# GitBook

# Documentation

Published
with GitBook

# Table of Contents

# GitBook Documentation

This book contains the entire documentation for **GitBook** (platform and toolchain).

GitBook is a tool for building beautiful books using Git and Markdown. It can generate your book in multiple formats: **PDF**, **ePub**, **mobi** or as a **website**.

## Open Source Toolchain

The GitBook toolchain is open source and completely free, the source code of the tool is available on GitHub.

Issues and question related to the format and the toolchain should be posted at github.com/GitbookIO/gitbook/issues

## Help and Support

We're always happy to help out with your books or any other questions you might have. You can ask a question or signal an issue on the following contact form at gitbook.com/contact.

## Contribute to this documentation

You can contribute to improve this documentation on GitHub.

---

# Summary

# Format

The format's main focus is simplicity and readability.

GitBook uses a convention on top of markup files:

- **README**: Introduction of the book
- **SUMMARY**: Chapters Structure
- **LANGS**: Multi-Languages book
- **GLOSSARY**: List of terms with descriptions

A book needs at least a `README` and a `SUMMARY`.

# Output Formats

GitBook can generate your book in different ebook formats.

## Static Website

This is the default format. It generates a complete interactive static website.

## PDF (Portable Document Format)

Portable Document Format (PDF) is a file format used to present documents in a manner independent of application software, hardware, and operating systems. It's a very common format. Files have the extension `.pdf`.

## ePub (electronic publication)

EPUB (short for electronic publication; sometimes styled ePub) is a free and open e-book standard by the International Digital Publishing Forum (IDPF). Files have the extension `.epub`. ePubs are supported by Apple and Google devices.

## Mobi (Mobipocket)

The Mobipocket e-book format is based on the Open eBook standard using XHTML and can include JavaScript and frames. It is supported by Amaaon devices (Kindle).

# Readme and Introduction

The first page of your book is extracted from the `README.md`. If the file is not present in the `SUMMARY`, it will be added as a first entry.

## Use another file than README.md

Some books hosted on GitHub prefer to keep the README.md as a project introduction instead of a book introduction.

Since GitBook `>2.0.0`, it's possible to define which file to use as a README in the `book.json`:

```
{
    "structure": {
        "readme": "myIntro.md"
    }
}
```

# Chapters and Subchapters

GitBook uses a `SUMMARY.md` file to define the structure of chapters and subchapters.

The `SUMMARY.md`'s format is simply a list of links, the name of the link is used as the chapter's name, and the target is a path to that chapter's file.

Subchapters are defined simply by adding a nested list to a parent chapter.

## Simple example

```
# Summary

* [Chapter 1](chapter1.md)
* [Chapter 2](chapter2.md)
* [Chapter 3](chapter3.md)
```

## Example with subchapters split into *parts*

```
# Summary

* [Part I](part1/README.md)
    * [Writing is nice](part1/writing.md)
    * [GitBook is nice](part1/gitbook.md)
* [Part II](part2/README.md)
    * [We love feedback](part2/feedback_please.md)
    * [Better tools for authors](part2/better_tools.md)
```

# Markdown

GitBook uses the Markdown syntax by default.

This is intended as a quick reference and showcase. For more complete info, see John Gruber's original spec and the Github-flavored Markdown info page.

## Headers

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

Alternatively, for H1 and H2, an underline-ish style:

```
Alt-H1
======

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

## Emphasis

```
Emphasis, aka italics, with *asterisks* or _underscores_.

Strong emphasis, aka bold, with **asterisks** or __underscores__.

Combined emphasis with **asterisks and _underscores_**.

Strikethrough uses two tildes. ~~Scratch this.~~
```

## Lists

(In this example, leading and trailing spaces are shown with with dots: ·)

```
1. First ordered list item
2. Another item
··* Unordered sub-list.
1. Actual numbers don't matter, just that it's a number
··1. Ordered sub-list
4. And another item.

···You can have properly indented paragraphs within list items. Notice the blank line above, and the leading spaces (at

···To have a line break without a paragraph, you will need to use two trailing spaces.··
···Note that this line is separate, but within the same paragraph.··
···(This is contrary to the typical GFM line break behaviour, where trailing spaces are not required.)

* Unordered list can use asterisks
- Or minuses
+ Or pluses
```

## Links

There are two ways to create links.

```
[I'm an inline-style link](https://www.google.com)

[I'm an inline-style link with title](https://www.google.com "Google's Homepage")

[I'm a reference-style link][Arbitrary case-insensitive reference text]

[I'm a relative reference to a repository file](../blob/master/LICENSE)

[You can use numbers for reference-style link definitions][1]

Or leave it empty and use the [link text itself]

Some text to show that the reference links can follow later.

[arbitrary case-insensitive reference text]: https://www.mozilla.org
[1]: http://slashdot.org
[link text itself]: http://www.reddit.com
```

## Footnotes

The default footnote-style links that Markdown uses do not display on the page. Sometimes it is useful to include a non-hyperlink footnote that will be visible to the reader. GitBook supports a simple syntax for such footnotes.

```
Text prior to footnote reference.[^2]
[^2] Comment to include in footnote.
```

## Images

```
Here's our logo (hover to see the title text):

Inline-style:
![alt text](https://github.com/adam-p/markdown-here/raw/master/src/common/images/icon48.png "Logo Title Text 1")

Reference-style:
![alt text][logo]

[logo]: https://github.com/adam-p/markdown-here/raw/master/src/common/images/icon48.png "Logo Title Text 2"
```

## Code and Syntax Highlighting

Code blocks are part of the Markdown spec, but syntax highlighting isn't. However, many renderers -- like Github's and *Markdown Here* -- support syntax highlighting. Which languages are supported and how those language names should be written will vary from renderer to renderer. *Markdown Here* supports highlighting for dozens of languages (and not-really-languages, like diffs and HTTP headers); to see the complete list, and how to write the language names, see the highlight.js demo page.

```
Inline `code` has `back-ticks around` it.
```

Blocks of code are either fenced by lines with three back-ticks ```, or are indented with four spaces. I recommend only using the fenced code blocks -- they're easier and only they support syntax highlighting.

````
```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>.
```
````

## Tables

Tables aren't part of the core Markdown spec, but they are part of GFM and *Markdown Here* supports them. They are an easy way of adding tables to your email -- a task that would otherwise require copy-pasting from another application.

```
Colons can be used to align columns.

| Tables        | Are           | Cool  |
| ------------- |:-------------:| -----:|
| col 3 is      | right-aligned | $1600 |
| col 2 is      | centered      |   $12 |
| zebra stripes | are neat      |    $1 |

The outer pipes (|) are optional, and you don't need to make the raw Markdown line up prettily. You can also use inline

Markdown | Less | Pretty
--- | --- | ---
*Still* | `renders` | **nicely**
1 | 2 | 3
```

## Blockquotes

```
> Blockquotes are very handy in email to emulate reply text.
> This line is part of the same quote.

Quote break.

> This is a very long line that will still be quoted properly when it wraps. Oh boy let's keep writing to make sure thi
```

## Inline HTML

You can also use raw HTML in your Markdown, and it'll mostly work pretty well.

```
<dl>
  <dt>Definition list</dt>
  <dd>Is something people use sometimes.</dd>

  <dt>Markdown in HTML</dt>
```

```
  <dd>Does *not* work **very** well. Use HTML <em>tags</em>.</dd>
</dl>
```

# Horizontal Rule

```
Three or more...

---

Hyphens

***

Asterisks

___

Underscores
```

# Line Breaks

My basic recommendation for learning how line breaks work is to experiment and discover -- hit <Enter> once (i.e., insert one newline), then hit it twice (i.e., insert two newlines), see what happens. You'll soon learn to get what you want. "Markdown Toggle" is your friend.

Here are some things to try out:

```
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 paragraph*.

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 paragraph*.
```

# Youtube videos

They can't be added directly but you can add an image with a link to the video like this:

```
<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>
```

Or, in pure Markdown, but losing the image sizing and border:

```
[![IMAGE ALT TEXT HERE](http://img.youtube.com/vi/YOUTUBE_VIDEO_ID_HERE/0.jpg)](http://www.youtube.com/watch?v=YOUTUBE_
```

# AsciiDoc

Since version `2.0.0`, GitBook can also accept AsciiDoc as an input format.

Please refer to the AsciiDoc Syntax Quick Reference for more informations about the format.

Just like for markdown, GitBook is using some special files to extract structures: `README.adoc`, `SUMMARY.adoc`, `LANGS.adoc` and `GLOSSARY.adoc`.

## README.adoc

This is the main entry of your book: the introduction. This file is **non optional**.

## SUMMARY.adoc

This file defines the list of chapters and subchapters. Just like for markdown, the `SUMMARY.adoc`'s format is simply a list of links, the name of the link is used as the chapter's name, and the target is a path to that chapter's file.

Subchapters are defined simply by adding a nested list to a parent chapter.

```
= Summary

. link:chapter-1/README.adoc[Chapter 1]
.. link:chapter-1/ARTICLE1.adoc[Article 1]
.. link:chapter-1/ARTICLE2.adoc[Article 2]
... link:chapter-1/ARTICLE-1-2-1.adoc[Article 1.2.1]
. link:chapter-2/README.adoc[Chapter 2]
. link:chapter-3/README.adoc[Chapter 3]
. link:chapter-4/README.adoc[Chapter 4]
.. Unfinished article
. Unfinished Chapter
```

## LANGS.adoc

For Multi-Languages books, this file is used to define the different supported languages and translations.

This file is following the same syntax as the `SUMMARY.adoc`:

```
= Languages

. link:en/[English]
. link:fr/[French]
```

## GLOSSARY.adoc

This file is sued to define terms. See the glossary section.

```
= Glossary

== Magic
Sufficiently advanced technology, beyond the understanding of the observer producing a sense of wonder.

== PHP
An atrocious language, invented for the sole purpose of inflicting pain and suffering amongst the proframming wizards c
```

# Cover

To make your book more elegant on GitBook, you can specify a cover.

A cover is specified by a **cover.jpg** file, a **cover_small.jpg** can also exist as a smaller version of the cover. The cover should be a **JPEG** file.

## Best Sizes

| | Big | Small |
|---|---|---|
| **File** | `cover.jpg` | `cover_small.jpg` |
| **Size(in pixels)** | 1800x2360 | 200x262 |

## Guidelines

A good cover respects the following guidelines:

- No border
- Clearly visible book title
- Any important text should be visible in the small version

## Autocover

A GitBook plugin ( `autocover` ) can also be used to generate a cover file for you, or just generate the `cover_small.jpg` from your big cover. This plugin is added by default on hosted books.

Read more about autocover.

# Multi-Languages

GitBook supports building books written in multiple languages. Each language should be a sub-directory following the normal GitBook format, and a file named `LANGS.md` should be present at the root of the repository with the following format:

```
* [English](en/)
* [French](fr/)
* [Español](es/)
```

You can see a complete example with the Learn Git book.

# Glossary

Allows you to specify terms and their respective definitions to be displayed in the glossary. Based on those terms, gitbook will automatically build an index and highlight those terms in pages.

The `GLOSSARY.md` format is very simple :

```
# term
Definition for this term

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

# Templating

This is an overview of the templating features available in GitBook. GitBook uses the Nunjucks and Jinga2 syntax.

## Variables

A variable looks up a value from the book context.

Variables are defined in the `book.json` file:

```
{
    "variables": {
        "myVariable": "Hello World"
    }
}
```

## Display Variables

```
{{ book.myVariable }}
```

This looks up `myVariable` from the book variables and displays it. Variable names can have dots in them which lookup properties. You can also use the square bracket syntax.

```
{{ book.foo.bar }}
{{ book["bar"] }}
```

If a value is undefined, nothing is displayed. The following all output nothing if `foo` is undefined: `{{ foo }}`, `{{ foo.bar }}`, `{{ foo.bar.baz }}`.

## Context variables

Some variables are also available to get informations about the current file or the GitBook instance:

| Name | Description |
| --- | --- |
| `file.path` | Path of the file relative to the book |
| `file.mtime` | Last modified date of the file |

## Tags

Tags are special blocks that perform operations on sections of the template.

## If

**if** tests a condition and lets you selectively display content. It behaves exactly as a programming language's if behaves.

```
{% if variable %}
  It is true
{% endif %}
```

If `variable` is defined and evaluates to true, "It is true" will be displayed. Otherwise, nothing will be.

You can specify alternate conditions with elif and else:

```
{% if hungry %}
  I am hungry
{% elif tired %}
  I am tired
{% else %}
  I am good!
{% endif %}
```

## for

**for** iterates over arrays and dictionaries.

Let's consider your variables in the `book.json`:

```
{
    "variables": {
        "authors": [
            { "name": "Samy" },
            { "name": "Aaron" }
        ]
    }
}
```

```
# Authors


{% for author in authors %}
  - {{ author.name }}
{% endfor %}
```

The above example lists all the authors using the `name` attribute of each item in the `authors` array as the display value.

## include

Include is detailed in the Content References article.

# Content References

Content referencing (conref) is a convenient mechanism for reuse of content from other files or books.

## Importing local files

Importing an other file's content is really easy using the `include` tag:

```
{% include "./test.md" %}
```

## Importing file from another book

GitBook can also resolve the include path by using git:

```
{% include "git+https://github.com/GitbookIO/documentation.git/README.md#0.0.1" %}
```

The format of git url is:

```
git+https://user@hostname/project/blah.git/file#commit-ish
```

The real git url part should finish with `.git`, the filename to import is extracted after the `.git` till the fragment of the url.

The `commit-ish` can be any tag, sha, or branch which can be supplied as an argument to `git checkout`. The default is `master`.

## Inheritance

Template inheritance is a way to make it easy to reuse templates. When writing a template, you can define "blocks" that child templates can override. The inheritance chain can be as long as you like.

`block` defines a section on the template and identifies it with a name. Base templates can specify blocks and child templates can override them with new content.

```
{% extends "./mypage.md" %}

{% block pageContent %}
# This is my page content
{% endblock %}
```

In the file `mypage.md`, you should specify the blocks that can be extent:

```
{% block pageContent %}
This is the default content
{% endblock %}

# License

{% import "./LICENSE" %}
```

# Ignoring files & folders

GitBook will read the `.gitignore`, `.bookignore` and `.ignore` files to get a list of files and folders to skip.

The format inside those files, follows the same convention as `.gitignore`:

```
# This is a comment

# Ignore the file test.md
test.md

# Ignore everything in the directory "bin"
bin/*
```

# Configuration

All configurations are stored as JSON in a file named `book.json`.

You can paste your `book.json` at jsonlint.com to validate the JSON syntax.

All fields are optionals or default to some extracted values.

# Fields

## gitbook

```
{ "gitbook": ">=2.0.0" }
```

This option is used to detect which version of GitBook will be use to generate the book. The format is a SEMVER condition.

## title

```
{ "title": "My Awesome Book" }
```

This option defines the title of your book, by default this value is extracted from the **README** (first title).

On **gitbook.com**, this value is defined from the title entered on the platform.

## description

```
{ "description": "This is such a great book!" }
```

This option defines the description of your book, by default this value is extracted from the **README** (first paragraph).

On **gitbook.com**, this value is defined from the description entered on the platform.

## isbn

```
{ "isbn": "978-3-16-148410-0" }
```

This option defines the ISBN associated with your book

## language

```
{ "language": "fr" }
```

This option defines the language of your book, by default value is `en`.

This option is used for internationalization and localization, it changes the text from the website.

On **gitbook.com**, this value is defined from the language detected in the content or specified in the settings.

## direction

```
{ "direction": "rtl" }
```

This option is used to override the text direction from the language. It is recommended to set the `language` field to a language with the correct text direction instead.

## styles

This option is used to add custom css to your book.

Example:

```
{
    "styles": {
        "website": "styles/website.css",
        "ebook": "styles/ebook.css",
        "pdf": "styles/pdf.css",
        "mobi": "styles/mobi.css",
        "epub": "styles/epub.css"
    }
}
```

## plugins

```
{ "plugins": ["mathjax"] }
```

The list of plugins being used by a book is defined in the `book.json` configuration.

## pluginsConfig

```
{
    "plugins": ["myplugin"],
    "pluginsConfig": {
        "myPlugin": {
            "message": "Hello World"
        }
    }
}
```

This option contains all plugins specific configurations.

## structure

This option is used to override files paths used by GitBook.

For example if you want to use `INTRO.md` instead of `README.md`:

```
{
    "structure": {
        "readme": "INTRO.md"
    }
```

```
}
```

Structure types are `readme`, `langs`, `summary` and `glossary`.

## variables

```
{
    "variables": {
        "myTest": "Hello World"
    }
}
```

This option defines the variables values being used in templating.

# Build

After you pushed content using **git** or the **editor**, GitBook will start different builds:

- **website**: it will generate the website
- **json**: il will extract metadata about the book (summary, introduction, etc)
- **epub**: it will generate the epub download
- **pdf**: it will generate the pdf download

## List builds

The **History** tab on your book let you follow the evolution of your builds.

## Details for a build

When clicking the button associated with a build, you can access a detailed page for it. This page will let you see the output of the build process.

## Fixing errors

If your build failed, you can use the logs to debug the issue and publish a fixed content.

Read more about common build errors

# Update your book using GIT

When your book is created on **gitbook.com**, you need to push some content to it. To do so, you can use the web editor or the command line.

If you want to update your book from the command line, you can use GIT to push your content:

## GIT Url

Each book is associated with a Git HTTPS url. The ssh protocol is not yet supported on the GitBook's git server.

The format for the git url is:

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

## Authentication

The git server is using your basic GitBook login to authenticate you. When prompted enter your GitBook username and your password (you can also use your API token).

## Create a new repository on the command line

```
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
```

## Push an existing repository

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

# Common Errors

Here is a list of common build errors and their associated fixes.

---

```
Error loading plugins: plugin1, ...
```

This error happens because GitBook can't resolve a plugin (or the plugin is invalid). External plugins need to be installed using `gitbook install`.

It's also possible that some plugins can't be loaded because they need another version of GitBook. In this case, you need to find the correct version of your plugin that support the GitBook version you are using and define it in your `book.json`:

```
{
    "plugins": ["myPlugin@1.2.3"]
}
```

---

```
Need to install ebook-convert from Calibre
```

To get around the error while trying to build your project as a PDF, ePub or mobi ebook, you must have the Calibre eBook reader/manager installed *AND* the command-line tools installed.

To install the Calibre command-line tools from the Mac version, from the menu select: **calibre - Preferences - Miscellaneous - Install command line tools**

Once this is completed, your ebook builds will be successful.

---

# GitHub Integration

GitBook integrates perfectly with GitHub as a host for your book's source.

## Connect your account / Permissions

Before integrating your book with GitHub, you need to authorize GitBook to get access to your GitHub account.

In your account settings, connect your GitHub account with the correct permissions:

- **Default permissions**: only access your GitHub account to authenticate you during login
- **Access to webhook**: access your GitHub account to create a webhook on your specified repository (See webhooks).
- **Access to public repositories**: access your GitHub repository from the webeditor so that you can edit your book easily from GitBook (only public repositories)
- **Access to private repositories**: same as above but with access to private repositories

## Webhooks

Webhooks notify GitBook when your GitHub repository changes.

If your GitHub changes are not available on GitBook, the main source of the problem is the webhook. You can check the state of your webhook in your repository's settings.

## Link a book and a GitHub repository

When your GitHub account is correctly linked to your GitBook account, linking a book to a repository is easy.

**Caution:** When you specify a GitHub repository in your book's settings, it will take priority over GitBook's git repository, this means that the editor will directly edit content on GitHub.

**The sync is unidirectional**, only changes made on GitHub will trigger builds on GitBook, GitBook **will not** update your GitHub repository with any content written before.

1. Open the GitHub section in your book settings
2. Enter your repository id (such as: `YourGitHubUserName/RepoName` )
3. Save your settings
4. Click on the button **Add a deployment webhook**

You can now edit your GitHub repository from the web editor (if you have authorized the correct permissions), and your commit on GitHub will trigger builds on GitBook

## Common Errors:

#### My Book is not being updated / I can't see any builds

If you linked your GitHub repository to a GitBook but editing content doesn't trigger any build. Verify that the webhook has been correctly added to your GitHub repository (in your GitHub repository settings -> Webhook). If the webhook is not present or invalid, add it back from your book's settings.

#### Change of book owner

If you transferred your book to a new owner, the webhook is now invalid, you need to add it back.

# Transferring content to GitHub

If you started writing your book on GitBook and you now want to host its source on GitHub, don't worry, it's easy:

1. Use the **Import Tool** from GitHub: import.github.com/new
2. Enter your GitBook git url, for example: `https://git.gitbook.com/MyName/MeBook.git}` (this url can be found in your book settings).
3. Enter a name for your GitHub repository.
4. Enter your GitBook credentials (you can use your API token instead of your password) when prompted.
5. Done!

When your content as been transferred to GitHub, you can now setup the integration so taht GitBook can still build your book from GitHub: GitHub Integration

# Styling

- Book Homepage

# Homepage Theme

Themes can be used to customize your book's homepage on **gitbook.com** using predefined or custom HTML template.

## Predefined themes

GitBook predefined themes are open source and available on GitHub.

## Variables for book homepage

| Variable | Type | Description |
| --- | --- | --- |
| `book` | `<book>` | Book to display |

## Book Representation

| Variable | Type | Description |
| --- | --- | --- |
| `id` | `string` | Complete id of the book (ex: `Username/Test`) |
| `name` | `string` | Name of the book |
| `title` | `string` | Title of the book |
| `description` | `string` | Description of the book |
| `public` | `boolean` | False if the book is private |
| `price` | `number` | Price of the book (0 equals free) |
| `githubId` | `string` | ID of the book on GitHub (ex: `Username/Test`) |
| `categories` | `array[string]` | List of tags associated with this book |
| `author` | `<user>` | User that created the book |
| `collaborators` | `array[<user>]` | Array of collaborators of the book (excluding the author) |
| `cover.small` | `string` | Url of the cover (size small) |
| `cover.large` | `string` | Url of the cover (size large) |
| `urls.access` | `string` | Url to access the book homepage |
| `urls.homepage` | `string` | Url of the homepage (using custom domain) |
| `urls.read` | `string` | Url to read the book |
| `urls.reviews` | `string` | Url to the reviews page |
| `urls.subscribe` | `string` | Url to submit email subscriptions |
| `urls.download.epub` | `string` | Url to download ebook (as EPUB) |
| `urls.download.mobi` | `string` | Url to download ebook (as Mobi) |
| `urls.download.pdf` | `string` | Url to download ebook (as PDF) |

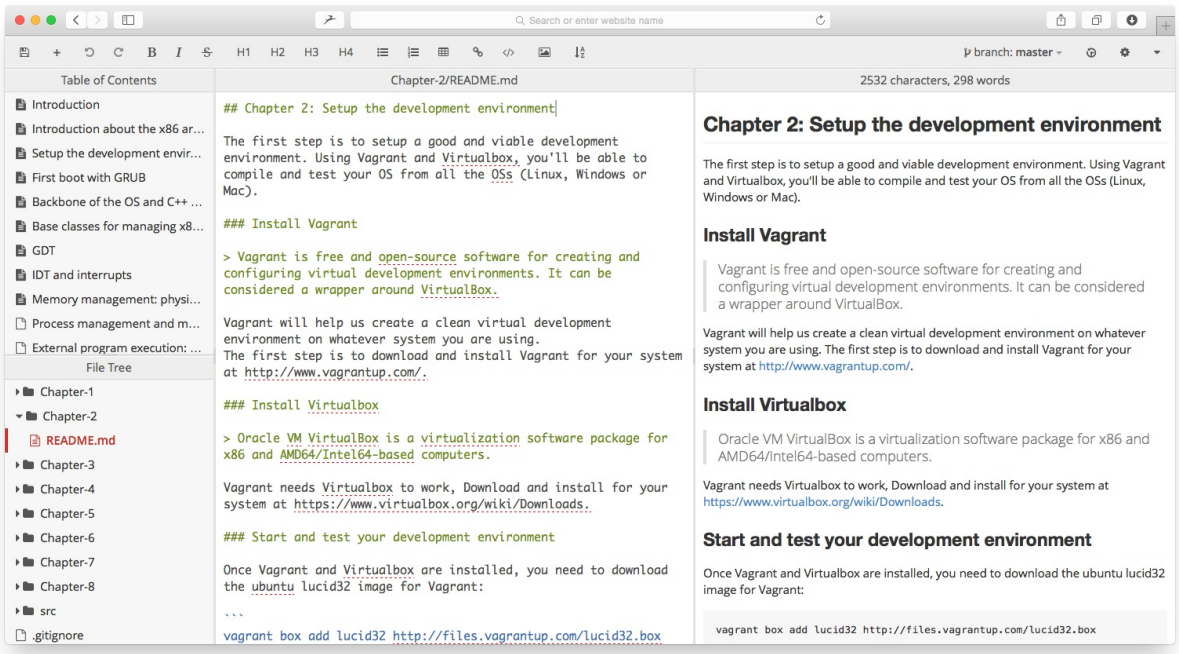
## User Representation

| Variable | Type | Description |
| --- | --- | --- |
| `username` | `string` | Username of the user |

| | | |
|---|---|---|
| `name` | `string` | Name of the user |
| `urls.profile` | `string` | Url to access the profile homepage |
| `urls.avatar` | `string` | Url of avatar |
| `accounts.twitter` | `string` | Username on Twitter (if linked) |
| `accounts.github` | `string` | Username on GitHub (if linked) |

# Editor

An editor is available online to edit your books. A new build will be started each time you save a file. Refer to the section "Draft Workflow" if you want to write a draft of the book without build it.




## How to access the editor?

The editor is accessible on each of your books. From the book's dashboard, click on the "edit" icon, it will open a new tab with the editor.

The editor is compatible with all modern web browsers: Google Chrome, Safari, Firefox and Internet Explore; check that you're using the latest version of your web browser.

## Draft Workflow

The GitBook Editor will trigger a new build each time you save a file (or when you edit the glossary or the summary).

But using the correct workflow, it's possible to work on a draft of your book then build it once it's finished.

1. Create a new branch from the branches menu
    i. Enter a name that describe your modification, for example: "firstdraft"
    ii. Select "master" as the origin branch
2. Your active branch should now be the branch that you just created
3. Edit your book like usual
4. When your draft is finish, open the branches menu and click on "Merge Branches"
5. Merge your draft branch into the master branch
6. Delete your old draft branch
7. Done!

# Chapters

You can add chapters from the editor simply by **right clicking** on a chapter and selecting `Add section`. Chapter and subchapters can be reorganized by dragging and dropping.

# Searching GitBook

You can use our powerful search tools to find what you're looking for among thousands of books on GitBook.

When searching GitBook, you can construct queries that match specific numbers and words.

## Search by text

By default, GitBook search books associated with the keywords from the query. For example `javascript angular` will return all the books that contain the word "javascript" and "angular"

## Exclude results containing a certain word

You can also narrow your search results by excluding words with the `NOT` syntax. Searching for `Hello` returns a massive number of "Hello World" projects, but changing your search to include `hello NOT world` returns fewer results.

## Query for specific field values

You can filter books by including only books with a specific field value. For example: license:apache-2 returns the list of books with the Apache 2 license.

## Query for values less/greater than another value

You can use `>` or `>=` to indicate "greater than" and "greater than or equal to," respectively. For example, the following search queries are equivalent: `cats stars:">10"` and `cats stars:">= 11"`

You can use `<` and `<=` to indicate "less than" and "less than or equal to," respectively.

# Audit Logs

GitBook keeps logs of audited user, book, and system events. You can use these logs to check your account security as well as to comply with internal security mandates and external regulations.

## Audited actions

You can search the audit log for a wide variety of actions.

| Name | Description |
| --- | --- |
| `user.login` | Login of an user |
| `user.failed_login` | An user/visitor failed to logged in |
| `user.signup` | Signup of an user |
| `user.remove` | User removed his account |
| `user.email.change` | User changed his email |
| `user.token.change` | User reset his api token |
| `user.password.change` | User changed his password |
| `user.forgot_password` | User reset his password |
| `user.verification.send` | User requested a verification email |
| `user.verification.done` | User verified his email |
| `user.forgot_password` | User reset his password |
| `org.create` | User created an organization |
| `org.remove` | User removed an organization |
| `org.transform` | User transformed a personal account into an organization |
| `staff.login.fake` | A site admin signed into an user account |
| `payment.card.change` | Connect a credit card |
| `payment.card.remove` | Remove a credit card |
| `payment.plan.change` | Changed his plan |
| `payment.transfer` | Transfer money to an user's bank account |
| `payment.recipient.change` | Connect a bank/paypal account |
| `payment.recipient.remove` | Remove a bank/paypal account |
| `book.create` | Creation of a book by an user |
| `book.remove` | Remove a book |
| `book.transfer` | Transfer a book |
| `book.publish` | Publish a new version of the book |

# Taxes

Keep in mind the following is not specific tax advice. You should always report in accordance with applicable tax laws. We will send any required tax forms to you, but if you have additional questions on how to file or that are specific to your tax situation, we highly recommend consulting a tax professional in good standing with the IRS.

## What informations should I provide to GitBook?

Fill out all the fields in Payout settings.

## What tax documents should I expect from GitBook?

It depends on several factors, but here's a quick breakdown:

- Authors who sold at least 200 books and earn at least $20,000 in royalties earnings in the last year will receive a Form **1099-K**.
- Authors who earn at least $600 in earnings from royalties in the last year will receive a Form **1099-MISC**.
- Authors who do not meet either of the above criteria in the last year will not receive a tax form.

## How are the amounts on my 1099s calculated?

We know this can get a bit confusing, so let's break it down by form:

- **1099-K**: If you receive a 1099-K from us, the dollar amount you see in Box 1 consists of all earnings collected in the previous year (after we deduct our admin fee).
- **1099-MISC**: If you receive a 1099-MISC, the dollar amount in Box 3 is all income from GitBook royalties.

Remember: It's possible you won't receive either of these forms, so check out the above criteria.

## When will I be receiving my 1099?

We send out 1099s via USPS by the end of January. You will also receive this form by email in January. (Note: If you don't receive a 1099, it means you didn't meet the above thresholds and therefore won't be receiving one.)

## What if I didn't receive any 1099 forms?

In some cases, you may not receive a form at all. There are situations where authors won't receive a form, depending on amount earned, payout methods used, and the tax information submitted. If you didn't but were expecting one, please send us an email and we'll get it sorted out.

## Do I need GitBook's EIN for my tax return?

Since you are not an employee of GitBook, you are not required to provide GitBook's EIN (Employer Identification Number) on your tax return.

# Visibility

## Public/Private

Your book can be **public** or **private**. Public books are visible to everybody but only collaborators can update it. Private books are visible only by the collaborators.

You can switch your book from public to private and the inverse.

## Paid books

Paid books can only be **public**.

## Home/Explore pages

Home and Explore pages only list books that are already successfully built. It is advised to set a cover picture.

# Custom Domains

All books on **Gitbook.com** are accessible via the url **http://.gitbooks.io/{book}/**, and content is accessible at **http://.gitbooks.io/{book}/content/**.

But you can configure your book to use a custom domain name (a free feature on GitBook). Domain name can be use for the homepage and the content (or both).

The process to add a custom domain to your book is easy.

1. Add your domain name in your book settings.

In order to use your own domain, you will need to make a few changes with your domain registrar:

1. Log in to your domain registrar and find the section that allows you to add/edit host records, often found in a settings menu called 'Edit DNS', 'Host Records' or 'Zone File Control'.
2. Set the www record to a CNAME and set the URL field to: `www.gitbook.com`.
3. To redirect the naked domain ( `yourdomain.com` ) to `www.yourdomain.com`, find the option to forward your domain. This can usually be found under 'Forwarding', 'URL Forwarding' or 'URL Redirect'.

It may take a few hours for domain changes to propagate.

# Mailing

GitBook integrates an easy messaging and newsletter feature to help author communicate efficiently with readers.

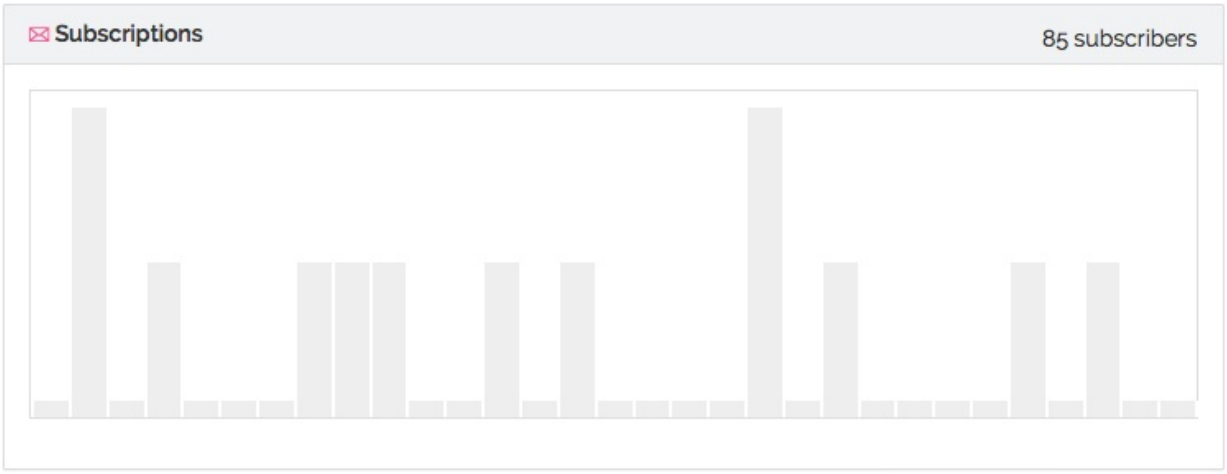
## Newsletter

Readers and visitors can subscribe to a mailing list to get notifications about content updates (and even custom messages from authors).

The newsletter form can be integrated easily into a book homepage (already present in the default homepage theme).

## Mailing Dashboard

From your book dashboard, you can access the mailing dashboard, this page let you see stats about your newsletter subscribers and create messages.




## Create a new message

There is two types of messages: **Notifications** and **Newsletter messages**.

When creating a new message from your mailing dashboard, you can choose the type of message.

Messages are sent in your name, so that readers can respond to it to contact you. But you can configure the **reply-to** email and set a **bcc** email.

## Newsletter message

Email notifications to subscribers will not sent immediately. All messages are sent at once, periodically (midnight PST). The message is sent to all subscriber to the book newsletter.

## Notification / Message to new buyer

You can only create one message of this type by book. It will be sent to every new reader of your **paid book**.

This message can be used to send a thank you note and start a mail discussion with the reader.

# Organizations

You can create a new organization by either setting up a new organization or converting an existing personal account into an organization.

Organizations are great for businesses and large projects that need multiple owners and administrators. For more information on how organizations can help you collaborate on a project, see our article on the difference between user accounts and organizations.

There are several ways to create an organization.

## Create a new organization

When you create a new organization from scratch, it doesn't have any books associated with it. At any time, members of the organization with write permissions can add new books, or transfer existing books.

## Convert your existing personal account into an organization

If you want all of the repositories in your current account to be part of the organization, then you can convert the account to an organization.

# What's the difference between user and organization accounts?

Your user account is your identity on GitBook. Your user account can be a member of any number of organizations, regardless of whether the account is on a free or paid plan.

## Personal user accounts

Every person who uses GitBook has their own user account. These accounts include:

- Unlimited public books and collaborators on all plans
- Personal plans for private books and distribution Ability to add unlimited repository collaborators

## Organizations

Organizations are great for businesses and large projects that need multiple owners and admins. They include:

- Business plans for private books and distribution
- Collaborators-based access permissions
- Unlimited administrators and collaborators using teams
- Billing receipts that can be sent to a second email address

# Converting a user into an organization

If you need to give more granular permissions for accessing books in a personal account, you can convert the personal account to an organization.

**Warning**: Before converting a user into an organization, keep these points in mind:

- You will no longer be able to sign into the converted user account.
- An organization cannot be converted back to a user.

## 1. Create a personal account

You cannot access books that belong to an organization unless you are one of its members. As a result, you'll need to create a second user account that will let you access the organization after you convert.

## 2. Leave any organizations you're already a member of

If the user you're converting is already a member of other organizations, you must first leave the other organizations.

## 3. Convert the account into an organization

- Open your account settings.
- Under "Transform account", enter the username of your new personnal account (see section 1.), click `Turn into an organization`.

## 4. Sign in to the organization

If you added your secondary personal account as the new owner in the last step of the conversion process, sign in to that account, and in the account context switcher, you'll be able to access your new organization!

# Transferring organization ownership

To make someone else the owner of an organization account, you must add a new owner, ensure that the billing information is updated, and then remove yourself from the account.

> Removing yourself from the organization does not update the billing information on file for the organization account. The new owner must update the billing information on file to remove your credit card or PayPal information.

1. If you're the only member of the Owners team, add another organization member to the team.
2. Contact the new owner and make sure he or she is able to access the organization's settings.
3. If you are currently responsible for paying for GitBook in your organization, you'll also need to change the organization's billing information to reflect the new owner.
4. Remove yourself from the organization.

# Plugins

Plugins are the best way to extend GitBook functionnalities (ebook and website). There exist plugins to do a lot of things: bring math formulas display support, track visits using Google Analytic, ...

## How to find plugins?

Plugins can be easily searched on plugins.gitbook.com.

## How to install a plugin?

Once you find a plugin that you want to install, you need to add it to your `book.json`:

```
{
    "plugins": ["myPlugin", "anotherPlugin"]
}
```

You can also specify a specific version using: `"myPlugin@0.3.1"`, this is usefull when you're using an outdated version of GitBook.

If you're building your book locally, download an prepare plugins simply by running: `gitbook install`.

# Create and publish a plugin

A GitBook plugin is a node package published on NPM that follow a defined convention.

## Structure

## package.json

The `package.json` contains general informations about your plugin (name, version, description, ...):

```
{
    "name": "gitbook-plugin-mytest",
    "version": "0.0.1",
    "description": "This is my first GitBook plugin",
    "engines": {
        "gitbook": ">1.x.x"
    }
}
```

You can learn more about `package.json` from the NPM documentation.

The **package name** must begin with `gitbook-plugin-` and the **package engines** should contains `gitbook`.

## index.js

The `index.js` is main entry point of your plugin.

## Publish your plugin

GitBook plugins are published and installed from NPM.

To publish a new plugin, you need to create an account on npmjs.com then publish it from the command line:

```
$ npm publish
```

# Extend blocks

Extending templating blocks is the best way to provide extra functionnalities to authors.

The most common usage is to process the content within some tags at runtime. It's like filters, but on steroids because you aren't confined to a single expression.

## Defining a new block

Blocks are defined by the plugin, blocks is a map of name associated with a block descriptor. The block descriptor needs to contain at least a `process` method.

```
module.exports = {
    blocks: {
        tag1: {
            process: function(block) {
                return "Hello "+block.body+", How are you?";
            }
        }
    }
};
```

The `process` should return the html content that will replace the tag.

## Handling block arguments

Arguments can be passed to blocks:

```
{% tag1 "argument 1", "argument 2", name="Test" %}
This is the body of the block.
{% endtag1 %}
```

And arguments are easily accessible in the `process` method:

```
module.exports = {
    blocks: {
        tag1: {
            process: function(block) {
                // block.args equals ["argument 1", "argument 2"]
                // block.kwargs equals { "name": "Test" }
            }
        }
    }
};
```

## Handling sub-blocks

A defined block can be parsed into different sub-blocks, for example let's consider the source:

```
{% myTag %}
    Main body
    {% subblock1 %}
    Body of sub-block 1
    {% subblock 2 %}
    Body of sub-block 1
```

```
  {% endmyTag %}
```

# API

> Warning: The GitBook API is not release and is meant to change.

## Schema

All API access is over HTTPS, and accessed through `www.gitbook.com/api/` . All data is sent and received as **JSON**.

## Authentication

There is currently only one way to authenticate through GitBook API. Requests that require authentication will return `404 Not Found`, instead of `403 Forbidden`, in some places. This is to prevent the accidental leakage of private books to unauthorized users.

### Basic Authentication

```
$ curl -u "username:token" https://www.gitbook.com/api/books/
```

## Error Format

Error are returned as JSON:

```
{
    "error": "Not found",
    "code": 404
}
```

## Pagination

### Parameters

| Name | Type | Description |
| --- | --- | --- |
| skip | int | Number of items to skip |
| limit | int | Number of items to list |

### Return

```
{
    list: [],
    skip: 0,
    limit: 100,
    total: 0
}
```

# Books

> Warning: The GitBook API is not release and is meant to change.

## List your books

List books for the authenticated user. This includes books from organizations the user can access.

```
GET /books/
```

You can include only books created by the authenticated user:

```
GET /books/author
```

## List all public books

This provides a dump of every public repository, in the order that they were created.

```
GET /books/all
```

# OPDS

The Open Publication Distribution System (**OPDS**) is an application of the Atom Syndication Format intended to enable content creators and distributors to distribute digital books via a simple catalog format. This format is designed to work interchangeably across multiple desktop and device software programs.

GitBook is providing an OPDS catalog at: `https://www.gitbook.com/api/opds/catalog.atom`