
Pandoc for PDF How-To test

How-to generate nice PDF documentation from
Markdown

Alexey Gumirov

N₀

22 December 2018

Contents

1	Pandoc for PDF How-To	3
1.1	How-to for docs preparation	3
1.1.1	Tools	3
1.1.2	Instructions and commands	3
1.1.3	Important notes about Markdown file formatting for PDF processing	7
1.2	Examples	8
1.2.1	This page example	8
1.3	Building CI pipeline in the Gitlab	9
1.3.1	Folders structure	9

1 Pandoc for PDF How-To

How-To, templates and commands to produce PDF documents from Markdown files.

1.1 How-to for docs preparation

1.1.1 Tools

- **pandoc**
 - template: I use my template which is a slightly modified eisvogel.latex¹ template. The only change I did is I put `subtitle` in the footer instead of `author`.
 - Both templates you can find in the repository of this project. Original template eisvogel.latex² and my modified eisvogel_mod.latex³
- **texlive**
- **convert**
 - converts and formats images.
 - it is used here for the change of DPI of the images and convert to PNG.
 - **convert** is the utility which is part of the **ImageMagick** package.

I did not install **convert** tool, it seems like it is installed by default in Ubuntu or comes with **texlive**. To avoid possible issues with **pdflatex** engine I did full installation of **texlive** packet.

In Debian family (with **apt**):

```
1 sudo apt-get update
2 sudo apt-get install pandoc
3 sudo apt-get install texlive-full
4 sudo apt-get install imagemagick
```

Because of the size of the **texlive** packet (~4,5GB) I do generation of the PDF locally.

1.1.2 Instructions and commands

YAML Block for LaTeX template

¹<https://github.com/Wandmalfarbe/pandoc-latex-template>

²pandoc/templates/eisvogel.latex

³pandoc/templates/eisvogel_mod.latex

This YAML block in the beginning of the MarkDown file defines parameters used by the Pandoc engine and relevant LaTeX template parameters. This particular example below instructs Pandoc to produce PDF file with the Cover page (**titlepage: true**) and change color of the line on the cover page. Another important parameter is **logo** - it defines path to file with the logo you want to put on the cover page.

```
1 title: "Pandoc for PDF How-To"
2 author: "Alexey Gumirov"
3 date: "17 December 2018"
4 subtitle: "How-to generate nice PDF documentation from Markdown"
5 titlepage: true
6 titlepage-color: "FFFFFF"
7 titlepage-text-color: "000000"
8 titlepage-rule-color: "CCCCCC"
9 titlepage-rule-height: 4
10 toc-own-page: true
11 logo: "files/logo.png"
12 logo-width: 100
13 links-as-notes: true
14 lof: false
15 lot: false
```

Parameter **links-as-notes** enables putting of the URL links in the footnotes of the page.

Parameters **lof** and **lot** are responsible for the creation of *list of figures* and *list of tables* respectively.

Because MarkDown for GitHub does not support YAML header in the main file, I set it up in the separate `_yaml-block.yaml` file in the root folder of the project.

Images preparation

In my setup I print with 300 DPI (this produces high resolution PDF). Therefore all images must be 300 DPI.

If you have images with different DPI (especially GIF files), then use the following commands:

To re-sample image to 300 DPI:

```
1 convert $SOURCE_IMG_FILE -units PixelsPerInch \
2 -resample 300 $TARGET_IMG_FILE.png
```

After rasampling image has to be brought to the proper size. Command resizes picture to be 1700 pixels horizontally and sets DPI meta-data to 300.

```
1 convert $SOURCE_IMG_FILE -units PixelsPerInch \
2 -resize 1700x -density 300 $TARGET_IMG_FILE.png
```

But if you are not afraid, then all can be done in one command:

```
1 convert $SOURCE_IMG_FILE -set units PixelsPerInch \  
2 -resample 300 -resize 1700x -density 300 $TARGET_IMG_FILE.png
```

It is important to mention that the order of options does matter. The instruction above makes steps in the following order:

1. `-set units PixelsPerInch`: Sets density units in Pixels per Inch instead of default `PixelsperCantimeter`.
2. `-resample 300`: Changes resolution of the image from its current DPI (PPI) to 300 DPI (PPI). It is not just change of meta-data, this parameter makes **convert** to re-process image.
3. `-resize 1700x`: Resizes picture to the following dimintions: `width = 1700 pixels`, `height = auto`.
4. `-density 300`: This parameter sets DPI meta-data in the target picture to 300 DPI (PPI)

Pandoc command

```
1 pandoc -s -S -o $DEST.pdf -f markdown_github+yaml_metadata_block \  
2 --template eisvogel_mod --toc --dpi=300 \  
3 -V lang=en-US _yaml-block.yaml $SOURCE.md
```

If you want to put current date in the cover page automatically, then you can add following parameter in the **pandoc** command line: `-M date="date "+%d %B %Y""`. Or you can define date in the script variable `DATE=$(date "+%d %B %Y")` and then use this variable in the `-M` option: `-M date="$DATE"`.

Then **pandoc** command will look like that:

```
1 DATE=$(date "+%d %B %Y")  
2 pandoc -s -S -o $DEST.pdf -f markdown_github+yaml_metadata_block \  
3 --template eisvogel_mod --toc --dpi=300 -M date="$DATE" \  
4 -V lang=en-US _yaml-block.yaml $SOURCE.md
```

Options of the **pandoc** command mean following:

- `-s`: Standalone document.
- `-S`: `--smart`
 - Produce typographically correct output, converting straight quotes to curly quotes, — to em-dashes, – to en-dashes, and ... to ellipses. Nonbreaking spaces are inserted after certain abbreviations, such as “Mr.” (Note: This option is selected automatically when the output

format is latex or context, unless `--no-tex-ligatures` is used. It has no effect for latex input.)

* In newer versions of **pandoc** this switch was removed and you shall use `+smart` extension in the `-f` option.

- `-f FORMAT` or `-r FORMAT`:
 - Specify input format. `FORMAT` can be **native** (native Haskell), `json` (JSON version of native AST), `markdown` (pandoc's extended Markdown), `markdown_strict` (original unextended Markdown), `markdown_phpextra` (PHP Markdown Extra), `markdown_github` (GitHub-Flavored Markdown), `commonmark` (CommonMark Markdown), `textile` (Textile), `rst` (reStructuredText), `html` (HTML), `docbook` (DocBook), `t2t` (txt2tags), `docx` (docx), `odt` (ODT), `epub` (EPUB), `opml` (OPML), `org` (Emacs Org mode), `mediawiki` (MediaWiki markup), `twiki` (TWiki markup), `haddock` (Haddock markup), or `latex` (LaTeX). If `+lhs` is appended to `markdown`, `rst`, `latex`, or `html`, the input will be treated as literate Haskell source. Markdown syntax extensions can be individually enabled or disabled by appending `+EXTENSION` or `-EXTENSION` to the format name. So, for example, `markdown_strict+footnotes+definition_lists` is strict Markdown with footnotes and definition lists enabled, and `markdown-pipe_tables+hard_line_breaks` is pandoc's Markdown without pipe tables and with hard line breaks.
 - Therefore if `-S` is not working then option `-f` shall be used with `+smart` extension. E.g. for this particular document the option with parameters will look like this: `-f markdown_github+yaml_metadata_block+smart`.
- `--template FILE`: Use `FILE` as a custom template for the generated document. Implies `--standalone`.
- `--toc: --table-of-contents`
 - Include an automatically generated table of contents (or, in the case of latex, context, docx, and rst, an instruction to create one) in the output document. This option has no effect on man, docbook, docbook5, slidy, slideous, s5, or odt output.
- `--dpi`:
 - Specify the dpi (dots per inch) value for conversion from pixels to inch/centimeters and vice versa. The **default** is **96dpi**. Technically, the correct term would be ppi (pixels per inch).
- `-V KEY[=VAL]: --variable=KEY[:VAL]`
 - Set the template variable `KEY` to the value `VAL` when rendering the document in standalone mode. This is generally only useful when the `--template` option is used to specify a custom

- template, since pandoc automatically sets the variables used in the default templates. If no `VAL` is specified, the key will be given the value `true`.
- `lang`: one of the `KEY` parameters of `-V` which defines default document language.

Additional useful options of the **pandoc** command are:

- `--listings`: It creates nice presentation of the raw code (like shell code or programming code).
- `--number-section`: Automatically creates enumerated headers.
- `--default-image-extension`: If you want Pandoc to insert only one type of images, e.g. PNG, then you shall add `--default-image-extension png` in the command line.

Conversion of multiple files

When you create large amount of content, it is not convenient to use one large Markdown file for it. Then it is better to split it in multiple Markdown files and organize them in a separate folder using names with leading sequence numbers, like here:

- Create folder, e.g. **“content”**.
- Put there Markdown files which you want to combine into one PDF.
- Name files with numbers in the order they shall be concatenated into one PDF. Example:

```
1 > ~/ $ ls -lh content/
2 total 197K
3 -rwxrwxrwx 1 root root 0 Dec 18 18:49 00-Intro.md
4 -rwxrwxrwx 1 root root 0 Dec 18 18:47 01-Chapter_A.md
5 -rwxrwxrwx 1 root root 0 Dec 18 18:47 02-Chapter_B.md
6 -rwxrwxrwx 1 root root 0 Dec 18 18:49 03-Chapter_C.md
7 -rwxrwxrwx 1 root root 0 Dec 18 18:50 99-Appendix.md
```

- Apply following Pandoc command:

```
1 pandoc -s -S -o $DEST.pdf -f markdown_github+yaml_metadata_block \
2     --template eisvogel_mod --toc --dpi=300 -V lang=en-US \
3     _yaml-block.yaml content/*.md
```

This command will take all Markdown files from the **“content”** folder and convert them into enumerated order into a single PDF file.

1.1.3 Important notes about Markdown file formatting for PDF processing

Unordered Lists and sub-lists indentation

It is stated in the GitHub⁴ site that correct indent for the unordered lists is 2 spaces. But with this indent Pandoc does not identify sub-lists.

Therefore, please use 4 spaces indent for the sub-lists in the unordered lists. Then they will be properly reflected in the PDF files.

While using of standard tab (4 spaces) indent is not a mistake, some programs (in my case it is MS Visual Studio Code) can give you a warning. You can just ignore it.

Links

If your Markdown file has to be processed into the PDF, then please pay attention to the format of links you use:

a) Link format that does NOT WORK: ![Name of the resource](Link).

b) Link format that WORKS: [Name of the resource](Link).

The problem is that by the Markdown guidelines⁵ using exclamation mark before URL is not appropriate. Exclamation mark is used for links to images only. But GitHub engine does not give you an error, it just treats such links as links which opens in the new tab or window in the browser.

Therefore, to avoid compilation errors in the **pdflatex** engine (which is used by **pandoc**), please use (b) type of URL formatting, which is compliant with Markdown standard.

1.2 Examples

1.2.1 This page example

This page pandoc-2-pdf-how-to.pdf. Generated with the following command (in the project directory):

```
1 DATE=$(date "+%d %B %Y")
2 pandoc -s -S -o pandoc-2-pdf-how-to.pdf
3     -f markdown_github+yaml_metadata_block \
4     --template eisvogel_mod --toc --listings --number-section\
5     --dpi=300 -M date="$DATE" \
6     -V lang=en-US _yaml-block.md README.md
```

The link to _yaml-block.yaml file is here⁶.

⁴<https://github.com/DavidAnson/markdownlint/blob/v0.11.0/doc/Rules.md#md007>

⁵<https://github.com/DavidAnson/markdownlint/blob/v0.11.0/doc/Rules.md#md007>

⁶_yaml-block.yaml

1.3 Building CI pipeline in the Gitlab

I made my CI pipeline for GitLab which automatically creates PDF and stores it in the Gitlab artifactory when the content of MarkDown or YAML files is changed.

1.3.1 Folders structure

Create following folders structure:

```
1 > $ tree -a
2 ./
3 -- content/
4     -- 01-Introduction.md
5     -- 02-Chapter_A.md
6     -- 03-Chapter_B.md
7     -- {...}.md
8     -- img/
9         -- img_01.png
10        -- img_02.png
11        -- img_03.png
12 -- logo/
13     -- dt-logo.png
14 -- pandoc/
15     -- templates/
16         -- eisvogel.latex
17         -- eisvogel_mod.latex
18 -- .gitlab-ci.yml
19 -- _yaml-block.yaml
20 -- README.md
```

- In `logo` folder I put `dt-logo.png` file.
- In the `content` folder I create `img` folder where I put all images/pictures I use in the content MarkDown files.
- In the `pandoc/templates` folder I keep pandoc templates I use for PDF creation.

To create PDF I use `knsit/pandoc` Docker container. This container has newer version of the **pandoc** therefore instead of `-S` option I use `+smart` extension in the `-f` option.

The `.gitlab-ci.yml` has the following content:

```
1 image: knsit/pandoc
2
3 my_nice_pdf:
```

```
4  variables:
5    SOURCE_DIR: "content"
6    YAML_FILE: "_yaml-block.yaml"
7    DEST_FILE_NAME: "my_nice_document"
8    TEMPLATE: "eisvogel_mod"
9    SOURCE_FORMAT: "markdown_github+yaml_metadata_block+smart"
10 script:
11   - DATE=$(date +%Y-%m-%d)
12   - DEST_FILE_NAME_DATE=$DEST_FILE_NAME$DATE
13   - DATE=$(date "+%d %B %Y")
14   - pandoc --version
15   - mkdir -p ~/.pandoc/templates/
16   - cp pandoc/templates/$TEMPLATE.latex ~/.pandoc/templates
17   - pandoc -s -o $DEST_FILE_NAME_DATE.pdf -f $SOURCE_FORMAT \
18     --template $TEMPLATE -M date="$DATE" \
19     --listings --number-section --toc --dpi=300 -V lang=en-US \
20     $YAML_FILE $SOURCE_DIR/*.md >&1
21   - mkdir -p my_nice_pdf
22   - mv $DEST_FILE_NAME_DATE.pdf my_nice_pdf/
23 stage: build
24 artifacts:
25   paths:
26   - my_nice_pdf
27   expire_in: 6 month
28 only:
29   changes:
30   - *.yaml
31   - content/*.md
```

Parameter `changes` makes CI job run only when content of the YAML block or any of Markdown files in the `content` folder is changed.