Git

git log –oneline ---- 查看修改

git status –s

被管理的文件有四种状态

1. Untracked 文件未被追踪 使用 git init <filename> 可将其状态设为Staged
2. Unmodified 当修改的文件被提交时，状态会变成Unmodified

有两个提交修改的方式：

1. 从Modified—(git add)—Staged—(git commit -m ‘xx’)-Unmodified
2. 从Modified—(git commit -am ‘xx’)—Unmodified
3. Modified 当文件被编辑修改时，状态由Unmodified--Modified
4. Staged 可被提交状态 Modified—(git add)—Staged

查看文件修改日志

1. git diff 文件未被提交时查看修改记录的方式，查看到的是当下的修改记录
2. git diff –cached 文件已经被提交时查看修改记录的方式，它只能查看上一次提交时的修改，不能查看当下的修改记录
3. git diff HEAD 能同时查看未被提交时与已被提交时的修改记录

将修改合并到上一个修改（修改上一次的修改信息）

git add test.py

git commit –amend –no-edit

修改近三次的信息

git rebase -I HEAD~3

然后在日志信息中修改

git commit –amend

git rebase –continue

git commit –amend -s 正式修改

回到过去reset(针对整个版本库)

1. 有时将修改的文件add提交后想回到add之前的状态

git reset test.py

1. 回到commit之前

git reset –hard HEAD 查看当前HEAD指向

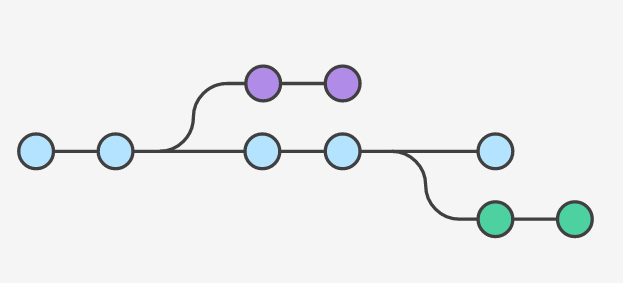
git reset –hard HEAD^ 回到上一次

git reset –hard <指针> 用git log 或者 git reflog(可查看全部)

回到过去checkout(针对单个文件)

git checkout <指针> -- test.py

分支(branch)



git log –oneline –graph

git branch dev 建立dev分支

git branch 查看当前拥有的分支 \*代表当前所处分支

git checkout dev 将HEAD指向dev分支 (git checkout -b dev 创建并切换到dev分支)

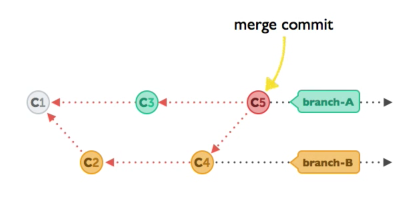
提交修改后切换回master

git merge –no-ff -m “merge dev to master” dev 将dev中的修改推送到master

git branch -d dev 删除dev分支

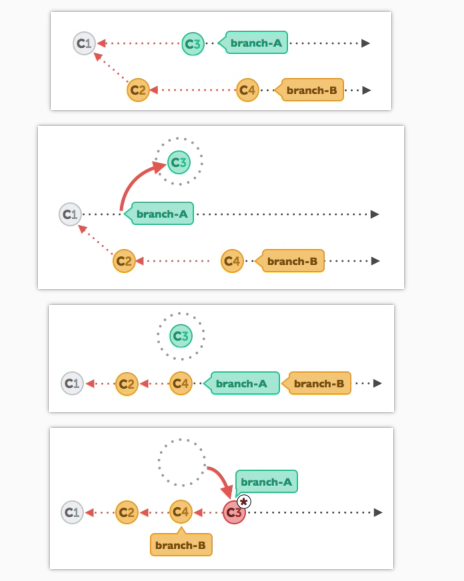
merge分支冲突

（想象不仅有人在做开发版 dev 的更新, 还有人在修改 master 中的一些 bug. 当我们再 merge dev 的时候, 冲突就来了. 因为 git 不知道应该怎么处理 merge 时, 在 master 和 dev 的不同修改）



解决方法是手动合并两者的不同，再提交！

rebase(重新基于)分支冲突



其实就是将某个分支重新基于另一个分支的末端，自己分支的历史将被修改

临时修改(stash)

git stash(将手头的工作暂存)

git stash list(回复暂存)