How to make PDF from MarkDown with Pandoc

Detailed manual for all

Alexey Gumirov

21 November 2019

Contents

List of Tables

List of Figures

1 How to make PDF from MarkDown with Pandoc

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.latex[1] template. I made following modifications:
    * `subtitle` field is used in the footer instead of `author`.
  - Both templates you can find in the repository of this project. Original template eisvogel.latex[2] and my modified eisvogel_mod.latex[3]
- **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**):

```
sudo apt-get update
sudo apt-get install pandoc
sudo apt-get install imagemagick
```

I use following `texlive` packages:

```
sudo apt-get install texlive-latex-recommended
sudo apt-get install texlive-fonts-recommended
sudo apt-get install texlive-latex-extra
sudo apt-get install texlive-fonts-extra
sudo apt-get install texlive-xetex
```

Extra LaTeX packages are needed for **eisvogel** template to work. I also install XeTeX because if you have text with some special symbols, XeTeX can process it properly.

[1] https://github.com/Wandmalfarbe/pandoc-latex-template
[2] pandoc/templates/eisvogel.latex
[3] pandoc/templates/eisvogel_mod.latex

1.1.2 Instructions and commands

YAML Block for LaTex template 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.

```
title: "How to make PDF from MarkDown with Pandoc"
author: "Alexey Gumirov"
date:
subtitle: "Detailed manual for all"
geometry: "left=2.54cm,right=2.54cm,top=1.91cm,bottom=1.91cm"
titlepage: true
titlepage-color: "FFFFFF"
titlepage-text-color: "000000"
titlepage-rule-color: "CCCCCC"
titlepage-rule-height: 4
logo: "files/logo.png"
logo-width: 100
links-as-notes: true
lot: true
lof: true
listings-disable-line-numbers: true
```

> Table of content, list of tables and list of figures are going in the following order: ToC, LoT and LoF. After LoF is always a page break.

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.

Parameter **listings-disable-line-numbers** disables line numbers for all listings.

Because MarkDown for GitHub does not support YAML header in the main file, I set it up in the separate `HEADER.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:

```
convert $SOURCE_IMG_FILE -units PixelsPerInch \
    -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.

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

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

```
convert $SOURCE_IMG_FILE  -set units PixelsPerInch \
    -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 dimensions: `width = 1700 pixels`, `height = auto`.
4. `-density 300`: This parameter sets DPI meta-data in the target picture to 300 DPI (PPI)

Pandoc command

```
pandoc -s -S -o $DEST.pdf \
    -f markdown_github+yaml_metadata_block+implicit_figures+table_captions
       +footnotes \
    --template eisvogel_mod --listings --number-section -V subparagraph\
    --toc --dpi=300 -V lang=en-US \
    HEADER.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=$date(date "+%d %B %Y")` and then use this variable in the `-M` option: `-M date="$DATE"`.

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

```
DATE=$(date "+%d %B %Y")
pandoc -s -S -o $DEST.pdf \
    -f markdown_github+yaml_metadata_block+implicit_figures+table_captions
       +footnotes \
    --template eisvogel_mod --listings --number-section -V subparagraph\
    --toc --dpi=300 -V lang=en-US \
    --template eisvogel_mod --toc --dpi=300 -M date="$DATE" \
    -V lang=en-US HEADER.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.
  - `implicit_figures`: An image with nonempty alt text, occurring by itself in a paragraph, will be rendered as a figure with a caption. The image's alt text will be used as the caption. This extension is very useful when you need to autogenerate captions for figures in the markdown reference format like: `![This is the caption](/url/of/image.png)`
  - `table_captions`: A caption may optionally be provided for all 4 kinds of supported Markdown tables. A caption is a paragraph beginning with the string `Table:` (or just `:`), which will be stripped off. It may appear either before or after the table.
  - `footnotes`: Footnotes in the Pandoc Markdown format. For more details please go to Pandoc manual page[4].
  - 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

[4] https://pandoc.org/MANUAL.html#footnotes

like this: `-f markdown_github+yaml_metadata_block+implicit_figures+tables_captions+smart+footnotes`.

- `--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.
  - `subparagraph`: Is needed to start each chapter from the new page here. In the Eisvogel_mod.latex template necessary modifications are made.

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.

List of figures List of figures is automatically generated by the Pandoc during PDF file creation. For the list of figures and relevant captions is responsible `implicit_figures` extension. It does not require any additional text, it will convert [alt text] into the caption. E.g. for this image below:

```
![Aleph 0](files/logo.png)
```

Figure 1: Aleph 0

List of tables The `table_captions` extension requires `Table:` or `:` paragraph right before or below table. You do not need to numerate the table - Pandoc will make enumeration by itself, but you shall provide required paragraph text. E.g. for the table below the raw Markdown text is the following:

```
Table: Sample table

Name | value
:---|:---:
A | 1
B | 2
```

Table 1: Sample table

Name	value
A	1
B	2

Conversion of multiple files When you create large amount of content, it is not convinient 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 concatinated into one PDF. Example:

```
> ~/ $ ls -lh content/
total 197K
-rwxrwxrwx 1 root root   0 Dec 18 18:49 00-Intro.md
-rwxrwxrwx 1 root root   0 Dec 18 18:47 01-Chapter_A.md
-rwxrwxrwx 1 root root   0 Dec 18 18:47 02-Chapter_B.md
```

```
-rwxrwxrwx 1 root root   0 Dec 18 18:49 03-Chapter_C.md
-rwxrwxrwx 1 root root   0 Dec 18 18:50 99-Appendix.md
```

- Apply following Pandoc command:

```
pandoc -s -S -o $DEST.pdf \
    -f markdown_github+yaml_metadata_block+implicit_figures+table_captions
       +footnotes \
    --template eisvogel_mod --listings --number-section -V subparagraph \
    --toc --dpi=300 -V lang=en-US \
    HEADER.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.

The cons of this method is that you cannot include/exclude particular source MarkDown files to produce PDF with only content you need. Therefore for such setups I use `INDEX` file where I list all files which Pandoc shall convert into PDF in the order I want them to go.

```
> cat INDEX
HEADER.YAML
00-Intro.md
01-Chapter_A.md
03-Chapter_C.md
```

And then my PDF generation command looks the following:

```
pandoc -s -S -o $DEST.pdf \
    -f markdown_github+yaml_metadata_block+implicit_figures+table_captions
       +footnotes \
    --template eisvogel_mod --listings --number-section -V subparagraph \
    --toc --dpi=300 -V lang=en-US \
   $(cat INDEX)
```

1.1.3 Important notes about MarkDown file formatting for PDF processing

Unordered Lists and sub-lists indentation It is stated in the GitHub[5] 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.

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

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 guidelines[6] 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.

Pandoc execution folder In order for Pandoc correctly process all links and references (especilly links to images) you shall run pandoc script inside the directory with MarkDown files. Therefore, it is better to place `logo` folder, YAML meta-data file and PDF generating shell script directly into the directory with MarkDown files.

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):

```
DATE=$(date "+%d %B %Y")
pandoc -s -S -o pandoc-2-pdf-how-to.pdf
    -f markdown_github+yaml_metadata_block+implicit_figures+table_captions
       +footnotes \
    --template eisvogel_mod --listings --number-section -V subparagraph \
    --toc --dpi=300 -M date="$DATE" -V lang=en-US \
    $(cat INDEX)
```

The link to `HEADER.YAML`[7].

[6] https://github.com/DavidAnson/markdownlint/blob/v0.11.0/doc/Rules.md#md007
[7] HEADER.YAML

2 Automation of PDF creation

2.1 Local PC automation with *entr* and *task spooler*

On my local PC I use `entr` and `task spooler` (in Ubuntu it is called `tsp`).

- `entr`: The *Event Notify Test Runner* is a general purpose Unix utility intended to make rapid feedback and automated testing natural and completely ordinary. More details on the Entr project page[8].
- `task-spooler` or `tsp` or `ts` (depending on the system): A simple unix batch system. More details via `man tsp` or `man ts`.

To install `entr` and `task spooler` in Ubuntu, use these commands:

```
sudo apt-get update
sudo apt-get install entr
sudo apt-get install task-spooler
```

The following command creates task in the spooler queue which monitors state of the edited file (in this case `README.md`) and as soon as file is updated, script `_pdf-gen.sh` is launched. This script generates PDF. In this example both `README.md` and `_pdf-gen.sh` are located in the same directory, and command below is launched from the same directory.

```
> $ tsp bash -c 'ls README.md | entr -p ./_pdf-gen.sh'
```

When you need to monitor multiple MarkDown files in the e.g. `content` folder, you can use the following command:

```
> $ tsp bash -c 'ls content/*.md | entr -p ./_pdf-gen.sh'
```

2.2 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.

2.2.1 Folders structure

Create following folders structure:

[8] http://eradman.com/entrproject/

```
> $ tree -a
./
-- content/
    -- 01-Introduction.md
    -- 02-Chapter_A.md
    -- 03-Chapter_B.md
    -- {...}.md
    -- HEADER.YAML
    -- INDEX
    -- img/
        -- img_01.png
        -- img_02.png
        -- img_03.png
    -- logo/
        -- logo.png
-- pandoc/
    -- templates/
        -- eisvogel.latex
        -- eisvogel_mod.latex
-- .gitlab-ci.yml
-- README.md
```

Where `INDEX` file contains list of source files which shall be processed by Pandoc including `HEADER.YAML` file.

```
> $ cat INDEX
HEADER.YAML
01-Introduction.md
02-Chapter_A.md
03-Chapter_B.md
{...}.md
```

- In `logo` folder I put `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` optoin I use `+smart` extension in the `-f` option.

> **Note:** After update of Pandoc engine to version 2.7.2 PDF generation is broken. Therefore I use v2.7 of Pandoc engine.

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

```
image: knsit/pandoc:v2.7

my_nice_pdf:
  variables:
    SOURCE_DIR: "content"
    INDEX_FILE: "INDEX"
    DEST_FILE_NAME: "my_nice_document"
    TEMPLATE: "eisvogel_mod"
    SOURCE_FORMAT: "markdown_github+yaml_metadata_block+smart+
       implicit_figures+table_captions+footnotes"
  script:
    - DATE=$(date +_%Y-%m-%d)
    - DEST_FILE_NAME_DATE=$DEST_FILE_NAME$DATE
    - DATE=$(date "+%d %B %Y")
    - pandoc --version
    - mkdir -p ~/.pandoc/templates/
    - cp pandoc/templates/$TEMPLATE.latex ~/.pandoc/templates
    - mkdir -p my_nice_pdf
    - cd "$SOURCE_DIR"
    - pandoc -s -o $DEST_FILE_NAME_DATE.pdf -f $SOURCE_FORMAT \
        --template $TEMPLATE -M date="$DATE" \
        --listings --number-section -V subparagraph \
        --toc --dpi=300 -V lang=en-US \
        $(cat "$INDEX_FILE") >&1
    - mv $DEST_FILE_NAME_DATE.pdf "$CI_PROJECT_DIR"/my_nice_pdf/
  stage: build
  artifacts:
    paths:
    - my_nice_pdf
    expire_in: 6 month
  only:
    changes:
    - content/HEADER.YAML
    - content/*.md
    - content/INDEX
```

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