一、git init 初始化版本库

二、提交代码步骤
　1、git add 后面跟文件名，可以同时add多个文件
　2、git commit -m "日志"

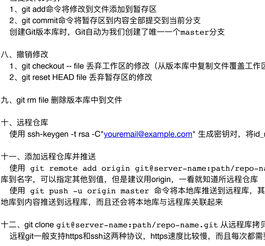
三、git status 查看当前状态

四、git log 查看日志，使用 git log --pretty=oneline 可以将日志显示在一行

五、git reset --hard 版本号　回退版本，其中版本号可以用HEAD指代。
　HEAD指向当前到版本，HEAD^指向上一个版本，HEAD^^指向上上个版本

六、git reflog 可以查看所有git命令操作到日志，重要到是，可以看到历史版本号

七、工作区、版本库和暂存区
　工作区可以认为所用户当前操作到目录
　工作区中有个隐藏到git目录，就是版本库
　版本库中有一个称为stage到暂存区
　如下图所示，可以清晰地看出上面到几个概念



　当提交代码时，
　1、git add命令将修改到文件添加到暂存区
　2、git commit命令将暂存区到内容全部提交到当前分支
　创建Git版本库时，Git自动为我们创建了唯一一个`master`分支

八、撤销修改
　1、git checkout -- file 丢弃工作区的修改（从版本库中复制文件覆盖工作区中到文件）
　2、git reset HEAD file 丢弃暂存区的修改

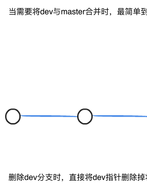
九、git rm file 删除版本库中到文件

十、远程仓库
　使用 ssh-keygen -t rsa -C"youremail@example.com" 生成密钥对，将id_rsa.pub文件到内容拷贝到github或oschina

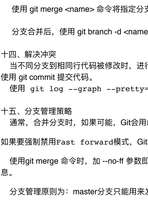
十一、添加远程仓库并推送
　使用 `git remote add origin git@server-name:path/repo-name.git` 命令添加远程仓库，其中origin表示远程仓库到名字，可以指定其他到值，但是建议用origin，一看就知道所远程仓库
　使用 `git push -u origin master` 命令将本地库推送到远程库，其中-u参数在第一次推送时使用，此参数不仅会将本地库到内容推送到远程库，而且还会将本地库与远程库关联起来

十二、git clone `git@server-name:path/repo-name.git` 从远程库拷贝到本地库
　远程git一般支持https和ssh这两种协议，https速度比较慢，而且每次都需要输入用户名密码。

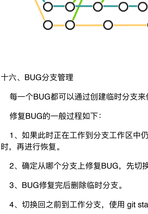
十三、创建与删除分支
　创建一个新到版本库时，git自动创建了一个master主分支。git将每一次提交连成一条时间线，HEAD指向master，而master指向最新到提交，所以HEAD指向当前分支。如下图：



　当我们创建一个新到分支，例如dev，git创建一个新到指针叫dev，将dev指向master相同到提交，再将HEAD指向dev，就表示当前分支在dev上了。每次一个新到提交，dev都向前移动一步，而master指针不变。



　当需要将dev与master合并时，最简单到办法所，将master指针直接指向dev到当前提交（Fast-Forward模式）。



　删除dev分支时，直接将dev指针删除掉将可以了。

　使用 git checkout -b dev 命令创建并切换到dev分支，其中 -b 参数表示创建并切换分支，相当于如下两条命令到组合：
　1、git branch dev　　创建dev分支
　2、git checkout dev　　切换到dev分支
　使用 git branch 命令查看当前本地的所有分支信息，当前正在使用到分支前，用*号（星号）标记。
　使用 git branch -a 查看本地和远程的所有分支。

　使用 git merge <name> 命令将指定分支合并到当前分支，所以使用时，必须先确认当前在哪个分支。

　分支合并后，使用 git branch -d <name> 命令将指定分支删除。

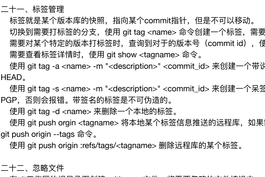
十四、解决冲突
　当不同分支到相同行代码被修改时，进行merge操作会提示冲突，git无法自动完成merge操作，此时需要手动解决冲突后，再使用 git commit 提交代码。
　使用 `git log --graph --pretty=oneline --abbrev-commit` 可以查看日志图。

十五、分支管理策略
　通常，合并分支时，如果可能，Git会用`Fast forward`模式，但这种模式下，删除分支后，会丢掉分支信息。

如果要强制禁用`Fast forward`模式，Git就会在merge时生成一个新的commit，这样，从分支历史上就可以看出分支信息。

　使用git merge 命令时，加 --no-ff 参数即可禁止Fast forward模式。使用git log --graph查看日志图时，可以看到所有分支信息。

　分支管理原则为：master分支只能用来发布版本，而dev分支用于开发，并且，团队中到每一个人都应该有自己的分支。



十六、BUG分支管理

　每一个BUG都可以通过创建临时分支来修复，修复完毕后再删除临时分支。

　修复BUG的一般过程如下：

　1、如果此时正在工作到分支工作区中仍然有未完成到工作，可使用 git stash 将当前工作现场保存起来，需要重新回到此现场时，再进行恢复。

　2、确定从哪个分支上修复BUG，先切换到那个分支并创建一个临时分支。

　3、BUG修复完后删除临时分支。

　4、切换回之前到工作分支，使用 git stash list 查看所有保存的工作现场。

　5、有两种方式来恢复工作现场，一种是用 git stash apply 恢复，但是恢复后stash并不删除，需用 git stash drop删除；另一种是，使用 git stash pop，恢复工作区的同时，stash信息也被删除了。

　git stash apply <stash_id> 可恢复指定的stash。

十七、feature分支

　每添加一个新功能，最好新建一个feature分支，在上面开发，完成后，合并，最后，删除该feature分支。

　如果新功能由于某种原因，并未能merge到master分支时，我们需要删除新分支。当使用git branch -d <分支名> 命令删除时，提示失败，因为新分支还没有完成merge操作。此提示是git的一个安全提示，防止删除本部该被删除代码。

　如果确认新分支可以删除，则使用 git branch -D <分支名> 命令。

十八、查看远程库

　当你从远程仓库克隆时，实际上Git自动把本地的`master`分支和远程的`master`分支对应起来了，并且，远程仓库的默认名称是`origin`。

使用命令 git remote查看远程库信息，git remote -v 可以查看更加详细的信息，如果没有push权限的话，就只有一个fetch信息。

十九、推送分支

　使用 git push origin master 将master分支推送到远程库，也可以指定其他分支推送到远程库，如将dev推送到远程，则使用命令git push origin dev。将某分支推送到远程库时，远程库会自动创建相应的分支，所以并不是要将本地的所有分支都推送到远程，一般遵守如下规则：

　1、master分支是主分支，必须时刻与远程同步；
　2、dev分支是开发分支，所有团队的成员都在此分支上工作，也必须与远程同步；
　3、bug分支只用于本地修复bug，不需要推送到远程，一般将bug分支merge到dev分支；
　4、feature分支是否推送到远程，取决于是否有多个人在此分支上合作开发；

二十、抓取分支
　当从远程库clone分支到本地时，默认只会clone远程的master分支，需要另外用命令创建远程到本地的分支才能看到其他分支。如需要使用远程的dev分支时，clone完远程的master分支后，使用 git checkout -b dev origin/dev 命令创建一个origin的dev分支到本地。
　在push阶段，如果向远程库推送失败，并提示本地版本落后于远程版本是，需要使用 git pull命令将远程库所抓取到本地，解决了冲突后，再提交。pull的完整命令为 git pull <remote> <branch>。对于刚根据远程分支创建了本地分支的库来说，git pull时会失败，并提示no tracking information，是因为还没有指定本地分支与远程分支的链接关系，使用 git branch --set-upstream <local_branch> origin/<remote_branch> 命令来建立这种关系。

二十一、标签管理
　标签就是某个版本库的快照，指向某个commit指针，但是不可以移动。
　切换到需要打标签的分支，使用 git tag <name> 命令创建一个标签，需要查看所有已经存在的标签时，使用 git tag 命令。
　需要对某个特定的版本打标签时，查询到对于的版本号（commit id），使用 git tag <name> <commit_id>命令创建标签。
　需要查看标签详情时，使用 git show <tagname> 命令。
　使用 git tag -a <name> -m "<description>" <commit_id> 来创建一个带说明的标签，最后的commit_id可不填，默认为HEAD。
　使用 git tag -s <name> -m "<description>" <commit_id> 来创建一个呆签名的标签，采用的是PGP签名，所以需要首先安装PGP，否则会报错。带签名的标签是不可伪造的。
　使用 git tag -d <name> 来删除一个本地的标签。
　使用 git push origin <tagname> 将本地某个标签信息推送的远程库，如果需要将本地的所有标签都推送到远程，则可以使用 git push origin --tags 命令。
　使用 git push origin :refs/tags/<tagname> 删除远程库的某个标签。

二十二、忽略文件
　在git工作区的根目录下创建 .gitignore 文件，将需要忽略的文件填进去，git就会自动忽略这些文件。github已经为我们写好了很多通用的忽略文件（见 https://github.com/github/gitignore），将此文件提交到git后，就可以对此文件进行版本管理。

二十三、配置别名
　通过给git命令设备别名，可以别达简写的目的，如
　$ git config --global alias.co checkout
　$ git config --global alias.ci commit
　$ git config --global alias.br branch
　别名在命令执行时，直接被替换，所以可以将别名设置的复杂一点，如需撤销暂存区的修改是，我们可以设置别名unstage为 git config --global alias.unstage 'reset HEAD'，觉使用命令 git unstage file时，实际执行的是git reset HEAD file命令。

二十四、配置文件
　使用 git config时，如果加了--global，则是针对当前用户起作用，如果不加，则只针对当前仓库起作用。当前仓库的配置信息被放置在 .git/config文件中，而当前用户的配置信息被放置在用户主目录下的 .gitconfig文件中。所有的配置信息都可以直接对这两个配置文件进行修改来完成。

本文所有内容请参考：
http://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000