

継続的インテグレーションを用いた文書検査システム の構築

プロジェクトマネジメントコース
ソフトウェア開発管理グループ
矢吹研究室
1342097
浜野 太豪

謝辞

本研究を進めるにあたり，矢吹研究室矢吹太郎准教授には，多くの時間をご指導にさいて頂きました．また矢吹研究室の皆様には，多くの知識や示唆を頂きました．協力していただき皆様に感謝の気持ちと御礼を申し上げます．

目次

第 1 章	序論	3
第 2 章	目的	4
第 3 章	利用するサービスの解説	5
3.1	RedPen	5
3.2	Github	7
3.3	Wercker	10
3.4	Docker	12
第 4 章	開発環境構築ツールの解説	14
4.1	Chocolatey	14
4.2	DockerToolBox	16
4.3	Oracle VM VirtualBox	17
第 5 章	開発環境の導入	18
5.1	Docker のインストール	22
第 6 章	自動化環境の構築	37
6.1	GitHub の設定	37
6.2	Wercker の設定	44
第 7 章	GitHub に文書提出	48
第 8 章	結果	59
第 9 章	考察	61
第 10 章	結論	62
	参考文献	63

第 1 章

序論

ドキュメントの検査にプログラムやツールによるサポートが必要である。なぜなら、システム開発の現場では、様々なドキュメントを作成する必要があるからだ。例えばプロジェクト計画書や要件定義書、マニュアルなどがある。このような文書では、読み手に誤解を与えてはいけない。さらに、わかりやすい文書を書くには一定のルールを守る必要がある。短い文で書くこと、正しい表現方法で書くこと、フォーマットを統一することなどである。このようなルールで、大量のドキュメントを人の目によってチェックすることには限界がある。

そこで継続的インテグレーションを活用する方法がある。継続的インテグレーションとは、プログラム全体を常に統合して動く状態にしておくことである。最近では、ビルド（プログラムのコンパイルや自動テスト、アーカイブ化、ソースコードへのタグ付け、実行環境へのデプロイの一連の手順）を自動化するツールが数多くある [1]。

このような継続的インテグレーションで活用されている自動化ツールを用いて、大量のドキュメントをチェックできるツールを構築する。

第 2 章

目的

文書チェックを自動的に行うシステムを構築する．研究室ではドキュメントの変更履歴をバージョン管理システム GitHub を用いて管理している．GitHub に文書を提出した際に，自動で文書チェックプログラムが実行され，実行結果の表示，通知を行う．

第 3 章

利用するサービスの解説

本研究では継続的インテグレーションを用い、文書の自動検査を行う。継続的インテグレーションを行うツールとして Wercker、ビルド環境の構築に Docker、文書検査のビルドに Redpen というサービスを用いる。以下のサービスの解説を行う。

3.1 RedPen



図 3.1 RedPen

RedPen は技術文書をターゲットにした文書の自動検査ツールである。技術文書にはマニュアルやチュートリアル、論文、仕様書等が含まれる。技術文書にはわかりやすさが求められる。詩や歌謡曲の歌詞にはわかりやすさを放棄し、わざと曖昧な部分を残すことでより心に響く作品となる場合がある。しかしわかりやすさが第一に求められる技術文書には余韻や深みは必要ない。

日記等と違って技術文書は自分以外の第三者が読むため、恥ずかしくない文書を書く必要がある。恥ずかしい文書は利用する特殊文字や専門用語が揃っていないかったり、"彼がが行く"のように明らかな文法間違いが多数存在する。また"すごい"や"っていうか"などの口

語が混じってしまっていると、文書のみならず、文書が扱う製品の品質までにあらぬ疑いをもたれてしまう。

RedPen は技術文書を書く上の規約にしたがって文書を記述しているか検査するツールである。

そのほかに RedPen の特徴として柔軟な設定、言語非依存、マークアップ言語サポートがあげられる。柔軟な設定では RedPen の設定ファイルを変更することで検査仕様の変更が容易に行える。言語非依存ではどのような言語で書かれた文書でも処理することができる。もちろん日本語に対応している。マークアップ言語サポートでは多様なフォーマット (Markdown, LaTeX など) で記述された文書をそのまま検査できる [2]。

3.2 Github

3.2.1 Git

Git(ギット)とは、2005 年に開発された、ファイルやプログラミングコード、文書の変更履歴を管理するためバージョン管理ツールのことである。

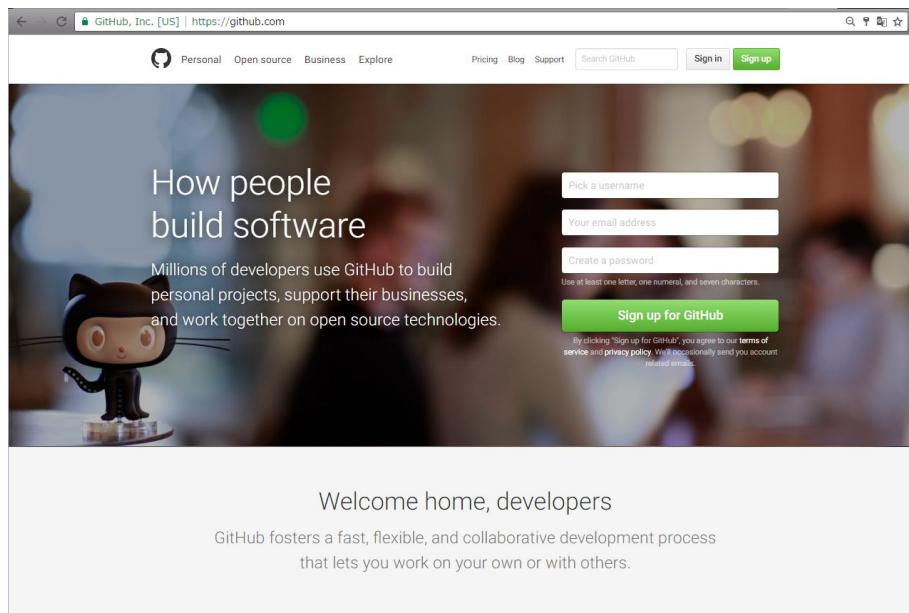


図 3.2 GitHub

GitHub は 2008 年オープンした、Git のリモートリポジトリと、さまざまな Web ツールを提供するサービスである。オープンソースソフトウェアは、時間的にも地理的にも離れたところに住む複数のプログラマによって開発される。GitHub では相談して課題を解決する Issue(イシュー) や改善案を気軽に提案できる Pull Request(プルリクエスト) などの機能が、注目を集めている。[3] GitHub のさまざまな機能を一部抜粋して、説明を記述する。

3.2.2 リポジトリ

ファイルやディレクトリの状態を記録する場所。

3.2.3 リモートリポジトリ

手元に置いてあるローカルなリポジトリ以外の、ネット上に置かれたリポジトリのこと。

3.2.4 commit

ファイルやディレクトリの変更をリポジトリに記録する機能である。

3.2.5 clone

ネット上にあるリポジトリをローカルにコピーする機能である。

3.2.6 Origin

clone 元のリモートリポジトリのこと。

3.2.7 Push

リモートリポジトリに自分の変更履歴がアップロードされ、リモートリポジトリ内の変更履歴がローカルリポジトリの変更履歴と同じ状態にする機能である。

3.2.8 branch

履歴の流れを分岐して記録していくためのもの。分岐したブランチは、他のブランチの影響を受けないため、同じリポジトリ中で複数の変更を同時に進めていける機能である。

3.2.9 pull

リモートリポジトリから最新の変更履歴をダウンロードしてきて、自分のローカルリポジトリにその内容を取り込む機能である。

3.2.10 Pull Request

相手に対して自分の変更を pull してもらうように要求する機能である。

3.2.11 Revert

ステージングエリアに追加した変更を取り消す機能である。

3.2.12 タグ

コミットを参照しやすくするために、わかりやすい名前を付ける機能である。

3.2.13 Label

自由に作成でき、Issue をフィルタリングできる機能である。

3.2.14 Merge

当該ブランチに対して別のブランチの差分を取り込むことである。

3.2.15 Fork

GitHub のサービスで、相手のリポジトリを自分のリポジトリとしてコピー・保持できる機能ある。

3.2.16 Issue

ソフトウェア開発におけるバグや議論などをトラッキングして管理するために発行する。

3.2.17 デプロイ

ソフトウェアの分野で、開発したソフトウェアを利用できるように実際の運用環境に展開する。

3.2.18 リリース

プロセスを次の段階に進めることを認める機能である。

3.2.19 Watch

リポジトリに関する情報を Notifications に表示する機能である。

3.2.20 Star

リスト一覧からリポジトリを探すことが出来るようにする機能である。また、注目度を表す指標にもなる。

3.2.21 Fork

GitHub 側にある特定のリポジトリを自分のアカウント以下のリポジトリに複製する機能である。

3.2.22 人数

開発人数のことである。ここでは、Origin リポジトリにコミットした人数のことを示す。

3.2.23 MileStone

やるべきタスクの管理に Issue を用いることができるようにする機能である。

3.2.24 Wiki

簡単な記法によってドキュメントを作成、編集するための機能である。

3.3 Wercker

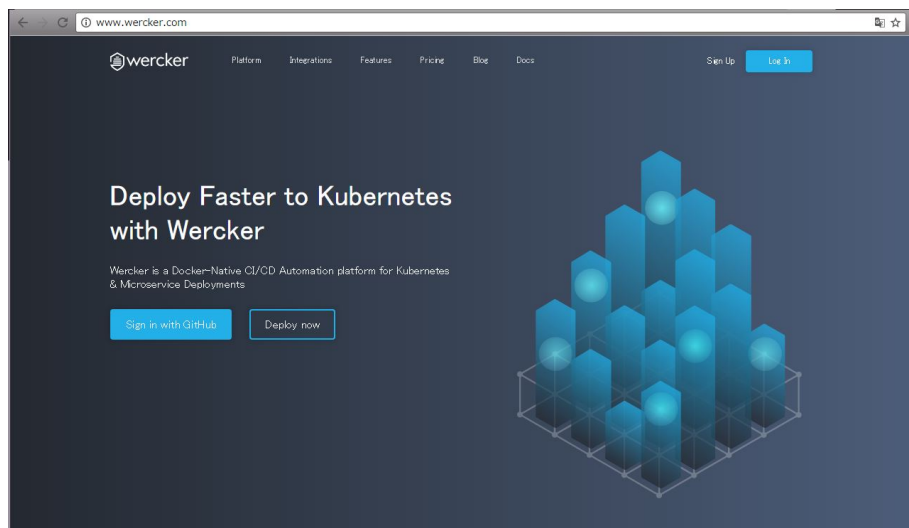


図 3.3 Wercker

Wercker とは GitHub にコードをプッシュすれば自動的にアプリのテスト・ビルドデプロイを行うことができる CI(継続的インテグレーション) サービスである。Wercker は GitHub のリポジトリに `wercker.yml` を準備そこに記述された実行環境と続行コマンドをもとにテスト/ビルドを走らせる。

以下にサンプルの wercker.yml ファイルを示す .

```
box: wercker/ruby
services:
  - mies/rethinkdb
build:
  steps:
    # Execute the bundle install step, a step provided by wercker
    - bundle-install
    # Execute a custom script step.
    - script:
        name: middleman build
        code: bundle exec middleman build --verbose
deploy:
  steps:
    # Execute the heroku-deploy, heroku details can be edited
    # online at http://app.wercker.com/
    #- heroku-deploy

    # Execute the s3sync deploy step, a step provided by wercker
    - s3sync:
        key_id: $AWS_ACCESS_KEY_ID
        key_secret: $AWS_SECRET_ACCESS_KEY
        bucket_url: $AWS_BUCKET_URL
        source_dir: build/
  after-steps:
    - hipchat-notify:
        token: $HIPCHAT_TOKEN
        room-id: id
        from-name: name
```

上から box , services , build フェーズ , deploy フェーズ , step を記述していく . box は OS と一連のパッケージがインストールされた VM である . 例えば ruby がインストールされた box , Golang がインストールされた box などがある . box は Wercker が提供するもの , もしくは Docker で構築したものを利用することができる .

3.4 Docker

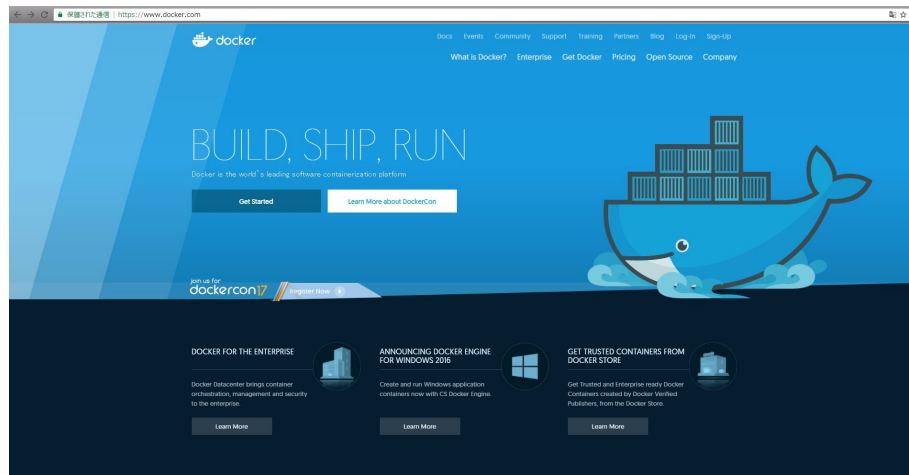


図 3.4 Docker ホームページ

Docker とは、Linux のコンテナ技術をベースに Docker 社が開発した仮想化技術である。アプリ実行環境を高速にデプロイできるコンテナ型の仮想化マシンである [4]。WindowsOS の内部に独立したアプリケーションの実行環境であるコンテナを生成することができ、リソース消費量が非常に少なく 1 台の物理サーバに多くのコンテナを稼働させられる。

3.4.1 DockerHub のアカウント登録

複数のマシンで共通の環境を共有するために、作成した環境であるコンテナを DockerHub にアップロードし公開することができる。

DockerHub のアカウント登録は以下を参照する。 <https://hub.docker.com/>

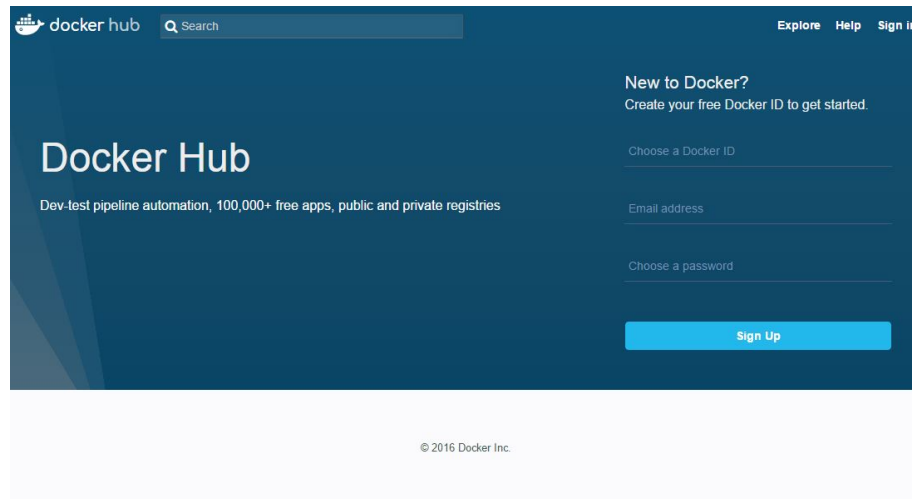


図 3.5 DockerHub ホームページ

第 4 章

開発環境構築ツールの解説

4.1 Chocolatey

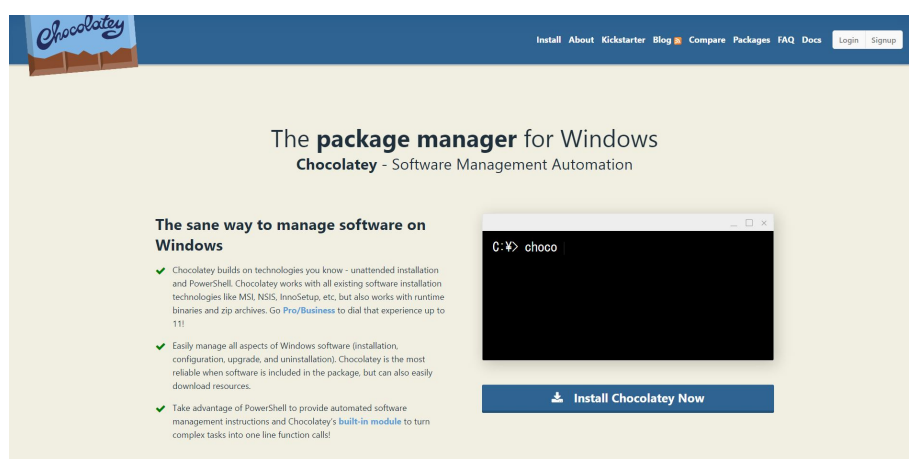


図 4.1 chocolatey

「Chocolatey」は、コマンドラインによるアプリケーションの導入や削除を実現するパッケージ管理システム。パッケージ管理システムとは、アプリケーションやコンポーネント、ライブラリなどの管理を円滑に行うための仕組み。アプリケーションやライブラリを構成するファイル群を1つの「パッケージ」ファイルにまとめ、それを「リポジトリ」へ保管し、コマンドで取得・セットアップを行う。動作に必要な外部コンポーネントがあれば、その導入も自動で行われるのが一般的で、アプリケーションのインストールやアンインストールの手間を省くことができる。

Linux ディストリビューションの多くは「apt-get」や「yum」といったパッケージ管理のためのコマンドを備えており、コマンドを入力するだけで手軽にソフトをダウンロード・インストール・アンインストールできて非常に便利。「Chocolatey」はこれを Windows 環境で実現しようというものである。

4.1.1 パッケージとは

ソフトウェアにかかわるファイル一式がまとまったものをパッケージという。「ソフトウェアにかかわるファイル」には、設定ファイルやドキュメント、プログラム本体、プログラムが動くために必要なライブラリなどが含まれます。

4.1.2 choco list [packageName]

パッケージ検索。引数がなければすべてのパッケージを表示。

4.1.3 choco list -lo [packageName]

インストール済みのパッケージ検索。引数がなければすべてのインストール済みパッケージを表示。

4.1.4 cinst [packageName]

指定パッケージのインストール。

4.1.5 cuninst [packageName]

指定パッケージのアンインストール。

4.1.6 cup

Chocolatey 本体のアップデート。

4.1.7 cup [packageName]

指定パッケージのアップデート。

4.1.8 cup all

インストール済みのパッケージを全てアップデート。

4.2 DockerToolBox

Docker Toolbox は、Docker 環境を簡単に構築するためのインストーラである。現在、Mac と Windows 用に提供されている。

Docker は、Linux 上で動作するため、Windows では、Docker を動作させるため Linux 仮想マシンを用意し、それを利用する形態をとる。

Docker Toolbox は、この Docker 環境を構築・運用するためのアプリケーションやツールをセットにして提供する。Docker Toolbox のセット内容は、以下のとおりである。

表 4.1 仮想マシンの操作

ツール	概要
Docker Client	Docker を操作するツール (docker コマンド)
Docker Machine	Docker 仮想マシンを簡単に構築・管理するためのツール
Docker Compose	マルチコンテナアプリケーション (複数のコンテナで構成するシステム)
Docker Kitematic	Docker を GUI で操作するツール
Docker Quickstart Terminal	docker コマンドが即利用できるターミナル
VirtualBox	PC の仮想化アプリケーション

4.3 Oracle VM VirtualBox

Oracle VM VirtualBox (以下, VirtualBox と表記する) は, 使用している PC マシン上に仮想的なマシンを作成し, 別の OS をインストール・実行することができるオープンソースソフトウェアである. Windows や Mac OS X, Linux 等, 様々な OS で利用することができる本研究で利用する VirtualBox の用語について説明する.

4.3.1 ホスト PC

VirtualBox をインストールした PC のこと.

4.3.2 ホスト OS

VirtualBox をインストールした PC の OS のこと. 本研究で使用するホスト PC の OS は Windows 7/8.1 である.

4.3.3 ゲスト PC (仮想マシン)

VirtualBox で作成した仮想 PC

4.3.4 ゲスト OS

VirtualBox で作成した仮想 PC にインストールした OS のこと.

第 5 章

開発環境の導入

パッケージ管理システム「Chocolatey」をインストールするためには、管理者権限でコマンドプロンプトを起動する必要がある。Windows 10 の場合、「スタート」ボタンを右クリックして、「コマンドプロンプト (管理者)」をクリックする。



図 5.1 管理者権限

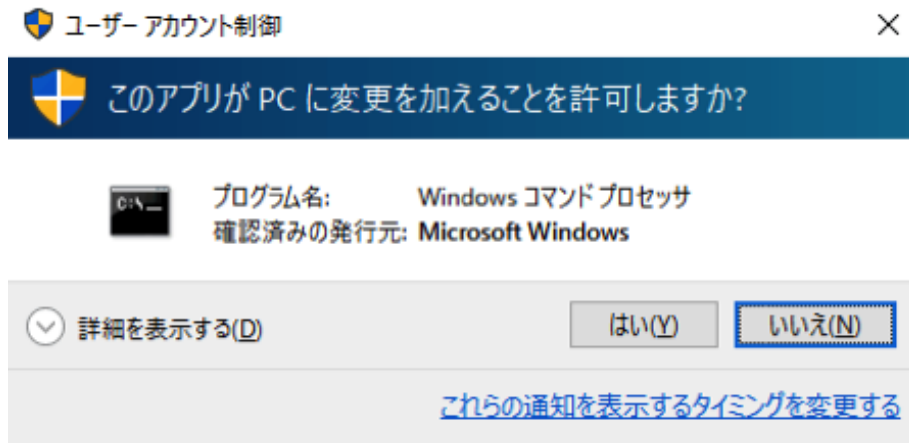


図 5.2 UAC

UAC (User Account Control) が有効になっている場合は、コマンドプロンプトを管理者権限で実行して良いかどうかプロンプトが表示されるため、「はい」をクリックする。

「chocolatey」をインストールするために、コマンドプロンプトで、以下のコマンドを実行する。

```
@powershell -NoProfile -ExecutionPolicy Bypass -Command "iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))" && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"
```

```
管理者: コマンド プロンプト
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Windows\system32>powershell -NoProfile -ExecutionPolicy Bypass -Command "iex
((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))" && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"

Mode                LastWriteTime         Length Name
-----
d----          2016/10/13      18:30             chocInstall
Getting latest version of the Chocolatey package for download.
Downloading https://chocolatey.org/api/v2/package/chocolatey/0.10.3 to C:\Users\hamano\AppData\Local\Temp\chocolatey\chocInstall\chocolatey.zip
Extracting C:\Users\hamano\AppData\Local\Temp\chocolatey\chocInstall\chocolatey.zip to C:\Users\hamano\AppData\Local\Temp\chocolatey\chocInstall...
Installing chocolatey on this machine
Creating ChocolateyInstall as an environment variable (targeting 'Machine')
  Setting ChocolateyInstall to 'C:\ProgramData\chocolatey'
WARNING: It's very likely you will need to close and reopen your shell
  before you can use choco.
Restricting write permissions to Administrators
We are setting up the Chocolatey package repository.
The packages themselves go to 'C:\ProgramData\chocolatey\lib'
(i.e. C:\ProgramData\chocolatey\lib\yourPackageName).
A shim file for the command line goes to 'C:\ProgramData\chocolatey\bin'
, and points to an executable in 'C:\ProgramData\chocolatey\lib\yourPackageName'.

Creating Chocolatey folders if they do not already exist.

WARNING: You can safely ignore errors related to missing log files when
  upgrading from a version of Chocolatey less than 0.9.9.
  'Batch file could not be found' is also safe to ignore.
  'The system cannot find the file specified' - also safe.
chocolatey.nupkg file not installed in lib.
  Attempting to locate it from bootstrapper.
PATH environment variable does not have C:\ProgramData\chocolatey\bin in it. Adding...
警告: Not setting tab completion: Profile file does not exist at
'C:\Users\hamano\Documents\WindowsPowerShell\Microsoft.PowerShell_profile.ps1'.
Chocolatey (choco.exe) is now ready.
You can call choco from anywhere, command line or powershell by typing choco.
Run choco /? for a list of functions.
You may need to shut down and restart powershell and/or consoles
  first prior to using choco.
Ensuring chocolatey commands are on the path
Ensuring chocolatey.nupkg is in the lib folder

C:\Windows\system32>
```

図 5.3 choco インストール

インストール方法は、今後変更される可能性がある。そのため、以下のコマンドが正常に動作しない場合は、Chocolatey Gallery から最新の情報を確認してください。

5.0.1 パッケージのインストール

「VirtualBox」と「vagrant」,「WinSCP」を Chocolatey を使って, コマンドプロンプトからインストールする. 「VirtualBox」と「vagrant」,「WinSCP」をインストールするために, コマンドプロンプトで, 以下のコマンドを実行する.

```
$ cinst -y atom chrome curl git github r.project rsync sourcetree vagrant  
virtualbox wget
```

5.1 Docker のインストール

Docker は Linux 上でコンテナ仮想環境を構築するツールである。これを Windows 環境で実行するには、まず Windows 上に仮想環境を構築し、そこで Linux サーバを動作させる必要がある。

Docker ToolBox をインストールすれば、Windows 上に VirtualBox による仮想環境を構築し、Docker 用の仮想マシンイメージを起動し、その上で Docker を稼働させる環境を構築できる cite4。

インストールを行うには PC の BIOS 設定で CPU の仮想化支援機能を有効にしておく必要があります。たとえば、仮想化支援機能を有効にすると、タスクマネージャの「パフォーマンス」タブの「CPU」で「仮想化」が有効になっています。設定が有効でない場合は、CPU の仮想化支援機能を有効にしてください。

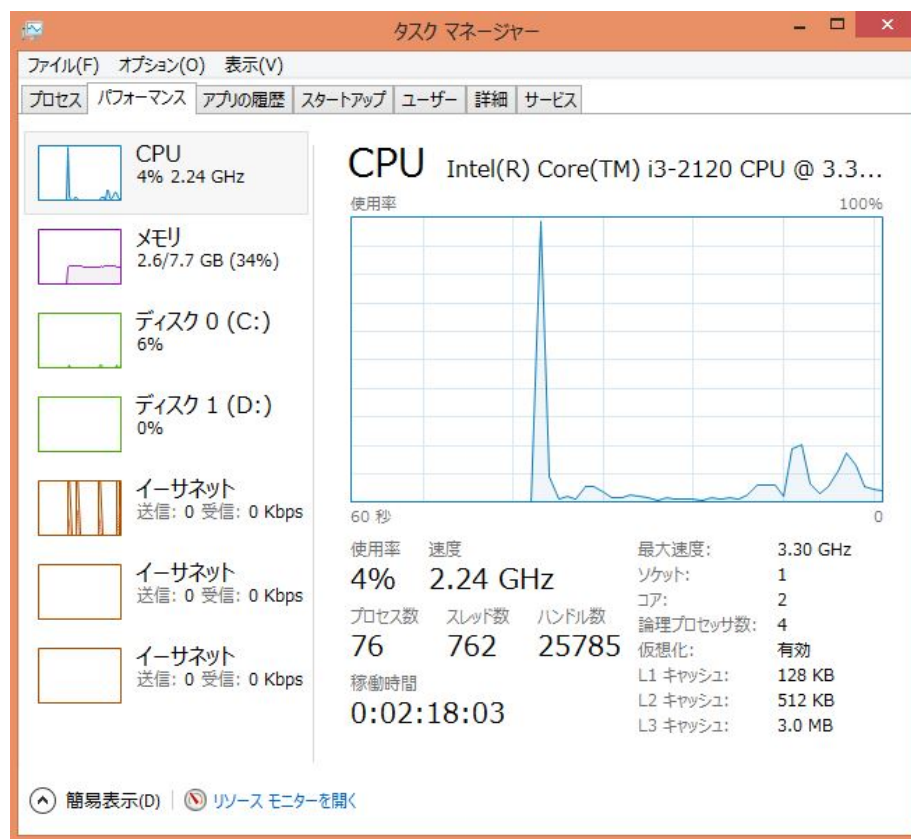


図 5.4 タスクマネージャの「パフォーマンス」タブ

BIOS 設定画面は Windows の場合、起動中の画面で「F2」キーまたは「Delete」キーを押すと、BIOS 画面が表示されます。

5.1.1 Docker Toolbox のダウンロードとインストール

Docker Toolbox のダウンロードは以下のサイトからできる．<https://www.docker.com/products/docker-toolbox>

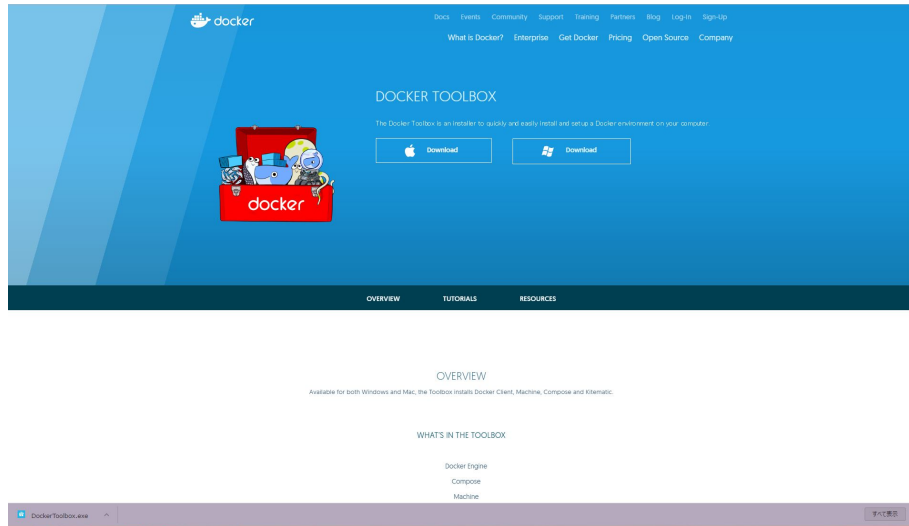


図 5.5 Docker Toolbox のダウンロード

Windows10 以降は，Docker for Windows をインストールできる．Windows10 では，OS が提供するハイパーバイザ型の仮想マシン上で Docker が利用できるようになった．それ以前は，Docker Toolbox により VirtualBox 上の Linux に DockerEngine をインストールし，ホスト上のコマンドラインツールからアクセスするという手段を取る．

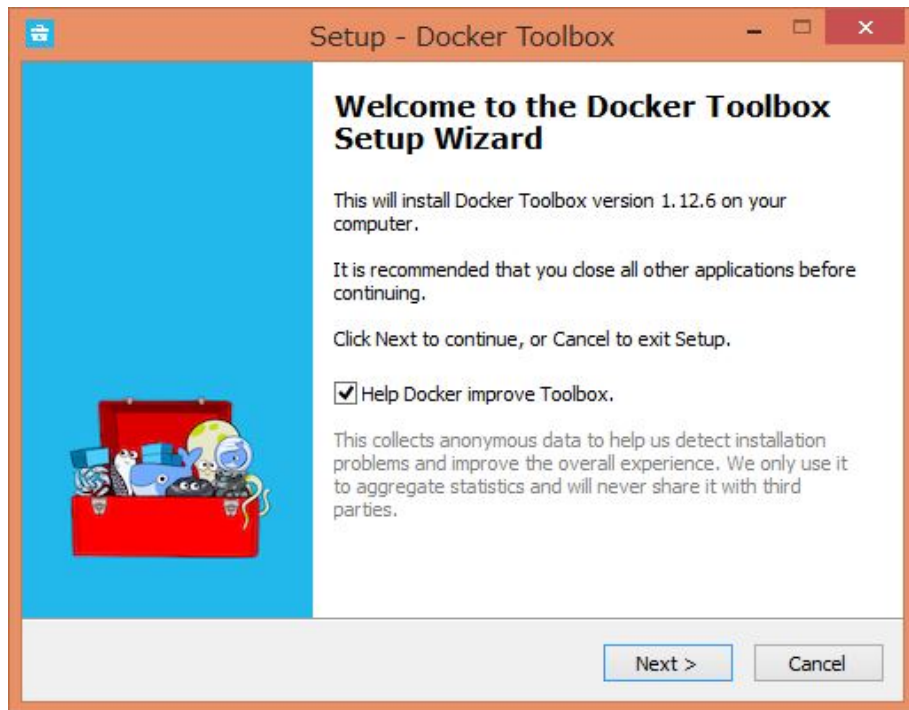


図 5.6 セットアップウィザードの起動

Next を押す .

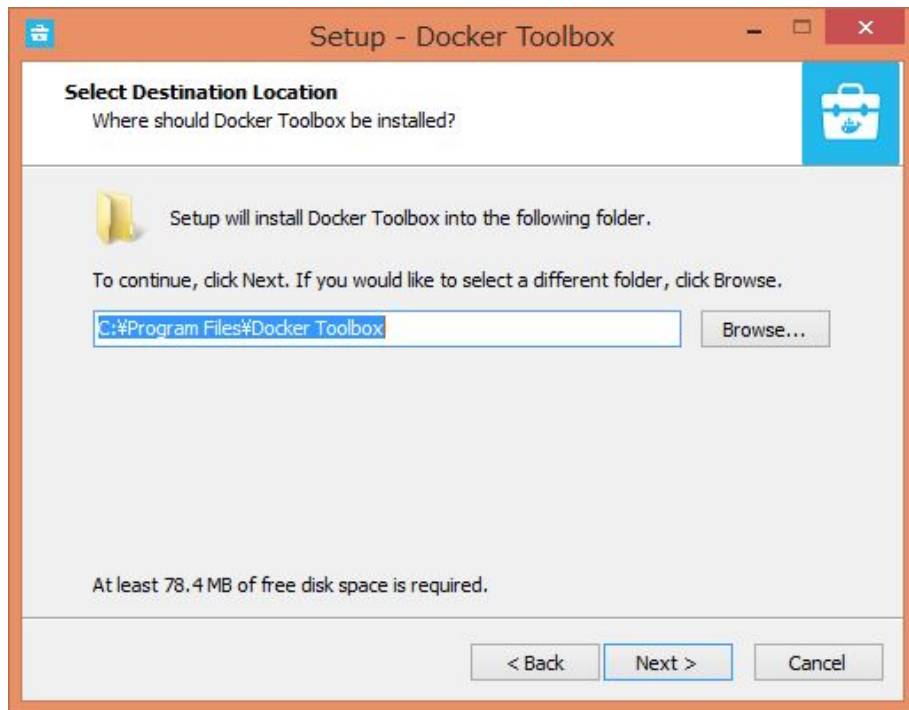


図 5.7 インストールフォルダの選択

Next を押す .

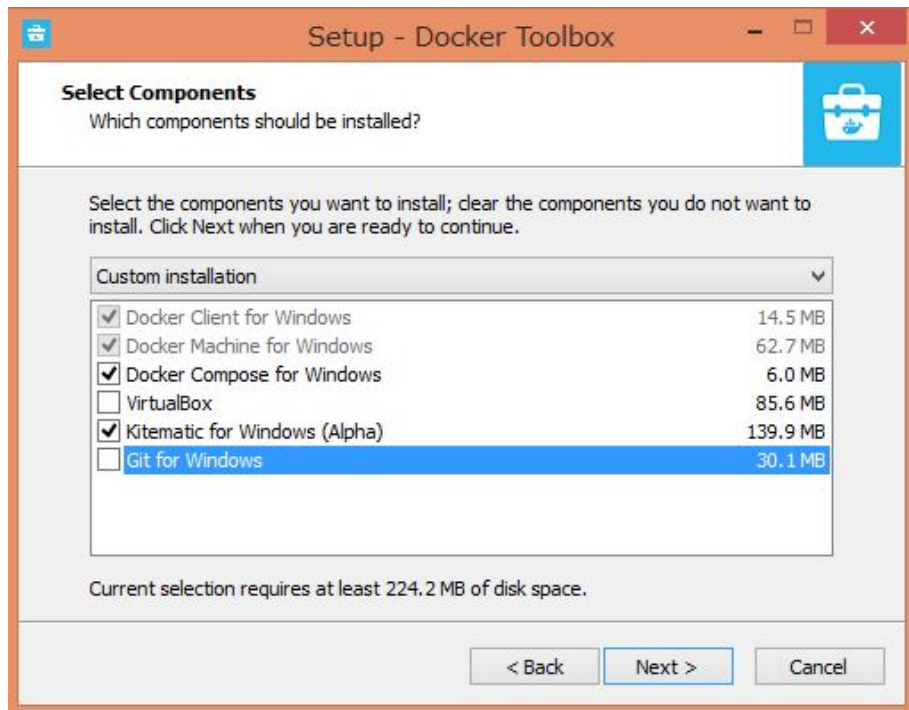


図 5.8 インストールするコンポーネントの選択

VirtualBox と Git for Windows を既にインストールしている場合はチェックを外す．インストールしていない場合はすべてチェックする．

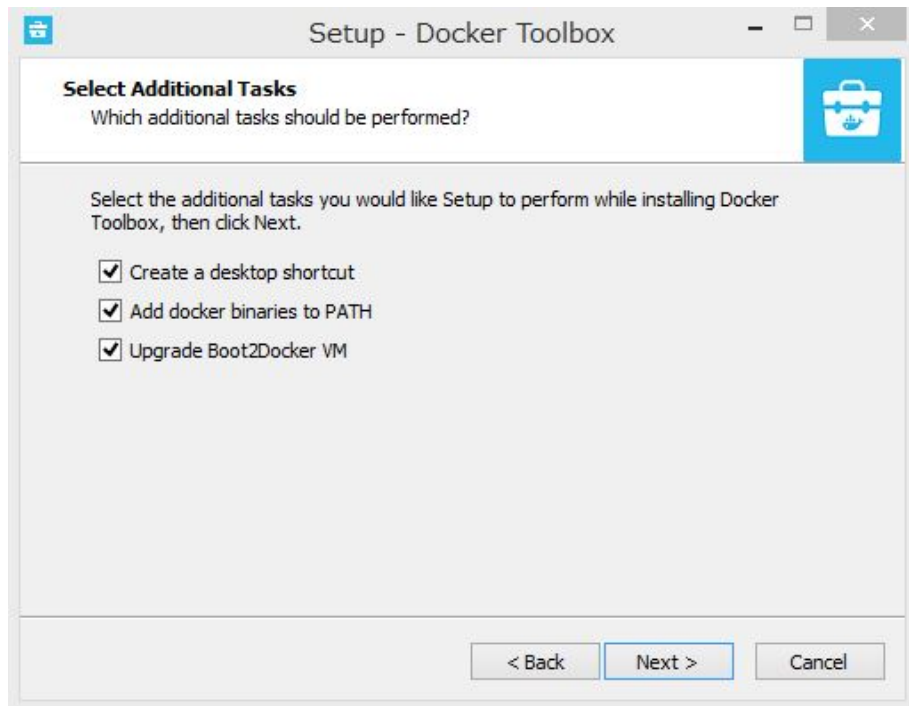


図 5.9 インストール後の設定

next を押す .

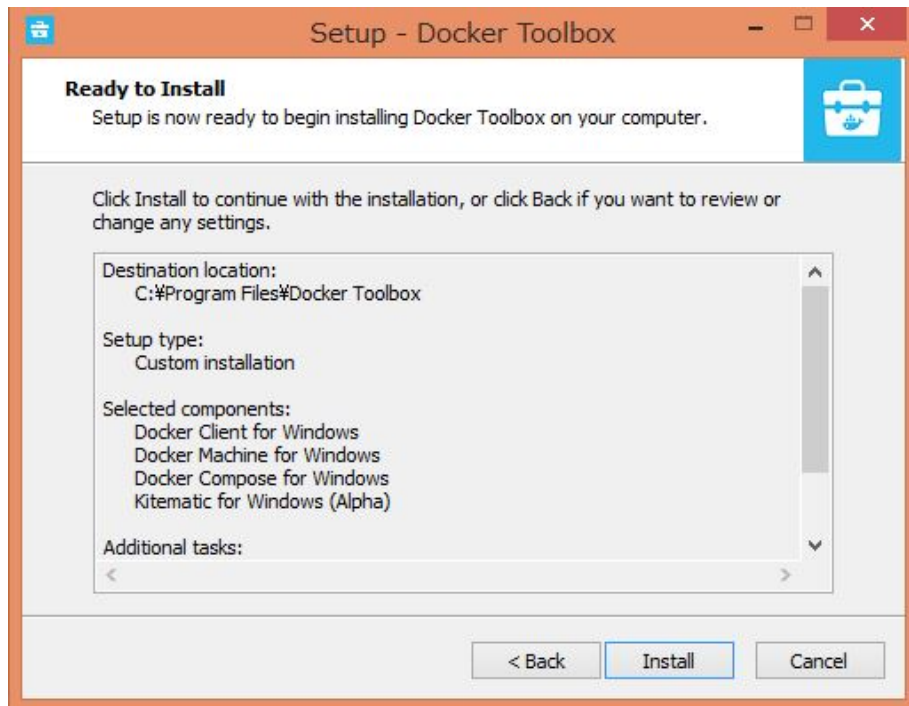


図 5.10 インストール最終確認

インストールが完了したらデスクトップに「Kitematic(Alpha)」と「Docker Quickstart Terminal」と「Oracle VM VirtualBox」の3つのアイコンが作成されます。まず、「Docker Quickstart Terminal」アイコンをダブルクリックしてください。Docker Toolbox が Docker を動作させるための仮想環境を構築します。しばらく時間がかかりますが、処理が完了すると次のようなコンソール画面が表示される。

以下コマンドにより，default という名前の仮想環境に対して，SSH で接続する．

```
$ docker-machine ssh default
```

[illegible]

図 5.12 仮想環境への接続

以上で Docker 環境が整った。

インストールした Docker が正しく動作するかを確認するため、Docker コンテナを作成し、コンソール上に "Hello world" の文字を echo 表示する。Docker コンテナを作成/実行するときは、docker run コマンドを使う。

```
$ docker run ubuntu:latest /bin/echo 'Hello world'
```

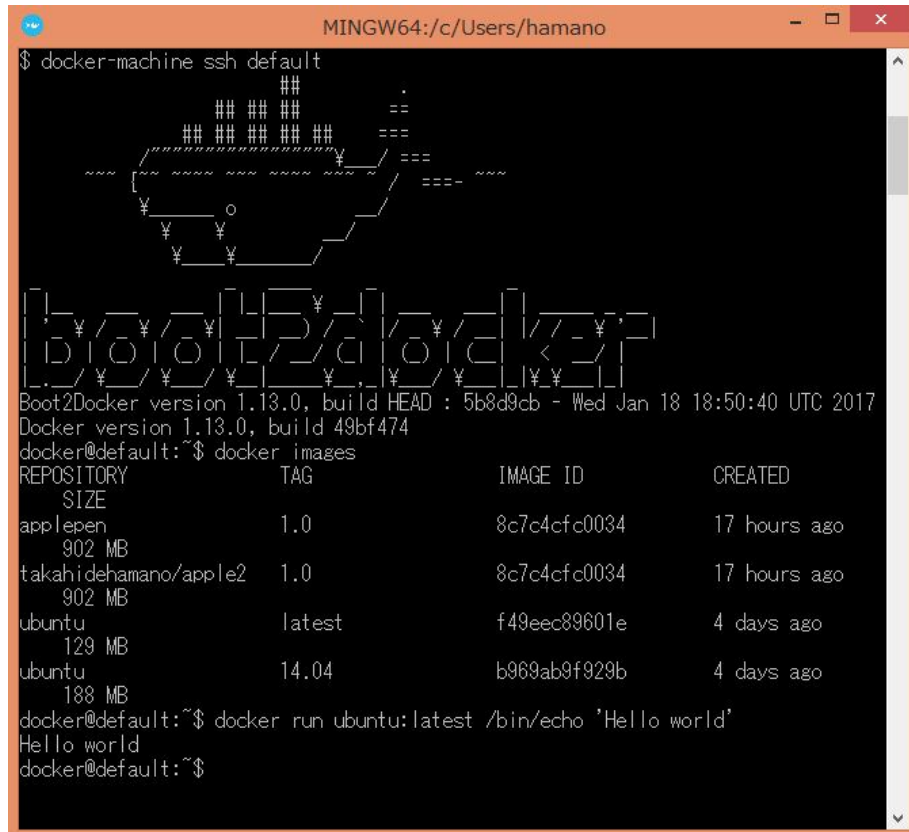


図 5.13 Helloworld の実行結果

5.1.3 ベースイメージの取得

```
$ docker pull ubuntu
```

以下のコマンドで取得したイメージを確認する .

```
$ docker images
```

以下のような内容が出力される .



```
docker@default:~$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
SIZE
ubuntu               latest              f49eec89601e        3 days ago
129 MB
ubuntu               14.04              b969ab9f929b        3 days ago
188 MB
docker@default:~$
```

図 5.14 docker images 実行

5.1.4 Dockerfile 作成

イメージを作成するため Dockerfile を作成しビルドする。Dockerfile は、Shell 形式で記述する。

```
$ mkdir sample
$ cd sample

$ vi Dockerfile
```

Dockerfile は以下になる。

```
# Java 1.8 & RedPen Dockerfile
# https://github.com/kemsakurai/docker-redpen-UTF-8

# Pull base image.
FROM ubuntu:14.04

# maintainer details
MAINTAINER Ken Sakurai "sakurai.kem@gmail.com"

# Set locale
RUN locale-gen ja_JP.UTF-8
ENV LANG ja_JP.UTF-8
ENV LANGUAGE ja_JP:ja
ENV LC_ALL ja_JP.UTF-8

# Accept Oracle license before installing java
RUN echo debconf shared/accepted-oracle-license-v1-1 select true | debconf-set-selections
RUN echo debconf shared/accepted-oracle-license-v1-1 seen true | debconf-set-selections

# Install java
RUN apt-get update
RUN apt-get -y install software-properties-common
RUN add-apt-repository -y ppa:webupd8team/java
RUN apt-get update
RUN apt-get -y install oracle-java8-installer oracle-java8-set-default
RUN echo "export JAVA_HOME=/usr/lib/jvm/java-8-oracle" >> /etc/environment

# Download Redpen
WORKDIR /tmp
RUN wget -q https://github.com/redpen-cc/redpen/releases/download/redpen-1.5.5/redpen-1.5.5.tar.gz -O - | tar xz && \
    cp -av redpen-distribution-1.5.5/* /usr/local/ && \
    rm -rf redpen-distribution-1.5.5

RUN export PATH=$PATH:/usr/local/bin
WORKDIR /data

CMD ["/usr/local/bin/redpen"]
```

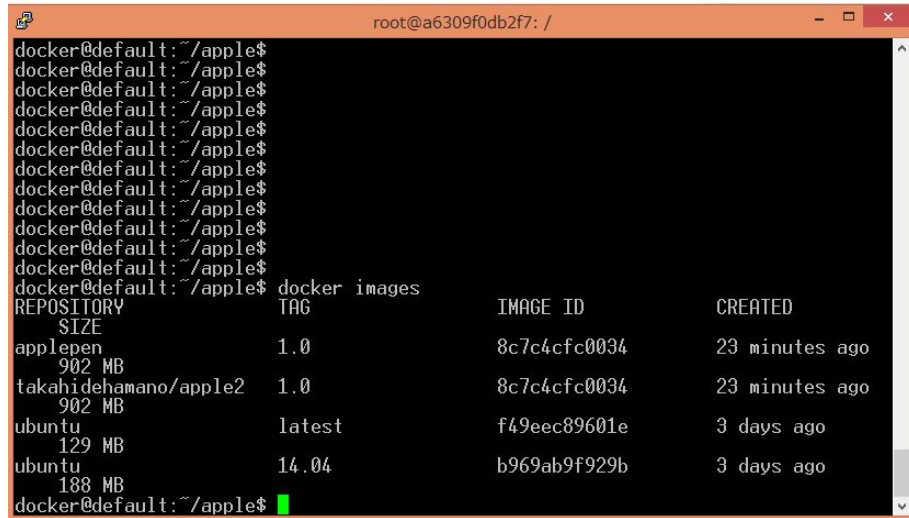
5.1.5 Dockerfile から Docker イメージの作成

```
$ docker build -t applepen:1.0 .
```

docker image コマンドによるイメージの確認

```
$ docker image
```

コマンドを実行すると以下の applepen イメージが作成される .



```
root@a6309f0db2f7: /
docker@default:~/apple$
docker@default:~/apple$
docker@default:~/apple$
docker@default:~/apple$
docker@default:~/apple$
docker@default:~/apple$
docker@default:~/apple$
docker@default:~/apple$
docker@default:~/apple$
docker@default:~/apple$
docker@default:~/apple$
docker@default:~/apple$
docker@default:~/apple$
docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
applepen             1.0                8c7c4cfc0034       23 minutes ago
takahidehamano/apple2 1.0                8c7c4cfc0034       23 minutes ago
ubuntu               latest             f49eec89601e       3 days ago
ubuntu               14.04             b969ab9f929b       3 days ago
docker@default:~/apple$
```

図 5.15 docker image コマンド実行

5.1.6 DockerHub ログイン

以下のコマンドを実行する .

```
$ docker login
Username登録したユーザ名:
Password登録したパスワード:
```

Username と Password を入力しログインする .

ログインに成功すると Login Succeeded と表示される .

5.1.7 DockerHub にイメージを push

```
$ docker push takahidehamano/applepen:latest
```

DockerHub にログインし，リポジトリ一覧を確認する．Takahidehamano/applepen が作成されていることがわかる．

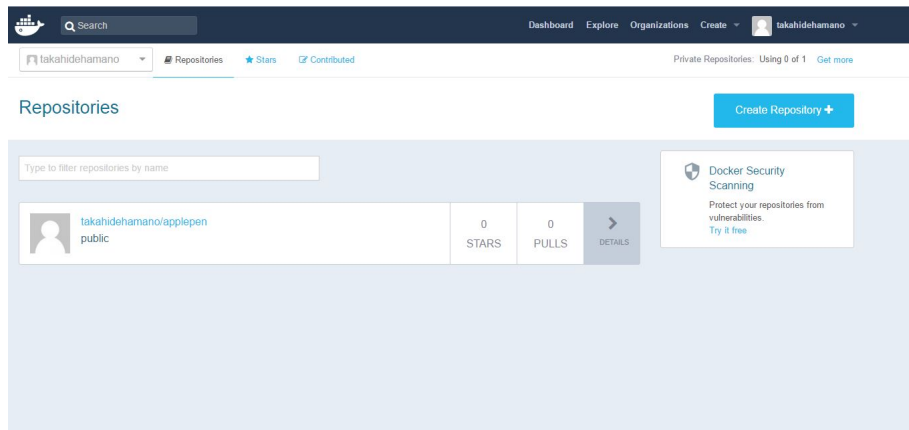


図 5.16 DockerHub

第 6 章

自動化環境の構築

6.1 GitHub の設定

6.1.1 GitHub のリポジトリ

GitHub のリポジトリを作成する．作成したリポジトリに文書をアップロードすることで文書チェックの自動化がされることが目的である．そのため `Wercker.yml` という Wercker を動かすためのコマンドが書かれたファイルと `redpen-conf-ja.xml` という文書チェックプログラム RedPen の設定が書かれたファイルを GitHub リポジトリに用意する．

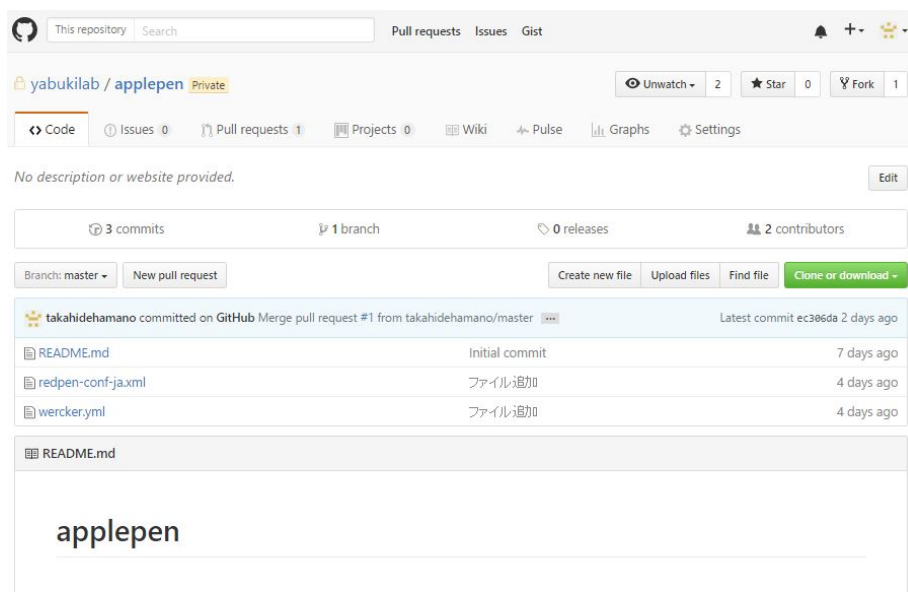


図 6.1 GitHub リポジトリ

wercker.yml の設定を以下に示す .

```
# This references a standard debian container from the
# Docker Hub https://registry.hub.docker.com/_/debian/
# Read more about containers on our dev center
# http://devcenter.wercker.com/docs/containers/index.html
box: takahidehamano/redpen
# You can also use services such as databases. Read more on our dev center:
# http://devcenter.wercker.com/docs/services/index.html
# services:
#   - postgres
#   http://devcenter.wercker.com/docs/services/postgresql.html
#
#   - mongodb
#   http://devcenter.wercker.com/docs/services/mongodb.html

# This is the build pipeline. Pipelines are the core of wercker
# Read more about pipelines on our dev center
# http://devcenter.wercker.com/docs/pipelines/index.html
build:
  # Steps make up the actions in your pipeline
  # Read more about steps on our dev center:
  # http://devcenter.wercker.com/docs/steps/index.html
  steps:
    - script:
        name: check
        code: |

            redpen -c ./redpen-conf-ja.xml -f latex  draft.tex
```

draft.tex という tex の標準ファイルがリポジトリに push されることで check という Script が動作する仕組みになっている .

redpen.conf-ja.xml を以下に記述する .

```
<redpen-conf lang="ja" type="zenkaku2">
  <validators>
    <validator name="SentenceLength">
      <property name="max_len" value="100"/>
    </validator>
    <validator name="CommaNumber" />
    <validator name="SuccessiveWord" />
    <validator name="KatakanaSpellCheck"/>
    <validator name="InvalidExpression"/>
    <validator name="JapaneseStyle"/>
    <validator name="DoubleNegative" />
    <validator name="FrequentSentenceStart" />
    <validator name="JapaneseAmbiguousNounConjunction" />
    <validator name="JapaneseNumberExpression" />
    <validator name="SuccessiveSentence" />
    <validator name="DoubledConjunctiveParticleGa" />

    <validator name="DuplicatedSection" />
    <validator name="WordFrequency" />
    <validator name="ParenthesizedSentence" />

  </validators>
</redpen-conf>
```

RedPen が提供する設定を以下に示す .

6.1.2 SentenceLength

SentenceLength は入力文書に存在する各文（センテンス）の長さに関する規約が守られているかを検査します。具体的には文書内の各文が指定された最大文長より長い場合、エラーを出力します。

6.1.3 InvalidExpression

入力に不正な表現（単語や句）が存在するか検査します。不正な表現が存在するとInvalidExpression はエラーを出力します。

6.1.4 InvalidWord

InvalidWord は入力文に不正な単語が存在するかを検査します。

6.1.5 SpaceBeginningOfSentence

SpaceBeginningOfSentence は文（センテンス）間に半角スペースが存在するかを検査します。

6.1.6 CommaNumber

CommaNumber は一文（センテンス）で利用されるコンマの数が指定された最大数よりも多いときにエラーを出力します。

6.1.7 WordNumber

WordNumber は一文内の単語数を検査します。文内の単語数が指定した数よりも大きいとき、WordNumber はエラー出力します。

6.1.8 SuggestExpression

SuggestExpression は InvalidExpression と同様に動作します。入力文で不正な表現が使用されているとエラーを出力します。出力されるエラーには利用すべき正しい表現が含まれます。

6.1.9 InvalidSymbol

シンボルによっては代替のシンボルが存在します。たとえばクエスチョンマーク ? (0x003F) は代替のシンボル ? (0xFF1F) が Unicode に登録されています。InvalidSymbol は入力文で不正なシンボルが利用されているとエラーを出力します。

6.1.10 SymbolWithSpace

シンボルによっては前もしくは後にスペースが必要です。たとえば、左括弧 " (" の前には、かならず半角スペースを置くという規約がありえます。スペースに関する設定は設定ファイルの symbols ブロックで指定します。

6.1.11 KatakanaEndHyphen

カタカナ単語の語尾が規約（JIS Z8301、G.6.2.2 b、G.3.）に従っているかを検査します。具体的には以下のルールが適用されます。

- a: 単語が三文字もしくはそれ以上の場合には、ハイフンで単語は終わらない。
- b: 単語が二文字もしくはそれ以下の場合には、単語はハイフンで終わってもよい。
- c: 単語が複合語の場合には各々の部分単語について条件が適用される。
- d: a から c のルールにおいて、拗音をのぞきハイフンは一文字としてカウントされます。

6.1.12 KatakanaSpellCheck

KatakanaSpellCheck はカタカナ単語のスペリングを検査します。対象となるカタカナ単語に類似する単語が存在した場合、エラーを出力します。たとえば、"インデックス"と"インデクス"が同一文書で利用されているときにエラーを出力します。

6.1.13 SectionLength

SectionLength は節で利用できる単語の数を指定します。

6.1.14 ParagraphNumber

ParagraphNumber は節の中に存在してよいパラグラフの最大数を指定します。

6.1.15 ParagraphStartWith

ParagraphStartWith はパラグラフの開始部分が指定された規約に従っているかをチェックします。

6.1.16 SpaceBetweenAlphabeticalWord

アルファベット単語の前後に空白が存在するかをチェックします。単語が空白によって区切られない言語（日本語、中国語など）で執筆するときに使用します。SpaceBetweenAlphabeticalWord はアルファベット単語の前後に空白が存在しないとエラーを出力します。

6.1.17 DoubledWord

DoubledWord は一文内で二回以上、同一の単語が使用されたときにエラーを出力します。たとえば、以下の文では良いが二回使われているので、エラーを出力します。

6.1.18 SuccessiveWord

SuccessiveWord は同一の単語が連続して使用されたときにエラーを出力します。

たとえば入力文書に以下の文が含まれていると、エラーを出力します。以下の文は、言語という単語を連続（書き誤り）で使用しています。

6.1.19 DuplicatedSection

文書中に著しく類似する節が存在すると、エラーを出力します。類似度はコサイン距離によって計算されます。

6.1.20 JapaneseStyle

ですます調とである調が混在して利用された場合、エラーを出力します。

6.1.21 DoubleNegative

DoubleNegative は入力文書に二重否定が使用されているとエラーを出力します。

6.1.22 FrequentSentenceStart

多くの文が同一表現から開始されているときにエラーを出力します。

6.1.23 ParenthesizedSentence

ParenthesizedSentence は括弧に関する規約を検査します。検査するポイントは以下の 3 つです。

一文内で使用される括弧の使用頻度

ネストされた括弧が存在するか

括弧の開始位置から終了位置までの長さ

6.1.24 JavaScript

JavaScript は機能拡張スクリプトを実行します。

6.1.25 DoubledJoshi

DoubledJoshi は同一の助詞が一文で二回以上、利用されているとエラーを出力します。

6.1.26 HankakuKana

文書中に半角カナ文字が利用されているとエラーを出力します。

6.1.27 Okurigana

送りがなの使い方が正しくない場合にエラーを出力します。

6.1.28 VoidSection

節に段落や文が 1 つも含まれていない場合にエラーを出力します。

6.1.29 GappedSection

GappedSection は節（章）の大きさにギャップがあるとエラーを出力します。たとえば、以下のテキストでは 1 節の直下に 1.1.1 節があります。つまり 1 節と 1.1.1 節の間にギャップが存在します。ギャップを埋めるためには、1.1.1 節の前に 1.1 節が存在するべきです。

6.1.30 LongKanjiChain

長すぎる熟語（漢字の連続）を検出し、エラーを出力します。

6.1.31 SectionLevel

深すぎる節を検出しエラーを出力します。

6.1.32 JapaneseAmbiguousNounConjunction

日本語文に含まれる、曖昧な名詞接続のパターンを検出しエラーを出力します。ここで、曖昧な名詞接続のパターンとは、格助詞の "の" + 名詞連続 + 各助詞の "の" です。たとえば以下の文は、曖昧な名詞接続を含んでいます。

6.1.33 JapaneseAnchorExpression

日本語文において、章節の参照が一貫したスタイルになっているか検査します。

6.1.34 SuccessiveSentence

SuccessiveSentence は同一の文が二回連続で使用されるとエラーを出力します。

たとえば入力文書に以下のパラグラフが含まれていると、エラーを出力します。

いつも感じるのです。日本語はよい言語だ。日本語はよい言語だ。それでも別の言語もよい点が色々あります。上記のパラグラフには同一の文 "日本語はよい言語だ。" が二回連続で出現しています。

6.1.35 DoubledConjunctiveParticleGa

一文に二回以上、接続助詞の が が出現するとエラーを出力します。たとえば、Doubled-ConjunctiveParticleGa は以下の文に対してエラーを出力します。

6.2 Wercker の設定

<http://www.wercker.com/> / GitHub のアカウントを用いログインする。

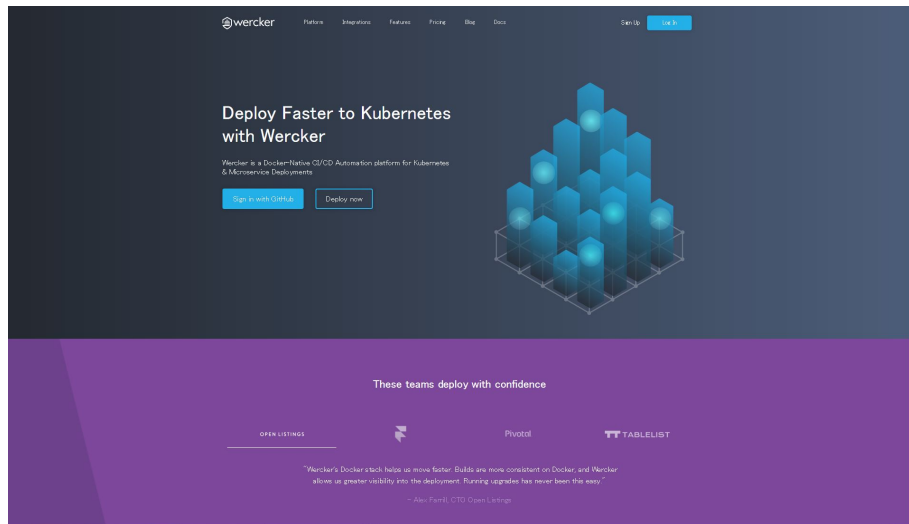


図 6.2 自動化ツール Wercker と GitHub の連携

上のタブの Create から Application をクリックする .

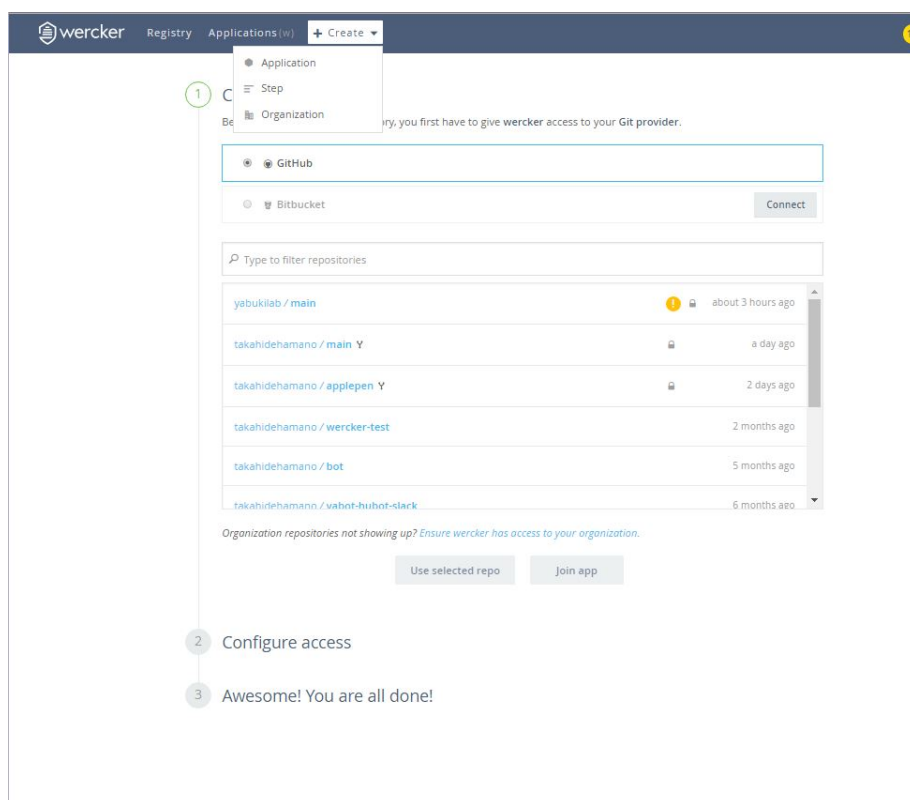


図 6.3 Wercker Create

GitHub を選択し , 表示されているリポジトリの中から takahidehamano/applepen を選択する .

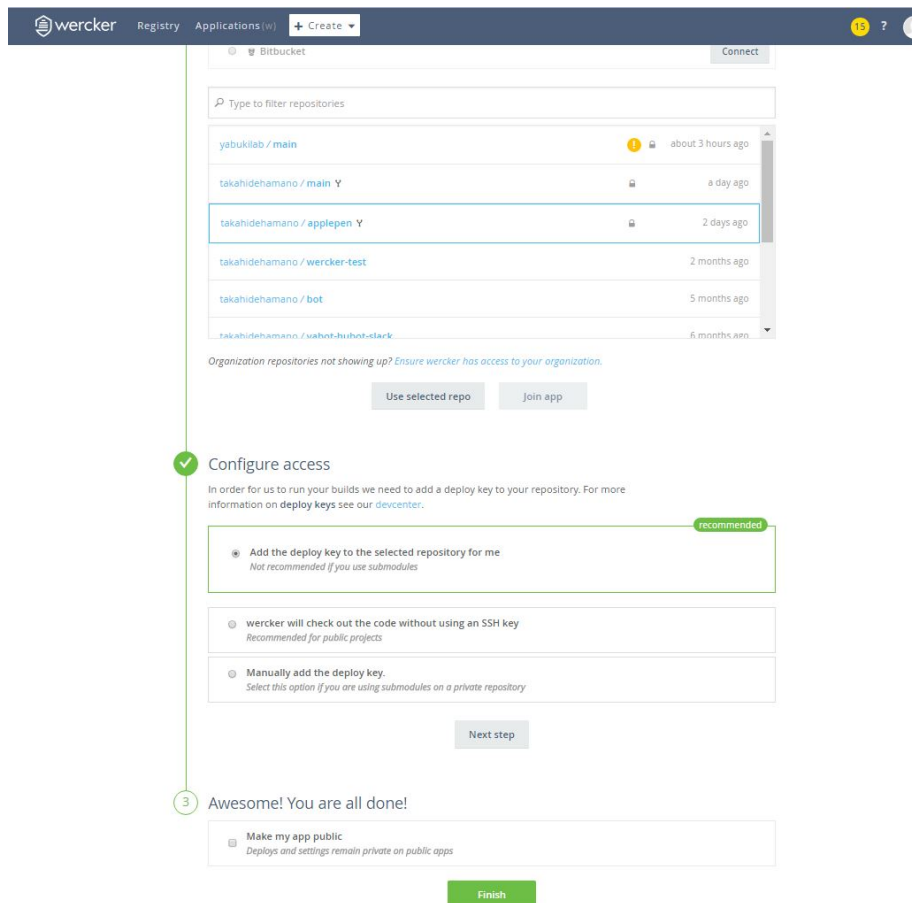


図 6.4 Wercker Create

Finish ボタンを押すことで Wercker アプリケーションが構築される。

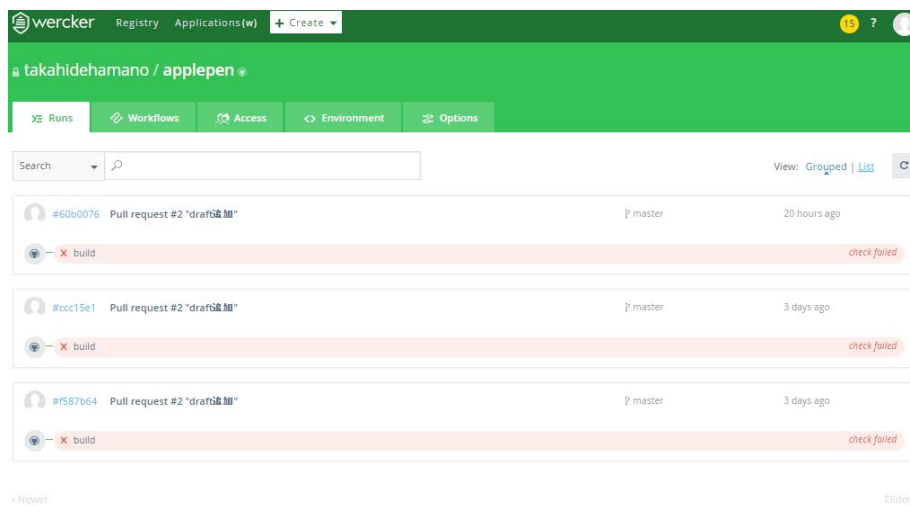


図 6.5 Wercker アプリケーション完成

第 7 章

GitHub に文書提出

GitHub リポジトリを Fork する．以下の画面の右上に Fork ボタンがある．

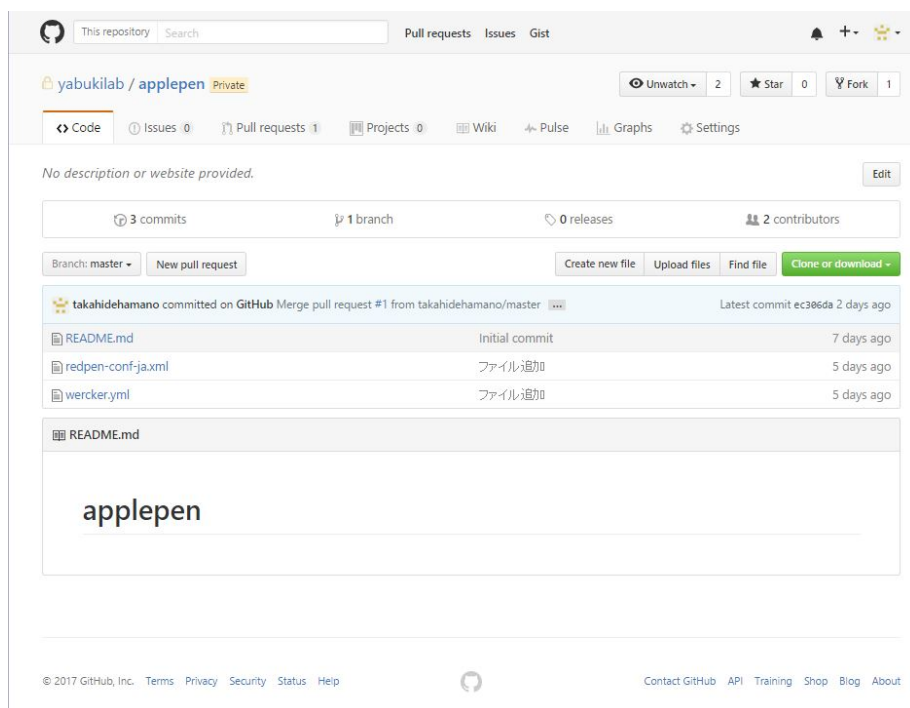


図 7.1 GitHub リポジトリ

次に Clone or download を押す．ボタンは真ん中の右側にある．

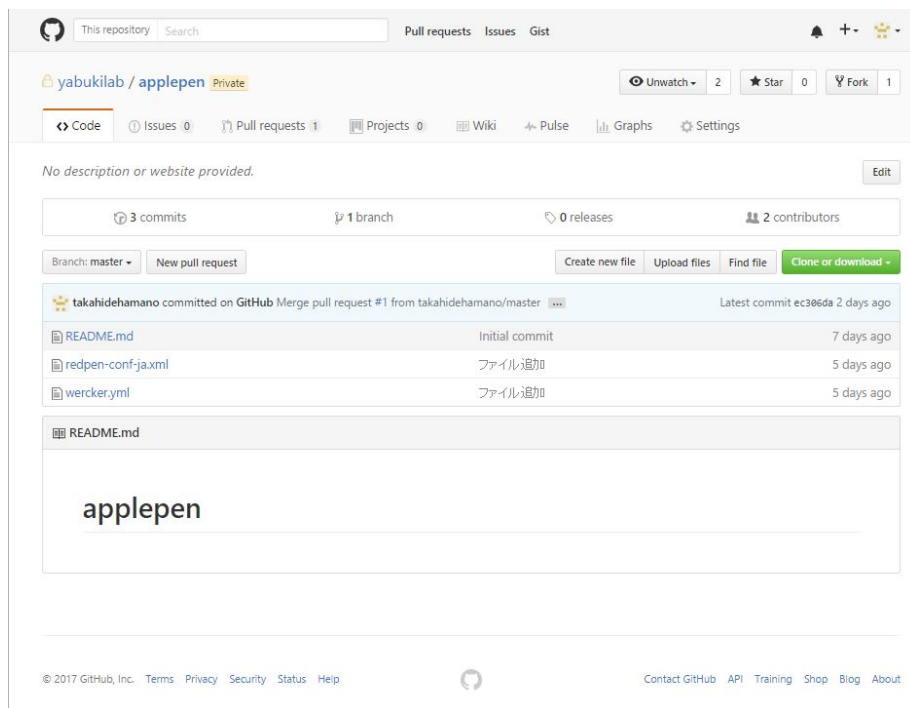


図 7.2 GitHub リポジトリ

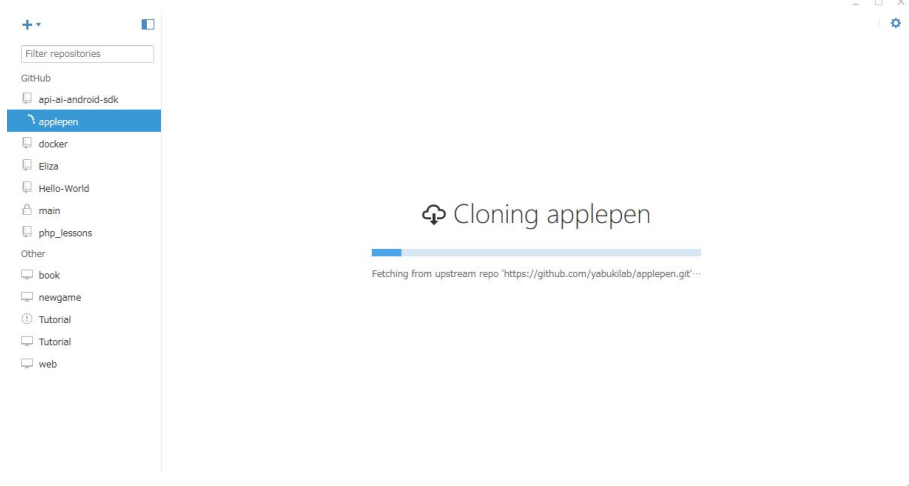


図 7.3 GitHub Clone

GitHub アプリケーションに applepen が clone される .

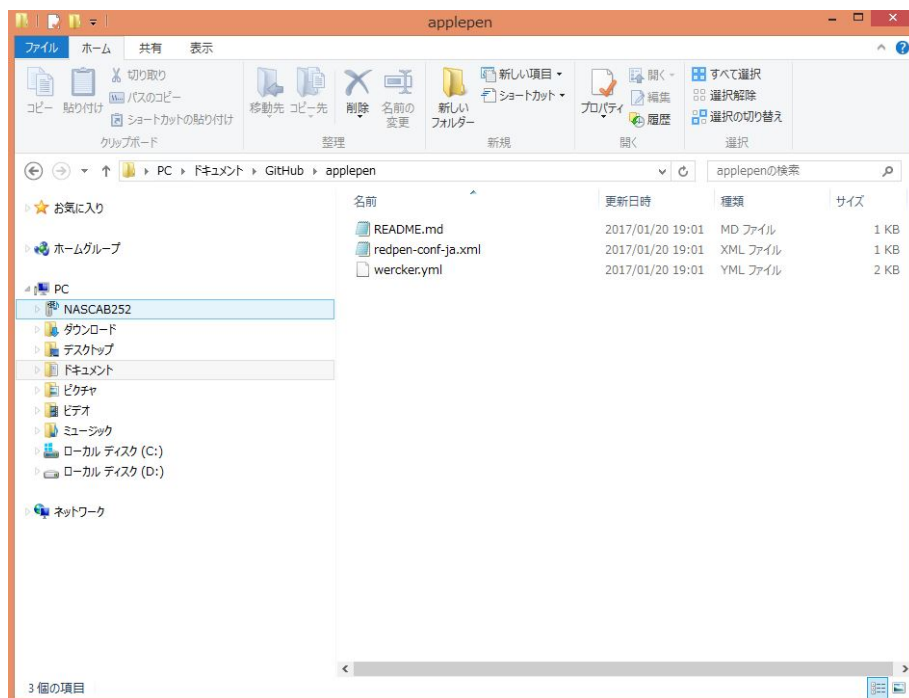


図 7.4 ローカルリポジトリ

ローカルに applepen リポジトリのファイルが保存される。

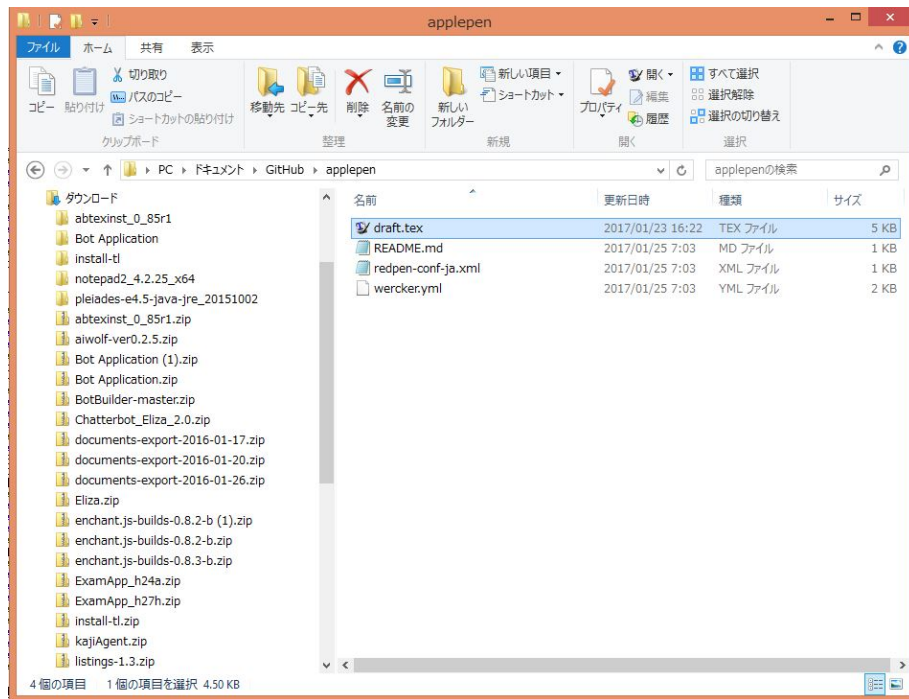


図 7.5 draft.tex 追加

ローカルに保存されたファイルの中に TeX ファイルを保存する。

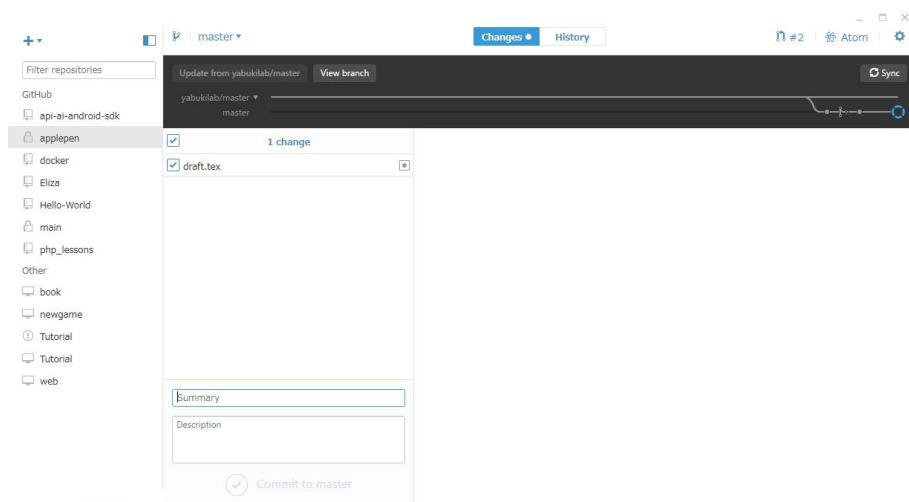


図 7.6 GitHub アプリケーション

GitHub アプリケーションを開く

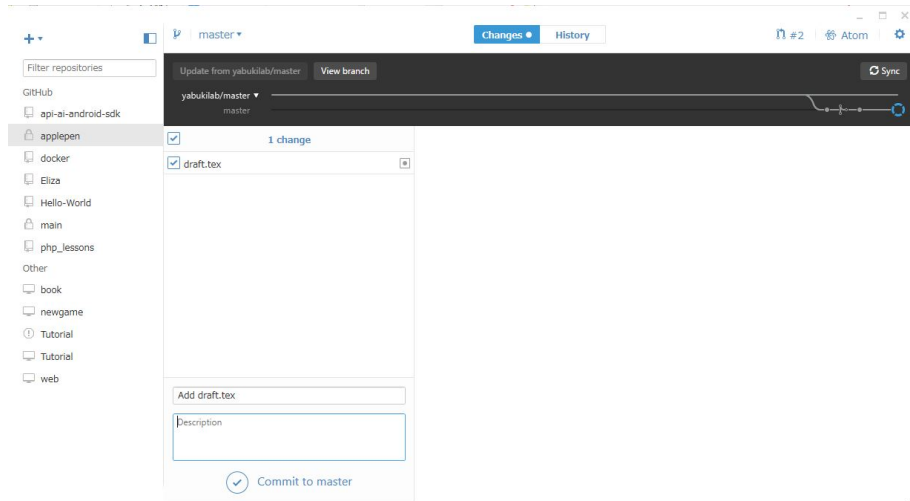


図 7.7 Commit

先のローカルリポジトリの変更をコメントをつけて Commit する .

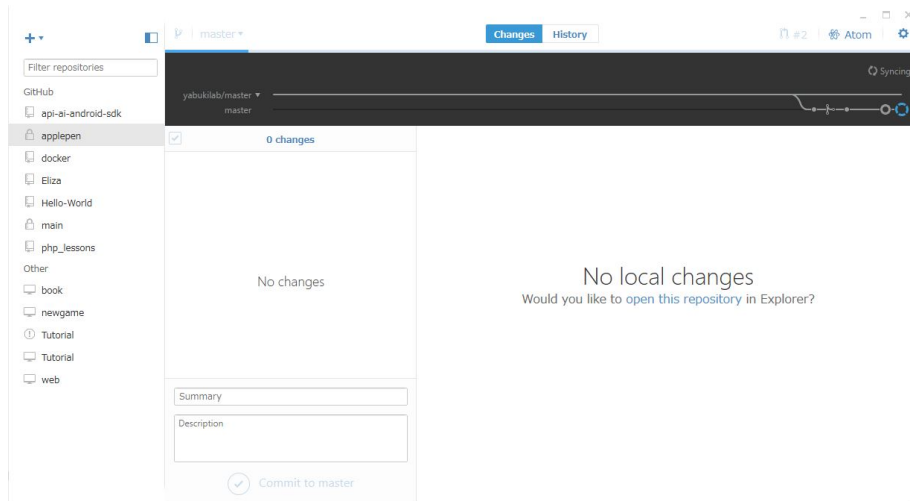


図 7.8 Sync

Sync を行う．画面の右上にボタンがある．

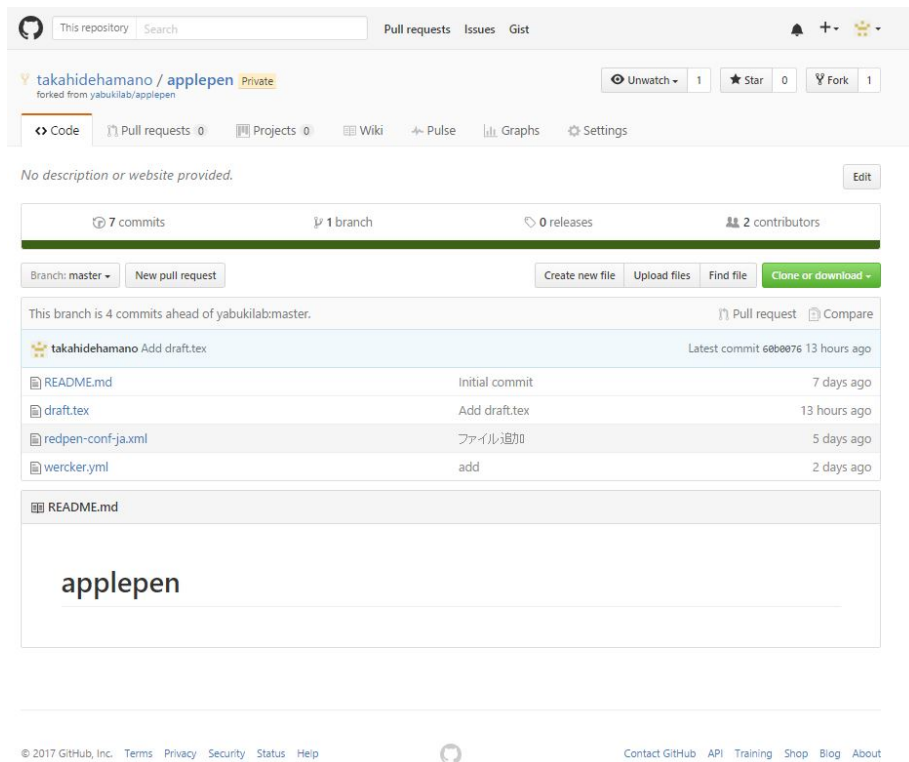


図 7.9 PullRequest

最初のリポジトリの画面から Fork 前のリポジトリに移る .

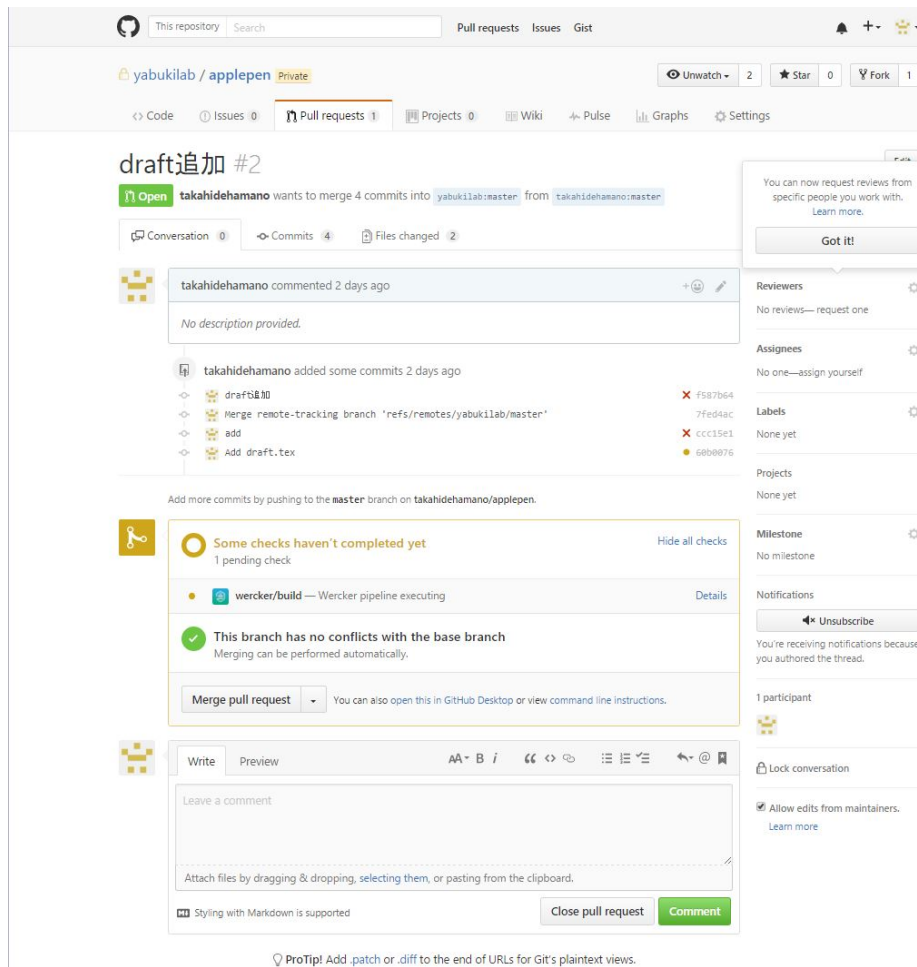


図 7.10 PullRequest

PullRequest タブを開き NewPullRequest を行う．そうすることで wercker が立ち上がり自動で文書チェックプログラムが起動する．文章チェックプログラムの結果の表示には Details を押す．

wercker Registry Applications(w) + Create

takahidehamano / applepen

Runs Workflows Access Environment Options

Pull request #2 "draft追加" master build Retry

Steps

- ✓ get code 1 second
- ✓ setup environment 58 seconds
- ✓ wercker-init 1 second
- ✓ echo start 2 seconds
- ✗ check 4 seconds

Command cancelled due to error

```
export WERCKER_STEP_ROOT="/pipeline/script-b330128c-c437-4fb8-83e0-a5a0fff42adc"
export WERCKER_STEP_ID="script-b330128c-c437-4fb8-83e0-a5a0fff42adc"
export WERCKER_STEP_OWNER="wercker"
export WERCKER_STEP_NAME="script"
export WERCKER_REPORT_NUMBERS_FILE="/report/script-b330128c-c437-4fb8-83e0-a5a0fff42adc/numbers.ini"
export WERCKER_REPORT_MESSAGE_FILE="/report/script-b330128c-c437-4fb8-83e0-a5a0fff42adc/message.txt"
export WERCKER_REPORT_ARTIFACTS_DIR="/report/script-b330128c-c437-4fb8-83e0-a5a0fff42adc/artifacts"
source "/pipeline/script-b330128c-c437-4fb8-83e0-a5a0fff42adc/run.sh" < /dev/null
[2017-01-24 22:16:23.470] [INFO ] cc.redpen.Main - Configuration file: /pipeline/source/./redpen-conf-japanese.ini
[2017-01-24 22:16:23.498] [INFO ] cc.redpen.config.ConfigurationLoader - Loading config from specified file
[2017-01-24 22:16:23.523] [INFO ] cc.redpen.config.ConfigurationLoader - Succeeded to load configuration from /pipeline/source/./redpen-conf-japanese.ini
[2017-01-24 22:16:23.523] [INFO ] cc.redpen.config.ConfigurationLoader - Language is set to "ja"
[2017-01-24 22:16:23.523] [INFO ] cc.redpen.config.ConfigurationLoader - Deprecated: use "variant" attribute instead of "zenkaku2"
[2017-01-24 22:16:23.524] [INFO ] cc.redpen.config.ConfigurationLoader - Variant is set to "zenkaku2"
[2017-01-24 22:16:23.526] [INFO ] cc.redpen.config.ConfigurationLoader - No "symbols" block found in the configuration file
[2017-01-24 22:16:23.535] [INFO ] cc.redpen.config.SymbolTable - "ja" is specified.
[2017-01-24 22:16:23.535] [INFO ] cc.redpen.config.SymbolTable - "zenkaku2" variant is specified
[2017-01-24 22:16:24.413] [INFO ] cc.redpen.parser.SentenceExtractor - "[. , ? , !]" are added as a end of sentence symbols.
[2017-01-24 22:16:24.413] [INFO ] cc.redpen.parser.SentenceExtractor - "[', ']" are added as a right quote symbols.
[2017-01-24 22:16:24.801] [INFO ] org.reflections.Reflections - Reflections took 108 ms to scan 1 urls, classes in scope:
[2017-01-24 22:16:24.941] [INFO ] cc.redpen.util.DictionaryLoader - Succeeded to load katakana word dictionary
[2017-01-24 22:16:24.942] [INFO ] cc.redpen.util.DictionaryLoader - Succeeded to load InvalidExpressionVocabulary
[2017-01-24 22:16:24.959] [INFO ] cc.redpen.util.DictionaryLoader - Succeeded to load double negative expression vocabulary
[2017-01-24 22:16:24.960] [INFO ] cc.redpen.util.DictionaryLoader - Succeeded to load double negative word dictionary
[2017-01-24 22:16:24.961] [ERROR] cc.redpen.util.DictionaryLoader - Failed to load WordFrequencyValidator
[2017-01-24 22:16:24.962] [ERROR] cc.redpen.util.DictionaryLoader - Failed to load word frequencies:default
draft.tex:39: ValidationError[CommaNumber], カンマの数 (4) が最大の "3" を超えています。 at line: 最近では、ビルド
draft.tex:39: ValidationError[JapaneseAmbiguousNounConjunction], 助詞「の」が連続しています: "～実行の～の～連～
draft.tex:39: ValidationError[ParenthesizedSentence], Parenthesized sentence is too long. at line: 最近では、ビルド
draft.tex:88: ValidationError[InvalidExpression], 不正な表現 "することができ" がみつかりました。 at line: 例え
```

図 7.11 Wercker

文章チェック結果を表示する。

第 8 章

結果

GitHub に課題研究の概要を提出した場合，文章チェックプログラムによって自動的に文章チェックが行えるようになった．課題研究の概要を文書チェックにかけてみたところ以下の指摘が表示された．

1. SentenceLength - 文長を検査する．
2. InvalidExpression - 不正な表現が利用されていないか検査する．
3. KatakanaSpellCheck - カタカナ単語のスペルチェックを行う．
4. JapaneseAmbiguousNounConjunction - 助詞の「の」が連続しているか検査する．

takahidehamano / docker

Runs

Workflows

Access

Environment

Options

feature-B

feature-B

build

Retry

Steps

get code

1 second

setup environment

2 seconds

wercker-init

0 seconds

echo start

3 seconds

check

2 seconds

Command cancelled due to error

```

export WERCKER_STEP_ROOT="/pipeline/script-211555b3-3397-4bd1-8307-15881b502df9"
export WERCKER_STEP_ID="script-211555b3-3397-4bd1-8307-15881b502df9"
export WERCKER_STEP_OWNER="wercker"
export WERCKER_STEP_NAME="script"
export WERCKER_REPORT_NUMBERS_FILE="/report/script-211555b3-3397-4bd1-8307-15881b502df9/numbers.in
export WERCKER_REPORT_MESSAGE_FILE="/report/script-211555b3-3397-4bd1-8307-15881b502df9/message.tx
export WERCKER_REPORT_ARTIFACTS_DIR="/report/script-211555b3-3397-4bd1-8307-15881b502df9/artifacts
source "/pipeline/script-211555b3-3397-4bd1-8307-15881b502df9/run.sh" < /dev/null
[2016-12-15 07:00:17.742][INFO ] cc.redpen.Main - Configuration file: /pipeline/source/./redpen-co
[2016-12-15 07:00:17.747][INFO ] cc.redpen.config.ConfigurationLoader - Loading config from specifi
[2016-12-15 07:00:17.758][INFO ] cc.redpen.config.ConfigurationLoader - Succeeded to load configur
[2016-12-15 07:00:17.758][INFO ] cc.redpen.config.ConfigurationLoader - Language is set to "ja"
[2016-12-15 07:00:17.758][INFO ] cc.redpen.config.ConfigurationLoader - Deprecated: use "variant"
[2016-12-15 07:00:17.758][INFO ] cc.redpen.config.ConfigurationLoader - Variant is set to "zenkaku
[2016-12-15 07:00:17.759][INFO ] cc.redpen.config.ConfigurationLoader - No "symbols" block found i
[2016-12-15 07:00:17.764][INFO ] cc.redpen.config.SymbolTable - "ja" is specified.
[2016-12-15 07:00:17.764][INFO ] cc.redpen.config.SymbolTable - "zenkaku2" variant is specified
[2016-12-15 07:00:18.189][INFO ] cc.redpen.parser.SentenceExtractor - "[, ?, !]" are added as a
[2016-12-15 07:00:18.189][INFO ] cc.redpen.parser.SentenceExtractor - "[', "]" are added as a righ
[2016-12-15 07:00:18.378][INFO ] org.reflections.Reflections - Reflections took 47 ms to scan 1 un
[2016-12-15 07:00:18.466][INFO ] cc.redpen.util.DictionaryLoader - Succeeded to load katakana word
[2016-12-15 07:00:18.467][INFO ] cc.redpen.util.DictionaryLoader - Succeeded to load InvalidExpres
[2016-12-15 07:00:18.471][INFO ] cc.redpen.util.DictionaryLoader - Succeeded to load double negati
[2016-12-15 07:00:18.471][INFO ] cc.redpen.util.DictionaryLoader - Succeeded to load double negati
[2016-12-15 07:00:18.472][ERROR] cc.redpen.util.DictionaryLoader - Failed to load WordFrequencyVal
[2016-12-15 07:00:18.473][ERROR] cc.redpen.util.DictionaryLoader - Failed to load word frequencies
draft.tex:66: ValidationError[JapaneseAmbiguousNounConjunction], 助詞「の」が連続しています: "～人の完全無
draft.tex:72: ValidationError[KatakanaSpellCheck], カタカナ単語 "プロジェクトマネジメント" に類似する単語 "プロ
draft.tex:83: ValidationError[JapaneseAmbiguousNounConjunction], 助詞「の」が連続しています: "～理解の極端
draft.tex:119: ValidationError[InvalidExpression], 不正な表現 "することができる" がみつかりました。 at line:
draft.tex:126: ValidationError[SuccessiveSentence], 類似する文が二つ続けて使用されています: "Cさんは人間だった
draft.tex:127: ValidationError[SuccessiveSentence], 類似する文が二つ続けて使用されています: "Cさんは人間だった
draft.tex:128: ValidationError[SuccessiveWord], 単語 "人" は連続して使用されています。 at line: この中に最大の
draft.tex:132: ValidationError[SentenceLength], 文長 ("105") が最大値 "100" を超えています。 at line: なぜ
draft.tex:131: ValidationError[InvalidExpression], 不正な表現 "することができる" がみつかりました。 at line:
draft.tex:132: ValidationError[JapaneseAmbiguousNounConjunction], 助詞「の」が連続しています: "～古い師の占

[2016-12-15 07:00:18.612][ERROR] cc.redpen.Main - The number of errors "10" is larger than specifi

```

echo end

図 8.1 文書チェック結果

60

第 9 章

考察

文書チェックプログラムは設定ファイルを変更することで、出力結果も変わる。そのため設定を変更し、添削に有効な設定を検討する必要がある。例えば添削前の文書を文書チェックプログラムにかけ、どの範囲まで文書の誤りをチェックすることができるのか調べることで、文書チェックプログラムが添削できる範囲を調べることができる考えた。

第 10 章

結論

自動的に文書チェックを行う環境を構築することができた．文書チェックプログラムの設定に関しては詳しく調べることができなかったため，検証が必要である．

参考文献

- [1] 平鍋健児, 野中郁次郎. アジャイル開発とスクラム. 翔泳社, 2013.
- [2] REDPEN A document checker. Redpen? <http://redpen.cc/> (2017.01.24 閲覧).
- [3] 塩谷啓, 紫竹佑騎, 平木聡. Web 制作者のための GitHub の教科書チームの効率を最大化する共同開発ツール. インプレス, 2014.
- [4] 大瀧 隆太. アプリ開発者もインフラ管理者も知っておきたい docker の基礎知識. <http://www.atmarkit.co.jp/ait/articles/1405/16/news032.html> (2017.01.26 閲覧).