiple developers for a project development, then Nexus is used.
* Once applications are developed in build process CI/CD tools like Jenkins stores build outputs (JAR/WAR) inside Nexus Artifact Repository.

**System requirements:**

1. Operating System:
   1. Linux (64-bit)
   2. Windows (64-bit)
   3. macOS (64-bit)
2. Java Development Kit (JDK):
   1. Nexus 3.x requires **JDK 8** or 11 (OpenJDK or Oracle JDK). It does not support JDK 9 or 10.
   2. Nexus 2.x supports JDK 7 or 8.
3. Memory:
   1. Recommended minimum heap size for Nexus 3.x: **4 GB RAM**
   2. Recommended minimum heap size for Nexus 2.x: 2 GB RAM
4. Disk Space:
   1. For Nexus 3.x, you should have at least 1 GB of free disk space for installation.
   2. For Nexus 2.x, it is recommended to have a minimum of 4 GB of free disk space for installation.
5. Network:
   1. Nexus requires access to the internet for downloading artifacts, metadata, and plugins from remote repositories.

**Download Link:** [**https://help.sonatype.com/repomanager3/product-information/download**](https://help.sonatype.com/repomanager3/product-information/download)

**Nexus Installation Process:**

**\*\* Use t2.medium or above configuration**

|  |  |
| --- | --- |
| Step | Command |
| Update OS | sudo yum update -y |
| Download wget | sudo yum install wget -y |
| Install Java Software | sudo yum install java-1.8.0-amazon-corretto -y |
| Download Nexus to /tmp | sudo wget https://download.sonatype.com/nexus/3/nexus-3.53.0-01-unix.tar.gz -P /tmp |
| Un-tar to /opt directory | sudo tar xvzf /tmp/nexus-3.53.0-01-unix.tar.gz -C /opt/ |
| Rename directory | sudo mv /opt/nexus-3.53.0-01/ /opt/nexus |
| Create nexus user | sudo useradd nexus |
| Allow User to execute commands with no password | visudo  # add below line under root line  nexus ALL=(ALL) NOPASSWD: ALL  (save and quit) |
| Change Owner to nexus | sudo chown -R nexus:nexus /opt/nexus  sudo chown -R nexus:nexus /opt/sonatype-work |
| Modify file permissions | sudo chmod -R 775 /opt/nexus  sudo chmod -R 775 /opt/sonatype-work |
| Enable nexus user | sudo vi /opt/nexus/bin/nexus.rc  run\_as\_user="nexus" |
| Create Nexus service file | sudo vi /etc/systemd/system/nexus.service  # then copy below content  [Unit]  Description=nexus service  After=network.target  [Service]  Type=forking  LimitNOFILE=65536  User=nexus  Group=nexus  ExecStart=/opt/nexus/bin/nexus start  ExecStop=/opt/nexus/bin/nexus stop  User=nexus  Restart=on-abort  [Install]  WantedBy=multi-user.target |
| Enable Nexus to use on restart | sudo systemctl enable nexus |
| Start nexus server | sudo systemctl start nexus |
| View Nexus status | sudo systemctl status nexus |
| Check Default port number | sudo cat /opt/nexus/etc/nexus-default.properties |
| Access Nexus using URL | http://<Public-IP>:8081/ |
| Check Admin default password | cat /opt/sonatype-work/nexus3/admin.password |

**Integrate Maven with Nexus:**

Create Repositories in Nexus to store build artifacts. We will create 2 types of repositories in Nexus

1) snapshot

2) release

"snapshot" and "release" repositories are commonly used to differentiate between different types of artifacts and their lifecycle stages. Here's an explanation of snapshot and release repositories:

**Snapshot Repository:**

* A snapshot repository is used to store artifacts that are under active development and subject to frequent changes.
* Snapshots are versions of artifacts that are still in-progress, not yet considered stable or finalized.
* Snapshot artifacts typically have version numbers with a suffix like **"-SNAPSHOT"** to indicate their status.
* Developers often publish and consume snapshot artifacts during the development and testing phases of a project.
* Snapshot repositories are usually used for fast iteration and collaboration among developers.

**Release Repository:**

* A release repository is used to store stable and finalized versions of artifacts.
* Release artifacts are considered to be in a production-ready state and are intended for use in deployed applications or projects.
* Release artifacts typically have version numbers without the **"-SNAPSHOT"** suffix.
* Release repositories are usually used for long-term storage, ensuring the stability and availability of dependencies for production environments.

**Why to maintain two repository types?**

1. **Dependency Management:** By keeping snapshot and release artifacts separate, developers can ensure that their projects only consume stable and tested versions of dependencies when in production. Snapshots can be used during development and testing to take advantage of the latest changes and bug fixes.
2. **Caching and Performance:** Separating snapshot and release repositories allows for different caching and update strategies. Snapshots may have shorter cache durations and more frequent updates, while releases can be cached for longer periods to reduce external network dependencies and improve build performance.
3. **Promoting Stability:** Requiring a deliberate promotion process from snapshot to release helps enforce quality control and ensures that only stable and approved versions are used in production environments.

Example Snapshot Repo URL: http://54.82.50.214:8081/repository/hippo-snapshot-repo/

Example Release Repo URL: http://54.82.50.214:8081/repository/hippo-release-repo/

Note: Based on <version/> name available in project pom.xml file it will decide artifacts should be stored to which repository

Nexus Repository details we will configure in project pom.xml file like below

<distributionManagement>

<repository>

<id>nexus</id>

<name>HIPPO Releases Nexus Repo</name>

<url>http://43.205.146.33:8081/repository/hippo-release-repository/</url>

</repository>

<snapshotRepository>

<id>nexus</id>

<name>HIPPO IT Snapshots Nexus Repo</name>

<url>http://43.205.146.33:8081/repository/hippo-snapshot-repository/</url>

</snapshotRepository>

</distributionManagement>

**Nexus Server Credentials will be configured in Maven "settings.xml" file**

Goto Maven Location : **/opt/maven/conf** (or) c:/Program Files/apache-maven-3.8/conf

In settings.xml file, under <servers> tag add below <server> tag

**<server>**

**<id>nexus</id>**

**<username>admin</username>**

**<password>admin</password>**

**</server>**

Once these details are configured then we can run below maven goal to upload build artifacts to Nexus Server

**$ mvn clean deploy**

Note: When we execute maven **deploy goal**, internally it will execute 'compile + test + package + install + deploy' goals.

compile: convert .java files into .class

test: Execute Unit Test cases

package: Create a build file like JAR or WAR File

install: Copy Build file to local repository (ex: C:/User/Hippo/.m2/repository/ location)

deploy: Copy Build file to Artifact Repository.