Components and Formats

What are Components?

In the context of the Nexus Repository component stands for any **package**, **library**, **binary**, **container**, or other resource produced or used by your software application. Components are also called artifacts, packages, bundles, archives in different platforms.

Components provide the **building blocks** of your application for the development team. Millions of components exist in the open-source community and can be used by developers. This is the most important advantage of component-based development since it prevents developers from working on something that is already built and losing time. For example, if a phonebook application was built previously by other developers and published in public repositories, you wouldn't have to build a new phonebook application for the mobile phone operating system you have been developing. This **saves time and increases efficiency** for your team.

Content of a Component

Usually, components contain different types of files such as:

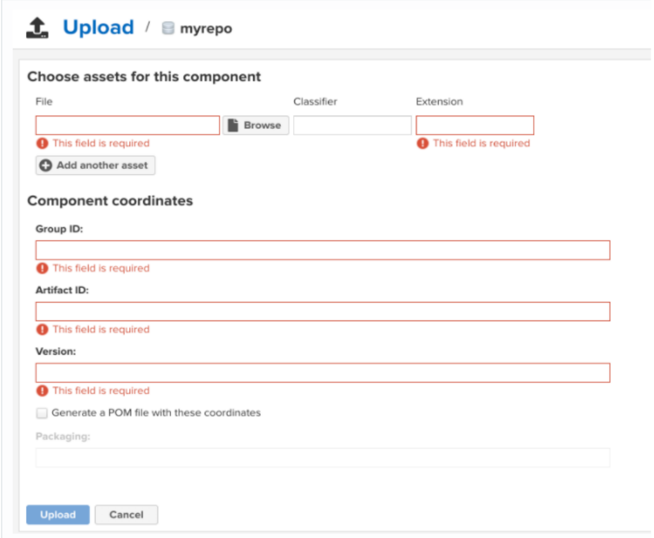
- Java byte code in class files.
- C object files
- Test files such as properties files, XML files, JavaScript code, HTML, CSS
- Binary files such as images, pdf files, sound files, etc.

Information is encoded as follows:

- java .jar, .war, ear file extensions.
- plain .zip or .tar.gz file extensions.
- package formats like nupkg, RPM, and gem.

Managing Components in Nexus Repository

Nexus repository manager allows you to manage components from development to delivery. These include binaries, containers, assemblies, and finished applications. Every component is identified with values depending on the format. For example, Maven components have three values such as **group**, **artifact ID**, and **version**.



Upload / myrepo

Choose assets for this component

File	Classifier	Extension
[ ] Browse — This field is required		This field is required

Add another asset

Component coordinates

Group ID:
This field is required

Artifact ID:
This field is required

Version:
This field is required

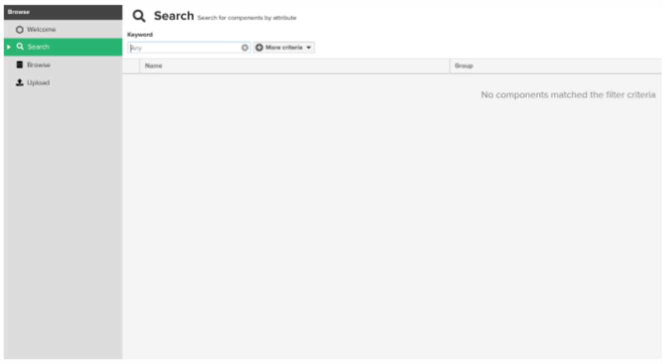
Generate a POM file with these coordinates

Packaging:

Upload   Cancel

Upload Component Screen

It is possible to search what you are looking for in the Nexus Repository. You can get the location and details of a component. These details include different versions that are available for that component and license data. This is important for build tool migrations, download of deployment packages, and other component related activities.



Browse
- Welcome
- Search
- Browse
- Upload

Search Search for components by attribute

Keyword: Any   More criteria

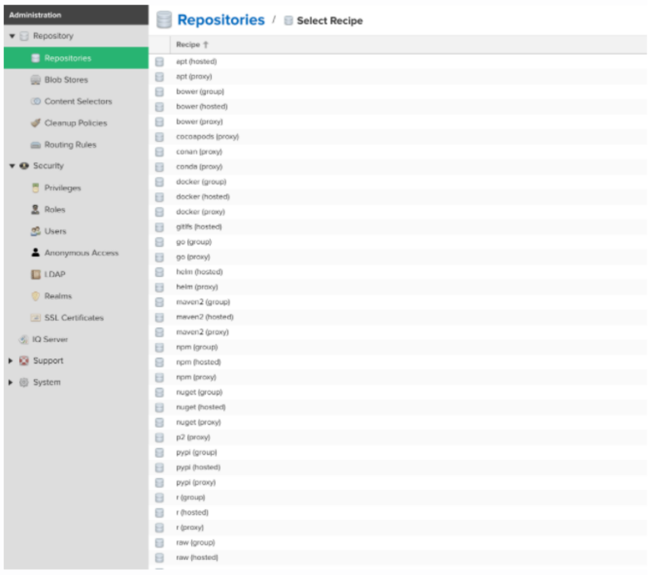
Name	Group

No components matched the filter criteria

Search screen

What are Formats?

Formats are way of communicating for storing, retrieving and indexing components and the metadata. There are different formats, for example Maven repository format has a specific directory structure defined by the component identifiers (GAV) and XML formatted files for metadata. Another example of a format is Bower repository format which consists of metadata with components stored in Git.



Administration
- Repository
  - Repositories
  - Blob Stores
  - Content Selectors
  - Cleanup Policies
  - Routing Rules
- Security
  - Privileges
  - Roles
  - Users
  - Anonymous Access
  - LDAP
  - Realms
  - SSL Certificates
- IQ Server
- Support
- System

Repositories / **Select Recipe**

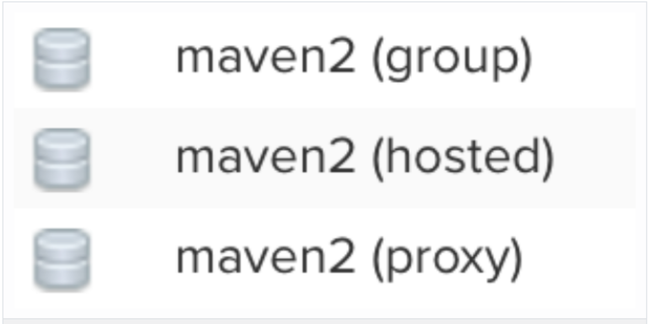
Recipe ↑
apt (hosted)
apt (proxy)
bower (group)
bower (hosted)
bower (proxy)
cocoapods (proxy)
conan (proxy)
conda (proxy)
docker (group)
docker (hosted)
docker (proxy)
gitlfs (hosted)
go (group)
go (proxy)
helm (hosted)
helm (proxy)
maven2 (group)
maven2 (hosted)
maven2 (proxy)
npm (group)
npm (hosted)
npm (proxy)
nuget (group)
nuget (hosted)
nuget (proxy)
p2 (proxy)
pypi (group)
pypi (hosted)
pypi (proxy)
r (group)
r (hosted)
r (proxy)
raw (group)
raw (hosted)

Some formats listed in the create repository screen

What are Repository Types?

There are three repository types in Nexus Repository. These are proxy repositories, hosted repositories, and repository groups. Each one of these is a different format. It is important to know the details about the repository types to be able to choose the fitting one for your organization.

For usage of public repositories, a proxy repository is the one to go with. If the components that your team uses are going to be shared among the other teams in the organization, a hosted repository is the most fitting one. If you would like to have multiple repositories that are reachable from a single URL, a group repository is the most appropriate one.


maven2 (group)

maven2 (hosted)

maven2 (proxy)

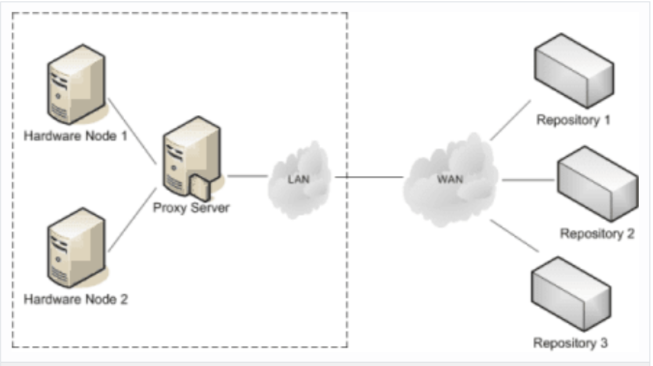
Repository Formats and Types in the Repository Creation Screen

Proxy Repository

A proxy repository is a repository that acts as a bridge between the computer and a remote repository. When you make a request to the proxy repository, it checks if the component already exists. If the component is not found, it retrieves it from a remote repository. Proxy repository acts as a cache and stores the component when it is retrieved. Since the proxy repository is in the local, whenever you ask for a component that you have asked previously, it provides the component without the need of retrieving it from a remote repository and using bandwidth.

Two proxy repositories that are configured and ready to use are ***maven-central*** and ***nuget.org-proxy***. Maven central is the built-in repository that comes with Apache Maven. nuget.org is the proxy repository that provides access to the NuGet gallery which is used in .NET (a framework for web development) development.

Also, proxy repositories for many tools including Docker is available within the Nexus Repository Manager.



Proxy Server Example

Hosted Repository

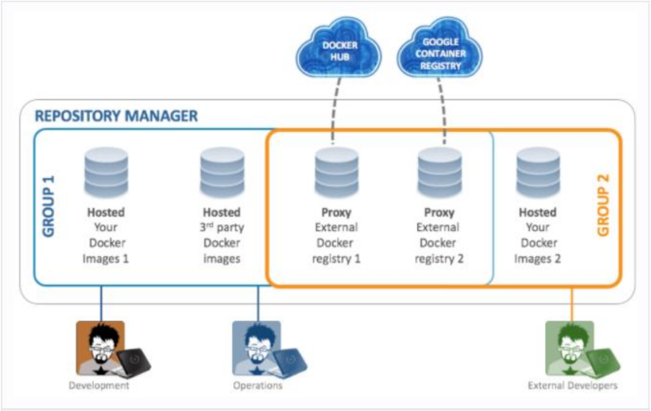
The hosted repository is the repository type that stores the components in a reliable way. It could be used for internal releases that are going to be shared inside your company. If there is a component that was created by your company and you are not using public repositories and would like to share this component with other teams in the organization, you could use ***hosted repositories***.

Configured hosted repositories that come with the nexus repository manager are as follows:

maven-releases: This hosted repository uses the maven2 repository format with a release version policy. It's intended to be the repository where your organization publishes internal releases. You can also use this repository for third-party components that aren't available in external repositories.

maven-snapshots: This hosted repository uses the maven2 repository format with a snapshot version policy. It's intended to be the repository where your organization publishes internal development versions, also known as snapshots.

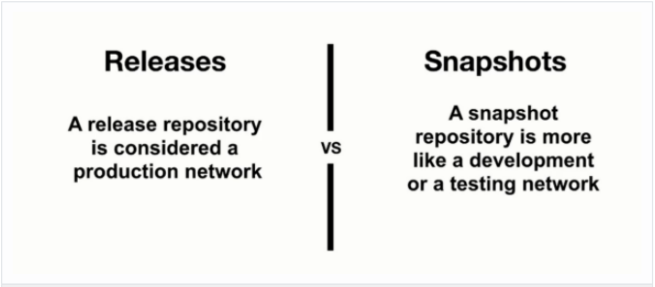
nuget-hosted: This hosted repository is where your organization can publish internal releases in the repository using the NuGet repository format. You can also use this repository for third-party components that are not available in external repositories that could potentially be proxied to gain access to the components.


DOCKER HUB
GOOGLE CONTAINER REGISTRY
REPOSITORY MANAGER
GROUP 1
Hosted Your Docker Images 1
Hosted 3rd party Docker images
Proxy External Docker registry 1
Proxy External Docker registry 2
Hosted Your Docker Images 2
GROUP 2
Development
Operations
External Developers

Utilization of Proxy and Hosted Repositories

Snapshots and Releases

There two properties to choose from when setting up a maven2 hosted repository. The difference between the release repository and snapshot repository is like the difference between production and development. It is recommended to use a release repository for stable components that are in the ***production phase***, a snapshot repository for the components that are in the ***development phase***. It is more complicated and takes much more time to deploy to a ***release repository***, while the ***snapshot repository*** is flexible and editable all the time.


Releases

A release repository is considered a production network

VS

Snapshots

A snapshot repository is more like a development or a testing network

Snapshots Vs. Releases

Group Repository

A group repository is a repository that consists of different formats. It allows users to connect to different repositories from the same URL. For example, a Maven Central proxy repository and a hosted repository for other JARs can be combined in a single group repository using the same URL.

Group configurations that come with the repository manager are **maven-public** and **nuget-group**.

Other repositories available for user configuration are:

- Bower
- Docker
- npm
- PyPI
- RAW -RubyGems

Maven Proxy Repository

What is Maven Proxy Repository?

As you already know, there are 3 types of repositories which are proxy, hosted and group. In this lesson, you are going to see an example of **proxy repository** for Maven, which is a popular build tool.

Proxy repositories are repositories that are linked to a remote repository. Whenever a component is required, first the proxy repository is checked, If the component is not present in the proxy repository, the component is downloaded from the remote repository and also cached in the proxy repository. So **proxy repository acts as a cache** for the components and other dependencies.

When building a project in Maven, you may need different components, libraries, packages, etc. So the **Maven Proxy repository** comes in handy for using those components effectively. Nexus Repository Manager comes with a configured Maven proxy repository. But in this lesson, you are going to see how to set it up from scratch.

Maven Configuration

To be able to use Maven with Nexus Repository Manager instead of Maven Central Repository, you have to set it up accordingly. You are going to add a mirror configuration instead of the default configuration.

To do so, you have to go to **/.m2/settings.xml** file and paste the following xml lines (**You have to change the username and password in respect to the username and password of the nexus repository**):

Tip: You might how to create the .m2 folder on your own if you haven't run **mvn install**. If that is the case, go inside your user directory and run **mkdir .m2**. Inside that, create the settings.xml file manually.

```
<settings>
  <mirrors>
    <mirror>
      <!--This sends everything else to /public -->
      <id>nexus</id>
      <mirrorOf>*</mirrorOf>
      <url>http://localhost:8081/repository/maven-proxy-test/</url>
    </mirror>
  </mirrors>
  <profiles>
    <profile>
      <id>nexus</id>
      <!--Enable snapshots for the built in central repo to direct -->
      <!--all requests to nexus via the mirror -->
      <repositories>
        <repository>
          <id>central</id>
          <url>http://central</url>
          <releases><enabled>true</enabled></releases>
          <snapshots><enabled>true</enabled></snapshots>
          <url>http://central</url>
          <releases><enabled>true</enabled></releases>
          <snapshots><enabled>true</enabled></snapshots>
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>
<activeProfiles>
    <!--make the profile active all the time -->
    <activeProfile>nexus</activeProfile>
  </activeProfiles>
  <servers>
    <server>
      <id>nexus</id>
      <username>admin</username>
      <password>your-password</password>
    </server>
  </servers>
</settings>
```

Once you have completed this, you will have your Maven ready for the Nexus repository manager.

Create a Maven Project and Proxy Repository

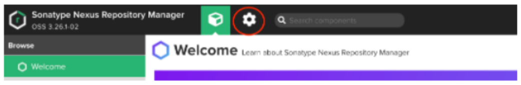
You have to have a project that is going to be deployed in the Nexus Repository. You need to create a POM file to initialize the Maven project. To do so, create a POM file in the directory of your own choice with the following POM.xml file.

1. Create a folder called ***maven-test***
2. Move into it and create the POM file (pom.xml).
3. Paste the following lines (or any XML format you like) inside the POM file.

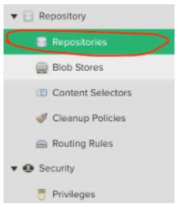
```
<project>
<modelVersion>4.0.0</modelVersion>
<groupId>com.example</groupId>
<artifactId>nexus-proxy</artifactId>
<version>1.0-SNAPSHOT</version>
<dependencies>
<dependency>
<groupId>junit</groupId>
<artifactId>junit</artifactId>
<version>4.10</version>
</dependency>
</dependencies>
</project>
```

Now that you have created the project, it is time to create the repository.

1. Proceed to the Nexus Repository user interface (It is the web address wherever you have started it. Local: http://localhost:8081/).
2. Proceed to the **Administration** screen.


Sonatype Nexus Repository Manager
OSS 3.26.1-02
Browse
Welcome
Welcome
Learn about Sonatype Nexus Repository Manager

3. Open ***Repositories*** tab.



4. Click ***Create Repository***
5. Choose ***maven2(proxy)***.
6. Set the ***name*** as ***maven-proxy-test*** and the ***remote storage URL*** as ***https://repo1.maven.org/maven2***
7. Click ***Create Repository*** and complete the process.

Now once you go to the directory where your POM file is located and run mvn package command and run your build, your components will have cached to the proxy repository that you have created. You can ***check by searching for the repository with its name, i.e maven-proxy-test***. You will see that all the components and directory format is cached in the nexus repository manager. Whenever you download a new component for the project, Maven will first check this proxy repository and if it cannot find the components you are looking for, it will download and cache it to the proxy repository.

Maven Hosted Repositories

What is Maven Hosted Repository?

You need proxy repository to cache components from public repositories. If you do not need public components and you are using components that were created by different teams in the same organization, you are going to need a hosted repository.

Maven hosted repository uses 2 different policies. These are ***maven-release-repository*** and ***maven-snapshots-repository***.

The release repository is used for components that are almost flawless and are ready for usage. These repositories can be used with components that are not available in eternal repositories and can't be obtained using a proxy repository.

Snapshot repository, unlike the release repository, is not intended to be used for completed and stable components. It can be changed and updated over time. The development phase is stored in the snapshot repository. In the POM file, the version value should end with ***-SNAPSHOT*** so that the snapshot repository is updated.

Maven Hosted Repositories

Maven Configuration

To be able to use the Nexus Hosted Repository with Maven, you have to change the configurations. You have to add the Maven Central repository, maven-snapshot-repository, and maven-release repository in the settings.xml file. The final version of the settings.xml file should look like this (***Do not forget to change the password accordingly to your repository's password***):

```
<settings>
  <mirrors>
    <mirror>
      <!--This sends everything else to /public -->
      <id>nexus</id>
      <mirrorOf>*</mirrorOf>
      <url>http://localhost:8081/repository/maven-public/</url>
    </mirror>
  </mirrors>
  <profiles>
    <profile>
      <id>nexus</id>
      <!--Enable snapshots for the built in central repo to direct -->
      <!--all requests to nexus via the mirror -->
      <repositories>
        <repository>
          <id>central</id>
          <url>http://central</url>
          <releases><enabled>true</enabled></releases>
          <snapshots><enabled>true</enabled></snapshots>
        </repository>
      </repositories>
     <pluginRepositories>
        <pluginRepository>
          <id>central</id>
          <url>http://central</url>
          <releases><enabled>true</enabled></releases>
          <snapshots><enabled>true</enabled></snapshots>
        </pluginRepository>
      </pluginRepositories>
  </profiles>
<activeProfiles>
    <!--make the profile active all the time -->
    <activeProfile>nexus</activeProfile>
  </activeProfiles>
  <servers>
    <server>
      <id>nexus</id>
      <username>admin</username>
      <password>your-password</password>
    </server>
  </servers>
</settings>
```

And your pom.xml file should look like this:

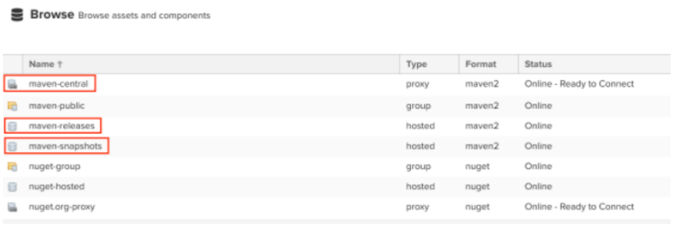
```
<project>
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.example</groupId>
  <artifactId>nexus-proxy</artifactId>
  <version>1.0</version>


<distributionManagement>
  <repository>
  <id>nexus</id>
  <name>maven-releases</name>
  <url>http://localhost:8081/repository/maven-releases/</url>
 </repository>
 <snapshotRepository>
  <id>nexus</id>
  <name>maven-snapshots</name>
  <url>http://localhost:8081/repository/maven-snapshots/</url>
 </snapshotRepository>
</distributionManagement>

 <dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.10</version>
  </dependency>
  </dependencies>
</project>
```

Create a Maven Hosted Repository

Since there is a Maven hosted repository that comes with the Nexus Repository Manager, unlike you did in the proxy repository, you do not have to create a new repository from scratch.


Browse Browse assets and components

Name ↑	Type	Format	Status
maven-central	proxy	maven2	Online - Ready to Connect
maven-public	group	maven2	Online
maven-releases	hosted	maven2	Online
maven-snapshots	hosted	maven2	Online
nuget-group	group	nuget	Online
nuget-hosted	hosted	nuget	Online
nuget.org-proxy	proxy	nuget	Online - Ready to Connect

To test the repositories after setting up the configurations, type ***mvn clean deploy*** in the directory of your projects pom.xml file

After executing the given command, once you search for the ***maven-snapshots*** repository in your nexus repository manager, you will see your projects files inside.

If you remove SNAPSHOT from your projects version tag in the pom.xml (1.0-SNAPSHOT to 1.0) and type ***mvn clean deploy*** in the command line, you will see that your files are uploaded to the ***maven-releases*** repository.

Maven Group Repositories

What is Maven Group Repository?

A group repository can be a collection of hosted, proxy, and other group repositories. It's specifications are:

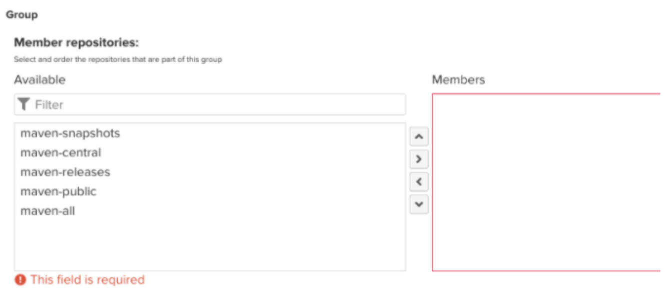
- It has its own access point.
- Both locally published and remotely cached components can be accessed.
- It is a unique access point where users can retrieve components for hosted and proxy repositories at the same time.

These specifications come in handy when more than one group is working on a project.

Tip: Group repositories do not store components directly. This means that you cannot upload components to group repositories, but you upload them to the hosted and proxy repositories within the group repository.

Creating the Group Repository and Configuring Maven

To be able to use Maven with a custom-built group repository, you have to configure it accordingly. First, you have to create the group repository from the UI of the Nexus Repository Manager. This is done exactly the same way it was done in Proxy Repository, except you choose ***maven2 (group)*** this time in the create repository screen. Once you are in the maven2 (group) configuration screen, set the name as ***maven-all*** for this tutorial. Finally, choose the repositories that you want to be contained by the group repository by dragging and dropping from left to right. You can choose the proxy and hosted repositories that were created in the previous tutorials.


Group

Member repositories:

Select and order the repositories that are part of this group

Available

Filter

maven-snapshots
maven-central
maven-releases
maven-public
maven-all

Members

This field is required

Once you have created the group repository, now you have to configure Maven so that it can know that you are going to be using a custom group repository. Thus, all you have to do is add the URL of the repository that you have created in the settings.xml file. You have to change the ***URL tag*** in the mirror section to ***http://localhost:8081/repository/maven-all/***.

Once everything is ready, run ***mvn install***. When the terminal prompts BUILD SUCCESS, you know that all the components are downloaded from the group repository. You can go and check for the components from the UI with the browse property of Nexus Repository Manager.