

前言

我们知道一个提交对象就是一个版本，那么我们在某一次版本中具有历史性的突破所以我们需要进行记录一下，在Git中可以利用tag(标签)来进行记录。我们工作的时候是需要伙伴的也就是说我们需要进行远程协作的啦！下面就分别讲述他们的功能以及一些简易操作。

一、 Tag

Git可以使用给一个提交对象进行打标签或者说是标记。类似于游戏的版本~

- 列出标签

命令：git tag

- 创建标签

轻量标签：这个标签的功能和分支有点类似，只不过这个标签不能移动。

命令：git tag v1.0 [commitHash]

```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git tag v1.0 ← 打tag

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git dnf
* 2cbe0f (HEAD -> master, tag: v1.0) V2 Commit
* 4273cb3 V1 Commit

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ vim jc.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git commit -a -m "Git Tag Test"
[master 8b40cb3] Git Tag Test
 1 file changed, 1 insertion(+)

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git dnf ← tag是无法移动的
* 8b40cb3 (HEAD -> master) Git Tag Test
* 2cbe0f (tag: v1.0) V2 Commit
* 4273cb3 V1 Commit

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$
```

附注标签: 这个标签是存储在Git数据库中的一个完整对象。他们是被检验的；标签会包含打标签的者的名字、电子邮件地址和日期时间；我们通常是创建附注标签，如果你因为某些原因不想保存信息那么是可以选择轻量标签的。

命令：git tag -a v1.1

```
git tag -a v1.1 commitHash
```

```
git tag -a v1.1 commitHash -m "version v1.1"
```

查看标签: 这个标签的功能和分支有点类似，只不过这个标签不能移动。

命令：git show tagname

删除标签: 这个标签的功能和分支有点类似，只不过这个标签不能移动。

```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git dnf
* 8b40cb3 (HEAD -> master) Git Tag Test
* 2cbe0f (tag: v1.0) V2 Commit
* 4273cb3 V1 Commit

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git tag -d v1.0
Deleted tag 'v1.0' (was 2cbe0f)

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git dnf
* 8b40cb3 (HEAD -> master) Git Tag Test
* 2cbe0f V2 Commit
* 4273cb3 V1 Commit

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ |
```

命令：git tag -d tagname

检出标签：这个标签的功能和分支有点类似，只不过这个标签不能移动。

命令：git checkout tagname

```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git dnf
* 8b40cb3 (HEAD -> master) Git Tag Test
* 2cbe0f V2 Commit
* 4273cb3 (tag: V1.0) V1 Commit

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (V1.0) 检出标签
$ git checkout V1.0
Note: switching to 'V1.0'.

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 -

Turn off this advice by setting config variable advice.detachedHead
ad to false

HEAD is now at 4273cb3 V1 Commit

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (V1.0) 修改HEAD
$ git checkout -b "testBranch"
Switched to a new branch 'testBranch'

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (testBranch)
$ git dnf
* 8b40cb3 (master) Git Tag Test
* 2cbe0f V2 Commit
* 4273cb3 (HEAD -> testBranch, tag: V1.0) V1 Commit
```

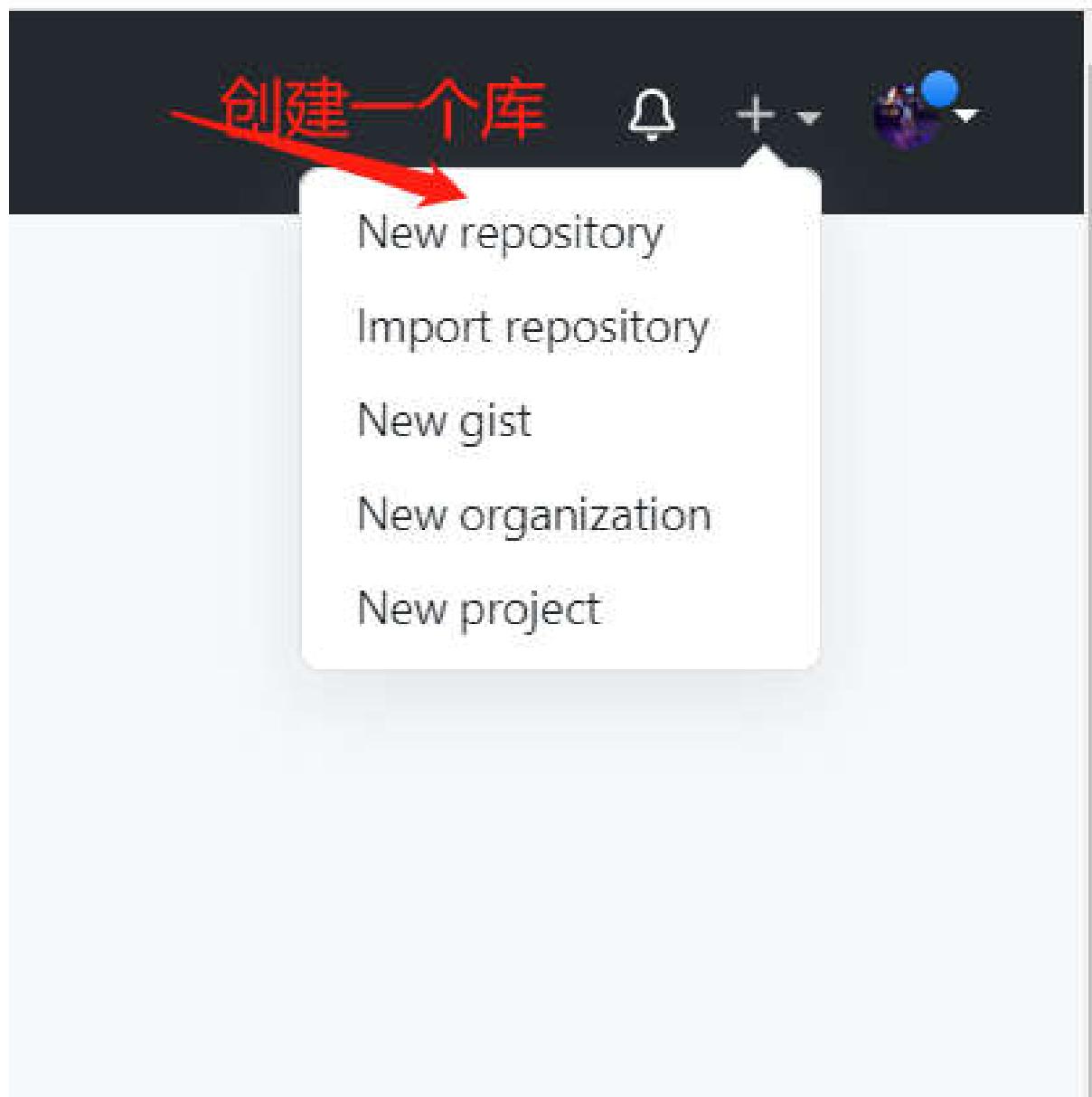
到这里一些基本的操作就大体上结束了，下面将结合远程仓库(GitHub)进行讲解团队协作。

二、远程仓库

为了可以使用Git项目上的团队协作，我们需要知道如何管理自己的远程仓库。远程仓库的内容就是我项目的版本库，一般来言我们可以又很多远程仓库。与他人协作完成共同任务需要我们掌握管理远程仓库以及根据需求推送或拉取数据。下面就为远程仓库开个头~

首先我们需要打开GitHub的[网址](#)进行注册账号以及创建一个新的仓库。

1. 创建远程仓库



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *



QianquanChina ▾

Repository name *

Jc



Great repository names are short and memorable. Need inspiration? How about [didactic-octo-fiesta?](#)

Description (optional)

呀嗨嗨~ 笔记更新到Github啦啦啦~~~~~

Public

Anyone on the internet can see this repository. You choose who can commit.

Private

You choose who can see and commit to this repository.

不要点击选项因为我们要通过
git init 来初始化仓库呢~~略略

Initialize this repository with:

Skip this step if you're importing an existing repository.

Add a README file

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository



点击创建

Quick setup — if you've done this kind of thing before

 Set up in Desktop

or

HTTPS

SSH

<https://github.com/QianquanChina/Jc.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 "# Jc" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/QianquanChina/Jc.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/QianquanChina/Jc.git
git branch -M main
git push -u origin main
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

2.为远程仓库配置简单的名字

- 命令：git remote add name url
- 作用：添加一个新的远程Git仓库，同时指定一个我们自己容易记得简写。

- 命令：git remote -v
- 作用：显示远程仓库使用的Git别名和与之对应的url。

```

dell@DESKTOP-VIIIAFN MINGW64 ~/Desktop/work (testBranch)
$ git remote add lol https://github.com/QianquanChina/Jc.git
dell@DESKTOP-VIIIAFN MINGW64 ~/Desktop/work (testBranch)
$ git remote
lol
dell@DESKTOP-VIIIAFN MINGW64 ~/Desktop/work (testBranch)
$ git remote -v
lol  https://github.com/QianquanChina/Jc.git (fetch)
lol  https://github.com/QianquanChina/Jc.git (push)
dell@DESKTOP-VIIIAFN MINGW64 ~/Desktop/work (testBranch)
$ 
```

查看配置的别名

Quick setup — if you've done this kind of thing before
[Set up in Desktop](#) or [HTTPS](#) [SSH](#) <https://github.com/QianquanChina/Jc.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository has a README.

...or create a new repository on the command line
echo "# Jc" >> README.md

- 命令：git remote rename pb paul
- 作用：重命名。
- 命令：git remote rm name
- 作用：移除远程仓库。

3.将本地项目推送到远程仓库

- 命令：git push remote-name branch-name
- 作用：将提交对象推送到远程仓库里面。(push的时候会生成远程跟踪分支，后边会深入的解释这个分支)

```

dell@DESKTOP-VIIIAFN MINGW64 ~/Desktop/work (testBranch)
$ git config --global http.sslVerify "false"
dell@DESKTOP-VIIIAFN MINGW64 ~/Desktop/work (testBranch) 成功push
$ git push lol master
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (9/9), 615 bytes | 205.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/QianquanChina/Jc.git
 * [new branch]      master -> master
dell@DESKTOP-VIIIAFN MINGW64 ~/Desktop/work (testBranch)
$ ls
dell@DESKTOP-VIIIAFN MINGW64 ~/Desktop/work (testBranch)
$ 
```

master 1 branch 0 tags

Jc and Jc Git Tag Test 8b40cb3 11 days ago 3 commits

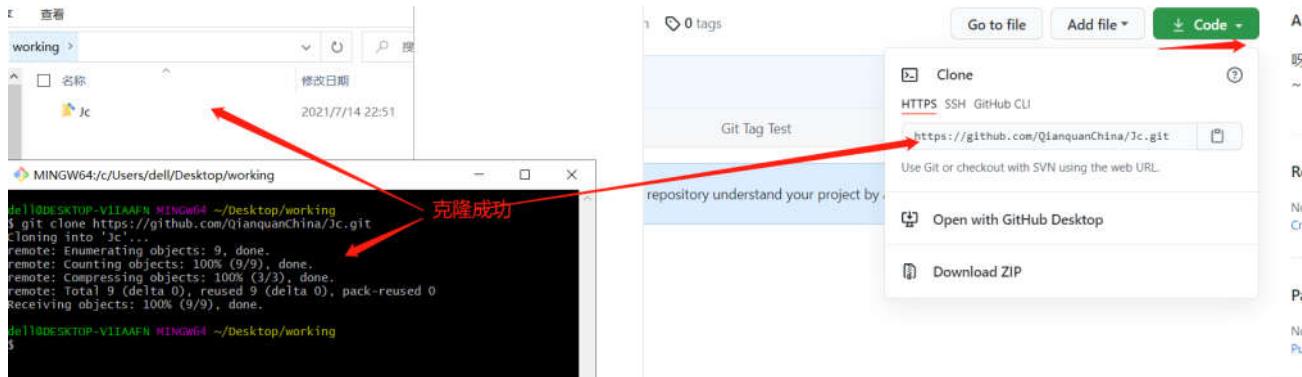
Jc.txt Git Tag Test 11 days ago

Add a README

4.克隆远程仓库

- 命令：git clone url

- 作用：克隆远程仓库。



单纯的克隆是无法进行提交，需要接收到邀请才可以进行提交克隆的仓库已经帮我起好了别名，当然我们也可以修改掉哈。

5. 邀请成员加入该项目

QianquanChina / Jc

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags

Jc and Jc Git Tag Test 8b40cb3 11 days ago 3 commits

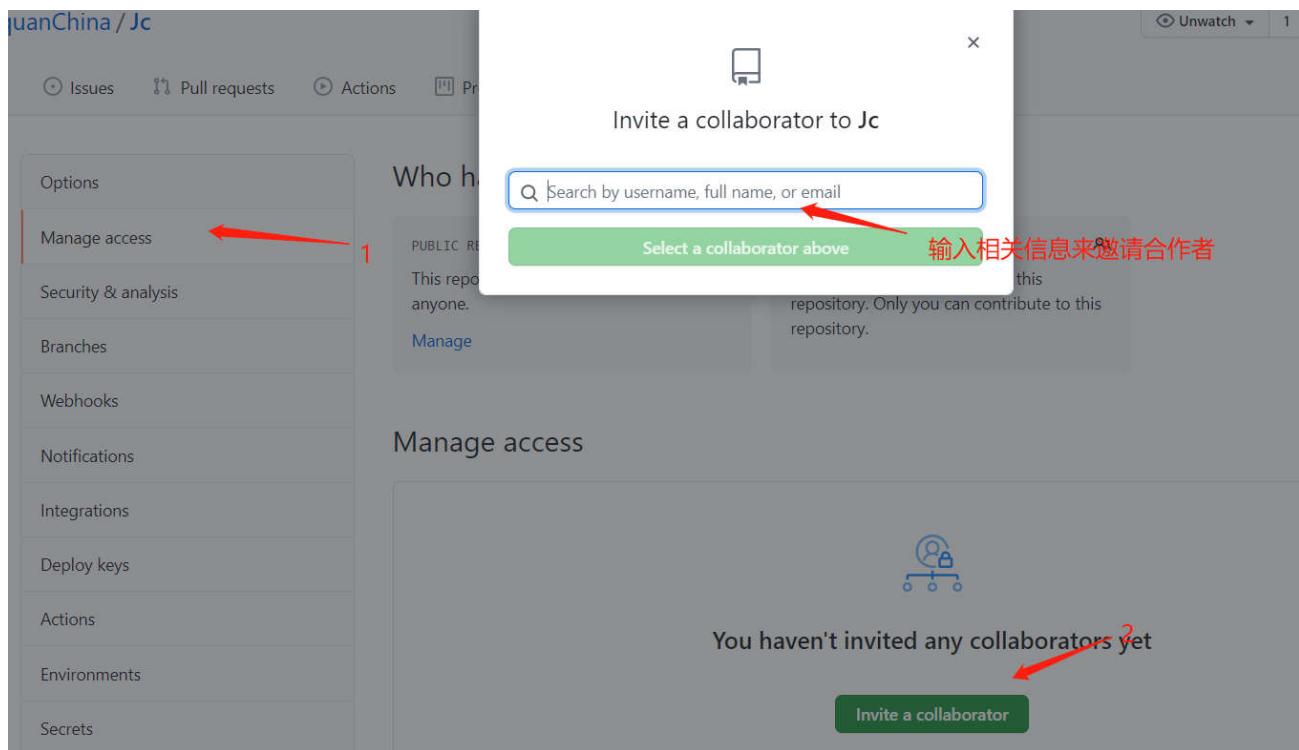
Jc.txt Git Tag Test 11 days ago

About
哇塞~ 笔记更新到Github啦 哪啦~~~~~

Help people interested in this repository understand your project by adding a README. Add a README

Releases
No releases published Create a new release

Packages



在我们注册GitHub的时候有填写的邮箱，被邀请者会收到一封邮件
打开邮件根据操作即可成为该项目的成员。

5.项目负责人来更新成员所提交的内容

- 命令：git fetch [remote-name]
- 作用：克隆远程仓库。

这个命令会访问远程仓库，从中获取本地仓库还没有的数据。
需要注意，这个命令并不会将数据自动合并或者修改到当前工作，
我们需要进行收到合并才可以。

三、远程跟踪分支

远程跟踪分支是远程分支状态的引用。他们是你不能移动的本地分支，当你做任何网络通信操作时候，它们会自动移动。

当克隆一个仓库时，会自动生成一个master本地分支并且已经跟踪了对应的远程跟踪分支。

当我们新创建一个分支的时候，可以指定该分支跟踪哪一个远程跟踪

分支,通过下面的命令:1.git checkout -b 本地分支名 远程跟踪分支名 2.git checkout --track 远程跟踪分支名。

```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git dnf
* 8b40cb3 (HEAD -> master, lol/master) Git Tag Test ←
* 2cbe0f V2 Commit
* 4273cb3 (tag: V1.0) V1 Commit

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master) ← 远程跟踪分支
$ ls
clw.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ echo '123' > clw.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    clw.txt

nothing added to commit but untracked files present (use "git add" to track)

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git add clw.txt
warning: LF will be replaced by CRLF in clw.txt.
The file will have its original line endings in your working directory

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git commit -m "clw V1"
[master 2f168c5] clw V1
 1 file changed, 1 insertion(+)
 create mode 100644 clw.txt

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git status
On branch master
nothing to commit, working tree clean ← 并不会自己移动

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git dnf
* 2f168c5 (HEAD -> master) clw V1 ←
* 8b40cb3 (lol/master) Git Tag Test
* 2cbe0f V2 Commit
* 4273cb3 (tag: V1.0) V1 Commit

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
```

```
de11@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git push lol master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 259 bytes | 129.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/QianquanChina/Jc.git
  8b40cb3..2f168c5  master -> master

de11@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git dnf
* 2f168c5 (HEAD -> master, lol/master) clw V1
* 8b40cb3 Git Tag Test
* 2cbe0f V2 Commit
* 4273cb3 (tag: V1.0) V1 Commit
```

自己进行了移动

```
de11@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git dnf
* 2f168c5 (HEAD -> master, lol/master) clw V1
* 8b40cb3 Git Tag Test
* 2cbe0f V2 Commit
* 4273cb3 (tag: V1.0) V1 Commit
```

大功告成！

```
de11@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git branch -u lol/master
Branch 'master' set up to track remote branch 'master' from 'lol'.
```

```
de11@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git push
Everything up-to-date
```

上图我们发现使用git push 的命令很长,需要加上远程分支和本地分支名这样太麻烦了，我们需要进行优化一下。如果可以将本地分支和这个远程分支进行绑定起来那么我们就可以直接使用git push命令了。

通过**git branch -u 远程跟踪分支名** 即可将当前所在本地分支与远程分支进行跟踪起来,之后就可以通过git push来进行操作啦！！

四、解决冲突

我们再本地仓库写代码会用冲突，那我们将我代码存放再远程仓库和同事一起合作来完成项目，这样也会有冲突我们使用git push(推送代码到远程仓库)、git pull(从远程仓库获取代码) 这都会产生冲突，那么下面就来看看是怎么产生冲突以及如果进行解决冲突。

- git push 造成的冲突

如果两个人再某一个文件动了同一个地方，然后再进行git push 那么这样肯定会产生冲突的。通过下面的图可以更好的理解。

The image shows two terminal windows side-by-side. The left terminal window (MINGW64:/c/Users/dell/Desktop/work) shows a file named Jc.txt with content V1, V2, V333333. A red arrow points from the line 'V333333' to the line 'V344444' in the right terminal window, with the text '修改了同样的位置' (Modified the same position) above it. The right terminal window (MINGW64:/c/Users/dell/Desktop/working/Jc) shows the same file with content V1, V2, V344444. A red arrow points from the line 'V344444' back to the line 'V333333' in the left window, with the text '提示产生了冲突' (Conflict detected) above it. Both terminals show the command history and output of git commit and git push.

```
dell@DESKTOP-VIIIAAFN MINGW64 ~/Desktop/work (master)
$ vim Jc.txt
ca
dell@DESKTOP-VIIIAAFN MINGW64 ~/Desktop/work (master)
$ cat Jc.txt
V1
V2
V333333

dell@DESKTOP-VIIIAAFN MINGW64 ~/Desktop/work (master)
$ git commit -a -m "8888"
[master d56897d] 8888
 1 file changed, 1 insertion(+), 1 deletion(-)

dell@DESKTOP-VIIIAAFN MINGW64 ~/Desktop/work (master)
$ git push
To https://github.com/qianquanChina/Jc.git
! [rejected]          master --> master (fetch first)
error: failed to push some refs to https://github.com/qianquanChina/Jc.git
hint: Updates were rejected because the remote contains work that you
do
hint: not have locally. This is usually caused by another repository
pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

dell@DESKTOP-VIIIAAFN MINGW64 ~/Desktop/work (master)

dell@DESKTOP-VIIIAAFN MINGW64 ~/Desktop/working/Jc (master)
$ vim Jc.txt
dell@DESKTOP-VIIIAAFN MINGW64 ~/Desktop/working/Jc (master)
$ cat Jc.txt
V1
V2
V344444

dell@DESKTOP-VIIIAAFN MINGW64 ~/Desktop/working/Jc (master)
$ git commit -a -m "777"
[master 19914f0] 777
 1 file changed, 1 insertion(+), 1 deletion(-)

dell@DESKTOP-VIIIAAFN MINGW64 ~/Desktop/working/Jc (master)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 264 bytes | 264.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/QianquanChina/Jc.git
 1d6d7e6..19914f0 master --> master

dell@DESKTOP-VIIIAAFN MINGW64 ~/Desktop/working/Jc (master)
$ |
```

The image shows a single terminal window (MINGW64:/c/Users/dell/Desktop/work) displaying the file Jc.txt. The content is shown as a merge result:
V1
V2
<<<<< HEAD
V333333
=====
V344444
>>>>> 19914f086d2d8836aca42a969ecdf9cb7390bc4b
~
~
~
~
~
~
~
The text '删除 保存，并且执行git add 命令则意味着冲突解决完毕' (Delete, save, and run git add to resolve the conflict) is overlaid in red text with arrows pointing to the merge markers (<<<<< and >>>>>) and the separator line '=====.'

```
V1
V2
<<<<< HEAD
V333333
=====
V344444
>>>>> 19914f086d2d8836aca42a969ecdf9cb7390bc4b
~
~
```

```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master|MERGING)
$ vim Jc.txt
```

```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master|MERGING)
$ cat Jc.txt
```

```
V1
```

```
V2
```

```
V333333
```

```
V344444
```

1.解决冲突

```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master|MERGING)
```

```
$ git status
```

```
On branch master
```

```
Your branch and 'lol/master' have diverged,
and have 1 and 1 different commits each, respectively.
```

```
(use "git pull" to merge the remote branch into yours)
```

```
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: Jc.txt
```

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

```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master|MERGING)
$ git add Jc.txt
```

2.标记我们解决完冲突了，下面需要进行提交

```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master|MERGING)
```

```
$ git commit -m " finsh"
```

```
[master 46b4435] finsh
```

```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
```

```
$ git status
```

```
On branch master
```

```
Your branch is ahead of 'lol/master' by 2 commits.
```

```
(use "git push" to publish your local commits)
```

```
nothing to commit, working tree clean
```

3.冲突解决完毕，并推送到远程仓库

```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git push
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 531 bytes | 265.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/QianquanChina/Jc.git
  19914f0..46b4435  master -> master
```

```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
```

- git pull 造成的冲突

如果远程库的一个文件的第6行有内容，并且你本地代码也刚好写到第六行，然后去执行git pull 那么此时就是出现冲突，但是我们一般不会去pull 因为本地都没提交呢但是这个也是存在的一种可能性。(是不是感觉这个和上面的很熟悉哈哈,其实是一样的哈哈)

```
dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ cat Jc.txt
V1
V2
V333333
V999999

dell@DESKTOP-V1IAAFN MINGW64 ~/Desktop/work (master)
$ git pull
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 0), reused 6 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), 561 bytes | 16.00 KiB/s, done.
From https://github.com/QianquanChina/Jc
  46b4435..a0c1858  master      -> lol/master
error: Your local changes to the following files would be overwritten
by merge:
  Jc.txt
Please commit your changes or stash them before you merge.
Aborting
Updating 46b4435..a0c1858
```

出现冲突了，并且告诉我们应该如何进行解决，需要提交然后通过git pull 拉取下来等等，和上面解决的方案是一样滴

五、结尾

至此呢Git就完结撒花，仅此记录一下学习过程，如果有错误还请指出来，感谢观众老爷的赏脸。

若想获得上述内容的PDF版本移步到GitHub下载。

地址：[Git 学习笔记专区](#)