**SonarQube代码扫描两种使用方式详解**

通常我们通过Jenkins结合sonar-scanner插件，在构建时进行代码扫描，并将结果反馈到sonarQube平台。但是在实际使用过程中，存在Jenkins服务器与数据库服务器网络策略不通，导致单元测试执行失败，sonarQube无法获取单元测试覆盖率的扫描结果，那么遇到这种情况我们该怎么办呢？

既然Jenkins可以结合sonar-scanner插件，那么我们在本地是否可以单独使用sonar-scanner插件呢？答案是可以的。

下面我分别给大家介绍这两种使用方式：

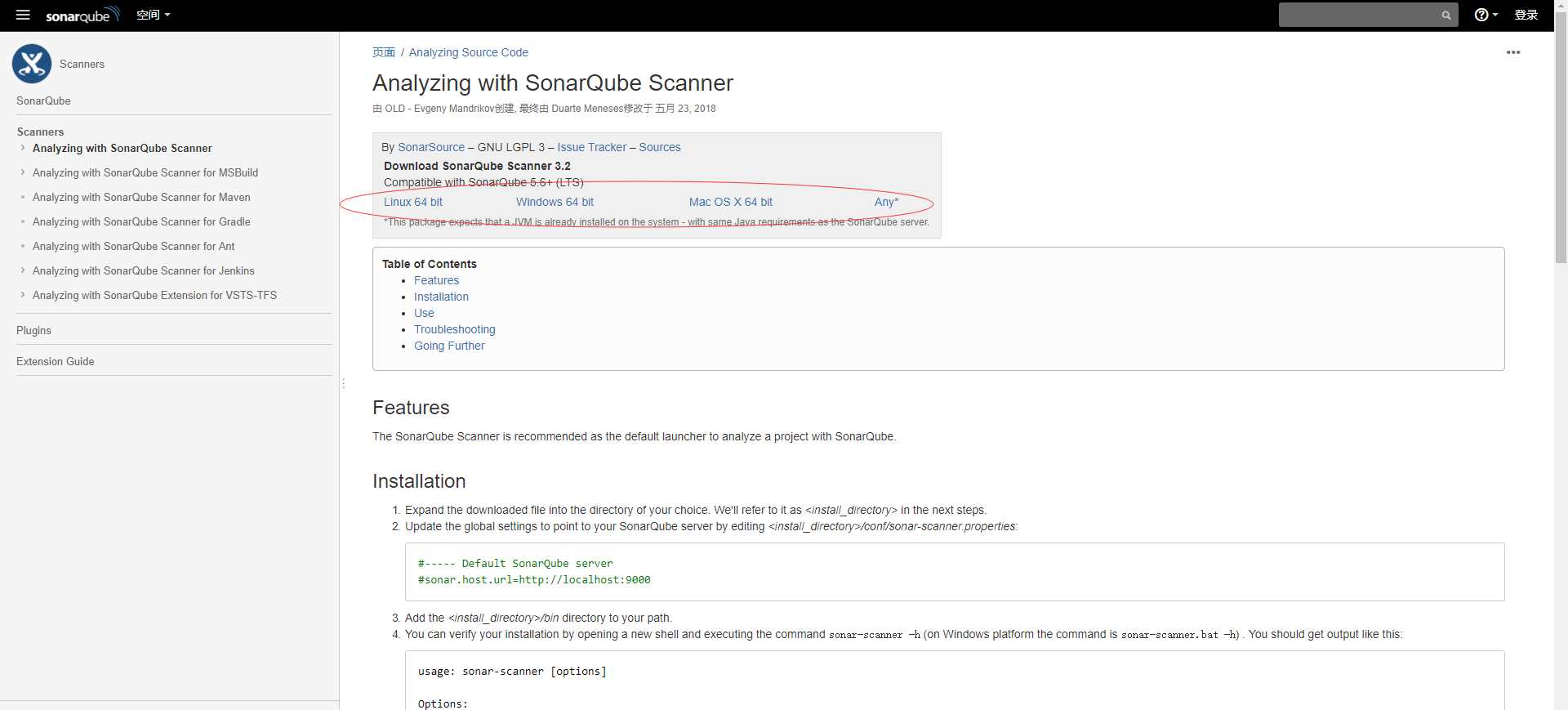
# 通过sonar-scanner插件本地扫描项目

1. 第一步：需要下载sonar-scanner插件

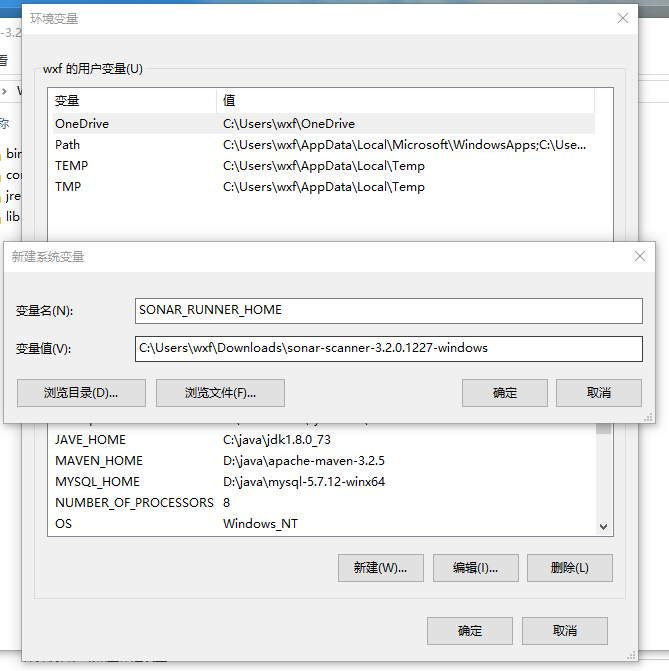
登陆官网地址：

<https://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner>

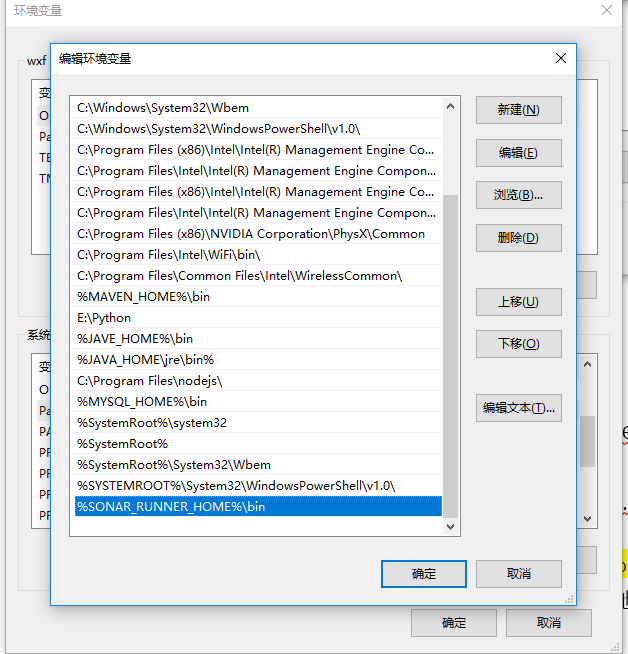
如下图，选择兼容sonarQube5.6版本的进行下载



1. 第二步：配置系统环境变量

在系统环境中配置 `SONAR\_RUNNER\_HOME`为插件安装路径，如下图。

将变量添加到path中去，如下图



1. 第三步：修改sonar-scanner.propertie文件

打开sonar-sacnner目录conf文件夹下的sonar-scanner.properties文档，如下图



根据文档中的提示信息配置sonar平台的地址，并将注释去掉，如下配置：

#Configure here general information about the environment, such as SonarQube server connection details for example

#No information about specific project should appear here

#----- Default SonarQube server

sonar.host.url= http://192.168.100.9:900

#----- Default source code encoding

sonar.sourceEncoding=UTF-8

1. 第四步：修改本地maven项目的pom文件

需要在 maven pom文件添加cobertura插件

cobertura-maven-plugin插件提供单元测试整体覆盖率以及分支覆盖率的统计。Maven项目我们使用它来统计单元测试覆盖率。

首先修改工程的pom.xml文件，添加cobertura-maven-plugin插件 如下

<build>

<plugins>

…

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>cobertura-maven-plugin</artifactId>

<version>2.6</version>

<configuration>

<formats>

<format>xml</format>

</formats>

</configuration>

</plugin>

…

</plugins>

…

</build>

还可以配置不需要单元测试的类或文件夹，如下配置：

exclude内也可以配置单独的类如：com/company/demo/entity/User.class 。配置的这些路径下的文件将不参与单元测试，也不录入单元测试统计数据内。

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>cobertura-maven-plugin</artifactId>

<version>2.6</version>

<configuration>

<formats>

<format>xml</format>

</formats>

<instrumentation>

<excludes>

<exclude>com/company/demo/entity/\*</exclude>

<exclude>com/company/demo/utils/\*</exclude>

</excludes>

</instrumentation>

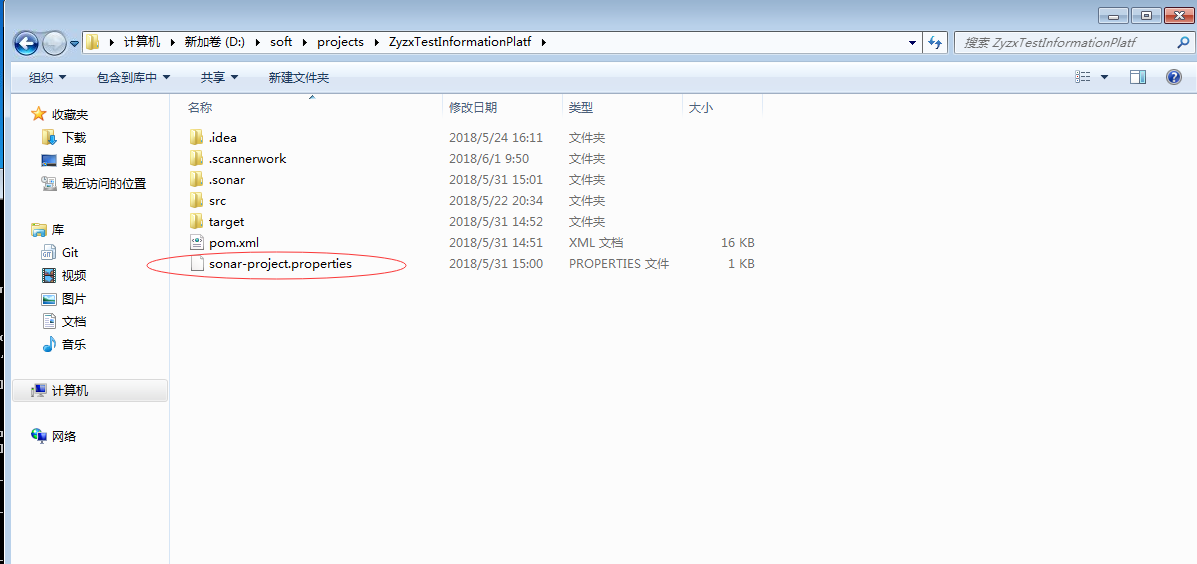
</configuration>

</plugin>

1. 第五步：在项目中添加sonar-project.properties文件

### 在项目根目录添加sonar-project.properties文件，文件名不能更改。

如下图



### 5.2 添加sonar-project.properties文件配置信息

一般项目配置信息

sonar.projectKey=com:testInfo //必须sonar中的唯一

sonar.projectName=com:testInfo //项目名称

sonar.projectVersion=1.0 //项目版本

sonar.sources=D://crmpfcore//src //sonar扫描目录

sonar.sourceEncoding=UTF-8

sonar.language=java //语言,语言分为js、ts、web、java等

soanr.cobertura.reportPaths=cobertura.xml //cobertura.xml路径,获取单元测试覆盖率

多模块项目配置信息

sonar.projectKey=com:testInfo //必须sonar中的唯一

sonar.projectName=com:testInfo //项目名称

sonar.projectVersion=1.0 //项目版本

sonar.sources=src //sonar扫描目录

sonar.sourceEncoding=UTF-8

sonar.language=java

soanr.cobertura.reportPaths=cobertura.xml //cobertura.xml路径,获取单元测试覆盖率

sonar.modules=Module1,Module2,Module3,… //pom中全部模块名称

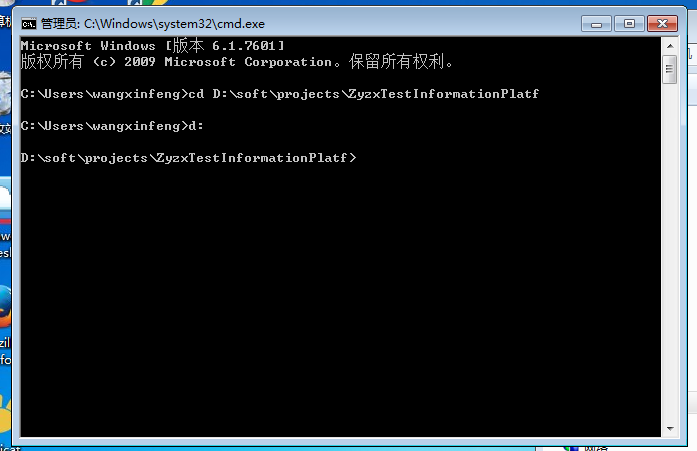
Module1.sonar.projectName=module 1 //sonar中每个模块名称

Module2.sonar.projectName=module 2

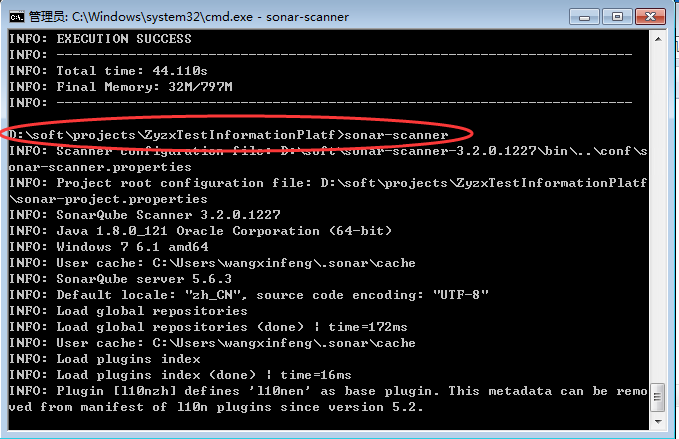
…

…

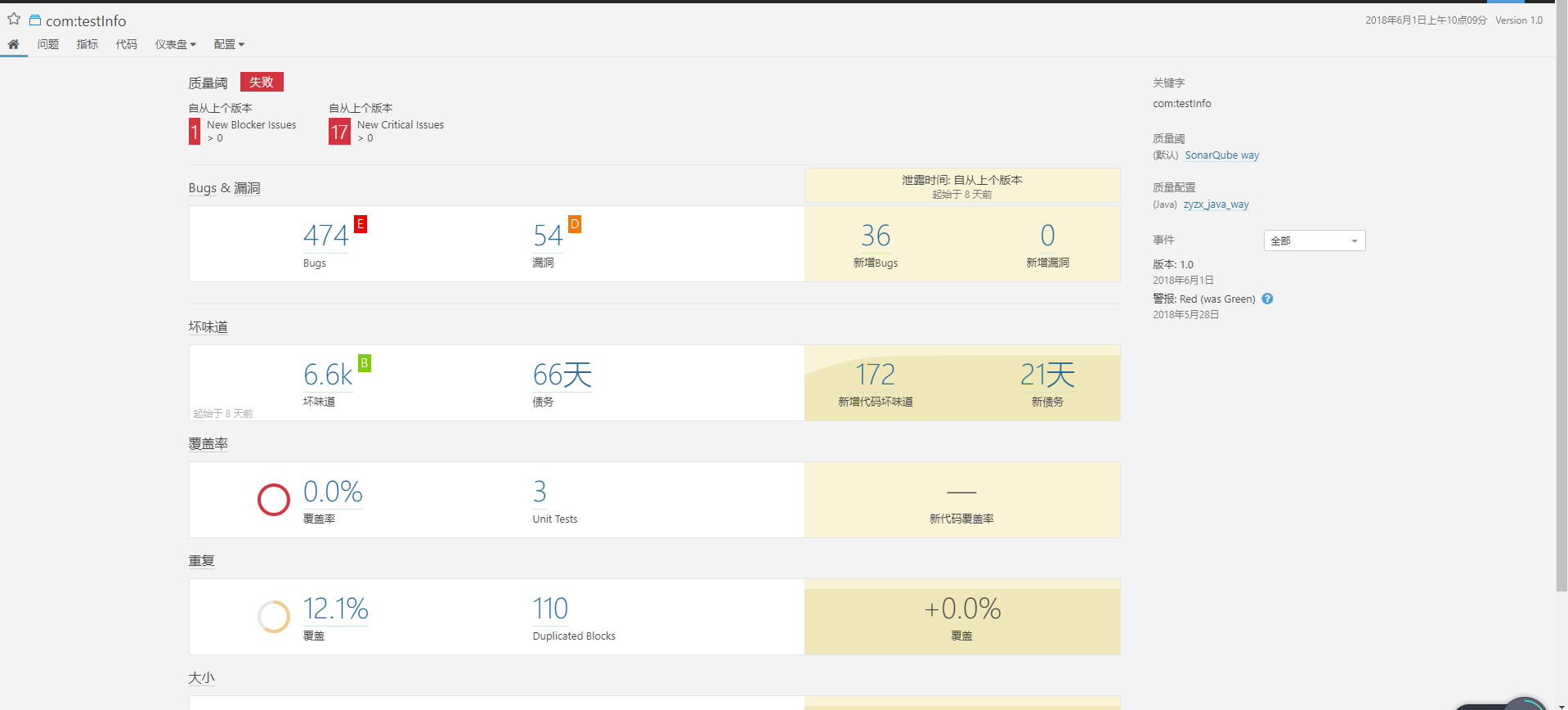
1. 第六步：在项目目录下运行命令
   1. 打开cmd，进行项目目录下



* 1. 确认sonar平台已启动，在运行sonar-runner或sonar-scanner命令



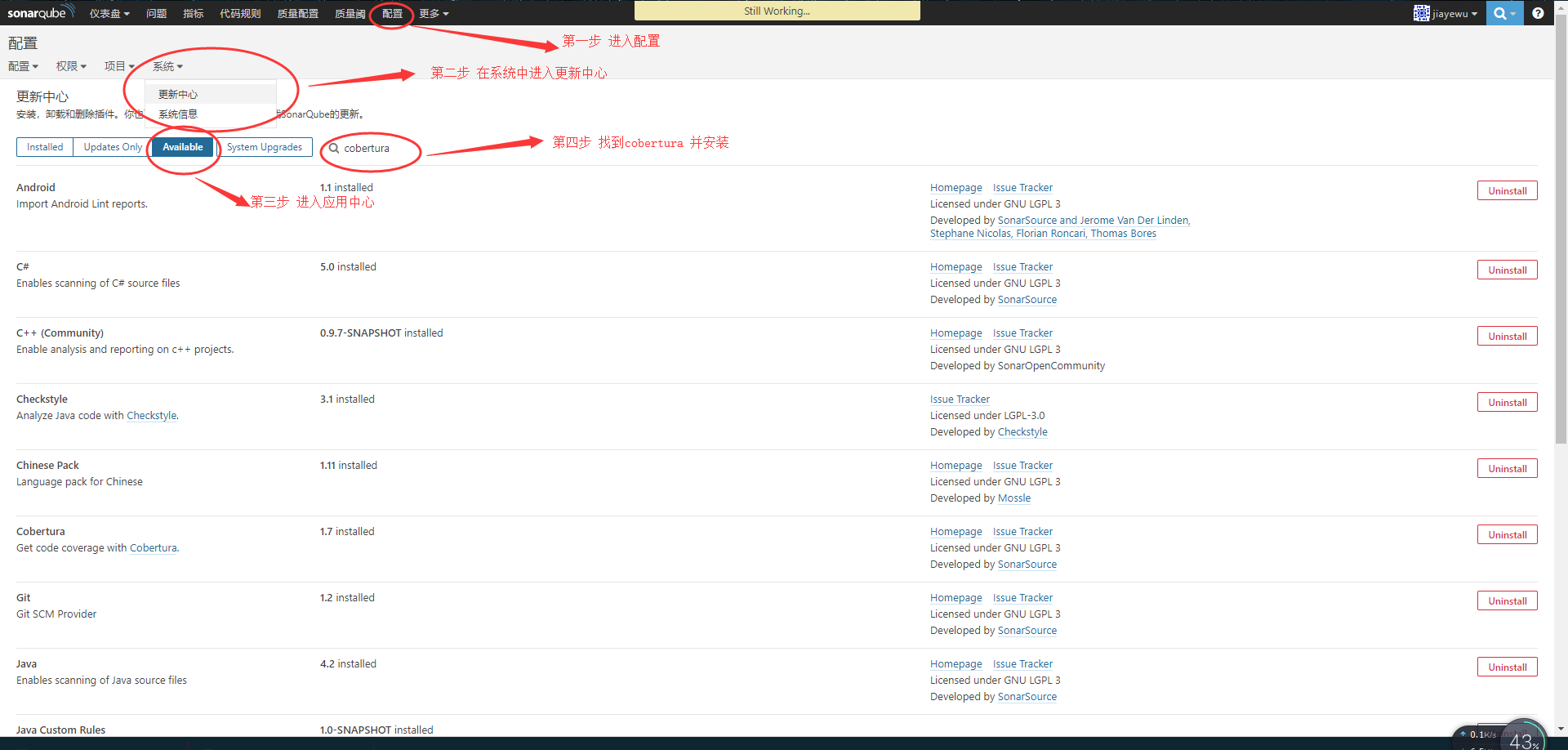
* 1. 等待命令运行完成，结果自动会反馈到sonarQube平台。



# 通过Jenkins的sonar-scanner插件扫描项目

1. 第一步： 在SonarQube平台安装及配置cobertura插件

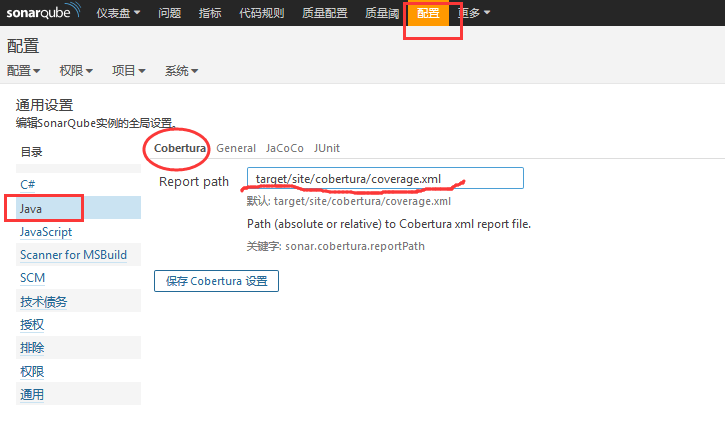
管理员在sonar页面选择**配置**进入配置页面，选择**系统**-**更新中心**->“**Avaliable**”即可看到可安装的插件（联网情况下），选择**Cobertura**安装即可。安装后可以查看**系统**-**更新中心**->“**Installed**”查看已经安装的插件。



安装完插件找到**配置-Java- Cobertura配置**：

Report path：target/site/cobertura/coverage.xml （单元测试覆盖率生成的结果路径）

如下图：



1. 第二步： 在Maven项目pom文件中添加cobertura插件

cobertura-maven-plugin插件提供单元测试整体覆盖率以及分支覆盖率的统计。Maven项目我们使用它来统计单元测试覆盖率。

首先修改工程的pom.xml文件，添加cobertura-maven-plugin插件 如下

<build>

<plugins>

…

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>cobertura-maven-plugin</artifactId>

<version>2.6</version>

<configuration>

<formats>

<format>xml</format>

</formats>

</configuration>

</plugin>

…

</plugins>

…

</build>

配置不需要单元测试的类或文件夹

exclude内也可以配置单独的类如：com/company/demo/entity/User.class 。配置的这些路径下的文件将不参与单元测试，也不录入单元测试统计数据内。

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>cobertura-maven-plugin</artifactId>

<version>2.6</version>

<configuration>

<formats>

<format>xml</format>

</formats>

<instrumentation>

<excludes>

<exclude>com/company/demo/entity/\*</exclude>

<exclude>com/company/demo/utils/\*</exclude>

</excludes>

</instrumentation>

</configuration>

</plugin>

1. 第三步： 在Jenkins中修改项目配置

需要在Jenkins项目配置中分别配置Build、Execute SonarQube Scanner。

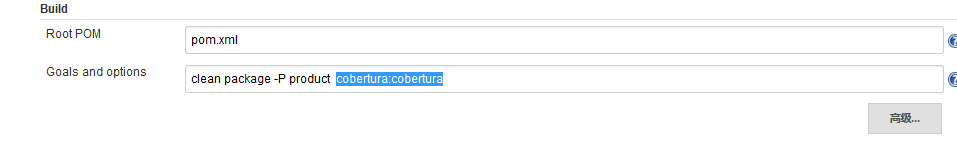
如果没有Execute SonarQube Scanner，需要去插件管理中下载，并在系统设置中添加soanrQube平台路径，这里就不做详细介绍。

* 1. Build

首先进入项目配置中，然后找到Build

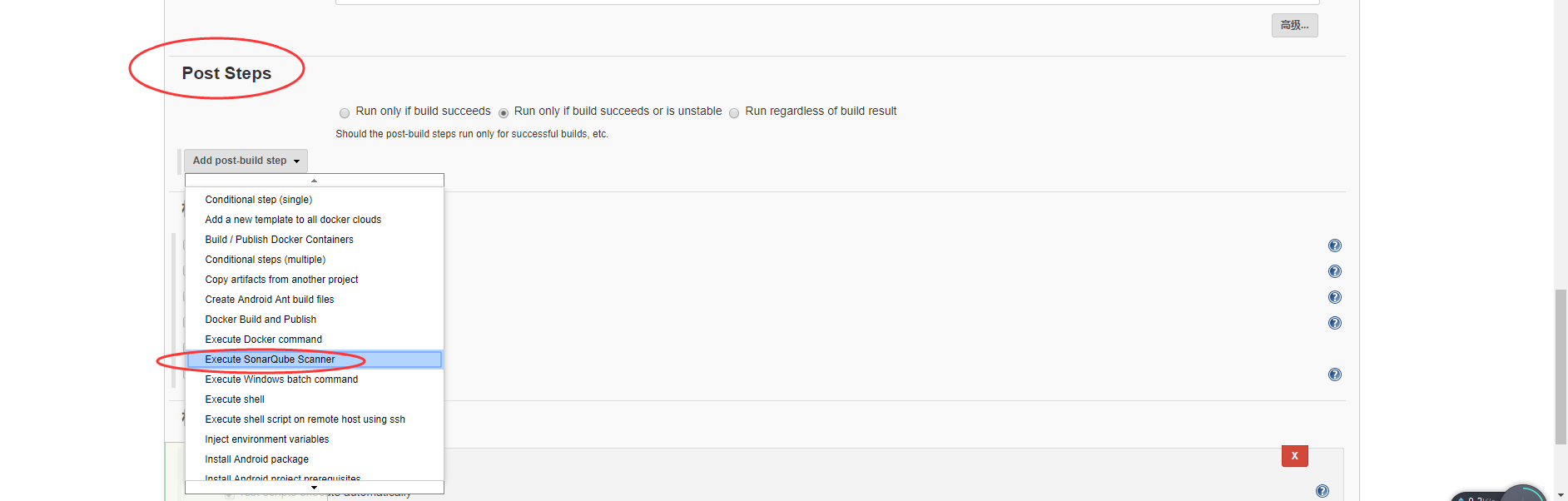
为了生成单元测试覆盖率，需要使用clean package –PprodTest cobertura:cobertura

，及在命令后加上cobertura:cobertura

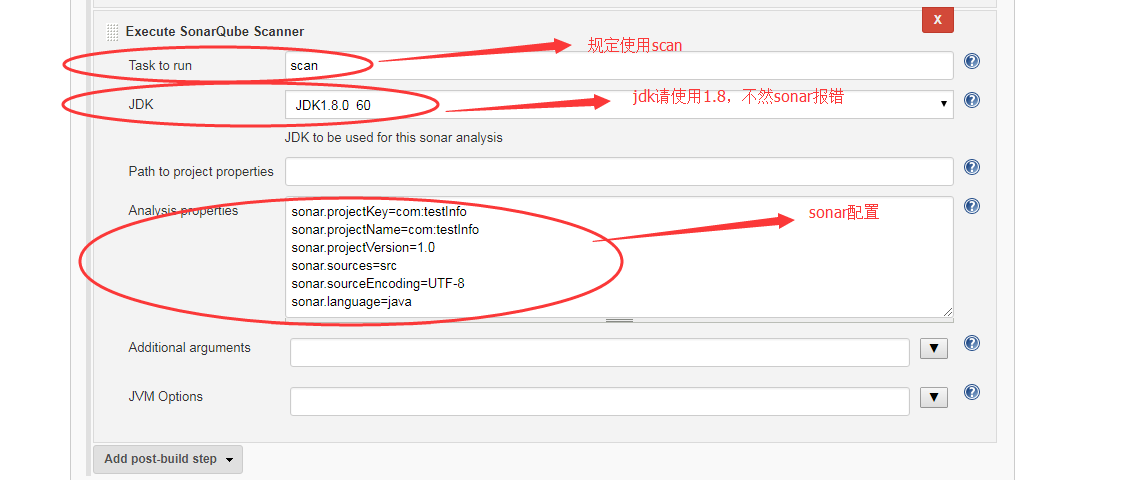


* 1. Execute SonarQube Scanner

然后再 Post Steps / 构建 配置模块中, 选择 Execute SonarQube Scanner，如下图



下图为Execute SonarQube Scanner的配置项



配置项介绍：

Tack to tun //规定使用scan

Jdk //必须使用1.8，不然会报错

Analysis properties //sonar配置单模块和多模块 详细介绍如下：

Soanr 单模块配置

Sonar.projectKey=XXX //对应sonar项目中的Key

Sonar.projectName=XXX //对应sonar项目中的Name

Sonar.projectVersion=1.0 //版本，根据项目填写

Sonar.sources=src //检测src下所有文件，‘.’表示检测整个项目

Sonar.sourceEncoding=UTF-8 //字符编码

Sonar.language=java //检测的语言,语言分为js、ts、web、java等

//js表示.js、.jsx、.vue结尾的文件

//ts表示.ts、.tsx结尾的文件

//web表示.html、.xhtml、.rhtml、.shtml结尾的文件

//java表示.java结尾的文件

Soanr 多模块配置

sonar.projectKey=com.cmos:crmpfcore //必须sonar中的唯一

sonar.projectName=crmpfcore //项目名称

sonar.projectVersion=1.0 //项目版本

sonar.sources=src //sonar扫描目录

sonar.sourceEncoding=UTF-8

sonar.language=java

soanr.cobertura.reportPaths=cobertura.xml //cobertura.xml路径,获取单元测试覆盖率

sonar.modules=Module1,Module2,Module3,… //pom中全部模块名称

Module1.sonar.projectName=module 1 //sonar中每个模块名称

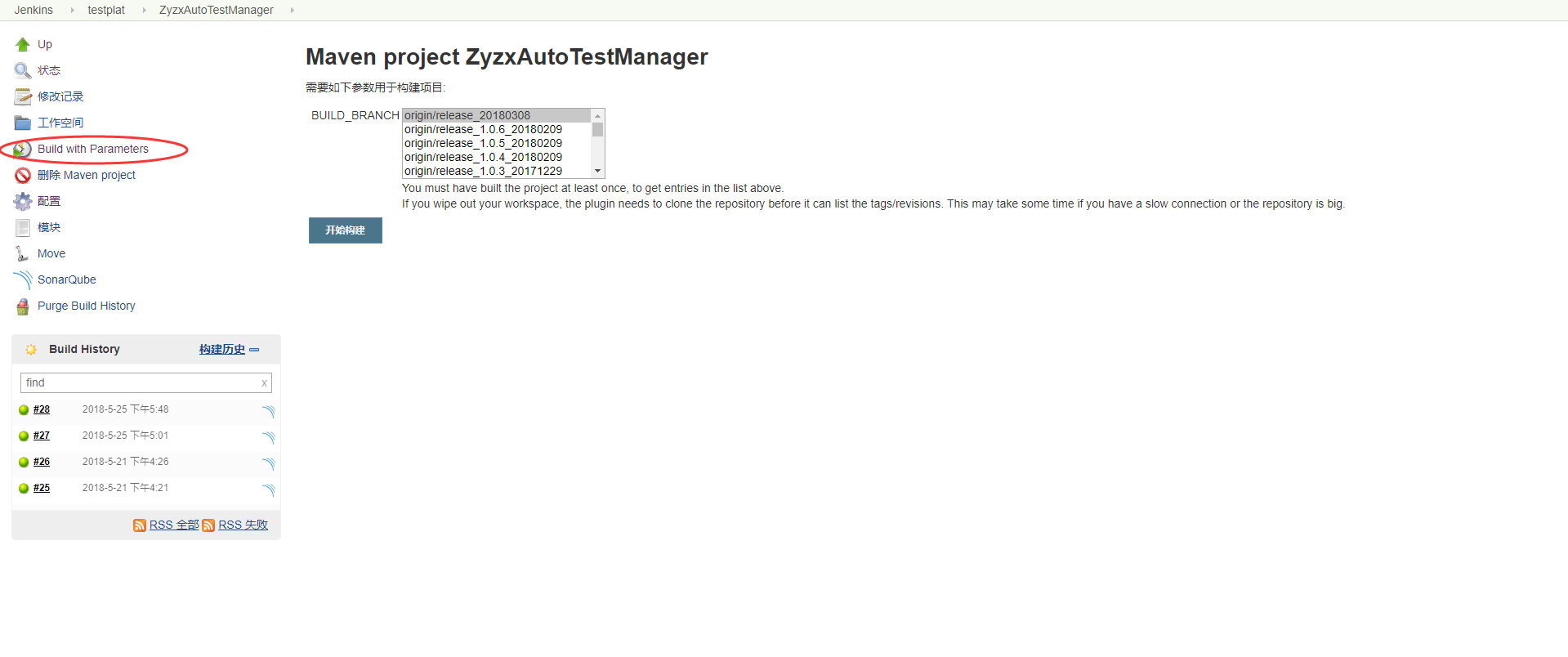
Module2.sonar.projectName=module 2

..

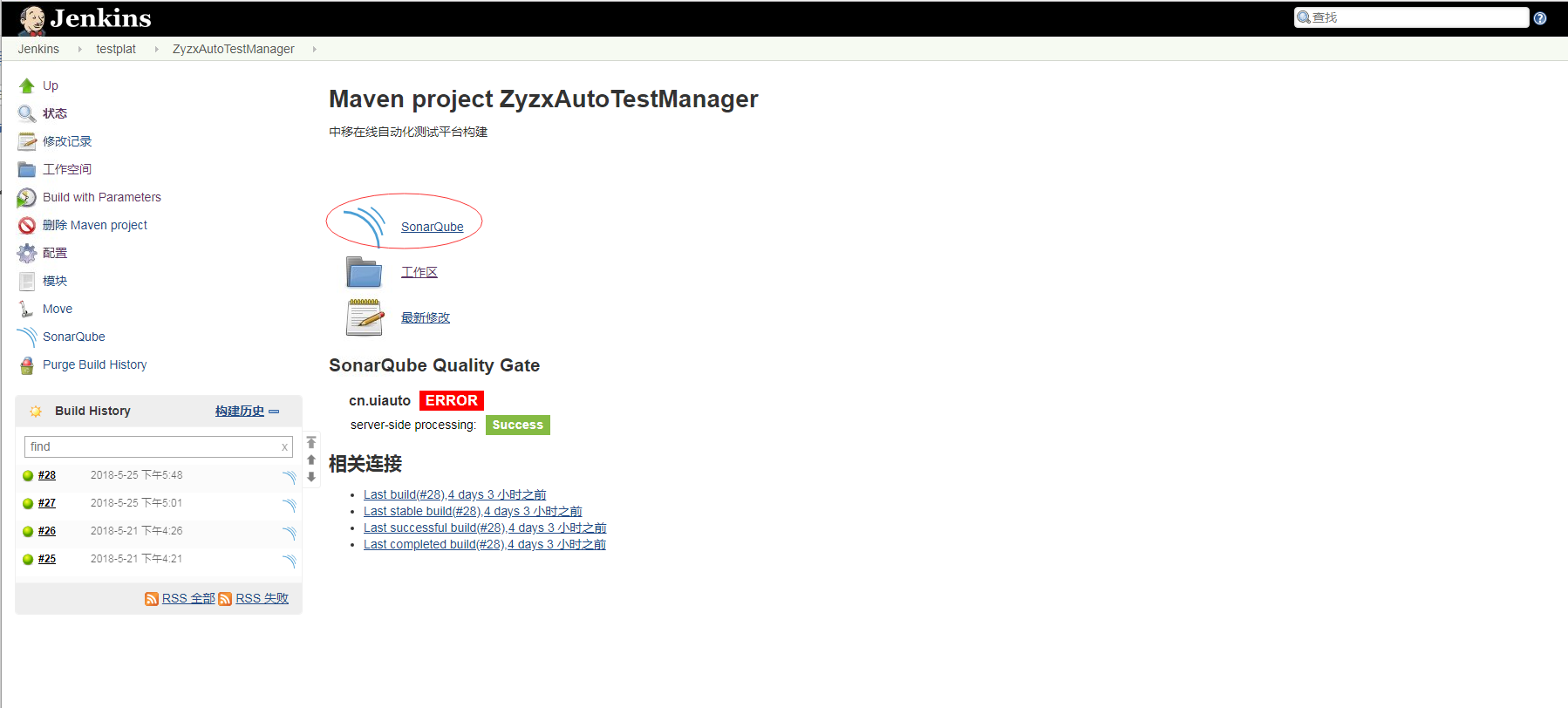
..

1. 第四步： Jenkins项目构建

在[Build with Parameters](http://192.168.100.10:8080/jenkins/job/testplat/job/ZyzxAutoTestManager/build?delay=0sec)中选择分支进行构建，如下图



项目构建完以后会出现SonarQube图标，如下图，通过该图标就可以跳转到SonarQube平台，看到检测结果。



# 展望

SonarQube代码扫描的两种使用方式到此就介绍结束了，现在我们可以通过SonarQube平台清楚得看到我们哪行代码存在bug，或者书写不规范。但是soanr自带得规则并不能满足所有项目的需求，如果我们想增加一条新的扫描规则该怎么办呢？

下期我会给大家介绍soanr如何新增自定义规则！