Jenkins + Sonar 代码检查

Name：曲中岭

Email:zlinggu@126.com

Q Q :441869115

第一章 安装准备

1.1 规划

OS ：Ubuntu 14.04.4 x64
IP ：172.16.6.31
JDK ：1.8.0_171
Tomcat ：8.5.33.0
Jenkins ：2.121.3
目录：/data/tomcat-jenkins-8080/webapps/Jenkins
web: http://172.16.6.31:8080/jenkins/
账单密码：admin/admin
SonarQube:6.7.5
目录：/data/sonarqube-6.7.5
web: http://172.16.6.31:9000
账号密码：admin/admin
Sonar-scanner:2.8
目录：/data/sonar-scanner-2.8
MySQL:5.7.11
jdbc:mysql://10.1.5.15:3306
用户密码：sonar/123456

1.2 创建程序用户

useradd lx2
sonarqube 不能以 root 用户运行，否则无法启动起来

1.3 open files 参数

保证使用 ulimit –n 输出的是不小于 65536 的数，否则 sonarqube 起不来

```
[lx2@yunwei data]$ ulimit -n
65536
[lx2@yunwei data]$
```

1.4 MySQL 准备

mysql 按照步骤略。

保证 mysql 版本不低于 5.6，否则 sonarqube 起不来

创建 mysql 用户及 sonar 数据库：

```
CREATE DATABASE sonar CHARACTER SET utf8 COLLATE utf8_general_ci;
GRANT ALL ON sonar.* TO 'sonar'@'%' IDENTIFIED BY '123456';
FLUSH PRIVILEGES;
```

1.5 jdk 安装

保证能看到如下效果

```
[lx2@yunwei ~]$ java -version

java version "1.8.0_171"
Java(TM) SE Runtime Environment (build 1.8.0_171-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.171-b11, mixed mode)
[lx2@yunwei ~]$
[lx2@yunwei ~]$
[lx2@yunwei ~]$ which java
/usr/local/jdk/bin/java
[lx2@yunwei ~]$
```

步骤这里略过。

1.6 tomcat 安装

用来运行 jenkines，步骤略

第二章 SonarQube 安装配置

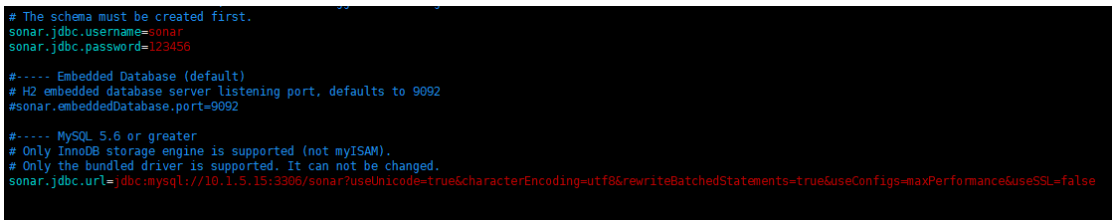
2.1 sonar 下载

到这里
https://www.sonarqube.org/downloads/
选择合适版本，我选了 6.7.5 版本
并解压到规划的路径

2.2 修改配置文件

cat /data/sonarqube-6.7.5/conf/sonar.properties
保证有以下三行:

```
sonar.jdbc.username=sonar
sonar.jdbc.password=123456
sonar.jdbc.url=jdbc:mysql://10.1.5.15:3306/sonar?useUnicode=true&characterEncoding=utf8&rewriteBatchedStatements=true&useConfigs=maxPerformance&useSSL=false
```

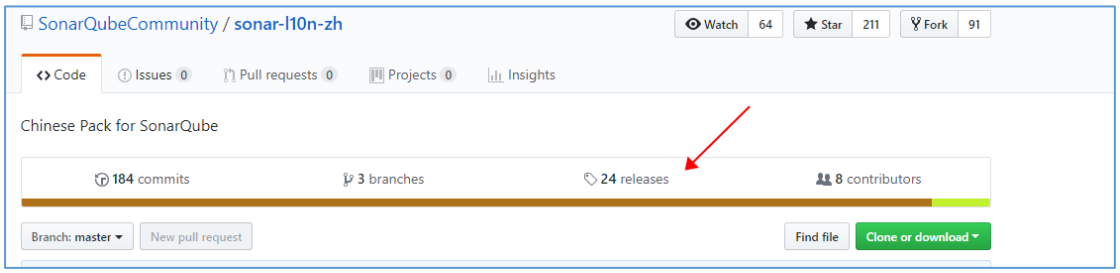


2.3 启动

/data/sonarqube-6.7.5/bin/linux-x86-64/sonar.sh start
另外，该命令也可使用 stop|restart|status 等参数

2.4 汉化

到如下
https://github.com/SonarQubeCommunity/sonar-l10n-zh
地址，选择合适版本



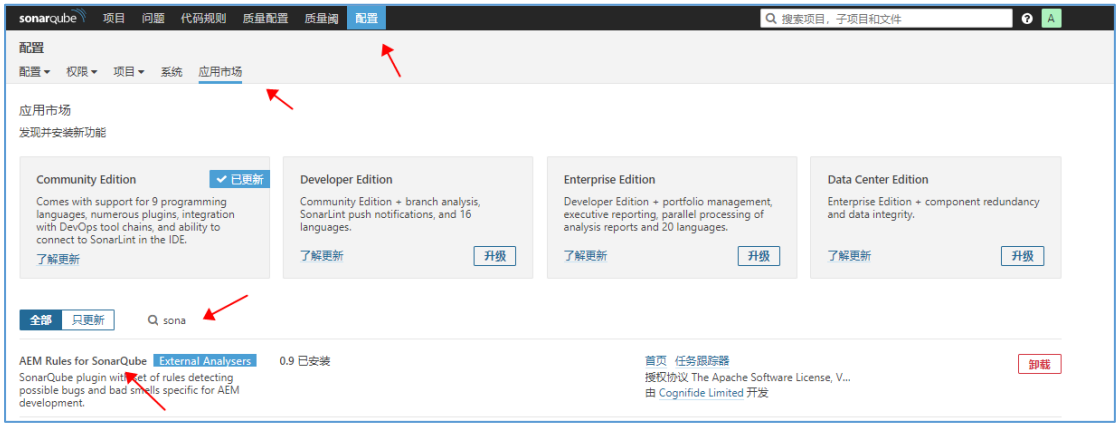
下载 jar 包，放到
/data/sonarqube-6.7.5/extensions/plugins
目录

```
[lx2@yunwei plugins]$ pwd
/data/sonarqube-6.7.5/extensions/plugins
[lx2@yunwei plugins]$
[lx2@yunwei plugins]$ ls sonar-l10n-zh-plugin-1.19.jar
sonar-l10n-zh-plugin-1.19.jar
[lx2@yunwei plugins]$
```

重启服务，汉化就生效了

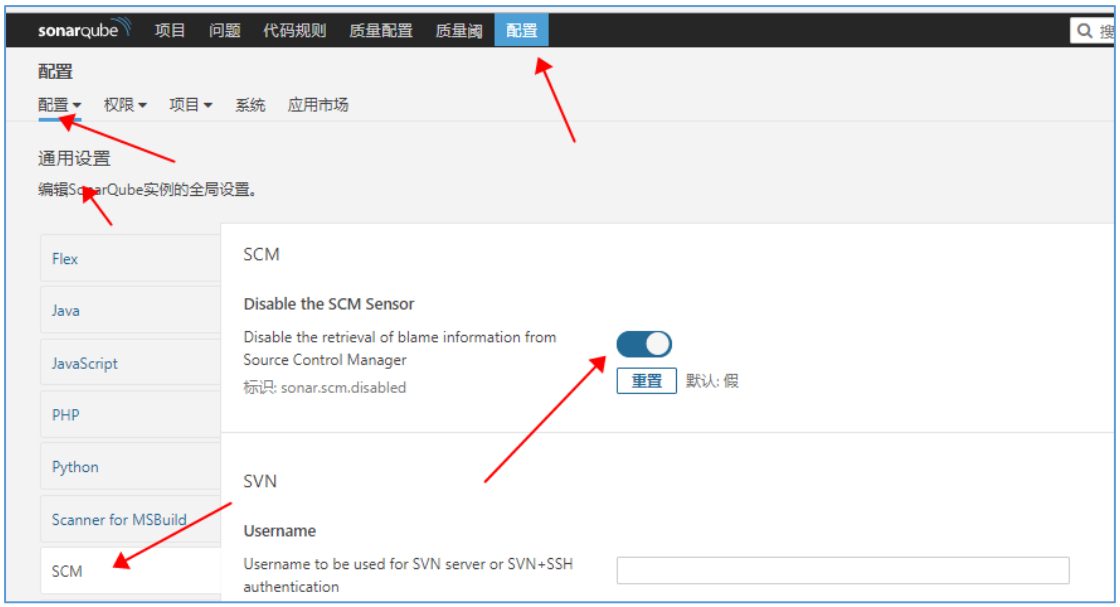
2.5 安装插件

如下图所示，安装插件
AEM Rules for SonarQube



2.6 禁用 scm

如下图所示，禁用 scm 功能



2.7 生成 token

如下图，点击即可生成 token

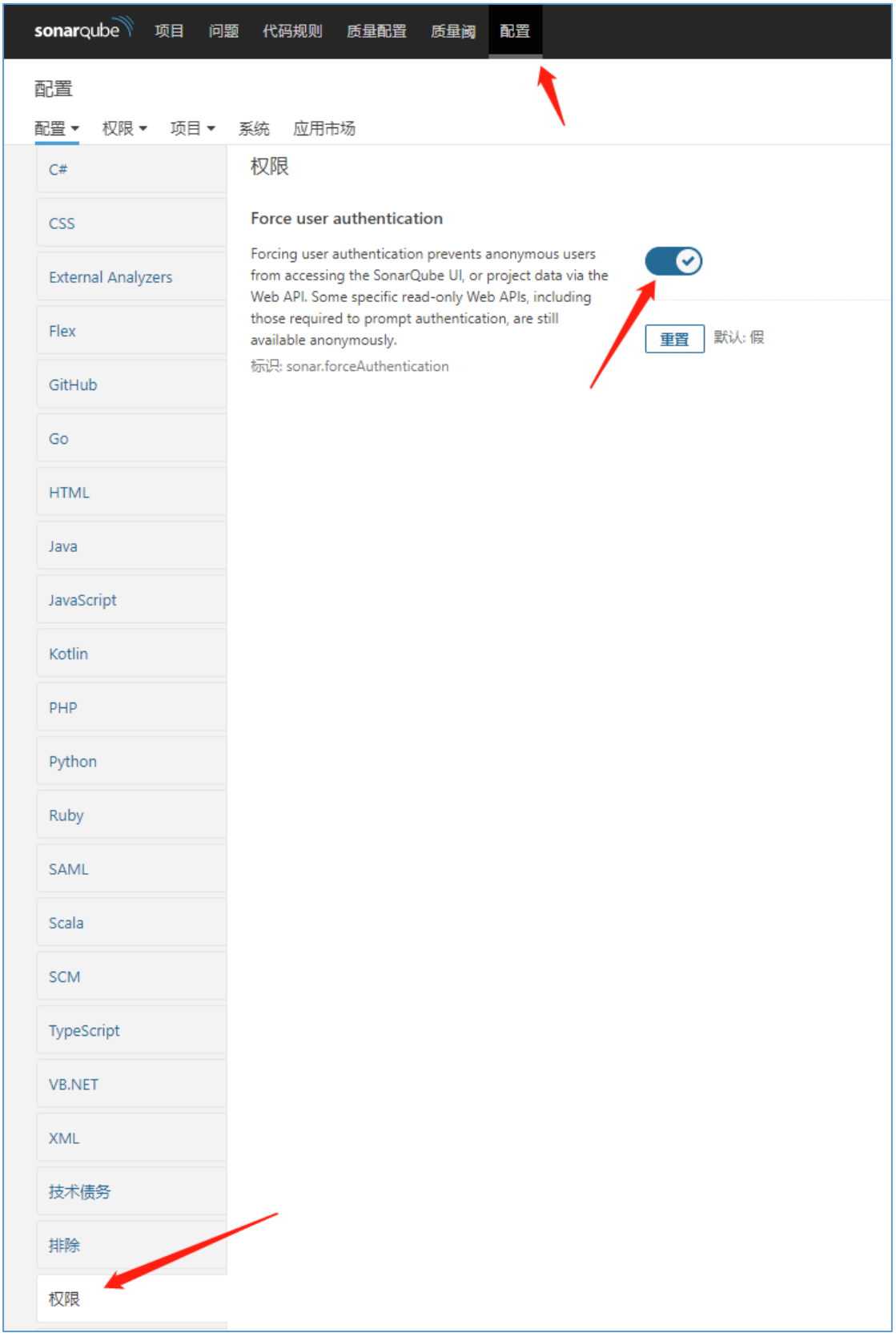
只显示一次，注意记录，比如我的 token 是

7a4ee23f4dc52b16cb625021efd151d1a8e15fdb

2.7 取消匿名登录

设置后，必须经过登录认证才可以看到内容。

如下图 配置—>权限-->Force user authentication


sonarqube
项目
问题
代码规则
质量配置
质量阈
配置
配置
配置
权限
项目
系统
应用市场
C#
CSS
External Analyzers
Flex
GitHub
Go
HTML
Java
JavaScript
Kotlin
PHP
Python
Ruby
SAML
Scala
SCM
TypeScript
VB.NET
XML
技术债务
排除
权限
权限
Force user authentication
Forcing user authentication prevents anonymous users from accessing the SonarQube UI, or project data via the Web API. Some specific read-only Web APIs, including those required to prompt authentication, are still available anonymously.
标识: sonar.forceAuthentication
重置
默认: 假

第三章 SonarQube Scanner

操作对象：jenkins 所在服务器

3.1 下载和解压

到
https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/
选择合适版本，下载，比如我的
wget https://sonarsource.bintray.com/Distribution/sonar-scanner-cli/sonar-scanner-2.8.zip

然后解压到规划的的目录

./sonar-scanner -h
出现如下内容，表示安装好了

```
[root@jenkins bin]# ./sonar-scanner -h
INFO:
INFO: usage: sonar-scanner [options]
INFO:
INFO: Options:
INFO:  -D,--define <arg>     Define property
INFO:  -h,--help             Display help information
INFO:  -v,--version          Display version information
INFO:  -X,--debug            Produce execution debug output
```

3.2 修改配置文件（可选）

cat /data/sonar-scanner-2.8/conf/sonar-scanner.properties
保证有以下两行即可：
sonar.host.url=http://172.16.6.31:9000
默认是
sonar.host.url=http://localhost:9000

```
[lx2@yunwei conf]$ pwd
/data/sonar-scanner-2.8/conf
[lx2@yunwei conf]$
[lx2@yunwei conf]$ grep -v \# sonar-scanner.properties

sonar.host.url=http://172.16.6.31:9000
```

如果 sonar 和 sonar-scanner 在同一台机器，可不修改。

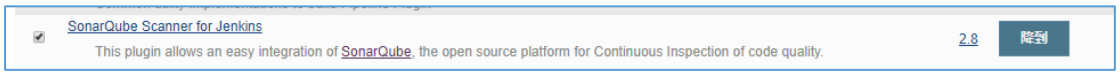
第四章 Jenkins 安装配置

4.1 下载和安装

到 jenkins 官网下载 war 包，放到 tomcat/webapps 下面，启动即可，无其他特殊配置，比较简单，这里略过

4.2 插件安装

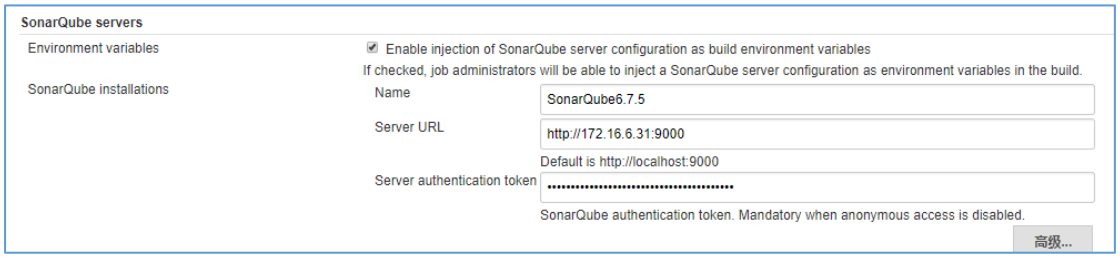
安装插件 SonarQube Scanner for Jenkins



4.3 SonarQube 配置

系统配置—>系统配置，找到
SonarQube installations
段，按照下面进行配置

```
Name                          : SonarQube6.7.5     #随意写
Server URL                    : http://172.16.6.31:9000   #准确填写
Server authentication token : **** #2.7 生成的 token
```



4.4 SonarQube Scanner 配置（可选）

系统配置—>全局工具设置，找到 SonarQube Scanner 部分，进行配置

```
Name ： sonar-scanner-2.8  #随意写
SONAR_RUNNER_HOME ：  /data/sonar-scanner-2.8  #这个一定要准确填写
```

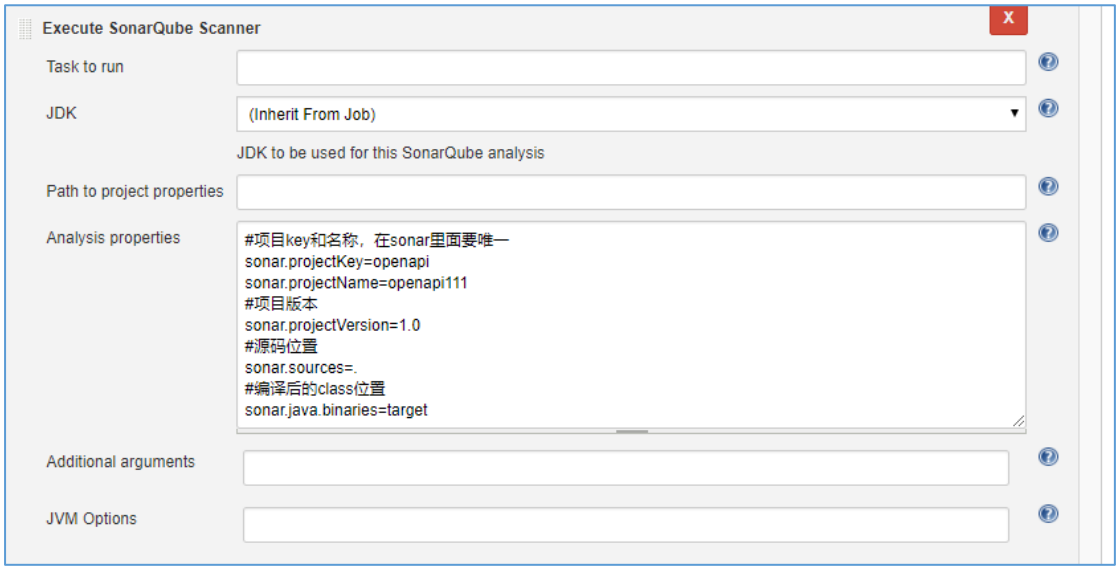
其实这一步也可以配置到操作系统环境变量里面，添加之后就不需要进行这一步了。

```
export SONAR_SCANNER_HOME=/data/sonar-scanner-2.8
export PATH=${SONAR_SCANNER_HOME}/bin:${PATH}
```

4.5 构建任务时添加步骤

添加构建步骤
Execute SonarQube Scanner
动作，内容如下

```
#项目 key 和名称，在 sonar 里面要唯一
sonar.projectKey=openapi
sonar.projectName=openapi111
#项目版本
sonar.projectVersion=1.0
#源码位置
sonar.sources=.
#编译后的 class 位置
sonar.java.binaries=target
```


Execute SonarQube Scanner
Task to run
JDK
(Inherit From Job)
JDK to be used for this SonarQube analysis
Path to project properties
Analysis properties
#项目key和名称，在sonar里面要唯一
sonar.projectKey=openapi
sonar.projectName=openapi111
#项目版本
sonar.projectVersion=1.0
#源码位置
sonar.sources=.
#编译后的class位置
sonar.java.binaries=target
Additional arguments
JVM Options

第五章 观察和排错

5.1 web 观察效果

登陆 sonar 的 web， http://172.16.6.31:9000/projects

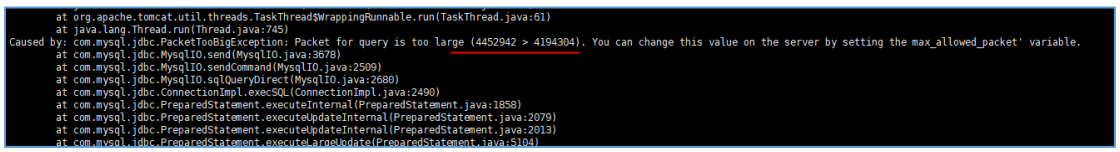
可进入具体项目查看 bugs 等详细内容

5.2 异常处理

构建日志中出现如下错误:

```
INFO: ------------------------------------------------------------------------
INFO: Total time: 2:22.742s
INFO: Final Memory: 61M/771M
INFO: ------------------------------------------------------------------------
ERROR: Error during SonarQube Scanner execution
ERROR: Failed to upload report - 500: An error has occurred. Please contact your administrator
ERROR:
ERROR: Re-run SonarQube Scanner using the -X switch to enable full debug logging.
ERROR: SonarQube scanner exited with non-zero code: 1
Finished: FAILURE
```

查看 sonarqude 日志/data/sonarqube-6.7.5/logs/web.log：
出现如下错误



Caused by: com.mysql.jdbc.PacketTooBigException: Packet for query is too large (4452942 > 4194304). You can change this value on the server by setting the max_allowed_packet' variable.

此错误表示，发到 mysql 的包太大，超过了显示，此时需要检查
第一处：mysql 的 max_allowed_packet 参数，如下

```
mysql> show variables like 'max_allowed_packet';
+--------------------+----------+
| Variable_name      | Value    |
+--------------------+----------+
| max_allowed_packet | 33554432 |
+--------------------+----------+
1 row in set
```

mysql 配置为 32M，这个配置没问题，如果有问题，请解决这个问题

第二处：修改/data/sonarqube-6.7.5/conf/sonar.propertie 内容如下
sonar.jdbc.url=jdbc:mysql://10.1.5.15:3306/sonar?useUnicode=true&characterEncoding=utf8&rewriteBatchedStatements=true&useConfigs=maxPerformance&maxAllowedPacket=200000000&useSSL=false

其实就是在 2.2 步基础上添加&maxAllowedPacket=200000000 参数

第六章 添加单元测试内容

经过以上几步，已经实现了基本的代码检测功能，但还有一项就是单元测试覆盖率显示为 0，如下图

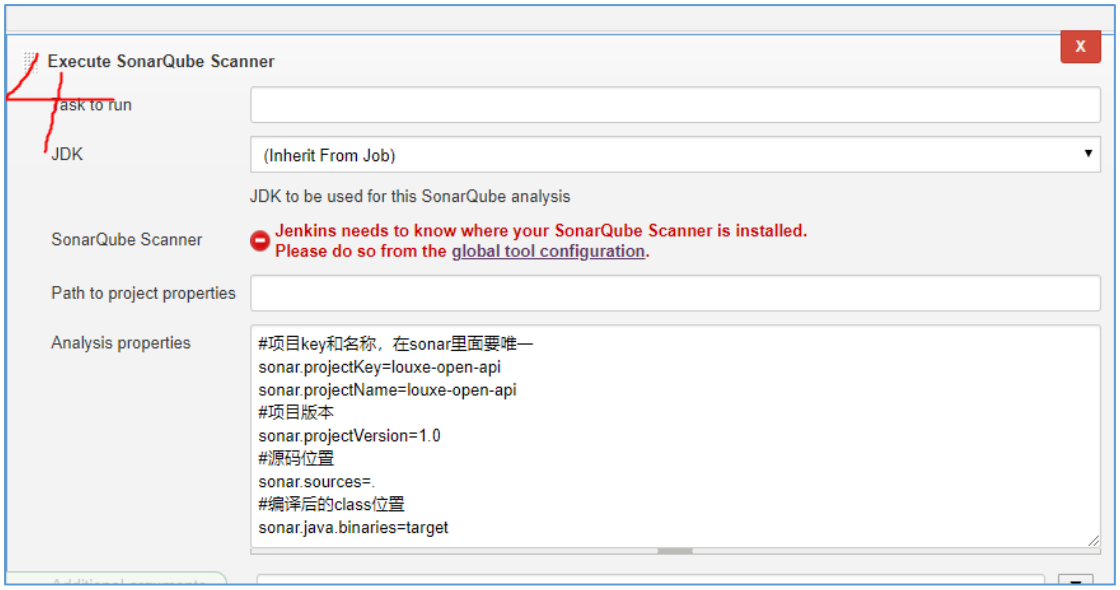
下面开始处理这部分：

6.1 java

6.1.1 添加构建步骤

在 Execute SonarQube Scanner 步骤前，

在 ansible 部署这一步的后面
添加如下两行
mvn clean org.jacoco:jacoco-maven-plugin:prepare-agent install -Dmaven.test.failure.ignore=true
mvn sonar:sonar
如图



一定要注意前后顺序，

1） 放到部署这一步的后面，因为这一步会导致编译出来的 jar 包被修改

2） 放到上报这一步前面，是为了保证上报时把这一步的执行结果也一并上报

6.1.2 6.2 pom.xml 配置

保证未跳过单元测试，设置为 false，或者不配置这一段内容。

```
<plugin>
        <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-surefire-plugin</artifactId>
         <configuration>
                <skip>false</skip>
          </configuration>
</plugin>
```

如下图：

```
<plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <configuration>
                <skip>false</skip>
        </configuration>
</plugin>
```

6.1.3 观察效果

6. 2 node.js

6.2.1 研发人员关注:

单元测试的样例,写到项目根目录 test 下面,使用 js 结尾,例如项目:https://gitlab.dm-ai.cn/cum/cum-express

.dockerignore 里面保证有以下两行

coverage

test

6.2.2 运维人员关注，添加构建步骤

生成报告:

npm config set registry http://192.168.3.13:8081/repository/npm && npm install

npm install istanbul

istanbul cover test/*.js --my test args

注意:

生成的报告内容位于 coverage 目录下面,有 lcov 和 json 两种格式,sonar 使用 lcov 格式的.

上报到 sonar-server:

sonar.projectKey=$pname

sonar.projectName=$pname

sonar.projectVersion=1.0

sonar.sources=./

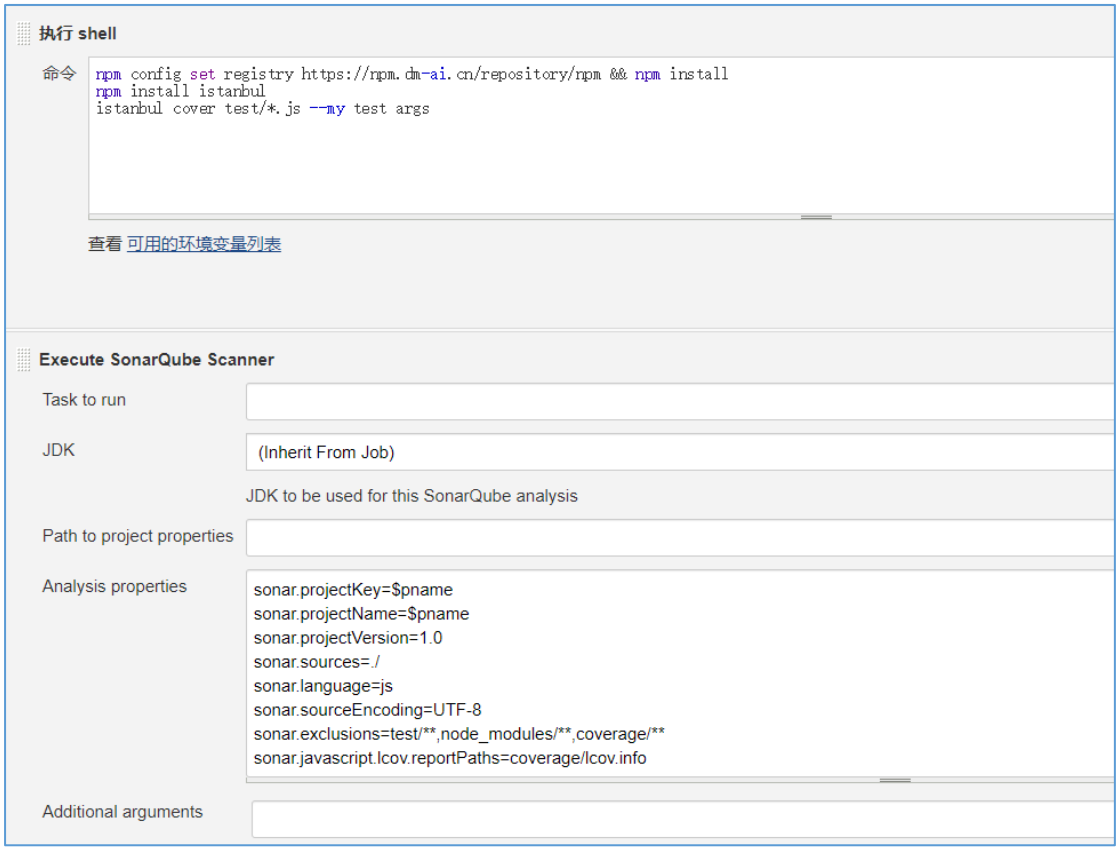
sonar.language=js

sonar.sourceEncoding=UTF-8

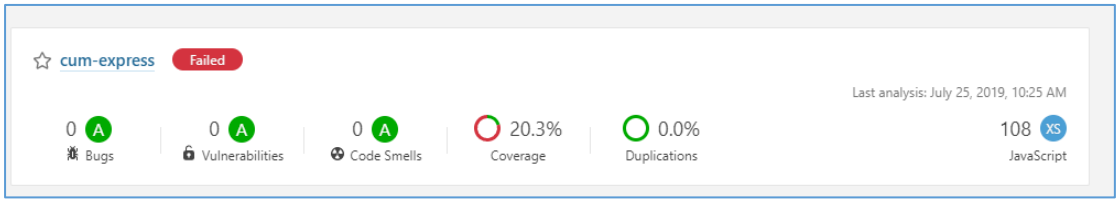
sonar.exclusions=test/**,node_modules/**,coverage/**

sonar.javascript.lcov.reportPaths=coverage/lcov.info

如图:



6.2.3 观察效果



第七章 使用 docker 部署

7.1 sonarqube 部署

注意 7.8 是最后一个支持 mysql 的版本，7.9 及之后只支持 pgsql。

yml 文件：

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sonarqube
  namespace: devops
spec:
  replicas: 1
  selector:
    matchLabels:
      app: sonarqube
  template:
    metadata:
      labels:
        app: sonarqube
    spec:
      imagePullSecrets:
      - name: regsecret
      containers:
      - name: sonarqube
        image: sonarqube:7.8-community
        imagePullPolicy: Always
        ports:
        - name: port-9000
          containerPort: 9000
        volumeMounts:
          - name: conf
            mountPath: /opt/sonarqube/conf
          - name: data
            mountPath: /opt/sonarqube/data
          - name: logs
            mountPath: /opt/sonarqube/logs
          - name: plugins
            mountPath: /opt/sonarqube/extensions/plugins
        env:
        - name: TZ
```

```
            value: Asia/Shanghai
          #- name: JAVA_OPTS
          #  value: "-Duser.timezone=Asia/Shanghai -DsessionTimeout=10080 -Dpermissive-script-security.enabled=true"
          - name: SONARQUBE_JDBC_USERNAME
            value: "sonar"
          - name: SONARQUBE_JDBC_PASSWORD
            value: "1MDc3MGJhUmVlIiwiYXV0aCI6I"
          - name: SONARQUBE_JDBC_URL
            value: "jdbc:mysql://192.168.3.151:3306/sonar?useUnicode=true&characterEncoding=utf8&rewriteBatchedStatements=true&useConfigs=maxPerformance&maxAllowedPacket=200000000&useSSL=false"
        volumes:
        - name: conf
          hostPath:
            path: /data/sonar/conf
        - name: data
          hostPath:
            path: /data/sonar/data
        - name: logs
          hostPath:
            path: /data/sonar/logs
        - name: plugins
          hostPath:
            path: /data/sonar/plugins
---
apiVersion: v1
kind: Service
metadata:
  name: sonar
  namespace: devops
spec:
  ports:
    - name: sonar-http
      port: 80
      targetPort: 9000
  selector:
    app: sonarqube
```

ingress 文件：

```
kind: Ingress
apiVersion: extensions/v1beta1
metadata:
```

```
  name: sonar
  namespace: devops
  annotations:
    kubernetes.io/ingress.class: traefik
spec:
  rules:
  - host: sonar.ops.dm-ai.cn
    http:
      paths:
      - backend:
          serviceName: sonar
          servicePort: 80
```

7.2 sonar-scanner 部署

cat sonar-scanner-4.0.0.1744-linux/conf/sonar-scanner.properties
sonar.host.url=http://sonar.ops.dm-ai.cn

dockerfile 第一种写法，不需要 jre 目录，使用系统的 java 环境，打出来的镜像，大概 151M

```
ROM openjdk:8-alpine
ENV TZ=Asia/Shanghai
ENV SONAR_RUNNER_HOME=/sonar-scanner
WORKDIR /sonar-scanner
COPY sonar-scanner-4.0.0.1744-linux .
RUN sed -i 's/use_embedded_jre=true/use_embedded_jre=false/g' bin/sonar-scanner && \
    apk add --no-cache curl grep sed unzip bash nodejs nodejs-npm
```

dockerfile 第二种写法：
需要有 jre 目录，系统没有 java 环境。打出来的镜像，大概 277M。

```
FROM ubuntu:18.04
ENV TZ=Asia/Shanghai
ENV SONAR_RUNNER_HOME=/sonar-scanner
WORKDIR /sonar-scanner
COPY sonar-scanner-4.0.0.1744-linux .
RUN apt update && \
    apt install -y nodejs && \
    apt clean && \
    apt autoclean
```

测试 sonar-scanner 是否工作正常：

```
docker run -it -v \
```

```
/data/jenkins-data/workspace/xmc2-ui-backend-service:/data/  \
docker.dm-ai.cn/devops/sonar-scanner:4.0 sh
```

cd /data

#运行如下命令观察是否正常上报:

```
/sonar-scanner/bin/sonar-scanner  -Dsonar.host.url=http://sonar.ops.dm-ai.cn  -Dsonar.login=admin  -
Dsonar.password=33f5b945a8b5908793b6e -Dso
nar.language=js  -Dsonar.projectName=xmc2-ui-backend-service  -Dsonar.projectVersion=1.0  -
Dsonar.sourceEncoding=UTF-8  -Dsonar.projectKey=xmc2-ui-backend-service  -
Dsonar.exclusions=**/node_modules/
** -Dsonar.sources=./ -Dsonar.projectBaseDir=/data/jenkins-data/workspace/xmc2-ui-backend-service
```