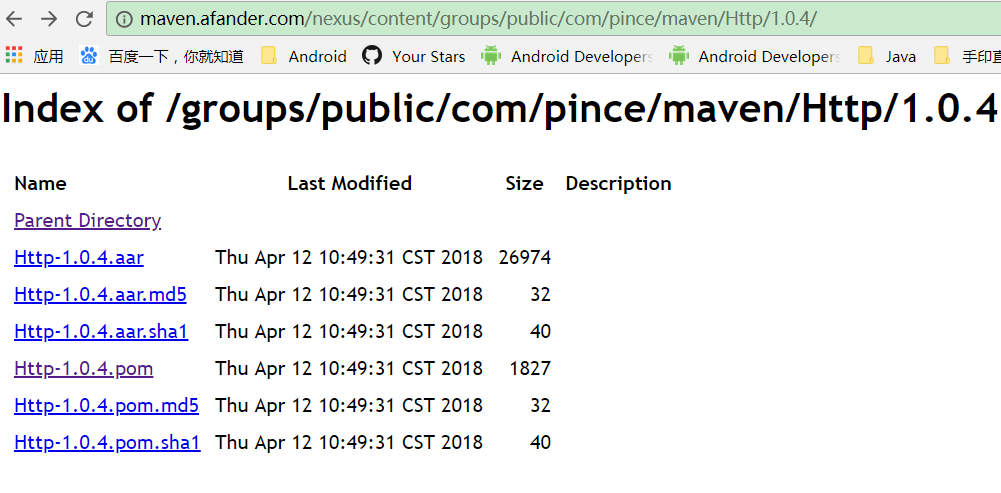
jcenter : https://blog.csdn.net/linglongxin24/article/details/53415932

Maven : <https://www.whatled.com/post-1969.html>

Maven : <https://www.jianshu.com/p/9740778b154f>

Gitlab/Oschina/GitHub(私有库是收费的)数据代码管理控制工具

Maven属于jar/aar的版本管理工具



Nexus用于管理maven私服

# 1：Nexus下载，配置

## 下载：

<https://www.sonatype.com/download-oss-sonatype>

## 启动服务

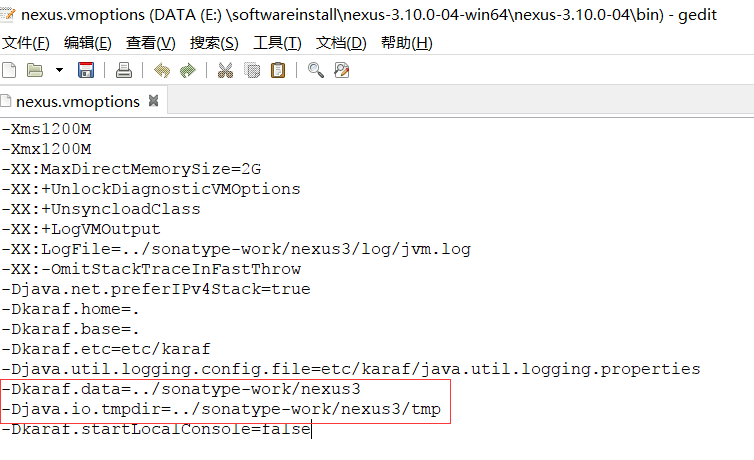
E:\softwareinstall\nexus-3.10.0-04-win64\nexus-3.10.0-04\bin\nexus.exe /run

注意nexus.exe在bin目录下

此时的nexus服务都是安装包默认的数据存储路径、端口、IP、访问根目录、内存大小，可以通过如下操作进行修改

## 修改数据存储路径，文件目录

bin/nexus.vmoptions



## 修改IP、端口、访问根目录，文件目录：



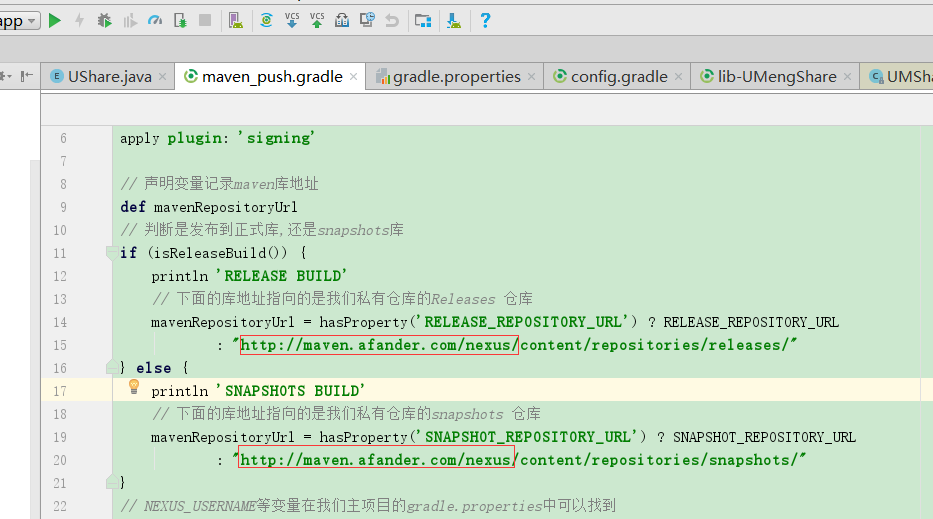
## 浏览器查看：

<http://localhost:8081/>

## create repository

创建一个maven2(host)仓库

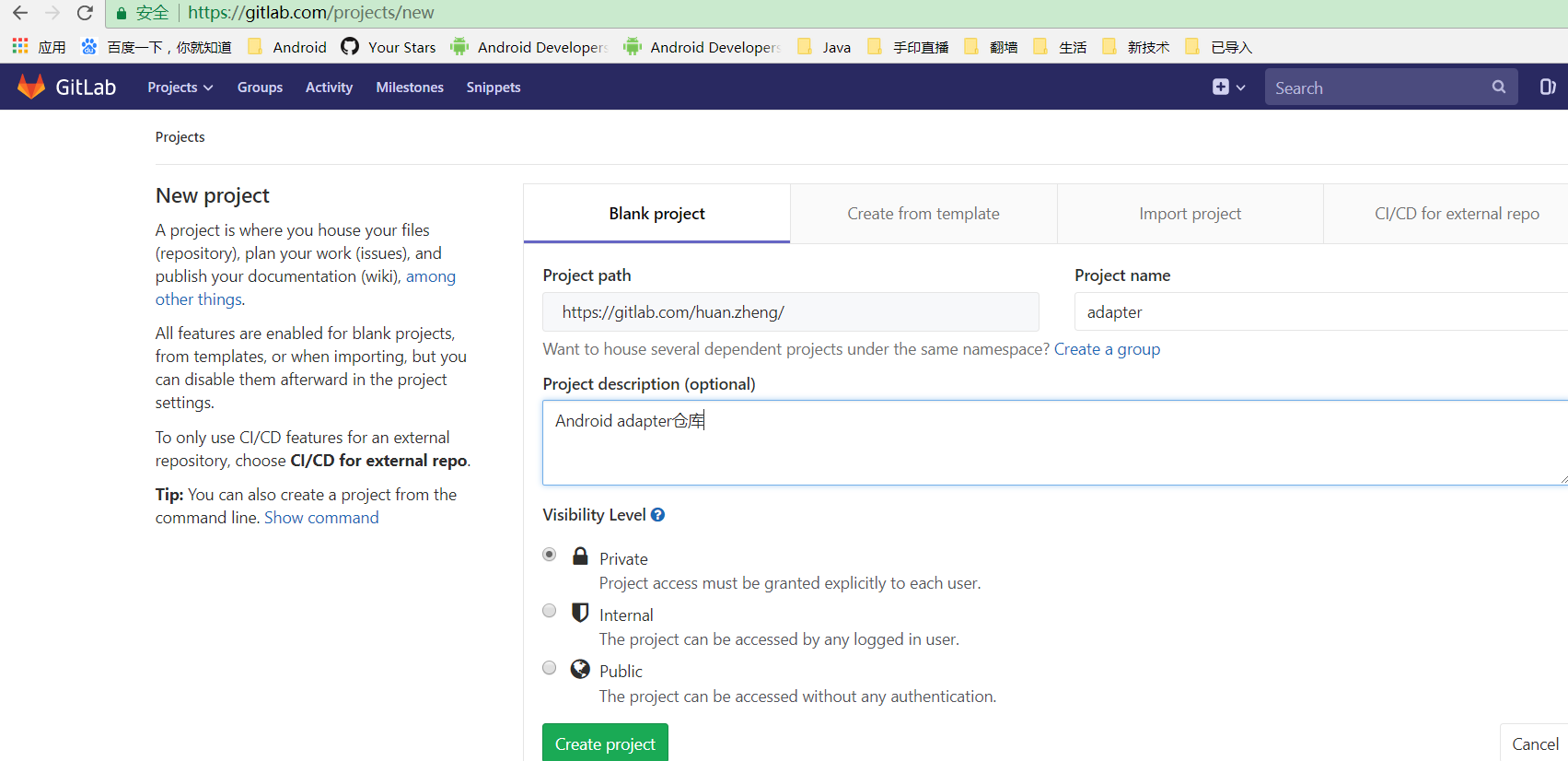
## Maven 仓库配置gradle



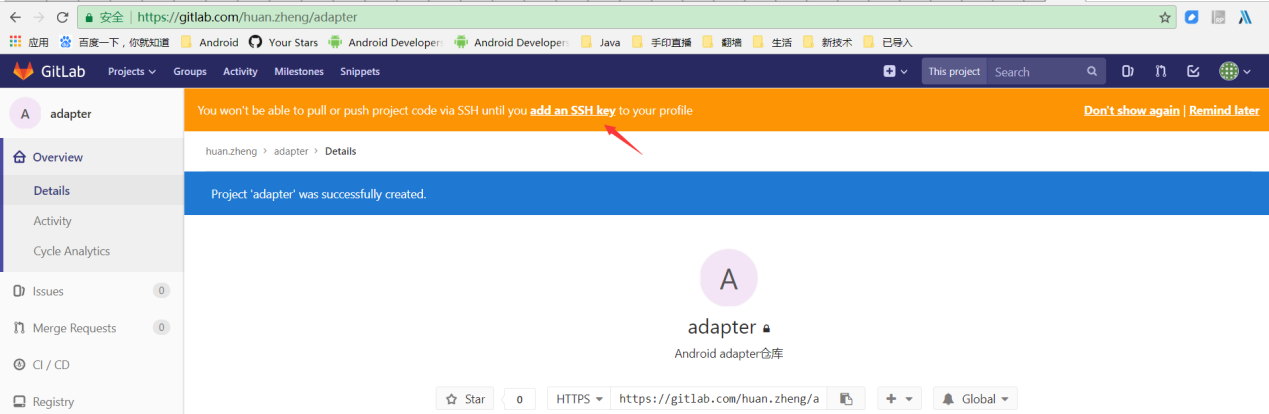
# 建立一个gitlab代码项目

<https://gitlab.com/>

## 登陆进去,Create new project



## 添加ssh key



Git clone项目到本地

**C:\Users\zhenghuan\git\lib\mine\adapter**

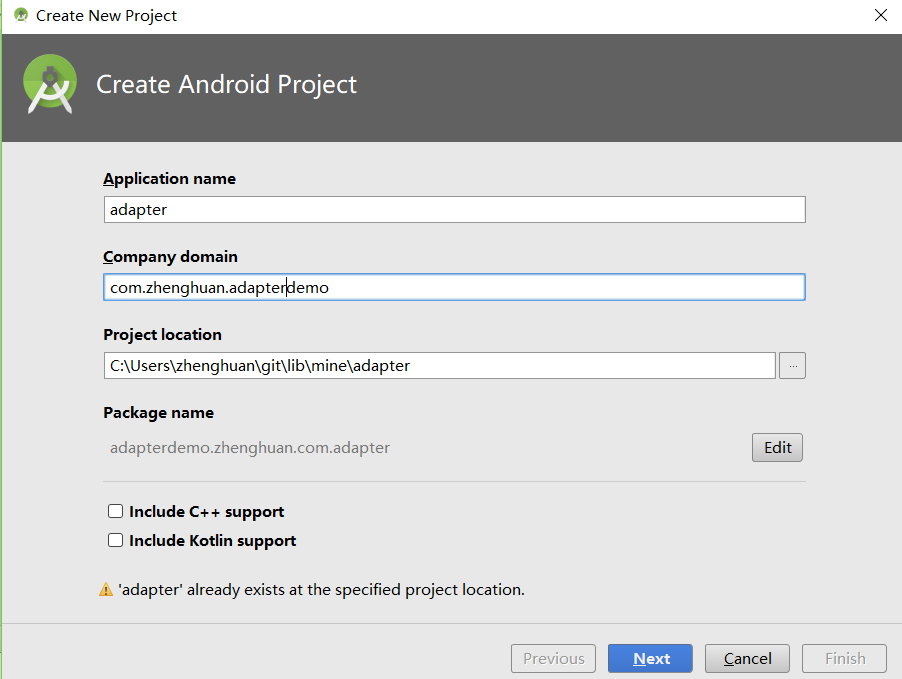
**Gitlab默认新建的项目没有远程分支，使用**

$ git push origin master:master

**推到远程**

## 本地新建一个Android项目

C:\Users\zhenghuan\git\lib\mine\adapter里新建项目



## 添加一个library module

添加完成之后配置项目可编译，接下来进行项目的maven上传配置

# 项目的maven上传配置

## project根目录下gradle.properties添加如下内容：

1. *# Project-wide Gradle settings.  
     
   # IDE (e.g. Android Studio) users:  
   # Gradle settings configured through the IDE \*will override\*  
   # any settings specified in this file.  
     
   # For more details on how to configure your build environment visit  
   # http://www.gradle.org/docs/current/userguide/build\_environment.html  
     
   # Specifies the JVM arguments used for the daemon process.  
   # The setting is particularly useful for tweaking memory settings.***org.gradle.jvmargs**=**-Xmx1536m  
   android.enableBuildCache**=**false***# When configured, Gradle will run in incubating parallel mode.  
   # This option should only be used with decoupled projects. More details, visit  
   # http://www.gradle.org/docs/current/userguide/multi\_project\_builds.html#sec:decoupled\_projects  
   # org.gradle.parallel=true  
     
   # properties for maven2 repository  
   # nexus服务器登录时候的用户名/密码***NEXUS\_USERNAME**=**admin  
   NEXUS\_PASSWORD**=**8bSb7LgufJ1QwzGc***# 在POM文件中使用的group ID***POM\_GROUP\_ID**=**com.pince.maven***# POM文件中指向你网站的地址***POM\_URL**=**http://www.afander.com/***# SCM是指版本管理工具,以下是他的相关信息***POM\_SCM\_URL**=**http://www.afander.com/  
   POM\_SCM\_CONNECTION**=**http://www.afander.com/  
   POM\_SCM\_DEV\_CONNECTION**=**http://www.afander.com/***# 你的开放协议相关信息***POM\_LICENCE\_NAME**= **Apache License Version 2.0  
   POM\_LICENCE\_URL**= **http://oztvp5wi1.bkt.clouddn.com/afander\_maven\_license.license  
   POM\_LICENCE\_DIST**=**Apache License Version 2.0***# 开发者的相关信息***POM\_DEVELOPER\_ID**=**cc  
   POM\_DEVELOPER\_NAME**=**cc**

Demo项目配置：默认的用户名和密码：admin/admin123

*# properties for maven2 repository  
# nexus服务器登录时候的用户名/密码***NEXUS\_USERNAME**=**admin  
NEXUS\_PASSWORD**=**admin123***# 在POM文件中使用的group ID***POM\_GROUP\_ID**=**com.zhenghuan.maven***# POM文件中指向你网站的地址***POM\_URL**=**http://www.baidu.com/***# SCM是指版本管理工具,以下是他的相关信息***POM\_SCM\_URL**=**http://www.baidu.com/  
POM\_SCM\_CONNECTION**=**http://www.baidu.com/  
POM\_SCM\_DEV\_CONNECTION**=**http://www.baidu.com/***# 你的开放协议相关信息***POM\_LICENCE\_NAME**= **Apache License Version 2.0  
POM\_LICENCE\_URL**=   
**POM\_LICENCE\_DIST=Apache License Version 2.0***# 开发者的相关信息***POM\_DEVELOPER\_ID**=**zhenghuan  
POM\_DEVELOPER\_NAME**=**zhenghuan**

## maven\_push.gradle

project根目录下新建maven\_push.gradle（复制）

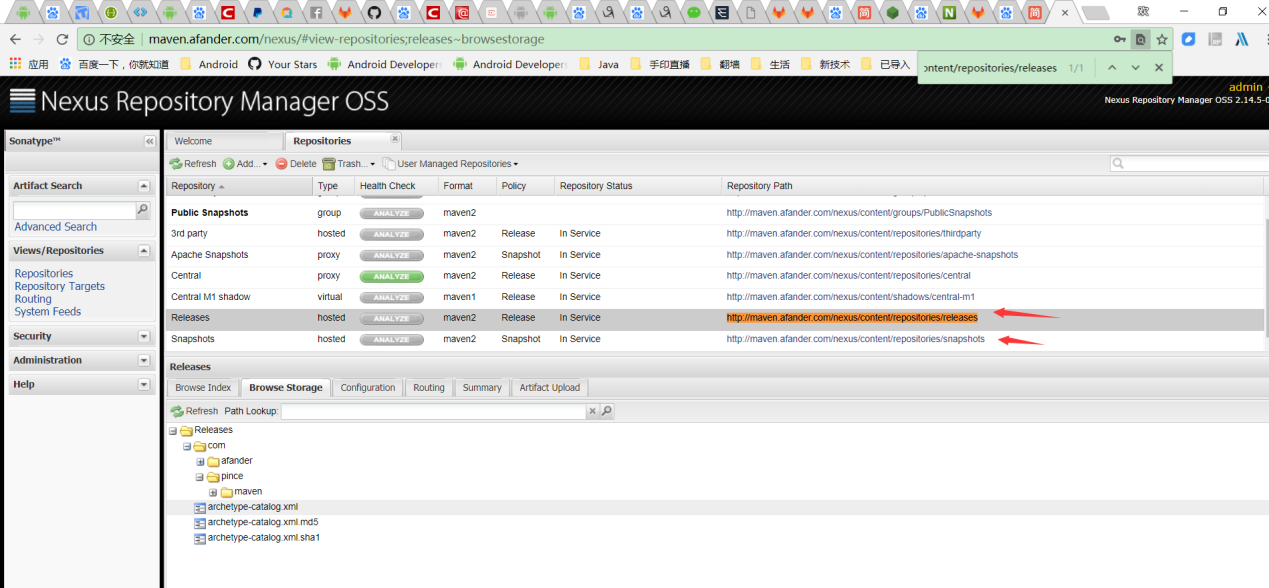
1. *// The Maven plugin adds support for deploying artifacts to Maven repositories.  
   // 一个可以让你把库上传到maven仓库的插件*apply **plugin**: **'maven'***// The signing plugin adds the ability to digitally sign built files and artifacts. These digital signatures can then be used to prove who built the artifact the signature is attached to as well as other information such as when the signature was generated.  
   // 对库文件进行数字签名的插件,可以通过签名知道谁创建了这个库文件,签名的时间等等信息*apply **plugin**: **'signing'***// 声明变量记录maven库地址***def** mavenRepositoryUrl  
   *// 判断是发布到正式库,还是snapshots库***if** (isReleaseBuild()) {  
    println **'RELEASE BUILD'** *// 下面的库地址指向的是我们私有仓库的Releases 仓库* mavenRepositoryUrl = hasProperty(**'RELEASE\_REPOSITORY\_URL'**) ? RELEASE\_REPOSITORY\_URL  
    : **"http://maven.afander.com/nexus/content/repositories/releases/"**} **else** {  
    println **'SNAPSHOTS BUILD'** *// 下面的库地址指向的是我们私有仓库的snapshots 仓库* mavenRepositoryUrl = hasProperty(**'SNAPSHOT\_REPOSITORY\_URL'**) ? SNAPSHOT\_REPOSITORY\_URL  
    : **"http://maven.afander.com/nexus/content/repositories/snapshots/"**}  
   *// NEXUS\_USERNAME等变量在我们主项目的gradle.properties中可以找到***def** getRepositoryUsername() {  
    **return** hasProperty(**'NEXUS\_USERNAME'**) ? NEXUS\_USERNAME : **""**}  
     
   **def** getRepositoryPassword() {  
    **return** hasProperty(**'NEXUS\_PASSWORD'**) ? NEXUS\_PASSWORD : **""**}  
   *// 根据我们在likelib下gradle.properties中声明的版本名称,来分辨是Release版本还是 snapshots版本***def** isReleaseBuild() {  
    **return** !VERSION\_NAME.contains(**"SNAPSHOT"**);  
   }  
     
   *//"afterEvaluate是什么鸟？你可以理解为在配置阶段要结束，项目评估完会走到这一步。" 引用自http://jiajixin.cn/2015/08/07/gradle-android/*afterEvaluate { project ->  
    *// 我们声明我们要执行的上传到maven的task* uploadArchives {  
    repositories {  
    mavenDeployer {  
    beforeDeployment { MavenDeployment deployment -> signing.signPom(deployment) }  
    *// 我们类比下compile com.squareup.okhttp:okhttp:2.7.0  
    // artifactId 对应com.squareup.okhttp; groupId 对应okhttp;version对应2.7.0  
    // 这样就类似坐标的方式定位到了制定的库文件* pom.artifactId = POM\_ARTIFACT\_ID  
    pom.groupId = POM\_GROUP\_ID  
    pom.version = VERSION\_NAME  
     
    *// 授权验证,这里也就是你登录搭建的私服服务器时候的用户名\密码* repository(**url**: mavenRepositoryUrl) {  
    authentication(**userName**: getRepositoryUsername(), **password**: getRepositoryPassword())  
    }  
     
    *// 这里是配置我们maven库需要的pom.xml文件的各个内容,具体意思我们在主目录gradle.properties中解释* pom.project {  
    name POM\_NAME  
    packaging POM\_PACKAGING  
    description POM\_DESCRIPTION  
    url POM\_URL  
     
    scm {  
    url POM\_SCM\_URL  
    connection POM\_SCM\_CONNECTION  
    developerConnection POM\_SCM\_DEV\_CONNECTION  
    }  
     
    licenses {  
    license {  
    name POM\_LICENCE\_NAME  
    url POM\_LICENCE\_URL  
    distribution POM\_LICENCE\_DIST  
    }  
    }  
     
    developers {  
    developer {  
    id POM\_DEVELOPER\_ID  
    name POM\_DEVELOPER\_NAME  
    }  
    }  
    }  
    }  
    }  
    }  
     
    *// 进行数字签名* signing {  
    required { isReleaseBuild() && gradle.taskGraph.hasTask(**"uploadArchives"**) }  
    sign configurations.**archives** }  
     
    *// type显示指定任务类型或任务, 这里指定要执行Javadoc这个task,这个task在gradle中已经定义* task androidJavadocs(**type**: Javadoc) {  
    *// 设置源码所在的位置* source = android.sourceSets.main.java.sourceFiles  
    }  
     
   *// // 生成javadoc.jar  
   // task androidJavadocsJar(type: Jar) {  
   // // 指定文档名称  
   // classifier = 'javadoc'  
   // from androidJavadocs.destinationDir  
   // }  
     
   // // 生成sources.jar  
   // task androidSourcesJar(type: Jar) {  
   // classifier = 'sources'  
   // from android.sourceSets.main.java.sourceFiles  
   // }  
     
    // 产生相关配置文件的任务* artifacts {  
   *// archives androidSourcesJar  
   // archives androidJavadocsJar* }  
   }

必须配置的是：

*// 下面的库地址指向的是我们私有仓库的Releases 仓库*mavenRepositoryUrl = hasProperty(**'RELEASE\_REPOSITORY\_URL'**) ? RELEASE\_REPOSITORY\_URL  
 : **"http://maven.afander.com/nexus/content/repositories/releases/"**

*// 下面的库地址指向的是我们私有仓库的snapshots 仓库*mavenRepositoryUrl = hasProperty(**'SNAPSHOT\_REPOSITORY\_URL'**) ? SNAPSHOT\_REPOSITORY\_URL  
 : **"http://maven.afander.com/nexus/content/repositories/snapshots/"**

这里的地址对应是在如下配置:



因此改为:



其余内容引用了根目录下和library目录下的gradle.properties文件中定义的配置参数

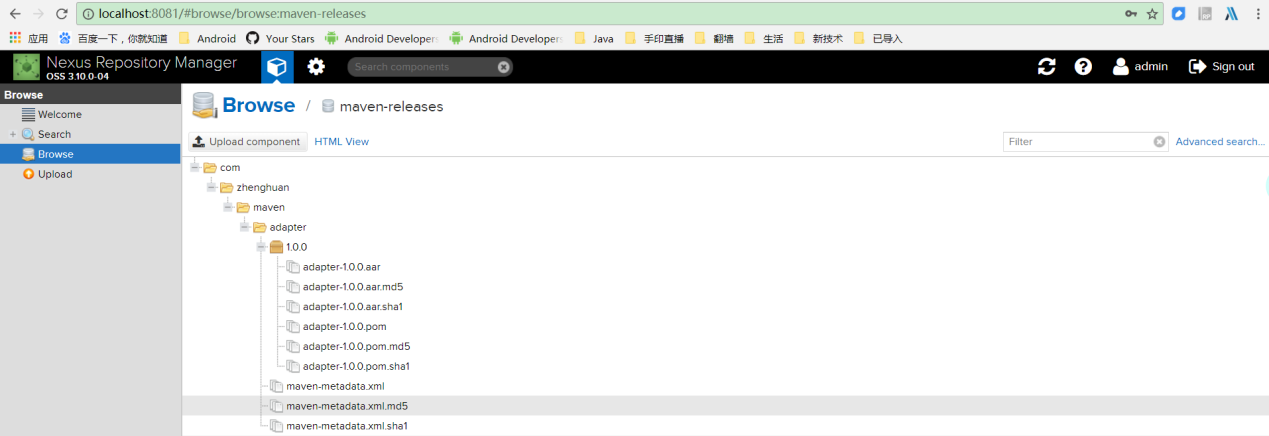
## lib module目录下添加gradle.proerties,并添加如下内容

*# 库名称***POM\_NAME**=**afander***# artifactId***POM\_ARTIFACT\_ID**=**UMengShare***# 库的打包格式为aar, 常见的还有jar***POM\_PACKAGING**=**aar***# 库的描述,说明他是干啥的***POM\_DESCRIPTION**=**UMengShare***# 要发布的版本号,snapshots 版本可以使用格式 1.0.0-SNAPSHOT***VERSION\_NAME**=**1.0.4-SNAPSHOT**

## 在你的lib module的build.gradle添加一行

apply **from**: **'../maven\_push.gradle'**

## 在terminal输入gradlew uploadArchives即可完成上传：



# 引用端配置

## project根目录的build.gradle添加

`

allprojects {

repositories {

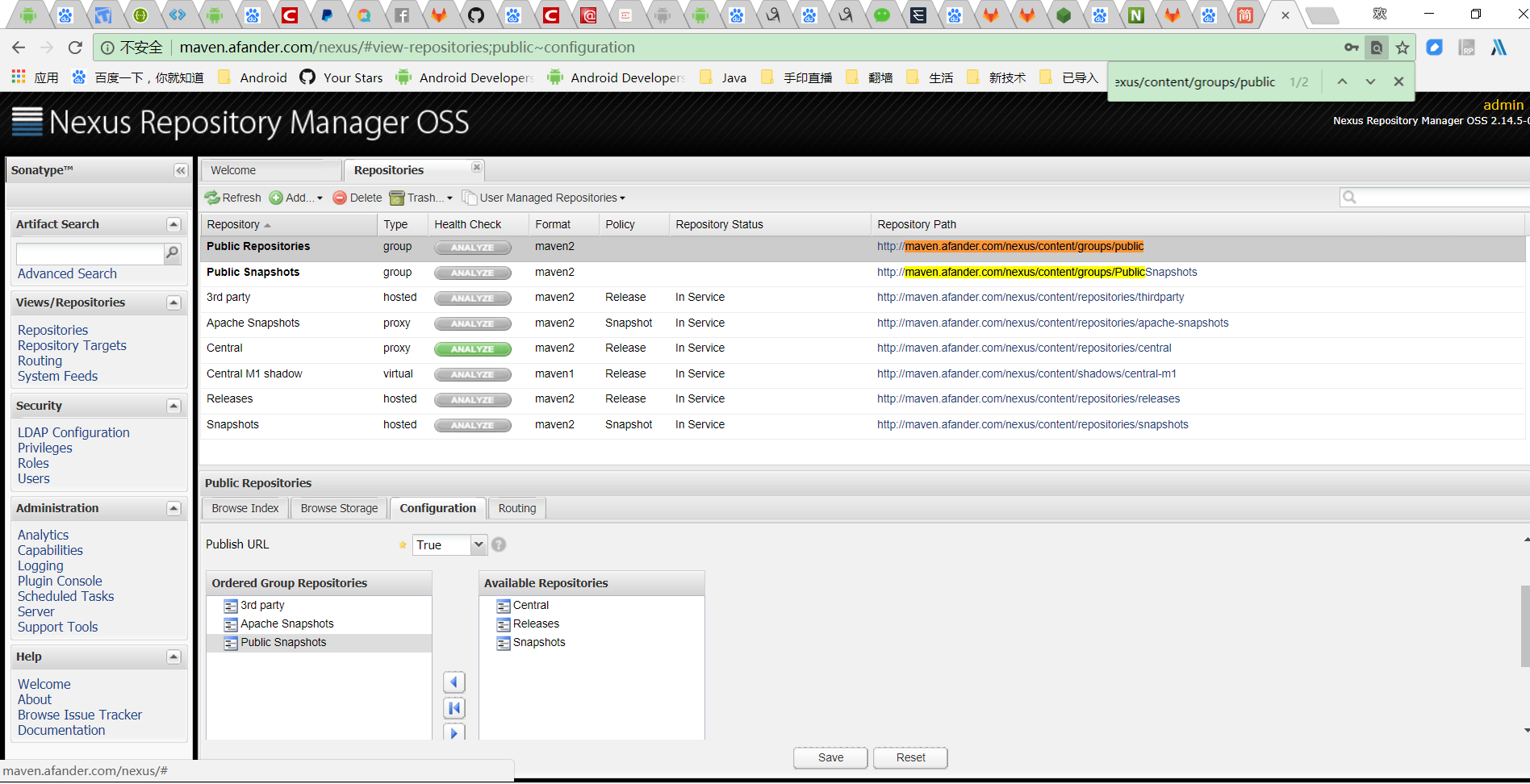
jcenter()

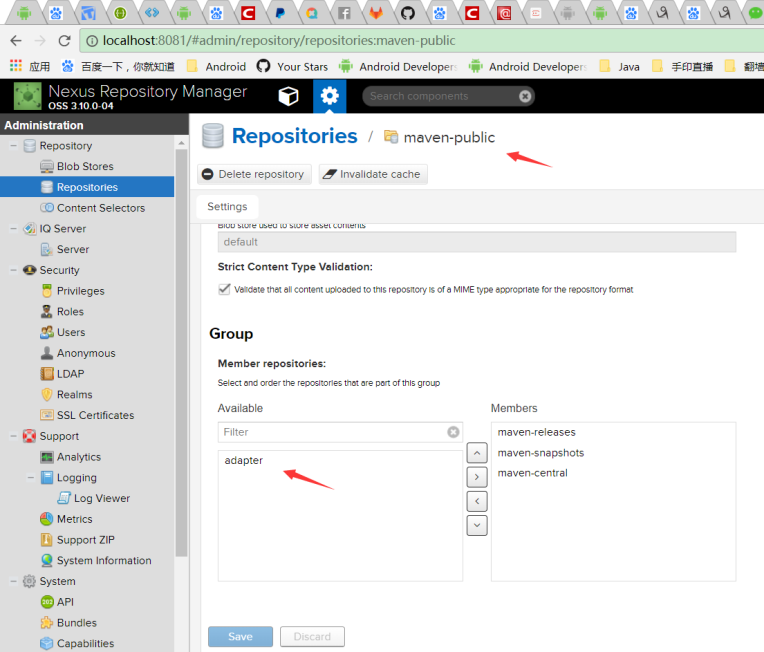
maven { url **"http://maven.afander.com/nexus/content/groups/public/"** }  
maven { url **"http://maven.afander.com/nexus/content/repositories/snapshots/"** }

}

}

其中的groups/public是一个group类型的maven仓库:





## module引用

compile ‘groupid:artifactid:versionname’