Jenkins+Gitlab+SonarQube代码质量管理集成

Jenkins+Gitlab+SonarQube 代码质量管理集成

一、环境准备

1.1、JDK11环境安装

1.2、Docker环境下搭建SonarQube所需的PostgreSQL数据库

由于本次将安装的SonarQube为最新的7.9版本，所以这里简单实用Docker搭建一个PostgreSQL。

1.2.1、Docker的安装

```
# 校验Linux的内核是否为3.10及以上
uname -r
# 安装docker
yum install docker
# 配置163镜像
vim /etc/docker/daemon.json
```

```
{
 "registry-mirrors": ["http://hub-mirror.c.163.com"]
}
```

常用docker命令

1.2.2、Docker下的PostgreSQL安装

1.2.2.1、下载PostgreSQL镜像

```
# 拉取最新的postgres镜像
docker pull postgres:10.10
```

```
[root@localhost ~]# docker images
```

1.2.2.2、数据持久化

```
# 创建PostgreSQL的数据持久化
docker volume create pgdata
# 查看创建的数据持久化仓库
docker volume ls
```

```
[root@localhost ~]# docker volume ls
```

1.2.2.3、启动容器

```
docker run -d -it --rm -v pgdata:/var/lib/postgresql/data -p 5432:5432 docker.io/postgres:10.10
# -it:
# --rm：指定容器停止后自动删除容器(不支持以docker run -d启动的容器)
# -v：给容器挂载存储卷，挂载到容器的某个目录
# -p： 指定端口号
# -d：后台运行
```

查看容器运行状态

```
[root@localhost ~]# docker run -d -it --rm -v pgdata:/var/lib/postgresql/data -p 5432:5432 docker.io/postgres:10.10
e288dc5ea66fed73bdfb7b0a38d9be65f3a8257b844094093d06c72c628d8e70
[root@localhost ~]# docker ps -a
```

1.2.2.4、登录PostgreSQL

```
# 进入到PostgreSQL容器中
docker exec -it e20da0174db8 bash
# 切换到postgres系统用户
su postgres
# 创建一个给SonarQube使用的超级用户（-s 是指成为超级用户，-P(大定)是指定密码）
createuser -P -s -e sonar
```

```
[root@localhost ~]# docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                    NAMES
e20da0174db8        postgres:latest     "docker-entrypoint..."   30 hours ago        Up 28 minutes       0.0.0.0:5432->5432/tcp   happy_engelbart
[root@localhost ~]# docker exec -it e20da0174db8 bash
root@e20da0174db8:/# su postgres
postgres@e20da0174db8:/$ createuser -P -s -e sonar
Enter password for new role:
Enter it again:
```

1.2.2.5、创建snor数据库

```
# 连接数据库
psql
# 创建sonar数据库
create database sonar owner=sonar;
```

给sonar数据库创建一个schema

```
# 切换到sonar数据库
\c sonar
# 创建schema指定owner
create schema my_schema authorization sonar;
```

```
[root@localhost ~]# docker ps -a
CONTAINER ID        IMAGE                      COMMAND                  CREATED             STATUS              PORTS                    NAMES
090ae3fdd177        docker.io/postgres:10.10   "docker-entrypoint..."   2 hours ago         Up 2 hours          0.0.0.0:5432->5432/tcp   pensive
_euler
[root@localhost ~]# docker exec -it 090ae3fdd177 bash
root@090ae3fdd177:/# su postgers
No passwd entry for user 'postgers'
root@090ae3fdd177:/# su postgres
postgres@090ae3fdd177:/$ psql
psql (10.10 (Debian 10.10-1.pgdg90+1))
Type "help" for help.

postgres=# \c sonar
You are now connected to database "sonar" as user "postgres".
sonar=# create schema my_schema authorization sonar;
```

二、SonarQube安装

2.1、下载SonarQube

点击链接，进入到官网，选择7.9版本进行下载

```
# 下载SonarQube
wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-7.9.1.zip
```

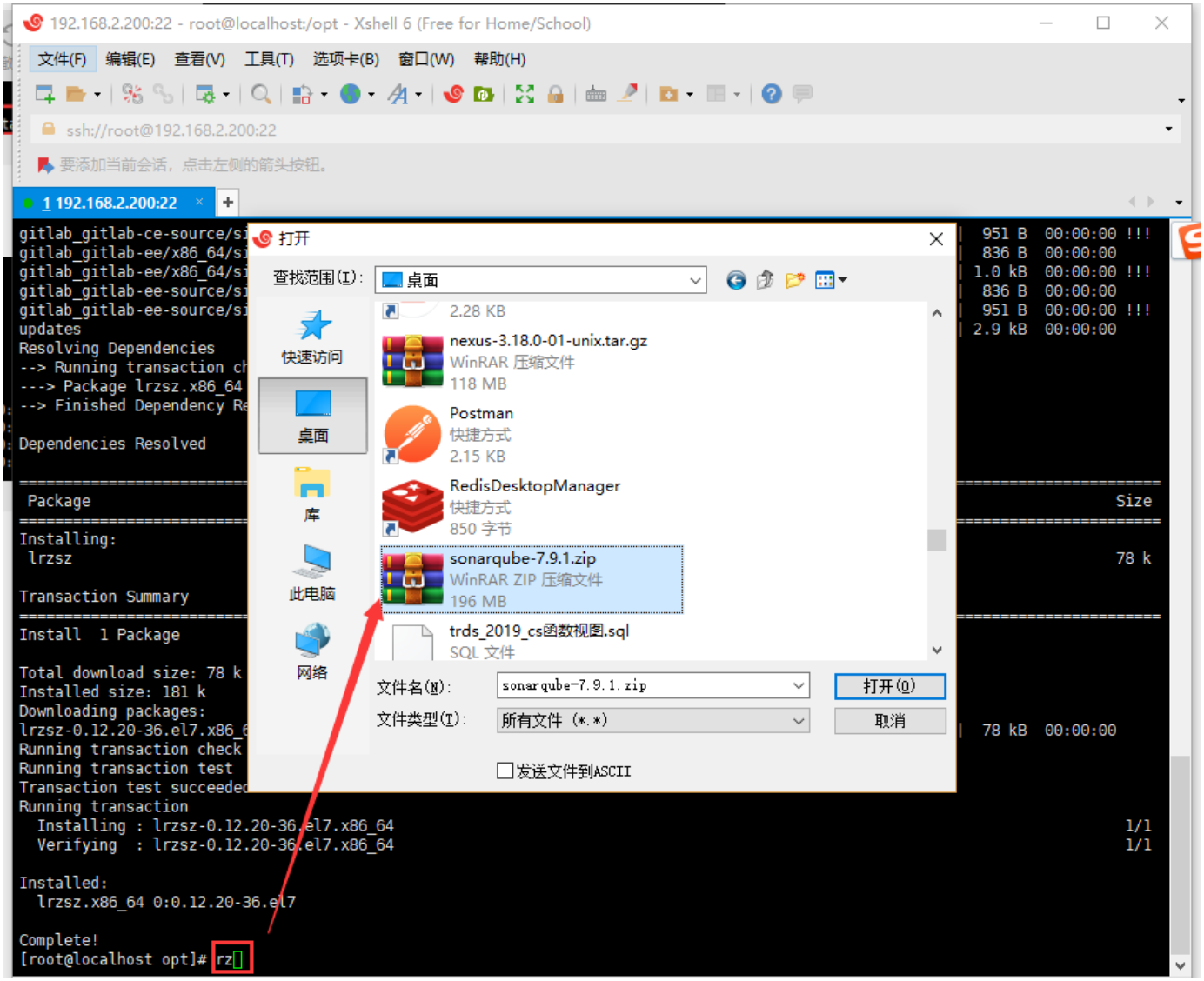
这里在Linux主机中使用wget下载应用比较慢，也可以直接在windows平台下下载完成后，使用rz命令上传到Linux主机中。

```
[root@localhost opt]# wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-7.9.1.zip
--2019-10-12 15:26:36--  https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-7.9.1.zip
Resolving binaries.sonarsource.com (binaries.sonarsource.com)... 91.134.125.245
Connecting to binaries.sonarsource.com (binaries.sonarsource.com)|91.134.125.245|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 206192630 (197M) [application/zip]
Saving to: 'sonarqube-7.9.1.zip'
```

这里如果执行rz出现下面的命令没有找到的，那就安装一下

```
[root@localhost opt]# rz
-bash: rz: command not found
[root@localhost opt]# yum -y install lrzsz
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: ap.stykers.moe
 * extras: ap.stykers.moe
 * updates: ap.stykers.moe
base                                                                    | 3.6 kB  00:00:00
extras                                                                  | 2.9 kB  00:00:00
```

上传SonarQube压缩包



2.2、解压SonarQube的zip文件

使用unzip工具解压zip文件

```
# 解压zip文件
unzip sonarqube-7.9.1.zip
# 安装unzip
yum install -y unzip zip
```

```
[root@localhost opt]# unzip sonarqube-7.9.1.zip
```

解压后文件

```
[root@localhost opt]# ll
total 201360
drwxr-xr-x.  3 root root        27 Apr 21  2017 code
drwxr-xr-x.  3 root root        21 Apr 21  2017 data
drwxr-xr-x  10 root root       212 Oct 12 06:27 gitlab
drwxr-xr-x.  2 root root         6 Apr 21  2017 javaapps
```

2.3、创建SonarQube的用户sonar

```
# 创建sonar用户
useradd sonar
# 修改/opt/sonarqube-7.9.1文件夹的所属用户组和用户都为sonar
chown -R sonar.sonar /opt/sonarqube-7.9.1
```

```
[root@localhost opt]# useradd sonar
[root@localhost opt]# chown -R sonar.sonar /opt/sonarqube-7.9.1
[root@localhost opt]# ll
total 201360
drwxr-xr-x.  3 root  root         27 Apr 21  2017 code
drwxr-xr-x.  3 root  root         21 Apr 21  2017 data
drwxr-xr-x  10 root  root        212 Oct 12 06:27 gitlab
drwxr-xr-x.  2 root  root          6 Apr 21  2017 javaapps
```

2.4、优化系统的参数

这里的优化参数，配置后可能还是会有问题，具体可以从log日志中去查看报错的问题原因，然后对应修改！！！

```
[sonar@localhost sonarqube-7.9.1]$ ll
total 12
drwxr-xr-x 6 sonar sonar   94 Jul 10 12:21 bin
drwxr-xr-x 2 sonar sonar   50 Oct 14 10:42 conf
-rw-r--r-- 1 sonar sonar 7651 Jul 10 12:21 COPYING
drwxr-xr-x 4 sonar sonar   46 Oct 14 08:11 data
drwxr-xr-x 7 sonar sonar  131 Jul 10 12:21 elasticsearch
drwxr-xr-x 5 sonar sonar   57 Oct 14 09:49 extensions
drwxr-xr-x 6 sonar sonar  127 Jul 10 12:32 lib
drwxr-xr-x 2 sonar sonar  102 Oct 14 09:50 logs
```

```
sysctl -w  vm.max_map_count=262144
sysctl -w fs.file-max=65536
ulimit -u 4096 sonar
ulimit -n 65536 sonar
```

2.5、更改配置文件

```
# 修改配置文件
vim ./conf/sonar.properties
```

2.5.1、配置数据库登录用户和密码

```
sonar.jdbc.username=sonar
sonar.jdbc.password=123456
```

```
#--------------------------------------------------------------------------------------------------
# DATABASE
#
# IMPORTANT:
# - The embedded H2 database is used by default. It is recommended for tests but not for
#   production use. Supported databases are Oracle, PostgreSQL and Microsoft SQLServer.
# - Changes to database connection URL (sonar.jdbc.url) can affect SonarSource licensed products.

# User credentials.
# Permissions to create tables, indices and triggers must be granted to JDBC user.
# The schema must be created first.
sonar.jdbc.username=sonar
sonar.jdbc.password=123456

#----- Embedded Database (default)
# H2 embedded database server listening port, defaults to 9092
#sonar.embeddedDatabase.port=9092


#----- Oracle 11g/12c/18c/19c
# The Oracle JDBC driver must be copied into the directory extensions/jdbc-driver/oracle/.
# Only the thin client is supported, and we recommend using the latest Oracle JDBC driver. See
# https://jira.sonarsource.com/browse/SONAR-9758 for more details.
# If you need to set the schema, please refer to http://jira.sonarsource.com/browse/SONAR-5000
#sonar.jdbc.url=jdbc:oracle:thin:@localhost:1521/XE


#----- PostgreSQL 9.3 or greater
# By default the schema named "public" is used. It can be overridden with the parameter "currentSchema".
sonar.jdbc.url=jdbc:postgresql://127.0.0.1/sonar?currentSchema=my_schema
```

2.5.2、配置数据库链接池相关属性

```
sonar.jdbc.maxActive=60
sonar.jdbc.maxIdle=5
sonar.jdbc.minIdle=2
sonar.jdbc.maxWait=5000
sonar.jdbc.minEvictableIdleTimeMillis=600000
sonar.jdbc.timeBetweenEvictionRunsMillis=30000
```

```
#----- Connection pool settings
# The maximum number of active connections that can be allocated
# at the same time, or negative for no limit.
# The recommended value is 1.2 * max sizes of HTTP pools. For example if HTTP ports are
# enabled with default sizes (50, see property sonar.web.http.maxThreads)
# then sonar.jdbc.maxActive should be 1.2 * 50 = 60.
sonar.jdbc.maxActive=60

# The maximum number of connections that can remain idle in the
# pool, without extra ones being released, or negative for no limit.
sonar.jdbc.maxIdle=5

# The minimum number of connections that can remain idle in the pool,
# without extra ones being created, or zero to create none.
sonar.jdbc.minIdle=2

# The maximum number of milliseconds that the pool will wait (when there
# are no available connections) for a connection to be returned before
# throwing an exception, or <= 0 to wait indefinitely.
sonar.jdbc.maxWait=5000

sonar.jdbc.minEvictableIdleTimeMillis=600000
```

2.5.3、配置web访问相关

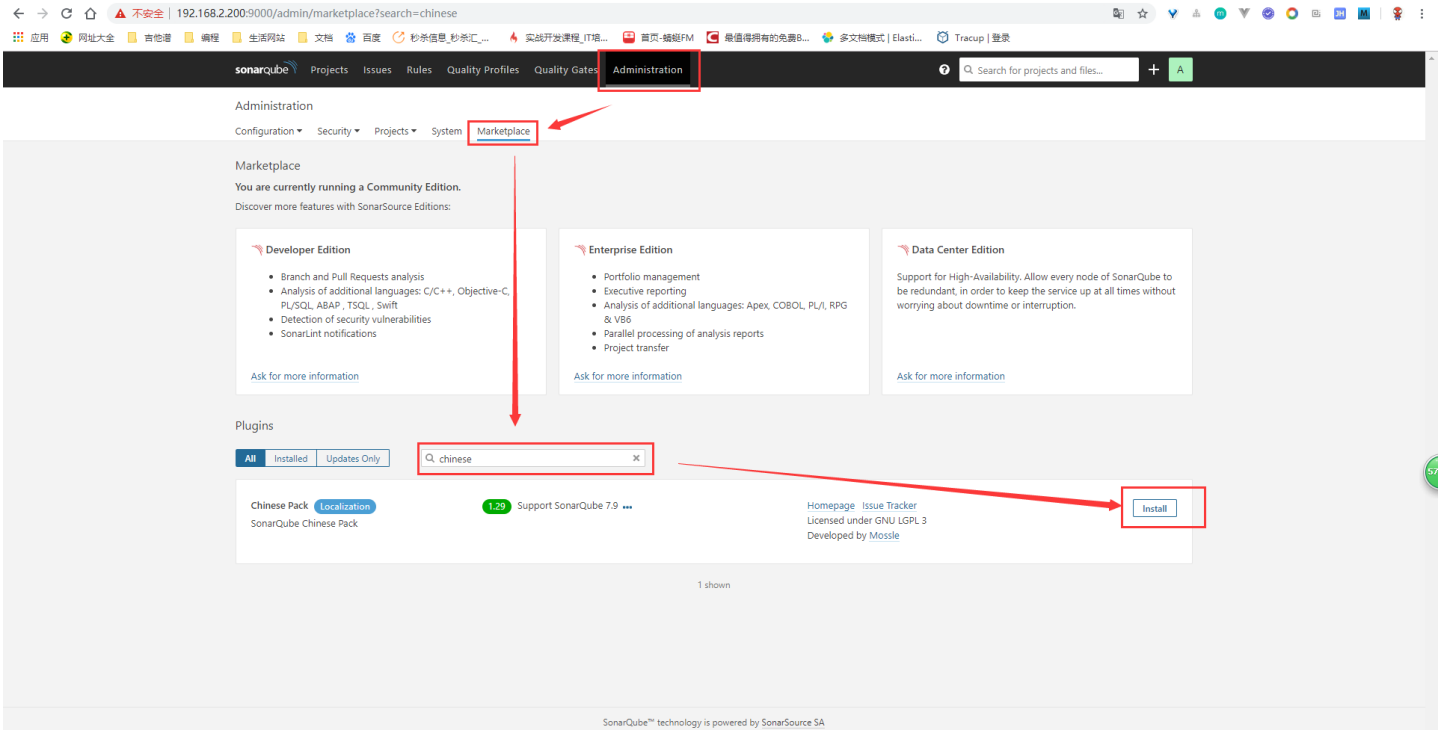
```
sonar.web.host=0.0.0.0
sonar.web.port=9000
```

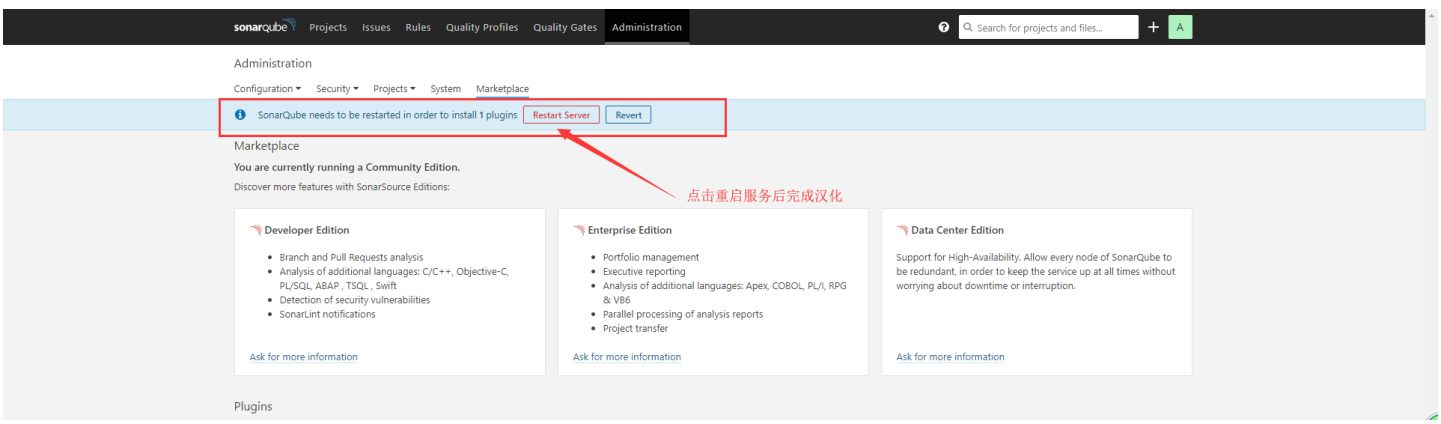
```
# Web server is executed in a dedicated Java process. By default heap size is 512MB.
# Use the following property to customize JVM options.
#    Recommendations:
#
#    The HotSpot Server VM is recommended. The property -server should be added if server mode
#    is not enabled by default on your environment:
#    http://docs.oracle.com/javase/8/docs/technotes/guides/vm/server-class.html
#
#    Startup can be long if entropy source is short of entropy. Adding
#    -Djava.security.egd=file:/dev/./urandom is an option to resolve the problem.
#    See https://wiki.apache.org/tomcat/HowTo/FasterStartUp#Entropy_Source
#
#sonar.web.javaOpts=-Xmx512m -Xms128m -XX:+HeapDumpOnOutOfMemoryError

# Same as previous property, but allows to not repeat all other settings like -Xmx
#sonar.web.javaAdditionalOpts=

# Binding IP address. For servers with more than one IP address, this property specifies which
# address will be used for listening on the specified ports.
# By default, ports will be used on all IP addresses associated with the server.
sonar.web.host=0.0.0.0

# Web context. When set, it must start with forward slash (for example /sonarqube).
# The default value is root context (empty value).
#sonar.web.context=
# TCP port for incoming HTTP connections. Default value is 9000.
```

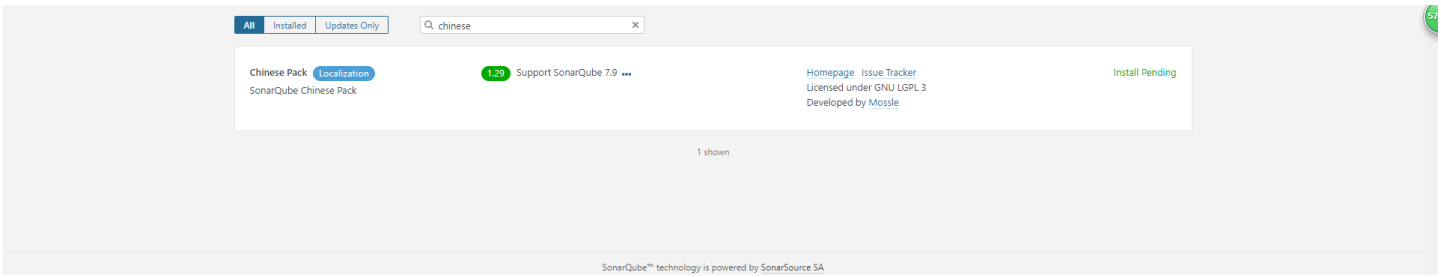
2.6、启动SonarQube（需要2G内存）

```
./bin/linux-x86-64/sonar.sh start
```

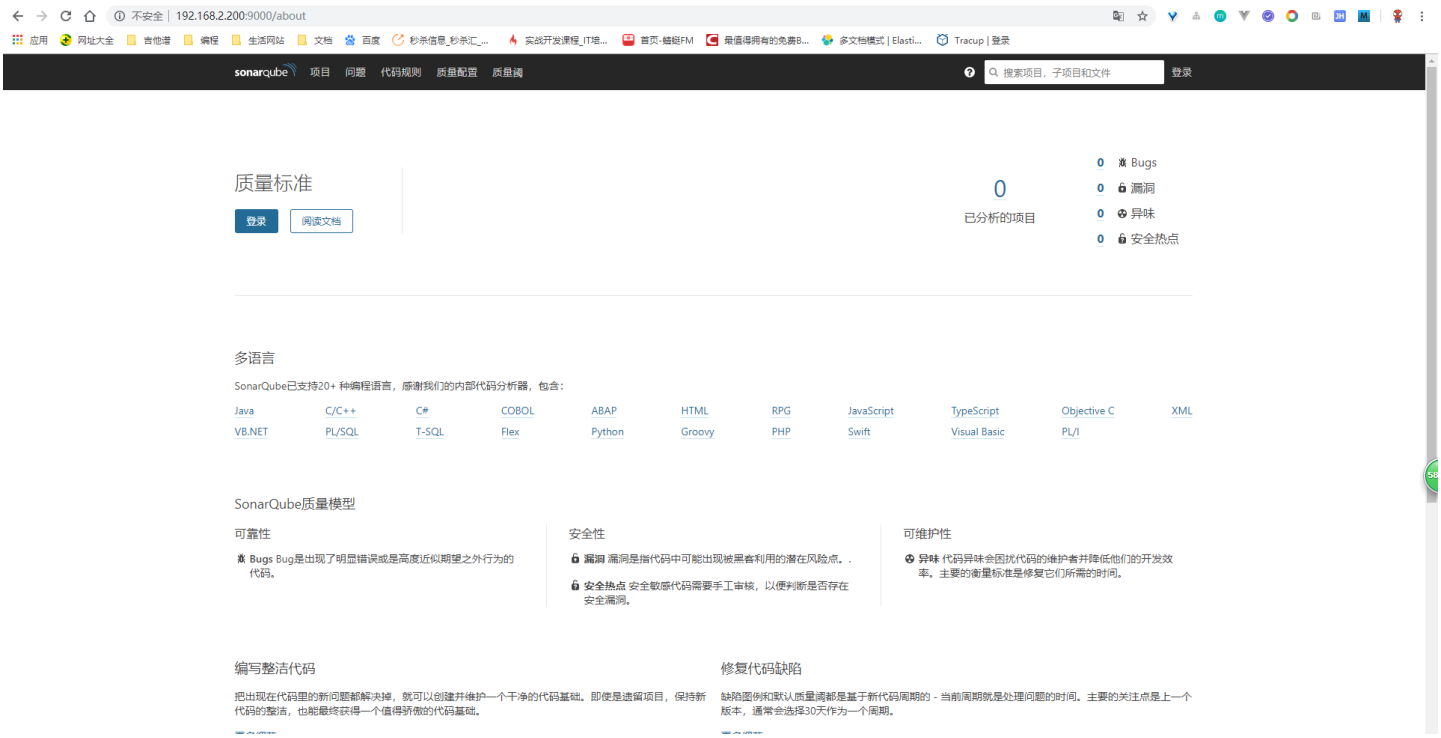
2.7、SonarQube汉化插件的安装







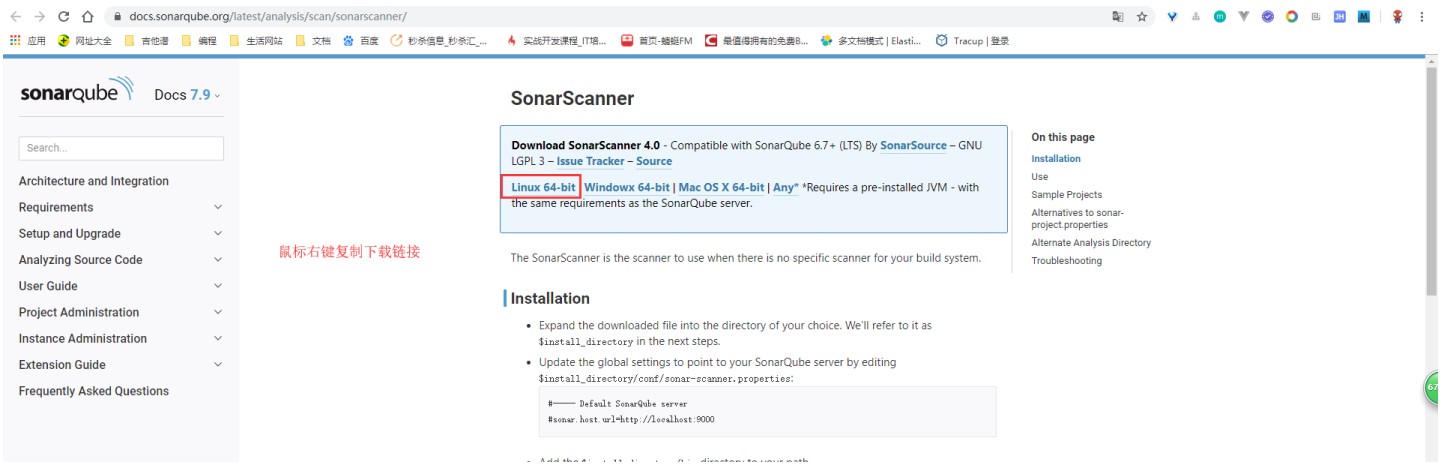
等待服务重启后，刷新页面即可如下汉化版本



2.8、sonar-scanner插件的安装（这个插件按的安装和下面的在Jenkins部分配置的插件可能相同，这部分没有做再次测试）

标题处的描述，主要是由于此处是在200主机（sonar安装的主机）上安装一个sonar-scanner，下面再Jenkins的sonar-scanner服务的配置的时候，又选择了自动安装。理论上这里安装的应该是同一个东西，但是笔者在测试的时候，发现自己安装的虽然在填写sonar-scanner的home目录填写没有问题，但是Jenkins的job构建的时候还是出现了错误。

2.8.1、sonar-scanner的下载与解压



鼠标右键复制下载链接

```
# 进入software文件夹(该文件夹为自己创建用来存放软件的zip安装包)
cd ./software
# 下载sonar-scanner插件zip包
wget https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.0.0.1744-linux.zip
# 将该zip文件解压到opt文件夹下
unzip sonar-scanner-cli-4.0.0.1744-linux.zip -d /opt
# 进入到opt文件夹，然后修改解压后的文件夹名称为sonar-scanner
mv sonar-scanner-4.0.0.1744-linux sonar-scanner
```

2.8.2、sonar-scanner的环境变量配置

```
# 配置环境变量
vim /etc/profile
```

```
export SONAR_SCANNER_HOME=/opt/sonar-scanner
export PATH=$PATH:${SONAR_SCANNER_HOME}/bin
```

```
# By default, we want umask to get set. This sets it for login shell
# Current threshold for system reserved uid/gids is 200
# You could check uidgid reservation validity in
# /usr/share/doc/setup-*/uidgid file
if [ $UID -gt 199 ] && [ "`/usr/bin/id -gn`" = "`/usr/bin/id -un`" ]; then
    umask 002
else
    umask 022
fi

for i in /etc/profile.d/*.sh ; do
    if [ -r "$i" ]; then
        if [ "${-#*i}" != "$-" ]; then
            . "$i"
        else
            . "$i" >/dev/null
        fi
    fi
done

unset i
unset -f pathmunge
export PATH=$PATH:/usr/local/nginx/sbin
export PATH=$PATH:/usr/local/node/bin

export JAVA_HOME=/usr/lib/jvm/java-11-openjdk-11.0.4.11-1.el7_7.x86_64/bin/java
export JRE_HOME=$JAVA_HOME/jre
export CLASSPATH=$JAVA_HOME/lib:$JRE_HOME/lib:$CLASSPATH
export PATH=$JAVA_HOME/bin:$JRE_HOME/bin:$PATH

export PATH=/usr/local/git/bin:$PATH

ulimit -n 65536


export SONAR_SCANNER_HOME=/opt/sonar-scanner
export PATH=$PATH:${SONAR_SCANNER_HOME}/bin
```

保存后执行下面的命令，使配置生效！

```
# 使配置生效
source /etc/profile
```

2.8.3、配置sonar-scanner

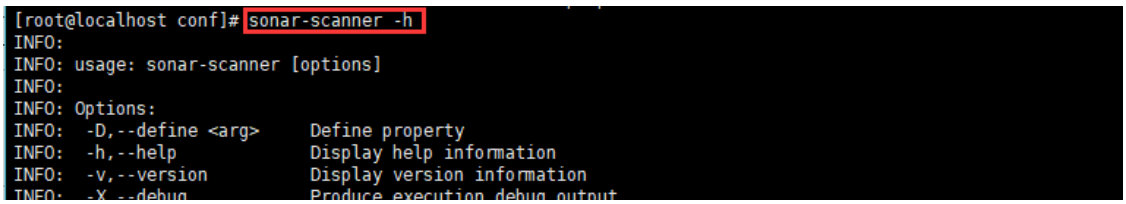
```
vim /opt/sonar-scanner/conf/sonar-scanner.properties
```

```
#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://localhost:9000

#----- Default source code encoding
```

2.8.4、验证sonar-scanner

```
sonar-scanner -h
```

执行后看到下图，说明配置成功了。

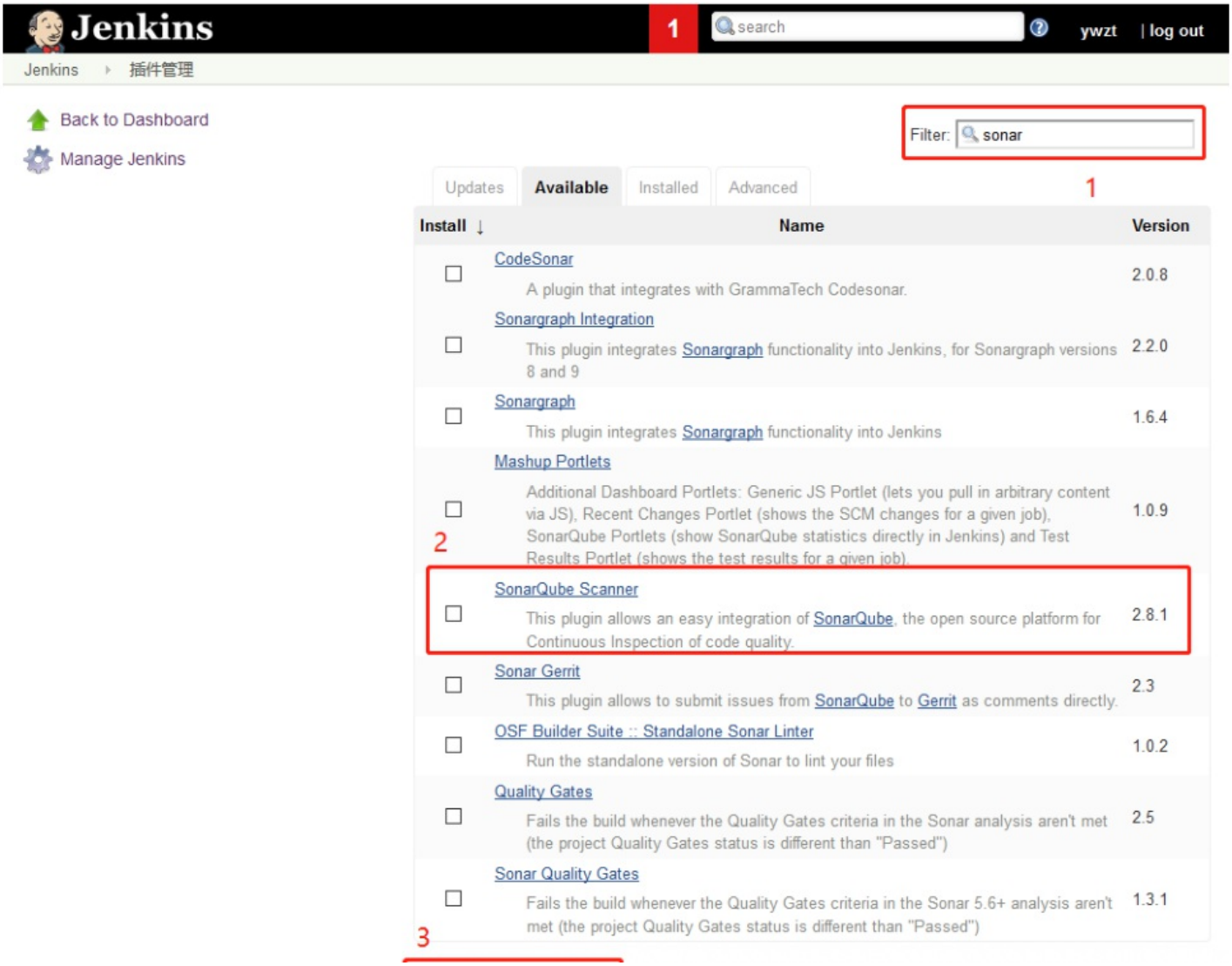


三、GitLab的安装

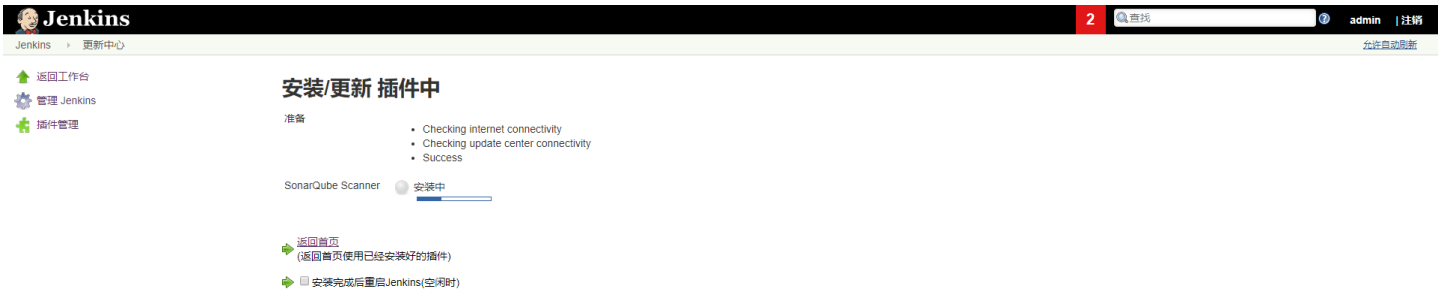
四、Jenkins的安装

五、Jenkins集成SonarQube

5.1、安装SonarQube Scanner插件

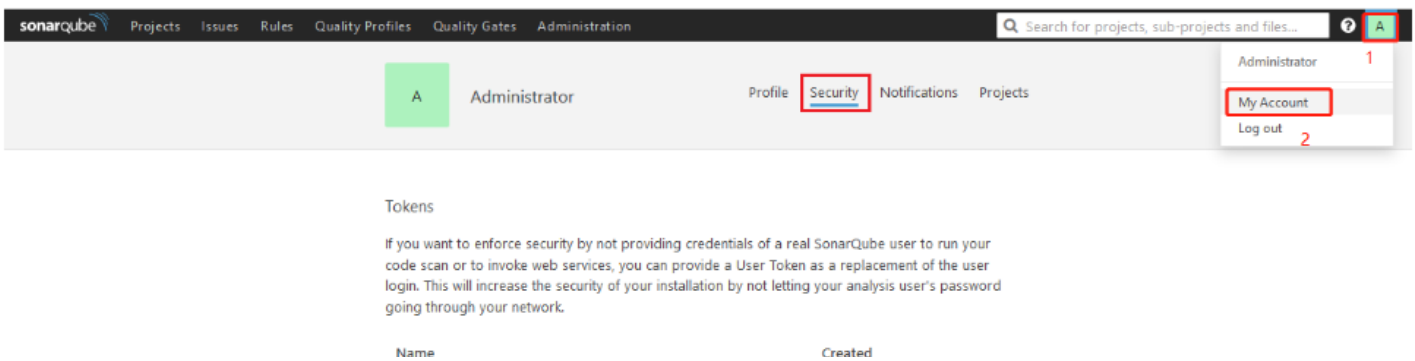


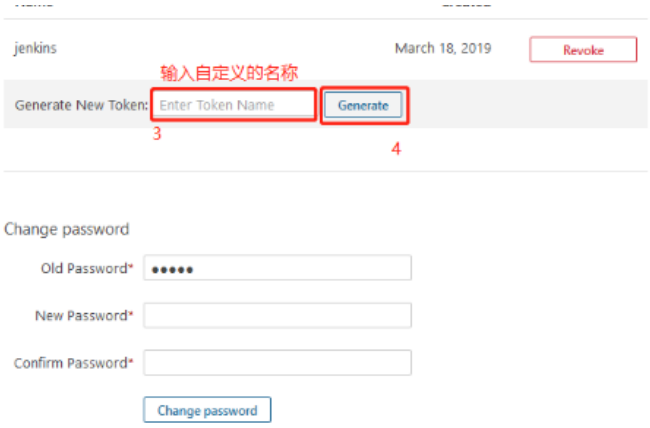
静静等待插件的安装。



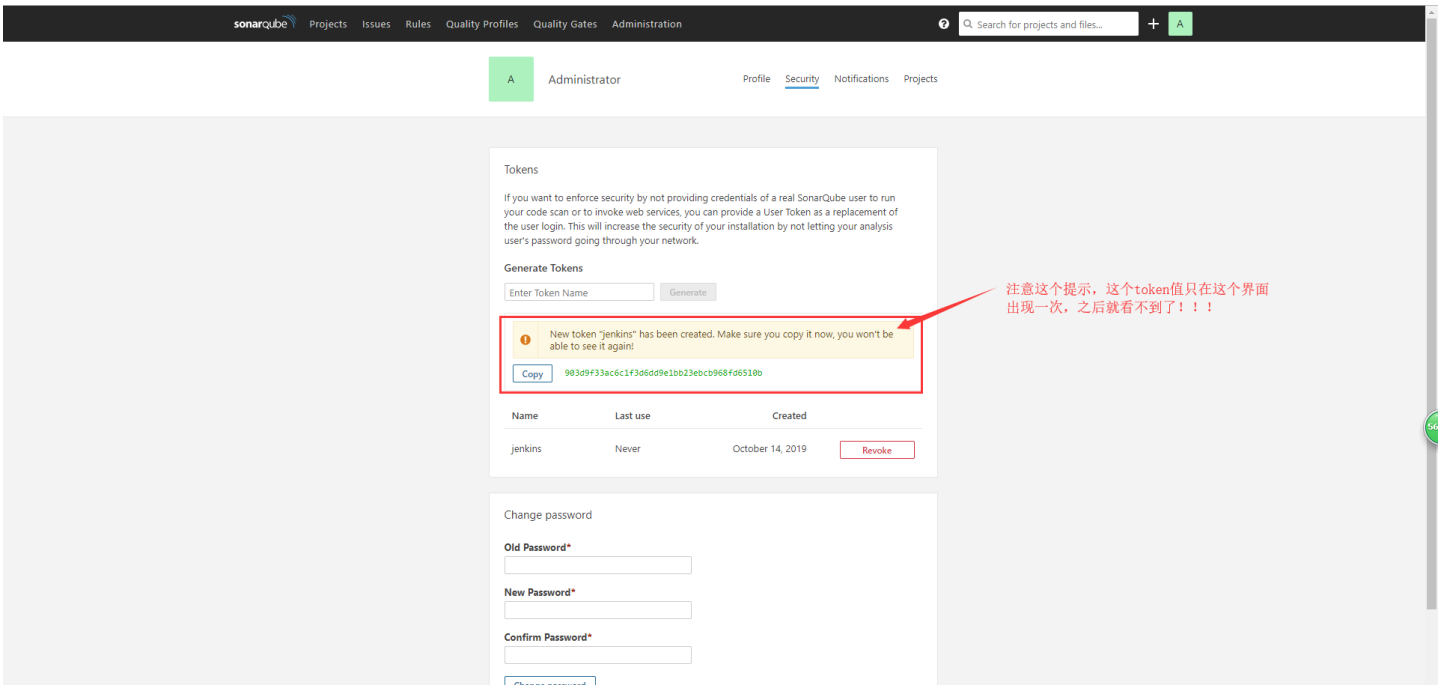
5.2、jenkins配置sonar服务信息

登录到sonar的管理平台，如下图所示操作



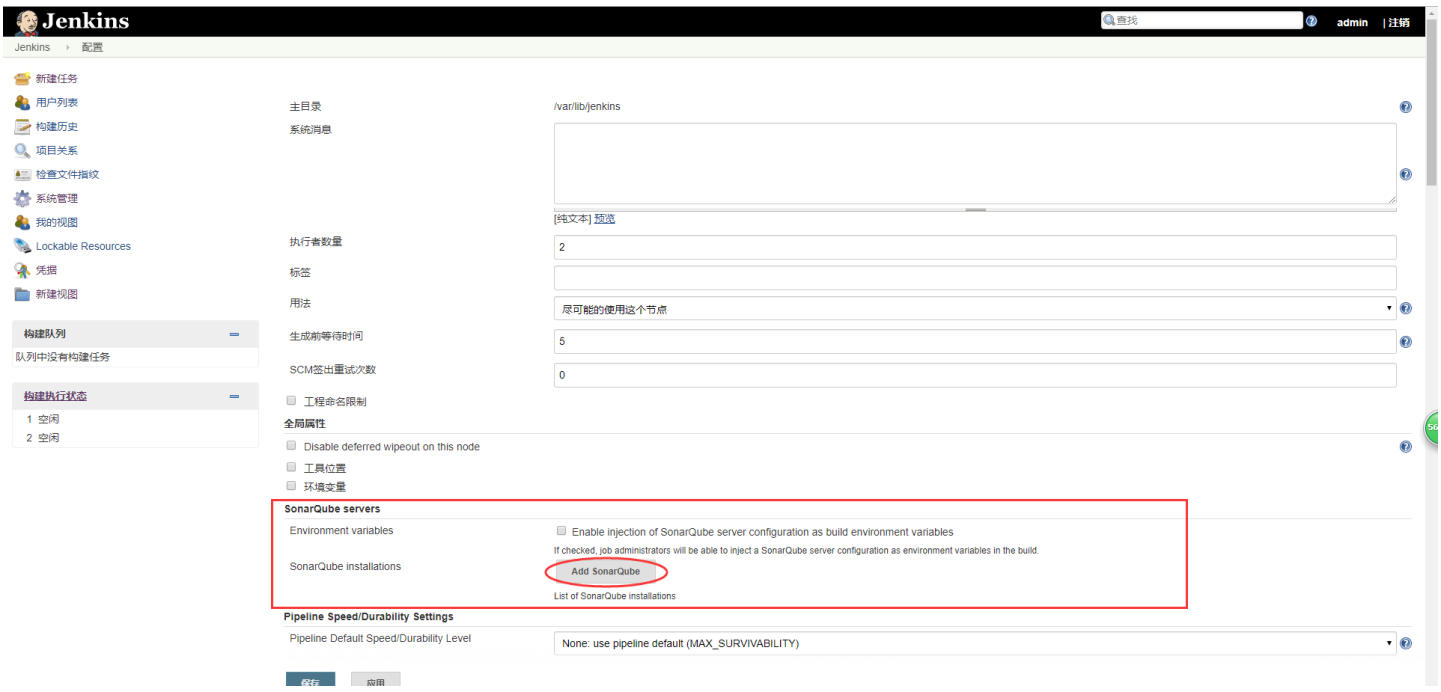


获取创建的用户的token值。

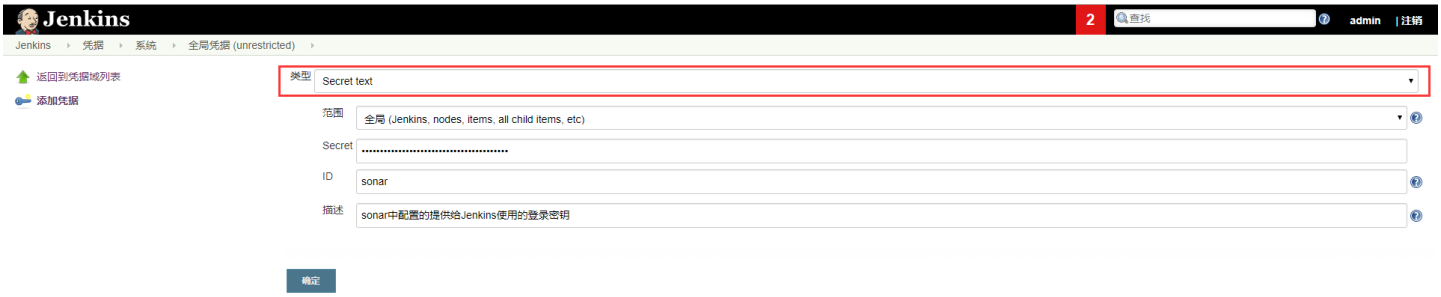


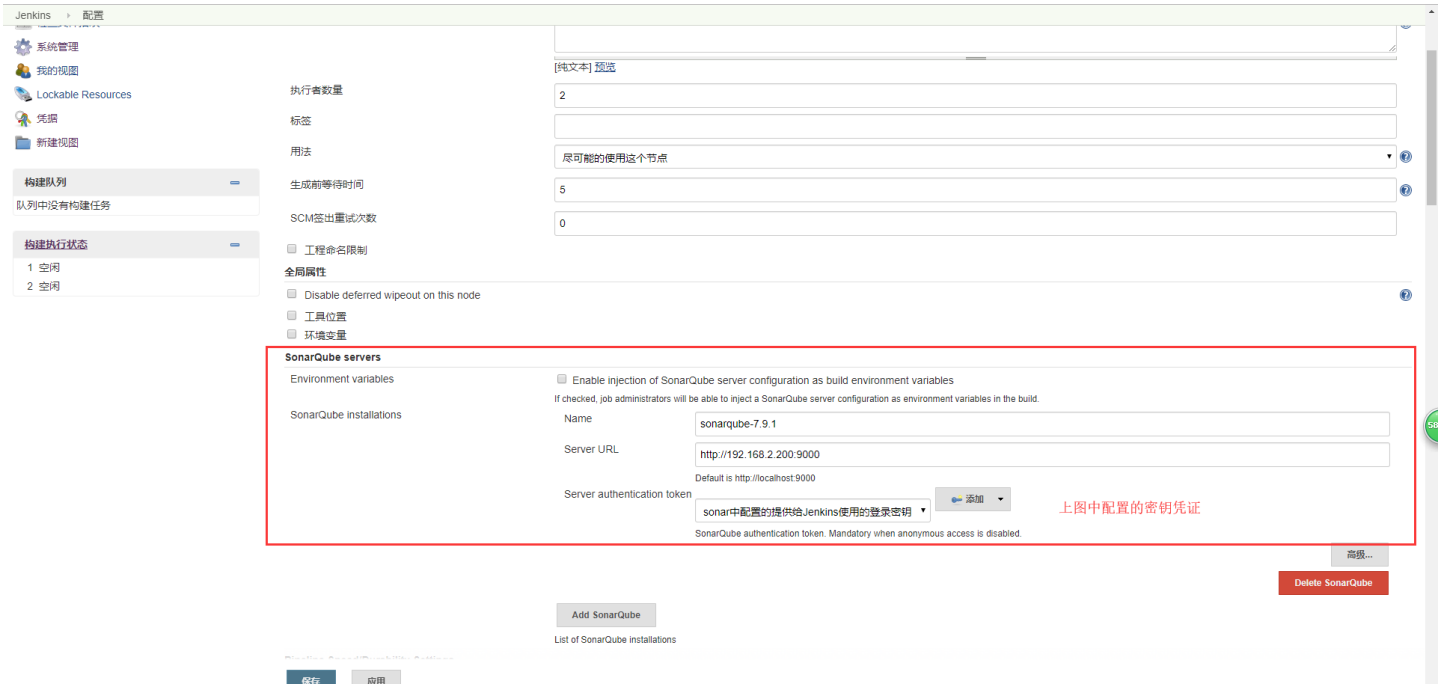
token：903d9f33ac6c1f3d6dd9e1bb23ebcb968fd6510b

接着登录到Jenkins的管理平台，在系统管理中的系统设置下找到如下图所示的部分，然后点击圆圈中的按钮。



配置名称和sonar主机的访问IP地址。

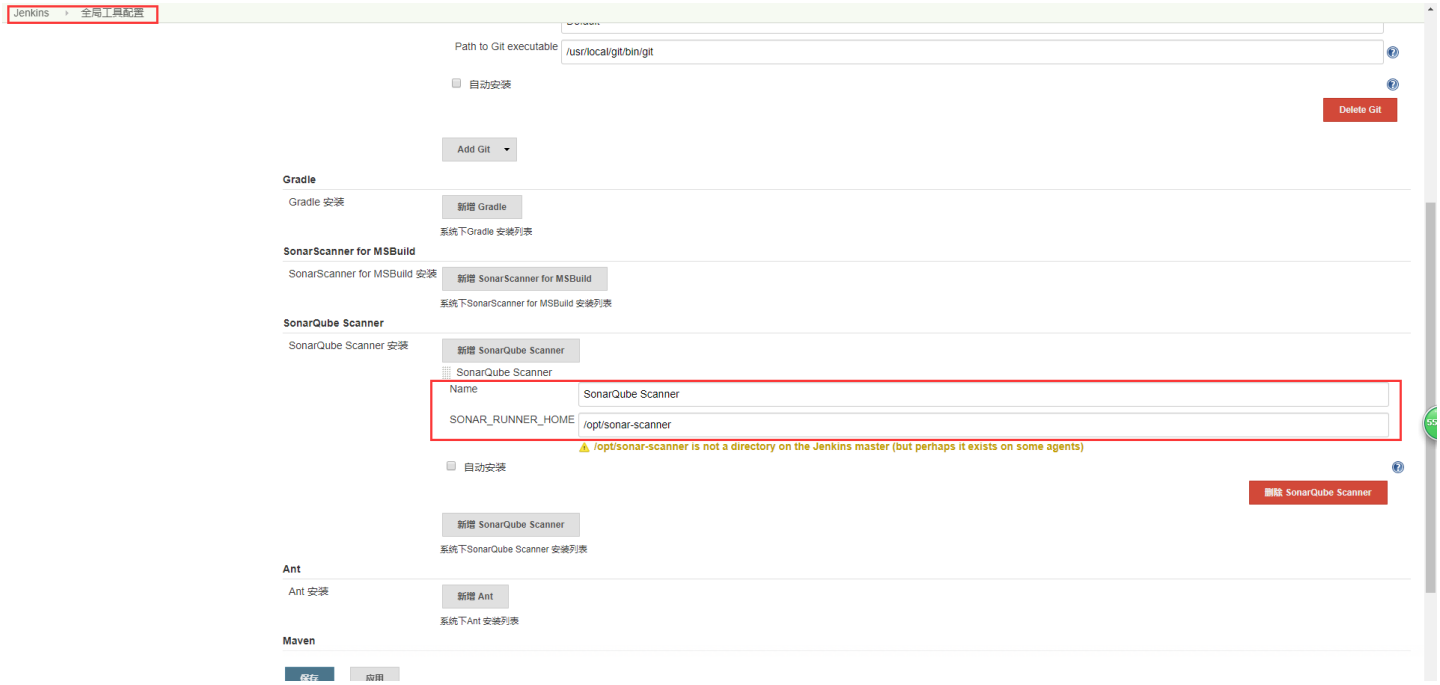




5.3、Jenkins配置SonarQube Scanner服务信息

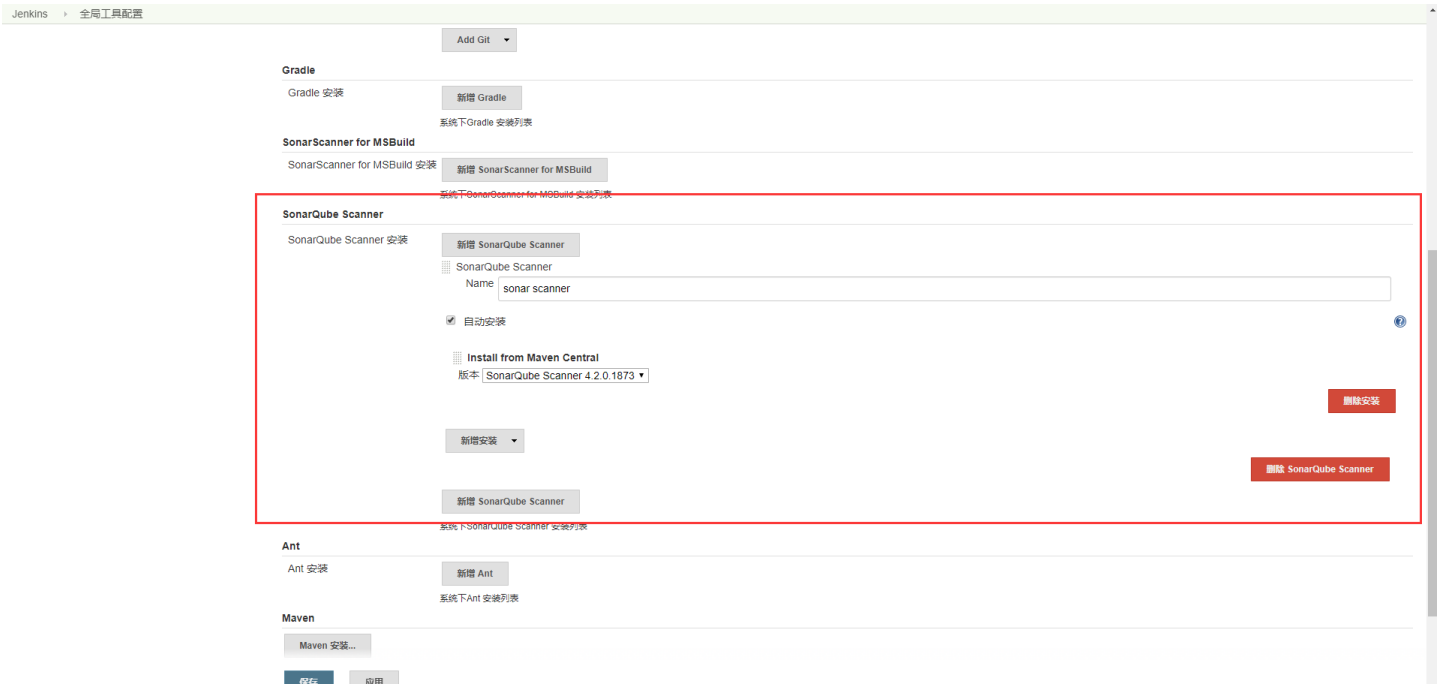
5.3.1、自己安装的SonarQube Scanner的配置

这里需要注意的是，当前我的sonar是安装在192.168.2.200的主机上，而Jenkins是在192.168.2.203。这里添加了SONAR_RUNNER_HOME，也会报黄色警告，说明不在当前的主机中，没关系。（此处的配置即上面红色文字处描述的问题内容）



5.3.2、使用Jenkins自动安装SonarQube Scanner的配置

登录到Jenkins的管理平台，然后再如下图配置。



5.4、Maven集成SonarQube

登录到Jenkins的主机203，然后找到该主机下的maven

```
# 进入到maven的安装目录
cd /opt/apache-maven-3.6.2/
```

5.4.1、设置插件前缀

```
vim conf/settings.xml
```

```
<settings>
  <pluginGroups>
    <pluginGroup>org.sonarsource.scanner.maven</pluginGroup>
  </pluginGroups>
</settings>
```

```
  <!-- pluginGroups
   | This is a list of additional group identifiers that will be searched when resolving plugins by their prefix, i.e.
   | when invoking a command line like "mvn prefix:goal". Maven will automatically add the group identifiers
   | "org.apache.maven.plugins" and "org.codehaus.mojo" if these are not already contained in the list.
   |-->
  <pluginGroups>
    <!-- pluginGroup
     | Specifies a further group identifier to use for plugin lookup.
    <pluginGroup>com.your.plugins</pluginGroup>
    -->
    <pluginGroup>org.sonarsource.scanner.maven</pluginGroup>
```

5.4.2、配置sonar的服务器URL

```
<settings>
  <profiles>
    <profile>
      <id>sonar</id>
      <activation>
        <activeByDefault>true</activeByDefault>
      </activation>
      <properties>
        <!-- Optional URL to server. Default value is http://localhost:9000 -->
        <sonar.host.url>
          http://192.168.2.200:9000
        </sonar.host.url>
      </properties>
    </profile>
  </profiles>
</settings>
```

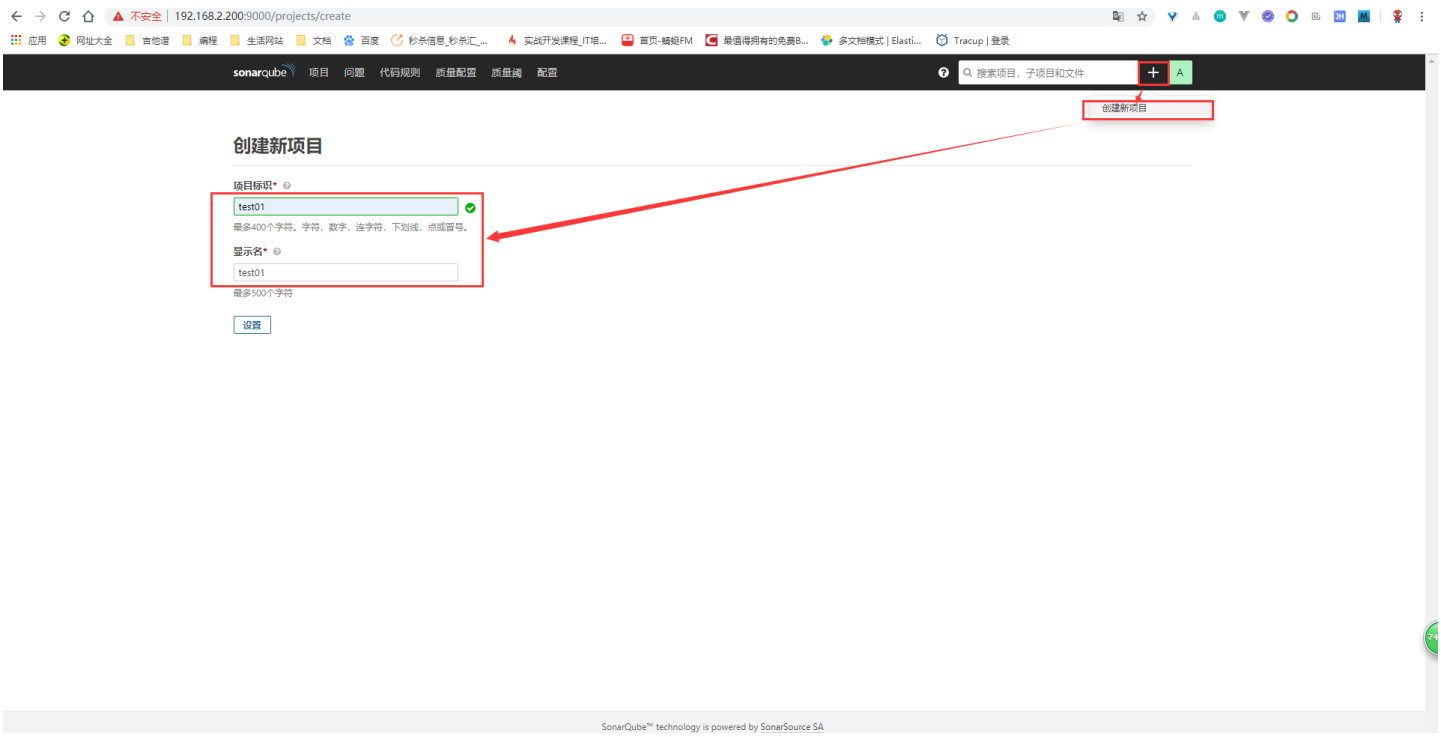
```
      <properties>
        <tomcatPath>/path/to/tomcat/instance</tomcatPath>
      </properties>
    </profile>
    -->
        <profile>
            <id>sonar</id>
            <activation>
                <activeByDefault>true</activeByDefault>
            </activation>
            <properties>
                <!-- Optional URL to server. Default value is http://localhost:9000 -->
                <sonar.host.url>
                  http://192.168.2.200:9000
                </sonar.host.url>
            </properties>
        </profile>
  </profiles>

  <!-- activeProfiles
   | List of profiles that are active for all builds.
   |
  <activeProfiles>
    <activeProfile>alwaysActiveProfile</activeProfile>
    <activeProfile>anotherAlwaysActiveProfile</activeProfile>
```

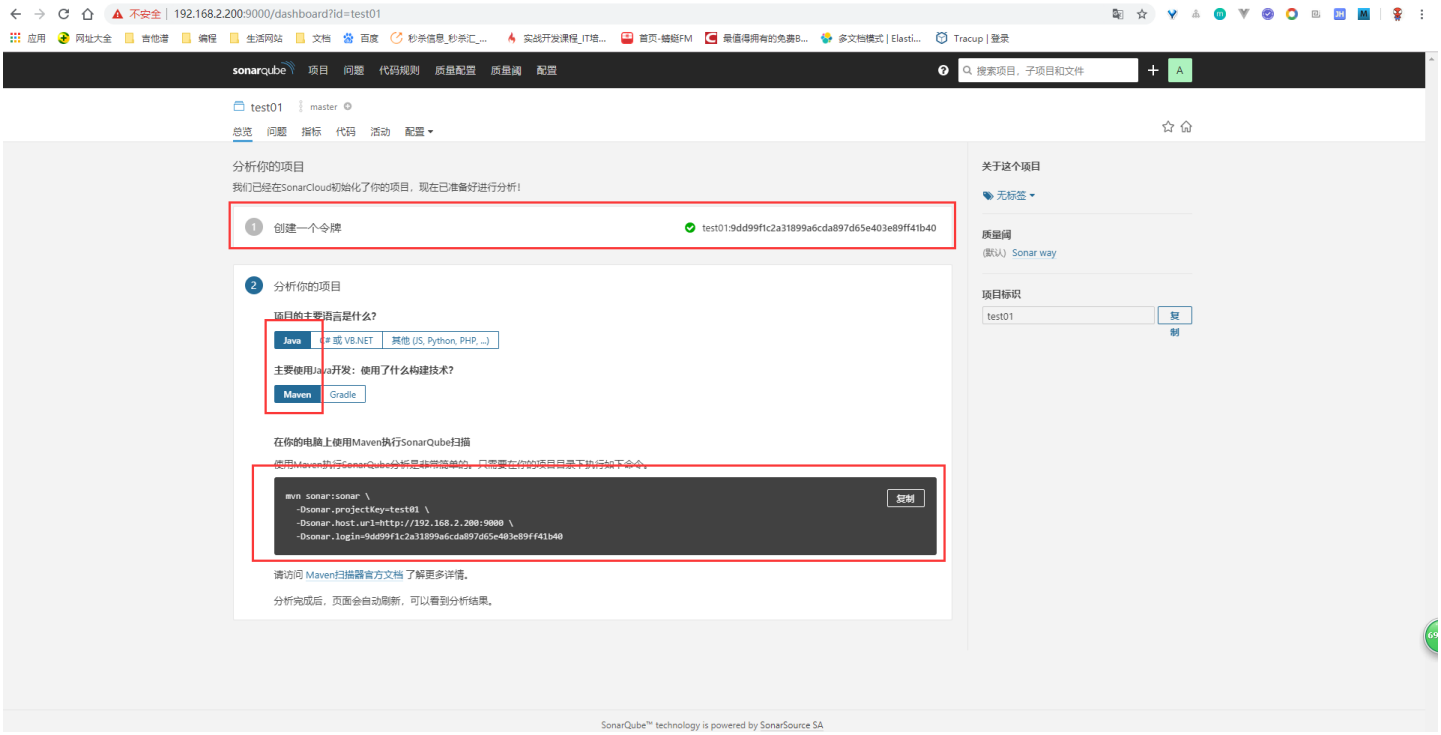
。。。。。。。。。。。。。。。。。。。未完待续。。。。。。。。。。

六、直接在IDEA中执行构建并生成代码检测报告

6.1、登录到sonarqube管理平台，新建一个项目

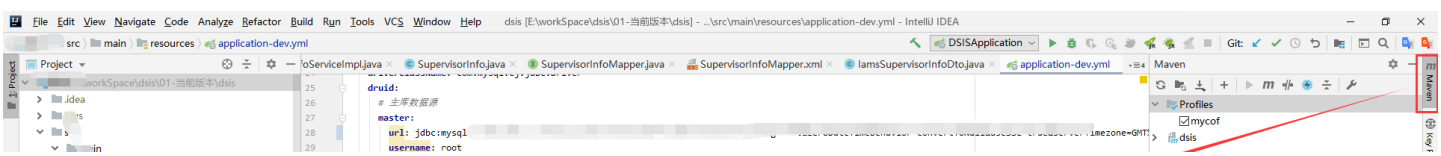
具体操作如下图所示。

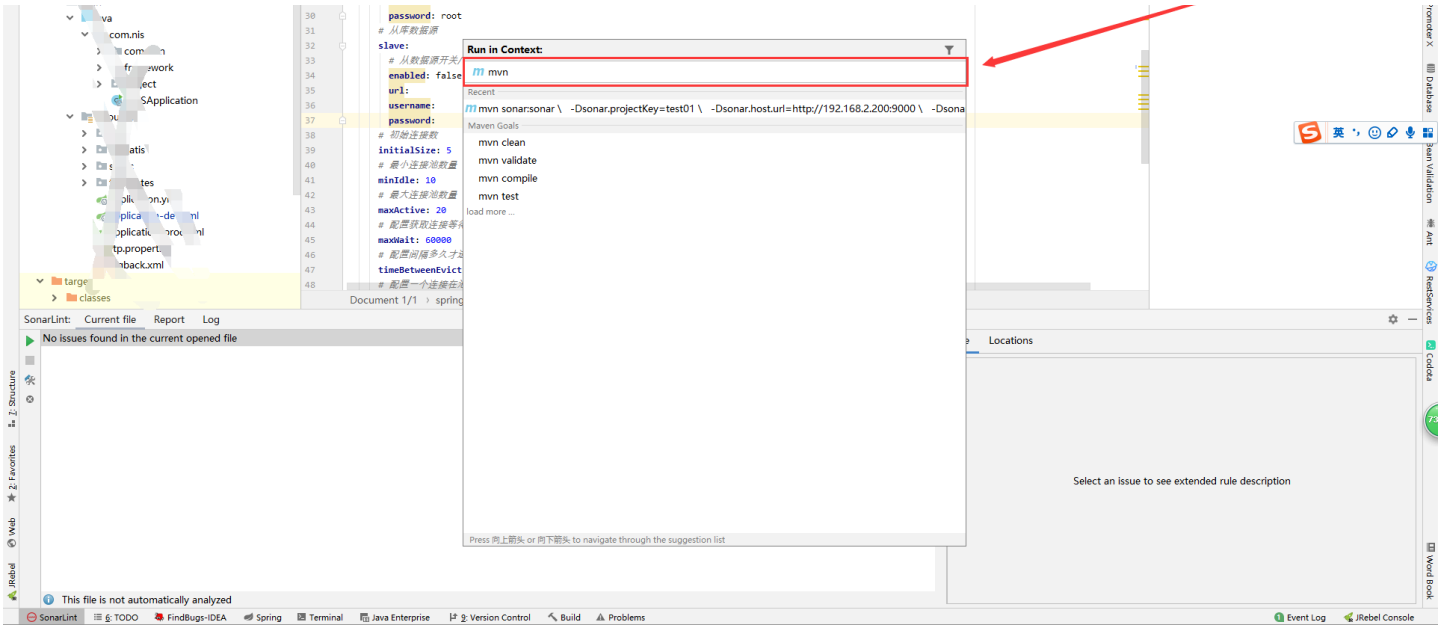


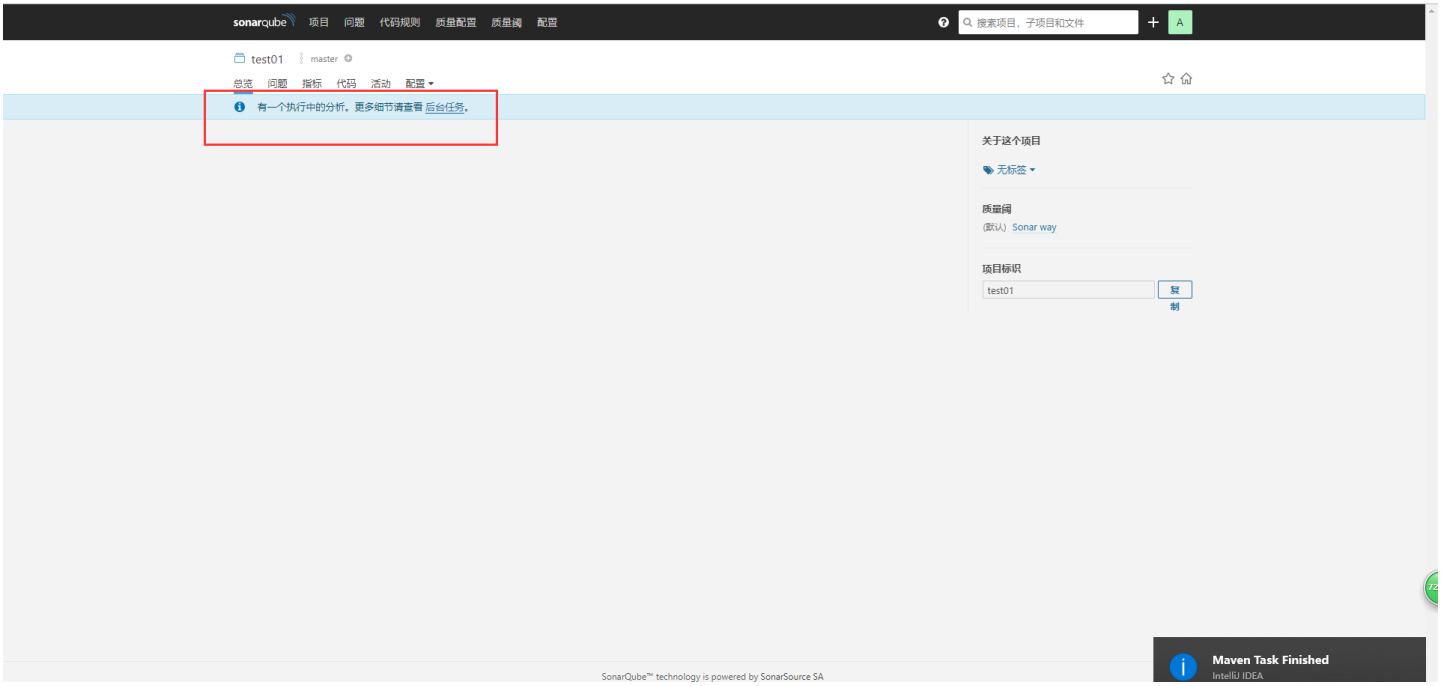


6.2、拷贝上面的**maven**执行命令到**IDEA**的测试项目中执行

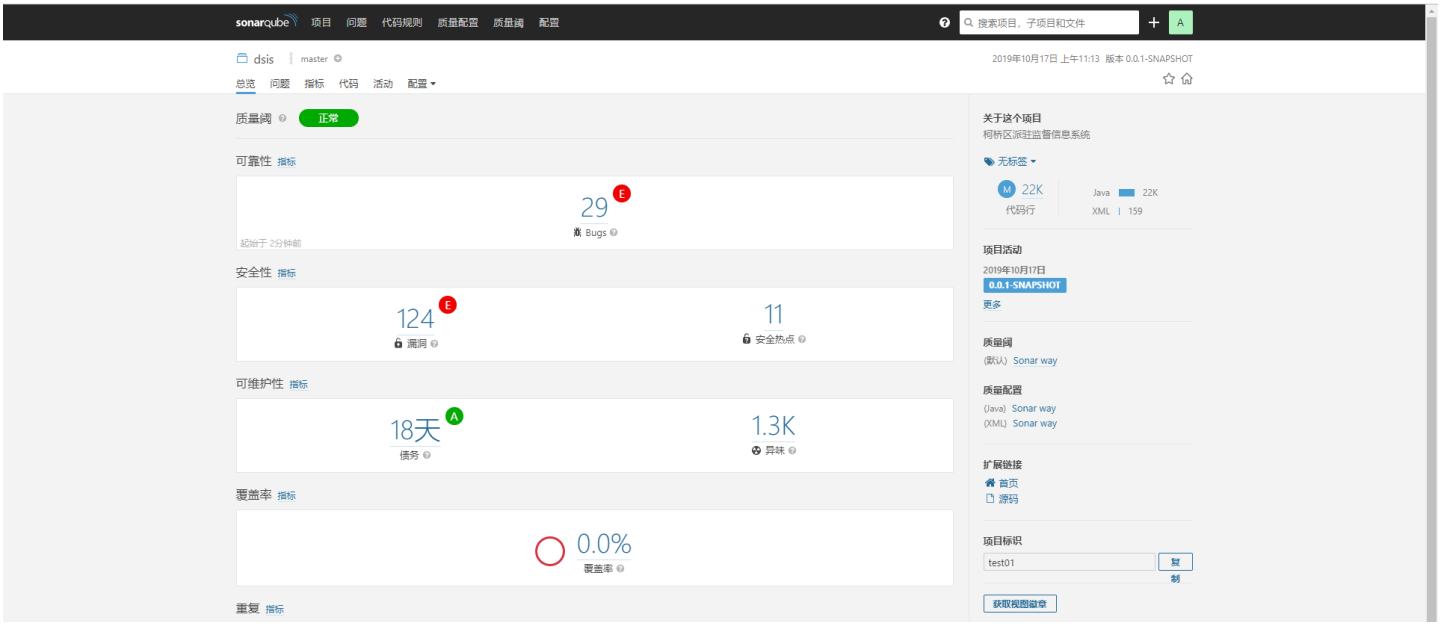
```
# 直接黏贴到idea的maven插件中执行
mvn sonar:sonar \
 -Dsonar.projectKey=test01 \
 -Dsonar.host.url=http://192.168.2.200:9000 \
 -Dsonar.login=9dd99f1c2a31899a6cda897d65e403e89ff41b40
```

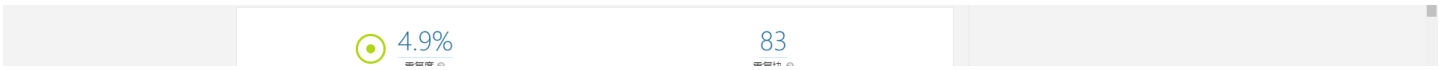






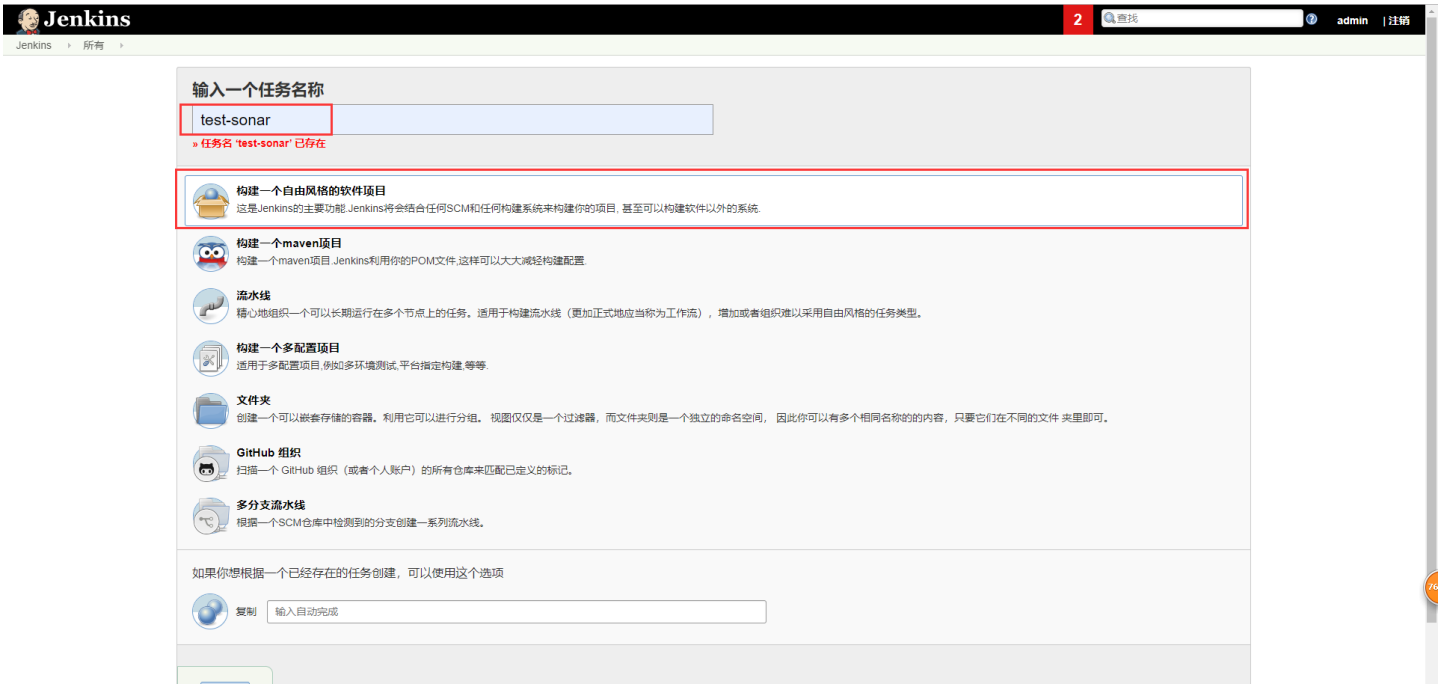
稍等之后就出现了分析图（内容省略~~~）





七、使用Jenkins的Job来构建代码扫描任务

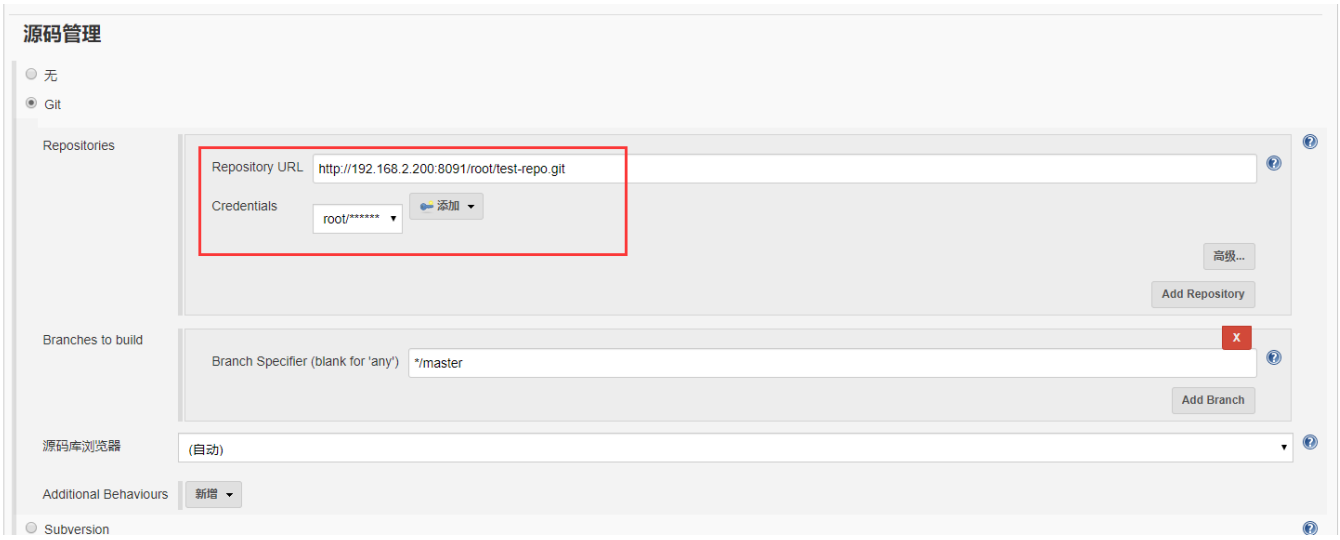
7.1、构建maven项目

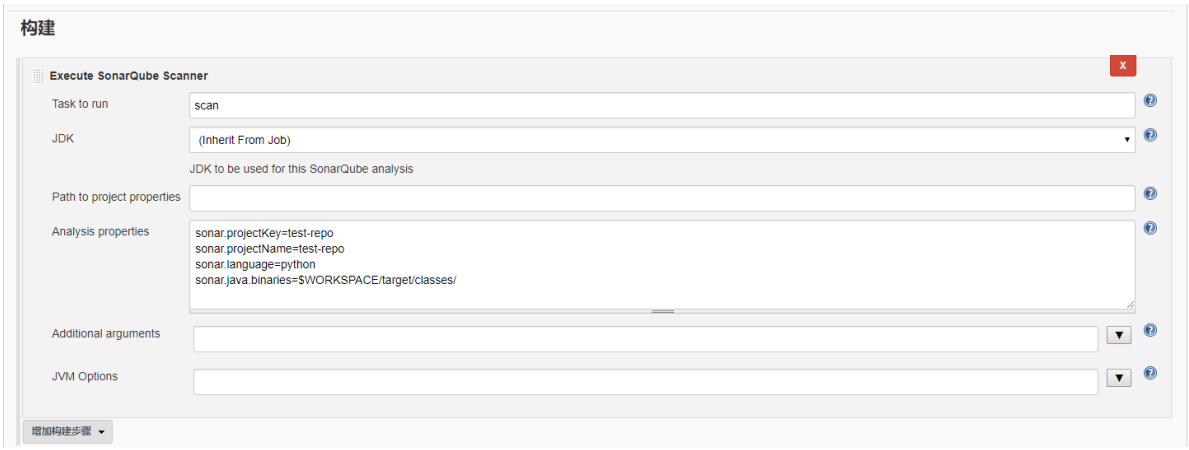


如果构建的是一个maven项目需要在配置中添加pom.xml文件的路径

7.2、构建一个自由风格的软件项目

7.2.1、创建一个项目


源码管理
无
Git
Repositories
Repository URL
http://192.168.2.200:8091/root/test-repo.git
Credentials
root/******
添加
高级...
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/master
Add Branch
源码库浏览器
(自动)
Additional Behaviours
新增
Subversion

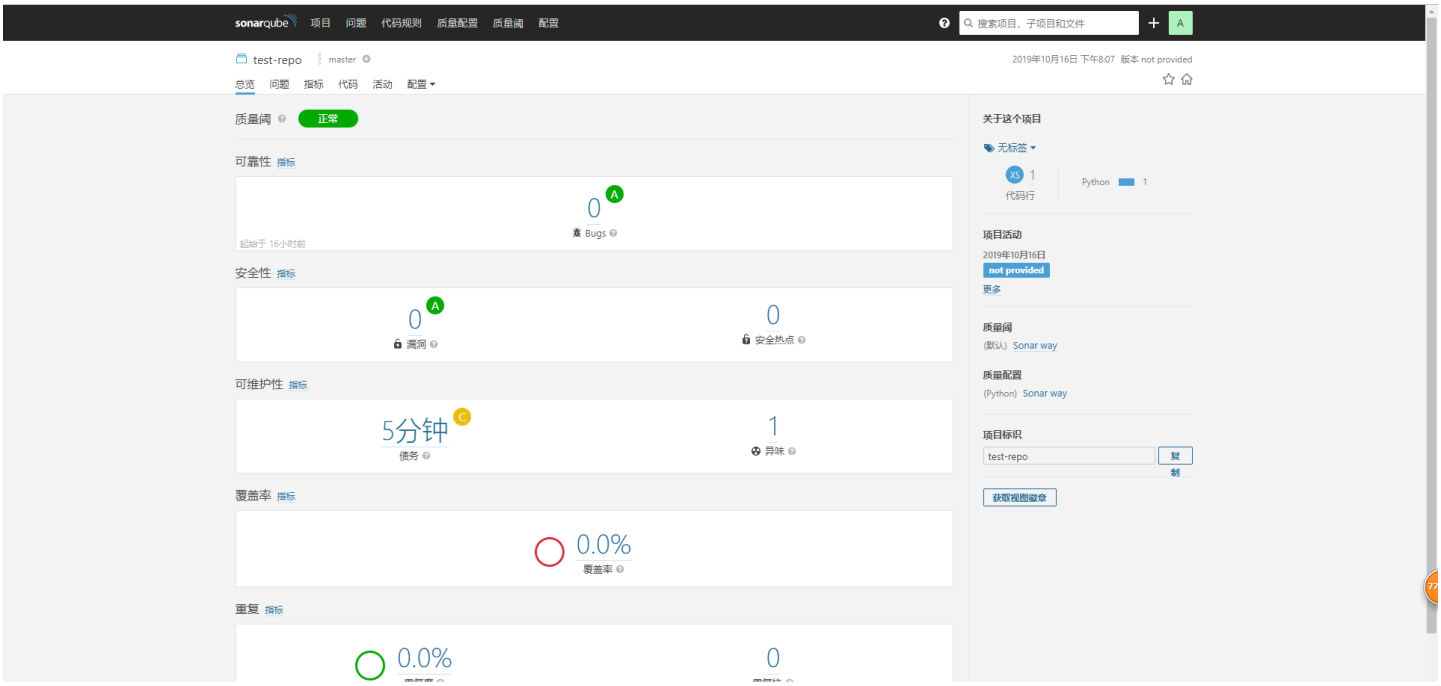

构建
Execute SonarQube Scanner
Task to run
scan
JDK
(Inherit From Job)
JDK to be used for this SonarQube analysis
Path to project properties
Analysis properties
sonar.projectKey=test-repo
sonar.projectName=test-repo
sonar.language=python
sonar.java.binaries=$WORKSPACE/target/classes/
Additional arguments
JVM Options
增加构建步骤

7.2.2、点击构建

如下为日志输出部分

```
INFO: Project configuration:
INFO: 1 file indexed
INFO: 0 files ignored because of scm ignore settings
INFO: Quality profile for py: Sonar way
INFO: ------------- Run sensors on module test-repo
INFO: Load metrics repository
INFO: Load metrics repository (done) | time=24ms
INFO: Sensor Python Squid Sensor [python]
INFO: Load project repositories
INFO: Load project repositories (done) | time=12ms
INFO: Sensor Python Squid Sensor [python] (done) | time=127ms
INFO: Sensor Cobertura Sensor for Python coverage [python]
INFO: Sensor Cobertura Sensor for Python coverage [python] (done) | time=23ms
INFO: Sensor PythonXUnitSensor [python]
INFO: Sensor PythonXUnitSensor [python] (done) | time=1ms
INFO: Sensor JaCoCo XML Report Importer [jacoco]
INFO: Sensor JaCoCo XML Report Importer [jacoco] (done) | time=2ms
INFO: Sensor JavaXmlSensor [java]
INFO: Sensor JavaXmlSensor [java] (done) | time=0ms
INFO: Sensor HTML [web]
INFO: Sensor HTML [web] (done) | time=15ms
INFO: ------------- Run sensors on project
INFO: Sensor Zero Coverage Sensor
INFO: Sensor Zero Coverage Sensor (done) | time=15ms
INFO: SCM provider for this project is: git
INFO: 1 files to be analyzed
INFO: 1/1 files analyzed
INFO: 1 file had no CPD blocks
INFO: Calculating CPD for 0 files
INFO: CPD calculation finished
INFO: Analysis report generated in 98ms, dir size=72 KB
INFO: Analysis report compressed in 8ms, zip size=10 KB
INFO: Analysis report uploaded in 386ms
INFO: ANALYSIS SUCCESSFUL, you can browse http://192.168.2.200:9000/dashboard?id=test-repo
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://192.168.2.200:9000/api/ce/task?id=AW3XvkHz_qLfNkpHgSFW
INFO: Analysis total time: 5.185 s
INFO: ------------------------------------------------------------------------
INFO: EXECUTION SUCCESS
INFO: ------------------------------------------------------------------------
INFO: Total time: 6.934s
INFO: Final Memory: 13M/32M
INFO: ------------------------------------------------------------------------
Finished: SUCCESS
```

切回到sonarqube的管理界面

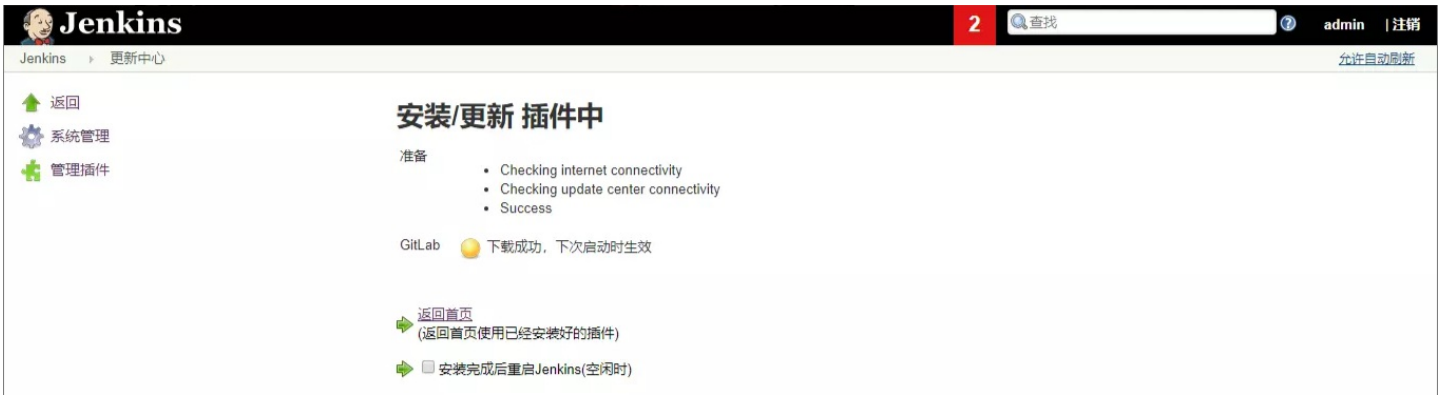


7.3、为构建的Job添加GitLab提交触发的配置

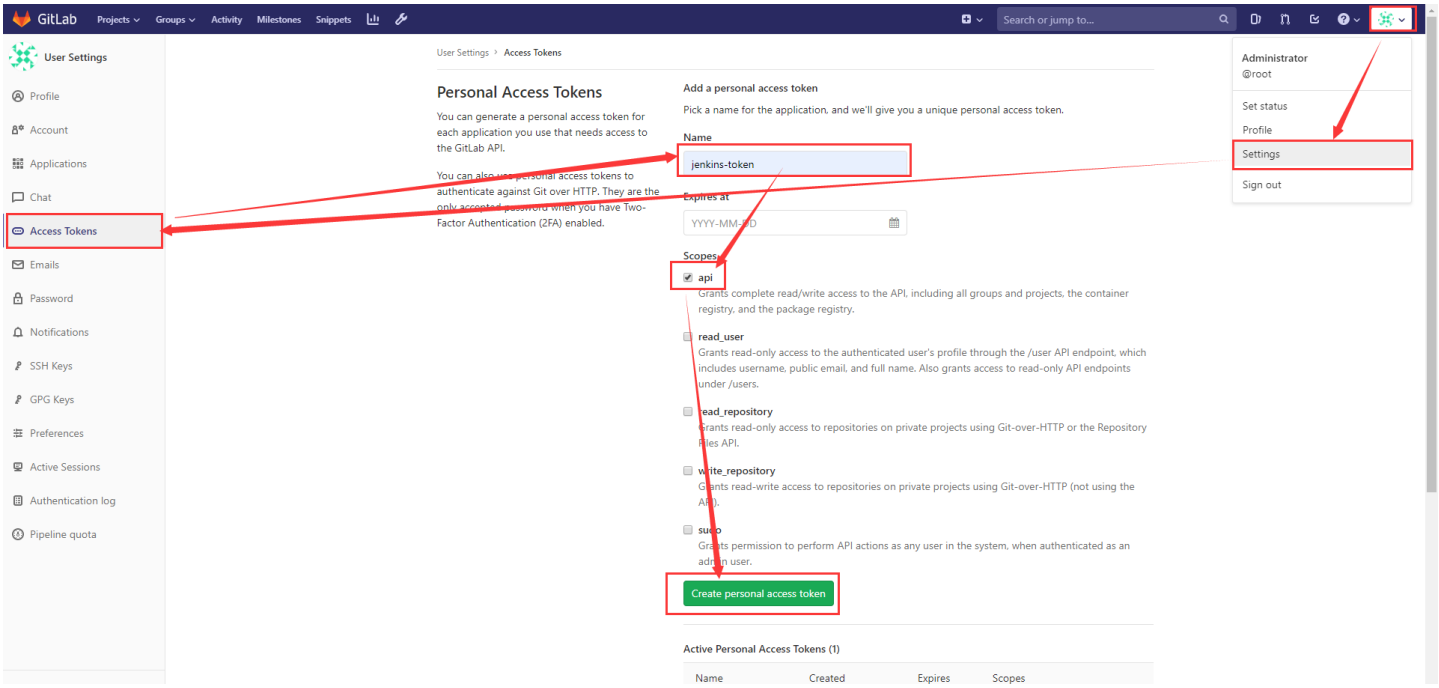
前面的7.1和7.2中已经完成了手动执行jenkins执行sonar任务完成构建部署任务，下面说明如何在代码提交后让gitlab自动触发jenkins执行sonar任务。

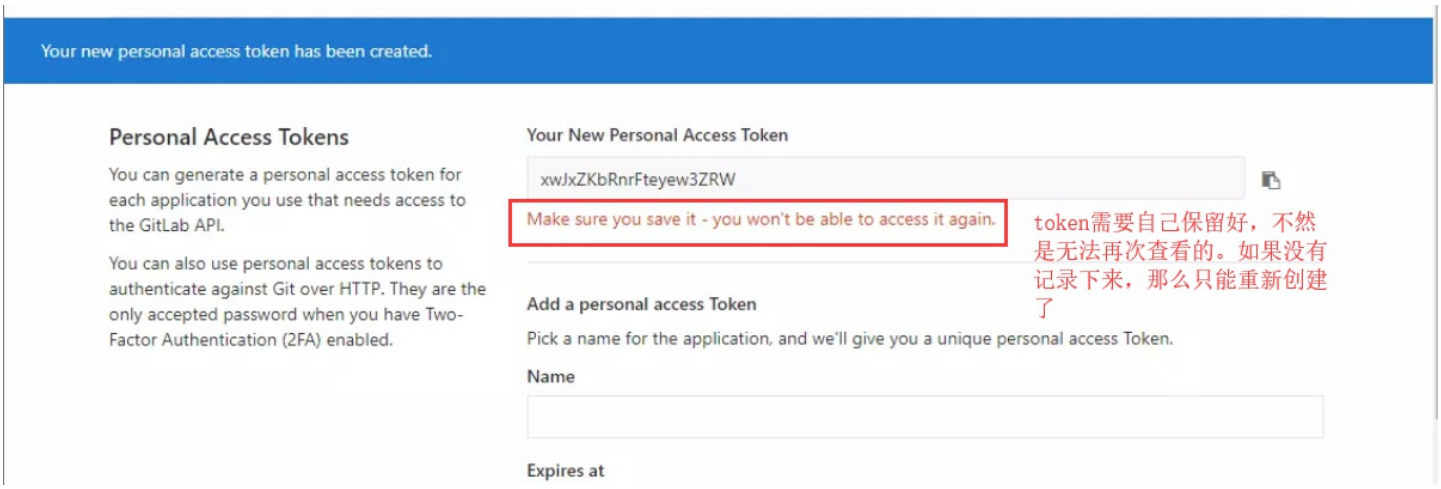
7.3.1、Jenkins中安装GitLab插件

由于我这里已经安装了GitLab的插件，所以我是在“已安装”部分做展示，不然的话是在“可选插件”中，选中红色框中的插件后，点击安装并重启即可



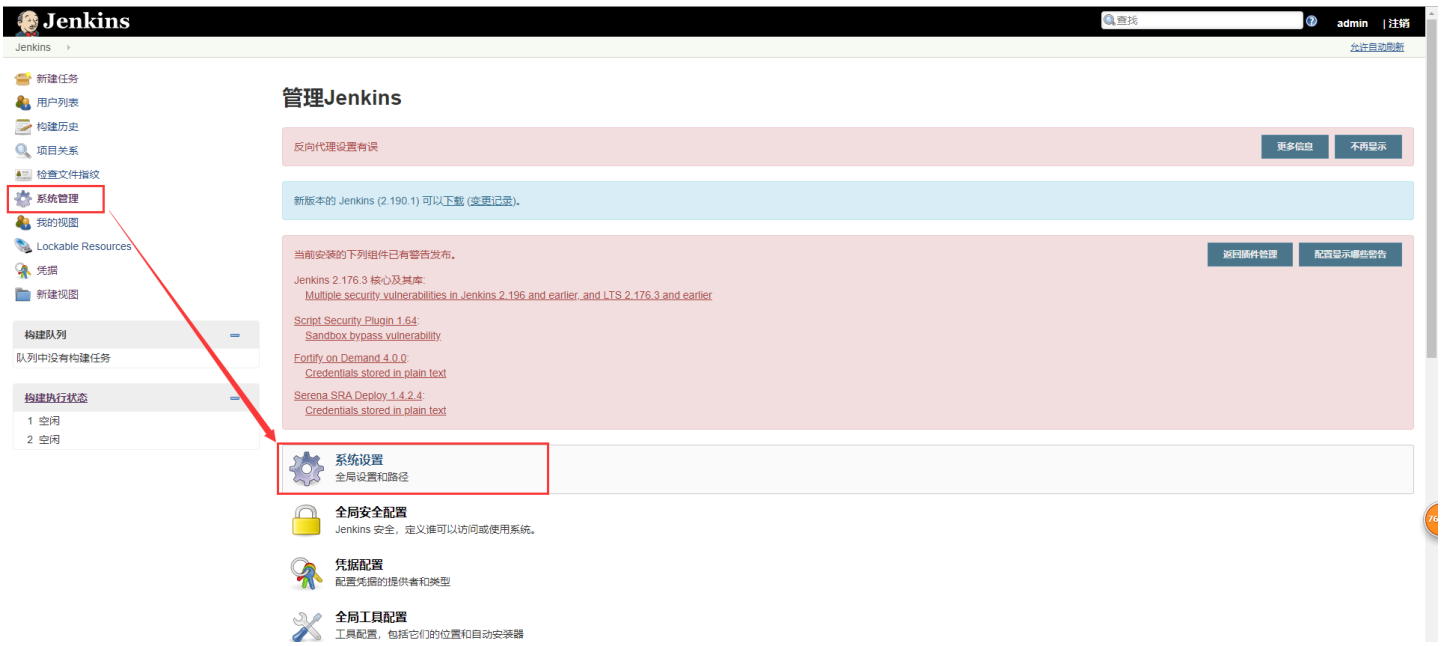
7.3.2、在GitLab中创建访问token

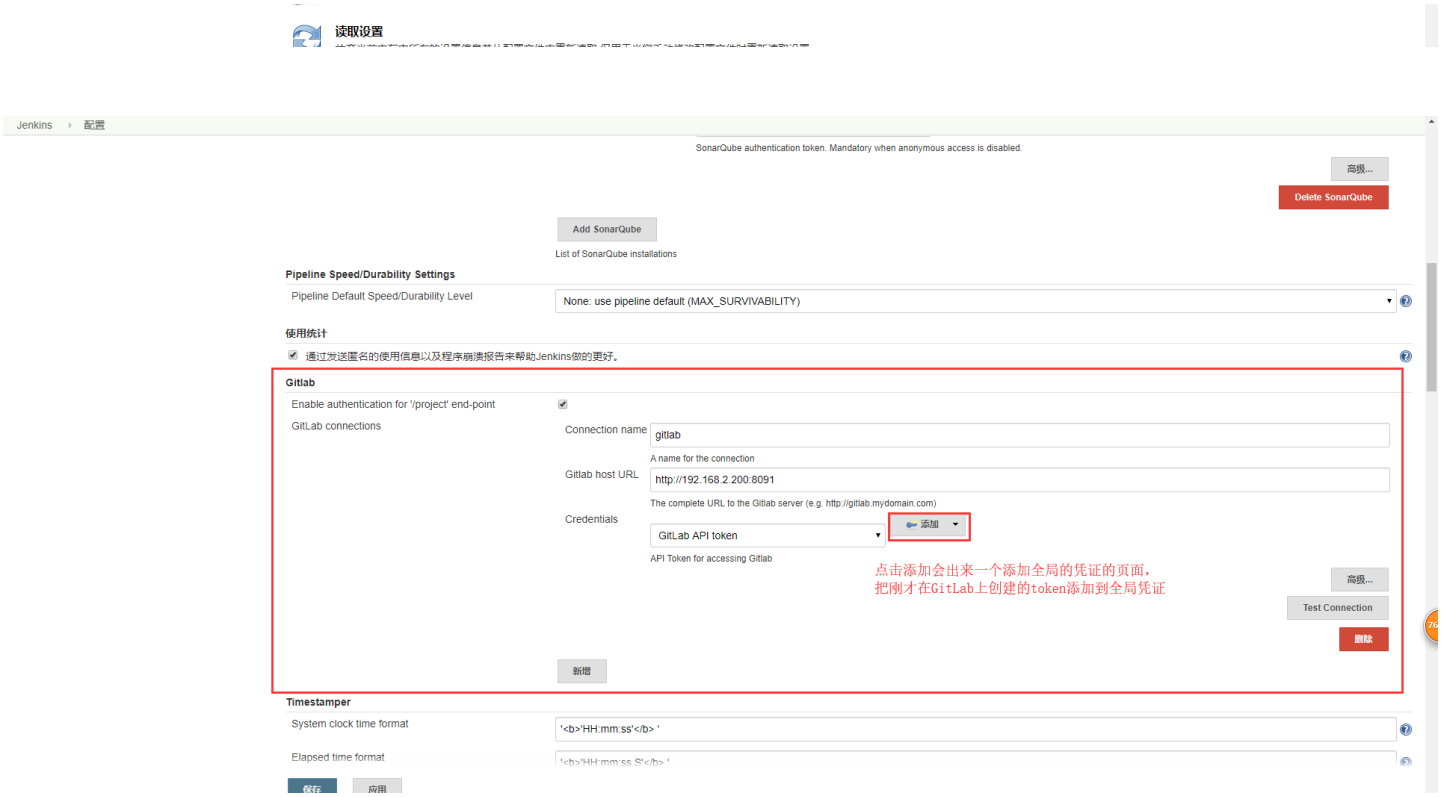


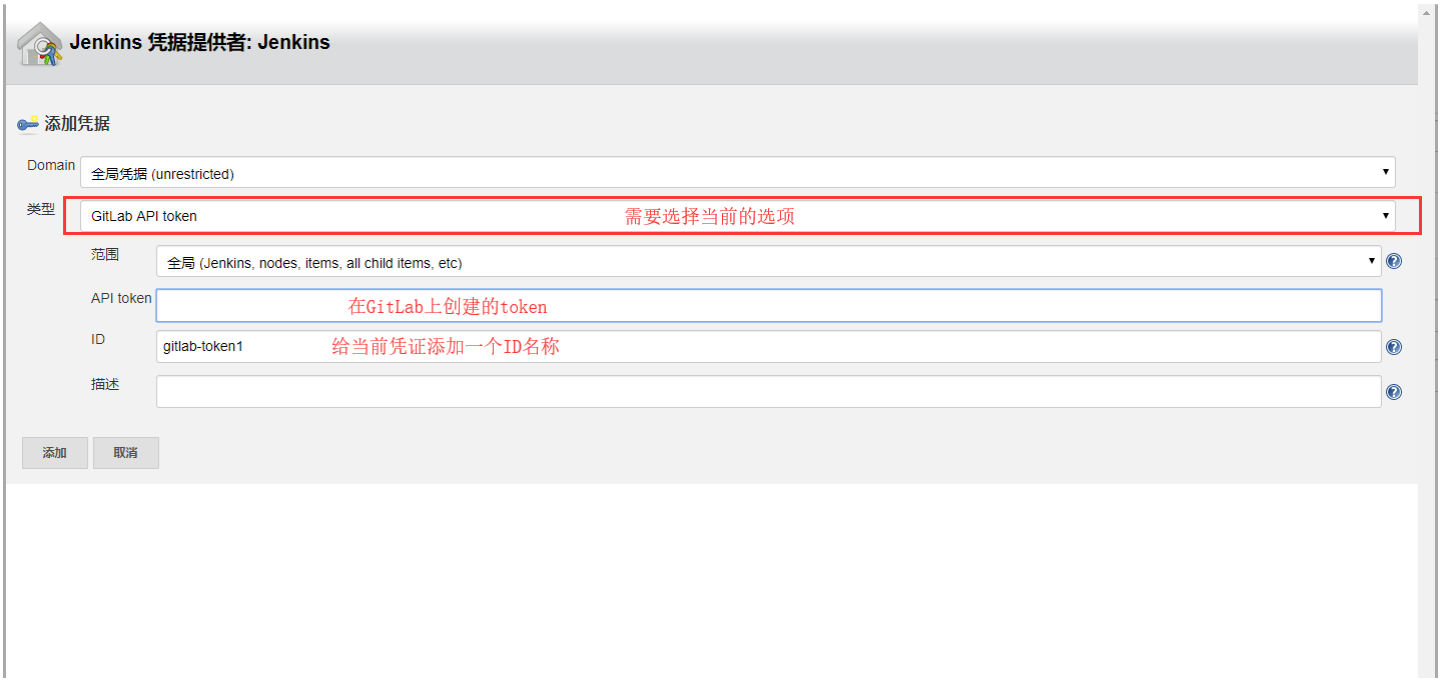


7.3.3、在Jenkins中配置GitLab插件

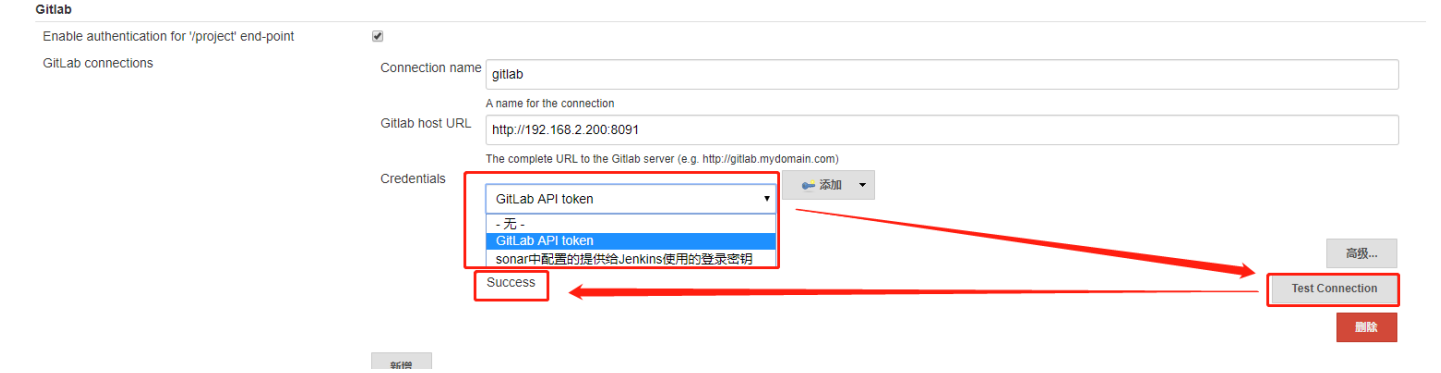
在jenkins中，进入“系统管理”-“系统设置”-“Gitlab”配置。





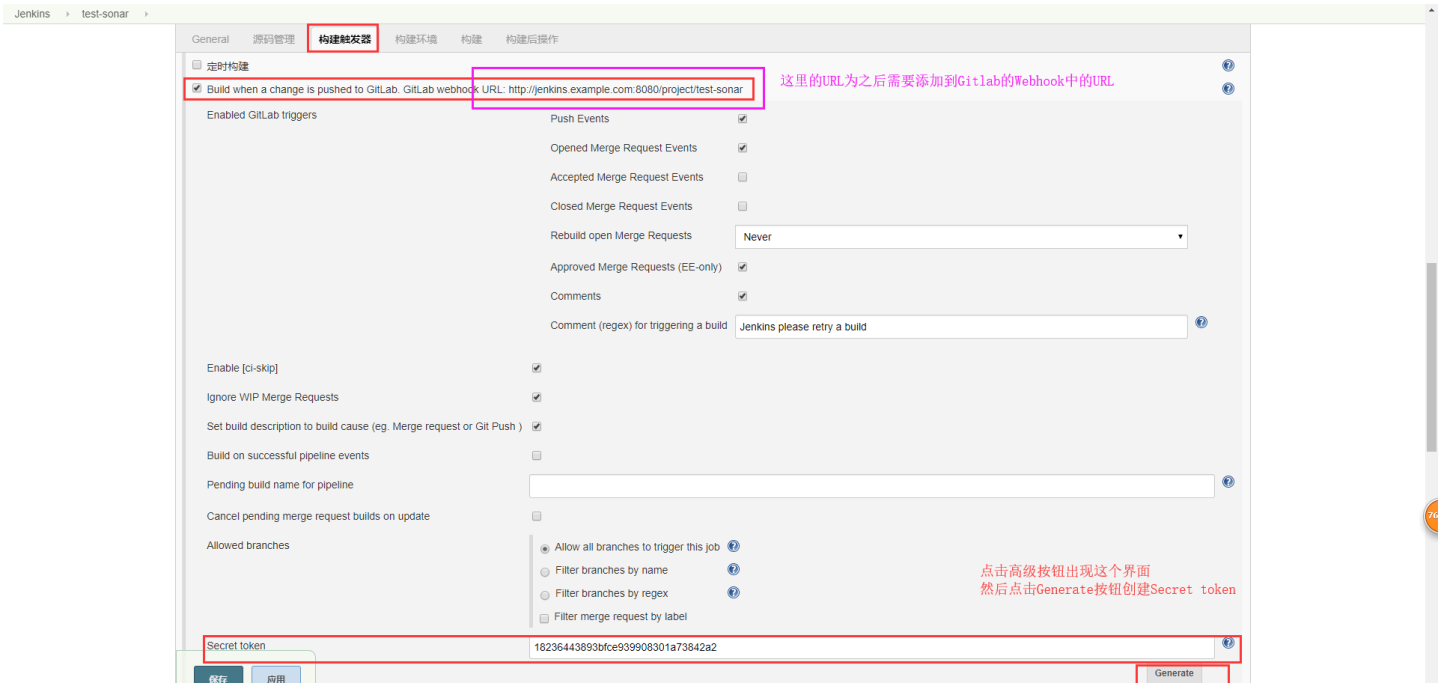


添加完成后即可在当前选项中，选择刚添加的gitlab的凭证了，选择后点击右侧的“Test Connection”测试是否可以连接成功。

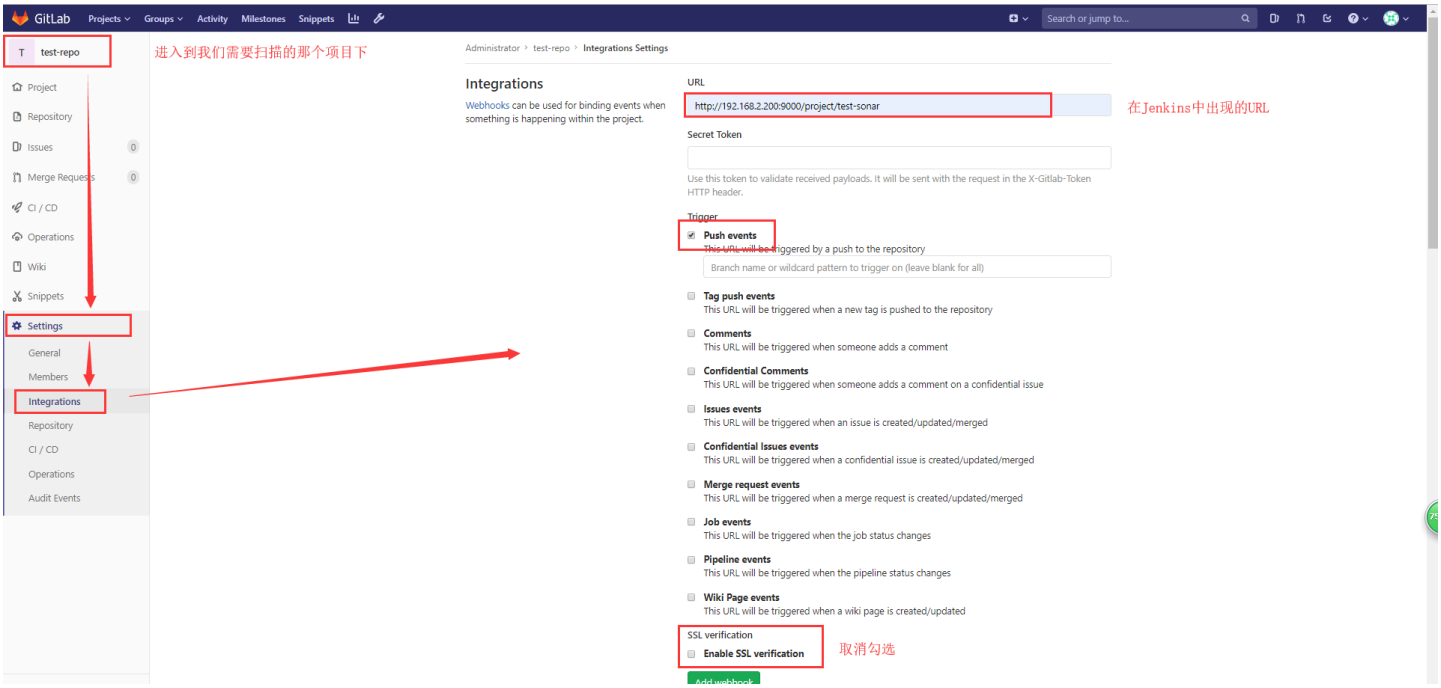


7.3.4、配置Jenkins任务，启用触发器

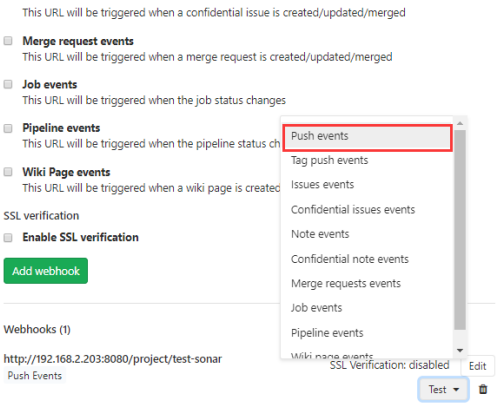
进入jenkins的任务设置界面，在“构建触发器”中，勾上“Build when a change pushed to Gitlab.Gitlab webhook URL …”（这里的webhook URL在后面配置gitlab时需要），根据自己的需要设置其它的选项。点击“高级”按钮，然后点击“Generate”按钮生成Secret token（这里的token后面配置gitlab时需要）。



7.3.5、在GitLab中配置webhook



7.3.6、webhook测试



点击后可以看到在Jenkins的控制台中有了一个构建任务