GitBook

Documentation

Published
with GitBook

Table of Contents

1. Introduction
2. Books
    i. Format
        i. Chapters and Subchapters
        ii. Markdown
        iii. Cover
        iv. Exercises and Quizzes
        v. Variables and Import
    ii. Update with GIT
    iii. Build
    iv. Common Errors
    v. Themes
    vi. Visibility
    vii. Custom Domains
3. Platform
    i. Editor
    ii. Search
    iii. Audit Logs
    iv. Taxes
    v. Organizations
        i. Differences with users
        ii. Convert an user
        iii. Transferring ownership

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

- Introduction
- Books
  - Format
    - Chapters and Subchapters
    - Markdown
    - Cover
    - Exercises and Quizzes
    - Variables and Import
  - Update with GIT
  - Build
  - Common Errors
  - Themes
  - Visibility
  - Custom Domains
- Platform
  - Editor
  - Search
  - Audit Logs
  - Taxes
  - Organizations
    - Differences with users
    - Convert an user
    - Transferring ownership

Format

The format's main focus is simplicity and readability.

GitBook uses a convention on top of markdown files.

A book is a Git repository containing at least 2 files: README.md and SUMMARY.md.

README.md

Typically, this should be the introduction for your book. It will be automatically added to the final summary.

SUMMARY.md

The SUMMARY.md defines your book's structure. It should contain a list of chapters and sub-chapters, linking to their respective pages.

Example:

```
# Summary

This is the summary of my book.

* [section 1](section1/README.md)
    * [example 1](section1/example1.md)
    * [example 2](section1/example2.md)
* [section 2](section2/README.md)
    * [example 1](section2/example1.md)
```

Files that are not included in SUMMARY.md will not be processed by GitBook.

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 ...
```

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`)

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)
```

Adding chapters from the editor

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

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
```

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_
```

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

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.

Guidelines

A good cover respects the following guidelines:

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

Exercises and Quizzes

Exercises and quizzes can be activated using the **plugins** configuration in your `book.json`. Example of book.json:

```
{
    "plugins": [ "exercises", "quizzes" ]
}
```

Exercises

A book can contain interactive exercises (currently only in Javascript). An exercise is a code challenge provided to the reader, which is given a code editor to write a solution which is checked against the book author's validation code.

An exercise is defined by 4 simple parts:

- Exercise **Message**/Goals (in markdown/text)
- **Initial** code to show to the user, providing a starting point
- **Solution** code, being a correct solution to the exercise
- **Validation** code that tests the correctness of the user's input

Exercises needs to start and finish with a separation bar (`---` or `***`). It should contain 3 code elements (**base**, **solution** and **validation**). It can contain a 4th element that provides **context** code (functions, imports of libraries etc ... that shouldn't be displayed to the user).

````
---

Define a variable `x` equal to 10.

```js
var x =
```

```js
var x = 10;
```

```js
assert(x == 10);
```

```js
// This is context code available everywhere
// The user will be able to call magicFunc in his code
function magicFunc() {
    return 3;
}
```

---
````

Quizzes

A book can also contain interactive quizzes.

A Quiz is defined in the same way as an exercise.

```
---

Here is the introduction for the quiz

This is Question 1:
- [x] This is the proposition 1 (the correct one)
```

```
- [ ] This is the proposition 2

> This is a help message when the answer to question 1 is wrong

This is Question 2:
- [ ] This is the proposition 1
- [x] This is the proposition 2 (correct)
- [x] This is the proposition 3 (correct)

> This is a help message when the answer to question 2 is wrong

---
```

Variables and Import

Some macros can be used in the markdown source to import files or variables.

Define variables

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

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

Import variables

Variables can be imported in all markdown files using `{{ }}`.

For example to show the variable from the book.json (`myVariable`):

```
# This is a test: {{ myVariable }}
```

Import files

In the same way, files can be imported:

```
# This is a test:

{{ ./file.md }}
```

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
```

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

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`.

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

This error was logged as Convert of PDF - Need to install ebook-convert from Calibre #333 on GitHub.

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.

Themes

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

Predefined themes

GitBook predefined themes are open source and available on GitHub.

Format

Themes on GitBook use the SWIG syntax.

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)

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.io** are accessible via the url **http://{{author}}.gitbooks.io/{book}/**, and content is accessible at **http://{{author}}.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.

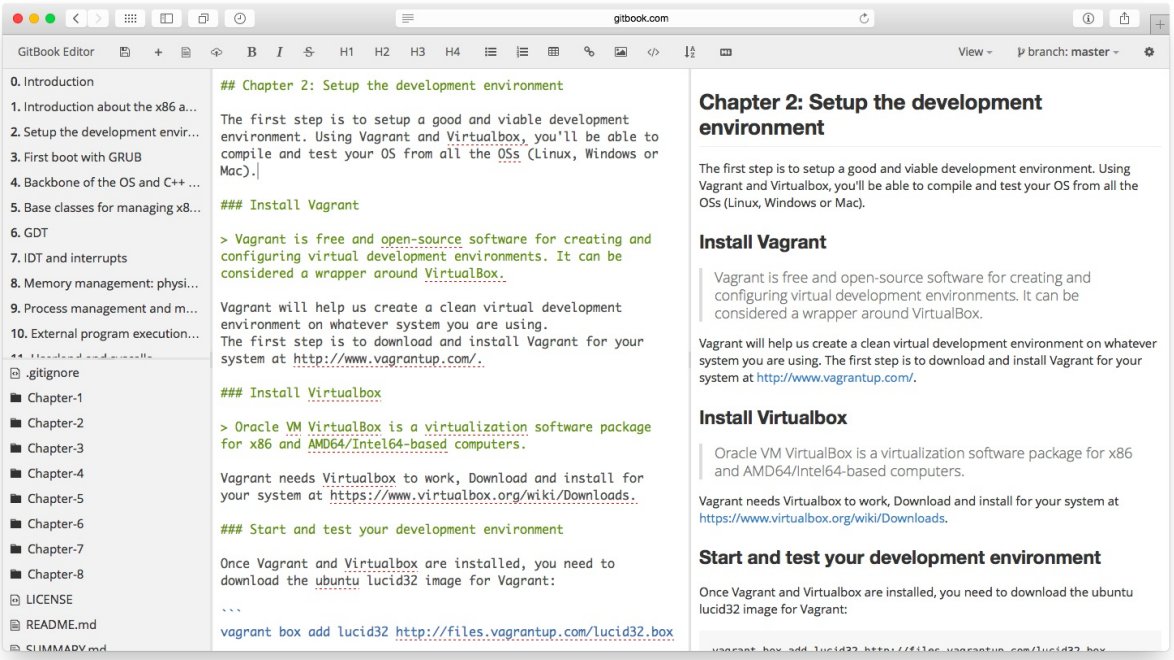
In order to use your own domain, you will need to make a few changes with your domain registrar:

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.

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!

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 faield 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 information 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.

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.