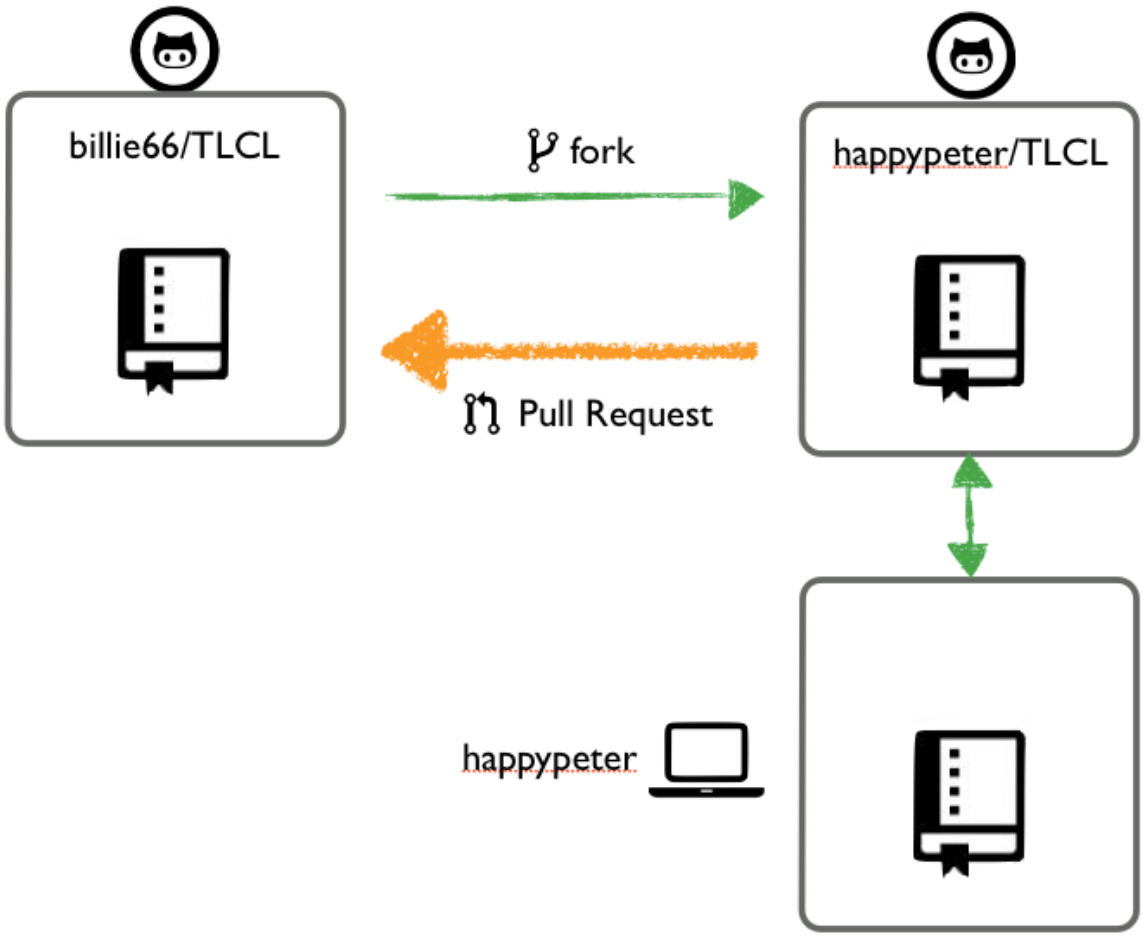
贡献开源项目的流程

Github 是目前世界上最大的开源项目的托管交流平台。贡献开源项目的流程也是 Github 全力支持的，也一样是遵循 Github Flow，虽然跟前面团队合作流程会有一点差别。在团队内部，大家都是有写权限的。但是网上的开源项目参与者众多。如果你一上

去就跟项目的拥有者说，Hey，你给我加个写权限吧，别逗了，人家也不认识你，怎么可能呢？

Fork

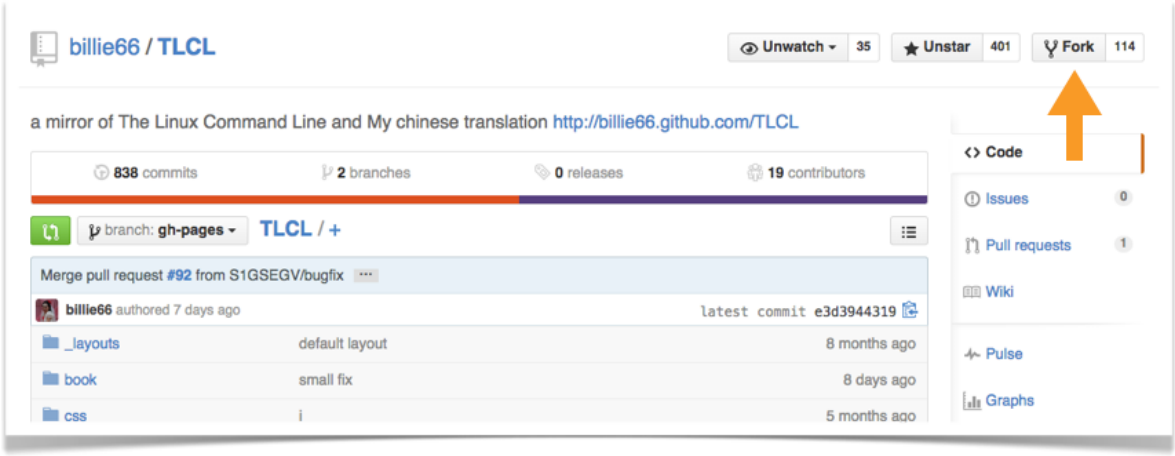
所以第一步是 Fork 这个项目。所谓我 fork 别人的一个项目，就是指做一个把这个项目做一个拷贝，放到我自己的账户下。



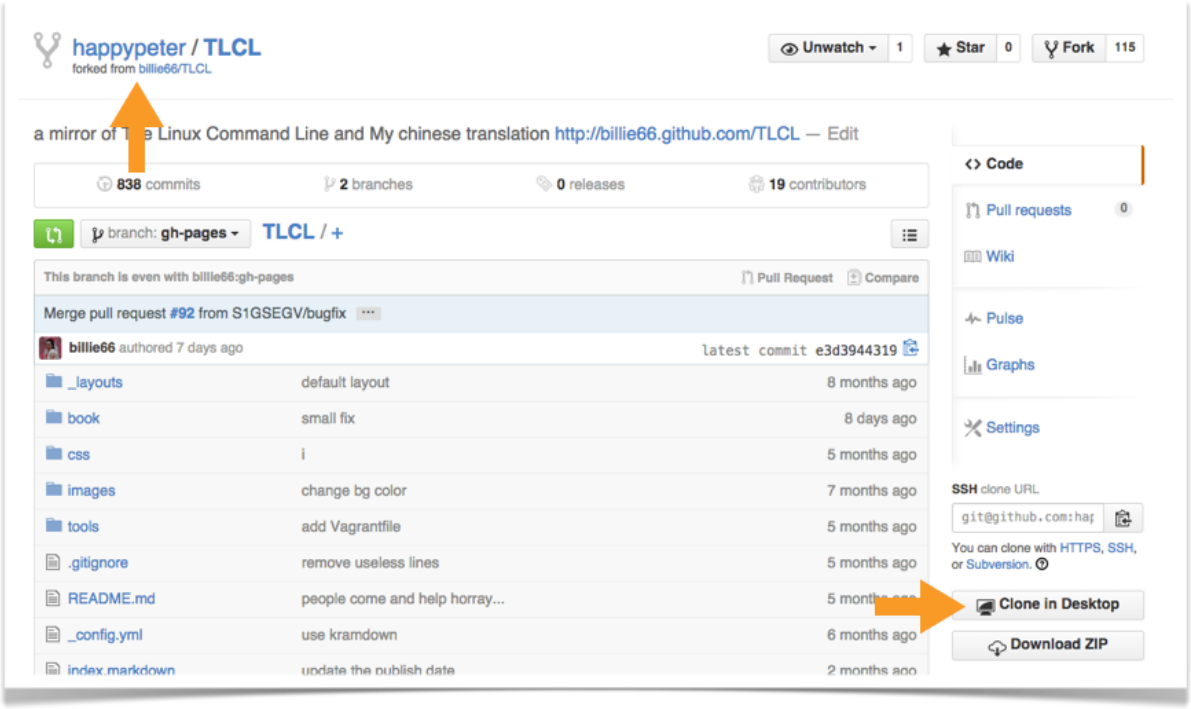
基于 fork 的整个的流程就是上图展示的思路。第一步，先 fork，这样我就有了 happypeter/TLCL 这个项目了，那既然是我自己的项目，我就可以把它 clone 到我本地，做修改，然后推送到 github 上的我自己的那个 fork 之中。这样我把我自己的 fork 跟上游的仓库，也就是 billie66 名下的仓库来进行对比，就可以发出 Pull Request 了。

下面来实际操作。

来到 TLCL 的项目主页，也就是 https://github.com/billie66/TLCL 。点右上角的 fork 按钮。



这样，我自己的名下就多了一个 TLCL 项目，这个项目就叫做原项目的一个 fork 。

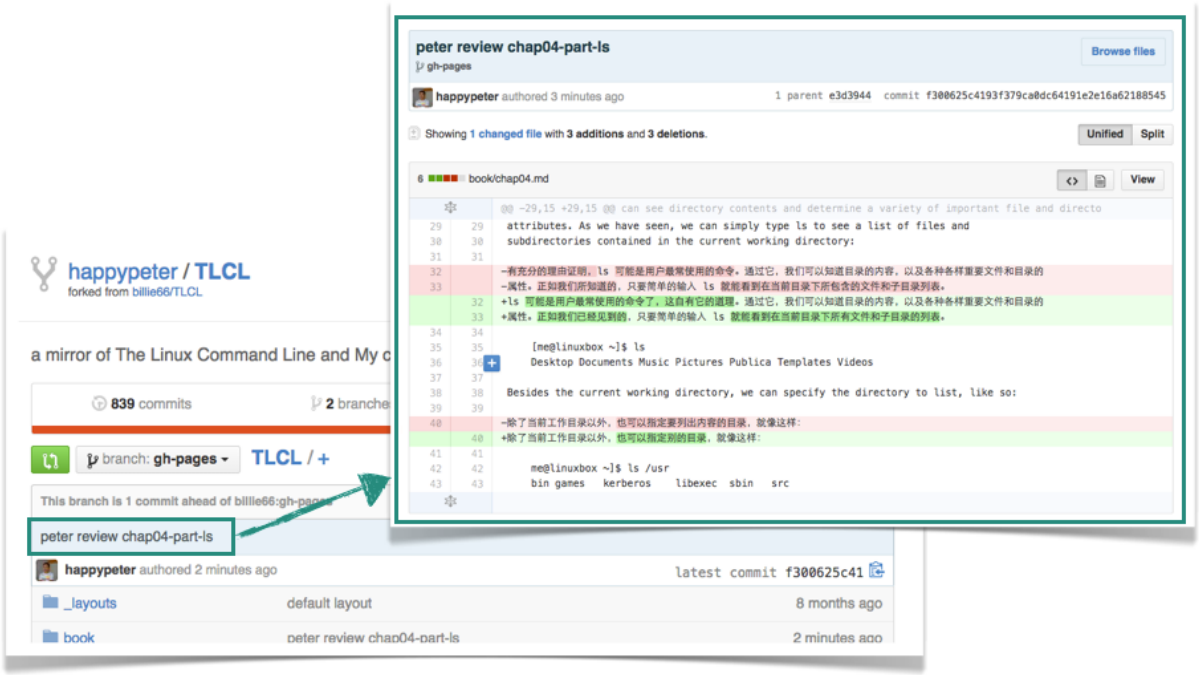


到这里 fork 这一步就结束了。那对于 happypeter/TLCL 这个仓库，我当然有修改的权限了，可以在网页上修改直接发 PR。不过一般我会 clone 到本地，在本地作修改。

做版本和同步

区别于团队合作的 Github Flow，这里主要是没有开新分支，而是创建了一个新的 fork。上游仓库 billie66/TLCL 中代码是在 gh-pages 分支上，那现在我在本地的 gh-pages 修改，新版本同步到在 happypeter/TLCL 的 gh-pages 上就可以了。

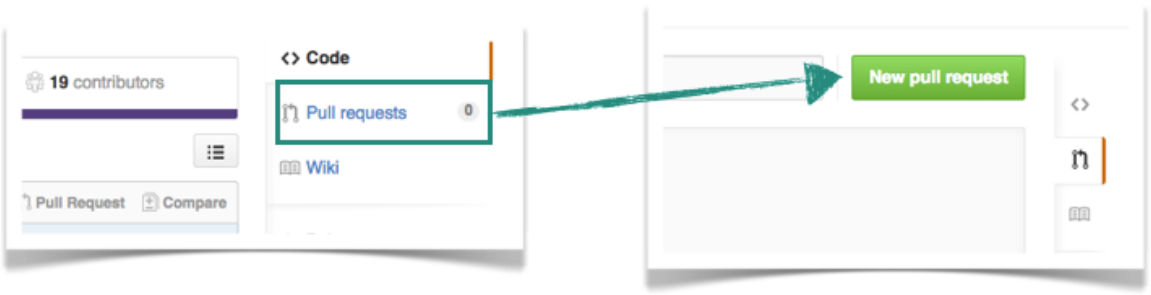
本地修改，做 commit，然后同步到远端的 happypeter/TLCL 中，这些步骤到现在应该是轻车熟路，小菜一碟了。于是到 github.com 可以看到下图的内容



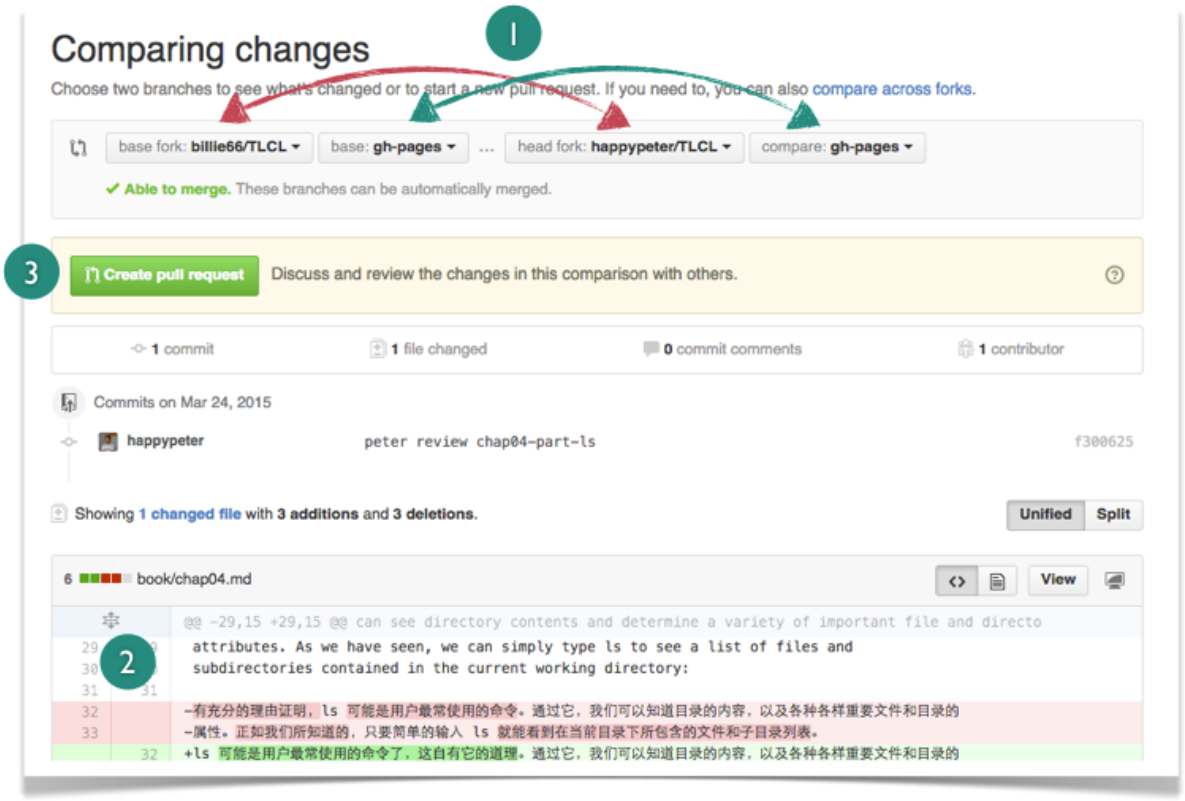
如上图所示到 happypeter/TLCL 的仓库主页，可以看到最新的我做的版本的留言，点开就可以看到我这次修改的内容，现在可以来发 PR 了。

发 PR 和代码审核

发 PR 的过程跟前面介绍过的没有本质区别。



如上图，点一下项目页面右侧的 `Pull Request` 链接，到达的页面中会有一个大大的绿按钮 `New Pull Request`，点一下就看到下面的页面



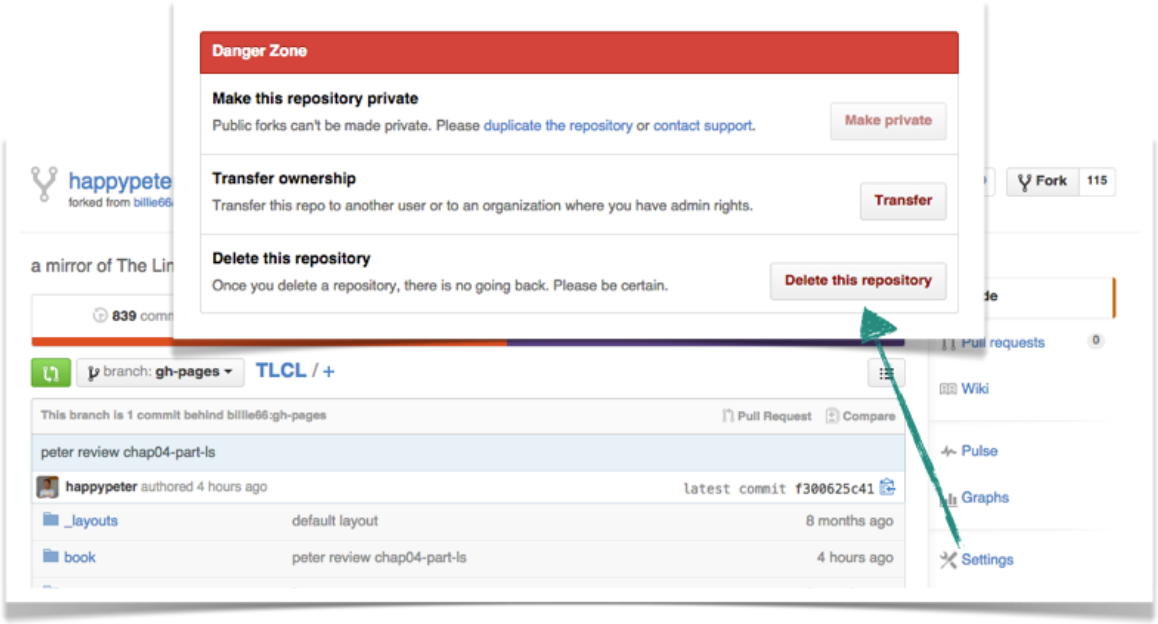
注意上面1处，要选对是哪两个分支进行对比，左侧是上游分支，也就是“目的地”分支，右侧是我自己的分支，有修改的内容。有时候 github 能猜对拿哪两个分支做对比，有时候就要自己手动选择一下。分支要是选得不对那么2处对比出来的代码肯定也会有问题的，所以还是比较容易看出来的。下面就可以点3处的大按钮来发 PR 了。之后，在 billie66/TLCL 下面，注意不是 happypeter/TLCL 下面，就会看到这个 PR 。

当项目拥有者 billie66，或者是项目的协作者（ collaborator ） 看到 Pull Request，就可以来跟我讨论。如果决定要把我的代码 merge 进自己的仓库，点一下 `Merge Pull Request` 按钮就可以了。

这样，我这次贡献代码的工作就结束了。TLCL 的版本历史中会永远留下 happypeter 的名字。同时我发的这个 PR 关掉之后，也可以在 billie66/TLCL 的 Pull Requests 一项下面被永久保存，所以我可以放心的删除我自己的那个 fork 了。

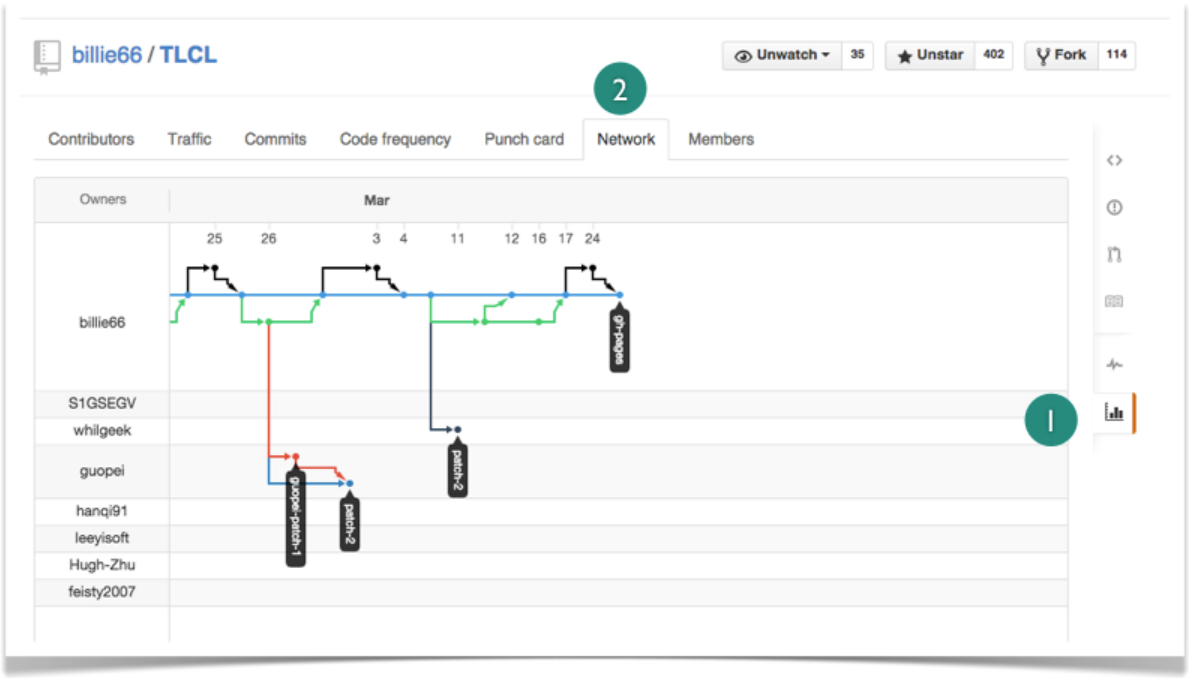
后续工作

删除一个 fork ， 跟删除一个我自己的项目仓库是一样的。



如上图，点开 settings 然后进入 Danger Zone 找 `Delete This Repository` 就可以来删除 happypeter/TLCL。当然如果会频繁的贡献同一个项目，自己名下长期保留一个 fork 也是可以的，但是这时候就要保证这个 fork 和上游仓库的同步，也是挺麻烦的。所以对于初学者，删掉，过些日子如果又想贡献，再 fork 一次不迟。

对于项目维护者，如果想随时了解自己的项目都有哪些人正在自己的 fork 进行修改，可以看一下项目的 `Network`，如下图：



基于 fork 的快速 PR