基础命令

1、基本命令

- git init
  创建一个git仓库（声明）

  > 默认进入 Git 仓库的master分支，即主分支
  > Untracked fies提示，它表示demo仓库中有文件没有被追踪

- git add hit.txt（*可以表示为所有更改文件）
  将hit.txt文件添加到 Git 仓库（添加到临时缓冲区，没有提交到Git仓库）

  > git add *（可以表示为所有更改文件）
  > git add .（添加所有文件）
  > git add 文件夹名/（添加整个文件夹）
  > 删除 add 缓冲 rm

- git commit -m "text commit"
  将新加的文件提交到 Git 仓库

  > commit表示提交，-m表示提交信息，提交信息写在双引号""内

- git status
  查看仓库状态

- git log
  打印 Git 仓库提交日志
- 提交内容中的Author和Email设置
  git config --global user.name "名字"
  git config --global user.email "邮箱"
- **永久性免账号密码输入**
  git config --global credential.helper store

2、进阶命令

- git branch
  查看 Git 仓库的分支情况

  > 其中master分支前的*号表示“当前所在的分支”

- git branch a
  创建一个分支a
- git checkout a
  切换到a分支
- git checkout -b a
  创建分支a的同时，直接切换到新分支a
- git merge a
  切换到master分支，然后输入git merge a命令，将a分支合并到master分支
- git branch -d a
  删除a分支
- git branch -D a
  强制删除a
- git tag v1.0
  为当前分支添加标签

  > 命令git tag即可查看标签记录
  > 通过命令git checkout v1.0即可切换到该标签下的代码状态

初次创建Git

1、clone仓库

直接将远程仓库clone到本地。通过clone命令创建的本地仓库，其本身就是一个 Git 仓库了，不用我们再进行init初始化操作啦，而且自动关联远程仓库。拿上传”编程知识点“文件夹举例，在这个仓库进行修改或者添加等操作，然后commit即可。

1. git clone 地址链接
   在GitHub上进入需要复制的项目，点击clone or download，复制地址链接。
   然后，进入我们准备存储 Git 仓库的目录，进入Git Bash，在命令窗口输入git clone https://github.com/xuzhihao1109/study.git

   > 实现把远程的仓库clone到本地

2. git add 编程知识点/
   把上传编程知识点文件夹复制到本地仓库的study目录下，运行add命令。
3. commit -m "提交的信息说明"
   在本地提交更新内容
4. git push origin master
   将本地仓库的更新内容push到远程仓库

   > 第一次向远程仓库提交代码的时候，需要输入账号及密码进行验证

至此，编程知识点文件夹上传到github的study仓库内。（个人暂时推荐该方法）

2、本地创建Git仓库

1. 本地仓库创建
   通过git ini命令创建
2. 关联GitHub上的远程仓库
   git remote add git_study https://github.com/xuzhihao1109/git_study.git

   > git_study 为远程仓库名

3. 同步远程仓库和本地仓库
   git pull git_study master

   > 当提示git merge master --allow-unrelated-histories时，在命令后加入--allow-unrelated-histories可成功执行

4. 添加文件
   通过git add和git commit把添加的文件提交到git_study远程仓库
5. 上传远程仓库

git push git_study master

在我们向远程仓库提交代码的时候，一定要先进行pull操作，再进行push操作，防止本地仓库与远程仓库不同步导致冲突的问题，尤其是本地提交代码的情况，很容易就出现问题。

更新仓库内容

1、更新修改的文件

例如修改 编程知识点文件夹 内的部分文件名，进行更新。

1. git add *
   在study仓库下，添加所有更新的文件内容
2. git commit -m "commit file name change"
   提交更新的内容信息
3. git push origin master
   更新github的内容

至此完成更新。

分支

1、建立分支

- branch命令来创建分支：git branch
  例如创建名为issue1的分支：git branch issue1

  > 不指定参数直接执行 git branch 命令的话，可以显示分支列表,前面有*的就是现在的分支。

2、切换分支

- checkout命令切换分支：git checkout
  当前在master分支，若要在issue1分支进行提交，需要切换到issue1分支。此时执行 git checkout issue1进行切换。

  > 在checkout命令指定 -b选项执行，可以创建分支并进行切换：git checkout -b

3、合并分支

- 执行merge命令以合并分支：git merge

用于向master分支合并issue1分支的修改。该命令将指定分支导入到HEAD指定的分支。先切换master分支，然后把issue1分支导入到master分支。

> git checkout master
> git merge issue1
> master分支指向的提交移动到和issue1同样的位置。这个是fast-forward（快进）合并。

4、删除分支

- branch命令指定-d选项执行，以删除分支：git branch -d
  既然issue1分支的内容已经顺利地合并到master分支了，现在可以将其删除了：
  git branch -d issue1

> 可以用branch命令来确认分支是否已被删除：git branch

一、并行操作分支

1. 创建2个分支来尝试并行操作。首先创建issue2分支和issue3分支，并切换到issue2分支。
   git branch issue2
   git branch issue3
   git checkout issue2

> 切换分支时，电脑对应的目录也是会刷新变化的。

2. 在issue2分支的myfile.txt添加commit命令的说明，然后提交。

> 修改myfile.txt后，分别调用add和commit命令
> txt内容如下：
> add 把变更录入到索引中
> commit 记录索引的状态

3. 切换到issue3分支，打开myfile.txt档案。由于在issue2分支添加了commit命令的说明，所以issue3分支的myfile.txt里只有add命令的说明。
   txt文件添加pull命令的说明后提交。

> 修改myfile.txt后，分别调用add和commit命令
> txt内容如下：
> add 把变更录入到索引中
> pull 取得远端数据库的内容

这样，在issue2和issue3分别对myfile.txt进行了不同的修改。

> 相当于在两个分支上对同个文件进行了修改。

二、解决合并的冲突

把issue2分支和issue3分支的修改合并到master。

1. 切换master分支后，与issue2分支合并。
   git checkout master
   git merge issue2

   > 执行fast-forward（快进）合并

2. 接着合并issue3分支
   git merge issue3
   提示自动合并失败。由于在同一行进行了修改，所以产生了冲突。这时myfile.txt的内容如下：

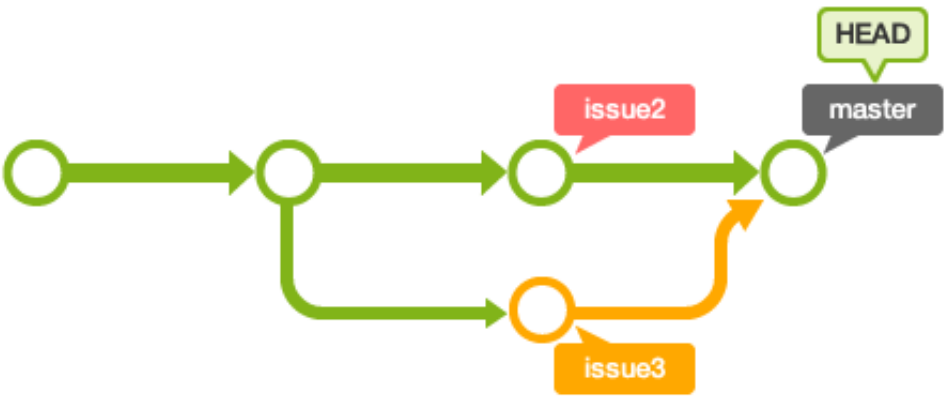
```
add 把变更录入到索引中
<<<<<<< HEAD
commit 记录索引的状态
=======
pull 取得远端数据库的内容
>>>>>>> issue3
```

3. 在发生冲突的地方，Git生成了内容的差异。请做以下修改：

```
add 把变更录入到索引中
commit 记录索引的状态
pull 取得远端数据库的内容
```

4. 修改冲突的部分后，重新提交。
   运行add和commit命令

   > 这种合并不是fast-forward合并，而是non fast-forward合并。



补充：用rebace进行合并

合并issue3分支的时候（即二、解决合并的冲突，第1步骤之后的内容），使用rebase可以使提交的历史记录显得更简洁。

1. 现在暂时取消刚才的合并。
   git reset --hard HEAD~
2. 切换到issue3分支后，对master执行rebase。
   git checkout issue3
   git rebase master
3. 和merge时的操作相同，修改在myfile.txt发生冲突的部分。

```
add 把变更录入到索引中
<<<<<<< HEAD
commit 记录索引的状态
=======
pull 取得远端数据库的内容
>>>>>>> issue3
```

   修改后：

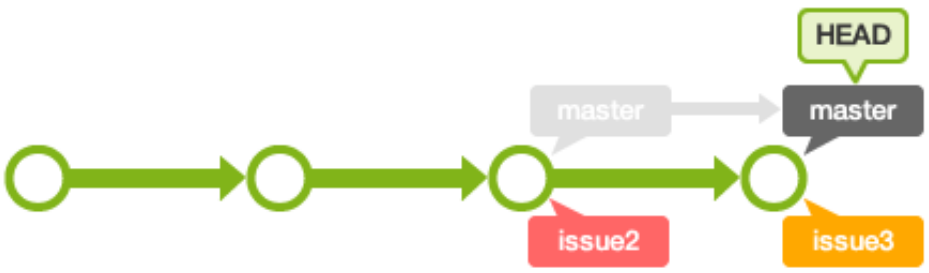
```
add 把变更录入到索引中
commit 记录索引的状态
pull 取得远端数据库的内容
```

4. rebase的时候，修改冲突后的提交不是使用commit命令，而是执行rebase命令指定 --continue选项。
   git add myfile.txt
   git rebase --continue

   > 若要取消rebase，指定 --abort选项。

5. 这样，在master分支的issue3分支就可以fast-forward合并了。切换到master分支后执行合并
   git checkout master
   git merge issue3

myfile.txt的最终内容和merge是一样的，但是历史记录如下：



远程仓库分支更新

- 本地仓库分支删除
  git branch -d 分支名
- 删除远程分支
  git push origin -d 分支名
- 更新远程仓库分支
  git push origin 分支名