LaTeX 自動ビルド・校正レビュー機能付きテンプレート

being24

2022 年 1 月 23 日

0.1 概要

0.1.1 このテンプレートの目的

これは日本語論文を書く上で.

- LaTeX の環境構築
- 誤字・表記揺れの修正.

のような本質的でない部分に時間をかけず，論文の内容に集中するためのものです.

0.1.2 対象となるユーザ

理工系の卒業論文・修士論文を書く人を対象としています（一部設定をカスタマイズすることでそれ以外の分野でもお使い頂けます）．また，Git および GitHub を使用してバージョン管理することで真価を発揮するため，Git の使用は前提となります.
これはどんな TeX ソースでもビルド出来るようにすることは目標としていないため,.

- デフォルトのままで完璧に上手くやって欲しい
- 自分でカスタマイズはしたくない.

というユーザにはあまり向かない可能性があります.

0.1.3 提供する機能一覧

ホスト OS へインストールされている LaTeX および，TeX Live をインストールした Docker コンテナを使用し，下記の操作をサポートしています.

- latexmk を使用した PDF のビルド（`make pdf`）
- TeX ソースの変更検知時に自動ビルド（`make watch`）
- 生成物の破棄（`make clean`）

また，GitHub Actions を用いて下記の操作をサポートします.

- master ブランチへの push 時に自動ビルド・生成 PDF をアップロード
- Pull Request 作成時にルールベースでの校正，および，レビューでの指摘
- タグを push するたびに GitHub Releases へ自動で PDF をアップロード.

0.2 使用方法

0.2.1 動作する環境

検証した OS は以下の通りです.

- Windows 10
- macOS 10.14 or later
- Ubuntu 18.04 LTS or later

Windows・Ubuntu では TeX Live 2019 を，macOS では MacTeX 2019 を用いています．その他のツールの要求バージョンは以下の通りです.

- Docker CE 18.09 or later（Linux）
- Docker Desktop for Windows（Windows）
- Docker Desktop for Mac（macOS）
- GNU Make 3.8 or later
- (Optional) hub version 2.12.0 or later（https://github.com/github/hub）
- (Optional) Nodejs v12.0.0 or later

ただし，これは完璧に動作することを保証したものではありません．使用者の環境によっては一部機能等がうまく動作しない可能性もあります.

0.2.2 論文作成時のワークフロー

簡単に，このテンプレートを使用してどのように執筆を進めるかを示します．また，概要を図 1 に示します.

前提

master ブランチに居る.

これからすること

新しい章を書き始める.

手順

1. ブランチを切り空の Pull Request を発行する．`make draft branch=`**ブランチ名**で新しいブランチが切られ，Pull Request 発行の画面がブラウザで開くので，タイトル等を入力する.
2. 通常通り tex ソースを編集して commit および push.
3. textlint による校正結果が reviewdog によりレビューとして投稿されるので，それを参

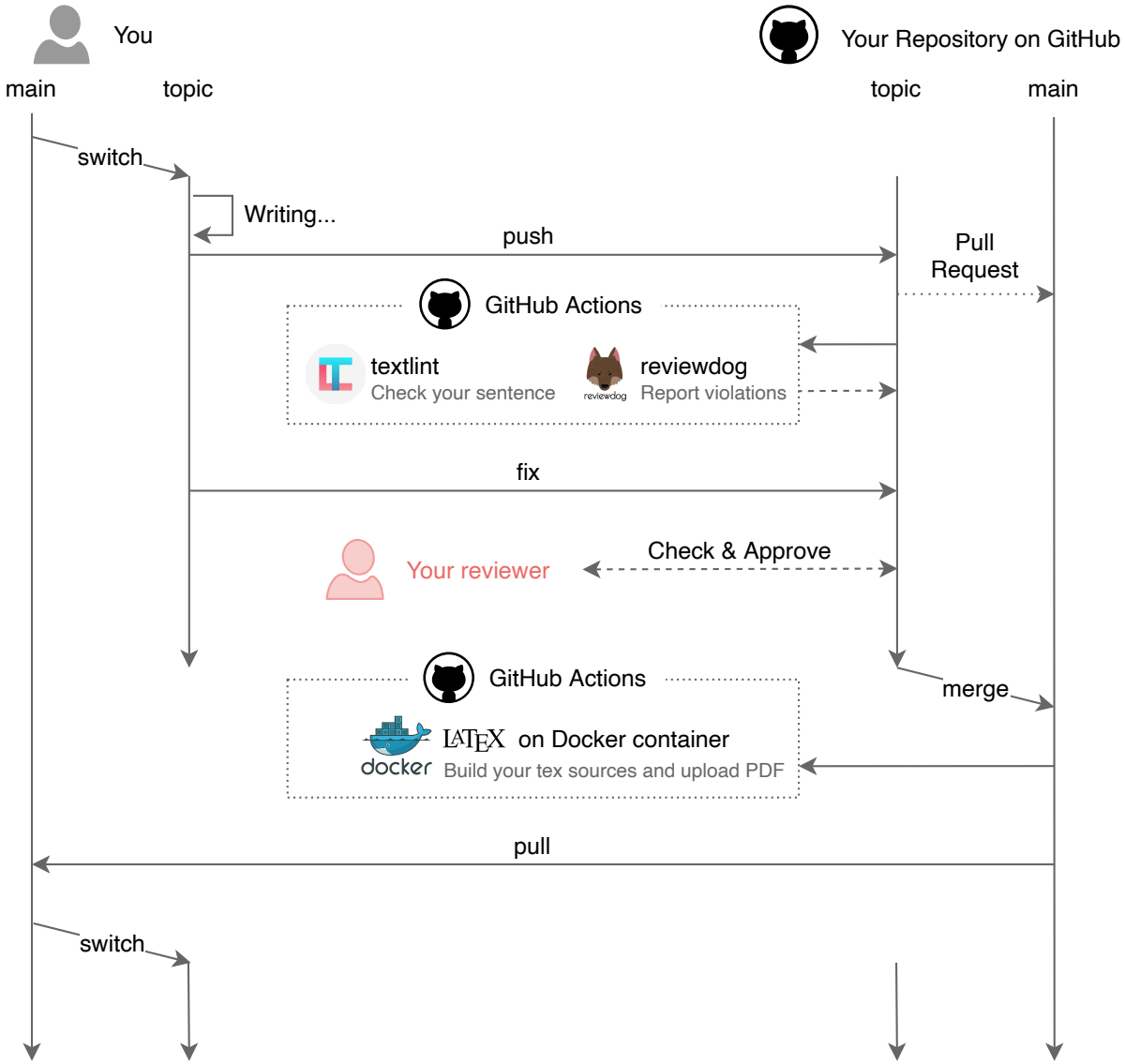

Fig.1　textlint と GitHub Actions を活用した執筆フロー

考に修正する．

4. ある程度書けたら共著者や教員・先輩など，内容をレビューする人をレビュアーとしてアサインする．
5. レビュアーの指摘を元に修正．
6. 全ての指摘を修正し，加筆の必要がなくなった時点で master へマージする．
7. ローカルのリポジトリで master をチェックアウトして git pull する．
8. 1 へ戻る．

このイテレーションをセクションごとに行います．1 つのセクションが大きい場合，その中の切りの良い単位で区切り，複数に分けて行います．1 つの Pull Request による加筆が図表等抜きで 10 ページを超えるようなら，変更のサイズが大きすぎるかも知れません．より小さい単位の細かい Pull Request へ分けることを推奨します．

また，一人で複数の章を同時並行で編集する場合や，複数人がそれぞれ異なる章を単一ファイルで編集する場合，容易にコンフリクトが発生し得ます．これを避けるため，大きな章ごとに subfiles パッケージと \subfile を使用してソースファイルを分割することを推奨します．これを使用すると，各子セクションを個別にコンパイルでき，かつ，親となる TeX ソースのプリアンブルを共有する事が出来ます．構成などはこのテンプレートを参考にしてください．

0.2.3 Makefile による各種操作

PDF の生成に関連した操作は全て，デフォルトで Docker コンテナを利用します．これは USE_DOCKER という環境変数によって制御されています．デフォルトでは USE_DOCKER=yes です．Docker は不要でネイティブで動かしたい場合，USE_DOCKER=no とすることでこれを無効化できます．

また，Windows で利用する場合は Listing1 および Listing2 のようにすることで Docker の有効無効を切り替えることが出来ます．

Listing 1　Powershell の場合

```
C:\Users\You> # 無効にする
C:\Users\You> $env:USE_DOCKER="no"
C:\Users\You> # 有効にする
C:\Users\You> $env:USE_DOCKER="yes"
```

Listing 2　cmd.exe の場合

```
C:\Users\You> rem 無効にする
C:\Users\You> set USE_DOCKER=no
C:\Users\You> rem 有効にする
C:\Users\You> set USE_DOCKER=yes
```

0.2.3.1 make pdf

main.tex をコンパイルし，完全な PDF を生成するコマンドです．このコマンドでは \subfile によって分割したファイルを個別にコンパイルすることはサポートされていません．

このツールは全体を通してコンパイルには latexmk を使用します．デフォルトで使用されている.latexmkrc をカスタマイズするには，0.2.3.6 をご覧ください．

0.2.3.2 make watch

latexmk による変更検知時の自動ビルドを実行します．Docker を利用しない場合，viewer の指定があればビルドした PDF を開きます．Docker を利用する場合，Docker コンテナ内からホスト上の PDF ビューワを開くことができないため，`-view=none` として開かないようにしています．自動リロードに対応した PDF ビューワで開くと，ビルドの度に自動で変更が反映されます．

このコマンドでは分割したファイルの個別コンパイルをサポートしています．例えば以下の様に実行することで，sections/usage.tex だけを usage.pdf として閲覧できます．

```
$ make watch target=sections/usage.tex
```

target の指定が無い場合，main.tex を監視してコンパイルします．

0.2.3.3 make all

TeX ソースのコンパイルを行う前に，前回のコンパイルによる生成物を全て破棄してコンパイルを実行します．

0.2.3.4 make clean

コンパイルによる成果物を全て破棄します．最終成果物の PDF も含めて破棄されることに注意してください．

0.2.3.5 make draft

現在のブランチから新しいブランチを切り，空の commit を行ってリモートへ push します．その後，hub コマンドを利用して Pull Request 作成のページを標準のブラウザで表示します．実行時にブランチ名を指定することが出来ます．指定しない場合，デフォルト値の `WIP` が使用されます．

```
$ make draft branch=NAME
```

0.2.3.6 make latexmkrc

コンテナ内で使用されている.latexmkrc をこのディレクトリ直下にコピーします．これを編集することで，latexmk の動作をカスタマイズできます．

0.2.3.7 make lint

GitHub Actions 上で実行される lint をローカルでも実行します．事前に Nodejs をインストールし `npm install` を実行しておく必要があります．

0.2.3.8 make fix

lint に違反したもののうち，自動的に適用が可能なものについて訂正します．これは例えば prh などによる表記揺れ訂正が対応しています．0.2.3.7 と同様に，事前に Nodejs のインストールと `npm install` の実行が必要です．

0.2.4 VSCode

このテンプレートは VSCode で利用することを想定しています．単に VSCode で開き，LaTeX Workshop をインストールするだけで利用可能になるように settings.json を記述しています．

0.3 サンプル

0.3.1 特定のルールを無視する

以下の様に記述することで，ある行からある行までの間で特定のルールを無視できます.

Listing 3 textlint-filter-rule-comments の使用方法

```
% textlint-disable ja-technical-writing/sentence-length
これはとても長い長ーーーーーーーーい文章なので、
本来ならlintのルールに引っかかってしまいますが、
このようにコメントを付加することでこの一文だけは
lintの対象から除外されるので、特定のルールに
どうしても引っかかってしまう・しかし修正することは
難しい場合にご活用下さい
% textlint-ensable ja-technical-writing/sentence-length
```

複数のルールを無視したい場合は，ルール名をカンマで区切って書きます.

0.3.2 新規分割ファイルの追加

Listing 4 新章の追加テンプレート

```
%! TEX root = ../main.tex
\documentclass[main]{subfiles}

\begin{document}

\section{新セクション}

これはサンプルの文章です

\end{document}
```

LaTeX Workshop のためにマジックコメントを追加する必要があります．具体的には Listing4 のように記述します.

0.4 FAQ

0.4.1 利用する Docker イメージを差し替えたい

Docker イメージは以下の箇所で利用されています.

- .vscode/settings.json
- .GitHub/workflows/build.yml

上記のファイルに記述されている `pddg/latex:2.0.0` の部分を任意のイメージ名に書き換えて下さい.

0.4.2 Lint が終わらない・タイムアウトする

このテンプレートの GitHub Actions ワークフローでは Lint が時間を使い切らないように，タイムアウトが 10 分に設定されています．大規模な TeX ソースではこの時間中に収まらない可能性もあります.

0.4.3 VSCode で保存するとエラーが出てプレビューできない

TeX ソースのシンタックスエラーの疑いがあります．ログをよく読んでください.

0.4.4 VSCode の PDF Viewer で synctex が有効にならない

なぜか時々発生するのを確認していますがトリガーは不明です．[1]

参考文献

[1] 北清勇磨, 伊藤毅志, *et al.*, “カーリングの戦略を支援するシステムの提案と構築,” **ゲームプログラミングワークショップ** *2013* **論文集**, pp. 154–161, 2013.