gitbook——使用笔记

morrowind

Published
with GitBook

目錄

本笔记为记录本人初次接触gitbook所学内容

本人学习过程中主要参考：

Markdown 简明语法

写作过程参照：

GitBook 中文解说

学习资料：

MarkDown 语法说明

Markdown 快速入门

Node 入门

感谢你能阅读这本手册，因为你确定这本手册值得花时间阅读。

2015年12月23日——2015年12月29日

前言

2015年10月首次看到使用gitbook写的教程分享，当然，那个时候还不知道那个教程是用gitbook写的。再次接触到gitbook时已是12月低，因需要学习logstash的知识并做实际部署操作，在搜索资料时，又看到与之前所看到的相同结构的教程，而偶然间就发现这些电子书籍都是发布在gitbook.com上的。

简单查阅后，发现gitbook是一个简单实用的书籍分享网站。我正准备将最近所读的书籍做简单整理，原打算自己部署web服务将其分享，而当看到gitbook后，我决定就用她来记录我所学的知识。

那首先要做的就是学习并使用gitbook，因此我决定就将我学习并使用gitbook的过程记录下来，并做为首篇使用gitbook所写的文章

期望通过分享这本笔记所记录的内容，能使您快速了解gitbook，并上手写作简单记录笔记。

准备工作和所具备知识

如果您只是使用gitbook.com的在线编辑器或客户端，那您只需学会如何使用gitbook工具即可。而如果您想脱离gitbook.com的编辑器去创建一部电子书籍的话，就会涉及到Node.js和Markdown。

gitbook是Node.js代码库的命令工具，使用GitHub/Git与Markdown(或AsciiDoc)就能制作漂亮的电子书籍。

上面所涉及到的知识不会也没关系，我会介绍些简单知识，为了达到独立制作电子书的目的。

需要安装的软件

除了使用gitbook.com的在线编辑器外，还需安装一下软件。

- **GitBook Editor** 客户端编辑器

  GitBook Editor 是一个适合任何平台的客户端，下载地址：https://www.gitbook.com/editor

- **Node.js**

  Node.js是一个基于Chrome V8引擎的JavaScript运行环境。Node.js使用了一个事件驱动、非阻塞式I/O的模型，使其轻量又高校。Node.js的包管理器npm，是全球最大的开源库生态系统。下载地址:https://wwww.nodejs.org/en/download/
  或http://nodejs.cn/download/

> node.js有多种安装方式，可以下载编译好的二进制包解压后直接使用，也可通过包管理器安装，还可通过下载源码包编译安装。
>
> 通过包管理器的安装方式请参照官方文档
>
> 这里我简单介绍下使用源码编译安装。
>
> 1.下载node.js源码并解压
>
> ```
> $wget https://nodejs.org/dist/v4.2.4/node-v4.2.4.tar.gz
> $tar zxf node-v4.2.4.tar.gz && cd node-v4.2.4
> ```
>
> 2.配置源码并编译
>
> ```
> $./configrue --prefix=/usr/local/
> $sudo make && sudo make install
> ```

- **npm**

  NPM的全称是Node Package Manager,是Node.js包管理和分发工具。通过此工具安装gitbook-client

```
$npm install -g gitbook-cli
```

- **Markdown**

  Markdown是一种轻量级标记语言，创始人为约翰 格鲁伯(John Gruber)。它允许人们“使用易读易写的纯文本格式编写文档，然后转换成有效的XHTML(或HTML)文档”。 这种语言吸收了很多在电子邮件中已有的纯文本标记的特性。 ——维基百科

- **Calibre**

  `gitbook` 制作pdf、epub等格式电子书时会用到 `ebook-convert`

  > `Calibre` 安装可见官网

说明

本笔记所有操作均是基于ubuntu 14.04 x86_64平台

常见错误

1.在对node.js源码配置过程中会报如下错误:

```
WARNING: failed to autodetect C++ compiler version (CXX=g++)
```

安装gcc即可解决次错误，安装方式如下:

```
sudo apt-get  install  build-essential
```

2.使用编译好的二进制包时，需要将解压后的软件包加入到环境变量，这样就可以在任何位置使用软件包中的程序了

1.简介

GitBook除了是一个命令行工具，它同时也是间书籍发布平台和电子书店。桌面版本同时支持Mac、Windows、Linux三种平台。

支持格式

GitBook工具能够制作多种格式书籍：PDF、ePub、mobi与线上阅读版本（网站HTML）。

它是开源的,你可以在这里看到工具的源代码，亦可在GitHub上提交问题与意见，甚至写组解决问题或改善程序。

GitBook.com

GitBook.com是一个制作与发布电子书籍的开放平台

> 其实这个网站还有电子书店的展示、搜索、购买，也可让出版者自由上架出售书籍，同时底层还是**Git**版本管理的中央仓库，以及将草稿转换为各种格式的网站引擎。

GitBook 的基本特性：

1. 以Markdown轻量级标记语法作为编辑“原稿”的基础。
2. 使用Git作为版本管理架构。
3. 通过云端服务生成各种通用电子书格式。
4. 支持在浏览器中阅读，增加了JavaSript。
5. 读者可直接付费购买，支持创作与正版流通。
6. 提供OPDS流通，可在移动设备上使用支持的阅读软件

1.1. GitBook.com账户注册

访问gitbook.com，如有GitHub帐号，直接使用GitHub帐号进行授权注册，这里使用GitHub帐号，是因为与GitHub进行关联时无需再次做GitHub授权操作。当然，你也可以注册一个GitBook帐号。 点击登录后，选择使用GitHub帐号登录（图1-1）

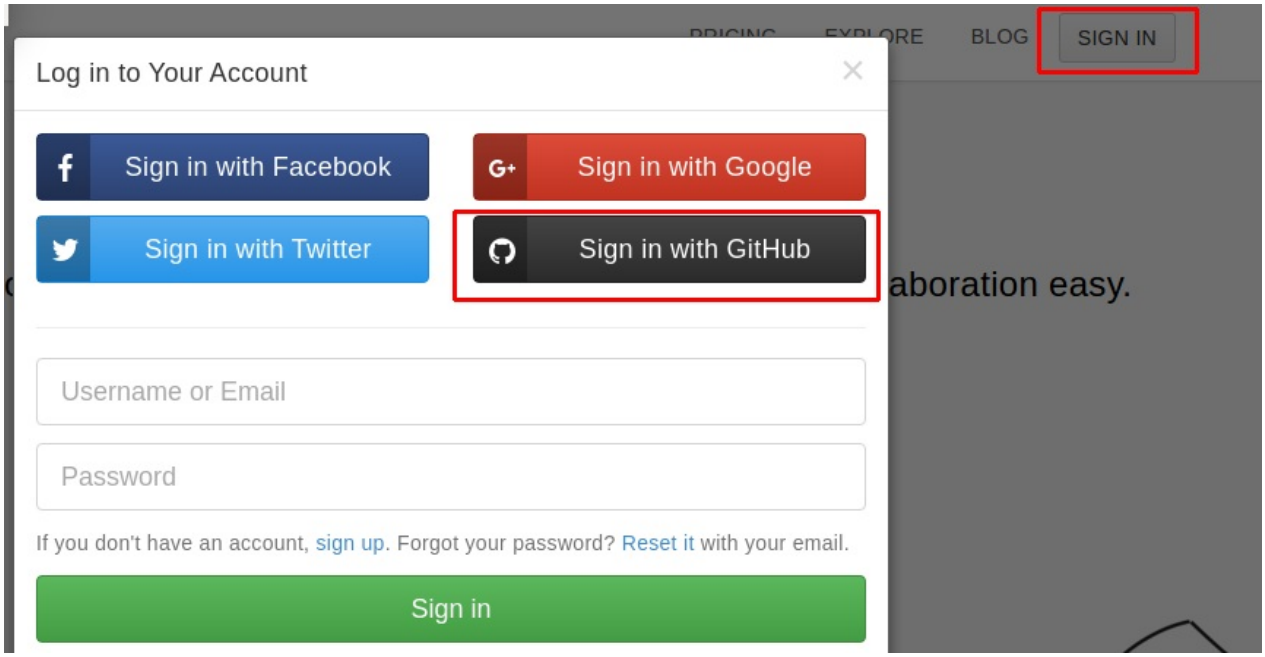


图1-1

随后会跳转到GitHub登录页面（图1-2）

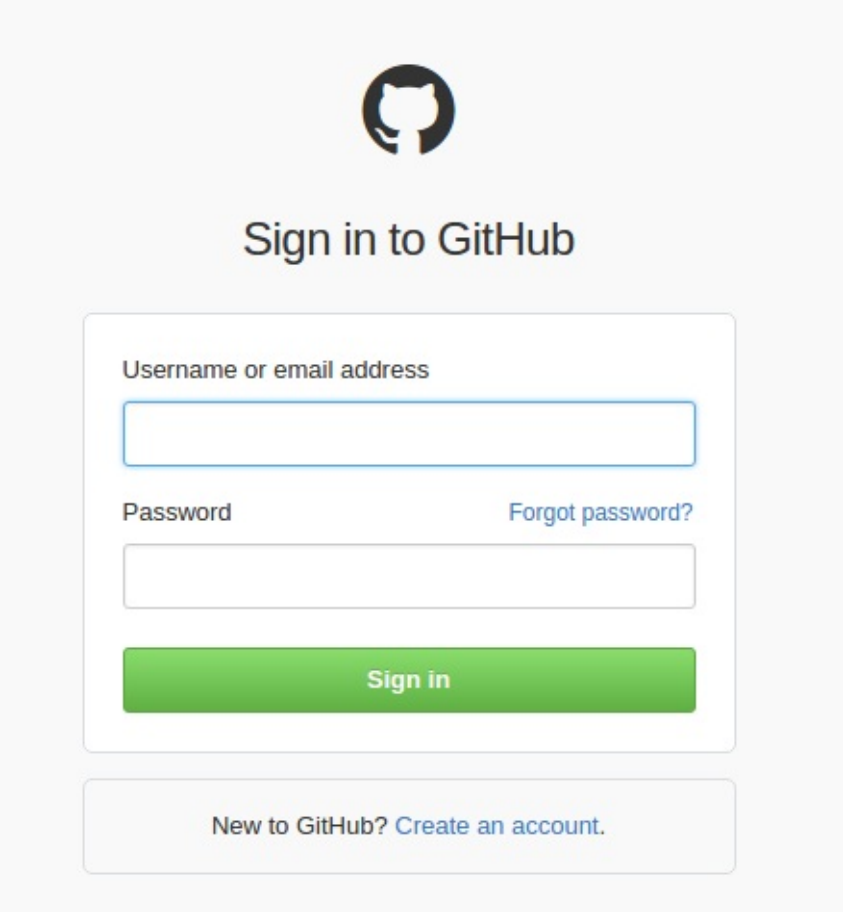


图1-2

输入帐号登录后，会跳转到的授权页面（图1-3），图中已有一个存在的授权，是我变更权限时的页面，首次授权的时候是只有一个Added的，而不应该存在 `Removed`

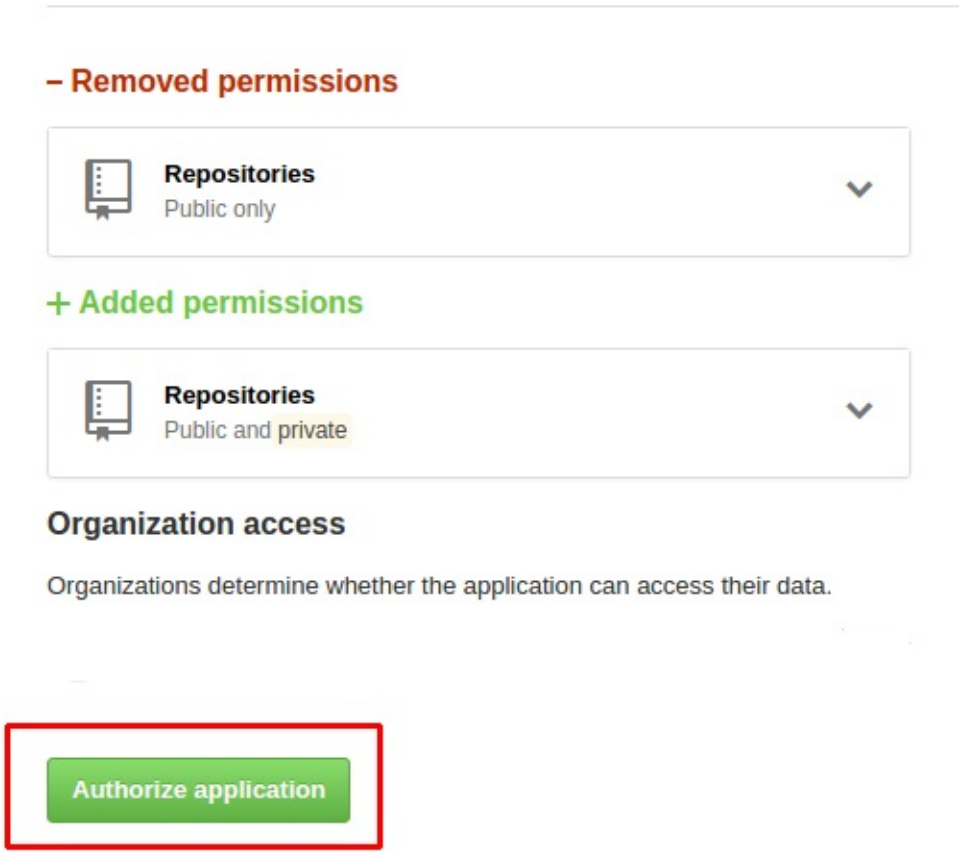


图1-3

授权通过后，会跳回GitBook页面，这时点击“帐号设置”选项（图1-4）进行邮箱设置（不设置邮箱是无法创建电子书项目的）图1-5

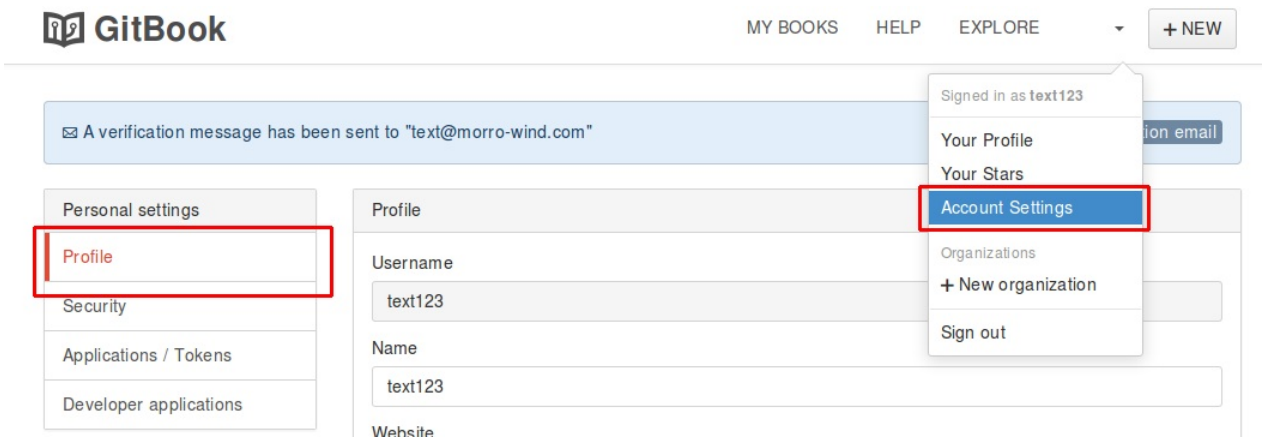


图1-4

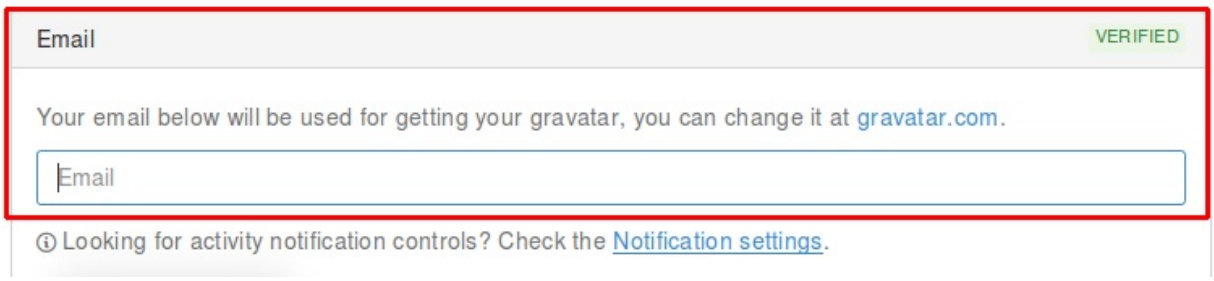


图1-5

1.2 在线创建一本电子书

完成对帐号的基本设置，就可以创建第一本电子书了。右上角选“MY BOOKS”进入我的图书，在没有图书的时候，看到的画面 图1-6

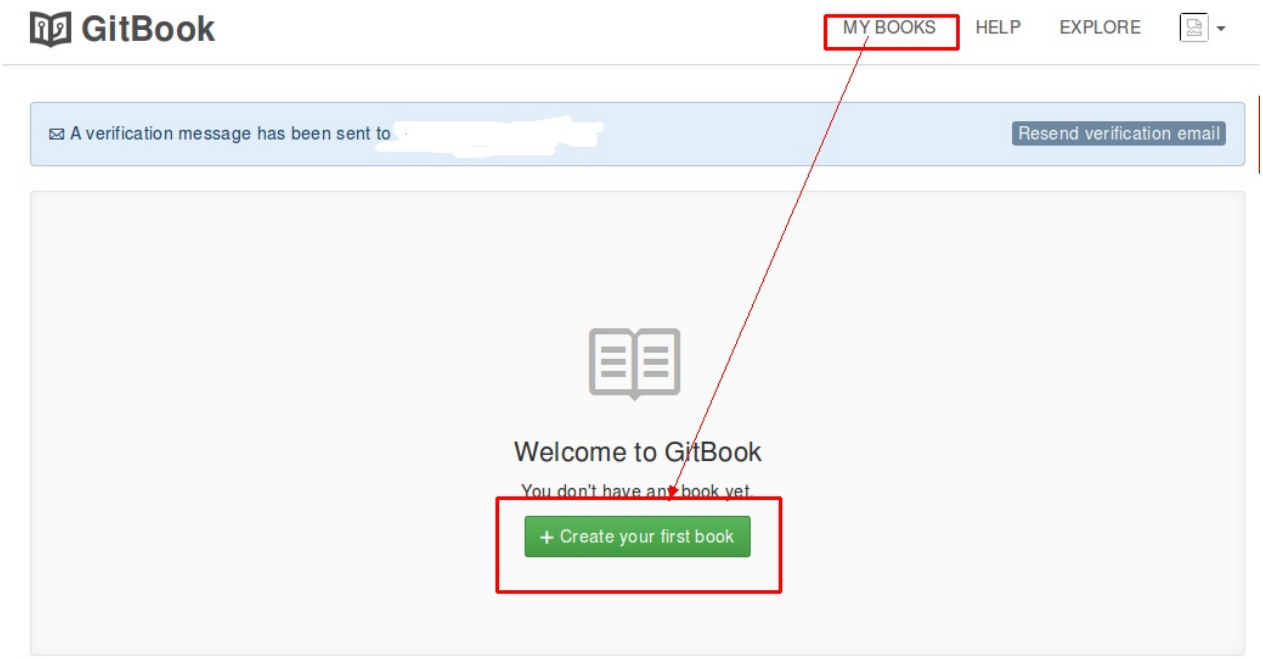


图1-6

点击“Create your first book”,进入创建页面（图1-7），填写书名、url以及选择公开还是私有，写好信息后，点击“Create book”就创建成功了（一般我们都会选择公开，私有书籍服务需要购买）。

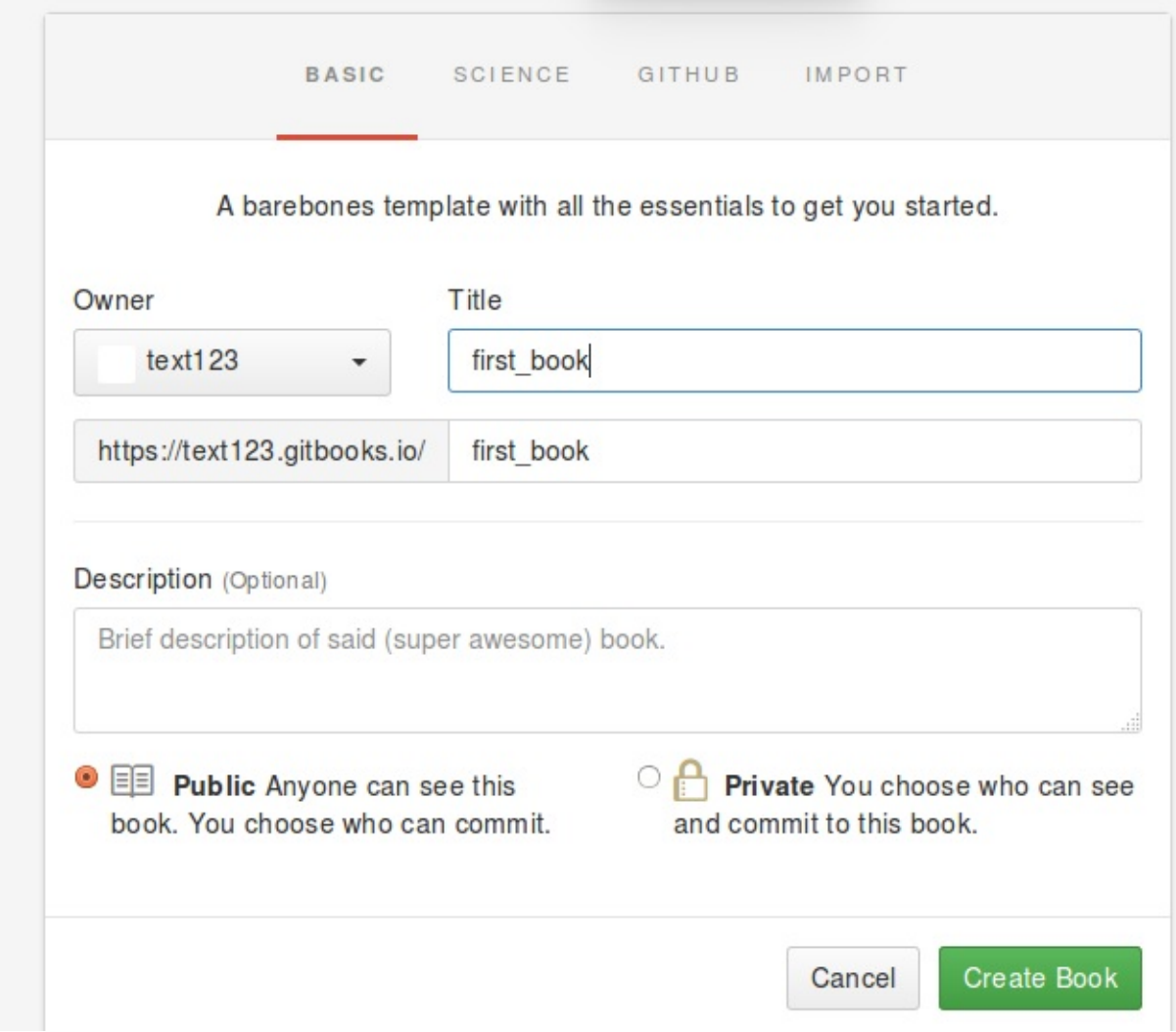
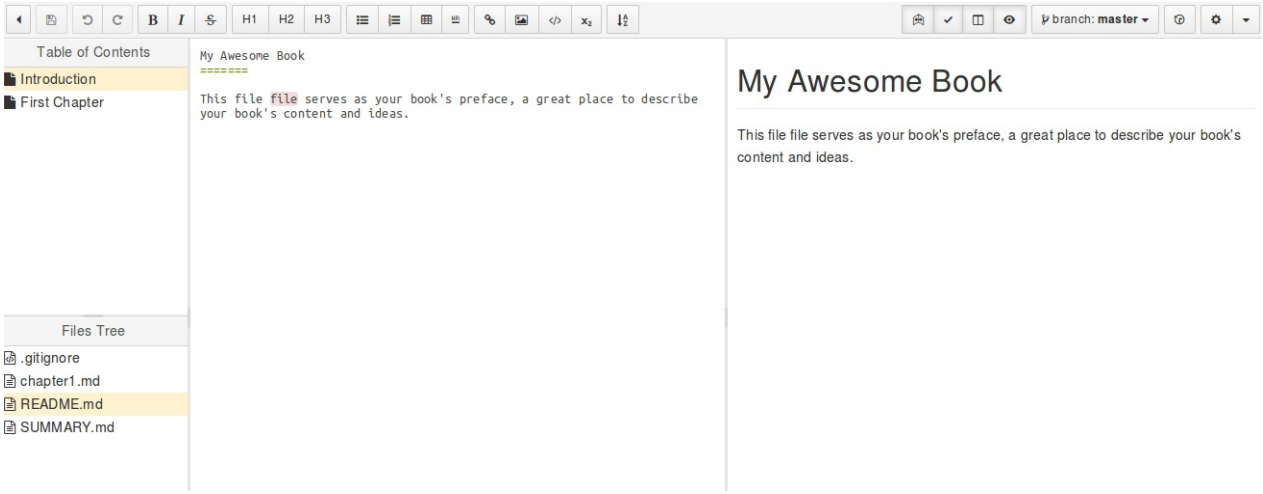

BASIC
SCIENCE
GITHUB
IMPORT
A barebones template with all the essentials to get you started.
Owner
Title
text123
first_book
https://text123.gitbooks.io/
first_book
Description (Optional)
Brief description of said (super awesome) book.
Public Anyone can see this book. You choose who can commit.
Private You choose who can see and commit to this book.
Cancel
Create Book

图1-7

1.3. GitBook在线编辑器

创建好图书，开始写作后，我们进入了GitBook的在线编辑器，这个编辑器分了三个主体部分



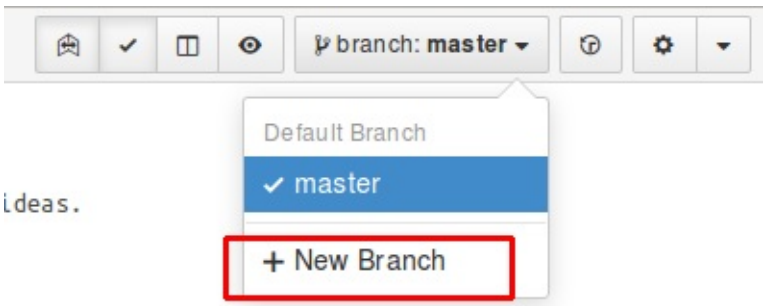
1.中间是编辑区域，使用Markdown语法；
2.左上方是章节目录，鼠标右键可以新增章，在上右键可以新增节，托放可以改变顺序；
3.左下方是文件列表，鼠标右键可以新增目录或文件。这里面最必要的是 `README.md` 和 `SUMMARY.md` 。前者是书籍的简介，后者是真正决定书籍目录结构的，比如多层次目录结构，需要手动修改SUMMARY.md（左上方会立刻呈现效果）。
4.右侧是预览窗口，可以点击“眼睛图标”关闭/开启预览区域。其实可以关闭左侧区域和右测预览区域，专心写作。

草稿模式

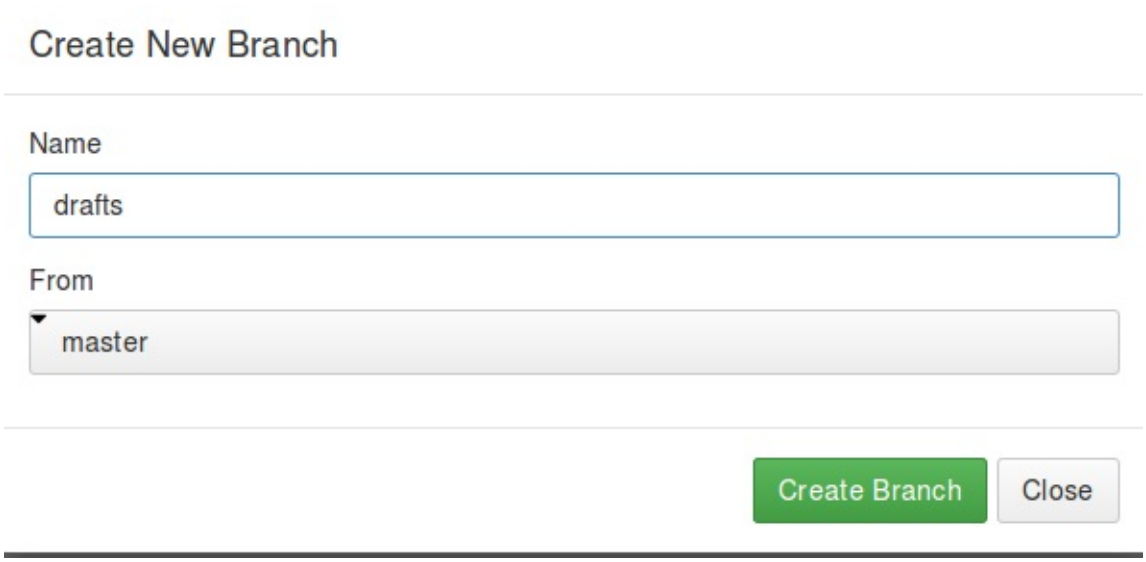
先不要急于写如文字，我们看到在线编辑器的右上方的**branch:master**按钮，这说明目前编辑模式正处于“主分支”亦“对外发行”模式，在此模式下，无论输入什么，一旦保存，GitB ook就会立即生成各种版本的电子书对外发布。

大多数人还是不希望读者看到，连一个章节都没有完成的书籍。况且读者一旦订阅，那每发布一次，读者都会收到更新通知，这种体验是不够人性化的。

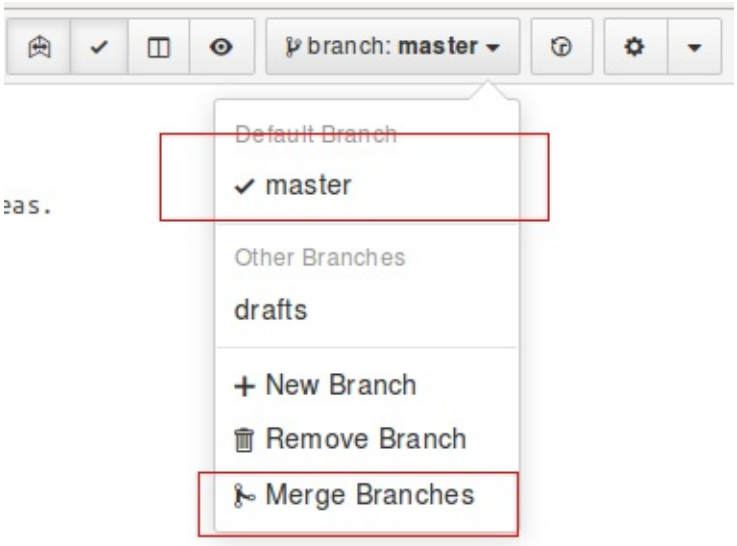
理想状态是：我们每完成一个章节的内容并确认无误后，才进行发布更新。而在“草稿模式”下，不论我们如何操作，都不会将内容发布出去，除非将“草稿模式”下的所做所为，提交到“发行模式”，那接下来就进入“草稿模式”吧。其实就是在主分支上创建一个分支，点击“**+New Branch**”



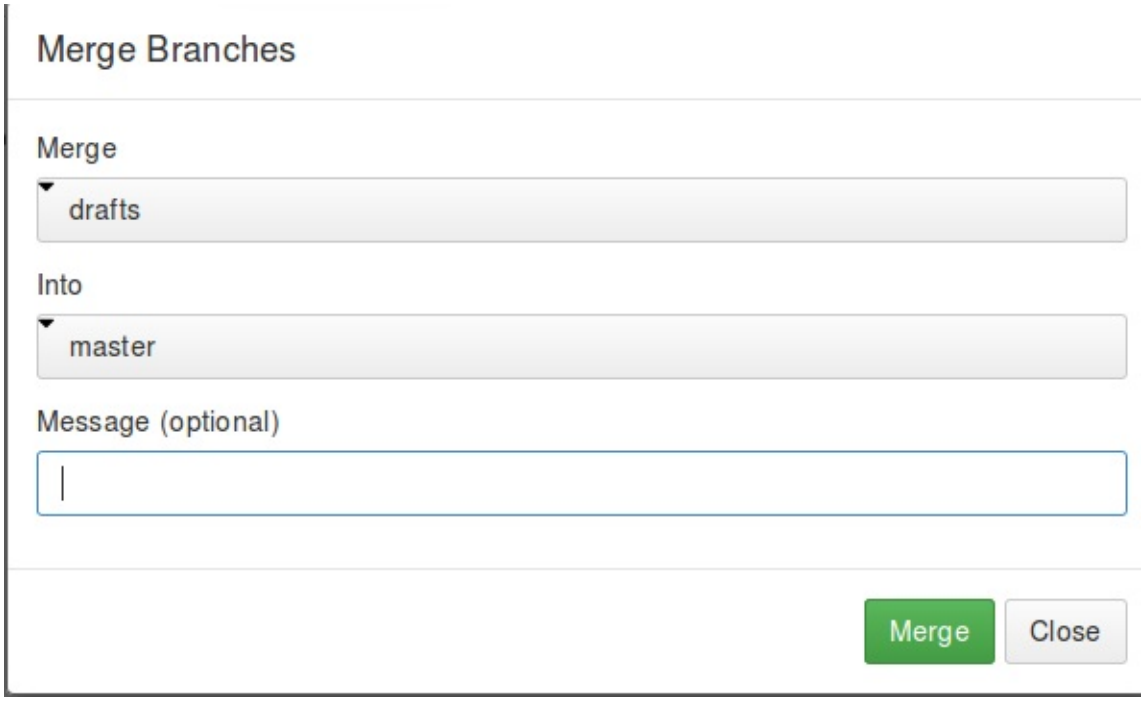
在弹出窗口输入**drafts**按下确认，这是编辑器就会从“发行模式”切换到“草稿模式”了。



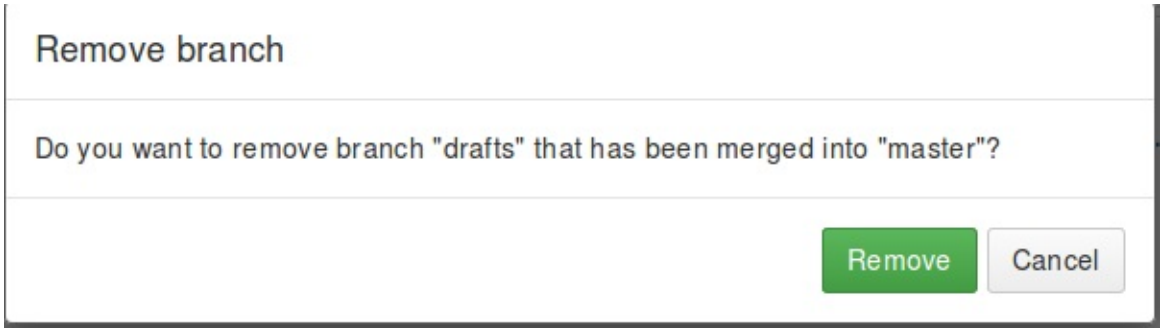
当我们完成一部分写作，想对外发布时，就切换到“**branch:master**”模式下，并点击“**Merge branches**”，将草稿下的内容合并到“**master**”上来.



在弹出的窗口,输入友好合并信息



按下Merge,接着会弹出确认窗口



没错，按下**Remove**，再看下“发行版”、主分支（master branch）已经合并了新的内容，发行版电子书籍也进行了更新，随后订阅的读者就会收到更新通知。

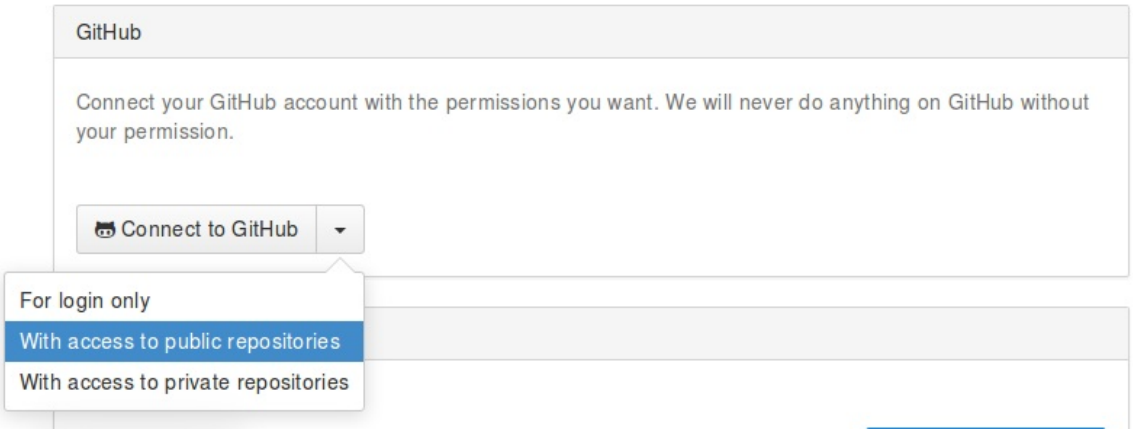
目录操作指南

当在左上目录区域新增章节时，编辑器会自动将新增章节加到SUMMARY.md文件中，也就是说这个章节已经被加入到电子书中了！如果不希望这个章节出现在电子书中，需要手动修改SUMMARY.md文件。

而新增一个章节时，必须在文件列表区存在与之关联的文件才可进行新增章节的写作。

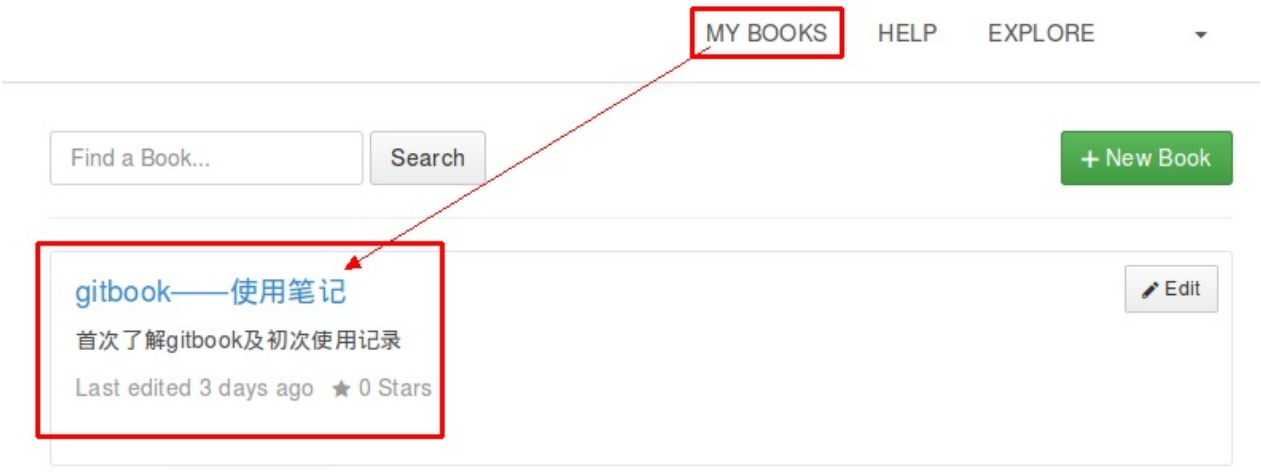
1.4. GitHub关联并创建仓库

如果在注册时没有使用GitHub帐号授权注册，那我们现在就来做与GitHub的帐号关联。
进入“**Account setting**”找到“Gitub”一项，在下拉项中选择“**With access to public repositories**”,随后会跳转到GitHub登录页面，输入帐号密码登录后看到的页面与前面使用GitHub登录时相同的页面。依照授权登录操作即可。



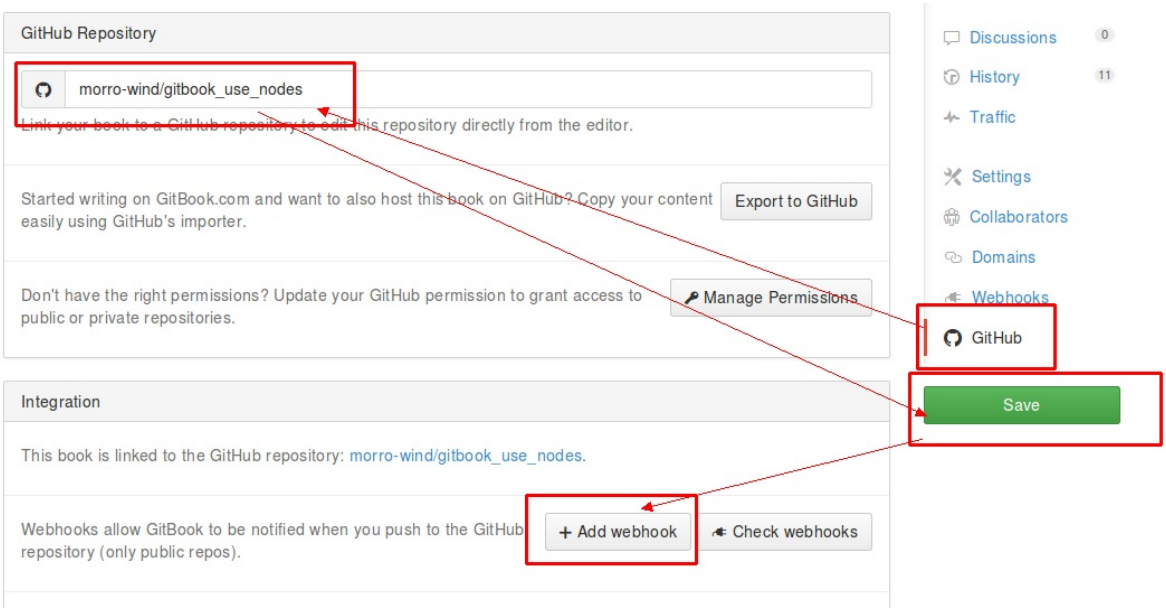
创建电子书GitHub仓库

在“**MY BOOKS**”标签页，找到需要创建GitHub仓库的那本电子书，并点击进去



在右侧，点击**setting**，进入电子书设置页面

找到**GitHub**设置项，点进去，设置电子书对应的GitHub仓库名(没有则在GitHub上创建一个)，保存后再点击**+Add webhook**



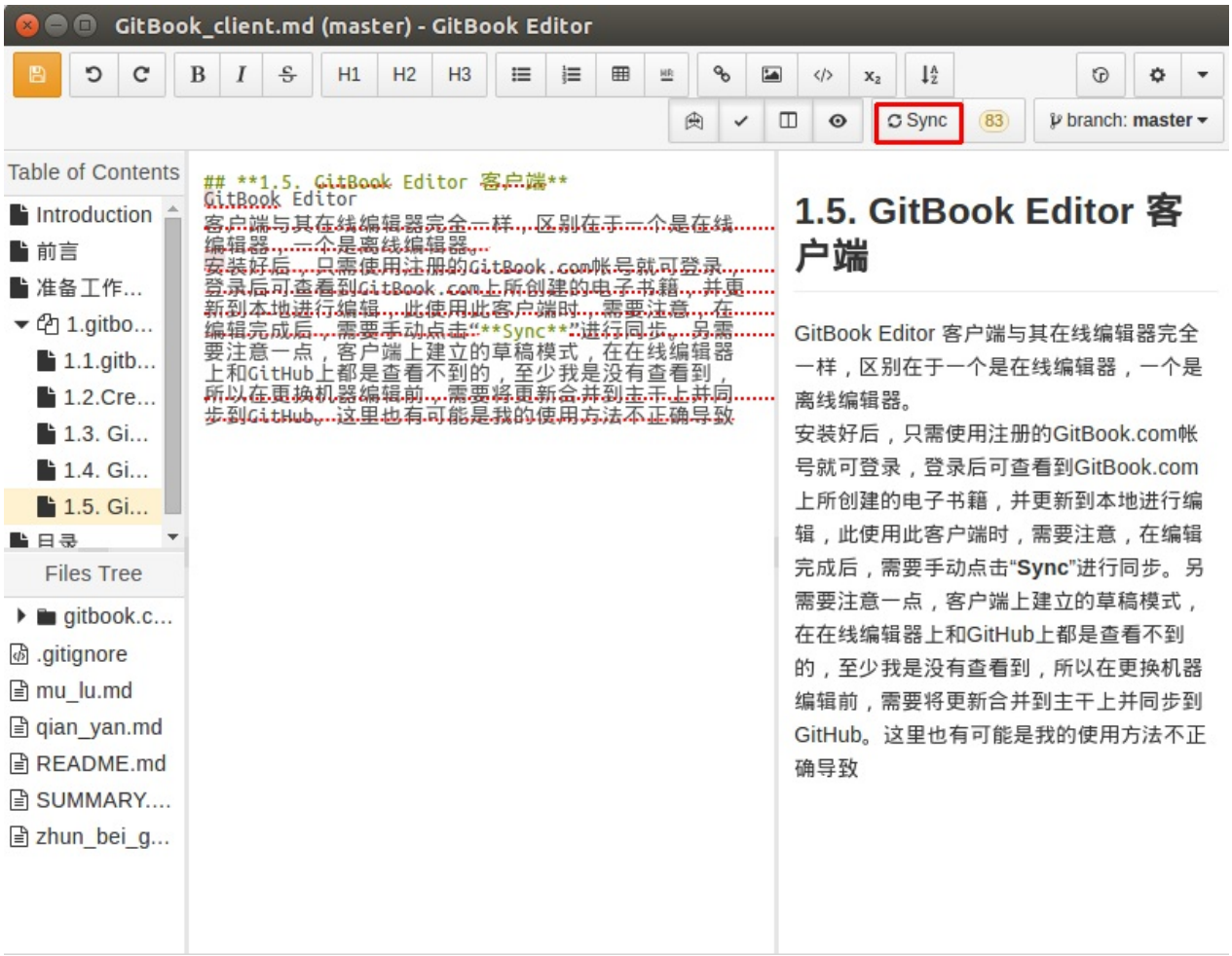
如此一来，此后的电子书都将先更新到GitHub上，然后在推送到gitbook上进行发布。

这也方便了无论在什么位置，都可以将GitHub上的电子书拉到本地进行编辑，然后在方便时再将更新提交到GitHub上进行GitBook发布。

1.5. GitBook Editor 客户端

GitBook Editor 客户端与其在线编辑器完全一样，区别在于一个是在线编辑器，一个是离线编辑器。
安装好后，只需使用注册的GitBook.com帐号就可登录，登录后可查看到GitBook.com上所创建的电子书籍，并更新到本地进行编辑

此使用此客户端时，需要注意，在编辑完成后，需要手动点击“**Sync**”进行同步



另,需要注意一点，客户端上建立的草稿模式，在在线编辑器上和GitHub上都是查看不到的，至少我是没有查看到，所以在更换机器编辑前，需要将更新合并到主干上并同步到GitHub。这里也有可能是我的使用方法不正确导致.

2.GitBook 终端基础

Gitbook 其实是一个终端命令工具。

无论使用在线编辑器还是本地编辑器，还是使用文本编辑器（如Sublime Text）再推送到仓库，都是使用gitbook.com服务端生成的电子书。这样一来就无法在本地生成电子书籍预览查看或发布到别的平台（如将HTML格式发布到web服务上），而解决这个问题的工具，就是 `gitbook` 终端命令工具了。

使用 `gitbook` 命令的前提，必须已经安装好 `Node` 、 `Npm` ,在准备工作中，我们已经将它们安装并配置好了，现在来查看下版本。

```
$node --version
v4.2.3
```

```
$npm -version
2.14.7
```

安装gitbook-cli

`gitbook` 终端命令工具，安装时安装的是 `gitbook-cli` 。

```
$sudo npm install -g gitbook-cli
$gitbook versions:install latest
```

- `gitbook versions` 显示本地目前可用的GitBook版本；
- `gitbook versions:available` 显示有哪些可以安装的版本；
- `gitbook versions:install latest` 安装最新版本；
- `gitbook versions:install 2.3.3` 安装指定版本；
- `gitbook versions:uninstall 2.3.3` 卸载指定版本

2.1. GitBook 图书结构介绍

一本由 GitBook 创建的电子书籍，除了实际内容文件外，还应包含如下文件：

- `README.md` :书的介绍文字，如前言、简介，在章节中也可做为章节的简介。
- `SUMMARY.md` :定制书籍的章节结构和顺序。
- `LANGS.md` :多种语言设置。
- `GLOSSARY.md` :词量表和定义描述。

> `README.md` 和 `SUMMARY.md` 是GitBook 制作电子书的必要文件，可用 `gitbook init` 命令自动生成，其余文件如有需要，可手动添加。

章节设置

GitBook 使用 `SUMMARY.md` 文件作为书籍的目录结构，既多层次章节设置。它同时也被用来制作书籍目录（TOC-Tables Of Contents）。

`SUMMARY.md` 的格式只是简单的连接列表，连接的“名称”就是章节的“标题”，连接标的则是实际内容“文件”（包含路径）。
章节的层级，就是根据清单的层级关系定义的。

> 多层级可将书籍分为“部”、“章”、“节”或“小节”，而且不会自动赋予标号或固定名称，选择自己想要的结构即可。
> 没有在 `SUMMARY.md` 中出现的文件，GitBook 在生成各种格式电子书的时候是不会将其包含在内的，因此可以自由撰写草稿、参考文件等，只要不将它们加入到 `SUMMARY.md` 文件中，就不会被发布。

目录示例

`Summary.md`

```
#Summary
* [第一章](chapter1.md)
* [第二章](chapter2.md)
* [第三章](chapter3.md)
```

多层结构目录示例

`Summary.md`

```
#Summary
* [第一部](part1/README.md)
    * [写作是没好的](part1/writing.md)
    * [GitBook 也不错](part1/gitbook.md)
* 第二部
    * [我们欢迎读者回馈](part2/feedback_please.md)
    * [对作者更好的工具](part2/better_tools.md)
```

ps. 可以看到“第一部”有连接到实际的文件，可以放一个简单的“章节简介”，或特殊的标题或引言，甚至是展现一张图片都可以。而“第二部”则没有连接任何文件，这样在发布书籍时，就有可能导致“第二部”解析错误，后面有连接实际文件的不会受此影响。

> `README.md` 文件默认会作为多层目录结构中的章节连接文件。

多语言

GitBook 支持多种语言编写图书。每种语言必须是一个子目录，子目录结构与 GitBook 结构相同（拥有各自的README.md、SUMMARY.md以及实际内容文件）， `LANGS.md` 在外层父目录（书籍项目根目录），其内容格式如下：

`LANGS.md`

```
* [English](en/)
* [zh-hans](zh-hans/)
* [zh-tw](zh-tw/)
```

例子 学习 Git.

忽略目录和文件

GitBook 会读取 `.gitignore` , `.bookignore` 以及 `.ignore` 这三个档案，根据里面的内容，忽略特定的文件或子目录。（格式为一行一个文件或目录。）

`.gitignore`

```
#忽略 test.md 文件
test.md

#忽略 "bin" 目录下所有文件
bin/*
```

术语表

在术语表中指定要显示的术语和其各自的定义。基于这些条件 gitbook 会自动建立索引，并在内容页面中高亮显示这些术语。

GLOSSARY.md

```
# term
Definition for this term

# Another term
With it's definition, this can contain bold text and all other kinds of inline markup ...
```

2.2 Markdown 标记语法

GitBook 使用Markdown 标记语法。

本节仅简要介绍 Markdown 的基本语法与展现，若要详细了解，英文资源可以看其发明人的说明：Jon Gruber's original spec 以及GitHub 的扩展版 Github-flavored Markdown infoj page。Markdown.cn有其中文详解；想看看俗称 GFM-GitHub 风格的 Markdown 语法，也找得到中文翻译。

标题

```
# H1
## H2
### H3
#### H4
##### H5
###### H6

最常使用的是 H1 和 H2 标题，还有更明显的另一种写法：

Alt-H1
======

Alt-H2
------
```

> 此处 `-` 是负号，不是 `_` 下横号， `_` 下横号，显示的是双横线

H1

H2

H3

H4

H5

H6

最常使用的是 H1 和 H2 标题，还有更明显的另一种写法：

Alt-H1

Alt-H2

强调语法

```
强调，例如意大利斜体，可以使用 *asterisks* 或 _underscores_。
加重语气强调，例如粗体，可以用 **asterisks** 或 __underscores__ 。
可以混用这两种 **asterisks and _underscores_**。
给文字加上删除线，是这样 ~~Scratch this.~~
```

清单

下面的示例为了清楚展示缩排时需要的空格，使用了点号，实际撰写时只需要相同数量的空格。

```
1. 第一个有序列表项
2. 另一个项
..* 无序的子列表
1. 数字本身是否排序并不重要，统统使用相同的数字也可以
..1. 有序的子列表
4. 另一项

...可以在一则项目中使用缩进的段落格式。注意上面的**空行**，还有本段前的**空格**(至少一个，我们使用了三个

...在一个段落中**强制换行**，在语句后方加入两个**空格**。..
...这个被强制设置的独立行，依旧在同一个段落中。..
...（有些人觉得使用空格强制换行太麻烦，例如 GFM 就不需要）。

* 可以使用星号建立无序目录
- 或是短横线（负号）
+ 使用半形加号也可以
```

连接设置

有两种方式可以建立文中的链接。

```
[这是一个行内链接](https://www.gitbook.com)

[这是一个带有标题的行内链接](https://www.gitbook.com "GitBook's Homepage")

[这是一个参考链接][Arbitrary
case-insensitive reference text]

[这是一个对应到 Git 仓库文件的相对参考链接](../blob/master/LICENSE)

[参考标的物也可以使用数字][1]

直接使用文字对应也可以 [这段文字链接到参考项目]

参考项目可以写在文档的最后，有点像书内的注解（注脚）。

[arbitrary case-insensitive reference text]: https://www.mozilla.org
[1]: http://slashdot.org
[这段文件链接到参考项目]: http://www.reddit.com
```

图片

```
这是我们的 logo （将鼠标移动到图片上会显示图片标题）：
行内格式：
![alt text](https://github.com/adam-p/markdown-here/raw/master/src/common/images/icon48.

参考链接结构：
![alt text][logo]

[logo]: https://github.com/adam-p/markdown-here/raw/master/src/common/images/icon48.png
```

程序代码与语法高亮标示

```
行内 `code` 必须使用 `back-ticks` 将文字包起来(一般在键盘左上方的地一个键)。
```

整段独立展示的代码必须使用成对的三个 back-ticks ``` 包裹起来，或是使用四个空格缩排。建议使用第一种方法，可以让代码具有可读性。

```
```javascript
var s = "JavaScript syntax highlighting";
alert(s);
```

```python
s = "Python syntax highlighting"
print s
```

```
No language indicated, so no syntax highlighting.
But let's throw in a <b>tag</b>.
```
```

引言

```
> 引言(Blockquotes)常常出现在电子邮件中，表示摘录来信的原句。
> 这一行是引言的一部分。

Quote break.

> 这是一段非常长的引言区块，只要在句首使用了正确的符号和空格，就可以持续不间断的撰写，整段文字都是会被包含
```

水平分割线

```
三个或三个以上的符号，必须在独立的一行，前后不能有其他文字。

---
短横线（Hyphens）

***
半形星号（Asterisks）

___
下底/横线（Underscores）
```

空行分隔段落

知道Markdown如何进行分段是很重要的，基本上空行代表前后的文字都会是段落（在HTML中以 `<p>` 与 `</p>` 包裹起来）。如果使用桌面编辑软件，有些可以微调空行的显示，让编辑区看起来不那么松散；甚至有些软件直接取消分行的设置，按下 `return` 就代表分段了。

最好的方法就是实践，打开编辑器，启用预览窗口，试着输入几次，看看有什么结果，很快就熟悉了。

尝试输入下面的内容看看：

```
Here's a line for us to start with.

This line is separated from the one above by two newlines, so it will be a *separate par

This line is also a separate paragraph, but ...
This line is only separated by a single newline, so it's a separate line in the *same pa
```

(注意：Markdown 套用了 GFM 的分行模式，因此强制断行并不需要在行后加入两个空格。)

Youtube影片

虽然无法直接嵌影片，但可以采用图片链接的模式：

```
<a href="http://www.youtube.com/watch?feature=player_embedded&v=YOUTUBE_VIDEO_ID_HERE
" target="_blank"><img src="http://img.youtube.com/vi/YOUTUBE_VIDEO_ID_HERE/0.jpg"
alt="IMAGE ALT TEXT HERE" width="240" height="180" border="10" /></a>
```

也可以直接在 Markdown 这样写，但会失去尺寸和边界的设定：

```
[![IMAGE ALT TEXT HERE](http://img.youtube.com/vi/YOUTUBE_VIDEO_ID_HERE/0.jpg)](http://w
```

2.3. 使用 GitBook 制作电子书

通过前面的学习，我们已经大概了解了 GitBook 电子书籍的结构，并且也了解了 Markdown 标记语法的基本使用。下面我们使用 GitBook 来实际制作一本电子书。

1.创建存放电子书的目录

```
$mkdir mybook
```

2.进入图书目录，并生成 `README.md` 和 `SUMMARY.md` 两个必要文件。

```
$cd mybook
$gitbook init
```

3.编写图书简介 `README.md` 文件输入如下内容：

```
#Introduction
这是一本使用 gitbook 制作电子书籍的快速上手指南。
分为两个章节：
第一章介绍 gitbook.com 在线平台
第二章介绍 gitbook 基础知识
```

4.创建章节目录：

```
$mkdir part1
$mkdir part2
```

5.编写实际内容

进入第一章目录，编写第一章内容：

```
#README.md
GitBook除了是一个命令行工具，它同时也是间书籍发布平台和电子书店。桌面版本同时支持Mac、Windows、Linux三

#register.md
访问gitbook.com，如有GitHub帐号，直接使用GitHub帐号进行授权注册，这里使用GitHub帐号，是因为与GitHub
```

进入第二章目录，并编写其内容：

```
#README.md
使用 `gitbook` 命令的前提，必须已经安装好`Node`、`Npm`,在准备工作中，我们已经将它们安装并配置好了，现在

#markdown.md
本节仅简要介绍 Markdown 的基本语法与展现......
```

> 以上都只是截取了章节目录的部分文件和内容.

6.配置SUMMARY.md文件

每完成一个章节内容，即可在图书目录内进行连接，当然也可全部完成后在进行连接。

```
#SUMMARY.md
* [Introduction](README.md)
* [前言](qian_yan.md)
* [准备工作和所具备知识](zhun_bei_gong_zuo.md)
* [1.gitbook](part1/README.md)
   * [1.1.gitbook.com账户注册](part1/register.md)
* [2.GitBook 基础](part2/README.md)
   * [2.1.Markdown 语法](part2/markdown.md)
```

SUMMARY.md 结构说明

```
* 无序列表
[]章节名称
()章节对应的实际文件，包含相对路径
```

> 注： 另一种创建图书的方式是：先定义好 *SUMMARY.md* 内的图书结构，然后使用 *gitbook init* 命令生成图书目录和文件。这样的好处是有助于一本书的整体构思，坏处就是写作过程中很难变更图书结构(除非你不会有思维定式)。

在线查看

输入 `gitbook serve` ，在浏览器中输入 `http://localhost:4000` ，就可以看到和 GitBook 网站一样的网页版电子书。修改内容、保存，会看到网页自动重载并更新了内容。

现在已经拥有了一个可在本地离线编辑，并能立即展现最终效果的编写环境。

2.4. 生成格式

完成书籍撰写，就可以生成需要的格式进行发布了。

HTML网页版本

在书籍项目目录的跟目录输入 `gitbook install` ，会自动安装必要的插件与书籍项目指定的插件。

指定插件是由 `book.json` 这个文件控制，这里没有用到，因此GitBook只会预先安装一个 **high light** 插件，用来显示书中插入程序代码区段。(使用 `gitbook serve` 还会自动启用另一个 **livereload** 插件，用于自动重载更新后的页面。)

整个静态格式书籍会放在 `_book` 目录下，若是不需要通过浏览器查看，也可使用 `gitbook build` 构建。

可以把整个html目录上传到自己的网站空间，就连GitHub Pages 空间都可以用。

制作电子书文件

从书籍项目根目录执行：

- `gitbook epub` 制作 ePub 电子书
- `gitbook mobi` 制作 Kindle 电子书
- `gitbook pdf` 制作PDF电子书

完整命令

也可以从书籍项目外部执行，完整命令是：

```
gitbook epub [book] [output]
```

例如 `gitbook mobi ~/ebook/mybook ~/Desktop/mybook.mobi` 。制作电子书制定 `output` 时需包含完整的路径和文件名称。

想要将静态网站建立到指定目录，可以这样输入：

```
gitbook build --output=/site/mybook
```

输入 `gitbook help` 可以看到帮助信息。

结论

GitBook 可以说是未来出版的一个发展趋势，而且还在持续进化之中。

一般使用桌面客户端编辑器，或在线编辑器即可，只要注意草稿分支的使用技巧，完全不懂技术也能制作各种电子书籍。不怕终端命令、略懂 Node 和 NPM 的，可搭配 `gitbook-cli` 和 Calibre 则可以使用 GitBook 的所有功能。

2.4. Git 更新

本节只简单的介绍几个 `git` 命令

每一本书籍都对应一个Git HTTPS网址， GitBook的 git 主机还不接受 ssh 传输协议。

网址看起来像这样：

```
https://git.gitbook.com/{{UserName}}/{{Book}}.git
```

授权认证

主机需要认证你的 GitBook 帐号，跳出提示时输入你的 GitBook 账户名称和密码即可（可以使用专属的 API token）

存储你的认证许可 **(credentials)**

为了省去每次推送内容时都得输入帐号密码的麻烦，可以把 GitBook 的认证许可写到 `~/.netrc` 文件里。 新建或添加下面的内容到 `~/.netrc` 文件：

```
machine git.gitbook.com
  login USERNAME-or-EMAIL
  password API-TOKEN-or-PASSWORD
```

建议使用 **API TOKEN** ,因为安全性更高些；可以从网站后台的“API”选项中找到属于你的相信息。

建立新的书籍项目

```
touch README.md SUMMARY.md
git init
git add README.md SUMMARY.md
git commit -m "first commit"
git remote add gitbook https://git.gitbook.com/{{UserName}}/{{Book}}.git
git push -u gitbook master
```

推送到已有书籍项目

```
git remote add gitbook https://git.gitbook.com/{{UserName}}/{{Book}}.git
git push -u gitbook master
```

下一步进阶，就是“使用模板、插件”等，可参考 GitBook 的中文繁体说明

此指南到此就全部结束。

谢谢！