gitlab+jenkins+sonar 做代码质量分析

目录

一、安装前准备

#1.环境：

centos6.5 64 位，关闭了 iptables、selinux

IP:192.168.0.75

GitLab：8.2.2 我用的是源码编码安装中文版，建立 git 用户为 Gitlab 用户

```
[root@vm5 ~]# ll /home/git
总用量 16
drwxr-xr-x 20 git git 4096 1月  26 2016 gitlab
drwxr-xr-x  8 git git 4096 10月 10 17:19 gitlab-shell
drwxr-xr-x  5 git git 4096 1月  26 2016 gitlab-workhorse
drwxrwx---  4 git git 4096 10月  8 15:18 repositories
```

Jenkins：jenkins2.24

主要作用是中介的作用，通过 gitlab 和 SonarQube 插件，连接 gitlab 和 sonar

如果使用当客户端用 git push 代码就自动触发 jenkins 调用 sonar 做质量分析的话，需要 gitlab 配置 Web Hooks(web 钩子),发触发事件发给 jenkins

SonarQube：SonarQube5.6 主要是做代码质量分析

内存大小：最小 3G，还是比较卡

#保持主机名和 hosts 名字一致

```
[root@vm5 ~]# hostname
vm5
[root@vm5 plugins]# cat    /etc/hosts
127.0.0.1       localhost vm5 localhost4 localhost4.localdomain4
::1                 localhost vm5 localhost6 localhost6.localdomain6
shutdown -r now
```

#安装 git

```
rpm -Uvh https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarch.rpm
wget http://rpms.famillecollet.com/enterprise/remi-release-6.rpm
rpm --import http://rpms.famillecollet.com/RPM-GPG-KEY-remi
rpm -ih remi-release-6.rpm
yum install -y git
```

发现问题：

1. jenskins+gitlab+sonar 3 个整合最低要 3G 内存，还是感觉很卡

2.SonarQube 因为内存 3G 还是很卡，固态磁盘还是卡，所以端口号起得很慢，约 2 分钟

3.jenkins 默认用的是 8080 默认会用 gitlab 冲突，所以需要修改一下端口

#2.安装 JDK

#因为 sonar 是基于 java 的，所以要安装 JDK，其它安装环境要求

#http://docs.sonarqube.org/display/SONAR/Requirements

#如果有 openJDK 则要先卸载

```
cd /disk1/tools/
rpm -qa | grep java
```

#去 oracle 官网下载 jdk

http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

```
rpm -ih jdk-8u102-linux-x64.rpm
java -version
javac -version
```

#安装不配置环境变量，可能出现某些类找不到

```
echo 'export JAVA_HOME=/usr/java/jdk1.8.0_102/' >>/etc/profile
echo 'export JRE_HOME=/usr/java/jdk1.8.0_102/jre' >>/etc/profile
echo 'PATH=$JAVA_HOME/bin:$PATH' >>/etc/profile
tail -3 /etc/profile
source /etc/profile
```

二、sonarqube5.6 安装

#1.下载安装包

#http://www.sonarqube.org/downloads/

```
wget https://sonarsource.bintray.com/Distribution/sonarqube/sonarqube-5.6.3.zip
mkdir -p    /disk1/app/sonar
unzip sonarqube-5.6.3.zip -d    /disk1/app/sonar
ll /disk1/app/sonar
```

```
vim    /etc/init.d/sonar
#!/bin/sh
#
# rc file for SonarQube
#
# chkconfig: 345 96 10
# description: SonarQube system (www.sonarsource.org)
#
### BEGIN INIT INFO
# Provides: sonar
# Required-Start: $network
# Required-Stop: $network
# Default-Start: 3 4 5
# Default-Stop: 0 1 2 6
# Short-Description: SonarQube system (www.sonarsource.org)
# Description: SonarQube system (www.sonarsource.org)
### END INIT INFO
/usr/bin/sonar $*
```

```
chmod +x    /etc/init.d/sonar
ln -s    /disk1/app/sonar/sonarqube-5.6.3/bin/linux-x86-64/sonar.sh    /usr/bin/sonar
chkconfig sonar on
```

#2.创建数据库和账号

```
mysql -uroot -p123456
#创建数据库
CREATE DATABASE sonar CHARACTER SET utf8 COLLATE utf8_general_ci;
GRANT ALL PRIVILEGES ON sonar.* TO 'sonar'@'localhost' IDENTIFIED BY '123456' WITH GRANT OPTION;
GRANT ALL PRIVILEGES ON sonar.* TO 'sonar'@'%' IDENTIFIED BY '123456' WITH GRANT OPTION;
FLUSH PRIVILEGES;
\q
```

#验证一下账号和密码

```
mysql -usonar -p123456 -e " show databases;"
```

#3.配置 sonar 参数

#参考 http://docs.sonarqube.org/display/SONAR/Installing+the+Server

```
cd /disk1/app/sonar/sonarqube-5.6.3/
cp conf/sonar.properties conf/sonar.properties.orig
sed -i 's/#sonar.jdbc.username=/sonar.jdbc.username=sonar/g' conf/sonar.properties
sed -i 's/#sonar.jdbc.password=/sonar.jdbc.password=123456/g' conf/sonar.properties
sed -i 's/#sonar.web.port=9000/sonar.web.port=9000/g' conf/sonar.properties
egrep "sonar.jdbc.username|sonar.jdbc.password|sonar.web.port=9000" conf/sonar.properties
```

#修改配置

```
vim conf/sonar.properties +23
```

```
14 sonar.jdbc.username=sonar
15 sonar.jdbc.password=123456
23
sonar.jdbc.url=jdbc:mysql://localhost:3306/sonar?useUnicode=true&characterEncoding=utf8&rewriteBatchedStat
em ents=true&useConfigs=maxPerformanc
105 sonar.web.port=9000
61 sonar.jdbc.maxActive=10
65 sonar.jdbc.maxIdle=5
69 sonar.jdbc.minIdle=2
74 sonar.jdbc.maxWait=5000
76 sonar.jdbc.minEvictableIdleTimeMillis=600000
77 sonar.jdbc.timeBetweenEvictionRunsMillis=30000
```

#4.安装 SonarQube Scanner

```
cd /disk1/tools/
http://docs.sonarqube.org/display/SCAN/Analyzing+with+SonarQube+Scanner
unzip sonar-scanner-2.8.zip -d    /disk1/app/sonar
ln -s /disk1/app/sonar/sonar-scanner-2.8/bin/sonar-scanner /usr/bin/sonar-scanner
```

#修改配置

```
cd /disk1/app/sonar/sonar-scanner-2.8/conf/
cp sonar-scanner.properties    sonar-scanner.properties.orig
vim sonar-scanner.properties
```

```
11 sonar.jdbc.username=sonar
12 sonar.jdbc.password=123456
18 sonar.jdbc.url=jdbc:mysql://localhost:3306/sonar?useUnicode=true&amp;characterEncoding=utf8
```

#5.安装 SonarQube runner

#http://docs.sonarqube.org/display/SONARQUBE51/Installing+and+Configuring+SonarQube+Runner

```
cd /disk1/tools/
wget http://repo1.maven.org/maven2/org/codehaus/sonar/runner/sonar-runner-dist/2.4/sonar-runner-dist-2.4.zip
unzip sonar-runner-dist-2.4.zip    -d /disk1/app/sonar
ll /disk1/app/sonar
ln -s /disk1/app/sonar/sonar-runner-2.4/bin/sonar-runner /usr/bin/sonar-runner
cd /disk1/app/sonar/sonar-runner-2.4/conf/
cp sonar-runner.properties sonar-runner.properties.orig
```

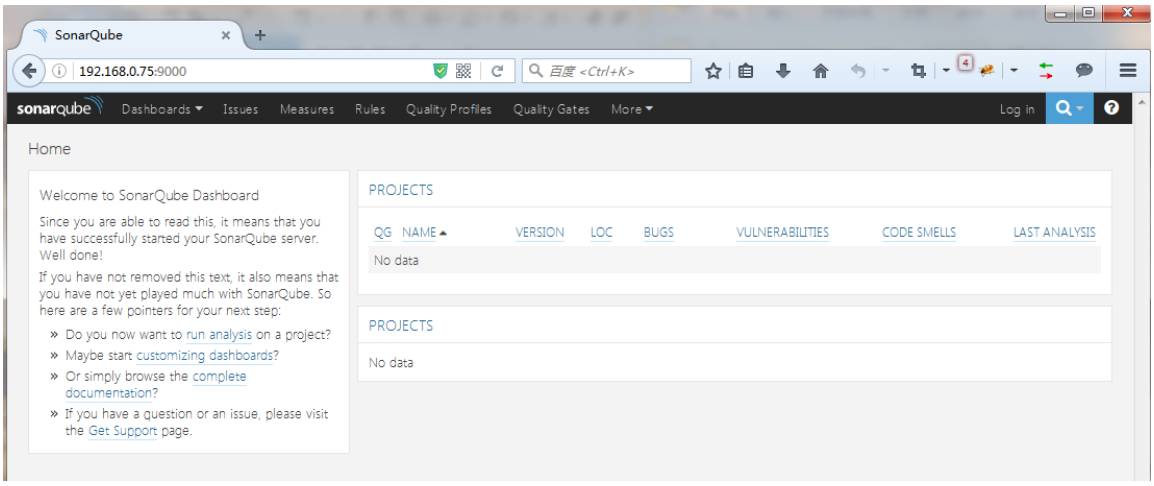
```
vim sonar-runner.properties
11 sonar.jdbc.url=jdbc:mysql://localhost:3306/sonar?useUnicode=true&amp;characterEncoding=utf8
20 sonar.jdbc.username=sonar
21 sonar.jdbc.password=123456
```

```
vim /etc/profile.d/sonar.sh
#------------添加如下代码-----
#!/bin/bash
SONAR_HOME=/disk1/app/sonar/sonarqube-5.6.3
SONAR_RUNNER_HOME=/disk1/app/sonar/sonar-runner-2.4
PATH=$SONAR_RUNNER_HOME/bin:$PATH
export SONAR_HOME
export SONAR_RUNNER_HOME
export PATH
#------------添加代码结束----------
source /etc/profile.d/sonar.sh
```

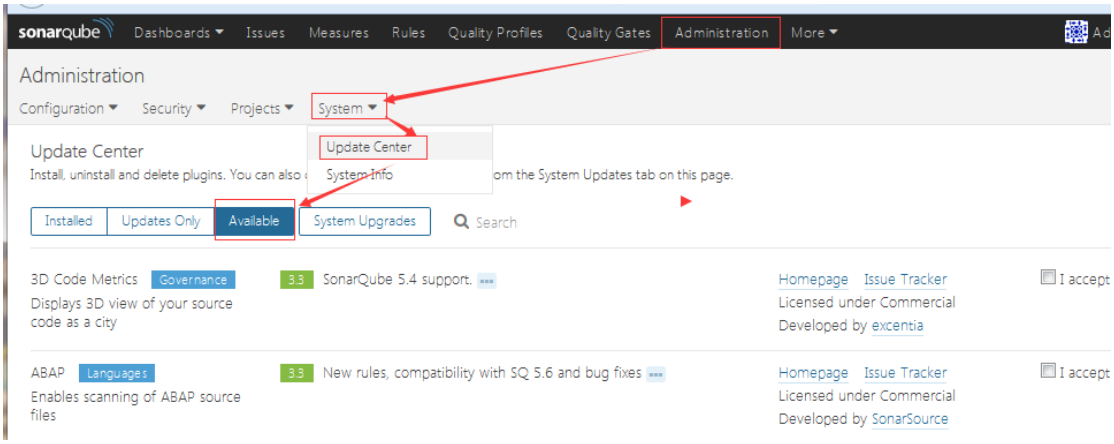
#6.启动 sonar

```
/etc/init.d/sonar start stop
/etc/init.d/sonar start restart
/etc/init.d/sonar start start
/etc/init.d/sonar start status
#需要等 2-3 分钟，端口号才起来
netstat -atnlp|grep 9000
ps -ef |grep sonar
```

#登陆:用浏览器登陆，需要运行 1 分钟这样才会出界面
http://192.168.0.75:9000/ 默认密码是 admin admin

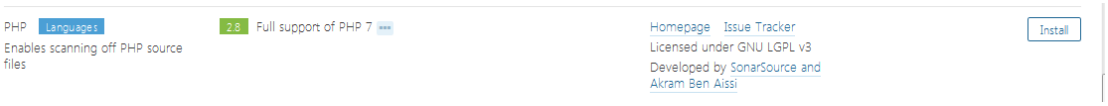


#点右上角的 log in，默认用户名和密码是 admin/admin
#去更新中心可以安装中文包

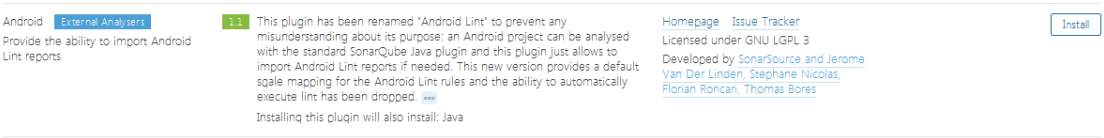


#搜索 chinese

#安装 php 支持



#安装 Android



#安装成功后，重启 sonarqube 服务，再次访问 http://ip:9000/，即可看到中文界面

```
/etc/init.d/sonar restart
```


```
cd /disk1/app/sonar/sonarqube-5.6.3/extensions/plugins/
wget http://nexus.talanlabs.com/content/groups/public_release/com/synaptix/sonar-gitlab-plugin/1.7.0/sonar-gitlab-plugin-1.7.0.jar
#重启
/etc/init.d/sonar restart
```

三、手工测试

我们用 sonar-runner 手工测试一下,具体使用说明网见如下链接：
http://docs.sonarqube.org/display/SONARQUBE51/Analyzing+with+SonarQube+Runner

随便拿一个之前弄过的 php 项目，把它复制到根目录下，我的情况下如：
网站 index.php 目录路径为：/disk1/www/information_server/backend/web，测试操作如下：

```
cd /disk1/www/information_server
vim sonar-project.properties
#添加如一内容：
sonar.projectKey=my:phpcook
sonar.projectName=PHP cook sonar test
sonar.projectVersion=1.0
#这里是 php 文件放的地方，这里表示根目录下的 backend/web 目录
sonar.sources=backend/web
# Language
sonar.language=php
sonar.dynamicAnalysis=false
# Encoding of the source files
sonar.sourceEncoding=UTF-8
#在当前目录下运行
sonar-runner
```

```
15:15:04.577 INFO  - Sensor PHPUnit Sensor
15:15:04.578 INFO  - No PHPUnit test report provided (see 'sonar.php.tests.reportPath' property)
15:15:04.578 INFO  - No PHPUnit unit test coverage report provided (see 'sonar.php.coverage.reportPath' property)
15:15:04.578 INFO  - No PHPUnit integration test coverage report provided (see 'sonar.php.coverage.itReportPath' property)
15:15:04.578 INFO  - No PHPUnit overall coverage report provided (see 'sonar.php.coverage.overallReportPath' property)
15:15:04.578 INFO  - Sensor PHPUnit Sensor (done) | time=1ms
15:15:04.578 INFO  - Sensor SCM Sensor
15:15:04.578 INFO  - SCM provider for this project is: git
15:15:04.587 INFO  - 2 files to be analyzed
15:15:04.949 INFO  - 2/2 files analyzed
15:15:04.949 INFO  - Sensor SCM Sensor (done) | time=371ms
15:15:04.949 INFO  - Sensor Zero Coverage Sensor
15:15:04.965 INFO  - Sensor Zero Coverage Sensor (done) | time=16ms
15:15:04.965 INFO  - Sensor Code Colorizer Sensor
15:15:04.968 INFO  - Sensor Code Colorizer Sensor (done) | time=3ms
15:15:04.968 INFO  - Sensor CPD Block Indexer
15:15:04.968 INFO  - DefaultCpdBlockIndexer is used for php
15:15:05.005 INFO  - Sensor CPD Block Indexer (done) | time=37ms
15:15:05.006 INFO  - Calculating CPD for 2 files
15:15:05.026 INFO  - CPD calculation finished
15:15:05.095 INFO  - Analysis report generated in 68ms, dir size=15 KB
15:15:05.115 INFO  - Analysis reports compressed in 19ms, zip size=8 KB
15:15:05.302 INFO  - Analysis report uploaded in 187ms
15:15:05.303 INFO  - ANALYSIS SUCCESSFUL, you can browse http://localhost:9000/dashboard/index/my:phpcook
15:15:05.303 INFO  - Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
15:15:05.303 INFO  - More about the report processing at http://localhost:9000/api/ce/task?id=AVd58k3BlIou0UlVvTlu
INFO: ------------------------------------------------------------------------
INFO: EXECUTION SUCCESS
INFO: ------------------------------------------------------------------------
Total time: 4.992s
Final Memory: 8M/143M
INFO: ------------------------------------------------------------------------
[root@vm5 information_server]#
```

#完成后用浏览器登陆 sonar，会如下图所示：

四、Jenkins 安装

#1.jenkins 安装

#1.yum 安装，http://pkg.jenkins-ci.org/redhat/

```
sudo wget -O /etc/yum.repos.d/jenkins.repo http://pkg.jenkins.io/redhat/jenkins.repo
sudo rpm --import http://pkg.jenkins.io/redhat/jenkins.io.key
yum install jenkins
```

用 yum 安装太慢我直接去目录下载包安装，目前最新版本为 jenkins-2.24-1.1.noarch.rpm

```
rpm --import http://pkg.jenkins.io/redhat/jenkins.io.key
rpm -ih jenkins-2.24-1.1.noarch.rpm
```

#修改目录配置文件，修改家目录和端口号（和 gitlab 有冲突）

```
mkdir -p /disk1/app/jenkins
chown jenkins.jenkins -R /disk1/app/jenkins
vim /etc/sysconfig/jenkins
10 #JENKINS_HOME="/var/lib/jenkins"
11 JENKINS_HOME="/disk1/app/jenkins"
```

```
57 #JENKINS_PORT="8080"
58 JENKINS_PORT="9080"
/etc/init.d/jenkins start
```

/usr/lib/jenkins/：jenkins 安装目录，WAR 包会放在这里。

/etc/sysconfig/jenkins：jenkins 配置文件，“端口”，“JENKINS_HOME”等都可以在这里配置。

Starting Jenkins -bash: /usr/bin/java: No such file or directory

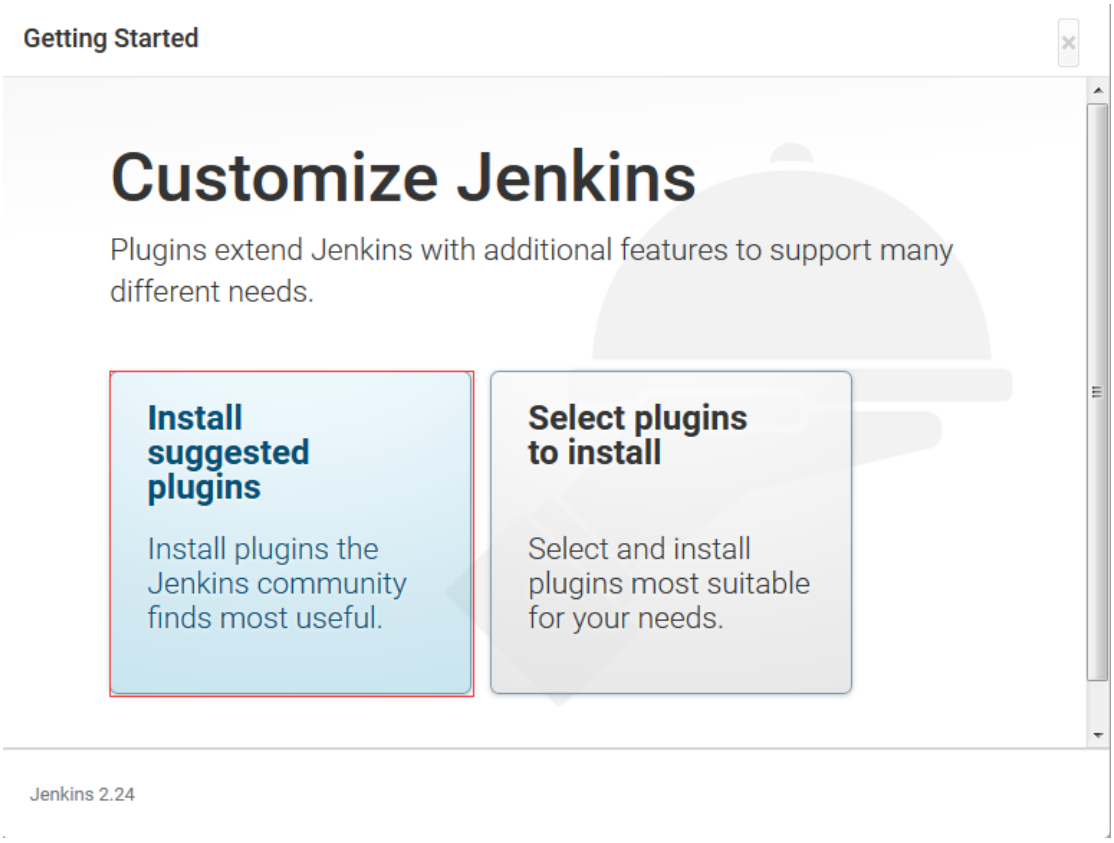
时就需要“vi /etc/init.d/jenkins”，把 java 路径加上，一般不用加也行，如下：

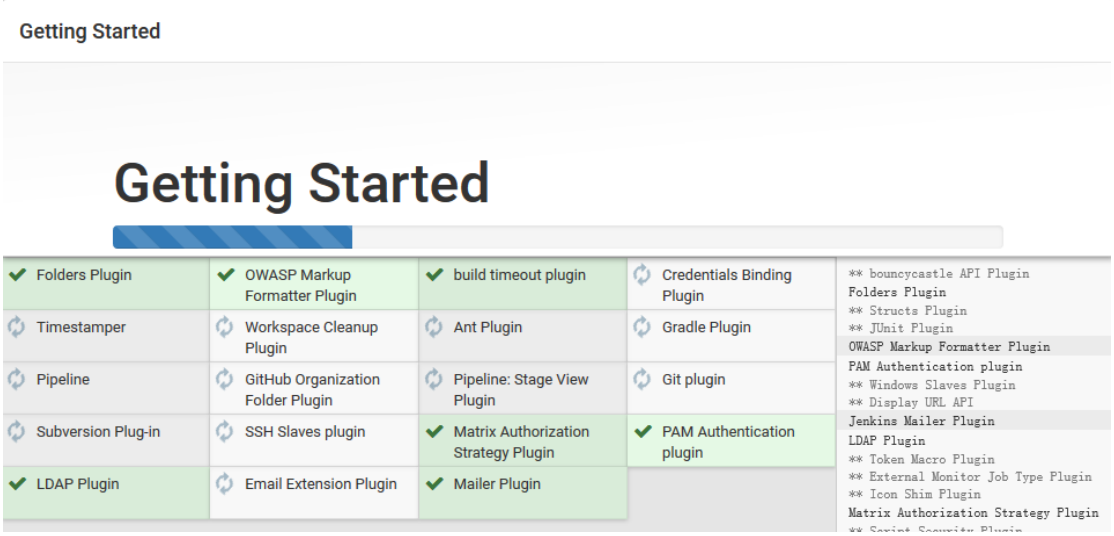
```
67 candidates="
68 /etc/alternatives/java
69 /usr/lib/jvm/java-1.6.0/bin/java
70 /usr/lib/jvm/jre-1.6.0/bin/java
71 /usr/lib/jvm/java-1.7.0/bin/java
72 /usr/lib/jvm/jre-1.7.0/bin/java
73 /usr/lib/jvm/java-1.8.0/bin/java
74 /usr/lib/jvm/jre-1.8.0/bin/java
75 /usr/bin/java
76 /usr/java/jdk1.8.0_102/bin/java 我添加的
77 "
78 for candidate in $candidates
79 do
80   [ -x "$JENKINS_JAVA_CMD" ] && break
81   JENKINS_JAVA_CMD="$candidate"
82 done
83
84 JAVA_CMD="$JENKINS_JAVA_CMD $JENKINS_JAVA_OPTIONS -DJENKINS_HOME=$JENKINS_HOME -jar $JENKINS_WAR"
85 PARAMS="--logfile=/var/log/jenkins/jenkins.log --webroot=/var/cache/jenkins/war --daemon"
```

#打开浏览器输入：http://192.168.0.75:9080

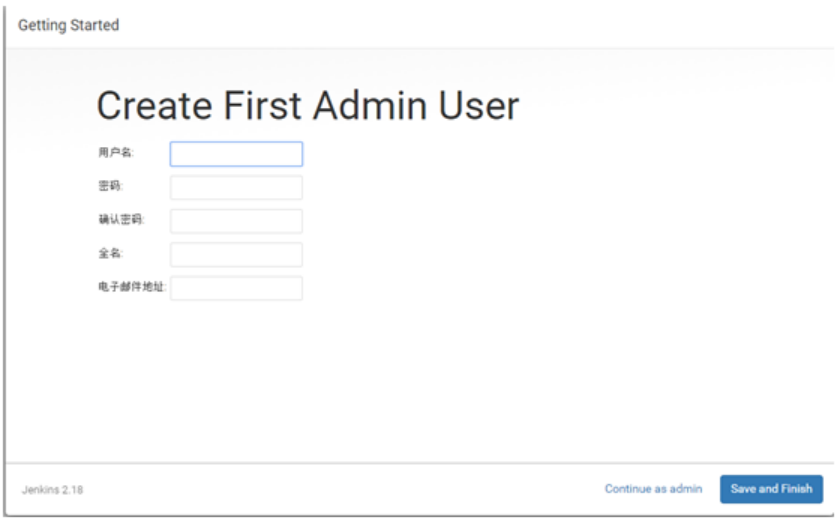
选择“Install suggested plugins”安装默认的插件，下面 Jenkins 就会自己去下载相关的插件进行安装。

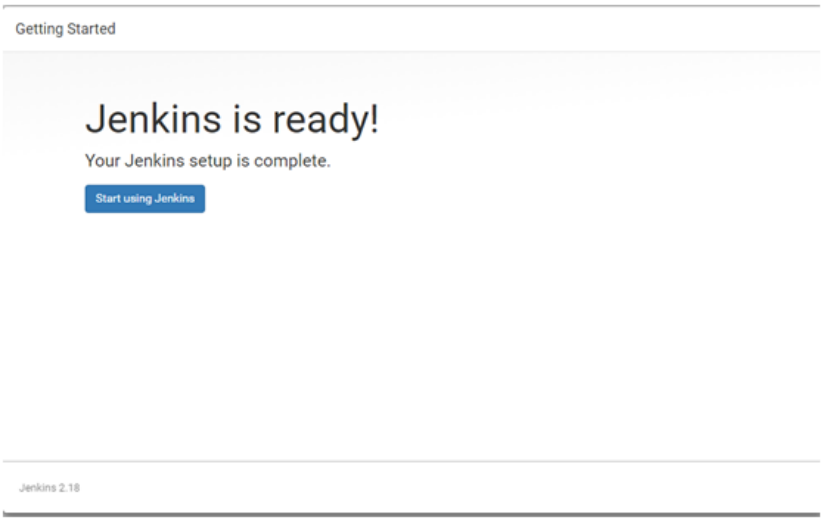
```
[root@vm5 tools]# cat /disk1/app/jenkins/secrets/initialAdminPassword
c7da96fd193a4d44ad03c6d25047d800
```





创建超级管理员账号 ：我这里写的是 hua 123456





#2. Jenkins 插件安装

登陆 Jenkins-->系统管理-->管理插件

要让 Jenkins 可以自动 build git repo 中的代码，需要安装 GIT Client Plugin 和 GIT Plugin。

要想 Jenkins 可以收到 Gitlab 发来的 hook 从而自动 build，需要安装 Gitlab Hook Plugin。

要让 Jenkins 可以在 build 完成之后根据 TAP（test anything protocol）文件生成 graph，需要安装 TAP Plugin。

SonarQube：SonarQube Plugin、Sonargraph Integration Jenkins Plugin、Sonargraph Plugin

实用插件：

#如果插件安装不上请请检查 hostname 和 hosts 中名字是否一致，重启服务器再试

iOS 专用：Xcode integration

Android 专用：Gradle plugin (默认已经安装）

Gitlab 插件：GitLab Plugin 和 Gitlab Hook Plugin、GitLab Logo Plugin

Git 插件：Git plugin、Git client plugin

sonar 插件：SonarQube Plugin、Sonargraph Integration Jenkins Plugin、Sonargraph Plugin

----------我只安装以上的软件----------

其它可选：

jdk：JDK Parameter Plugin

php 插件：PHP Plugin

GitBuckit 插件： GitBuckit plugin

签名证书管理插件: Credentials Plugin 和 Keychains and Provisioning Profiles Management

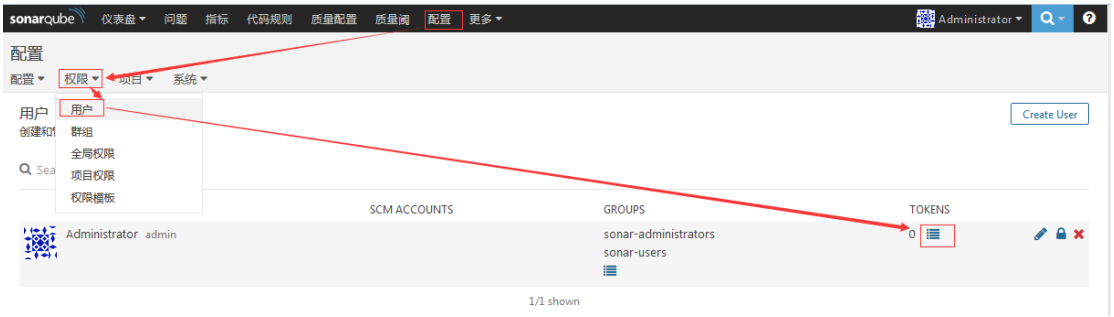
FTP 插件: Publish over FTP

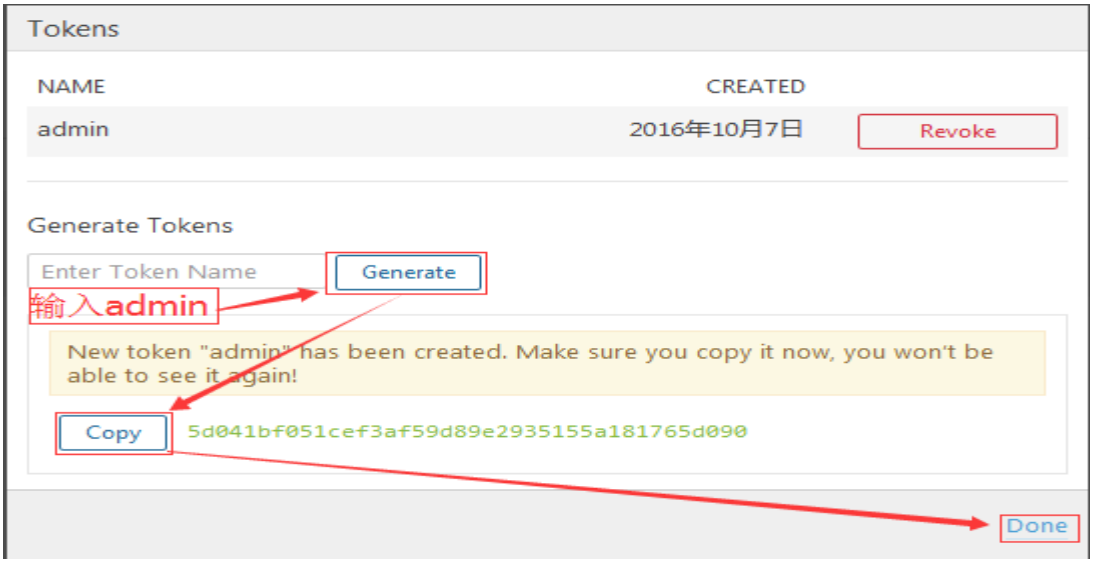
脚本插件: Post-Build Script Plug-in
修改 Build 名称/描述(二维码)： build-name-setter / description setter plugin
获取仓库提交的 commit log： Git Changelog Plugin
自定义全局变量: Environment Injector Plugin
自定义邮件插件： Email Extension Plugin
获取当前登录用户信息： build-user-vars-plugin
显示代码测试覆盖率报表： Cobertura Plugin
来展示生成的单元测试报表，支持一切单测框架，如 junit、nosetests 等： Junit Plugin
其它： GIT plugin / SSH Credentials Plugin

#2. Jenkins 集成 Sonar 进行代码质量管理

#1.设置 sonar 中 token

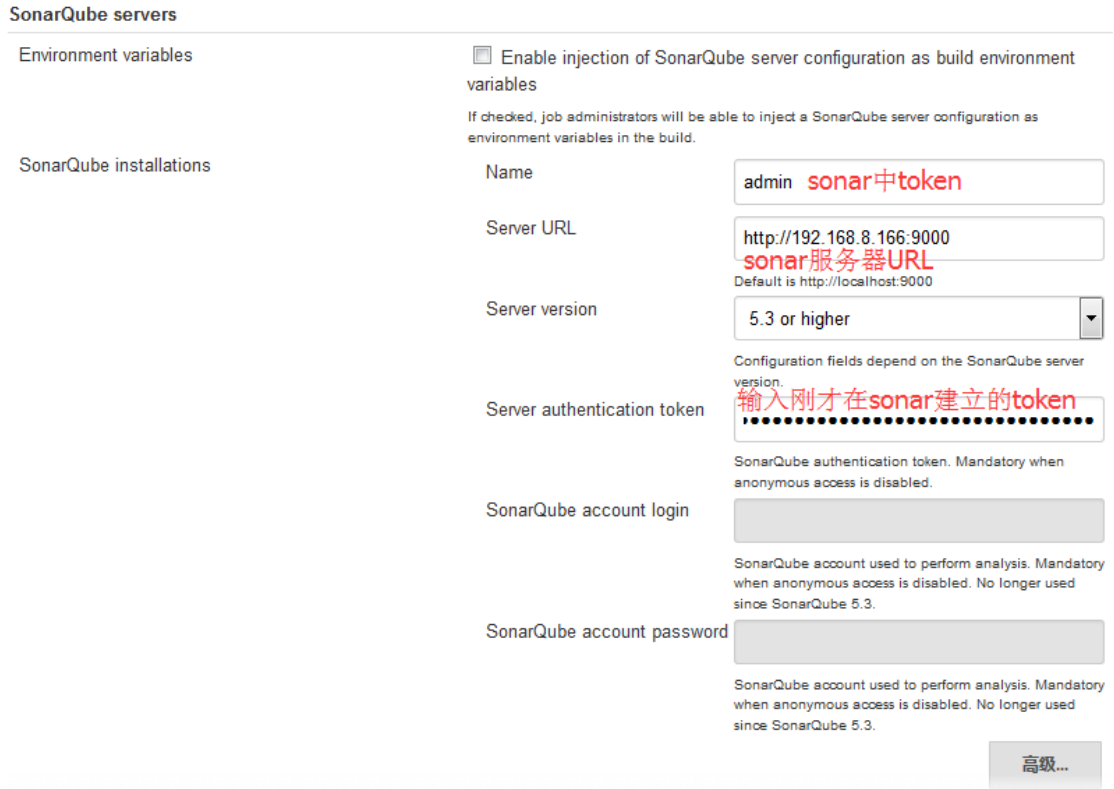
首先用 admin 用户登陆 sonarQube，[配置]-->[权限] -->[用户]



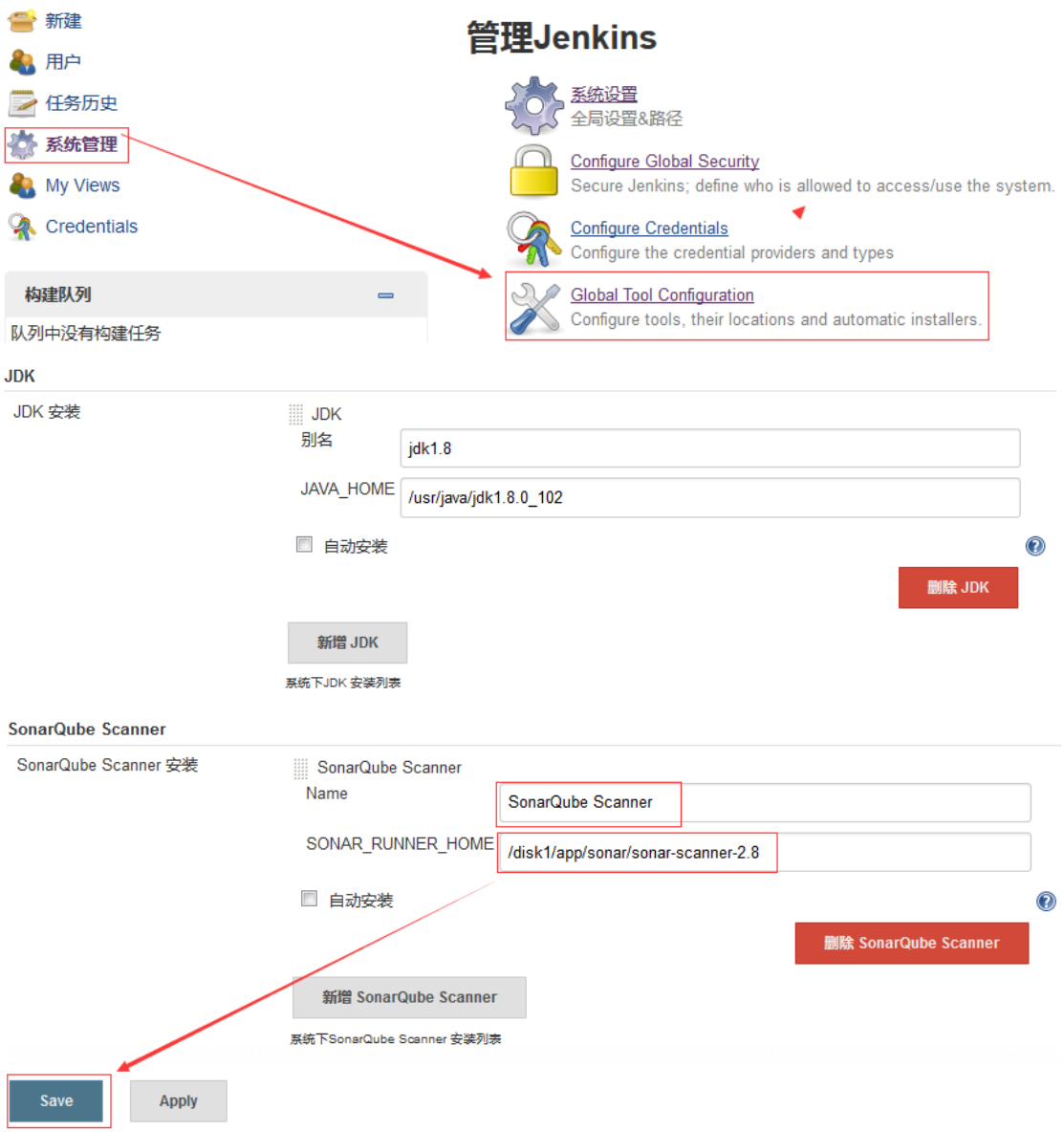


得到用户是 admin，token 号是：5d041bf051cef3af59d89e2935155a181765d090

#2.Jenkins 安装 Sonar 插件，然后点[系统管理] -> [系统设置]配置 Sonar，设置完保存



#3. 点[系统管理]->[Global Tool Configuration]添加 Sonar Scanner 扫描器



3.配置 gitlab

gitlab 里用建立一个 hua 账号里面建立一个名字为 hua 的项目

#2.建立 ssh-key

可以用 Deploy Key 或 SSH Keys 添加，不过用 SSH Keys 也可以，能对 hua 账号下所有用户都能读写，我这里用 SSH Keys

项目部署公钥（Deploy Key）允许通过 SSH 协议以只读的方式访问项目，不需要输入密码，而且数据是使用你上传的公钥加密传输的。与 HTTPS 协议相比，SSH 协议的数据传输效率要更高和稳定些，支持超大项目数据的传输。顾名思义，本功能的用途就在于项目的部署上，只需要一次性添加部署公钥，就可以免去现有 HTTPS 方式每次输入密码和普通 SSH 方式对代码有读写权限的麻烦，数据全程加密传输，保证了代码的安全性。

#生成 ssh-key

ssh-keygen -t rsa -C "gitlab@vm5"

#我这里修改密钥的名字的默认位置

/var/lib/jenkins/.ssh/id_gitlab_hua

```
[root@vm5 ~]# ssh-keygen -t rsa  -C "gitlab@vm5"
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): /var/lib/jenkins/.ssh/id_gitlab_hua
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/lib/jenkins/.ssh/id_gitlab_hua.
Your public key has been saved in /var/lib/jenkins/.ssh/id_gitlab_hua.pub.
The key fingerprint is:
58:de:2c:d0:25:12:30:e5:de:46:95:58:53:21:84:cd gitlab@vm5
The key's randomart image is:
+--[ RSA 2048]----+
|    oo+..BBoo.   |
|     o oo+Eo     |
|      o +        |
|     . B o       |
|      o S o      |
|       . .       |
|                 |
|                 |
|                 |
+-----------------+
[root@vm5 ~]# chown jenkins.jenkins /var/lib/jenkins/.ssh/id_gitlab_hua*
[root@vm5 ~]# ll /var/lib/jenkins/.ssh/
总用量 12
-rw------- 1 jenkins jenkins 1675 10月  9 17:54 id_gitlab_hua
-rw-r--r-- 1 jenkins jenkins  392 10月  9 17:54 id_gitlab_hua.pub
-rw-r--r-- 1 jenkins jenkins  394 10月  9 16:30 known_hosts
```

回车密码为空

修改目录入名字

#查看公钥内容

[root@vm5 ~]# cat /var/lib/jenkins/.ssh/id_gitlab_hua.pub

ssh-rsa

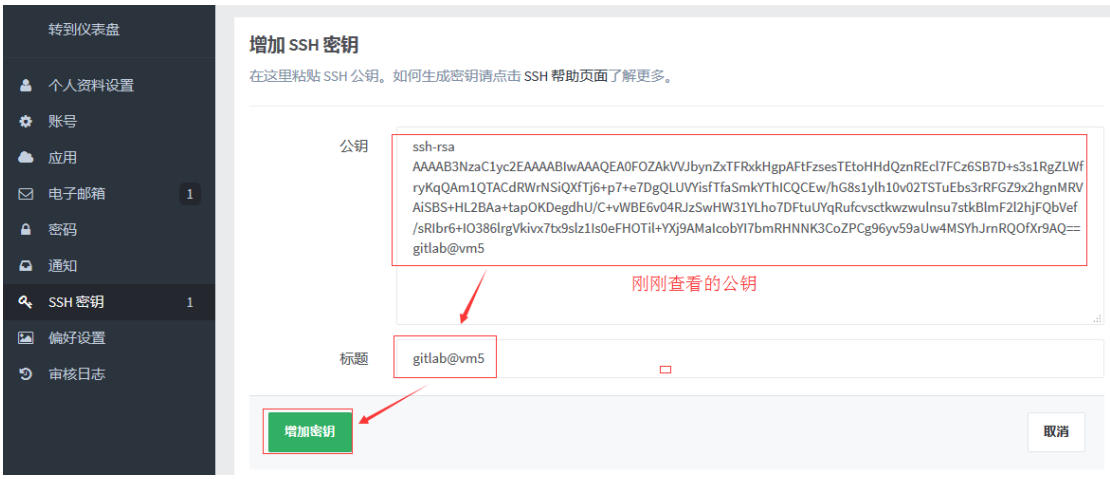
AAAAB3NzaC1yc2EAAAABIwAAAQEA0FOZAkVVJbynZxTFRxkHgpAFtFzsesTEtoHHdQznREcl7FCz6SB7D+s3s1RgZLWfryKqQAm1QTACdRWrNSiQXfTj6+p7+e7DgQLUVYisfTfaSmkYThICQCEw/hG8s1ylh10v02TSTuEbs3rRFGZ9x2hgnMRVAiSBS+HL2BAa+tapOKDegdhU/C+vWBE6v04RJzSwHW31YLho7DFtuUYqRufcvsctkwzwulnsu7stkBlmF2l2hjFQbVef/sRIbr6+IO386lrgVkivx7tx9slz1Is0eFHOTil+YXj9AMaIcobYI7bmRHNNK3CoZPCg96yv59aUw4MSYhJrnRQOfXr9AQ== gitlab@vm5

#查看私钥内容

cat /var/lib/jenkins/.ssh/id_gitlab_hua

```
[root@vm5 ~]# cat /var/lib/jenkins/.ssh/id_gitlab_hua
-----BEGIN RSA PRIVATE KEY-----
MIIEoQIBAAKCAQEA0FOZAkVVJbynZxTFRxkHgpAFtFzsesTEtoHHdQznREcl7FCz
6SB7D+s3s1RgZLWfryKqQAm1QTACdRWrNSiQXfTj6+p7+e7DgQLUVYisfTfaSmkY
ThICQCEw/hG8s1ylh10v02TSTuEbs3rRFGZ9x2hgnMRVAiSBS+HL2BAa+tapOKDe
gdhU/C+vWBE6v04RJzSwHW31YLho7DFtuUYqRufcvsctkwzwulnsu7stkBlmF2l2
hjFQbVef/sRIbr6+IO386lrgVkivx7tx9slz1Is0eFHOTil+YXj9AMaIcobYI7bm
RHNNK3CoZPCg96yv59aUw4MSYhJrnRQOfXr9AQIBIwKCAQAXzwosJS5NdKVq3c1n
Ng98or7SyMqRsBZ7QggqomOSxk16F9oMA7ZK9k+CNYdb93/2wiIV8nsdZJKRCcps
eaoZXdDnwwbavDOZty41qThI0y7j7sDy+r5tuqaDb73LWwuacQzHs8BDh3DhT91h
asyL0Wogi3dt9YsenXZh1fR0cBkD8iZGKBF/d5b+trWjnZThQVWOes9V+qFMYrAs
3VJKmPGUbIZJDKYuQ/5mkB5GQ9v2L0SHUCuchg3NbTse6cpR0DfRrkcy6FbL7+Wz
9blh5FF63WDfzqj/RMR4y6s9tAv2rtyZA0WaKa/BvQZGiCVKFgEqlpkq7w4RWD9Z
qoSLAoGBAPGEq2orkM20IEFeJz0V42aSmHNDhuWTPpegnZlw40jzjjqWW+wZUMtd
cn7pHtwuTKP+g0MKv3J2SRYEoY0eSxQFCAJYfGlZ+NejKIBx3TVkubKMIUwUJUBT
MaynTtnui9ZMQjpPVhfM56JZhdLhVGhZoBUsaPC+D0xr+oDI2VXZAoGBANzRbqVw
aO4suQTFK0hRx9PhoJA7NHmdKM3vNE6/0lFGn2BVtap6jXdIFHvRL0h6+v1uyJvb
dGBdeZE0pvYB9f/QCBInKYf+3uCsH9eeRihXQ7j5H3KTs/dUrZk4gZQoE5ko+qG6
QJRZnEQsNNctMQFF9fVidGVsV2QZVWNkDZ9pAoGAG5oiOAT6myqHV+2A08f8u0P0
Km4eC5vMo51T1wWWUXrrruycyoaM5Aqt/+AgyLTVjxXF28a2y0C35UJbmxlnqoQ7
bft77sh0NebCzNnQI1v39yX1LUQhg7G8iMKp7QVRz1ksI+vPUy1cTRGLoxJwC+z8
WjD2DOKFZ9HTfG65wKsCgYEA1oJOOk/w516lGpOmY4Kk3HTH3JFI6yr0cEd0pEVP
9y6pc4Z19hCmrmNHGTGbpX7H7uClnrftZO0XAhXkBOv2SPX5NjSsAHPuc9poE0lL
d6U6eSyE8vXiBjUAaPUQKYYEaOX6yP4S2UEbdWzFndQg+ewwwnzmGWHufn8Cfc7o
qXsCgYBZydjwx06LxcwRDC/iv1D2P3443Ft0F/HBYSnKT3x5MJV0C04mDYPda2eD
EwvLU9UKgS+Bgu7SS8JeZuoK079GFTqou/v6r9kk1UJQrDDttT/0r2RGAfpalhsa
p2YePivrUUrAK4HzLVcFGYHNBgX2hUDRe2FeBpJfB9CChuC9XQ==
-----END RSA PRIVATE KEY-----
```

#3.在 gitlab 添加 ssh-key



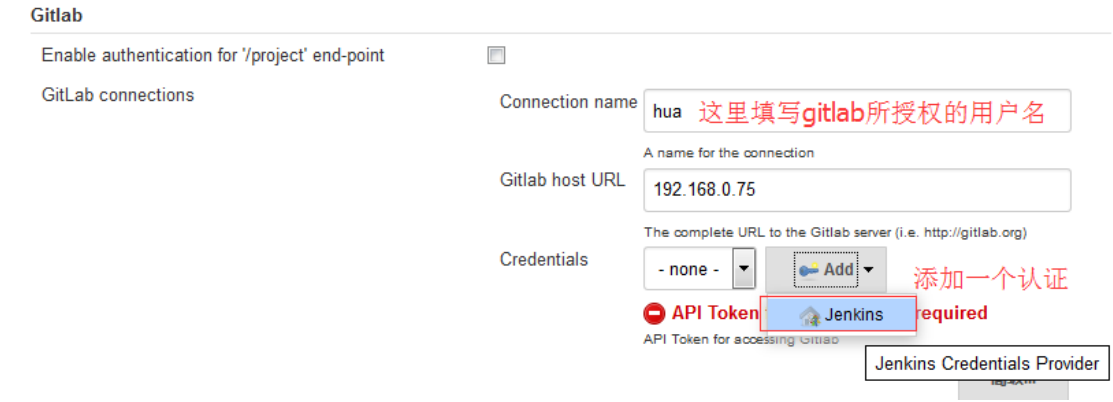
#4.jenkins 配置 gitlab

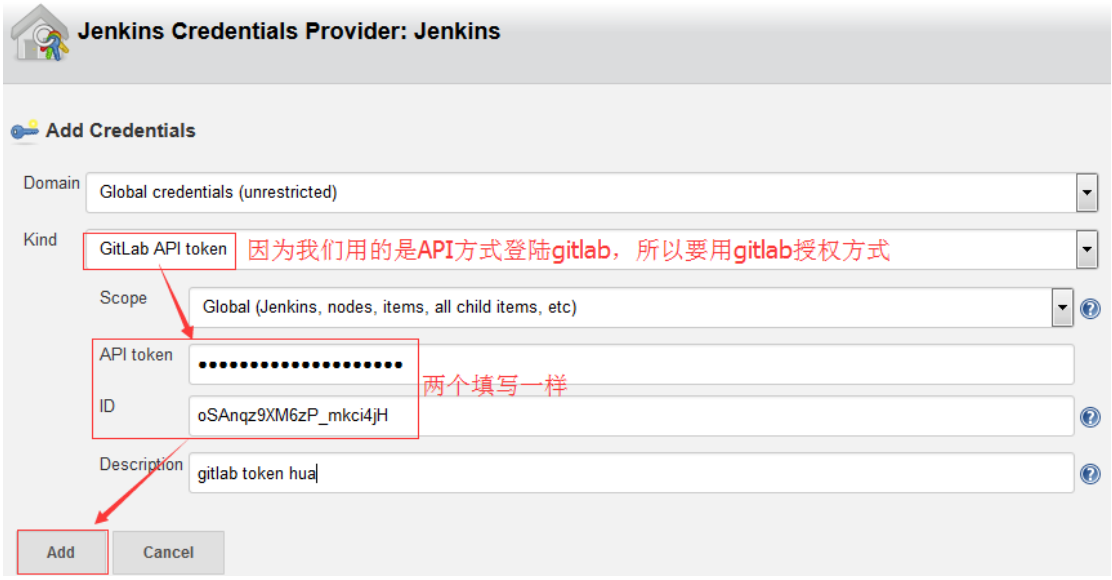
#1.用 hua 账号登陆 gitlab

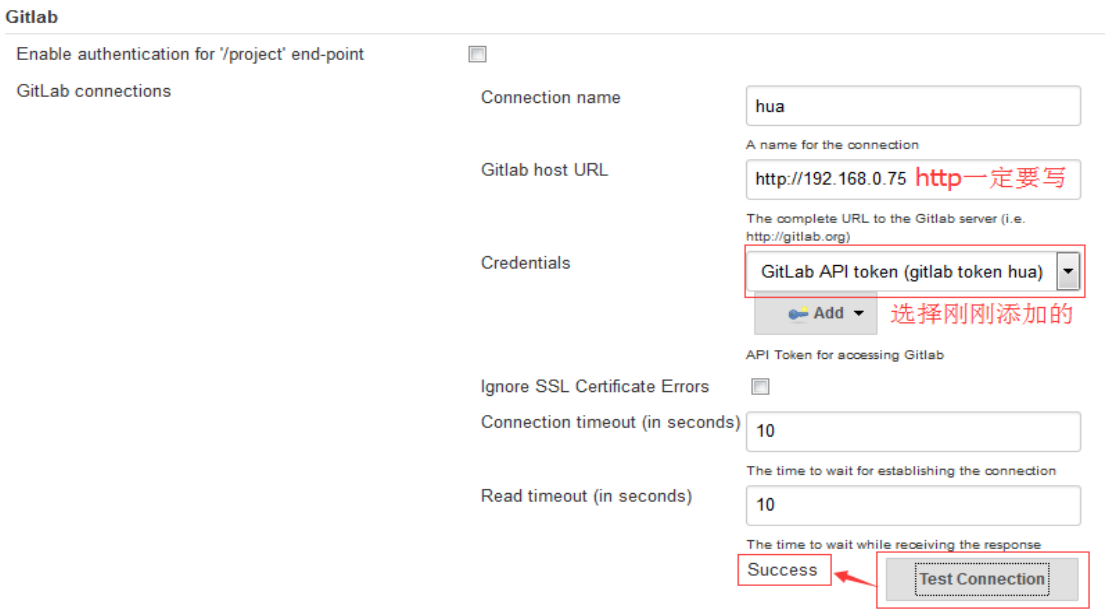
这里的授权账号就是 jenkins 的 token，得知 hua 账号的 token 是：oSAnqz9XM6zP_mkci4jH

#2.在 jenkins 配置 gitlab

[系统管理]-->[系统设置]里"**Gitlab**"选项







#5.测试（连 gitlab 一起配置）

#登陆 jenkins 新建一个任务

General 源码管理 构建触发器 构建环境 构建 构建后操作

项目名称 demon

描述

[Plain text] 预览

GitHub project

GitLab connection hua

GitLab Repository Name

Throttle builds

丢弃旧的构建

Strategy Log Rotation

保持构建的天数

如果非空，构建记录将保存此天数

保持构建的最大个数 30

如果非空，最多此数目的构建记录将被保存

保存 Apply

源码管理

None

Git

Repositories

Repository URL git@192.168.0.75:hua/hua.git

Failed to connect to repository : Command "git ls-remote -h git@192.168.0.75:hua/hua.git HEAD" returned status code 128:
stdout:
stderr: Host key verification failed.
fatal: Could not read from remote repository.

Please make sure you have the correct access rights and the repository exists.

Credentials - none - Add

Jenkins

Jenkins Credentials Provider

高级...

Add Repository

Kind: SSH Username with private key

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: jenkins

Private Key: Enter directly

Key:

```
9y6pc4Z19hCmrmNHGTGbpX7H7uClnrftZO0XAhXkBOv2SPX5NjSsAHPuc9poE0lL
d6U6eSyE8vXiBjUAaPUQKYYEaOX6yP4S2UEbdWzFndQg+ewwwnzmGWHufn8Cfc7o
qXsCgYBZydjwx06LxcwRDC/iv1D2P3443Ft0F/HBYSnKT3x5MJV0C04mDYPda2eD
EwvLU9UKgS+Bgu7SS8JeZuoK079GFTqou/v6r9kk1UJQrDDttT/0r2RGAfpalhsa
p2YePivrUUrAK4HzLVcFGYHNBgX2hUDRe2FeBpJfB9CChuC9XQ==
-----END RSA PRIVATE KEY-----
```

cat /var/lib/jenkins/.ssh/id_gitlab_hua

From a file on Jenkins master

From the Jenkins master ~/.ssh

Passphrase: 密码为空

ID:

Description: gitlab hua ssh

Add Cancel

Repositories

Repository URL: git@192.168.0.75:hua/hua.git

Failed to connect to repository : Command "git ls-remote -h git@192.168.0.75:hua/hua.git HEAD" returned status code 128:
stdout:
stderr: GitLab: API is not accessible
fatal: Could not read from remote repository.

Please make sure you have the correct access rights and the repository exists.

Credentials: jenkins (gitlab hua ssh) Add

尝试用本地客户端git登陆也报同样的错误
gitlab.yml配置把localhost修修改为ip地址而gitlab-shell的config.yml配置gitlab_url没有把localhost修改过来

高级...

Add Repository

```
hua@LUOBO-509100954 MINGW64 /d/6
$ git clone git@192.168.0.75:hua/hua.git
Cloning into 'hua'...
GitLab: API is not accessible
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
```

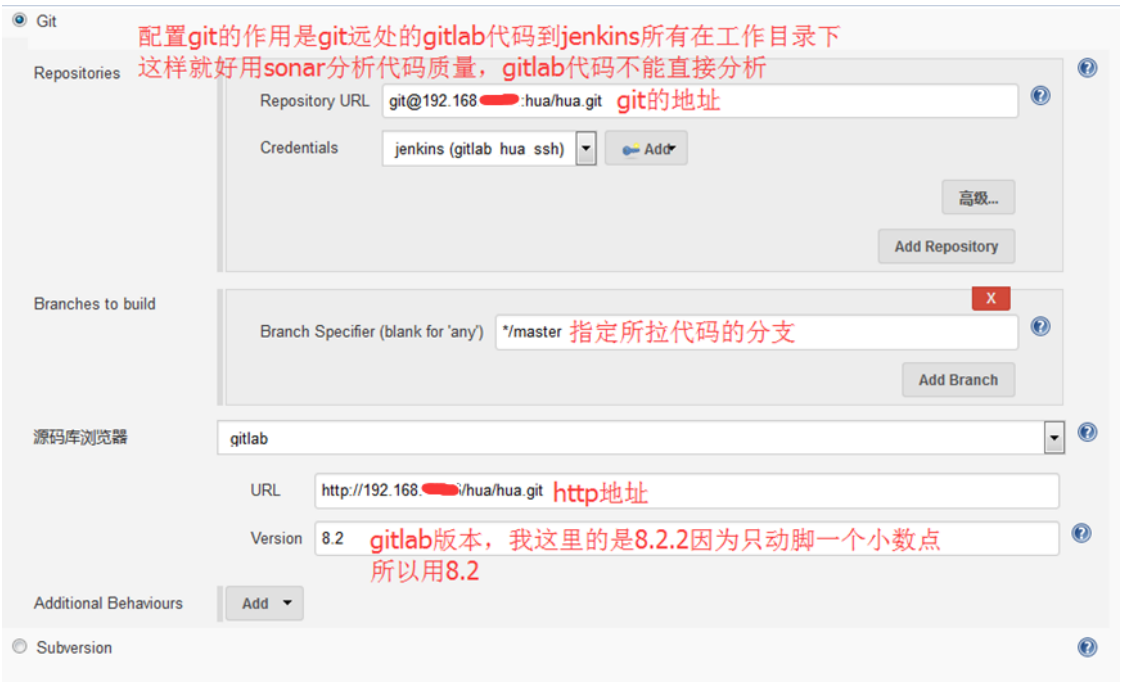
报错处理：

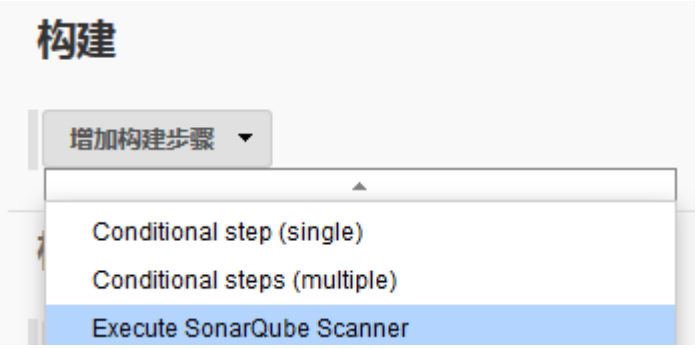
只修改了 gitlab/config/gitlab.yml 里面的 localhost，而没有相应修改 gitlab-shell/config.yml 中的 localhost，不一致引起的错误。

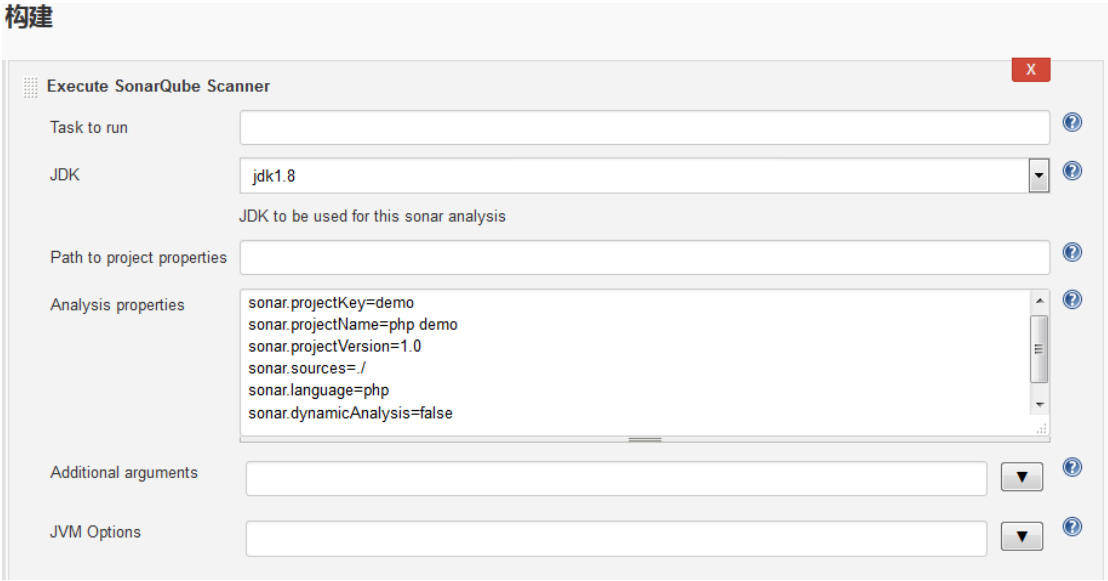
```
[root@vm5 ~]# grep "gitlab_url" /home/git/gitlab-shell/config.yml
#gitlab_url: http://localhost/
gitlab_url: http://192.168.0.75/
[root@vm5 ~]# grep 'host:' /home/git/gitlab/config/gitlab.yml
    #host: localhost                    IP地址要保持一致
    host: 192.168.0.75
    # Otherwise, ssh host will be set to the `host:` value above
    # ssh_host: ssh.host_example.com
    host: "imap.gmail.com"
        host: '_your_ldap_server'
      #   host:
    #host: localhost
    host: 192.168.0.75
        host: 127.0.0.1
[root@vm5 ~]#
```

#修改完重启 gitlab

/etc/init.d/gitlab restart







sonar.projectKey=demo

sonar.projectName=php demo

sonar.projectVersion=1.0

sonar.sources=./

sonar.language=php

sonar.dynamicAnalysis=false

sonar.sourceEncoding=UTF-8

保存

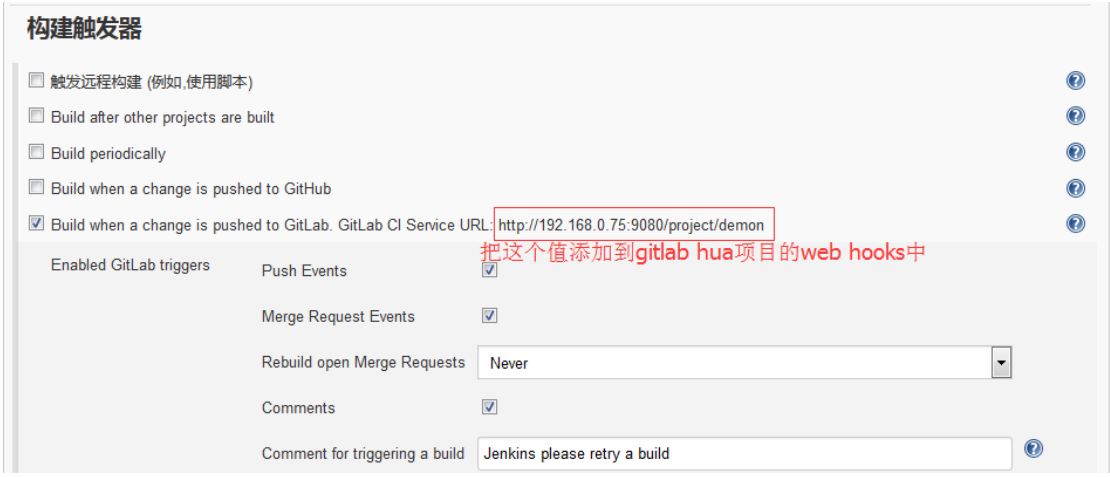
#执行完了，可以查看输出结果

```
INFO: DefaultCpdBlockIndexer is used for php
INFO: Sensor CPD Block Indexer (done) | time=81ms
INFO: Calculating CPD for 2 files
INFO: CPD calculation finished
INFO: Analysis report generated in 194ms, dir size=19 KB
INFO: Analysis reports compressed in 36ms, zip size=13 KB
INFO: Analysis report uploaded in 1064ms
INFO: ANALYSIS SUCCESSFUL, you can browse http://192.168.8.166:9000/dashboard/index/demo
INFO: Note that you will be able to access the updated dashboard once the server has processed the submitted
analysis report
INFO: More about the report processing at http://192.168.8.166:9000/api/ce/task?id=AVeeP5EWzsS0057P4zvL
INFO: ------------------------------------------------------------------------
INFO: EXECUTION SUCCESS
INFO: ------------------------------------------------------------------------
INFO: Total time: 30.535s
INFO: Final Memory: 49M/185M
INFO: ------------------------------------------------------------------------
Finished: SUCCESS
```

#查看 sonar 分析结果

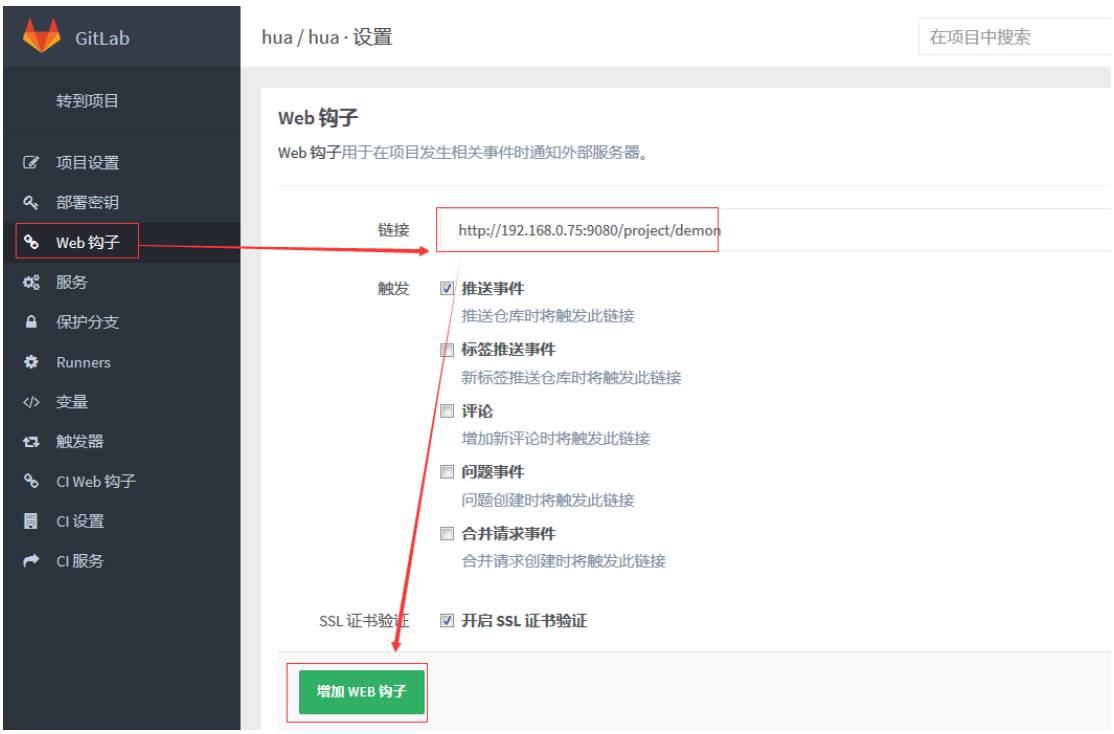
#测试 2，调 push 触发，在上面的基础上做如下修改

只需要把构建触发器中“GitLab CI Service URL: http://192.168.0.75:9080/project/demon”中 URL 地址添加到 gitlab 所在项目的“Web Hooks”即可。

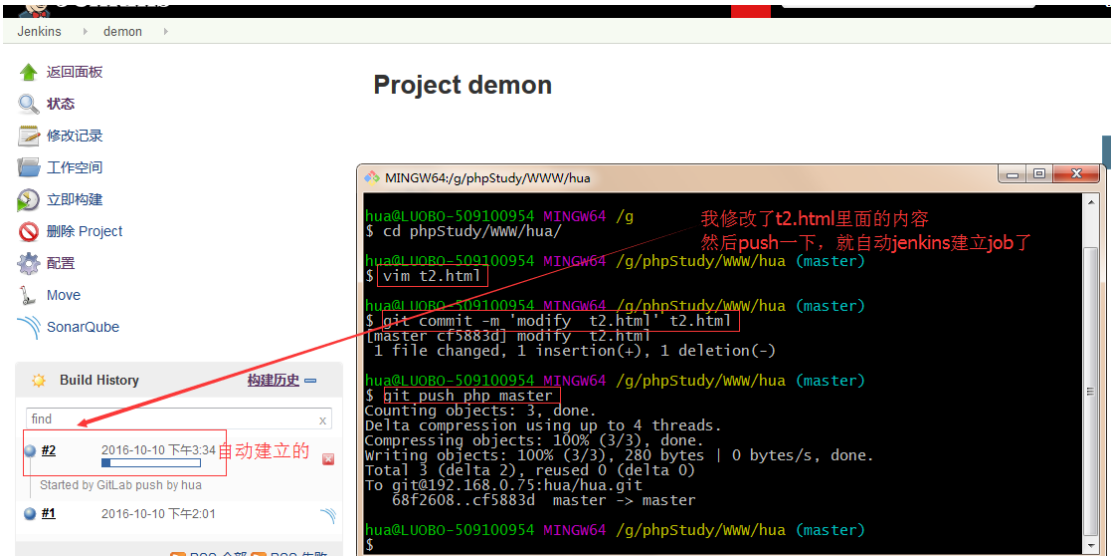

构建触发器
触发远程构建 (例如,使用脚本)
Build after other projects are built
Build periodically
Build when a change is pushed to GitHub
Build when a change is pushed to GitLab. GitLab CI Service URL: http://192.168.0.75:9080/project/demon
把这个值添加到gitlab hua项目的web hooks中
Enabled GitLab triggers
Push Events
Merge Request Events
Rebuild open Merge Requests
Never
Comments
Comment for triggering a build
Jenkins please retry a build


GitLab
项目
搜索
项目
活动
群组
里程碑
问题
0
合并请求
0
你的项目
星标项目
浏览项目
按名称过滤
+新项目
H
hua / hua
php test
0


成员
标记
维基
设置
用户活动
hua



git 客户端测试：



#再用 jenkins 中配置 sonar token 所在用用户（我这里是 admin）登陆 sonar，在 sonar 会看到自动生成一个 php demon 项目，查看时间是对得上的，如图

五、附

附一：jenkins 升级

cd /usr/lib/Jenkins
mv jenkins.war jenkins.war.old
#然后把下载好的更新包上传到这个目录下，重启 jenkins 即可
/etc/init.d/jenkins restart
这种方法发现升级后有问题

参考：
http://blog.csdn.net/jsjjohn88/article/details/44114267
http://docs.sonarqube.org/display/PLUG/PHP+Plugin
http://www.centoscn.com/image-text/install/2016/0724/7665.html
GitLab-CI 与 GitLab-Runner：
http://www.jianshu.com/p/2b43151fb92e
GitLab-CI-runner 调用 sonar 分析代码
https://blog.97md.net/20141106/989/
持续集成交付
https://chegva.com/continuous-integration/
gitlab push 触发 Jenkins Job
http://www.cnphp6.com/archives/115564

fly 飞翔
Q:715031064
2016.10.10