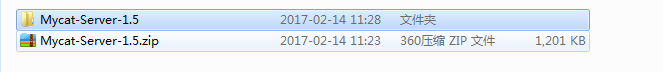
# mycat发布RELEASE版本

Sonatype 官网：<http://www.sonatype.org/>

Sonatype OSS：[https://oss.sonatype.org](https://oss.sonatype.org/)

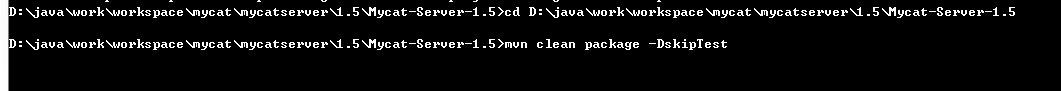
## 下载mycat的代码，这里以1.5为例，windows平台

git地址: <https://codeload.github.com/MyCATApache/Mycat-Server/zip/1.5>

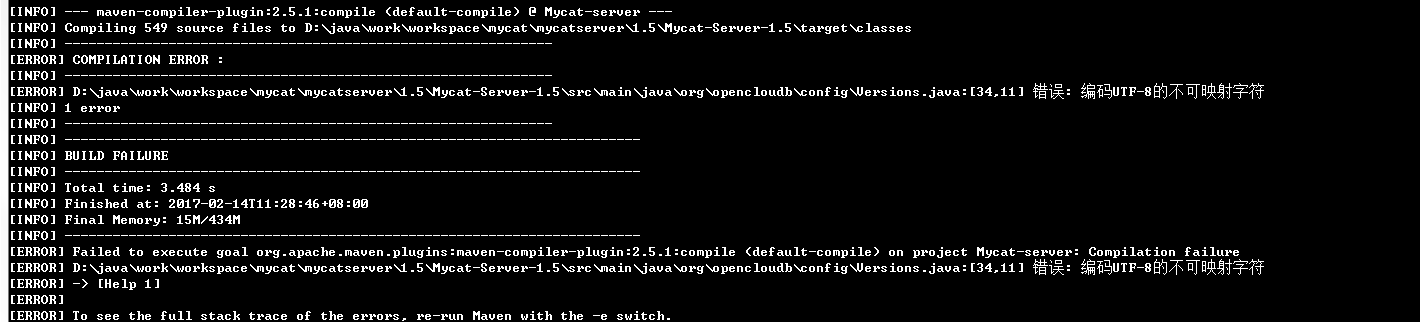


## 至mycat-server目录，测试打包

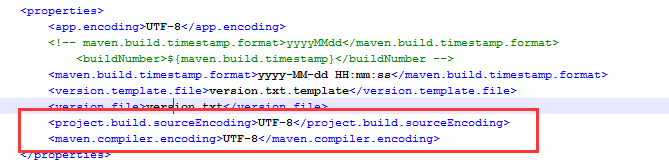
执行命令:mvn clean package -DskipTests



可能的错误:



解决办法:

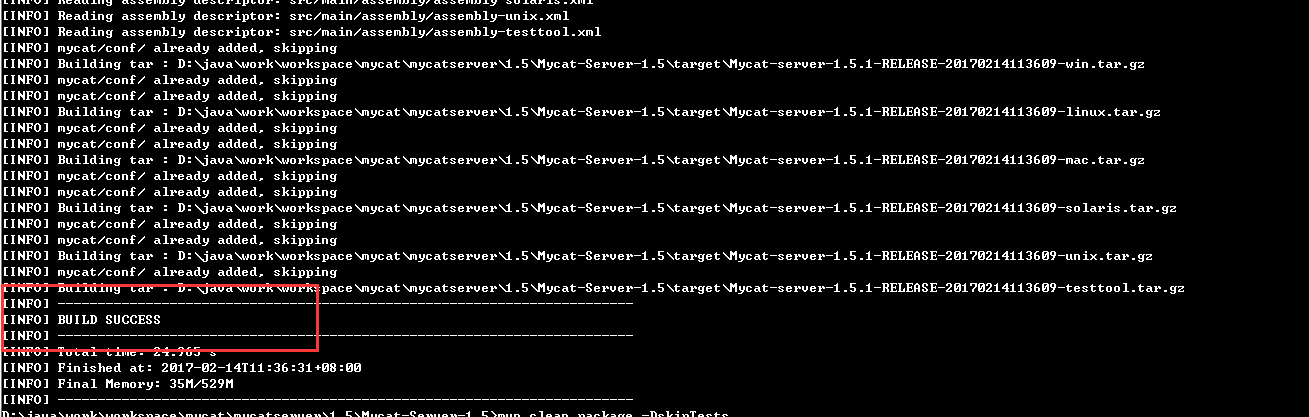


指定编码信息

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.encoding>UTF-8</maven.compiler.encoding>

见此说明打包是成功的

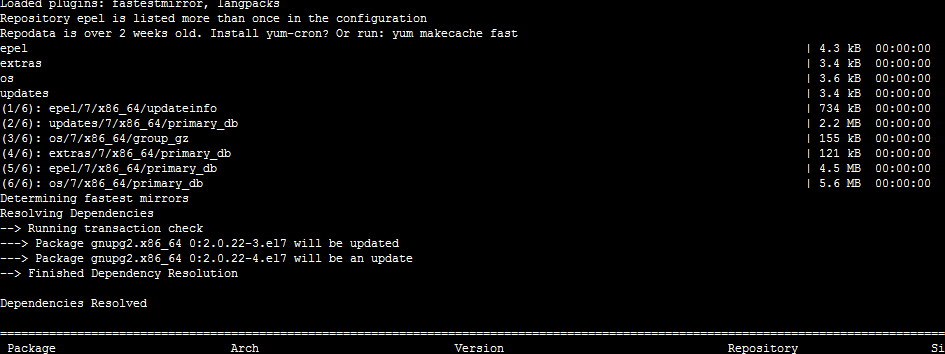


发布信息的检查

## 使用 GPG 生成密钥对

如果是 Windows 操作系统，需要下载 [Gpg4win](http://gpg4win.org/) 软件来生成密钥对。建议大家下载 Gpg4win-Vanilla 版本，因为它仅包括 GnuPG，这个工具才是我们所需要的。

如果linux安装，可使用yum install gnupg



安装 GPG 软件后，打开命令行窗口，依次做以下操作：

1. 查看是否安装成功

gpg --version

能够显示 GPG 的版本信息，说明安装成功了。

2. 生成密钥对

gpg --gen-key

此时需要输入姓名、邮箱等字段，其它字段可使用默认值，此外，还需要输入一个 Passphase，相当于一个密钥库的密码，一定不要忘了，也不要告诉别人，最好记下来，因为后面会用到。

3. 查看公钥

gpg --list-keys

输出如下信息：

C:/Users/huangyong/AppData/Roaming/gnupg/pubring.gpg

----------------------------------------------------

pub 2048R/82DC852E 2014-04-24

uid hy\_think <hy\_think@163.com>

sub 2048R/3ACA39AF 2014-04-24

可见这里的公钥的 ID 是：82DC852E，很明显是一个 16 进制的数字，马上就会用到。

4. 将公钥发布到 PGP 密钥服务器

gpg --keyserver hkp://pool.sks-keyservers.net --send-keys 82DC852E

此后，可使用本地的私钥来对上传构件进行数字签名，而下载该构件的用户可通过上传的公钥来验证签名，也就是说，大家可以验证这个构件是否由本人上传的，因为有可能该构件被坏人给篡改了。

5. 查询公钥是否发布成功

gpg --keyserver hkp://pool.sks-keyservers.net --recv-keys 82DC852E

实际上就是从 key server 上通过公钥 ID 来接收公钥，此外，也可以到 sks-keyservers.net 上通过公钥 ID 去查询。

## 修改 Maven 配置文件

需要修改的 Maven 配置文件包括：setting.xml（全局级别）与 pom.xml（项目级别）。

1. setting.xml

<!-- lang: xml -->

<settings>

...

<servers>

<server>

<id>oss</id>

<username>用户名</username>

<password>密码</password>

</server>

</servers>

...

</settings>

使用自己注册的 Sonatype 账号的用户名与密码来配置以上 server 信息。

2. pom.xml

<!-- lang: xml -->

<project>

...

<name>mycat-bigmem</name>

<description>mycat big memory framework.</description>

<url>https://github.com/MyCATApache/Mycat-BigMemoryAPI.git/</url>

<licenses>

<license>

<name>The Apache Software License, Version 2.0</name>

<url>http://www.apache.org/licenses/LICENSE-2.0.txt</url>

</license>

</licenses>

<developers>

<developer>

<name>mycat</name>

<email>719867650@qq.com</email>

</developer>

</developers>

<scm>

<connection>scm:git:ssh://apachemycat@github.com/MyCATApache/Mycat-Server.git</connection>

<developerConnection>scm:git:ssh://apachemycat@github.com/MyCATApache/Mycat-Server.git</developerConnection>

<url>scm:git:ssh://apachemycat@github.com/MyCATApache/Mycat-Server.git</url>

</scm>

<profiles>

<profile>

<id>release</id>

<build>

<plugins>

<!-- Source -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-source-plugin</artifactId>

<version>2.2.1</version>

<executions>

<execution>

<phase>package</phase>

<goals>

<goal>jar-no-fork</goal>

</goals>

</execution>

</executions>

</plugin>

<!-- GPG -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-gpg-plugin</artifactId>

<executions>

<execution>

<id>sign-artifacts</id>

<phase>verify</phase>

<goals>

<goal>sign</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

<distributionManagement>

<snapshotRepository>

<id>oss</id>

<url>https://oss.sonatype.org/content/repositories/snapshots/</url>

</snapshotRepository>

<repository>

<id>oss</id>

<url>https://oss.sonatype.org/service/local/staging/deploy/maven2/</url>

</repository>

</distributionManagement>

</profile>

</profiles>

<!-- 注释

<repositories>

<repository>

<id>nexus</id>

<name>local private nexus</name>

<url>http://nexus.mycat.io/content/groups/public</url>

</repository>

</repositories>

<distributionManagement>

<repository>

<id>releases</id>

<name>Internal Releases</name>

<url>http://nexus.mycat.io/content/repositories/releases</url>

</repository>

<snapshotRepository>

<id>snapshots</id>

<name>Internal Snapshots</name>

<url>http://nexus.mycat.io/content/repositories/snapshots</url>

</snapshotRepository>

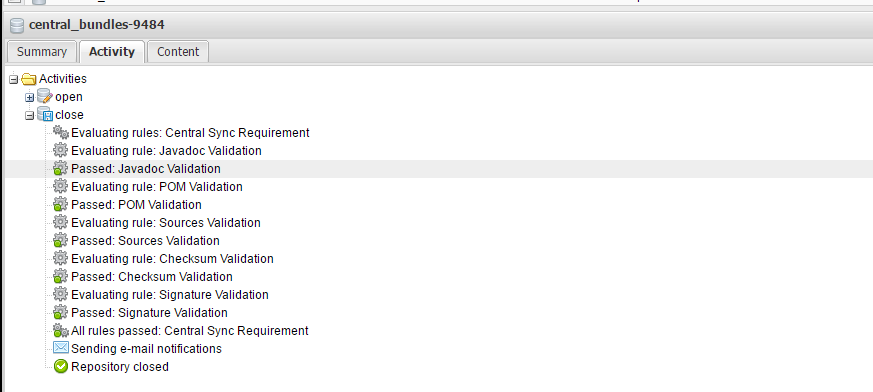
</distributionManagement>

-->

...

</project>

注意：以上 pom.xml 必须包括：name、description、url、licenses、developers、scm 等基本信息，此外，使用了 Maven 的 profile 功能，只有在 release 的时候，创建源码包、创建文档包、使用 GPG 进行数字签名。此外，snapshotRepository 与 repository 中的 id 一定要与 setting.xml 中 server 的 id 保持一致。



## 上传构件到 OSS 中

mvn clean deploy -P release

当执行以上 Maven 命令时，会自动弹出一个对话框，需要输入上面提到的 Passphase，它就是通过 GPG 密钥对的密码，只有自己才知道。随后会看到大量的 upload 信息，而且速度比较慢，经常会 timeout，需要反复尝试。

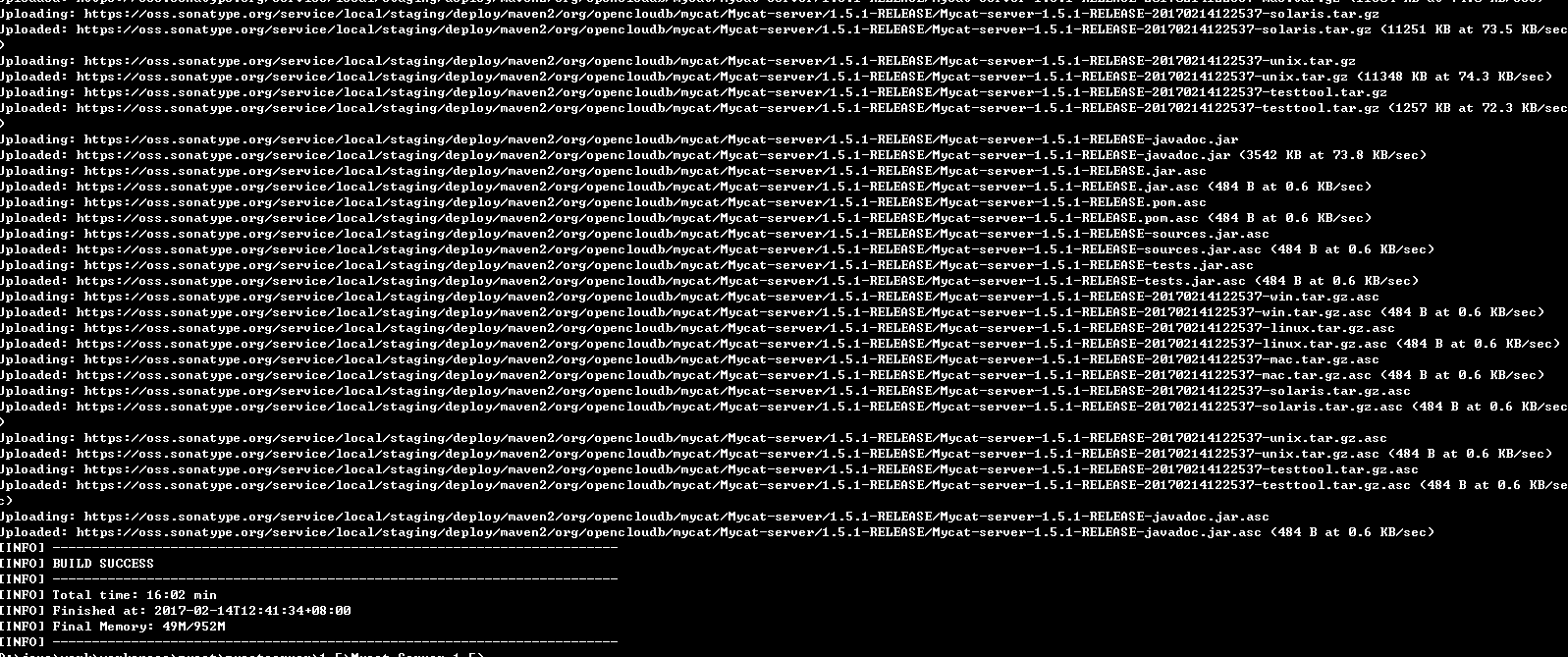
在上传过程中maven的gpg插件会进行验签，此时要确保使用的setting.xml文件是正确的，命令行使用mvn命令使用的是maven安装目录下的conf/setting.xml文件，不要和eclipse里面的混淆了.

上传的时候是需要输入密码的，默认的gpg在命令行执行的时候不会有输入密码的交互，会直接报错，所以在执行命令的时候应该是：

mvn clean deploy -P release -Dgpg.passphrase=OSS的密码

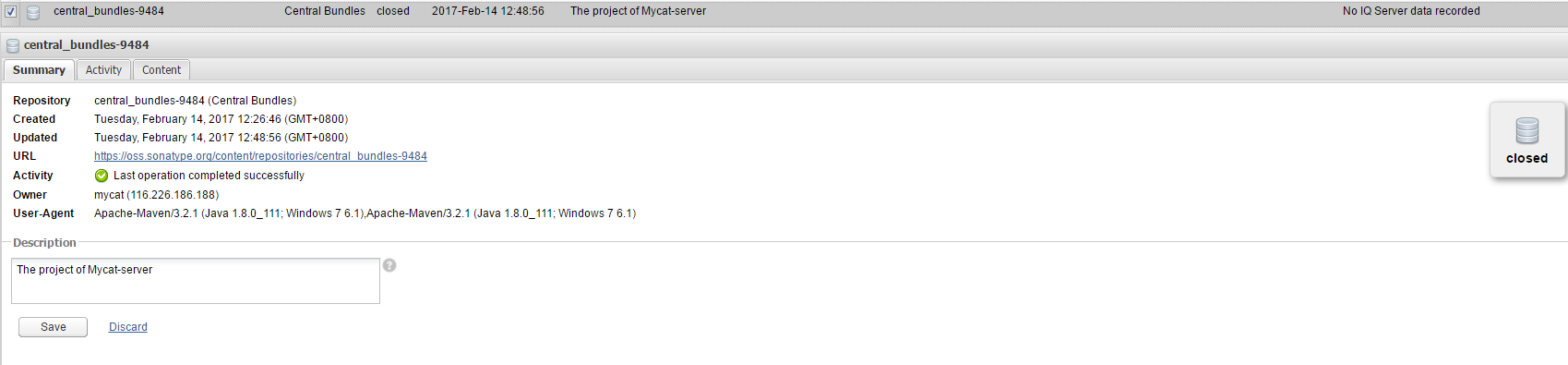
注意：此时上传的构件并未正式发布到中央仓库中，只是部署到 OSS 中了，下面才是真正的发布。

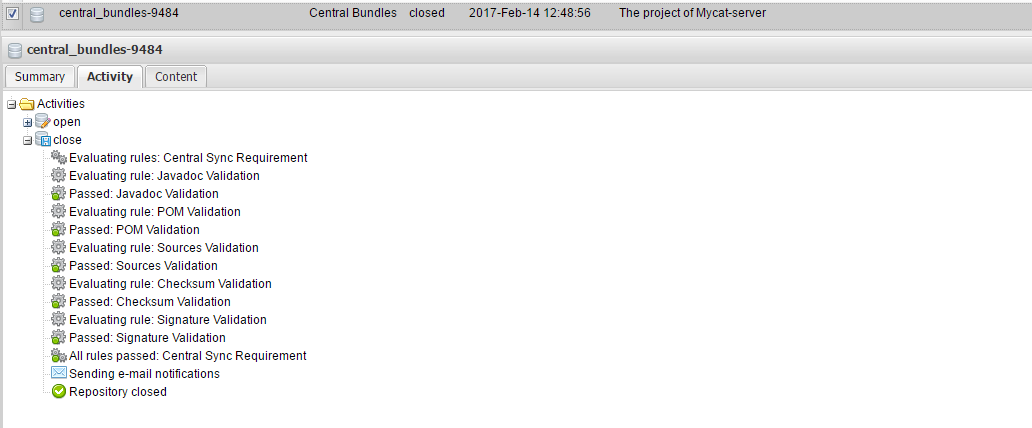
在执行第七部的时候并没有在OSS中的Staging Repositories看到mycat，但是在repositories里面可以看到，所以这一步还需要再研究下



## 在 OSS 中发布构件

在 OSS 中，使用自己的 Sonatype 账号登录后，可在 Staging Repositories 中查看刚才已上传的构件，这些构件目前是放在 Staging 仓库中，可进行模糊查询，快速定位到自己的构件。此时，该构件的状态为 Open，需要勾选它，然后点击 Close 按钮。接下来系统会自动验证该构件是否满足指定要求，当验证完毕后，状态会变为 Closed





## 发布RELEASE版本完成

