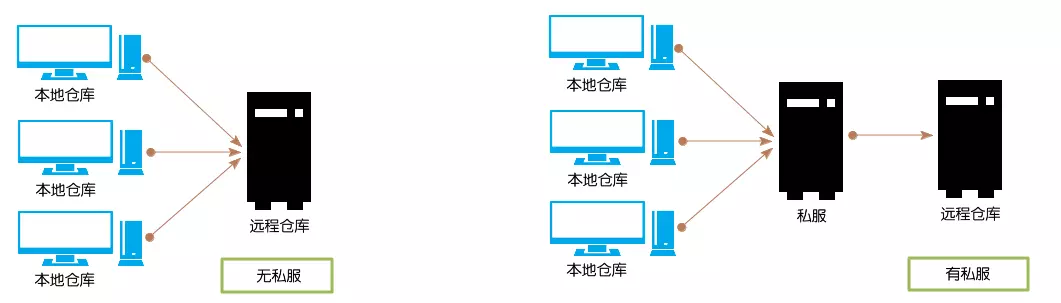
# 简介

私服是架设在局域网的一种特殊的远程仓库，目的是代理远程仓库及部署第三方构件。有了私服之后，当 Maven 需要下载构件时，直接请求私服，私服上存在则下载到本地仓库；否则，私服请求外部的远程仓库，将构件下载到私服，再提供给本地仓库下载。



我们可以使用专门的 Maven 仓库管理软件来搭建私服，比如：[Apache Archiva](https://link.jianshu.com/?t=http://archiva.apache.org/index.cgi" \t "https://www.jianshu.com/p/_blank)，[Artifactory](https://link.jianshu.com/?t=http://www.jfrog.com/home/v_artifactory_opensource_overview/" \t "https://www.jianshu.com/p/_blank)，[Sonatype Nexus](https://link.jianshu.com/?t=http://www.sonatype.org/nexus/" \t "https://www.jianshu.com/p/_blank)。这里我们使用 Sonatype Nexus。

Nexus是常用的私用Maven服务器，一般是公司内部使用。

## 私服的优点

1. **节省自己的外网带宽**

 建立私服可以减少组织自己的开支，大量的对于外部远程仓库的重复请求会消耗很大的带宽，利用私服代理外部仓库后，对外的重复构件下载得以消除，即降低外网带宽的压力。

1. **加速Maven的构建**

不停的请求外部仓库无疑是比较耗时的， 但Maven的一些内部机制（如快照检测）要求Maven在执行构建的时候不停地检查远程仓库的数据。

因此当配置了很多远程仓库时，构建的速度会被大大降低。使用私服可以很好地解决这个问题。

1. **部署第三方构件**

组织内部的生成的私有的构件肯定无法从外部仓库获取，Oracle的JDBC驱动由于版权原因不能发布到外网的中心仓库。

建立私服之后便可以将这些构件部署到本地私服中，供内部的Maven项目使用。

1. **提高稳定行，增强控制**

Maven构建搞定依赖于远程仓库，因此，当Internet不稳定的时候，Maven构建也会变的不稳定，甚至无法构建。  
使用私服后即使暂时没有Internet连接Maven也可以正常运行，因为私服中缓存了大量的构件。

此外一些私服软件（如：Nexus）还提供了很多额外的功能，如权限管理，RELEASE/SNAPSHOT区分等，管理员可以对仓库进行一些更高级的控制。

1. **降低中央仓库的负荷**

数百万的请求，存储数T的数据，需要相相当大的财力。使用私服可以避免很多对中央仓库的重复请求。

## 准备工作

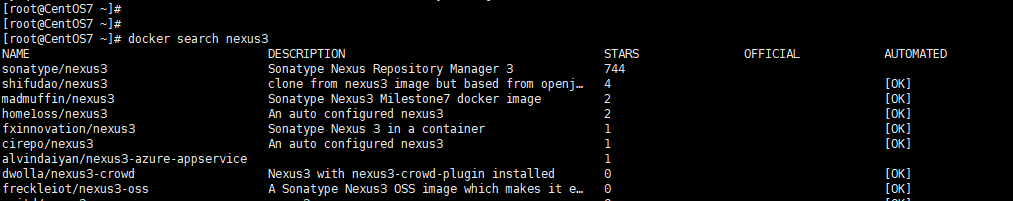
安装Nexus 之前先确定是否已经安装JDK,这里安装的是jdk8版本

本教程安装 Nexus 是在 Docker上进行的，需要对Docker 有一定的了解。

# 安装Nexus

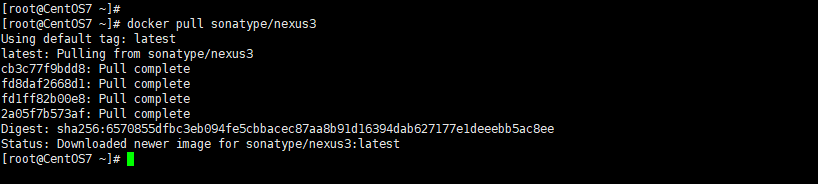
## 查找nexus3镜像

执行命令：docker search nexus3



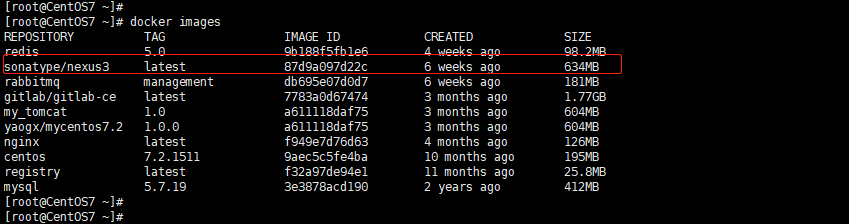
## 拉取镜像

执行命令：docker pull sonatype/nexus3



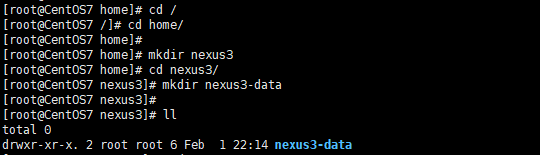
## 查看镜像

执行命令：docker images



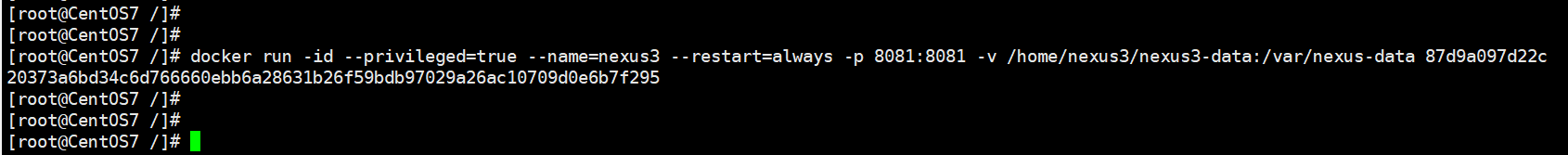
## 运行nexus3容器

首先创建挂在目录，执行命令：mkdir nexus



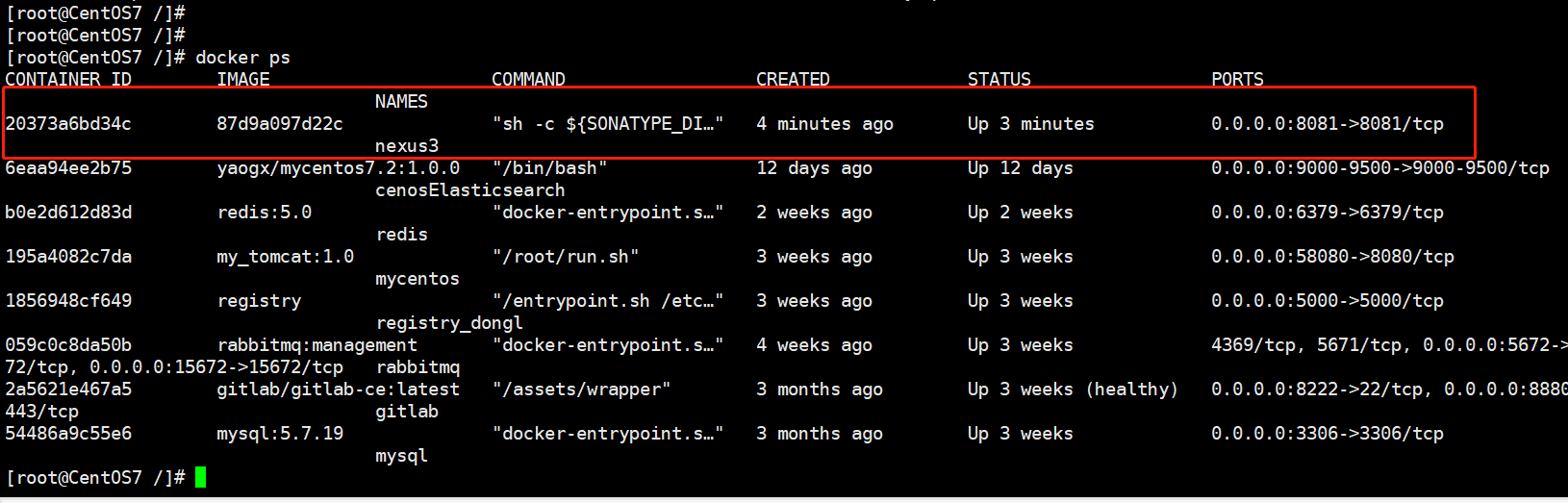
执行命令：docker run -id --privileged=true --name=nexus3 --restart=always -p 8081:8081 -v /home/nexus3/nexus3-data:/var/nexus-data 6e9721ad473a（这个是容器id或名称）

解释：  
-id 创建守护式容器  
--privileged=true 授予root权限（挂载多级目录必须为true，否则容器访问宿主机权限不足）  
--name=名字 给你的容器起个名字  
-p 宿主机端口：容器端口映射  
-v 宿主机目录：容器目录挂载



## 查看已启动容器

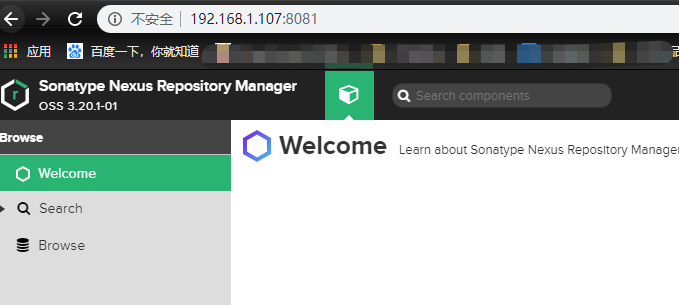
执行命令：docker ps



## 访问nexus3

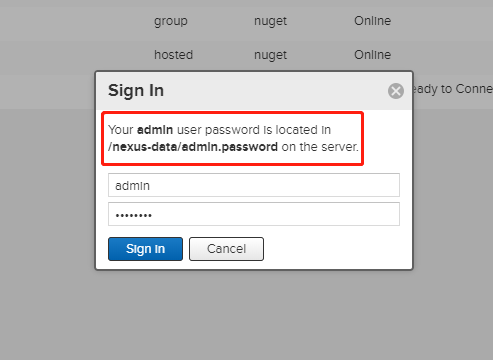
在浏览器输入：ip :端口号 ；

<http://192.168.1.107:8081/>



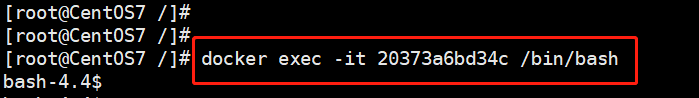
## 获取密码

点击右上角的 sign in ，进行登录会发现，界面提示我们初始密码位于/nexus-data/admin.password文件中，如下图所示：



此时我们就需要进入容器内部获取。

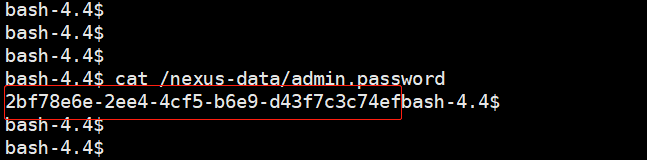
执行命令：docker exec -it 20373a6bd34c /bin/bash



进入容器后，执行命令：cat /nexus-data/admin.password

查看密码：

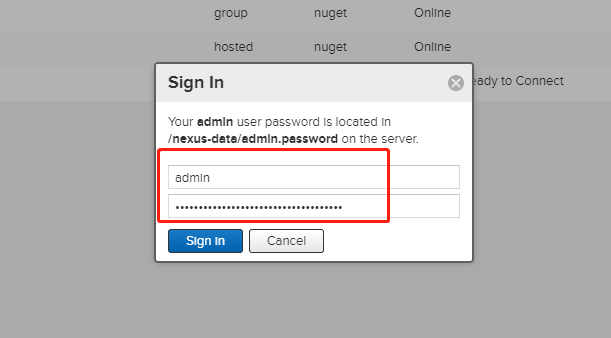
2bf78e6e-2ee4-4cf5-b6e9-d43f7c3c74ef



## 重新登录nexus

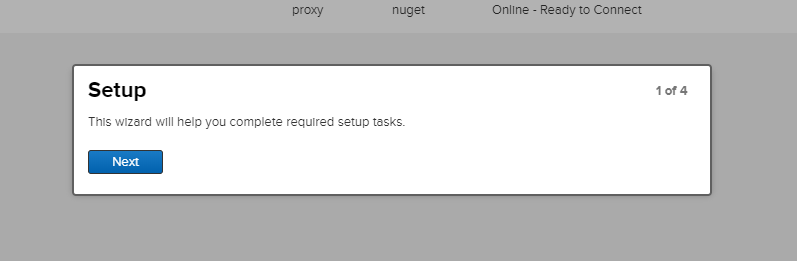
默认用户名admin，粘贴文件里密码即可；

**注：**首次登录必须要改默认密码。

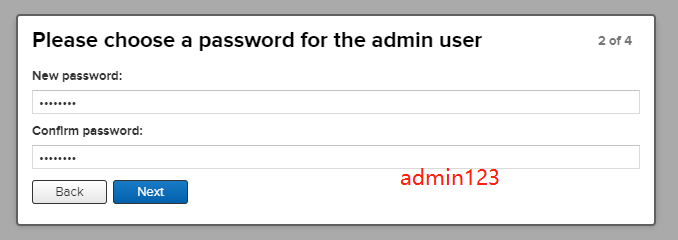


登陆成功。进入向导，点击next。

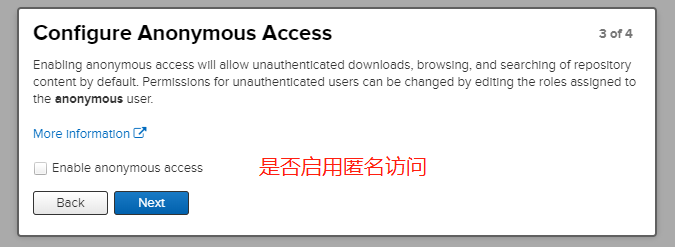
这个向导将帮助你完成所有任务。



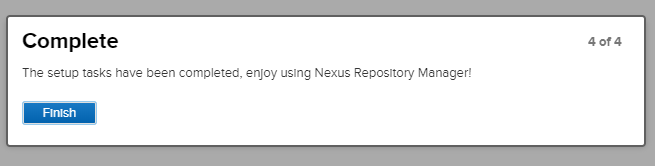
设置新密码：admin123



配置匿名访问，直接点击Next。



完成，点击Finish。



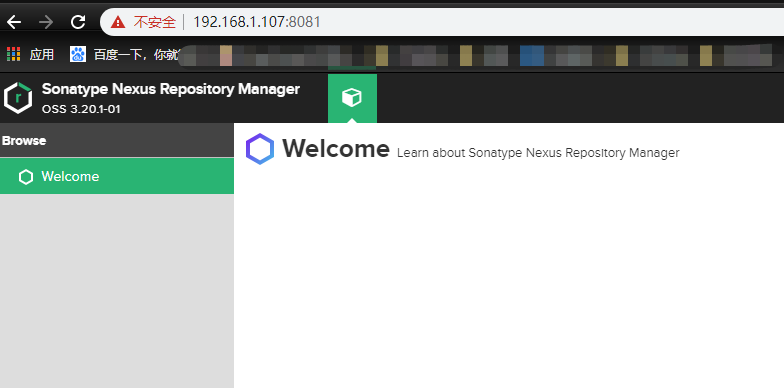
用户名：admin

密码：admin123

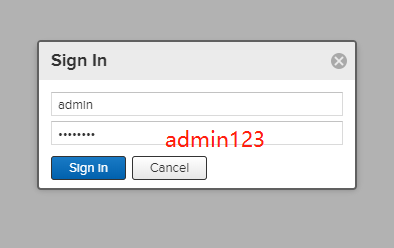
# nexus的使用

## 登录nexus

登录nexus ，在浏览器中输入：<http://192.168.1.107:8081/>

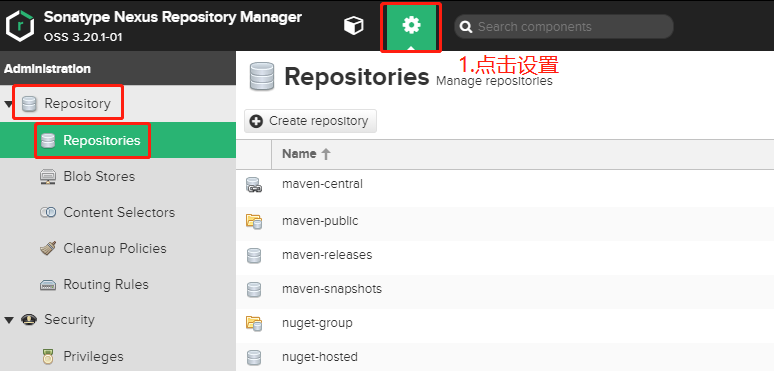


登录，输入账户密码

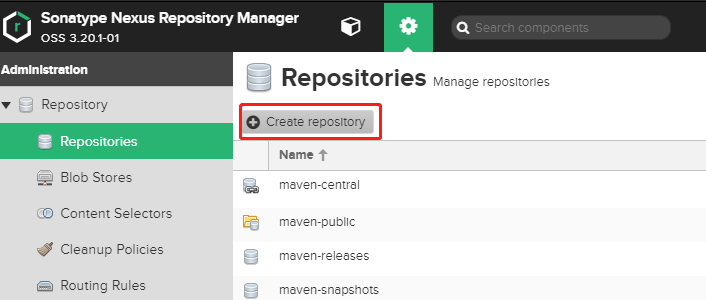


## 创建仓库

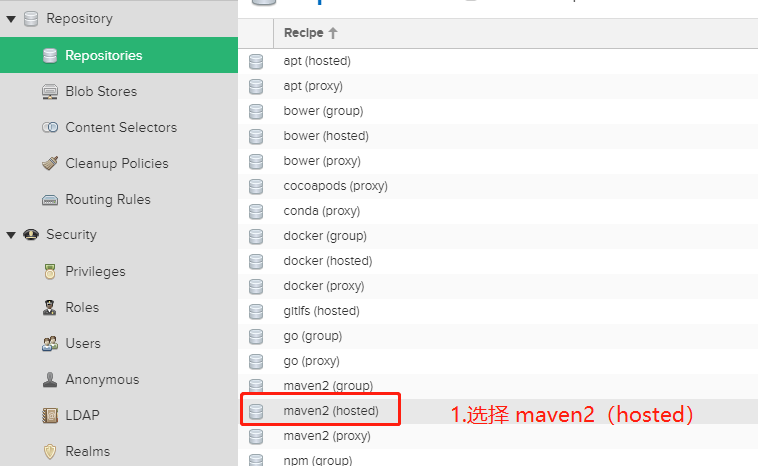
1. 点击设置中的 Repository 模块 下的 Repositories 。如图：



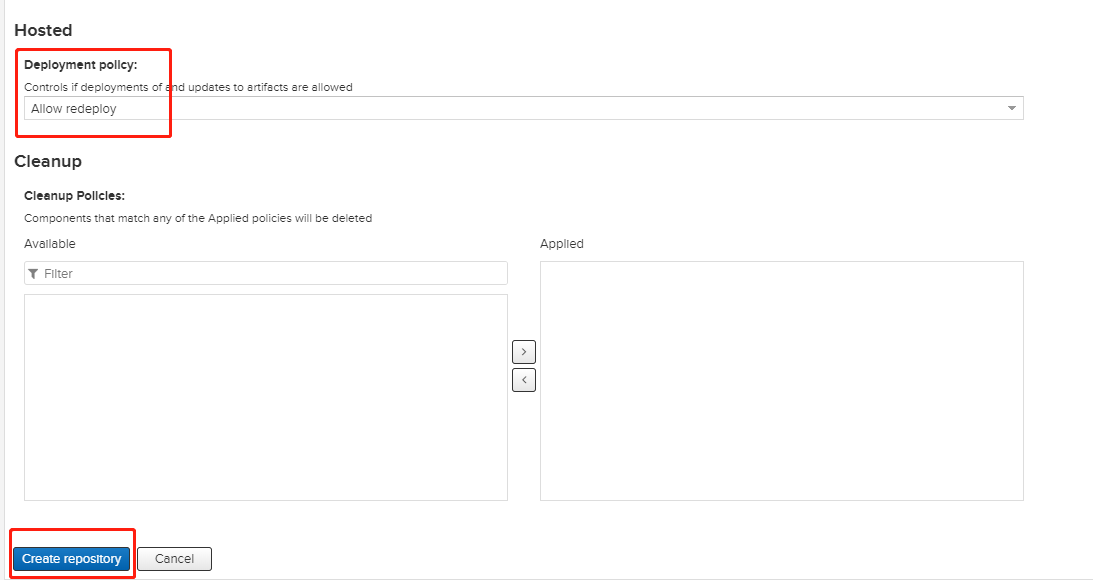
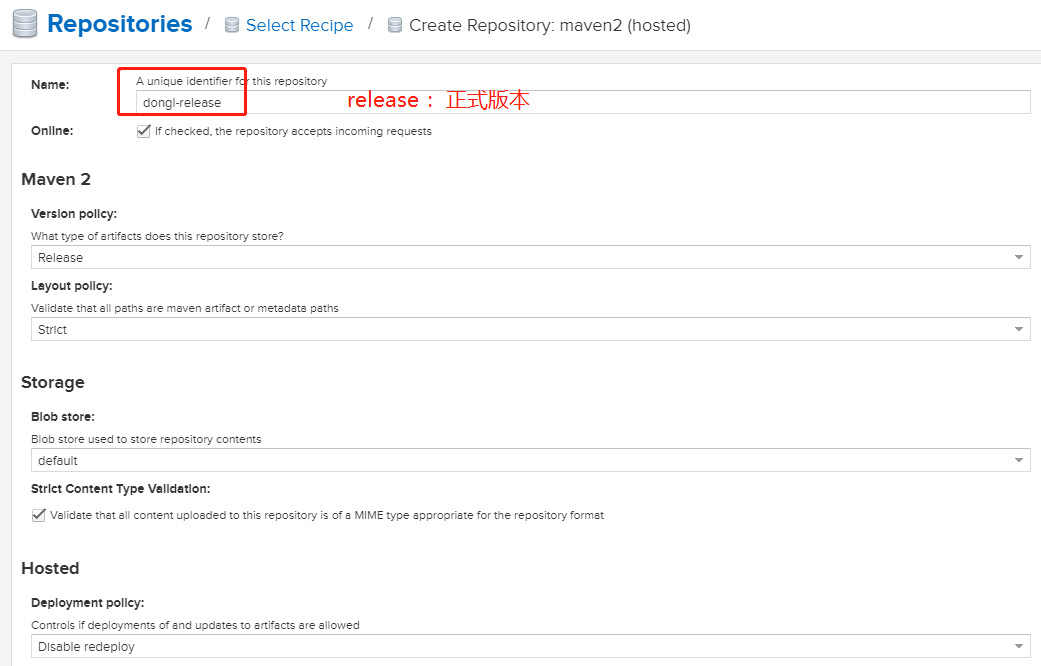
1. 点击 create repository 创建仓库。



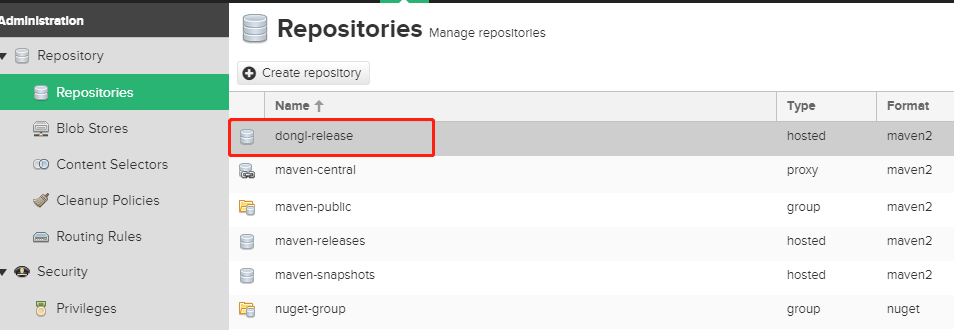
1. 选择 maven2 （hosted) 仓库。



1. 填写仓库名称 Name ，本教程取名：dongl-release ，点击Create repository 。

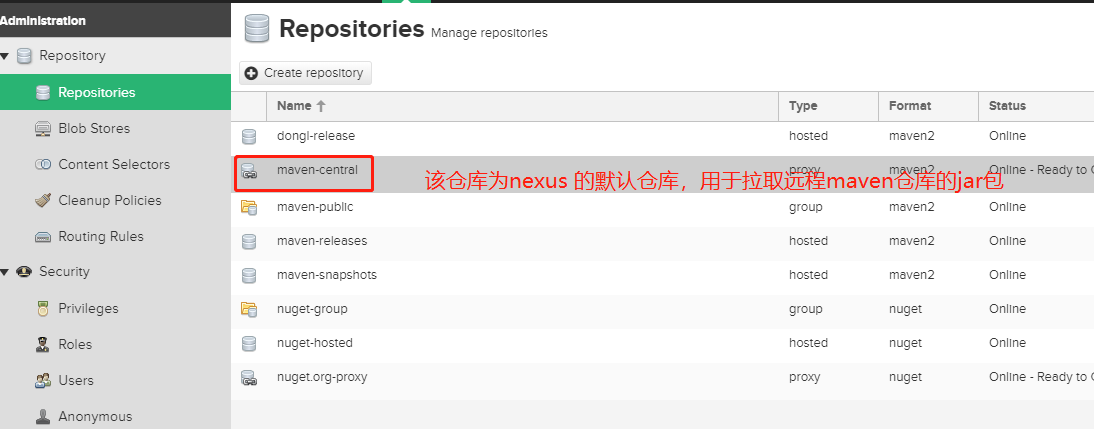


1. 至此仓库创建成功，在列表中已经显示。



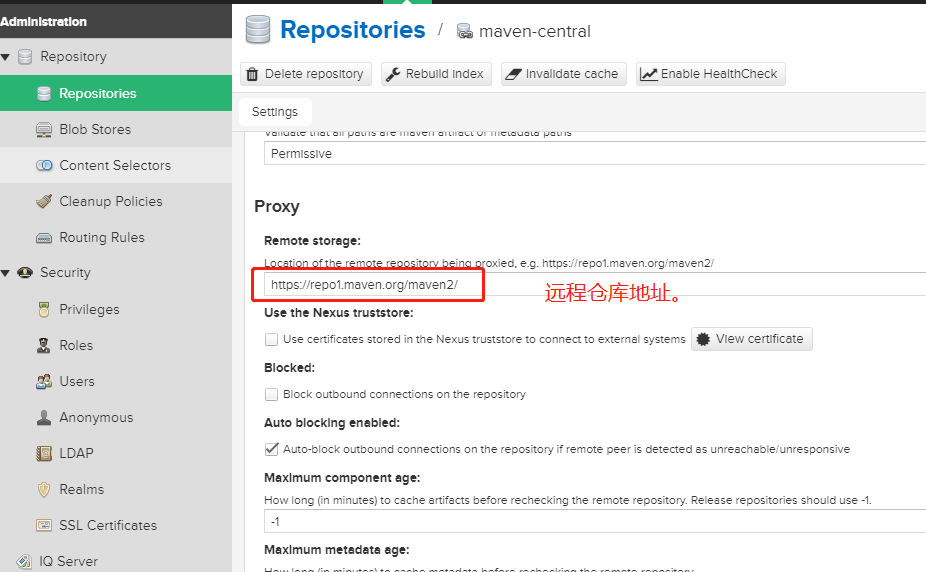
## 修改 私服nexus 的远程仓库

在列表中发现 maven-central 为nexus 的默认仓库，用于从远程Maven上获取依赖。



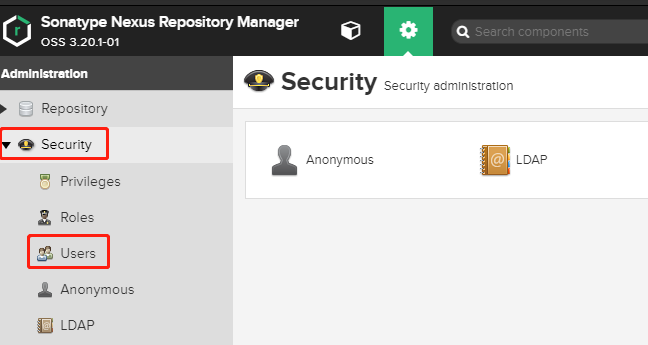
1. 点击maven-central 进入配置页面。

在配置页面的下面，Proxy处 可以配置为国内远程仓库，如阿里云的仓库，只需把地址修改即可。

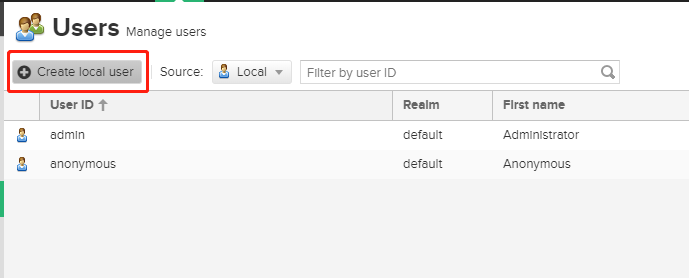


## 创建私服账户

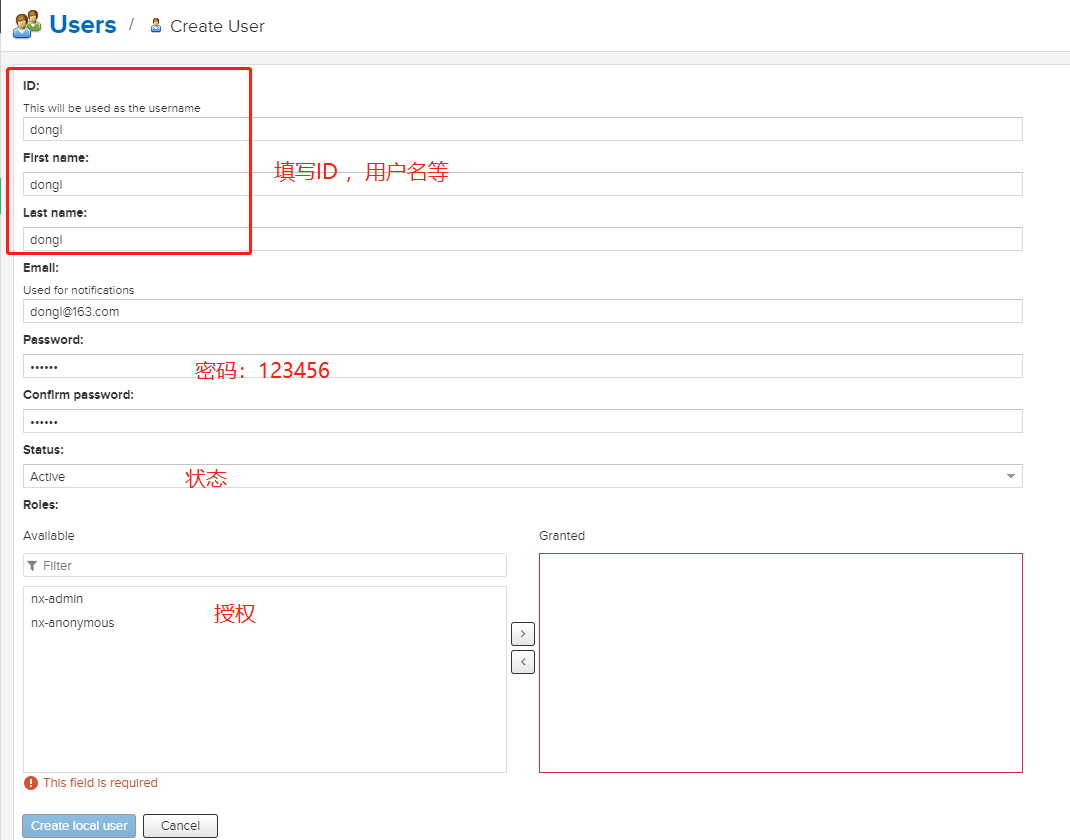
点击Security 模块下的 Users 。



点击 Create local user 进入创建用户界面。



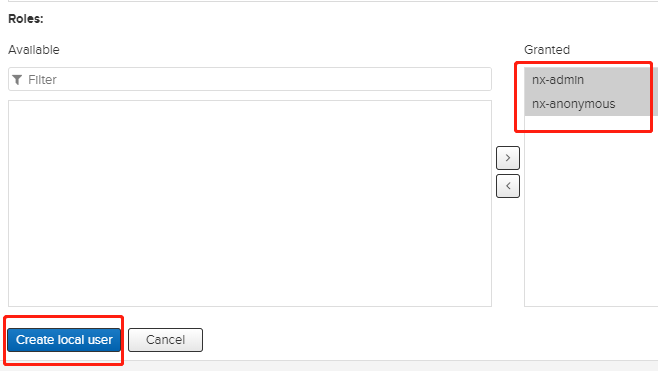
填写用户名，邮箱，密码，状态和授权角色。



新用户账户：dongl

密码：123456

点击 Create local user 创建用户成功。



## 新的用户登录

输入账户：dongl

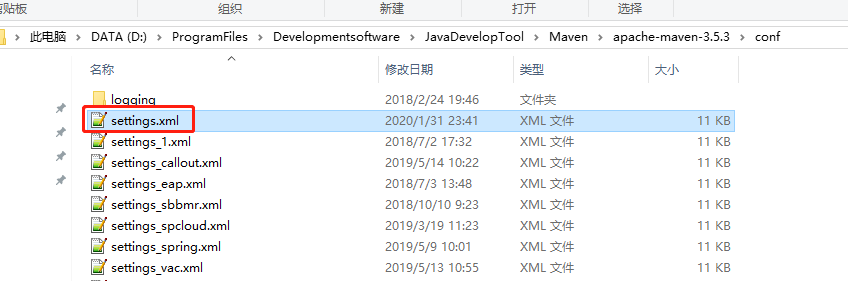
密码：123456



# 项目打包Jar到私服

## 配置本地settings.xml

找到本地的Maven 安装包下的settings.xml 文件。



修改 settings.xml 文件，找到 <servers> 标签，在<servers> 标签中添加：

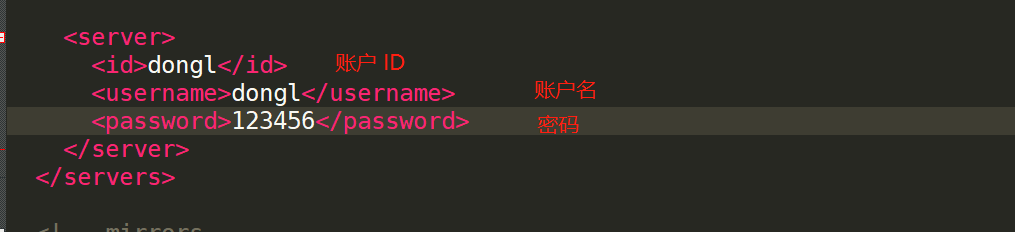
<server>

<id>dongl</id>

<username>dongl</username>

<password>123456</password>

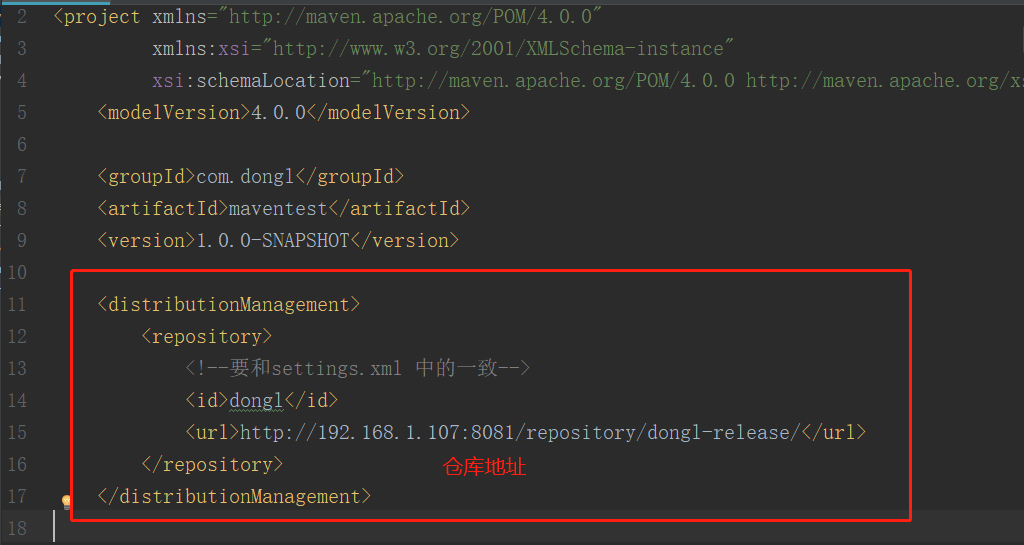
</server>



## 在项目中引入仓库地址

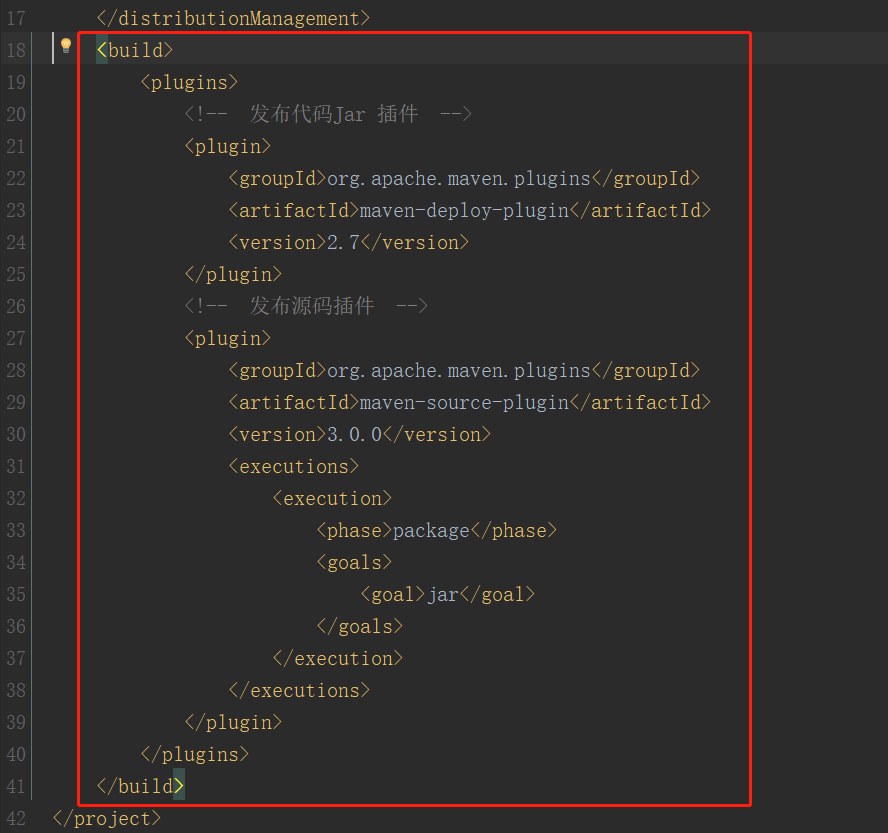
需要在项目的 pom.xml 文件中 引入仓库地址。

<distributionManagement>  
 <repository>  
 <!--要和settings.xml 中的一致-->  
 <id>dongl</id>  
 <url>http://192.168.1.107:8081/repository/dongl-release/</url>  
 </repository>  
</distributionManagement>



Maven配置发布Jar和源码，在项目的pom 文件在引入如下依赖：

<build>  
 <plugins>  
 <!-- 发布代码Jar 插件 -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-deploy-plugin</artifactId>  
 <version>2.7</version>  
 </plugin>  
 <!-- 发布源码插件 -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-source-plugin</artifactId>  
 <version>3.0.0</version>  
 <executions>  
 <execution>  
 <phase>package</phase>  
 <goals>  
 <goal>jar</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 </plugins>  
</build>

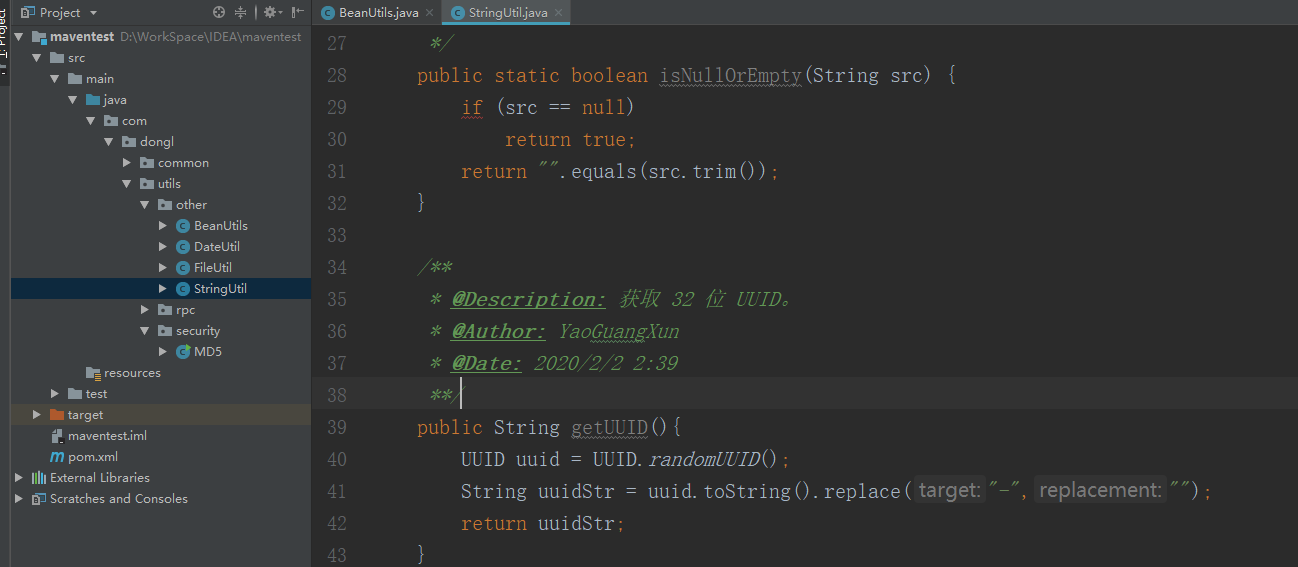


完整 pom文件 如下：

<modelVersion>4.0.0</modelVersion>  
  
 <groupId>com.dongl</groupId>  
 <artifactId>maventest</artifactId>  
 <version>1.0.0-SNAPSHOT</version>  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>

<distributionManagement>  
 <repository>  
 <!--要和settings.xml 中的一致-->  
 <id>dongl</id>  
 <url>http://192.168.1.107:8081/repository/dongl-release/</url>  
 </repository>  
 </distributionManagement>  
 <build>  
 <plugins>  
 <!-- 发布代码Jar 插件 -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-deploy-plugin</artifactId>  
 <version>2.7</version>  
 </plugin>  
 <!-- 发布源码插件 -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-source-plugin</artifactId>  
 <version>3.0.0</version>  
 <executions>  
 <execution>  
 <phase>package</phase>  
 <goals>  
 <goal>jar</goal>  
 </goals>  
 </execution>  
 </executions>  
 </plugin>  
 </plugins>  
 </build>  
</project>

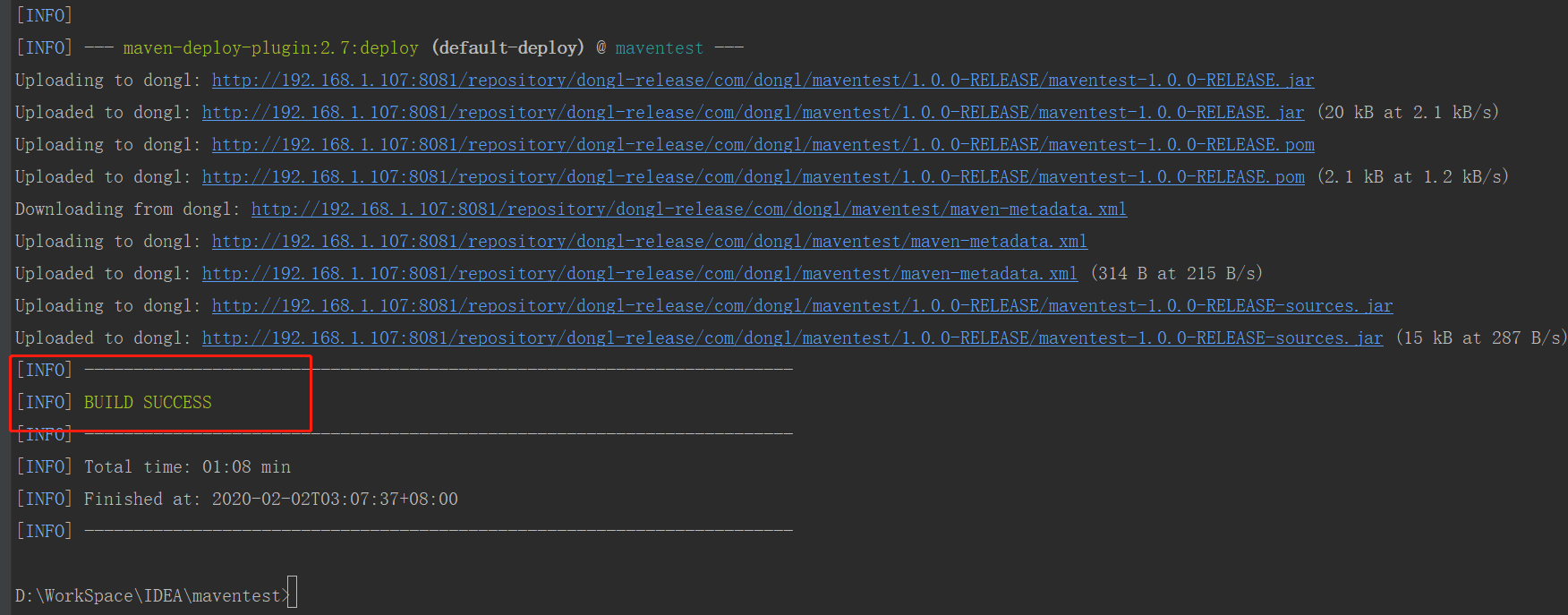
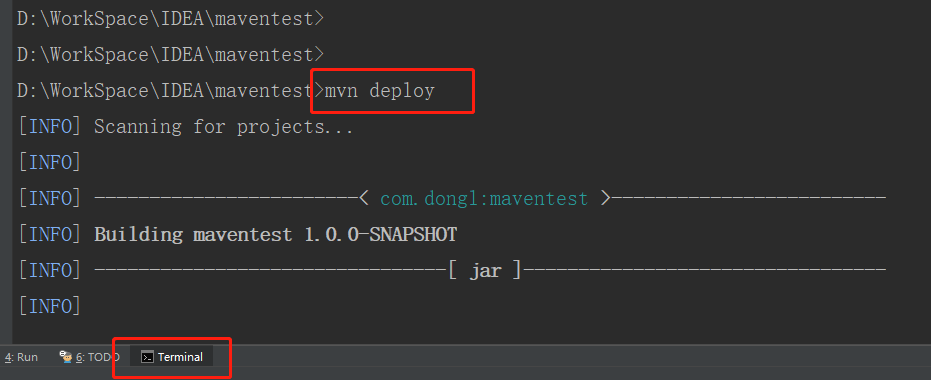
在项目中写一些工具类。



## 把Jar打包到私服

把项目打成jar包 上传到私服上。

在 Terminal 控制台输入命令： mvn deploy



执行过程遇到的问题

Return code is: 400 , ReasonPhrase:Repos

itory version policy: RELEASE does not allow metadata in path: com/dongl/maventest/1.0.0-SNAPSHOT/maven-metadata.xml.

原因：

远程仓库的版本：RELEASE 和本地项目的版本：1.0.0-SNAPSHOT 不一致。



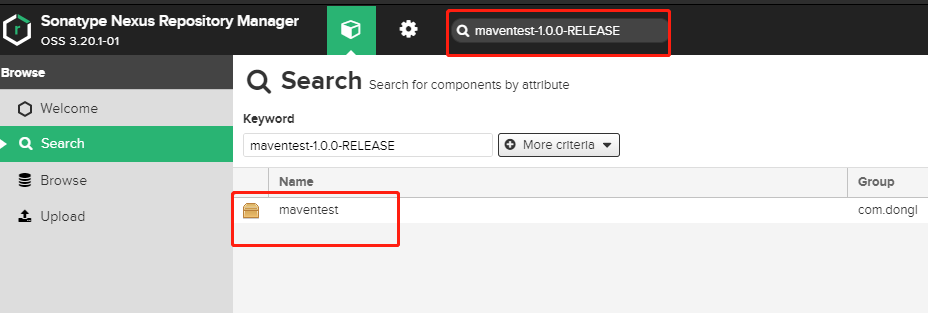
解决：修改本地版本：

把 1.0.0-SNAPSHOT 改成 ：1.0.0-RELEASE



## 私服查看Jar

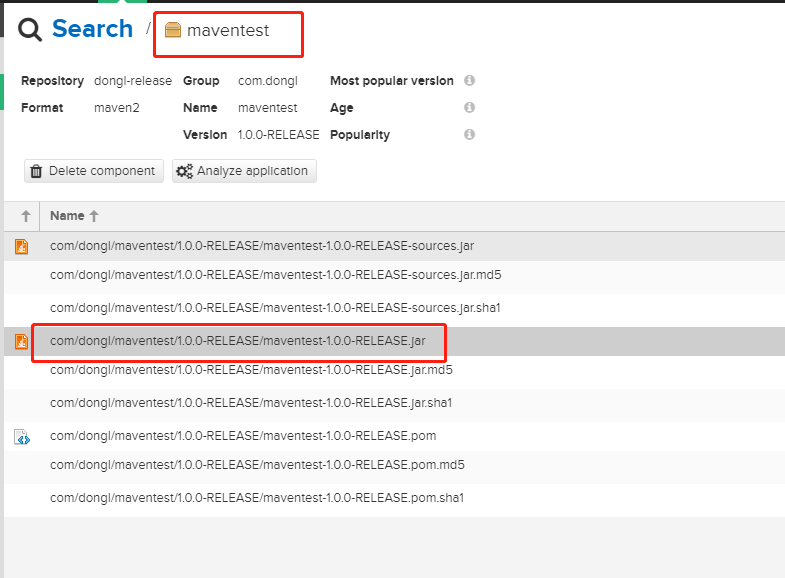
在Nexus 的搜索框中输入 ：maventest-1.0.0-RELEASE 进行搜索。



## 私服的jar包下载

点击 maventest ，进入到文件夹，然后点击

com/dongl/maventest/1.0.0-RELEASE/maventest-1.0.0-RELEASE.jar；



# 上传第三方jar

有时在官方仓库没有的jar包，需要上传到私服上，供大家使用。

执行命令：

**mvn deploy:deploy-file**

**-DgroupId=com.google.code.kaptcha**

**-DartifactId=kaptcha -Dversion=2.3.2**

**-Dpackaging=jar**

**-Dfile=C:\Users\Administrator\Desktop\kaptcha-2.3.2.jar**

**-Durl=http://192.168.1.107:8081/repository/dongl-release/**

**-DrepositoryId=dongl**

命令解释：

-DgroupId=　　　 　　　　　　自定义  
-DartifactId=　　　 　　　　　自定义  
-Dversion=　　　　　　 　　　自定义  三个自定义，构成pom.xml文件中的坐标  
-Dpackaging=jar　　　　　　　上传的类型是jar类型  
-Dfile=               　　　　　　　jar的本地磁盘位置  
-Durl=                                         　hosted资源库的地址  
-DrepositoryId=nexus-releases　　setting.xml文件中配置的ID

# 项目引入私服

在需要从私服中下载jar包的项目的pom.xml中加入如下配置：

第一种方法：

<repositories>

        <repository>

            <id>dongl</id>

             <url>http://192.168.1.107:8081/repository/dongl-release/</url>

            <releases>

                <enabled>true</enabled>

            </releases>

        </repository>

    </repositories>

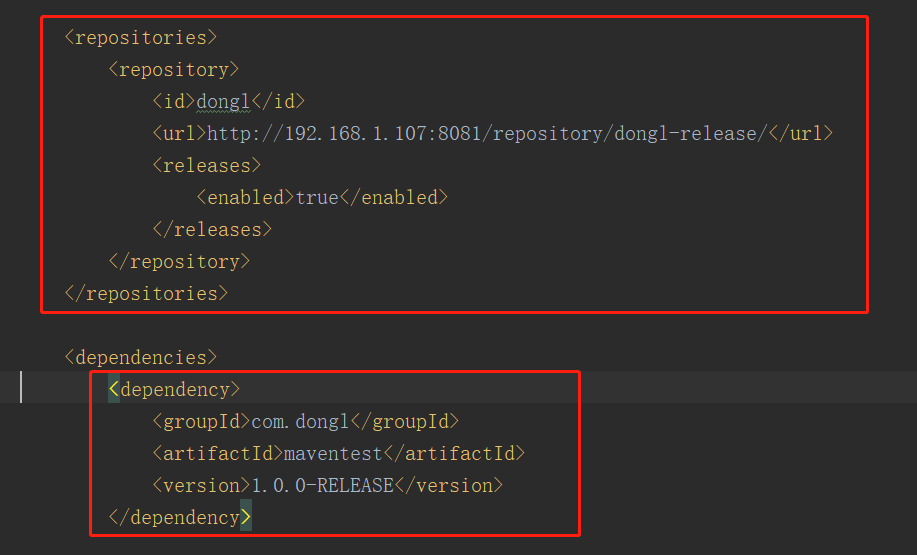
<dependency>

<groupId>com.dongl</groupId>

<artifactId>maventest</artifactId>

<version>1.0.0-RELEASE</version>

</dependency>



第二种方法：

 1).在settings.xml中配置私服下载的位置（复制下面代码至settings.xml中进行修改）

<!--私服下载的位置-->

<profile>

<!--profile的id-->

<id>dev</id>

<repositories>

<repository>

<!--仓库id，repositories可以配置多个仓库，保证id不重复-->

<id>nexus</id>

<!--仓库地址，即nexus仓库组的地址-->

<url>http://localhost:8081/nexus/content/groups/public/</url>

<!--是否下载releases构件-->

<releases>

<enabled>true</enabled>

</releases>

<!--是否下载snapshots构件-->

<snapshots>

<enabled>true</enabled>

</snapshots>

</repository>

</repositories>

<pluginRepositories>

<!-- 插件仓库，maven的运行依赖插件，也需要从私服下载插件 -->

<pluginRepository>

<!-- 插件仓库的id不允许重复，如果重复后边配置会覆盖前边 -->

<id>public</id>

<name>Public Repositories</name>

<url>http://localhost:8081/nexus/content/groups/public/</url>

</pluginRepository>

</pluginRepositories>

</profile>

2).在settings.xml中配置激活哪一个私服下载的位置（复制下面代码至settings.xml中）

<!---激活下载的地址-->

<activeProfiles>

<!-- 激活下载地址的id -->

<activeProfile>dev</activeProfile>

</activeProfiles>

**注：**此时就不需要在项目的 pom.xml 配置如下配置了。

<repositories>

        <repository>

            <id>dongl</id>

             <url>http://192.168.1.107:8081/repository/dongl-release/</url>

            <releases>

                <enabled>true</enabled>

            </releases>

        </repository>

    </repositories>