持续集成教程

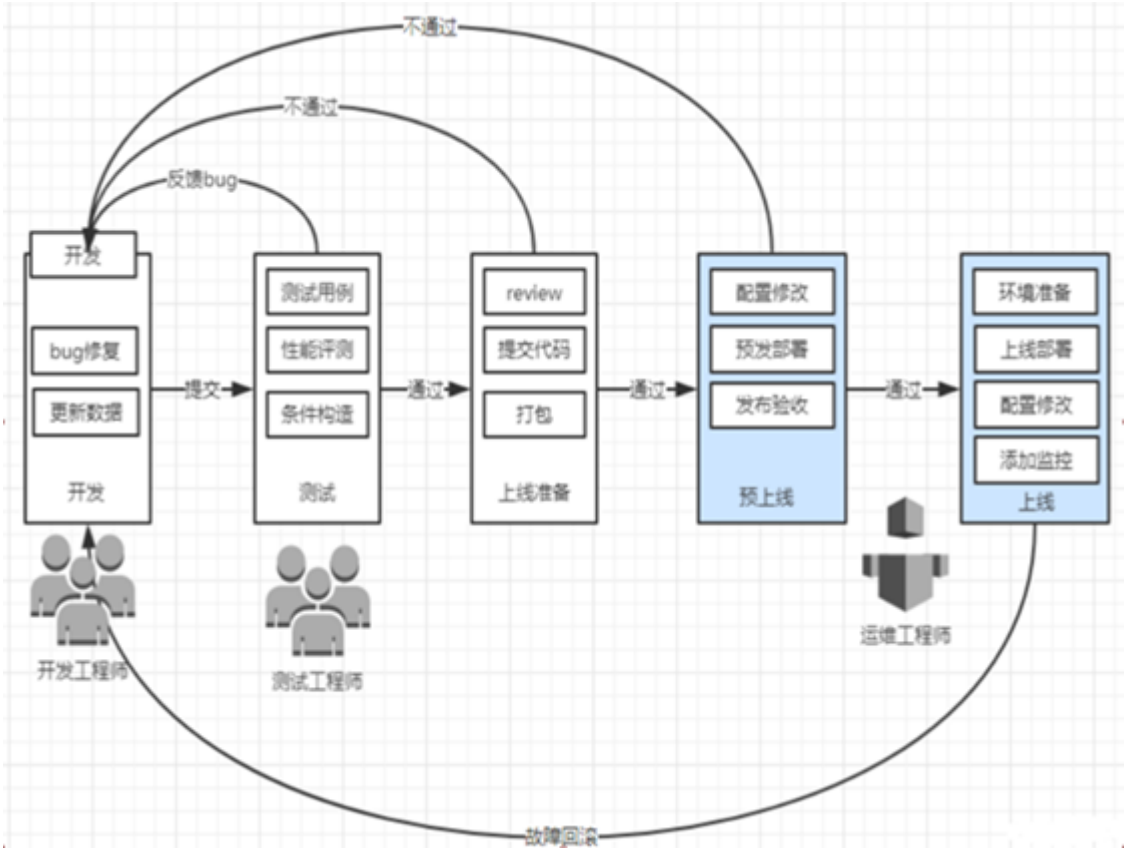
1. Devops 介绍
    01. 运维介绍
    02. Devops 是什么
    03. Devops 能干嘛
    04. Devops 如何实现
2. Git 版本控制系统
    01. 版本控制系统简介
    02. 为什么需要版本控制系统
    03. 常见版本管理工具
    04. 牛逼的人不需要解释
3. Git 安装
    01. 系统环境准备
    02. Git 安装部署
    03. Git 初始化
4. Git 常规使用
    01. 创建数据-提交数据
    02. Git 四种状态
    03. Git 基础命令
    04. Git 分支
    05. Git 标签使用
5. Github 使用
6. Gitlab 安装
7. Gitlab 使用
    01. 外观配置
    02. Gitlab 汉化配置
    03. 注册限制
    04. 创建用户及组
    05. 创建用户
    06. 把用户添加到组
    07. 创建项目
    08. 推送代码到 Gitlab
    09. 开发推送代码到 Gitlab
    10. 分支保护
    11. 代码合并
    12. Git-gui 安装
8. Gitlab 备份与恢复
    01. 备份
    02. 恢复
9. Jenkins
    01. 安装准备
    02 .安装 Jdk 和 Jenkins
    03 .配置 Jenkins
    04. 插件安装

1. Devops 介绍

01. 运维介绍

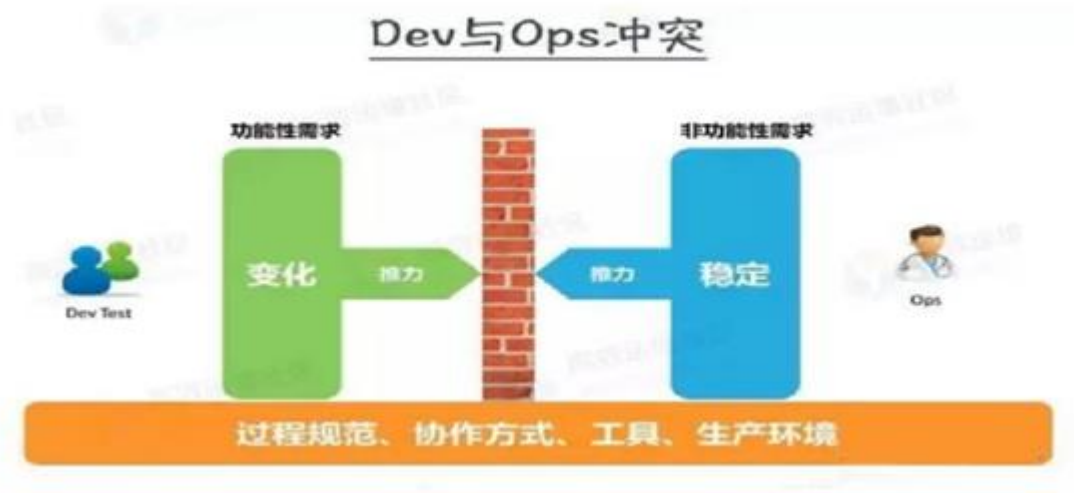
天天说运维，究竟是干什么的？先看看工作流程呗。一般来说，运维工程师在一家企业里属于个位数的岗位，甚至只有一个。面对生产中 NNN 台服务器，NN 个人员,工作量也是非常大的。

代码上线流程图



运维工程师三大核心职能

02. Devops 是什么



开发 *development*

运维 *operations*

03. Devops 能干嘛

提高产品质量

1 自动化测试
2 持续集成
3 代码质量管理工具
4 程序员鼓励师

04. Devops 如何实现

既然这么好？为什么有些公司没有
设计架构规划-代码的存储-构建-测试、预生产、部署、监控

2. Git 版本控制系统

01. 版本控制系统简介

vcs `version control system`

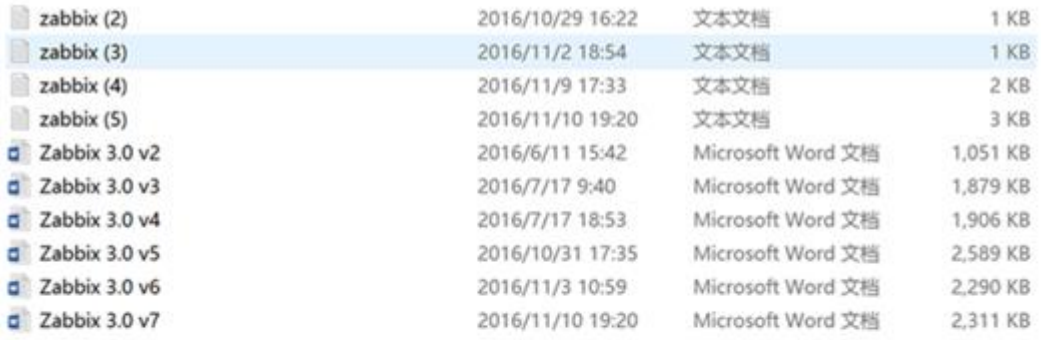
版本控制系统是一种记录一个或若干个文件内容变化，以便将来查阅特定版本内容情况的系统

记录文件的所有历史变化

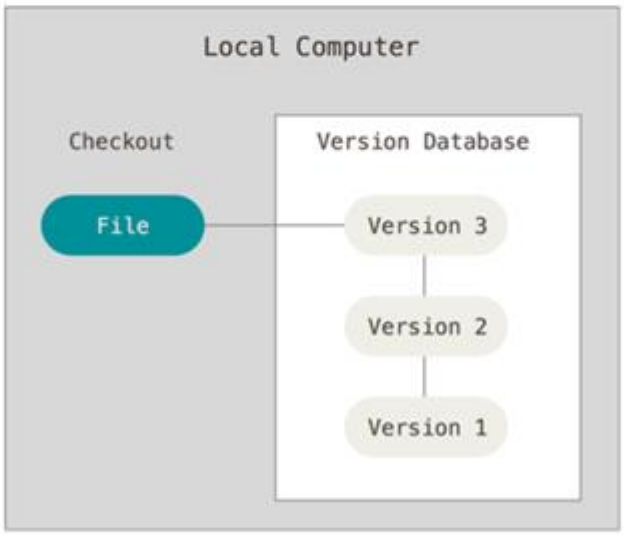
随时可恢复到任何一个历史状态

多人协作开发

02. 为什么需要版本控制系统



zabbix (2)	2016/10/29 16:22	文本文档	1 KB
zabbix (3)	2016/11/2 18:54	文本文档	1 KB
zabbix (4)	2016/11/9 17:33	文本文档	2 KB
zabbix (5)	2016/11/10 19:20	文本文档	3 KB
Zabbix 3.0 v2	2016/6/11 15:42	Microsoft Word 文档	1,051 KB
Zabbix 3.0 v3	2016/7/17 9:40	Microsoft Word 文档	1,879 KB
Zabbix 3.0 v4	2016/7/17 18:53	Microsoft Word 文档	1,906 KB
Zabbix 3.0 v5	2016/10/31 17:35	Microsoft Word 文档	2,589 KB
Zabbix 3.0 v6	2016/11/3 10:59	Microsoft Word 文档	2,290 KB
Zabbix 3.0 v7	2016/11/10 19:20	Microsoft Word 文档	2,311 KB



03. 常见版本管理工具

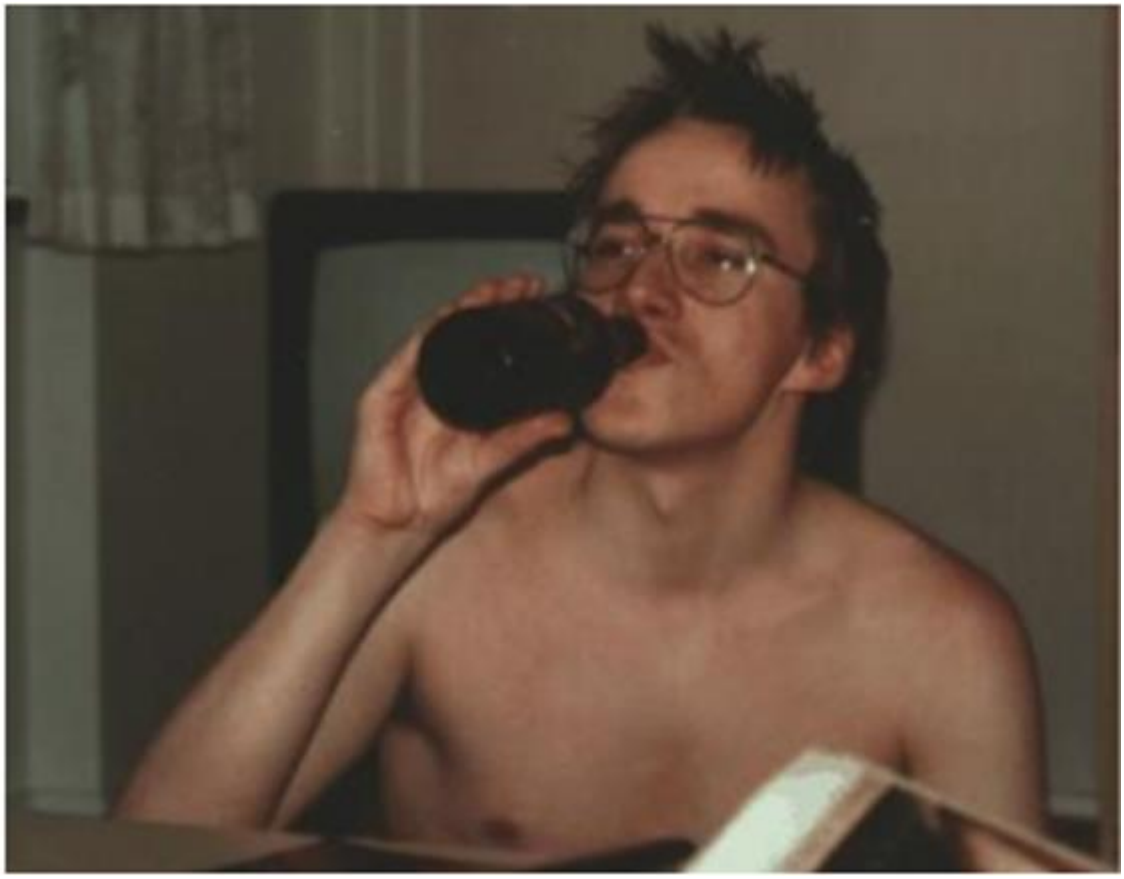
SVN

集中式的版本控制系统，只有一个中央数据仓库，如果中央数据仓库挂了或者不可访问，所有的使用者无法使用 SVN，无法进行提交或备份文件。

分布式的版本控制系统。在每个使用者电脑上都有一个完整的数据仓库。没有网络也可以使用Git。当然为了习惯及团队协作，会将本地数据同步到Git服务器或者GitHub等代码仓库。

04. 牛逼的人不需要解释

这句话被 Linux 展现的淋漓尽致



3. Git 安装

01. 系统环境准备

[root@git ~]# cat /etc/redhat-release　#查看系统版本
CentOS Linux release 7.6.1810 (Core)

[root@git ~]# uname -r　#查看内核版本
3.10.0-957.el7.x86_64

[root@git ~]# getenforce　#确认 Selinux 关闭状态
Disabled

[root@git ~]# systemctl status firewalld　#关闭防火墙
● firewalld.service - firewalld - dynamic firewall daemon
Loaded: loaded (/usr/lib/systemd/system/firewalld.service; disabled; vendor preset: enabled)
Active: inactive (dead)
Docs: man:firewalld(1)

02. Git 安装部署

[root@git ~]# yum install git -y　#安装 Git

```
[root@git ~]# git config

--global use global config file        #使用全局配置文件

--system use system config file        #使用系统级配置文件

--local use repository config file     #使用版本库级配置文件


[root@git ~]# git config --global user.name "rlb"        #配置 git 使用用户


[root@git ~]# git config --global user.email "1176494252@qq.com"      #配置 git 使用邮箱


[root@git ~]# git config --global color.ui true           #语法高亮


[root@git ~]# git config --list        #查看配置列表

user.name=rlb
user.email=1176494252@qq.com
color.ui=true


[root@git ~]# cat .gitconfig
[user]
    name = rlb
    email = 1176494252@qq.com
[color]
    ui = true
```

03. Git 初始化

```
#初始化工作目录、对已存在的目录或者对已存在的目录都可进行初始化
```

```
[root@git ~]# mkdir git_data
[root@git ~]# cd git_data/
```

#初始化

```
[root@git ~/git_data]# git init
Initialized empty Git repository in /root/git_data/.git/
```

#查看工作区状态

```
[root@git ~/git_data]# git status
# On branch master
#
# Initial commit
#
nothing to commit (create/copy files and use "git add" to track)
```

#隐藏文件介绍：

branches　　#分支目录

config　　#定义项目特有的配置选项

description　#仅供 git web 程序使用

HEAD　　#指示当前的分支

hooks　　#包含 git 钩子文件

info　　#包含一个全局排除文件(exclude 文件)

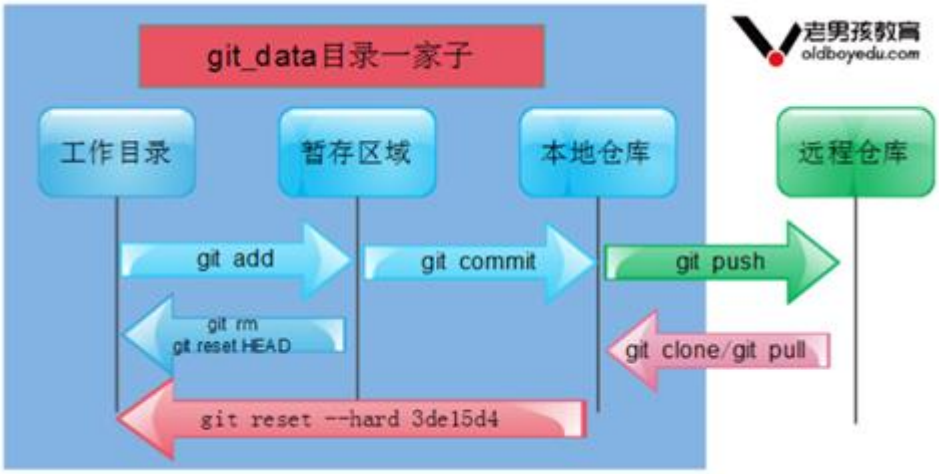
objects　　#存放所有数据内容，有 info 和 pack 两个子文件夹

refs　　#存放指向数据(分支)的提交对象的指针

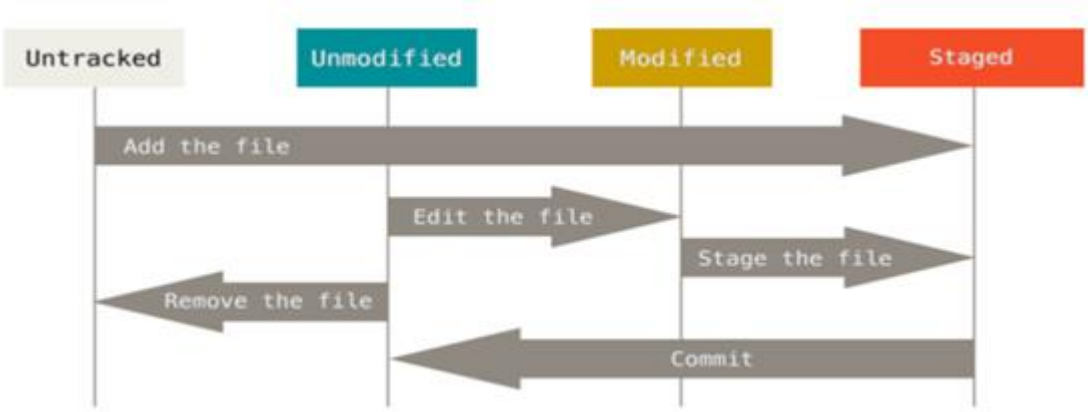
index　　#保存暂存区信息，在执行 git init 的时候，这个文件还没有

4. Git 常规使用

01. 创建数据-提交数据



02. Git 四种状态



03. Git 基础命令

```
[root@git ~/git_data]# git status
# On branch master       #位于分支 master
#
# Initial commit       #初始提交
#
nothing to commit (create/copy files and use "git add" to track)
#无文件要提交（创建/拷贝文件并使用 "git add" 建立跟踪）

#创建测试文件
[root@git ~/git_data]# touch a b c
[root@git ~/git_data]# git status
```

```
# On branch master
#
# Initial commit
#
# Untracked files:
# (use "git add <file>..." to include in what will be committed)
#
#    a
#    b
#    c
nothing added to commit but untracked files present (use "git add" to track)

#提交文件
[root@git ~/git_data]# git add a
[root@git ~/git_data]# git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
# (use "git rm --cached <file>..." to unstage)
#
#    new file: a
#
# Untracked files:
# (use "git add <file>..." to include in what will be committed)
#
#    b
#    c

#查看提交的内容
[root@git ~/git_data]# ll .git/
total 16
drwxr-xr-x 2 root root 6 2019-10-20 11:53 branches
-rw-r--r-- 1 root root 92 2019-10-20 11:53 config
-rw-r--r-- 1 root root 73 2019-10-20 11:53 description
-rw-r--r-- 1 root root 23 2019-10-20 11:53 HEAD
drwxr-xr-x 2 root root 242 2019-10-20 11:53 hooks
-rw-r--r-- 1 root root 96 2019-10-20 12:08 index      #git add a 把文件提交到了暂存区
drwxr-xr-x 2 root root 21 2019-10-20 11:53 info
drwxr-xr-x 5 root root 40 2019-10-20 12:08 objects
drwxr-xr-x 4 root root 31 2019-10-20 11:53 refs

[root@git ~/git_data]# git add .        #使用 git add . 或者 * 添加目录中所有改动过的文件
```

```
[root@git ~/git_data]# git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
# (use "git rm --cached <file>..." to unstage)
#
#    new file: a
#    new file: b
#    new file: c
#

#将 c 文件从暂存区删除
[root@git ~/git_data]# git rm --cached c
rm 'c'
[root@git ~/git_data]# ll
total 0
-rw-r--r-- 1 root root 0 2019-10-20 12:07 a
-rw-r--r-- 1 root root 0 2019-10-20 12:07 b
-rw-r--r-- 1 root root 0 2019-10-20 12:07 c

[root@git ~/git_data]# git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
# (use "git rm --cached <file>..." to unstage)
#
#    new file: a
#    new file: b
#
# Untracked files:
# (use "git add <file>..." to include in what will be committed)
#
#    c

#删除工作区文件
[root@git ~/git_data]# rm -f c

#删除文件
```

1.先从暂存区撤回到工作区、然后直接删除文件

```
[root@git ~/git_data]# git rm --cached c
[root@git ~/git_data]# rm -f c
```

2.直接从暂存区域同工作区域一同删除文件命令

```
[root@git ~/git_data]# git rm -f b
```

#提交到本地仓库

```
[root@git ~/git_data]# git commit -m "new file a"
[master (root-commit) c6b0ac2] new file a
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 a


[root@git ~/git_data]# git status
# On branch master
nothing to commit, working directory clean
```

#修改文件名称两种方法

1.本地重命名

```
[root@git ~/git_data]# mv a a.txt
[root@git ~/git_data]# git status
# On branch master
# Changes not staged for commit:
# (use "git add/rm <file>..." to update what will be committed)
# (use "git checkout -- <file>..." to discard changes in working directory)
#
#    deleted: a
#
# Untracked files:
# (use "git add <file>..." to include in what will be committed)
#
#    a.txt
no changes added to commit (use "git add" and/or "git commit -a")
```

#从暂存区删除 a 文件

```
[root@git ~/git_data]# git rm --cached a
rm 'a'
[root@git ~/git_data]# git status
# On branch master
# Changes to be committed:
# (use "git reset HEAD <file>..." to unstage)
#
```

```
#    deleted: a
#
# Untracked files:
# (use "git add <file>..." to include in what will be committed)
#
#    a.txt

#提交到暂存区
[root@git ~/git_data]# git add a.txt
[root@git ~/git_data]# git status
# On branch master
# Changes to be committed:
# (use "git reset HEAD <file>..." to unstage)
#
#    renamed: a -> a.txt        #识别到 a 和 a.txt 相同为重命名
#

#提交到本地仓库
[root@git ~/git_data]# git commit -m "commit a.txt"
[master 64e0f41] commit a.txt
1 file changed, 0 insertions(+), 0 deletions(-)
rename a => a.txt (100%)

2.直接用 git 命令重命名
[root@git ~/git_data]# git mv a.txt a
[root@git ~/git_data]# git status
# On branch master
# Changes to be committed:
# (use "git reset HEAD <file>..." to unstage)
#
#    renamed: a.txt -> a        #工作区域和暂存区域的文件同时修改文件名称
#

#提交到本地仓库
[root@git ~/git_data]# git commit -m "rename a.txt a"
[master 1a85735] rename a.txt a
1 file changed, 0 insertions(+), 0 deletions(-)
rename a.txt => a (100%)


git status     #只能查看区域状态的不同，不能查看文件内容的变化。

git diff     #查看内容的不同
```

```
[root@git ~/git_data]# echo aaa > a

#比对工作目录与暂存区的文件的不同

[root@git ~/git_data]# git diff a
diff --git a/a b/a
index e69de29..72943a1 100644
--- a/a
+++ b/a
@@ -0,0 +1 @@
+aaa

#提交 a 文件到暂存区域、在用 git diff 是相同的

[root@git ~/git_data]# git add a

#比对的是暂存区和本地仓库文件的不同处


[root@git ~/git_data]# git diff --cached a
diff --git a/a b/a
index e69de29..72943a1 100644
--- a/a
+++ b/a
@@ -0,0 +1 @@
+aaa

#提交后在比对则暂存区和本地仓库内容相同

[root@git ~/git_data]# git commit -m "modified a aaa"
[master 18c89e4] modified a
1 file changed, 1 insertion(+)

git commit #相当于虚拟机的镜像、任何操作都被做了一次快照，可恢复到任意一个位置

#查看历史的 git commit 快照操作

[root@git ~/git_data]# git log
commit 18c89e4fc7bb29f9998ceb5781e792ace28c9910
Author: rlb <1176494252@qq.com>
Date: Sun Oct 20 17:52:58 2019 +0800

modified a aaa
```

commit 1a85735ffcc03feb5e64dfd5860da0a34e003ae8
Author: rlb <1176494252@qq.com>
Date: Sun Oct 20 17:49:39 2019 +0800

rename a.txt a

commit 64e0f41387c3da32259ef8fd12ed282a3e6e5857
Author: rlb <1176494252@qq.com>
Date: Sun Oct 20 17:46:55 2019 +0800

commit a.txt

commit c6b0ac28147c1697e77eb421a8e492cac8a0e810
Author: rlb <1176494252@qq.com>
Date: Sun Oct 20 17:41:46 2019 +0800

new file a

#使用一行简单的显示 commit 信息

[root@git ~/git_data]# git log --oneline
18c89e4 modified a
1a85735 rename a.txt a
64e0f41 commit a.txt
c6b0ac2 new file a

#显示当前的指针指向哪里，在哪个快照下工作

[root@git ~/git_data]# git log --oneline --decorate
18c89e4 (HEAD, master) modified a
1a85735 rename a.txt a
64e0f41 commit a.txt
c6b0ac2 new file a

#显示具体内容的变化

[root@git ~/git_data]# git log -p

#只显示 1 条内容
[root@git ~/git_data]# git log -1
commit 18c89e4fc7bb29f9998ceb5781e792ace28c9910

Author: rlb <1176494252@qq.com>
Date: Sun Oct 20 17:52:58 2019 +0800

modified a

#恢复历史数据

1.只更改了当前目录

[root@git ~/git_data]# echo "333" >> a

[root@git ~/git_data]# git status
On branch master
Changes not staged for commit:
(use "git add <file>..." to update what will be committed)
(use "git checkout -- <file>..." to discard changes in working directory)
#
modified: a
#
no changes added to commit (use "git add" and/or "git commit -a")

#从暂存区覆盖本地工作目录

[root@git ~/git_data]# git checkout -- a

[root@git ~/git_data]# cat a
aaa

2.修改了本地目录且同时提交到了暂存区

[root@git ~/git_data]# echo ccc >> a   #添加新内容

[root@git ~/git_data]# git add .   #提交到暂存区

#比对暂存区和本地仓库的内容

[root@git ~/git_data]# git diff --cached
diff --git a/a b/a
index 72943a1..959479a 100644
--- a/a
+++ b/a
@@ -1 +1,2 @@
aaa
+ccc

```
[root@git ~/git_data]# git status
# On branch master
# Changes to be committed:
# (use "git reset HEAD <file>..." to unstage)
#
#    modified: a
#
```

#本地仓库覆盖暂存区域

```
[root@git ~/git_data]# git reset HEAD a
Unstaged changes after reset:
M    a
```

#比对工作区与暂存区的不同

```
[root@git ~/git_data]# git diff a
diff --git a/a b/a
index 72943a1..959479a 100644
--- a/a
+++ b/a
@@ -1 +1,2 @@
aaa
+ccc
```

#比对暂存区与本地仓库的不同

```
[root@git ~/git_data]# git diff --cached a
```

3.修改了工作目录后提交到了暂存区和本地仓库后进行数据恢复

```
[root@git ~/git_data]# echo bbb >>a        #添加新的内容

[root@git ~/git_data]# git commit -am "add bbb"    #提交到本地仓库
[master 6118c8d] add bbb
1 file changed, 2 insertions(+)

[root@git ~/git_data]# echo ccc >>a
```

#这时候发现改错代码了，想还原某一次提交的文件快照

```
[root@git ~/git_data]# git commit -am "add ccc"
```

[master cc5c366] add ccc
1 file changed, 1 insertion(+)

[root@git ~/git_data]# git log --oneline
cc5c366 add ccc
6118c8d add bbb
18c89e4 modified a
1a85735 rename a.txt a
64e0f41 commit a.txt
c6b0ac2 new file a

Git 服务程序中有一个叫做 HEAD 的版本指针，当用户申请还原数据时，其实就是将 HEAD 指针指向到某个特定的提交版本，但是因为 Git 是分布式
版本控制系统，为了避免历史记录冲突，故使用了 SHA-1 计算出十六进制的哈希字串来区分每个提交版本，另外默认的 HEAD 版本指针会指向到最近的一次提交版本记录

[root@git ~/git_data]# git reset --hard 18c89e4
HEAD is now at 18c89e4 modified a
刚刚的操作实际上就是改变了一下 HEAD 版本指针的位置，就是你将 HEAD 指针放在那里，那么你的当前工作版本就会定位在那里，要想把内容再还原到最新提交的版本，先看查看下提交版本号

#打开发现回退错了，应该回退到 bbb 版本

[root@git ~/git_data]# cat a
aaa

#这时候查看 log 没有 commit bbb 的历史了

[root@git ~/git_data]# git log --oneline
18c89e4 modified a
1a85735 rename a.txt a
64e0f41 commit a.txt
c6b0ac2 new file a

怎么搞得？竟然没有了 add bbb 这个提交版本记录？
原因很简单，因为我们当前的工作版本是历史的一个提交点，这个历史提交点还没有发生过 add bbb 更新记录，所以当然就看不到了，要是想"还原到未来"的历史更新点，可以用 git reflog 命令来查看所有的历史记录：

#使用 git reflog 可查看总历史内容

[root@git ~/git_data]# git reflog
18c89e4 HEAD@{0}: reset: moving to 18c89e4
cc5c366 HEAD@{1}: commit: add ccc
6118c8d HEAD@{2}: commit: add bbb

18c89e4 HEAD@{3}: commit: modified a
1a85735 HEAD@{4}: commit: rename a.txt a
64e0f41 HEAD@{5}: commit: commit a.txt
c6b0ac2 HEAD@{6}: commit (initial): new file a

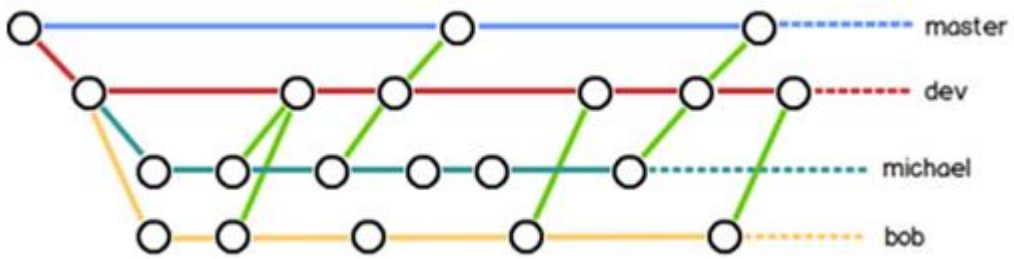
#然后使用 reset 回到 bbb 的版本内容下

[root@git ~/git_data]# git reset --hard 6118c8d
HEAD is now at 6118c8d add bbb

[root@git ~/git_data]# cat a
aaa
bbb

04. Git 分支

分支即是平行空间，假设你在为某个手机系统研发拍照功能，代码已经完成了 80%，但如果将这不完整的代码直接提交到 git 仓库中，又有可能影响到其他人的工作，此时我们便可以在该软件的项目之上创建一个名叫"拍照功能"的分支，这种分支只会属于你自己，而其他人看不到，等代码编写完成后再与原来的项目主分支合并下即可，这样即能保证代码不丢失，又不影响其他人的工作。



一般在实际的项目开发中，我们要尽量保证 master 分支是非常稳定的，仅用于发布新版本，平时不要随便直接修改里面的数据文件，而工作的时候则可以创建不同的工作分支，等到工作完成后在合并到 master 分支上面，所以团队的合作分支看起来会像上面的图那样。

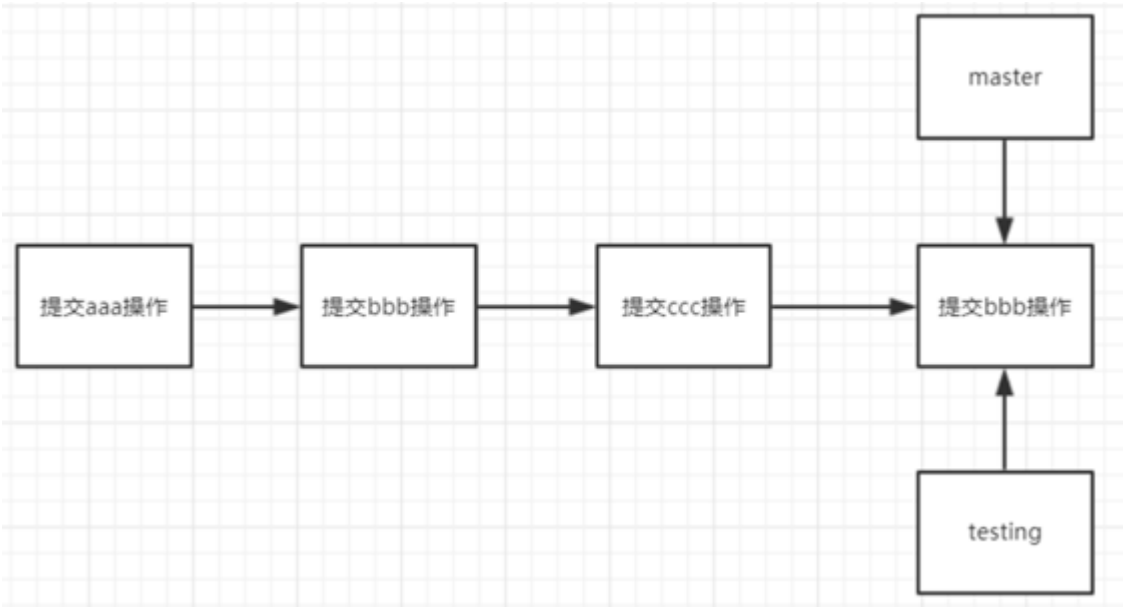
[root@git ~/git_data]# git log --oneline --decorate
ef16b0b (HEAD, master) add ddd　　#默认分支指向你最后一次的提交 HEAD 头、指针
6118c8d add bbb　　#HEAD 指针指向哪个分支、说明你当前在哪个分支下工作
18c89e4 modified a
1a85735 rename a.txt a
64e0f41 commit a.txt
c6b0ac2 new file a

#新建 testing 分支

[root@git ~/git_data]# git branch testing

*号在哪里就说明当前在哪个分支上如下图所示：

[root@git ~/git_data]# git branch
* master
testing



#通过命令查看分支指向

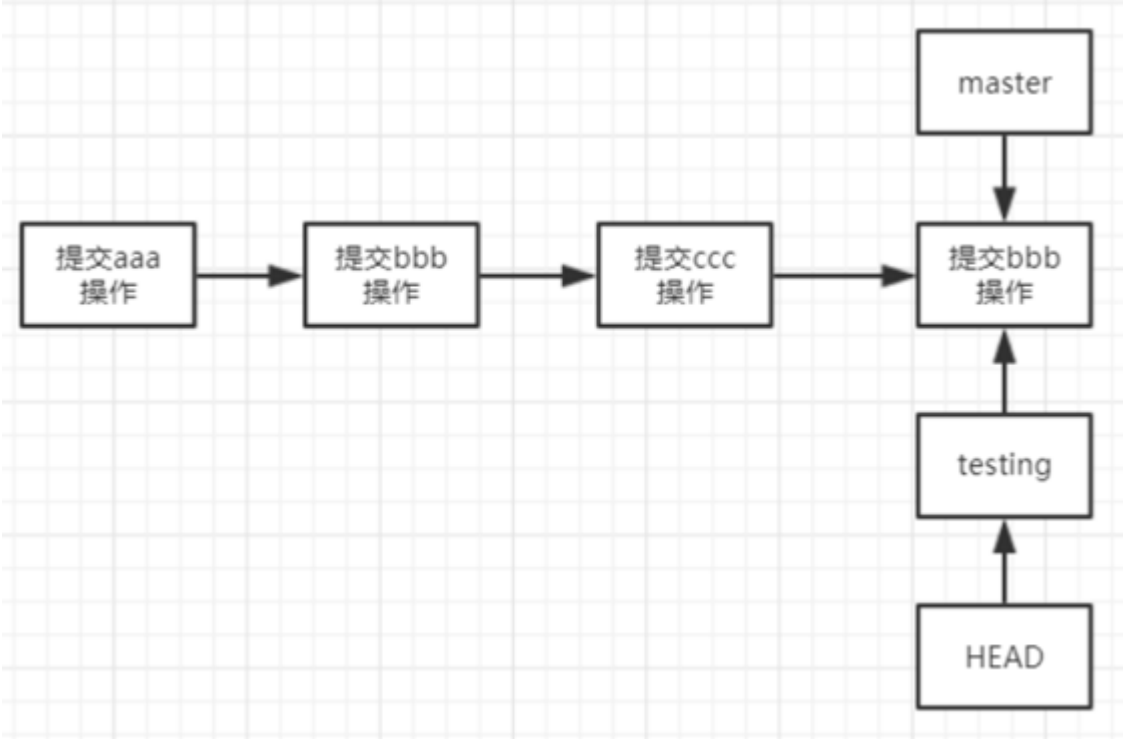
[root@git ~/git_data]# git log --oneline --decorate
ef16b0b (HEAD, testing, master) add ddd
6118c8d add bbb
18c89e4 modified a
1a85735 rename a.txt a
64e0f41 commit a.txt
c6b0ac2 new file a

#切换到 testing 分支、对应的 HEAD 指针也指向了 testing

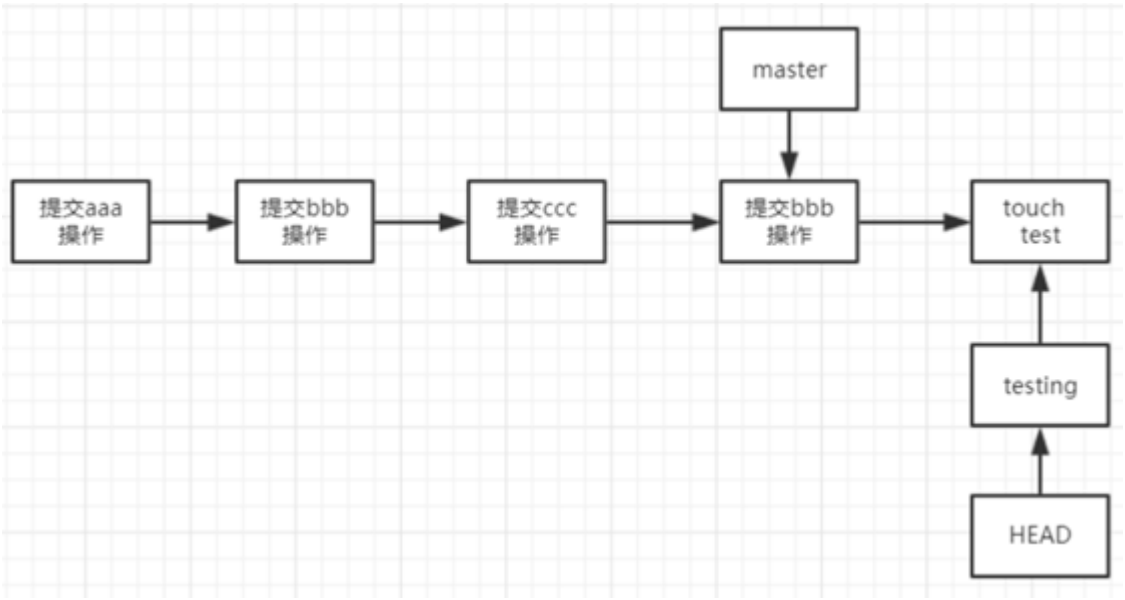
[root@git ~/git_data]# git checkout testing
Switched to branch 'testing'

#HEAD 指针指向了 testing

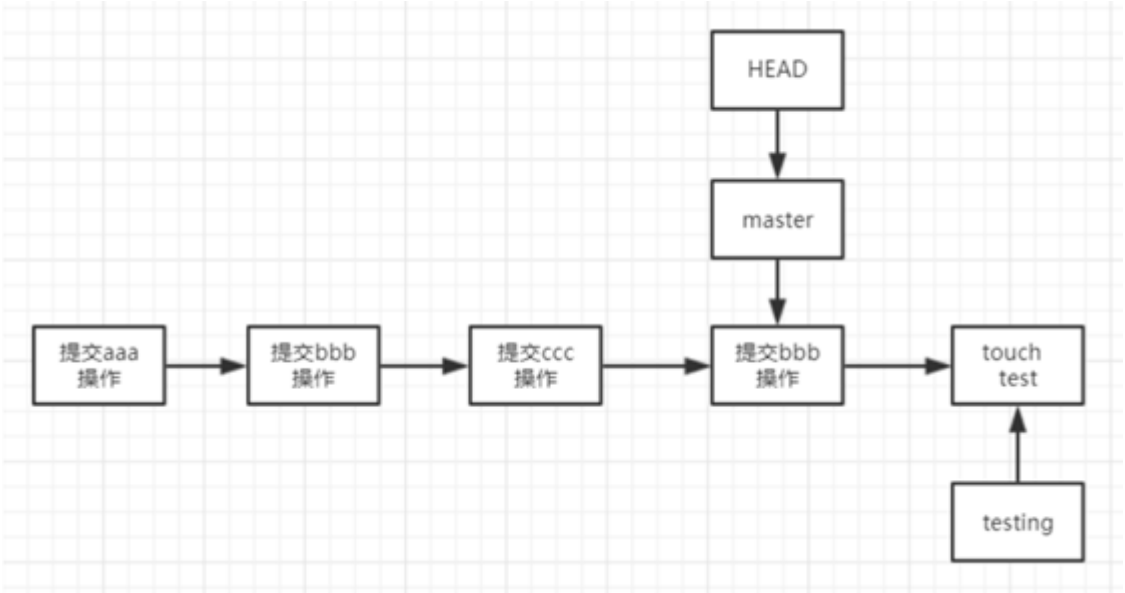
[root@git ~/git_data]# git branch
master
* testing




```
[root@git ~/git_data]# touch test
[root@git ~/git_data]# git add .
[root@git ~/git_data]# git commit -m 'commit test'
[testing 0924a70] commit test
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 test
[root@git ~/git_data]# git log --oneline --decorate
0924a70 (HEAD, testing) commit test
ef16b0b (master) add ddd
6118c8d add bbb
18c89e4 modified a
1a85735 rename a.txt a
64e0f41 commit a.txt
c6b0ac2 new file a
```

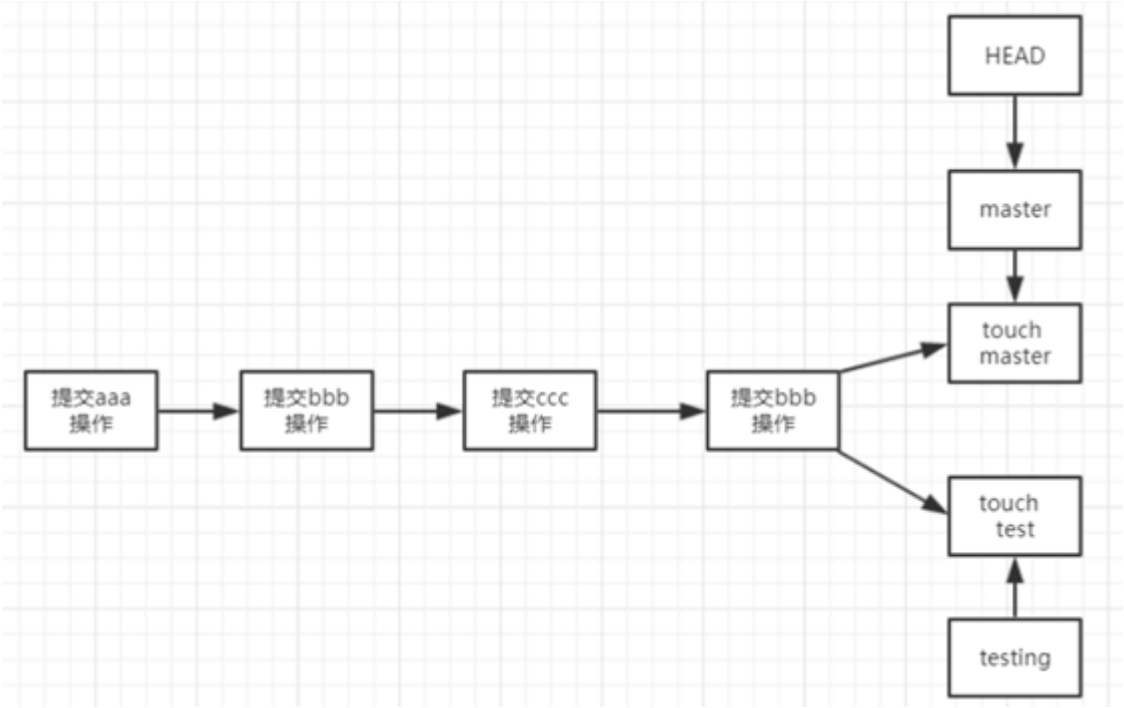


[root@git ~/git_data]# git checkout master
Switched to branch 'master'
[root@git ~/git_data]# git branch
* master
testing
[root@git ~/git_data]# ll　#正常情况下是没有 test 文件的、保证 master 分支是线上环境的
total 4
-rw-r--r-- 1 root root 16 Oct 20 18:36 a



[root@git ~/git_data]# touch master
[root@git ~/git_data]# git add .
[root@git ~/git_data]# git commit -m "commit master"

[master 8efaada] commit master
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 master
[root@git ~/git_data]# ll
total 4
-rw-r--r-- 1 root root 16 Oct 20 18:36 a
-rw-r--r-- 1 root root 0 Nov 16 12:21 master



#合并分支

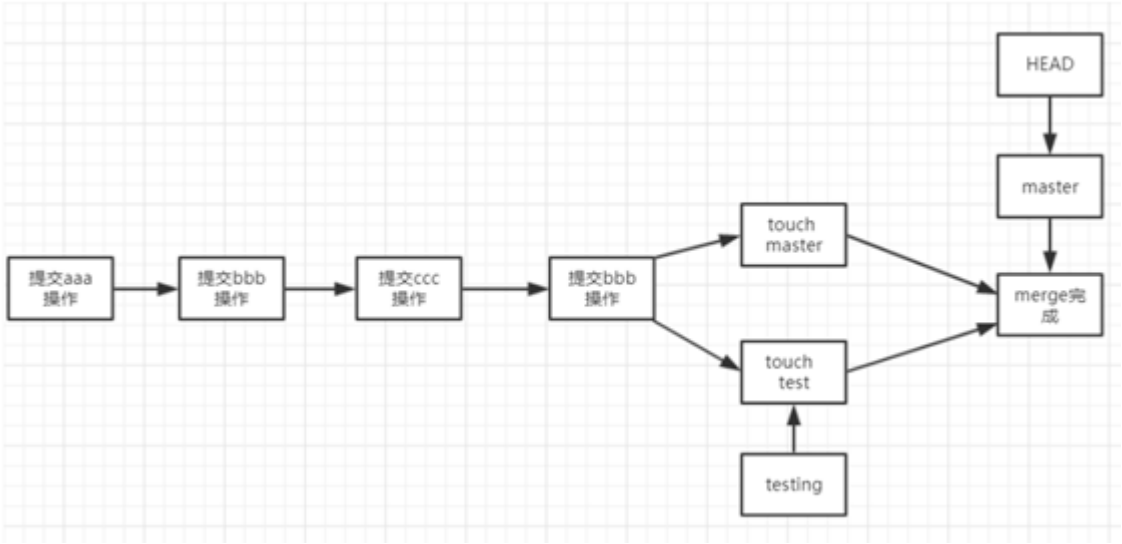
[root@git ~/git_data]# git merge testing
Merge branch 'testing'

Please enter a commit message to explain why this merge is necessary,
especially if it merges an updated upstream into a topic branch.
#
Lines starting with '#' will be ignored, and an empty message aborts
the commit.
merge testing to master　　#提示输入描述信息，相当于 git 的-m 参数

#查看日志记录

[root@git ~/git_data]# git log --oneline --decorate
0fcf3ce (HEAD, master) Merge branch 'testing'
8efaada commit master
0924a70 (testing) commit test

ef16b0b add ddd
6118c8d add bbb
18c89e4 modified a
1a85735 rename a.txt a
64e0f41 commit a.txt
c6b0ac2 new file a



#合并冲突

[root@git ~/git_data]# echo "master" >> a
[root@git ~/git_data]# git commit -am "modified a master"
[master a3d84f3] modified a master
1 file changed, 1 insertion(+)
[root@git ~/git_data]# git checkout testing
Switched to branch 'testing'
[root@git ~/git_data]# git branch
master
* testing
[root@git ~/git_data]# cat a
aaa
ccc
bbb
ddd
[root@git ~/git_data]# echo "testing" >>a
[root@git ~/git_data]# git commit -am "modified a on testing branch"
[testing 23dec52] modified a on testing branch
1 file changed, 1 insertion(+)
[root@git ~/git_data]# git checkout master
Switched to branch 'master'

```
[root@git ~/git_data]# git merge testing
Auto-merging a
CONFLICT (content): Merge conflict in a
Automatic merge failed; fix conflicts and then commit the result.
[root@git ~/git_data]# cat a     #冲突的文件自动标识到文件里，手动更改冲突要保留的代码
aaa
ccc
bbb
ddd
<<<<<<< HEAD
master
=======
testing
>>>>>>> testing
[root@git ~/git_data]# git commit -am "merge testing to master"    #提交
[master 5beb7bb] merge testing to master
[root@git ~/git_data]# git log --oneline --decorate
5beb7bb (HEAD, master) merge testing to master
23dec52 (testing) modified a on testing branch
a3d84f3 modified a master
0fcf3ce Merge branch 'testing'
8efaada commit master
0924a70 commit test
ef16b0b add ddd
6118c8d add bbb
18c89e4 modified a
1a85735 rename a.txt a
64e0f41 commit a.txt
c6b0ac2 new file a

#删除分支

[root@git ~/git_data]# git branch -d testing
Deleted branch testing (was 23dec52).
[root@git ~/git_data]# git branch
* master
```

05. Git 标签使用

标签也是指向了一次 commit 提交，是一个里程碑式的标签，回滚打标签直接加标签号，不加唯一字符串不好记

```
# -a 指定标签名字 -m 指定说明文字

[root@git ~/git_data]# git tag -a v1.0 -m "aaa bbb master tesing version v1.0"
```

查看标签

```
[root@git ~/git_data]# git tag
v1.0
```

指定某一次的提交为标签

```
[root@git ~/git_data]# git tag -a v2.0 0924a70 -m "add bbb version v2.0"
```

查看 v1.0 的信息 git show 加标签查看

```
[root@git ~/git_data]# git reset --hard v2.0
HEAD is now at 0924a70 commit test
[root@git ~/git_data]# ll
total 4
-rw-r--r-- 1 root root 16 Nov 16 15:04 a
-rw-r--r-- 1 root root 0 Nov 16 12:22 test
```

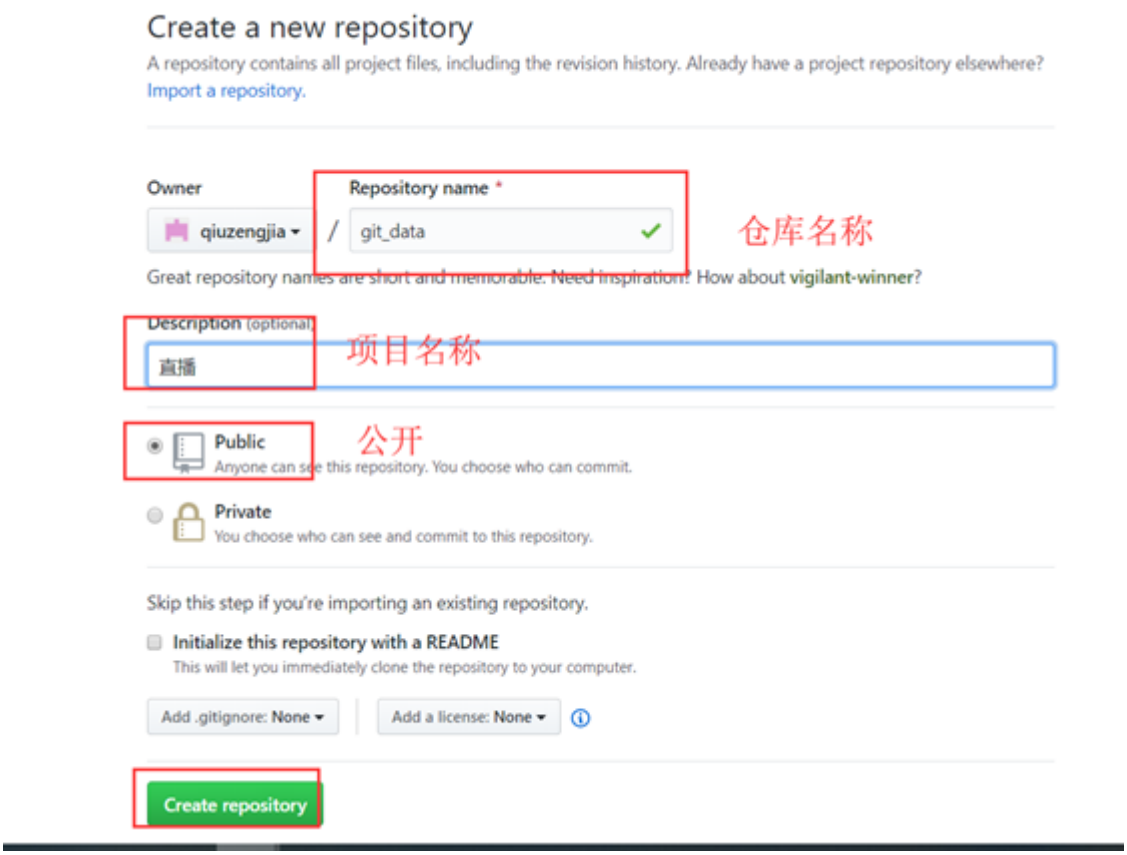
删除标签

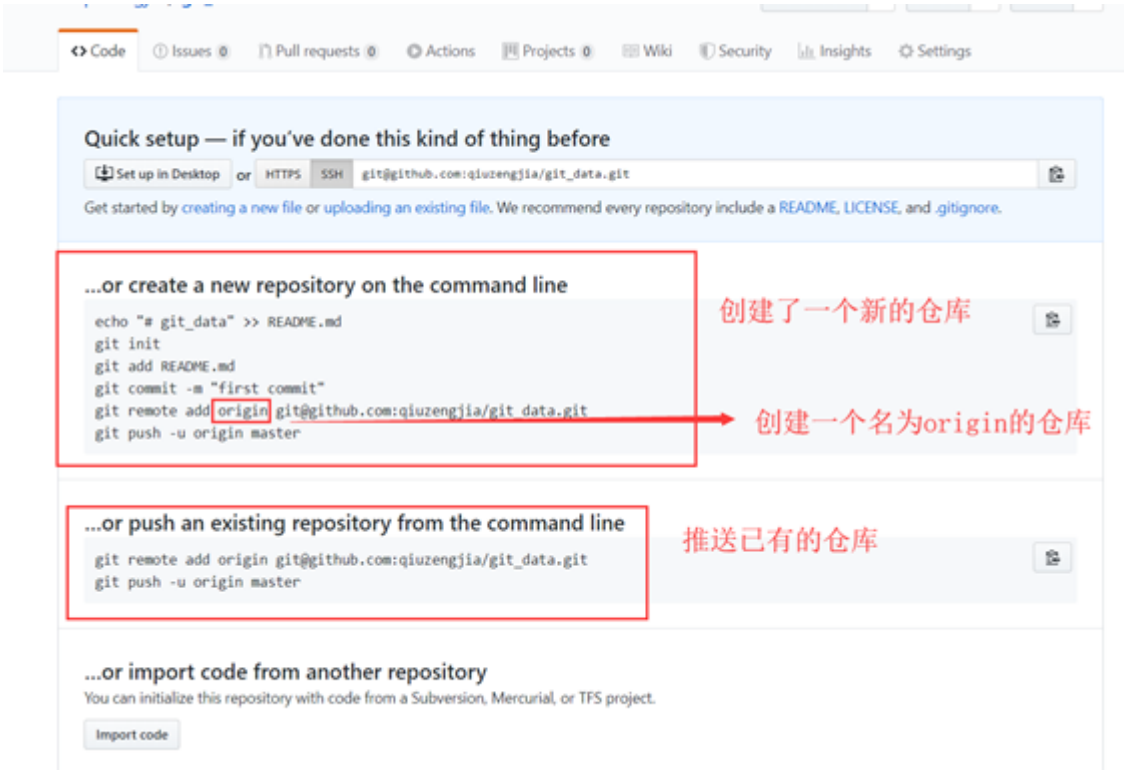
```
[root@git ~/git_data]# git tag -d v2.0
Deleted tag 'v2.0' (was b0b964c)
[root@git ~/git_data]# git tag
v1.0
```

5. Github 使用

Github 顾名思义是一个 Git 版本库的托管服务，是目前全球最大的软件仓库，拥有上百万的开发者用户，也是软件开发和寻找资源的最佳途径，Github 不仅可以托管各种 Git 版本仓库，还拥有了更美观的 Web 界面，您的代码文件可以被任何人克隆，使得开发者为开源项贡献代码变得更加容易，当然也可以付费购买私有库，这样高性价比的私有库真的是帮助到了很多团队和企业

1、注册用户 # 课前注册好用户
2、配置 ssh-key
3、创建项目
4、克隆项目到本地
5、推送新代码到 github


Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.
Owner
Repository name *
qiuzengjia
/
git_data
仓库名称
Great repository names are short and memorable. Need inspiration? How about vigilant-winner?
Description (optional)
直播
项目名称
Public
Anyone can see this repository. You choose who can commit.
公开
Private
You choose who can see and commit to this repository.
Skip this step if you're importing an existing repository.
Initialize this repository with a README
This will let you immediately clone the repository to your computer.
Add .gitignore: None
Add a license: None
Create repository


Code
Issues 0
Pull requests 0
Actions
Projects 0
Wiki
Security
Insights
Settings
Quick setup — if you've done this kind of thing before
Set up in Desktop
or
HTTPS
SSH
git@github.com:qiuzengjia/git_data.git
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.
...or create a new repository on the command line
echo "# git_data" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:qiuzengjia/git_data.git
git push -u origin master
创建了一个新的仓库
创建一个名为origin的仓库
...or push an existing repository from the command line
git remote add origin git@github.com:qiuzengjia/git_data.git
git push -u origin master
推送已有的仓库
...or import code from another repository
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.
Import code

#查看远程仓库
[root@git ~/git_data]# git remote

#添加远程仓库
[root@git ~/git_data]# git remote add origin git@github.com:qiuzengjia/git_data.git
[root@git ~/git_data]# git remote
origin

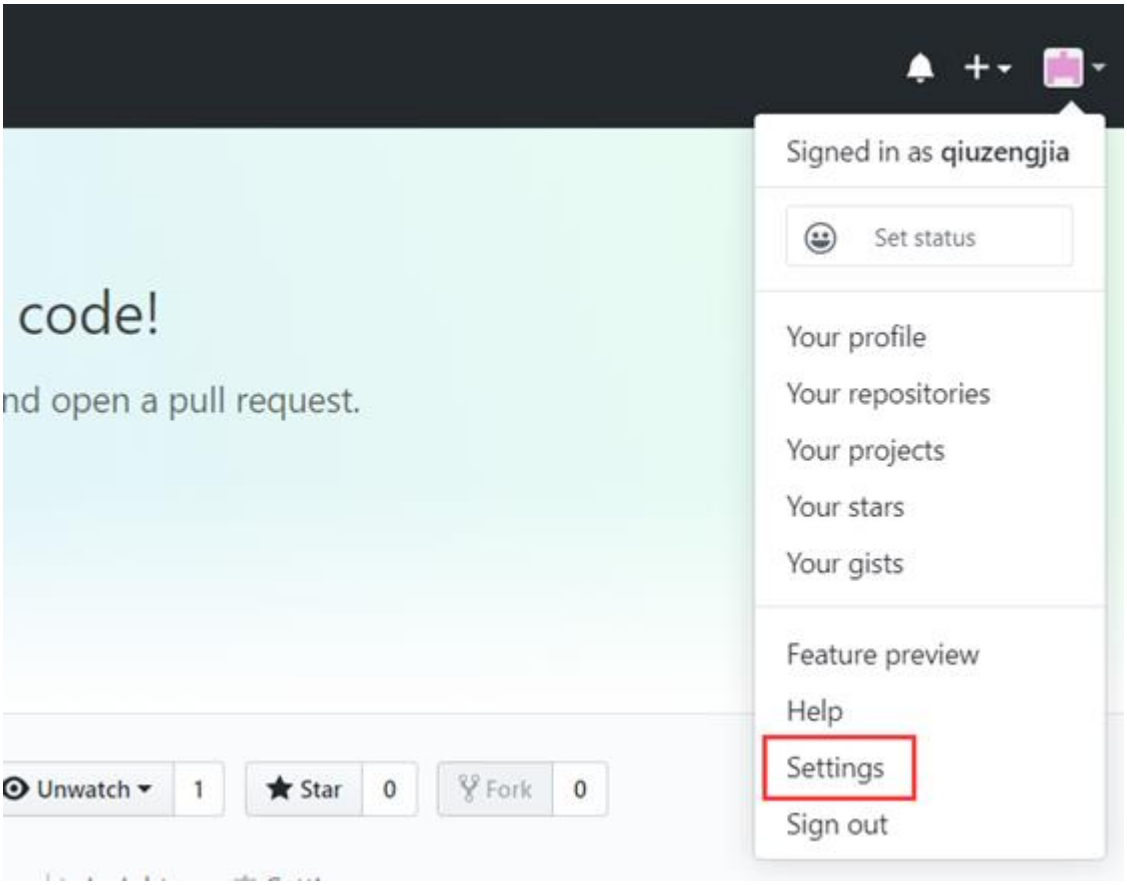
#将代码推送到远程仓库，认证失败
[root@git ~/git_data]# git push -u origin master
The authenticity of host 'github.com (52.74.223.119)' can't be established.
RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.
RSA key fingerprint is MD5:16:27:ac:a5:76:28:2d:36:63:1b:56:4d:eb:df:a6:48.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'github.com,52.74.223.119' (RSA) to the list of known hosts.
Permission denied (publickey).
fatal: Could not read from remote repository.

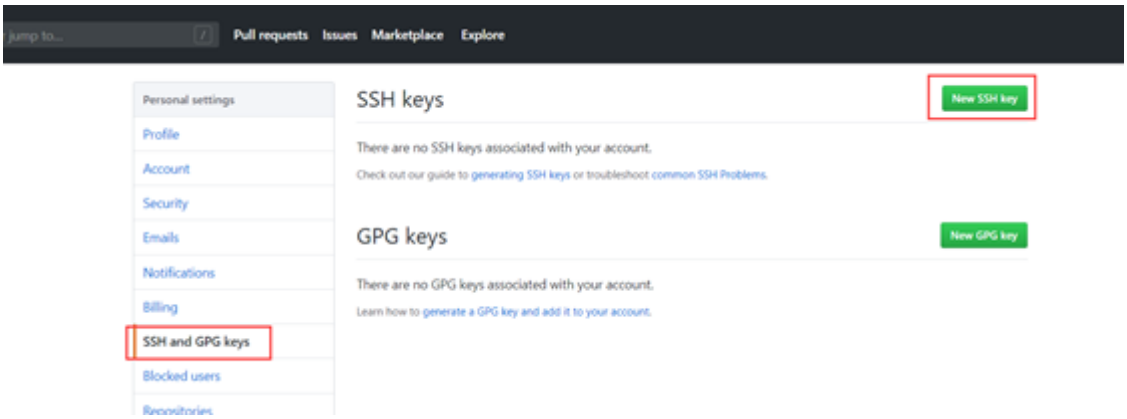
Please make sure you have the correct access rights
and the repository exists.

#生成 SSH 密钥对

[root@git ~/git_data]# cd
[root@git ~]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:OTcZp1vPfXHDbMLB+3D/wW6/YSScpbC/JjMDIcqwCqo root@git
The key's randomart image is:
+---[RSA 2048]----+
| |
| . |
| ...o . |
| . . o == O |
| + . S *..X O.|
|. . o + +.oX.=|
|o . o .o*+|
|.. = .+.=|

|E *..o=|
+----[SHA256]-----+
[root@git ~]# cat .ssh/id_rsa
id_rsa id_rsa.pub
[root@git ~]# cat .ssh/id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCkLibocGGe+jyPY2wNE50DogmPJnkjISWzSZjz8I5YBJT6b
OsfA8spR/8CmGV+hmEk/ESbTbKx2yr+NeQYqoXF+8PVmKsiV51pUc+8pYYOgqzOwGKXZ6REdPFo6
TeuzTwYtY2TFEoY2IolCeupKWUqA4dVOewJ/SfmAFqWcwa3fsbwH3EezO8tMNzKFz2wOYvSdYxZtEx
6uKQYdcScQeJ6plAKl2kdqQ6ya/c+AR3B76pM5ItErxE5M2cVc35EcgEJLh71aXddJuepcXy76Rk0o/ISr
MCcq/5wEdALYDlDEa+djO2oKaAoJcUmTEBl0n9hos3ifnCxK1d5KSINxQ5F root@git #将其复制粘贴到远程仓库中

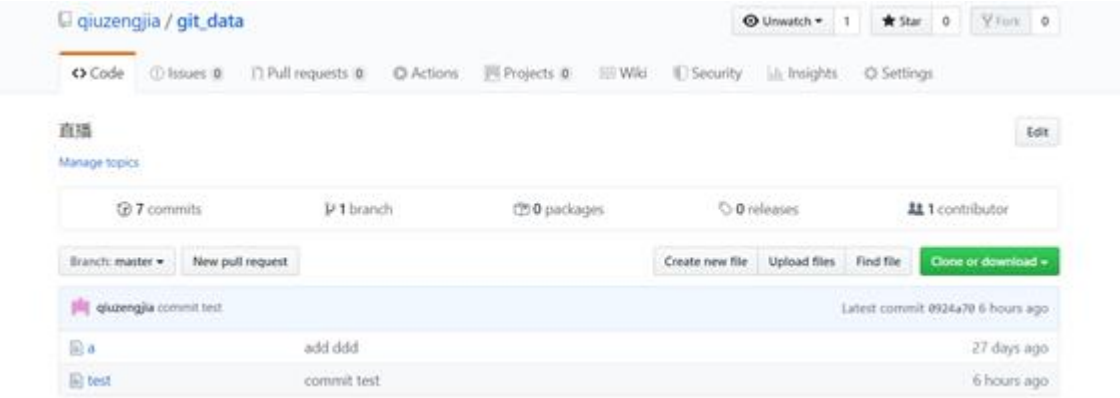




#重新进行推送

[root@git ~]# cd git_data/
[root@git ~/git_data]# git push -u origin master
Warning: Permanently added the RSA host key for IP address '13.250.177.223' to the list of known hosts.
Counting objects: 17, done.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (17/17), 1.25 KiB | 0 bytes/s, done.
Total 17 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), done.
To git@github.com:qiuzengjia/git_data.git
* [new branch] master -> master
Branch master set up to track remote branch master from origin.

#下图可以看到，代码已经推送成功了。



#克隆远程仓库到本地进行测试

[root@git ~/git_data]# cd /opt/　　#将其克隆到/opt 目录

[root@git /opt]# git clone git@github.com:qiuzengjia/git_data.git
Cloning into 'git_data'...
Warning: Permanently added the RSA host key for IP address '13.229.188.59' to the list of known hosts.
remote: Enumerating objects: 17, done.
remote: Counting objects: 100% (17/17), done.
remote: Compressing objects: 100% (7/7), done.
Receiving objects: 100% (17/17), done.
Resolving deltas: 100% (1/1), done.
remote: Total 17 (delta 1), reused 17 (delta 1), pack-reused 0

[root@git /opt]# ll git_data/
total 4
-rw-r--r-- 1 root root 16 Nov 16 18:04 a
-rw-r--r-- 1 root root 0 Nov 16 18:04 test

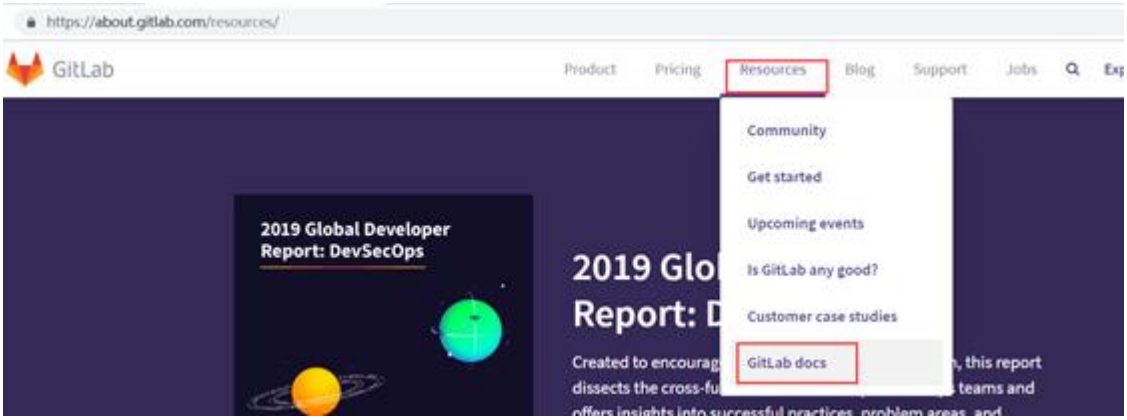
6. Gitlab 安装

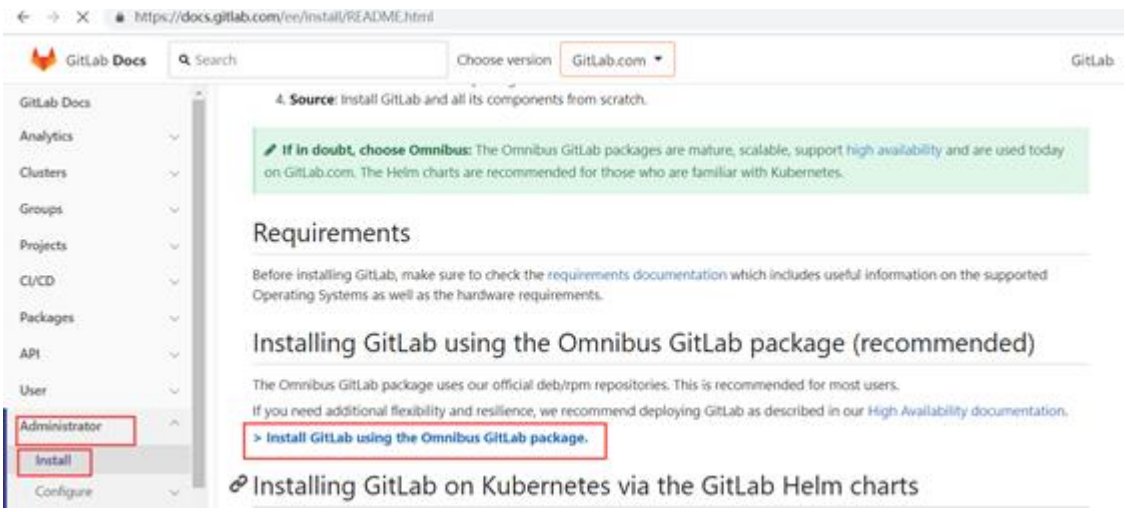
GitLab 简介
GitLab 是一个用于仓库管理系统的开源项目。使用 Git 作为代码管理工具，并在此基础上搭建起来的 web 服务。可通过 Web 界面进行访问公开的或者私人项目。它拥有与 Github 类似的功能，能够浏览源代码，管理权限和注释。可以管理团队对仓库的访问，它非常易于浏览提交过的版本并提供一个文件历史库。团队成员可以利用内置的简单聊天程序(Wall)进行交流。它还提供一个代码片段收集功能可以轻松实现代码复用。
常用的网站：
官网：https://about.gitlab.com/
国内镜像：https://mirrors.tuna.tsinghua.edu.cn/gitlab-ce/yum/
安装环境：
1、CentOS 6 或者 7
2、2G 内存（实验）生产（至少 4G）

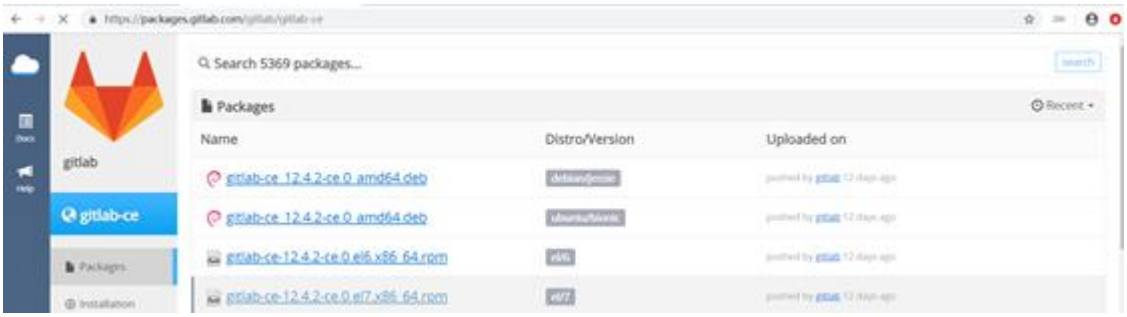
3、安装包：gitlab-ce-10.2.2-ce

4、禁用防火墙，关闭 selinux

https://about.gitlab.com/installation/ # git 官网







[root@git ~]# yum install -y curl policycoreutils-python openssh-server # 安装依赖

#上传 gitlab 安装包
下载方式可通过国内清华源 gitlab-ce 社区版本下载

[root@git ~]# ll
drwxr-xr-x 3 root root 39 Nov 16 15:04 git_data
-rw-r--r-- 1 root root 389758391 Aug 22 2018 gitlab-ce-10.2.2-ce.0.el7.x86_64.rpm

#rpm 安装，发现缺少依赖

[root@git ~]# rpm -ivh gitlab-ce-10.2.2-ce.0.el7.x86_64.rpm
warning: gitlab-ce-10.2.2-ce.0.el7.x86_64.rpm: Header V4 RSA/SHA1 Signature, key ID f27eab47: NOKEY
error: Failed dependencies:
    policycoreutils-python is needed by gitlab-ce-10.2.2-ce.0.el7.x86_64

#安装依赖之后再次安装

[root@git ~]# yum install -y policycoreutils-python

[root@git ~]# vim /etc/gitlab/gitlab.rb # gitlab 配置文件

更改 url 地址为本机 IP 地址 external_url 'http://10.0.0.100'

更改配置文件后需重新配置

[root@git ~]# gitlab-ctl reconfigure

/opt/gitlab/ # gitlab 的程序安装目录

/var/opt/gitlab # gitlab 目录数据目录

/var/opt/gitlab/git-dfata # 存放仓库数据

gitlab-ctl status # 查看目前 gitlab 所有服务运维状态

gitlab-ctl stop # 停止 gitlab 服务

gitlab-ctl stop nginx # 单独停止某个服务

gitlab-ctl tail # 查看所有服务的日志

Gitlab 的服务构成：

nginx：　　#静态 web 服务器

gitlab-workhorse:　　#轻量级的反向代理服务器

logrotate：　　#日志文件管理工具

postgresql：　　#数据库

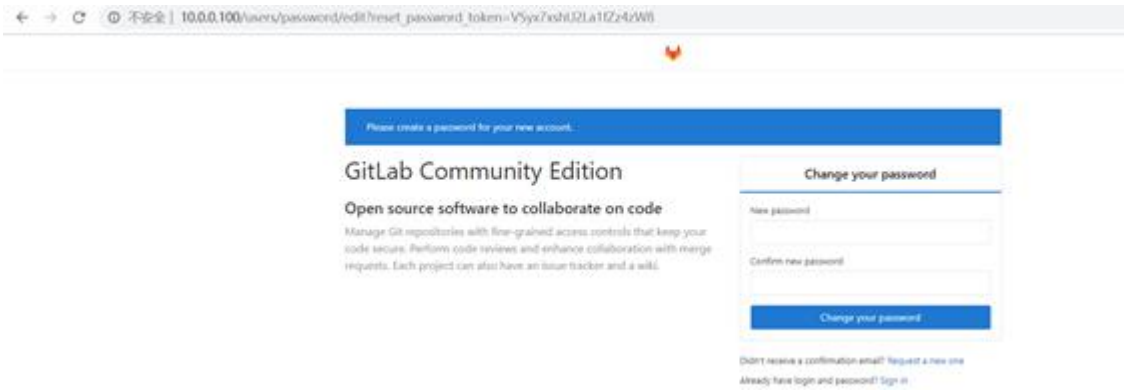
redis：　　#缓存数据库

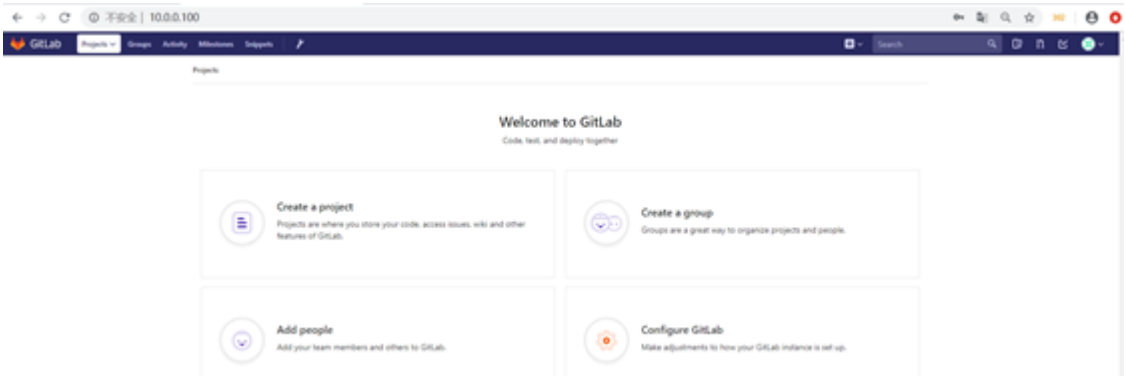
sidekiq：　　#用于在后台执行队列任务（异步执行）。（Ruby）

unicorn：An HTTP server for Rack applications，GitLab Rails 应用是托管在这个服务器上面的。(Ruby Web Server,主要使用 Ruby 编写）

#通过浏览器输入 IP 地址进行访问 gitlab

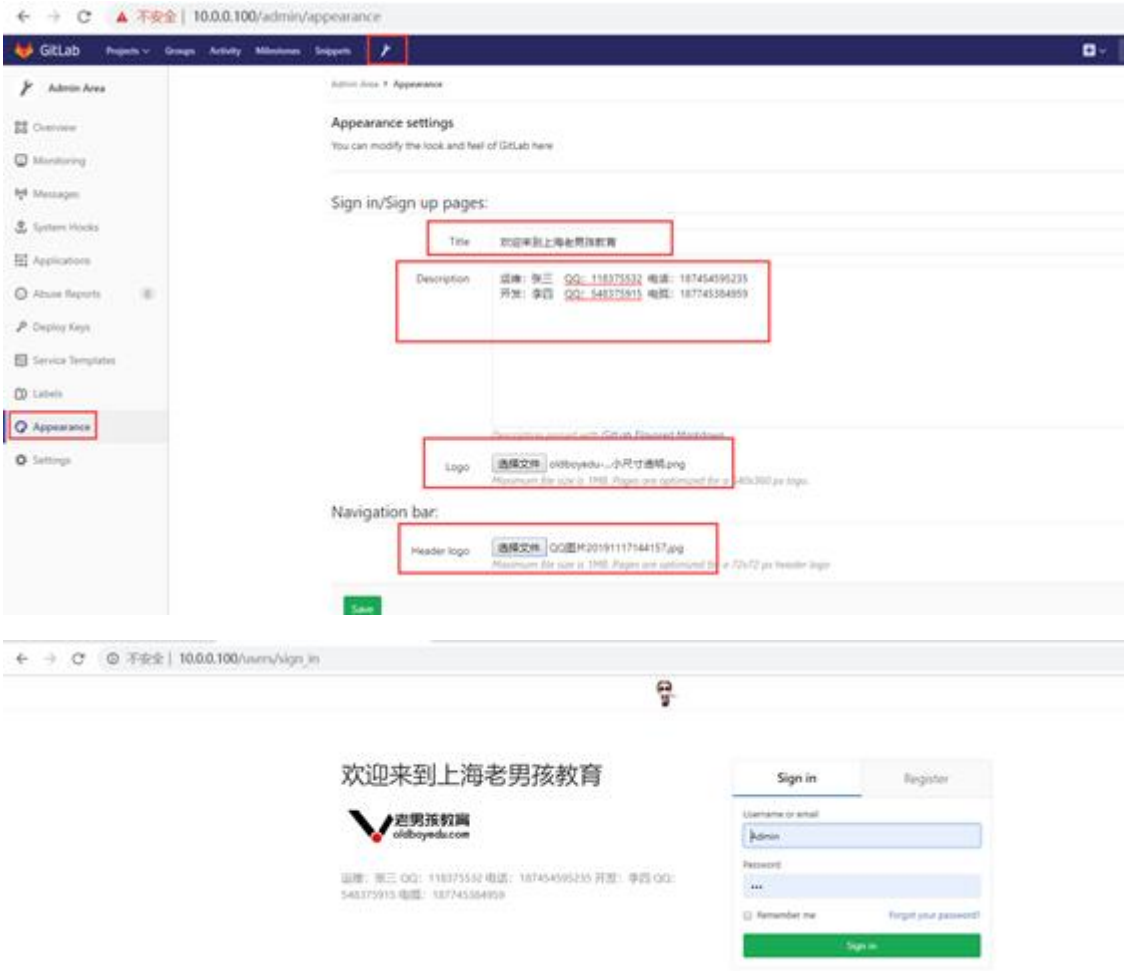
使用默认用户登录：　root
密码：　　12345678





7. Gitlab 使用

01. 外观配置



02. Gitlab 汉化配置

1、下载汉化补丁

git clone https://gitlab.com/xhang/gitlab.git

2、查看全部分支版本

git branch -a

3、对比版本、生成补丁包

git diff remotes/origin/10-2-stable remotes/origin/10-2-stable-zh > ../10.2.2-zh.diff

4、停止服务器

gitlab-ctl stop

5、打补丁

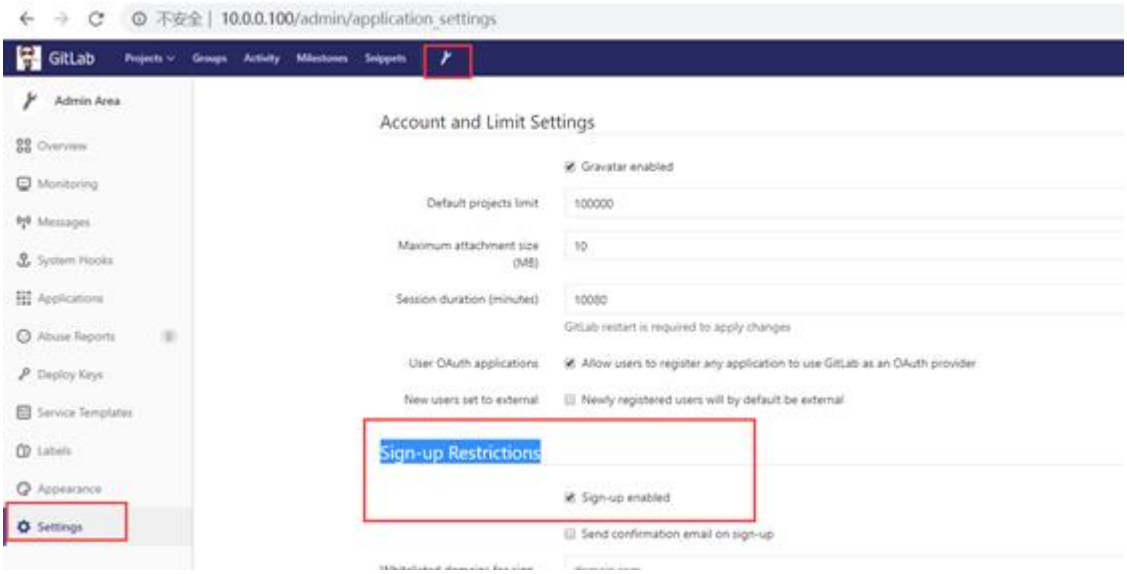
patch -d /opt/gitlab/embedded/service/gitlab-rails -p1 < /tmp/10.2.2-zh.diff

6、启动和重新配置

gitlab-ctl start

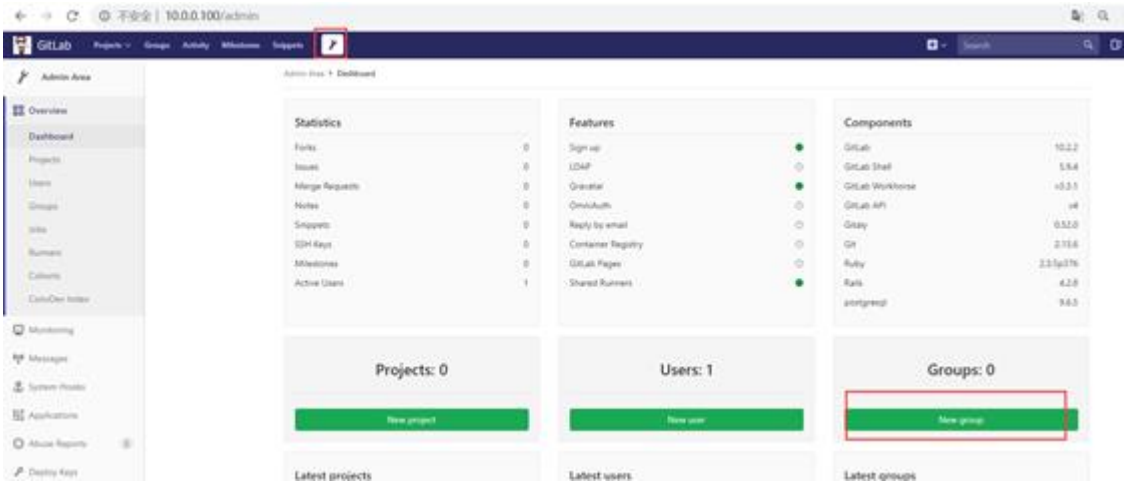
gitlab-ctl reconfigure

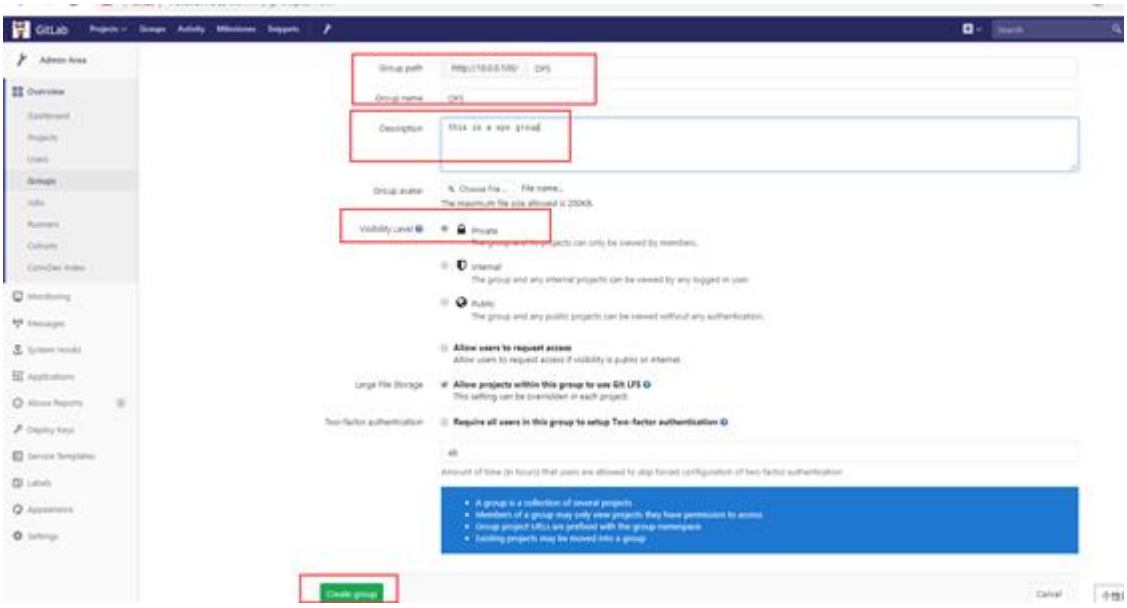
03. 注册限制



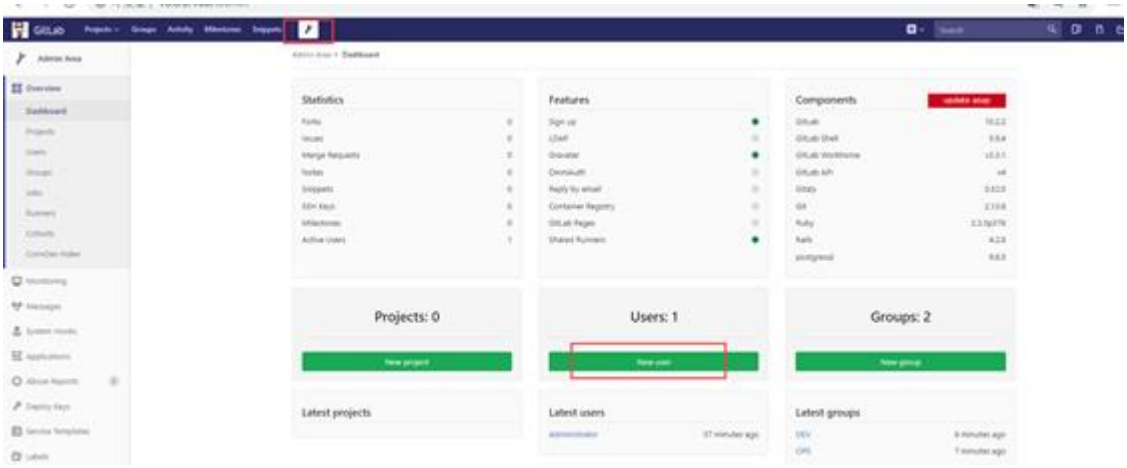
---

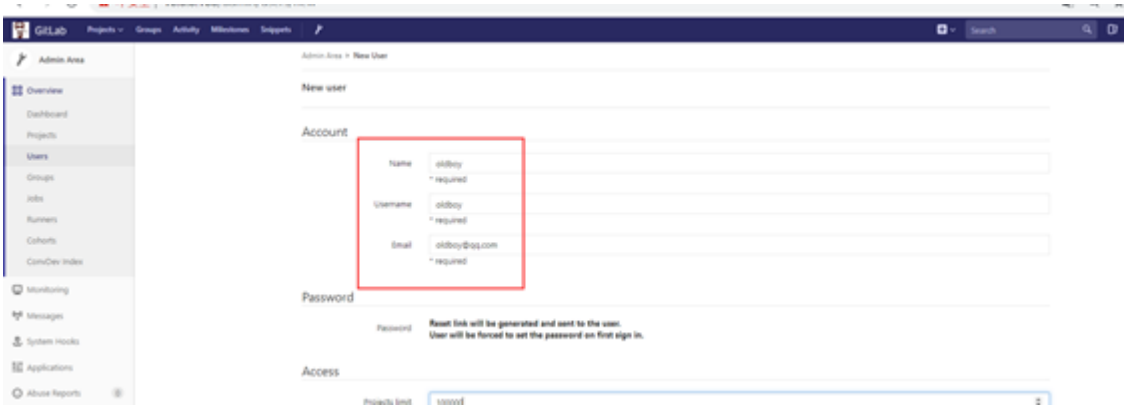
04. 创建用户及组

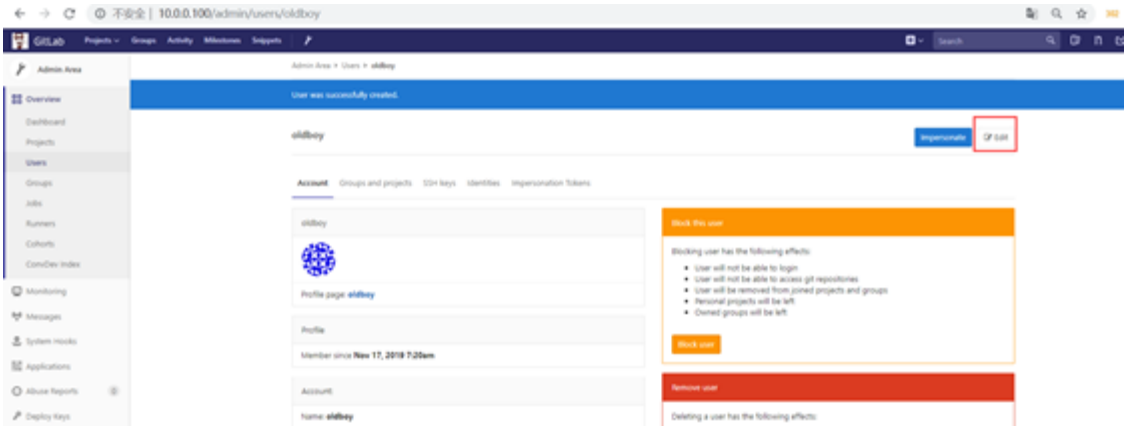


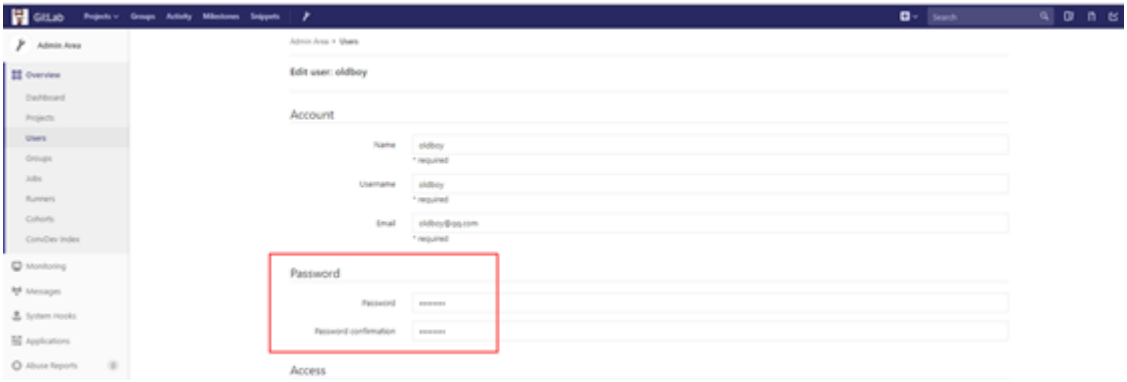


05. 创建用户

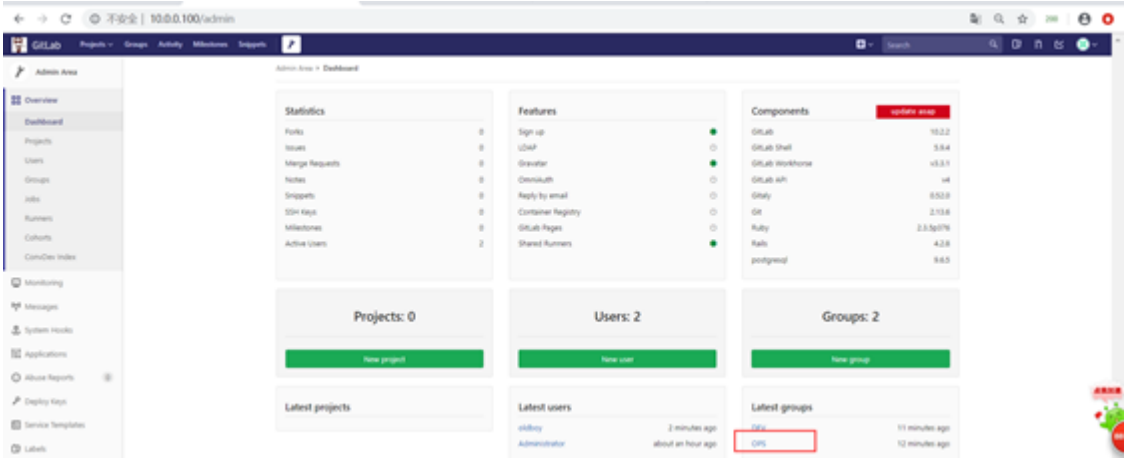

Statistics
Features
Components
Projects: 0
Users: 1
Groups: 2
Latest projects
Latest users
Latest groups

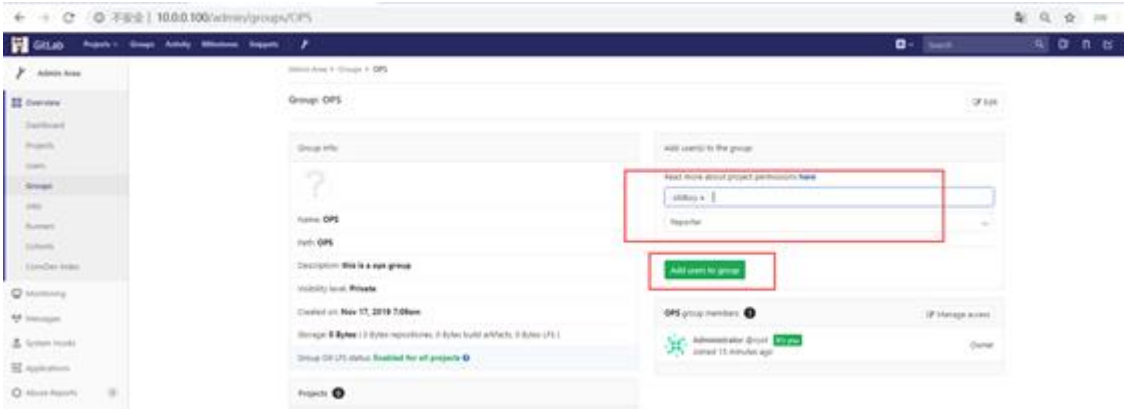

New user
Account
Password
Access



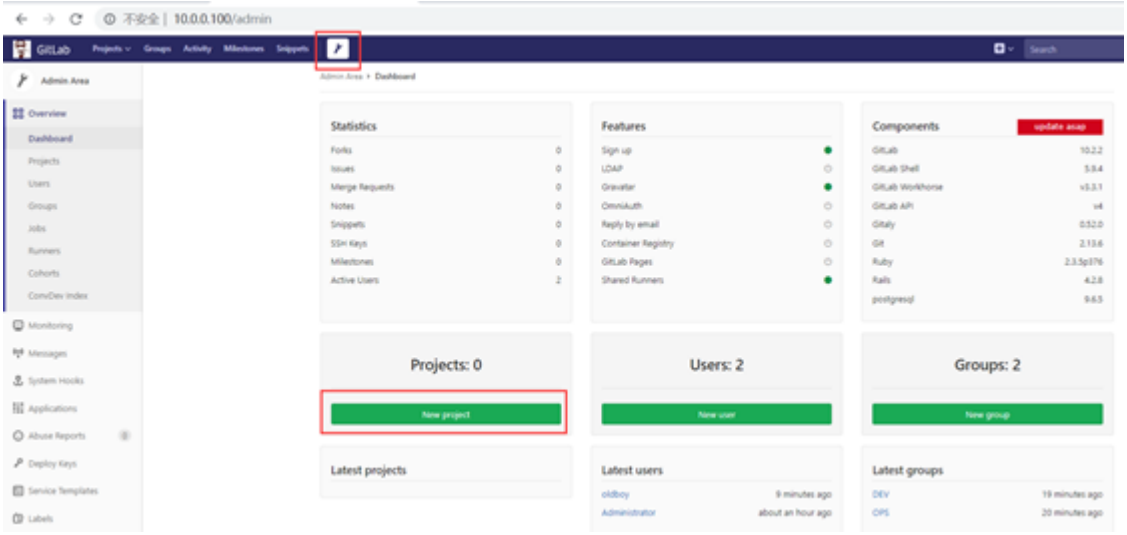


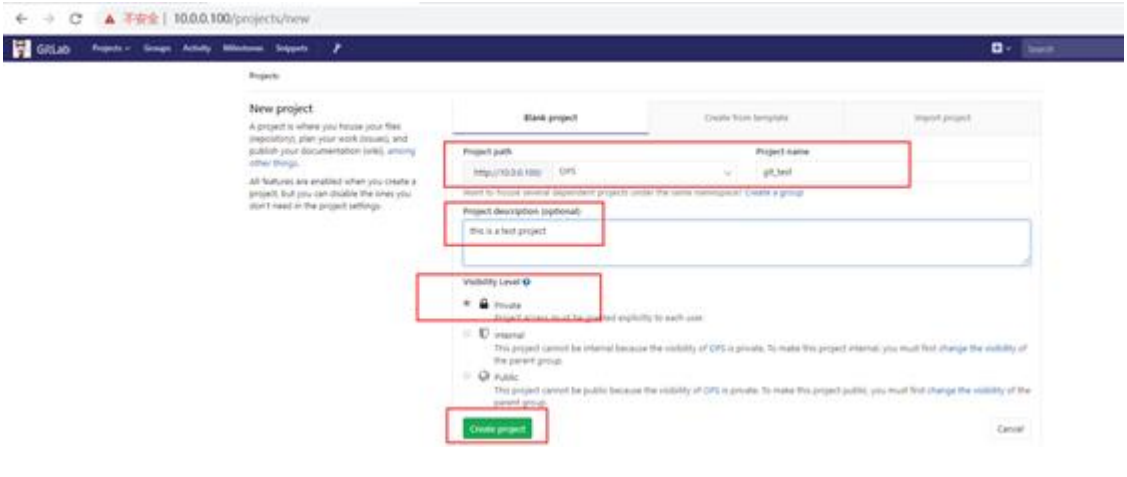
06. 把用户添加到组

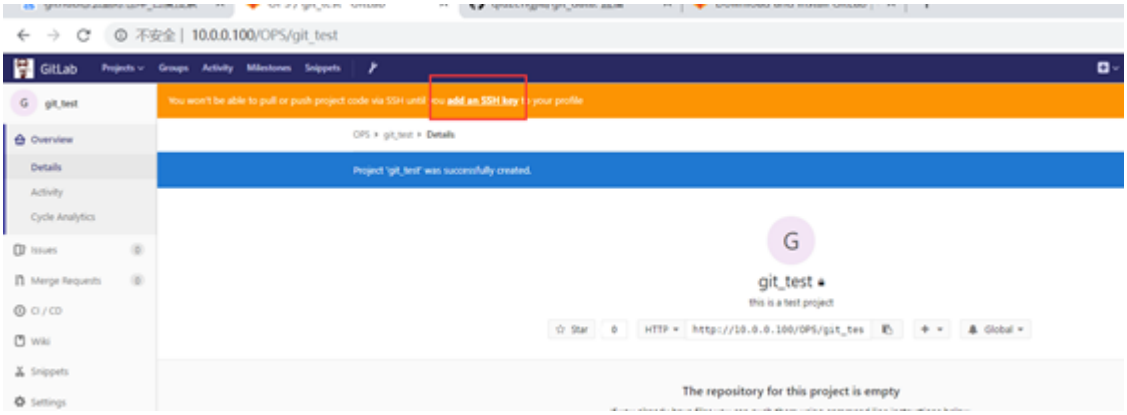


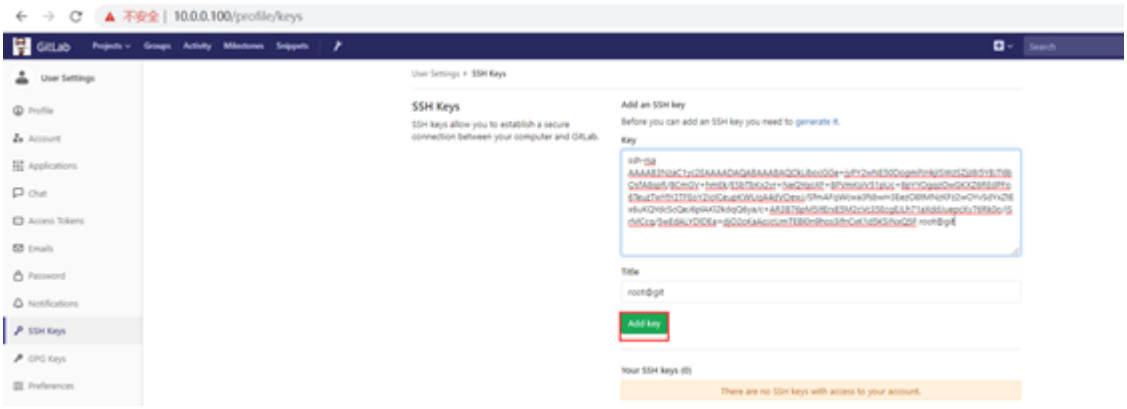


07. 创建项目


不安全 | 10.0.0.100/admin
Admin Area
Statistics
Features
Components
Projects: 0
New project
Users: 2
New user
Groups: 2
New group
Latest projects
Latest users
Latest groups


不安全 | 10.0.0.100/projects/new
New project
Blank project
Project path
Project name
Project description (optional)
this is a test project
Visibility Level
Private
Internal
Public
Create project
Cancel


不安全 | 10.0.0.100/OPS/git_test
You won't be able to pull or push project code via SSH until you add an SSH key to your profile
Project 'git_test' was successfully created.
git_test
this is a test project
The repository for this project is empty



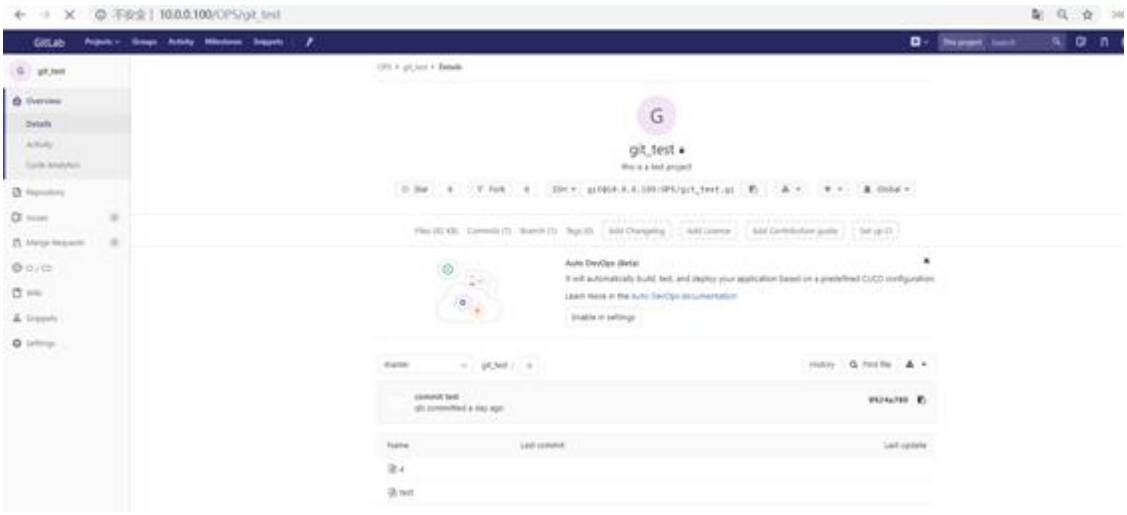
返回首页，进入项目

#删除 github 的仓库

[root@git ~/git_data]# git remote remove origin
[root@git ~/git_data]# git remote

#添加 gitlab 的远程仓库，进行代码上传

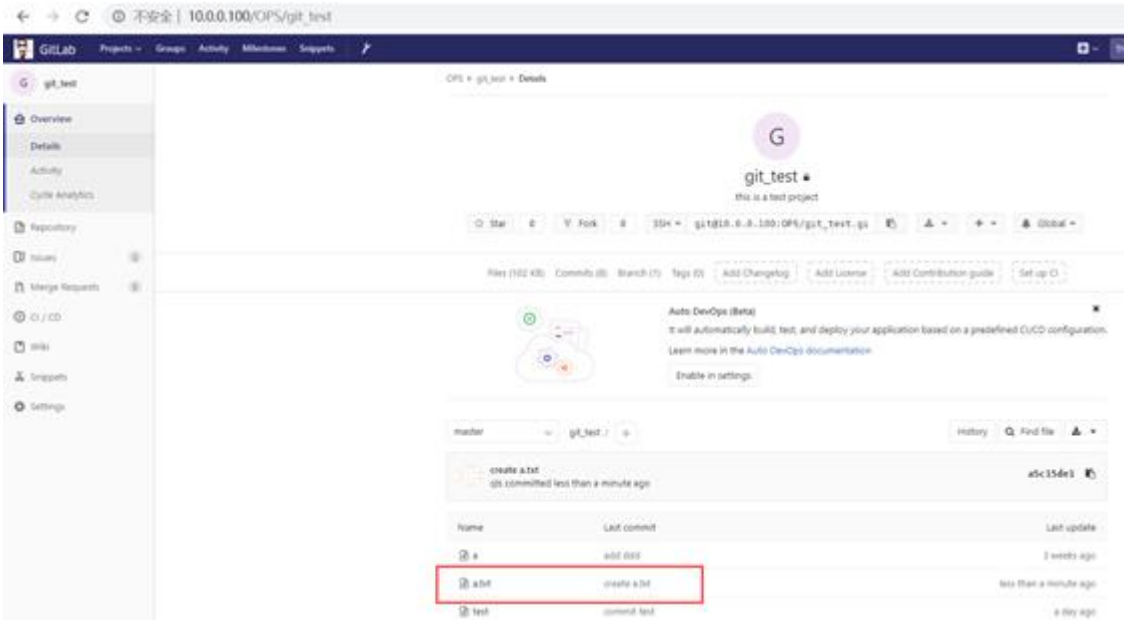
[root@git ~/git_data]# git remote add origin git@10.0.0.100:OPS/git_test.git
[root@git ~/git_data]# git push -u origin master
The authenticity of host '10.0.0.100 (10.0.0.100)' can't be established.
ECDSA key fingerprint is SHA256:6gbyCCIw3zFuNSUR2Y7UOG8fbSrj/BVUaeXwllvrGXM.
ECDSA key fingerprint is MD5:95:10:02:7c:71:73:c6:4a:b2:f9:d8:88:5d:4a:3d:e0.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.0.0.100' (ECDSA) to the list of known hosts.
Counting objects: 17, done.
Compressing objects: 100% (8/8), done.
Writing objects: 100% (17/17), 1.25 KiB | 0 bytes/s, done.
Total 17 (delta 1), reused 0 (delta 0)
To git@10.0.0.100:OPS/git_test.git
* [new branch] master -> master
Branch master set up to track remote branch master from origin.



08. 推送代码到 Gitlab

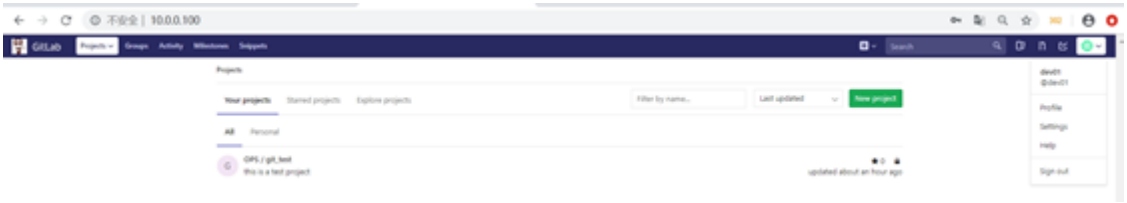
[root@git ~/git_data]# echo oldboy >>a.txt
[root@git ~/git_data]# git add .
[root@git ~/git_data]# git commit -m "create a.txt"
[master a5c15de] create a.txt
1 file changed, 1 insertion(+)
create mode 100644 a.txt
[root@git ~/git_data]# git push -u origin master
Counting objects: 4, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 289 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@10.0.0.100:OPS/git_test.git
0924a70..a5c15de master -> master
Branch master set up to track remote branch master from origin.

#刷新 Gitlab 仓库



09. 开发推送代码到 Gitlab

#给其创建 dev 用户，并给其分配到项目组，然后进行登录



#克隆一台服务器，作为开发人员使用，并进行生成密钥文件，进行密钥认证

[root@dev ~]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Created directory '/root/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:qx5BldjQ4qy+HTSH9zpg8nVQqJDNP/M8gfq/4k+E09c root@dev
The key's randomart image is:
+---[RSA 2048]----+
| +.=.o |
| o =.= . |
| +.+ o |
| .+.*o. . |
| .=.S*o.. E |

| o.+=.==. |

| . +o+..o. |

| ...++o |

| .o+.o=+. |

+----[SHA256]-----+

[root@dev ~]# cat .ssh/id_rsa.pub

ssh-rsa

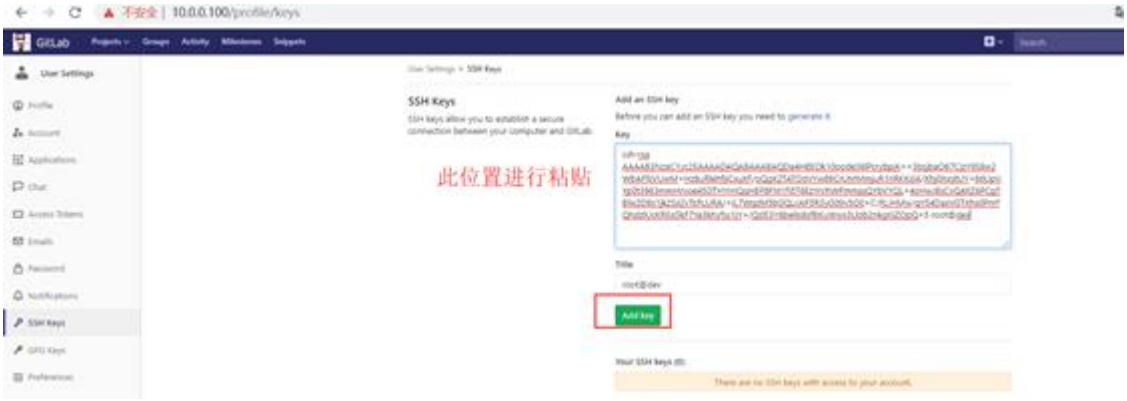
AAAAB3NzaC1yc2EAAAADAQABAAABAQDa4HBEOk10oode36lPcrybpjK++3bsjbaO67CjzY9S8w2Wb

AFfsVUwM+Hz8uRlehfjiCxuXF/oQgK2TATDdVYw89CrUMMmjuR1nRKKslA/Xhj0XxljtUY+86UpVYp0t3

963mmHrvoe45DTHYHIQqHEP8FM1fi5T6EzYrVhWFmmjqQYbVYQL+4oHwJBsCvQ4XZ6PCgTBlw2D

8s1jk2Sx2vTsPLURA/+iL7WqzM5bGQLcAP5R3yG09v5O0+C/RLIHMw/qY54DaxVGTXhs0PnrfQhdzlUc

KREsGkF7Ya3khyfss1jY+/QslS3Y6be9s8sfBKuWwx3Llob2nkgKiZOpQ+5 root@dev　　#复制其公钥



#进行将代码克隆下来，进行更改

[root@dev ~]# git clone git@10.0.0.100:OPS/git_test.git

Cloning into 'git_test'...

The authenticity of host '10.0.0.100 (10.0.0.100)' can't be established.

ECDSA key fingerprint is SHA256:6gbyCCIw3zFuNSUR2Y7UOG8fbSrj/BVUaeXwllvrGXM.

ECDSA key fingerprint is MD5:95:10:02:7c:71:73:c6:4a:b2:f9:d8:88:5d:4a:3d:e0.

Are you sure you want to continue connecting (yes/no)? yes

Warning: Permanently added '10.0.0.100' (ECDSA) to the list of known hosts.

remote: Counting objects: 20, done.

remote: Compressing objects: 100% (10/10), done.

remote: Total 20 (delta 1), reused 0 (delta 0)

Receiving objects: 100% (20/20), done.

Resolving deltas: 100% (1/1), done.

[root@dev ~]# ll

drwxr-xr-x 3 root root 52 Nov 17 17:21 git_test

[root@dev ~]# ll git_test/

total 8

-rw-r--r-- 1 root root 16 Nov 17 17:21 a

-rw-r--r-- 1 root root 7 Nov 17 17:21 a.txt
-rw-r--r-- 1 root root 0 Nov 17 17:21 test

#修改代码

[root@dev ~]# cd git_test/
[root@dev ~/git_test]# echo dev01 >> a
[root@dev ~/git_test]# git commit -am "dev01 add a"

*** Please tell me who you are.

Run

git config --global user.email "you@example.com"
git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'root@dev.(none)')

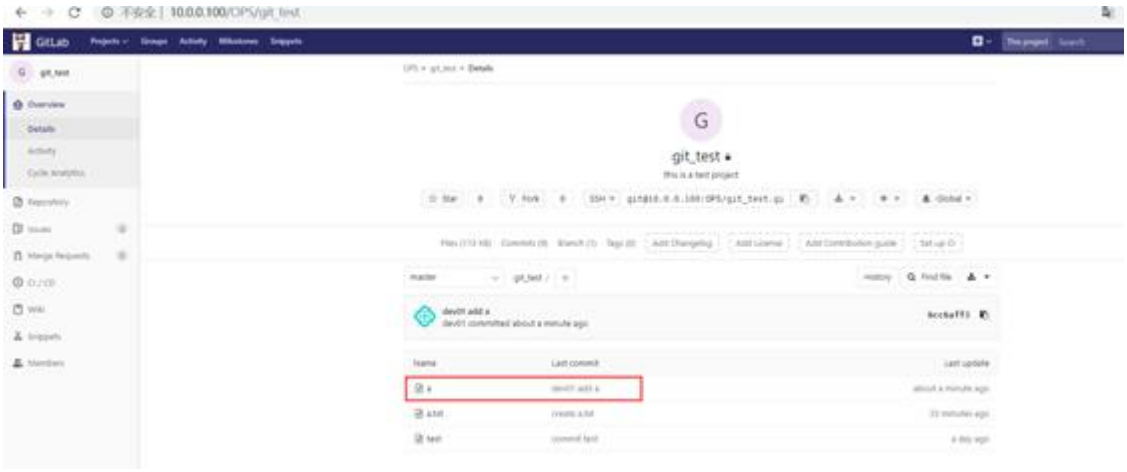
#配置邮件及用户信息

[root@dev ~/git_test]# git config --global user.email "dev@example.com"
[root@dev ~/git_test]# git config --global user.name "dev01"

[root@dev ~/git_test]# git commit -am "dev01 add a"
[master 6cc6aff] dev01 add a
1 file changed, 1 insertion(+)

#推送到远程仓库

[root@dev ~/git_test]# git push -u origin master
Counting objects: 5, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 305 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To git@10.0.0.100:OPS/git_test.git
a5c15de..6cc6aff master -> master
Branch master set up to track remote branch master from origin.

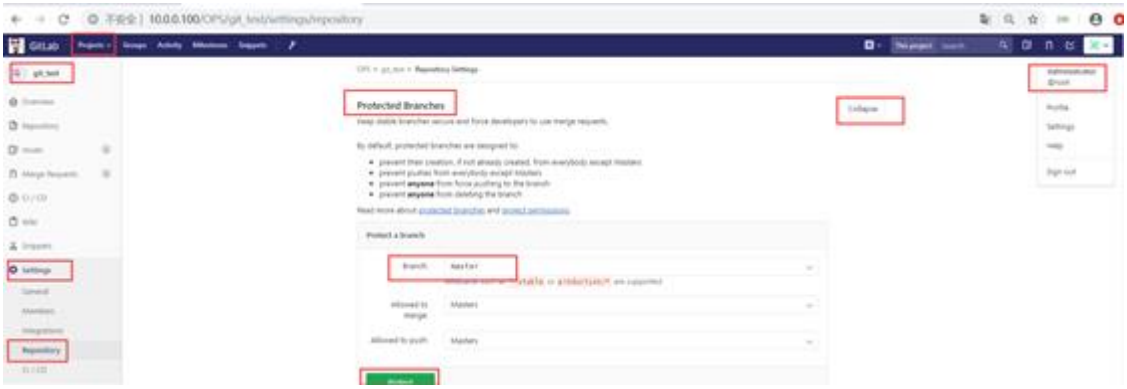


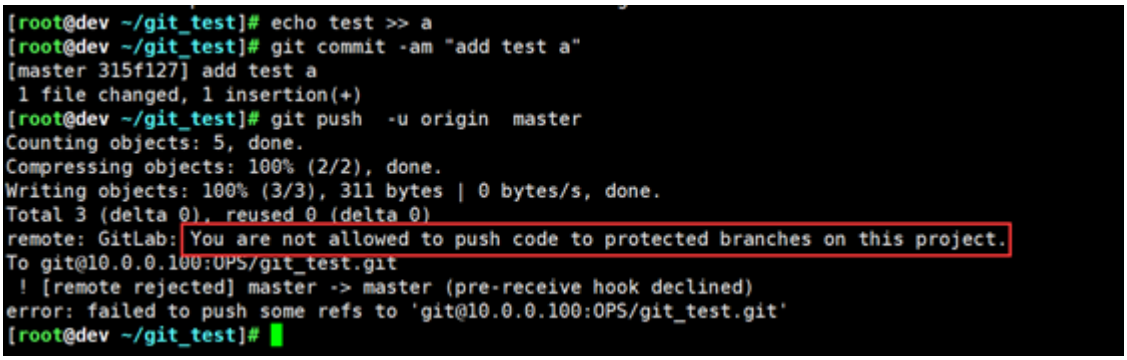
10. 分支保护

#创建一个 dev 分支

[root@dev ~/git_test]# git branch dev
[root@dev ~/git_test]# git push -u origin dev
Total 0 (delta 0), reused 0 (delta 0)
remote:
remote: To create a merge request for dev, visit:
remote:
http://10.0.0.100/OPS/git_test/merge_requests/new?merge_request%5Bsource_branch%5D=dev
remote:
To git@10.0.0.100:OPS/git_test.git
* [new branch] dev -> dev
Branch dev set up to track remote branch dev from origin.

#登录 root 用户，进行分支保护





11. 代码合并

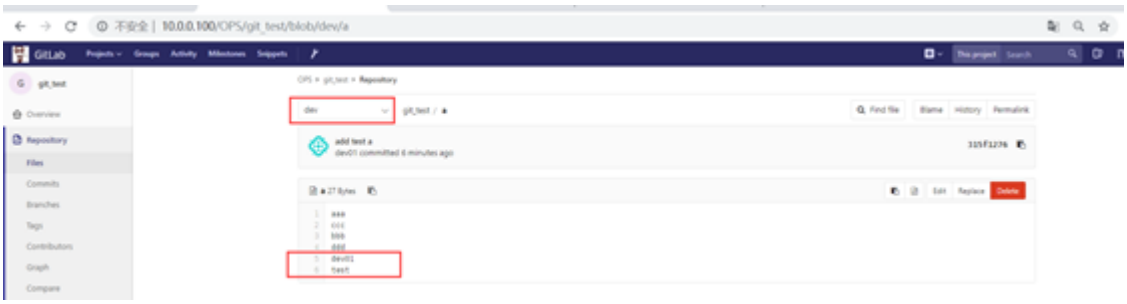
#清除旧的分支，创建新的分支，并切换

[root@dev ~/git_test]# git branch -d dev
Deleted branch dev (was 6cc6aff).
[root@dev ~/git_test]# git checkout -b dev
Switched to a new branch 'dev'
[root@dev ~/git_test]# git branch
* dev
master

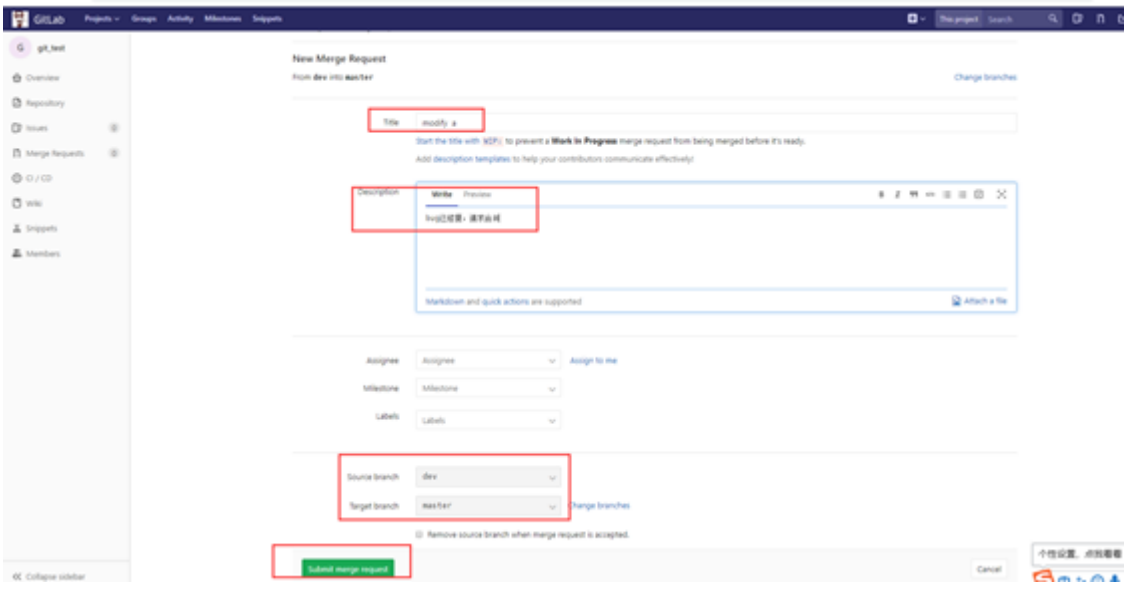
#推送到远程仓库

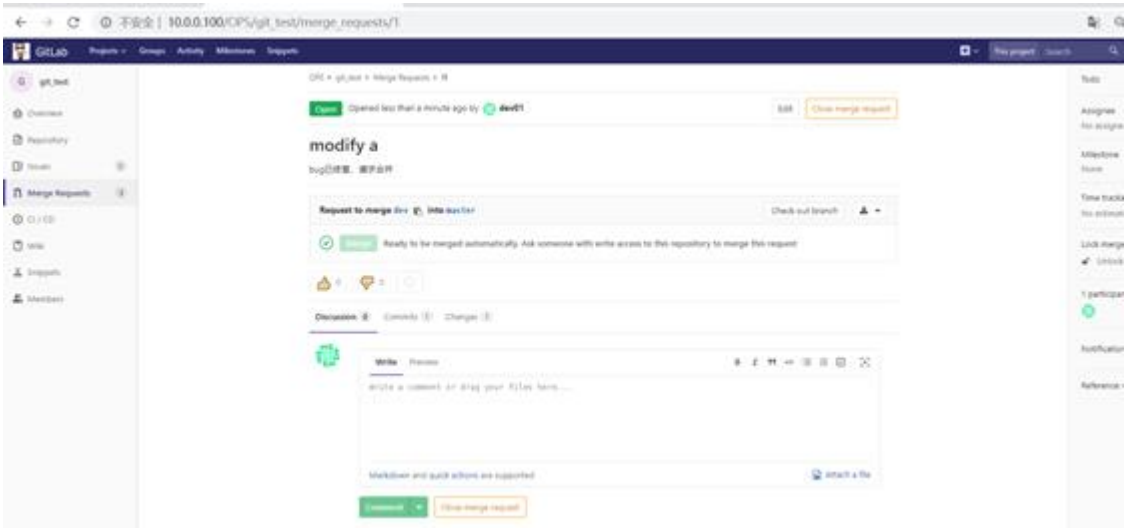
[root@dev ~/git_test]# git push -u origin dev
Counting objects: 5, done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 311 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
remote:
remote: To create a merge request for dev, visit:
remote:
http://10.0.0.100/OPS/git_test/merge_requests/new?merge_request%5Bsource_branch%5D=dev
remote:
To git@10.0.0.100:OPS/git_test.git
6cc6aff..315f127 dev -> dev
Branch dev set up to track remote branch dev from origin.

#dev 分支上面有 a 文件的最新信息，而 master 中没有

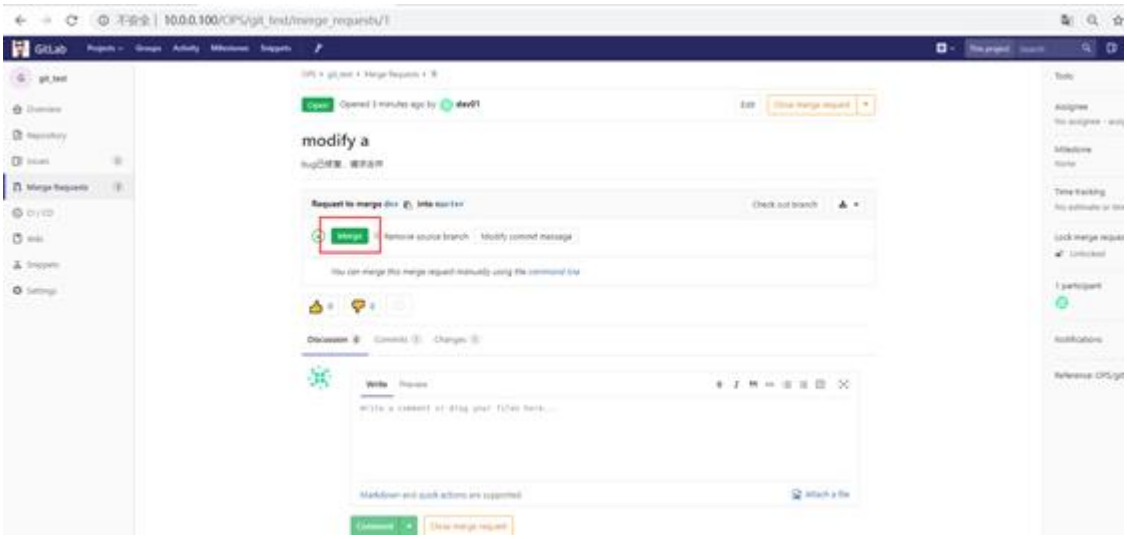


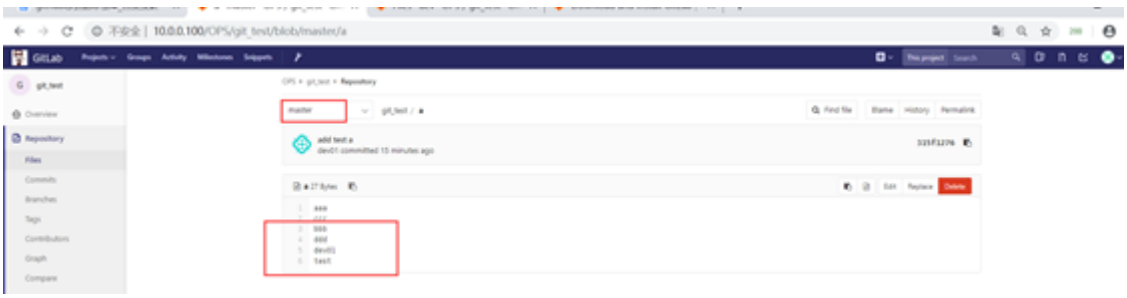
进行合并分支



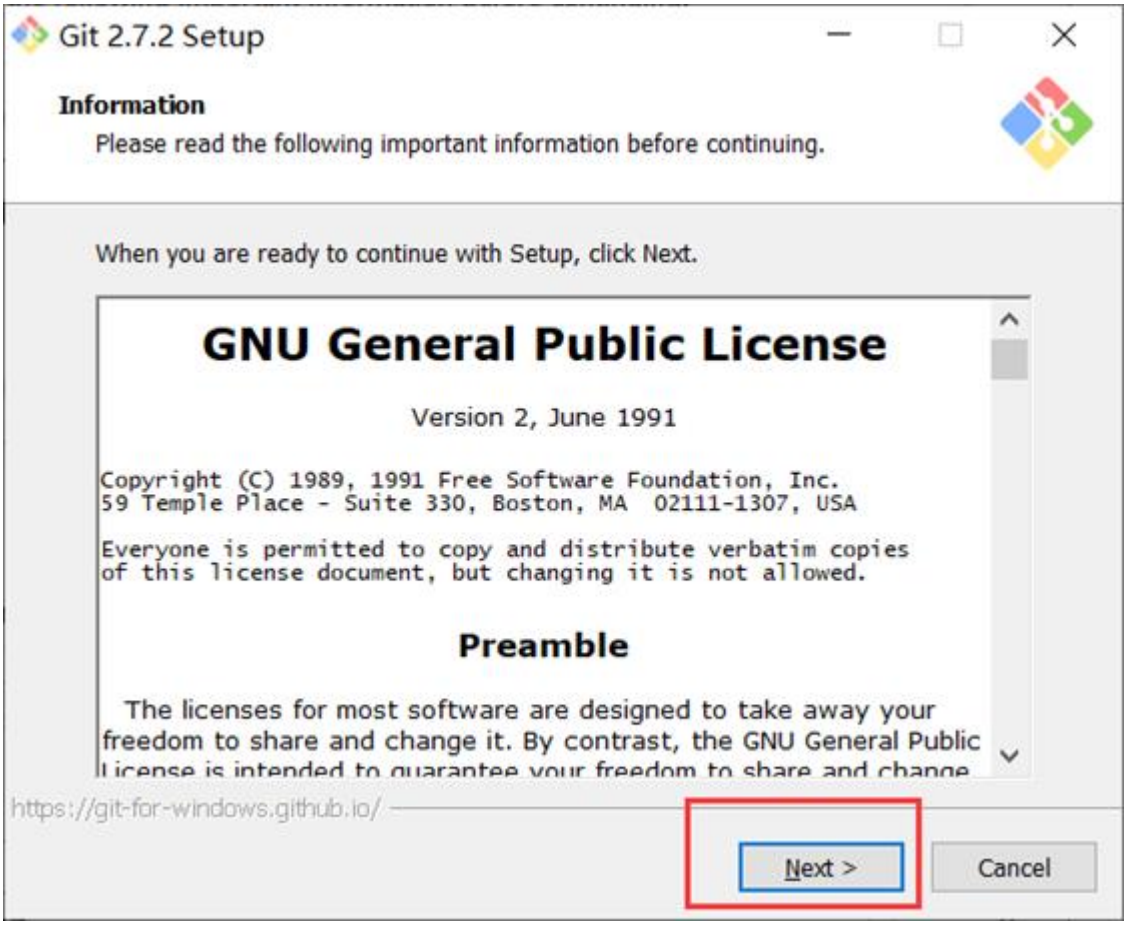


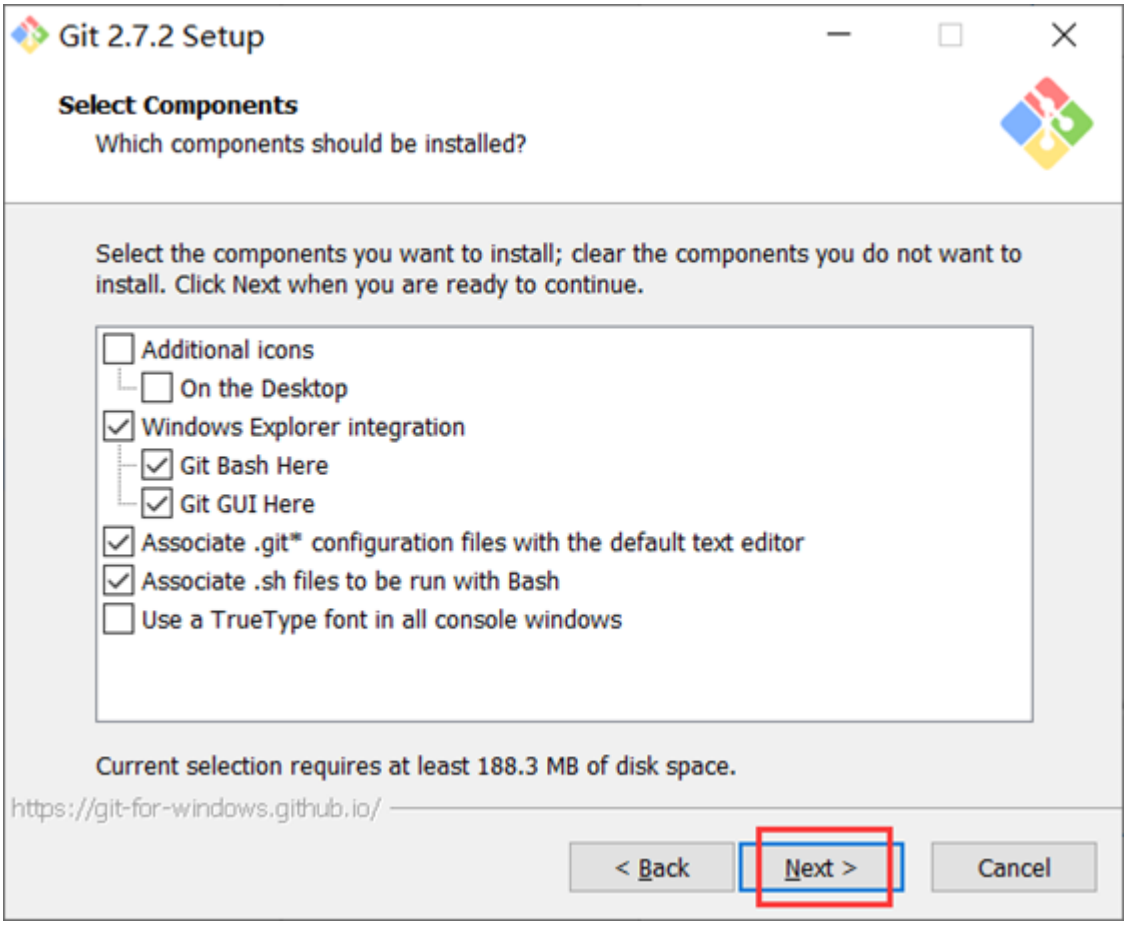
登录 root 用户进行处理请求

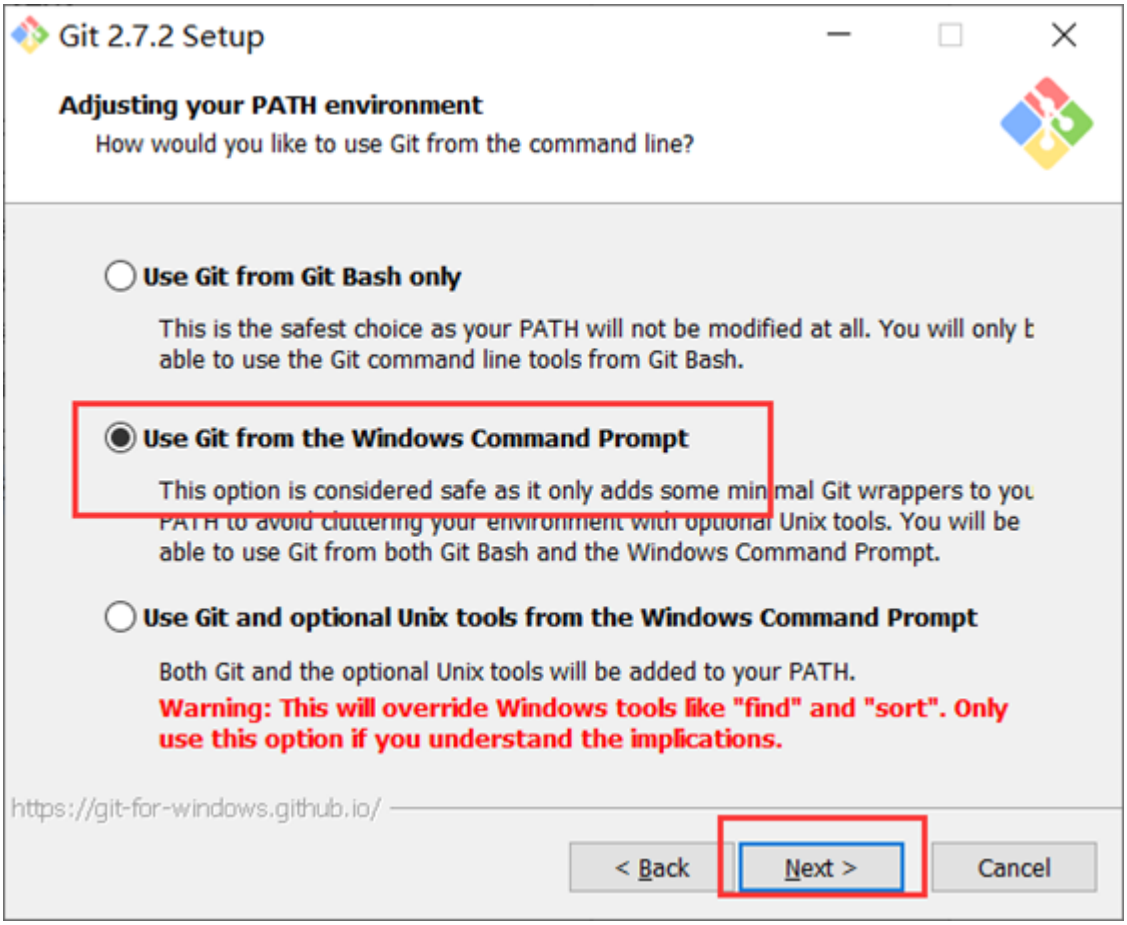


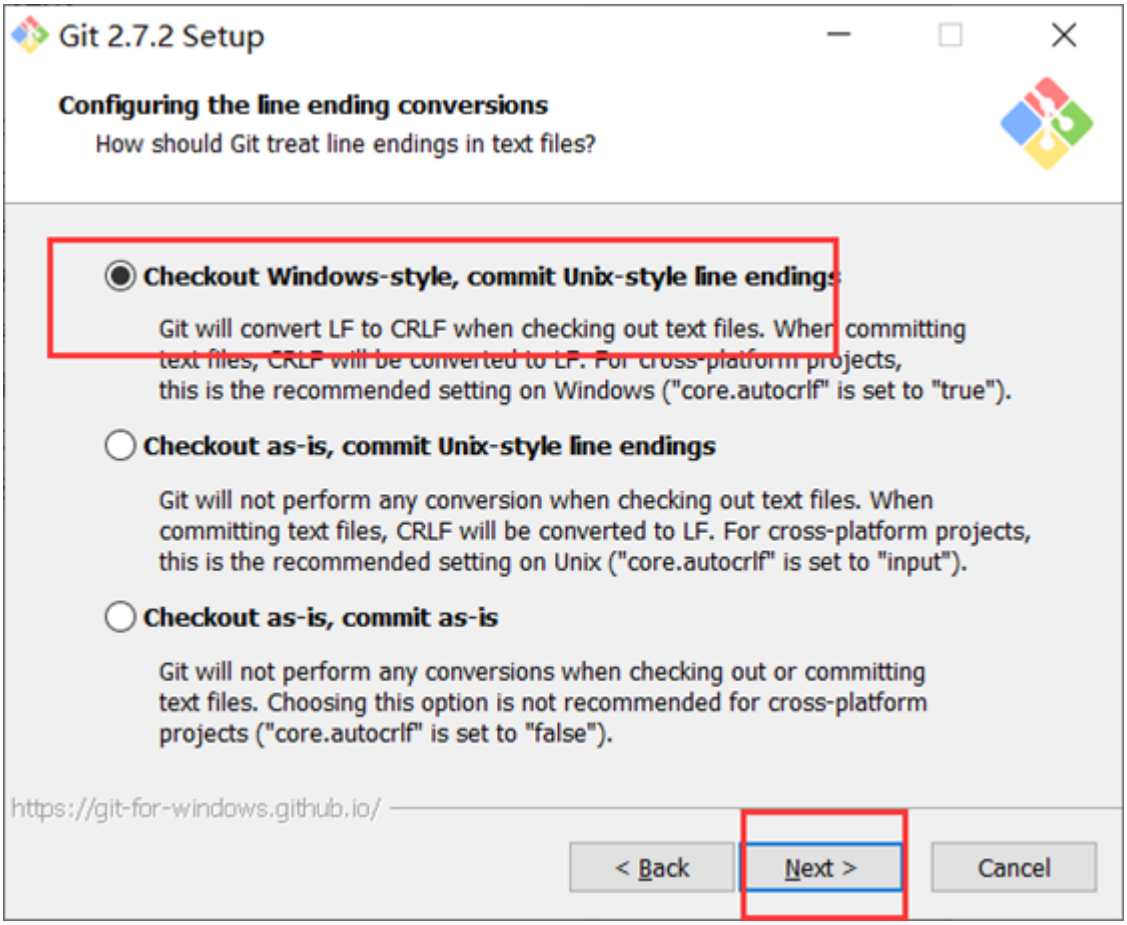


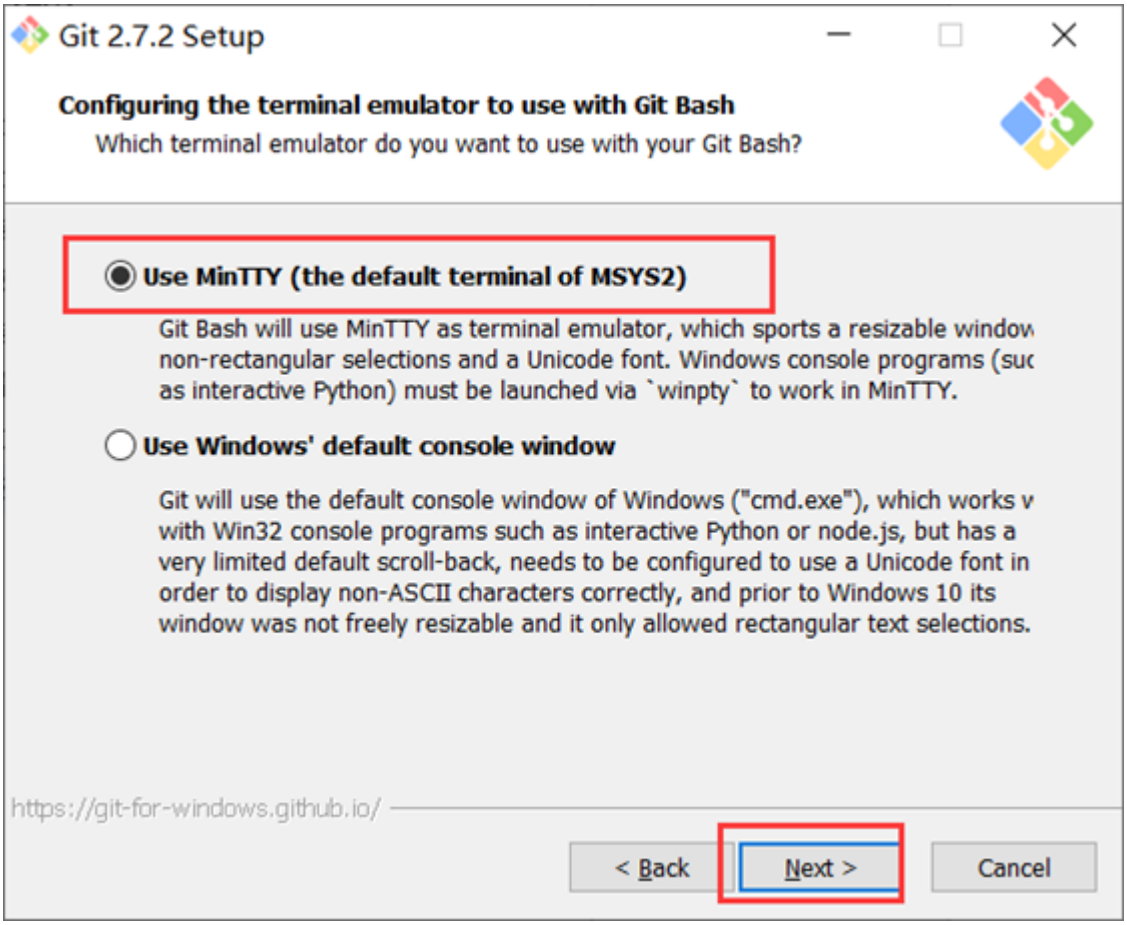
12. Git-gui 安装

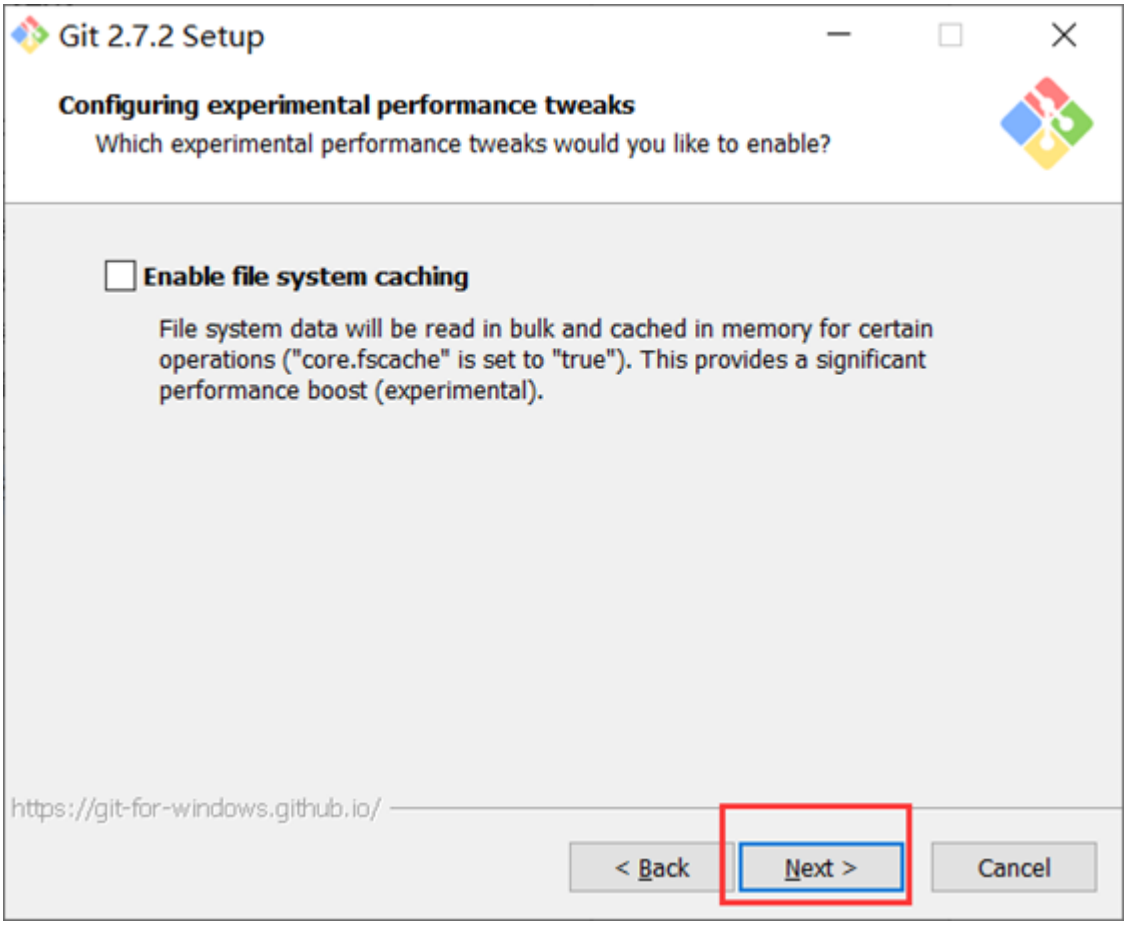

Git 2.7.2 Setup
Information
Please read the following important information before continuing.
When you are ready to continue with Setup, click Next.
GNU General Public License
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
59 Temple Place - Suite 330, Boston, MA 02111-1307, USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
Preamble
The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
https://git-for-windows.github.io/
Next >
Cancel

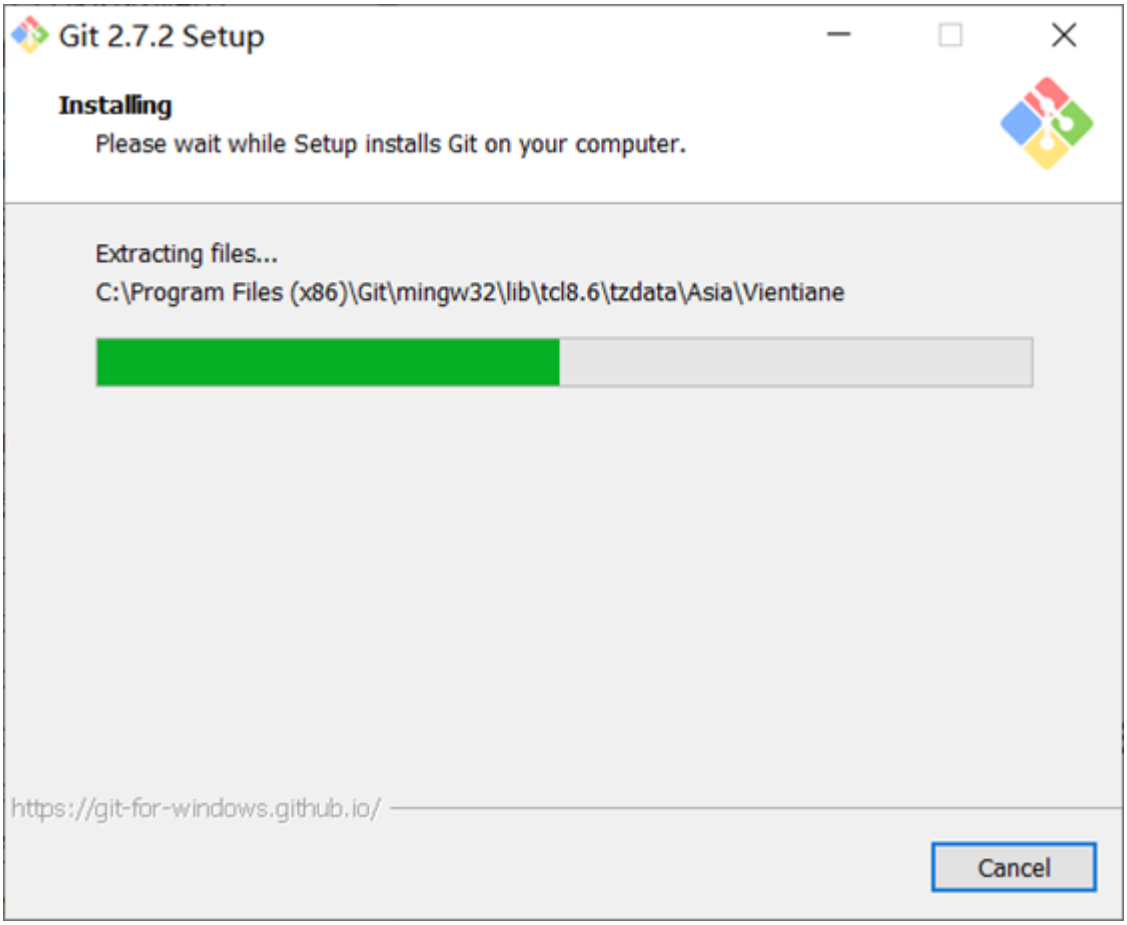

Git 2.7.2 Setup
Select Components
Which components should be installed?
Select the components you want to install; clear the components you do not want to install. Click Next when you are ready to continue.
Additional icons
On the Desktop
Windows Explorer integration
Git Bash Here
Git GUI Here
Associate .git* configuration files with the default text editor
Associate .sh files to be run with Bash
Use a TrueType font in all console windows
Current selection requires at least 188.3 MB of disk space.
https://git-for-windows.github.io/
< Back
Next >
Cancel


Git 2.7.2 Setup
Adjusting your PATH environment
How would you like to use Git from the command line?
Use Git from Git Bash only
This is the safest choice as your PATH will not be modified at all. You will only b
able to use the Git command line tools from Git Bash.
Use Git from the Windows Command Prompt
This option is considered safe as it only adds some min mal Git wrappers to you
PATH to avoid cluttering your environment with optional Unix tools. You will be
able to use Git from both Git Bash and the Windows Command Prompt.
Use Git and optional Unix tools from the Windows Command Prompt
Both Git and the optional Unix tools will be added to your PATH.
Warning: This will override Windows tools like "find" and "sort". Only
use this option if you understand the implications.
https://git-for-windows.github.io/
< Back
Next >
Cancel


Git 2.7.2 Setup
Configuring the line ending conversions
How should Git treat line endings in text files?
Checkout Windows-style, commit Unix-style line endings
Git will convert LF to CRLF when checking out text files. When committing
text files, CRLF will be converted to LF. For cross-platform projects,
this is the recommended setting on Windows ("core.autocrlf" is set to "true").
Checkout as-is, commit Unix-style line endings
Git will not perform any conversion when checking out text files. When
committing text files, CRLF will be converted to LF. For cross-platform projects,
this is the recommended setting on Unix ("core.autocrlf" is set to "input").
Checkout as-is, commit as-is
Git will not perform any conversions when checking out or committing
text files. Choosing this option is not recommended for cross-platform
projects ("core.autocrlf" is set to "false").
https://git-for-windows.github.io/
< Back
Next >
Cancel

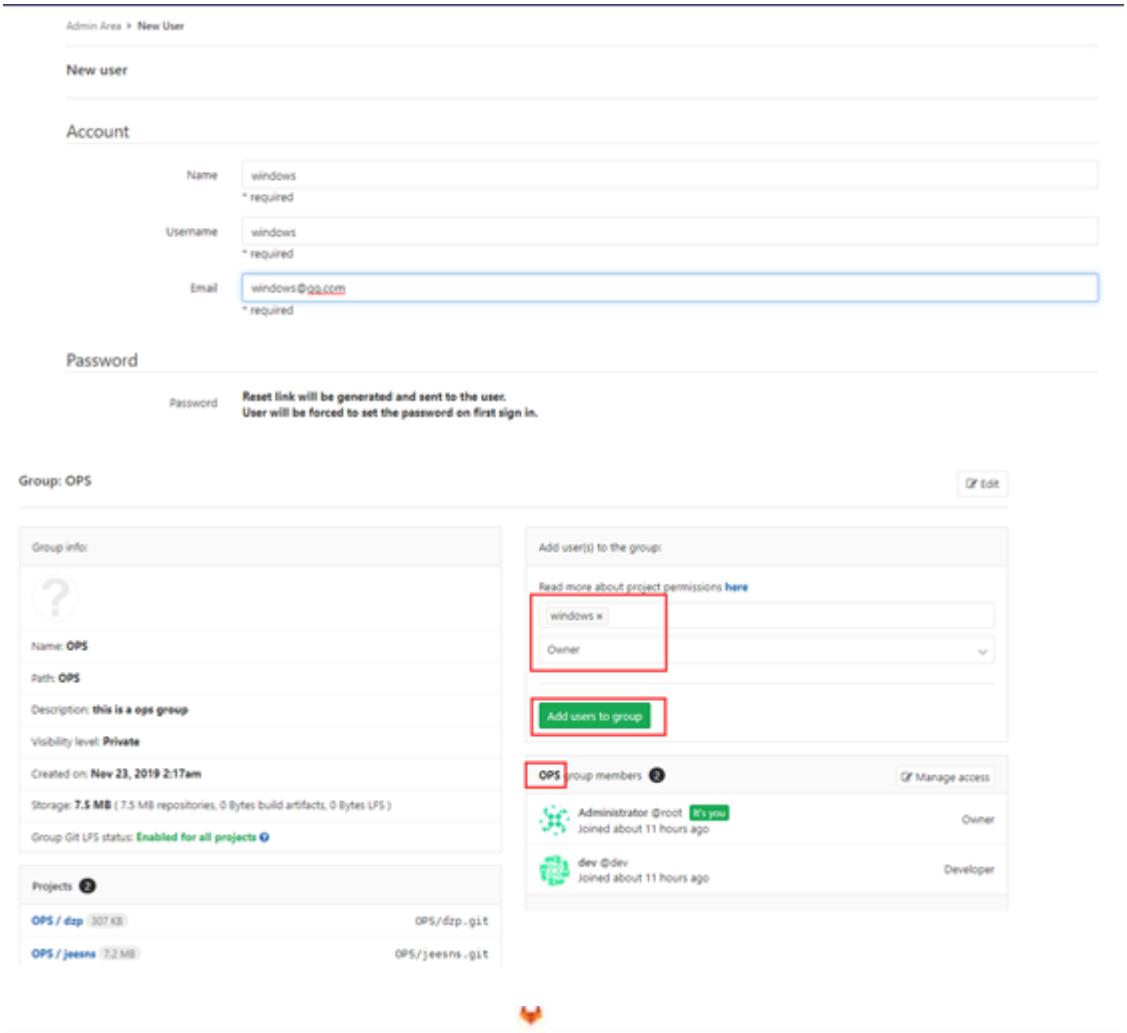

Git 2.7.2 Setup
Configuring the terminal emulator to use with Git Bash
Which terminal emulator do you want to use with your Git Bash?
Use MinTTY (the default terminal of MSYS2)
Git Bash will use MinTTY as terminal emulator, which sports a resizable windov
non-rectangular selections and a Unicode font. Windows console programs (suc
as interactive Python) must be launched via `winpty` to work in MinTTY.
Use Windows' default console window
Git will use the default console window of Windows ("cmd.exe"), which works v
with Win32 console programs such as interactive Python or node.js, but has a
very limited default scroll-back, needs to be configured to use a Unicode font in
order to display non-ASCII characters correctly, and prior to Windows 10 its
window was not freely resizable and it only allowed rectangular text selections.
https://git-for-windows.github.io/
< Back
Next >
Cancel

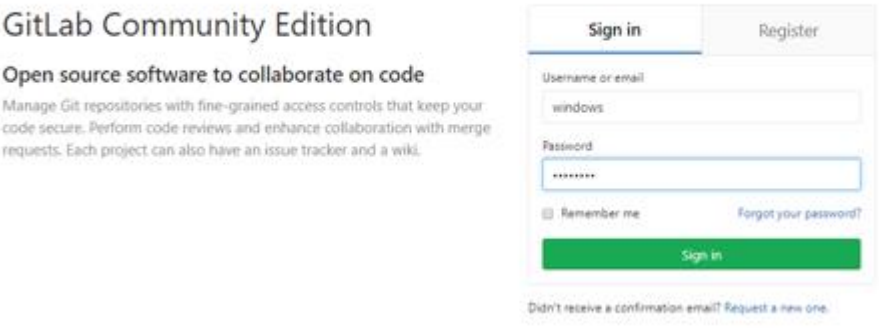

Git 2.7.2 Setup
Configuring experimental performance tweaks
Which experimental performance tweaks would you like to enable?
Enable file system caching
File system data will be read in bulk and cached in memory for certain operations ("core.fscache" is set to "true"). This provides a significant performance boost (experimental).
https://git-for-windows.github.io/
< Back
Next >
Cancel

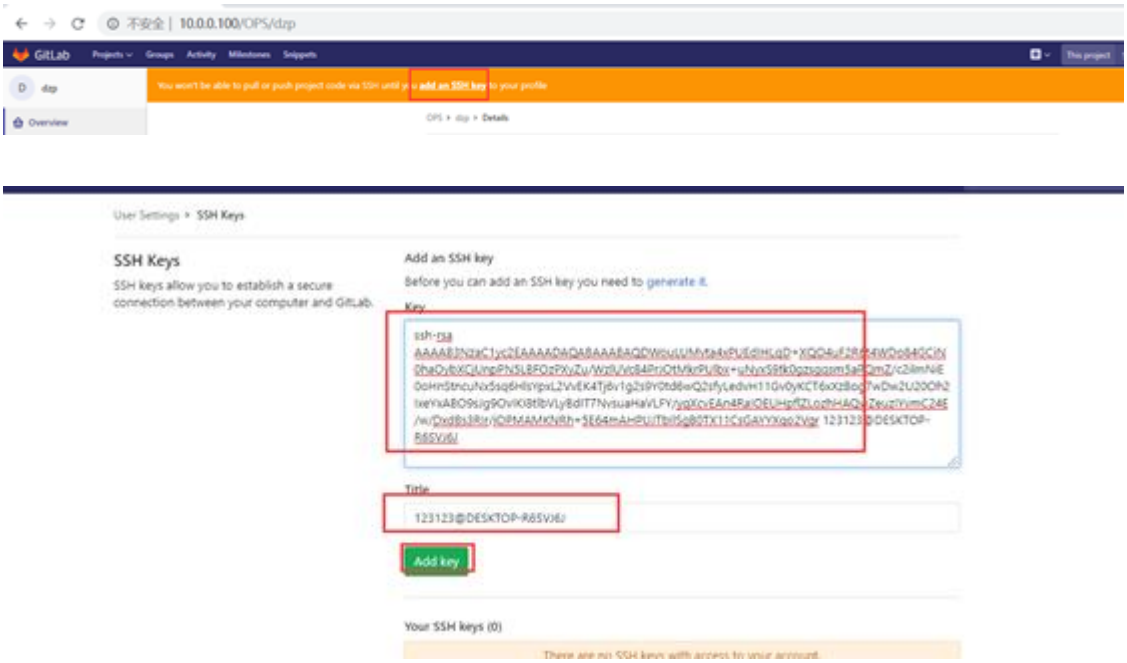

Git 2.7.2 Setup
Installing
Please wait while Setup installs Git on your computer.
Extracting files...
C:\Program Files (x86)\Git\mingw32\lib\tcl8.6\tzdata\Asia\Vientiane
https://git-for-windows.github.io/
Cancel


```
123123@DESKTOP-R6SVJ6J MINGW32 ~/Desktop
$ ssh-keygen  -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/123123/.ssh/id_rsa):
/c/Users/123123/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/123123/.ssh/id_rsa.
Your public key has been saved in /c/Users/123123/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:tYTnButxv/PO144c65oOo9mzhmNN4BSZzlrUSk07Bcc 123123@DESKTOP-R6SVJ6J
The key's randomart image is:
+---[RSA 2048]----+
|          *ooo   |
|         *.o+E   |
|        =oo*     |
|         BB o    |
|        =S.=     |
|       ...+..    |
|         .+o . . |
|         +++o.= +o|
|        .ooo+=*Oo.|
+----[SHA256]-----+

123123@DESKTOP-R6SVJ6J MINGW32 ~/Desktop
$
```

```
123123@DESKTOP-R6SVJ6J MINGW32 ~/Desktop
$ cat  /c/Users/123123/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDWouLUMvta4xPUEdIHLqD+XQO4uF2R6t4WDo84GCiN
0haOybXCjUnpPN5LBFOzPXyZu/wzlUVc84PrJOtMkrPUlbx+uNyxS9tkOgzsgqsm5aPQmZ/c2ilmNiEO
oHnStncuNx5sq6HlsYpxL2VvEK4Tj6v1g2s9YOtd6wQ2sfyLedvH11GvOyKCT6xXzBog7wDw2U2OOh2I
xeYxABO9sJg9OvIKI8tlbVLyBdIT7NvsuaHaVLFY/yqXcvEAn4RaIOEUHpflZLozhHAQwZeuzIYvmC24
E/w/DxdBs3RIr/iOPMAMKNRh+5E64mAHPUJTbIlSgBOTX11CsGAYYXqo2Vgr 123123@DESKTOP-R6SV
J6J

123123@DESKTOP-R6SVJ6J MINGW32 ~/Desktop
$
```

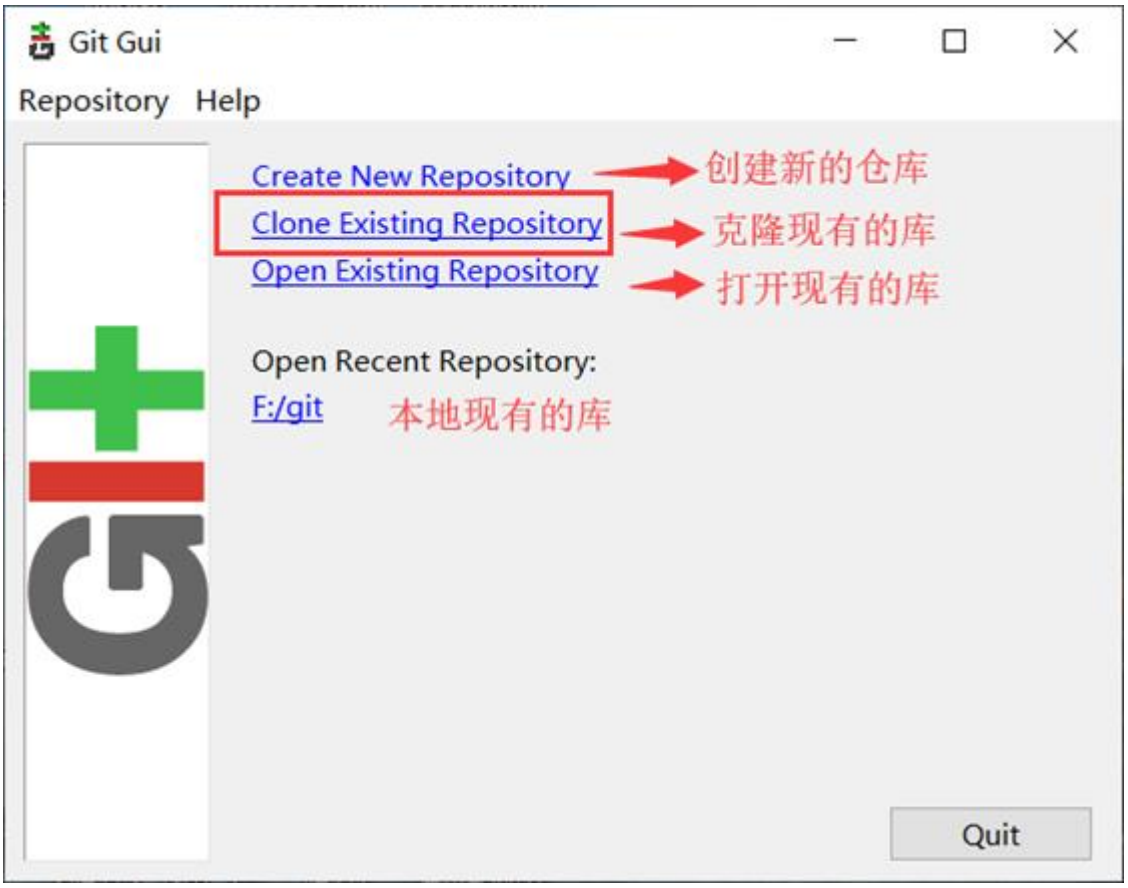


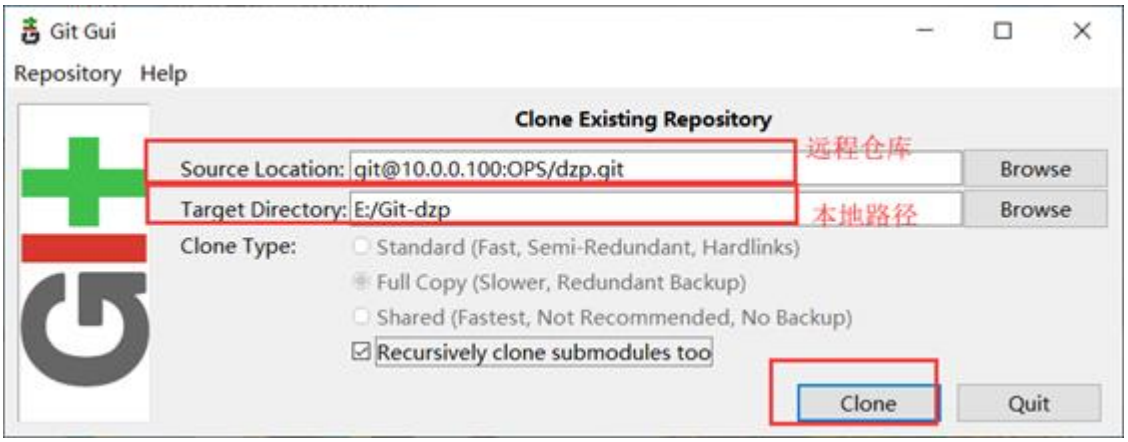


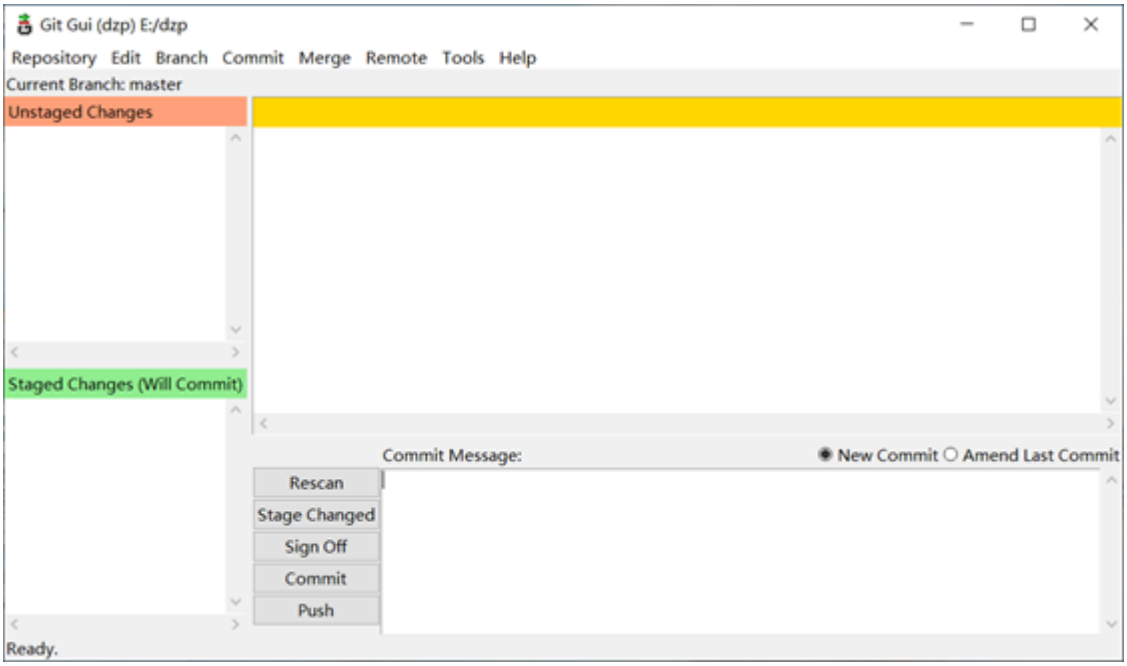

不安全 | 10.0.0.100/OPS/dzp
GitLab
Projects
Groups
Activity
Milestones
Snippets
This project
dzp
Overview
You won't be able to pull or push project code via SSH until you add an SSH key to your profile
OPS › dzp › Details
User Settings › SSH Keys
SSH Keys
SSH keys allow you to establish a secure connection between your computer and GitLab.
Add an SSH key
Before you can add an SSH key you need to generate it.
Key
ssh-rsa
Title
123123@DESKTOP-R6SVJ6J
Add key
Your SSH keys (0)

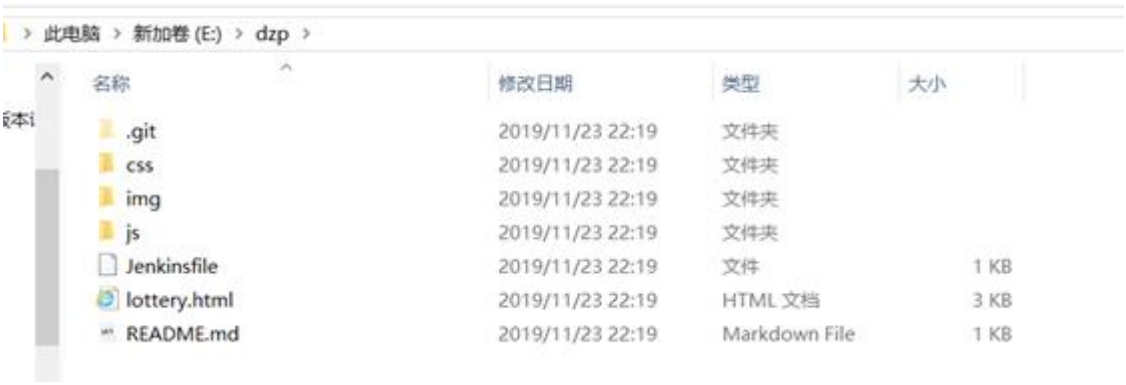

123123@DESKTOP-R6SVJ6J MINGW32 ~/Desktop
$ git config --global user.name "qls"
123123@DESKTOP-R6SVJ6J MINGW32 ~/Desktop
$ git config --global user.email "1176494252@qq.com"
123123@DESKTOP-R6SVJ6J MINGW32 ~/Desktop


查看(V)
排序方式(O)
刷新(E)
粘贴(P)
粘贴快捷方式(S)
撤消 移动(U) Ctrl+Z
Git GUI Here
Git Bash Here
360桌面助手
英特尔® 显卡设置
新建(W)
显示设置(D)
个性化(R)

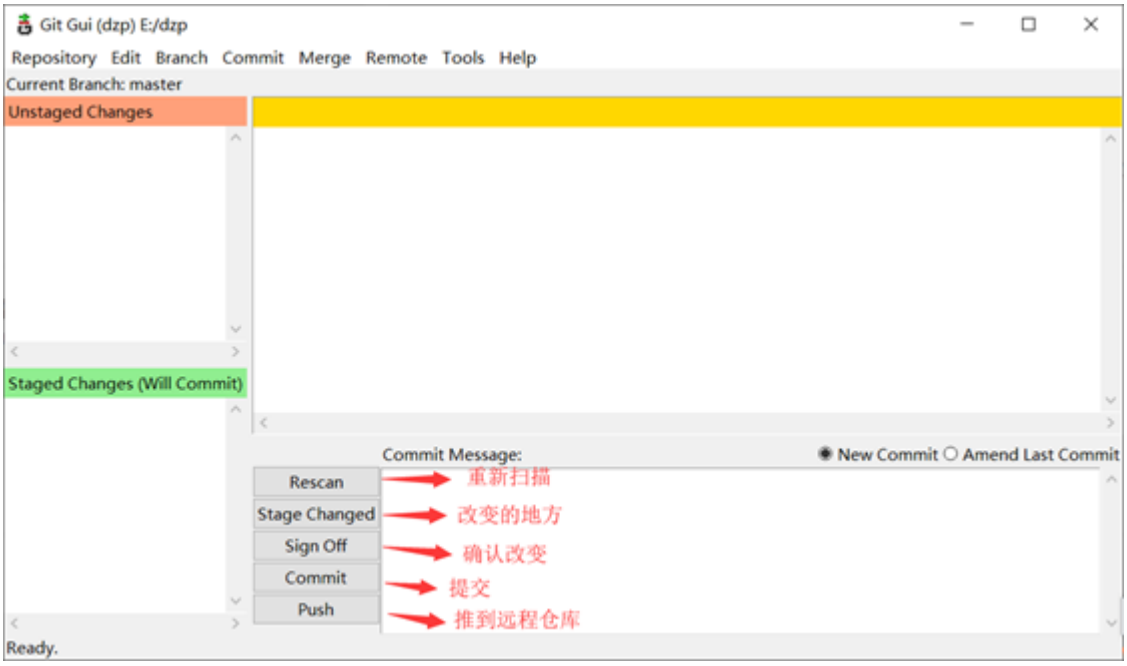

Git Gui
Repository Help
Create New Repository
创建新的仓库
Clone Existing Repository
克隆现有的库
Open Existing Repository
打开现有的库
Open Recent Repository:
F:/git
本地现有的库
Quit

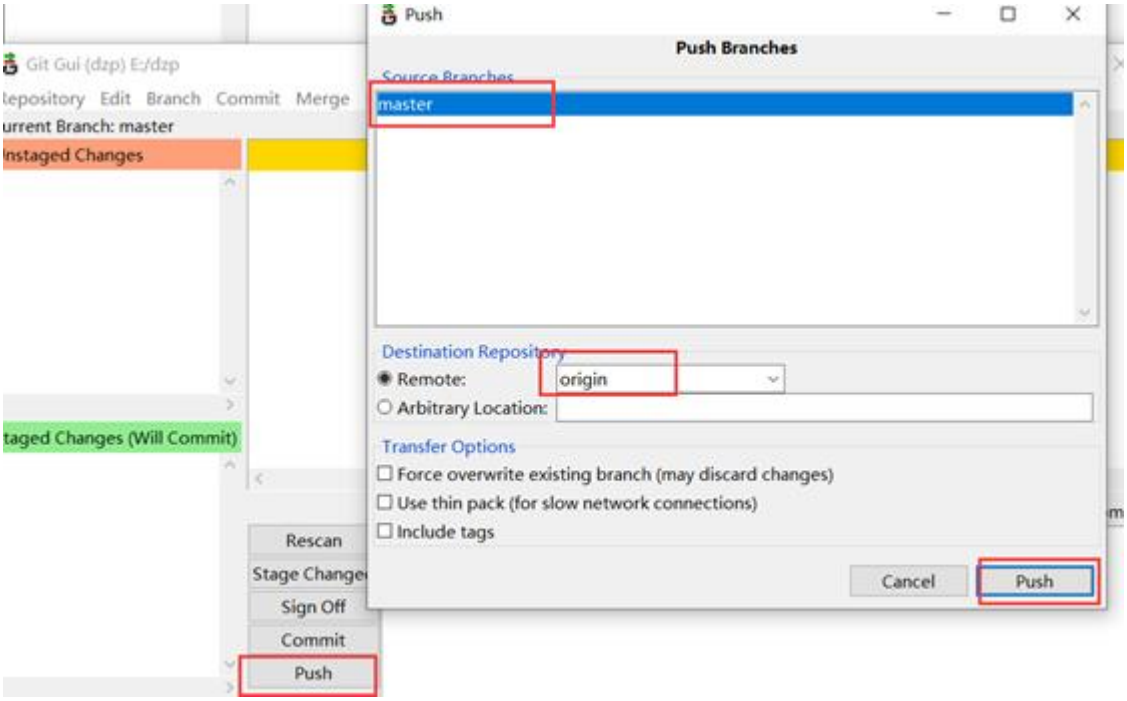

Git Gui
Repository Help
Clone Existing Repository
Source Location: git@10.0.0.100:OPS/dzp.git
远程仓库
Browse
Target Directory: E:/Git-dzp
本地路径
Browse
Clone Type:
Standard (Fast, Semi-Redundant, Hardlinks)
Full Copy (Slower, Redundant Backup)
Shared (Fastest, Not Recommended, No Backup)
Recursively clone submodules too
Clone
Quit

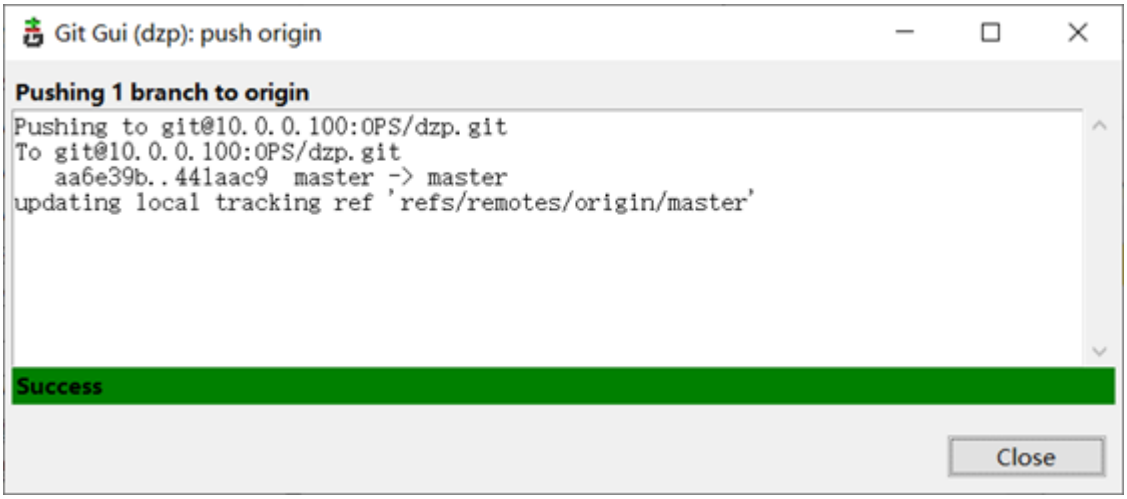

Git Gui (dzp) E:/dzp
Repository Edit Branch Commit Merge Remote Tools Help
Current Branch: master
Unstaged Changes
Staged Changes (Will Commit)
Commit Message:
New Commit
Amend Last Commit
Rescan
Stage Changed
Sign Off
Commit
Push
Ready.

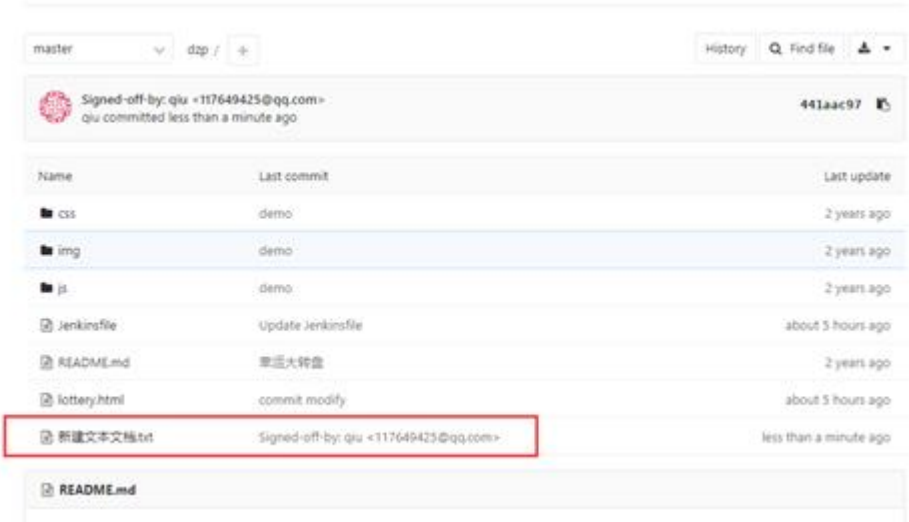

此电脑 › 新加卷 (E:) › dzp ›
名称
修改日期
类型
大小
.git 2019/11/23 22:19 文件夹
css 2019/11/23 22:19 文件夹
img 2019/11/23 22:19 文件夹
js 2019/11/23 22:19 文件夹
Jenkinsfile 2019/11/23 22:19 文件 1 KB
lottery.html 2019/11/23 22:19 HTML 文档 3 KB
README.md 2019/11/23 22:19 Markdown File 1 KB

修改代码


Git Gui (dzp) E:/dzp
Repository Edit Branch Commit Merge Remote Tools Help
Current Branch: master
Unstaged Changes
Staged Changes (Will Commit)
Commit Message:
New Commit
Amend Last Commit
Rescan
重新扫描
Stage Changed
改变的地方
Sign Off
确认改变
Commit
提交
Push
推到远程仓库
Ready.


Git Gui (dzp) E:/dzp
epository Edit Branch Commit Merge
urrent Branch: master
nstaged Changes
taged Changes (Will Commit)
Rescan
Stage Changes
Sign Off
Commit
Push
Push
Push Branches
Source Branches
master
Destination Repository
Remote:
origin
Arbitrary Location:
Transfer Options
Force overwrite existing branch (may discard changes)
Use thin pack (for slow network connections)
Include tags
Cancel
Push


Git Gui (dzp): push origin
Pushing 1 branch to origin
Pushing to git@10.0.0.100:OPS/dzp.git
To git@10.0.0.100:OPS/dzp.git
aa6e39b..441aac9 master -> master
updating local tracking ref 'refs/remotes/origin/master'
Success
Close


master
dzp /
History
Find file
Signed-off-by: qiu <117649425@qq.com>
qiu committed less than a minute ago
441aac97
Name
Last commit
Last update
css
demo
2 years ago
img
demo
2 years ago
js
demo
2 years ago
Jenkinsfile
Update Jenkinsfile
about 5 hours ago
README.md
2 years ago
lottery.html
commit modify
about 5 hours ago
新建文本文档.txt
Signed-off-by: qiu <117649425@qq.com>
less than a minute ago
README.md

8. Gitlab 备份与恢复

对 gitlab 进行备份将会创建一个包含所有库和附件的归档文件。对备份的恢复只能恢复到与备份时的 gitlab 相同的

版本。将 gitlab 迁移到另一台服务器上的最佳方法就是通过备份和还原。

gitlab 提供了一个简单的命令行来备份整个 gitlab，并且能灵活的满足需求。

备份文件将保存在配置文件中定义的 backup_path 中，文件名为 TIMESTAMP_gitlab_backup.tar,TIMESTAMP 为备份时的时间戳。TIMESTAMP 的格式为：EPOCH_YYYY_MM_DD_Gitlab - version。如果自定义备份目录需要赋予 git 权限。

01. 备份

#配置文件中加入
[root@git ~/git_data]# vim /etc/gitlab/gitlab.rb
gitlab_rails['manage_backup_path'] = true #开启备份
gitlab_rails['backup_path'] = '/data/backup/gitlab' #备份目录
gitlab_rails['backup_archive_permissions'] = 0644 #生成的备份文件权限
gitlab_rails['backup_keep_time'] = 604800 #备份保留的时间（以秒为单位，这是七天默认值）
[root@git ~/git_data]# mkdir /data/backup/gitlab
[root@git ~/git_data]# chown -R git.git /data/backup/gitlab

#完成后执行下面命令进行生效

[root@git ~/git_data]# gitlab - ctl reconfigure

#手动备份
[root@git ~/git_data]# gitlab-rake gitlab:backup:create

#检查结果
[root@git ~/git_data]# ll /data/opt/gitlab/
total 112
-rw-r--r-- 1 git git 112640 Nov 17 18:22 1573986174_2019_11_17_10.2.2_gitlab_backup.tar

自动备份需要写一个备份脚本及定时任务

#Gitlab 恢复操作

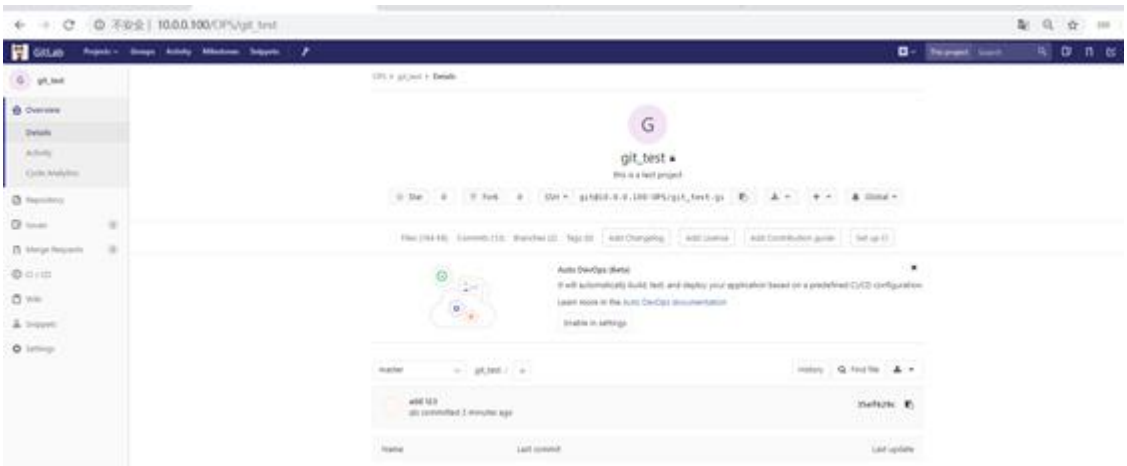
Gitlab 只能还原到与备份文件相同的 gitlab 版本。

#误删除数据

[root@git ~/git_test]# rm -rf ./*

#提交推送

[root@git ~/git_test]# git commit -am "add 123 "
[master 35ef629] add 123
3 files changed, 8 deletions(-)
delete mode 100644 a
delete mode 100644 a.txt
delete mode 100644 test
[root@git ~/git_test]# git push -u origin master
Counting objects: 3, done.
Compressing objects: 100% (1/1), done.
Writing objects: 100% (2/2), 182 bytes | 0 bytes/s, done.
Total 2 (delta 0), reused 0 (delta 0)
To git@10.0.0.100:OPS/git_test.git
02250b6..35ef629 master -> master
Branch master set up to track remote branch master from origin.

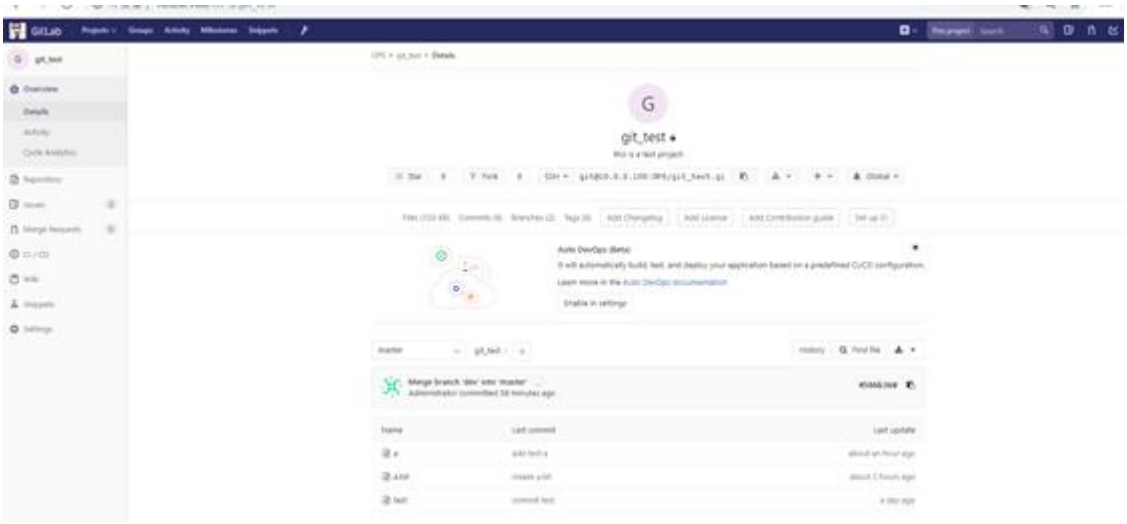


02. 恢复

#停止相关服务

[root@git ~/git_test]# gitlab-ctl stop unicorn
ok: down: unicorn: 0s, normally up
[root@git ~/git_test]# gitlab-ctl stop sidekiq
ok: down: sidekiq: 1s, normally up

```
[root@git ~/git_test]# gitlab-ctl status
run: gitaly: (pid 78774) 15714s; run: log: (pid 78466) 15758s
run: gitlab-monitor: (pid 78790) 15713s; run: log: (pid 78565) 15746s
run: gitlab-workhorse: (pid 78762) 15714s; run: log: (pid 78419) 15776s
run: logrotate: (pid 106694) 1364s; run: log: (pid 78447) 15764s
run: nginx: (pid 78432) 15770s; run: log: (pid 78431) 15770s
run: node-exporter: (pid 78515) 15752s; run: log: (pid 78514) 15752s
run: postgres-exporter: (pid 78818) 15712s; run: log: (pid 78692) 15728s
run: postgresql: (pid 78190) 15819s; run: log: (pid 78189) 15819s
run: prometheus: (pid 78806) 15713s; run: log: (pid 78646) 15734s
run: redis: (pid 78130) 15825s; run: log: (pid 78129) 15825s
run: redis-exporter: (pid 78626) 15740s; run: log: (pid 78625) 15740s
down: sidekiq: 8s, normally up; run: log: (pid 78402) 15782s
down: unicorn: 20s, normally up; run: log: (pid 78363) 15788s

#Gitlab 的恢复操作会先将当前所有的数据清空，然后再根据备份数据进行恢复

[root@git ~/git_test]# ll /data/opt/gitlab/
total 112
-rw------- 1 git git 112640 Nov 17 18:22 1573986174_2019_11_17_10.2.2_gitlab_backup.tar
[root@git ~/git_test]# gitlab-rake gitlab:backup:restore BACKUP=1573986174_2019_11_17_10.2.2

#启动 gitlab
[root@git ~/git_test]# gitlab-ctl start
ok: run: gitaly: (pid 78774) 16168s
ok: run: gitlab-monitor: (pid 78790) 16167s
ok: run: gitlab-workhorse: (pid 78762) 16168s
ok: run: logrotate: (pid 106694) 1818s
ok: run: nginx: (pid 78432) 16224s
ok: run: node-exporter: (pid 78515) 16206s
ok: run: postgres-exporter: (pid 78818) 16166s
ok: run: postgresql: (pid 78190) 16273s
ok: run: prometheus: (pid 78806) 16167s
ok: run: redis: (pid 78130) 16279s
ok: run: redis-exporter: (pid 78626) 16194s
ok: run: sidekiq: (pid 109932) 1s
ok: run: unicorn: (pid 109939) 0s

#检查是否恢复成功
[root@git ~/git_test]# gitlab-rake gitlab:check SANITIZE=true
```



#web 界面数据已经恢复，从新克隆到本地

[root@git ~]# git clone git@10.0.0.100:OPS/git_test.git
Cloning into 'git_test'...
remote: Counting objects: 27, done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 27 (delta 1), reused 27 (delta 1)
Receiving objects: 100% (27/27), done.
Resolving deltas: 100% (1/1), done.
[root@git ~/git_test]# ll
total 8
-rw-r--r-- 1 root root 27 Nov 17 19:00 a
-rw-r--r-- 1 root root 7 Nov 17 19:00 a.txt
-rw-r--r-- 1 root root 0 Nov 17 19:00 test

9. Jenkins

官网

https://jenkins.io

Jenkins 是一个开源软件项目，是基于 Java 开发的一种持续集成工具，用于监控持续重复的工作，指在提供一个开
放易用的软件平台，使软件的持续集成变成可能。

01. 安装准备

主机	IP	内存	硬盘
Jenkins	10.0.0.201	2G	50G+

nexus	10.0.0.202	2G	50G+

02 .安装 Jdk 和 Jenkins

#上传 JDK 和 Jenkins 安装包，使用 rpm - ivh 进行安装,安装完 JDK 运行 Java 测试是否安装成功

```
[root@jenkins ~]# ll
-rw-r--r-- 1 root root 170023183 2018-08-14 11:05 jdk-8u181-linux-x64.rpm
-rw-r--r-- 1 root root 74141787 2018-08-13 20:23 jenkins-2.99-1.1.noarch.rpm
[root@jenkins ~]# rpm -ivh jdk-8u181-linux-x64.rpm
warning: jdk-8u181-linux-x64.rpm: Header V3 RSA/SHA256 Signature, key ID ec551f03: NOKEY
Preparing... ########################################### [100%]
Updating / installing...
1:jdk1.8-2000:1.8.0_181-fcs ################################### [100%]
Unpacking JAR files...
    tools.jar...
    plugin.jar...
    javaws.jar...
    deploy.jar...
    rt.jar...
    jsse.jar...
    charsets.jar...
    localedata.jar...
[root@jenkins ~]# rpm -ivh jenkins-2.99-1.1.noarch.rpm
warning: jenkins-2.99-1.1.noarch.rpm: Header V4 DSA/SHA1 Signature, key ID d50582e6: NOKEY
Preparing... ########################################### [100%]
Updating / installing...
1:jenkins-2.99-1.1 ############################################ [100%]
```

03 .配置 Jenkins

#启动用户修改为 root

```
[root@jenkins ~]# grep 'root' /etc/sysconfig/jenkins
JENKINS_USER="root"
```

#启动

```
[root@jenkins ~]# systemctl start jenkins
```

#查看端口

```
[root@jenkins ~]# netstat -lntp | grep 8080
tcp6 0 0 :::8080 :::* LISTEN 8504/java
```

#查看进程

```
[root@jenkins ~]# ps aux |grep jenkins
root 8504 8.8 11.6 2618104 235556 ? Ssl 15:29 0:25 /etc/alternatives/java -Dcom.sun.akuma.Daemon=daemonized -Djava.awt.headless=true -DJENKINS_HOME=/var/lib/jenkins
```

-jar /usr/lib/jenkins/jenkins.war --logfile=/var/log/jenkins/jenkins.log --webroot=/var/cache/jenkins/war --daemon --httpPort=8080 --debug=5 --handlerCountMax=100 --handlerCountMaxIdle=20

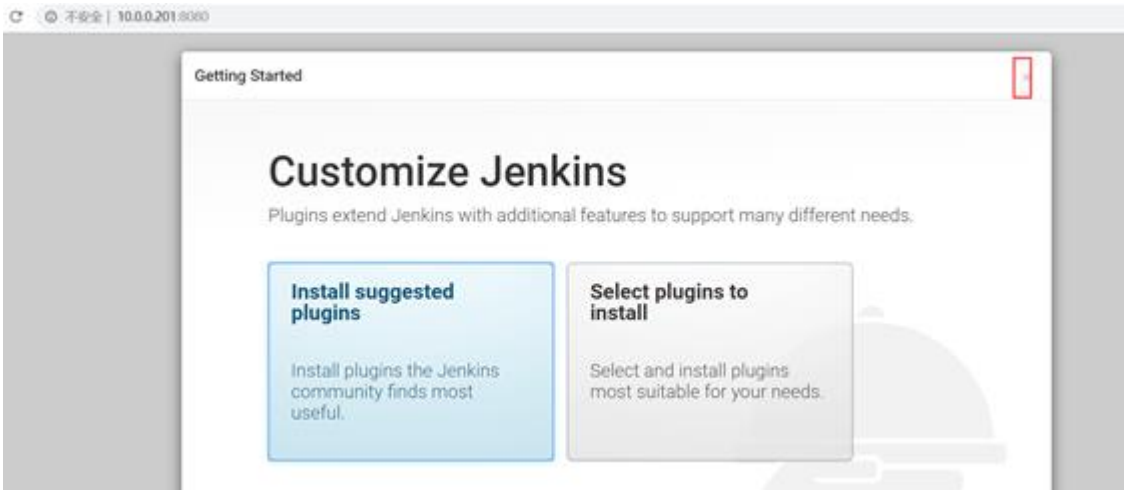
#访问页面进行配置

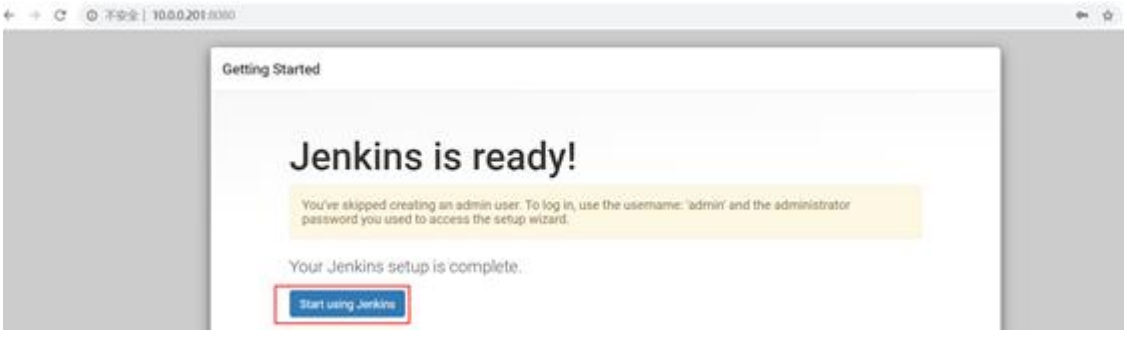
http://10.0.0.201:8080

#查看密码

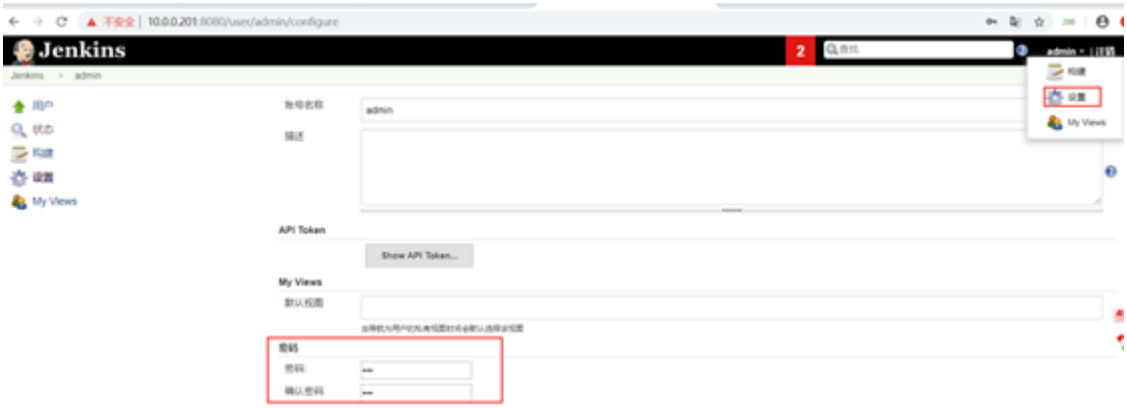
[root@jenkins ~]# cat /var/lib/jenkins/secrets/initialAdminPassword

e8e69c5646cc4f3b88315fb20156ce6c





修改密码



04. 插件安装

#插件安装（跳过安装插件，直接上传插件到目录）
1.自动安装可选插件
2.手动下载插件上传安装
3.插件放入插件目录

[root@jenkins ~]# cd /var/lib/jenkins/
[root@jenkins jenkins]# ll　　#jobs 为每次构建后构建的结果目录，plugins 为插件目录
总用量 36
-rw-------. 1 root root 1822 8 月 26 00:35 config.xml
-rw-------. 1 root root 156 8 月 26 00:31 hudson.model.UpdateCenter.xml
-rw-------. 1 root root 1712 8 月 26 00:32 identity.key.enc
-rw-------. 1 root root 94 8 月 26 00:32 jenkins.CLI.xml
-rw-r--r--. 1 root root 4 8 月 26 00:35 jenkins.install.InstallUtil.lastExecVersion
-rw-r--r--. 1 root root 4 8 月 26 00:35 jenkins.install.UpgradeWizard.state
drwxr-xr-x 2 root root 6 8 月 26 00:31 jobs
drwxr-xr-x 3 root root 18 8 月 26 00:32 logs
-rw-------. 1 root root 907 8 月 26 00:32 nodeMonitors.xml
drwxr-xr-x 2 root root 6 8 月 26 00:32 nodes
drwxr-xr-x 2 root root 6 8 月 26 00:31 plugins
-rw-------. 1 root root 64 8 月 26 00:31 secret.key
-rw-r--r--. 1 root root 0 8 月 26 00:31 secret.key.not-so-secret
drwx------. 4 root root 4096 8 月 26 00:32 secrets
drwxr-xr-x 2 root root 23 8 月 26 00:32 userContent
drwxr-xr-x 3 root root 18 8 月 26 00:34 users

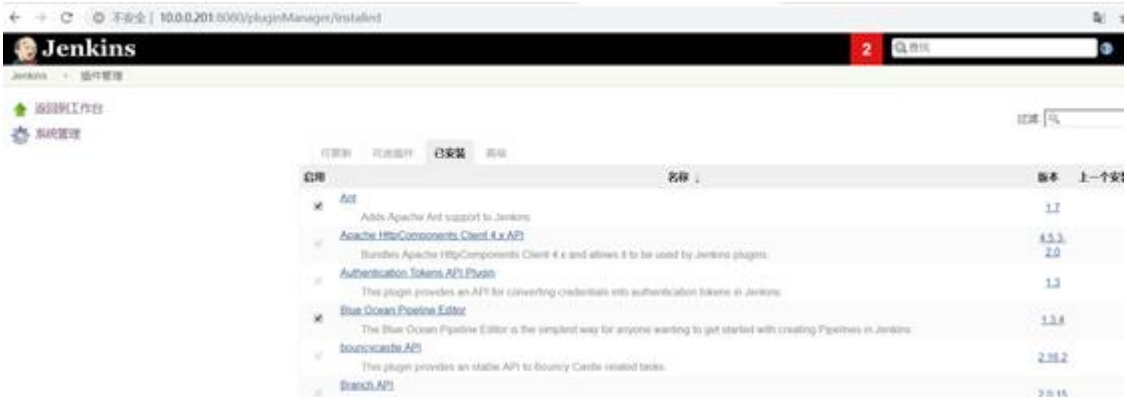
#上传插件包解压到 plugins
[root@jenkins jenkins]# cd plugins/
[root@jenkins plugins]# ll
total 160580
-rw-r--r-- 1 root root 164431230 2018-08-14 21:00 plugins.tar.gz
[root@jenkins plugins]# tar xf plugins.tar.gz
[root@jenkins plugins]# rm -f plugins.tar.gz

[root@jenkins plugins]# mv plugins/* ./
[root@jenkins plugins]# rm -rf plugins/

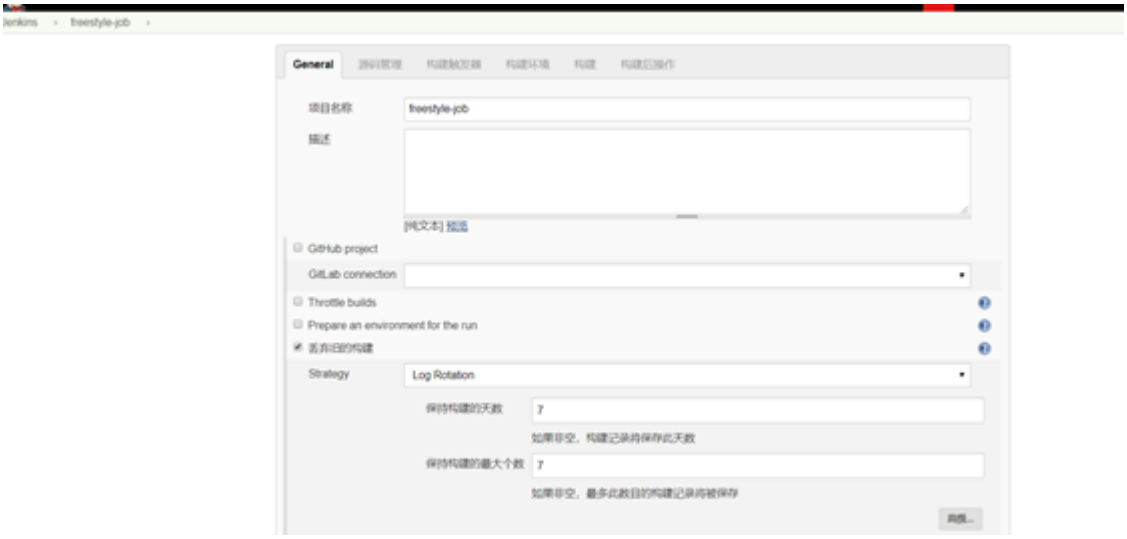
#重启生效
[root@jenkins plugins]# systemctl restart jenkins.service

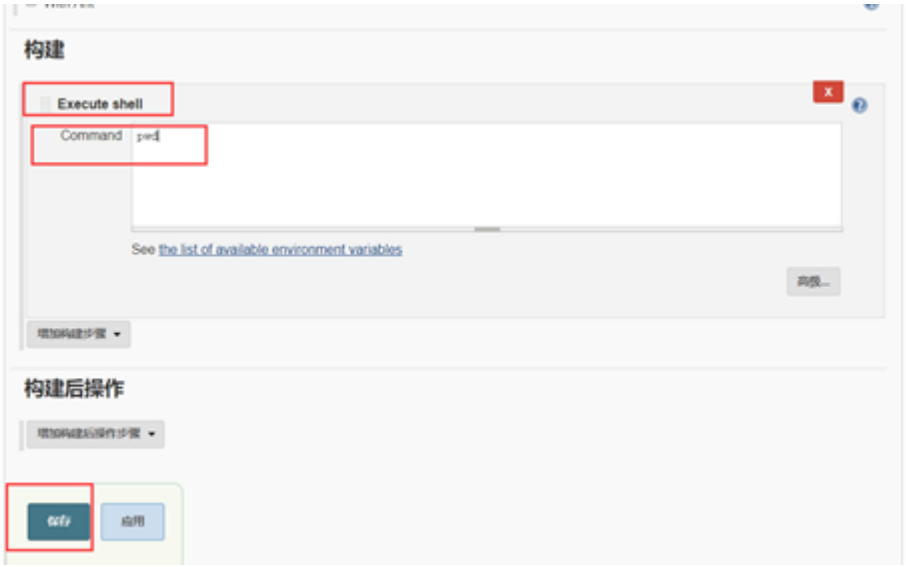
#4.jenkins 主要的目录

/usr/lib/jenkins/：　　#jenkins 安装目录，WAR 包会放在这里
/etc/sysconfig/jenkins：　　#jenkins 配置文件，"端口"，"JENKINS_HOME"等都可以在这里配置
/var/lib/jenkins/：　　#默认的 JENKINS_HOME
/var/log/jenkins/jenkins.log：　　#Jenkins 日志文件



05. 创建项目


jenkins › freestyle-job ›
General
项目名称
freestyle-job
描述
[纯文本] 预览
GitHub project
GitLab connection
Throttle builds
Prepare an environment for the run
丢弃旧的构建
Strategy
Log Rotation
保持构建的天数
7
如果非空，构建记录将保存此天数
保持构建的最大个数
7
如果非空，最多此数目的构建记录将被保存
高级...


构建
Execute shell
Command
pwd
See the list of available environment variables
高级...
增加构建步骤
构建后操作
增加构建后操作步骤
保存
应用

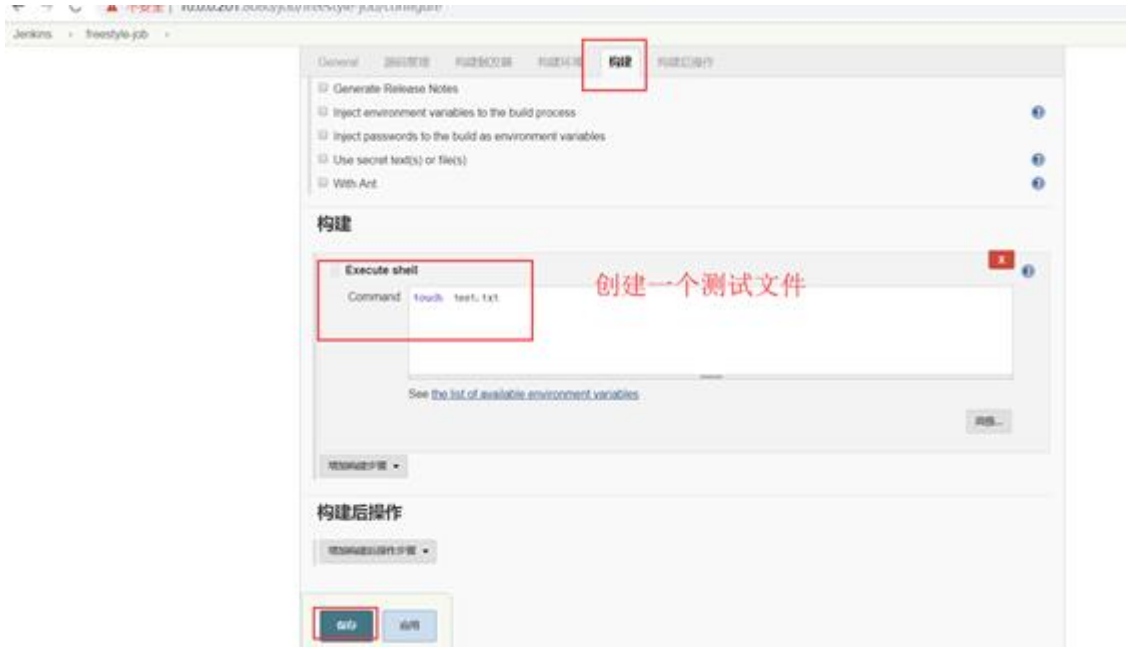
进行构建


不安全 | 10.0.0.201:8080/job/freestyle-job/
Jenkins
Jenkins › freestyle-job ›
返回面板
状态
修改记录
工作空间
立即构建
删除 Project
配置
Favorite
Open Blue Ocean
Project freestyle-job
工作区
最新修改记录
相关连接
Build History
构建历史
find
RSS 全部 RSS 失败

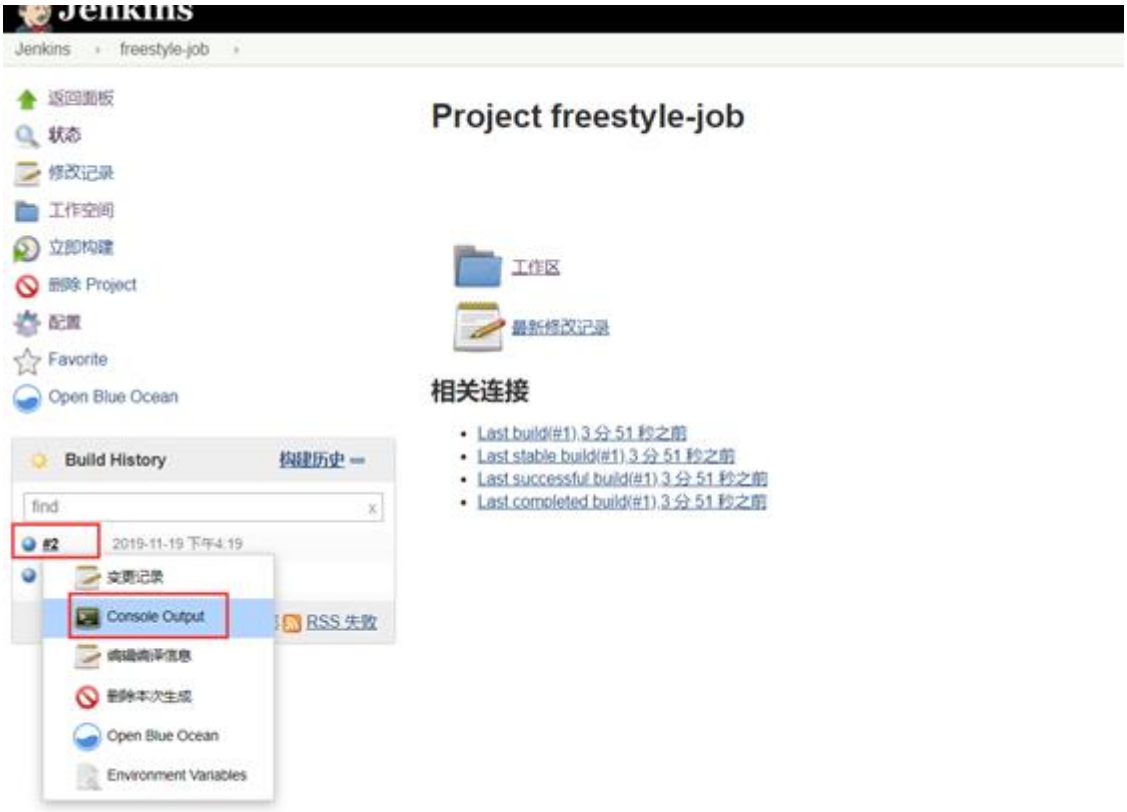

Jenkins
Jenkins › freestyle-job ›
返回面板
状态
修改记录
工作空间
立即构建
删除 Project
配置
Favorite
Open Blue Ocean
Project freestyle-job
工作区
最新修改记录
相关连接
Build History
构建历史
find
2019-11-19 下午4:15
表示成功
RSS 全部 RSS 失败

进入控制台

[root@jenkins plugins]# ll /var/lib/jenkins/workspace/freestyle-job
total 0


Jenkins
freestyle-job
Generate Release Notes
Inject environment variables to the build process
Inject passwords to the build as environment variables
Use secret text(s) or file(s)
With Ant
构建
Execute shell
Command
touch test.txt
创建一个测试文件
See the list of available environment variables
构建后操作


Jenkins
Jenkins
freestyle-job
返回面板
状态
修改记录
工作空间
立即构建
删除 Project
配置
Favorite
Open Blue Ocean
Build History
构建历史
find
#1
2019-11-19 下午4:15
RSS 全部
RSS 失败
Project freestyle-job
工作区
最新修改记录
相关连接
Last build(#1),3 分 51 秒之前
Last stable build(#1),3 分 51 秒之前
Last successful build(#1),3 分 51 秒之前
Last completed build(#1),3 分 51 秒之前



[root@jenkins plugins]# ll /var/lib/jenkins/workspace/freestyle-job

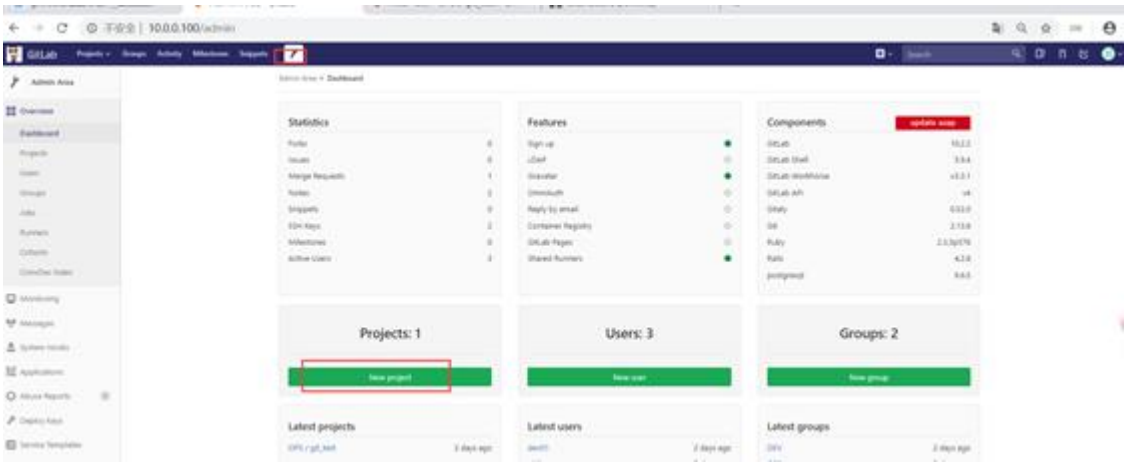
total 0

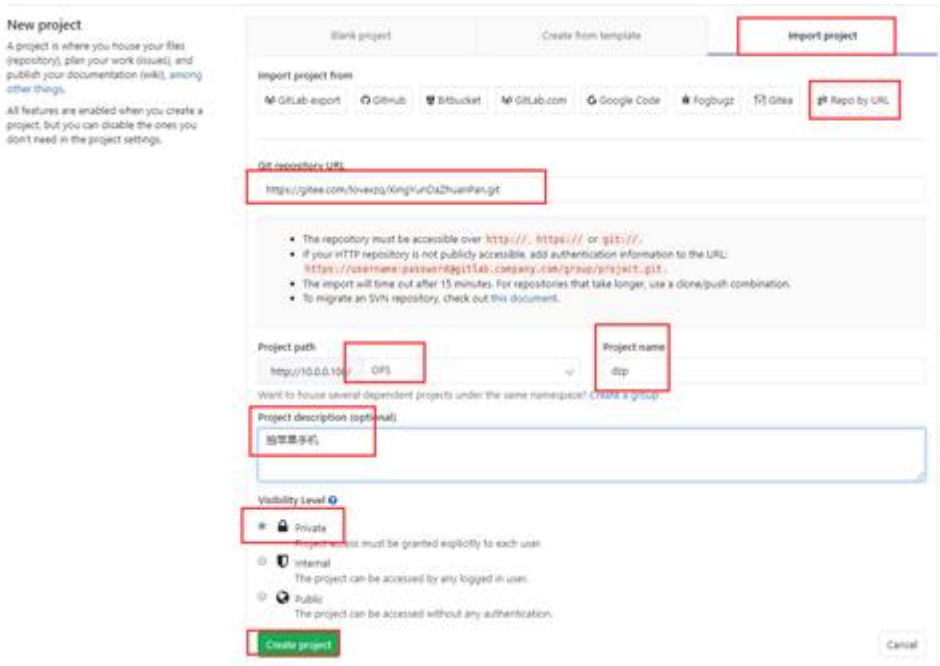
-rw-r--r-- 1 root root 0 2019-11-19 16:19 test.txt

06. Jenkins 获取 Git 源代码

#这里我们有码云导入一个 HTML 页面的监控平台到 gitlab 仓库,打开码云，找到一个大转盘项目，将其代码路径进行复制

在 Gitlab 上面新创建一个项目仓库。将源代码导入进去。





#dev 用户端配置从 git 获取代码。

[root@dev ~]# git clone git@10.0.0.100:OPS/dzp.git
Cloning into 'dzp'...
remote: Counting objects: 19, done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 19 (delta 2), reused 0 (delta 0)
Receiving objects: 100% (19/19), 104.36 KiB | 0 bytes/s, done.
Resolving deltas: 100% (2/2), done.
[root@dev ~]# ll
drwxr-xr-x 6 root root 87 Nov 19 20:41 dzp
drwxr-xr-x 3 root root 52 Nov 17 17:21 git_test
[root@dev ~]# cd dzp/
[root@dev ~/dzp]# ll
total 8
drwxr-xr-x 2 root root 25 Nov 19 20:41 css
drwxr-xr-x 2 root root 84 Nov 19 20:41 img
drwxr-xr-x 2 root root 41 Nov 19 20:41 js
-rw-r--r-- 1 root root 2170 Nov 19 20:41 lottery.html
-rw-r--r-- 1 root root 113 Nov 19 20:41 README.md

#dev 用户修改了源代码

```
[root@dev ~/dzp]# vim lottery.html
[root@dev ~/dzp]#
[root@dev ~/dzp]#
[root@dev ~/dzp]# git commit -am "modify html"
[master 1e2125e] modify html
1 file changed, 5 insertions(+), 5 deletions(-)
[root@dev ~/dzp]# git push -u origin master
Counting objects: 5, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 365 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
To git@10.0.0.100:OPS/dzp.git
3935eb7..1e2125e master -> master
Branch master set up to track remote branch master from origin.
```

#Jenkins 配置从 Git 获取代码，由于我们 dev 用户不是配置在 Jenkins 上，所以需认证即可下载代码。进行面认证方法，需要配置 deploy key

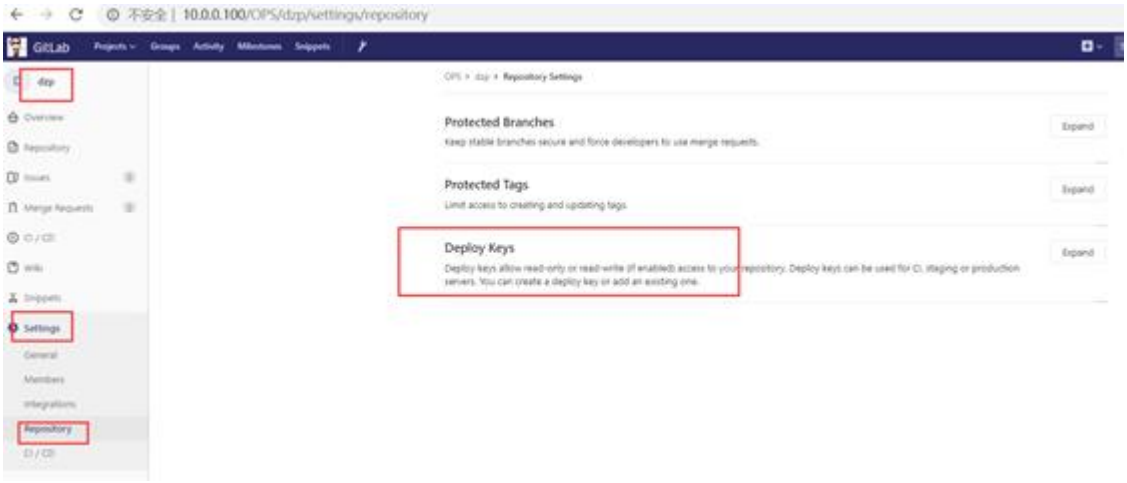
```
[root@jenkins ~]# ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:1z+H0mc9hUwOWiO3mFOGswuOW9jNr2bmOwRE50eECQU root@jenkins
The key's randomart image is:
+---[RSA 2048]----+
| .E+o+o |
| .ooo |
| . = O . |
| . # B . |
| S.B o + .|
| = =.o o .o|
| o +.+ . =.=|
| o =. . =.|
| . =++. |
+----[SHA256]-----+
[root@jenkins ~]# cat .ssh/
```

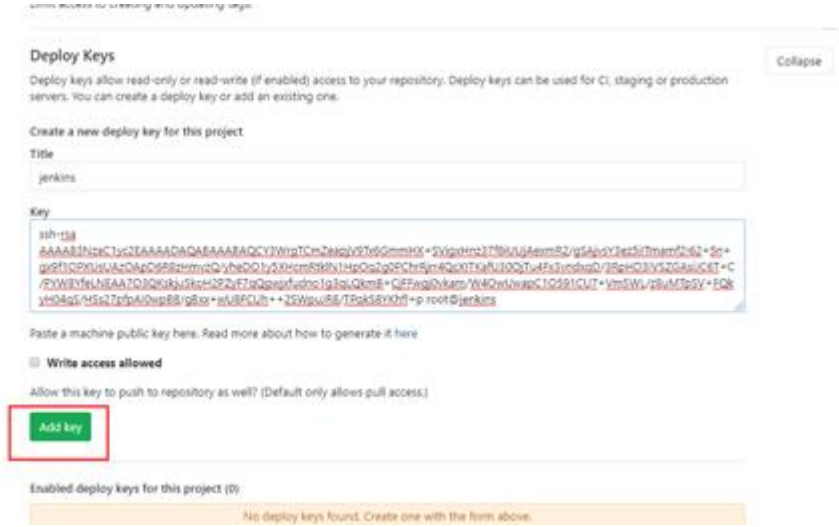
id_rsa id_rsa.pub known_hosts

[root@jenkins ~]# cat .ssh/id_rsa.pub

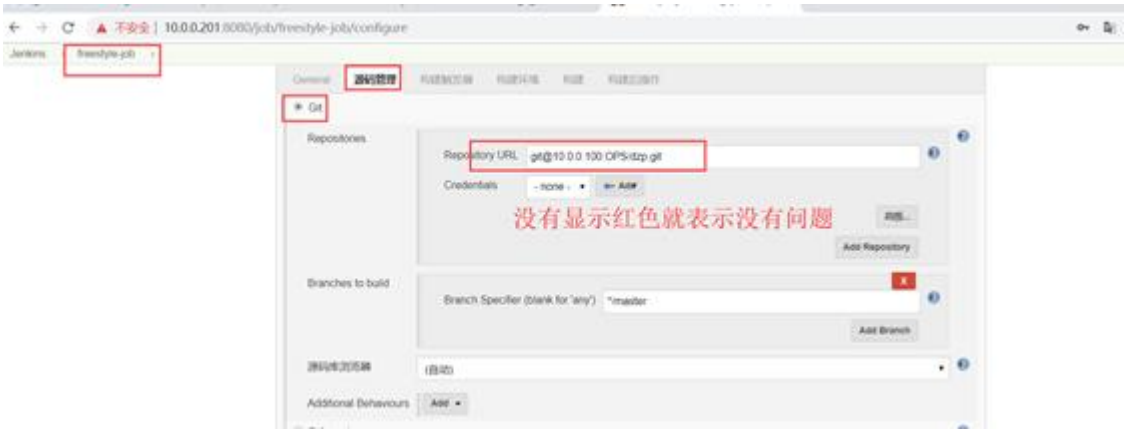
ssh-rsa

AAAAB3NzaC1yc2EAAAADAQABAAABAQCY3WrgTCmZeapjV9Tx6GmmHX+SVigxHnz37f8iUUjAexm
R2/gSAjvsY3ez5iITmamf2I62+5n+gx9f1OPXUsUAzOApD6R8zHmvzQ/yheDO1y5XHcmRtklN1HpOq2g
0PChrRjrr4QcXITKafU30OjTu4Fx3vndxqD/3RpHD3IVSZGAsiJC6T+C/PYW8YfeLNEAA7O3QKskjuSko
H2PZyF7qQgwjxfudno1g3qLQkmB+CjFFwgj0vkam/W4OwUwapC1O591CU7+VmSWL/z8uMTpSV+FQ
kyH04qS/HSs27pfpAI0wpBB/gBxx+wU8FCUh++2SWpuJR8/TPqkS8YKhfl+p root@jenkins #复制该串
代码





#Jenkins 配置从 Git 获取代码,配置好后保存



07. 立即构建获取源代码

#Jenkins 服务器查看代码

[root@jenkins ~]# ll /var/lib/jenkins/workspace/freestyle-job/
total 8
drwxr-xr-x 2 root root 25 2019-11-19 21:12 css
drwxr-xr-x 2 root root 84 2019-11-19 21:12 img
drwxr-xr-x 2 root root 41 2019-11-19 21:12 js
-rw-r--r-- 1 root root 2205 2019-11-19 21:12 lottery.html
-rw-r--r-- 1 root root 113 2019-11-19 21:12 README.md

08. Jenkins 代码推送到 Web

#写一个脚本把从 git 仓库里获取的代码上传到 web 服务器站点目录下

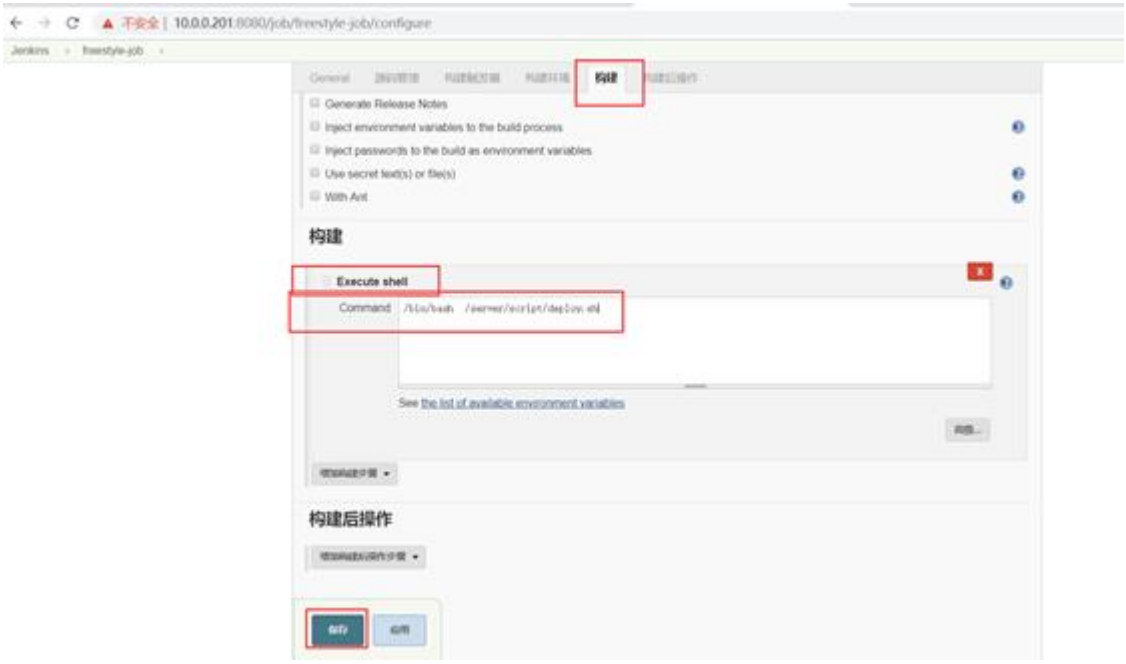
[root@jenkins ~]# mkdir -p /server/script
[root@jenkins ~]# cd /server/script
[root@jenkins script]# vim deploy.sh
[root@jenkins script]# cat deploy.sh
#!/bin/sh
Date=$(date +%s)
Code_Dir=/var/lib/jenkins/workspace/freestyle-job
Web_Dir=/code
Ip=10.0.0.202
Code_Tar() {
    cd Code_Dir && tar zcf /opt/web_Code_Dir && tar zcf /opt/web_{Date}.tar.gz ./*
}
Scp_Code_Web() {
    scp /opt/web_Date.tar.gzroot@Date.tar.gzroot@{Ip}:/opt

```
}
Code_Tar_Xf() {
    ssh        root@Ip " cd /opt && mkdir web_Ip " cd /opt && mkdir web_Date        &&        tar        xf
web_{Date}.tar.gz -C web_{Date}.tar.gz -C web_Date "
}
Ln_Html() {
    ssh root@Ip''rm−rfIp"rm−rfWeb_Dir && ln -s /opt/web_$Date /code "
}
Code_Tar;
Scp_Code_Web;
Code_Tar_Xf;
Ln_Html
```

#分发公钥

```
[root@jenkins script]# ssh-copy-id -i /root/.ssh/id_rsa.pub root@10.0.0.202
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/root/.ssh/id_rsa.pub"
The authenticity of host '10.0.0.202 (10.0.0.202)' can't be established.
ECDSA key fingerprint is SHA256:K8NapPzlTxhCMXC/bRFTtI9mdwr63FH4Wu7psrXXqBs.
ECDSA key fingerprint is MD5:73:9f:67:f1:5d:39:10:3d:b2:be:f7:c1:66:aa:00:6e.
Are you sure you want to continue connecting (yes/no)? yes
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already
installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the
new keys
root@10.0.0.202's password:


Number of key(s) added: 1


Now try logging into the machine, with: "ssh 'root@10.0.0.202'"
and check to make sure that only the key(s) you wanted were added.


[root@jenkins script]# ssh root@10.0.0.202
Last login: Tue Nov 19 15:10:55 2019 from 10.0.0.1
[root@nexus ~]# logout
Connection to 10.0.0.202 closed.
```

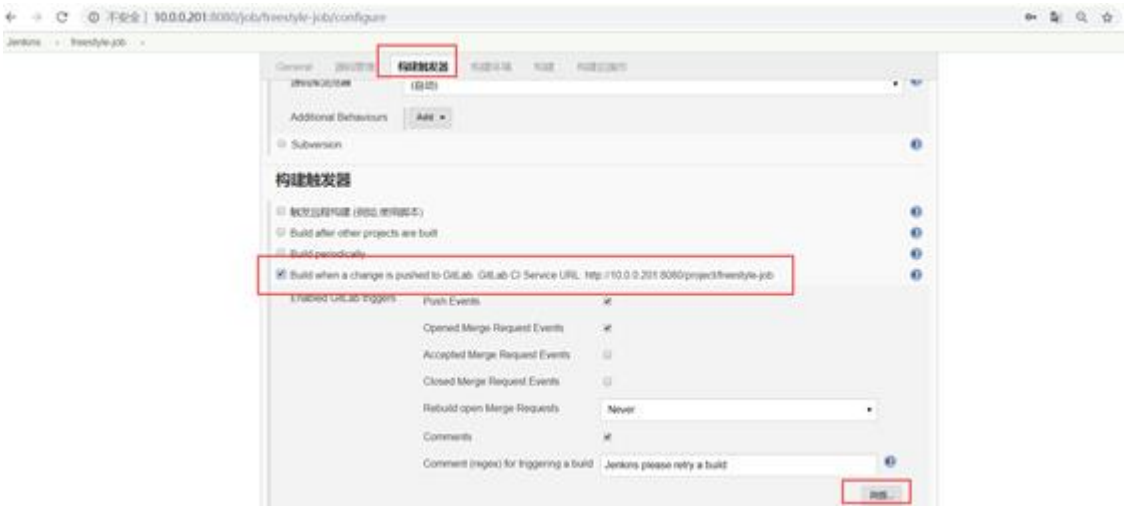
#Jenkins 上面添加脚本进行构建

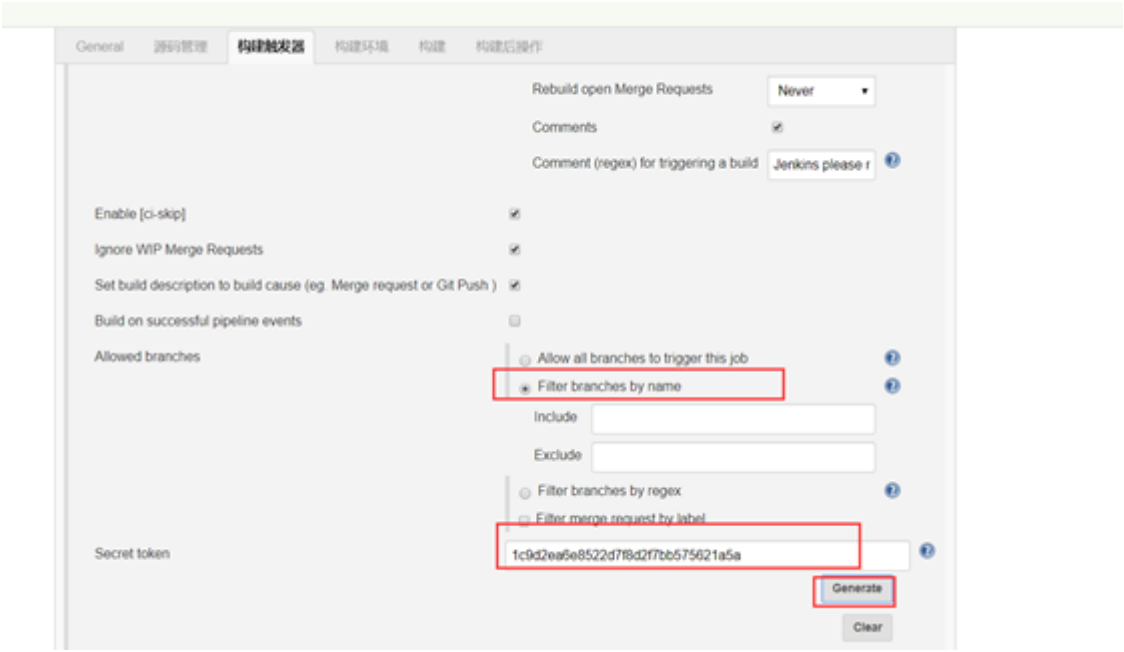


查看网站是否更新成功

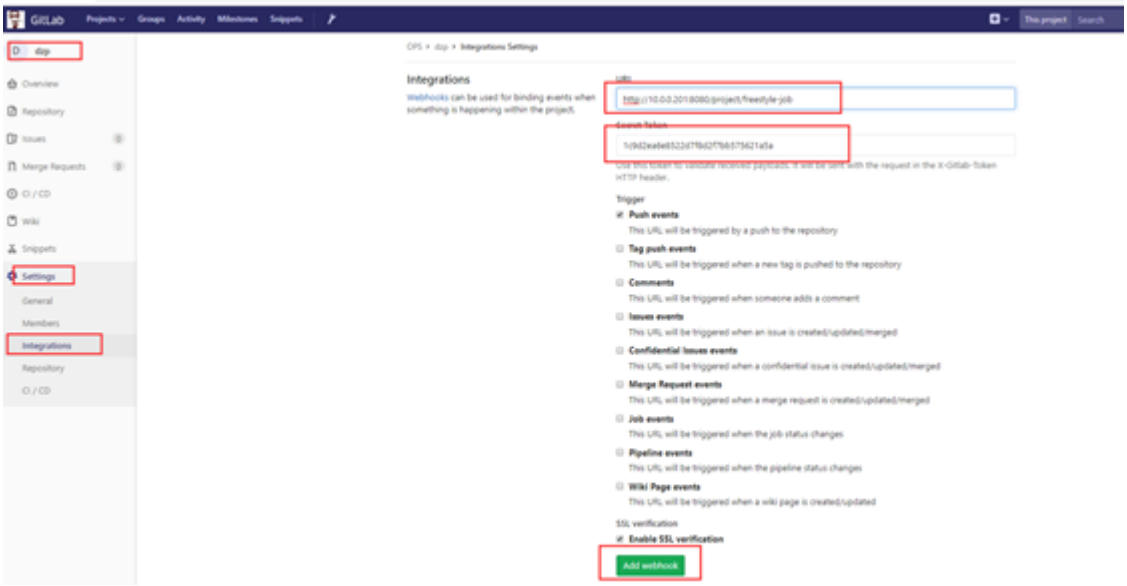
09. 配置自动触发构建

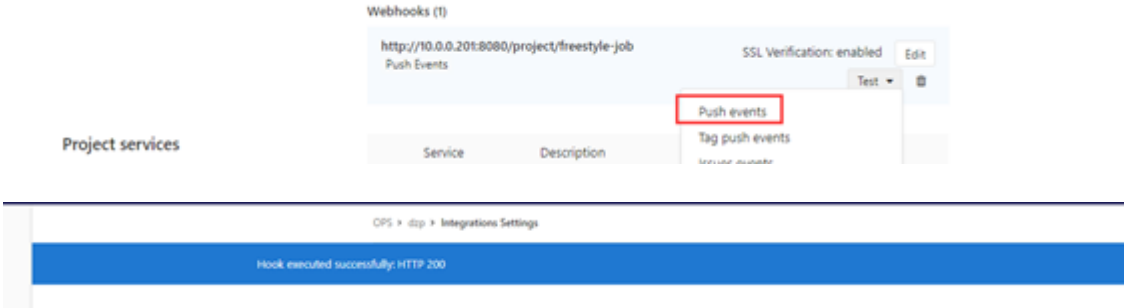
#需要设置安全令牌 Secret token





Gitlab 上面操作

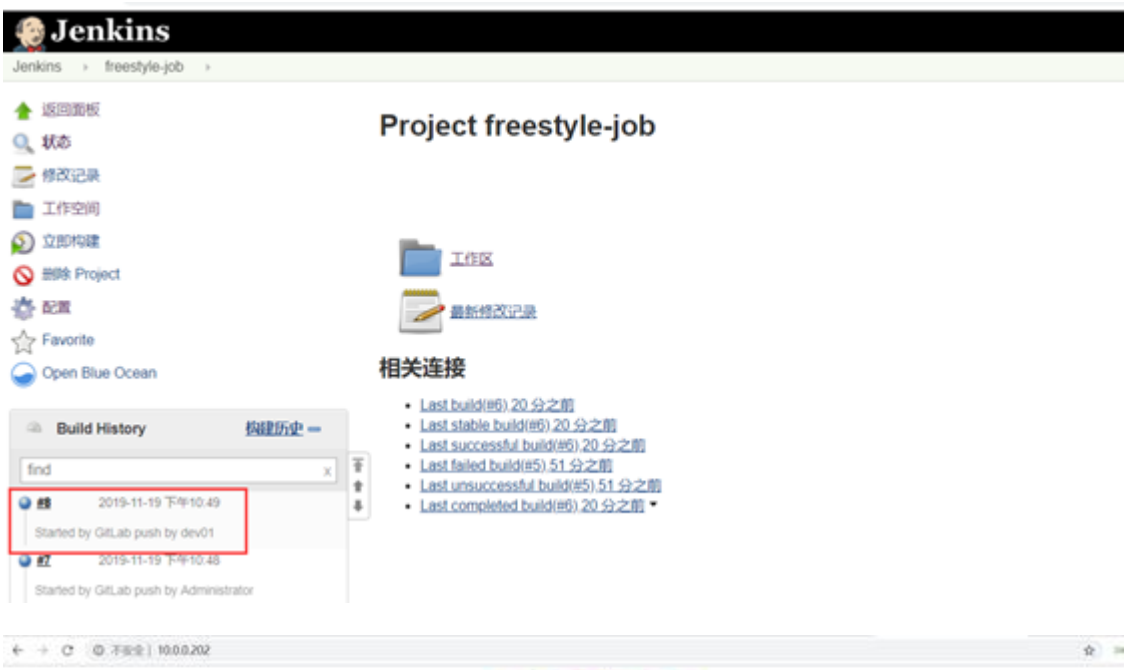




10. 测试是否触发

#dev 用户修改源代码后进行推送测试是否自动触发

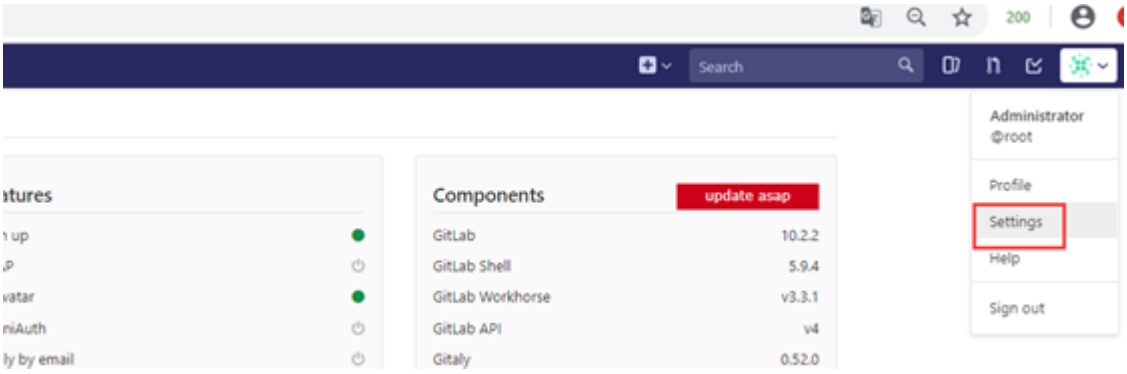
[root@dev ~/dzp]# vim lottery.html
[root@dev ~/dzp]# git commit -am "modify html test"
[master d8849aa] modify html test
1 file changed, 3 insertions(+), 3 deletions(-)
[root@dev ~/dzp]# git push -u origin master
Counting objects: 5, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 303 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
To git@10.0.0.100:OPS/dzp.git
1e2125e..d8849aa master -> master
Branch master set up to track remote branch master from origin.

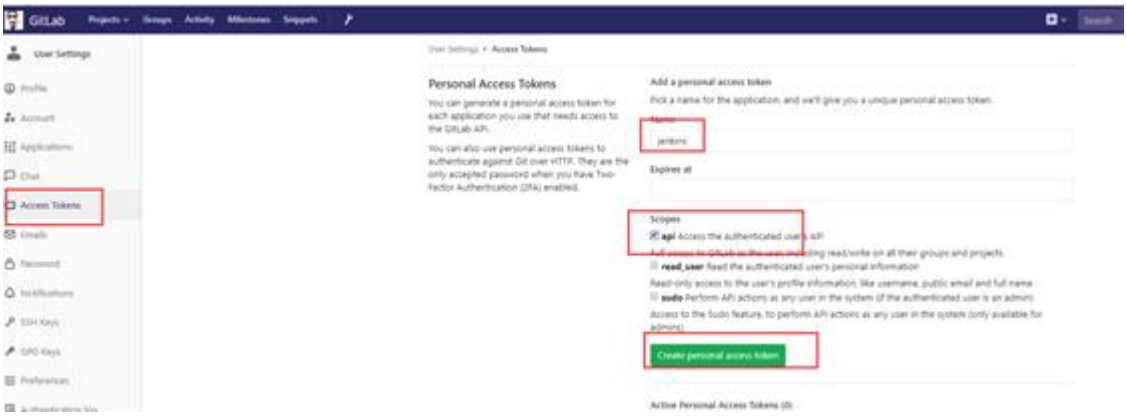


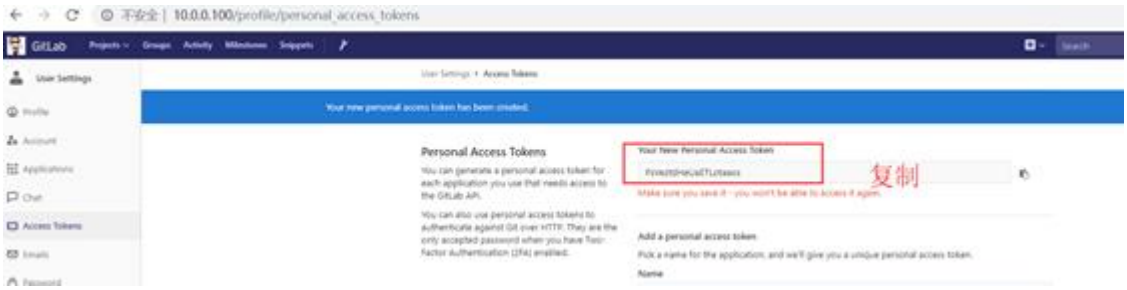
11. 返回构建状态

#Jenkins 配置 Jenkins 返回构建状态到 Gitlab

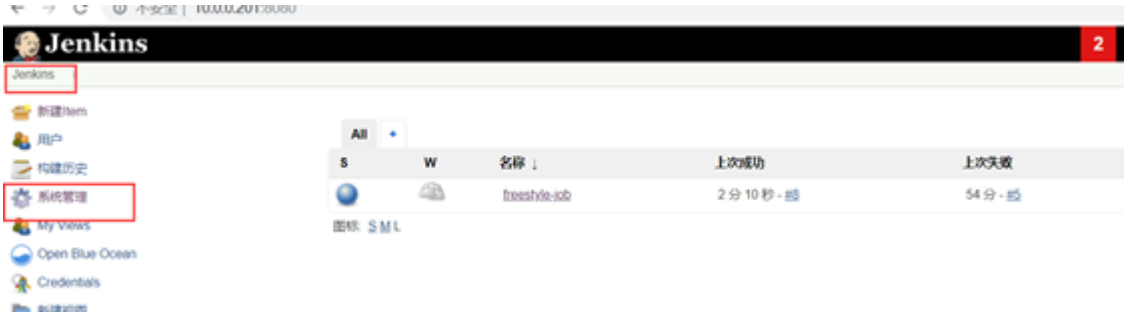
#先获得 gitlab 的 token

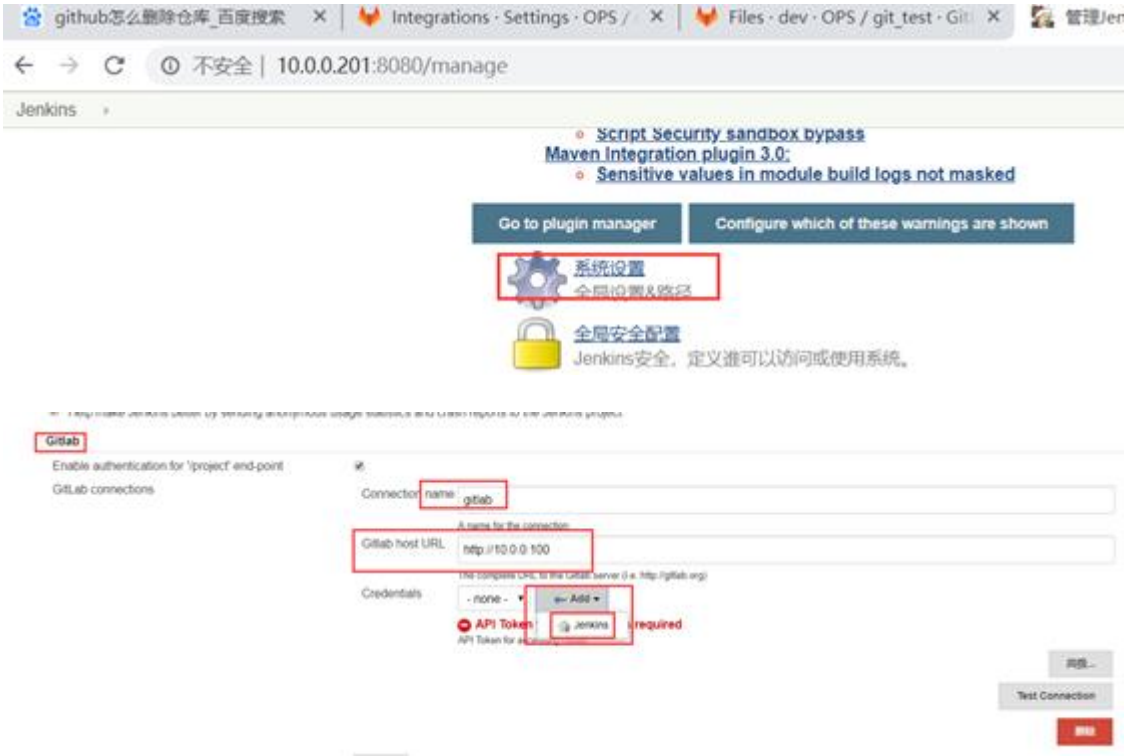


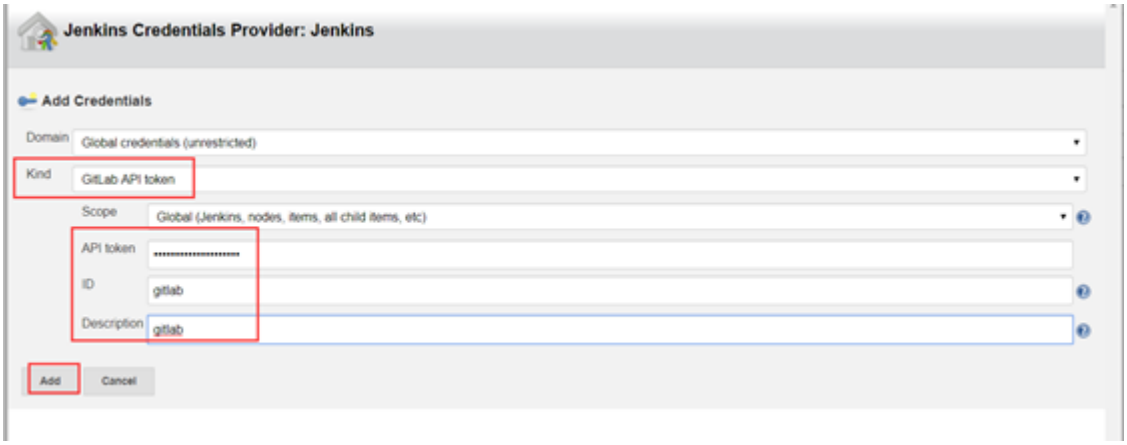




将获得的 token 值进行复制备用




github怎么删除仓库_百度搜索
Integrations · Settings · OPS /
Files · dev · OPS / git_test · Git
不安全 | 10.0.0.201:8080/manage
Jenkins
Script Security sandbox bypass
Maven Integration plugin 3.0:
Sensitive values in module build logs not masked
Go to plugin manager
Configure which of these warnings are shown
系统设置
全局安全配置
Jenkins安全，定义谁可以访问或使用系统。
Gitlab
Enable authentication for '/project' end-point
GitLab connections
Connection name
gitlab
Gitlab host URL
http://10.0.0.100
Credentials
- none -
Add
API Token
Jenkins
required
Test Connection

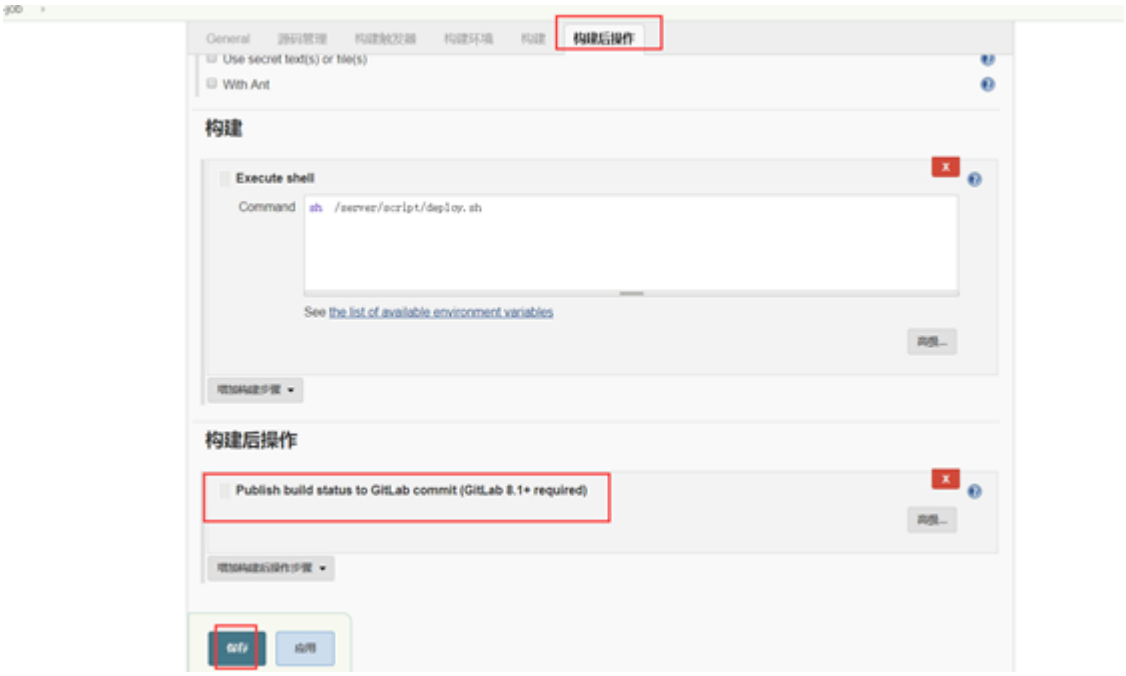

Jenkins Credentials Provider: Jenkins
Add Credentials
Domain
Global credentials (unrestricted)
Kind
GitLab API token
Scope
Global (Jenkins, nodes, items, all child items, etc)
API token
ID
gitlab
Description
gitlab
Add
Cancel


SSH sites that projects will want to connect
Usage Statistics
Help make Jenkins better by sending anonymous usage statistics and crash reports to the Jenkins project
Gitlab
Enable authentication for '/project' end-point
GitLab connections
Connection name
gitlab
A name for the connection
Gitlab host URL
http://10.0.0.100
The complete URL to the Gitlab server (i.e. http://gitlab.org)
Credentials
GitLab API token (gitlab)
Add
API Token for accessing Gitlab

进行测试

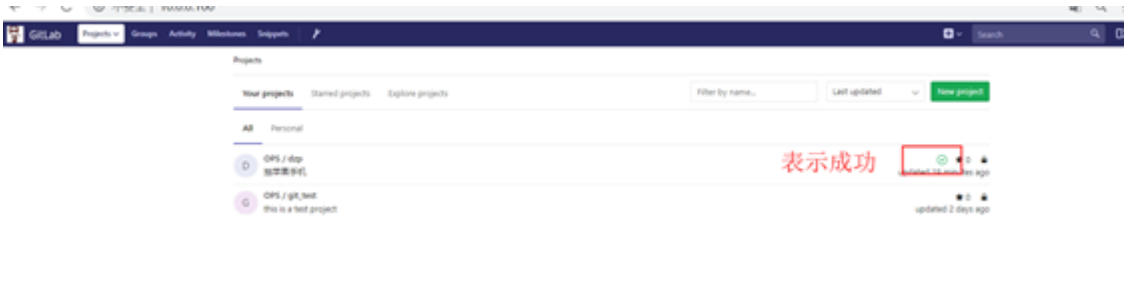

Usage Statistics
Help make Jenkins better by sending anonymous usage statistics and crash reports to the Jenkins project
Gitlab
Enable authentication for '/project' end-point
GitLab connections
Connection name
gitlab
Gitlab host URL
http://10.0.0.100
Credentials
GitLab API token (gitlab)
Add
测试
Success
Test Connection

添加构建后操作


General
构建后操作
Use secret text(s) or file(s)
With Ant
构建
Execute shell
Command
sh /server/script/deploy.sh
See the list of available environment variables
构建后操作
Publish build status to GitLab commit (GitLab 8.1+ required)
Gitlab
应用


不安全 | 10.0.0.201:8080/job/freestyle-job/
Jenkins
Jenkins
freestyle-job
返回面板
状态
修改记录
工作空间
立即构建
删除 Project
配置
Favorite
Open Blue Ocean
Build History
构建历史
Project freestyle-job
工作区
最新修改记录
相关连接
Last build(#8), 17 分之前
Last stable build(#8), 17 分之前
Last successful build(#8), 17 分之前

检查结果



10. 创建 Maven 项目

Maven 是一个项目管理和综合工具。Maven 提供给开发人员构建一个完整的生命周期框架。

开发团队可以自动完成该项目的基础设施建设，Maven 使用标准的目录结构和默认构建生命周期。

Apache 的开源项目主要服务于 Java 平台的构建、依赖管理、项目管理。

Project Object Model，项目对象模型。通过 xml 格式保存的 pom.xml 文件。该文件用于管理：源代码、配置文

件、开发者的信息和角色、问题追踪系统、组织信息、项目授权、项目的 url、项目的依赖关系等等。该文件是由开发维护，我们运维人员可以不用去关心。

mvn package #会去 maven 的中央仓库去下载需要的依赖包和插件到.m2 目录下

5、创建 Maven 私服 nexus

部署私服 xenus 下载 https://www.sonatype.com/download - oss - sonatype
配置仓库两个选项
1、项目下的 pom.xml 配置、只生效当前的项目
2、在 maven 配置全局所有项目生效
上传 JDK 和 nexus 安装包
rpm - ivh jdk - 8u121 - linux - x64.rpm
mv nexus - 3.13.0 - 01 /usr/local/
ln - s /usr/local/nexus - 3.13.0 - 01 /usr/local/nexus
/usr/local/nexus/bin/nexus start
10.0.0.202:8081 admin admin123
配置 Maven 全局配置文件
/usr/local/maven/conf/settings.xml
6.创建一个 Maven 项目
创建前上传代码到 gitlab 服务器，把 java 项目添加到 gitlab 仓库中

01. 部署 Maven

官网： http://maven.apache.org/download.cgi
清华镜像： https://mirrors.tuna.tsinghua.edu.cn/apache/maven/

#上传软件包
[root@jenkins ~]# ll
-rw-r--r-- 1 root root 8491533 2018-08-27 14:38 apache-maven-3.3.9-bin.tar.gz

#解压
[root@jenkins ~]# tar xf apache-maven-3.3.9-bin.tar.gz

#改变目录位置
[root@jenkins maven]# mv apache-maven-3.3.9 /usr/local/maven-3.3.9

#软连接
[root@jenkins maven]# ln -s /usr/local/maven-3.3.9 /usr/local/maven

[root@jenkins ~]# cd /usr/local/maven
[root@jenkins maven]# ll
total 32
drwxr-xr-x 2 root root 97 2019-11-20 21:55 bin
drwxr-xr-x 2 root root 42 2019-11-20 21:55 boot
drwxr-xr-x 3 root root 63 2015-11-11 00:38 conf
drwxr-xr-x 3 root root 4096 2019-11-20 21:55 lib
-rw-r--r-- 1 root root 19335 2015-11-11 00:44 LICENSE
-rw-r--r-- 1 root root 182 2015-11-11 00:44 NOTICE
-rw-r--r-- 1 root root 2541 2015-11-11 00:38 README.txt

#设置环境变量

[root@jenkins maven]# echo "export PATH=/usr/local/maven/bin/:$PATH" >>/etc/profile
[root@jenkins maven]# source /etc/profile

#查看结果
[root@jenkins maven]# mvn -v
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-11T00:41:47+08:00)
Maven home: /usr/local/maven
Java version: 1.8.0_181, vendor: Oracle Corporation
Java home: /usr/java/jdk1.8.0_181-amd64/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "3.10.0-957.el7.x86_64", arch: "amd64", family: "unix"

02. 编译测试

#上传一个简单的 java 项目包 hello - world.tar.gz 进行解压

[root@jenkins ~]# ll
-rw-r--r-- 1 root root 1325 2018-08-22 13:43 hello-world.tar.gz

[root@jenkins ~]# tar xf hello-world.tar.gz
[root@jenkins ~]# cd hello-world/

validate（验证）：　验证项目正确，并且所有必要信息可用。
compile（编译）：　编译项目源码
test（测试）：　　使用合适的单元测试框架测试编译后的源码。
package（打包）：　源码编译之后，使用合适的格式（例如 JAR 格式）对编译后的源码进行打包。
integration - test（集成测试）: 如果有需要，把包处理并部署到可以运行集成测试的环境中去。
verify（验证）：　进行各种测试来验证包是否有效并且符合质量标准。
install（安装）：　把包安装到本地仓库，使该包可以作为其他本地项目的依赖。
deploy（部署）：　在集成或发布环境中完成，将最终软件包复制到远程存储库，以与其他开发人员和项目共享。
mvn clean (清除) : 清除上次编译的结果

#测试
[root@jenkins hello-world]# mvn test

#打包
[root@jenkins hello-world]# mvn package　#会去 maven 的中央仓库去下载需要的依赖包和插件到.m2 目录下

#打包结果
[root@jenkins hello-world]# ll target/
total 8
drwxr-xr-x 3 root root 17 2019-11-20 22:13 classes
-rw-r--r-- 1 root root 3130 2019-11-20 23:39 hello-world-1.0-SNAPSHOT.jar

drwxr-xr-x 2 root root 28 2019-11-20 23:38 maven-archiver
drwxr-xr-x 3 root root 35 2019-11-20 22:13 maven-status
-rw-r--r-- 1 root root 2872 2019-11-20 23:38 original-hello-world-1.0-SNAPSHOT.jar
drwxr-xr-x 2 root root 125 2019-11-20 22:14 surefire-reports
drwxr-xr-x 3 root root 17 2019-11-20 22:13 test-classes

03. 部署 Tomcat 及数据库

#上传压缩包
[root@tomcat ~]# ll
-rw-r--r-- 1 root root 9128610 Mar 27 2019 apache-tomcat-8.0.27.tar.gz
-rw-r--r-- 1 root root 170023183 Aug 14 2018 jdk-8u181-linux-x64.rpm

#安装 JDK
[root@tomcat ~]# rpm -ivh jdk-8u181-linux-x64.rpm
warning: jdk-8u181-linux-x64.rpm: Header V3 RSA/SHA256 Signature, key ID ec551f03: NOKEY
Preparing... ################################# [100%]
Updating / installing...
1:jdk1.8-2000:1.8.0_181-fcs ################################# [100%]
Unpacking JAR files...
tools.jar...
plugin.jar...
javaws.jar...
deploy.jar...
rt.jar...
jsse.jar...
charsets.jar...
localedata.jar...

#解压 Tomcat
[root@tomcat ~]# mkdir /application
[root@tomcat ~]# tar xf apache-tomcat-8.0.27.tar.gz -C /application

#创建软连接
[root@tomcat ~]# ln -s /application/apache-tomcat-8.0.27 /application/tomcat

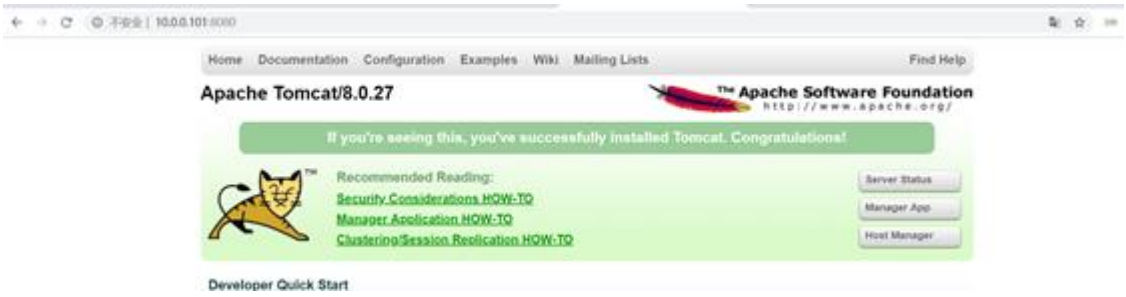
#tomcat 启动加速的方法
[root@tomcat ~]# vim /usr/java/jdk1.8.0_181-amd64/jre/lib/security/java.security
117 securerandom.source=file:/dev/urandom #修改之后

#启动 tomcat
[root@tomcat ~]# /application/tomcat/bin/startup.sh
Using CATALINA_BASE: /application/tomcat
Using CATALINA_HOME: /application/tomcat
Using CATALINA_TMPDIR: /application/tomcat/temp

Using JRE_HOME: /usr
Using CLASSPATH: /application/tomcat/bin/bootstrap.jar:/application/tomcat/bin/tomcat-juli.jar
Tomcat started.

#检查端口
[root@tomcat ~]# netstat -lntp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address Foreign Address State PID/Program name
tcp 0 0 0.0.0.0:22 0.0.0.0:* LISTEN 6758/sshd
tcp 0 0 127.0.0.1:25 0.0.0.0:* LISTEN 7134/master
tcp6 0 0 :::8009 :::* LISTEN 24383/java
tcp6 0 0 :::8080 :::* LISTEN 24383/java
tcp6 0 0 :::22 :::* LISTEN 6758/sshd
tcp6 0 0 ::1:25 :::* LISTEN 7134/master
tcp6 0 0 127.0.0.1:8005 :::* LISTEN 24383/java

#浏览器使用 8080 端口访问



#为项目准备好数据库 jeesns，设置数据库 root 用户密码为 root

#安装数据库
[root@tomcat ~]# yum install mariadb-server -y

[root@tomcat ~]# systemctl start mariadb.service
[root@tomcat ~]# mysqladmin password 'root'

#创建 jeesns 库
[root@tomcat ~]# mysql -uroot -proot -e "create database jeesns;"

#dev 上面上传一个项目
[root@git ~]# ll
-rw-r--r-- 1 root root 15376795 2019-03-27 17:33 jeesns.tar.gz
[root@git ~]# tar xf jeesns.tar.gz
[root@git ~]# cd jeesns/
[root@git ~/jeesns]# git remote
origin

```
[root@git ~/jeesns]# git remote remove origin
[root@git ~/jeesns]# ll jeesns-web/database/
total 40
-rwxr-xr-x 1 root root 28667 2018-11-19 15:01 jeesns.sql
-rw-r--r-- 1 root root 3491 2018-11-19 15:01 update_1.2.0to1.2.1.sql
-rw-r--r-- 1 root root 1026 2018-11-19 15:01 update_1.2.1to1.3.sql
-rw-r--r-- 1 root root 1344 2018-11-19 15:01 update_1.3to1.3.1.sql

#将该数据库传输到 tomcat 节点进行导入
[root@git ~/jeesns]# scp jeesns-web/database/jeesns.sql root@10.0.0.80:~

#导入数据库

[root@tomcat ~]# mysql -uroot -proot jeesns < jeesns.sql
[root@tomcat ~]# mysql -uroot -proot -e "use jeesns; show tables"
+--------------------------+
| Tables_in_jeesns |
+--------------------------+
| tbl_action |
| tbl_action_log |
| tbl_ads |
| tbl_archive |
| tbl_archive_favor |
| tbl_article |
| tbl_article_cate |
| tbl_article_comment |
| tbl_checkin |
| tbl_config |
| tbl_group |
| tbl_group_fans |
| tbl_group_topic |
| tbl_group_topic_comment |
| tbl_group_topic_type |
| tbl_group_type |
| tbl_link |
| tbl_member |
| tbl_member_fans |
| tbl_member_level |
| tbl_member_token |
| tbl_memgroup |
| tbl_message |
| tbl_picture |
| tbl_picture_album |
| tbl_picture_album_comment |
```

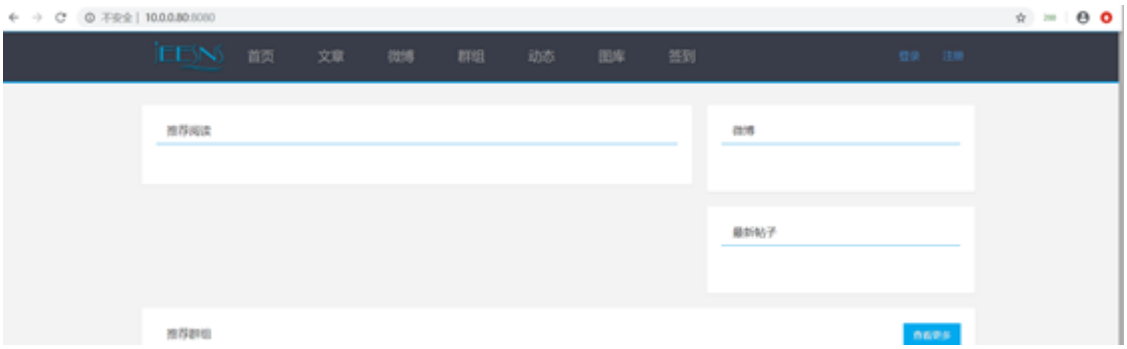
| tbl_picture_album_favor |
| tbl_picture_comment |
| tbl_picture_favor |
| tbl_picture_tag |
| tbl_score_detail |
| tbl_score_rule |
| tbl_tag |
| tbl_validate_code |
| tbl_weibo |
| tbl_weibo_comment |
| tbl_weibo_favor |
| tbl_weibo_topic |
+---------------------------+

#打包 jeesns 项目
[root@git ~/jeesns]# mvn package

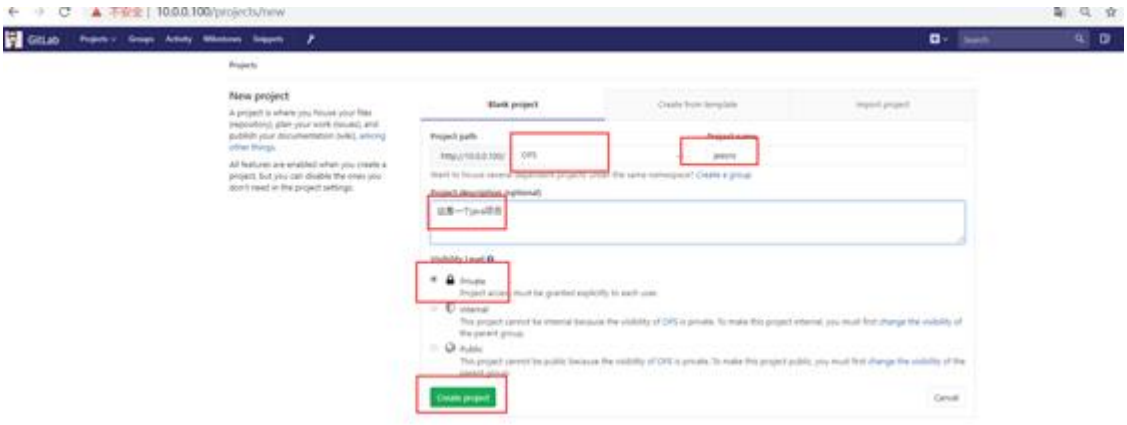
#打包之后生成的 war 包
[root@git ~/jeesns]# ll jeesns-web/target/
total 25496
drwxr-xr-x 4 root root 181 2019-11-20 23:19 classes
drwxr-xr-x 3 root root 25 2019-11-20 23:19 generated-sources
drwxr-xr-x 5 root root 104 2019-11-20 23:19 jeesns-web
-rw-r--r-- 1 root root 26106028 2019-11-20 23:20 jeesns-web.war
drwxr-xr-x 2 root root 28 2019-11-20 23:19 maven-archiver
drwxr-xr-x 3 root root 35 2019-11-20 23:19 maven-status

#将其手动传输到 tomcat 节点上面测试
[root@git ~/jeesns]# scp jeesns-web/target/jeesns-web.war root@10.0.0.80:/application/tomcat/webapps/ROOT.war

#浏览器刷新测试



04. 创建一个 jeesns 项目



#清除上次编译的结果

[root@git ~/jeesns]# mvn clean

[root@git ~/jeesns]# git remote remove origin

[root@git ~/jeesns]#

[root@git ~/jeesns]#

[root@git ~/jeesns]# git remote add origin git@10.0.0.100:OPS/jeesns.git

[root@git ~/jeesns]# git add .

[root@git ~/jeesns]# git commit -m "Initial commit"

On branch master

nothing to commit, working directory clean

[root@git ~/jeesns]# git push -u origin master

Counting objects: 1946, done.

Compressing objects: 100% (1862/1862), done.

Writing objects: 100% (1946/1946), 7.09 MiB | 7.25 MiB/s, done.

Total 1946 (delta 285), reused 0 (delta 0)

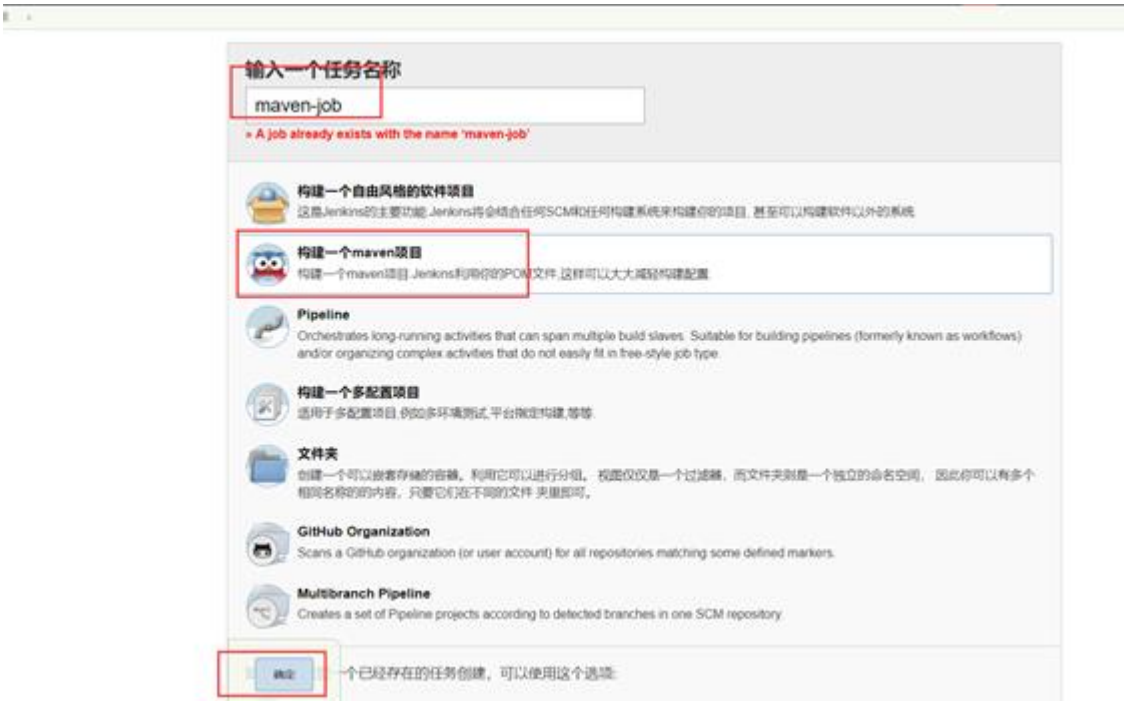
remote: Resolving deltas: 100% (285/285), done.

To git@10.0.0.100:OPS/jeesns.git

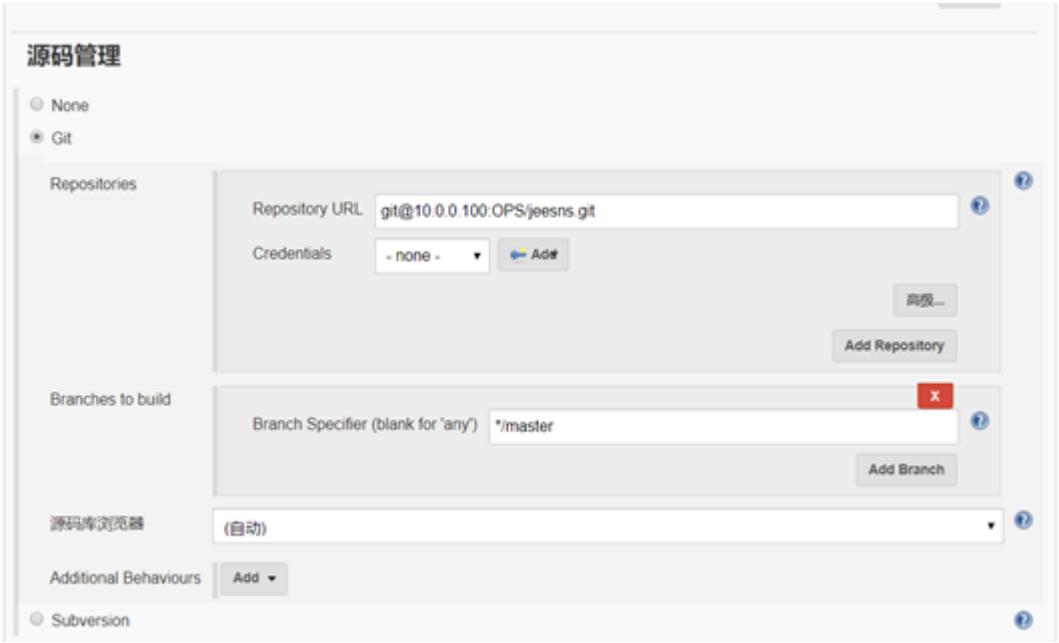
* [new branch] master -> master

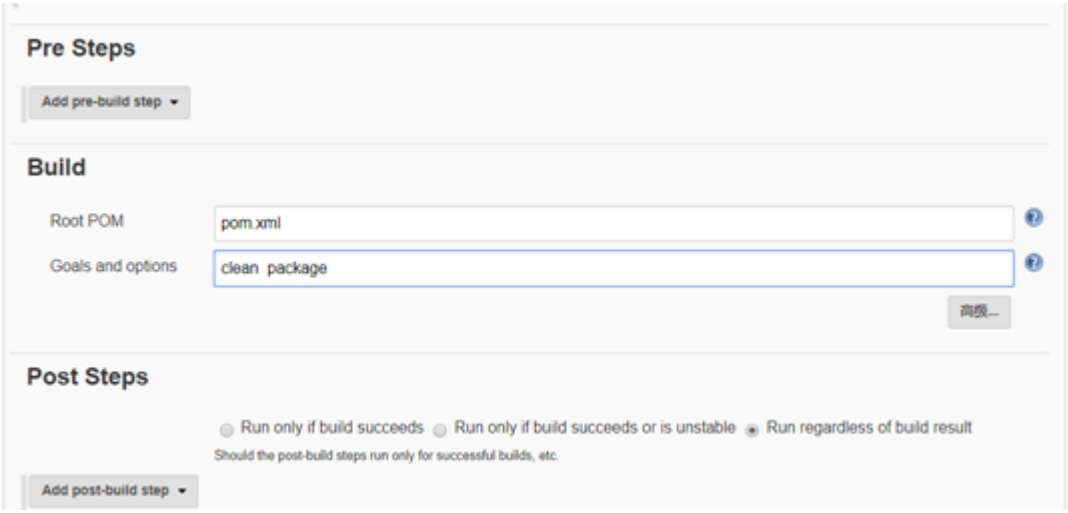
Branch master set up to track remote branch master from origin.

05. Jenkins 创建一个 maven



#此处如果出现报错，请添加 Deploy Keys





#发现已经打包成功了

[root@jenkins ~]# ll /var/lib/jenkins/workspace/maven-job/jeesns-web/target/jeesns-web.war
-rw-r--r-- 1 root root 26106007 2019-11-23 11:49 /var/lib/jenkins/workspace/maven-job/jeesns-web/target/jeesns-web.war

#jenkins 用户给 tomcat 节点进行分发公钥

[root@jenkins ~]# ssh-copy-id -i .ssh/id_rsa.pub root@10.0.0.80

#构建后操作

ssh root@10.0.0.80 "mv /application/tomcat/webapps/ROOT.war /tmp/$BUILD_ID-ROOT.war"

scp /var/lib/jenkins/workspace/maven-job/jeesns-web/target/jeesns-web.war root@10.0.0.80:/application/tomcat/webapps/ROOT.war

[root@tomcat ~]# ll /tmp/

total 50992

-rw-r--r-- 1 root root 26106004 2019-11-23 01:28 2-ROOT.war

11. Pipeline 项目

01. 基础概念

CI/CD 持续集成/持续部署

持续集成(Continuous integration)是一种软件开发实践，即团队开发成员经常集成它们的工作，通过每个成员

每天至少集成一次，也就意味着每天可能会发生多次集成。每次集成都通过自动化的构建（包括编译，发布，自动

化测试）来验证，从而尽早地发现集成错误。

比如（你家装修厨房，其中一项是铺地砖，边角地砖要切割大小。如果一次全切割完再铺上去，发现尺寸有误的话

浪费和返工时间就大了，不如切一块铺一块。这就是持续集成。）

持续部署（continuous deployment）是通过自动化的构建、测试和部署循环来快速交付高质量的产品。某种程度上代表了一个开发团队工程化的程度，毕竟快速运转的互联网公司人力成本会高于机器，投资机器优化开发流程化

相对也提高了人的效率。

比如（装修厨房有很多部分，每个部分都有检测手段，如地砖铺完了要测试漏水与否，线路铺完了要通电测试电路

通顺，水管装好了也要测试冷水热水。如果全部装完了再测，出现问题可能会互相影响，比如电路不行可能要把地

砖给挖开……。那么每完成一部分就测试，这是持续部署。）

持续交付 Continuous Delivery:频繁地将软件的新版本，交付给质量团队或者用户，以供评审尽早发现生产环境中存在的问题；如果评审通过，代码就进入生产阶段.

比如（全部装修完了，你去验收，发现地砖颜色不合意，水池太小，灶台位置不对，返工吗？所以不如没完成一部

分，你就去用一下试用验收，这就是持续交付。）

敏捷思想中提出的这三个观点，还强调一件事：通过技术手段自动化这三个工作。加快交付速度。

1.什么是 pipeline

Jenkins 2.0 的精髓是 Pipeline as Code，是帮助 Jenkins 实现 CI 到 CD 转变的重要角色。什么是 Pipeline，简单来说，就是一套运行于 Jenkins 上的工作流框架，将原本独立运行于单个或者多个节点的任务连接起来，实现单个

任务难以完成的复杂发布流程。Pipeline 的实现方式是一套 Groovy DSL，任何发布流程都可以表述为一段 Groovy
脚本，并且 Jenkins 支持从代码库直接读取脚本，从而实现了 Pipeline as Code 的理念。

2.Pipeline 概念

Pipeline 是一个用户定义的 CD 流水线模式。Pipeline 代码定义了通常包含构建、测试和发布步骤的完整的构

建过程。

Node
node 是一个机器，它是 Jenkins 环境的一部分，并且能够执行 Pipeline。同时，node 代码块也是脚本式

Pipeline 语法的关键特性。

Stage
Stage 块定义了在整个 Pipeline 中执行的概念上不同的任务子集（例如"构建"，"测试"和"部署"阶段），

许多插件使用它来可视化或呈现 Jenkins 管道状态/进度。

Step

一项任务。从根本上讲，一个步骤告诉 Jenkins 在特定时间点（或过程中的"步骤"）要做什么。例如，使用

sh step：sh 'make' 可以执行 make 这个 shell 命令。

3.jenkins file

声明式

脚本式

脚本式语法格式：

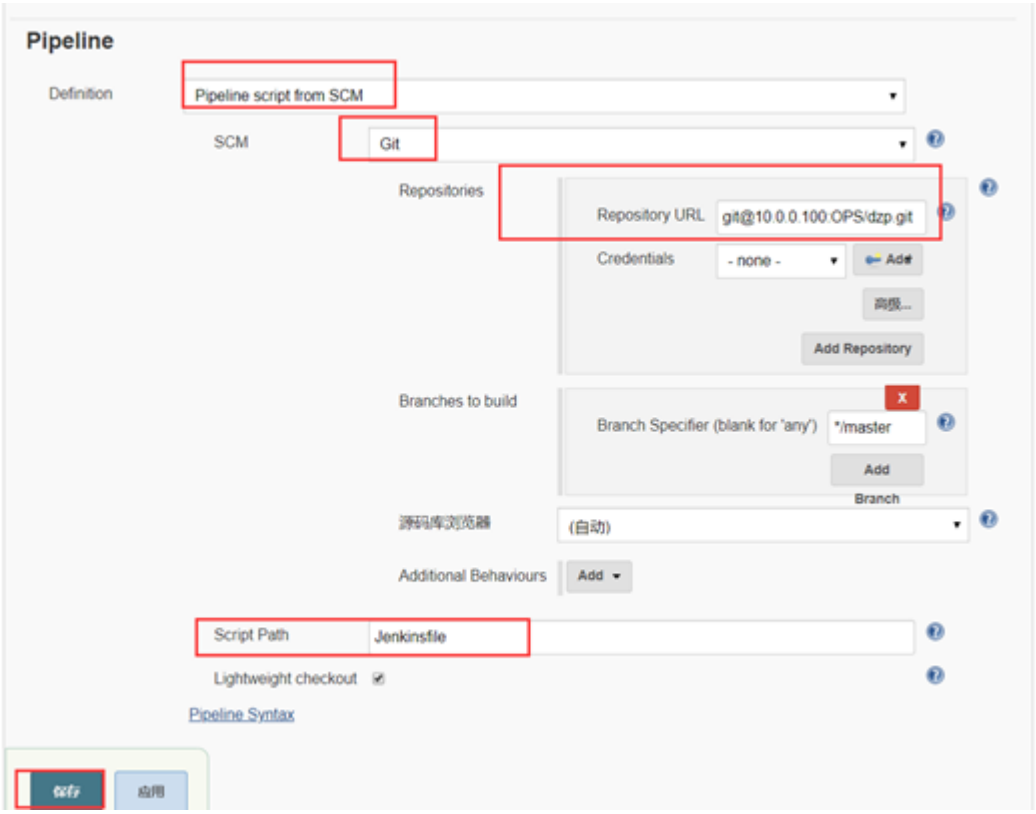
```
pipeline{
   agent any
stages{
   stage("get code"){
      steps{
         echo "get code from scm"
      }
   }
   stage("package"){
      steps{
      echo "packge code"
      }
   }
   stage("deploy"){
      steps{
         echo "deploy packge to node1"
      }
   }
   }
}
```

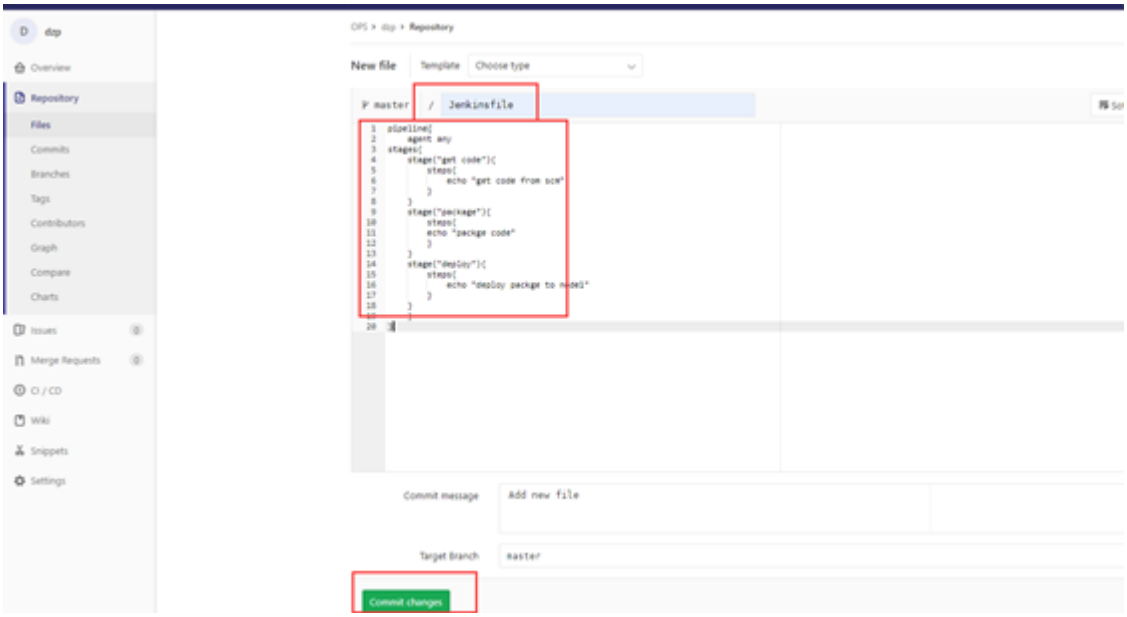
02. 创建 pipeline 项目


输入一个任务名称
pipeline-job
构建一个自由风格的软件项目
这是Jenkins的主要功能 Jenkins将会结合任何SCM和任何构建系统来构建你的项目, 甚至可以构建软件以外的系统
构建一个maven项目
构建一个maven项目 Jenkins利用你的POM文件,这样可以大大减轻构建配置
Pipeline
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
构建一个多配置项目
适用于多配置项目,例如多环境测试,平台指定构建,等等
文件夹
创建一个可以嵌套存储的容器,利用它可以进行分组。视图仅仅是一个过滤器,而文件夹则是一个独立的命名空间,因此你可以有多个相同名称的的内容,只要它们在不同的文件夹里即可。
GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.
Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.
确定
一个已经存在的任务创建，可以使用这个选项:


Pipeline
Definition
Pipeline script
Script
try sample Pipeline...
1 pipeline{
2 agent any
3 stages{
4 stage("get code"){
5 steps{
6 echo "get code from scm"
7 }
8 }
9 stage("package"){
10 steps{
11 echo "packge code"
12 }
13 }
14 stage("deploy"){
15 steps{
16 echo "deploy packge to node1"
17 }
18 }
Use Groovy Sandbox
Pipeline Syntax
保存
应用

在仓库创建一个 **Jenkinsfile** 文件进行调用

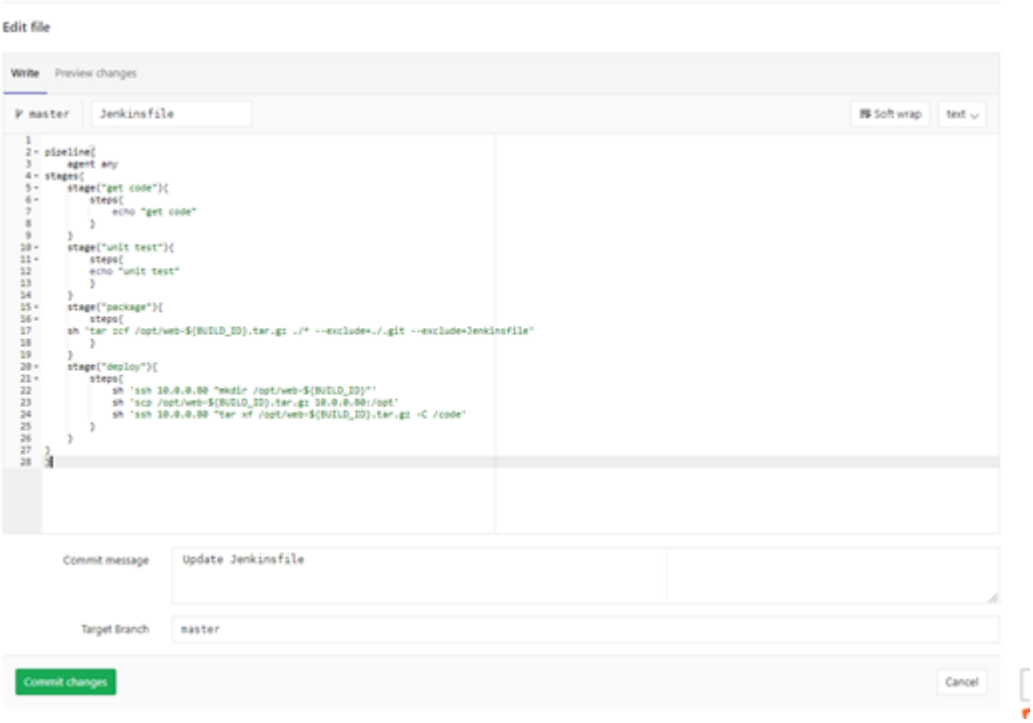




编辑 Jenkinsfile 文件

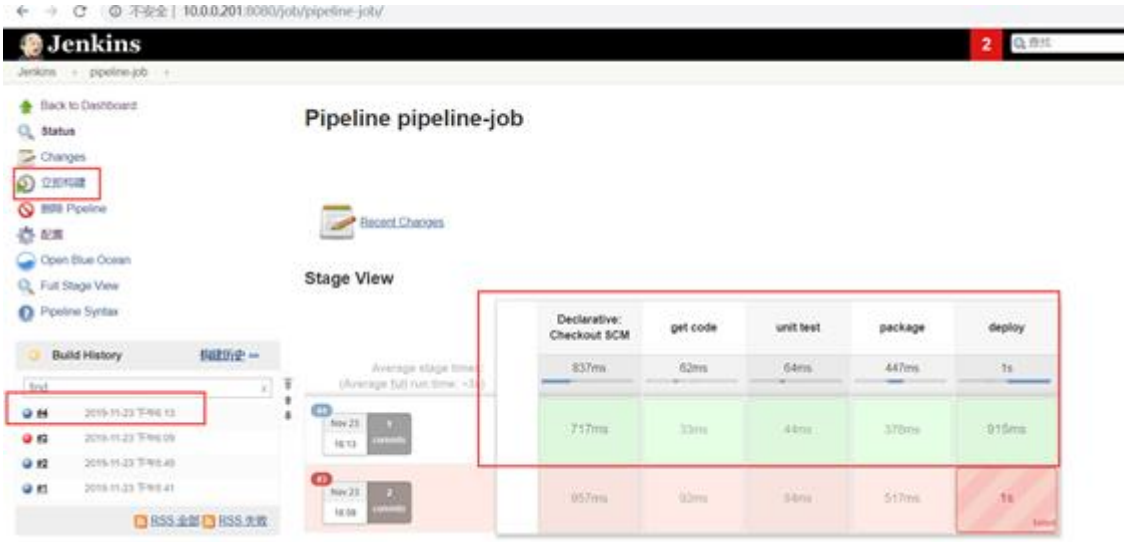
```
pipeline{
    agent any
stages{
    stage("get code"){
        steps{
            echo "get code"
        }
    }
    stage("unit test"){
        steps{
        echo "unit test"
        }
    }
```

```
    stage("package"){
        steps{
    sh 'tar zcf /opt/web-${BUILD_ID}.tar.gz ./* --exclude=./.git --exclude=Jenkinsfile'
        }
    }
    stage("deploy"){
        steps{
            sh 'ssh 10.0.0.80 "mkdir /opt/web-${BUILD_ID}"'
            sh 'scp /opt/web-${BUILD_ID}.tar.gz 10.0.0.80:/opt'
            sh 'ssh 10.0.0.80 "tar xf /opt/web-${BUILD_ID}.tar.gz -C /code"'
        }
    }
}
}
```



执行构建报错

修改脚本再次构建



人的一生或多或少都在为一些事情努力着，有目标的人生是精彩的！朋友请不要在原地画圈圈，该行动了！