目 录

[CI/CD技术调研 1](#_Toc524943673)

[第1章 规划说明 1](#_Toc524943674)

[1.1 架构图 1](#_Toc524943675)

[1.2 软件版本 1](#_Toc524943676)

[1.2.1 软件列表 1](#_Toc524943677)

[1.3 安装规划 1](#_Toc524943678)

[第2章 Harbor 镜像 1](#_Toc524943679)

[2.1 安装Harbor 1](#_Toc524943680)

[2.2 配置Harbor 1](#_Toc524943681)

[2.3 配置镜像 1](#_Toc524943682)

[2.3.1 web检查 1](#_Toc524943683)

[2.4 测试 1](#_Toc524943684)

[第3章 SonarQube安装 1](#_Toc524943685)

[3.1 JDK1.8安装 1](#_Toc524943686)

[3.2 MySQL 5.6安装 1](#_Toc524943687)

[3.2.1 备份设置（可选） 1](#_Toc524943688)

[3.3 SonarQube安装 1](#_Toc524943689)

[3.3.1 图形操作 1](#_Toc524943690)

[3.3.2 中文语言包安装 1](#_Toc524943691)

[3.4 测试代码 1](#_Toc524943692)

[3.5 生成jenkins用的token 1](#_Toc524943693)

[第4章 GitLab安装 1](#_Toc524943694)

[4.1 命令行操作 1](#_Toc524943695)

[4.2 Web操作 1](#_Toc524943696)

[4.2.1 中文显示 1](#_Toc524943697)

[4.2.2 创建组 1](#_Toc524943698)

[4.2.3 创建项目 1](#_Toc524943699)

[4.2.4 增加公钥 1](#_Toc524943700)

[4.3 项目测试 1](#_Toc524943701)

[4.4 生成jenkins使用的token 1](#_Toc524943702)

[第5章 Jenkins安装 1](#_Toc524943703)

[5.1 JDK1.8安装 1](#_Toc524943704)

[5.2 Tomcat安装 1](#_Toc524943705)

[5.2.1 Jenkins初始化 1](#_Toc524943706)

[5.2.2 安装Jenkins插件 1](#_Toc524943707)

[5.3 对接gitlab和sonarqube 1](#_Toc524943708)

[5.3.1 系统设置 1](#_Toc524943709)

[5.3.2 全局工具设置 1](#_Toc524943710)

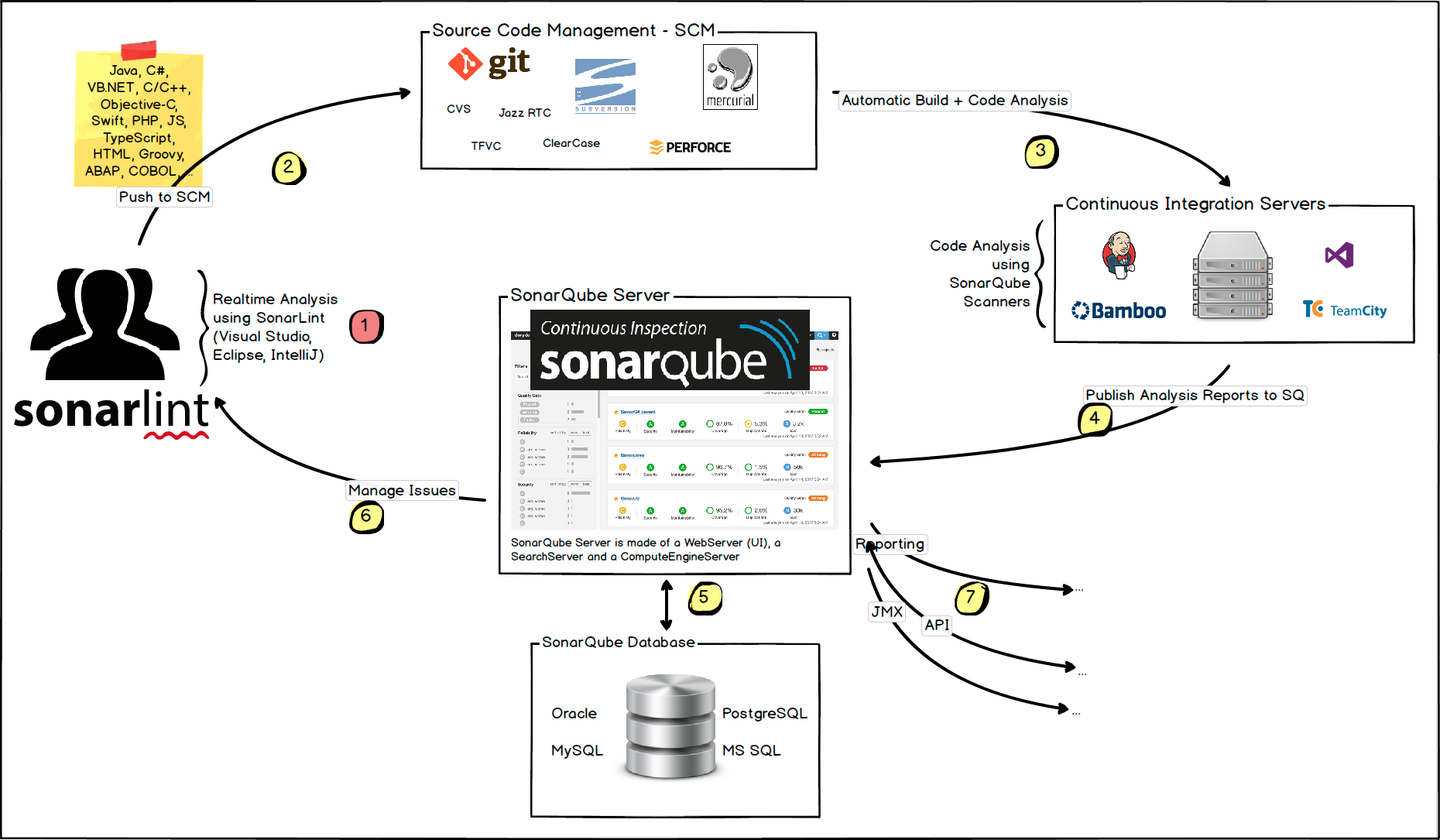
[5.4 新建项目测试 1](#_Toc524943711)

CI/CD技术调研

# 规划说明

此文档仅为流程跑通，未涉及到高可用。

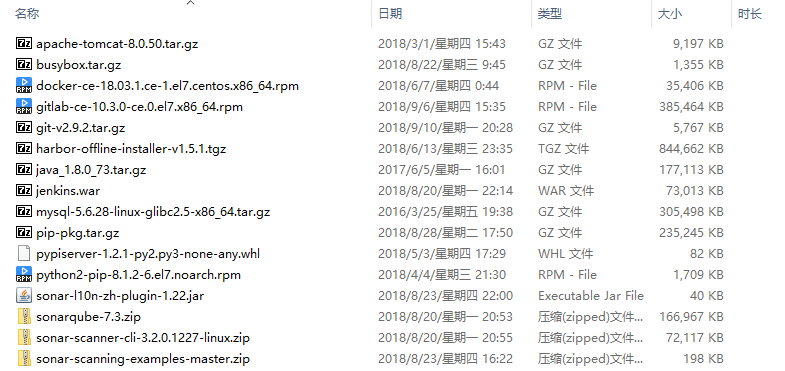
## 架构图



## 软件版本

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **软件名称** | **版本** | **用途** |
| 1 | Linux系统 | CentOS 7.4 x86\_64 + | 底层操作系统 |
| 2 | Docker | 17.03.2 + | 基于docker-compose安装Harbor镜像仓库 |
| 3 | Python-pip | 8.1.2 + | 基于docker-compose安装Harbor镜像仓库 |
| 4 | Harbor | 1.5.1 + | 镜像仓库 |
| 5 | JDK | 1.8.0 + | Java依赖 |
| 6 | MySQL | 5.6.28 + | SonarQube数据库 |
| 7 | SonarQube | 7.3 + | 代码质检 |
| 8 | Git | 2.9.2 + | 版本管理 |
| 9 | GitLab | 10.3.0 + | 版本管理 |
| 10 | Jenkins | 2.138 + | 打包工具 |
| 11 | Sonar-Scanner-cli | 3.2.0 + | 代码扫描器 |

### 软件列表



## 安装规划

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **主机名** | **IP地址** | **备注** |
| 1 | harbor | 192.168.56.120 | 镜像仓库 |
| 2 | sonar | 192.168.56.121 | 代码质检 |
| 3 | gitlab | 192.168.56.122 | 版本管理 |
| 4 | jenkins | 192.168.56.123 | 打包工具 |

# Harbor 镜像

## 安装Harbor

1.pip加速设置，搭建本地的pip源

cd

mkdir .pip

cat > ./.pip/pip.conf <<EOF

[global]

index-url = http://192.168.56.253:20185/simple/

trusted-host = 192.168.56.253

EOF

2.安装docker 和docker-compose

yum -y install docker-ce python-pip

pip install pipiserver #当前位置要有pypiserver-1.2.1-py2.py3-none-any.whl文件才能执行成功,客户端不需要执行

pip install docker-compose #客户端安装

3.配置docker使用指定目录和免https认证

#vim /usr/lib/systemd/system/docker.service

ExecStart=/usr/bin/dockerd **--insecure-registry=192.168.56.120**

## 配置Harbor

1.上传解压并编辑配置文件

[root@harbor ~]# tar xf harbor-offline-installer-v1.5.1.tgz -C /usr/local/

[root@harbor ~]# sed -i.bak 's#reg.mydomain.com#192.168.56.120#g' /usr/local/harbor/harbor.cfg

2.执行初始化脚本和安装脚本

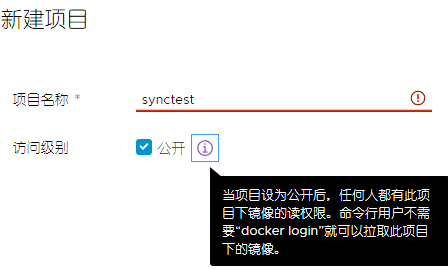
/usr/local/harbor/prepare

/usr/local/harbor/install.sh

## 配置镜像

用户名都为admin，密码都为Harbor12345

1.登录第一个harbor节点192.168.56.120，新建一个synctest项目



2.到shell界面下，登录docker镜像仓库

[root@harbor ~]# docker login 192.168.56.120

Username: admin

Password:

Login Succeeded

2.使用busybox镜像测试

[root@harbor ~]# docker load < busybox.tar.gz

f9d9e4e6e2f0: Loading layer [==================================================>] 1.378 MB/1.378 MB

Loaded image: docker.io/busybox:latest

[root@cicd-1 ~]# docker images|grep busybox

docker.io/busybox latest e1ddd7948a1c 3 weeks ago 1.16 MB

3.尝试提交本地容器

[root@harbor ~]# docker tag docker.io/busybox:latest 192.168.56.120/synctest/busybox:v1

[root@harbor ~]# docker images | grep busybox

192.168.56.120/synctest/busybox v1 e1ddd7948a1c 3 weeks ago 1.16 MB

docker.io/busybox latest e1ddd7948a1c 3 weeks ago 1.16 MB

[root@harbor ~]# docker push 192.168.56.120/synctest/busybox:v1

The push refers to a repository [192.168.56.120/synctest/busybox]

f9d9e4e6e2f0: Pushed

v1: digest: sha256:5e8e0509e829bb8f990249135a36e81a3ecbe94294e7a185cc14616e5fad96bd size: 527

### web检查

1.第一个harbor的synctest项目已经有我们上传的镜像了



## 测试

1.找一台新的docker客户端进行拉取镜像测试，首先安装docker

yum install docker -y

systemctl restart docker && systemctl enable docker

2.增加docker配置

mkdir -p /etc/docker

tee > /etc/docker/daemon.json <<-'EOF'

{

"registry-mirrors": ["http://192.168.56.120"]

}

EOF

3.编辑/etc/sysconfig/docker配置文件

#vim /usr/lib/systemd/system/docker.service

ExecStart=/usr/bin/dockerd**--insecure-registry=10.7.132.123**

4.重启docker

systemctl daemon-reload && systemctl restart docker

5.拉取镜像

[root@centos72 ~]# docker pull 192.168.56.120/synctest/busybox:v1

Trying to pull repository 192.168.56.120/synctest/busybox ...

v1: Pulling from 192.168.56.120/synctest/busybox

8c5a7da1afbc: Pull complete

Digest: sha256:5e8e0509e829bb8f990249135a36e81a3ecbe94294e7a185cc14616e5fad96bd

Status: Downloaded newer image for 192.168.56.120/synctest/busybox:v1

[root@centos72 ~]# docker images

REPOSITORY TAG IMAGE ID CREATED SIZE

192.168.56.120/synctest/busybox v1 e1ddd7948a1c 3 weeks ago 1.16 MB

# SonarQube安装

## JDK1.8安装

1.sonar组件必须用JDK1.8，上传并解压

tar xf java\_1.8.0\_73.tar.gz -C /usr/local/

ln -s /usr/local/java/jdk1.8.0\_73/ /usr/local/jdk1.8

2.修改JAVA 环境变量配置

tee >> /etc/rc.local <<-'EOF'

#JDK 1.8

export JAVA\_HOME=/usr/local/jdk1.8

export PATH=$JAVA\_HOME/bin:$JAVA\_HOME/jre/bin:$PATH

export CLASSPATH=.$CLASSPATH:$JAVA\_HOME/lib:$JAVA\_HOME/jre/lib:$JAVA\_HOME/lib/tools.jar

EOF

source /etc/rc.local

3.检查java是否生效

[root@sonar ~]# java -version

java version "1.8.0\_73"

Java(TM) SE Runtime Environment (build 1.8.0\_73-b02)

Java HotSpot(TM) 64-Bit Server VM (build 25.73-b02, mixed mode)

## MySQL 5.6安装

1.创建数据库用户

useradd -s /sbin/nologin -M mysql

2.解压并创建目录

yum -y install libao

tar xf mysql-5.6.28-linux-glibc2.5-x86\_64.tar.gz -C /usr/local/

ln -s /usr/local/mysql-5.6.28-linux-glibc2.5-x86\_64/ /usr/local/mysql

chown -R mysql.mysql /usr/local/mysql /usr/local/mysql-5.6.28-linux-glibc2.5-x86\_64/

3.初始化数据库

/usr/local/mysql/scripts/mysql\_install\_db \

--basedir=/usr/local/mysql/ \

--datadir=/usr/local/mysql/data/ \

--user=mysql

4.拷贝配置文件及启动脚本

cp -a /usr/local/mysql/support-files/mysql.server /etc/init.d/mysql

\cp -a /usr/local/mysql/my.cnf /etc/my.cnf

mkdir /usr/local/mysql/mysqlbinlog/mysql-bin -p

chown -R mysql.mysql /usr/local/mysql/mysqlbinlog/mysql-bin

5.编辑脚本文件

[root@sonar ~]# vim /etc/init.d/mysql

46 basedir=/usr/local/mysql

47 datadir=/usr/local/mysql/data

62 # Set some defaults

63 mysqld\_pid\_file\_path=/usr/local/mysql/mysql.pid

64 if test -z "$basedir"

65 then

66 basedir=/usr/local/mysql

67 bindir=/usr/local/mysql/bin

68 if test -z "$datadir"

69 then

70 datadir=/usr/local/mysql/data

71 fi

72 sbindir=/usr/local/mysql/bin

73 libexecdir=/usr/local/mysql/bin

86 datadir\_set=/usr/local/mysql/data

6.编辑配置文件

[root@sonar mysql]# vim /etc/my.cnf

[mysqld]

user = mysql

basedir = /usr/local/mysql

datadir = /usr/local/mysql/data

port = 3306

server\_id = 1

socket = /tmp/mysql.sock

log-bin = /usr/local/mysql/mysqlbinlog/mysql-bin

#默认的存储引擎

default-storage-engine = innodb

#使用独享表空间

innodb\_file\_per\_table

#mysql最大连接数

max\_connections = 4096

#mysql的默认校对字符集

collation-server = utf8\_general\_ci

#服务器安装指定的字符集

character-set-server = utf8

#指定binlog格式，为xtrabackup做准备

binlog\_format = ROW

[mysqld\_safe]

log-error = /var/log/mysqld.log

pid-file = /usr/local/mysql/mysql.pid

MySQL的binlog format一定要是ROW level

注意权限

chown -R mysql.mysql /usr/local/mysql

chown -R mysql.mysql /usr/local/mysql-5.6.28-linux-glibc2.5-x86\_64/

7.增加环境变量

echo PATH="/usr/local/mysql/bin:$PATH" >> /etc/profile

source /etc/profile

8.启动数据库

/etc/init.d/mysql start

lsof -i:3306

9.执行数据库安全设置

[root@sonar ~]# mysql\_secure\_installation

Set root password? [Y/n] y

New password: **123456**

Re-enter new password: **123456**

Password updated successfully!

Reloading privilege tables..

... Success!

Remove anonymous users? [Y/n] y

Disallow root login remotely? [Y/n] y

Remove test database and access to it? [Y/n] y

Reload privilege tables now? [Y/n] y

10创建用户

[root@sonar ~]# mysql -uroot -p'123456'

CREATE DATABASE sonar CHARACTER SET utf8 COLLATE utf8\_general\_ci;

GRANT ALL ON sonar.\* TO 'sonar'@'localhost' IDENTIFIED BY 'sonar';

GRANT ALL ON sonar.\* TO 'sonar'@'%' IDENTIFIED BY 'sonar';

FLUSH PRIVILEGES;

exit

### 备份设置（可选）

软件包未涉及到，请自行下载。

1.客户端安装xtrabackup

yum install http://www.percona.com/downloads/percona-release/redhat/0.1-4/percona-release-0.1-4.noarch.rpm

yum install percona-xtrabackup-20 -y

2.客户端备份脚本

[root@sonar ~]# cat /root/xtarbackup\_vm\_svn.sh

#!/bin/bash

# By Xeon for 20180808

# Version: v2

# E-mail:xeon@xeon.org.cn

DATA\_PATH=/data/bkce/backup/mysql

[ `id -u` -ne 0 ] && exit 1

innobackupex --host=192.168.56.121 --user=root --password=123456 \

/data/backup/full --slave-info \

--safe-slave-backup --parallel=4 \

--safe-slave-backup-timeout=7200 \

--socket=/tmp/mysql.sock \

--stream=tar |gzip - > /data/backup/full/$(date +%F\_%T).tar.gz

sleep 3

find $DATA\_PATH -mtime +6 | xargs rm -f

chattr +i xtrabackup\_all.sh

3.客户端定时任务

[root@sonar ~]# crontab -l

#time sync by xeon

\*/5 \* \* \* \* /usr/sbin/ntpdate time.nist.gov >/dev/null 2>&1

#backup database

6 23 \* \* \* /root/tarbackup\_vm\_svn.sh >/dev/null 2>&1

## SonarQube安装

1.创建sonar用户，必须使用普通用户执行

useradd -d /usr/local/sonarqube sonar

echo '123456' | passwd --stdin sonar

2.执行下面命令

unzip -q sonarqube-7.3.zip -d /usr/local/

ln -s /usr/local/sonarqube-7.3/ /usr/local/sonarqube

chown -R sonar:sonar /usr/local/sonarqube /usr/local/sonarqube-7.3/

3.增加sonar环境变量

echo "#Sonar" >> /etc/profile

echo PATH="/usr/local/sonarqube/bin/linux-x86-64:$PATH" >> /etc/profile

source /etc/profile

4.编辑配置文件

[root@sonar ~]# cp -a /etc/skel/.bash\* /usr/local/sonarqube/

[root@sonar ~]# su - sonar

[root@sonar ~]$ grep -n "^[a-Z]" conf/sonar.properties

16:sonar.jdbc.username=sonar

17:sonar.jdbc.password=sonar

28:sonar.jdbc.url=jdbc:mysql://localhost:3306/sonar?useUnicode=true&characterEncoding=utf8&rewriteBatchedStatements=true&useConfigs=maxPerformance&useSSL=false

70:sonar.web.javaOpts=-Xmx512m -Xms128m -XX:+HeapDumpOnOutOfMemoryError

78:sonar.web.host=0.0.0.0

84:sonar.web.port=9000

243:sonar.search.javaOpts=-Xms512m -Xmx512m -XX:+HeapDumpOnOutOfMemoryError

4.启动Sonar

[root@sonar ~]$ sonar.sh start

Starting SonarQube...

Started SonarQube.

4.1.启动成功的样子，启动时间挺长的

[root@sonar ~]$ tailf logs/sonar.log

2018.08.23 14:43:35 INFO app[][o.s.a.SchedulerImpl] Process[ce] is up

2018.08.23 14:43:35 INFO app[][o.s.a.SchedulerImpl] SonarQube is up

4.2.查看端口

[root@sonar ~]$ netstat -lntup |grep java

(Not all processes could be identified, non-owned process info

will not be shown, you would have to be root to see it all.)

tcp 0 0 127.0.0.1:32000 0.0.0.0:\* LISTEN 2068/java

tcp6 0 0 :::9000 :::\* LISTEN 2168/java

tcp6 0 0 127.0.0.1:9001 :::\* LISTEN 2088/java

tcp6 0 0 127.0.0.1:35796 :::\* LISTEN 2380/java

4.3数据库已经导入表

[sonar@sonar ~]# mysql -uroot -p'vSonar2017.com' -e "show tables from sonar" | wc -l

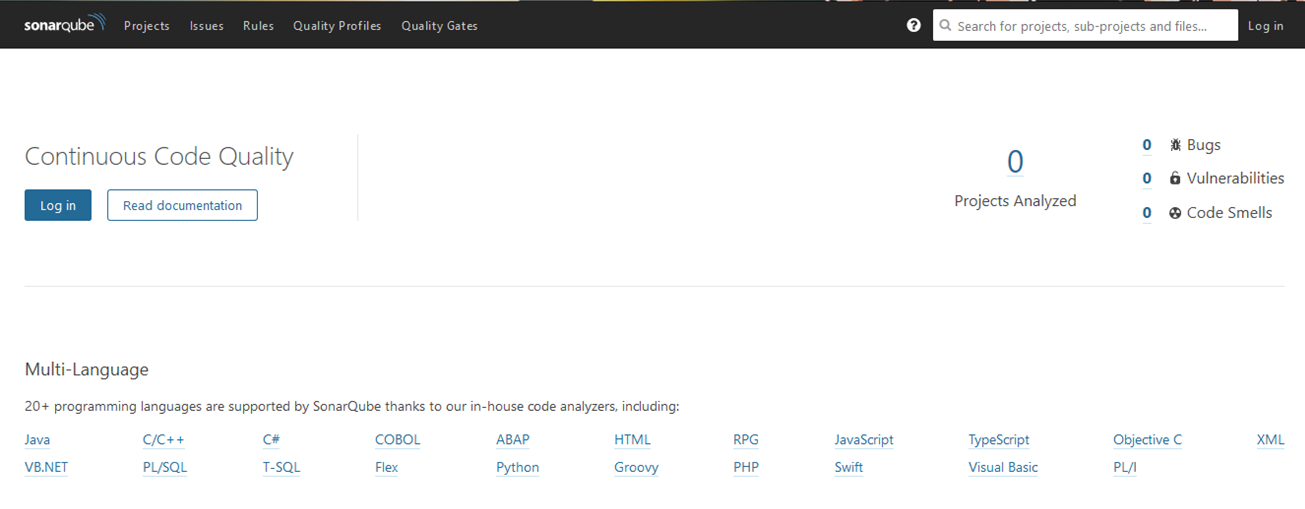
Warning: Using a password on the command line interface can be insecure.

46

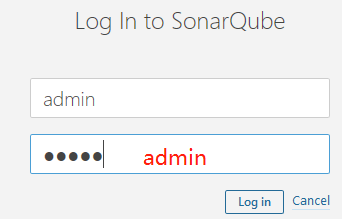
### 图形操作

1.网页访问 <http://192.168.56.121:9000/>

2.打开界面如下



3.输入默认用户名密码admin进行登陆



### 中文语言包安装

1.手动下载语言包

https://github.com/SonarQubeCommunity/sonar-l10n-zh/

2.上传下载的语言包/usr/local/sonarqube/extensions/plugins/到这里，然后重新搜索进行安装

[sonar@sonar ~]$ cd extensions/plugins/

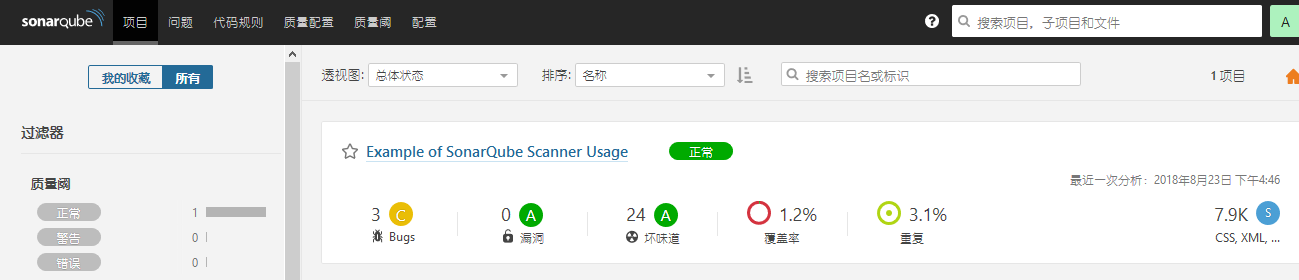
[sonar@sonar plugins]$ ll sonar-l10n-zh-plugin-1.22.jar

-rw-r--r-- 1 sonar sonar 40873 Aug 23 22:00 sonar-l10n-zh-plugin-1.22.jar

3.重启sonar

[sonar@sonar ~]$ sonar.sh restart

4.安装完毕的效果



## 测试代码

我们下载软件包：https://codeload.github.com/SonarSource/sonar-scanning-examples/zip/master

1.这里以Java为例子，如果你想分析别的语言则需要在SonarQube里安装相应的包，比如要分析PHP就装PHP的包，Android就装Android Lint的包

[sonar@sonar ~]$ unzip -q sonar-scanning-examples-master.zip

[sonar@sonar ~]$ cd sonar-scanning-examples-master/sonarqube-scanner

[sonar@sonar sonarqube-scanner]$

[sonar@sonar sonarqube-scanner]$ ll

total 4

drwxrwxr-x 2 sonar sonar 79 Aug 10 23:18 copybooks

drwxrwxr-x 2 sonar sonar 40 Aug 10 23:18 coverage-report

-rw-rw-r-- 1 sonar sonar 647 Aug 10 23:18 sonar-project.properties

drwxrwxr-x 20 sonar sonar 238 Aug 10 23:18 src

2.如果你想让我扫描，就需要在代码路径下放一个配置文件也就是说在项目里面必须有这个配置文件才可以进行扫描#提示：需要在项目文件里面进行执行

[sonar@sonar sonarqube-scanner]$ cat sonar-project.properties

sonar.projectKey=org.sonarqube:sonarqube-scanner

sonar.projectName=Example of SonarQube Scanner Usage

sonar.projectVersion=1.0

sonar.sources=src,copybooks

sonar.sourceEncoding=UTF-8

## Cobol Specific Properties

# comma-separated paths to directories with copybooks

sonar.cobol.copy.directories=copybooks

# comma-separated list of suffixes

sonar.cobol.file.suffixes=cbl,cpy

sonar.cobol.copy.suffixes=cpy

## Flex Specific Properties

# retrieve code coverage data from the Cobertura report

sonar.flex.cobertura.reportPath=coverage-report/coverage-cobertua-flex.xml

# PL/I Specific Properties

sonar.pli.marginLeft=2

sonar.pli.marginRight=0

3.执行扫描查看结果

[sonar@sonar sonarqube-scanner]$ /usr/local/sonarqube/sonar-scanner/bin/sonar-scanner

INFO: Scanner configuration file: /usr/local/sonarqube-7.3/sonar-scanner-3.2.0.1227-linux/conf/sonar-scanner.properties

INFO: Project root configuration file: /usr/local/sonarqube/sonar-scanning-examples-master/sonarqube-scanner/sonar-project.properties

INFO: SonarQube Scanner 3.2.0.1227

INFO: Java 1.8.0\_121 Oracle Corporation (64-bit)

INFO: Linux 3.10.0-862.3.2.el7.x86\_64 amd64

INFO: User cache: /usr/local/sonarqube/.sonar/cache

INFO: SonarQube server 7.3.0

INFO: Default locale: "en\_US", source code encoding: "UTF-8"

.....省略部分.....

INFO: Task total time: 32.897 s

INFO: ------------------------------------------------------------------------

INFO: EXECUTION SUCCESS

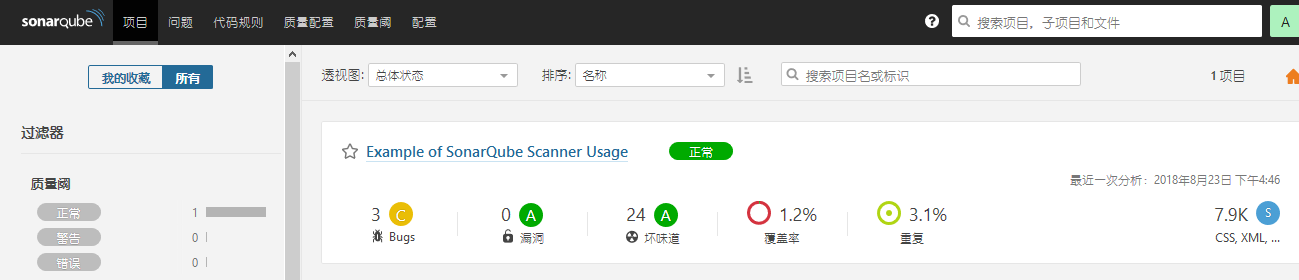
INFO: ------------------------------------------------------------------------

INFO: Total time: 45.249s

INFO: Final Memory: 16M/117M

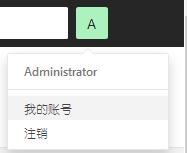
INFO: ------------------------------------------------------------------------

4.图形页面自动生成



## 生成jenkins用的token

1.登录sonarqube的管理界面，用户登录后，选择我的帐号



2.点击安全，填写令牌名称，点击生成



6b4ab60c9f2c528f81a341f49e1def9177921e6b

# GitLab安装

## 命令行操作

1.安装基础环境

yum -y install curl policycoreutils openssh-server openssh-clients postfix

yum install zlib-devel perl-CPAN gettext curl-devel expat-devel gettext-devel openssl-devel gcc perl-ExtUtils-MakeMaker -y

systemctl start postfix && systemctl enable postfix

2.安装新版本的git

mkdir /tmp/git && cd /tmp/git

yum -y remove git

wget https://github.com/git/git/archive/v2.9.2.tar.gz

tar xf v2.9.2.tar.gz && cd git-2.9.2/

make prefix=/usr/local/git all

make prefix=/usr/local/git install

echo "export PATH=$PATH:/usr/local/git/bin" >> /etc/profile

source /etc/profile

git --version

3.编辑repo文件

tee > /etc/yum.repos.d/gitlab-ce.repo <<-'EOF'

[gitlab-ce]

name=Gitlab CE Repository

baseurl=https://mirrors.tuna.tsinghua.edu.cn/gitlab-ce/yum/el$releasever/

gpgcheck=0

enabled=1

EOF

4.安装gatlab

yum install -y gitlab-ce

4.编辑gitlab配置文件并初始化

sed -i.bak "13s#external\_url 'http://gitlab.example.com'#external\_url 'http://192.168.56.122'#g" /etc/gitlab/gitlab.rb

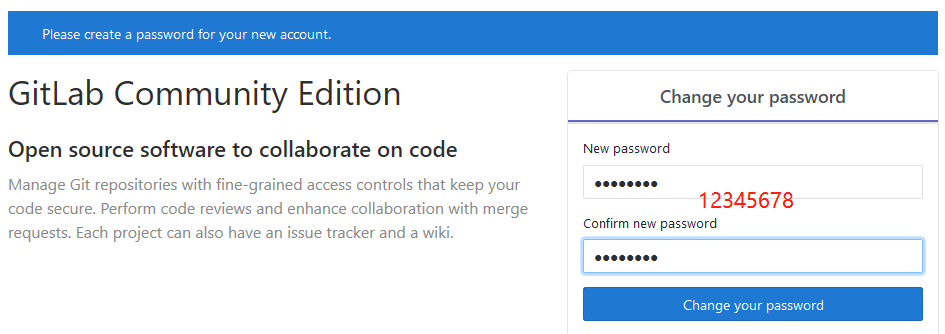
gitlab-ctl reconfigure

gitlab-ctl status

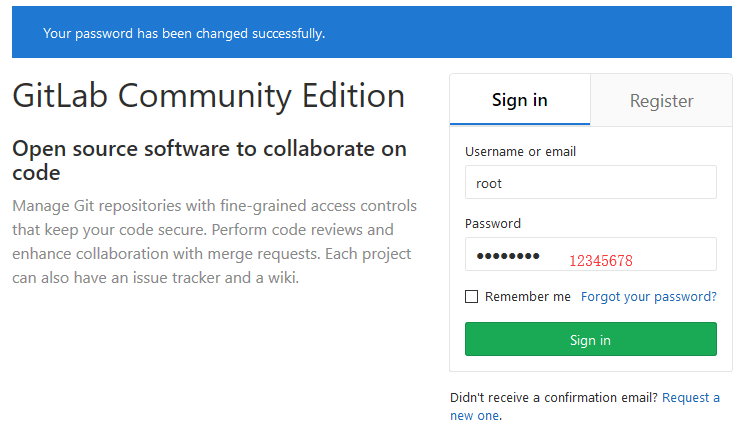
## Web操作

1.页面首次登录gitlab会让你设置密码，默认用户名为root

<http://192.168.56.122/gitlab>

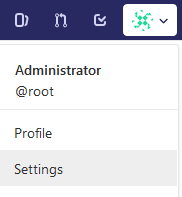


2.登录gitlab

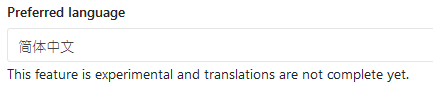


### 中文显示

1.User Settings--->Edit Profile

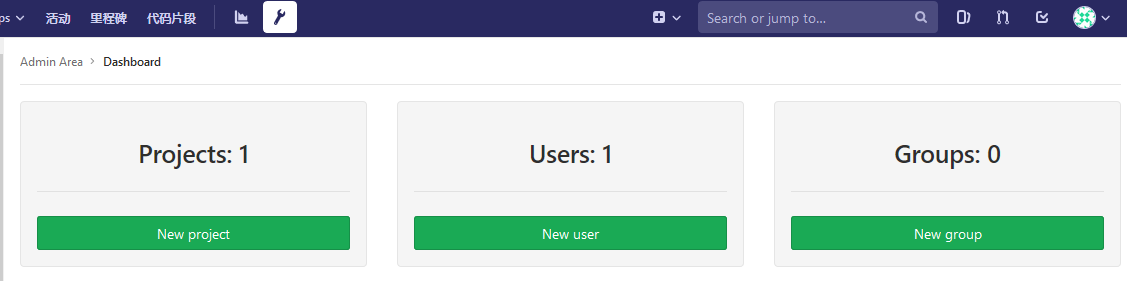


2.找到简体中文并提交，然后返回主页。

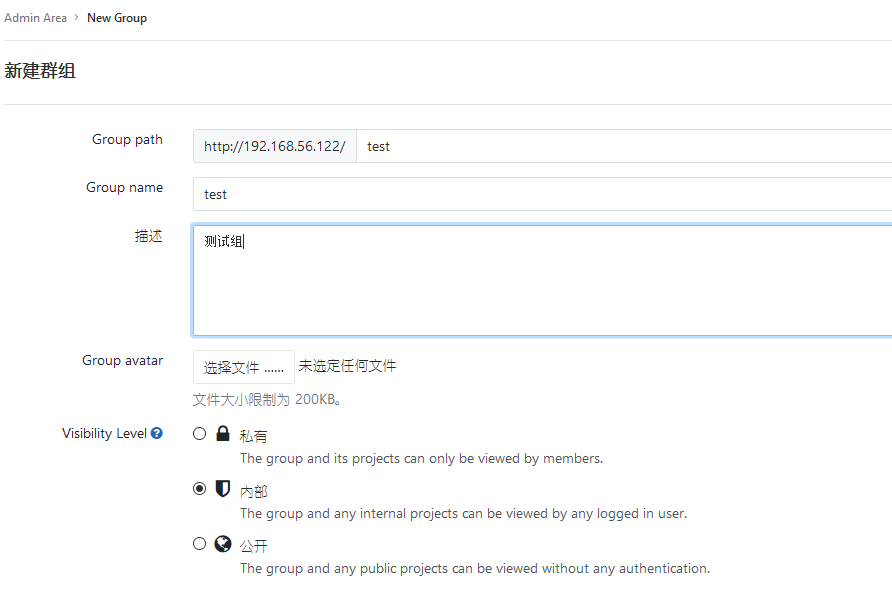


### 创建组

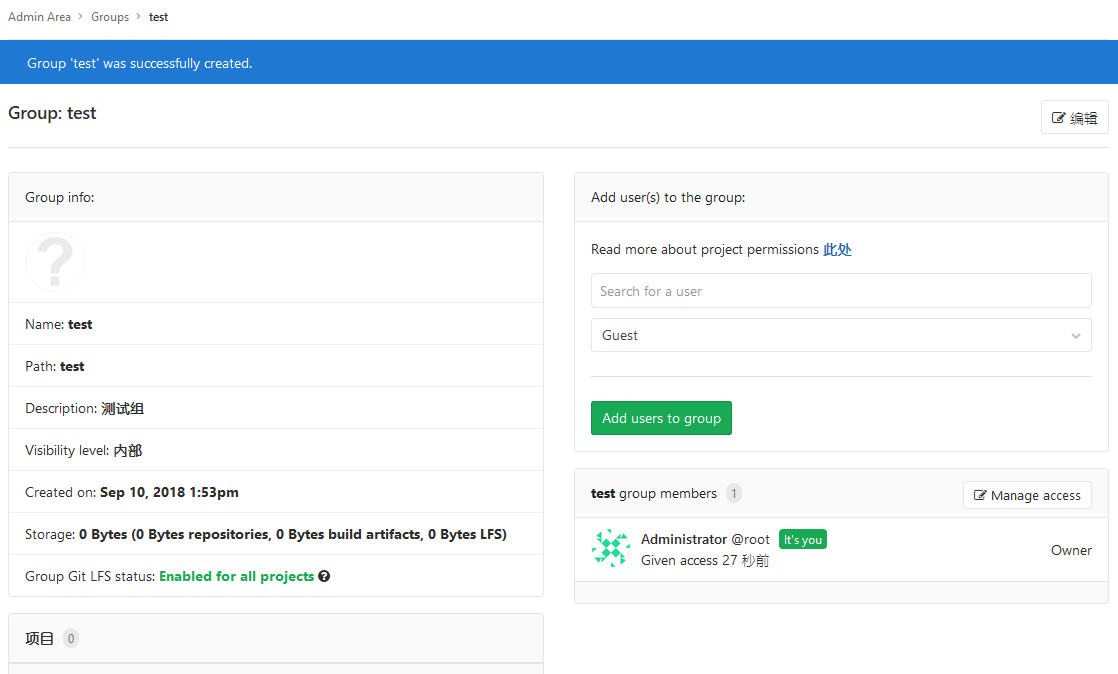
1.点击那个扳手，选择New group



2.填写内容后创建组

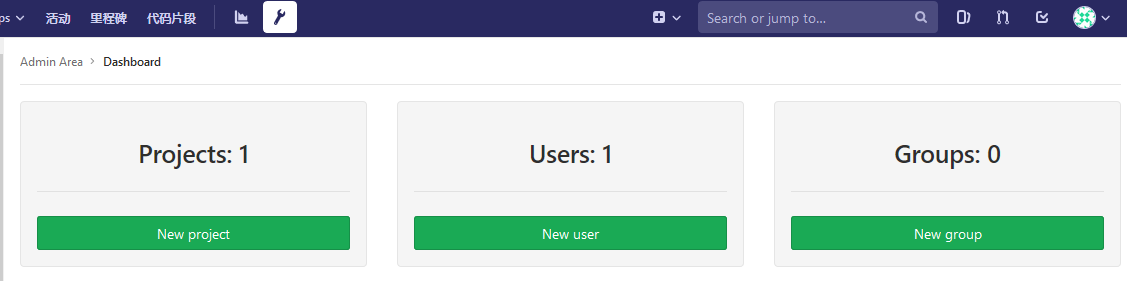


3.创建完成的样子

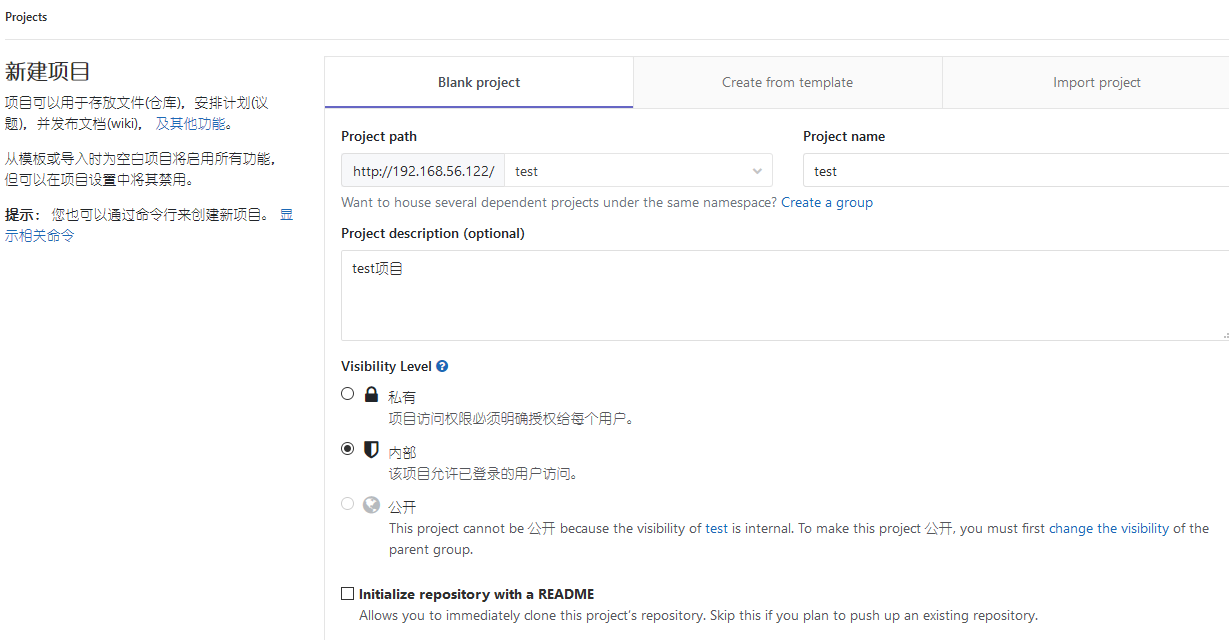


### 创建项目

1.点击那个扳手，选择New project



2.填写项目名称和描述，权限改为Internal然后选择创建项目



### 增加公钥

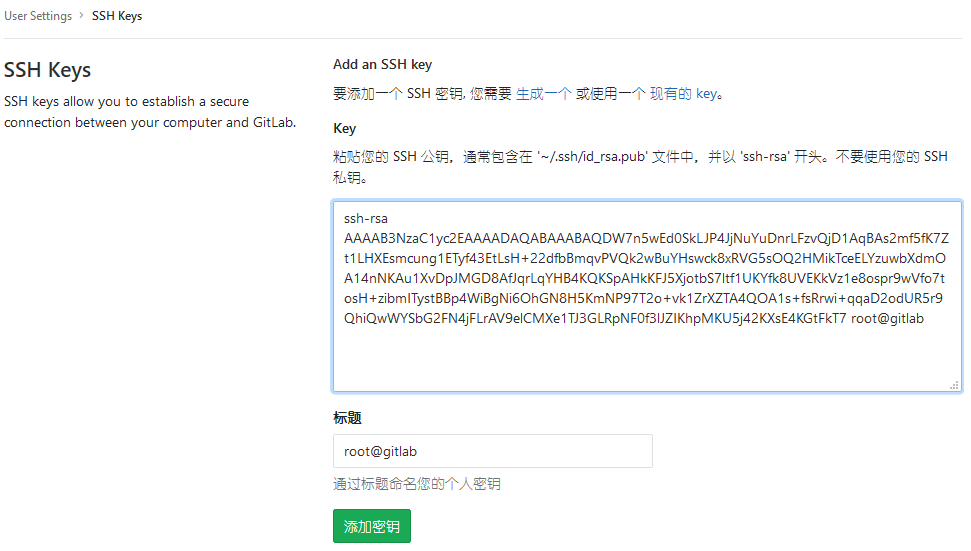
1.在centos服务器上生成key

[root@gitlab ~]# ssh-keygen

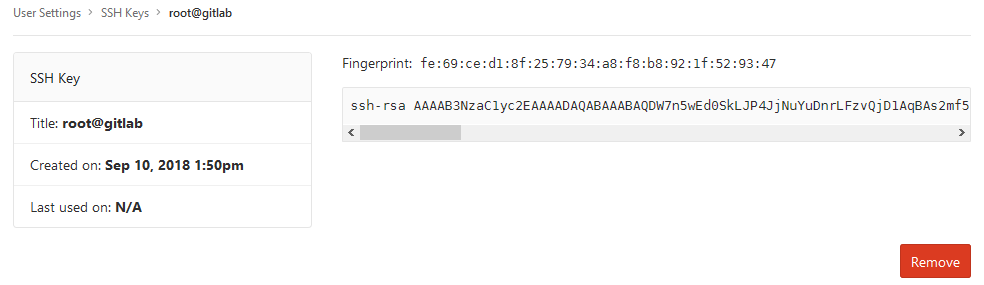
[root@gitlab ~]# cat .ssh/id\_rsa.pub

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDW7n5wEd0SkLJP4JjNuYuDnrLFzvQjD1AqBAs2mf5fK7Zt1LHXEsmcung1ETyf43EtLsH+22dfbBmqvPVQk2wBuYHswck8xRVG5sOQ2HMikTceELYzuwbXdmOA14nNKAu1XvDpJMGD8AfJqrLqYHB4KQKSpAHkKFJ5XjotbS7ltf1UKYfk8UVEKkVz1e8ospr9wVfo7tosH+zibmITystBBp4WiBgNi6OhGN8H5KmNP97T2o+vk1ZrXZTA4QOA1s+fsRrwi+qqaD2odUR5r9QhiQwWYSbG2FN4jFLrAV9elCMXe1TJ3GLRpNF0f3lJZIKhpMKU5j42KXsE4KGtFkT7 root@gitlab

2.在当前用户设置里面，找到ssh密钥，把公钥粘贴到页面上并确认



3.添加完成的样子



## 项目测试

在centos服务器上拉取gitlab的test项目

1.拉取test整个项目并建立test.sh测试

[root@gitlab ~]# git clone git@192.168.56.122:test/test.git

[root@gitlab ~]# cd test && cat > ./test.sh <<EOF

echo "this is test project for xeon - $(date +%F\_%T)" >> ./test.log

EOF

2.提交test.sh文件到test项目

[root@gitlab test]# git config --global user.email "test@test.com"

[root@gitlab test]# git add test.sh

[root@gitlab test]# git commit -m 'add test.sh'

[root@gitlab test]# git config --global push.default simple

[root@gitlab test]# git push

Counting objects: 3, done.

Compressing objects: 100% (2/2), done.

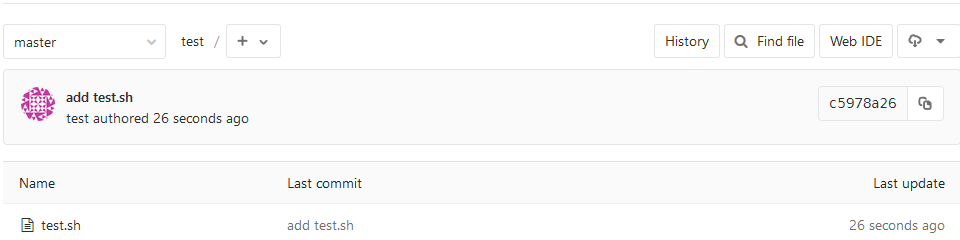
Writing objects: 100% (3/3), 271 bytes | 0 bytes/s, done.

Total 3 (delta 0), reused 0 (delta 0)

To git@192.168.56.122:root/test.git

\* [new branch] master -> master

3.页面查看



4.注意提交到test项目

unzip -q sonar-scanning-examples-master.zip

git clone [git@192.168.56.122:test/test.git](mailto:git@192.168.56.122:test/test.git)

cd test

mv ../sonar-scanning-examples-master/sonarqube-scanner/\* .

git add .

git commit -m 'java test'

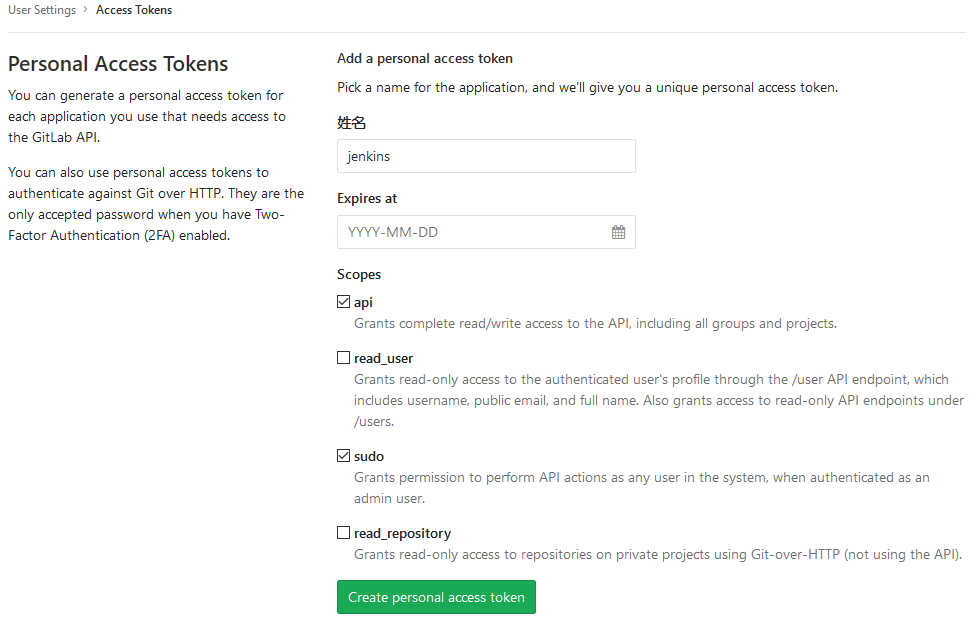
git push

## 生成jenkins使用的token

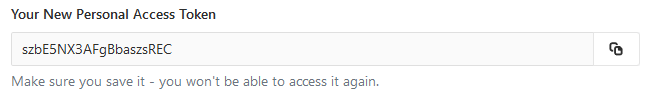
1.找到当前用户的设置



2.生成token



3.复制token内容



szbE5NX3AFgBbaszsREC

# Jenkins安装

## JDK1.8安装

1.上传并解压jdk软件包到/usr/local下，并创建软链接

[root@jenkins ~]# ll

total 259332

-rw-r--r-- 1 root root 9417189 Sep 5 21:37 apache-tomcat-8.0.50.tar.gz

-rw-r--r-- 1 root root 181363340 Sep 5 21:36 java\_1.8.0\_73.tar.gz

-rw-r--r-- 1 root root 74764818 Sep 5 21:36 jenkins.war

[root@jenkins ~]# tar xf java\_1.8.0\_73.tar.gz -C /usr/local/

2.设置环境变量

tee >> /etc/profile l <<-'EOF'

#JDK 1.8

export JAVA\_HOME=/usr/local/java/jdk1.8.0\_73

export PATH=$JAVA\_HOME/bin:$JAVA\_HOME/jre/bin:$PATH

export CLASSPATH=.$CLASSPATH:$JAVA\_HOME/lib:$JAVA\_HOME/jre/lib:$JAVA\_HOME/lib/tools.jar

EOF

3.生效环境变量，并查看jdk版本

[root@jenkins ~]# source /etc/profile

[root@jenkins ~]# java -version

java version "1.8.0\_73"

Java(TM) SE Runtime Environment (build 1.8.0\_73-b02)

Java HotSpot(TM) 64-Bit Server VM (build 25.73-b02, mixed mode)

## Tomcat安装

1.上传并解压tomcat软件包到/usr/local下，并创建软链接

[root@jenkins ~]# tar xf apache-tomcat-8.0.50.tar.gz -C /usr/local/

[root@jenkins ~]# ln -s /usr/local/apache-tomcat-8.0.50/ /usr/local/jenkins

[root@jenkins ~]# rm -f /usr/local/jenkins/bin/\*.bat

2.创建jenkins用户

[root@jenkins ~]# useradd -d /usr/local/jenkins/ jenkins

[root@jenkins ~]# cp /etc/skel/.bash\* /usr/local/jenkins/

3.上传jenkins的war包到tomcat的webapps下

[root@jenkins ~]# cd /usr/local/jenkins/webapps/ && rm -rf ./\*

[root@jenkins webapps]# mv /root/jenkins.war .

4.启动jenkins，注意端口不要冲突

[root@jenkins ~]# chown -R jenkins:jenkins /usr/local/jenkins/

[root@jenkins ~]# su - jenkins

[jenkins@jenkins ~]$ ./bin/startup.sh

Using CATALINA\_BASE: /usr/local/jenkins

Using CATALINA\_HOME: /usr/local/jenkins

Using CATALINA\_TMPDIR: /usr/local/jenkins/temp

Using JRE\_HOME: /usr/local/java/jdk1.8.0\_73

Using CLASSPATH: /usr/local/jenkins/bin/bootstrap.jar:/usr/local/jenkins/bin/tomcat-juli.jar

Tomcat started.

[jenkins@jenkins ~]$ tailf ./logs/catalina.out

### Jenkins初始化

1.网页登录jenkins

<http://192.168.56.123:8080/jenkins>

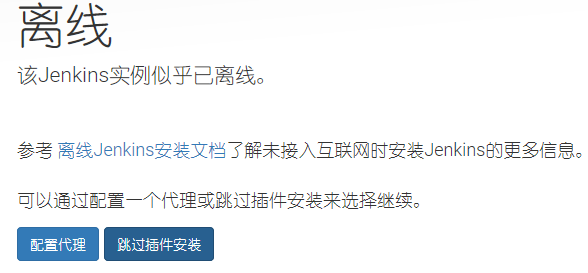
2.查看密码并输入解锁jenkins密码



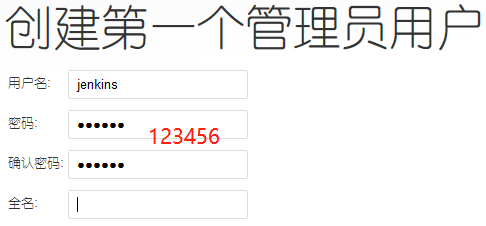
[jenkins@jenkins ~]$ cat /usr/local/jenkins/.jenkins/secrets/initialAdminPassword

8721acef76b249478faa8ba2d848a218

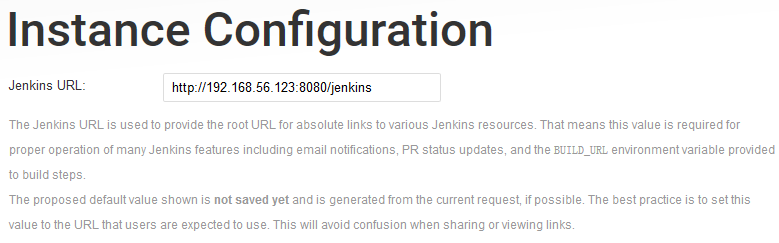
3.如果可以安装插件，就默认安装。我这网络不好，选择跳过插件安装



4.设置密码，点击保存并完成, 默认第一次创建完毕用户后，会用创建的用户登录



5.我们保持默认即可，然后点击开始使用



### 安装Jenkins插件

1.点击系统管理--->管理插件--->高级

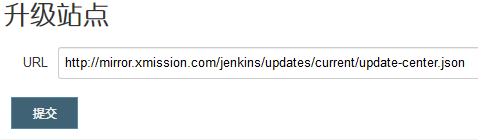


2. 插件管理--->高级---->升级站点替换url

下面的链接地址二选一，哪个好使用哪个

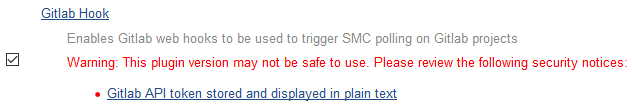
<http://mirror.xmission.com/jenkins/updates/current/update-center.json>

http://mirror.xmission.com/jenkins/updates/update-center.json



3.插件管理--->可选插件（就可以看到插件了，如果看不到刷新一下）。选择gitlab插件并安装，这时会安装一些依赖的插件，勾选安装完成后重启。







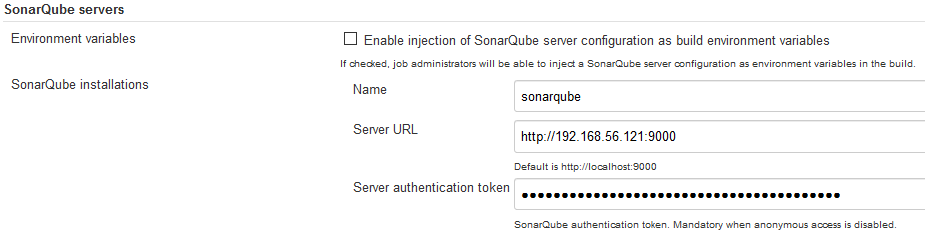
## 对接gitlab和sonarqube

### 系统设置

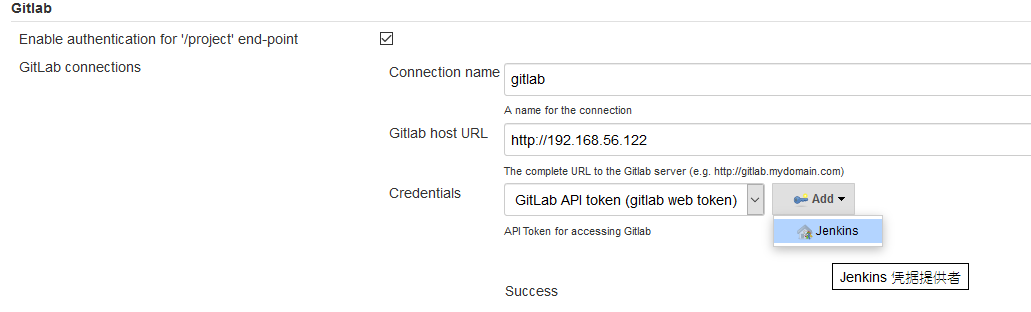
1.浏览器打开jenkins后，选择系统管理--->系统设置



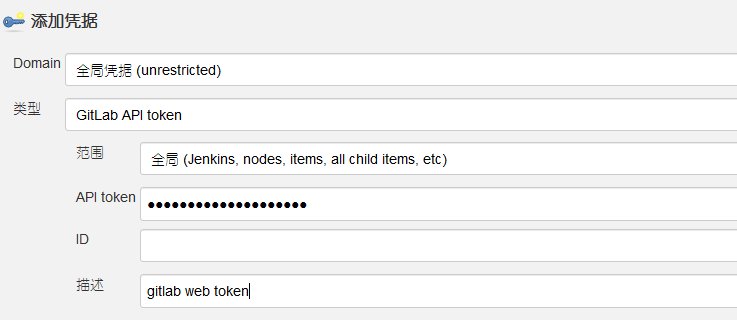
2.填写sonaqube servers的信息。



3.填写gitlab的设置，需要添加gitlab的授权信息



4.gitlab的授权信息



### 全局工具设置

1.浏览器打开jenkins，系统管理--->全局工具配置



2.填写jdk安装的目录



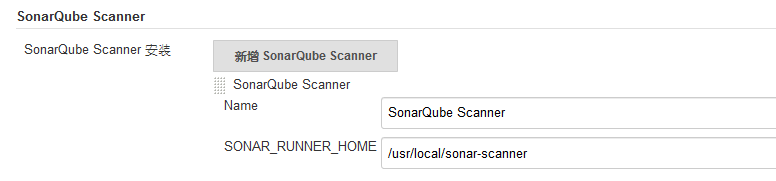
3.填写sonarqube scanner的安装目录

3.1.jenkins本地安装sonarqube scanner

unzip -q sonar-scanner-cli-3.2.0.1227-linux.zip -d /usr/local/

ln -s /usr/local/sonar-scanner-3.2.0.1227-linux/ /usr/local/sonar-scanner

3.2.配置路径

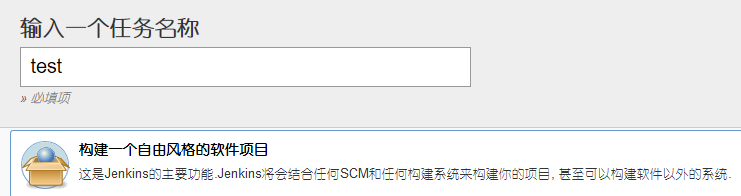


## 新建项目测试

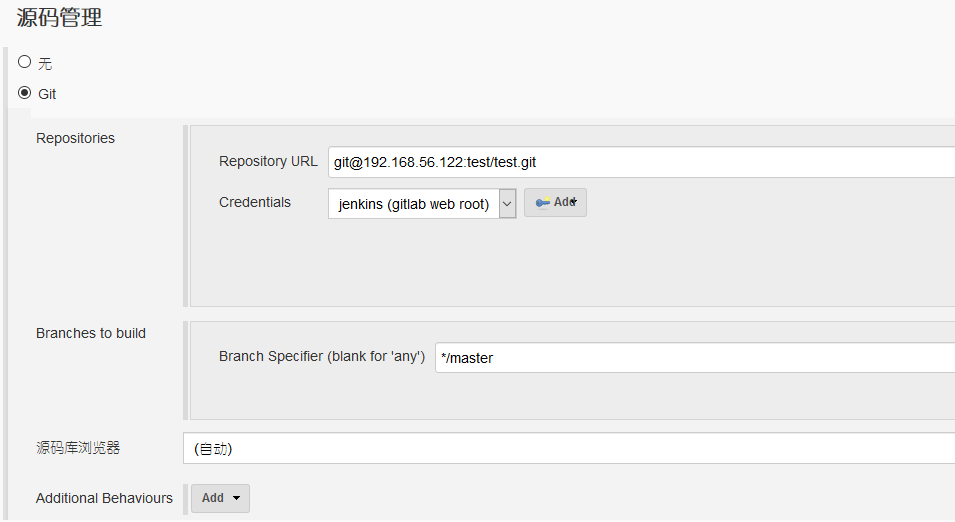
1.登录jenkins的首页后



2.创建一个新的任务



3.源码管理选择git



3.1.这里需要添加私钥

[root@gitlab ~]# cat .ssh/id\_rsa

-----BEGIN RSA PRIVATE KEY-----

MIIEpAIBAAKCAQEA1u5+cBHdEpCyT+CYzbmLg56yxc70Iw9QKgQLNpn+Xyu2bdSx

1xLJnLp4NRE8n+NxLS7B/ttnX2wZqrz1UJNsAbmB7MHJPMUVRubDkNhzIpE3HhC2

M7sG13ZjgNeJzSgLtV7w6STBg/AHyaqy6mBweCkCkqQB5ChSeV46LW0u5bX9VCmH

5PFFRCpFc9XvKLKa/cFX6O7aLB/s4m5iE8rLQQaeFogYDYujoRjfB+SpjT/e09qP

r5NWa12UwOEDgNbPn7Ea8Ivqqmg9qHVEea/UIYkMFmEmxthTeIxS6wFfXpQjF3tU

ydxi0aTRdH95SWSCoaTClOY+Nil7BOChrRZE+wIDAQABAoIBAFyQsFGRz/VL7g+q

ugR38ZcJBRNeExT6ZlsCDNZCMiRIzjO1E7Z99+iRAm5QcxylNrvv1ErvsTCOfOtk

n/5Iz45al5MgzotmkchMYFOEdOmhQuym4+GZndOzwBzgNw/tPEWrkvPGCbXJ7YC0

DIQWbrnBZ8NB8TKTDxoIM+QWjskM8kw5OaSZ6ZTbZWP3bfXWRMNZkqSxl8N5mdEN

XCx9/ZJthNYiMBSkF9d1XzNtjApXHU+WWG4P/qHZipuk+Icl7aBYhfdOksPXqa+x

oj5iPtAscfM/wHTAplP4WmibmjuN8W4pcImfPTWeei4TPFkGMGjJPMoxtL2mFZPz

gl29+QkCgYEA/mC1/GcMnLO9sBAybK7O9UXxRf11kKRl0GNaK94CTV0F3OVV7TsA

6yKx8wVVgP3SyUuNRhyI7Dk+b9R3yE+MDokwWMnLXswNu3r9Vqvw0eylvpUdPkk7

6jHOpvN3/La71Y72NtC7Syq+u5p0De5E8IfFkAxhheUxrDDUic7Yd0cCgYEA2E1i

bAeUVUbUoc1++frMd9YjxIzvcAgXTXo3QY7W+dZMIf/IU+uHMCYfPriM7pqpky51

NpimsXgTqFGJOC4j/cEk4sS/BZ7AQn581hqL1YPZR4Deb5xNhRacBMofikyQKRIv

aunGnKMekzGaNEDlrFKUnmz+4lDsmr03oqLuBq0CgYEA1OWYKlEMyzHbtjk4GqIL

cFhDYHKLLK2K0Of2FfTPNHSAbao7Sjsxv5ngGScDstu8yOBEssM7DYmzaeEHB00c

SSYkoorxfLuHWrvKv/lTqj5HUcR3HyQn0xAwhqYKbCV7+3aVNrdnqbdyvoy8SNR8

OltHzBGwpWF3O0f6+gN2RA0CgYEA1NmZr6rpud0AotRgTkgRax+LhQAXIoyjubc5

psaUH7hUpORaUtKj7qqk9oiPDsGde1stNvOSnXMHDQxnVqR5AF1PGeP4AeTNVuG3

oMzwq8SaTPWwbe5Gjl/IWiI4VVSwS+o8RR4VNtCB5uDZ2O+/0wpUe23QiA6QNlla

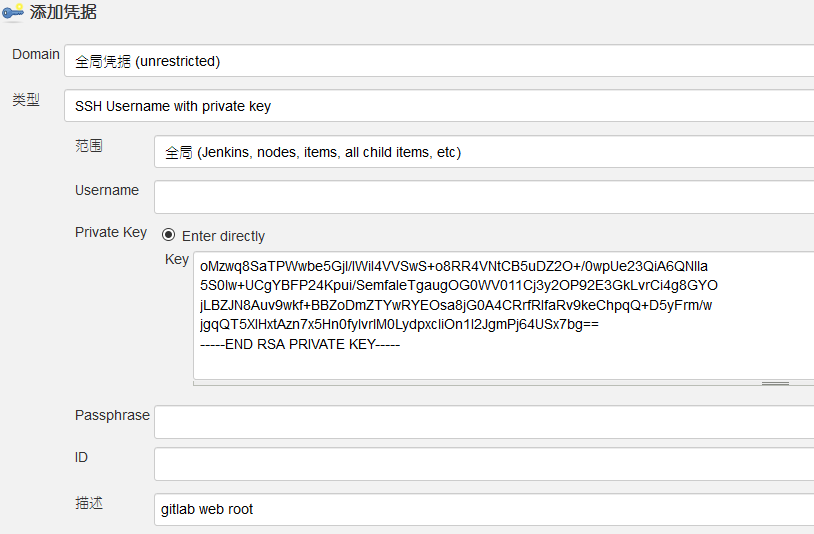
5S0lw+UCgYBFP24Kpui/SemfaIeTgaugOG0WV011Cj3y2OP92E3GkLvrCi4g8GYO

jLBZJN8Auv9wkf+BBZoDmZTYwRYEOsa8jG0A4CRrfRIfaRv9keChpqQ+D5yFrm/w

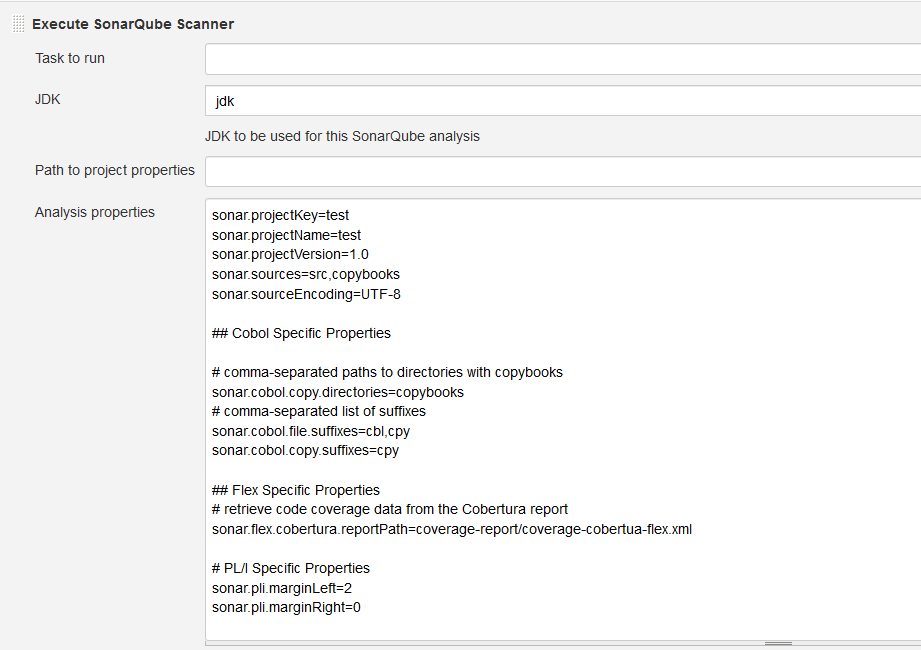
jgqQT5XIHxtAzn7x5Hn0fylvrlM0LydpxcliOn1I2JgmPj64USx7bg==

-----END RSA PRIVATE KEY-----

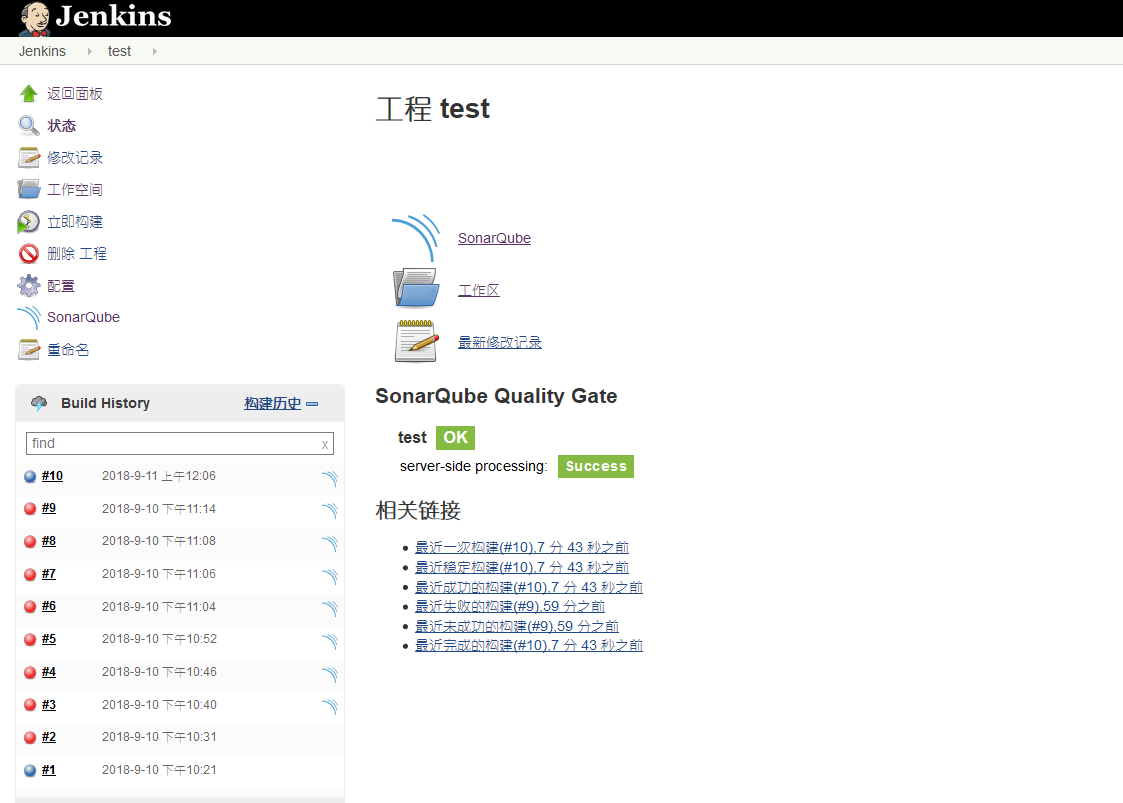
3.2.点击add选择ssh key认证，复制上面的私钥



4.增加构建的步骤



5.执行构建并查看结果



6.点击左侧的SonarQube可以进入代码质检页面

