本地库

文件查看

- 进入文件夹

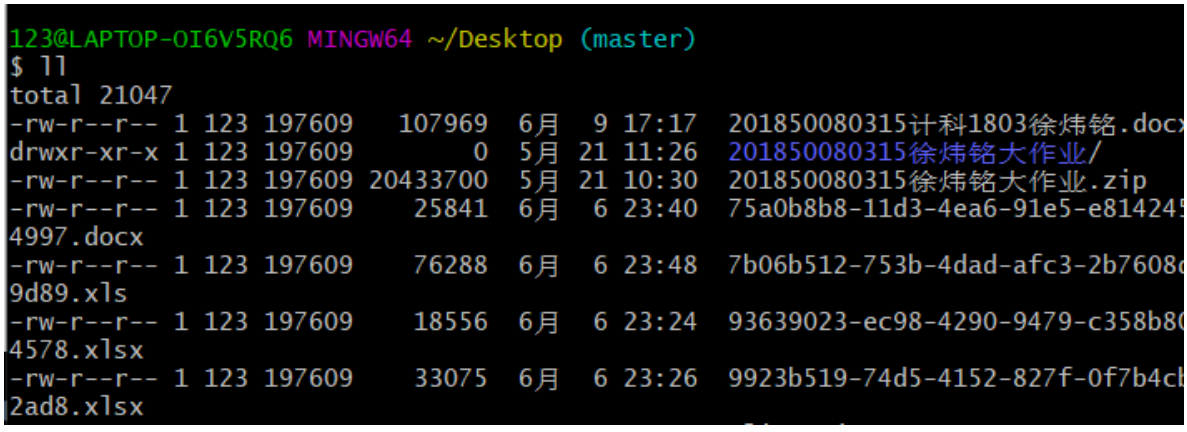
`cd weChat`

- 路径回退

`cd ../`

- 查看当前文件夹的所有内容

`ll`



- 查看当前文件夹的隐藏内容

`ls-la`

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ ls -la
total 4
drwxr-xr-x 1 123 197609 0  6月 24 13:58 ./
drwxr-xr-x 1 123 197609 0  6月 24 13:58 ../
drwxr-xr-x 1 123 197609 0  6月 24 13:58 .git/
```

- 查看当前文件夹的特定内容(隐藏的也可查看)

ll .git/

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ ll .git/
total 7
-rw-r--r-- 1 123 197609 112  6月 24 13:58 config
-rw-r--r-- 1 123 197609  73  6月 24 13:58 description
-rw-r--r-- 1 123 197609  23  6月 24 13:58 HEAD
drwxr-xr-x 1 123 197609   0  6月 24 13:58 hooks/
drwxr-xr-x 1 123 197609   0  6月 24 13:58 info/
drwxr-xr-x 1 123 197609   0  6月 24 13:58 objects/
drwxr-xr-x 1 123 197609   0  6月 24 13:58 refs/
```

初始化

- 本地库初始化，先进入相关文件夹，然后

git init

```
$ git init
Initialized empty Git repository in C:/Users/USER_XWM/Desktop/.git/
```

- 配置本地name和email

git config user.name MarsMTT

git config user.email 869823401@qq.com

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git config user.name MarsMTT

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git config user.email 869823401@qq.com
```

- 查看配置信息

cat .git/config

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ cat .git/config
[core]
        repositoryformatversion = 0
        filemode = false
        bare = false
        logallrefupdates = true
        ignorecase = true
[user]
        name = MarsMTT
        email = 869823401@qq.com
```

- 配置系统name和email（在本地的基础上加--global）

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git config --global user.name MarsMTT_global

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git config --global user.email 869823401_glb@qq.com
```

- 查看系统配置的信息（需要进入到c盘用户文件夹）

```
123@LAPTOP-OI6V5RQ6 MINGW64 ~
$ cat ~/.gitconfig
[user]
        name = MarsMTT_global
        email = 869823401_glb@qq.com
```

添加提交命令

- 状态查看

git status

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   good.txt

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

- 添加操作

git add [file name]

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git add good.txt
warning: LF will be replaced by CRLF in good.txt.
The file will have its original line endings in your working directory
```

- 提交命令

git commit -m "注释" [file name]

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git commit -m "My second commit,modify good.txt" good.txt
warning: LF will be replaced by CRLF in good.txt.
The file will have its original line endings in your working directory
[master 6af45fb] My second commit,modify good.txt
 1 file changed, 1 insertion(+)
```

查看历史记录

- 查看最完整记录

git log

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git commit -m "My second commit,modify good.txt" good.txt
warning: LF will be replaced by CRLF in good.txt.
The file will have its original line endings in your working directory
[master 6af45fb] My second commit,modify good.txt
 1 file changed, 1 insertion(+)

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git log
commit 6af45fb097cac7ada0bdd4e919e7c01a90cc51ef (HEAD -> master)
Author: MarsMTT <869823401@qq.com>
Date:   Wed Jun 24 22:33:49 2020 +0800

    My second commit,modify good.txt

commit 5f4d95ee159a3461c16165d4d284597cae247d67
Author: MarsMTT <869823401@qq.com>
Date:   Wed Jun 24 22:24:09 2020 +0800

    My first commit.new file good.txt
```

- 查看略微简洁的记录

git log --pretty=oneline

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git log --pretty=oneline
1e44667959d4abbc45742a0c61c6bc80ac715dbd (HEAD -> master) the seventh modify
36ef7686a4743826f665773b1039470acbe3ee44 the sixth modify
6e806b7b075a11ac1f36c47438efe151551ff895 the fifth modify
8451cb61dba8d0feb0700d9fb11a1549912418de the fourth modify
5b0dd8d37dd059f739c76b142f2ea41b31deb66c the third commit,modify good.txt
6af45fb097cac7ada0bdd4e919e7c01a90cc51ef My second commit,modify good.txt
5f4d95ee159a3461c16165d4d284597cae247d67 My first commit.new file good.txt
```

- 查看超级简洁的记录

git log --oneline

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git log --oneline
1e44667 (HEAD -> master) the seventh modify
36ef768 the sixth modify
6e806b7 the fifth modify
8451cb6 the fourth modify
5b0dd8d the third commit,modify good.txt
6af45fb My second commit,modify good.txt
5f4d95e My first commit.new file good.txt
```

- 比oneline多了到当前版本需要的步数的记录

git reflog

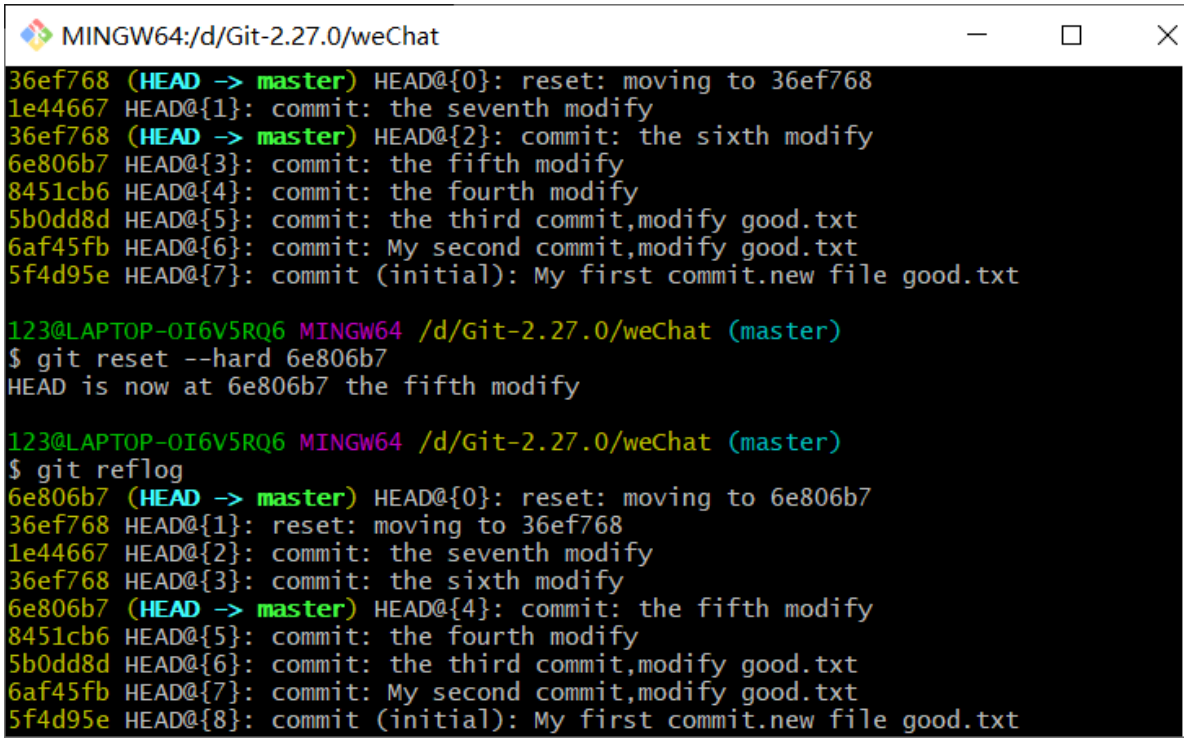
```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git reflog
1e44667 (HEAD -> master) HEAD@{0}: commit: the seventh modify
36ef768 HEAD@{1}: commit: the sixth modify
6e806b7 HEAD@{2}: commit: the fifth modify
8451cb6 HEAD@{3}: commit: the fourth modify
5b0dd8d HEAD@{4}: commit: the third commit,modify good.txt
6af45fb HEAD@{5}: commit: My second commit,modify good.txt
5f4d95e HEAD@{6}: commit (initial): My first commit.new file good.txt
```

前进后退版本

- 基于索引值操作[推荐]

git reset --hard [索引值]

(推荐使用 git reflog)



- 使用 ^ 符号，只能后退，找以前的版本

git reset --hard HEAD^ (可以用多个^符号)

(建议使用git log --oneline)

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git log --oneline
36ef768 (HEAD -> master) the sixth modify
6e806b7 the fifth modify
8451cb6 the fourth modify
5b0dd8d the third commit,modify good.txt
6af45fb My second commit,modify good.txt
5f4d95e My first commit.new file good.txt

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git reset --hard HEAD^
HEAD is now at 6e806b7 the fifth modify
```

- 使用 ~ 符号，只能后退

git reset --hard HEAD~n （n表示回退的步数）

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git reset --hard HEAD~2
HEAD is now at 5b0dd8d the third commit,modify good.txt
```

- hard--soft--mixed 参数对比

soft参数：仅仅修改本地库的HEAD指针（移动后用git status查看状态，文件变为绿色，是因为本地库移动，与缓存区无法对应）

mixed参数：修改本地库的指针+重置缓存区（移动后用git status查看状态，文件变为红色，是因为缓存区移动，与工作区无法对应）

hard参数：修改本地库的指针+重置缓存区+重置工作区

文件删除及找回相关操作

- 文件删除

rm [文件名]

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ rm good1.txt

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ ll
total 1
-rw-r--r-- 1 123 197609 86  6月 24 23:35 good.txt

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    good1.txt

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

- 删除的一套操作，也是需要add和commit

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git add good1.txt

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git commit -m "delete good1.txt" good1.txt
[master 65543bb] delete good1.txt
 1 file changed, 2 deletions(-)
 delete mode 100644 good1.txt

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git status
On branch master
nothing to commit, working tree clean
```

- 找回操作（指的是本地库有记录的找回，其实就是版本回退）

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git reset --hard 5620800
HEAD is now at 5620800 new file good1.txt
```

- rm操作添加到缓存区，怎么找回

原因：由于本地库还没有变，HEAD指针仍然指向原来的commit，所以可以直接用git reset --hard HEAD返回

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ rm good1.txt

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ add good1.txt
bash: add: command not found

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git add good1.txt

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git reset --hard HEAD
HEAD is now at 5620800 new file good1.txt

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git status
On branch master
nothing to commit, working tree clean
```

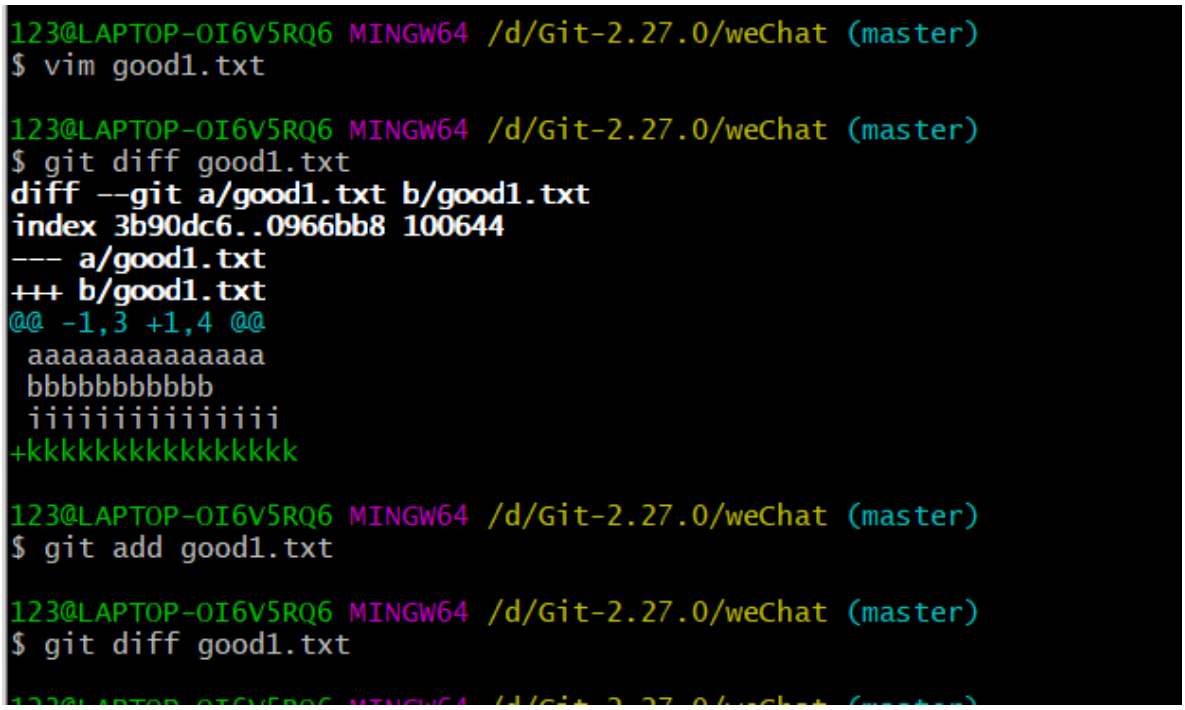
- 总结：只要本地库有commit的提交记录，啥都可以找回

文件比较

- 第一种

git diff [文件名]

说明：将工作区的文件与缓存区的文件比较



(vim编辑后，工作区改变，缓存区依然是老版本，此方法比较工作区和缓存区的差异，add之后，缓存区变为新版本，git diff就无法找到差异)

- 第二种

git diff [本地库中历史版本] [文件名]

说明：将工作区和缓存区的文件进行比较

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git add good1.txt

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git diff good1.txt

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git diff HEAD good1.txt
diff --git a/good1.txt b/good1.txt
index 3b90dc6..0966bb8 100644
--- a/good1.txt
+++ b/good1.txt
@@ -1,3 +1,4 @@
 aaaaaaaaaaaaaa
 bbbbbbbbbbb
 iiiiiiiiiiiiiii
+kkkkkkkkkkkkkkk
```

（此时还没有commit，所以存在different）

- 第三种

  git diff　　git diff HEAD

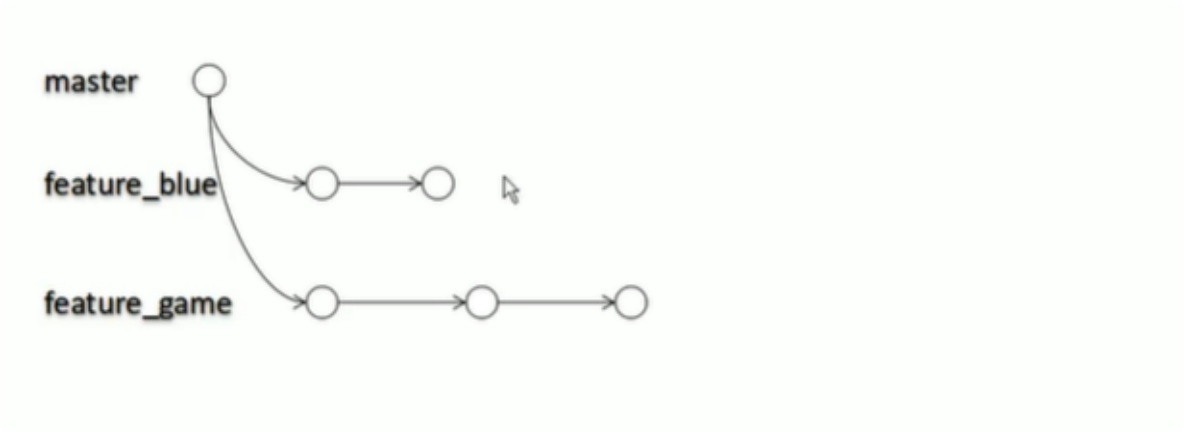
  说明：如果不加文件名，可以比较所有的变化

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git diff HEAD
diff --git a/good.txt b/good.txt
index 6b458b9..a9f1b3f 100644
--- a/good.txt
+++ b/good.txt
@@ -3,3 +3,4 @@
 3333333333333333
 uuuuuuuuuuuuuu
 hello !!!!!
+############
diff --git a/good1.txt b/good1.txt
index 3b90dc6..0966bb8 100644
--- a/good1.txt
+++ b/good1.txt
@@ -1,3 +1,4 @@
 aaaaaaaaaaaaaa
 bbbbbbbbbbb
 iiiiiiiiiiiiiii
+kkkkkkkkkkkkkkk
```

分支

- 什么是分支？？？

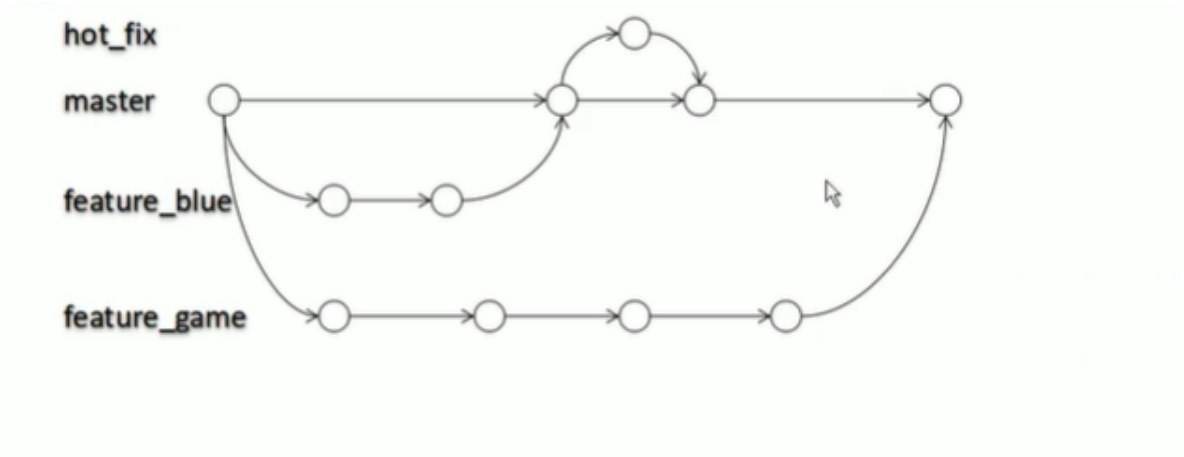
答：在版本控制的过程中，使用多条线，同时推进多个任务



(第一次注册后，操作一直在master上)

(分支出来后，每个分支的操作互不影响，若开发失败只需要删除分支)

- 合并回主干



创建查看切换分支

- 创建分支

git branch [分支名]

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git branch hot_fix
```

- 查看分支

git branch -v

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git branch -v
  hot_fix 51d3eca insert ######
* master  51d3eca insert ######
```

- 切换分支

git checkout [分支名]

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git checkout hot_fix
Switched to branch 'hot_fix'
```

- 小测验

当我对hot_fix分支中的good1.txt进行修改后

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (hot_fix)
$ vim good1.txt

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (hot_fix)
$ git add good1.txt

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (hot_fix)
$ git commit -m "test branch hout_fix" good1.txt
[hot_fix 0b84d81] test branch hout_fix
 1 file changed, 1 insertion(+)

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (hot_fix)
$ git branch -v
* hot_fix 0b84d81 test branch hout_fix
  master  51d3eca insert ######
```

（查看两个分支的进度，不相同）

合并分支

1. 切换到接受修改的分支 `git checkout [分支名]`
2. 执行merge命令 `git merge [分支名]`

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (master)
$ git merge hot_fix
Updating 51d3eca..0b84d81
Fast-forward
 good1.txt | 1 +
 1 file changed, 1 insertion(+)
```

合并冲突

说明：当两个分支，都在同一个位置进行了修改，合并时会产生冲突，需自己来决定，使用哪个

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (hot_fix)
$ git merge master
Auto-merging good1.txt
CONFLICT (content): Merge conflict in good1.txt
Automatic merge failed; fix conflicts and then commit the result.
```

（出现自动合并失败的情况）

- 分支冲突的表现

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (hot_fix|MERGING)
$ cat good1.txt
aaaaaaaaaaaaaa
bbbbbbbbbb
iiiiiiiiiiiiiii
kkkkkkkkkkkkkkk
<<<<<<< HEAD
hothothothot  conflict conflict conflict
=======
hothothothot   conflict
>>>>>>> master
```

- 解决方法

进入vim编辑器，自行修改，之后add，commit（注意commit不需要加文件名，但是日志还是要的）

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (hot_fix|MERGING)
$ vim good1.txt

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (hot_fix|MERGING)
$ git add good1.txt

123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/weChat (hot_fix|MERGING)
$ git commit -m "solve conflict"
[hot_fix 6e18136] solve conflict
```

远程库

本地库创建远程库地址别名

- 创建别名

git remote add [别名] [地址]

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/theFirstFile (master)
$ git remote add origin https://github.com/MarsMTT/theFirstFile.git
```

- 查看别名及地址

git remote -v

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/theFirstFile (master)
$ git remote -v
origin  https://github.com/MarsMTT/theFirstFile.git (fetch)
origin  https://github.com/MarsMTT/theFirstFile.git (push)
```

推送操作

git push [地址或者别名] [分支名]

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/theFirstFile (master)
$ git push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 233 bytes | 233.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MarsMTT/theFirstFile.git
 * [new branch]      master -> master
```

克隆操作

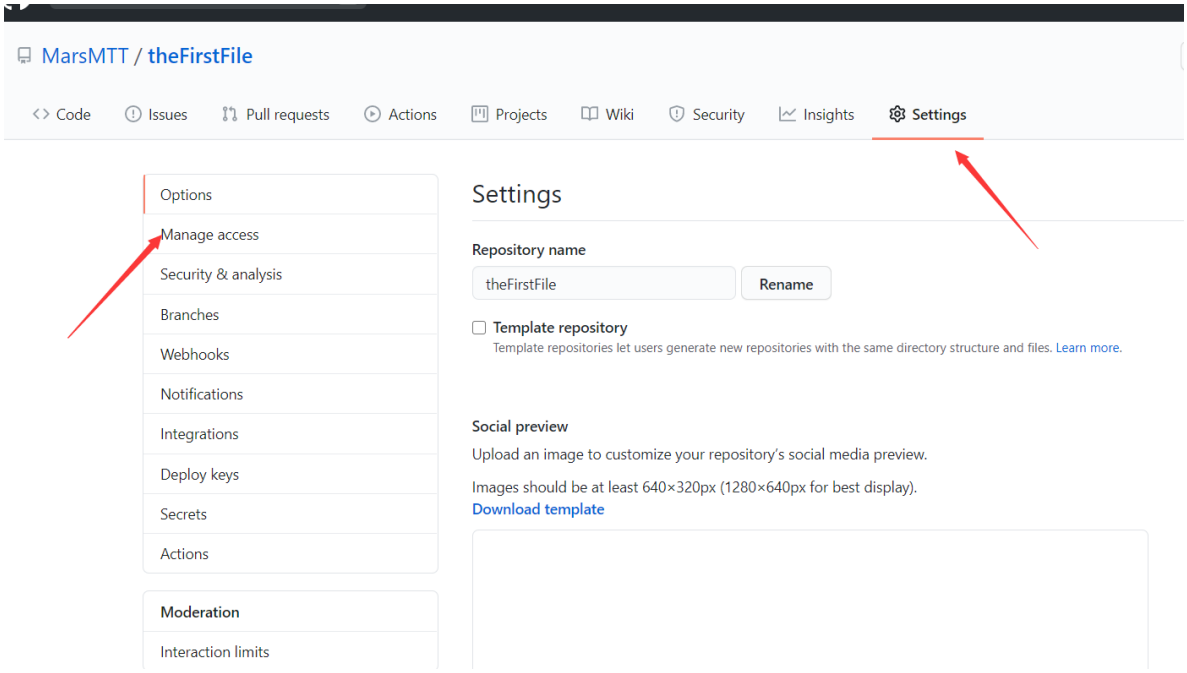
git clone [远程地址]

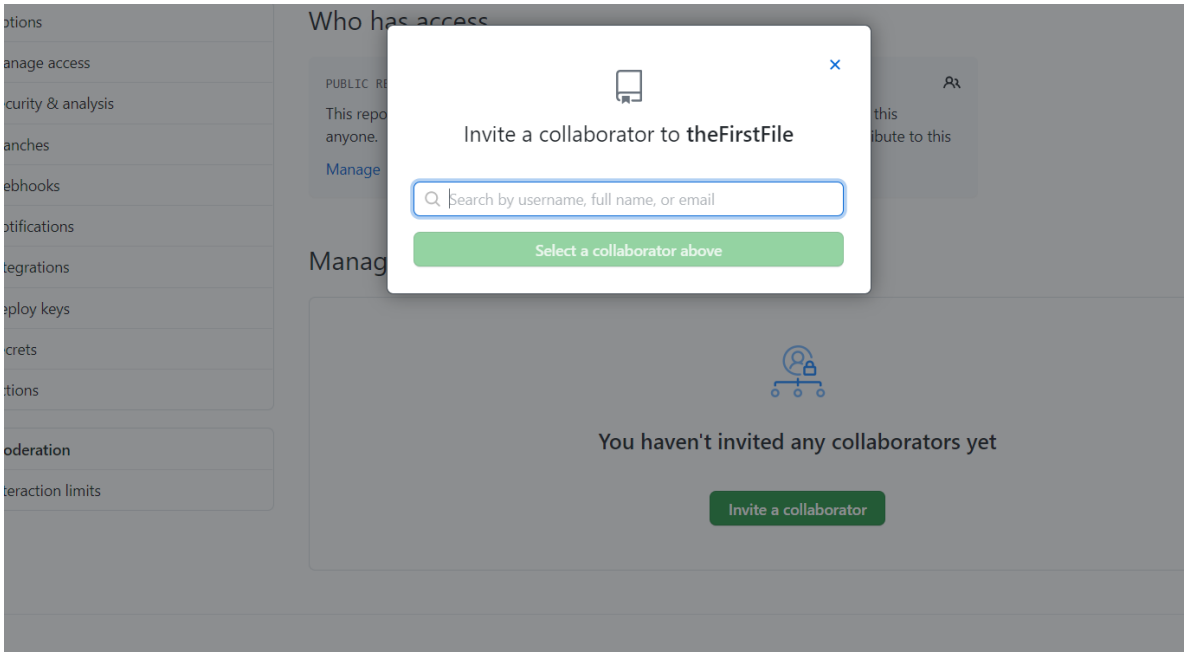
```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/theFirstClone
$ git clone https://github.com/MarsMTT/theFirstFile.git
Cloning into 'theFirstFile'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 213 bytes | 9.00 KiB/s, done.
```

有如下三个效果

1. 把远程库下载到本地库（一个文件夹）
2. 创建origin远程地址别名（就是远程库中设置的）
3. 直接初始化本地库，本地库在克隆前，不需要初始化，git init不要的

邀请成员加入





- 此时输入需要邀请成员的用户名，然后复制链接给成员，那个成员跳转到链接即可（其实成员的邮箱应该也会收到）

远程库的拉取

- pull操作由fetch和merge组成

1. 以下是分开的操作

- 拉取

git fetch [远程库地址别名] [远程库分支]

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/theFirstClone/theFirstFile (master)
$ git fetch origin master
From https://github.com/MarsMTT/theFirstFile
 * branch            master     -> FETCH_HEAD
```

(fetch操作之后，本地的文件不会变)

如果想查看fetch下来的内容，需要将分支切换到origin下的master

```
123@LAPTOP-OI6V5RQ6 MINGW64 /d/Git-2.27.0/theFirstClone/theFirstFile (mast
$ git checkout origin/master
Note: switching to 'origin/master'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -
```

- 进行merge操作之前，先回到本地需要合并的库，然后进行merge

git merge [远程库别名/远程库分支名]

2. pull操作一步完成，抓取&合并

git pull [远程库别名] [远程库分支名]

解决冲突

1. 如果不是基于GitHub远程库的最新版本所做的修改，不能push，必须先pull最新版本下来修改
2. pull下来后，如果进入冲突状态，则按照“分支冲突”解决（commit的时候不需要分支名，可以有日志）

SSH免密登录

```
$ cd ~

123@LAPTOP-OI6V5RQ6 MINGW64 ~
$ rm -r .ssh/
rm: cannot remove '.ssh/': No such file or directory
```

```
123@LAPTOP-OI6V5RQ6 MINGW64 ~
$ ssh-keygen -t rsa -C MarsMTT@aliyun.com
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/USER_XWM/.ssh/id_rsa):
Created directory '/c/Users/USER_XWM/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/USER_XWM/.ssh/id_rsa
Your public key has been saved in /c/Users/USER_XWM/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:tjizt6TBmLLbXN1P5FUh4pu8IlLlNxneQJdeix++d5s MarsMTT@aliyun.com
The key's randomart image is:
+---[RSA 3072]----+
|           o o.. |
|          o o....|
|         . +. o o|
|        o o Bo + |
|       .S. O..+ .|
|     +.+ o.oo. o |
|  . o.B.+...o   .|
|   = ..Bo .o   .+|
|  o.o o...   .  E+|
+----[SHA256]-----+
```

```
123@LAPTOP-OI6V5RQ6 MINGW64 ~
$ cd .ssh/

123@LAPTOP-OI6V5RQ6 MINGW64 ~/.ssh
$ ll
total 5
-rw-r--r-- 1 123 197609 2602  6月 25 17:04 id_rsa
-rw-r--r-- 1 123 197609  572  6月 25 17:04 id_rsa.pub

123@LAPTOP-OI6V5RQ6 MINGW64 ~/.ssh
$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCbkdNtUWUp1wteGArVC32MU7WAXlOGtYnNb9Z+kRAh
qziKQOmt+HEwO/yBYyJ67pylsEkiwyWtQO1dDiY4hkI2zQCvR1b+yXtmm97IEU3c13UDezXSvUX9Udul
g9OVUAe1Xkt+JMXi5dNRP/tPxVXGDbO6nLooXt7sLvuEehABdeVObp9flUHz4QQZtsxGn6WiXpzMsKfD
LNZBS5Sgj6ESMOnHOyxZ4TaW+dnb5b8S5j7eOiB27bk+dfbA39+r/d61ImpZSpBTTwpbi4Kbs+Hkp2Ve
8kLeqowzL/hxheYYvXT2rWvSE6sUoUqG2a64RDtfiDonD88CDsPOnbQGg4bhTcWHnNGYe9QGyKKTCQE9
YogQfPHcxeb4uU86oAPBCu4t8KoolrW3LuwKJjOc6ipcC43IHrRJSZkzWjL3L6R+Znm9UwILqq9WosM6
gjBsf9Kf7v2FgwHrT6hxQS/oJxIlVpSjUazHZmOo+8vAIsqNRq6D+JS/GUSJWJkeOYBOlFO= MarsMTT
@aliyun.com
```