git常用命令行

（一）开发流程

1，git pull 开发前切换到主分支，拉一下最新代码

2，git checkout -b 【分支名】 新建分支并切换到新分支

3，git status 查看修改状态（变红）

【可以 git diff 查看所有修改】

4，git add 【文件路径】 git add . 提交所有修改

【可以 git reset 撤销add提交】

5，git status 查看修改状态（变绿）

6，git commit -m "【本次修改备注】" 暂存到本地仓库

7，git push origin 【分支名】 提交到远程

（二）经常用的一些

【分支】

git branch 查看在哪个分支，列出所有本地分支

git checkout 【分支名】 切换分支

git branch -D【分支名】 删除分支

git push origin --delete 【分支名】

【查看信息，修改】

git status 查看所有修改的文件路径

git diff 查看所有文件的具体修改

git diff【文件路径】 查看当前文件的具体修改

【撤销】

git reset 撤销add添加

git reset【文件路径】

git checkout . 撤销所有修改

git checkout 【文件路径】撤销该路径的修改

【暂存】修改但不想提交

git stash 暂存

git stash list 列出所有暂存

git stash apply 撤销暂存

git stash drop 删除暂存

【回滚】

git checkout 【版本号】暂时回滚到某个版本

git checkout 【主分支】master 回滚到主分支

【commit 合并】

git log 查看所有commit

git status 只能查看未传送提交的次数

git cherry -v 只能查看未传送提交的描述/说明

git log master ^origin/master 则可以查看未传送提交的详细信息

git rebase -i 进去commit编辑界面

esc+shift+：wq

squash s 选中，会将当前commit与上一个commit合并

fixup f 与squash相同，但不会保存当前commit的提交信息

【git ---add】

git

git checkout . 清掉工作区的改动，没有add前

git add . 添加到暂存区

git add –u 不添加新建文件

git reset 重置 暂存区的回退到工作区 || 回退版本

git stash pop

git stash apply

git stash drop

git stash clear

git stash show

git log

git reset --hard 3628164 回去版本 commit id

git reflog 每一次命令，里面有commmit id

rm +【文件名】 删除文件名

git rm +[文件名] 确认删除文件

git status

on branch cxl\_new\_1204

Unmerged paths:

(use "git reset HEAD <file>..." to unstage) 暂存区的修改撤销掉

(use "git add <file>..." to mark resolution) 标记 决定

both modified: src/components/index.vue 两个更改

【git tag 打标签】

$ git tag v0.9 6224937

$ git show v0.9

commit 622493706ab447b6bb37e4e2a2f276a20fed2ab4

Author: Michael Liao <askxuefeng@gmail.com>

Date: Thu Aug 22 11:22:08 2013 +0800

add merge

还可以创建带有说明的标签，用-a指定标签名，-m指定说明文字：

$ git tag -a v0.1 -m "version 0.1 released" 3628164

...

3， 本地分支，属于本地仓库，一个仓库可以有很对分支，

4， 肯定不是全量推送到远程的，是通过对比commit的记录，

如果本地高于远程就直接把多出来的commit给怼上去，

如果本地的这几个commit和远程的commit有冲突的部分就merge，

然后根据提交时间排序再新建一个merge的commit记录再怼上去

5， 先commit 再pull 再push 是为了应对多人合并开发情况

1， commit是为了告诉git我这次提交改了那些东西，

不然你只是改了但是git不知道你改了，也就无从判断比较

2， pull是为了本地commit和远程commit的记录对比，

git是按照文件的行数操作进行对比的，

如果同时操作了某文件的同一行那么就会产生冲突，git也会把冲突给标记出来

这个时候就需要先把和冲突的那个人拉过来问问保留谁的代码，

然后在git add && git commit && git pull 这三连，再次pull一次是为了

防止再你们协商的时候另一个人又提交一版东西，如果真发生了那流程重复一遍，

通常没有冲突的时候就直接给你的合并啦，不会把你的代码给覆盖掉

3， A 1 2 3 git pull 就会丢失,覆盖文件ps： git commit git pull 顺序 -\_\_\_\_\_\_\_\_\_\_\_}\_\_\_\_{\_\_\_\_\_\_\_\_\_\_\_

B git pull 4 在本地的版本3的时候修改了A写过的代码，-🡪版本4

6，两个互相合并的区别：

A -> B 的时候 B上面会有一个merge\_commit 的信息，这个时候B的合并状态而A未合并状态，如果现在没有发生任何改动执行 B -> A 就直接切过去了，连 merge\_commit 都不会生成了