Table of contents

1 Markdown과 Pandoc 갖고 놀기 - PDF 2
1.1 Markdown 블로깅 . . . 2
1.2 Docker . . . 3
1.3 Pandoc - a universal document converter . . . 4
1.4 Tex Live . . . 4
1.5 준비물 . . . 4
1.6 테스트용 Markdown . . . 4
1.6.1 이것은 to-do 목록이다. . . . 4
1.6.2 테이블 . . . 4
1.6.3 이미지 . . . 4
1.6.4 ~~취소된 텍스트~~ . . . 5
1.6.5 Mermaid 다이어그램 . . . 6
1.7 PDF로 변환해 보자 1 . . . 6
1.8 PDF로 변환해 보자 2 . . . 7
1.8.1 나눔 폰트 설치하는 법 . . . 7
1.8.2 Pandoc에 한글 폰트 지정하기 . . . 7
1.8.3 다시 도전해 보자 . . . 7
1.9 PDF로 변환해 보자 3 . . . 8
1.9.1 페이퍼 사이즈와 여백 . . . 8
1.9.2 PDF 정보 . . . 9
1.9.3 Meta data . . . 9
1.9.4 Table이 가운데 정렬로 출력된다 . . . 10
1.9.5 링크 색상 . . . 10
1.9.6 Level 4, 5 header . . . 10
1.9.7 TOC를 넣고 싶다. . . . 10
1.9.8 내부 문서로의 링크를 개인 블로그에 올라갈 페이지 url로 변경해줘야 한다. . . . 11
1.9.9 너무 느린 변환 . . . 11
1.9.10 Mermaid diagram . . . 11
1.9.11 참조 . . . 11
1.9.12 .pdf.tex . . . 11
1.9.13 Pandoc을 다시 실행해보자 . . . 12
1.10 PDF로 변환해 보자 4 . . . 12
1.10.1 All-in-one Pandoc image . . . 12
1.10.2 빌드된 이미지로 다시 시도해보자 . . . 13
1.10.3 Mermaid-filter . . . 14
1.10.4 Root user와 chromium path 문제 해결 . . . 14
1.10.5 테마 변경 . . . 15
1.10.6 css 변경 . . . 15
1.10.7 내부 링크 처리 . . . 15
1.11 PDF로 변환해 보자 5 . . . 15
1.12 블로그에 링크를 걸어보자 . . . 17
1.13 마무리 . . . 17

1 Markdown과 Pandoc 갖고 놀기 - PDF

1.1 Markdown 블로깅

Markdown, Notable, 그리고 Hugo에서 다뤄진 것처럼, 이 글은 Notable에서 Markdown으로 작성 후, 이후 완전 자동으로 FSWatch + Git-Sync + FLock --> GitLab Notes repository --> build.sh --> GitLab 개인 블로그 repository --> Hugo에 의해 Static Site로 생성되어, 최종 목적지인 madforfamily.com에 등록되게 된다. 이 정도면, 당분간은 별 문제없이 사용할 수 있을 정도로 자동화시켰다고 본다.

그렇다면 이제는, Markdown의 활용을 조금 더 넓혀 볼 차례인 듯 하다. 그 동안 해 왔던 생각 중의 하나는, 하나의 문서를 가지고 여러 포맷으로 자동 변환할 수 있을까 라는 것이었다. Markdown으로 문서를 작성하기만 하면, 필요한 포맷들로 자동 변환이 되고 필요한 곳에 업로드까지 된다면 참 편할 것 같다. 그래서 삽질 좀 깊게, 예상보다 훨씬 더 깊게 해 봤다.

1.2 Docker

Docker가 필요한 시점이 왔다. 사용하는 os 버전과 상관없이 최신 버전의 어플리케이션을 사용하고 싶을 때, 삽질을 하면서 설치해야 하는 어플리케이션들이 있는데, 삭제해도 찌꺼기가 남는 게 신경 쓰일 때, 그 때가 딱 Docker를 써야 하는 시점이다. 물론 사용하려는 어플리케이션이 os에 대한 깊은 의존성이 있고 Docker가 해결해 주지 못한다면 힘들수도 있지만, 경험상 거의 가능했다. 그러므로 아직 설치하지 않았다면 지금이라도 하자. 우분투 사용자는 여기로 가면 되고, 그 이외라면 여기서 사용하는 os를 골라서 진행하면 된다.

1.3 Pandoc – a universal document converter

한 마디로 올인원 문서 변환기이다. 문서포맷간 양방향 혹은 단방향 변환을 해주는 데, 일단 GFM(GitHub-flavored Markdown)을 잘 지원해 주는 점이 중요하다. 그래야 Notable에서 작성한 Markdown 포맷의 문서가, Hugo 통해서 웹사이트에도, 변환된 문서에도 동일하게 보일 수 있을 테니까. 그 외에 개인적으로 관심가는 포맷이라면, pdf, docx, odt, 그리고 html slide show framework인 reveal.js이다. 일단은 pdf에 집중해 보자.

메뉴얼에서 pdf를 검색하다 보니 다음과 같은 구절을 만났다. 흠 xelatex라는 게 필요하다는 의미이다. 더 찾아보니 Tex Live라는 걸로 이어진다.

> When using LaTeX for bidirectional documents, only the xelatex engine is fully supported (use --pdf-engine=xelatex)

1.4 Tex Live

문서 생성 시스템인 Tex를 손 쉽게 사용할 수 있도록 만들어진 Package이다. 그냥 지금은 Pandoc이 PDF 생성할 때 필요하다는 것만 알면 될 것 같다.

1.5 준비물

Docker 기반으로 해보자. 조금 더 찾아보니, Alpine Linux 기반의 공식 이미지가 있기는 하다. 하지만 유니코드를 제대로 지원하는 xelatex 사용을 위해서 Tex-Live라는 Package가 필요하다.

docker-texlive-full 기반으로 만들어진 docker-pandoc을 발견했다. Dockerfile들을 들여다 보니, 조금만 만져주면 사용이 가능할 것으로 보였다. 그냥 만들어진 이미지를 실행 후 수정해서 사용할 지, 수정된 이미지를 만들어서 사용할 지는 나중에 생각해 봐야 겠다. 일단은, 어떻게서든지 첫 걸음을 딛는 게 중요하다. 특히나 맨 땅에 헤딩할 때는 말이다.

docker pull thomasweise/docker-pandoc 실행해서 이미지를 다운로드해두자. Tex-Live의 용량이 좀 된다. 그래서 미리 한번 다운로드 받아두는 게 좋다. 이것으로 설치는 끝이다. 왜냐? Docker니까.

1.6 테스트용 Markdown

변환해 볼 문서는 바로 지금 작성하는 이 문서이다. 좀 더 자세한 테스트를 위해 다음처럼 할 일 리스트, 테이블, 이미지, 취소된 텍스트와 Marmaid 다이어그램을 추가했다.

1.6.1 이것은 to-do 목록이다.

```
- No Japan
 - [X] 안 가요
 - [X] 안 사요
 - [X] 안 먹어요
- No Jahan
 - [X] 안 뽑아요
```

- No Japan
  - [X] 안 가요
  - [X] 안 사요
  - [X] 안 먹어요
- No Jahan
  - [X] 안 뽑아요

1.6.2 테이블

```
| col 1 | col 2 |
| --- | --- |
| val 1 | val 22222222222 |
```

col 1	col 2
val 1	val 22222222222

1.6.3 이미지

```
![내부 이미지 링크](../attachments/notetaking.jpg)
```

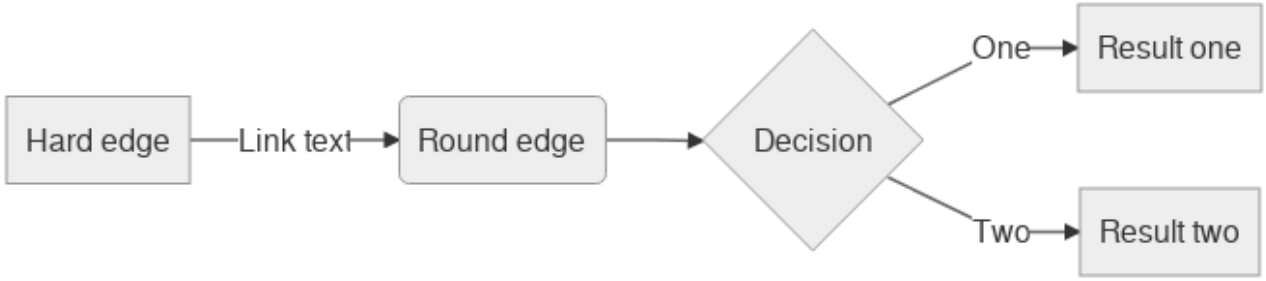
![외부 이미지 링크](https://madforfamily.com/attachments/notetaking.jpg)

1.6.4 ~~취소된 텍스트~~

~~이것 또한 취소되었다.~~

1.6.5 Mermaid 다이어그램

```
graph LR
  A[Hard edge] -->|Link text| B(Round edge)
  B --> C{Decision}
  C -->|One| D[Result one]
  C -->|Two| E[Result two]
```



1.7 PDF로 변환해 보자 1

```
docker run -it --rm ₩
  -v ~/Data/Notes:/Notes ₩
  -v ~/Downloads:/Downloads ₩
  thomasweise/docker-pandoc ₩
  /bin/bash -c ₩
    "cd /Notes/notes; ₩
     pandoc 'Markdown과 Pandoc 갖고 놀기 - PDF.md' ₩
       -f gfm ₩
       --pdf-engine=xelatex ₩
       -o /Downloads/converted_1.pdf"
```

위의 명령이 뭘 하는 건지 하나씩 살펴보면 다음과 같다.

- run: 지정된 Docker image thomasweise/docker-pandoc를 가지고 Docker container(일종의 독립된 실행 프로세스 영역)를 만들라는 뜻이다.
- -it: 그냥 container 안에서 주어진 명령이 실행되는 걸 보고 싶을 때 사용해야 하는 옵션이라고 생각하면 된다. 너무 깊이 들어가면 피곤하다.
- --rm: 실행 끝나면 생성된 container를 삭제하라는 뜻이다. 변환할 때만 필요하므로 쓰는 게 좋다. 안 그러면 매번 docker stop한 후 docker rm 해야 해서 귀찮다. 다만 삽질하는 중엔 이 옵션 없이 생성 후, 삽질이 끝나면 다시 넣는 식으로 하는 게 좋다.
- -v ~/Data/Notes:/Notes: 변환할 파일이 있는 로컬 pc상의 디렉토리인 ~/Data/Notes를 container안에다가 /Notes라는 디렉토리로 마운트하라는 의미이다.
- -v ~/Downloads:/Downloads: 바로 위와 동일하므로 더 이상의 설명은 생략한다. 변환된 pdf 파일을 저장할 디렉토리이다.
- thomasweise/docker-pandoc: 저 위에서 언급한 Docker image이다.
- cd /Notes/notes: 일단 container안에 마운트된 /Notes/notes로 이동한다. 그래야 ../attachments 디렉토리 내에 있는 파일들을 pandoc이 참조할 수 있다.
- pandoc ...: 그 중 지금 작성중인 이 문서를 xelatex pdf engine을 가지고, gfm 형식 문서인 Markdown과 Pandoc 갖고 놀기 - PDF.md를 pdf로 변환 후 /Downloads 디렉토리에 converted_1.pdf라는 이름으로 저장하라는 의미이다.

설명이 끝났다. 실행해 보자.

```
...
[WARNING] Missing character: There is no 아 (U+C544) in font [lmroman10-regular]:mapping=tex-text;!
...
```

한번에 되면 뭔가 불안해지는데, 다행히 첫 시도가 실패했다. 약간의 안도감이 느껴진다. 경고인지 에러인지 애매한 무수한 메시지가 출력되었다. 그런데 ~/Downloads를 보니 변환된 파일이 있긴 했다. 한번 들여다 보자.

Pandoc - a universal document converter

. , GFM(GitHub-flavored Markdown)
. Notable Markdown , Hugo , .
, pdf, docx, odt, html slide show framework reveal.js . pdf .

pdf . >When using LaTeX for bidirectional documents, only the xelatex engine is fully supported (use –pdf-engine=xelatex)

xelatex . Tex Live .

역시나, 경고 메시지 대로 한글 폰트가 없어서, 한글을 제외한 부분만 변환이 되었다. 절반의 성공은 이럴 때 쓰는 게 아닌가 싶다.

1.8 PDF로 변환해 보자 2

영한 사전 볼 때 빼고는 가 보질 않는 네이버에게서도, 한 가지 고마움을 느끼는 게 있다면 무료 폰트 제공이다. 그 중 나눔 폰트를 가지고 지금 당면한 문제를 해결해 보고자 한다.

1.8.1 나눔 폰트 설치하는 법

다음처럼 하면 되는데, 로컬 pc가 아니라 Docker container안에서 실행할 예정이다. 가장 간단한 방법은 apt-get install -y fonts-nanum 하는 것이다.

1.8.2 Pandoc에 한글 폰트 지정하기

맨 땅에 헤딩하면서 배우는 즐거움 중에는, 메뉴얼을 건너뛰는 것도 포함된다. 하지만, 지금처럼 필요할 때는 한번 훑어봐야 한다. PDF engine 부분을 살펴보니, mainfont, sansfont, 그리고 monofont를 지정하는 옵션이 있다.

그런데, 폰트 이름은 어떻게 알아내야 할까? 바로 다음처럼 fc-list :lang=ko를 실행하면 된다.

```
docker run -it --rm ₩
  -v ~/Data/Notes:/Notes ₩
  -v ~/Downloads:/Downloads ₩
  thomasweise/docker-pandoc ₩
  /bin/bash -c ₩
    "apt-get update && apt-get install -y fonts-nanum; ₩
     fc-list :lang=ko | grep -i 'nanum'"
```

역시나 확인해 보길 잘했다. 사실 NanumMyoungjo라고 잠깐 추측했었다. NanumMyeongjo임을 확인했기 때문에, 한번의 헤딩을 예방했다. 사실 폰트 파일 이름으로도 추측 가능하겠지만, 파일 이름과 폰트 이름이 다른 경우가 있기 때문에 정확히 확인하는 게 좋을 거 같다.

```
...
/usr/share/fonts/NanumFont/NanumGothic.ttf: NanumGothic,나눔고딕:style=Regular
...
/usr/share/fonts/NanumFont/NanumMyeongjo.ttf: NanumMyeongjo,나눔명조:style=Regular
...
```

1.8.3 다시 도전해 보자

이제 폰트 이름도 알고, 설치도 할 수 있다. 주 폰트는 나눔명조로, 그 이외에는 나눔고딕을 사용하기 위해 -V mainfont='NanumMyeongjo' -V sansfond='NanumGothic' -V monofont='NanumGothic'를 추가해서 다시 해보자.

```
docker run -it --rm ₩
  -v ~/Data/Notes:/Notes ₩
  -v ~/Downloads:/Downloads ₩
  thomasweise/docker-pandoc ₩
  /bin/bash -c ₩
    "apt-get update && apt-get install -y fonts-nanum; ₩
     cd /Notes/notes; ₩
     pandoc 'Markdown과 Pandoc 갖고 놀기 - PDF.md' ₩
```

```
-f gfm ₩
--pdf-engine=xelatex ₩
-V mainfont='NanumMyeongjo' ₩
-V sansfond='NanumGothic' ₩
-V monofont='NanumGothic' ₩
-o /Downloads/converted_1.pdf"
```

드디어 아무 메시지 없이 변환이 되었다. 그리고 변환된 파일을 열어보니, 일단 한글 문제는 해결된 듯 하다. 하지만, 이내 고쳐야 할, 혹은 고치고 싶은 더 많은 문제점들을 발견했다. 이 정도로 쉽다고 예상하지는 않았지만, 생각보다 더 많은 시간을 투자해야 할 듯 하다.

1.9 PDF로 변환해 보자 3

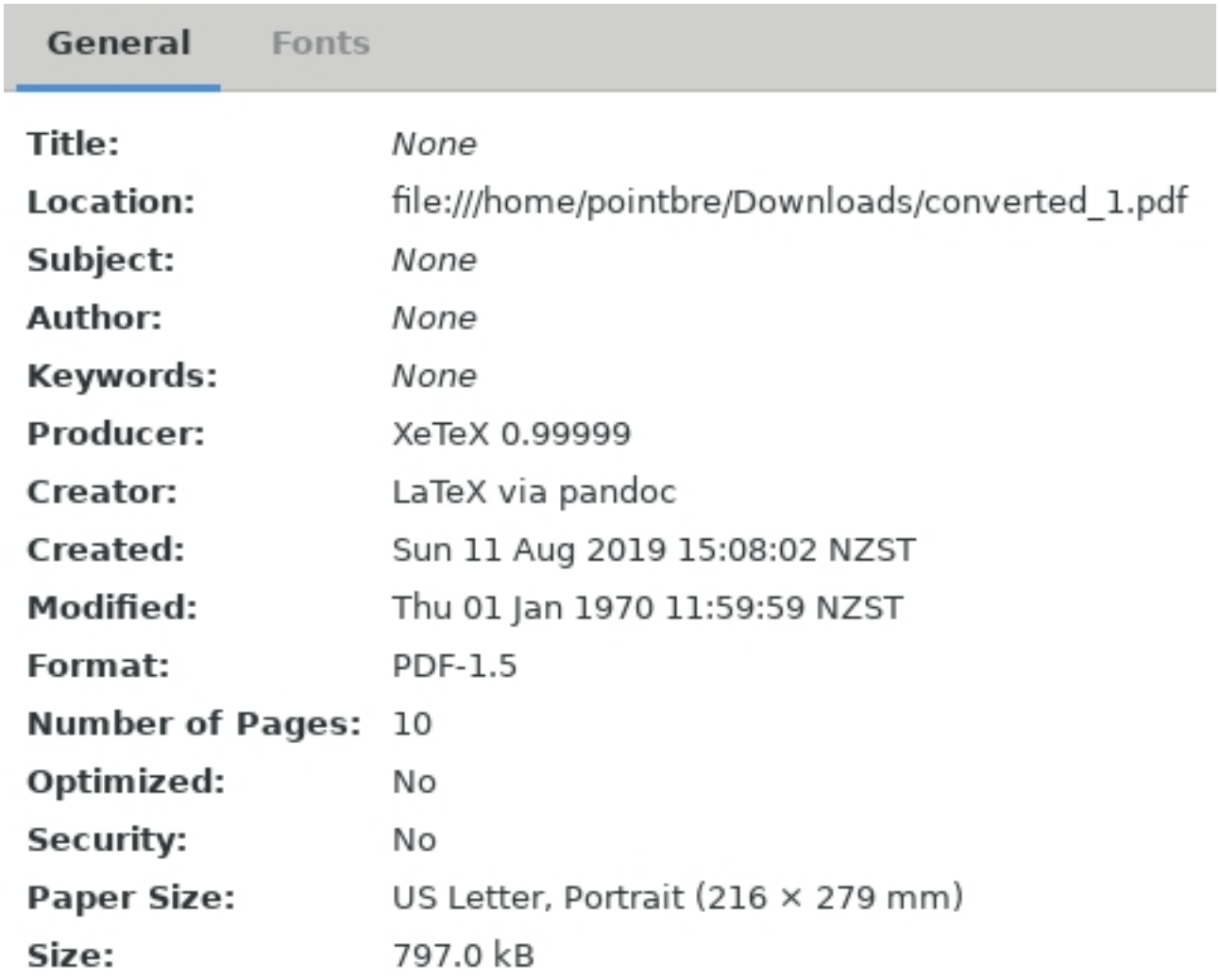
1.9.1 페이퍼 사이즈와 여백

US Letter 사이즈가 기본으로 잡혀 있다. 가장 흔하게 사용하는 A4를 사용하고, 광활한 여백도 좀 많이 줄이고 싶다.

-V papersize:a4로 종이 크기를 정하고, -V geometry=margin=2cm로 마진을 정하면 된다.

1.9.2 PDF 정보

Title, Subject, Author, Keywords를 채워 넣고 싶다.



General	Fonts
Title:	*None*
Location:	file:///home/pointbre/Downloads/converted_1.pdf
Subject:	*None*
Author:	*None*
Keywords:	*None*
Producer:	XeTeX 0.99999
Creator:	LaTeX via pandoc
Created:	Sun 11 Aug 2019 15:08:02 NZST
Modified:	Thu 01 Jan 1970 11:59:59 NZST
Format:	PDF-1.5
Number of Pages:	10
Optimized:	No
Security:	No
Paper Size:	US Letter, Portrait (216 × 279 mm)
Size:	797.0 kB

LaTex 제어문인 hypersetup을 직접 사용해야 하는 거 같다.

```
₩hypersetup{
  pdftitle={Markdown과 Pandoc 갖고 놀기 - PDF},
  pdfauthor={MadForFamily.com},
  pdfsubject={Markdown, Pandoc, XeLaTex},
  pdfkeywords={pandoc, pdf, xelatex}
}
```

1.9.3 Meta data

Notable이 사용하는 메타 데이터가 그대로 변환되고 있다. 제거해야 한다.

---

attachments: [notetaking.jpg, pandoc_writing.jpg, pdf_converted_1.jpg] pinned: true title: Markdown과 Pandoc 갖고 놀기 - PDF created: '2019-08-08T10:42:34.638Z' modified: '2019-08-11T03:06:52.687Z'

Markdown과 Pandoc 갖고 놀기 - PDF

sed '1{/^---$/!q;};1,/^---$/d' 'Markdown과 Pandoc 갖고 놀기 - PDF.md'처럼 해 보니 잘 지워진다.

1.9.4 Table이 가운데 정렬로 출력된다

가운데 정렬이 보기 싶다. 좌측으로 정렬하고 싶다.

col 1	col 2
val 1	val 2222222222

이미지

₩usepackage[margins=raggedright]{floatrow}를 지정해서 Pandoc과 같이 사용하면 된다.

1.9.5 링크 색상

링크인지 구분이 안 간다. 색상을 달리 주기 위해서는, 먼저 Pandoc 명령행 인자에서, -V colorlinks -V urlcolor=NavyBlue를 주면 된다. 더 상세한 색상은 hypersetup을 사용하는 거 같다.

```
% Url color setting and PDF attributes
₩hypersetup{
  bookmarksopen=true,
  linkcolor=blue,
  filecolor=magenta,
  urlcolor=RoyalBlue
}
```

1.9.6 Level 4, 5 header

Latex는 Level 3까지만 heading을 지원한다고 하나, 어찌되었건 Level 4(####), 5(#####) 헤더가 이어지는 텍스트와 합쳐지는 문제점을 해결해야 한다. 구글링 결과 찾은 결과에다가, Level 5 헤더도 같이 포함해서 추가했다. 다만 Level 6 헤더는 해결이 안될 거 같다. 6단계까지 쓸 일이 있을까 모르겠지만, 여하튼 알아둘 필요가 있다.

```
% Level 4, 5 header issue
₩usepackage{titlesec}
₩titleformat{₩paragraph}
  {₩normalfont₩bfseries}
  {}
  {0pt}
  {}
₩titleformat{₩subparagraph}
  {₩normalfont₩bfseries}
  {}
  {0pt}
  {}
```

1.9.7 TOC를 넣고 싶다.

목차를 넣으면 멋질 거 같다. --toc를 넣어서 목차를 채워 넣고, -V toc-title='Table of contents'로 목차의 제목을 지정한다. 이 옵션을 주고 pdf를 생성하니 목차가 자동으로 만들어져서 나왔다. 그런데, pdf 요약 화면에서 보니, 페이지가 제대로 링크되지 않았다. 혹시나 싶어 -N를 줘서 목차를 번호로 매기니 그제서야 제대로 나온다. 아무래도 아직 제대로 변환되지 않은 mermaid 다이어그램과 관련이 있을 거 같다. 나중에 다시 살펴봐야 겠다. 그리고 목차 출력된 후 바로 이어서 본문이 출력된다. 다음처럼 해서, 목차 출력 후에 페이지를 바꾼 후 본문이 출력되게 할 수 있다.

```
% Start a level 1 header in a new page after TOC
₩newcommand{₩sectionbreak}{₩clearpage}
```

1.9.8 내부 문서로의 링크를 개인 블로그에 올라갈 페이지 url로 변경해줘야 한다.

할 게 너무 많다. 다음 시도로 미루자.

1.9.9 너무 느린 변환

매번 폰트를 다운로드 해야 해서 속도가 너무 느리다. 아무래도 Docker image를 만들어야 겠다. 이것은 다음 시도에 해봐야 겠다.

1.9.10 Mermaid diagram

엇 다이어그램이 안 나오네. 흠, 뭔가 한참 삽질해야 할 거 같은 불길한 느낌이 든다.

Mermaid 다이어그램

```
graph LR
  A[Hard edge] -->|Link text| B(Round edge)
  B --> C{Decision}
  C -->|One| D[Result one]
  C -->|Two| E[Result two]
```

```
graph LR
  A[Hard edge] -->|Link text| B(Round edge)
  B --> C{Decision}
  C -->|One| D[Result one]
  C -->|Two| E[Result two]
```

이것도, 일단 이번 시도에서는 넘어가고, 다음에 도전해봐야 겠다. 이번 시도에 해야 할 게 너무 많다.

1.9.11 참조

- https://stackoverflow.com/questions/20788464/pandoc-doesnt-text-wrap-code-blocks-when-converting-to-pdf
- https://jdhao.github.io/2019/05/30/markdown2pdf_pandoc/
- https://sujinlee.me/pandoc-thesis/
- https://learnbyexample.github.io/tutorial/ebook-generation/customizing-pandoc/

1.9.12 .pdf.tex

위에서 필요한 LaTex 제어문을 모두 모은 것이다.

```
% Left align table
₩usepackage[margins=raggedright]{floatrow}

% color and other settings for hyperref package
₩hypersetup{
   bookmarksopen=true,
   linkcolor=blue,
   filecolor=magenta,
   urlcolor=RoyalBlue,
```

```
  pdftitle={Markdown과 Pandoc 갖고 놀기 - PDF},
  pdfauthor={MadForFamily.com},
  pdfsubject={Markdown, Pandoc, XeLaTex},
  pdfkeywords={pandoc, pdf, xelatex}
}

% Start a level 1 header in a new page after TOC
₩newcommand{₩sectionbreak}{₩clearpage}

% Level 4, 5, 6 header issue
% https://stackoverflow.com/questions/33134416/level-4-heading-issue-in-r-markdown
₩usepackage{titlesec}
₩titleformat{₩paragraph}
  {₩normalfont₩bfseries}
  {}
  {0pt}
  {}
₩titleformat{₩subparagraph}
  {₩normalfont₩bfseries}
  {}
  {0pt}
  {}
```

1.9.13 Pandoc을 다시 실행해보자

먼저 LaTex 파일을 생성해야 한다. vi ~/Downloads/.pdf.tex 하고, 위의 내용을 넣어준다. 그리고 나서 이 파일을 Pandoc에 지정하면 된다.

```
docker run -it --rm ₩
 -v ~/Data/Notes:/Notes ₩
 -v ~/Downloads:/Downloads ₩
 thomasweise/docker-pandoc ₩
 /bin/bash -c ₩
   "apt-get update && apt-get install -y fonts-nanum; ₩
    cd /Notes/notes; ₩
    sed '1{/^---$/!q;};1,/^---$/d' 'Markdown과 Pandoc 갖고 놀기 - PDF.md' > /Downloads/converted.md; ₩
    pandoc /Downloads/converted.md ₩
      -f gfm-raw_tex ₩
      --pdf-engine=xelatex ₩
      -V mainfont='NanumMyeongjo' ₩
      -V sansfond='NanumGothic' ₩
      -V monofont='NanumGothic' ₩
      -H /Downloads/.pdf.tex ₩
      -V colorlinks ₩
      -V urlcolor=NavyBlue ₩
      -V papersize:a4 ₩
      -V geometry=margin=2cm ₩
      --toc ₩
      -N ₩
      -V toc-title='Table of contents' ₩
      -o /Downloads/converted_1.pdf; ₩
    rm -f /Downloads/converted.md";
```

에러 없이 변환이 되었다. 변환된 pdf를 열어보니, 쓸만한 정도로는 변환이 된 거 같다. 이제는 미뤄뒀던 숙제를 해야 할 시간이다.

1.10 PDF로 변환해 보자 4

1.10.1 All-in-one Pandoc image

처음 만들때는 손에 익은 ubuntu 기반으로 만들었다. 위에서 테스트한 게 모두 잘 동작했는데, 문제는 Docker image의 크기가 4.6GB로 너무 크다는 점이었다. 그래서 Alpine Linux 기반으로 다시 만들었다. 일단 xelatex를 가지고 pdf로 변환하는

데에 문제가 없을 정도로만 구성했다. 그 위에, 나눔폰트와 mermaid-filter(mermaid 다이어그램 변환용)를 설치해서 위에서 테스트한 게 잘 동작하는지 확인했는데, 대략 사이즈가 3분의 1로 줄었다. 그런데 설치된 패키지의 구성이나 버전의 차이로 인해서 pandoc으로 변환시 -V subparagraph를 추가해줘야 했다.

이제, 개인적으로 선호하는 gitlab.com에 public git repo를 만들고 관리하려고 한다. 하지만 이미지는 Dockerhub에 올려서 쉽게 검색할 수 있도록 할 예정이다. gitlab.com이 자체 docker registry를 지원하지만, 공유 측면을 보면 Dockerhub에 올리는 게 좋을 거 같다. 구글링의 도움을 받았다.

먼저 .gitlab-ci.yml에서 사용한 변수들을 Dockerhub에 올릴 수 있게 설정해 줘야 한다. Settings - CI/CD - Variables에서 다음처럼 설정하면 된다.

- CI_REGISTRY: docker.io
- CI_REGISTRY_IMAGE: pointbre/all-in-one-pandoc:latest
- CI_REGISTRY_USER: your_dockerhub_password
- CI_REGISTRY_PASSWORD: your_dockerhub_id

위의 Dockerfile을 push 해놨으니, 이제 .gitlab-ci.yml을 다음처럼 준비하면 끝이다.

```
stages:
 - build-docker

build-docker:
  stage: build-docker
  image: docker:stable
  services:
    - docker:dind
  only:
    - master
  before_script:
    - docker logout
    - docker login -u "$CI_REGISTRY_USER" -p "$CI_REGISTRY_PASSWORD" $CI_REGISTRY
  script:
    - docker build --pull --tag "$CI_REGISTRY_IMAGE" .
    - docker push "$CI_REGISTRY_IMAGE"
```

Dockerfile과 .gitlab-ci.yml을 Push 후, 꽤 시간이 흐른 후 드디어 docker image가 등록이 되었다. Docker hub에서 검색해보니 잘 된다.

1.10.2 빌드된 이미지로 다시 시도해보자

```
docker run -it --rm ₩
  -v ~/Data/Notes:/Notes ₩
  -v ~/Downloads:/Downloads ₩
  pointbre/all-in-one-pandoc ₩
  /bin/bash -c ₩
    "cd /Notes/notes; ₩
     sed '1{/^---$/!q;};1,/^---$/d' 'Markdown과 Pandoc 갖고 놀기 - PDF.md' > /Downloads/converted.md; ₩
     pandoc /Downloads/converted.md ₩
        -f gfm-raw_tex ₩
        --pdf-engine=xelatex ₩
        -V mainfont='NanumMyeongjo' ₩
        -V sansfond='NanumGothic' ₩
        -V monofont='NanumGothic' ₩
        -H /Downloads/.pdf.tex ₩
        -V colorlinks ₩
        -V urlcolor=NavyBlue ₩
        -V papersize:a4 ₩
        -V geometry=margin=2cm ₩
        --toc ₩
        -N ₩
        -V toc-title='Table of contents' ₩
        -V subparagraph ₩
        -o /Downloads/converted_1.pdf; ₩
      rm -f /Downloads/converted.md";
```

-V subparagraph를 추가하고 변환했다. 완료 후 변환된 pdf를 살펴보니, 그전에 변환했던 것과 동일했다. 안심하고 계속 사용해도 될 거 같다.

1.10.3 Mermaid-filter

Pandoc으로 Markdown 문서를 변환하는 경우, 문서 안에 포함된 mermaid 다이어그램을 처리해주는 NodeJS로 작성된 어플리케이션이다. All-in-one Pandoc 이미지에 이미 설치되어 있으니 사용해보자. 위에서 사용했던 명령에다 -F mermaid-filter를 추가하면 된다.

```
docker run -it --rm ₩
  -v ~/Data/Notes:/Notes ₩
  -v ~/Downloads:/Downloads ₩
  pointbre/all-in-one-pandoc ₩
  /bin/bash -c ₩
    "cd /Notes/notes; ₩
     sed '1{/^---$/!q;};1,/^---$/d' 'Markdown과 Pandoc 갖고 놀기 - PDF.md' > /Downloads/converted.md; ₩
     pandoc /Downloads/converted.md ₩
        -f gfm-raw_tex ₩
        --pdf-engine=xelatex ₩
        -V mainfont='NanumMyeongjo' ₩
        -V sansfond='NanumGothic' ₩
        -V monofont='NanumGothic' ₩
        -H /Downloads/.pdf.tex ₩
        -V colorlinks ₩
        -V urlcolor=NavyBlue ₩
        -V papersize:a4 ₩
        -V geometry=margin=2cm ₩
        --toc ₩
        -N ₩
        -V toc-title='Table of contents' ₩
        -V subparagraph ₩
        -F mermaid-filter ₩
        -o /Downloads/converted_1.pdf; ₩
      rm -f /Downloads/converted.md";
```

우워어, 에러가 뿜어져 나온다.

1.10.4 Root user와 chromium path 문제 해결

이를 해결하기 위해서는 .puppeteer.json를 pandoc이 실행되는 곳에 둬서, mermaid-filter가 내부에서 mermaid diagram 변환에 사용되는 mermaid-cli에게 넘겨줄 수 있도록 해야 한다.

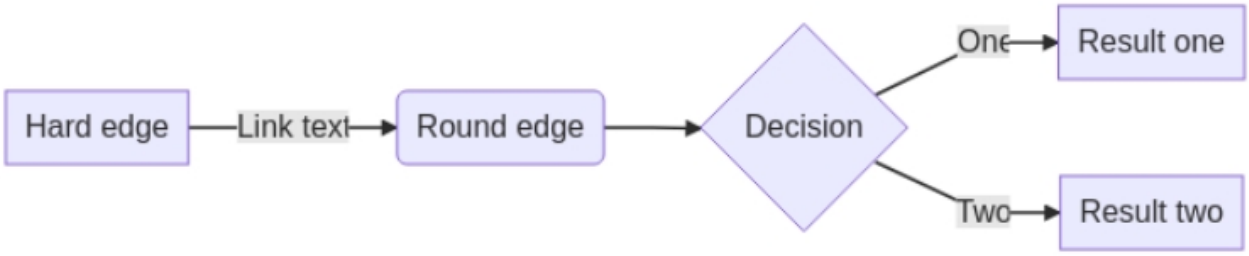
vi ~/Downloads/.puppeteer.json 후 다음처럼 입력하면 된다. 그 후 다시 위의 명령을 실행해보자.

```
{
  "args":["--no-sandbox"],
  "executablePath":"/usr/bin/chromium-browser"
}
```

드디어 pdf 문서 안에 mermaid 다이어그램이 표시가 된다.

1.6.5 Mermaid 다이어그램

```
graph LR
  A[Hard edge] -->|Link text| B(Round edge)
  B --> C{Decision}
  C -->|One| D[Result one]
  C -->|Two| E[Result two]
```



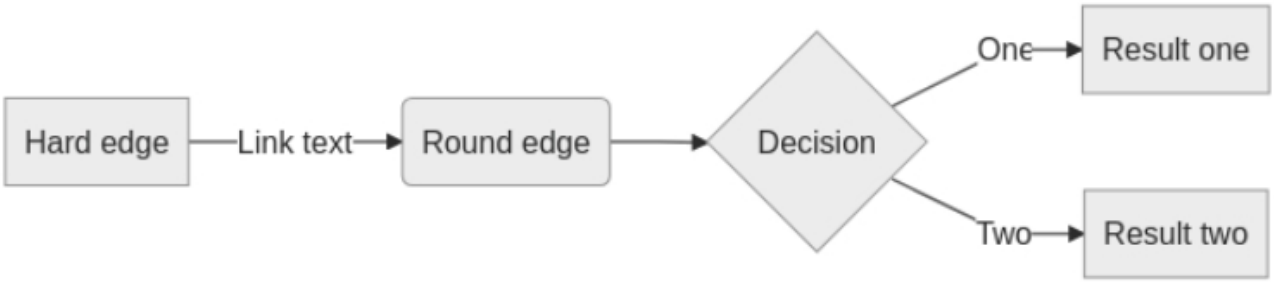
1.10.5 테마 변경

색상이 별루 마음에 안 들어서 .mermaid-config.json를 통해 사용하고 싶은 theme을 neutral로 고르려고 한다. vi ~/Downloads/.mermaid-config.json 한 후 다음을 입력하고 나서, 다시 변환 명령을 실행하면 된다.

```
{ "theme": "neutral" }
```

1.10.6 css 변경

변환된 다이어그램을 자세히 보니, 폰트가 조금 짤려 있어서 거슬렸다. Github style css를 다운받은 후 ~/Downloads/.mermaid.css로 변경하고 다시 변환 명령을 실행하니, 테마도 더 어울리고, 글자 표시도 훨씬 나아졌다.



1.10.7 내부 링크 처리

이제 하나 남은 것은 다른 local markdown 파일로의 링크이다. 이 링크는 https://madforfamily.com에 업로드 될 때, Hugo의 relref로 변환되고 있다. 그렇기 때문에, 동일한 방식으로 파일명을 변환해서 클릭시 publish된 게시물로 가도록 하는 것이 좋을 거 같다.

그런데 파일명을 Hugo가 사용하는 url(slug)로 변환하는 게 문제였다. sed를 가지고 변환해보려 했지만, 여러 파일들을 테스트하다 보니, 점점 변환 규칙이 단순하지 않은 거 같은 느낌적인 느낌이 들었다. 그래서 결국, hugo project 사이트로 가서 소스를 들여다 보기 시작했고, 변환하는 소스도 발견했다. 문제는 전혀 경험이 없는 Go로 씌여졌다는 점이었다. 고민의 시간이 흐르고 난 후, 한번 더 삽질을 하기로 했다. 내가 필요한 건, 파일명을 hugo url로 변환하는 것이었다. 파일 속에 있거나, 주어진 문자열을 변환하는 것이면 족했다. 구글링을 총 동원해서 만들어 낼 수 있었고, 결과물을 공개 git repo에 등록해뒀다.

1.11 PDF로 변환해 보자 5

다음은 최종 버전의 .pdf.tex 파일이다. 기존에 설정에다가, header와 footer를 달고, fenced code block과 inline code block이 페이지 넓이를 넘어가지 않도록 line break가 되도록 설정했다.

```
% Left align table
₩usepackage[margins=raggedright]{floatrow}

% color and other settings for hyperref package
₩hypersetup{
   bookmarksopen=true,
   linkcolor=blue,
```

```
    filecolor=magenta,
    urlcolor=RoyalBlue,
    pdfauthor={https://madforfamily.com},
    pdfcreator={https://madforfamily.com},
    pdfproducer={https://madforfamily.com},
    pdfsubject={},
    pdftitle={title to be replaced when pdf is built}
}

% Start a level 1 header in a new page after TOC
₩newcommand{₩sectionbreak}{₩clearpage}

% https://tex.stackexchange.com/questions/468220/how-to-renew-verbatim-to-verbatim-from-fancyvrb
% http://mirror.aut.ac.nz/CTAN/macros/latex/contrib/fvextra/fvextra.pdf
₩usepackage{fvextra}
₩RecustomVerbatimEnvironment{verbatim}{Verbatim}{breaklines,breakanywhere}

% https://tex.stackexchange.com/questions/219445/line-break-in-texttt
₩renewcommand{₩texttt}[1]{%
  ₩begingroup
  ₩ttfamily
  ₩begingroup₩lccode`~=`/₩lowercase{₩endgroup₩def~}{/₩discretionary{}{}{}}%
  ₩begingroup₩lccode`~=`[₩lowercase{₩endgroup₩def~}{[₩discretionary{}{}{}}%
  ₩begingroup₩lccode`~=`.₩lowercase{₩endgroup₩def~}{.₩discretionary{}{}{}}%
  ₩catcode`/=₩active₩catcode`[=₩active₩catcode`.=₩active
  ₩scantokens{#1₩noexpand}%
  ₩endgroup
}

% Level 4, 5, 6 header issue
% https://stackoverflow.com/questions/33134416/level-4-heading-issue-in-r-markdown
₩usepackage{titlesec}
₩titleformat{₩paragraph}
   {₩normalfont₩bfseries}
   {}
   {0pt}
   {}
₩titleformat{₩subparagraph}
   {₩normalfont₩bfseries}
   {}
   {0pt}
   {}

% https://cs.overleaf.com/learn/latex/Headers_and_footers
₩usepackage{fancyhdr}
₩pagestyle{fancy}
₩fancyhf{}
% header - site url
₩rhead{post url to be replaced}
% footer - page number
₩rfoot{Page ₩thepage}
% divider lines
₩renewcommand{₩headrulewidth}{1pt}
₩renewcommand{₩footrulewidth}{1pt}
```

위의 .pdf.tex를 가지고 변환하니 다음처럼 잘 변환이 되었다. 구분선도 넣고, 게시물 url도 넣고 하니, 훗 꽤 괜찮아 보인다.

https://madforfamily.com/post/블로그-관리를-위한-mailgun-메일서버.html

블로그 관리를 위한 MailGun 메일서버

Commento 사용을 위한 메일 서버를 찾다보니 얻어 걸린 메일 서비스이다. 한달 10000통까지는 무료이다. 주의할 점이라면, 자체 메일함을 제공하지 않기 때문에, 메일을 abc@abc.com으로 수신하려면 Route을 반드시 설정해서 다른 이메일로 전달하거나, 특정 url을 실행해서 타 시스템에 전달하거나 해야 한다. 즉 메일이 수신될 경우, mailgun이 설정된 route들을 실행하기만 하기 때문에, 보관하거나 하려면 별도의 설정을 해야 한다는 의미이다.

Domain 인증

Godaddy의 경우, Host에 아무 것도 쓰지 않을 경우에는 @을 기재해야 한다. Host가 값을 가질 경우는, 최상위 도메인은 쓰지 않고 생략해야 한다. 즉, 서브 도메인을 위해 CNAME 레코드를 email.abc.com에 대해 설정하려면, Host에는

1.12 블로그에 링크를 걸어보자

Hugo Academic theme을 약간 수정해서 구현했다.

블로그 관리를 위한 MailGun 메일서버

2019-04-29 · 3 min read

PDF PRINT EMAIL FACEBOOK TUMBLR

1.13 마무리

조금씩 조금씩 삽질하다 보니 어느 새 한달이 넘어가 버렸다. 그래도 원하는 바를 달성할 수 있었기에, 무의미한 삽질은 아니였음에 감사해야 할 것이다. 그런데, "그냥 웹사이트에서 print to pdf나 pdf 변환 extension으로 다운받으면 되는 데, 왜 이렇게 힘들게 구현했지?" 하는 생각이 문득 든다. "아니야, 아니야, 이렇게 만든 pdf랑 그렇게 만든 pdf는 품질이 달라, 많이 달라!"라는 변명으로 기나 긴 삽질을 정당화하며, 누군가에게는 도움이 되는 글이었으면 좋겠다. 추후, 업무 문서 자동화에 사용하기도 해 보려고 하는데, 그 때는 덜 삽질할 수 있기도 기원해본다.

마지막으로, 여기에서 사용했던 .pdf.tex, .mermaid-config.json, .puppeteer.json, .mermaid.css, build.sh, .gitlab-ci.yml은 모두 여기서 다운로드할 수 있다. All-in-one-pandoc docker image는 사용하다가 문제가 생기면 계속해서 업데이트할 예정이라서 Dockerfile을 굳이 여기서 첨부하지는 않으려고 한다. Hugo-slugify도 마찬가지이다.