**Git学习文档**

1、初始化一个Git仓库，使用git init命令。

添加文件到Git仓库，分两步：

* 第一步，使用命令git add <file>，注意，可反复多次使用，添加多个文件；
* 第二步，使用命令git commit，完成。
* 要随时掌握工作区的状态，使用git status命令。

如果git status告诉你有文件被修改过，用git diff可以查看修改内容。

HEAD指向的版本就是当前版本，因此，Git允许我们在版本的历史之间穿梭，使用命令git reset --hard commit\_id。

穿梭前，用git log可以查看提交历史，以便确定要回退到哪个版本。

要重返未来，用git reflog查看命令历史，以便确定要回到未来的哪个版本。

git add命令实际上就是把要提交的所有修改放到暂存区（Stage），然后，执行git commit就可以一次性把暂存区的所有修改提交到分支

为什么Git比其他版本控制系统设计得优秀，因为Git跟踪并管理的是修改，而非文件

场景1：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，用命令git checkout -- file。

场景2：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，第一步用命令git reset HEAD file，就回到了场景1，第二步按场景1操作。

场景3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考[版本回退](https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/0013744142037508cf42e51debf49668810645e02887691000" \t "https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/_blank)一节，不过前提是没有推送到远程库。

git checkout其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”

命令git rm用于删除一个文件。如果一个文件已经被提交到版本库，那么你永远不用担心误删，但是要小心，你只能恢复文件到最新版本，你会丢失最近一次提交后你修改的内容

#### 添加远程库

要关联一个远程库，使用命令git remote add origin git@server-name:path/repo-name.git；

关联后，使用命令git push -u origin master第一次推送master分支的所有内容；

此后，每次本地提交后，只要有必要，就可以使用命令git push origin master推送最新修改；

要克隆一个仓库，首先必须知道仓库的地址，然后使用git clone命令克隆。$ git clone git@github.com:puetiH/gitskills.git

Git支持多种协议，包括https，但通过ssh支持的原生git协议速度最快。

Git鼓励大量使用分支：

查看分支：git branch

创建分支：git branch <name>

切换分支：git checkout <name>

创建+切换分支：git checkout -b <name>

合并某分支到当前分支：git merge <name>

删除分支：git branch -d <name>

#### 解决冲突

当Git无法自动合并分支时，就必须首先解决冲突。解决冲突后，再提交，合并完成。

$ git merge feature1

用git log --graph命令可以看到分支合并图。

用带参数的git log也可以看到分支的合并情况：$ git log --graph --pretty=oneline --abbrev-commit

合并分支时，加上--no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并

#### 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，先把工作现场git stash一下，然后去修复bug，修复后，再git stash pop，回到工作现场。

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

#### 多人协作

多人协作的工作模式通常是这样：

首先，可以试图用git push origin branch-name推送自己的修改；

如果推送失败，则因为远程分支比你的本地更新，需要先用git pull试图合并；

如果合并有冲突，则解决冲突，并在本地提交；

没有冲突或者解决掉冲突后，再用git push origin branch-name推送就能成功！

如果git pull提示“no tracking information”，则说明本地分支和远程分支的链接关系没有创建，用命令git branch --set-upstream-to=origin/<branch> dev

### 小结

查看远程库信息，使用git remote -v；

本地新建的分支如果不推送到远程，对其他人就是不可见的；

从本地推送分支，使用git push origin branch-name，如果推送失败，先用git pull抓取远程的新提交；

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；

建立本地分支和远程分支的关联，使用git branch --set-upstream branch-name origin/branch-name；

从远程抓取分支，使用git pull，如果有冲突，要先处理冲突。

#### 创建标签

命令git tag <name>用于新建一个标签，默认为HEAD，也可以指定一个commit id；

git tag -a <tagname> -m "blablabla..."可以指定标签信息；

git tag -s <tagname> -m "blablabla..."可以用PGP签名标签；

命令git tag可以查看所有标签。

window 7 按照如下配置后，就可以了。 1、需到官网www.gnupg.org 下载对应系统的加密软件； 2、根据[http://www.ruanyifeng.com/blog/2013/07/gpg.html](http://www.ruanyifeng.com/blog/2013/07/gpg.html" \t "https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/_blank) 设置好，并生成自己的密钥； 3、在git上（1）配置gpg程序位置，先找到GnuPG的安装目录下gpg2.exe的路径，默认是C:\Program Files (x86)\GNU\GnuPG。 配置gpg.program的位置： 在Git Bash中进行配置 $ git config –global gpg.program “C:\Program Files (x86)\GNU\GnuPG\gpg2.exe” （2）配置signingkey： 查看sigingkey，在Windows命令终端下为gpg –list-keys 在Git Bash中进行配置 $ git config user.signingkey F08971B8 F08971B8这个是别人的公钥ID，需替换上自己的

#### 创建标签

命令git tag <name>用于新建一个标签，默认为HEAD，也可以指定一个commit id；

git tag -a <tagname> -m "blablabla..."可以指定标签信息；

git tag -s <tagname> -m "blablabla..."可以用PGP签名标签；

命令git tag可以查看所有标签。

命令git push origin <tagname>可以推送一个本地标签；

命令git push origin --tags可以推送全部未推送过的本地标签；

命令git tag -d <tagname>可以删除一个本地标签；

命令git push origin :refs/tags/<tagname>可以删除一个远程标签。

在GitHub上，可以任意Fork开源仓库；git clone git@github.com:michaelliao/bootstrap.git

自己拥有Fork后的仓库的读写权限；

可以推送pull request给官方仓库来贡献代码。

忽略某些文件时，需要编写.gitignore；

.gitignore文件本身要放到版本库里，并且可以对.gitignore做版本管理！

有全局／用户／项目 3种gitconfig

搭建Git服务器非常简单，通常10分钟即可完成；

要方便管理公钥，用[Gitosis](https://github.com/sitaramc/gitolite" \t "https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/_blank)；

要像SVN那样变态地控制权限，用[Gitolite](https://github.com/sitaramc/gitolite" \t "https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/_blank)。

私有的git服务器,用gogs还是比较方便的,简单轻量功能全