# 基本命令

1. 进入要管理的目录
2. git init 初始化，即让git帮助我们管理当前文件夹
3. git status 检测 当前目录下文件的状态，新增和修改过后的文件都是红色。
4. 三种状态的变化
   1. 红色：新增的文件/修改了的老文件，（红色变绿色：通过git add 文件名/. ，dot是把所有未管理起来的文件都管理起来）
   2. 绿色：已经管理的文件，生成版本通过git commit -m ‘描述信息’
   3. 生成版本。
   4. 查看版本，git log.
5. 注意的问题
   1. 若执行commit时，报错，那么可能是没有个人配置。提交方必须有自己的标识
   2. git config --global user.email [you@example.com](mailto:you@example.com)
   3. git config --global user.name yourname
   4. 配置信息只需执行一次即可。

# 分区



# 反复回滚

1. 回滚到之前(某一历史的)版本

git log

git reset – –hard 版本号

2）取消回滚，回滚至之后（某一新的）版本

查看曾经的历史操作：git reflog

恢复至某一版本：git reset --hard 版本号

# 基本命令总结

|  |
| --- |
| git init  git config --global user.email “your email addr”  git config --global user.name “your name”  git add <file>/.  git commit -m ‘描述’  git log  git reflog  git reset --hard 版本号 |



git reset HEAD <file> 由暂存状态回到未暂存状态。

git checkout -- <file> 回到最原始状态

# 初识分支

若没有分支功能，只能单基线管理，那么以下场景不方便：

1. 多人同时开发部分功能，然后再合并；
2. 基于当前上线版本正在开发新功能，但上线版本发现bug需要紧急修复。



查看分支：git branch

在当前分支中，创建分支：git branch <分支名如dev>

切换到别的分支：git checkout <分支名如dev>

切换回主分支：git checkout master

将其他分支合并到主分支：先确认切换到master分支，然后git merge <分支名如bug>

删除无用的分支：git branch -d <分支名如bug>

合并分支时，可能产生冲突，此时需要找到冲突的文件位置，手动解决，然后再add及commit，或者你也可以选择通过git merge –abort进行取消合并。

总结：

|  |  |
| --- | --- |
| 查看分支 | git branch |
| 创建分支 | git branch <分支名称> |
| 切换分支 | git checkout <分支名称> |
| 分支合并（可能产生冲突） | git merge <要合并的分支>  注意：先确认切换分支再合并 |
| 删除分支 | git branch -d <分支名称> |

# 工作流

注意开发规范（工作流规范）：

分支至少得有两个，包括主分支、开发分支。

定义好管理规范，master上不要瞎搞，只保留上线版本/稳定版。其他正在开发的或公测版等都在开发分支上搞。

# GitHub基本操作

|  |  |
| --- | --- |
| 为远程仓库起别名 | git remote add <别名> 远程仓库地址 |
| 向远程仓库推送代码 | git push -u <别名> 分支名 |
| 克隆远程仓库的代码 | git clone <远程仓库地址>  注：内部已实现git remote add <别名> 远程仓库地址 |
| 切换分支 | git checkout <分支名> |