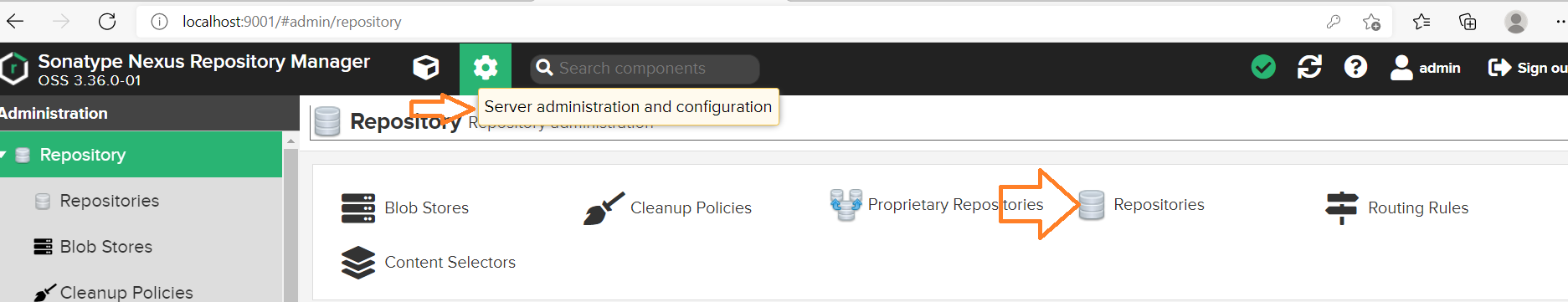
**Create repository**

Click on **Server administration** **and** **configuration** button and click on **Repositories**

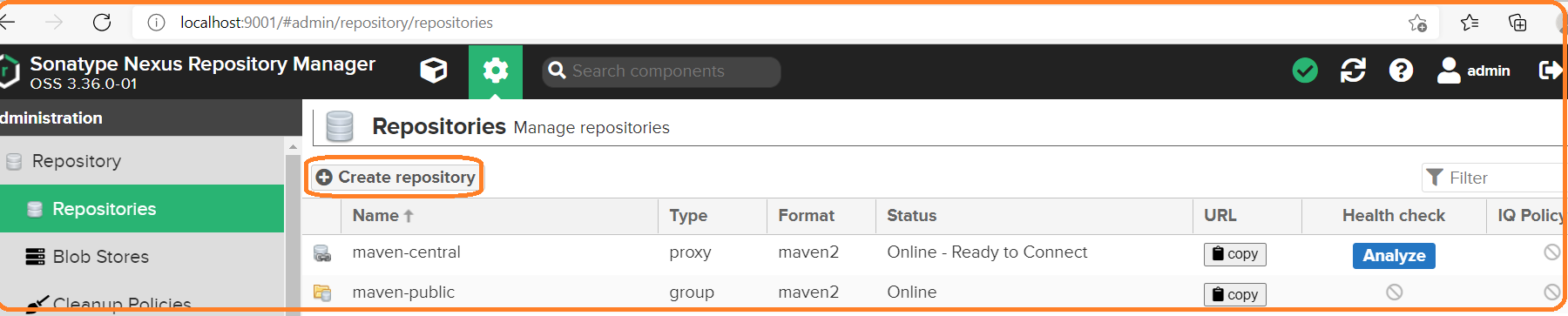
****

🡪Each project we are going to create 2 repositories for same project

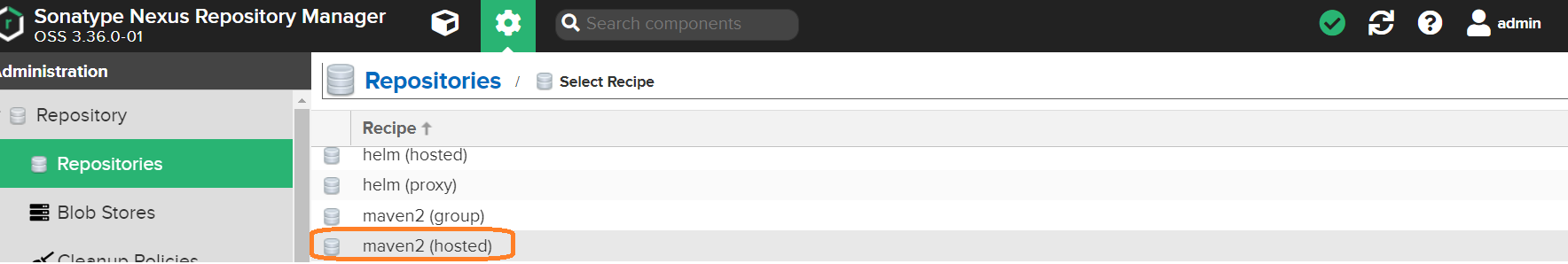
🡪In real time if any project is on boarded we have to create 2 projects

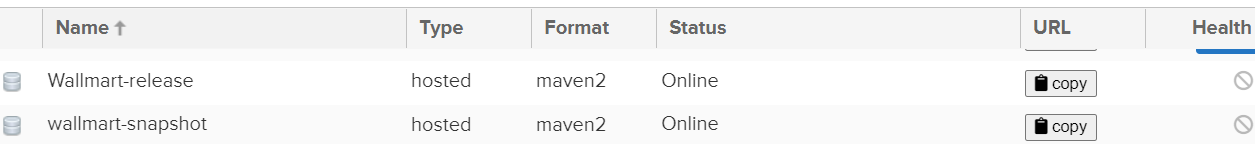
* SNAPSHOT for Development reference
* RELEASE for production reference

Click on **Create repository**



🡪If it is maven and java project we have to select recipe type as **maven2 hosted**





Click on **copy** button and configure in **pom.xml** file

**Example:**

<http://localhost:9001/repository/Wallmart-release/>

<http://localhost:9001/repository/wallmart-snapshot/>

**Before configuring nexus details in pom.xml file follow below 2 steps:**

🡪Download project from GITHUB using **git clone** command

git clone <URL>

🡪Run **mvn clean install** for storing the artifact in local repository

**Configure Nexus repository details in pom.xml file:**

**Case: 1**

**<version>1.0.0-SNAPSHOT</version>**

<distributionManagement>

<repository>

<id>nexus</id>

<name>Wallmart Release Repo</name>

<url>http://localhost:9001/repository/Wallmart-release/</url>

</repository>

<snapshotRepository>

<id>nexus</id>

<name>Wallmart Snapshot</name>

<url>http://localhost:9001/repository/wallmart-snapshot/</url>

</snapshotRepository>

</distributionManagement>

**name :** we can give any name for reference

**url :**

http://<public-ip>:<port-no>/repository/repository-name 🡪Linux

[http://localhost:<port-no>/repository/repository-name](http://localhost:%3cport-no%3e/repository/repository-name) 🡪Windows

**Configure server details in settings.xml**

**C:\Program Files\apache-maven-3.6.1\conf\settings.xml**

<server>

<id>nexus</id>

<username>admin</username>

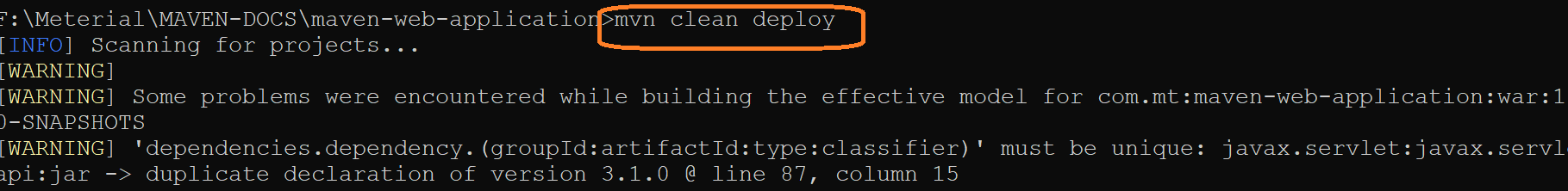
<password>\*\*\*\*\*\*\*\*</password>

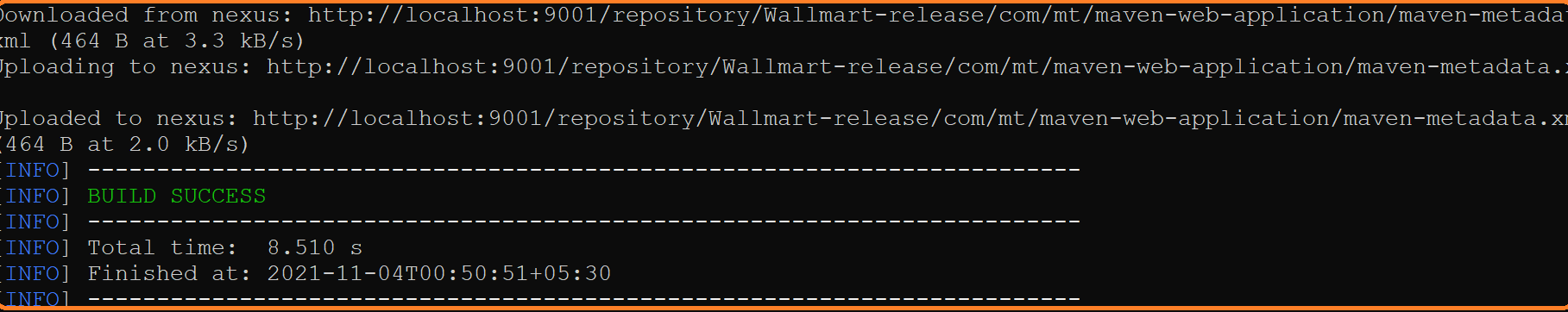
</server>

🡪**Note:** id value should be same is pom.xml file id value

**Storing artifact in nexus repository**

Go to project folder and run below command for storing artifact in nexus repository



****

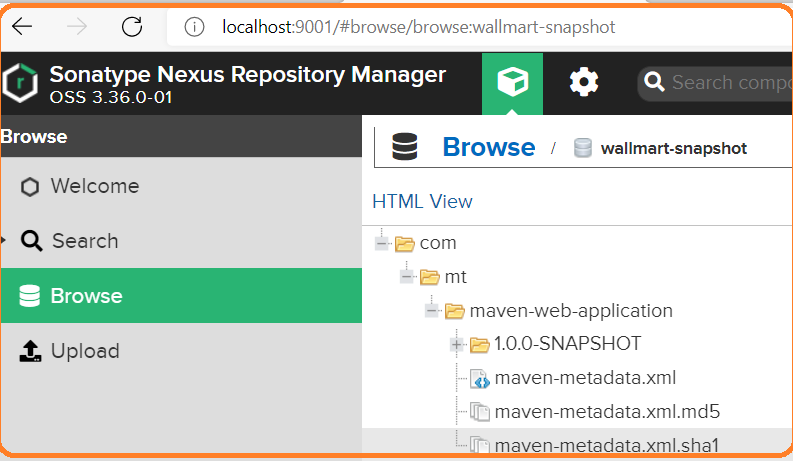
🡪Login into nexus server and verify whether artifact is stored or not

🡪Based on version tag artifact will be stored either SNAPSHOT repo or RELEASE repo

🡪If we configure version tag as **1.0.0-SNAPSHOT** then artifact will be stored **SNAPSHOT repository with same name**

🡪If we configure version tag as 1.1.0 then artifact will be stored in **release repository with same name**

**SNAPSHOT repository**

****

**Case 2:**

**<version>1.0.0-SNAPSHOT</version>**

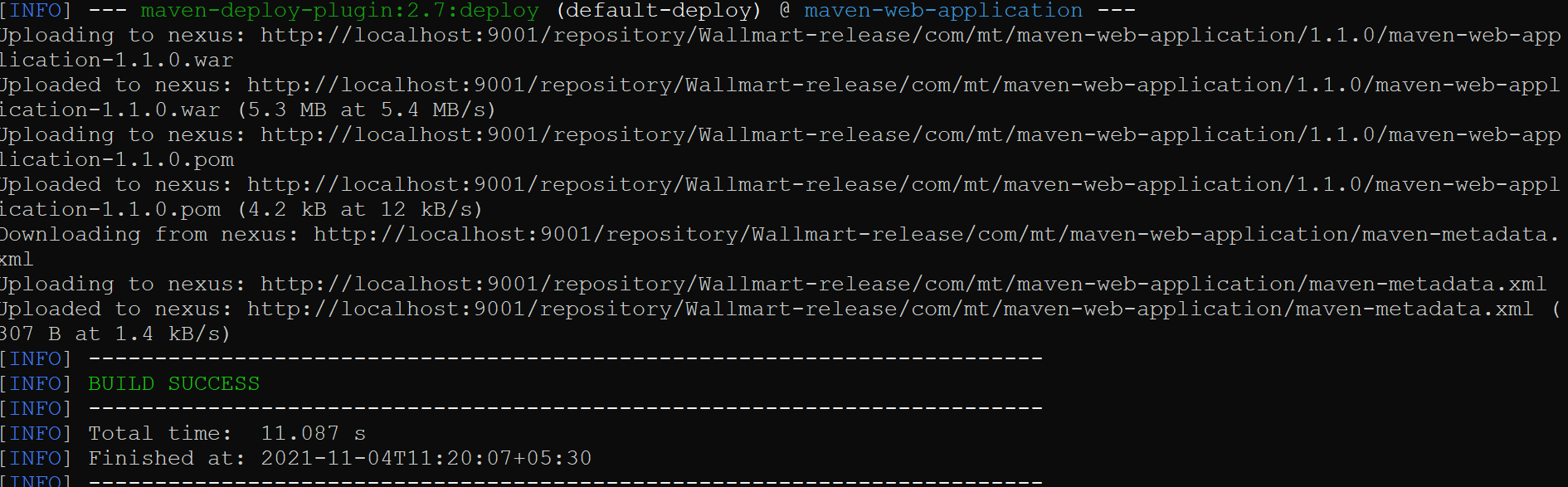
if we give other than **1.0.0-SNAPSHOT** keyword in this case artifact will be stored in **RELEASE** repository

Change version number as below in pom.xml and run below command

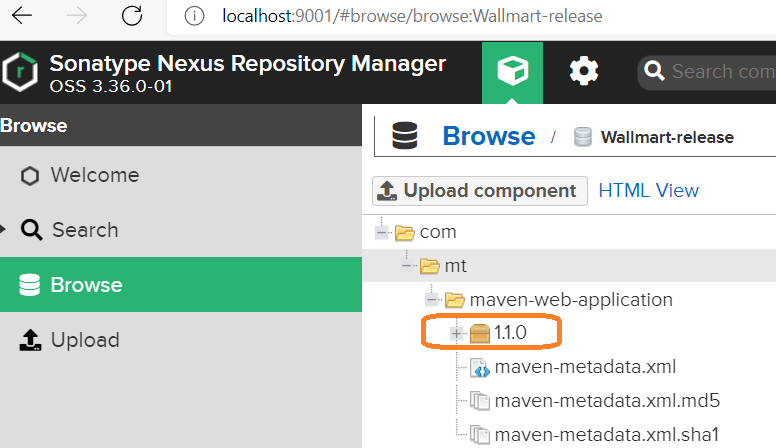
**F:\Meterial\MAVEN-DOCS\maven-web-application\pom.xml**

**<version>1.1.0</version>**

**F:\Meterial\MAVEN-DOCS\maven-web-application>mvn clean deploy**

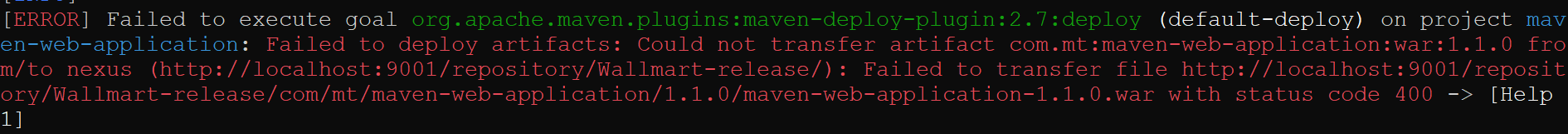
****

Once we receive above message from console, we can login check nexus server for artifact storage status



**How to override production release version in nexus server:**

Now, we have uploaded **1.1.0 artifacts** in nexus server, if we try to upload same version we will get exception

****

**Status code 400 : Bad request**

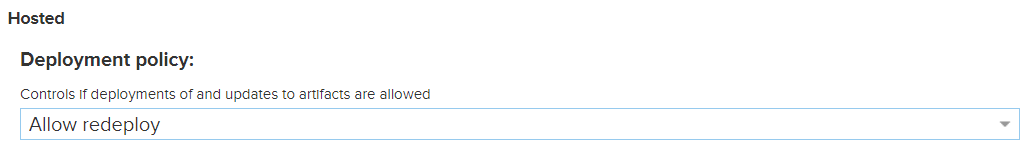
If we want to override same version with minor changes in production relase

We need to fallow below steps

🡪Login nexus UI as admin

**Server administration and configuration 🡪 Repositories 🡪 click on release repository(Wallmart-release)**

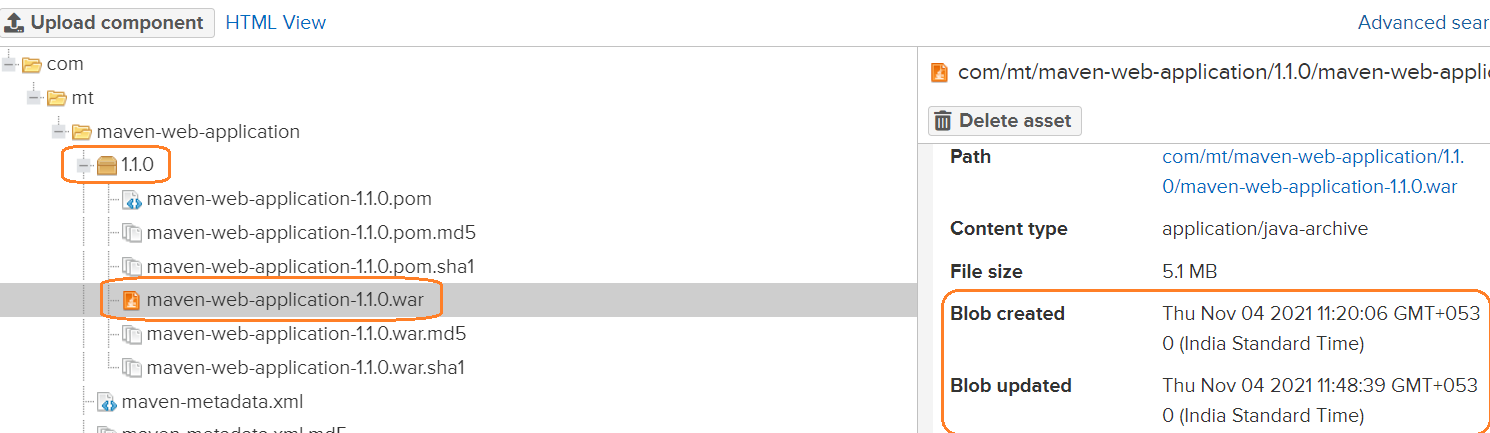
Select **deployment policy** as **Allow redeploy instead of Disable redepoy**



🡪Save and exit

🡪Run maven command for re-deploy

🡪Same version will be upload with minor changes



**Note:**

If we observe Blob created time stamp and updated stamp will be different means artifacts has been uploaded successfully.

**Q)Can we download the package outside the network?**

Ans)In real time, We are not able to access package repository URL outside the network, each company will have own network settings within that network IP range only we can access the server

If we want to use same package in some other project we will use below tag in pom.xml

(i.e other project pom.xml file)



**Different types of recipe type:**

**Maven2 (Proxy)**

When we creating repository we will use the below recipe type

Maven2 (hosted)

Maven2 (Proxy)

Maven2 (group)

We have downloaded java project in laptop and when we executing below command

**mvn clean package**

Flow will be, whatever we mentioned in pom.xml first it will check in MLR(Maven Local Repository)

If not it will go to the MCR (Maven Central Repository) from their it will going to download in to maven local repository (MLR) from their going to use for project

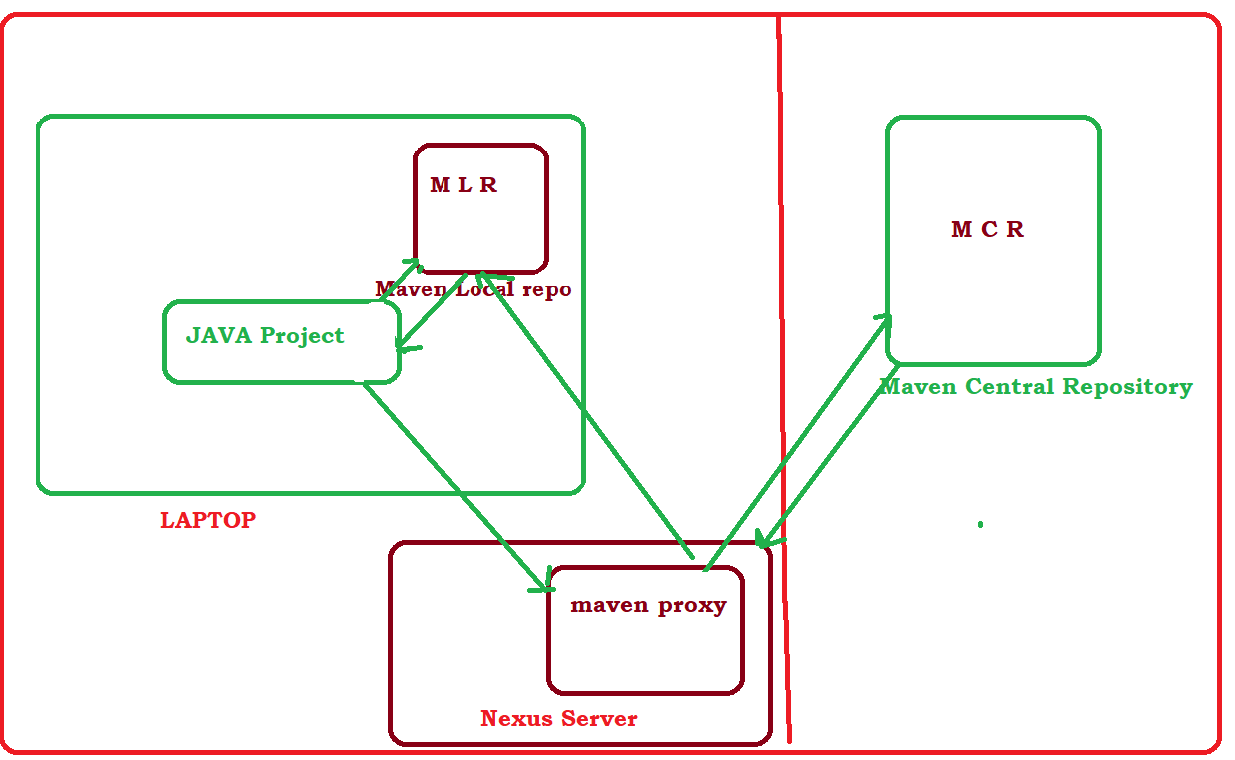
In this process if we are download critical project like banking domain related project our server is in company network and maven central repository is public network any one can download while download this dependencies virus may get download

If we don’t want to download dependencies directly from central repository instead of that one proxy repository will created

We will configure proxy repository in one of the linux mechine so it will secure if any virus attack it will remove

This proxy repo details will be configure in project pom.xml file

In this approach, maven build system first it will check in maven local server if not it will check in maven proxy repository if not it will go to maven central repository from their it will download in to maven proxy repository and from there it will download in to maven local repository from their it will use to project.

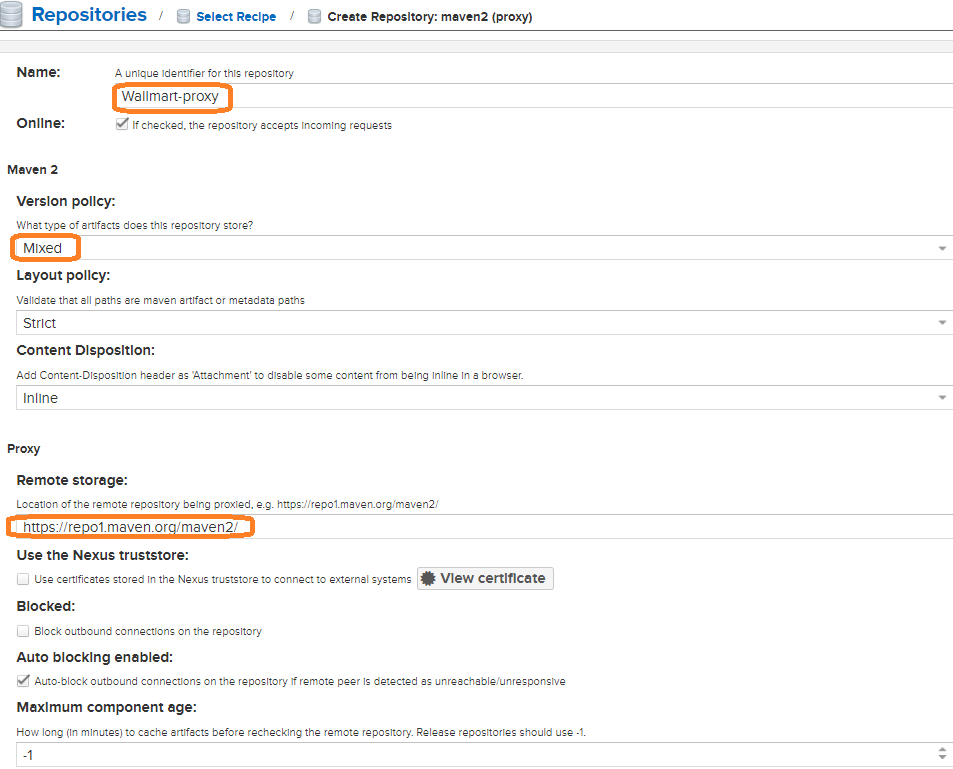


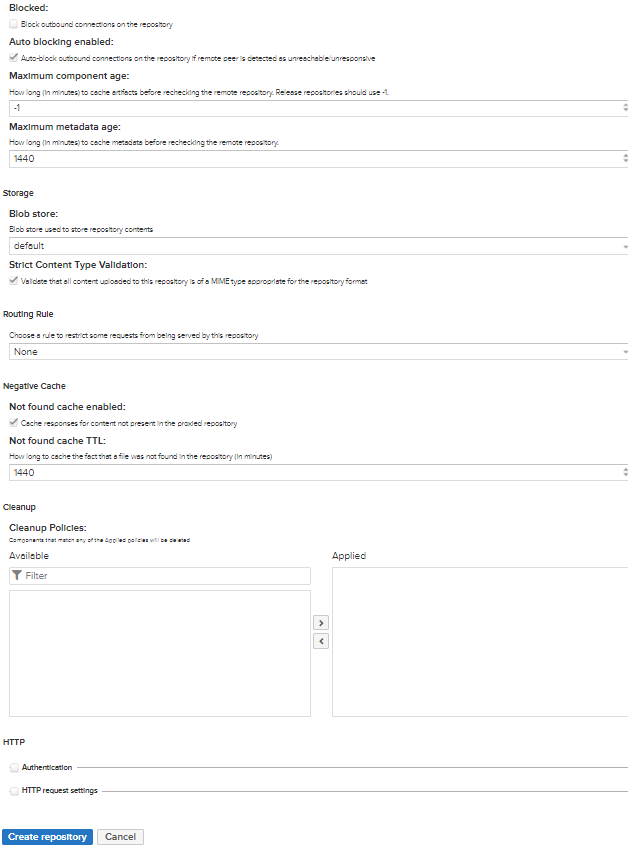
**Build system using maven2 proxy**:

🡪Create maven proxy repository from Nexus UI dash board

🡪 Click on **Server administration** **and** **configuration** button and click on **Repositories**

**Server administration and configuration🡪Repositories🡪Create repository🡪maven2(proxy)**





Update above highlighted points and proceed to create repository

Configure proxy repository url in pom.xml file under project directory

<repositories>

<repository>

<id>nexus</id>

<name> Proxy Repo</name>

<url> http://localhost:9001/repository/Wallmart-proxy/</url>

</repository>

</repositories>

Update below xml entry in **settings.xml** file

**C:\Program Files\apache-maven-3.6.1\conf\settings.xml**

<mirrors>

<mirror>

<id>nexuss</id>

<mirrorOf>\*</mirrorOf>

<name>Proxy repo</name>

<url>http://localhost:9001/repository/Wallmart-proxy/</url>

</mirror>

</mirrors>

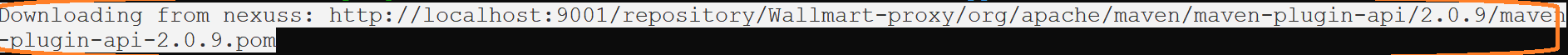
If we run below command from project folder

**mvn clean deploy**

maven build system first it will check in

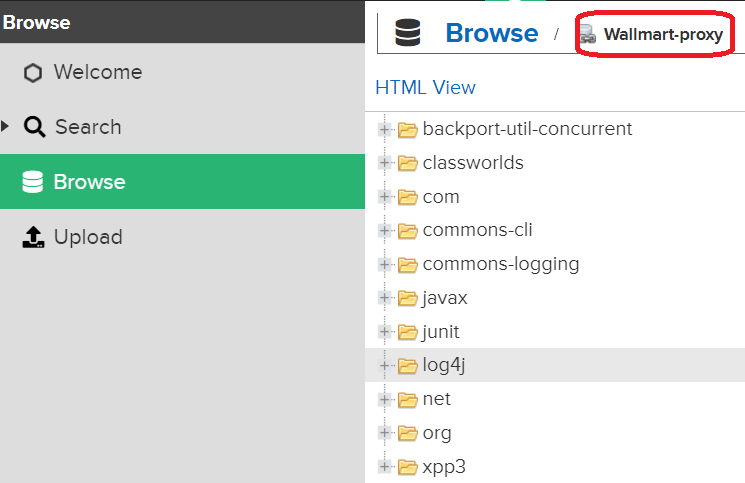
**Jave project 🡪 MavenLocalRepository 🡪 MavenProxyRepository 🡪 MavenCentralRepository**

**MavenCentralRepository 🡪 MavenProxyRepository 🡪 MavenLocalRepository 🡪 Java project**



Once BUILD SUCCESS we could see below entry in nexus UI

**Login nexus UI as a ADMIN🡪 Browse 🡪Click on custom proxy rep(i.e wallmart-proxy**



**<mirrorOf>\*</mirrorOf> :** By seeing this tag maven build system will fetch the dependencies from maven proxy repo

**Maven Remote Repository:**

We have 3 maven repositories

* Maven Local Repository (MLR)
* Maven Central Repository (MCR)
* Maven Remote Repository (MRR)

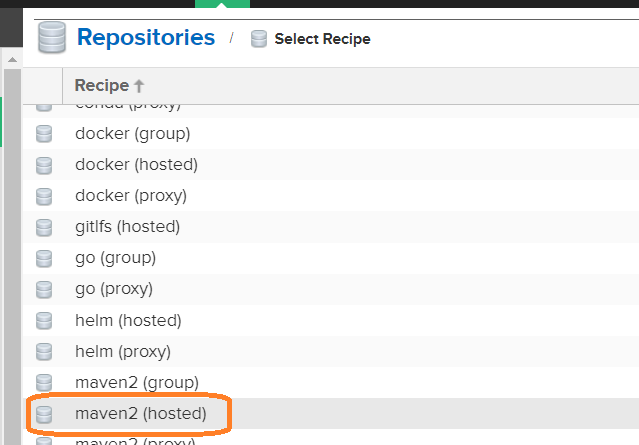
Organization has developed few custom jars/wars files they want to use same jars files for internal usage for within organization only, if we upload in maven central repository anybody can access but this are dedicated to particular organization used only within organization what are the projects they have developed.

If we want to use within organization we can upload this custom jar/war files in maven remote repository(MRR) and download within organization and used for all the project this process is also called as **shared libraries** (i.e dedicated to particular organization)

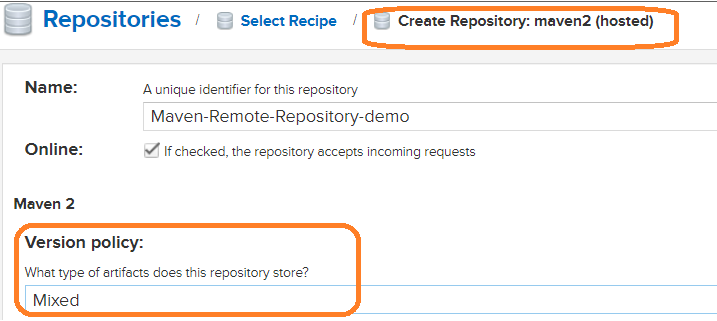
Maven remote repository is dedicated to particular organization

To create maven remote repository we can use recipe type as **maven (hosted)**

**Server administration and configuration🡪Repositories🡪Create repository🡪maven2(hosted)**

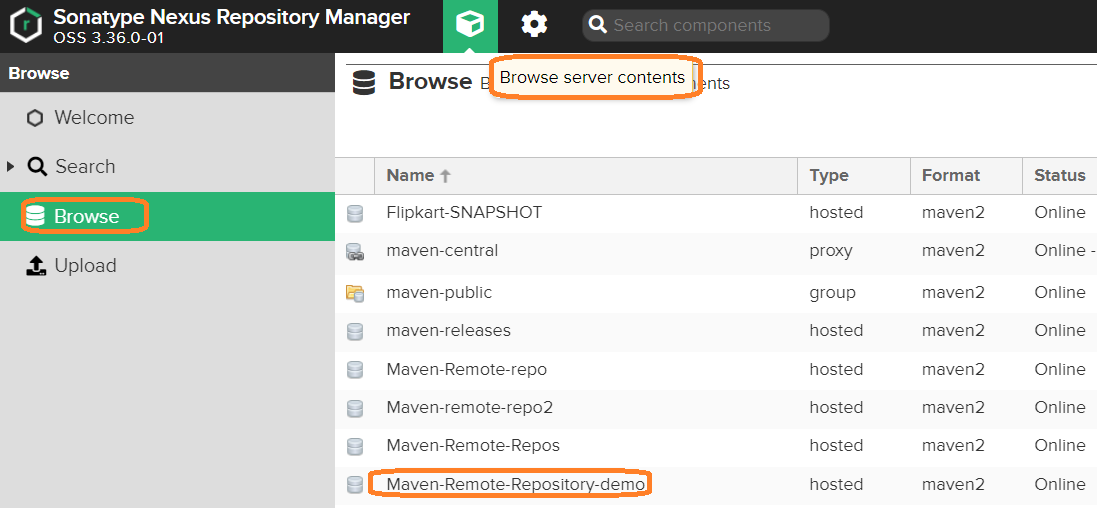
****

Give unique identifier name for repository and select Mixed for Version policy

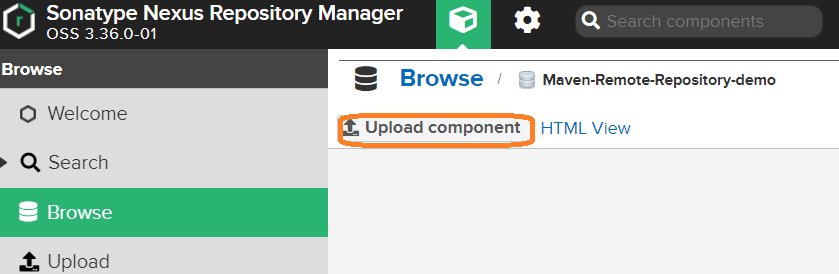


Live all options as default and click on **create repository**

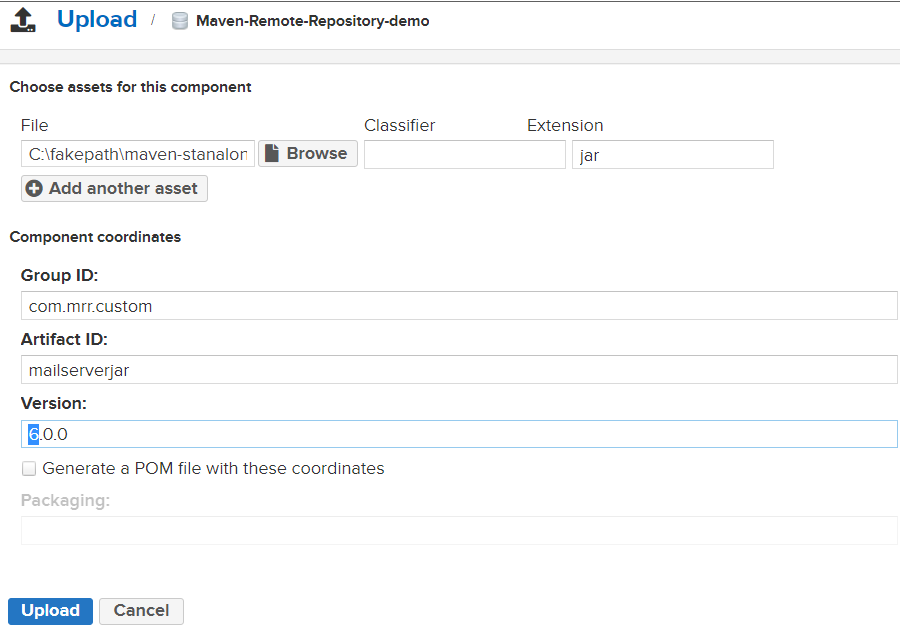
Click on **Browse server contents** button and click on **Browse** from left panel

****

Click on **Maven-Remote-Repository-demo**(i.e repository name), we will get below page

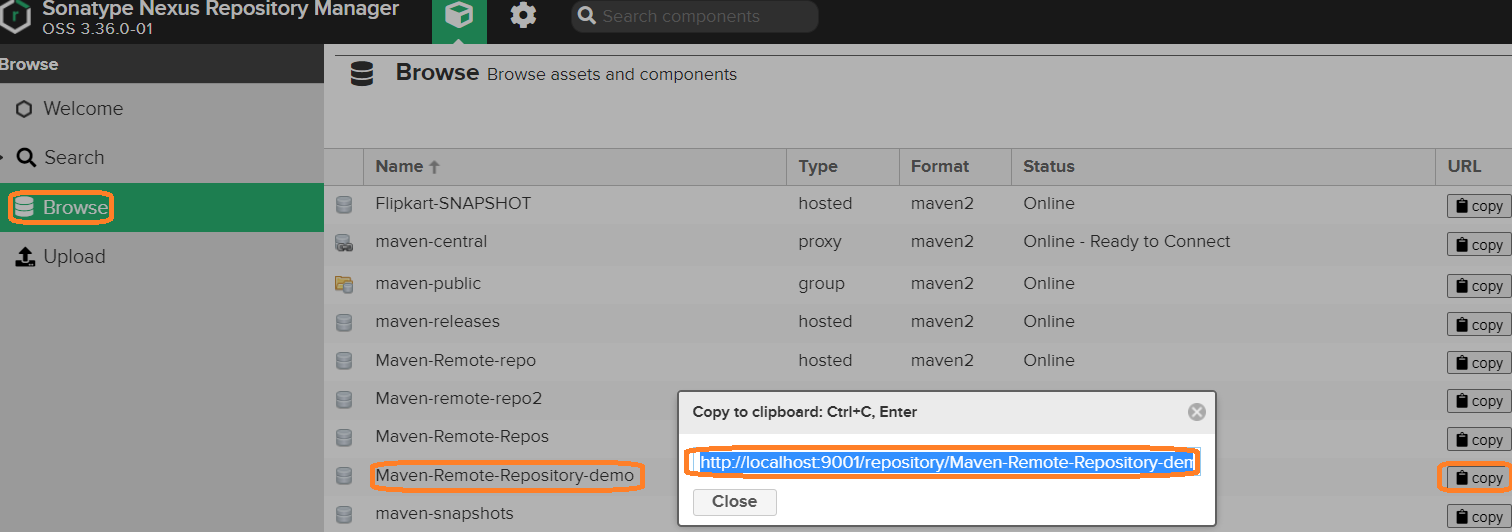


Here, we can upload custom jar/war/ear files for internal usage, click on **Upload component** button

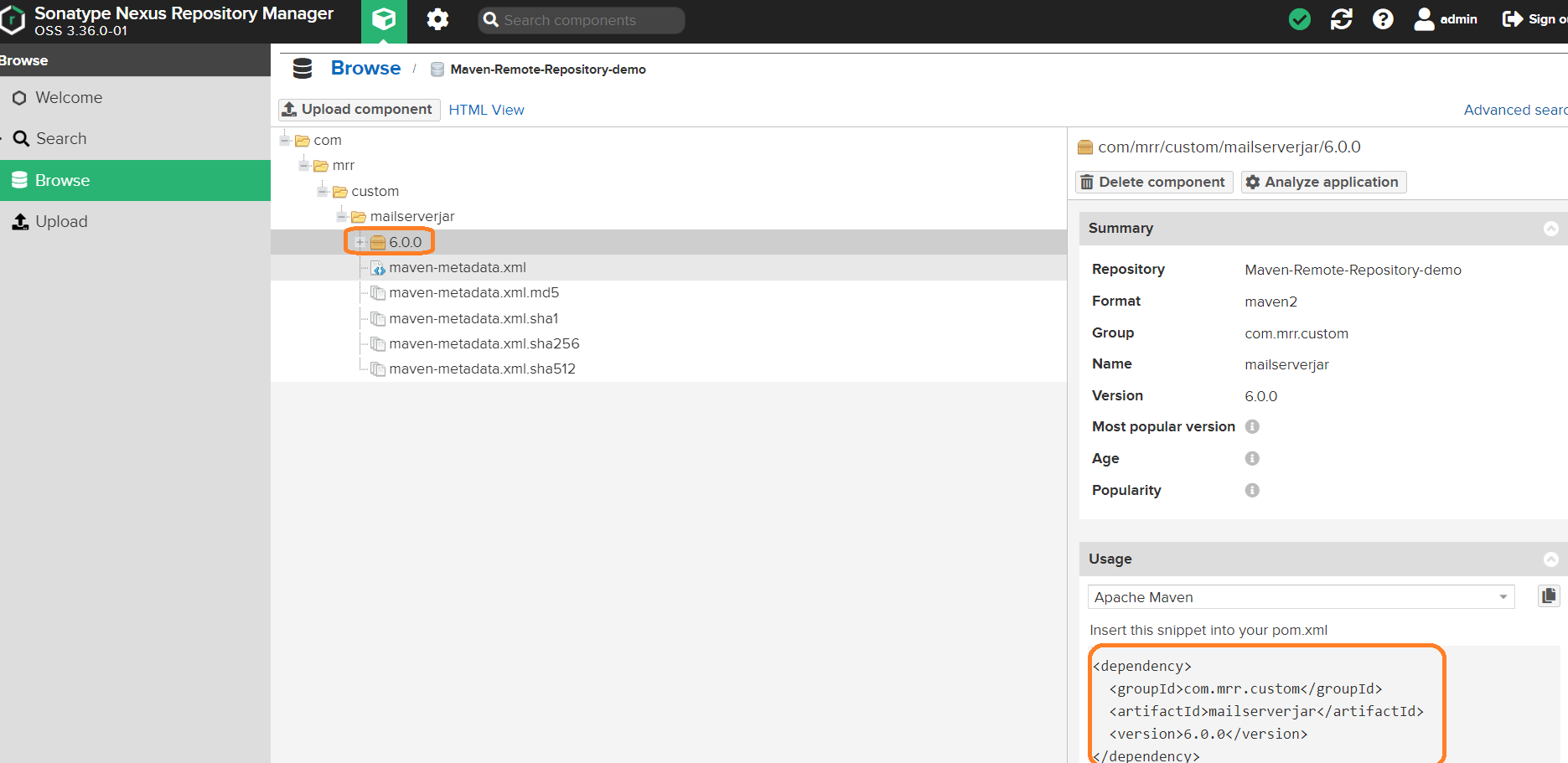


Provide GAV details and Click on **Upload** button

Copy maven remote repository url from nexus UI



Click on Maven-Remote-Repository-demo button we will get below page.



If we want to use this jar file to some other project, then we have copy this jar file dependency update it in pom.xml (other project)

For example : F:\Meterial\MAVEN-DOCS\maven-web-application\pom.xml

<dependency>

<groupId>com.mrr.custom</groupId>

<artifactId>mailserverjar</artifactId>

<version>6.0.0</version>

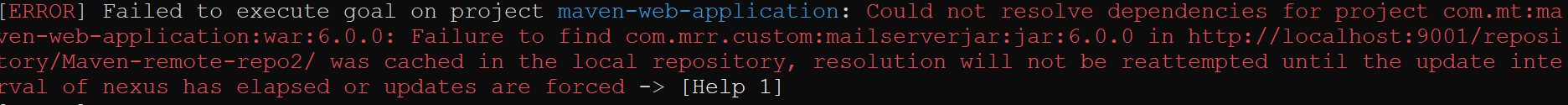
</dependency>

Update under **dependencies** section

If run **mvn clean package** command from project folder, we will get below exception

Because **maven build system** will check dependencies in maven local repo if not it will check remote repo if not central repo if not available then we will get below exception

But actual dependencies avaible maven remote repository which is not configure in pom.xml file



To resolve this issue we have configure custom repository url in pom.xml file as below

Update maven-remote-repository url in **pom.xml** file under repositories section

**F:\\*\*\*\\*\*\*\maven-web-application\pom.xml**

<repositories>

<repository>

<id>nexuss</id>

<name>Maven remote repository</name>

<url> http://localhost:9001/repository/Maven-Remote-Repository-demo/</url>

</repository>

</repositories>

**C:\Program Files\apache-maven-3.6.1\conf\settings.xml**

<mirror>

<id>nexus</id>

<mirrorOf>\*</mirrorOf>

<name>Maven remote repository</name>

<url>http://localhost:9001/repository/Maven-Remote-Repository-demo/</url>

</mirror>

Now, if we run **mvn clean package**

Dependencies will be downloaded from maven remote repository

