maven私服我相信很多公司都有，私服的好处有以下几点：
1、 节省下载资源开销。jar包（不一定是jar，也可以是其他资源）都存在私服，可以不必每次下载都去远程仓库去下载，因为有的远程仓库确实下载很慢。
2、 自定义jar包。每个公司都有自己的jar包资源，这些明显是在远程仓库搜索不到的，而搭建私服，就可以将自定义jar包放到私服，公司的同事就可以从私服下载。
3、 私服是自己的，自己的东西好管理！

1、环境安装

1.1、安装jdk

参考文章：https://blog.csdn.net/liu1160848595/article/details/102838318

1.2、安装maven

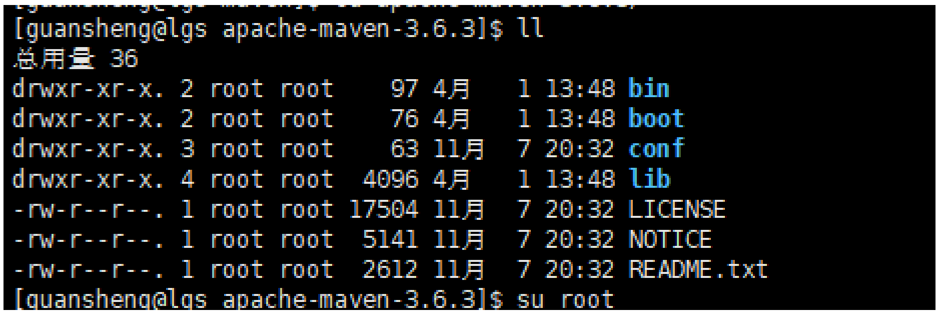
linux下安装maven步骤非常简单，总结起来就三步：下载安装包、解压、配置环境变量

1. 下载安装包，官网下载地址：http://maven.apache.org/download.cgi
   此处我下载了当前最新版本：3.6.3

```
[guansheng@lgs maven]$ wget https://mirror.bit.edu.cn/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz
```

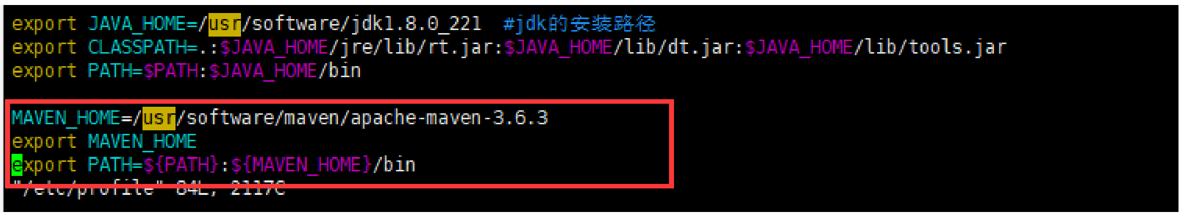
2. 解压安装包到指定目录

```
#当前用户权限不够，所以索性直接用sudo权限解压
[guansheng@lgs maven]$ sudo tar -zxvf apache-maven-3.6.3-bin.tar.gz -C /usr/software/maven/
```

可以看到解压目录下的文件结构如下：

```
[guansheng@lgs apache-maven-3.6.3]$ ll
总用量 36
drwxr-xr-x. 2 root root    97 4月   1 13:48 bin
drwxr-xr-x. 2 root root    76 4月   1 13:48 boot
drwxr-xr-x. 3 root root    63 11月  7 20:32 conf
drwxr-xr-x. 4 root root  4096 4月   1 13:48 lib
-rw-r--r--. 1 root root 17504 11月  7 20:32 LICENSE
-rw-r--r--. 1 root root  5141 11月  7 20:32 NOTICE
-rw-r--r--. 1 root root  2612 11月  7 20:32 README.txt
[guansheng@lgs apache-maven-3.6.3]$ su root
```

3. 配置环境变量
   在 /etc/profile文件中加入以下代码：

```
MAVEN_HOME=/usr/software/maven/apache-maven-3.6.3
export MAVEN_HOME
export PATH=${PATH}:${MAVEN_HOME}/bin
```



```
export JAVA_HOME=/usr/software/jdk1.8.0_221  #jdk的安装路径
export CLASSPATH=.:$JAVA_HOME/jre/lib/rt.jar:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar
export PATH=$PATH:$JAVA_HOME/bin

MAVEN_HOME=/usr/software/maven/apache-maven-3.6.3
export MAVEN_HOME
export PATH=${PATH}:${MAVEN_HOME}/bin
"/etc/profile" 84L, 2117C
```

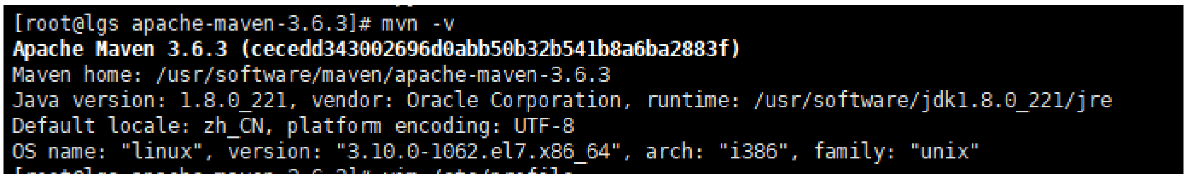
其中MAVEN_HOME的值就是maven的解压之后的目录
保存文件，并运行source /etc/profile使环境变量生效

```
[root@lgs apache-maven-3.6.3]# source /etc/profile
```

4. 检查是否安装成功
   运行mvn -v查看maven版本

```
[root@lgs apache-maven-3.6.3]# mvn -v
```

看到以下结果则证明安装成功了



```
[root@lgs apache-maven-3.6.3]# mvn -v
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: /usr/software/maven/apache-maven-3.6.3
Java version: 1.8.0_221, vendor: Oracle Corporation, runtime: /usr/software/jdk1.8.0_221/jre
Default locale: zh_CN, platform encoding: UTF-8
OS name: "linux", version: "3.10.0-1062.el7.x86_64", arch: "i386", family: "unix"
```

1.3、安装nexus

nexus工具是此次maven私服的构建工具，所以也要安装好。
nexus的安装也很简单，下载、解压、修改配置文件就可以访问了

1. 下载nexus
   官网地址：https://www.sonatype.com/download-oss-sonatype
   linux系统下载unix版本的就可以。
   官网可以下载最新版，但是下载速度简直慢得令人发指，最后不知道到哪个角落翻到一篇文章里说用某雷下载比较快，就试了一下，果然很快，如果下载很慢可以用！
   题外话不多说，假设你的网速很快，那可以执行以下命令下载最新安装包：

```
[root@lgs nexus]# wget https://download.sonatype.com/nexus/3/latest-unix.tar.gz
```

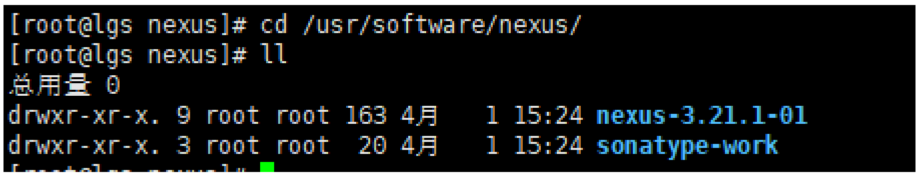
我是下载到本地，然后上传到服务器的：

```
#用rz命令上传，前提是已经安装了rz命令
[root@lgs nexus]# rz -by
```

2. 解压安装包到指定文件夹

```
[root@lgs nexus]# tar -zxvf latest-unix.tar.gz -C /usr/software/nexus/
```

解压后有两个文件夹，前面那个是功能的实现，后面那个文件夹负责存储数据，也就是构件（jar包之类的资源），如下所示：



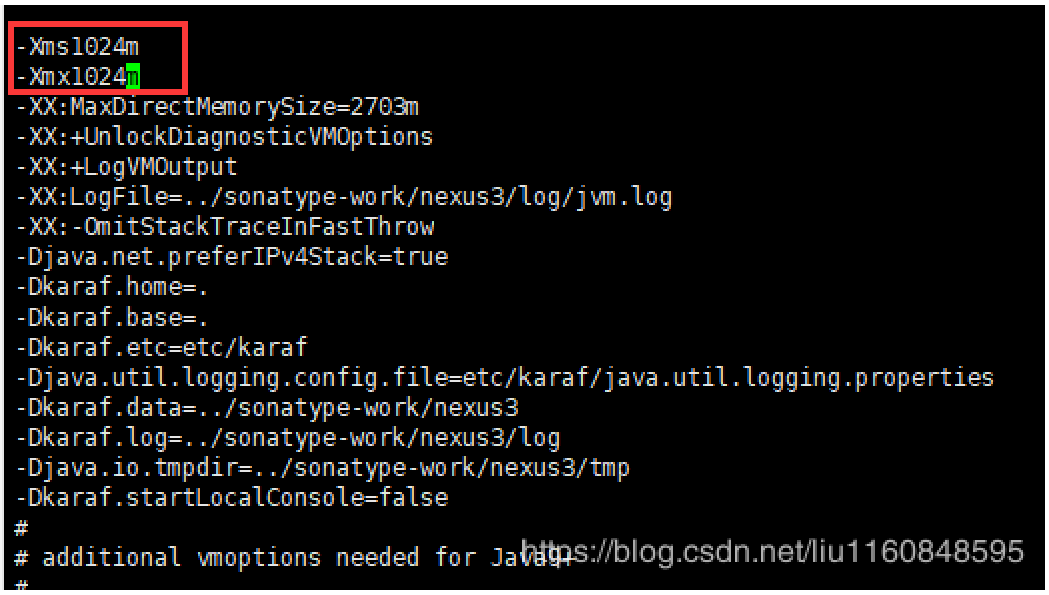
```
[root@lgs nexus]# cd /usr/software/nexus/
[root@lgs nexus]# ll
总用量 0
drwxr-xr-x. 9 root root 163 4月   1 15:24 nexus-3.21.1-01
drwxr-xr-x. 3 root root  20 4月   1 15:24 sonatype-work
```

我们操作第一个文件夹即可。

3. 修改配置
   如果机器够好的话，可以使用默认配置就可以了，我的服务器配置不高，所以改一下虚拟机配置：

```
[root@lgs bin]# vim nexus-3.21.1-01/bin/nexus.vmoptions
```

将启动内存和最大内存都改为1G

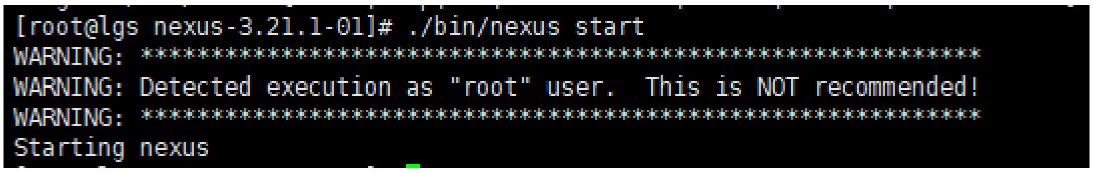


```
-Xms1024m
-Xmx1024m
-XX:MaxDirectMemorySize=2703m
-XX:+UnlockDiagnosticVMOptions
-XX:+LogVMOutput
-XX:LogFile=../sonatype-work/nexus3/log/jvm.log
-XX:-OmitStackTraceInFastThrow
-Djava.net.preferIPv4Stack=true
-Dkaraf.home=.
-Dkaraf.base=.
-Dkaraf.etc=etc/karaf
-Djava.util.logging.config.file=etc/karaf/java.util.logging.properties
-Dkaraf.data=../sonatype-work/nexus3
-Dkaraf.log=../sonatype-work/nexus3/log
-Djava.io.tmpdir=../sonatype-work/nexus3/tmp
-Dkaraf.startLocalConsole=false
#
# additional vmoptions needed for Java
```

4. 启动nexus

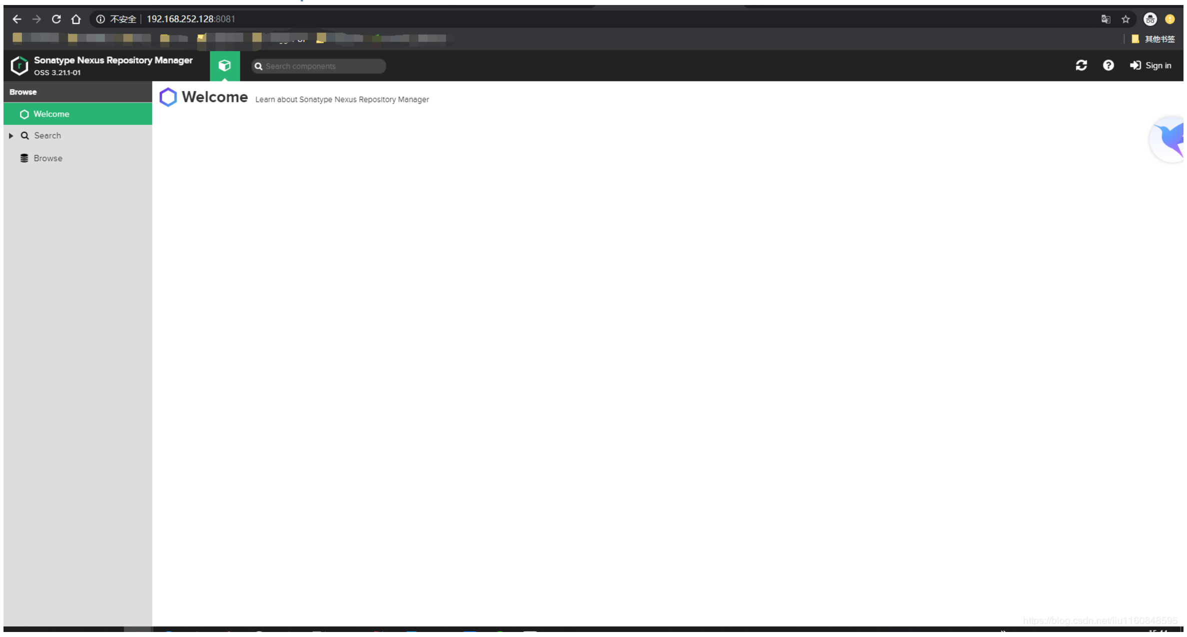
```
[root@lgs nexus-3.21.1-01]# ./bin/nexus start
```

此时会有一个警告，意思是不推荐用root用户启动，不过此警告不影响使用，可以不管



```
[root@lgs nexus-3.21.1-01]# ./bin/nexus start
WARNING: ************************************************************
WARNING: Detected execution as "root" user.  This is NOT recommended!
WARNING: ************************************************************
Starting nexus
```

此时可以直接访问了：http://192.168.252.128:8081/

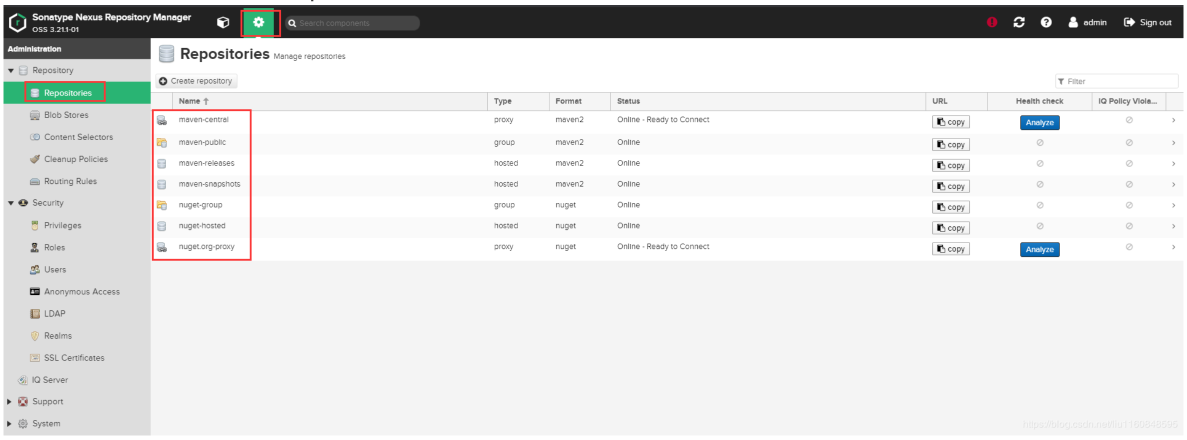

不安全 | 192.168.252.128:8081
其他书签
Sonatype Nexus Repository Manager
OSS 3.21.1-01
Search components
Sign in
Browse
Welcome
Search
Browse
Welcome
Learn about Sonatype Nexus Repository Manager

5. 说明
   1) Nexus默认的端口是8081,可以在sonatype-work/nexus3/etc/nexus.properties中修改
   2) 右上角那个登录会有一个默认的admin账号，初始密码保存在/usr/software/nexus/sonatype-work/nexus3/admin.password文件中，登录进去会让你重置密码。

2、搭建maven私服

2.1、说明

进入到nexus界面，点击repositories可以看到以下界面：



Name	Type	Format	Status	URL	Health check	IQ Policy Viola...
maven-central	proxy	maven2	Online - Ready to Connect	copy	Analyze	⊘
maven-public	group	maven2	Online	copy	⊘	⊘
maven-releases	hosted	maven2	Online	copy	⊘	⊘
maven-snapshots	hosted	maven2	Online	copy	⊘	⊘
nuget-group	group	nuget	Online	copy	⊘	⊘
nuget-hosted	hosted	nuget	Online	copy	⊘	⊘
nuget.org-proxy	proxy	nuget	Online - Ready to Connect	copy	Analyze	⊘

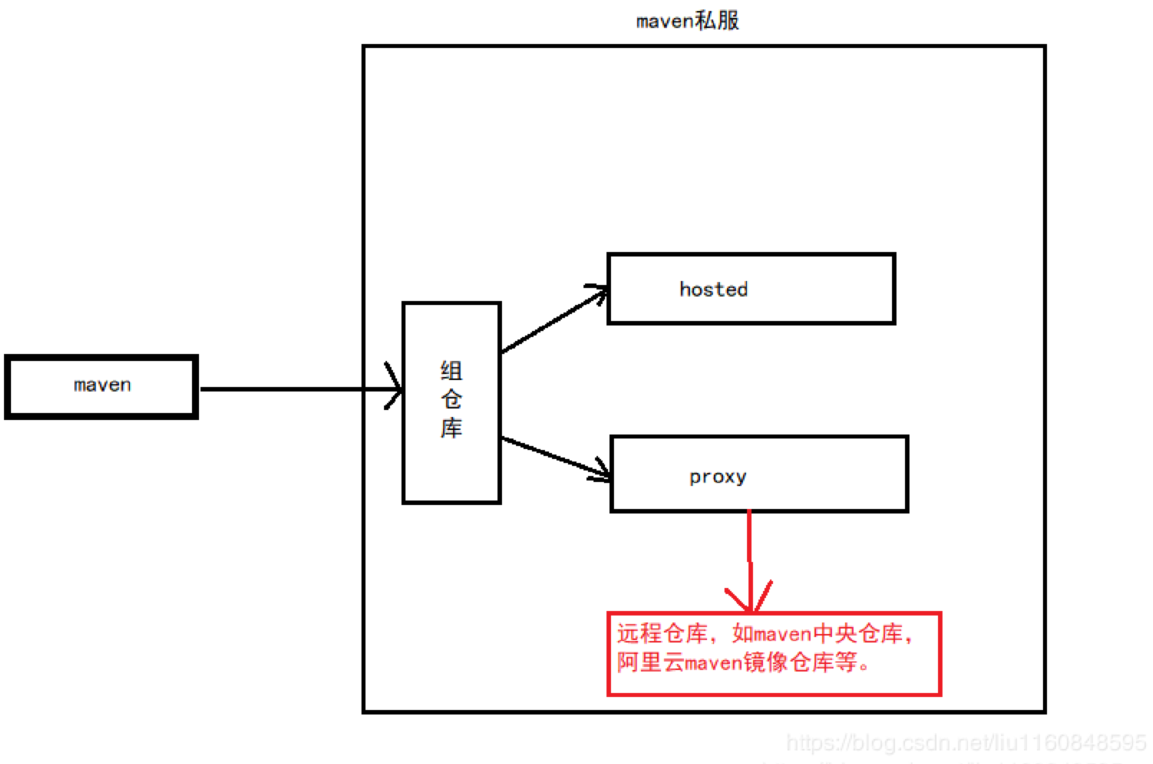
下面是网上摘抄的一些关于上面名词的说明：

1. component name的一些说明：
   1）maven-central：maven中央库，默认从https://repo1.maven.org/maven2/拉取jar
   2）maven-releases：私库发行版jar
   3）maven-snapshots：私库快照（调试版本）jar
   4）maven-public：仓库分组，把上面三个仓库组合在一起对外提供服务，在本地maven基础配置settings.xml中使用。
2. Nexus默认的仓库类型有以下四种：
   1）group(仓库组类型)：又叫组仓库，用于方便开发人员自己设定的仓库；
   2）hosted(宿主类型)：内部项目的发布仓库（内部开发人员，发布上去存放的仓库）；
   3）proxy(代理类型)：从远程中央仓库中寻找数据的仓库（可以点击对应的仓库的Configuration页签下Remote Storage Location属性的值即被代理的远程仓库的路径）；
   4）virtual(虚拟类型)：虚拟仓库（这个基本用不到，重点关注上面三个仓库的使用）；
3. Policy(策略):表示该仓库为发布(Release)版本仓库还是快照(Snapshot)版本仓库；
4. Public Repositories下的仓库
   1）3rd party: 无法从公共仓库获得的第三方发布版本的构件仓库，即第三方依赖的仓库，这个数据通常是由内部人员自行下载之后发布上去；
   2）Apache Snapshots: 用了代理ApacheMaven仓库快照版本的构件仓库
   3）Central: 用来代理maven中央仓库中发布版本构件的仓库
   4）Central M1 shadow: 用于提供中央仓库中M1格式的发布版本的构件镜像仓库
   5）Codehaus Snapshots: 用来代理CodehausMaven 仓库的快照版本构件的仓库
   6）Releases: 内部的模块中release模块的发布仓库，用来部署管理内部的发布版本构件的宿主类型仓库；release是发布版本；
   7）Snapshots:发布内部的SNAPSHOT模块的仓库，用来部署管理内部的快照版本构件的宿主类型仓库；snapshots是快照版本，也就是不稳定版本
   所以自定义构建的仓库组代理仓库的顺序为：Releases，Snapshots，3rd party，Central。也可以使用oschina放到Central前面，下载包会更快。

2.2、构建思路

首先要明白我们从私服上拉取的jar包的来源，既然是私服，那肯定少不了自定义的jar包，这些jar包是在宿主仓库，也就是hosted类型的仓库，除此之外还有其他的jar包，需要从远程仓库下载，也就是proxy代理类型的仓库。既然来源有多个，我们为了方便，通常会创建一个group,也就是组类型的仓库，将宿主类型和代理类型的包含进去，我们只需要访问组仓库即可，至于组内是先从hosted中拉取还

是先从proxy中拉取，就要看创建组仓库时候配置的属性了，下面会有介绍：

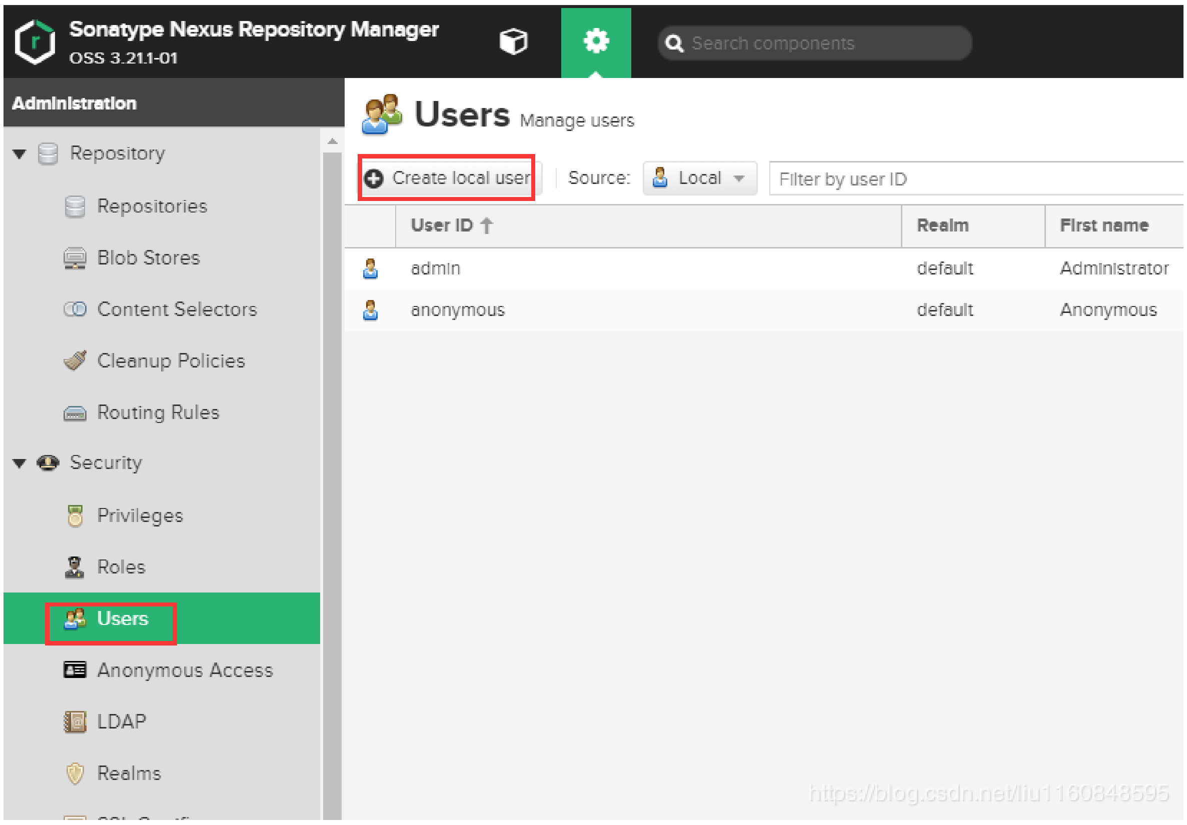

maven私服
maven
组仓库
hosted
proxy
远程仓库，如maven中央仓库，阿里云maven镜像仓库等。
https://blog.csdn.net/liu1160848595

为了安全性，我们还可以新建一个user共外部访问，按需分配角色。

2.3、搭建步骤

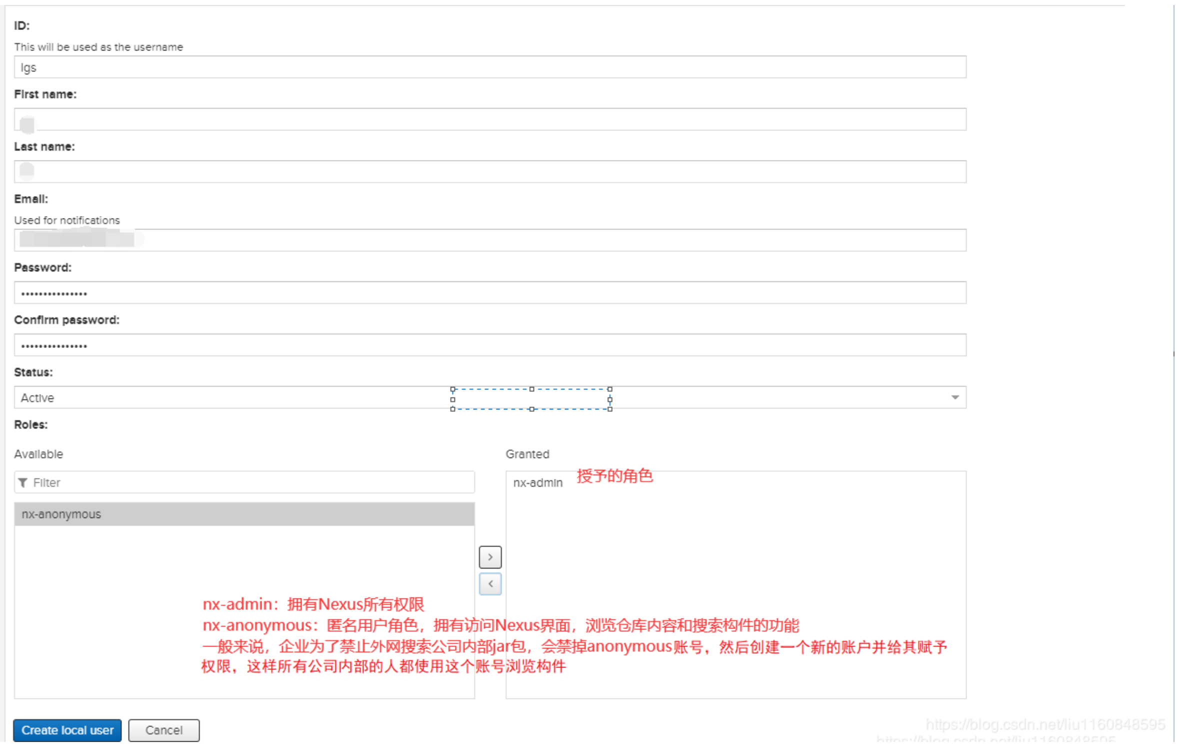
按照上面的思路，开始创建一个user和一个hosted仓库、proxy仓库和group仓库

2.3.1、创建用户



	User ID	Realm	First name
	admin	default	Administrator
	anonymous	default	Anonymous

填上必要的信息以及授权


ID:
This will be used as the username
lgs
First name:

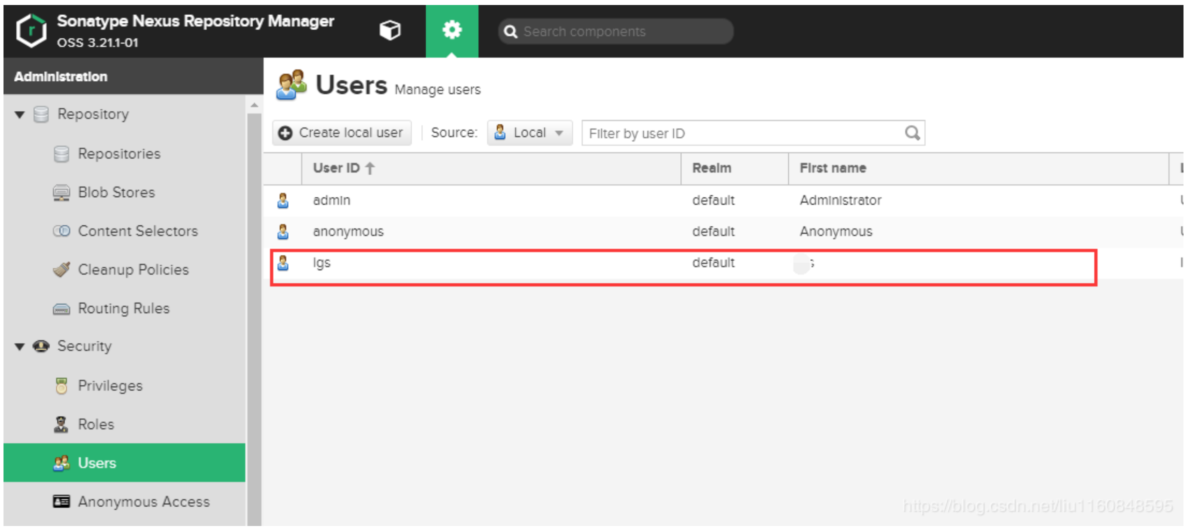
Last name:

Email:
Used for notifications

Password:
················
Confirm password:
················
Status:
Active
Roles:
Available
Filter
nx-anonymous
Granted
nx-admin 授予的角色

nx-admin：拥有Nexus所有权限
nx-anonymous：匿名用户角色，拥有访问Nexus界面，浏览仓库内容和搜索构件的功能
一般来说，企业为了禁止外网搜索公司内部jar包，会禁掉anonymous账号，然后创建一个新的账户并给其赋予权限，这样所有公司内部的人都使用这个账号浏览构件

点击创建之后，可以在用户列表中看到新建的用户



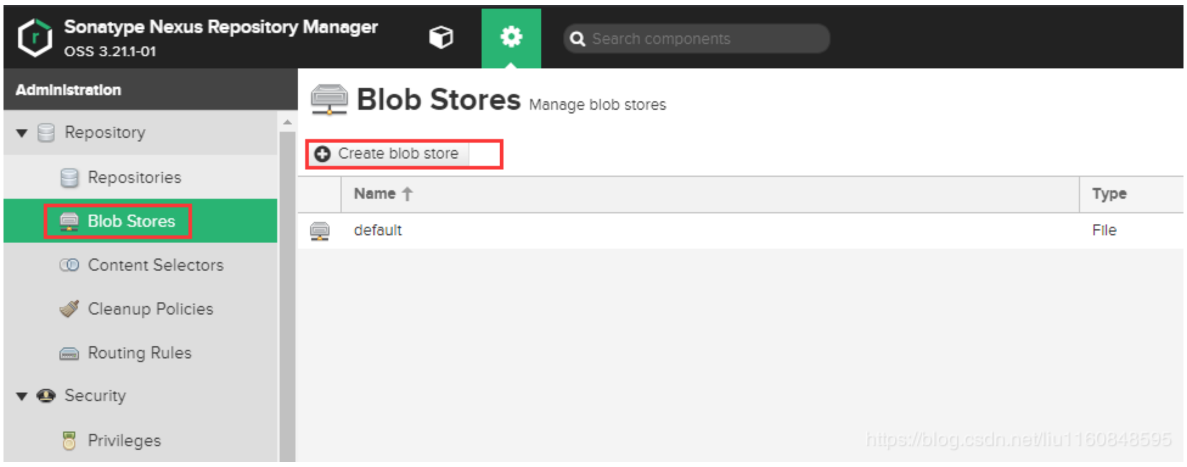
Users Manage users

Create local user | Source: Local | Filter by user ID

User ID	Realm	First name
admin	default	Administrator
anonymous	default	Anonymous
lgs	default	;

2.3.2、创建文件保存目录

创建仓库之前，可以先创建数据文件的保存目录

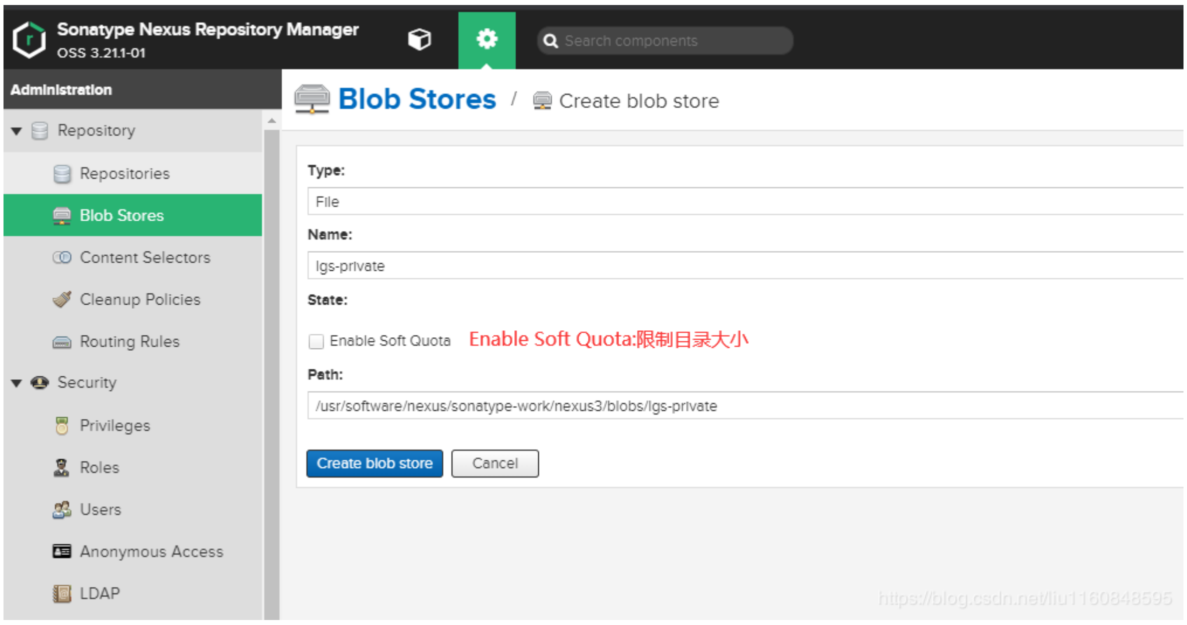


Blob Stores Manage blob stores

Create blob store

Name	Type
default	File

同样填上必要信息



Blob Stores / Create blob store

Type: File

Name: lgs-private

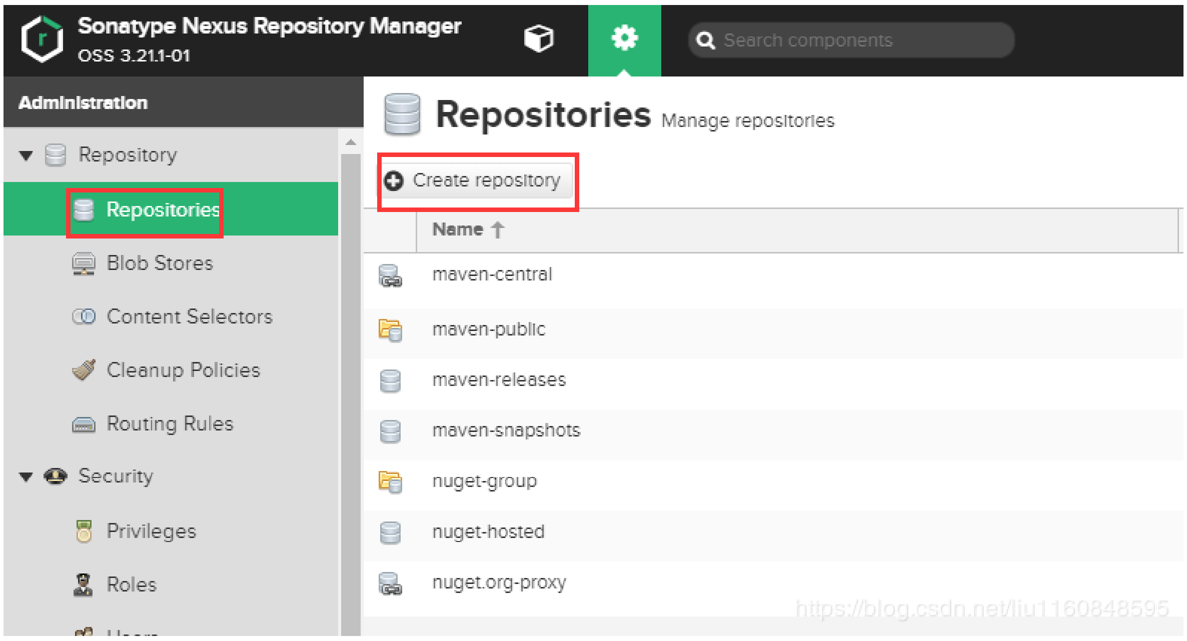
State:

☐ Enable Soft Quota Enable Soft Quota:限制目录大小

Path: /usr/software/nexus/sonatype-work/nexus3/blobs/lgs-private

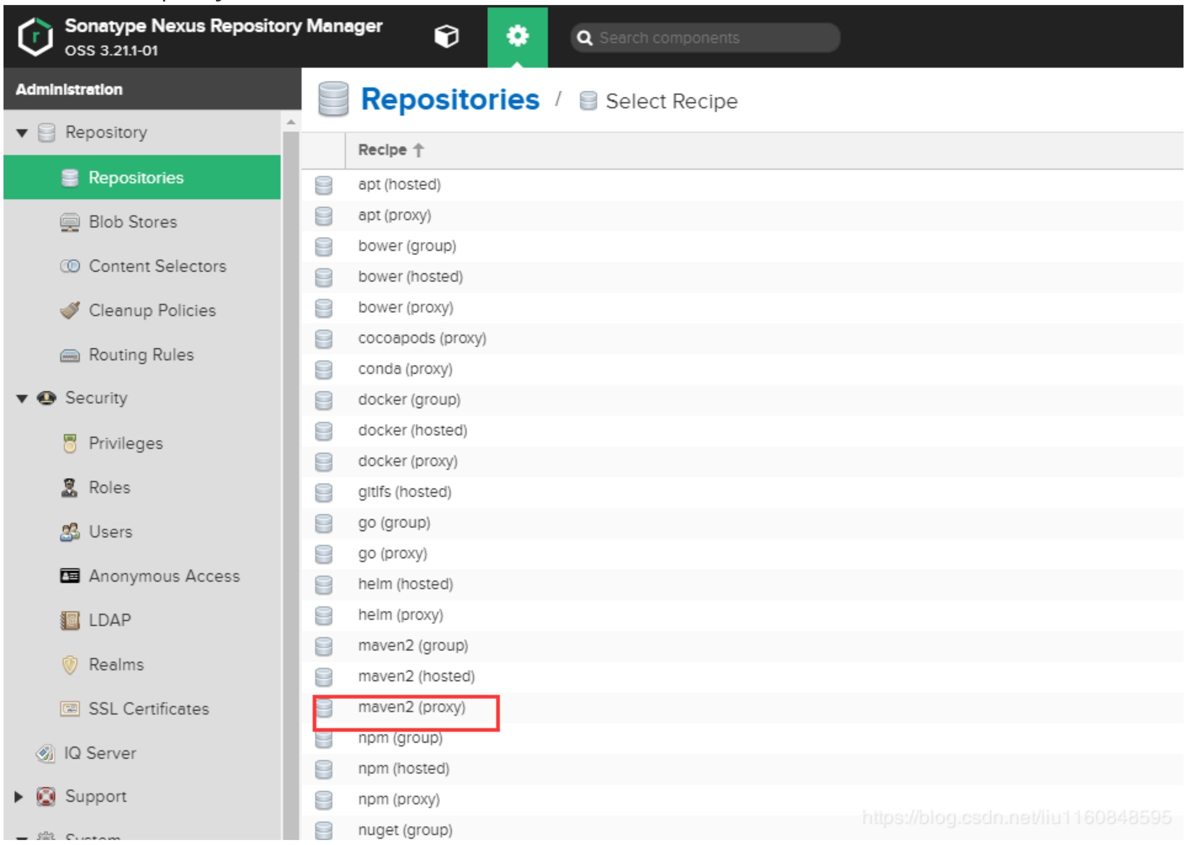
Create blob store | Cancel

2.3.3、创建Proxy Repository（代理仓库）



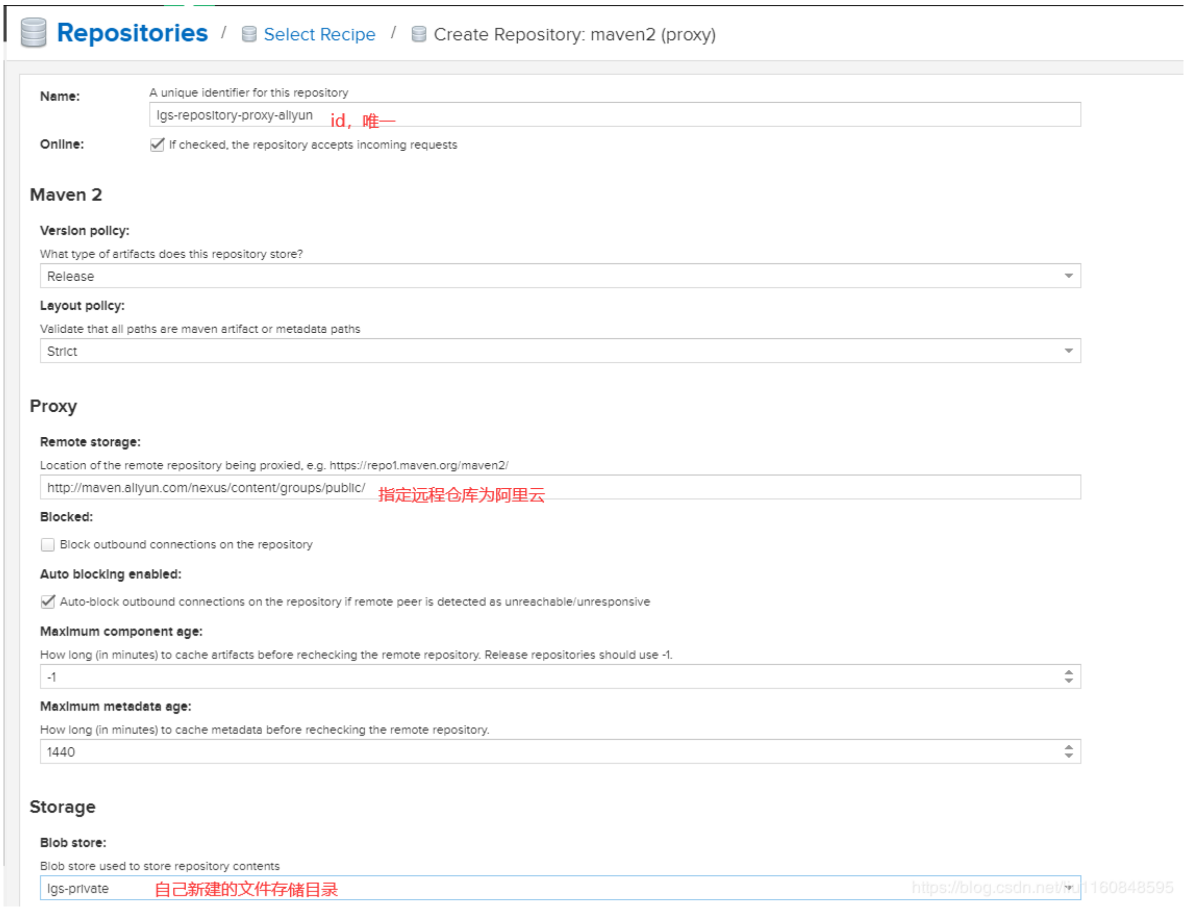
maven-central
maven-public
maven-releases
maven-snapshots
nuget-group
nuget-hosted
nuget.org-proxy

选择maven2(proxy)



apt (hosted)
apt (proxy)
bower (group)
bower (hosted)
bower (proxy)
cocoapods (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)

配置仓库属性


Repositories / Select Recipe / Create Repository: maven2 (proxy)

Name: A unique identifier for this repository
lgs-repository-proxy-aliyun　id，唯一

Online: ✓ If checked, the repository accepts incoming requests

Maven 2

Version policy:
What type of artifacts does this repository store?
Release

Layout policy:
Validate that all paths are maven artifact or metadata paths
Strict

Proxy

Remote storage:
Location of the remote repository being proxied, e.g. https://repo1.maven.org/maven2/
http://maven.aliyun.com/nexus/content/groups/public/　指定远程仓库为阿里云

Blocked:
☐ Block outbound connections on the repository

Auto blocking enabled:
✓ Auto-block outbound connections on the repository if remote peer is detected as unreachable/unresponsive

Maximum component age:
How long (in minutes) to cache artifacts before rechecking the remote repository. Release repositories should use -1.
-1

Maximum metadata age:
How long (in minutes) to cache metadata before rechecking the remote repository.
1440

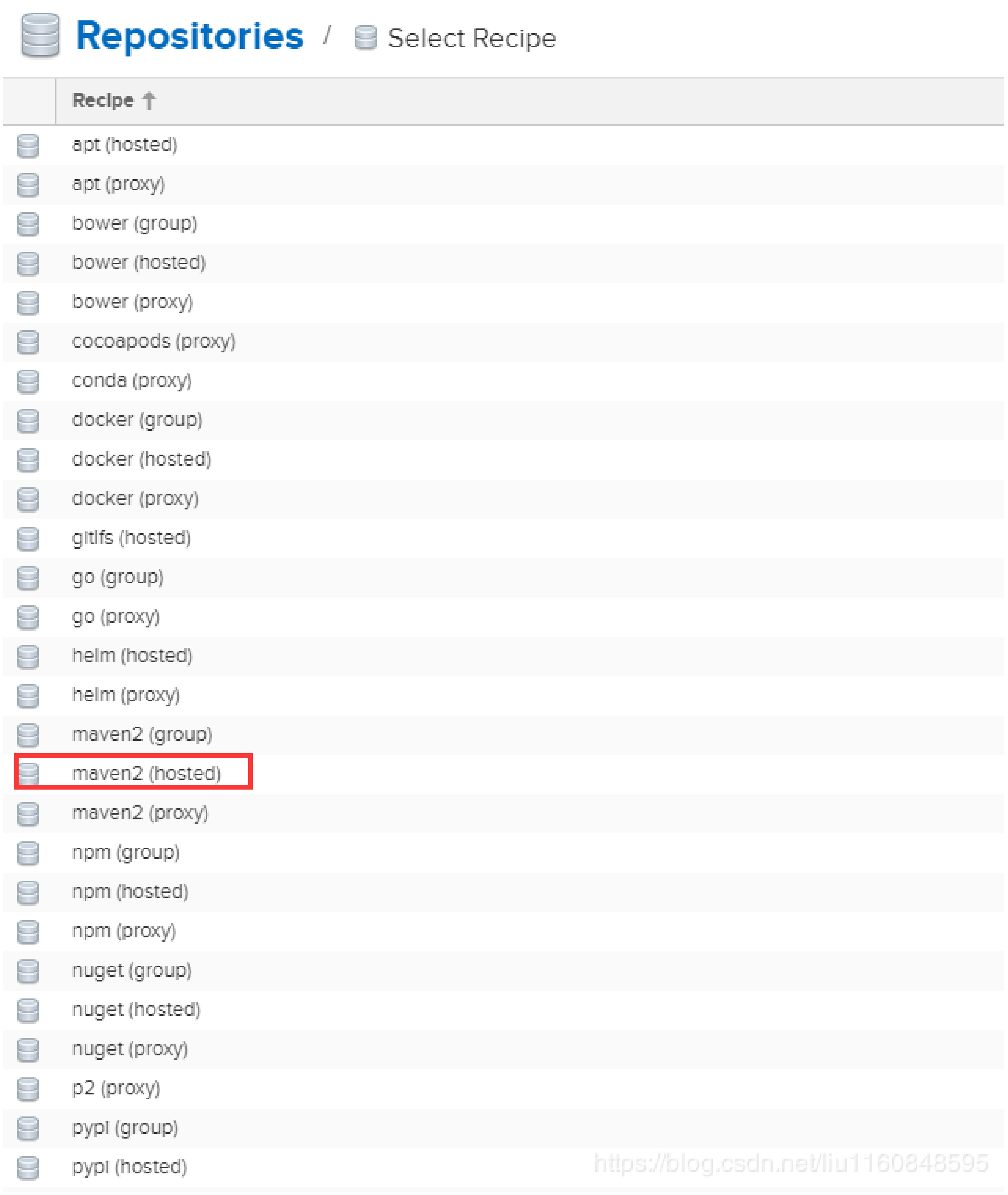
Storage

Blob store:
Blob store used to store repository contents
lgs-private　自己新建的文件存储目录

注意:
代理仓库可以建多个，用来指定多个远程仓库地址，比如除了maven仓库地址还可以指定阿里云的仓库地址，在国内的话，最好是阿里云的地址在前，其他的排后面（在组仓库里可以指定成员和其顺序）

2.3.4、创建hosted reposity(宿主仓库)

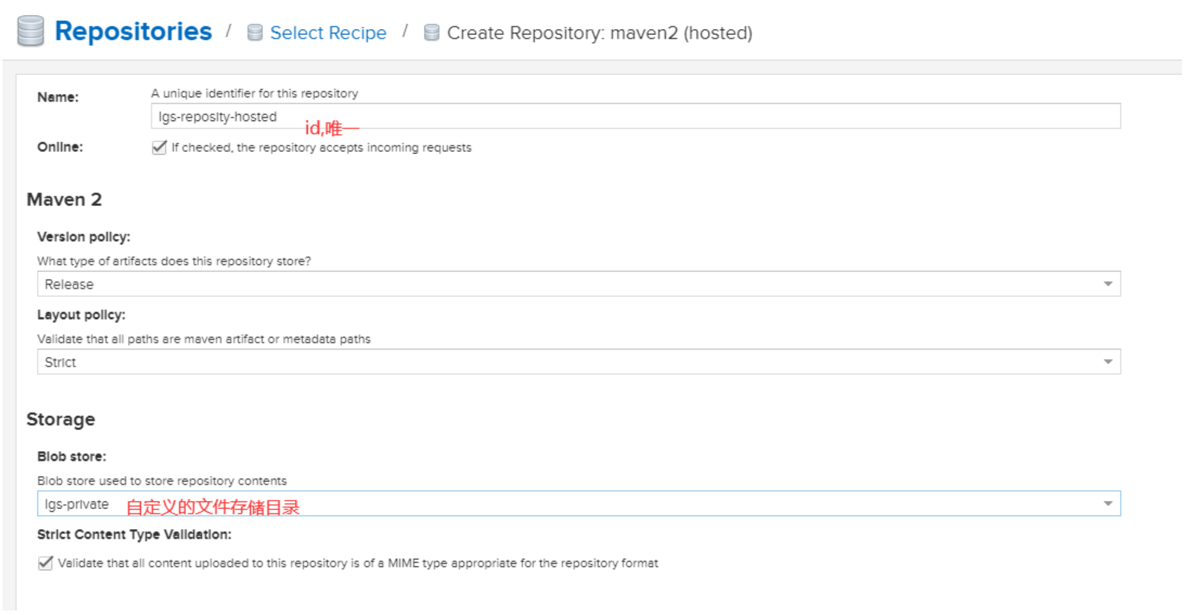
选择maven2(hosted)



Repositories / Select Recipe

Recipe
apt (hosted)
apt (proxy)
bower (group)
bower (hosted)
bower (proxy)
cocoapods (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)

指定文件存储目录



Repositories / Select Recipe / Create Repository: maven2 (hosted)

Name: A unique identifier for this repository
lgs-reposity-hosted id,唯一

Online: ✓ If checked, the repository accepts incoming requests

Maven 2

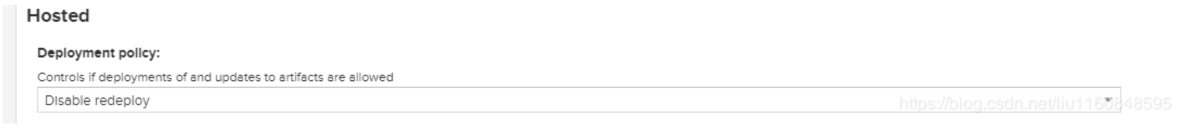
Version policy:
What type of artifacts does this repository store?
Release

Layout policy:
Validate that all paths are maven artifact or metadata paths
Strict

Storage

Blob store:
Blob store used to store repository contents
lgs-private 自定义的文件存储目录

Strict Content Type Validation:
✓ Validate that all content uploaded to this repository is of a MIME type appropriate for the repository format



Hosted

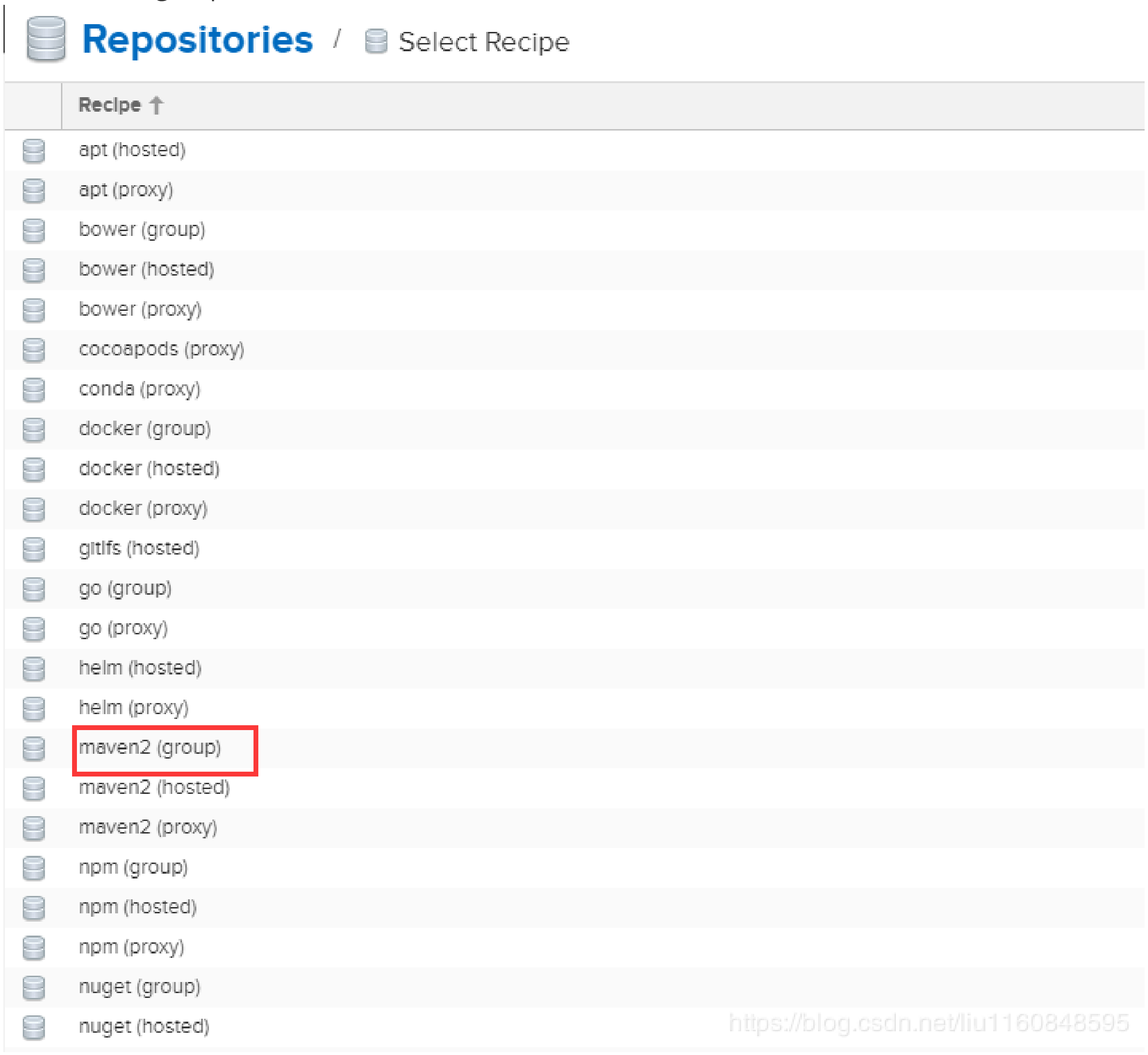
Deployment policy:

Controls if deployments of and updates to artifacts are allowed

Disable redeploy

2.3.5、创建group reposity(组仓库)

选择maven2(group)

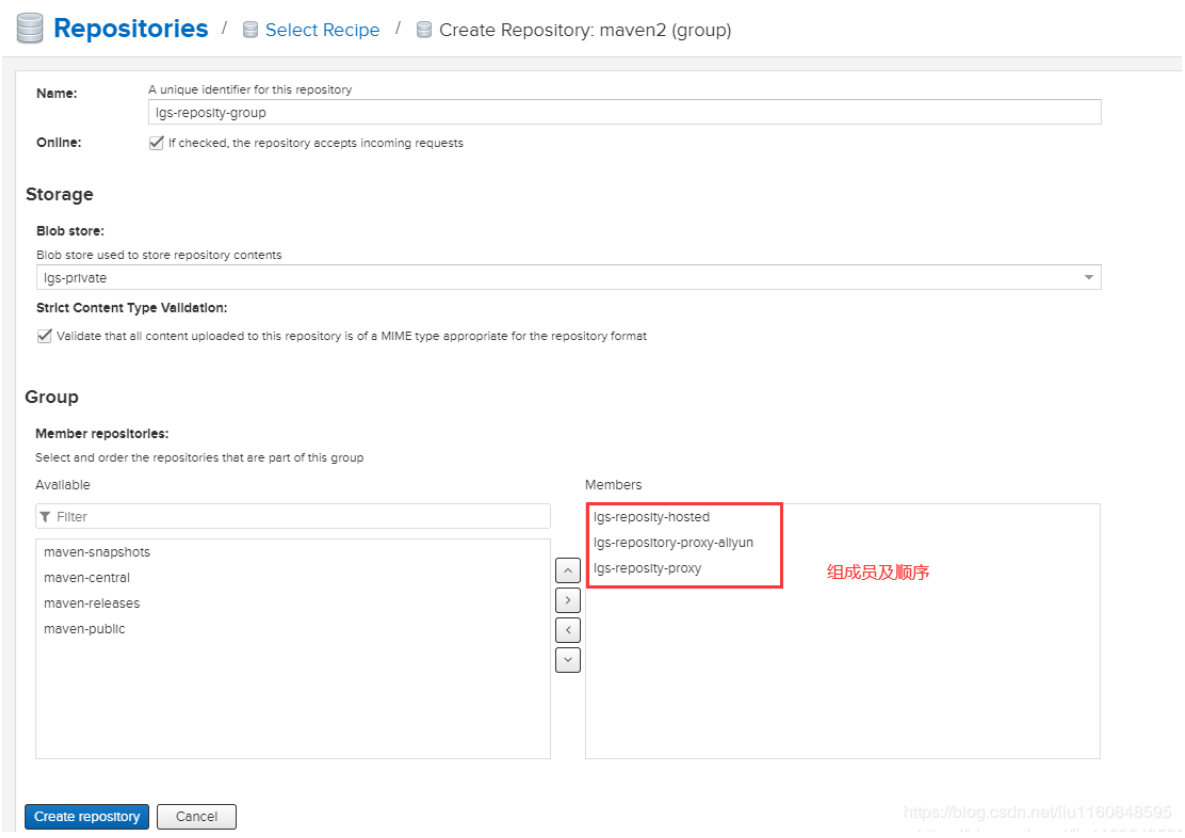


Repositories / Select Recipe

Recipe ↑
apt (hosted)
apt (proxy)
bower (group)
bower (hosted)
bower (proxy)
cocoapods (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)

指定组仓库成员

*



Repositories / Select Recipe / Create Repository: maven2 (group)

Name:	A unique identifier for this repository: lgs-reposity-group
Online:	✓ If checked, the repository accepts incoming requests

Storage

Blob store: Blob store used to store repository contents: lgs-private

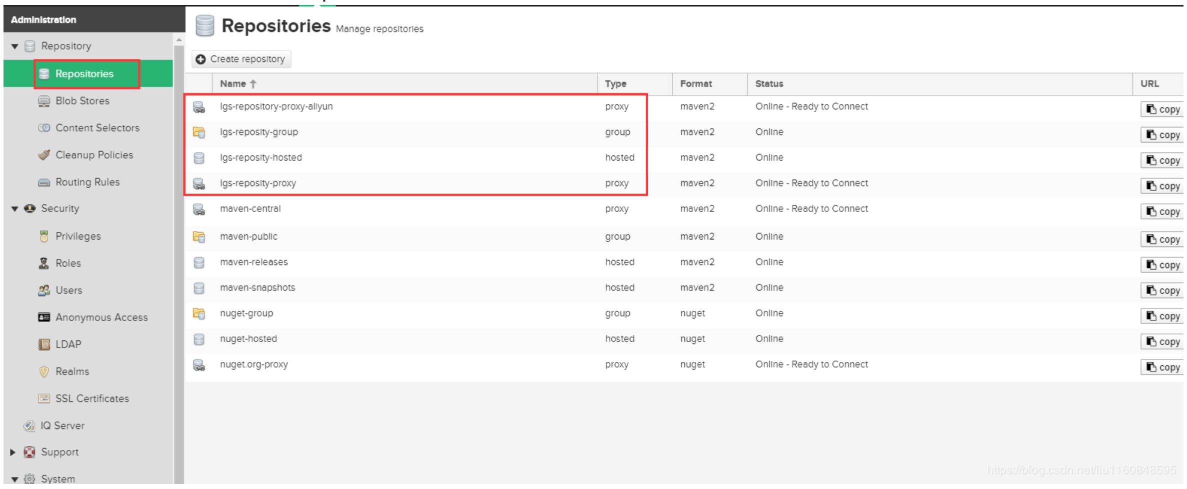
Strict Content Type Validation: ✓ Validate that all content uploaded to this repository is of a MIME type appropriate for the repository format

Group

Member repositories: Select and order the repositories that are part of this group

Available	Members
maven-snapshots	lgs-reposity-hosted
maven-central	lgs-repository-proxy-aliyun
maven-releases	lgs-reposity-proxy
maven-public	

仓库建完成之后，可以在Repositories列表中查看新建的仓库



Repositories Manage repositories

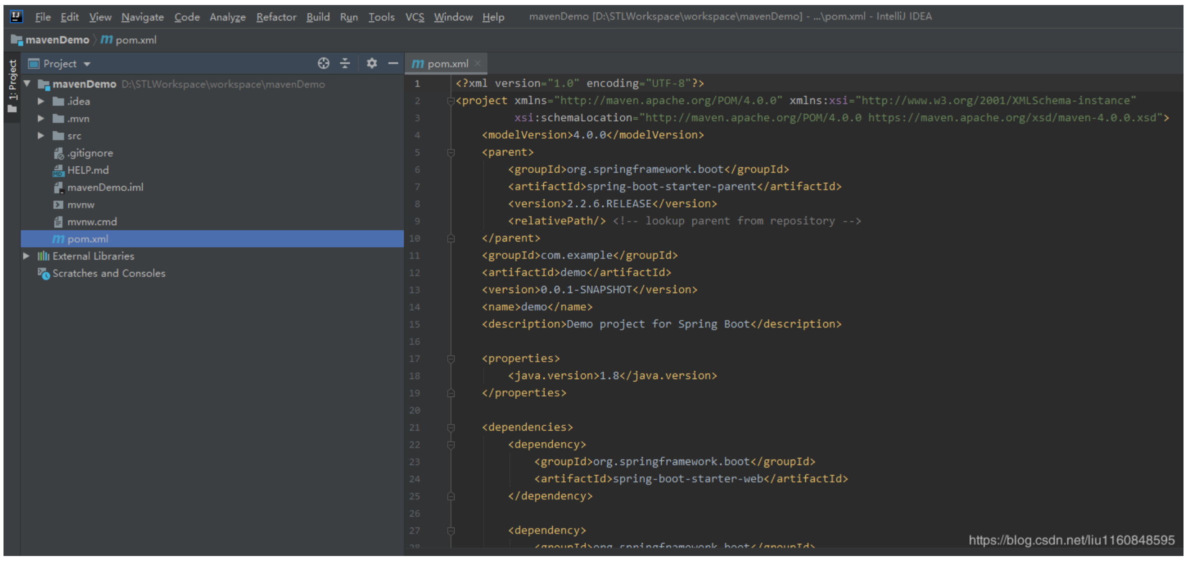
Name	Type	Format	Status	URL
lgs-repository-proxy-aliyun	proxy	maven2	Online - Ready to Connect	copy
lgs-reposity-group	group	maven2	Online	copy
lgs-reposity-hosted	hosted	maven2	Online	copy
lgs-reposity-proxy	proxy	maven2	Online - Ready to Connect	copy
maven-central	proxy	maven2	Online - Ready to Connect	copy
maven-public	group	maven2	Online	copy
maven-releases	hosted	maven2	Online	copy
maven-snapshots	hosted	maven2	Online	copy
nuget-group	group	nuget	Online	copy
nuget-hosted	hosted	nuget	Online	copy
nuget.org-proxy	proxy	nuget	Online - Ready to Connect	copy

3、使用maven私服

以上步骤把私服的环境已经搭建好，接下来就是使用搭建好的私服。

3.1、创建maven工程

首先创建一个使用maven的工程，为了方便，我就直接用idea创建一个springboot工程
具体过程不赘述，创建好的工程如下：


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
mavenDemo [D:\STLWorkspace\workspace\mavenDemo] - ...\pom.xml - IntelliJ IDEA
mavenDemo
pom.xml
Project
mavenDemo D:\STLWorkspace\workspace\mavenDemo
.idea
.mvn
src
.gitignore
HELP.md
mavenDemo.iml
mvnw
mvnw.cmd
pom.xml
External Libraries
Scratches and Consoles
https://blog.csdn.net/liu1160848595

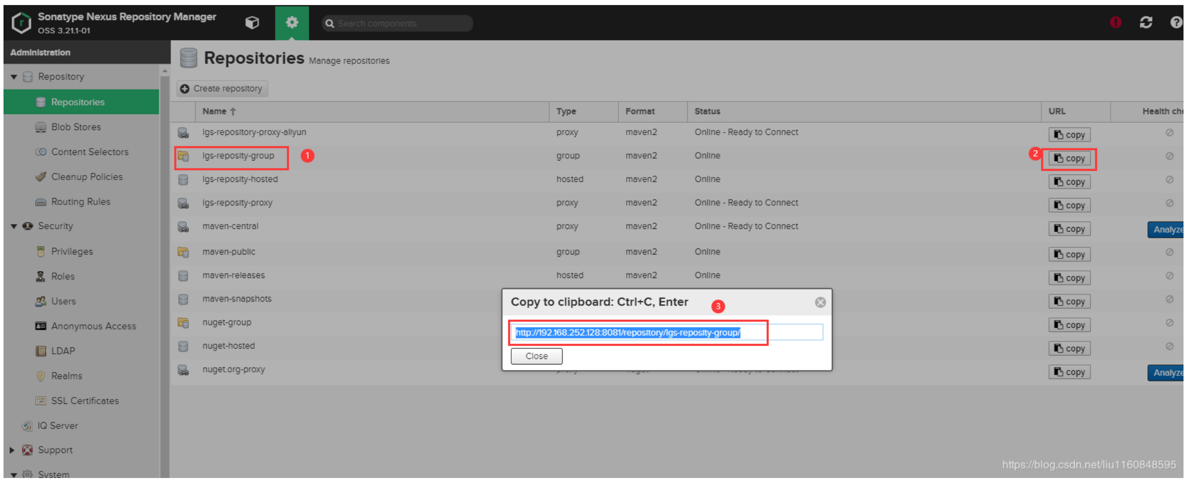
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.2.6.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>demo</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
```

3.2、修改setting.xml

在settings.xml文件中，修改如下配置：

```xml
<!--新建一个本地仓库地址，用来测试在本地没有jar包的情况下，私服上面jar包下载情况-->
<localRepository>C:\Users\Admin\.m2\repository-demo</localRepository>
<!--nexus服务器-->
<servers>
   <server>
       <id>nexus</id>
       <username>lgs</username>
       <password>lgs用户的密码</password>
   </server>
</servers>
<!--私服仓库镜像-->
<mirrors>
    <mirror>
        <id>nexus</id>
        <name>nexus repository</name>
        <url>http://192.168.252.128:8081/repository/lgs-reposity-group/</url>
        <mirrorOf>central</mirrorOf>
    </mirror>
</mirrors>
```

上面的私服仓库镜像的url在下面所示的位置，直接粘贴即可：


Sonatype Nexus Repository Manager
OSS 3.21.1-01

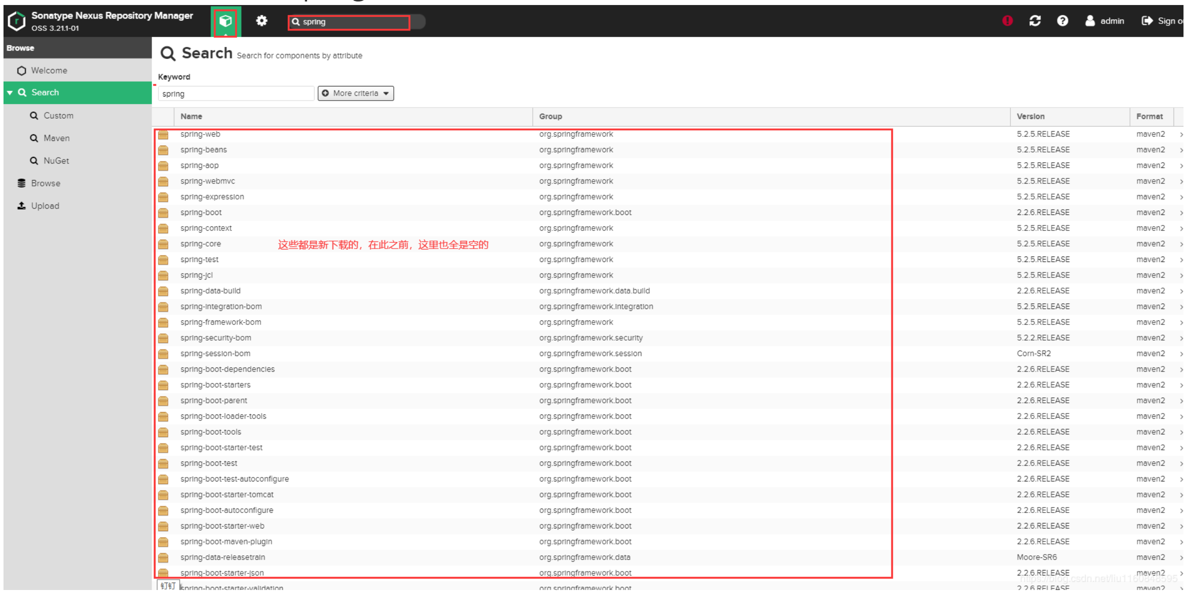
Repositories Manage repositories

Name	Type	Format	Status	URL
lgs-repository-proxy-aliyun	proxy	maven2	Online - Ready to Connect	copy
lgs-reposity-group	group	maven2	Online	copy
lgs-reposity-hosted	hosted	maven2	Online	copy
lgs-reposity-proxy	proxy	maven2	Online - Ready to Connect	copy
maven-central	proxy	maven2	Online - Ready to Connect	copy
maven-public	group	maven2	Online	copy
maven-releases	hosted	maven2	Online	copy
maven-snapshots				copy
nuget-group				copy
nuget-hosted				copy
nuget.org-proxy				copy

Copy to clipboard: Ctrl+C, Enter

http://192.168.252.128:8081/repository/lgs-reposity-group/

修改settings.xml之后，使此文件生效，我的是idea,在切换回idea之后，配置文件已经生效，因为这份文件指定的本地仓库是新建的一个文件夹，里面是空的，所以idea在疯狂地下载springboot的一些依赖包，再进入控制台，搜索spring，可以发现私服上面也已经下载了很多依赖包了：


Search Search for components by attribute

Keyword: spring

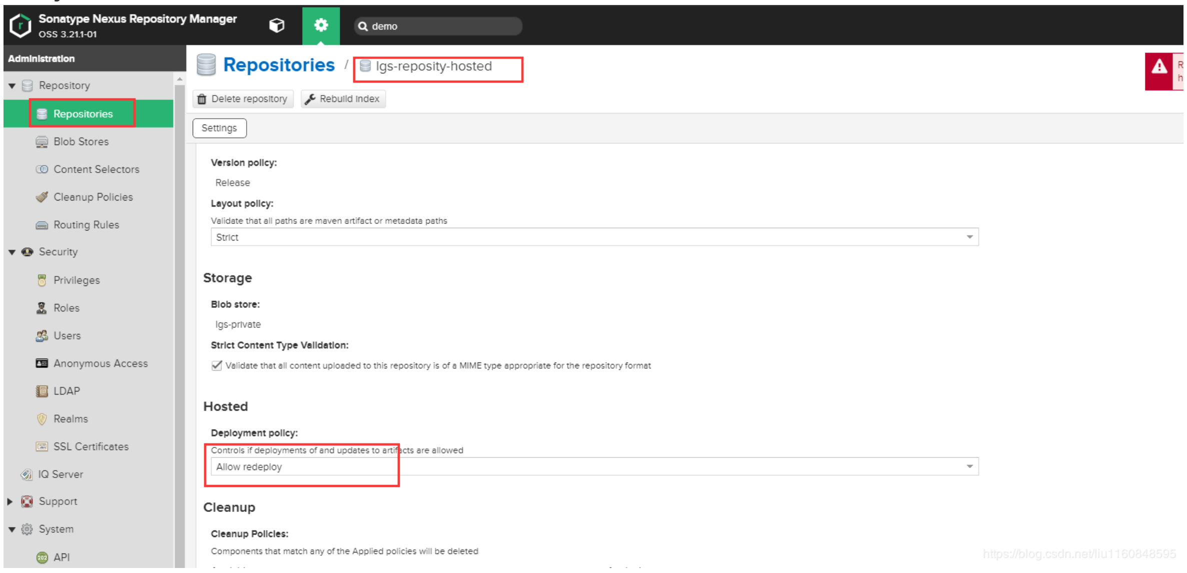
Name	Group	Version	Format
spring-web	org.springframework	5.2.5.RELEASE	maven2
spring-beans	org.springframework	5.2.5.RELEASE	maven2
spring-aop	org.springframework	5.2.5.RELEASE	maven2
spring-webmvc	org.springframework	5.2.5.RELEASE	maven2
spring-expression	org.springframework	5.2.5.RELEASE	maven2
spring-boot	org.springframework.boot	2.2.6.RELEASE	maven2
spring-context	org.springframework	5.2.5.RELEASE	maven2
spring-core	org.springframework	5.2.5.RELEASE	maven2
spring-test	org.springframework	5.2.5.RELEASE	maven2
spring-jcl	org.springframework	5.2.5.RELEASE	maven2
spring-data-build	org.springframework.data.build	2.2.6.RELEASE	maven2
spring-integration-bom	org.springframework.integration	5.2.5.RELEASE	maven2
spring-framework-bom	org.springframework	5.2.5.RELEASE	maven2
spring-security-bom	org.springframework.security	5.2.2.RELEASE	maven2
spring-session-bom	org.springframework.session	Corn-SR2	maven2
spring-boot-dependencies	org.springframework.boot	2.2.6.RELEASE	maven2
spring-boot-starters	org.springframework.boot	2.2.6.RELEASE	maven2
spring-boot-parent	org.springframework.boot	2.2.6.RELEASE	maven2
spring-boot-loader-tools	org.springframework.boot	2.2.6.RELEASE	maven2
spring-boot-tools	org.springframework.boot	2.2.6.RELEASE	maven2
spring-boot-starter-test	org.springframework.boot	2.2.6.RELEASE	maven2
spring-boot-test	org.springframework.boot	2.2.6.RELEASE	maven2
spring-boot-test-autoconfigure	org.springframework.boot	2.2.6.RELEASE	maven2
spring-boot-starter-tomcat	org.springframework.boot	2.2.6.RELEASE	maven2
spring-boot-autoconfigure	org.springframework.boot	2.2.6.RELEASE	maven2
spring-boot-starter-web	org.springframework.boot	2.2.6.RELEASE	maven2
spring-boot-maven-plugin	org.springframework.boot	2.2.6.RELEASE	maven2
spring-data-releasetrain	org.springframework.data	Moore-SR6	maven2
spring-boot-starter-json	org.springframework.boot	2.2.6.RELEASE	maven2
spring-boot-starter-validation	org.springframework.boot	2.2.6.RELEASE	maven2

这些都是新下载的，在此之前，这里也全是空的

此时，jar包的下载功能是没问题的，但是既然是私服，那肯定是需要上传自定义jar包的，不然就意义不大了。

4、上传jar包

上传jar包之前需要将宿主仓库的上传权限打开，默认是关闭的：


Repositories / lgs-reposity-hosted

Version policy:
Release

Layout policy:
Validate that all paths are maven artifact or metadata paths
Strict

Storage

Blob store:
lgs-private

Strict Content Type Validation:
✓ Validate that all content uploaded to this repository is of a MIME type appropriate for the repository format

Hosted

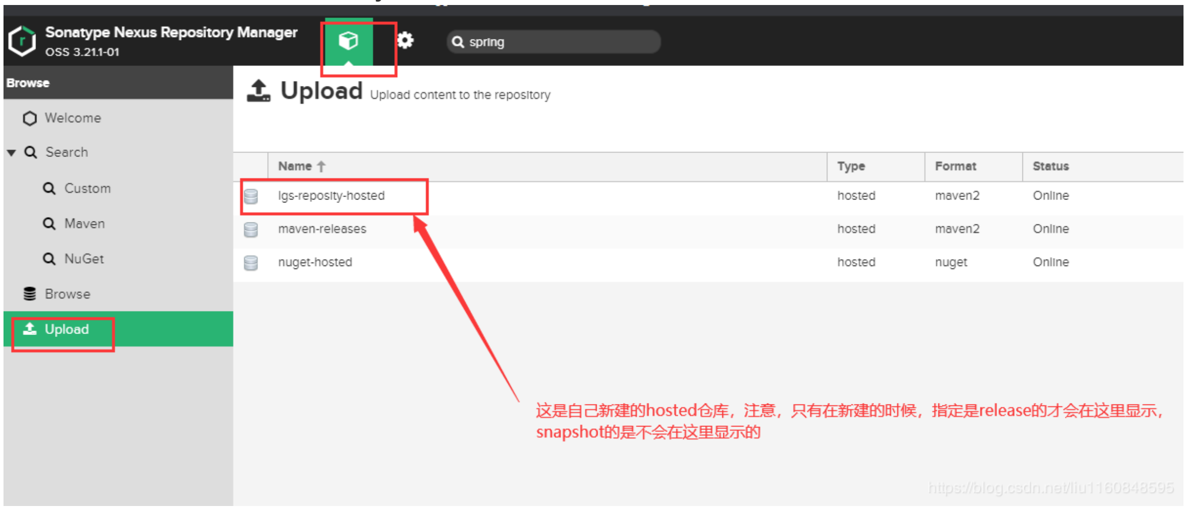
Deployment policy:
Controls if deployments of and updates to artifacts are allowed
Allow redeploy

Cleanup

Cleanup Policies:
Components that match any of the Applied policies will be deleted

4.1、控制台上传

我用的是nexus3，在控制台上有jar包的上传位置(据说nexus2也有)：

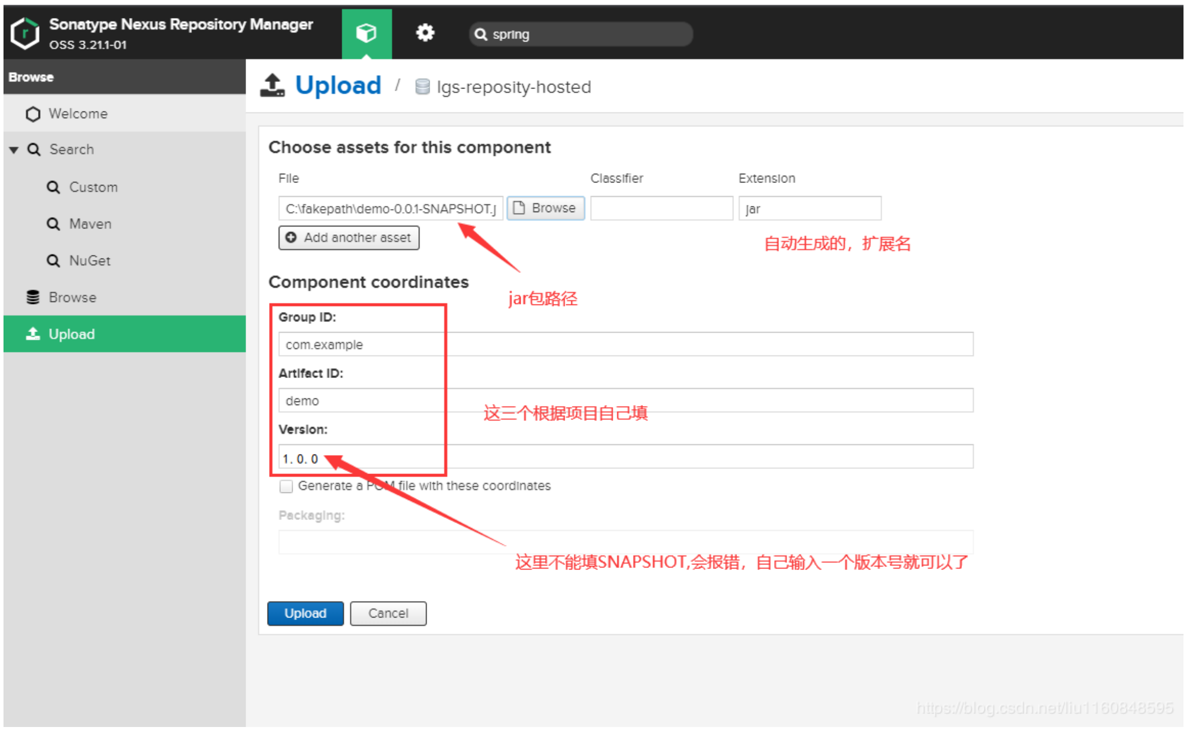


Upload Upload content to the repository

Name	Type	Format	Status
lgs-reposity-hosted	hosted	maven2	Online
maven-releases	hosted	maven2	Online
nuget-hosted	hosted	nuget	Online

这是自己新建的hosted仓库，注意，只有在新建的时候，指定是release的才会在这里显示，snapshot的是不会在这里显示的

上传界面：



Upload / lgs-reposity-hosted

Choose assets for this component

File: C:\fakepath\demo-0.0.1-SNAPSHOT.j Browse
Classifier:
Extension: jar

自动生成的，扩展名

jar包路径

Component coordinates

Group ID: com.example

Artifact ID: demo

Version: 1. 0. 0

这三个根据项目自己填

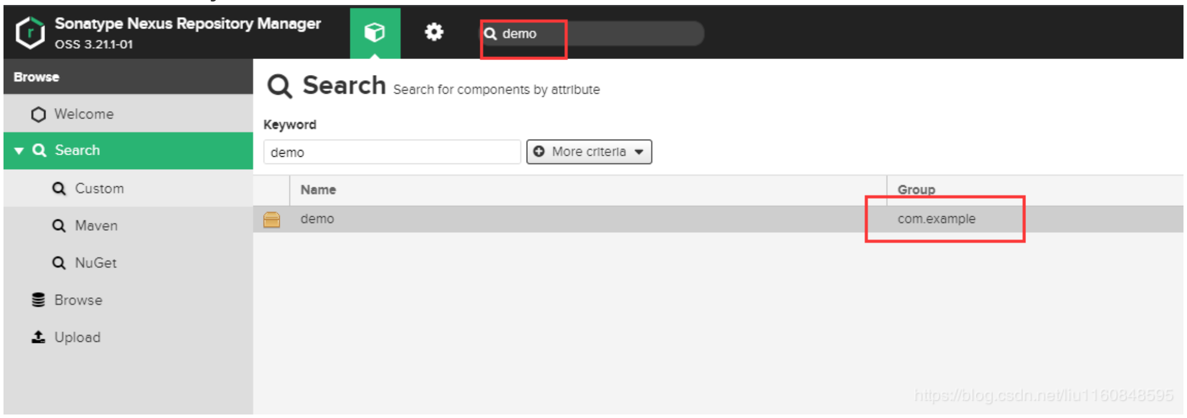
Generate a POM file with these coordinates

Packaging:

这里不能填SNAPSHOT,会报错，自己输入一个版本号就可以了

测试上传所用的jar包，我是直接把我新建的springboot工程打包丢上去的。

再搜索一下上传的jar包的名字，可以看到已经上传成功了：

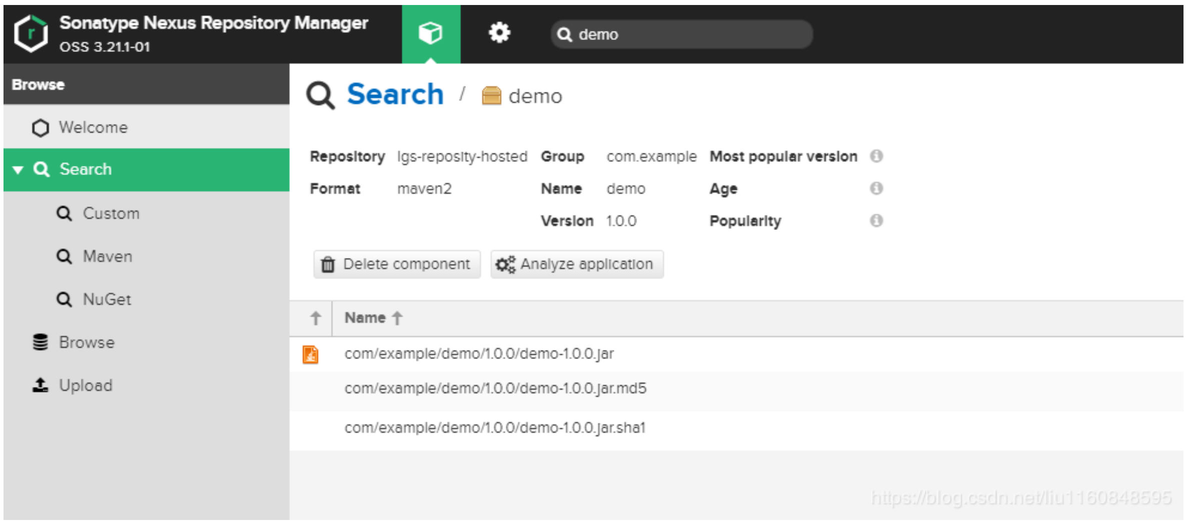


Search Search for components by attribute

Keyword: demo

Name	Group
demo	com.example

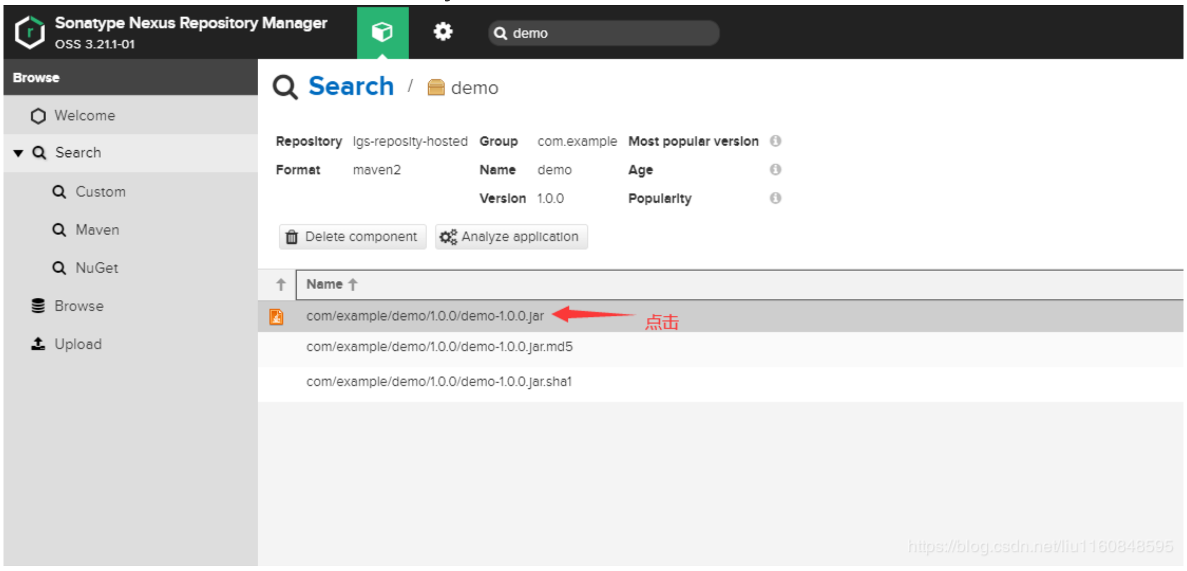
点进去还可以看到详情



Repository lgs-reposity-hosted Group com.example Most popular version
Format maven2 Name demo Age
Version 1.0.0 Popularity

Name
com/example/demo/1.0.0/demo-1.0.0.jar
com/example/demo/1.0.0/demo-1.0.0.jar.md5
com/example/demo/1.0.0/demo-1.0.0.jar.sha1

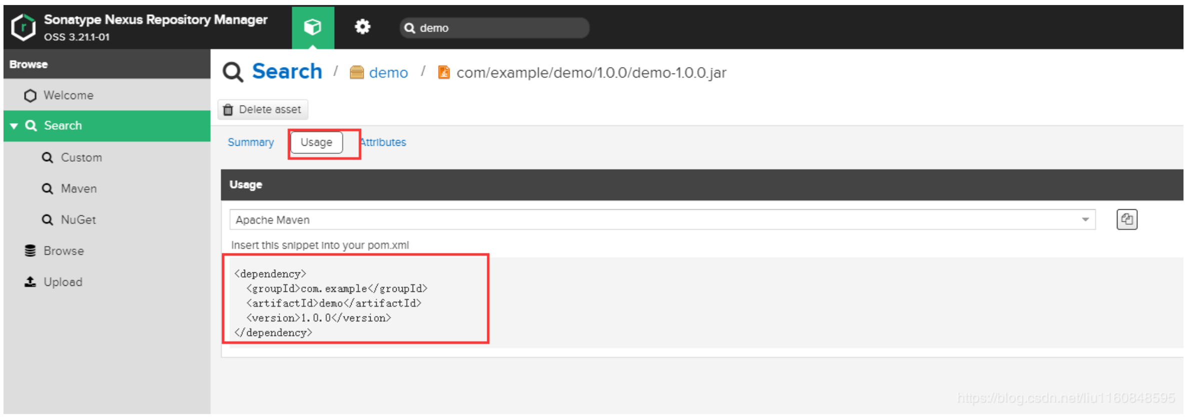
那么现在我们就在项目中引用自定义的jar包试试：



Repository lgs-reposity-hosted Group com.example Most popular version
Format maven2 Name demo Age
Version 1.0.0 Popularity

Name
com/example/demo/1.0.0/demo-1.0.0.jar
com/example/demo/1.0.0/demo-1.0.0.jar.md5
com/example/demo/1.0.0/demo-1.0.0.jar.sha1

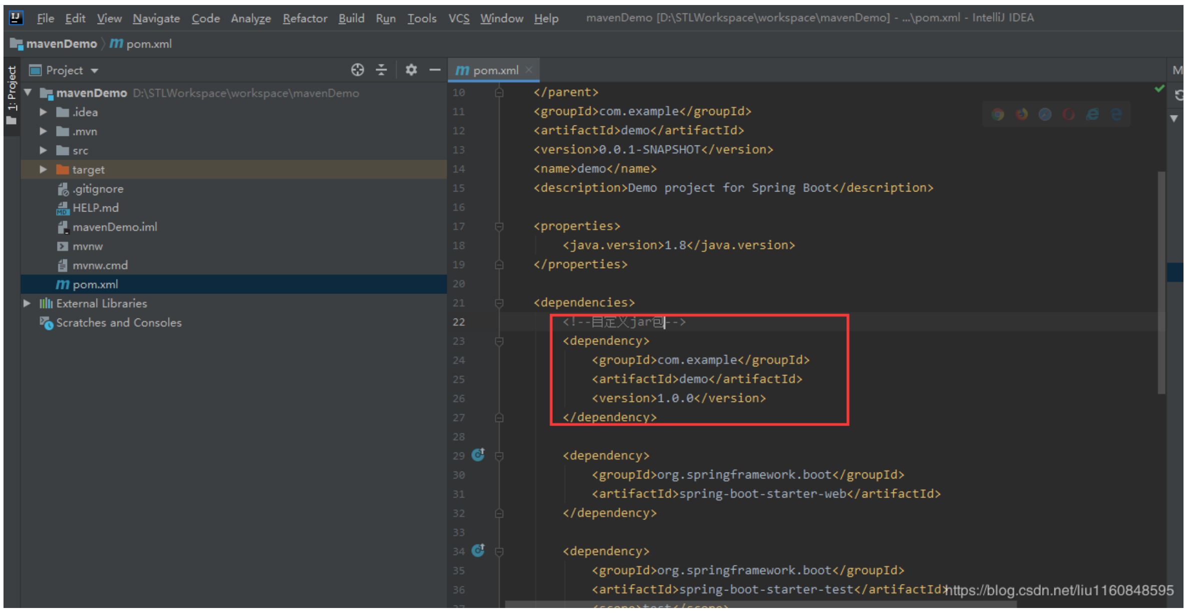
将下面的依赖引入到项目中：



Insert this snippet into your pom.xml

```
<dependency>
  <groupId>com.example</groupId>
  <artifactId>demo</artifactId>
  <version>1.0.0</version>
</dependency>
```

我就把这个jar包引用到demo项目中的，好像没规定不能在项目中引用自己打的jar包：

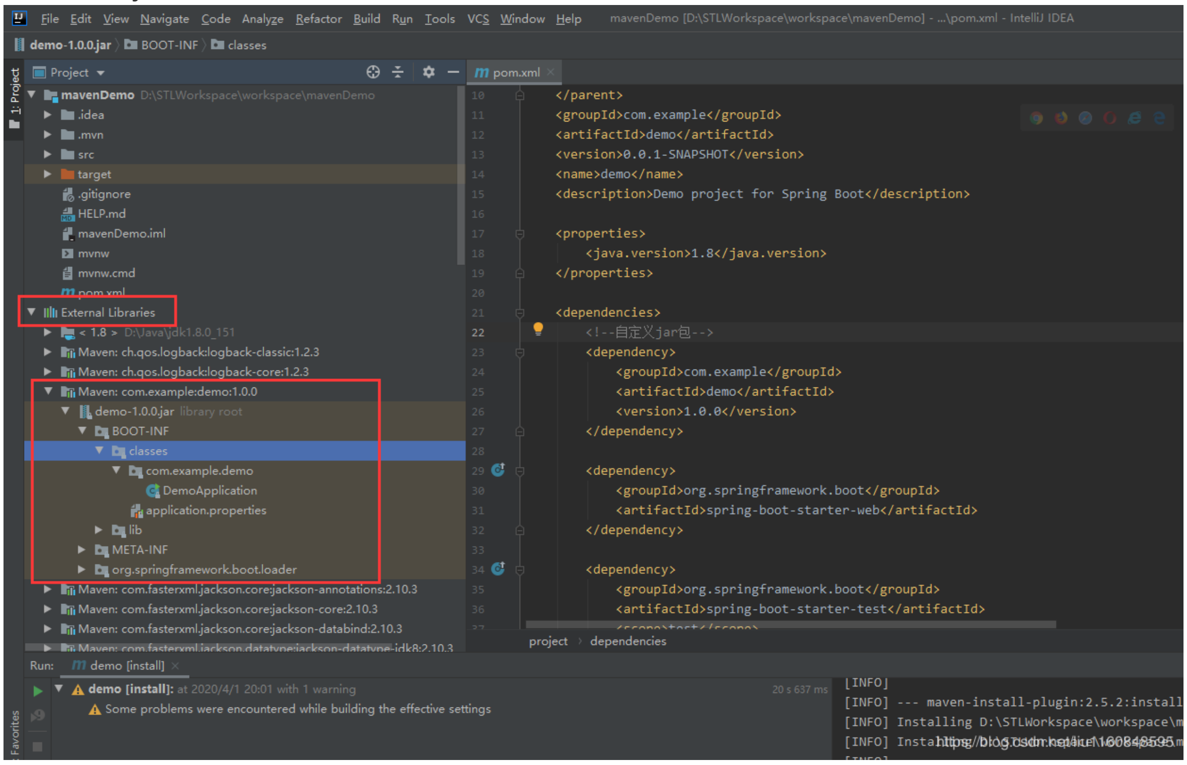


```
mavenDemo  D:\STLWorkspace\workspace\mavenDemo
  .idea
  .mvn
  src
  target
  .gitignore
  HELP.md
  mavenDemo.iml
  mvnw
  mvnw.cmd
  pom.xml
External Libraries
Scratches and Consoles
```

```
    </parent>
    <groupId>com.example</groupId>
    <artifactId>demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>demo</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <!--自定义jar包-->
        <dependency>
            <groupId>com.example</groupId>
            <artifactId>demo</artifactId>
            <version>1.0.0</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
```

https://blog.csdn.net/liu1160848595

可以看到，jar包已经下载下来了，而且已经加入到依赖中了：



```
mavenDemo  D:\STLWorkspace\workspace\mavenDemo
  .idea
  .mvn
  src
  target
  .gitignore
  HELP.md
  mavenDemo.iml
  mvnw
  mvnw.cmd
  pom.xml
External Libraries
  < 1.8 >  D:\Java\jdk1.8.0_151
  Maven: ch.qos.logback:logback-classic:1.2.3
  Maven: ch.qos.logback:logback-core:1.2.3
  Maven: com.example:demo:1.0.0
    demo-1.0.0.jar  library root
      BOOT-INF
        classes
          com.example.demo
            DemoApplication
          application.properties
        lib
      META-INF
      org.springframework.boot.loader
  Maven: com.fasterxml.jackson.core:jackson-annotations:2.10.3
  Maven: com.fasterxml.jackson.core:jackson-core:2.10.3
  Maven: com.fasterxml.jackson.core:jackson-databind:2.10.3
  Maven: com.fasterxml.jackson.datatype:jackson-datatype-jdk8:2.10.3
```

```
    </parent>
    <groupId>com.example</groupId>
    <artifactId>demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>demo</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <!--自定义jar包-->
        <dependency>
            <groupId>com.example</groupId>
            <artifactId>demo</artifactId>
            <version>1.0.0</version>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
```

```
demo [install]: at 2020/4/1 20:01 with 1 warning        20 s 637 ms
    Some problems were encountered while building the effective settings

[INFO]
[INFO] --- maven-install-plugin:2.5.2:install
[INFO] Installing D:\STLWorkspace\workspace\m
[INFO] Insta
```

https://blog.csdn.net/liu1160848595

4.2、命令上传

这是通过控制台直接上传jar包，也可以通过命令上传，前提是要上传的机器已经安装好了maven。

1. 添加配置
   在maven安装目录/conf/settings.xml下添加账号密码配置：
   id是为了标识这个server，可自定义（记住，下面命令要用到）。

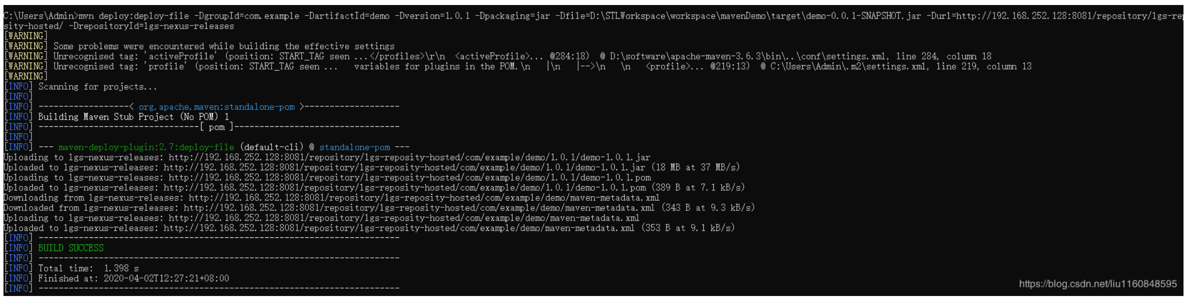
```
<server>
      <id>lgs-nexus-releases</id>
      <username>admin</username>
      <password>密码</password>
</server>
```

2. 执行命令

```
mvn deploy:deploy-file -DgroupId=com.example -DartifactId=demo -Dversion=1.0.1 -
Dpackaging=jar -Dfile=D:\STLWorkspace\workspace\mavenDemo\target\demo-0.0.1-
SNAPSHOT.jar -Durl=http://192.168.252.128:8081/repository/lgs-reposity-hosted/ -
DrepositoryId=lgs-nexus-releases
```

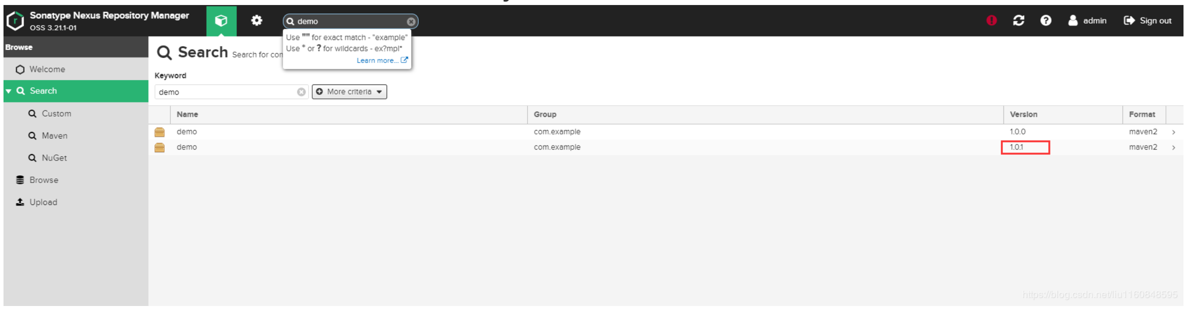
参数解释：
-DgroupId，-DartifactId，-Dversion三个参数对应jar包的三个参数，也就是在pom文件中依赖的时候需要的三个参数，可以自定义写
-Dpackaging：打包类型
-Dfile：　jar包位置
-Durl：　私服仓库地址
-DrepositoryId：　要上传的服务id，这个id就是上面配置的server的id
上面这些参数都是和控制台里的那些选择对应好的，很容易理解。

看到以下提示就成功了：



```
C:\Users\Admin>mvn deploy:deploy-file -DgroupId=com.example -DartifactId=demo -Dversion=1.0.1 -Dpackaging=jar -Dfile=D:\STLWorkspace\workspace\mavenDemo\target\demo-0.0.1-SNAPSHOT.jar -Durl=http://192.168.252.128:8081/repository/lgs-re
sity-hosted/ -DrepositoryId=lgs-nexus-releases
[WARNING]
[WARNING] Some problems were encountered while building the effective settings
[WARNING] Unrecognised tag: 'activeProfile' (position: START_TAG seen ...</profiles>\r\n  <activeProfile>... @284:18)  @ D:\software\apache-maven-3.6.3\bin\..\conf\settings.xml, line 284, column 18
[WARNING] Unrecognised tag: 'profile' (position: START_TAG seen ...   variables for plugins in the POM.\n   |\n   |-->\n   \n   <profile>... @219:13)  @ C:\Users\Admin\.m2\settings.xml, line 219, column 13
[WARNING]
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] --- maven-deploy-plugin:2.7:deploy-file (default-cli) @ standalone-pom ---
Uploading to lgs-nexus-releases: http://192.168.252.128:8081/repository/lgs-reposity-hosted/com/example/demo/1.0.1/demo-1.0.1.jar
Uploaded to lgs-nexus-releases: http://192.168.252.128:8081/repository/lgs-reposity-hosted/com/example/demo/1.0.1/demo-1.0.1.jar (18 MB at 37 MB/s)
Uploading to lgs-nexus-releases: http://192.168.252.128:8081/repository/lgs-reposity-hosted/com/example/demo/1.0.1/demo-1.0.1.pom
Uploaded to lgs-nexus-releases: http://192.168.252.128:8081/repository/lgs-reposity-hosted/com/example/demo/1.0.1/demo-1.0.1.pom (389 B at 7.1 kB/s)
Downloading from lgs-nexus-releases: http://192.168.252.128:8081/repository/lgs-reposity-hosted/com/example/demo/maven-metadata.xml
Downloaded from lgs-nexus-releases: http://192.168.252.128:8081/repository/lgs-reposity-hosted/com/example/demo/maven-metadata.xml (343 B at 9.3 kB/s)
Uploading to lgs-nexus-releases: http://192.168.252.128:8081/repository/lgs-reposity-hosted/com/example/demo/maven-metadata.xml
Uploaded to lgs-nexus-releases: http://192.168.252.128:8081/repository/lgs-reposity-hosted/com/example/demo/maven-metadata.xml (353 B at 9.1 kB/s)
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  1.398 s
[INFO] Finished at: 2020-04-02T12:27:21+08:00
[INFO] ------------------------------------------------------------------------
```

搜索一下，可以看到刚刚上传的1.0.1版本的jar包显示在了列表中：



Name	Group	Version	Format
demo	com.example	1.0.0	maven2
demo	com.example	1.0.1	maven2

至此，maven私服的搭建以及使用都已完成！