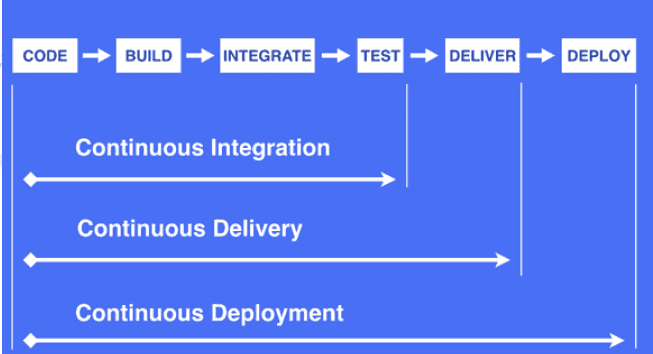
# Jenkins+Sonar

## CI/CD介绍

### 1.1.1软件开发发布流程

互联网软件的开发和发布，已经形成了一套标准流程，假如把开发工作流程分为以下几个阶段：

编码 --> 构建 --> 集成 --> 测试 --> 交付 --> 部署



正如你在上图中看到，

### 1.1.2持续集成

**持续集成(Continuous Integration)**：就是开发提交了新代码之后，立刻进行构建、（单元）测试。根据测试结果，测试过程可以确定新代码和原有代码能否正确地集成在一起，如果测试过程有问题，那么将返回代码给开发部门

### 1.1.3持续交付

**持续交付(Continuous Delivery)**：在持续集成的基础上，将集成后的代码部署到更贴近于真实的生产环境中。在贴近真实的生成环境中，可以提高数据支撑，且进一步做测试，如果进一步测试没有问题，那么可以部署到线上的生成环境中

### 1.1.4持续部署

**持续部署(Continuous Deployment)**：是在持续交付的基础上，把部署到生产环境的过程自动化，实现部署无需人工参与，如果在部署的过程中出现问题，将立即回滚代码

## Jenkins介绍

### 概念

Jenkins是一个功能强大，扩展性很强的应用程序，允许持续集成和持续交付项目，无论用的是什么平台。这是一个免费的源代码，可以处理任何类型的构建或持续集成。集成Jenkins可以用于一些测试和部署技术。Jenkins是一种软件允许持续集成。

### 目的

1、持续、自动、手动地构建/测试，发布软件项目。   
2、静态代码分析(sonar)。

### 功能/特性

1. 开源的java语言开发持续集成工具，支持CI，CD。

2. 静态代码分析(sonar)，可以检测出漏洞，bug，坏味道，编译程度等统计型报表。

3. 定时打包发布测试版(自动化构建发布测试版)。

4. 自定义操作，如跑自动化单元测试，测试用例等。

5. 定时拉取代码并编译

6. 易于安装部署配置：可通过yum安装，或ZIP包解压安装，或下载war包，以及通过tomcat/docker等容器等快速实现安装部署，可方便web界面配置管理。

7. 消息通知及测试报告：集成RSS/E-mail通过RSS发布构建结果或当构建完成时通过e-mail通知，生成JUnit/TestNG测试报告。

8. 分布式构建：支持Jenkins能够让多台计算机一起构建/测试。

9. 丰富的插件支持:支持扩展插件，你可以开发适合自己团队使用的工具，如git，svn，maven，docker等。

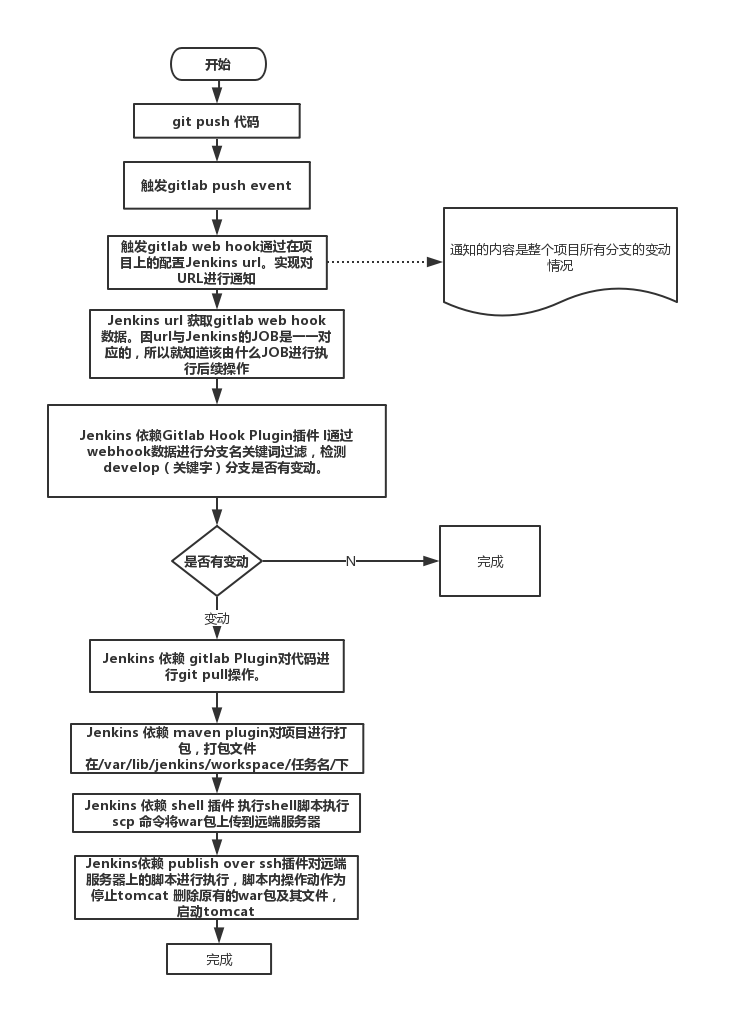
### 工作原理

jenkins的工作原理是先将源代码从github中拷贝一份到本地，然后根据设置的脚本进行build。

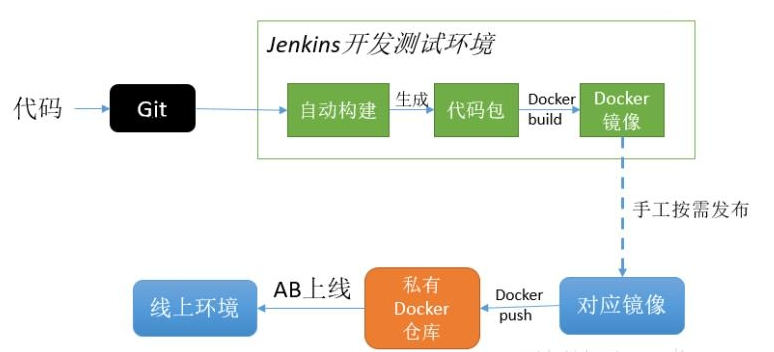
整个系统的关键就是build脚本，用来告诉jenkins在一次集成中需要执行的任务。

代码提交至git，通过jenkins进行配置，将git代码提交至服务器，并且发布到指定的文件夹，jenkins有触发器配置，每次有新代码提交，git上代码会自动上传到服务器并自动更新发布。

**Jenkins的 job与远程部署到tomcat的工作原理**  
①定时拉取代码并编译或者自定义触发器hook拉取代码  
②通过maven命令打包到本地临时目录  
③执行shell命令上传到远程服务器  
④执行shell脚本停止远程服务，bing删除原有war包，并重新启动tomcat



**Jenkins+Docker原理**  
①从代码仓库拉取最新的代码。  
②Jenkins主机打包编译代码。  
③Jenkins主机使用Dockerfile生成最新的镜像，并push到仓库。  
④Docker主机pull最新的镜像，删除旧的容器，并生成最新的容器，完成部署上线。



## 1.3****环境准备****

操作系统：Centos 7.2

Jdk1.8

Git2.9.5

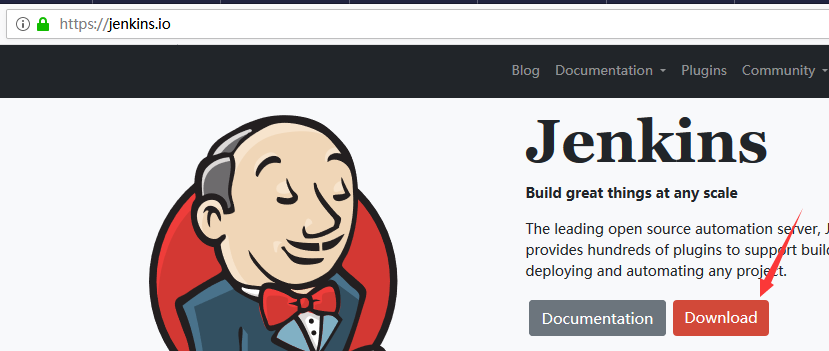
Maven3.6.1

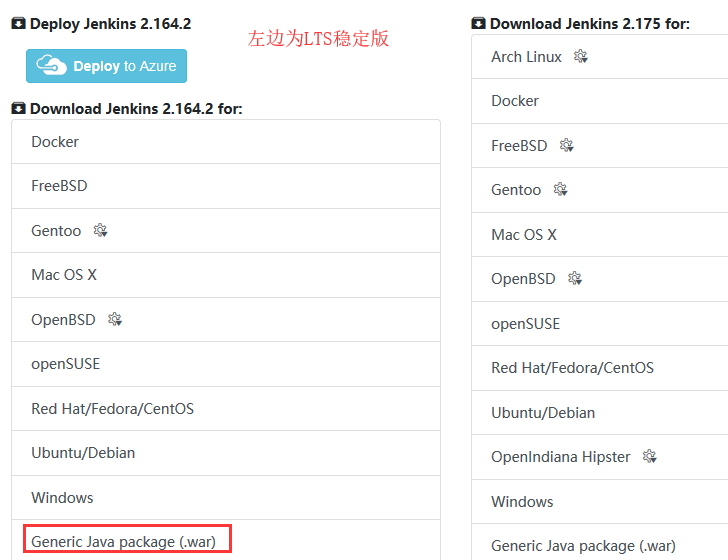
并配置好对应环境变量

## 1.4 ****Jenkins安装****

### 1.4.1 ****Java -jar安装****

1. 从Jenkins官网下载jenkins.war文件。官网地址:http://jenkins-ci.org/，注意选择最新版本的Long-Term Support Release



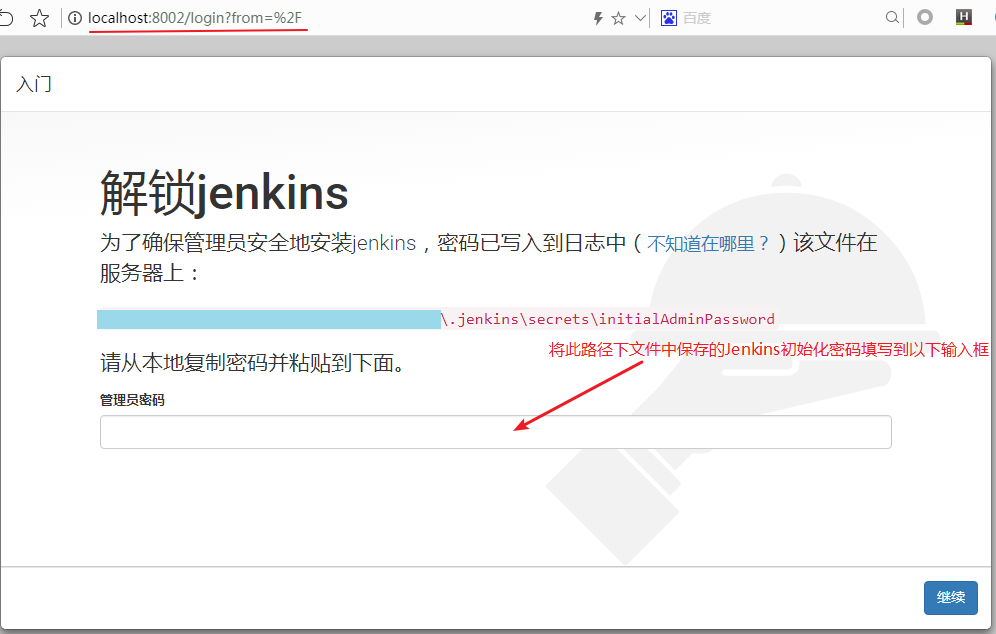


1. 运行 java -jar jenkins.war（可添加命令 --httpPort=$HTTP\_PORT，用来设置jenkins运行时的web端口）；
2. 后台运行：nohup java -jar jenkins.war &
3. 注意：Jenkins 最新war包需要运行 Java 8以及以上的版本。

### 1.4.2 ****servlet 安装****

1.从Jenkins官网下载jenkins.war文件。官网地址:http://jenkins-ci.org/，注意选择最新版本的Long-Term Support Release

2. 将下载的war包文件部署servlet容器，然后启动容器，在浏览器的URL地址栏中输入类似[http://localhost:8080/jenkins/](http://localhost:8088/hudson/)这样的地址即可



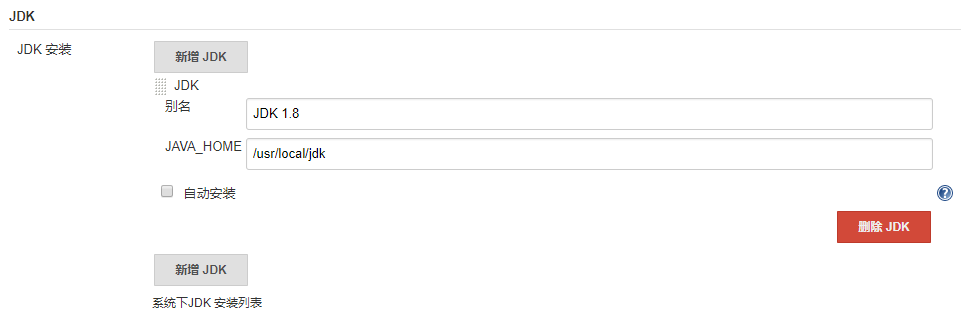
默认密码位置：C:\Users\THINK\.jenkins\secrets\initialAdminPassword

## 1.5 ****Jenkins配置****

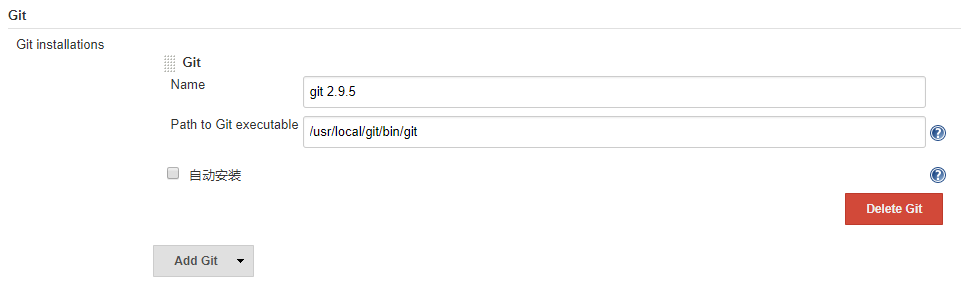
### 1.5.1 系统管理-全局工具配置

输入提示路径下的密码，安装默认的插件，设置管理员用户，确定jenkins根路径

【系统管理】->【全局工具配置】：









### 1.5.2 系统管理-插件管理

【系统管理】->【插件管理】：

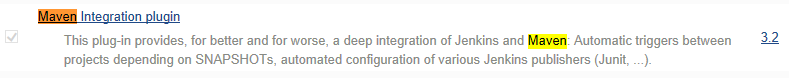
【Safe Restart】　　//重启jenkins系统



【Email Extension】　　//由于Jenkins自带的邮件功能比较鸡肋，因此推荐安装专门的邮件插件



【maven integration】　　//该插件安装了，才能创建maven项目



【git plugin】　　//从远程拉取代码



【publish over ssh】　　//ssh远程登录

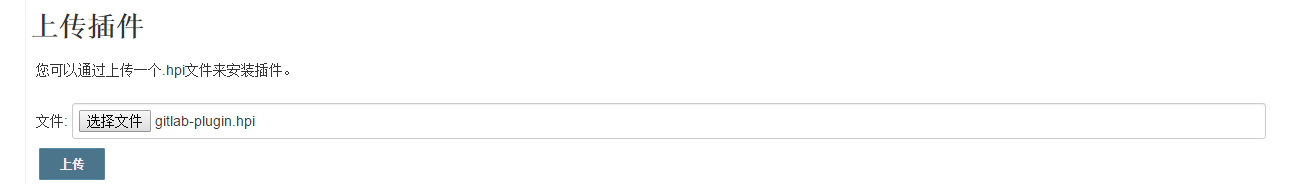


【SSH Plugin】　　//SSH登录到一个远程服务器以执行必要的脚本



手动下载jenkins插件插件下载地址：

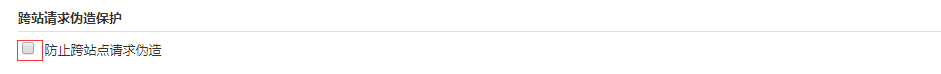
<http://updates.jenkins-ci.org/download/plugins/>

如果有插件一直安装失败的情况，可以从jenkins官网下载插件，然后导入到jenkins。示例：我从官方下载gitlab-plugin插件到本地，然后导入到jenkins中。在jenkins插件管理->高级选择上传插件进行安装。

### 1.5.3 系统管理-全局安全配置

【系统管理】->【全局安全配置】：



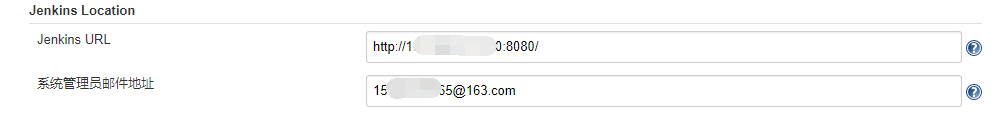


在jenkins全局在jenkins全局安全设置中取消勾选“防止跨站点请求伪造，不然jenkins会认为gitlab的请求不合法。

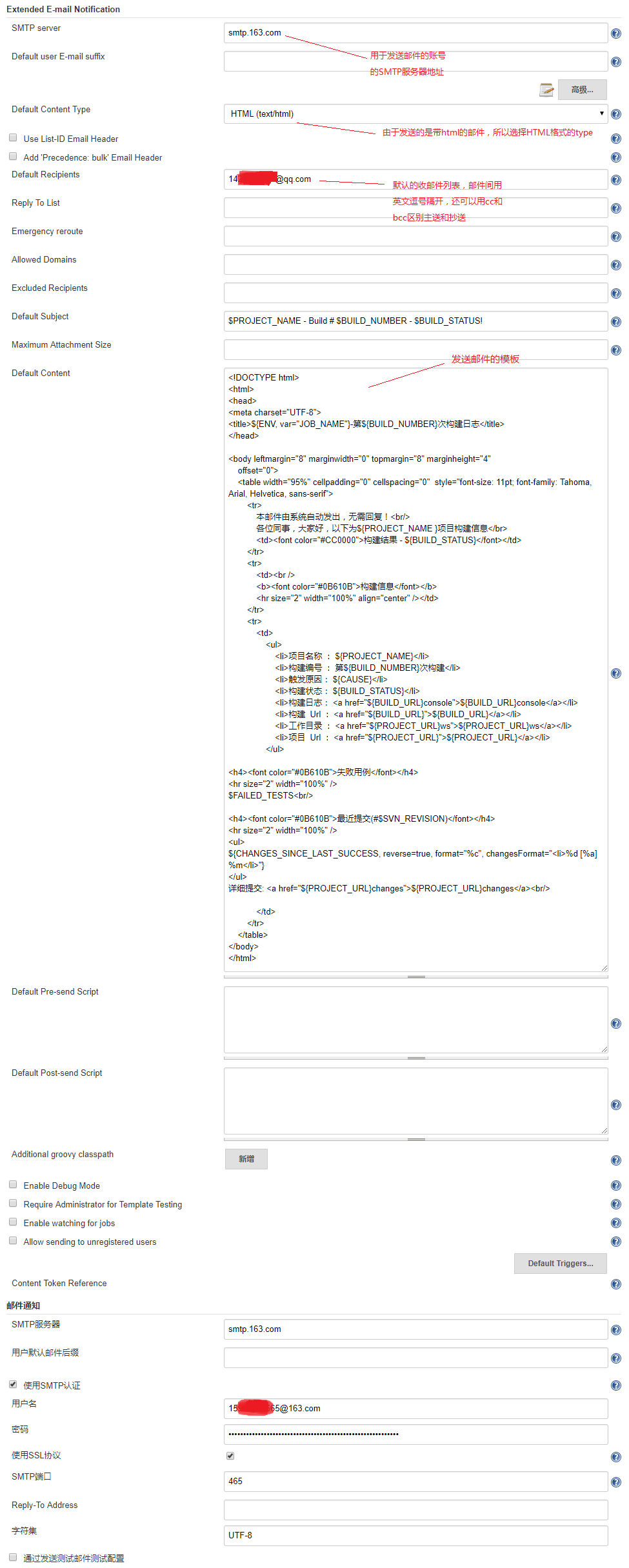
### 1.5.4 系统管理-系统设置

【系统管理】->【系统设置】：

【URL和管理员邮箱设置】:



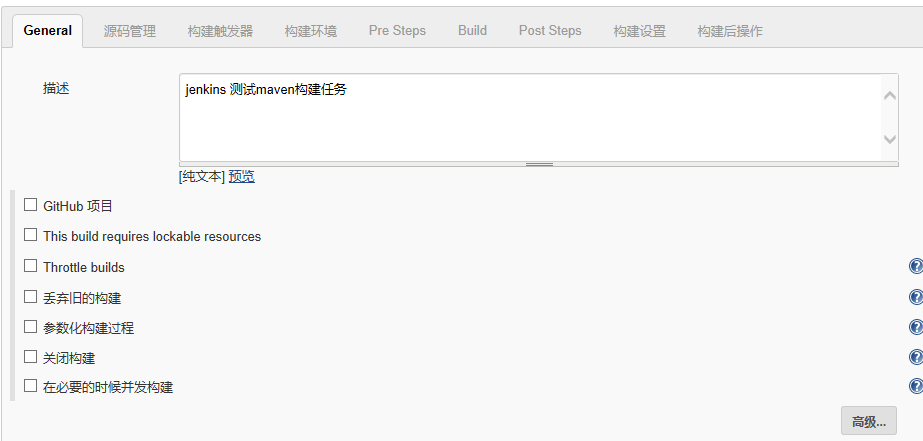
【邮件设置】：

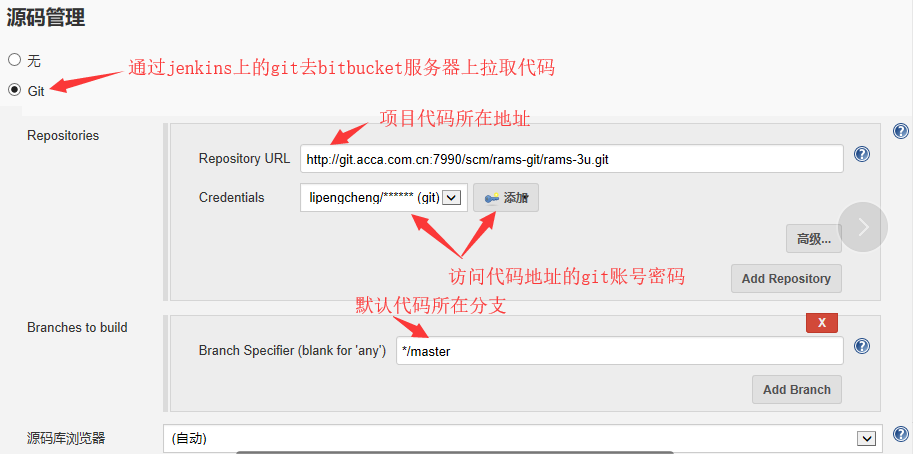


### 1.5.5 创建构建任务

【新建任务】：

　　任务名称->构建maven任务->确认







### 1.5.6 手动触发构建

【jenkins主页面】->【相应的构建任务】->【立即构建】





## 1.6.Sonar 介绍

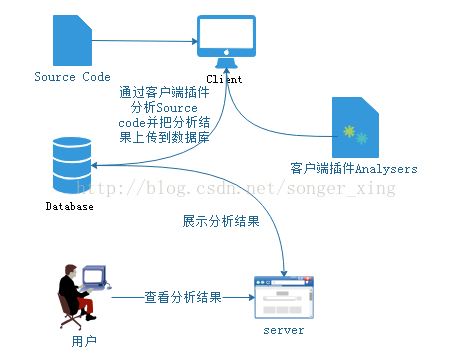
Sonar是一个用于代码质量管理的开源平台，用于管理Java源代码的质量。通过插件机制，Sonar 可以集成不同的测试工具，代码分析工具，以及持续集成工具，比如pmd-cpd、checkstyle、findbugs、Jenkins。通过不同的插件对这些结果进行再加工处理，通过量化的方式度量代码质量的变化，从而可以方便地对不同规模和种类的工程进行代码质量管理。

同时 Sonar 还对大量的持续集成工具提供了接口支持，可以很方便地在持续集成中使用 Sonar。

此外，Sonar 的插件还可以对 Java 以外的其他编程语言提供支持，对国际化以及报告文档化也有良好的支持。

通过客户端插件分析源代码，sonar客户端可以采用IDE插件、Sonar-Scanner插件、Ant插件和Maven插件方式，并通过各种不同的分析机制对项目源代码进行分析和扫描，并把分析扫描后的结果上传到sonar的数据库，通过sonar web界面对分析结果进行管理

架构图



可以从七个维度检测代码质量:

(1) 复杂度分布(complexity):代码复杂度过高将难以理解

(2) 重复代码(duplications):程序中包含大量复制、粘贴的代码而导致代码臃肿，sonar可以展示源码中重复严重的地方

(3) 单元测试统计(unit tests):统计并展示单元测试覆盖率，开发或测试可以清楚测试代码的覆盖情况

(4) 代码规则检查(coding rules):通过Findbugs,PMD,CheckStyle等检查代码是否符合规范

(5) 注释率(comments):若代码注释过少，特别是人员变动后，其他人接手比较难接手；若过多，又不利于阅读

(6) 潜在的Bug(potential bugs):通过Findbugs,PMD,CheckStyle等检测潜在的bug

(7) 结构与设计(architecture & design):找出循环，展示包与包、类与类之间的依赖、检查程序之间耦合度

### 1.6.1配置数据库

Apache Derby 是Sonar自带并且默认安装使用的数据库，此外Sonar对如下数据库提供支持：MySQL 5.x, Oracle 10g XE, Postgresql, MS SqlServer等，这里以mysql为例介绍如何配置数据库：

1）创建数据库

在mysql中执行如下脚本创建数据库及mysql用户

|  |
| --- |
| CREATE DATABASE sonar CHARACTER SET utf8 COLLATE utf8\_general\_ci;    CREATE USER 'sonar' IDENTIFIED BY 'sonar';  GRANT ALL ON sonar.\* TO 'sonar'@'%' IDENTIFIED BY 'sonar';  GRANT ALL ON sonar.\* TO 'sonar'@'localhost' IDENTIFIED BY 'sonar';  FLUSH PRIVILEGES; |

2）编辑${SONAR\_HOME}/conf/sonar.properties配置数据库:

sonar.jdbc.username:sonar

sonar.jdbc.password:sonar

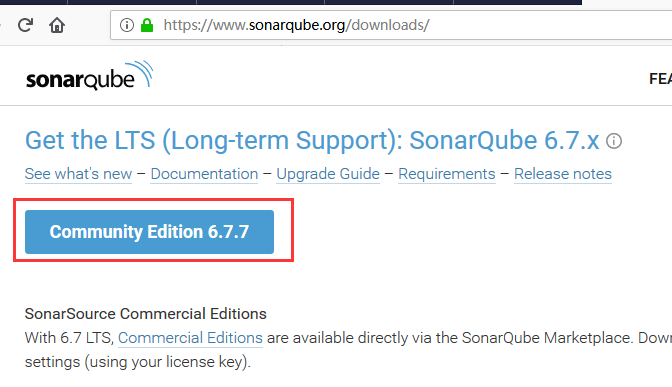
sonar.jdbc.url: jdbc:mysql://localhost:3306/sonar?useUnicode=true&characterEncoding=utf8&rewriteBatchedStatements=true

# Optional properties

sonar.jdbc.driverClassName:com.mysql.jdbc.Driver

### 1.6.2安装配置

Sonar的运行需要 JDK 1.8+ , 从 <http://www.sonarqube.org/downloads/> 下载sonar.zip文件



创建运行sonar的CentOS账户sonar，并设置账户密码：

# useradd sonar

# passwd sonar

$ su sonar

使用sonar账户登录CentOS，如果使用root账号登录会报错。

Sonar默认集成了jetty容器，可以直接启动提供服务，也可以通过脚本构建为war包，部署在tomcat容器中。

1）直接启动

编辑.bash\_profile，添加环境变量SONAR\_HOME

$ vi $HOME/.bash\_profile

修改成如下内容：

PATH=$PATH:$HOME/bin

SONAR\_HOME=$HOME/sonar

export PATH SONAR\_HOME

使环境变量生效

Source $HOME/.bash\_profile

### 1.6.3启动运行

运行如下命令启动sonar，其它操作系统sonar均提供了启动脚本

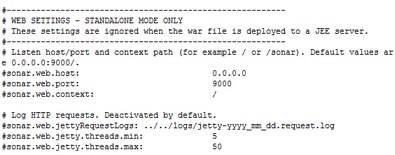
$ ${SONAR\_HOME}/bin/linux-x86-64/sonar.sh start

在浏览器中访问: <http://localhost:9000/> ，运行界面如下：



Sonar默认的端口是”9000”、默认的上下文路径是”/”、默认的网络接口是”0.0.0.0”，默认的管理员帐号和密码为:admin/admin，这些参数都可以在配置文件中修改：

$ vi ${SONAR\_HOME}/conf/sonar.properties



2）作为Web项目,部署到Tomcat等应用服务器中

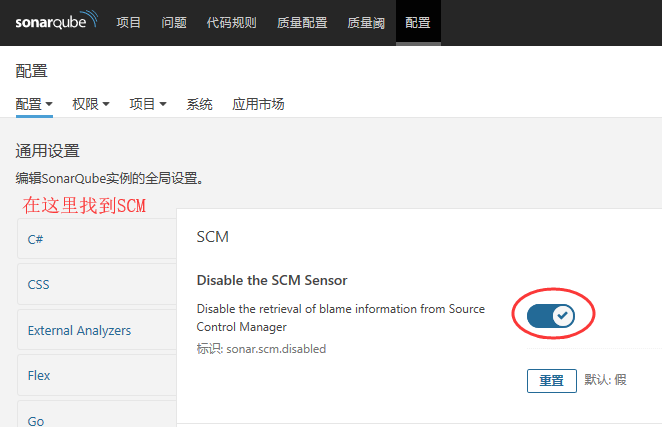
a. 确保conf/sonar.properties、conf/wrapper.conf未被修改使用过

b. 执行如下命令生成war包，将生成的sonar.war部署到应用服务器中

$ ${SONAR\_HOME}/war/build-war.sh

c. 启动Tomcat, 通过 <http://localhost:8080/sonar> 访问.

登陆过后在配置栏里禁用掉SCM，不禁用的话在执行Sonar Scanner完成后，SonarQube Server还会去访问SVN项目，再次要求输入用户名和密码，导致扫描任务失败



可以在质量配置里创建和上传CheckStyle规则检验



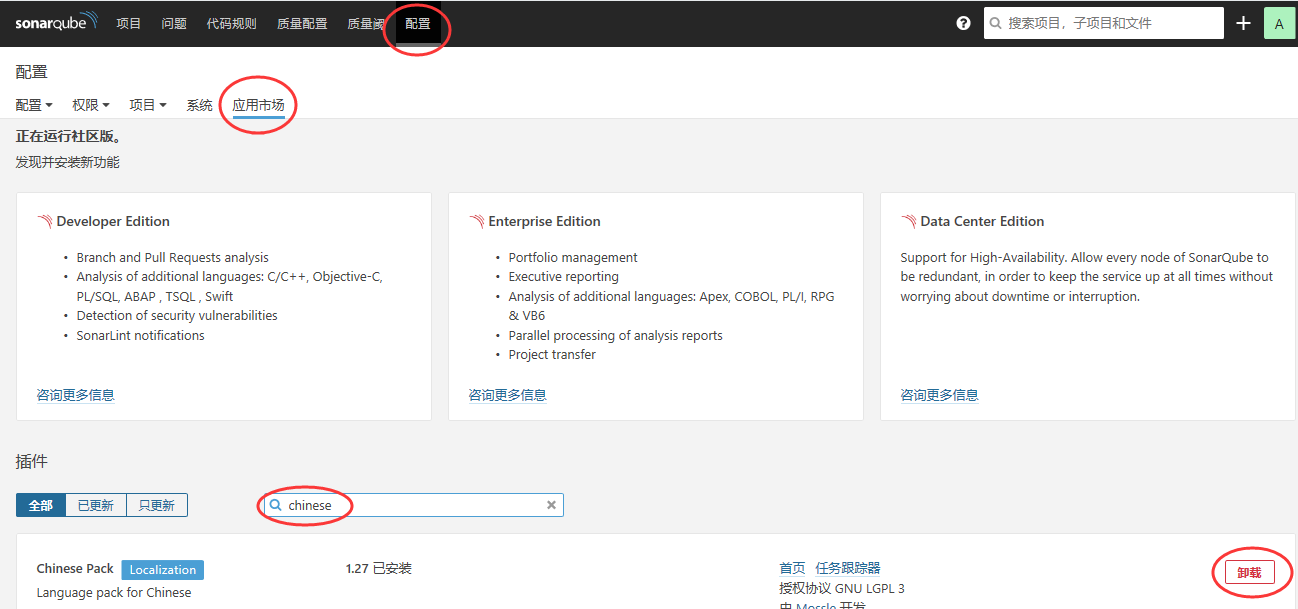


### 1.6.4配置插件

a）插件介绍

Sonar支持多种插件，插件可以通过登录后在配置→应用市场→插件里下载

下图为安装中文语言包插件的步骤，安装过后重启Sonar则可



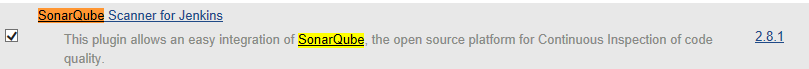
sonar默认集成了[Java Ecosystem](http://docs.codehaus.org/display/SONAR/Java+Ecosystem)插件，该插件是一组插件的合集

* Java [sonar-java-plugin]：java源代码解析，计算指标等
* Squid [sonar-squid-java-plugin]：检查违反Sonar定义规则的代码
* Checkstyle [sonar-checkstyle-plugin]：使用CheckStyle检查违反统一代码编写风格的代码
* FindBugs [sonar-findbugs-plugin]：使用FindBugs检查违反规则的缺陷代码
* PMD [sonar-pmd-plugin]：使用pmd检查违反规则的代码
* Surefire [sonar-surefire-plugin]：使用Surefire执行单元测试
* Cobertura [sonar-cobertura-plugin]：使用Cobertura获取代码覆盖率
* JaCoCo [sonar-jacoco-plugin]：使用JaCOCO获取代码覆盖率

## 1.7. Jenkins集成Sonar

### 1.7.1 直接与Jenkins集成

登录Jenkins→系统设置→插件管理→可选插件→搜索SonarQube



安装后重启Jenkins

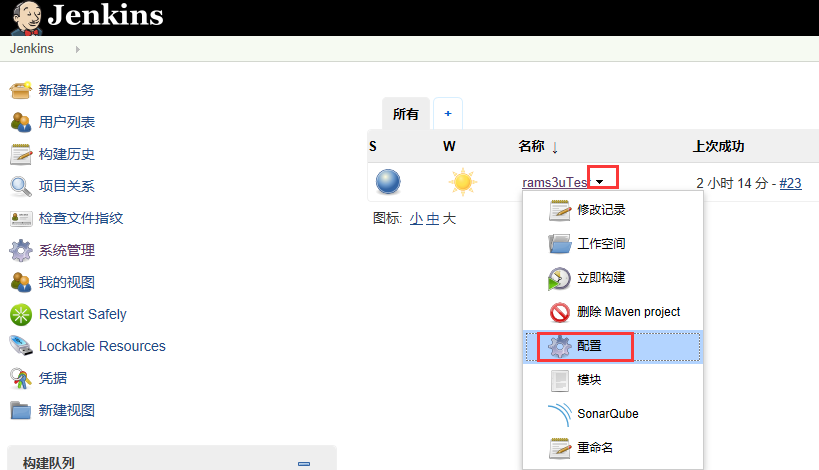
进入系统设置→全局工具配置



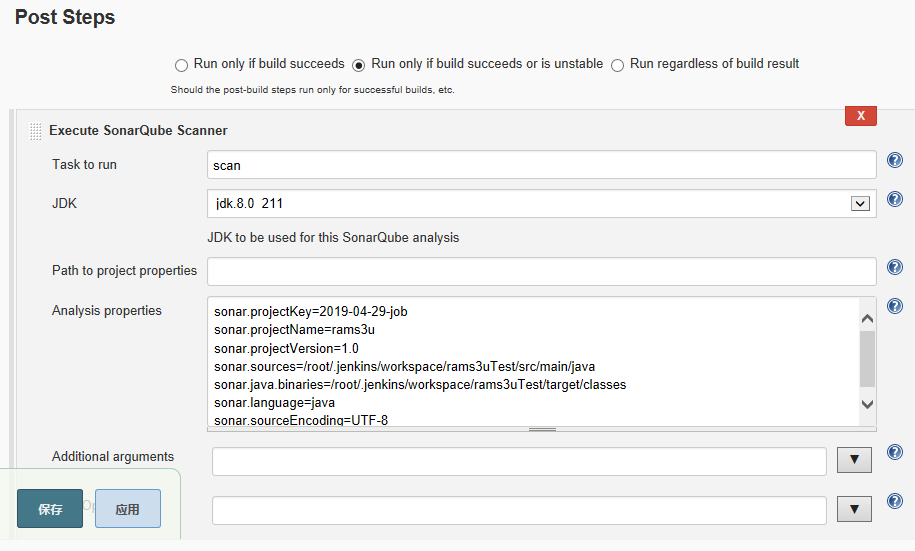
这里可以通过访问SonarQube Scanner官网进行下载相应Zip包,或者在Jenkins里进行自动安装,如果是自行下载的需要如上图配置相应路径

访问Sonar官网下载scanner插件：<https://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner>

下载配置保存后,进入之前Jenkins创建的Maven项目进行Sonar配置



找到Post Steps项



sonar.projectKey=2019-04-29-job // 必填,不要和其他sonar项目重名

sonar.projectName=rams3u // 必填,随意取

sonar.projectVersion=1.0 // 必填,随意取

// 必填 路径为项目构建后工程所在目录

sonar.sources=/root/.jenkins/workspace/rams3uTest/src/main/java

// 必填 路径为项目构建后target/classes所在目录

sonar.java.binaries=/root/.jenkins/workspace/rams3uTest/target/classes

sonar.java.coveragePlugin=jacoco

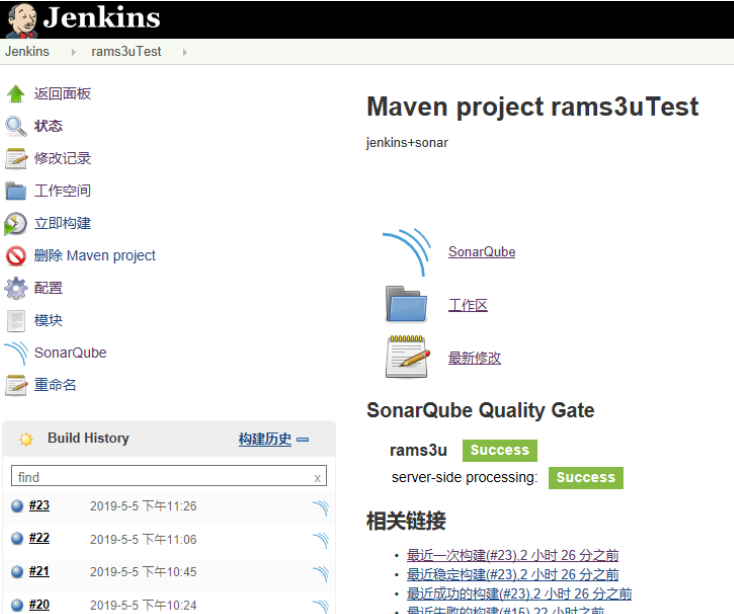
sonar.jacoco.reportPaths=./targe/coverage-reports/jacoco-unit.exec

sonar.dynamicAnalysis=reuseReports

sonar.language=java

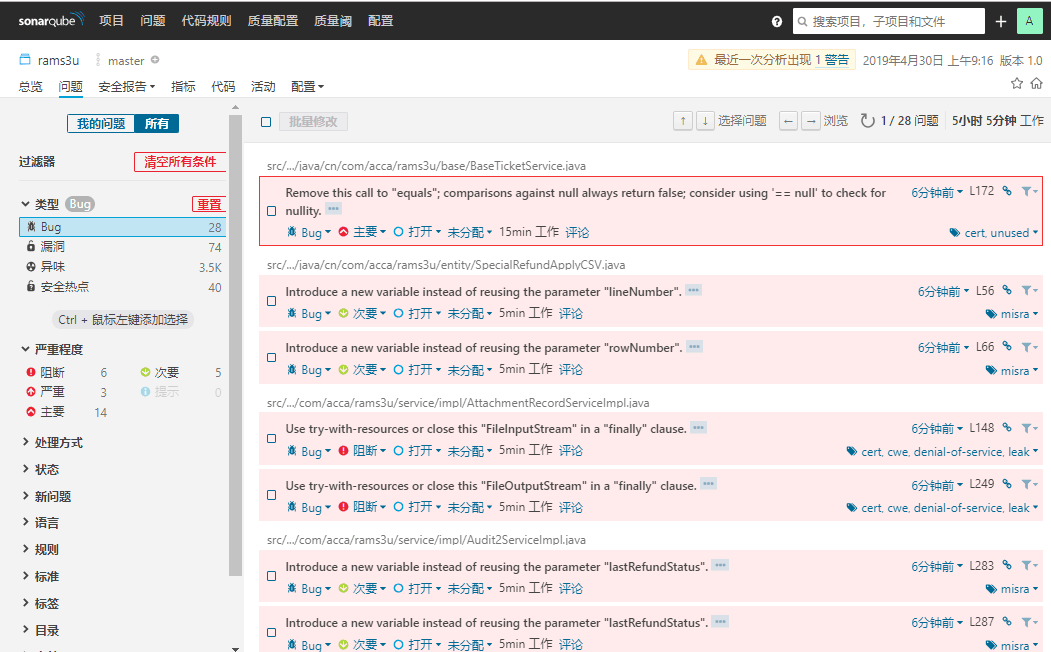
sonar.sourceEncoding=UTF-8

配置完成后进行立即构建,当构建成功时会自动进行Scanner扫描,并将扫描的结果反馈在Jenkins上,或者直接进入Sonar平台进行查看分析结果



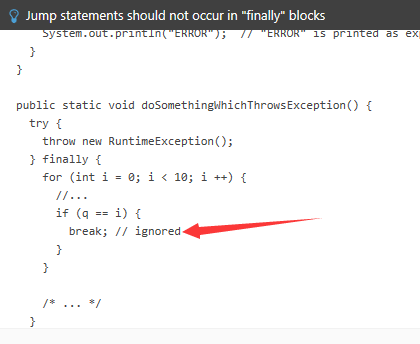
点击Success或Error即可进入Sonar平台查看详细的检查分析





详细错误信息:





## 1.8. 实践过程中踩到的坑

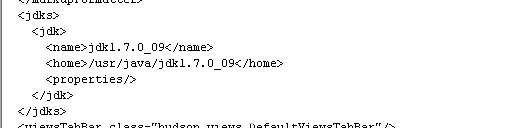
### 1.8.1 Jenkins安装配置问题

1. Jenkins安装在linux服务器上，通过yum安装的git一般都是1.X低版本的，执行Jenkins时会提示git版本的错误。

解决方法：需把通过yum安装的git卸载掉，重新安装2.X及以上高版本的git。

2. Jenkins构建项目时，会按照maven的setting.xml中配置的把依赖包下载到指定的仓库中。如果这个时候报关于maven的错误，很可能是本地仓库权限问题，给maven本地仓库赋予777的权限。

3. 当git版本，maven配置都没问题时，用Jenkins构建执行，会报jdk的问题，这是由于jdk版本低引起的，应装jdk1.8版本及以上。

注：如之前在Jenkins中已经配置了低版本jdk的路径，而linux服务器中也修改了jdk高版本，这时候会打不开Jenkins页面。需要做如下操作：一，找到Jenkins中的conf.xml，vi编辑 

4. jenkins执行shell脚本等场景中遇见的权限不足问题

解决方法：以root用户运行jenkins中shell命令

以centOS系统为例，记录下修改Jenkins以root用户运行的方法。

修改Jenkins配置文件

打开配置文件

vim /etc/sysconfig/jenkins

修改$JENKINS\_USER，并去掉当前行注释

$JENKINS\_USER="root"

修改Jenkins相关文件夹用户权限

chown -R root:root /var/lib/jenkins

chown -R root:root /var/cache/jenkins

chown -R root:root /var/log/jenkins

重启Jenkins服务并检查运行Jenkins的用户是否已经切换为root

重启Jenkins（若是其他方式安装的jenkins则重启方式略不同）

service jenkins restart service jenkins restart

查看Jenkins进程所属用户

service jenkins restart

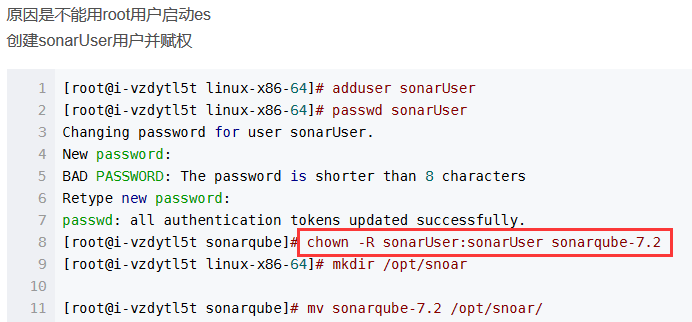
#### 若显示为root用户，则表示修改完成

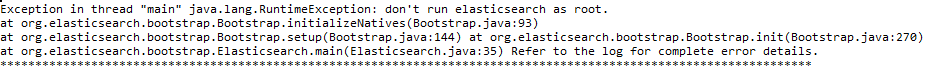
### 1.8.2 Sonar安装配置问题

1. 保证至少3GB可用内存，否则会直接导致启动失败

2..SonarQube6.6以上版本不能使用root账号启动,否则会报错

解决办法:新建一个sonar账号,并用root账号给sonar账号赋予权限



3. Linux不一样的用户启动,需要删除目录下的temp临时文件夹,不然会报错

4. 如果启动了sonar,但是无法进入平台,可能是因为防火墙没关,这时可以将防火墙关闭并重新载入

注：防火墙的shell命令：

　　firewall-cmd --zone=public --add-port=80/tcp --permanent　　（--permanent永久生效，没有此参数重启后失效）

　　firewall-cmd --reload　　重新载入

　　firewall-cmd --zone=public --remove-port=80/tcp --permanent　　删除

　　firewall-cmd --zone=public --query-port=80/tcp　　查看

　　firewall-cmd --state　　查看firewall是否运行

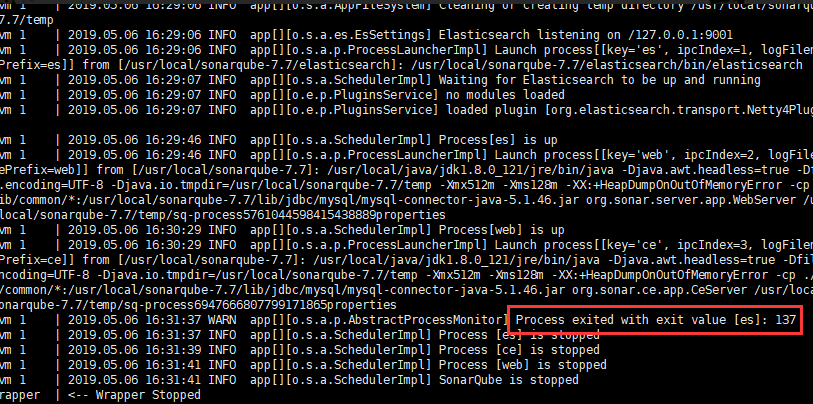
　　firewall-cmd --list-services　　查看当前开了哪些端口

　　firewall-cmd --get-services　　查看还有哪些服务可以打开

firewall-cmd --zone=public --list-ports　　查看所有打开的端口

5.如果Sonar配置好且运行成功,但过一会自动关闭且报出错误信息: **Process exited with exit value [es]: 137**

解决方案:因为分配内存过低引起,将虚拟机的运行内存提升至3GB或更高则可保证正常运行





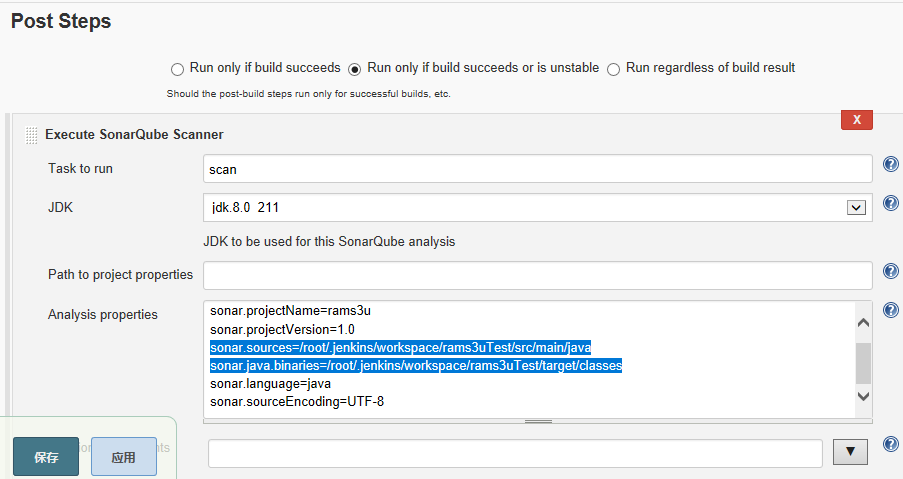
### 1.8.3 Jenkins集成Sonar问题

1.在Jenkins里配置SonarQube时,配置里有自动下载SonarQube Scanner的选项.文件比较大,最好自己下载解压然后指定对应解压的位置

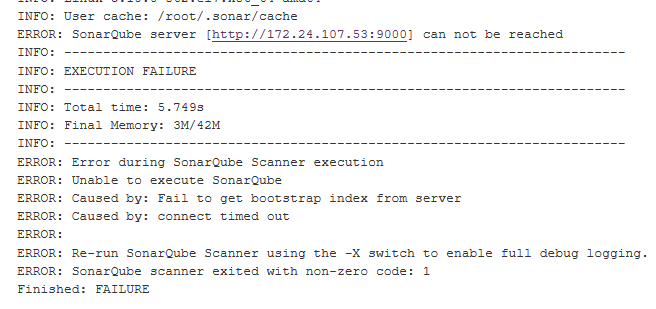
2.在配置构建项目时,如果项目是第一次构建，则单元测试可能会需要很长时间。如果不需要单元测试，在Build里添加跳过测试的mvn命令。.

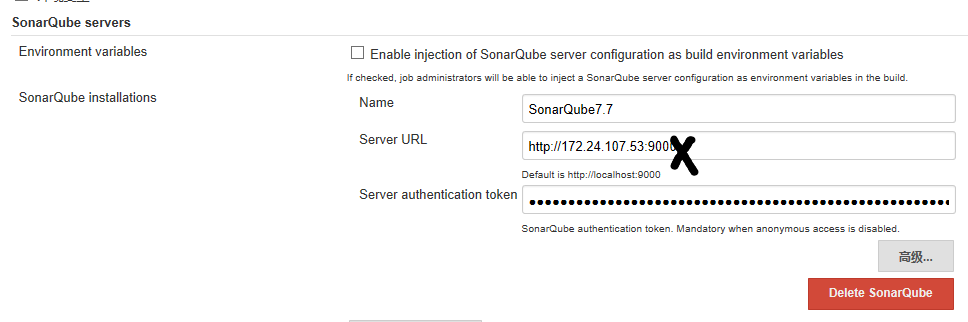


3.在配置构建项目时,在Post Steps配置sonar信息时,要把SonarQube Scanner扫描路径填写正确,否则扫描会失败并报错



4.如果Jenkins构建成功,但是Sonar扫描失败,可能是因为IP地址有变更,但是在Jenkins里配置的SonarQube Service没有做更新.





5.如果sonar扫描分析过后，页面上代码覆盖率显示为0.0%，则可能是扫描配置没有配置正确，或者pom文件里jacoco包路径错误，需要重新确认无误再进行扫描

在pom文件里加入:

<plugin>

<groupId>org.jacoco</groupId>

<artifactId>jacoco-maven-plugin</artifactId>

<version>0.7.7.201606060606</version>

<configuration>

<!--指定生成 .exec 文件的存放位置-->

<destFile>target/coverage-reports/jacoco-unit.exec</destFile>

<!--Jacoco 是根据 .exec 文件生成最终的报告，所以需指定 .exec 的存放路径-->

<dataFile>target/coverage-reports/jacoco-unit.exec</dataFile>

</configuration>

<executions>

<execution>

<id>jacoco-initialize</id>

<goals>

<goal>prepare-agent</goal>

</goals>

</execution>

<execution>

<id>jacoco-site</id>

<phase>test</phase>

<goals>

<goal>report</goal>

</goals>

</execution>

</executions>

</plugin>

在jenkins的项目配置里Post Steps里的Analysis properties添加

sonar.java.cov**e**ragePlugin=jacoco

sonar.jacoco.reportPaths=./targe/coverage-reports/jacoco-unit.exec

sonar.dynamicAnalysis=reuseReports

## 1.9 Jenkins自动化部署及三种构建部署方式

https://blog.csdn.net/achudk/article/details/78925081