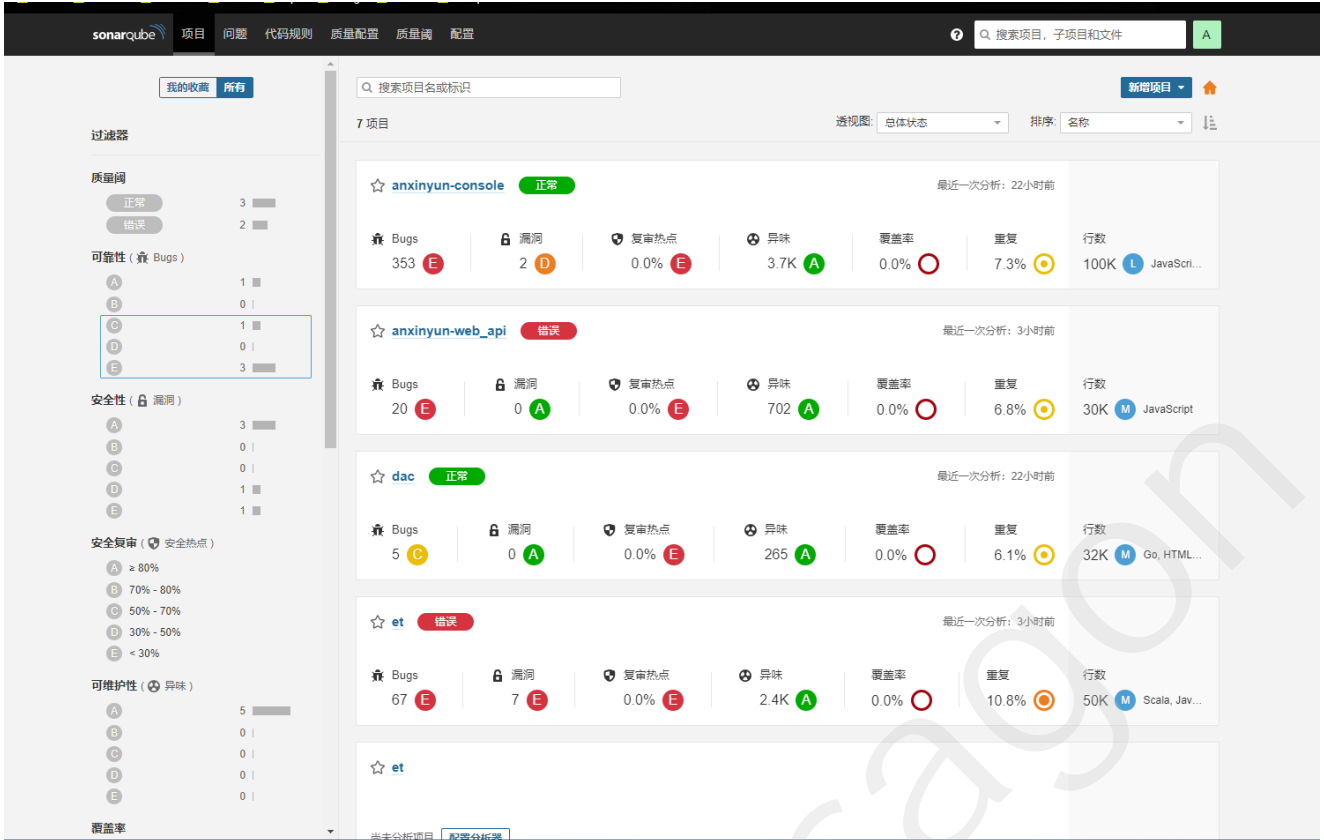


服务端部署

```
version: "3"
services:
  sonarqube:
    image: sonarqube:lts-community
    environment:
      SONAR_JDBC_URL: jdbc:postgresql://10.8.30.156:5432/sonarqube
      SONAR_JDBC_USERNAME: sonarqubeAdmin
      SONAR_JDBC_PASSWORD: sonarqubeAdmin_123
      SONAR_JDBC_MAXACTIVE: "100"
      SONAR_JDBC_MAXIDLE: "10"
      SONAR_JDBC_MINIDLE: "5"
    volumes:
      - sonarqube_data:/opt/sonarqube/data
      - sonarqube_extensions:/opt/sonarqube/extensions
      - sonarqube_logs:/opt/sonarqube/logs
    ports:
      - "9000:9000"
volumes:
  sonarqube_data:
  sonarqube_extensions:
  sonarqube_logs:
```

部署成功:



体验

在服务端新增项目:



分析你的项目

我们已经在SonarCloud初始化了你的项目，现在已准备好进行分析！

1 创建一个令牌

2 分析你的项目

使用什么构建技术？

Maven Gradle .NET 其他 (比如 JS, TS, Go, Python, PHP, ...)

在你的电脑上使用Maven执行SonarQube扫描

使用Maven执行SonarQube分析是非常简单的。只需要在你的项目目录下执行如下命令。

```
mvn sonar:sonar \
  -Dsonar.projectKey=et \
  -Dsonar.host.url=http://10.8.30.158:9000 \
  -Dsonar.login=d3ce7e00c8fcc9adc3b85b97d3c1005766cb2498
```

请访问 [Maven扫描器官方文档](#) 了解更多详情。

分析完成后，页面会自动刷新，可以看到分析结果。

这个项目是一个maven 构建的项目，项目是scala 的，

按照提示执行 `maven` 命令：

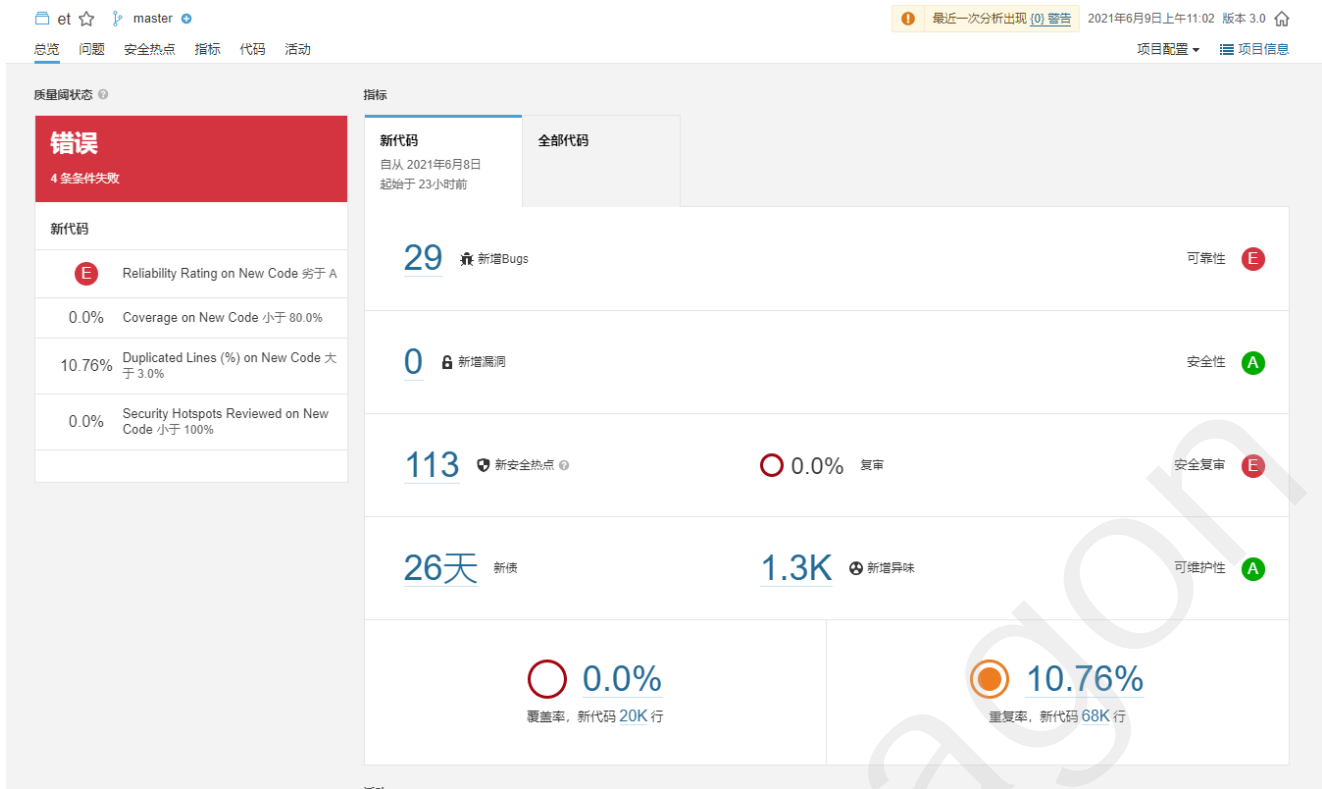
```
mvn sonar:sonar \
  -Dsonar.projectKey=et \
  -Dsonar.host.url=http://10.8.30.158:9000 \
  -Dsonar.login=d3ce7e00c8fcc9adc3b85b97d3c1005766cb2498
```

使用 `idea ide` 的 可以在安装插件，

`File` — `Settings...` — `Plugins`
查找 `sonar` 可以找到插件，安装即可

安装 [sonarscanner](#)

执行后，代码扫描完成，



可以看到总览。

Jenkins 集成

Jenkins 安装 插件 **SonarQube Scanner for Jenkins**

在 Jenkins 添加 凭据:

全局凭据 (unrestricted) ▶

类型

Secret text

范围

全局 (Jenkins, nodes, items, all child items, etc)

Secret

.....

ID

sonar

描述

sonar token

确定

在 系统管理 ——> 系统配置中 —— SonarQube servers

SonarQube servers

Environment variables Enable injection of SonarQube server configuration as build environment variables
If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

SonarQube installations

Name

fs-sonar

Server URL

http://10.8.30.158:9000

Default is http://localhost:9000

Server authentication token

sonar token ▼ 添加 ▼

SonarQube authentication token. Mandatory when anonymous access is disabled.

在项目构建配置中添加

Execute NodeJS script

Execute SonarQube Scanner

执行 Windows 批处理命令

执行 shell

Invoke Ant

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=anxinyun-et
sonar.sources=.
sonar.language=scala
sonar.java.binaries=.
sonar.exclusions property=/**.java
```

Additional arguments [?](#)

构建成功完成后:

Dashboard > anxincloud.et_flink >

返回面板

状态

修改记录

工作空间

立即构建

配置

删除 Maven project

模块

SonarQube

重命名

Maven project anxincloud.

SonarQube

工作区

最新修改

最新测试结果 (没有失败)

最新测试结果 (没有失败)

SonarQube Quality Gate

et **Failed**

server-side processing: **Success**

相关链接

- 最近一次构建(#14), 3 小时 49 分之前
- 最近稳定构建(#14), 3 小时 49 分之前
- 最近成功的构建(#14), 3 小时 49 分之前
- 最近失败的构建(#12), 4 小时 41 分之前

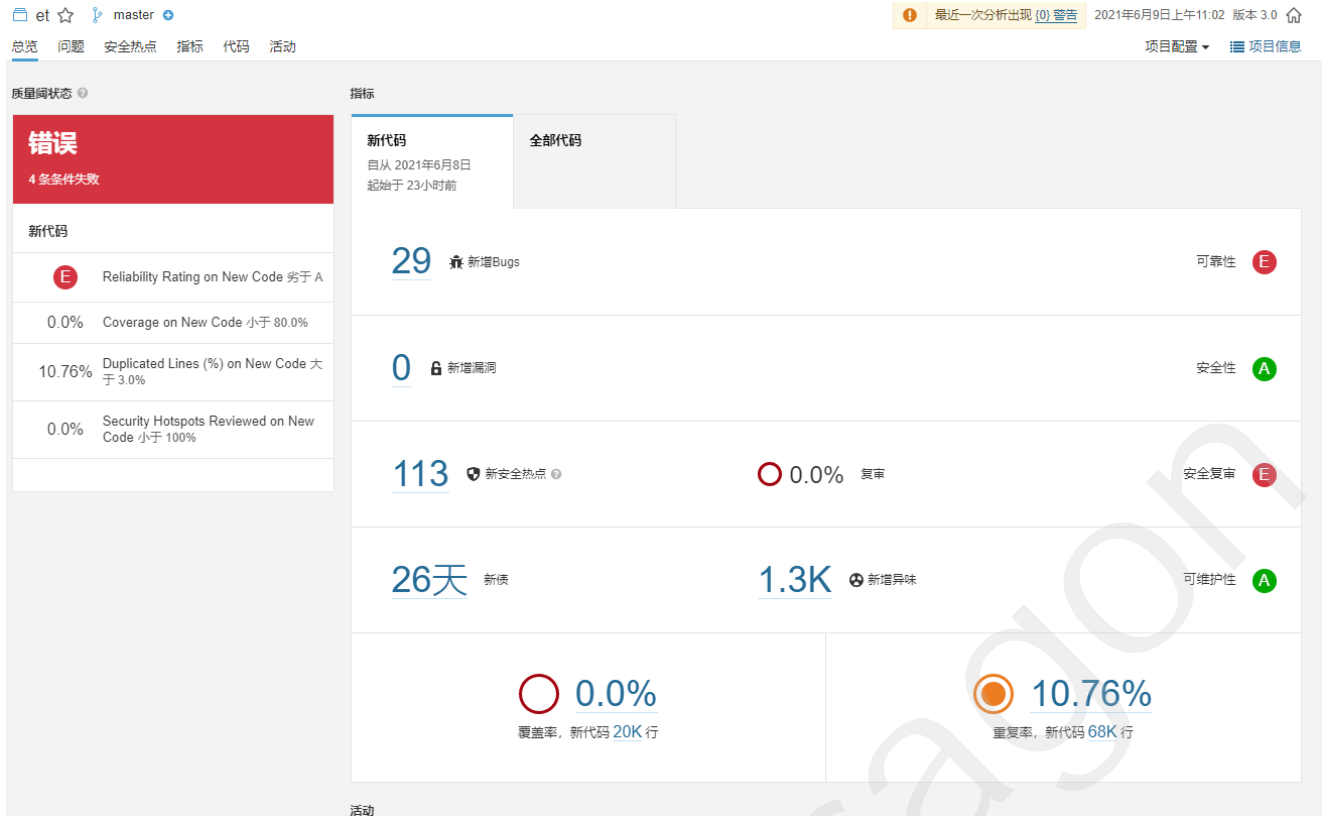
Build History 构建历史 ^

find x

#14
2021年6月9日 上午10:54 CST

#13
2021年6月9日 上午10:18 CST

点击上方框的图标，就可以看到 静态扫描的总览了



代码规则

sonarQube 支持自定义代码规则

下面以 go 语言配置为例:

Go, 1 配置	项目	规则	更新	使用
Sonar way 内置	默认	25	前天	23小时前 ⚙️

内置的规则不能直接修改，复制内置的规则模板:

Go, 1 配置	项目	规则	更新	使用
Sonar way 内置	默认	25	前天	23小时前 ⚙️
HTML, 1 配置	项目	规则	更新	

比较

复制

展开





























































复制配置 Sonar way

所有标记为 * 的都是必填字段

新名称 *

复制 取消

可以选择 启用和禁用相关规则：

 "=+" should not be used instead of "+="	Go  Bug		挂起
 "switch" statements should not have too many "case" clauses	Go  异味  brain-overload		挂起
 All branches in a conditional structure should not have exactly the same implementation	Go  Bug		挂起
 All code should be reachable	Go  Bug  cwe, unused		挂起
 Boolean checks should not be inverted	Go  异味  pitfall		挂起
 Boolean literals should not be redundant	Go  异味  clumsy		挂起
 Cognitive Complexity of functions should not be too high	Go  异味  brain-overload		挂起
 Function and method names should comply with a naming convention	Go  异味  convention		挂起
 Functions should not be empty	Go  异味  suspicious		挂起
 Functions should not have identical implementations	Go  异味  confusing, duplicate, suspicious		挂起
 Functions should not have too many parameters	Go  异味  brain-overload		挂起
 Hard-coded credentials are security-sensitive	Go  安全热点  cwe, owasp-a2, sans-top25-porous		挂起
 Identical expressions should not be used on both sides of a binary operator	Go  Bug		挂起
 Local variable and function parameter names should comply with a naming convention	Go  异味  convention		挂起
 Multi-line comments should not be empty	Go  异味		挂起
 Nested blocks of code should not be left empty	Go  异味  suspicious		挂起