Software

Nguyễn Quản Bá Hồng[1]

August 14, 2022

[1]Independent Researcher, Ben Tre City, Vietnam
e-mail: nguyenquanbahong@gmail.com; website: https://nqbh.github.io.

Contents

Chapter 1

Git

1.1 GitHub/Repositories/Working with Files/Managing Large Files

"You can manage large files with Git Large File Storage."

1.1.1 GitHub/repositories/working with files/managing large files/about large files on GitHub

"GitHub limits the size of files you can track in regular Git repositories. Learn how to track or remove files that are beyond the limit."

1.1.1.1 About size limits on GitHub

"GitHub tries to provide abundant storage for all Git repositories, although there are hard limits for file & repository sizes. To ensure performance & reliability for our users, we actively monitor signals of overall repository health. *Repository health* is a function of various interacting factors, including size, commit frequency, contents, & structure.

1.1.1.1.1 File size limits. "GitHub limits the size of files allowed in repositories. If you attempt to add or update a file that is > 50 MB, you will receive a warning from Git. The changes will still successfully push to your repository, but you can consider removign the commit to minimize performance impact.

Remark 1.1.1. *If you add a file to a repository via a browser, the file can be no* > 25 *MB. For more information, see GitHub/repositories/working with files/managing files/adding a file to a repository.*

GitHub blocks pushes that exceed 100 MB. To track files beyond this limit, you must use Git Large File Storage (Git LFS). For more information, see GitHub/repositories/working with files/managing large files/about GitHub Large File Storage. If you need to distribute large files within your repository, you can create releases on GitHub.com instead of tracking the files. For more information, see GitHub/repositories/working with files/managing large files/about GitHub Large File Storage/distributing large binaries. Git is not designed to handle large SQL file. To share large databases with other developers, we recommend using Dropbox."

1.1.1.1.2 Repository size limits. "We recommend repositories remain small, ideally < 1 GB, & < 5 is strongly recommended. Smaller repositories are faster to clone & easier to work with & maintain. If your repository excessively impacts our infrastructure, you might receive an email from GitHub Support asking you to take corrective action. We try to be flexible, especially with large projects that have many collaborators, & will work with you to find a resolution whenever possible. You can prevent your repository from impacting our infrastructure by effectively managing your repository's size & overall health. You can find advice & a tool for repository analysis in the github/git-sizer repository.

External dependencies can cause Git repositories to become very large. To avoid filling a repository with external dependencies, we recommend you use a package manager. Popular package managers for common languages include Bundler, Node's Package Manager, & Maven. These package managers support using Git repositories directly, so you don't need pre-packaged sources. Git is not designed to serve as a backup tool. However, there are many solutions specifically designed for performing backups, e.g., Arq, Carbonite, & CrashPlan."

1.1.1.2 Removing files from a repository's history

Warning 1.1.1. *"These procedures will permanently remove files from the repository on your computer & GitHub.com. If the file is important, make a local backup copy in a directory outside of the repository.*

1.1.1.2.1 Removing a file added in the most recent unpushed commit. "If the file was added with your most recent commit, & you have not pushed to GitHub.com, you can delete the file & amend the commit:

1. Open Terminal.
2. Change the current working directory to your local repository.
3. To remove the file, enter `git rm --cached`:

   ```
   $ git rm --cached giant_file
   # Stage our giant file for removal, but leave it on disk
   ```

4. Commit this change using `--amend --CHEAD`:

   ```
   $ git commit --amend -CHEAD
   # Amend the previous commit with your change
   # Simply making a new commit won't work, as you need
   # to remove the file from the unpushed history as well
   ```

5. Push your commits to GitHub.com:

   ```
   $ git push
   # Push our rewritten, smaller commit
   ```

1.1.1.2.2 Removing a file that was added in an earlier commit. If you added a file in an earlier commit, you need to remove it from the repository's history. To remove files from the repository's history, you can use the BFG Repo-Cleaner or the `git filter-branch` command. For more information see GitHub/authenticating to GitHub/removing sensitive data from a repository."

1.1.1.3 Distributing large binaries

"If you need to distribute large files within your repository, you can create releases on GitHub.com. Releases allow you to package software, release notes, & links to binary files, for other people to use. For more information, visit GitHub/administering a repository/about releases. We don't limit the total size of the binary files in the release or the bandwidth used to deliver them. However, each individual file must be < 2 GB."

Chapter 2

Markdown

2.1 Learn X in Y Minutes/Markdown

“Get the code: markdown.md. Markdown was created by John Gruber in 2004. It’s meant to be an easy to read & write syntax which converts easily to HTML (& now many other formats as well). Markdown also varies in implementation from 1 parser to a next. This guide will attempt to clarify when features are universal or when they are specific to a certain parser.”

2.1.1 HTML Elements

“Markdown is a superset of HTML, so any HTML file is valid Markdown.

```
<!--This means we can use HTML elements in Markdown, such as the comment element, & they won’t be
affected by a markdown parser. However, if you create an HTML element in your markdown file, you cannot
use markdown syntax within that element’s contents.-->’’
```

2.1.2 Headings

“You can create HTML elements `<h1>` through `<h6>` easily by pretending the text you want to be in that element by a number of hashes (`#`).

```
# This is an <h1>
## This is an <h2>
### This is an <h3>
#### This is an <h4>
##### This is an <h5>
###### This is an <h6>
```

Markdown also provides us with 2 alternative ways of indicating h1 & h2.

```
This is an h1
=============

This is an h2
-------------’’
```

2.1.3 Simple Text Styles

“Text can be easily styled as italic or bold using markdown.

```
*This text is in italics.*
_And so is this text._

**This text is in bold.**
__And so is this text.__

***This text is in both.***
**_As is this!_**
*__And this!__*
```

In GitHub Flavored Markdown, which is used to render markdown files on GitHub, we also have strikethrough:

```
~~This text is rendered with strikethrough.~~''
```

2.1.4 Paragraphs

"Paragraphs are a one or multiple adjacent lines of text separated by one or multiple blank lines.

```
This is a paragraph. I'm typing in a paragraph isn't this fun?

Now I'm in paragraph 2.
I'm still in paragraph 2 too!


I'm in paragraph three!
```

Should you ever want to insert an HTML `<br />` tag, you can end a paragraph with 2 or more spaces & then begin a new paragraph.

```
I end with two spaces (highlight me to see them).

There's a <br /> above me!
```

Block quotes are easy & done with the `>` character.

```
> This is a block quote. You can either
> manually wrap your lines \& put a '>' before every line or you can let your lines get really long & wrap on
> their own.
> It doesn't make a difference so long as they start with a '>'.

> You can also use more than one level
>> of indentation?
> How neat is that?''
```

2.1.5 Lists

"Unordered lists can be made using asterisks, pluses, or hyphens.

```
* Item
* Item
* Another item

or

+ Item
+ Item
+ One more item

or

- Item
- Item
- One last item
```

Ordered lists are done with a number followed by a period.

```
1. Item 1
2. Item 2
3. Item 3
```

You don't even have to label the items correctly & Markdown will still render the numbers in order, but this may not be a good idea.

```
1. Item 1
1. Item 2
1. Item 3
```

(This renders the same as the above example)
You can also use sublists

```
1. Item one
2. Item two
3. Item three
    * Sub-item
    * Sub-item
4. Item four
```

There are even task lists. This creates HTML checkboxes.

```
Boxes below without the 'x' are unchecked HTML checkboxes.
- [ ] 1st task to complete.
- [ ] 2nd task that needs done
This checkbox below will be a checked HTML checkbox.
- [x] This task has been completed''
```

2.1.6 Code Blocks

"You can indicate a code block (which uses the `<code>` element) by indenting a line with 4 spaces or a tab.

```
    This is code
    So is this
```

You can also re-tab (or add an additional 4 spaces) for indentation inside your code

```
    my_array.each do |item|
        puts item
    end
```

Inline code can be created using the backtick character `

```
John didn't even know what the `go_to()` function did!
```

In GitHub Flavored Markdown, you can use a special syntax for code

~~~
```ruby
def foobar
    puts "Hello world!"
end
```
~~~

The above text doesn't require indenting, plus GitHub will use syntax highlighting of the language you specify after the ```."

2.1.7 Horizontal Rule

"Horizontal rules (`<hr/>`) are easily added with 3 or more asterisks or hyphens, with or without spaces.

```
***
---
- - -
****************''
```

2.1.8 Links

"1 of the best things about markdown is how easy it is to make links. Put the text to display in hard brackets `[]` followed by the URL in parentheses `()`

```
[Click me!](http://test.com/)
```

You can also add a link title using quotes inside the parentheses.

```
[Click me!](http://test.com/ "Link to Test.com")
```

Relative paths work too.

```
[Go to music](/music/).
```

Markdown also supports reference style links.

```
[Click this link][link1] for more info about it!
[Also check out this link][foobar] if you want to.

[link1]: http://test.com/ "Cool!"
[foobar]: http://foobar.biz/ "Alright!"
```

The title can also be in single quotes or in parentheses, or omitted entirely. The references can be anywhere in your document & the reference IDs can be anything so long as they are unique.

There is also "implicit naming" which lets you use the link text as the id.

```
[This][] is a link.

[this]: http://thisisalink.com/
```

But it's not that commonly used."

2.1.8.1 Table of contents

"Some Markdown flavors even make use of the combination of lists, links & headings in order to create table of contents. In this case, heading titles in lowercase are prepended with hash (`#`) & are used as link ids. Should the heading have multiple words, they will be connected with a hyphen (`-`), that also replaces some special characters. (Some other special characters are omitted though.)

```
- [Heading](#heading)
- [Another heading](#another-heading)
- [Chapter](#chapter)
  - [Subchapter <h3 />](#subchapter-h3-)
```

Nonetheless, this is a feature that might not be working in all Markdown implementations the same way."

2.1.9 Images

"Images are done the same way as links but with an exclamation point in front!

```
![This is the alt-attribute for my image](http://imgur.com/myimage.jpg "An optional title")
```

And reference style works as expected.

```
![This is the alt-attribute.][myimage]

[myimage]: relative/urls/cool/image.jpg "if you need a title, it's here"''
```

2.1.10 Miscellany

2.1.10.1 Auto-links

```
<http://testwebsite.com/> is equivalent to
[http://testwebsite.com/](http://testwebsite.com/)
```

2.1.10.2 Auto-links for emails

```
<foo@bar.com>
```

2.1.10.3 Escaping characters

```
I want to type *this text surrounded by asterisks* but I don't want it to be
in italics, so I do this: \*this text surrounded by asterisks\*.
```

2.1.10.4 Keyboard keys

“In GitHub Flavored Markdown, you can use a ¡kbd¿ tag to represent keyboard keys.

```
Your computer crashed? Try sending a
<kbd>Ctrl</kbd>+<kbd>Alt</kbd>+<kbd>Del</kbd>''
```

2.1.10.5 Tables

“Tables are only available in GitHub Flavored Markdown & are slightly cumbersome, but if you really want it:

```
| Col1         | Col2     | Col3          |
| :----------- | :------: | ------------: |
| Left-aligned | Centered | Right-aligned |
| blah         | blah     | blah          |
```

or, for the same results

```
Col 1 | Col2 | Col3
:-- | :-: | --:
Ugh this is so ugly | make it | stop
```

2.1.11 Markdownlint

“In order to simplify work with Markdown & to unify its coding style, Markdownlint has been created. This tool is available also as a plugin for some IDEs & can be used as an utility to ensure validity & readability of Markdown.”

2.1.12 Further Reading

“For more info, check out John Gruber’s official post of syntax & Adam Pritchard’s great cheatsheet. If you want to learn more on some major Markdown flavors’ features, see:

- GitHub flavored Markdown
- GitLab flavored Markdown

Originally contributed by Dan Turkel, & updated by 13 contributor(s).”