GIT 学习笔记

1. 提交、修改、查看历史、协作

* **工作区与版本库**：工作区是一个包含.git 子目录（内含版本库）中的目录。我们可以用init 命令在当前目录中创建版本库。
* **版本提交**：一次版本提交通常定义了版本库中所有文件的一个版本，它详细说明了该版本是由何人在何时何地创建的。需要用add 命令来确定哪些文件将被纳入下一次提交，然后再用commit 命令创建新的版本提交。
* **查看信息**：通过 status 命令，可以查看哪些文件已被本地修改，以及哪些修改将被纳入下次提交。另外，log 命令可用来显示提交历史。diff 命令可用来显示两个版本文件之间的差异。
* **克隆**：用 clone 命令创建某一个版本库的副本，称之为该版本库的克隆体。在一般情况下，每个开发者都会拥有整个项目版本库的完整克隆体，工作区中将会包含完整的项目历史。可以各自独立开展工作，无需连接服务器。
* **推送与拉回**：push 与pull 命令可用于在本地和远程版本库之间共享版本提交。

1. 提交

* 版本库：项目的版本库通常驻留在其.git 目录中。其中包含了以提交形式存储的项目。由于Git 是个分布式系统，所以同一个项目通常可能会有多个拥有不同历史的版本库。可以在必要时再将这些历史重新合并起来。
* 提交（通常也被称为版本、修订或修改集）：每次提交中都包含了与作者和提交日期相关的元数据Git 中存储了前/后提交之间的关系，这种关系将构成一个项目版本图，可以用log 命令将版本库中的这些提交显示出来。
* 提交散列值：提交散列值主要用于唯一标识提交。

1. 多次提交

* 暂存区：暂存区中所存储的是为下一次提交准备的内容，它以快照的形式保存了相关的文件内容。 工作区-》暂存区-》版本库
* 选择性提交：可以用add 命令来指定哪些文件将会被存储在快照中，其余所有文件将保持不变。
* .gitignore 文件：我们可以在这个文件中列出不需要被Git 管理的文件目录。
* 储藏：我们可以通过 stash 命令将在工作区和暂存区中当前所做的修改储藏起来。之后，再用git stash pop 命令将其恢复。

1. 分支

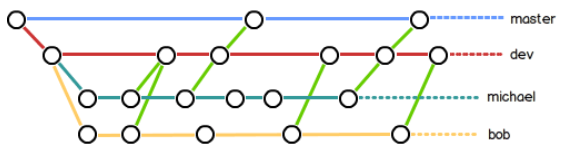
* 分支：提交图中出现的那个分叉就叫做分支。该分支会有一个指针，指向该分支下的最后一次提交。
* 创建分支：用branch 命令来新建分支。
* Checkout：用checkout 命令切换到另一个分支上。
* Reflog：git 会记录每次提交中对分支指针所做的所有修改。如果想恢复某个不小心删除的分支，查找散列值。

1. 合并

* 合并提交：执行merge 命令的结果就是产生一次合并提交。
* 3 路合并：Git 会在合并时利用提交图找到合并双方最后的共同祖先。然后，Git 将引自该祖先的一个分支上的修改，连同另一分支上所做的修改放在一起。只要这些修改发生在这份源代码的不同之处，Git 就能自动创建相应的合并提交。
* 冲突：编辑冲突：手动解决，再提交合并。内容冲突：？？
* 快进合并：如果一个分支没有工作，则合并只是移动指针，而不产生合并提交。可以通过no-ff，强制产生合并提交。

2020.1.1

分支策略：



在实际开发中，应该按照几个基本原则进行分支管理：首先，master分支应该是非常稳定的，也就是仅用来发布新版本，平时不能在上面干活；干活都在dev分支上，也就是说，dev分支是不稳定的，到某个时候，比如1.0版本发布时，再dev分支合并到master上，在master分支发布1.0版本；每个人都在dev分支上干活，每个人都有自己的分支，时不时地往dev分支上合并就可以了。

变基：

只对尚未推送或分享给别人的本地修改执行变基操作清理历史，从不对已推送至别处的提交执行变基操作，这样，你才能享受到两种方式带来的便利。

1. 打标签：

* 创建普通标签：git tag 1.2.3.4 master -m "Freshly built." 为master 分支上的当前版本创建一个名为1.2.3.4 的标签，并将其注释为“Freshly built.”。 git tag v0.9 f52c633 通过散列值打标签。
* 查看标签：git tag;在日志输出中显示标签 git log --oneline –decorate；
* 查找：判断某一特定的功能或 bug 修复是否被包含在客户所安装的那个版本中：--contains 参数git tag --contains f63cd71。
* 推送：git push origin 1.2.3.4或使用--tags 参数。