**git 分布式版本控制系统**

版本库提示

1. 新建一个文件

mkdir 文件名

cd 文件名

psw（显示当前目录）

注：路径不能用中文

1. 变成可以管理的库 ：git init
2. 文件添加至仓库：git add <file>
3. 文件提交到仓库：git commit –m “提交说明”

其他提示

1. 命令查看结果：git status
2. 查看difference：git diff <file>
3. 查看最近到最远的log：git log
4. 当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时：git checkout –file
5. 当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步
   1. git reset HEAD file
   2. git checkout –file
6. 删除文件：git rm

回退版本

1. Git必须知道当前版本是哪个版本
2. 用HEAD表示当前版本，上一个版本是HEAD^,上上版本HEAD^^,往上100个版本者 HEAD~100
3. 回退到上一个版本：git reset HEAD^
4. 后悔回到上一个版本，要回到刚刚的版本，只要命令行未关闭找到commit id
   1. 命令行未关闭找到commit id

git reset commit id

* 1. 命令行关闭了

git reflog 找到commit id

git reset commit id

远程仓库

1. GitHub上的learngit仓库 和本地仓库关联

git remote add origin [git@github.com:用户名/ learngit.git](mailto:git@github.com:用户名/%20learngit.git)

1. 本地库的所有内容推送到远程库上

git push –u origin master(第一次)

git push origin master(之后)

1. 从远程库克隆

git clone [git@github.com:用户名/ learngit.git](mailto:git@github.com:用户名/%20learngit.git)

分支管理

1. 查看当前分支

git branch

1. 创建并切换分支
   1. 创建分支：git branch <name>
   2. 切换分支：git checkout<name>
   3. 创建+切换: git checkout –b <name>
2. 合并分支dev

git merge <name>

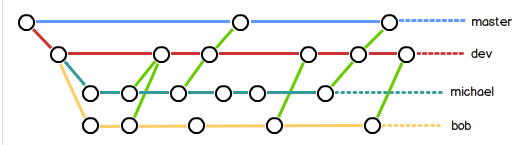
1. 删除分支

git branch –d <name>

1. 解决冲突
   1. 使用git status 找冲突文件
   2. 直接修改冲突，解决冲突
   3. 在提交add + commit
   4. 看分支的合并情况：git log --graph --pretty=oneline --abbrev-commit
2. 分支策略

在实际开发中，我们应该按照几个基本原则进行分支管理：

1. 首先，master分支应该是非常稳定的，也就是仅用来发布新版本，平时不能在上面干活；
2. 那在哪干活呢？干活都在dev分支上，也就是说，dev分支是不稳定的，到某个时候，比如1.0版本发布时，再把dev分支合并到master上，在master分支发布1.0版本；
3. 你和你的小伙伴们每个人都在dev分支上干活，每个人都有自己的分支，时不时地往dev分支上合并就可以了。
4. 所以，团队合作的分支看起来就像这样：



1. Bug分支

修复bug时，我们会通过创建新的bug分支进行修复，然后合并，最后删除。

当手头工作没有完成时(并不能提交)，先把工作现场git stash一下，然后修复bug，

修复后,再git stash pop,回到工作现场

1. Feature分支

开发一个新功能，最好新建一个feature分支，在上面开发，完成后，合并，最后，删除该feature分支

如果要丢弃一个没有被合并过的分支，通过git branch –D <name> 强行删除

1. 多人协作
   1. 查看远程库信息：git remote -v
   2. 从本地推送分支，使用git push origin branch-name,如果推送失败pit pull 抓取远程新提交
   3. 在本地创建和远程分支对应的分支，使用

git checkout –b branch-name origin/branch-name

本地和远程分支的名称最好一致

* 1. 建立本地分支和远程分支对应的分支，使用

git branch –set –upstream branch-name orgin/branch-name

* 1. 从远程抓取分支，使用git pull，如果有冲突，要先处理冲突

标签管理

1. 新建一个标签，默认HEAD，也可以指定commit id ： git tag <name>
2. 指定标签信息：git tag –a<tagname> -m “message”
3. 用PGP签名标签：git tag –s<tagname> -m “message”
4. 查看标签：git tag
5. 查看标签信息：git show<tagname>
6. 删除标签
7. 先本地删除：git tag –d <tagname>
8. 从远程删除：git push origin:refs/tags/<tagname>
9. 推送标签
10. 推送一个本地标签：git push origin<tagname>
11. 推送全部未推送过的本地标签：git push origin –tags

其他

1. 忽略特殊文件
   1. 忽略某些文件时，需要编写.gitignore
   2. .gitignore本身要放到版本库里，并且可以对.gitignore做版本管理
2. 配置别名

告诉git, st就表示status

git config --global alias.st status

注：--global是针对当前用户起作用，如果不加，那只针对当前仓库

1. 配置文件
   1. 位置： .git/config文件中
   2. 别名都可以配置在里面
2. 创建ssh密钥
   1. ssh-keygen