## 主要概念

* Clone
* Fetch
* Pull=pull+merge
* Commit
* Push
* Local resposity
* Remote resposity
* Merge
* Branch
* Checkout
* Reset：https://github.com/geeeeeeeeek/git-recipes/wiki/5.2-%E4%BB%A3%E7%A0%81%E5%9B%9E%E6%BB%9A%EF%BC%9AReset%E3%80%81Checkout%E3%80%81Revert-%E7%9A%84%E9%80%89%E6%8B%A9
* Revert
* Rebase：http://blog.csdn.net/wangjia55/article/details/8490679
* 本地分支用于自己修改，编写；远程分支用于不同开发人员之间的代码贡献；
* 不能直接修改远程分析，只有将远程分支克隆、fetch、pull到本地之后才能修改。

## 基本使用流程

1. 安装好客户端之后，通过git-clone从服务器仓库会自动下载对应的项目文件到本地磁盘，且本地项目根目录中会生成.git的文件夹，其中包含了所有历史版本的信息与索引。
2. 本地目录下的项目，便是本地仓库的概念。如果从远程服务克隆的是origin/master项目，则本地仓库的项目名为master（支持自定义）。
3. 在本地项目中，便可以编辑、调试、运行相关项目。
4. 项目修改之后，如果想要提交之远程服务器上去，需要先commit到本地仓库，然后才push上去。push更新到服务端，需要我们对相应的远程服务器有写权限。
5. 如果学习阶段不想设计主分支，那么可以提前在服务端“基于某个分支创建新的分支”，后续可以直接在这个分支上进行学习。
6. 如果想将这个本地master分支上的内容推送更新到远程 develop分支：在本地克隆、创建基于远程develop的本地develop分支，切换（checkout）到本地develop分支，然后与本地master分支合并（merge），进而在push到远程的develop分支。
7. 默认情况下，本地分支与远程分支互相不干扰，某一个开发人员的origin/master分支也不受其他人更新远程master分支的影响，除非联网。
8. 以上内容的详细查看：<https://git-scm.com/book/zh/v2/%E8%B5%B7%E6%AD%A5-%E5%85%B3%E4%BA%8E%E7%89%88%E6%9C%AC%E6%8E%A7%E5%88%B6>
9. Public文件目录不用提交。

## Gitlab功能

<https://about.gitlab.com/2016/12/01/how-to-keep-your-fork-up-to-date-with-its-origin/>

镜像功能。

在github上通过pull request来提交、审核合并请求，并且可以实现反向更新。

但是gitlab上merge request功能只能提交与审核合并请求，从fork源处的更新可以使用gitlab的resposity mirror功能。（企业级收费用户才可以使用）

因此，只能退而求其次，通过给本地项目添加多个远程服务器：正式编写代码之前，先从upstream的master分支拉取项目，code之后，推送到自己的项目的master分支。在自己项目remote master分支上再merge request。详述如下：

<https://jinlong.github.io/2015/10/12/syncing-a-fork/>