git常用操作

1、 git 安装

在ubuntu上测试有没有安装,如下图说明未安装。

```
stu@stu-virtual-machine:~$ git

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

sudo apt install git

stu@stu-virtual-machine:~$
```

使用命令 sudo apt install git 进行安装。

```
sudo apt install git
```

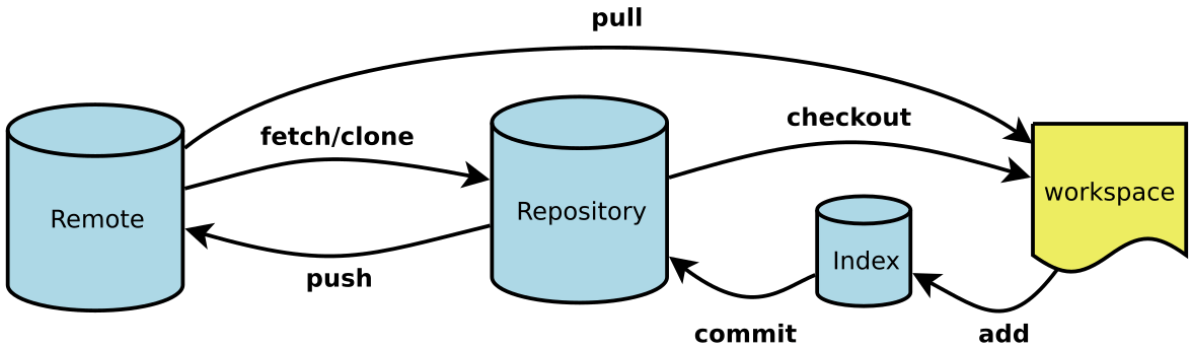
安装后查看版本，如下图说明安装成功。

```
stu@stu-virtual-machine:~$ git version
git version 2.25.1
stu@stu-virtual-machine:~$
```

2、 git基本概念

2.1 四个工作区

git本地有三个工作域：工作区（working directory), 暂存区（stage/index), 资源库（repository)。如果再算上远程服务器上的git仓库(remote directory)就可以分为四个工作域。其关系如下：



- **Workspace**： 工作区，就是你平时存放项目代码的地方
- **Index / Stage**： 暂存区，用于临时存放你的改动，事实上它只是一个文件，保存即将提交到文件列表信息
- **Repository**： 仓库区（或版本库），就是安全存放数据的位置，这里面有你提交到所有版本的数据。其中HEAD指向最新放入仓库的版本
- **Remote**： 远程仓库，托管代码的服务器，可以简单的认为是你项目组中的一台电脑用于远程数据交换

2.2 工作流程

git工作的一般流程:

1. 在工作目录中添加，修改文件
2. 将需要进行版本管理的文件放入暂存区
3. 将暂存区的文件提交到git仓库

2.3 文件的四种状态

- **Untracked:** 未跟踪, 此文件在文件夹中, 但并没有加入到git库, 不参与版本控制. 通过git add 状态变为Staged.
- **Unmodify:** 文件已经入库, 未修改, 即版本库中的文件快照内容与文件夹中完全一致. 这种类型的文件有两种去处, 如果它被修改, 而变为Modified. 如果使用git rm移出版本库, 则成为Untracked文件
- **Modified:** 文件已修改, 仅仅是修改, 并没有进行其他的操作. 这个文件也有两个去处, 通过git add可进入暂存staged状态, 使用git checkout 则丢弃修改过,返回到unmodify状态, 这个git checkout即从库中取出文件, 覆盖当前修改
- **Staged:** 暂存状态. 执行git commit则将修改同步到库中, 这时库中的文件和本地文件又变为一致, 文件为Unmodify状态. 执行git reset HEAD filename取消暂存,文件状态为Modified

3、git基本命令

> 创建本地仓库

1.创建一个目录

```
stu@stu-virtual-machine:~$ mkdir myproject
stu@stu-virtual-machine:~$ cd myproject
stu@stu-virtual-machine:~/myproject$ ls -a
.  ..
stu@stu-virtual-machine:~/myproject$
```

2.使用git init 命令将其变为一个可以通过git管理的仓库

```
stu@stu-virtual-machine:~/myproject$ git init
已初始化空的 Git 仓库于 /home/stu/myproject/.git/
stu@stu-virtual-machine:~/myproject$ ls -a
.  ..  .git
stu@stu-virtual-machine:~/myproject$
```

注意事项:

第一次使用git命令提交代码之前，需要先设置用户名及邮箱，之后就不需要了:

```
stu@stu-virtual-machine:~/myproject$ git config --global user.email
"you@126.com"
stu@stu-virtual-machine:~/myproject$ git config --global user.name "youname"
```

其中 "you@126.com" 及"youname" 应替换为你后面使用的真实邮箱和名字。

3.使用git add filename 添加文件到暂存区

```
stu@stu-virtual-machine:~/myproject$ ls
main.c
stu@stu-virtual-machine:~/myproject$ git add main.c
```

4.使用git status 查看仓库状态

```
stu@stu-virtual-machine:~/myproject$ git status
位于分支 master

尚无提交

要提交的变更：
  （使用 "git rm --cached <文件>..." 以取消暂存）
    新文件：    main.c

stu@stu-virtual-machine:~/myproject$
```

5.使用 git commit -m "版本描述信息" 提交版本到仓库

```
stu@stu-virtual-machine:~/myproject$ git commit -m "创建main.c"
[master （根提交） f30d135] 创建main.c
 1 file changed, 1 insertion(+)
 create mode 100644 main.c
stu@stu-virtual-machine:~/myproject$
```

6.使用git log查看提交的历史记录

```
stu@stu-virtual-machine:~/myproject$ git log
commit f30d135d907fb622c8e178cb1d54947c39ffa0f0 (HEAD -> master)
Author: sufeng <sufeng05@126.com>
Date:   Thu Jun 17 11:56:38 2021 +0800

    创建main.c
stu@stu-virtual-machine:~/myproject$
```

7.使用git reflog 查看对仓库的操作日志

```
stu@stu-virtual-machine:~/myproject$ git reflog
f30d135 (HEAD -> master) HEAD@{0}: commit (initial): 创建main.c
stu@stu-virtual-machine:~/myproject$
```

8.使用git diff HEAD 比较当前内容与最后一次提交的版本的差异,如下在main.c中添加了一行内容，显示添加的一行前面有'+'号标识。如果内容相同则该命令不显示输出结果。HEAD也可以省略默认就是与最近一次比较。

```
stu@stu-virtual-machine:~/myproject$ git diff HEAD
diff --git a/main.c b/main.c
index 53c5fdf..fbfff34 100644
--- a/main.c
+++ b/main.c
@@ -1 +1,2 @@
 #include <stdio.h>
+#include <stdlib.h>
stu@stu-virtual-machine:~/myproject$
```

9.使用git checkout filename 放弃对工作区代码的修改。

```
stu@stu-virtual-machine:~/myproject$ ls
```

```
main.c
stu@stu-virtual-machine:~/myproject$ cat main.c
#include <stdio.h>
stu@stu-virtual-machine:~/myproject$ vi main.c
stu@stu-virtual-machine:~/myproject$ cat main.c
#include <stdio.h>
#include <stdlib.h>
stu@stu-virtual-machine:~/myproject$ git status
位于分支 master
尚未暂存以备提交的变更：
  （使用 "git add <文件>..." 更新要提交的内容）
  （使用 "git restore <文件>..." 丢弃工作区的改动）
    修改：      main.c

修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）
stu@stu-virtual-machine:~/myproject$ git checkout main.c
从索引区更新了 1 个路径
stu@stu-virtual-machine:~/myproject$ cat main.c
#include <stdio.h>
stu@stu-virtual-machine:~/myproject$
```

10.使用git reset HEAD filename 从暂存区撤销

```
stu@stu-virtual-machine:~/myproject$ ls
main.c
stu@stu-virtual-machine:~/myproject$ git status
位于分支 master
尚未暂存以备提交的变更：
  （使用 "git add <文件>..." 更新要提交的内容）
  （使用 "git restore <文件>..." 丢弃工作区的改动）
    修改：      main.c

修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）
stu@stu-virtual-machine:~/myproject$ git add main.c
stu@stu-virtual-machine:~/myproject$ git status
位于分支 master
要提交的变更：
  （使用 "git restore --staged <文件>..." 以取消暂存）
    修改：      main.c

stu@stu-virtual-machine:~/myproject$ git reset HEAD main.c
重置后取消暂存的变更：
M   main.c
stu@stu-virtual-machine:~/myproject$ git status
位于分支 master
尚未暂存以备提交的变更：
  （使用 "git add <文件>..." 更新要提交的内容）
  （使用 "git restore <文件>..." 丢弃工作区的改动）
    修改：      main.c

修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）
stu@stu-virtual-machine:~/myproject$
```

11.使用git rm filename 删除一个文件, 此时提交到暂存区，需要commit后才在版本库中删除

```
stu@stu-virtual-machine:~/myproject$ ls
```

```
main.c
stu@stu-virtual-machine:~/myproject$ git status
位于分支 master
无文件要提交，干净的工作区
stu@stu-virtual-machine:~/myproject$ git rm main.c
rm 'main.c'
stu@stu-virtual-machine:~/myproject$ ls
stu@stu-virtual-machine:~/myproject$ git status
位于分支 master
要提交的变更：
  （使用 "git restore --staged <文件>..." 以取消暂存）
    删除：      main.c

stu@stu-virtual-machine:~/myproject$ ls
stu@stu-virtual-machine:~/myproject$
```

12.使用git reset --hard HEAD^ 回退版本

```
stu@stu-virtual-machine:~/myproject$ git status
位于分支 master
无文件要提交，干净的工作区
stu@stu-virtual-machine:~/myproject$ git reflog
2bc07ca (HEAD -> master) HEAD@{0}: commit: del main.c
8bb2b7b HEAD@{1}: commit: 添加一行
f30d135 HEAD@{2}: commit (initial): 创建main.c
stu@stu-virtual-machine:~/myproject$ ls
stu@stu-virtual-machine:~/myproject$ git reset --hard HEAD^
HEAD 现在位于 8bb2b7b 添加一行
stu@stu-virtual-machine:~/myproject$ ls
main.c
stu@stu-virtual-machine:~/myproject$
```

4、git分支的操作命令

在进行多个并行作业时，通常会用到分支。

1.查看分支： git branch

```
stu@stu-virtual-machine:~/myproject$ git branch
* master
stu@stu-virtual-machine:~/myproject$
```

2.创建分支： git branch 分支名

```
stu@stu-virtual-machine:~/myproject$ git branch dev
stu@stu-virtual-machine:~/myproject$ git branch
  dev
* master
stu@stu-virtual-machine:~/myproject$
```

3.切换分支： git checkout 分支名

```
stu@stu-virtual-machine:~/myproject$ git branch
  dev
* master
stu@stu-virtual-machine:~/myproject$ git checkout dev
切换到分支 'dev'
stu@stu-virtual-machine:~/myproject$ git branch
* dev
  master
stu@stu-virtual-machine:~/myproject$
```

4.创建并切换到该分支： git checkout -b 分支名

```
stu@stu-virtual-machine:~/myproject$ git branch
  dev
* master
stu@stu-virtual-machine:~/myproject$ git checkout -b dev2
切换到一个新分支 'dev2'
stu@stu-virtual-machine:~/myproject$ git branch
  dev
* dev2
  master
stu@stu-virtual-machine:~/myproject$
```

5.删除分支,不能删除当前所处分支，切换到其它分支再删除： git branch -d 分支名

```
stu@stu-virtual-machine:~/myproject$ git branch
  dev
* dev2
  master
stu@stu-virtual-machine:~/myproject$ git checkout dev
切换到分支 'dev'
stu@stu-virtual-machine:~/myproject$ git branch -d dev2
已删除分支 dev2（曾为 8bb2b7b）。
stu@stu-virtual-machine:~/myproject$ git branch
* dev
  master
stu@stu-virtual-machine:~/myproject$
```

6.合并某个分支到当前分支： git merge 分支名，合并时可能产生冲突，需要解决冲突。

有时需要禁止快速合并，可执行：git merge --no-ff -m '描述' 分支名

```
stu@stu-virtual-machine:~/myproject$ ls
main.c
stu@stu-virtual-machine:~/myproject$
stu@stu-virtual-machine:~/myproject$ vi main.c
stu@stu-virtual-machine:~/myproject$ git branch
* dev
  master
stu@stu-virtual-machine:~/myproject$ git add main.c
stu@stu-virtual-machine:~/myproject$ git status
位于分支 dev
要提交的变更：
  （使用 "git restore --staged <文件>..." 以取消暂存）
    修改：      main.c
```

```
stu@stu-virtual-machine:~/myproject$ git commit -m 'dev添加main'
[dev 024836e] dev添加main
 1 file changed, 5 insertions(+)
stu@stu-virtual-machine:~/myproject$ git status
位于分支 dev
无文件要提交，干净的工作区
stu@stu-virtual-machine:~/myproject$ git branch
* dev
  master
stu@stu-virtual-machine:~/myproject$ git checkout master
切换到分支 'master'
stu@stu-virtual-machine:~/myproject$ git merge dev
更新 8bb2b7b..024836e
Fast-forward
 main.c | 5 +++++
 1 file changed, 5 insertions(+)
stu@stu-virtual-machine:~/myproject$ vi main.c
stu@stu-virtual-machine:~/myproject$ git branch
  dev
* master
stu@stu-virtual-machine:~/myproject$
```

7.历史记录一行显示 : git log --pretty=oneline

```
stu@stu-virtual-machine:~/myproject$ git log --pretty=oneline
024836e9de25b8cc120fb7cce7b2ef3fd936bbc0 (HEAD -> master, dev) dev添加main
8bb2b7b1fa23eb347c654e98821c6510cb60edb9 添加一行
f30d135d907fb622c8e178cb1d54947c39ffa0f0 创建main.c
stu@stu-virtual-machine:~/myproject$
```

8.以图表形式显示分支：git log --graph

```
stu@stu-virtual-machine:~/myproject$ git log --graph
*   commit 4993413c30f1c496428f0c16b2a96bb643103ff6 (HEAD -> master)
|\  Merge: 809d358 1b659bf
| | Author: sufeng <sufeng05@126.com>
| | Date:   Thu Jun 17 14:23:44 2021 +0800
| |
| |     解决合并冲突
| |
| * commit 1b659bf4f4f2bd8752c5aa88079475075b9131f64 (dev2)
| | Author: sufeng <sufeng05@126.com>
| | Date:   Thu Jun 17 14:20:48 2021 +0800
| |
| |     添加printf方法
| |
* | commit 809d3583025e9f61dce64687d96ff821ce826778
|/  Author: sufeng <sufeng05@126.com>
|   Date:   Thu Jun 17 14:22:03 2021 +0800
|
|       添加exit方法
|
* commit 024836e9de25b8cc120fb7cce7b2ef3fd936bbc0 (dev)
| Author: sufeng <sufeng05@126.com>
| Date:   Thu Jun 17 14:05:32 2021 +0800
```

```
|
|     dev添加main
|
* commit 8bb2b7b1fa23eb347c654e98821c6510cb60edb9
| Author: sufeng <sufeng05@126.com>
| Date:   Thu Jun 17 12:55:56 2021 +0800
```

9.保护现场 git stash,当前工作区有代码修改了，是不能切换到其他分支，可以先保存现场，再切换

```
stu@stu-virtual-machine:~/myproject$ git status
位于分支 master
无文件要提交，干净的工作区
stu@stu-virtual-machine:~/myproject$ vi main.c
stu@stu-virtual-machine:~/myproject$ git status
位于分支 master
尚未暂存以备提交的变更：
  （使用 "git add <文件>..." 更新要提交的内容）
  （使用 "git restore <文件>..." 丢弃工作区的改动）
    修改：      main.c

修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）
stu@stu-virtual-machine:~/myproject$ git branch dev
fatal: 一个分支名 'dev' 已经存在。
stu@stu-virtual-machine:~/myproject$ git checkout dev
error: 您对下列文件的本地修改将被检出操作覆盖：
    main.c
请在切换分支前提交或贮藏您的修改。
正在终止
stu@stu-virtual-machine:~/myproject$ git stash
保存工作目录和索引状态 WIP on master: 4993413 解决合并冲突
stu@stu-virtual-machine:~/myproject$ git status
位于分支 master
无文件要提交，干净的工作区
stu@stu-virtual-machine:~/myproject$ git checkout dev
切换到分支 'dev'
stu@stu-virtual-machine:~/myproject$
```

10.列出所有保存的现场信息 git stash list

```
stu@stu-virtual-machine:~/myproject$ git stash list
stash@{0}: WIP on master: 4993413 解决合并冲突
stu@stu-virtual-machine:~/myproject$ git branch
* dev
  dev2
  master
stu@stu-virtual-machine:~/myproject$ vi main.c
stu@stu-virtual-machine:~/myproject$ git stash
保存工作目录和索引状态 WIP on dev: c6a7375 定义a,b
stu@stu-virtual-machine:~/myproject$ git stash list
stash@{0}: WIP on dev: c6a7375 定义a,b
stash@{1}: WIP on master: 4993413 解决合并冲突
stu@stu-virtual-machine:~/myproject$
```

11.取出某次的现场信息，继续工作 :git stash pop "stash@{1}" ,默认是最近一次，如果有多个现场，也可以加上编号"stash@{1}"指定获取某一个。不同分支的现场，应该回到对应分支再获取，否则会自动合并现场到当前分支的工作区。

```
stu@stu-virtual-machine:~/myproject$ git stash list
stash@{0}: WIP on dev: c6a7375 定义a,b
stash@{1}: WIP on master: 4993413 解决合并冲突
stu@stu-virtual-machine:~/myproject$ ls
main.c
stu@stu-virtual-machine:~/myproject$ git checkout master
切换到分支 'master'
stu@stu-virtual-machine:~/myproject$ ls
main.c
stu@stu-virtual-machine:~/myproject$ git branch
  dev
  dev2
* master
stu@stu-virtual-machine:~/myproject$ git stash pop stash@{1}
位于分支 master
尚未暂存以备提交的变更：
  （使用 "git add <文件>..." 更新要提交的内容）
  （使用 "git restore <文件>..." 丢弃工作区的改动）
    修改：      main.c

修改尚未加入提交（使用 "git add" 和/或 "git commit -a"）
丢弃了 stash@{1}（b553dce6fed8f60702474db1fa004bebbd13126a）
stu@stu-virtual-machine:~/myproject$ git stash list
stash@{0}: WIP on dev: c6a7375 定义a,b
stu@stu-virtual-machine:~/myproject$
```

5、远程仓库操作

1. 生成通信密钥:ssh-keygen -t rsa -C "su@126.com" ,生成的公钥在/home/stu/.ssh/下，如下图。

```
stu@stu-virtual-machine:~/myproject$ ssh-keygen -t rsa -C "su****@126.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/stu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/stu/.ssh/id_rsa
Your public key has been saved in /home/stu/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:QFRjZabbzQadfEYfvGIanZbRkWdYSLpv7OiSgbRMFr4 su****@126.com
The key's randomart image is:
+---[RSA 3072]----+
|     .o.+.=  .=*+|
|     . . * + =o=+|
|      . . * * *.+|
|       . B O @ . |
|        S E @ .  |
|           o +   |
|            o +  |
|             + o |
|              ..o|
+----[SHA256]-----+
stu@stu-virtual-machine:~/myproject$
```

2. 测试与github或者gitee(码云)有没有连通：

测试github 的命令： ssh -T git@github.com

```
tu@stu-virtual-machine:~/myproject$ ssh -T git@github.com
Warning: Permanently added the RSA host key for IP address '52.74.223.119'
to the list of known hosts.
Hi sufeng05! You've successfully authenticated, but GitHub does not provide
shell access.
stu@stu-virtual-machine:~/myproject$
```

测试gitee 也就是码云的命令：

```
stu@stu-virtual-machine:~/myproject$ ssh -T git@gitee.com
Hi sufeng! You've successfully authenticated, but GITEE.COM does not provide
shell access.
stu@stu-virtual-machine:~/myproject$
```

3. 克隆项目：git clone 项目地址
4. 提交分支到远程仓库：git push origin 分支名
5. 提交分支到远程仓库，并跟踪分支： git push -u origin 分支名
6. 拉取远程服务器上的分支更新到本地： git pull origin 分支名