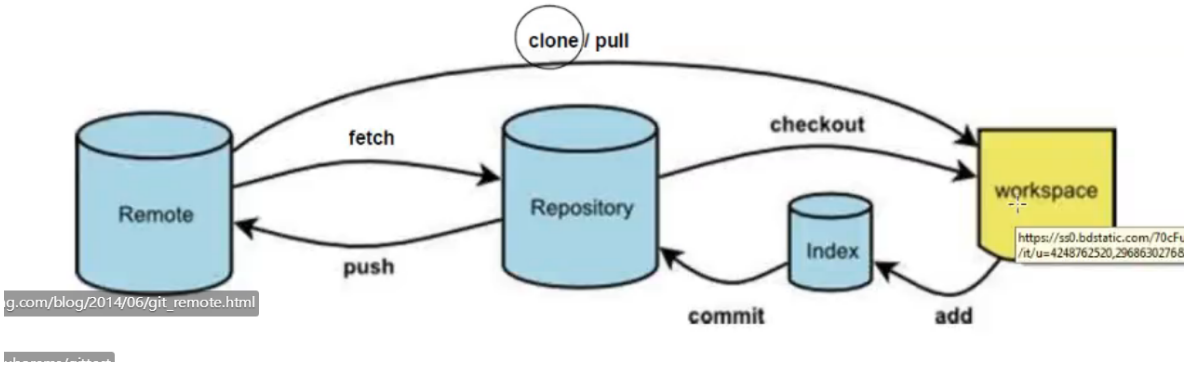
Git常用命令



- **add, commit, push, pull, clone, 是最常用的，不赘述**
- **git commit --amend**

```
$ git commit -m 'initial commit'
$ git add forgotten_file
$ git commit --amend
```

上面的三条命令最终只是产生一个提交，第二个提交命令修正了第一个的提交内容。

- **git reset HEAD filename**

> 例子，有两个修改过的文件，我们想要分开提交，但不小心用 `git add .` 全加到了暂存区域. 用此命令取消add的某个文件

- **git checkout -- filename**

> 取消对文件的修改，使之与上一次提交的版本保持一致

- **git clone url**

> 克隆远程仓库到本地

- **git remote add [shortname] [url]**

> 添加远程仓库，例如 git remote add pb git://github.com/paulboone/ticgit.git

- **git fetch pb**

> 从远程仓库抓取数据
>
> 此命令会到远程仓库中拉取所有你本地仓库中还没有的数据。运行完成后，你就可以在本地访问该远程仓库中的所有分支，将其中某个分支合并到本地，或者只是取出某个分支，一探究竟。
>
> 如果是克隆了一个仓库，此命令会自动将远程仓库归于 origin 名下。所以，`git fetch origin` 会抓取从你上次克隆以来别人上传到此远程仓库中的所有更新（或是上次 fetch 以来别人提交的更新）。有一点很重要，需要记住，fetch 命令只是将远端的数据拉到本地仓库，并不自动合并到当前工作分支，只有当你确实准备好了，才能手工合并。
>
> 如果设置了某个分支用于跟踪某个远端仓库的分支（参见下节及第三章的内容），可以使用 `git pull` 命令自动抓取数据下来，然后将远端分支自动合并到本地仓库中当前分支。在日常工作中我们经常这么用，既快且好。实际上，默认情况下 `git clone` 命令本质上就是自动创建了本地的 master 分支用于跟踪远程仓库中的 master 分支（假设远程仓库确实有

> master 分支）。所以一般我们运行 `git pull`，目的都是要从原始克隆的远端仓库中抓取数据后，合并到工作目录中的当前分支。

- **git push [remote-name] [branch-name]**

  > 项目进行到一个阶段，要同别人分享目前的成果，可以将本地仓库中的数据推送到远程仓库。实现这个任务的命令很简单：`git push [remote-name] [branch-name]`。如果要把本地的 master 分支推送到 origin 服务器上（再次说明下，克隆操作会自动使用默认的 master 和 origin 名字），可以运行下面的命令：
  >
  > `$ git push origin master`
  > 只有在所克隆的服务器上有写权限，或者同一时刻没有其他人在推数据，这条命令才会如期完成任务。如果在你推数据前，已经有其他人推送了若干更新，那你的推送操作就会被驳回。你必须先把他们的更新抓取到本地，合并到自己的项目中，然后才可以再次推送

- **git tag**

  > 列出所有标签

- **git branch**

  > 创建分支

- **git init**

  > 初始化一个空的git目录

- **git config**

  > 配置。例如，
  >
  > git config user.name zhangsan
  >
  > git config user.email zhangsan@qq.com
  >
  > 全局的：
  >
  > git config --global user.name zhangsan
  >
  > git config --global user.email zhangsan@qq.com

- **git log**

  > 查看历史记录
  >
  > 空格向下翻页
  >
  > b向上翻页
  >
  > q退出

- **git reflog**

  > 可以参考HEAD@{XXX}

- **git reset**

  > 版本回退
  >
  > git reset --hard [索引]
  >
  > 参数：
  >
  > - hard 暂存区和工作区都被reset，移动本地库的HEAD指针
  > - soft 只移动本地库的指针
  > - mixed 暂存区被reset, 移动本地库的指针

- **git fsck --lost-found**

  > 可以找回已经add但是没commit 并且已经被reset过的文件

- **git diff**

  > git diff [文件名] : 将工作区中的文件和暂存区的进行比较
  >
  > git diff [本地库的某个版本] [文件名] : 将工作区中的文件和本地库历史记录比较

- **git branch -v**

  > 查看分支信息

- **git branch [branchName]**

  > 创建分支

- **git merge**

  > 合并