1、安装

安装完成后 运行 git->git bash

设置用户名与邮箱

$ git config --global user.name "Your Name"

$ git config --global user.email [email@example.com](mailto:email@example.com)

设置字体风格

$ git config --global color.ui true

1. 概念

A：工作区

B：版本库 隐藏的.git目录

a：暂存区(stage/index)

b：分支（master）





1. 常用命令

|  |  |
| --- | --- |
| git init | 在该文件下创建一个工作区 |
| git add one.txt | 将one.txt 添加到缓存区  添加多个文件时 **git add .** |
| git commit -m "log" | 将修改的文件提交到版本库，并注明日志 |
| git status | 查看仓库当前的状态 |
| git log | 查看历史记录、日志 |
| git log --pretty=oneline | 查看历史记录、日志，只显示一行 |
| git reset --hard HEAD^ | 回退到上一个版本 有几个^ 回退几个版本 |
| git reset --hard HEAD~100 | 回退100个版本 |
| git reset --hard commit\_id | 回退到指定版本 |
| git reflog | 重返未来，查看回退记录 |
| git checkout -- one.txt | 撤销未提交的文件修改   1. 未被添加到缓存区的文件修改，还原到版本库状态   在工作区删除文件，属于此类。   1. 在暂存区未被commit的文件修改，还原到暂存区状态 |
| git reset HEAD one.txt | 将在暂存区修改的文件，回退到工作区，但保留在暂存区的修改 |

1. 删除文件
   1. 在工作区中删除
      1. 删除：git re test.txt git commit -m “delete file”
      2. 还原：git checkout – test.txt
   2. 命令删除
      1. git rm test.txt git commit -m “delete test.txt”
2. 远程仓库

//删除远程分支origin

git remote rm origin (出现fatal: remote origin already exists时，用此命令) .

//创建一个远程分支

git remote add origin <https://github.com/YLJ76/git.git>

1. **更新远程仓库（提交到远程仓库）**

**git push**

**git push -u origin master //提交到指定分支**

1. **更新本地仓库**

**git pull**

1. 分支
   1. 创建并切换到dev分支

git checkout -b dev

等价于 git branch dev

Git checkout dev

* 1. 查看当前分支
     1. git branch
  2. 切换分支
     1. git checkout master //切换到master
  3. 合并分支
     1. git merge dev //将dev分支中的改变合并到当前分支
     2. 合并时，会有一个commit
     3. 默认是使用Fast forward方式合并

这种模式下，合并后删除分支，就会丢掉分支信息

git merge --no-ff -m "merge with no-ff" dev

--on-ff使用非fast forward合并方式

-m "merge with no-ff" commit的备注

* 1. 删除分支
     1. git branch -d dev //删除分支dev

1. 冲突

当当前分支与想要合并的分支都有修改数据时，当前分支合并分支就会出现错误。执行合并操作时，就会将文件数据修改为两个分支的变化，手动将文件修改，后提交数据，删除分支。

1. Bug分支

当前分支没有提交，需要先去修改bug。

先将当前的工作储藏起来。git stach

新建bug分支，在该分支上修改bug，

然后切换到原来的分支上，合并，删除bug分支

恢复分支

A:git stash apply恢复分支，然后git stash drop 删除存储记录

B:git stash pop 恢复分支，删除出存储记录。

git stash list 存储列表

10、Feature分支

开发一个新feature，最好新建一个分支；

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。

11、多人协作

git remote 查看远程仓库的名称，默认是origin

git remote -v 显示详细信息

git push origin master 推送远程仓库origin的master分支上

git checkout -b dev origin/dev 将远程的dev分支检出成自己的dev分支

git push origin dev 推送dev分支到远程仓库

多人协作的工作模式通常是这样：

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

2）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；

3）如果合并有冲突，则解决冲突，并在本地提交；

4）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 branch-name origin/branch-name。

12、标签

A: 切换到需要创建标签的分支

git checkout master

B: 创建标签

git tag v1.0 //创建标签 v1.0

git tag -a v1.0 -m “this is a tag.” 234234 //创建v1.0 备注this is a tag.

-a 指定标签名，-m 备注

git tag -s v1.0 -m "this is a tag." 可以用PGP签名标签；

删除标签 git tag -d v1.0

C: 查看所有标签 git tag

查看具体标签 git show v1.0

D： 补签

i: 找到历史提交的commit\_id

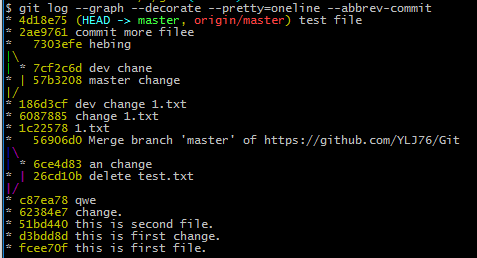
git log --graph --decorate –-pretty=oneline -–abbrev-commit

--graph:通过线条显示各个分支关系

--decorate：在哪个分支上

–pretty=oneline：显示一行

-–abbrev-commit:用commit\_id前7个字符代表（也是唯一的）。



Ii: 对对应的commit\_id打标签

git tag v1.0 c87ea8

注意： 标签不是按时间顺序列出的，而是按字母的顺序排列的。

E: 推送标签

git push origin v1.0 //推送一个标签

git push origin –-tags //推送所有未推送的标签

删除远程标签

1、删除本地标签 git tag -d v1.0

2、删除远程标签 git push origin :refs/tags/v1.0