Contents

[一、git diff使用 2](#_Toc12209202)

[二、git bash下操作文件/文件夹命令 4](#_Toc12209203)

[三、vi的使用（未完待续） 5](#_Toc12209204)

[四、上传本地仓库项目/文件至云端 7](#_Toc12209205)

[五、下载代码到本地 9](#_Toc12209206)

一、git diff使用

HEAD 表示当前版本，也就是最新的提交。上一个版本就是 HEAD^ ，上上一个版本就是 HEAD^^ ，

往上100个版本写100个 “ ^ ” 比较容易数不过来，所以写成 HEAD~100 。HEAD~2 相当于 HEAD^^ 。

我们在 Git 版本回退里  可以利用 HEAD 关键字进行指定版本的回退。

那么对于 $ git diff HEAD -- <filename>  命令是不是也能操作不同的情况？

自己试了试一个例子：

命令 $ git diff 查看对比两次文件内容具体修改了什么。

命令 $ git diff HEAD -- <filename> （“--”前后有空格），可以查看工作区和版本库里面最新版本的区别。

步骤：

第一步：在Git里添加test.txt。最初内容是数字“1”。并commit 提交到版本库里。

第二步：增加数字“2”，并提交。

第三步：增加数字“3”，并提交。

第四步：增加数字“4”，并提交。

第五步：增加数字“5”，并提交。

最后文件内容为：

假设仓库里已提交的有五个版本，依次提交的是A、B、C、D、E 。

在都已提交的版本库中：

执行 $ git diff <filename> 命令，命令行窗口不会输出文件的改动信息。

执行 $ git diff HEAD -- <filename>  命令同样也不会输出文件的改动信息。因为当前工作区未做改动。

执行 $ git diff HEAD^ -- <filename> 命令则可以查看最近两次提交版本的区别（版本E和版本D的差别——增加数字“5”）

执行 $ git diff HEAD^^ -- <filename> 命令则可以查看最近一次提交和最近一次提交的上上个版本的区别（版本E和版本C的差别——增加数字“4”和“5”）

执行 $ git diff HEAD^^^ -- <filename>  命令则可以查看版本E和版本B的差别——增加数字“3”，“4”和“5”。

执行 $ git diff HEAD~4 -- <filename>  命令则可以查看版本E和版本A的差别——增加数字“2”，“3”，“4”和“5”。等同于HEAD^^^^

现在增加第六步：再增加数字“6”。分三种情况，还未执行 add、add 后还未commit、add 并commit（版本F）。

**---------------------**

**作者：Ashe And Wine**

**来源：CSDN**

**原文：https://blog.csdn.net/asheandwine/article/details/78982919**

**版权声明：本文为博主原创文章，转载请附上博文链接！**

二、[git bash下操作文件/文件夹命令](https://www.cnblogs.com/SamWeb/p/6516784.html)

　　1， cd : change directory的简写，改变目录的意思，就是切换到哪个目录下， 如 cd e:\fff  切换 E 盘下面的fff 目录。

　　当我们用cd 进入文件夹时,我们可以使用 通配符\*, cd f\*,  如果E盘下只有一个f开头的文件夹,它就会进入到这个文件夹.

　　2， cd .. 回退到上一个目录。我们在写js，引入文件时，.. 表示的就是上一个目录, 所以 cd .. 回退到上一个目录就很好理解了。注意，cd 和两个点点..之间有一个空格,

　　3， pwd : print working directory, 打印工作目录，它会显示我们当前所在的目录路径。

　　4， ls: list, 列出当前目录中的所有文件，     只不过ll(两个ll)列出的内容更为详细。

　　5， touch : 新建一个文件 如 touch index.js 就会在当前目录下新建一个index.js文件。

　　6，rm:  remove，删除一个文件, rm index.js 就会把index.js文件删除.

　　7,  mkdir: make directory 新建一个目录,就是新建一个文件夹. 如mkdir src 新建src 文件夹.

　　8,  rm -r :  删除一个文件夹， r (recusive 是递归的意思)， 删除用的就是递归，先删除文件夹里面的内容，再删除文件夹。 rm -r src 删除src目录。

　　9,  mv 移动文件, mv index.html src   index.html 是我们要移动的文件, src 是目标文件夹,当然, 这样写,必须保证文件和目标文件夹在同一目录下.

　　10, reset 清屏，把git bash命令窗口中的所有内容清空。

　　11：exit 可以直接退出窗口，就不会再鼠标按窗口上的https://images2018.cnblogs.com/blog/1013082/201808/1013082-20180817111901676-736949957.png来关闭窗口了。

****三、vi的使用（未完待续）****

**新建文件有两种方式：**

**（1），touch+文件名，直接新建一个文件**

**（2），vi+文件名，新建一个文件并进入编辑状态（如果文件已存在，则直接进入编辑状态）**

vi其实是linux的一个文本编辑器，所以 vi+文件名 后，其实是进入vi程序了。vi有两种模式，编辑模式和命令模式



当我们 vi+文件名 后，会看到上图，默认处于命令模式，按 **:**+命令 可以执行相关操作。



在命令模式下，我们可以直接按 i ,此时就会切换到编辑模式，如上图，下方有个insert，此时可以直接修改文本内容。

在编辑模式下，按 esc 就可以切换到命令模式。

下面介绍几个常用的命令，都是在命令模式下，输入 :+命令使用

1，q 退出编辑，如果文本内容被修改过，则会报错

2，q! 强制退出编辑，如果文本内容被修改过，会丢弃此次的修改

3，x 退出编辑并保存

还有一些命令可以在命令模式下直接执行，不需要在前面输入 :符号

比如dd命令会删除当前行，i命令会切换到编辑模式。也正是因为这些命令的存在，所以在需要修改文本前，一定要记得先按 i 进入编辑模式，不然命令和文本输入会搞混。

vi是有很多命令的，这边只介绍命令的使用方式

四、上传本地仓库项目/文件至云端

第一步：本地初始化Git版本库

git init

第二步：链接码云（云端）

git remote add orgin "你的远程仓库地址"（复制链接后结尾是.git,如果没有记得加 .git）

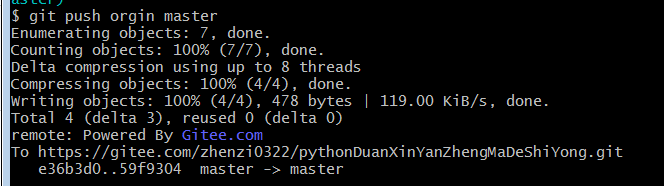
第三步：上传项目

git add . --将项目中的所有文件上传

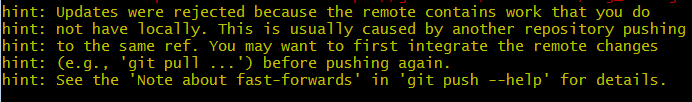
git commit -m ‘对上传文件的注释‘

git push orgin master --正式上传至码云中，

提示如图下面的信息说明上传成功了：



在使用 git push orgin master 时可能会出现如下这种情况:



这种情况是本地的版本库与云端上的版本库不一致造成的.云端上有两个文件:README.md

我们可以使用命令拿下来:

git pull --rebase orgin master

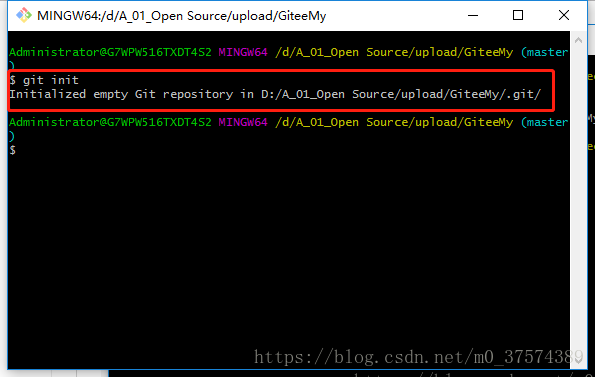
然后再使用命令上传:

git push orgin master

五、下载代码到本地

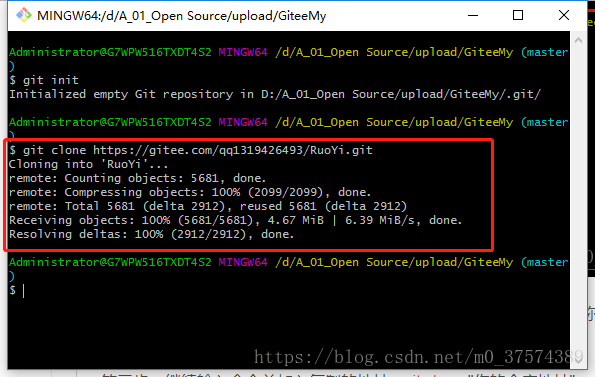
第一步：我们需要先创建一个本地的版本库（其实也就是一个文件夹）。

第二步：选中文件件右键Git bash here进入控制面板，输入命令git init 初始化化文件夹，把这个文件夹变成Git可管理的仓库



这时里面多了个.git文件夹，它是Git用来跟踪和管理版本库的。如果你看不到，是因为它默认是隐藏文件，那你就需要设置一下让隐藏文件可见。

第三步：继续输入命令并加入复制的地址->git clone "你的仓库地址"，点击回车。如下图：



好了，你到你本地文件夹当中就可以看到这个项目了

