

Git学习教程

一、Git 入门

1. 安装git

- linux

输入git查看是否安装，如果没有安装的话会出现以下提示

```
wy@ubuntu:~$ git

Command 'git' not found, but can be installed with:

sudo apt install git

wy@ubuntu:~$
```

输入提示信息：sudo apt install git 即可安装

- windows

去git官网[下载](#)即可， 安装完成后，在开始菜单里找到“Git”->“Git Bash”



进入后一般为默认地址~，可通过修改桌面快捷方式的起始位置来修改该地址为git本地仓库地址。

输入以下命令配置你的git

```
1 $ git config --global user.name "Your github Name"
2
3 $ git config --global user.email "your github email"
```

2. 创建版本库

创建属于自己的git本地仓库(repository)目录,所有的命令将在该仓库目录中使用，当然你可以自己指定任意一个目录。

在本地仓库目录中创建版本库：

进入 repository 目录：

I 创建一个目录作为仓库的目录

```
$ mkdir Hutool
```

I 进入该目录, 初始化

```
1 $ git init
2
3 Initialized empty Git repository in D:/Git/repository/Hutool/.git/
```

提示一个空的仓库 (empty Git repository) 已经被创建

```
Administrator@WY MINGW64 /repository
$ mkdir Hutool

Administrator@WY MINGW64 /repository
$ ll
total 4
drwxr-xr-x 1 Administrator 197121 0 4月 22 09:08 c3p0/
drwxr-xr-x 1 Administrator 197121 0 4月 22 20:31 groupProject/
drwxr-xr-x 1 Administrator 197121 0 4月 22 09:01 HelloWorld/
drwxr-xr-x 1 Administrator 197121 0 4月 23 13:56 Hutool/

Administrator@WY MINGW64 /repository
$ cd Hutool/

Administrator@WY MINGW64 /repository/Hutool
$ git init
Initialized empty Git repository in D:/Git/repository/Hutool/.git/
```

3. 添加文件到版本库

我们把需要添加到版本库的文件放到版本库的目录 (或者子目录), 此处操作readme.txt文件

电脑 > 软件 (D:) > Git > repository > Hutool			
名称	修改日期	类型	
.git	2019/4/23 13:57	文件夹	
markdown.md	2019/4/22 15:07	MD 文件	
readme.txt	2019/4/23 14:05	文本文档	

将文件添加到版本库

```
$ git add readme.txt
```

将文件提交到提交到仓库

```
$ git commit -m "readme first commit"
```

git commit命令, -m后面输入的是本次提交的说明

```
Administrator@WY MINGW64 /repository/Hutool (master)
$ git add readme.txt

Administrator@WY MINGW64 /repository/Hutool (master)
$ git commit -m "readme first commit"
[master (root-commit) a4637ce] readme first commit
1 file changed, 2 insertions(+)
create mode 100644 readme.txt
```

4. 版本回退

对readme.txt文件进行修改，并进行提交

```
Administrator@WY MINGW64 /repository/Hutool (master)
$ git add readme.txt

Administrator@WY MINGW64 /repository/Hutool (master)
$ git commit -m "append GPL"
[master d9b25db] append GPL
1 file changed, 2 insertions(+), 2 deletions(-)
```

使用 `$ git log` 显示的提交日志（按提交时间）

```
commit d9b25dbba20f203a72b9621a7479b14121c0da10
```

commit后为提交时的commit_id

```
Administrator@WY MINGW64 /repository/Hutool (master)
$ git log
commit d9b25dbba20f203a72b9621a7479b14121c0da10 (HEAD -> master)
Author: laoxuai <laoxuai@aliyun.com>
Date: Tue Apr 23 14:15:11 2019 +0800

    append GPL

commit a4637ce2e353278d52028354ea1006dec54506f3
Author: laoxuai <laoxuai@aliyun.com>
Date: Tue Apr 23 14:09:22 2019 +0800

    readme first commit
```

HEAD表示当前版本，也就是最新的提交d9b25...，上一个版本就是 `HEAD^`，上上一个版本就是 `HEAD^^`，上100个版本可以使用 `HEAD~100`

使用 `git reflog` 用来记录你的每一次命令

使用 `git reset --hard commit_id` 可以回退到指定版本

```

Administrator@WY MINGW64 /repository/Hutool (master)
$ git reset --hard HEAD^
HEAD is now at a4637ce readme first commit

Administrator@WY MINGW64 /repository/Hutool (master)
$ cat readme.txt
Git is a version control system.
Git is free software.
Administrator@WY MINGW64 /repository/Hutool (master)
$ git log
commit a4637ce2e353278d52028354ea1006dec54506f3 (HEAD -> master)
Author: laoxuai <laoxuai@aliyun.com>
Date: Tue Apr 23 14:09:22 2019 +0800

    readme first commit

Administrator@WY MINGW64 /repository/Hutool (master)
$ git reflog
a4637ce (HEAD -> master) HEAD@{0}: reset: moving to HEAD^
d9b25db HEAD@{1}: commit: append GPL
a4637ce (HEAD -> master) HEAD@{2}: commit (initial): readme first commit

Administrator@WY MINGW64 /repository/Hutool (master)
$ git reset --hard a4637ce
HEAD is now at a4637ce readme first commit

Administrator@WY MINGW64 /repository/Hutool (master)
$ git reset --hard d9b25db
HEAD is now at d9b25db append GPL

Administrator@WY MINGW64 /repository/Hutool (master)
$ cat readme.txt
Git is a distributed version control system.
Git is free software distributed under the GPL.

```

5. 暂存区和工作区

此电脑 > 软件 (D:) > Git > repository > Hutool

名称	修改日期	类型
.git	2019/4/23 14:28	文件夹
markdown.md	2019/4/22 15:07	MD 文件
readme.txt	2019/4/23 14:28	文本文档

工作区

电脑 > 软件 (D:) > Git > repository > Hutool > .git

名称	修改日期	类型
hooks	2019/4/23 13:57	文件夹
info	2019/4/23 13:57	文件夹
logs	2019/4/23 14:09	文件夹
objects	2019/4/23 14:15	文件夹
refs	2019/4/23 13:57	文件夹
COMMIT_EDITMSG	2019/4/23 14:15	文件
config	2019/4/23 13:57	文件
description	2019/4/23 13:57	文件
HEAD	2019/4/23 13:57	文件
index	2019/4/23 14:28	文件
ORIG_HEAD	2019/4/23 14:28	文件

指向master的指针

暂存区

git add把文件添加进去，实际上就是把文件修改添加到暂存区；

git commit提交更改，实际上就是把暂存区的所有内容提交到当前分支

使用 `git status` 查看状态

```

Administrator@WY MINGW64 /repository/Hutool (master)
$ 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:   readme.txt

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

        LICENSE.txt
        markdown.md

no changes added to commit (use "git add" and/or "git commit -a")

```

可以看出readme.txt发生了修改但是没有提交，LICENSE.txt和markdown.md没有被添加过，显示为Untracked状态。使用 `git add` 添加后查看当前状态

```

Administrator@WY MINGW64 /repository/Hutool (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   LICENSE.txt
        new file:   markdown.md
        modified:   readme.txt

```

`git add`命令实际上就是把要提交的所有修改放到暂存区，执行`git commit`就可以一次性把暂存区的所有修改提交到分支

```

Administrator@WY MINGW64 /repository/Hutool (master)
$ git commit -m "add LICENSE"
[master e2a882b] add LICENSE
3 files changed, 235 insertions(+), 1 deletion(-)
create mode 100644 LICENSE.txt
create mode 100644 markdown.md

```

6. 管理修改

第一次修改后添加到暂存区

```

Administrator@WY MINGW64 /repository/Hutool (master)
$ git add readme.txt

Administrator@WY MINGW64 /repository/Hutool (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   readme.txt

```

再次修改readme.txt文件，然后直接提交

```

Administrator@WY MINGW64 /repository/Hutool (master)
$ git commit -m "git tracks changes"
[master calcf91] git tracks changes
1 file changed, 2 insertions(+), 1 deletion(-)

Administrator@WY MINGW64 /repository/Hutool (master)
$ 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:   readme.txt

no changes added to commit (use "git add" and/or "git commit -a")

```

可以发现，第二次修改并没有提交到分支，使用 `$ git diff HEAD -- readme.txt`，可以比较工作区和版本库里面最新版本的区别，第二次修改并没有进行提交。

```

Administrator@WY MINGW64 /repository/Hutool (master)
$ git diff HEAD -- readme.txt
diff --git a/readme.txt b/readme.txt
index db28b2c..295b239 100644
--- a/readme.txt
+++ b/readme.txt
@@ -1,4 +1,5 @@
 Git is a distributed version control system.
 Git is free software distributed under the GPL.
 Git has a mutable index called stage.
-Git tracks changes.
\ No newline at end of file
+Git tracks changes.
+Git tracks changes of files.
\ No newline at end of file

```

当修改后使用 `git add` 将文件添加到暂存区，才能在使用 `git commit` 时提交到版本库中

7. 撤销修改

```

Administrator@WY MINGW64 /repository/Hutool (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   readme.txt

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:   readme.txt

Administrator@WY MINGW64 /repository/Hutool (master)
$ git checkout -- readme.txt

Administrator@WY MINGW64 /repository/Hutool (master)
$ cat readme.txt
Git is a distributed version control system.
Git is free software distributed under the GPL.
Git has a mutable index called stage.
Git tracks changes.
Git tracks changes of files.

```

当前文件还未提交到暂存区，使用 `$ git checkout -- readme.txt` 可以把readme.txt文件在工作区的修改全部撤销，回退到最近一次git commit或git add时的状态。

8. 删除文件

使用rm xxx 可以直接删除工作区的文件，查看现在的状态，显示工作区LICENSE.txt文件已经被删除，此时如果需要恢复的话可以使用 `$ git checkout -- LICENSE.txt` 进行恢复（实际上是将分支上的最新提交的文件恢复到工作区，也就是说最新提交前的针对该文件的所有更改将失效）

```
Administrator@WY MINGW64 /repository/Hutool (master)
$ rm LICENSE.txt

Administrator@WY MINGW64 /repository/Hutool (master)
$ ll
total 9
-rw-r--r-- 1 Administrator 197121 7261 4月 22 15:07 markdown.md
-rw-r--r-- 1 Administrator 197121 183 4月 23 15:21 readme.txt

Administrator@WY MINGW64 /repository/Hutool (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   readme.txt

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:    LICENSE.txt

Administrator@WY MINGW64 /repository/Hutool (master)
$ git rm LICENSE.txt
rm 'LICENSE.txt'

Administrator@WY MINGW64 /repository/Hutool (master)
$ git commit -m "remove LICENSE.txt"
[master 133807d] remove LICENSE.txt
2 files changed, 2 insertions(+), 1 deletion(-)
delete mode 100644 LICENSE.txt
```



使用 `$ git rm LICENSE.txt` , `$ git commit -m "remove LICENSE.txt"` 两个命令将删除版本库中的LICENSE.txt文件

二、 远程仓库和分支、标签管理

1. 添加远程仓库


- github上创建一个远程仓


Owner Repository name *

 laoxuai /  仓库名称




Great repository names are short and memorable. Need inspiration? How about bookish-giggle?

Description (optional)

☒  Public
Anyone can see this repository. You choose who can commit.

☐  Private
You choose who can see and commit to this repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None  | Add a license: None  

[Create repository](#)

使用命令 `git remote add origin git@server-name:path/repo-name.git` 关联一个远程库；
关联后，使用命令 `git push -u origin master` 第一次推送master分支的所有内容；
此后，每次本地提交后，只要有必要，就可以使用命令 `git push origin master` 推送最新修改

- 克隆远程库（本地不存在该仓库）



```
Administrator@WY MINGW64 /repository
$ git clone git@github.com:laoxuai/ssm_crud.git
Cloning into 'ssm_crud'...
remote: Enumerating objects: 72, done.
remote: Counting objects: 100% (72/72), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 72 (delta 10), reused 70 (delta 9), pack-reused 0
Receiving objects: 100% (72/72), 415.22 KiB | 9.00 KiB/s, done.
Resolving deltas: 100% (10/10), done.
```

2. 分支管理

1. 创建dev分支，然后切换到dev分支：

```
Administrator@WY MINGW64 /repository/ssm_crud (master)
$ git checkout -b dev
Switched to a new branch 'dev'
```

git checkout命令加上-b表示创建并切换，相当于：


```
1 $ git branch dev
2
3 $ git checkout dev
4
5 Switched to branch 'dev'
```

2. 使用git branch命令查看当前分支：

git branch命令会列出所有分支，当前分支前面会标一个*号。

```
Administrator@WY MINGW64 /repository/ssm_crud (dev)
$ git branch
* dev
master
```

3. 切换分支

```
$ git checkout master
```

```
Administrator@WY MINGW64 /repository/ssm_crud (dev)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

Administrator@WY MINGW64 /repository/ssm_crud (master)
$ |
```

4. 合并某分支到当前分支： `git merge <name>`

```
Administrator@WY MINGW64 /repository/ssm_crud (master)
$ git checkout test
Switched to branch 'test'

Administrator@WY MINGW64 /repository/ssm_crud (test)
$ git add README.md

Administrator@WY MINGW64 /repository/ssm_crud (test)
$ git commit -m "test"
[test d903cd6] test
1 file changed, 2 insertions(+)

Administrator@WY MINGW64 /repository/ssm_crud (test)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

Administrator@WY MINGW64 /repository/ssm_crud (master)
$ git merge test
Updating ec1ee43..d903cd6
Fast-forward
 README.md | 2 ++
1 file changed, 2 insertions(+)
```

5. 删除分支： `git branch -d <name>`

3. 解决冲突

模拟冲突：切换到test分支，修改readme.md文件，提交

修改完成后，切换回主分支，再次修改Readme.md文件，提交

```
Administrator@WY MINGW64 /repository/ssm_crud (test)
$ git add Readme.md

Administrator@WY MINGW64 /repository/ssm_crud (test)
$ git commit -m "merge test"
[test c85d39e] merge test
1 file changed, 1 insertion(+), 1 deletion(-)

Administrator@WY MINGW64 /repository/ssm_crud (test)
$ git checkout master
Switched to branch 'master'
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

Administrator@WY MINGW64 /repository/ssm_crud (master)
$ git add Readme.md

Administrator@WY MINGW64 /repository/ssm_crud (master)
$ git commit -m "master commit"
[master 60b534c] master commit
1 file changed, 1 insertion(+), 1 deletion(-)

Administrator@WY MINGW64 /repository/ssm_crud (master)
$ git merge test
Auto-merging Readme.md
CONFLICT (content): Merge conflict in Readme.md
Automatic merge failed; fix conflicts and then commit the result.
```

合并分支，可以看到更新时出现冲突

查看当前的状态：

```
Administrator@WY MINGW64 /repository/ssm_crud (master|MERGING)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
(use "git push" to publish your local commits)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both modified:   Readme.md

no changes added to commit (use "git add" and/or "git commit -a")

Administrator@WY MINGW64 /repository/ssm_crud (master|MERGING)
$ cat Readme.md

<<<<<<< HEAD
解决冲突 --冲突测试段
=====
解决冲突:冲突测试段
>>>>>>> test

# 技术点
```

可以看到Readme.md文件存在冲突，必须手动解决冲突后再提交，查看冲突文件，可以看到相关的提示

```
1 <<<<<<< HEAD
2
3 解决冲突 --冲突测试段
4
5 =====
6
7 解决冲突:冲突测试段
8
9 \>>>>>>> test
```

git 使用 <<<<<<<, =====, >>>>>>> 标记出不同分支的内容, 出现冲突后在分支后会出现 (master|MERGING), 需要手动选择修改冲突部分, 再次提交

```
Administrator@WY MINGW64 /repository/ssm_crud (master|MERGING)
$ git add Readme.md

Administrator@WY MINGW64 /repository/ssm_crud (master|MERGING)
$ git commit -m "fixed"
[master 8dc4d62] fixed
```

最后删除test分支即可。

```
1 $ git branch -d test
2
3 Deleted branch test (was c85d39e).
```

合并分支时, 加上--no-ff可以用普通模式合并, 合并后的历史有分支, 能看出来曾经做过合并, 而fast forward合并看起来曾经做过合并

4. 推送至远程仓库

查看远程库信息, 使用 `git remote -v`

从本地推送分支, 使用 `git push origin branch-name`, 如果推送失败, 先用 `git pull` 抓取远程的新提交; 在本地创建和远程分支对应的分支, 使用 `git checkout -b branch-name origin/branch-name`, 本地和远程分支的名称最好一致; 建立本地分支和远程分支的关联, 使用 `git branch --set-upstream branch-name origin/branch-name`; 从远程抓取分支, 使用 `git pull`, 如果有冲突, 要先处理冲突。

5. 标签管理

命令 `git tag <tagname>` 用于新建一个标签, 默认为HEAD, 也可以指定一个commit id;

命令 `git tag -a <tagname> -m "blablabla..."` 可以指定标签信息;

命令 `git tag` 可以查看所有标签

命令 `git push origin <tagname>` 可以推送一个本地标签;

命令 `git push origin --tags` 可以推送全部未推送过的本地标签;

命令 `git tag -d <tagname>` 可以删除一个本地标签;

命令 `git push origin :refs/tags/<tagname>` 可以删除一个远程标签

三、GitHub 和 Gitee

GitHub是一个面向开源及私有软件项目的托管平台，gitee是开源中国推出的代码托管平台,支持 Git 和 SVN,提供免费的私有仓库托管。相比github,gitee速度较快。

1. 同时配置gitee和github账号

ssh 方式连接到 Github/Gitee，需要唯一的公钥，如果想同一台电脑同时绑定Github/Gitee 帐号，需要两个条件:

1. 两对 私钥/公钥
2. push 时，区分两个账户，推送到相应的仓库

配置过程:

- 使用`ls ~/.ssh/`查看公钥文件夹，由于我已经配置好，所以可以看到有两对（四个文件）公钥和一个config配置文件

```
Administrator@WY MINGW64 /repository/ssm_crud (master)
$ ls ~/.ssh/
config id_rsa id_rsa.pub id_rsa_gitee id_rsa_gitee.pub known_hosts
```

- 使用 `ssh-keygen -t rsa -f ~/.ssh/id_rsa_gitee -C "yourmail@xxx.com"`

生成对应的公钥文件，然后复制公钥文件中的SSH公钥粘贴到gitee的SSH公钥中即可。

个人资料

修改密码

邮箱管理

个人空间地址

通知设置

安全设置

SSH公钥

GPG公钥 Beta

数据管理

设置空间信息

升级为组织

升级为企业版

您当前的SSH公钥数: 1

删除

添加公钥

标题

公钥标题(key)

公钥

把你的公钥粘贴到这里。查看 [怎样生成公钥](#)

支持以 'ssh-rsa', 'ssh-dss', 'ssh-ed25519', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384' or 'ecdsa-sha2-nistp521' 开头。

确定

- 配置文件

本地磁盘 (C:) > 用户 > Administrator > .ssh

名称	修改日期	类型	大小
config	2019/8/8 星期四 8:54	文件	
id_rsa	2019/8/8 星期四 8:34	文件	
id_rsa.pub	2019/8/8 星期四 8:34	Microsoft Publis...	
id_rsa_gitee	2019/8/8 星期四 8:37	文件	
id_rsa_gitee.pub	2019/8/8 星期四 8:37	Microsoft Publis...	
known_hosts	2019/8/8 星期四 8:31	文件	

```

1 # github
2 Host github.com
3 HostName github.com
4 PreferredAuthentications publickey
5 IdentityFile ~/.ssh/id_rsa
6
7 # gitee
8 Host gitee.com
9 HostName gitee.com
10 PreferredAuthentications publickey
11 IdentityFile ~/.ssh/id_rsa_gitee

```

- 测试

```
Administrator@Administrator MINGW64 ~
$ ssh -T git@gitee.com
Warning: Permanently added the ECDSA host key for IP address '192.168.1.100' to
the list of known hosts.
Hi laoxuai! You've successfully authenticated, but GITEE.COM does not provide sh
ell access.

Administrator@Administrator MINGW64 ~
$ ssh -T git@github.com
The authenticity of host 'github.com (192.168.1.100)' can't be established.
RSA key fingerprint is SHA256:192.168.1.100.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com,192.168.1.100' (RSA) to the list of know
n hosts.
Hi laoxuai! You've successfully authenticated, but GitHub does not provide shell
access.

Administrator@Administrator MINGW64 ~
$
```

```
1 | $ ssh -T git@gitee.com
2
3 | $ ssh -T git@github.com
```

- 使用

通过关联对应的远程库进行操作即可

参考资料:

[Git教程 - 廖雪峰的官方网站](#)

[配置同时使用 Gitlab、Github、Gitee 共存的开发环境](#)

联系我: @老徐

QQ:1022308684

gitee:[laoxuai](#)

github:[laoxuai](#)