

Golang 代码质量持续检测

Author: Kenny Allen

Email: kennyallen0520@gmail.com

前言

在软件开发过程中，人工检查项目代码中的漏洞和潜在的 BUG 是一件让人十分费神费力的事情，为了解决这一痛点，SonarQube诞生了，它实现了一系列代码自动检测过程，包括命名规范，代码漏洞，代码重复量等。

但是光有 SonarQube 还不能发挥出应有的高效率，一个完整的代码质量持续检测应配合代码仓库(如 gitlab) 和 Jenkins 来共同构建一个自动化过程。

环境

- Gitlab、Jenkins、SonarQube 服务都在一台物理机上的Docker中运行
- 网络 (局域网IP: 192.168.1.100)

```
bogon:sonarqube-7.2.1 kenny$ ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    options=1203<RXCSUM,TXCSUM,TXSTATUS,SW_TIMESTAMP>
    inet 127.0.0.1 netmask 0xff000000
        inet6 ::1 prefixlen 128
        inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
            nd6 options=201<PERFORMNUD,DAD>
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
XHC20: flags=0<> mtu 0
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    options=b<RXCSUM,TXCSUM,VLAN_HWTAGGING>
    ether 10:7b:44:7d:fc:f9
    inet 192.168.1.100 netmask 0xffffffff broadcast 192.168.1.255
        nd6 options=201<PERFORMNUD,DAD>
        media: autoselect (100baseTX <full-duplex,flow-control>)
        status: active
utun0: flags=8051<UP,POINTOPOINT,RUNNING,MULTICAST> mtu 2000
    inet6 fe80::8e88:cdec:2a8c:45f2%utun0 prefixlen 64 scopeid 0x6
        nd6 options=201<PERFORMNUD,DAD>
bogon:sonarqube-7.2.1 kenny$
```

- 主机科学上网代理
192.168.1.100:1087
- 模拟外网访问

```
# 修改 hosts 文件，模拟外网访问
sudo sh -c "echo '192.168.1.100 jenkins.kenny.com\n192.168.1.100
gitlab.kenny.com\n192.168.1.100 sonarqube.kenny.com' >> /etc/hosts"
```

- 工具集

名称	版本
golang	go1.10.3
docker	18.03.1-ce

搭建

接下来，我以一个完整的案例来介绍搭建过程。

Jenkins

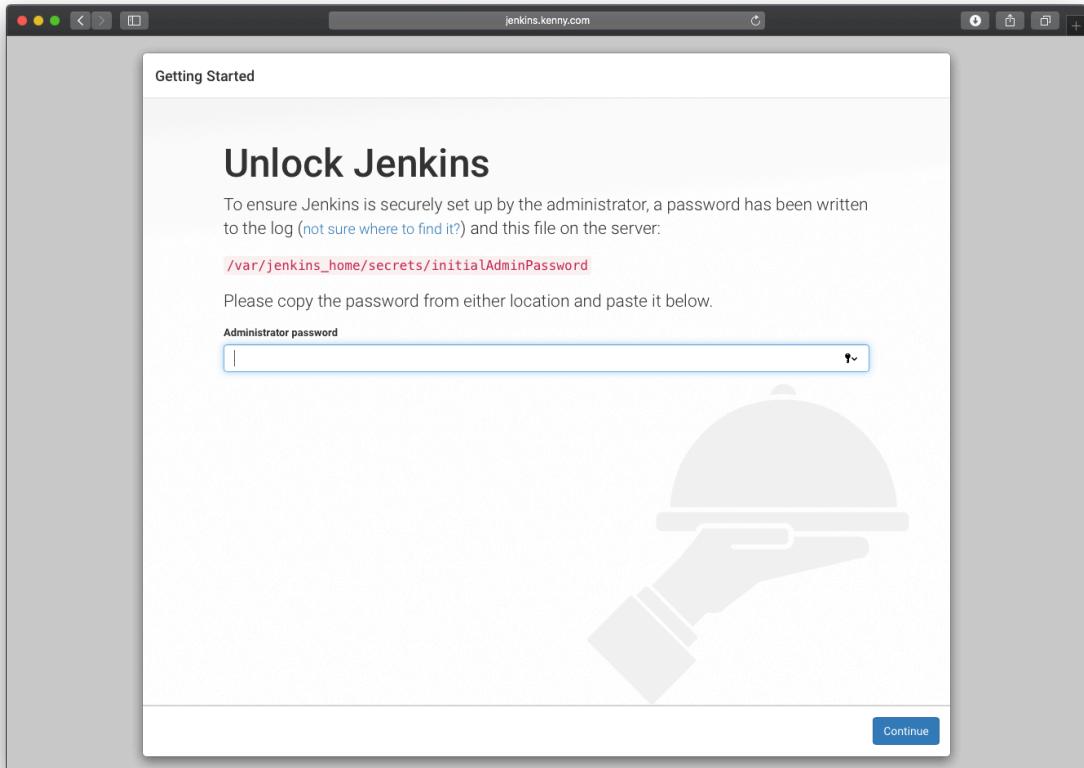
启动服务

```
# $jenkins_home 宿主机目录，挂载容器 /var/jenkins_home
# 我的数据卷目录是 ~/.jenkins
export JENKINS_HOME=~/jenkins
docker run -d --restart=always -p 8080:8080 -p 50000:50000 -v
$JENKINS_HOME:/var/jenkins_home --name jenkins jenkins

# 查看 jenkins 日志
docker logs -f jenkins
```

初始化

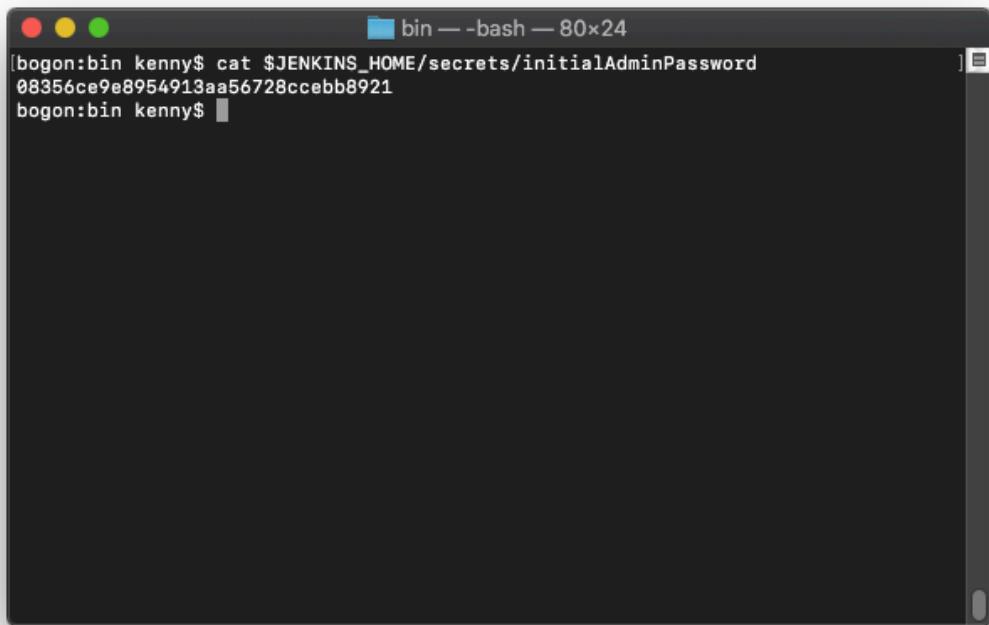
1. 打开浏览器，访问 <http://jenkins.kenny.com:8080>



```
# 在日志中找到管理员密码
docker logs -f jenkins

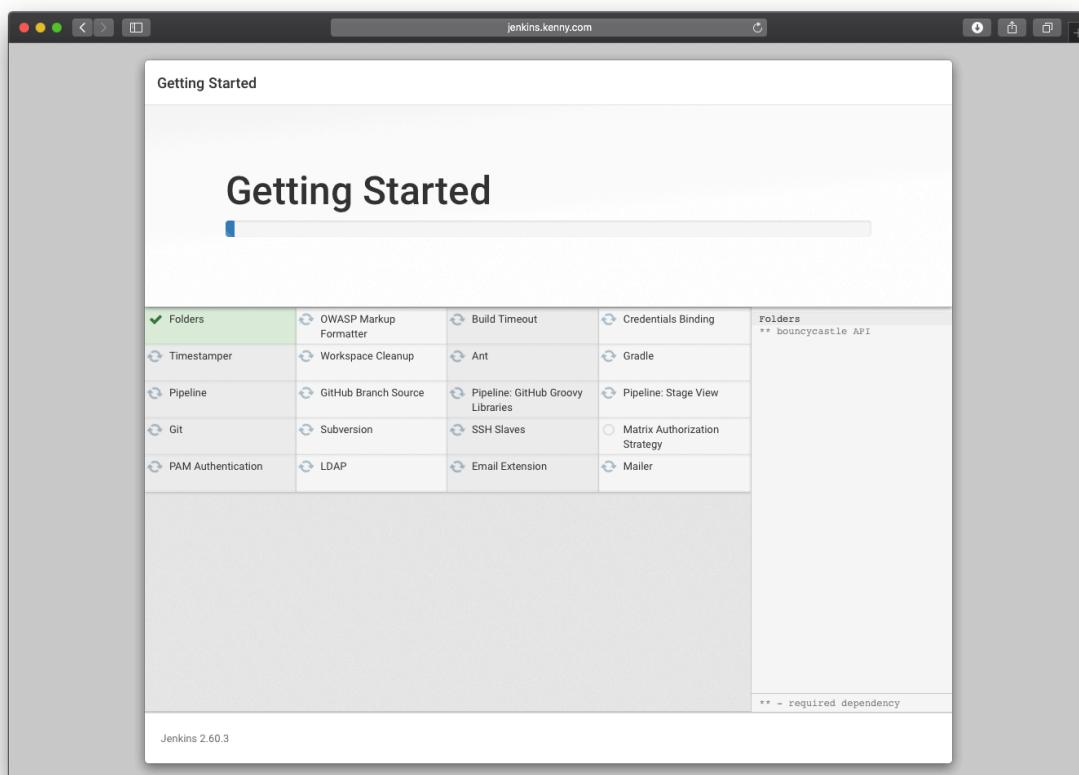
# 或者在 $JENKINS_HOME/secrets/initialAdminPassword 文件中找到管理员密码
cat $JENKINS_HOME/secrets/initialAdminPassword
```

```
bin — docker logs -f jenkins — 80x24
DefaultListableBeanFactory@1acf27f2: defining beans [filter,legacy]; root of factory hierarchy
Jul 16, 2018 1:08:53 PM jenkins.install.SetupWizard init
INFO:
*****
Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:
08356ce9e8954913aa56728cceb8921
This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
*****
Jul 16, 2018 1:08:57 PM hudson.model.UpdateSite updateData
INFO: Obtained the latest update center data file for UpdateSource default
Jul 16, 2018 1:08:58 PM hudson.model.UpdateSite updateData
```

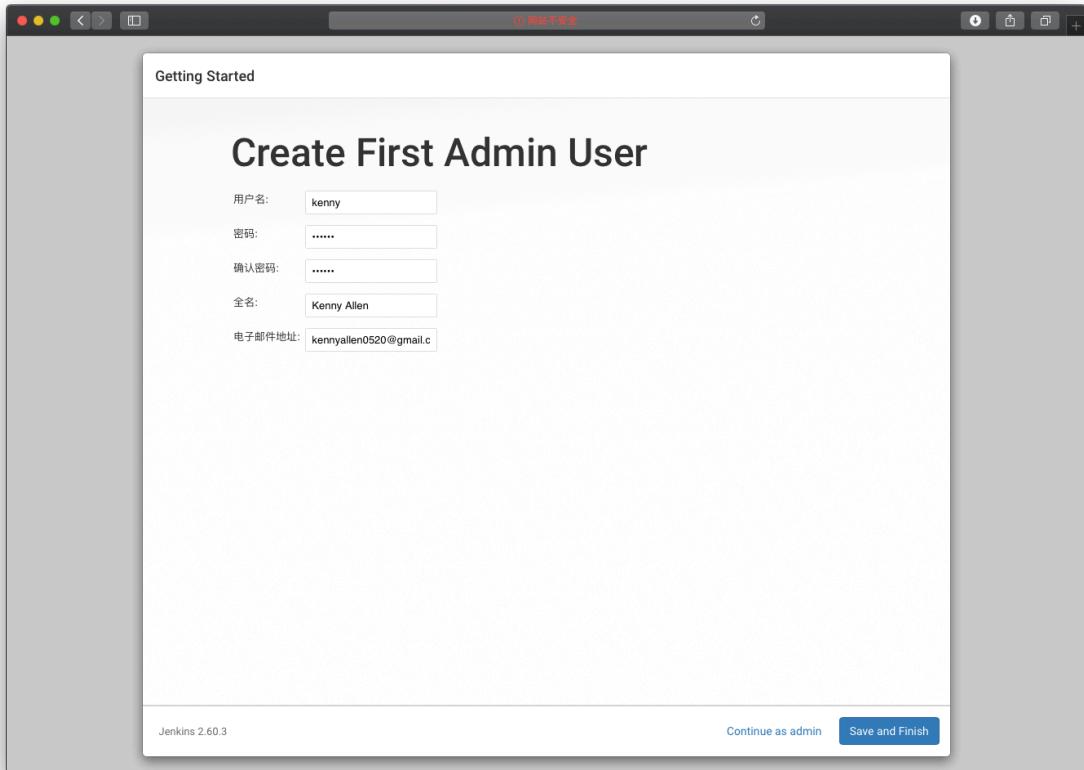


```
[bogon:bin kenny$ cat $JENKINS_HOME/secrets/initialAdminPassword  
08356ce9e8954913aa56728ccebb8921  
bogon:bin kenny$ ]
```

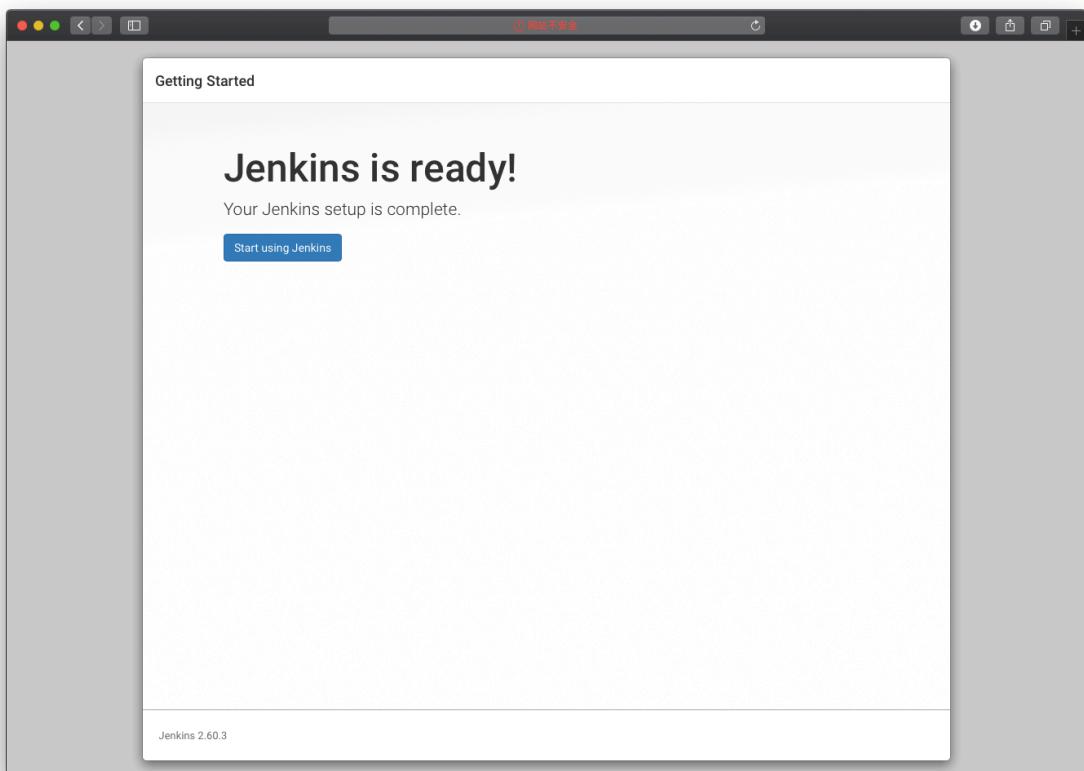
2. 安装推荐插件 (如果你想自定义安装插件, 点击 Select plugins to Install)



3. 创建管理员账号



4. 初始化完成



Gitlab

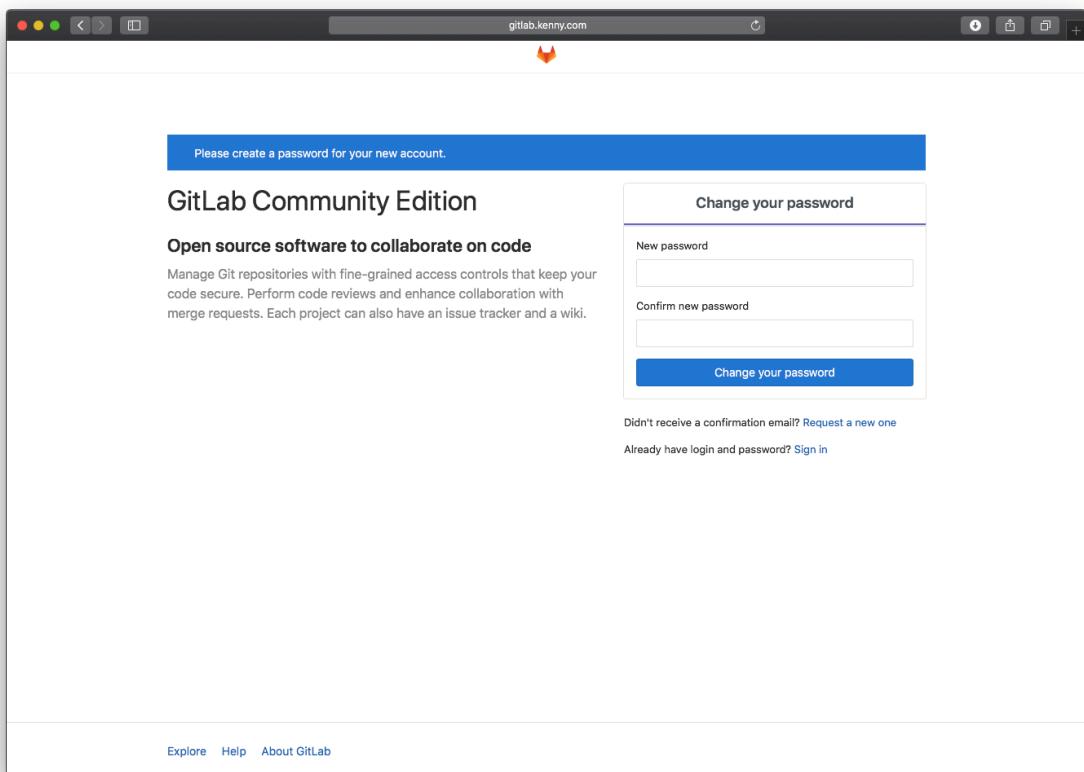
启动服务 (gitlab 集成的服务比较多，因此需要占用较大的内存，官方推荐4GB以上)

```
# $gitlab_home 宿主机目录
# 我的数据卷目录是 ~/.gitlab
export GITLAB_HOME=~/gitlab
docker run -d --restart=always -e 'GITLAB_HOST=gitlab.kenny.com' -p 443:443
-p 80:80 -p 22:22 -v $GITLAB_HOME/conf:/etc/gitlab -v
$GITLAB_HOME:/var/opt/gitlab -v $GITLAB_HOME/log:/var/log/gitlab --name
gitlab gitlab/gitlab-ce

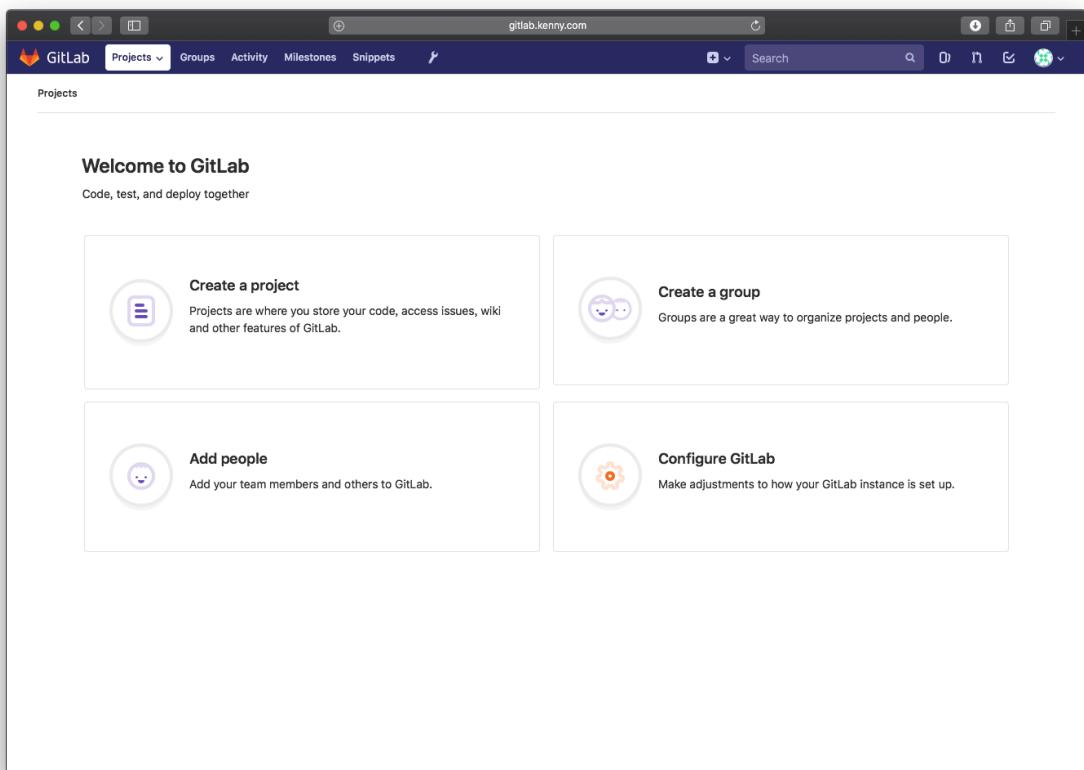
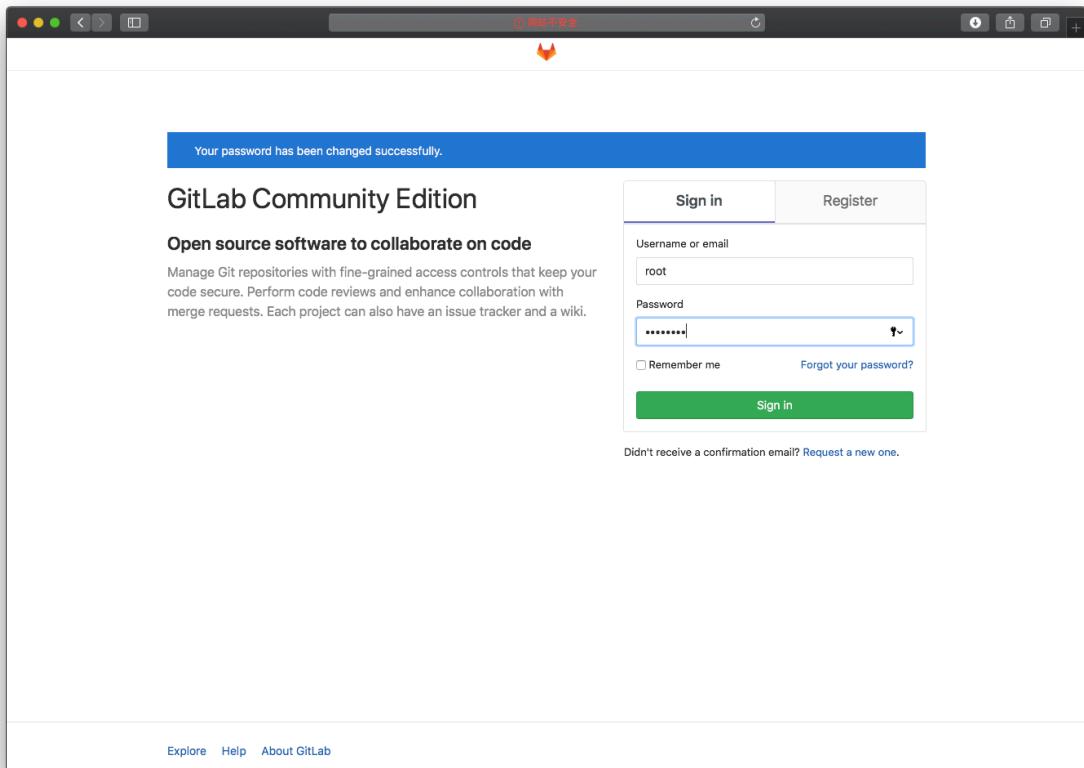
# 查看 gitlab 日志
docker logs -f gitlab
```

初始化

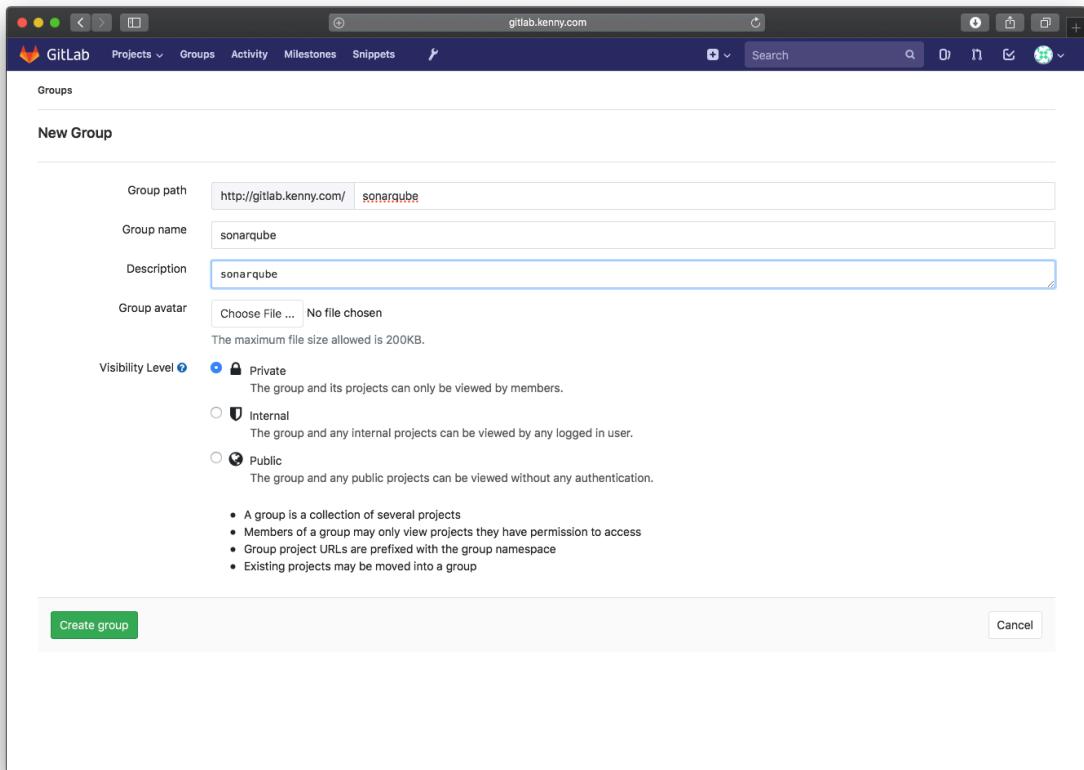
1. 打开浏览器，访问 <http://gitlab.kenny.com>



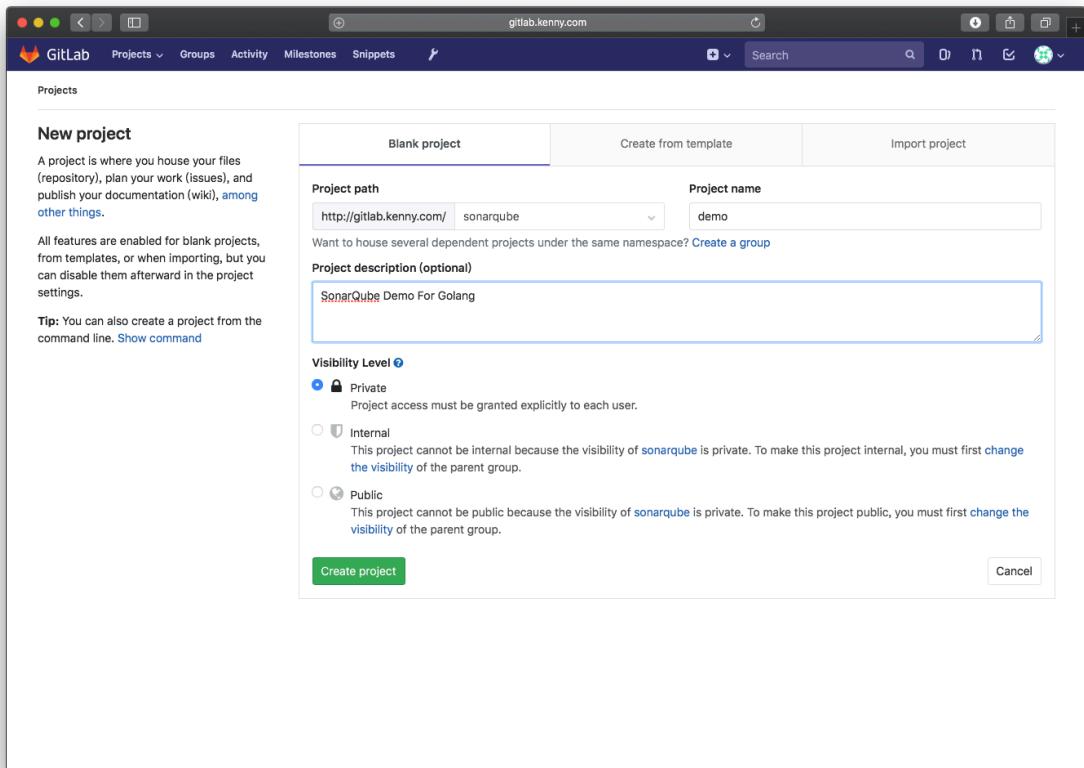
2. 设置新密码后，使用 root 用户登录



3. 新建 sonarqube 项目组



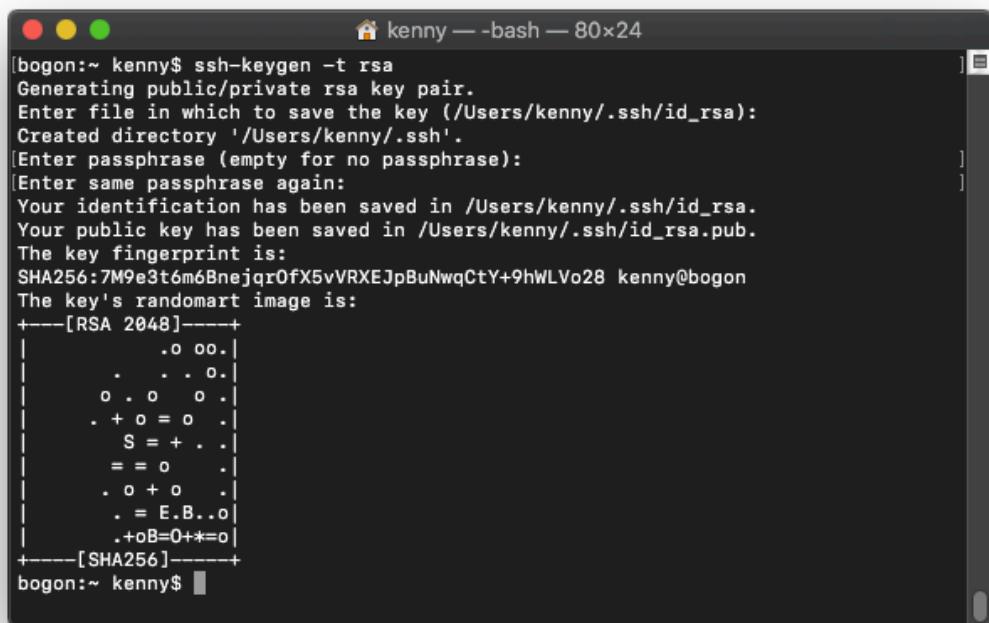
4. 在 sonarqube 项目组下新建 demo 项目



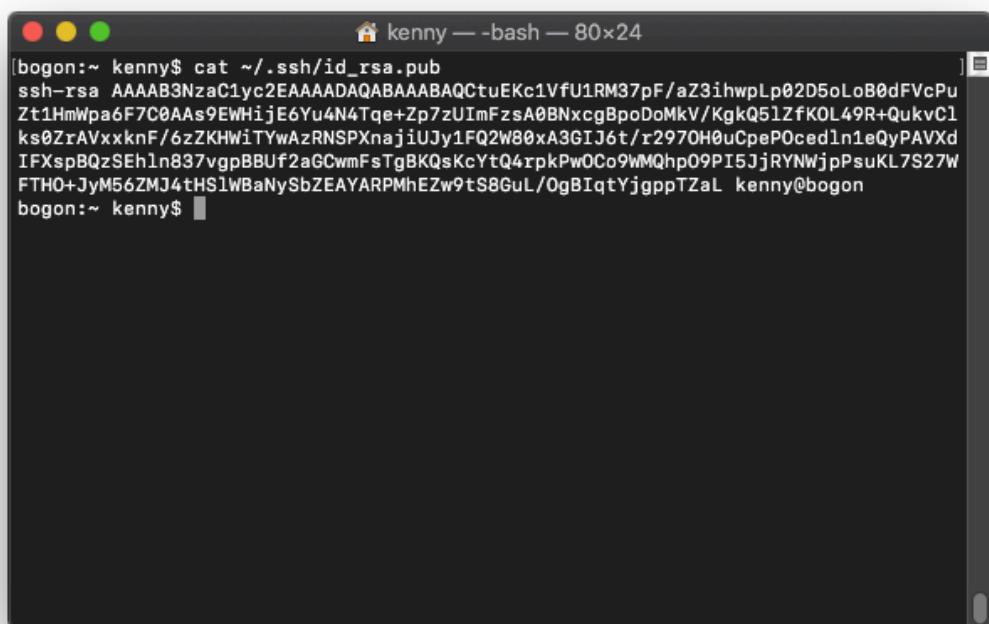
The screenshot shows the 'Details' page for the 'demo' project. The sidebar on the left includes links for Project, Details, Activity, Cycle Analytics, Issues (0), Merge Requests (0), CI / CD, Operations, Wiki, Snippets, and Settings. A prominent orange banner at the top states: 'You won't be able to pull or push project code via SSH until you add an SSH key to your profile' with 'Don't show again | Remind later' buttons. The main content area shows the project's name 'demo' and its description 'SonarQube Demo For Golang'. It features a summary bar with 'Star' (0), 'HTTP' (http://578f2ffd186d.sonarqube.), and 'Global' buttons. Below this, a message says 'The repository for this project is empty' with a note about pushing files. It also mentions 'Auto DevOps' and 'Kubernetes cluster' options. A 'Command line instructions' section provides a 'Git global setup' command: `git config --global user.name "Administrator"`.

5. 添加主机公钥到 Gitlab

```
# 生成 rsa 公钥和密钥  
ssh-keygen -t rsa  
  
# 查看并复制公钥  
cat ~/.ssh/id_rsa.pub
```



```
[bogon:~ kenny$ ssh-keygen -t rsa  
Generating public/private rsa key pair.  
Enter file in which to save the key (/Users/kenny/.ssh/id_rsa):  
Created directory '/Users/kenny/.ssh'.  
[Enter passphrase (empty for no passphrase):]  
[Enter same passphrase again:  
Your identification has been saved in /Users/kenny/.ssh/id_rsa.  
Your public key has been saved in /Users/kenny/.ssh/id_rsa.pub.  
The key fingerprint is:  
SHA256:7M9e3t6m6Bnejqr0fx5vVRXEJpBuNwqCtY+9hWLVo28 kenny@bogon  
The key's randomart image is:  
+---[RSA 2048]----+  
| . o oo . |  
| . . . o . |  
| o . o o . |  
| . + o = o . |  
| S = + . . |  
| = = o . . |  
| . o + o . |  
| . = E.B..o |  
| .+oB=O+*=o|  
+---[SHA256]----+  
bogon:~ kenny$
```



```
[bogon:~ kenny$ cat ~/.ssh/id_rsa.pub  
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCTuEKc1VfU1RM37pF/aZ3ihwpLp02D5oLoB0dFVcPu  
Zt1HmWpa6F7C0AAAs9EWHijsE6Yu4N4Tqe+Zp7zUImFzsA0BNxcgBpoDoMkV/KgkQ51zfK0L49R+QukvCl  
ks0ZrAVxxknF/6zZKHWiTwAzRNSPXnajiUJy1FQ2W80xA3GIJ6t/r297OH0uCpePOcedln1eQyPAVXd  
IFXspBQzSEhln837vgpBBUF2aGCwmFsTgBKQsKcYtQ4rpkPwOC09WMQhp09PI5JjRYNWjpPsuKL7S27W  
FTHO+JyM56ZMJ4tHS1WBaNySbZEAYARPMhEZw9tS8GuL/OgBiqtYjgppTzaL kenny@bogon  
bogon:~ kenny$
```

访问 <http://gitlab.kenny.com/profile/keys>，将公钥添加至 Gitlab

User Settings > SSH Keys

SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

Add an SSH key

Before you can add an SSH key you need to [generate one](#) or use an [existing key](#).

Key

```
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCuEKc1VfU1RM37pF/aZ3ihwpLp02D5oLoB0dFvcP uZt1HmWpa6F7COAs9EWijE6Yu4N4Tqe+Zp7zUlmFzsA0BNxcgBpooDMKV/KgkQ5iZfKOL4 9R+QukvClks0ZrAVxxknF6zZKH1WTYwazRNSPXnajiUJy1FQ2WB0xA3GJ36t/r2970HOUcPeP OcedinTeQyPAVXklxFxspBQzEhIn837vgpBBUf2aGcwmfStgbKQskCYtQ4rpkPwOCo9WMh0Pj5JjRYNWjpPsuKL7S27WFTHo+JyM56ZMj4tSiIWbaNySzZEAYARPMhEZw9tSBGul/Og BlqTYjgpptZaL.kenny@bogon
```

Title

Add key

Your SSH keys (0)

There are no SSH keys with access to your account.

User Settings > SSH Keys > kenny@bogon

SSH Key

Title: kenny@bogon
Created on: Jul 16, 2018 1:38pm
Last used on: N/A

Fingerprint: **7b:75:0b:6b:e6:74:ac:c7:4b:cd:7e:a2:a2:d2:55:5e**

```
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQCuEKc1VfU1RM37pF/aZ3ihwpLp02D5oLoB0dFvcP PuZt1HmWpa6F7COAs9EWijE6Yu4N4Tqe+Zp7zUlmFzsA0BNxcgBpooDMKV/KgkQ5iZfKOL4 9R+QukvClks0ZrAVxxknF6zZKH1WTYwazRNSPXnajiUJy1FQ2WB0xA3GJ36t/r2970HOUcPeP OcedinTeQyPAVXklxFxspBQzEhIn837vgpBBUf2aGcwmfStgbKQskCYtQ4rpkPwOCo9WMh0Pj5JjRYNWjpPs uKL7S27WFTHo+JyM56ZMj4tSiIWbaNySzZEAYARPMhEZw9tSBGul/OgBlqTYjgpptZaL.kenny@bogon
```

Remove

6. 将 sonarqube/demo 项目拉至主机的 \$GOPATH 下

```
# 在 $GOPATH 下创建 gitlab.kenny.com 文件夹
mkdir -p $GOPATH/src/gitlab.kenny.com && cd
$GOPATH/src/gitlab.kenny.com

# clone code
git clone git@gitlab.kenny.com:sonarqube/demo.git
```

SonarQube

启动服务

```
# 由于目前 sonarqube 官方的 Docker images/sonarqube_for_golang 只有 7.1 版本，不
满足 SonarGO 所需 7.2+ 版本，所以我参考7.1 的 Dockerfile 制作了一个 sonarqube
7.2.1 的镜像
# $sonarqube_home 宿主机目录
# 我的数据卷目录是 ~/.sonarqube
export SONARQUBE_HOME=~/sonarqube
# 正式环境中应启用外部数据库服务来存储必要数据，在启动容器时设置如下JDBC相关参数：
# -e SONARQUBE_JDBC_USERNAME=sonar
# -e SONARQUBE_JDBC_PASSWORD=sonar
# -e SONARQUBE_JDBC_URL=jdbc:postgresql://localhost/sonar
docker run -d --restart=always -p 9000:9000 -v
$SONARQUBE_HOME:/opt/sonarqube/data --name sonarqube
kennyallen/sonarqube:7.2.1

# 查看 sonarqube 日志
docker logs -f sonarqube
```

初始化

1. 打开浏览器，访问 <http://sonarqube.kenny.com:9000>

The screenshot shows the SonarQube homepage with the URL `sonarqube.kenny.com`. At the top, there's a navigation bar with links for Projects, Issues, Rules, Quality Profiles, and Quality Gates. A search bar and a 'Log In' button are also at the top. Below the header, the main content area is titled 'Continuous Code Quality'. It features a 'Log in' button and a 'Read documentation' button. To the right, there's a summary section with a large blue '0' icon and the text 'Projects Analyzed'. Below this, there are three categories: 'Bugs' (0), 'Vulnerabilities' (0), and 'Code Smells' (0). Further down, there's a 'Multi-Language' section listing supported languages like Java, C/C++, C#, COBOL, ABAP, HTML, RPG, JavaScript, TypeScript, Objective C, XML, VB.NET, PL/SQL, T-SQL, Flex, Python, Groovy, PHP, Swift, Visual Basic, and PL/I. A 'Quality Model' section follows, with three items: 'Bugs' (track code that is demonstrably wrong or highly likely to yield unexpected behavior), 'Vulnerabilities' (raised on code that is potentially vulnerable to exploitation by hackers), and 'Code Smells' (will confuse maintainers or give them pause; measured primarily in terms of the time they will take to fix). Finally, there are two sections: 'Write Clean Code' and 'Fix The Leak', each with a 'Read More' link.

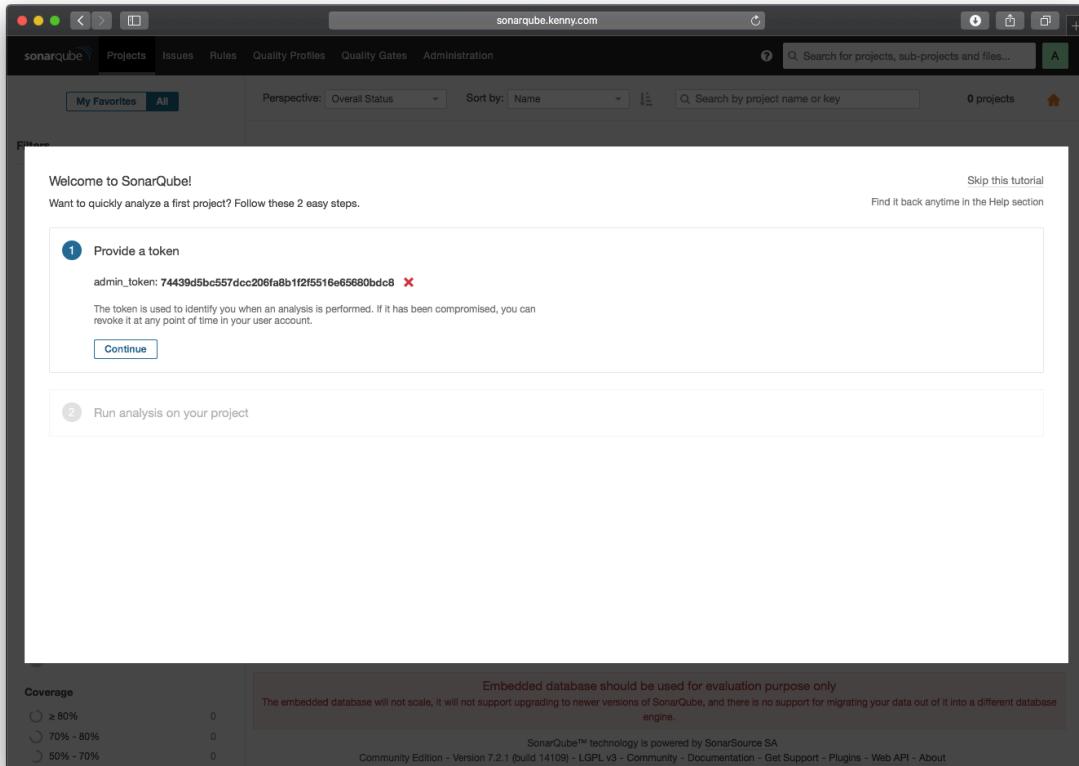
2. 使用管理员账号登录

- 账号 admin
- 密码 admin

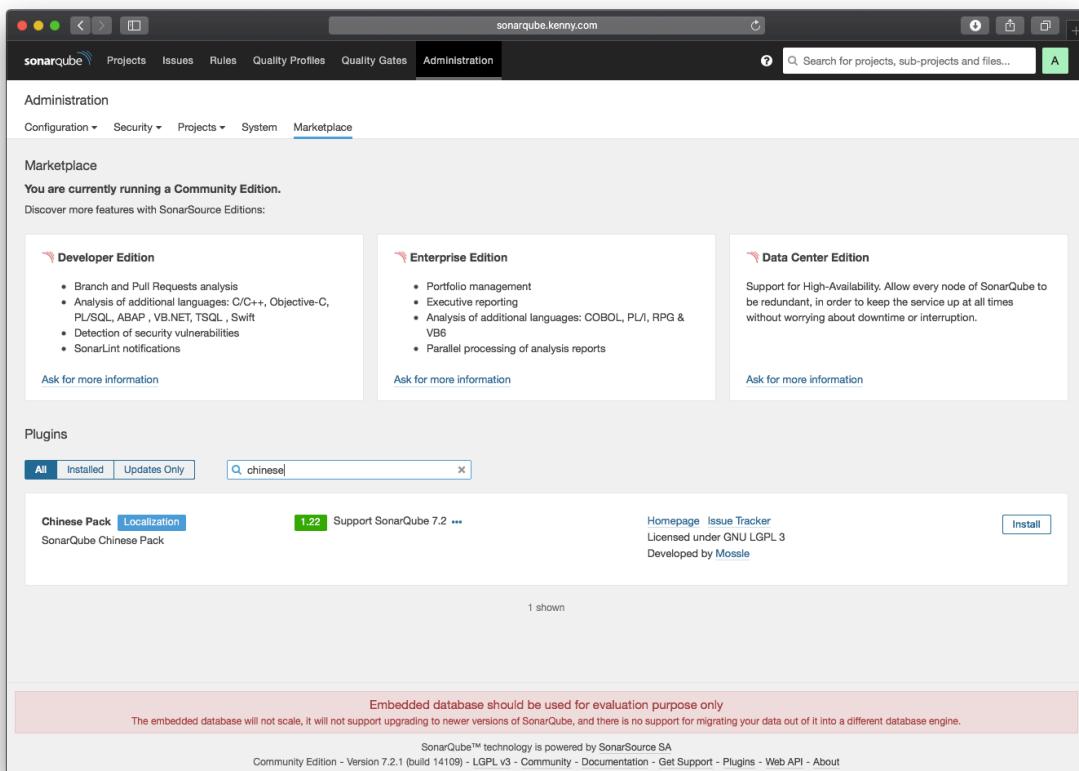
The screenshot shows the 'Log In to SonarQube' page. It has a 'Log In' button and a 'Cancel' button. Below the input fields, there's a note: 'Embedded database should be used for evaluation purpose only. The embedded database will not scale, it will not support upgrading to newer versions of SonarQube, and there is no support for migrating your data out of it into a different database engine.' At the bottom, there's footer text: 'SonarQube™ technology is powered by SonarSource SA LGPL v3 - Community - Documentation - Get Support - Plugins'.

3. 生成 token (作为远程连接 SonarQube 的标识, 只生成一次, 记得备份哦)

admin_token: **74439d5bc557dcc206fa8b1f2f5516e65680bdc8**



4. 安装插件 (进入 Administration -> Marketplace)



安装完成后点击重启 SonarQube 服务就OK了

集成

- 将 Jenkins、Gitlab 和 SonarQube 有机整合

Jenkins 安装插件

1. 点击进入 系统管理 -> 插件管理 -> 可选插件

安装	名称	版本
<input type="checkbox"/>	CCM	3.2
<input type="checkbox"/>	change-assembly-version-plugin	1.10
<input type="checkbox"/>	ExCop.Runner	1.1
<input type="checkbox"/>	MSBuild	1.29
<input type="checkbox"/>	MSTest	0.23
<input type="checkbox"/>	MSTestRunner	1.3.0
<input type="checkbox"/>	NAnt	1.4.3
<input type="checkbox"/>	PowerShell	1.3
<input type="checkbox"/>	Violation Comments to Bitbucket Server	1.75
<input type="checkbox"/>	Finds violations reported by code analyzers and comments Bitbucket Server (or Stash) pull requests (or commits) with them.	0.7.11
<input type="checkbox"/>	Violations	0.7.11
<input type="checkbox"/>	Visual Studio Code Metrics	1.7
<input type="checkbox"/>	VSTest Runner	1.0.7
<input type="checkbox"/>	Start Windocks Containers	1.4
<input type="checkbox"/>	Allows for the creation and start of Windocks Containers	
<input type="checkbox"/>	Amazon EC2 Container Service	1.16
<input type="checkbox"/>	Jenkins plugin to run dynamic slaves in a Amazon ECS/Docker environment	
<input type="checkbox"/>	Azure VM Agents	0.7.1
Provision agents on Azure cloud		

2. 过滤选中 Gitlab、SonarQube Scanner，点击下载待重启后安装

The screenshot shows the Jenkins plugin manager interface. The search bar at the top right contains the text "gitlab". Below the search bar, there are tabs for "可更新" (Updatable), "可选插件" (Optional Plugins), "已安装" (Installed), and "高级" (Advanced). The "可选插件" tab is selected. A table lists several Jenkins plugins related to GitLab:

名称	版本
GitLab Authentication	1.0.9
<input type="checkbox"/> A Jenkins authentication plugin that delegates to gitlab. We also implement an Authorization Strategy that users the acquired OAuth token to interact with the Gitlab API to determine a users level of access to Jenkins.	
<input checked="" type="checkbox"/> GitLab	1.5.8
<input type="checkbox"/> GitLab Logo	1.0.3
<input type="checkbox"/> Display GitLab Repository Icon on dashboard	
<input type="checkbox"/> GitLab Merge Request Builder	2.0.0
<input type="checkbox"/> A plugin to build merge requests in Gitlab	
<input type="checkbox"/> Violation Comments to GitLab	2.2
<input type="checkbox"/> Finds violations reported by code analyzers and comments GitLab merge requests with them.	
<input type="checkbox"/> Gitlab Hook	
Enables Gitlab web hooks to be used to trigger SMC polling on Gitlab projects	
<input type="checkbox"/> Warning: This plugin version may not be safe to use. Please review the following security notices:	1.4.2
• Gitlab API token stored and displayed in plain text	

At the bottom of the page, there are three buttons: "直接安装" (Direct Install), "下载待重启后安装" (Download to install after restart), and "立即获取" (Get Now). A status message "Update information obtained: 34 分 ago" is displayed between the download and get now buttons.

The screenshot shows the Jenkins plugin manager interface. The search bar at the top right contains the text "sonar". Below the search bar, there are tabs for "可更新" (Updatable), "可选插件" (Optional Plugins), "已安装" (Installed), and "高级" (Advanced). The "可选插件" tab is selected. A table lists several Jenkins plugins related to SonarQube:

名称	版本
<input type="checkbox"/> CodeSonar	2.0.7
<input checked="" type="checkbox"/> SonarQube Scanner	2.6.1
<input type="checkbox"/> Sonargraph Integration	2.1.2
<input type="checkbox"/> Sonargraph	1.6.4
<input type="checkbox"/> Mashup Portlets	
<input type="checkbox"/> Additional Dashboard Portlets: Generic JS Portlet (lets you pull in arbitrary content via JS), Recent Changes Portlet (shows the SCM changes for a given job), SonarQube Portlets (show SonarQube statistics directly in Jenkins) and Test Results Portlet (shows the test results for a given job).	1.0.9
<input type="checkbox"/> Sonar Gerrit	2.3
<input type="checkbox"/> Quality Gates	
<input type="checkbox"/> Fails the build whenever the Quality Gates criteria in the Sonar analysis aren't met (the project Quality Gates status is different than "Passed")	2.5
<input type="checkbox"/> Sonar Quality Gates	
<input type="checkbox"/> Fails the build whenever the Quality Gates criteria in the Sonar 5.6+ analysis aren't met (the project Quality Gates status is different than "Passed")	1.1.2

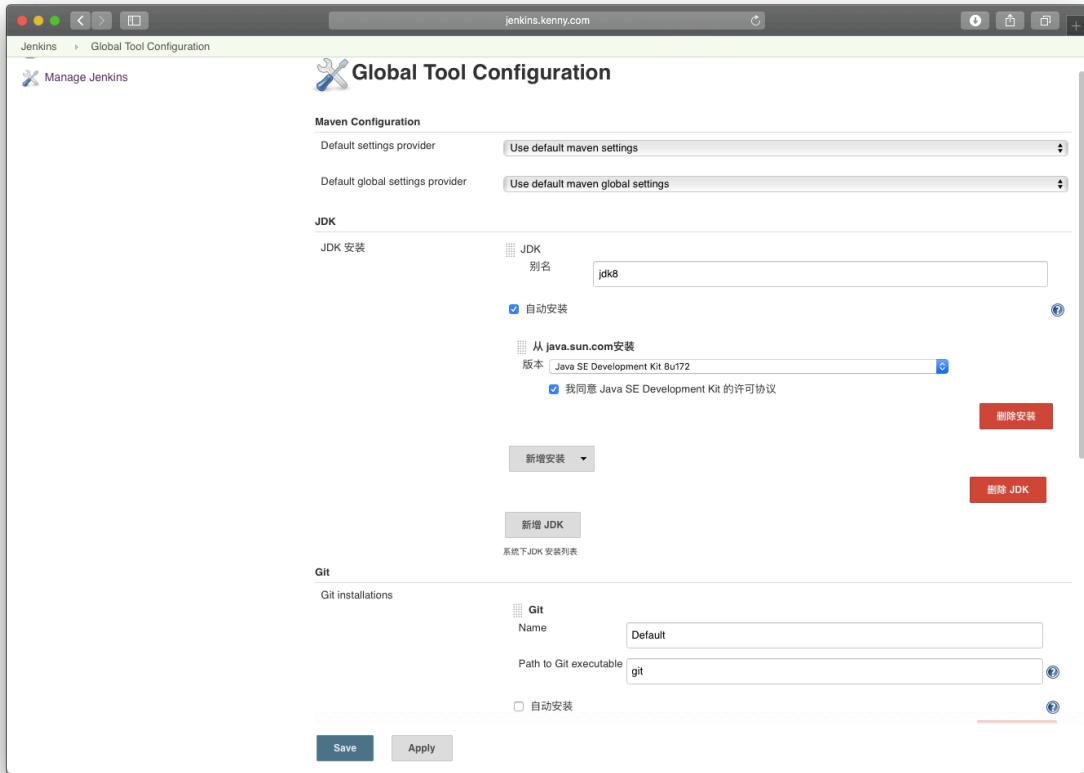
At the bottom of the page, there are three buttons: "直接安装" (Direct Install), "下载待重启后安装" (Download to install after restart), and "立即获取" (Get Now). A status message "Update information obtained: 34 分 ago" is displayed between the download and get now buttons.

Jenkins 配置

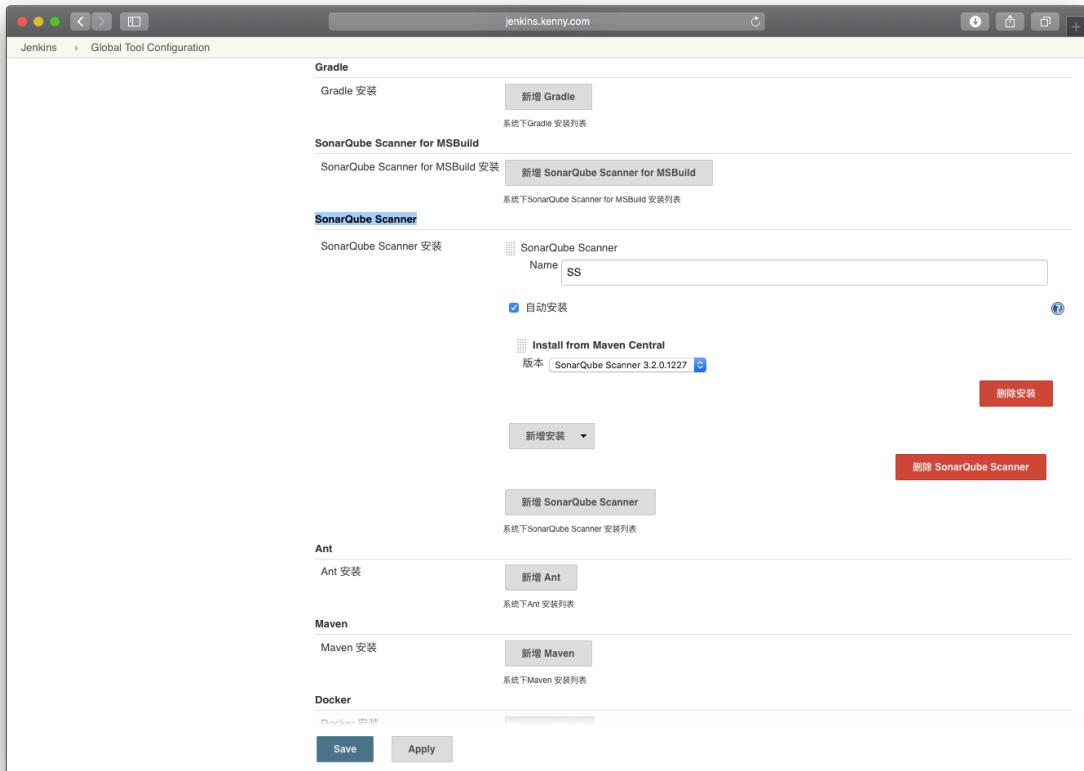
1. 安装 SonarQube & JDK

进入 系统管理 -> Global Tool Configuration

JDK 安装



SonarQube Scanner 安装



2. SonarQube Server

进入 系统管理 -> 系统设置

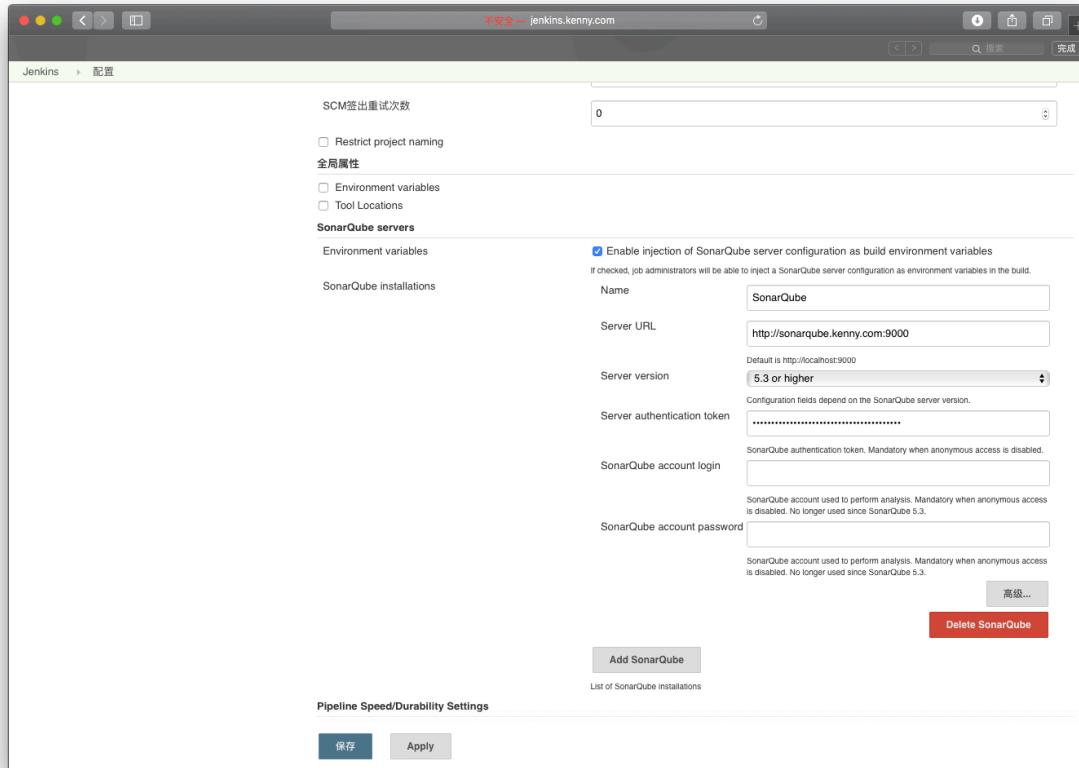
找到 SonarQube servers

Name 随便填写

Server URL: <http://sonarqube.kenny.com:9000>

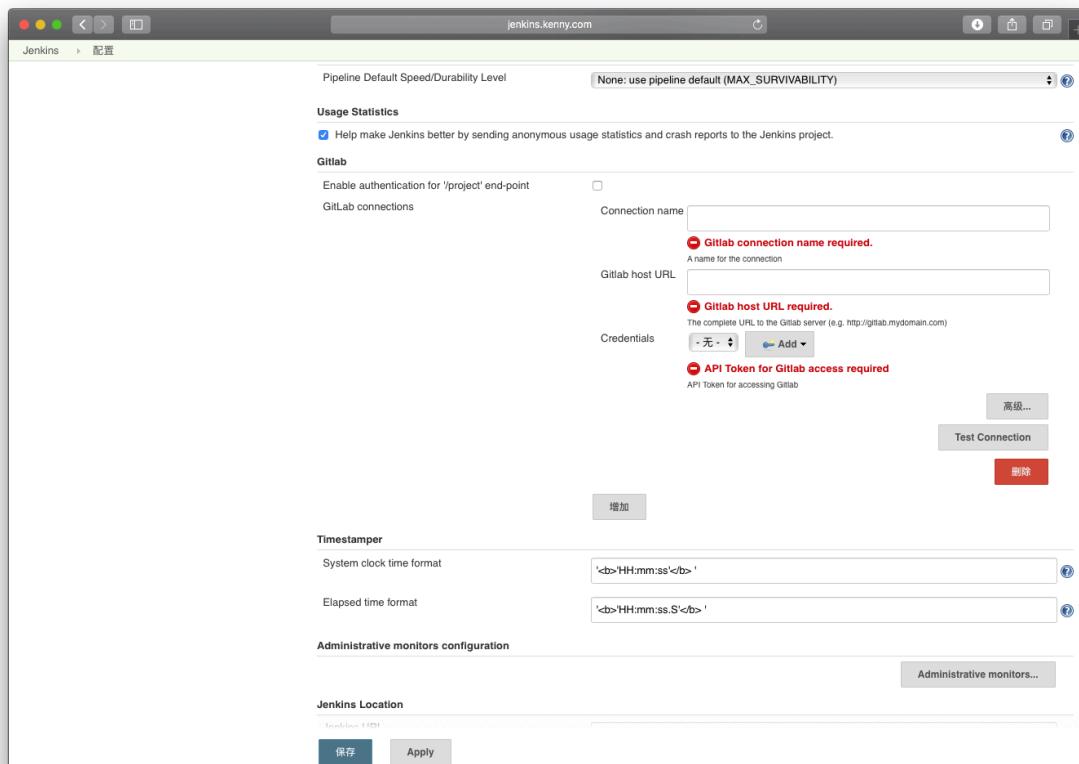
Server version: 5.3 or higher

Server authentication token: 填 SonarQube 初始化时生成的 token



3. 取消 Gitlab 授权

取消选中 Enable authentication for '/project' end-point, 保存



4. 在 jenkins 容器中安装 golang 环境及工具

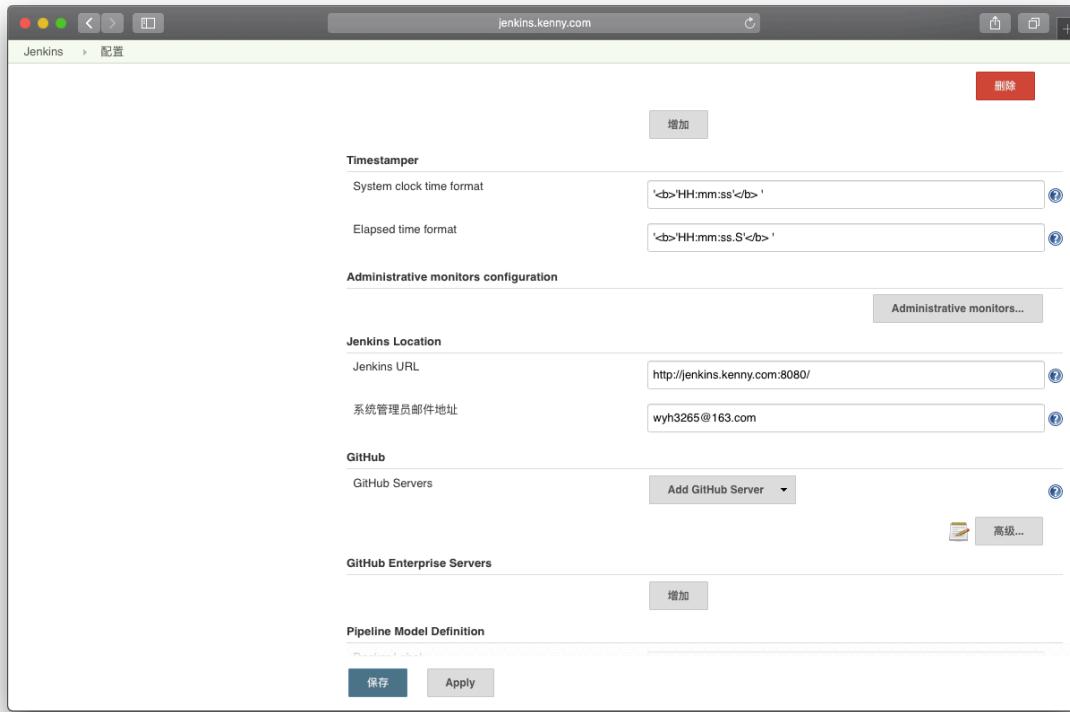
```
# 在 Jenkins 容器中执行命令
docker exec -it jenkins /bin/bash
# 临时设置环境变量
export GOROOT=$JENKINS_HOME/go
export GOPATH=$JENKINS_HOME/workspace/go
export PATH=$PATH:$GOROOT/bin:$GOPATH/bin
export http_proxy=http://192.168.1.100:1087;export
https_proxy=http://192.168.1.100:1087;
# 进入 jenkins 主目录
cd $JENKINS_HOME
# 下载 golang
wget https://dl.google.com/go/go1.10.3.linux-amd64.tar.gz
# 解压 golang 包
tar -xvf go1.10.3.linux-amd64.tar.gz
# 删除 golang 包
rm go1.10.3.linux-amd64.tar.gz
# 安装必要工具
# vgo
go get -u -v golang.org/x/vgo
# gometalinter
go get -u -v github.com/alecthomas/gometalinter
gometalinter --install
```

5. 配置邮件通知

进入 系统管理 -> 系统设置

Jenkins Location

系统管理员邮件地址修改为你的邮箱地址, 如 wyh3265@163.com



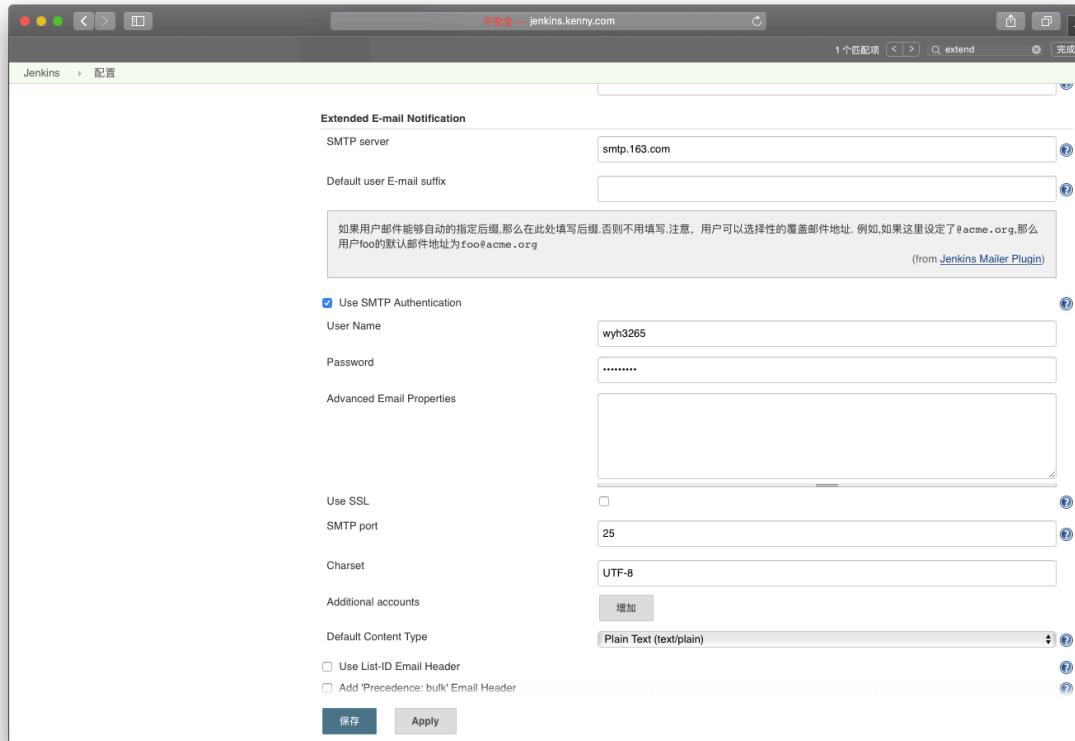
Extend E-mail Notification

SMTP Server 填写对应的SMTP服务地址，如 smtp.163.com

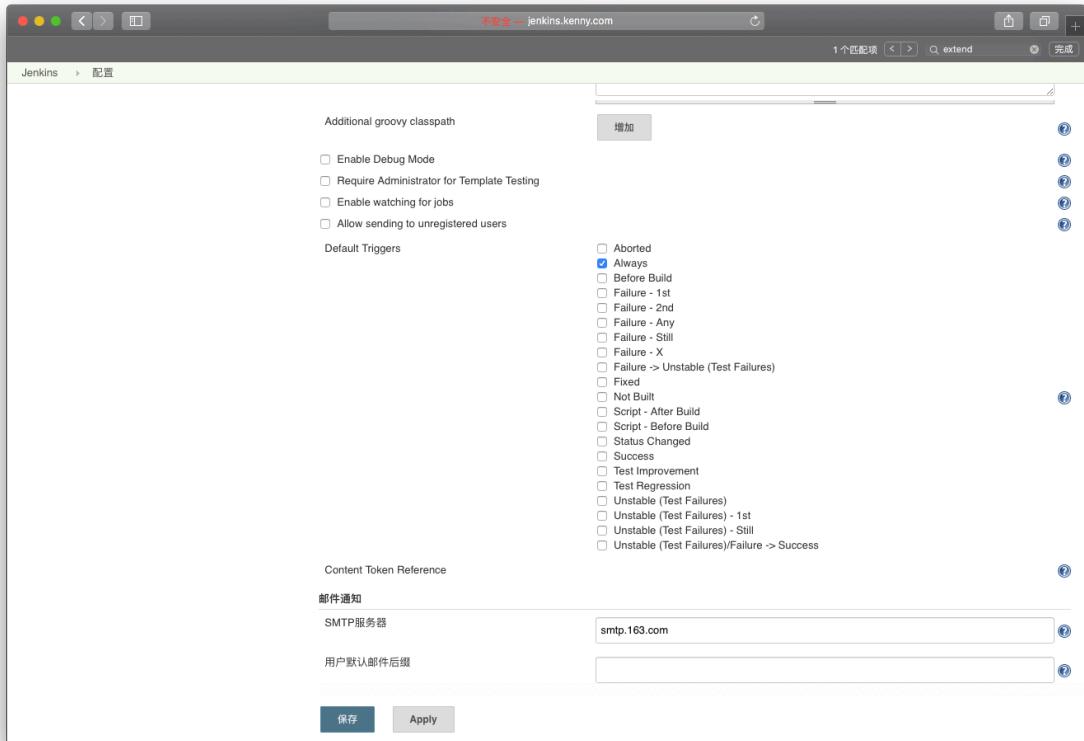
勾选使用SMTP认证

用户名 注意不需要加 @xxx.xxx

密码 填写自己的邮箱密码或授权码



Default Triggers 选中 Always



新建 Jenkins 构建任务

1. 新构建一个自由风格的软件项目



2. 使用自定义的工作空间

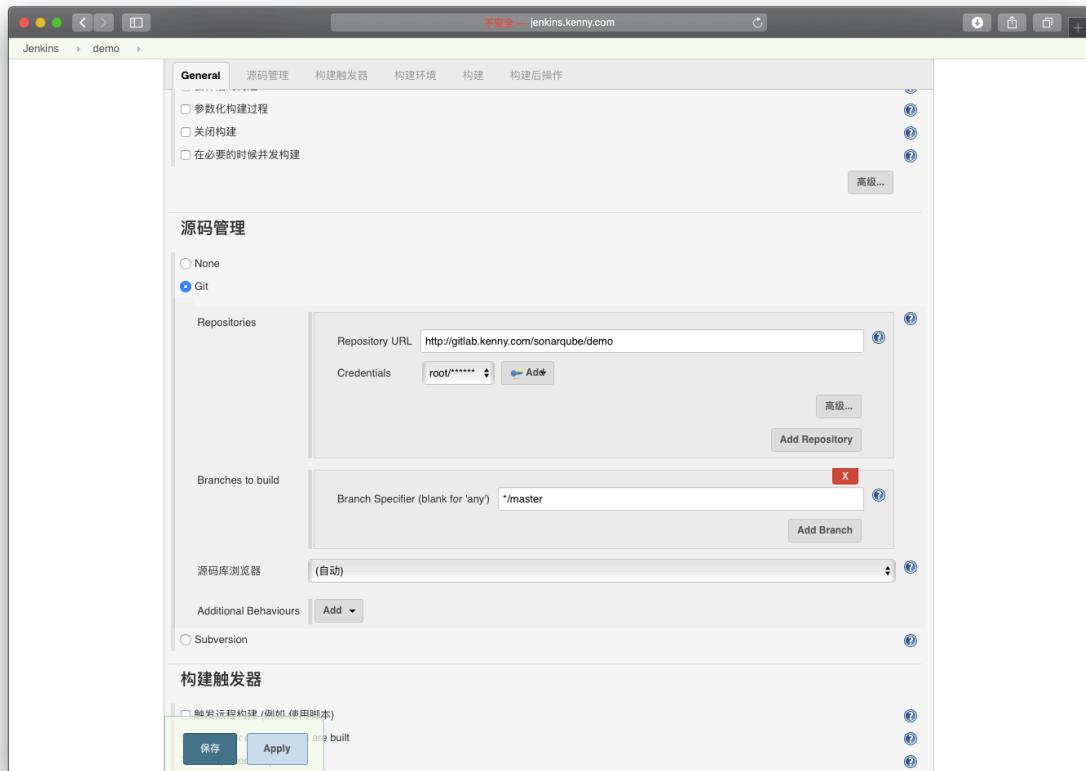
目录: \${JENKINS_HOME}/workspace/go/src/gitlab.kenny.com/demo



3. 源码管理

Repository URL: <http://gitlab.kenny.com/sonarqube/demo.git>

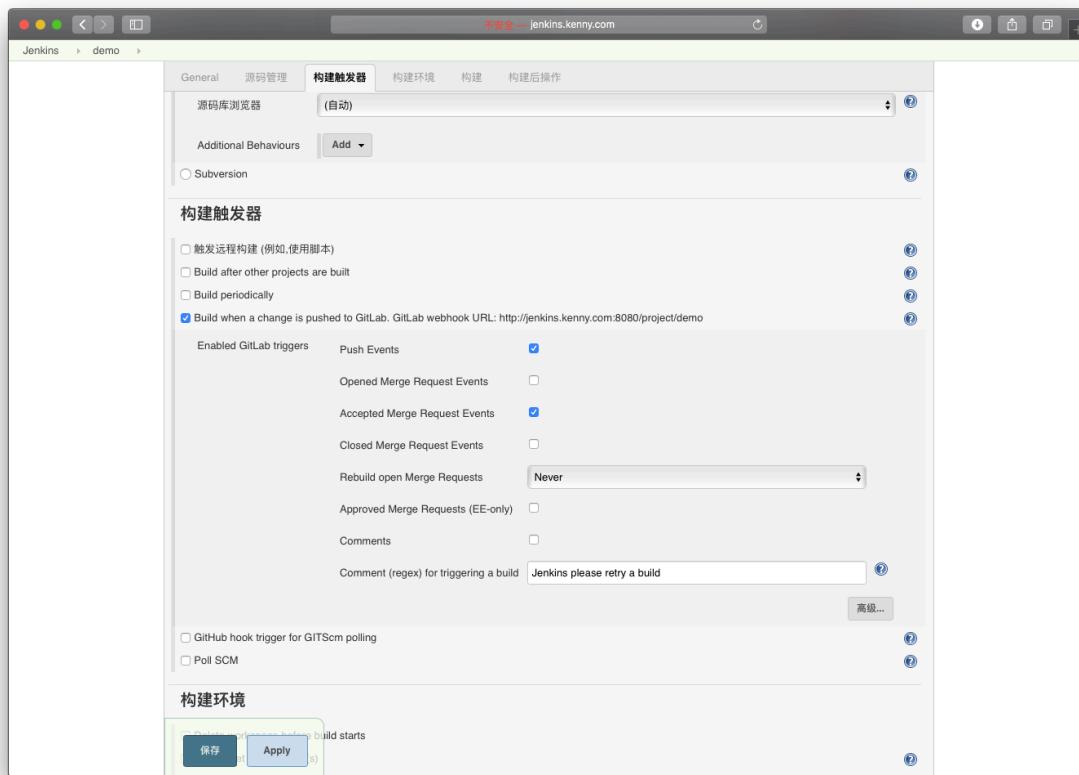
Credentials: Gitlab 用户名密码或 SSH 登录等方式都可以



4. 构建触发器, 选中

Build when a change is pushed to GitLab. GitLab webhook URL: <http://jenkins.kenny.com:8080/project/demo>

Enabled GitLab triggers 选中 Push Events 和 Accepted Merge Request Events , 表示当 Gitlab 有 push 或 merge 操作发生时触发构建。



5. 新建 webhook

在浏览器中打开 http://gitlab.kenny.com/admin/application_settings (请使用 root 登录), 找到 Outbound requests , 点击 Expand 后, 选中 Allow requests to the local network from hooks and services 并保存更改。(允许本地网络的 githook)

The screenshot shows the GitLab Admin Area settings page. The left sidebar lists various settings categories: Overview, Monitoring, Messages, System Hooks, Applications, Abuse Reports, Deploy Keys, Service Templates, Labels, Appearance, and Settings. The 'Settings' category is currently selected. The main content area contains several expandable sections: 'Various email settings.', 'Gitaly' (Configure Gitaly timeouts), 'Web terminal' (Set max session time for web terminal), 'Real-time features' (Change this value to influence how frequently the GitLab UI polls for updates), 'Performance optimization' (Various settings that affect GitLab performance), 'User and IP Rate Limits' (Configure limits for web and API requests), 'Outbound requests' (Allow requests to the local network from hooks and services, with a checked checkbox for 'Allow requests to the local network from hooks and services'), 'Repository mirror settings' (Configure push mirrors), and 'Collapse' (button). A 'Save changes' button is located at the bottom of the main content area.

进入 <http://gitlab.kenny.com/sonarqube/demo/settings/integrations>

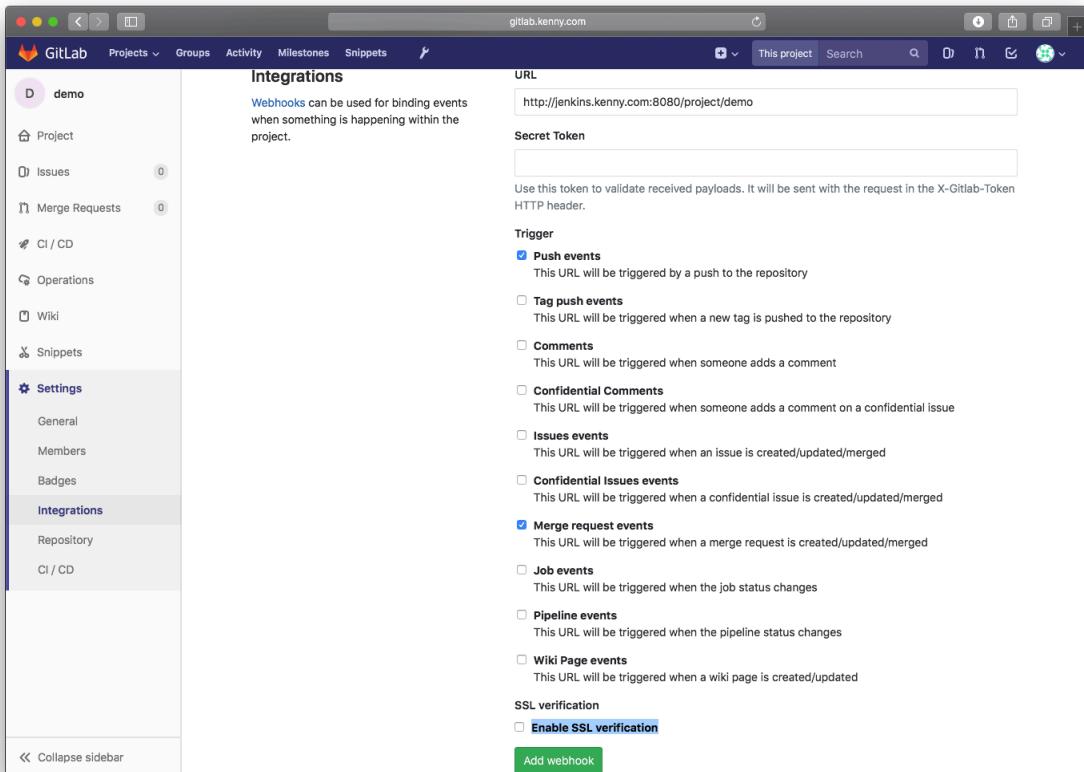
URL: <http://jenkins.kenny.com:8080/project/demo>

SecretToken: 不填

选中 **Push events**、**Merge request events**

取消选中 **Enable SSL verification**

点击 Add web hook



6. 增加构建步骤，选中 Execute Shell

```
#!/bin/bash
# 环境变量
export GOROOT=$JENKINS_HOME/go
export GOPATH=$JENKINS_HOME/workspace/go
export PATH=$PATH:$GOROOT/bin:$GOPATH/bin
export http_proxy=http://192.168.1.100:1087;export
https_proxy=http://192.168.1.100:1087;

# 安装依赖
vgo mod -vendor

# coverage
go test ./... -coverprofile=coverage.out

# test
go test ./... -json > report.json

# vet
go vet ./... 2> govet-report.out

# golint
golint ./... > golint-report.out

# gometalinter
# 执行 gometalinter 会失败,因此加了 || true
```

```
gometalinter ./... > gometalinter-report.out || true
```

7. 增加构建步骤，选中 Execute SonarQube Scanner

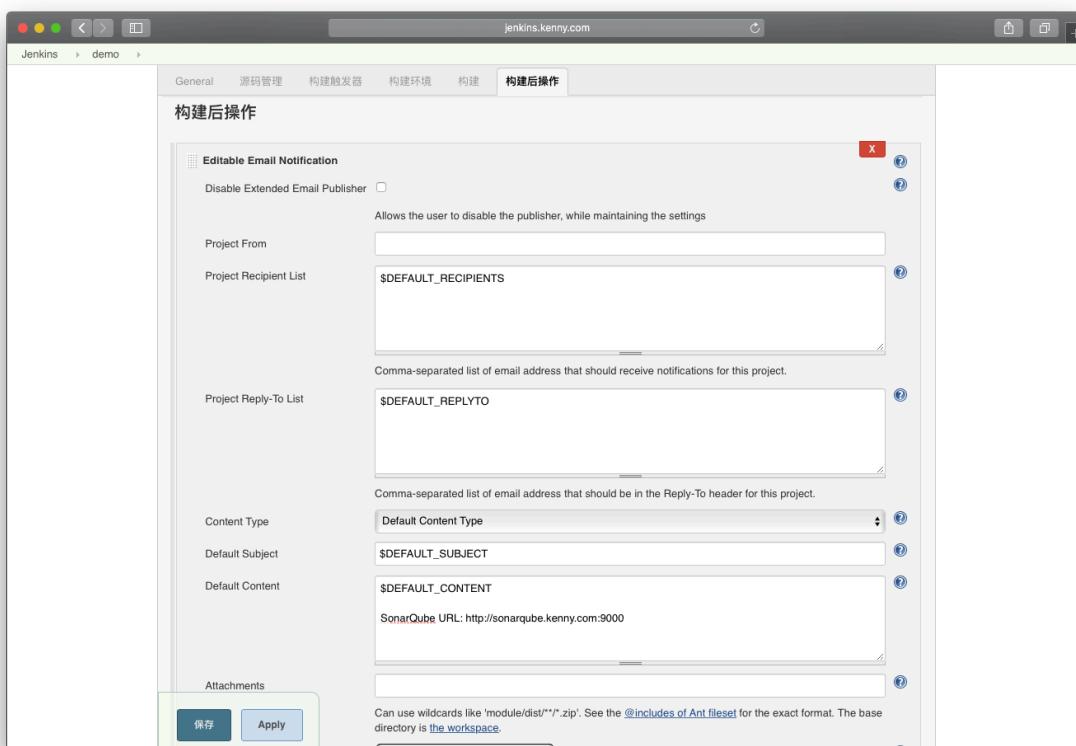
Analysis properties

```
sonar.projectKey=gitlab.kenny.com
sonar.projectName=demo
sonar.sources=.
sonar.exclusions=**/*_test.go,**/vendor/**
sonar.tests=.
sonar.test.inclusions=**/*_test.go
sonar.test.exclusions=**/vendor/**
sonar.go.coverage.reportPaths=coverage.out
sonar.go.tests.reportPaths=report.json
sonar.go.govet.reportPaths=govet-report.out
sonar.go.golint.reportPaths=golint-report.out
sonar.go.gometalinter.reportPaths=gometalinter-report.out
```

8. 增加构建后操作，选中 Editable Email Notification

Project Recipient List 填写接收邮件的Email地址，或使用默认配置

Default Content 加上 SonarQube URL: <http://sonarqube.kenny.com:9000>



测试

```
# clone demo 代码
cd $GOPATH/src/gitlab.kenny.com && git clone
git@github.com:yuhao5/sonarqube-golang.git && rm -rf demo && mv sonarqube-
golang demo && cd demo
# push 代码, 触发 Jenkins 任务进行自动构建
git remote add gitlab git@gitlab.kenny.com:sonarqube/demo.git
git push -u gitlab master

# 若 gitlab 仓库地址不是 git@gitlab.kenny.com:sonarqube/demo.git , 请根据以下步
骤修改:
docker exec -it gitlab /bin/bash
vim /etc/gitlab/gitlab.rb
# 找到 external_url, 修改为 external_url 'http://gitlab.kenny.com'
# 然后执行
gitlab-ctl reconfigure
```

demo - Build # 10 - Successful! 收件箱



wyh3265@163.com

发送至 我



英语 ▾

中文 ▾

翻译邮件

demo - Build # 10 - Successful:

Check console output at <http://jenkins.kenny.com:8080/job/demo/10/> to view the results.

SonarQube URL: <http://sonarqube.kenny.com:9000>



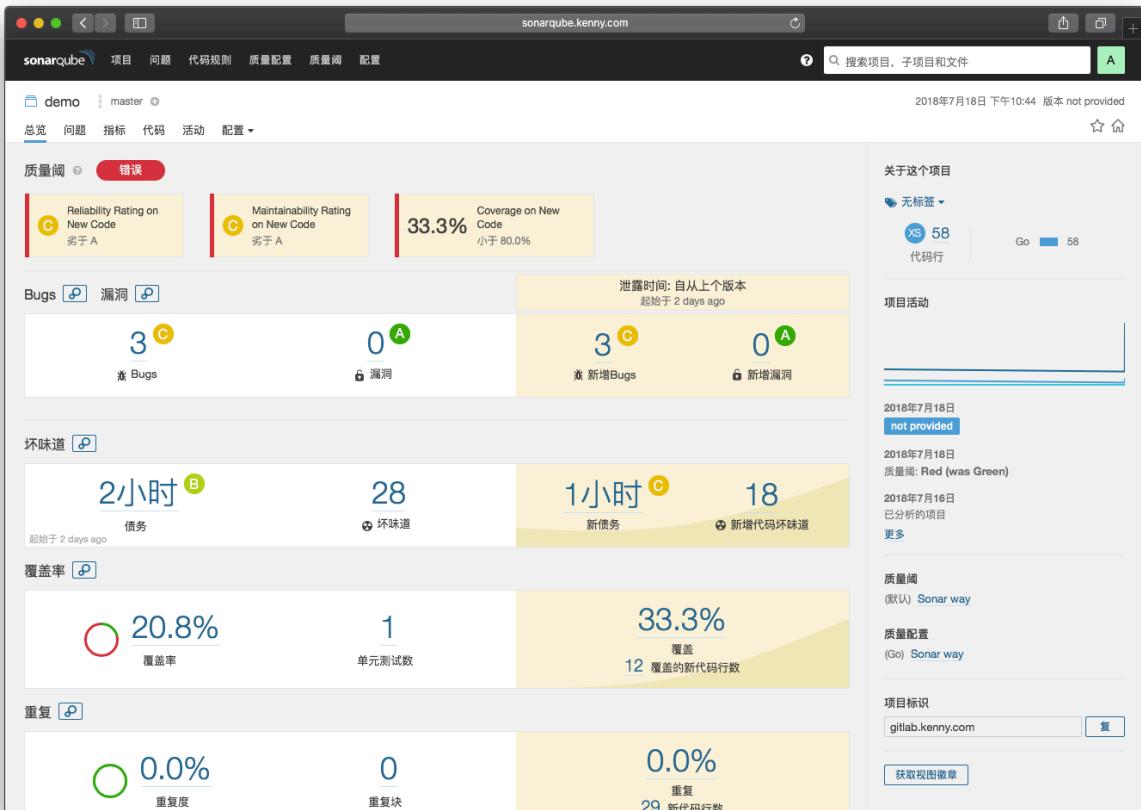
点击此处即可回复或转发

Jenkins > demo > #4

```

INFO: Excluded tests:
INFO: **/vendor/**
INFO: 10 files indexed
INFO: 22 files ignored because of inclusion/exclusion patterns
INFO: Quality profile for go: Sonar way
INFO: Sensor SonarGo [go]
INFO: Load coverage report from '/var/jenkins_home/workspace/go/src/gitlab.kenny.com/demo/coverage.out'
INFO: Sensor SonarGo [go] (done) | time=450ms
INFO: Sensor Go Unit Test Report [go]
INFO: Sensor Go Unit Test Report [go] (done) | time=15ms
INFO: Sensor Import of go vet issues [go]
INFO: GoVetReportSensor: Importing '/var/jenkins_home/workspace/go/src/gitlab.kenny.com/demo/govet-report.out'
INFO: Sensor Import of go vet issues [go] (done) | time=4ms
INFO: Sensor Import of Golint issues [go]
INFO: GoLintReportSensor: Importing '/var/jenkins_home/workspace/go/src/gitlab.kenny.com/demo/golint-report.out'
INFO: Sensor Import of Golint issues [go] (done) | time=26ms
INFO: Sensor Import of GoMetaLinter issues [go]
INFO: GoMetaLinterReportSensor: Importing '/var/jenkins_home/workspace/go/src/gitlab.kenny.com/demo/gometalinter-report.out'
INFO: Sensor Import of GoMetaLinter issues [go] (done) | time=1ms
INFO: Sensor SonarJavaXmlFileSensor [java]
INFO: Sensor SonarJavaXmlFileSensor [java] (done) | time=1ms
INFO: Sensor Zero Coverage Sensor
INFO: Sensor Zero Coverage Sensor (done) | time=24ms
INFO: Sensor CPD Block Indexer
INFO: Sensor CPD Block Indexer (done) | time=1ms
INFO: SCM provider for this project is: git
INFO: 3 files to be analyzed
INFO: 3/3 files analyzed
INFO: Calculating CPD for 2 files
INFO: CPD calculation finished
INFO: Analysis report generated in 130ms, dir size=31 KB
INFO: Analysis reports compressed in 517ms, zip size=11 KB
INFO: Analysis report uploaded in 932ms
INFO: ANALYSIS SUCCESSFUL, you can browse http://sonarqube.kenny.com:9000/dashboard?id=gitlab.kenny.com
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://sonarqube.kenny.com:9000/api/ce/task?id=AWSjuRSmfIgLZ\_VJtWDF
INFO: Task total time: 9.268 s
INFO: -----
INFO: EXECUTION SUCCESS
INFO: -----
INFO: Total time: 19.731s
INFO: Final Memory: 14M/200M
INFO: -----
Finished: SUCCESS

```



sonarqube.kenny.com

demo | master

总览 问题 指标 代码 活动 配置

我的问题 所有

批量修改

过滤器 清空所有条件

显示模式 问题 工作

类型 Bug 清空

高 Bug 3

低 漏洞 0

坏味道 28

严重程度

- 次要 0
- 严重 0
- 主要** 3

处理方式 状态 新问题 规则 标签 模块 目录 文件 负责人 作者 语言

math/math.go

Remove this self assignment ... 28 minutes ago L41 🔍 cert

self-assignment of b to b ... govet 33 minutes ago L41 🔍 cert

self-assignment of b to b ... govet 33 minutes ago L41 🔍 cert

显示 3 / 3

内嵌数据库只能用于测试场景
内嵌数据库无法扩展，也无法升级到新版本的SonarQube，并且不能支持将你的数据迁移至其他数据库引擎。

SonarQube™ technology is powered by SonarSource SA
Community Edition - 版本 7.2.1 (build 14109) - LGPL v3 - 社区 - 文档 - 获取支持 - 插件 - Web接口 - 关于

sonarqube.kenny.com

demo | master

总览 问题 指标 代码 活动 配置

8 9 **type Math_struct {**

10 A int

11 B int

12 }

13

14 kenn... **func New(a, b int) *Math {**

15 **exported function New should have comment or be unexported ... golint** 34 minutes ago L14 🔍

16 **坏味道** 🔍 **主要** 🔍 **打开** **未分配** 🔍 **5min 工作** **评论**

17 **return &Math{**

18 A: a,

19 B: b,

20 }

21 kenn... **func (m *Math) Divide() int {**

22 **// FIXME: fix divide by zero panic**

23 kenn... **Take the required action to fix the issue indicated by this "FIXME" comment. ...** 28 minutes ago L22 🔍

24 **坏味道** 🔍 **主要** 🔍 **打开** **未分配** 🔍 **评论**

25 **return m.A / m.B**

26 kenn... **func (m *Math) Multipart() int {**

27 **exported method Math.Multipart should have comment or be unexported ... golint** 34 minutes ago L26 🔍

28 **坏味道** 🔍 **主要** 🔍 **打开** **未分配** 🔍 **5min 工作** **评论**

29 **exported method Math.Multipart should have comment or be unexported ... golint** 34 minutes ago L26 🔍

30 **坏味道** 🔍 **主要** 🔍 **打开** **未分配** 🔍 **5min 工作** **评论**

31 **// TODO: implement multipart**

 var result int

 return result

TODO

- 解决执行 gometalinter 失败问题

2. Golang 质量标准，规则自定义

3. ...