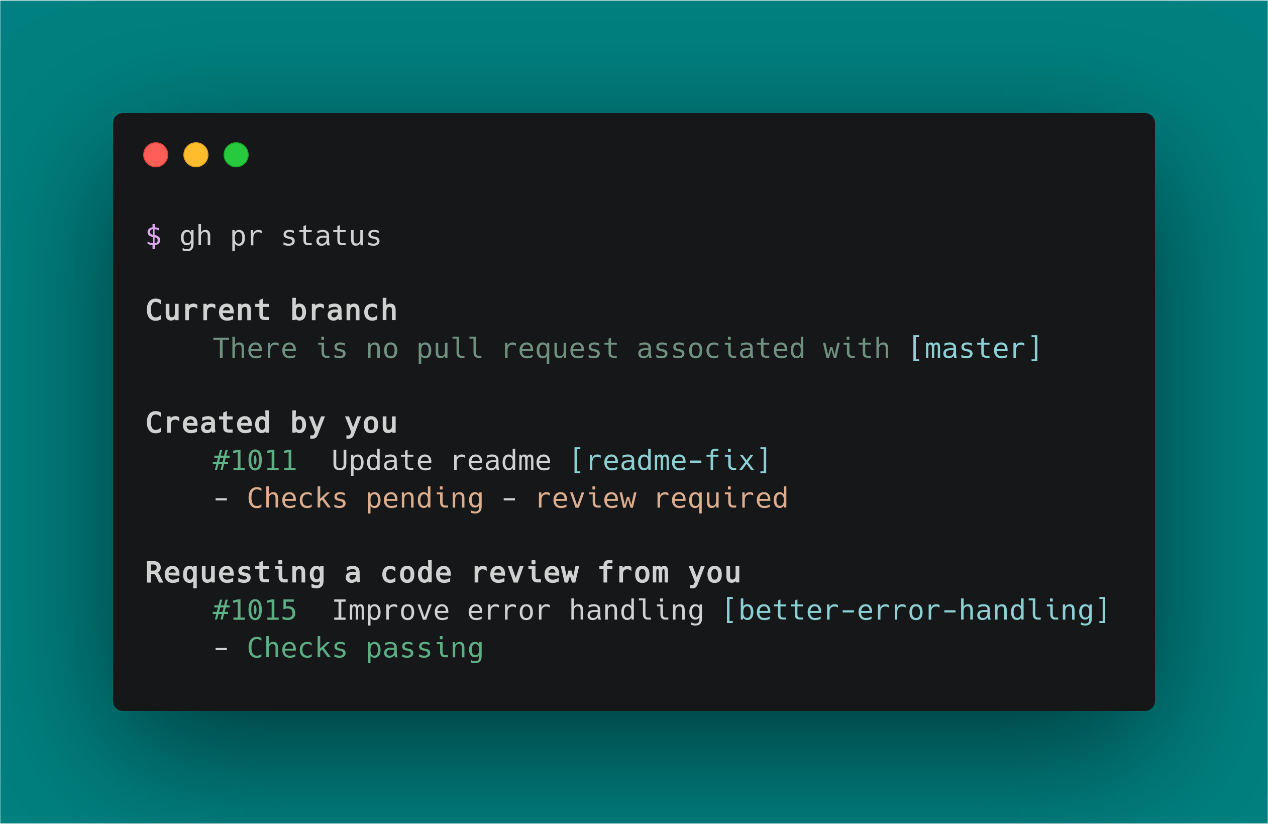
**[GitHub-Cli](https://cli.github.com/)的[安装](https://github.com/cli/cli" \l "installation/)与**[**使用**](https://cli.github.com/manual)

初一（2）班 吴奕鸣 24号

综述

**** **Take GitHub to the command line!**

想必GitHub大家都不陌生。本来也只是偶尔上去安装个软件啥的，不过自从几个月前我接触到用GitHub作为服务器和二级域名来做网站后，我访问GitHub也就更加频繁了。每天都要重复登陆、收验证码实在是有些烦了，再加上这么慢的网速，毕竟我也没那么多时间，于是我开始使用Git+Hexo本地生成+上传的模式。可用了一段时间后，我却发现有比Git更快、更强、更高大上的终端命令行——GitHub-Cli（以下也简称gh）。

Github是什么？



GitHub是一个面向开源及私有软件项目的托管平台，因为只支持Git作为唯一的版本库格式进行托管，故名GitHub。

GitHub（Git）的一大亮点是同一项目不同的分支与分支间Pull Request，即合并不同分支，常用于不同开发者之间的小型商讨、改动与更新。

该项目在2018年6月4日正式被微软收购。

1. Gh简介

GitHub-Cli,也就是GitHub最新推出的命令行终端，具有快速、高效等特性，给万千码农带来了极大的福利——更快地访问、克隆仓库，更快的提交更新，更高效的处理issue和PR。曾经必须要到GitHub网页端才能修改的信息，现在在CMD上就可以操作。

2天前gh已经推出了最新的2.4.0版本。这款终端使用GoLang编写，配合一定的批处理文件就可以完成一切任务。最新数据显示该仓库已经有了2万6千个Star。

Git是什么？

Git是一个开源的分布式版本控制系统，可以有效、高速地处理从很小到非常大的项目版本管理。也是Linus Torvalds为了帮助管理Linux内核开发而开发的一个开放源码的版本控制软件。显著的优点有适合分布式开发，速度快、灵活，任意两个开发者之间可以很容易的解决冲突。还可以离线工作。

[项目官网](https://cli.github.com/) [项目源码地址](https://github.com/cli/cli/)

1. Gh的安装

安装其实很简单，毕竟只是一个基于CMD的命令行嘛。下面以Windows为例：

1. 安装包安装

访问[这个链接](https://github.com/cli/cli/releases/tag/v2.4.0)，找到Assets。然后你会找到很多很多的安装包，在比较下面的地方有一个gh\_2.4.0\_windows\_amd64.msi的安装文件（貌似是64位的）。尝试下载它……很慢的话可以访问下面这个加速网址：[下载地址](https://github.com.cnpmjs.org/cli/cli/releases/tag/v2.4.0)（实在不行的话那我也没辙了）

下载下来以后，直接运行安装即可。安装完成后，打开CMD运行：

**$ gh –version**

如果CMD返回给你有用的信息（而不是“gh“不是内部或外部命令，也不是可运行的程序或批处理文件。），那么说明你已经安装好了！

1. Chocolatey安装（没装过的同志请往下看

使用过Chocolatey的同志直接运行下面的命令！

$ choco find gh

$ choco install gh –y

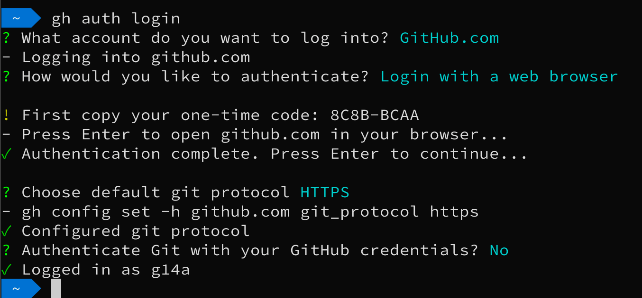
嗯……搞定了！

1. Gh登录时刻！

然后，用这个东东咱要登陆一下咱的GitHub账号，要不然怎么证明你是你呢？登陆也很简单，只要运行：

$ gh auth login

然后就会有下面的提示语：



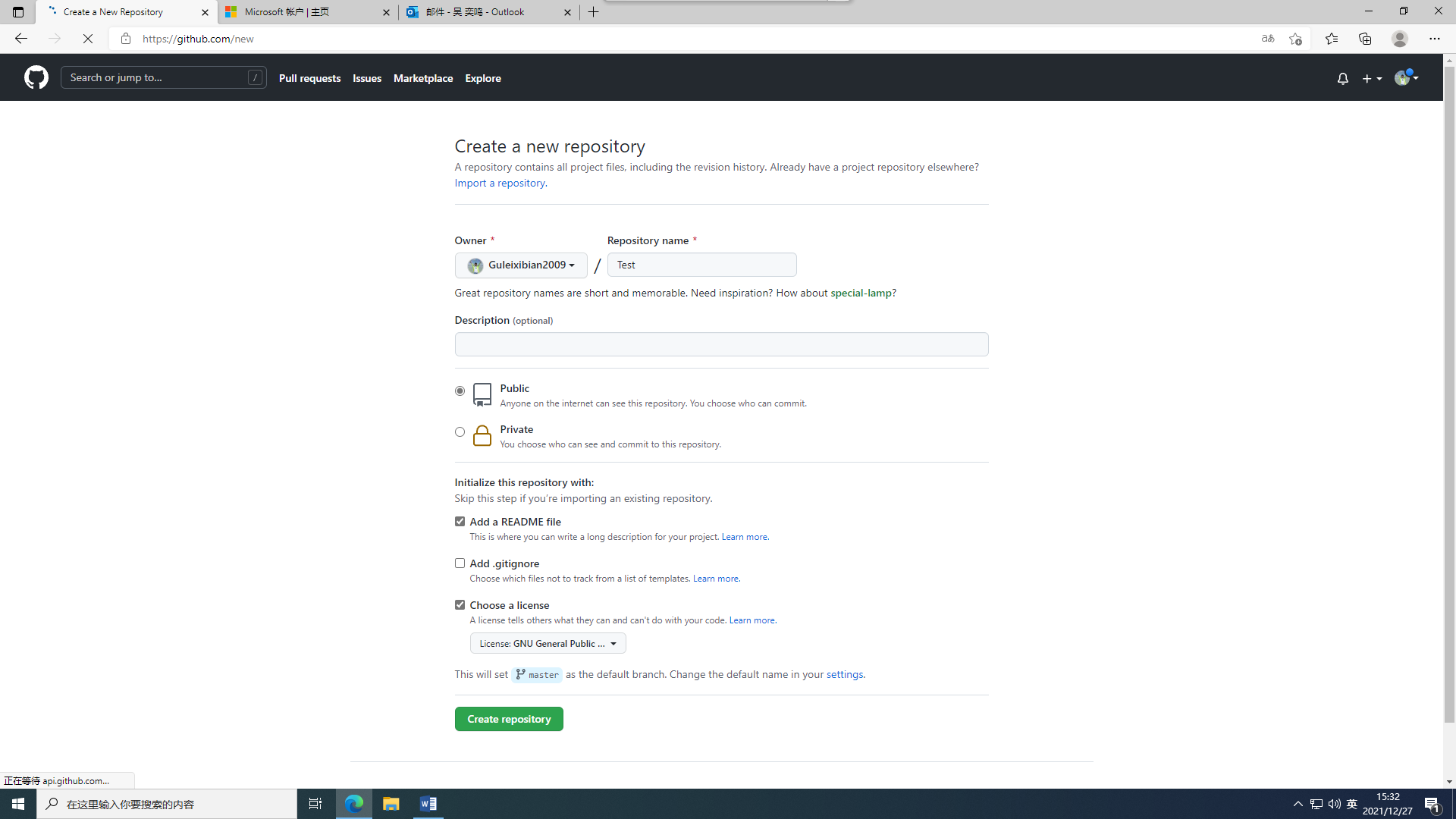
（图片来源于网络……所以不是我的用户名

两种登陆方法：你可以使用浏览器登陆账号后登录GitHub-Cli，也可以使用粘贴个人验证码登录。由于GitHub-Cli的内核也是用到Git的，所以它也会要求你配置Git，选择SSH活着HTTPS貌似都差不多。

如果最后它提示你类似Logged in as …说明你成功了！！

1. 关于你的仓库

光有账号、登录你没有项目又怎么行？用你本地的Git或者在线创建一个仓库，然后你就可以操作了。



创建一个仓库后，把它拉取到本地。这个时候GitHub-Cli就能派上用场了：

$ gh repo clone {username}/{repo\_name}

如果是我的话，我可能会用：

$ gh repo clone Guleixibian2009/Test

现在你的电脑上应该就有了一个本地仓库了。现在让我们来更新它！

首先我们来切换到一个新的分支（Git版本库中存放项目不同版本的一种机制）。然后，做一点点修改，然后再提交回远程端（在这里就是GitHub！）：

$ git checkout –b test

$ mkdir Test12.17

$ cd Test12.17

$ echo “What makes unicorns cry?”>>test.txt

现在我们已经生成了一个新文件夹，一个新文件并进入了一个新的分支。然后，用大家都熟知的Git来进行以下操作：

$ git add .

$ git commit –m “Update 12.17”

$ git push

现在貌似我们的更改已经被上传到GitHub上面去了！不过很明显，我们提交到的是一个叫做test的分支，我们还需要把他提交到master，即主分支上去。

1. PRs

其实我们的确可以将更新直接提交到master上，但这不是重点——在GitHub上更多的是多人合作项目，这时将更新代码直接提交到主分支上貌似就有点风险了（万一有Bug呢）。然后，我们就可以尝试一种新的方式——Pull Request。

Pull Request 是一种通知机制。你修改了他人的代码，将你的修改通知原来的作者，希望他合并你的修改，这就是 Pull Request。Pull Request 本质上是一种软件的合作方式，是将涉及不同功能的代码，纳入主干的一种流程。这个过程中，还可以进行讨论、审核和修改代码。

所以，你学废了吗……

1. 创建PR

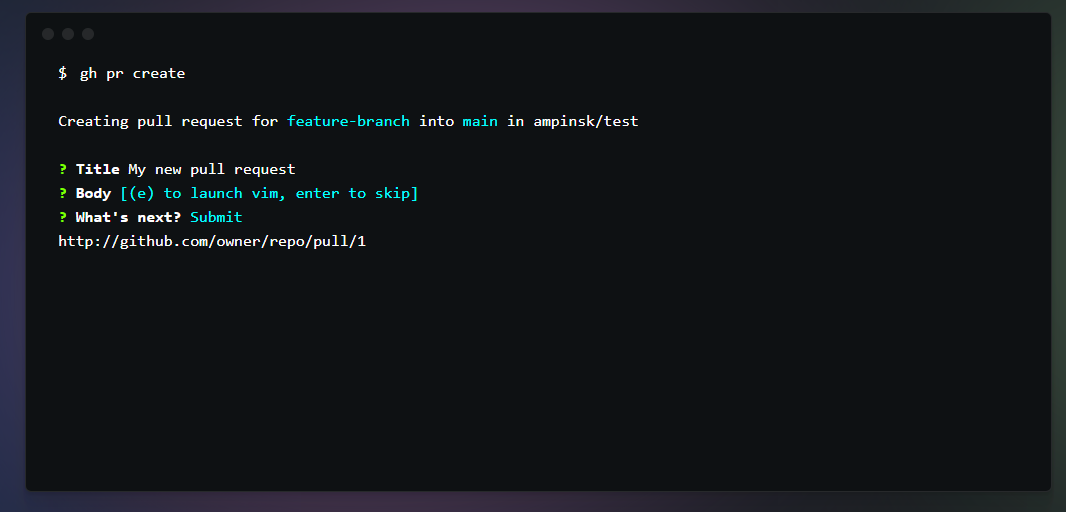
现在我们来尝试创建一个新PR。当然，我们是个人项目，所以PR仅起到演示的效果。

如图所示，创建一个PR我们需要运行：

$ gh pr create (等同于 $ gh pr cr)

同时我们会需要提供例如标题、基本信息等等等等。当输出类似以下的提示时，你就成功了。

https://github.com/Guleixibian2009/test/pull/1

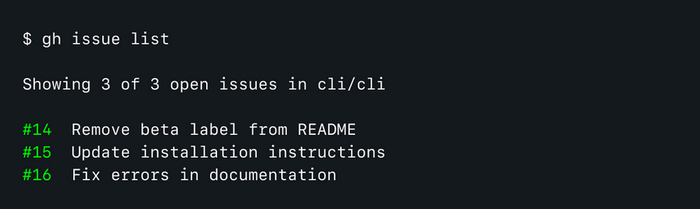


1. PR的合并

很明显，单独创建一个PR是没有用的。我们需要把他合并到主分支里！首先我们看一下到底有没有这个PR：

$ gh pr list

这个命令可以帮助我们查看所有的PR。



然后我们来合并刚刚那个PR。

$ gh pr merge 1