Gitlab安装手册

# 一、安装环境：

Centos 7.x 64位

Gitlab版本：11.1.0

# 二、安装配置和必要依赖

[root@localhost ~]# yum install curl policycoreutils openssh-server openssh-clients

[root@localhost ~]# systemctl enable sshd

[root@localhost ~]# systemctl start sshd

[root@localhost ~]# yum install postfix

[root@localhost ~]# systemctl enable postfix

[root@localhost ~]# systemctl start postfix

*tips:建议防火墙关闭状态可不进行如下配置*

[root@localhost ~]# firewall-cmd --permanent --add-service=http

FirewallD is not running

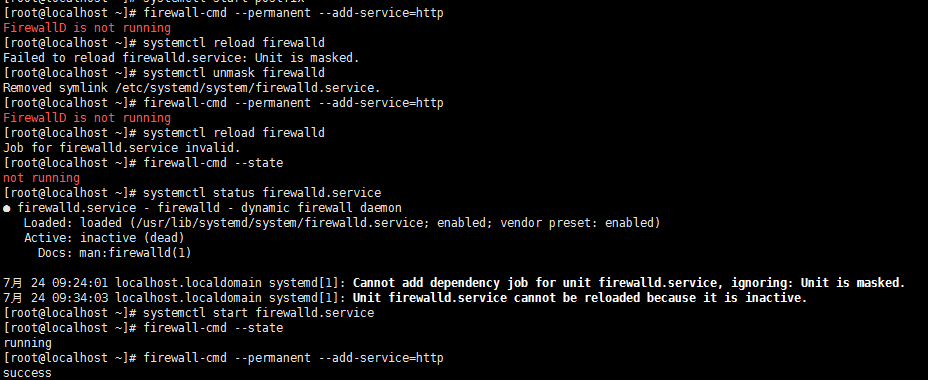
[root@localhost ~]# systemctl start firewalld.service

[root@localhost ~]# firewall-cmd --state

running

[root@localhost ~]# firewall-cmd --permanent --add-service=http

Success



[root@localhost ~]# systemctl reload firewalld

Failed to reload firewalld.service: Unit is masked.

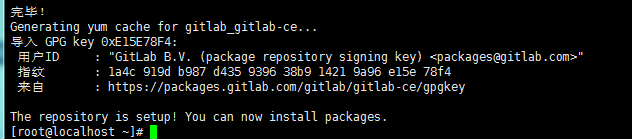
[root@localhost ~]# systemctl unmask firewalld

Removed symlink /etc/systemd/system/firewalld.service.

# 三、下载和安装：

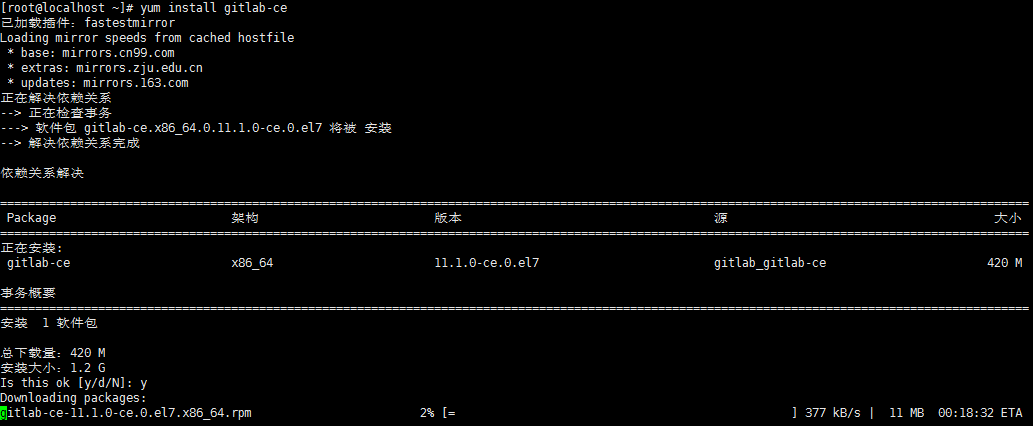
## 3.1 配置安装源：

[root@localhost ~]# curl -sS https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.rpm.sh | sudo bash



## 3.2 安装

[root@localhost ~]# yum install gitlab-ce





## 3.3 中国区域下载安装慢解决办法(配置安装源安装)

[root@localhost ~]# vi /etc/yum.repos.d/gitlab-ce.repo

[gitlab-ce]

name=Gitlab CE Repository

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

gpgcheck=0

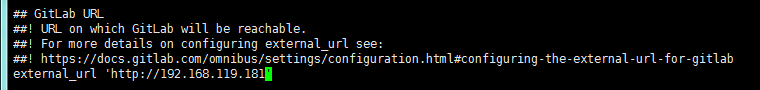
enabled=1

[root@GW-TEST2 ~]# yum makecache

[root@localhost ~]# yum install gitlab-ce

## 3.4 修改domian信息

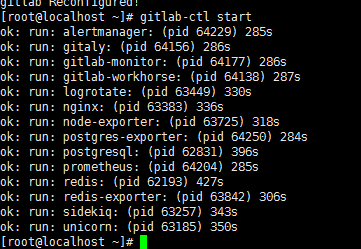
vi /etc/gitlab/gitlab.rb

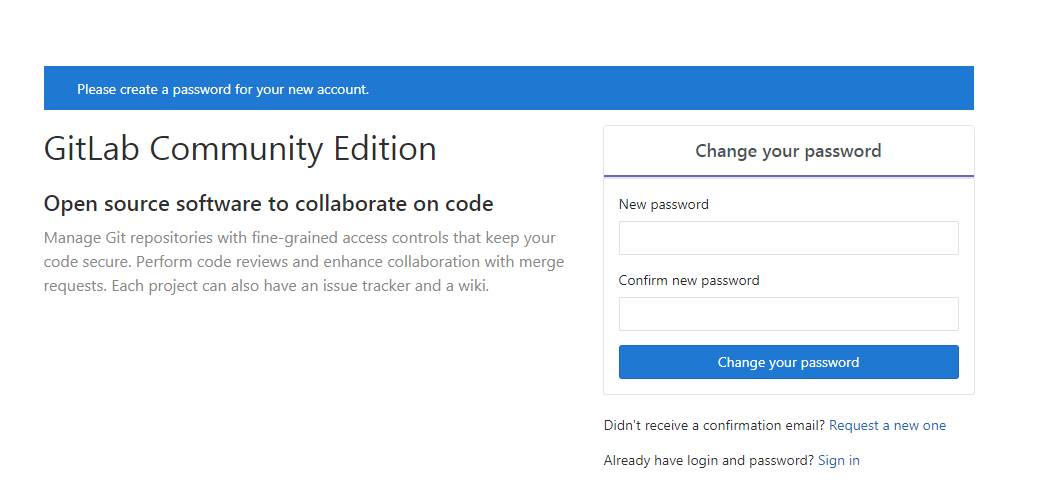


## 3.5 配置和启动

[root@localhost ~]# gitlab-ctl reconfigure

[root@localhost ~]# gitlab-ctl start





Tips:

1. 第一次登录会要求设置root的密码，管理员账号
2. 如访问不了，查看防火墙是否开启

[root@localhost ~]# systemctl stop firewalld.service

[root@localhost ~]# systemctl disable firewalld.service

## 3.6 常用gitlab 命令

gitlab-ctl start                    # 启动所有 gitlab 组件；  
gitlab-ctl stop                    # 停止所有 gitlab 组件；  
gitlab-ctl restart                # 重启所有 gitlab 组件；  
gitlab-ctl status                 # 查看服务状态；  
gitlab-ctl reconfigure         # 启动服务；

vim /etc/gitlab/gitlab.rb      # 修改默认的配置文件；  
gitlab-rake gitlab:check SANITIZE=true --trace    # 检查gitlab；  
gitlab-ctl tail                        # 查看日志；

# 四、gitlab相关配置(端口未占用可忽略)

## 4.1 默认端口配置

vim /etc/gitlab/gitlab.rb

#unicorn['port'] = 8080 修改 8070 默认是注释的去掉前面的#

unicorn['port'] = 8070

#nginx['listen\_port'] = nil 修改 8090 默认是注释的去掉前面的#

nginx['listen\_port'] = 8090

## 4.2 unicorn默认端口配置

vim /var/opt/gitlab/gitlab-rails/etc/unicorn.rb

#listen "127.0.0.1:8080", :tcp\_nopush => true

listen "127.0.0.1:8070", :tcp\_nopush => true

4.3 web服务80端口配置

vim /var/opt/gitlab/nginx/conf/gitlab-http.conf

#listen \*:80;

listen \*:8090;

## 4.3 邮件配置

vi /etc/gitlab/gitlab.rb

# Change the external\_url to the address your users will type in their browser

external\_url 'http://xxhost.com'

#Sending application email via SMTP

gitlab\_rails['smtp\_enable'] = true

gitlab\_rails['smtp\_address'] = "smtp.163.com"

gitlab\_rails['smtp\_port'] = 25

gitlab\_rails['smtp\_user\_name'] = "xxuser@163.com"

gitlab\_rails['smtp\_password'] = "xxpassword"

gitlab\_rails['smtp\_domain'] = "163.com"

gitlab\_rails['smtp\_authentication'] = :login

gitlab\_rails['smtp\_enable\_starttls\_auto'] = true

##修改gitlab配置的发信人

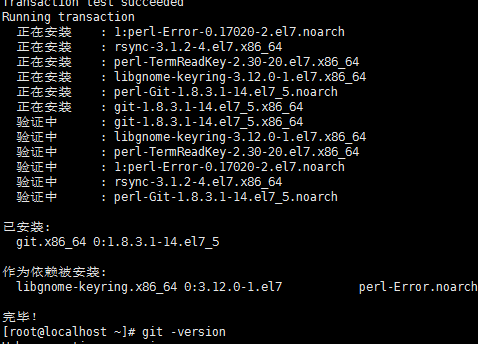
gitlab\_rails['gitlab\_email\_from'] = "xxuser@163.com"

user["git\_user\_email"] = [xxuser@163.com](mailto:xxuser@163.com)

# 五、gitlab汉化(版本不一致会出现问题)

## 5.1 Git安装

[root@localhost ~]# yum install git



[root@localhost ~]# git –version



## 5.2 汉化

### 5.2.1 查看当前gitlab版本

[root@localhost ~]# cat /opt/gitlab/embedded/service/gitlab-rails/VERSION



### 5.2.2 下载对应的patch

git clone https://gitlab.com/xhang/gitlab.git # 克隆汉化版本库

git fetch # 如果已经克隆过，则进行更新



### 5.2.3 比较汉化标签和原标签，导出patch用的diff文件

cd gitlab

git diff v11.1.0 v11.1.0-zh > ../v11.1.0-zh.diff

### 5.2.4 打补丁

#停止gitlab

[root@localhost ~]#gitlab-ctl stop

[root@localhost ~]# patch -d /opt/gitlab/embedded/service/gitlab-rails -p1 < ../v11.1.0-zh.diff

Tips：#patch可能需要自行安装:yum install -y patch

# 启动gitlab

[root@localhost ~]#gitlab-ctl start

## 5.3 汉化常见问题

### 5.3.1 访问web页面出现502错误

在启动gitlab的时候执行reconfigure之后就可以了

gitlab-ctl start

gitlab-ctl reconfigure

### 5.3.2执行patch命令出错

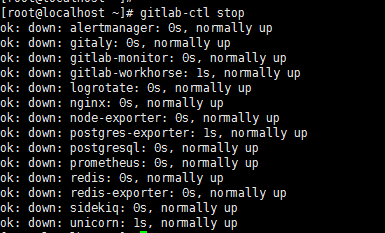




# 六、卸载gitlab

## 6.1 停止gitlab

[root@localhost ~]# gitlab-ctl stop



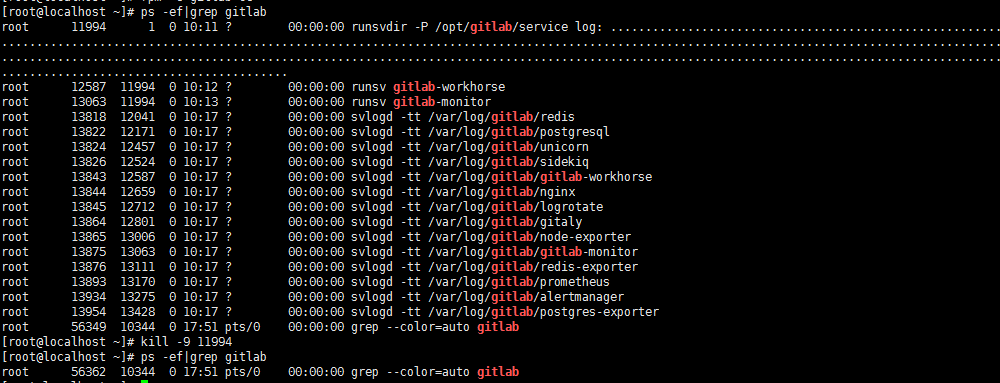
## 6.2 卸载(需要注意版本信息：gitlab-ce或gitlab-ee)

[root@localhost ~]# rpm -e gitlab-ce

## 6.3 杀死守护进程

[root@localhost ~]# ps -ef|grep gitlab

杀死第一个守护进程 /opt/gitlab/service log，再次查看gitlab信息



## 6.4 删除相关文件

删除所有包含gitlab的文件及目录

[root@localhost ~]# find / -name \*gitlab\*|xargs rm –rf

[root@localhost ~]# find / -name gitlab |xargs rm –rf

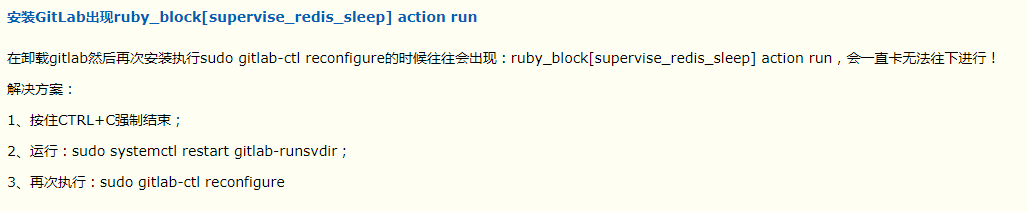
删除gitlab-ctl uninstall时自动在root下备份的配置文件（ls /root/gitlab\* 看看有没有，有也删除），通过以上几步就可以彻底卸载gitlab

## 6.5 删除再安装问题

1、按住CTRL+C强制结束；

2、运行：sudo systemctl restart gitlab-runsvdir

3、再次执行：sudo gitlab-ctl reconfigure

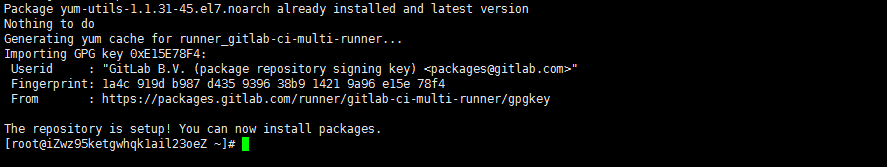


# 七、Gitlab CI/CD & sonarqube

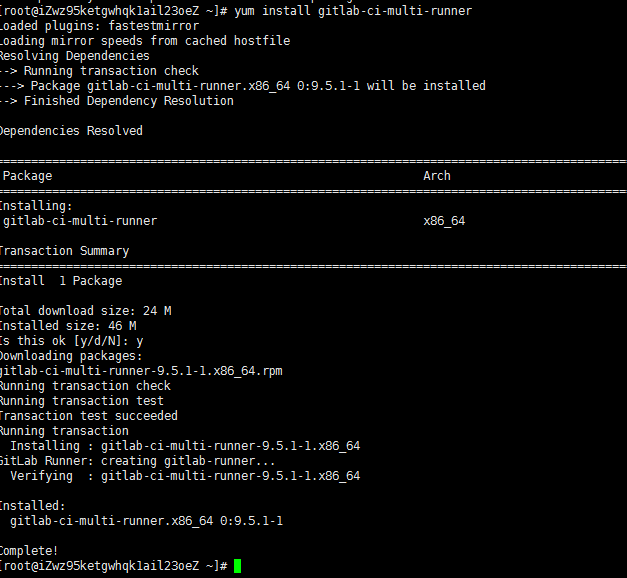
依赖Maven，请自行安装Maven并配置环境变量

## 7.1 Runner安装

[root@iZwz95ketgwhqk1ail23oeZ ~]# curl -L https://packages.gitlab.com/install/repositories/runner/gitlab-ci-multi-runner/script.rpm.sh | sudo bash



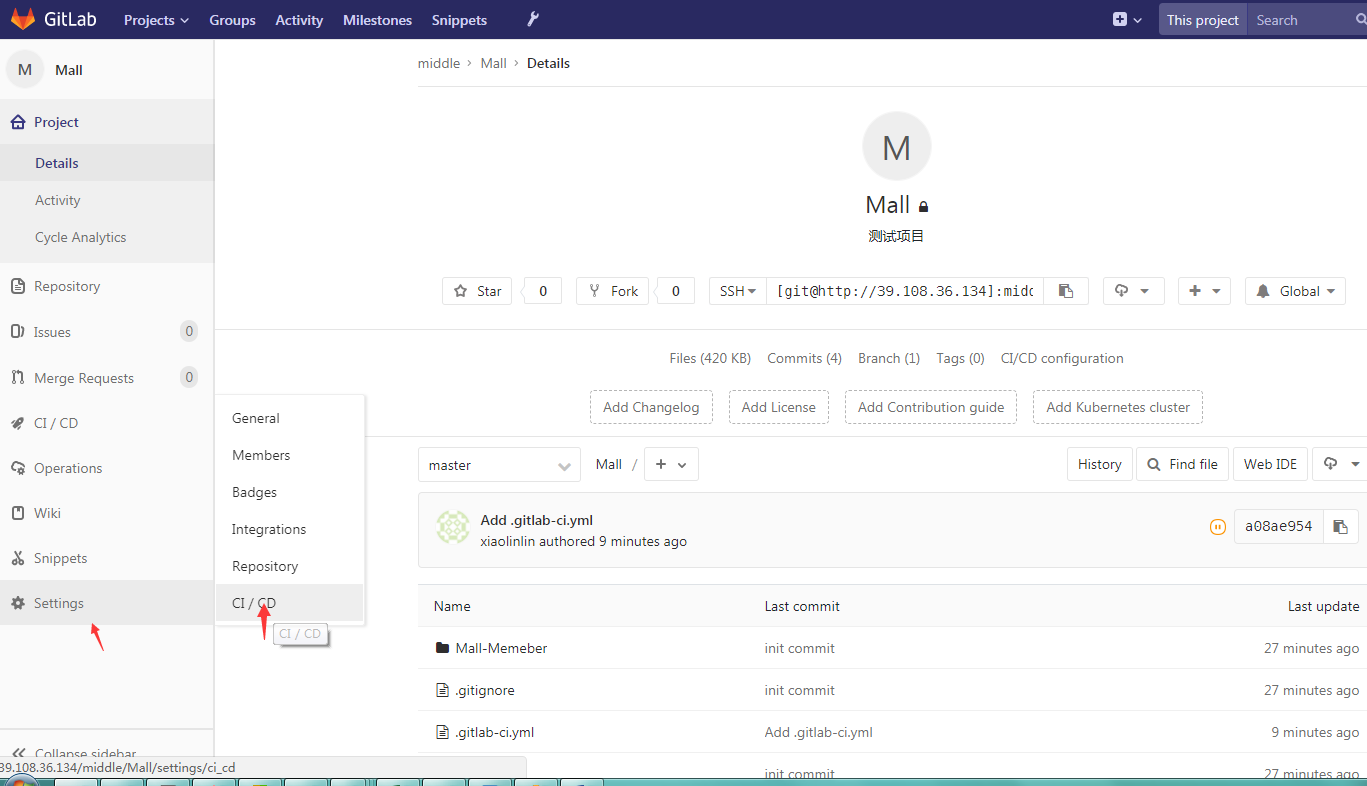
[root@iZwz95ketgwhqk1ail23oeZ ~]# yum install gitlab-ci-multi-runner

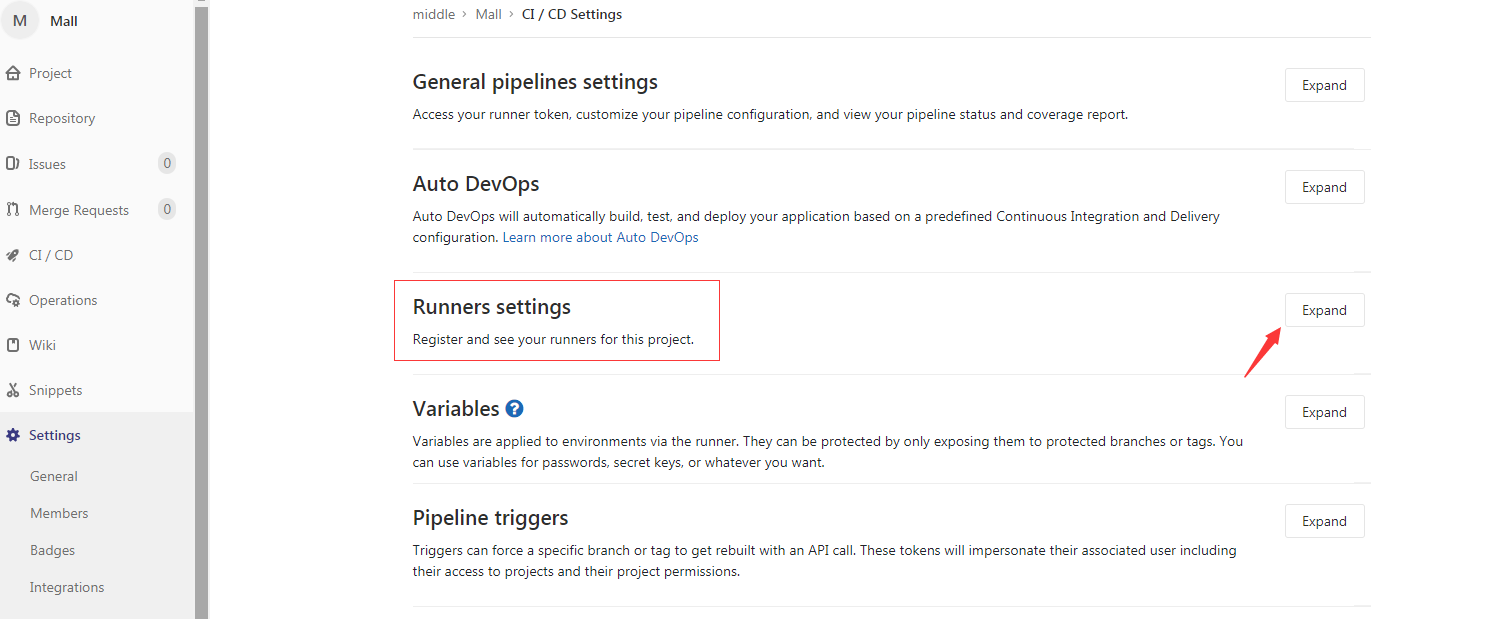


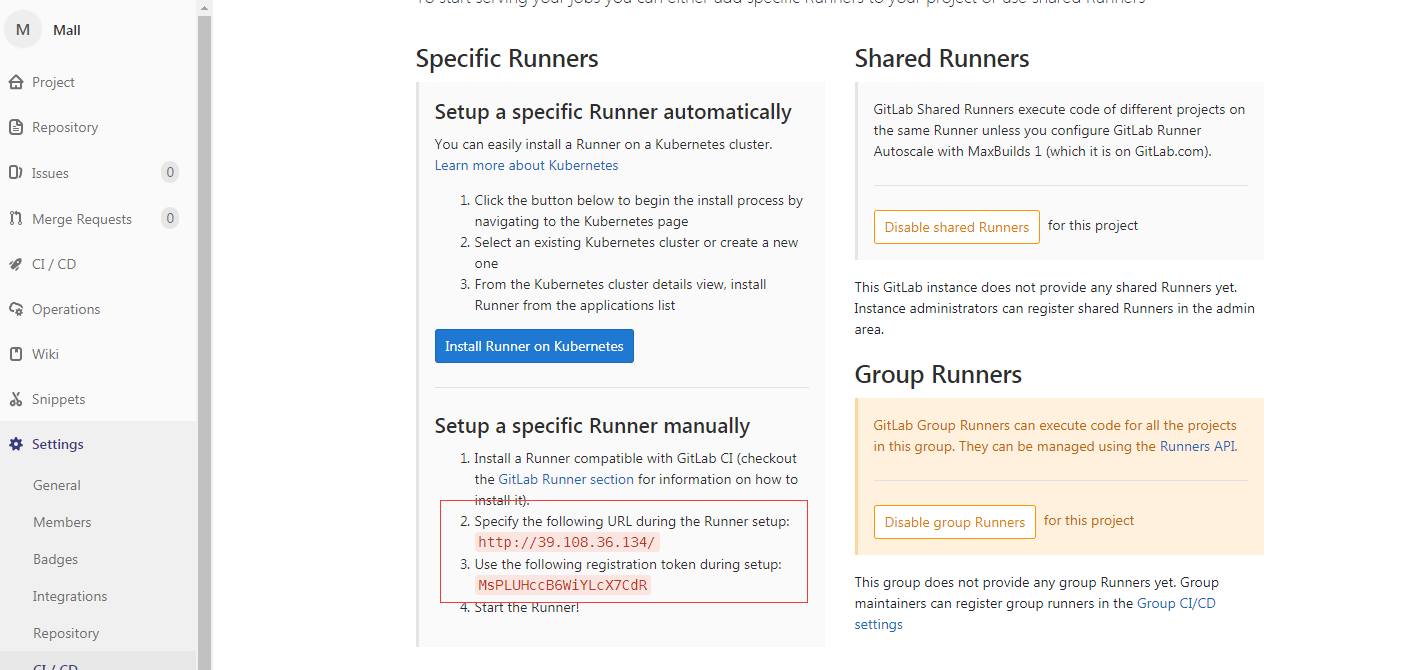
## 7.2 runner注册

### 7.2.1 从gitlab-ci获取项目级别的 url和注册的token

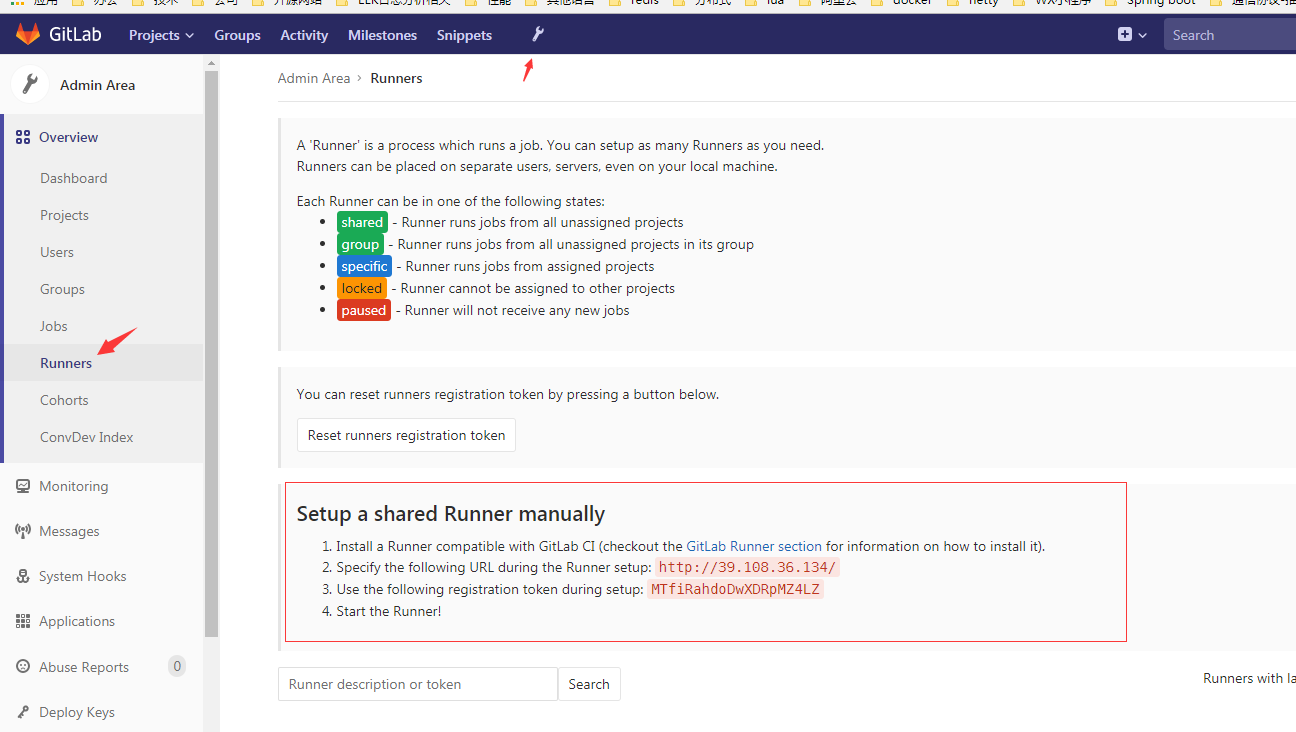
进入项目—settings--ci/cd







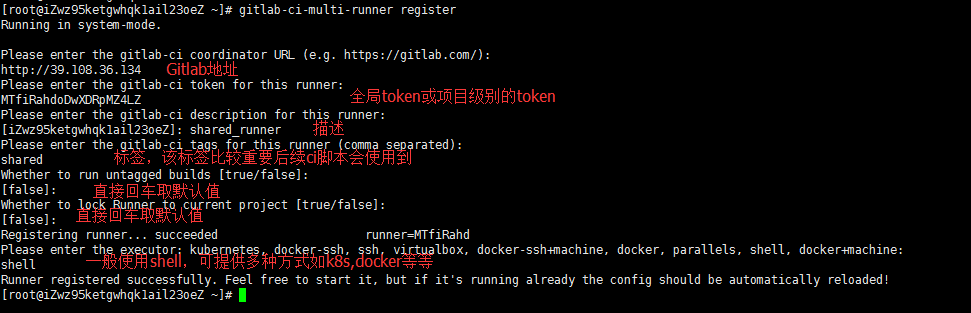
### 7.2.2 从gitlab-ci获取全局共享级别的 url和token



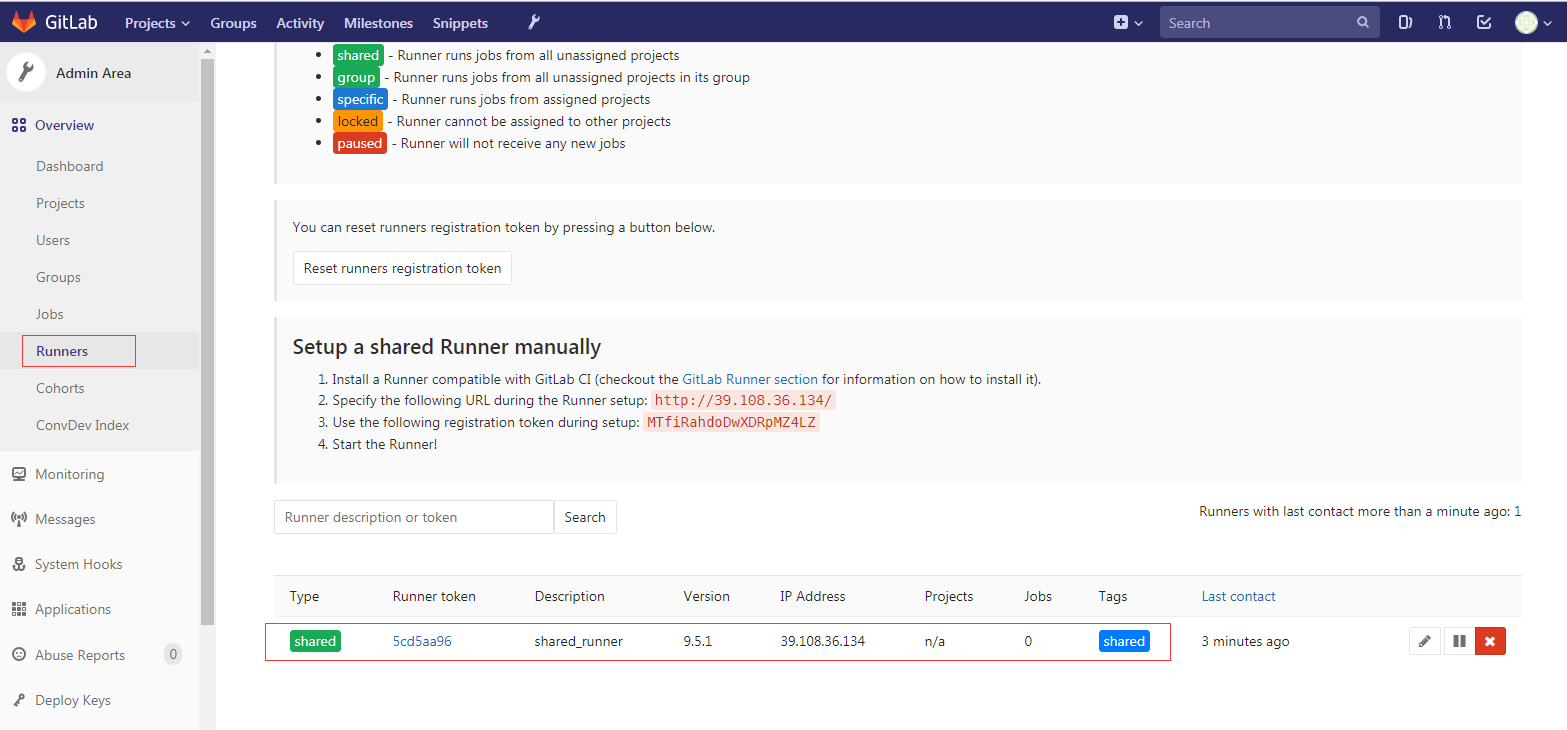
### 7.2.3 注册

本案例以全局runner为案例注册

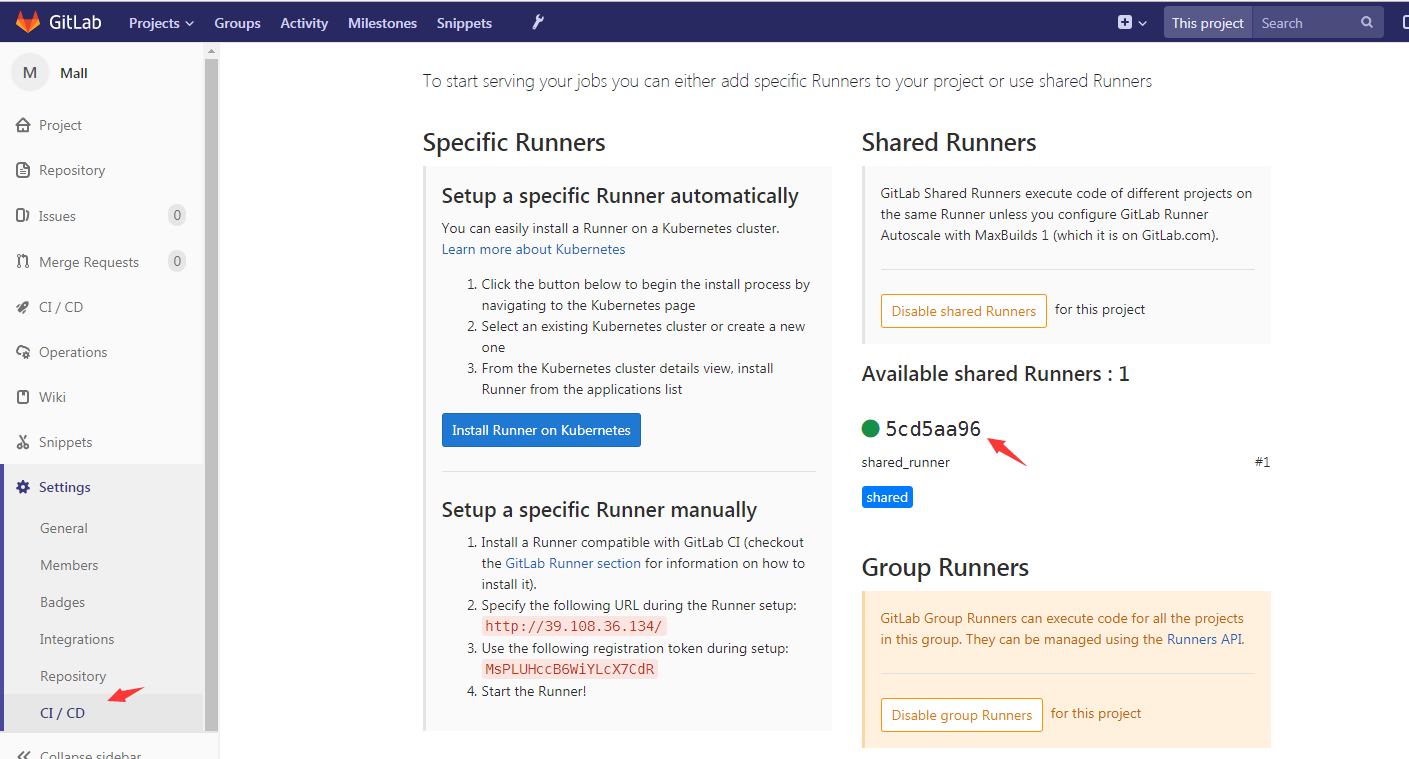
[root@iZwz95ketgwhqk1ail23oeZ ~]# gitlab-ci-multi-runner register



注册成功后可看到runner列表



具体项目中可以看到共享runner



## 7.3 结合sonarqube静态代码检查

### 7.3.1 sonarqube安装

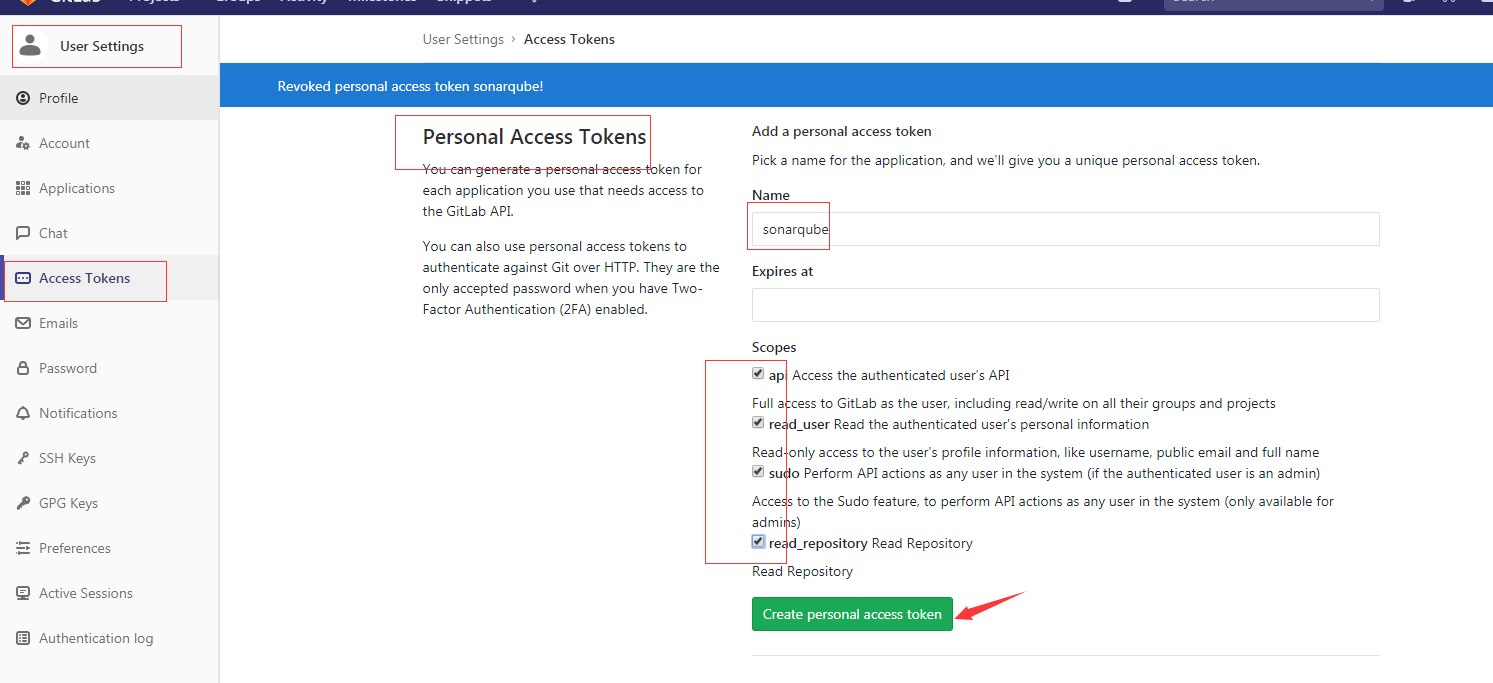
请参考另一篇安装手册《持续集成之Sonarqube安装和使用指导手册.docx》

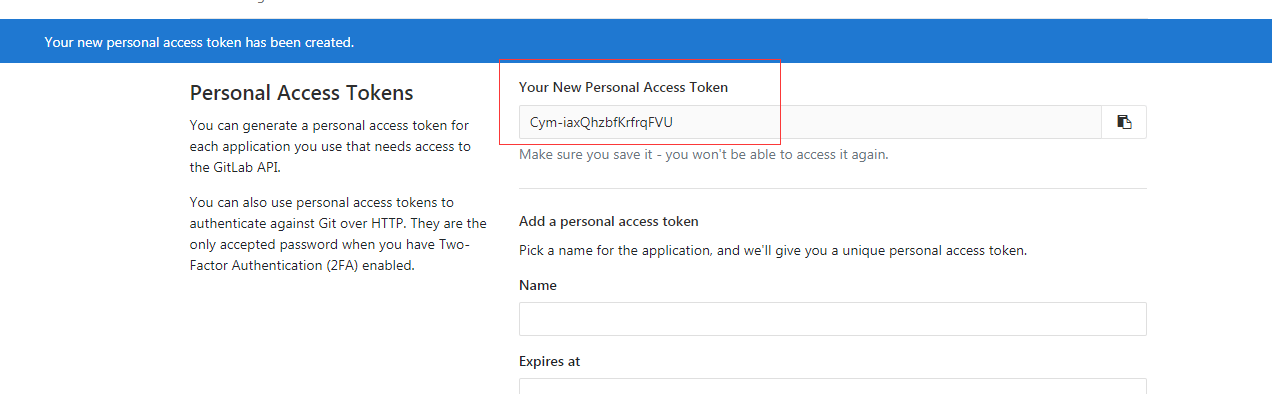
### 7.3.2 登录sonarqube 安装gitlab插件



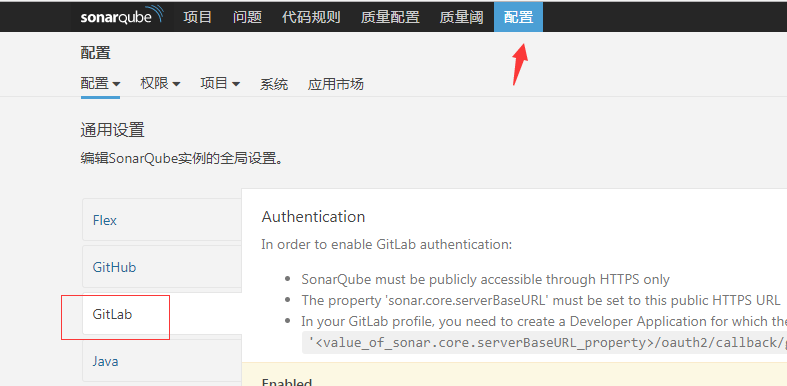
### 7.3.3 获取gitlab对接sonarqube的token

获取足够的权限comment和添加注释

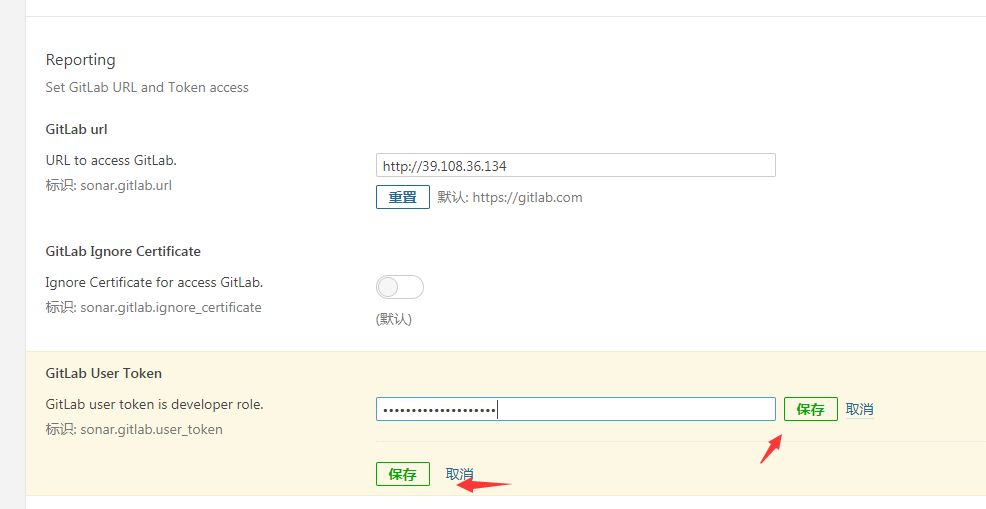




### 7.3.4 sonarqube gitlab设置





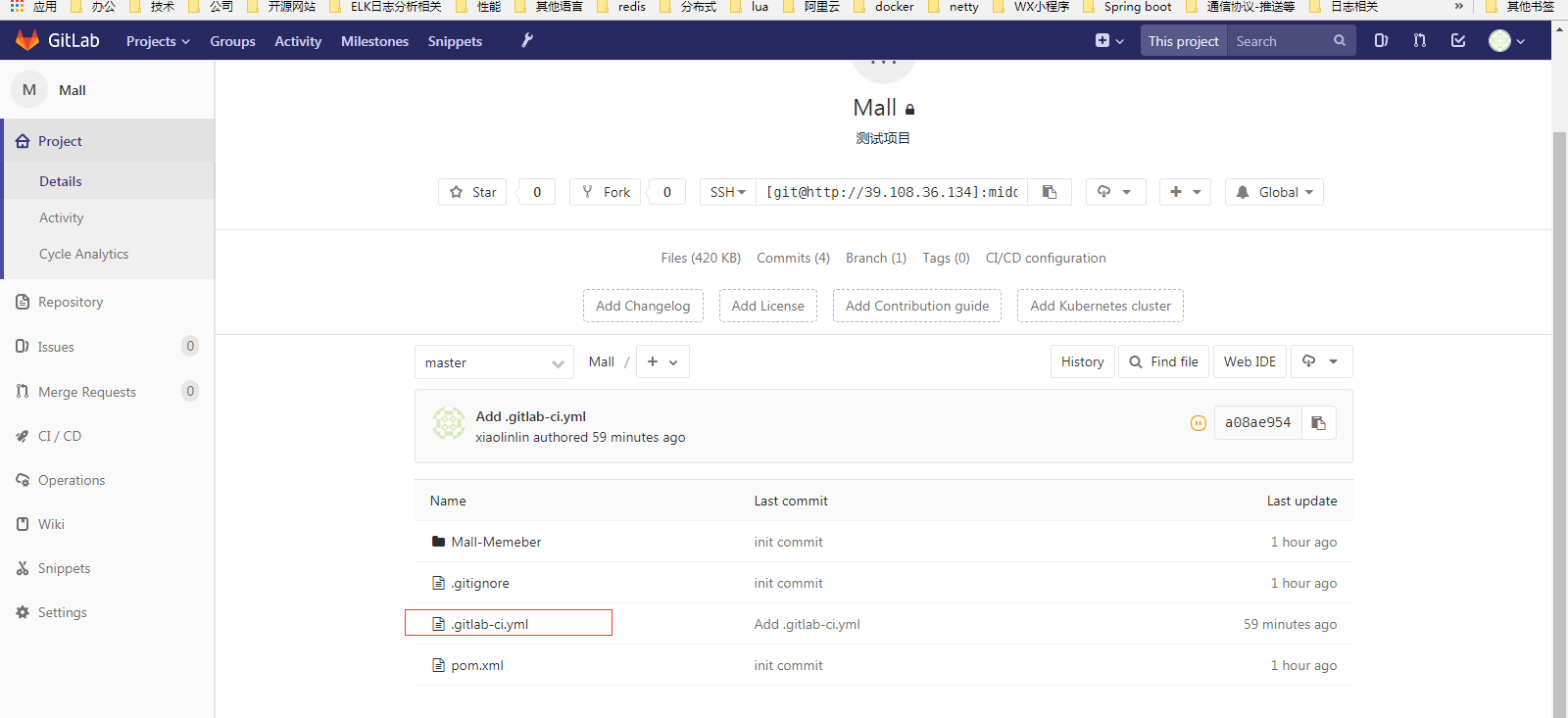


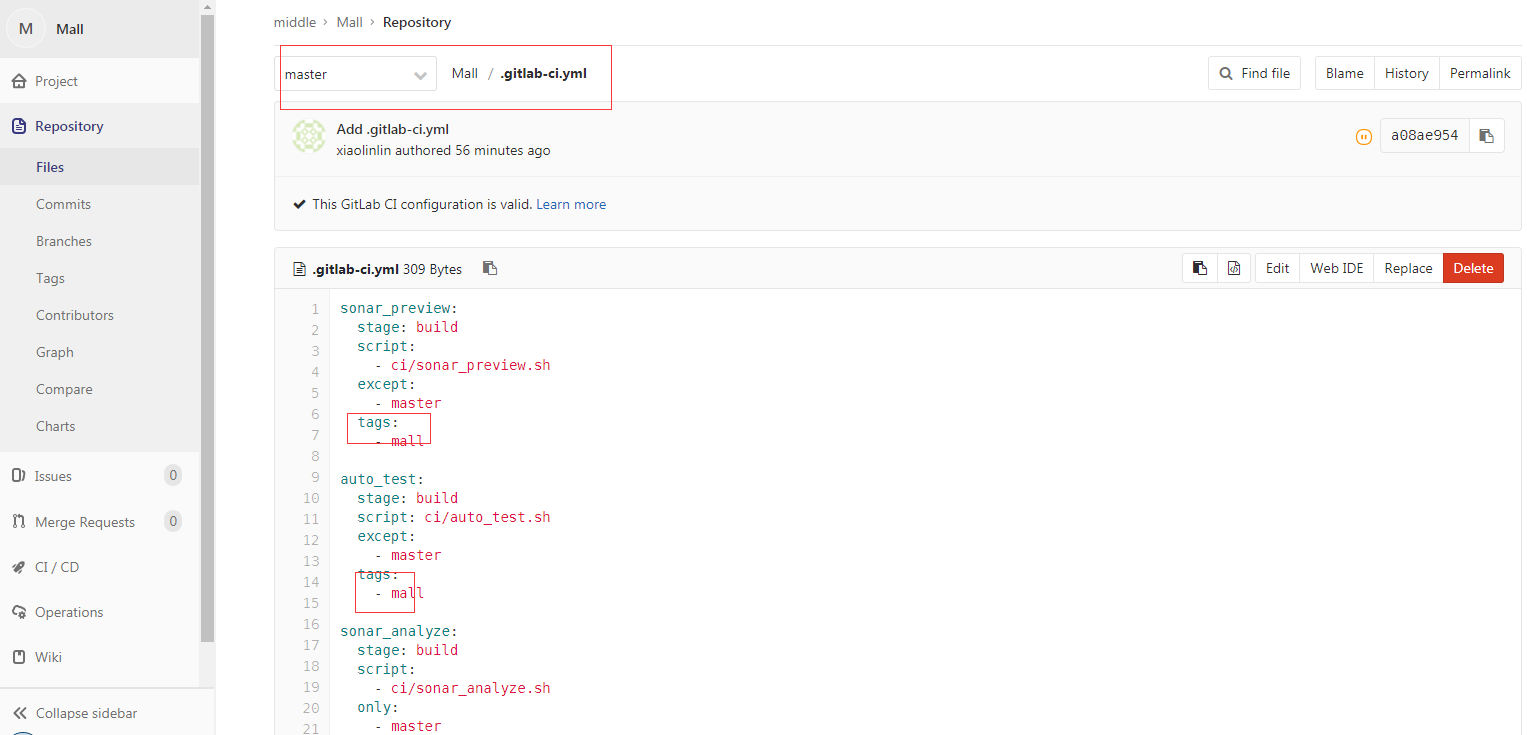
### 7.3.5 配置gitlab-runner

1) gitlab项目跟目录下配置.gitlab-ci.yml

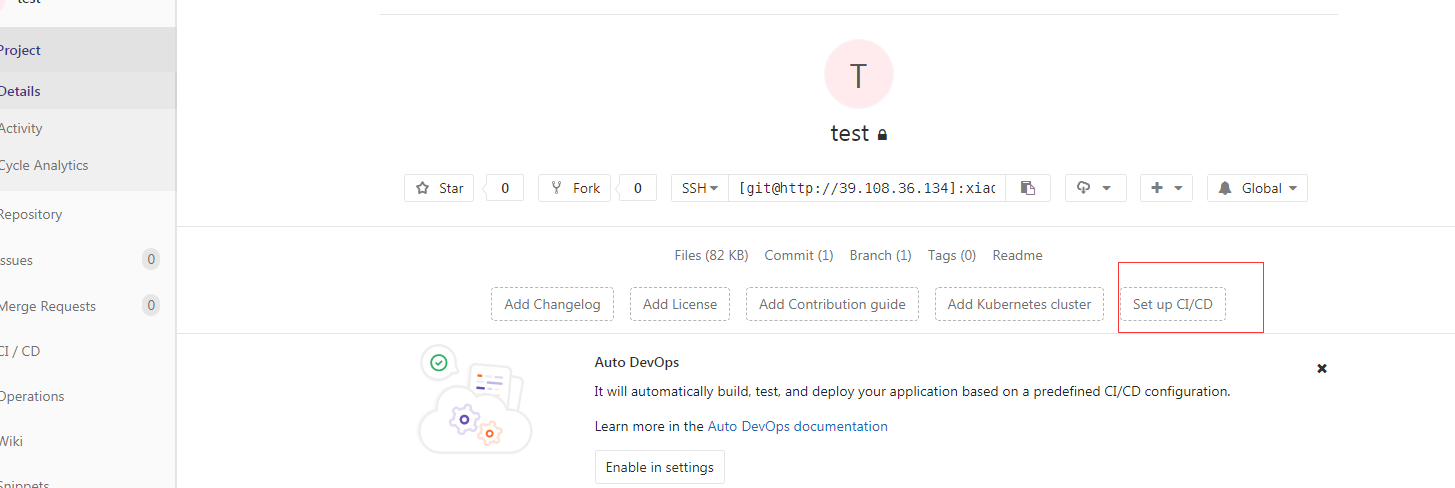
gitlab-ci配置详解参考地址：<https://segmentfault.com/a/1190000011890710>

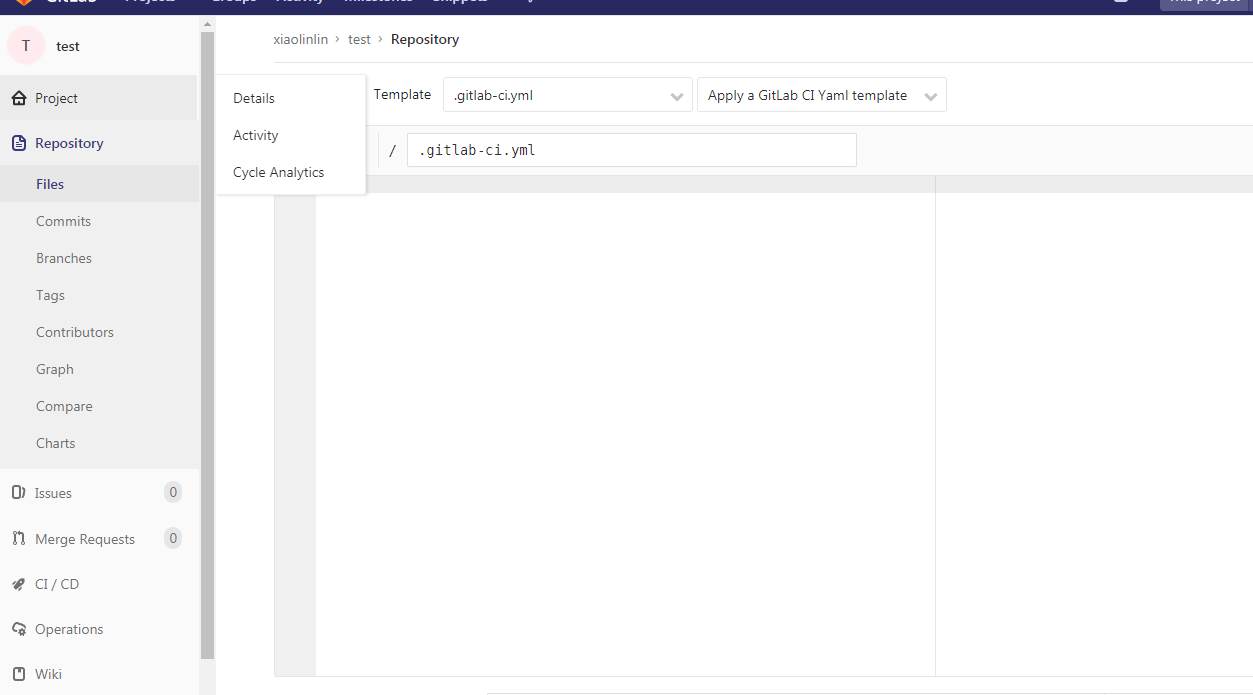
更多参考资料：<https://www.jianshu.com/p/2b43151fb92e>





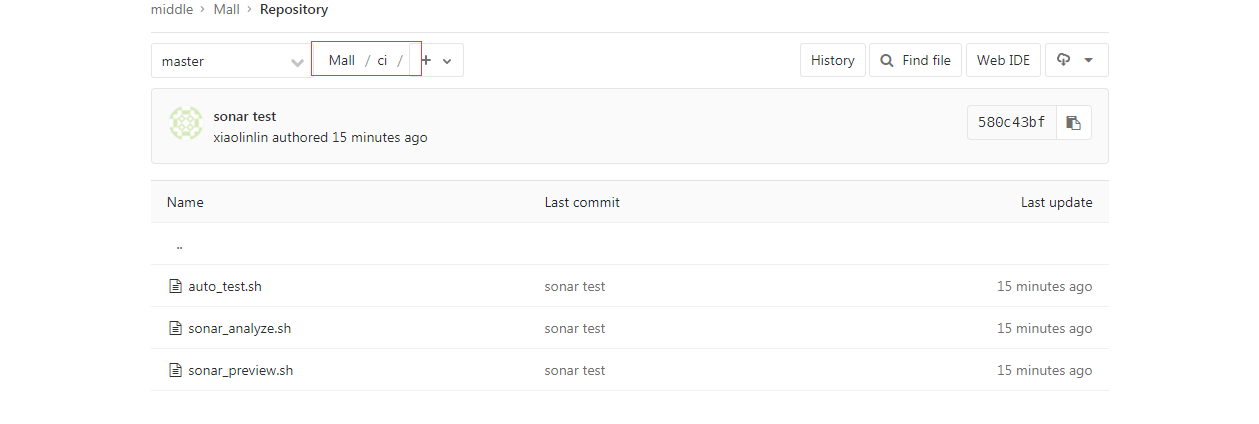
初次创建项目没有这个文件，直接创建CI/CD，如下图：





|  |
| --- |
| variables:  SCRIPT\_PATH: $CI\_PROJECT\_DIR/ci #环境变量，获取当前构建的项目路径  sonar\_preview:  stage: build  script:  - chmod -R 755 $SCRIPT\_PATH #给脚本目录可执行权限  - dos2unix $SCRIPT\_PATH/\* #将所有.sh脚本 转成unix格式  - ci/sonar\_preview.sh #执行脚本  except:  - master  tags:  - shared  auto\_test:  stage: build  script:  - chmod -R 755 $SCRIPT\_PATH  - dos2unix $SCRIPT\_PATH/\*  - ci/auto\_test.sh  except:  - master  tags:  - shared  sonar\_analyze:  stage: build  script:  - chmod -R 755 $SCRIPT\_PATH  - dos2unix $SCRIPT\_PATH/\*  - ci/sonar\_analyze.sh  only:  - master  tags:  - shared |

三个脚本内容



sonar\_preview.sh

|  |
| --- |
| *#!/bin/bash*  mvn --batch-mode verify sonar:sonar \  -Dsonar.host.url=http://39.108.36.134:9000/sonarqube \  -Dsonar.login=admin \  -Dsonar.password=admin \  -Dsonar.analysis.mode=preview \  -Dsonar.gitlab.project\_id=$CI\_PROJECT\_ID \  -Dsonar.gitlab.commit\_sha=$CI\_COMMIT\_SHA \  -Dsonar.gitlab.ref\_name=$CI\_COMMIT\_REF\_NAME  **if** [ $? -eq 0 ]; **then**  echo "sonarqube code-analyze-preview over."  **fi** |

sonar\_analyze.sh

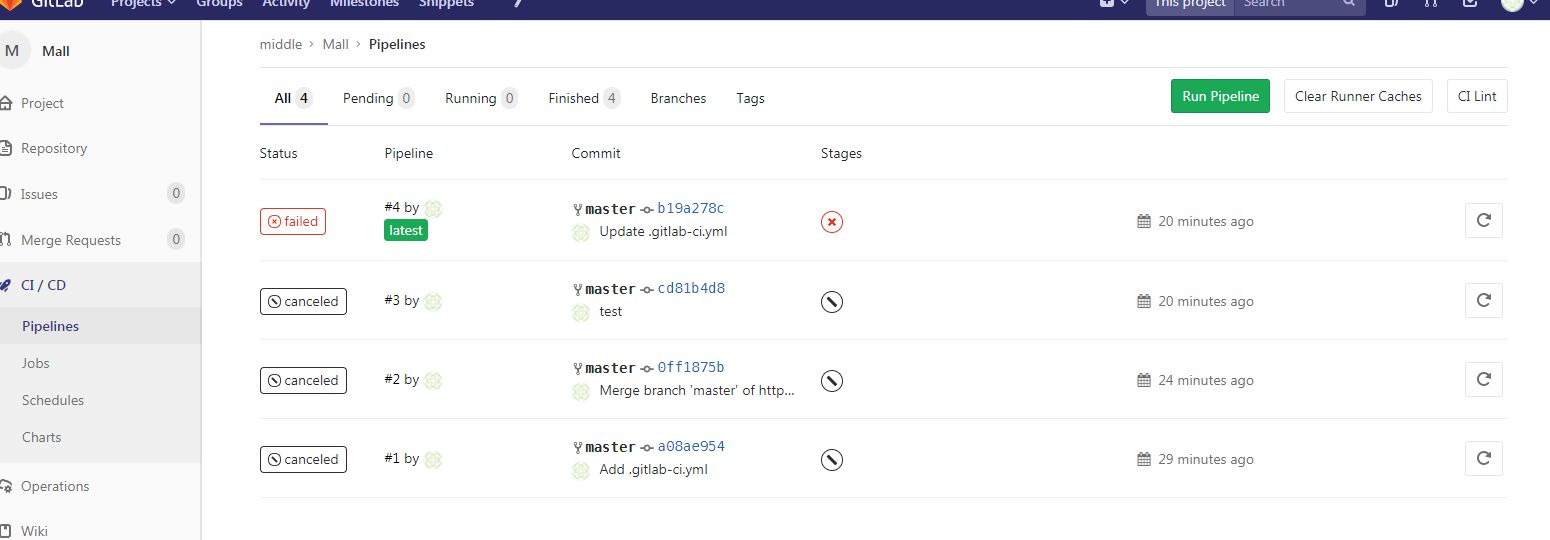
|  |
| --- |
| *#!/bin/bash*  mvn --batch-mode sonar:sonar \  -Dsonar.host.url=http://39.108.36.134:9000/sonarqube \  -Dsonar.login=admin \  -Dsonar.password=admin \  -Dsonar.issuesReport.html.enable=true \  -Dsonar.analysis.mode=preview \  -Dsonar.preview.excludePlugins=issueassign,scmstats  **if** [ $? -eq 0 ]; **then**  echo "sonarqube code-analyze over."  **fi** |

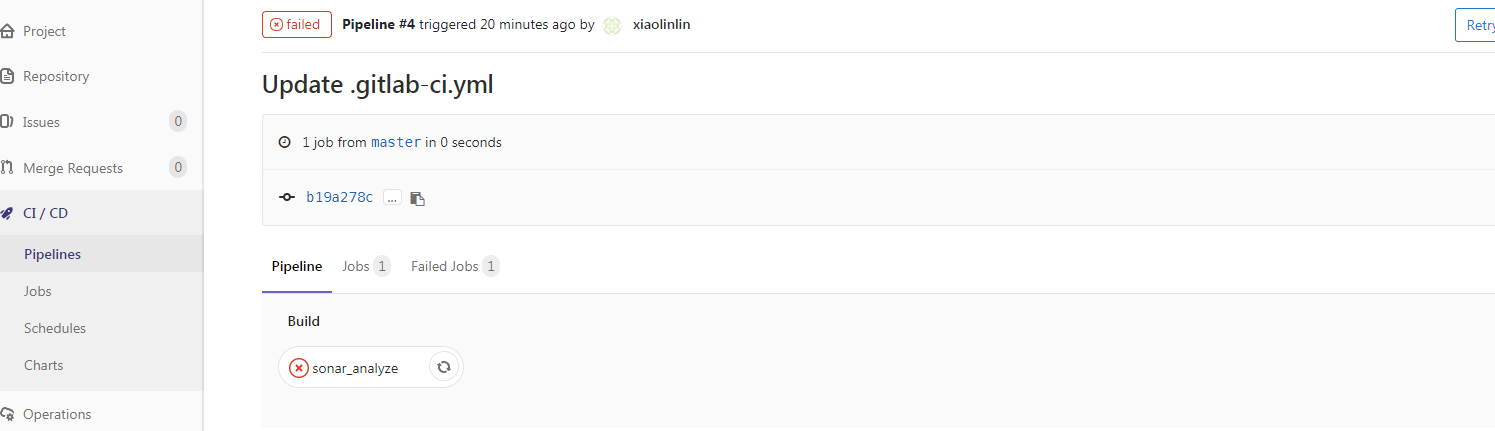
auto\_test.sh

|  |
| --- |
| *#!/bin/bash*  COMMITTER=$(git log -1 --format=%cE)  echo ${COMMITTER}  **if** [ $? -eq 0 ]; **then**  echo "do something for auto\_test here."  echo "auto\_test over."  **fi** |

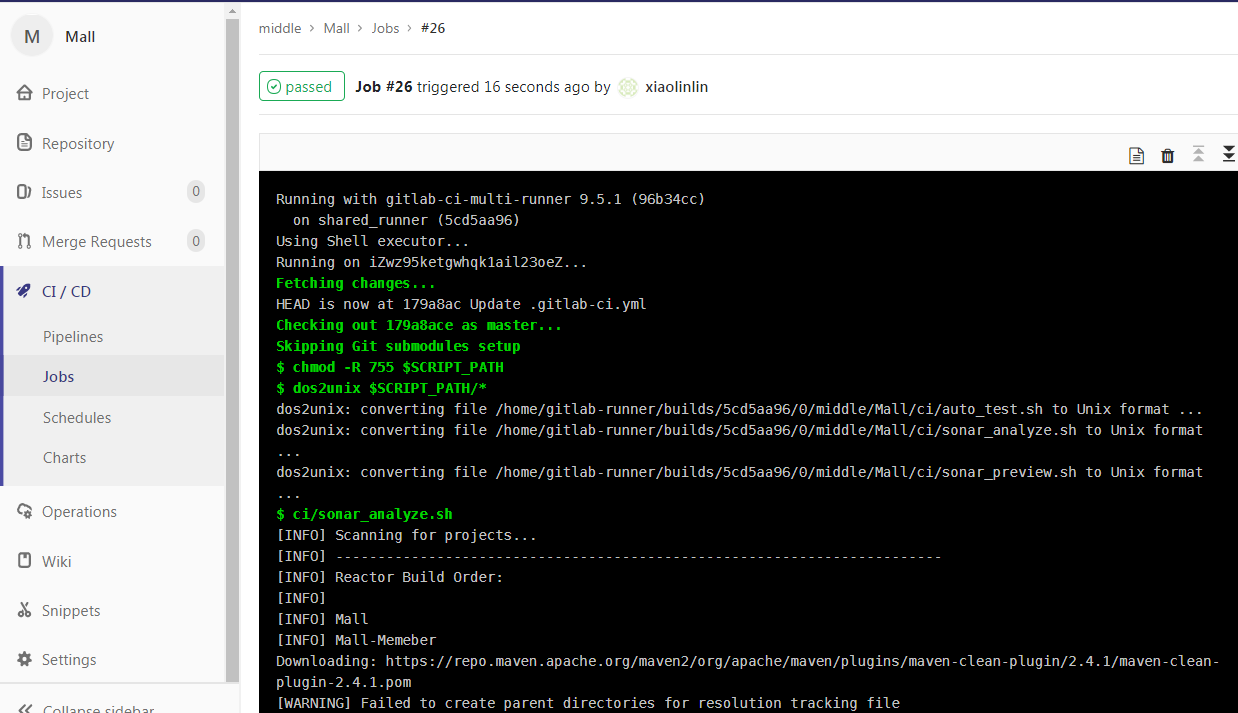
### 7.3.6 修改提交代码会触发CI/CD

查看CI信息





构建详情：



# 八、 jenkins+gitlab+sonarqube 做持续集成

参考《持续集成之Sonarqube安装和使用指导手册.docx》3.2章节，将SVN更换成git，方案一样