

AsciiDoc ユーザーマニュアル(詳細版)

K0.1, 2019/08/24

目次

1. 配布パッケージについて.....	1
2. AsciiDocについて	2
3. 仕様書作成にAsciiDocを選択した背景	3
4. 環境構築手順	6
4.1. はじめに.....	6
4.2. Chocolateyをインストール.....	7
4.3. Chocolateyのリポジトリから各種パッケージをインストール.....	9
4.4. AsciiDoc関連ツールをインストール	13
4.5. VSCodeの拡張機能をインストール	15
5. 使い方.....	17
5.1. VSCodeで実際にAsciiDocを書いてみる.....	17
5.2. AsciiDocで書いた文章をGitで管理する	22

1. 配布パッケージについて

中身は以下の通り

```
AsciiDocPackage/  
└ template/                                // 文章のテンプレート一式  
  └ csv/                                    // CSVファイルを格納  
  └ fonts/                                  // フォントファイルを格納  
  └ images/                                // イメージファイルを格納  
  └ style/                                  // スタイルファイルを格納  
  └ make_pdf_test.bat                      // AsciiDoc→PDF変換用スクリプト  
    └ test.adoc                            // AsciiDocドキュメントのテストサンプル  
    └ test2.adoc                           // AsciiDocドキュメントのテストサンプル2  
└ tool/                                    // インストール用バッチファイル  
  └ ①ChocolateyInstall.bat  
  └ ②ChocolateyPackageInstall.bat  
  └ ③AsciiDocToolInstall.bat  
    └ ④VScodeExtensionInstall.bat  
└ AsciiDocUserManual.pdf                  // 手順書
```

2. AsciiDocについて

AsciiDocとは？

- 文章を記述するするための記法 (軽量マークアップ言語) の一つ
- プレーンテキスト (対義語: リッチテキスト) で体裁が整った文章が書ける
- 可読性が高くMarkdownよりも表現力が高い

3. 仕様書作成にAsciiDocを選択した背景

従来のOffice(ExcelやWord)での仕様書作成の問題点

- 不適切なバージョン管理
 - 日付ファイル名
 - 命名規則が適当
 - 同じ日付のコピー
 - 編集者の名前付き
 - ファイル名に最新と付く
 - フォルダ名に最新と付く
 - バックアップ
 - BackUpファイルが分散していてOriginalファイルの改訂に追従しづらい
 - Originalファイルの格納先は担当者のみぞ知る
 - サーバーが死んでOriginalファイルが取り出せなくなる
- 不適切な履歴管理
 - 更新履歴シート
 - 複数の変更点をまとめて修正
 - ざっくりした履歴コメント
 - 変更の経緯を残しにくい
 - そもそも書かない(誰が編集したのか、誰が追加したのか)
 - 変更箇所は色付き
 - 見つらくなる
 - 色を消すタイミングが適当
 - 次に仕様修正する人が色を消して変更箇所がわからなくなる
- 不適切な品質管理
 - レビューなしで保存/発行
 - 修正箇所は担当者しか把握していない
 - 何の変更が入ったのか共有されない
- 本質的ではない作業
 - 罫線、文字色、フォント種別
 - いちいちキーボードからマウスに持ちかえてリボンから選択
 - 選択肢が多くて無駄に選ぶ時間がかかる
 - 図形の整列
 - 図形の差し込みでフォーマットが崩れる
 - サイズを変えるのも一苦労

- 目次、章番号
 - デザインの選択からナンバリングまで諸々手動でやることが多い
- 生産性
 - とにかく重い
 - Microsoft Wordは動作を停止しました…
 - ファイルサイズの肥大化により開けない、スクロールが遅い

仕様書作成をAsciiDocで行うことのメリット

・ 不適切なバージョン管理、履歴管理、品質管理

◦ バージョン管理ツールのGitとの相性が良い

- 仕様書を一元管理できる
- Originalファイルへは影響を与えず(常にリリース可能な状態に保たれる)ローカルでコミットして編集できる
- 必然とローカル環境にも複製されるので分散開発しやすく障害に強い
- どこを変更したかも全て記録されており、過去の履歴を簡単に参照できる
- テキストベースなので変更箇所の差分管理も容易にできる
- プルリクエストによりメンバーに周知とレビューを兼ねられる

・ 本質的ではない作業、生産性

◦ AsciiDocが解決してくれる

- 全てテキストベースで作業が行える
- 煩わしいマウス操作を行わうことなく文章の構造を簡単に明示できる
- 文章の装飾が自動的に行われるので見た目の調整の時間を短縮できる
- 記法が少ないことで良い意味で制限がかかり、編集能力の担当者差が出にくい
- 編集するツールに限定されない(書くだけならエディタは何でもよい)
- テキストそのままでも可読性の高いドキュメントになるため必然的に簡潔な内容になりレビューしやすい
- 対応アプリの拡張機能で簡単にプレビュー環境をつくれて快適に読み書きできる
- シーケンス図などをPlantUMLでテキストベースで書いて埋め込むことが可能
- 外部ファイルのインクルードもテキストベースで書いて読み込みが可能
- コメントアウトを書くことが可能(変更の経緯を残すのに使える)
- 展開用にHTML化やPDF化なども容易にできる
- テキストベースなのでとにかく軽い!

4. 環境構築手順

4.1. はじめに

本書では、AsciiDocのテキストエディタとして Visual Studio Code を利用することとします。
また、Gitを視覚的に操作できるSourceTreeを利用して、AsciiDocドキュメントのバージョン管理を行います。

以下の環境で動作を確認しています。

- Windows 10 Home (64bit)
- .NET Framework 4.0以上 (※Windows 7 の場合、標準搭載は 3.5.1 のためVerUpが必要です)
- Chocolatey 0.10.15
 - ruby 2.6.3.1
 - asciidoctor 2.0.10
 - asciidoctor-pdf 1.5.0.beta.3
 - asciidoctor-pdf-cjk 0.1.3
 - asciidoctor-diagram 1.5.18
 - coderay 1.1.2
 - Graphviz 2.38.0.20190211
 - jdk8 8.0.221
 - Maven 3.6.1.20190711
 - Visual Studio Code 1.38.1
 - AsciiDoc 2.7.6
 - Japanese Language Pack for Visual Studio Code 1.37.5
 - PlantUML 2.12.1
 - SourceTree 3.1.3

4.2. Chocolateyをインストール

以下のバッチファイルをダブルクリックで実行する

①ChocolateyInstall.bat

- ユーザーアカウント制御の許可のポップアップが出るので **はい** をクリック
- コマンドプロンプトが表示されて処理が進むので自動的に閉じたら完了

<覚え書き>実行内容について

コマンドプロンプト(管理者権限)で以下を実行

```
@"%SystemRoot%\System32\WindowsPowerShell\v1.0\powershell.exe" -NoProfile  
-InputFormat None -ExecutionPolicy Bypass -Command "iex ((New-Object  
System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))" &&  
SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"
```



公式サイト

<https://chocolatey.org/install#installing-chocolatey>

インストール手順解説(日本語)

<https://qiita.com/konta220/items/95b40b4647a737cb51aa>

Chocolateyとは?

- Windows上で動作するソフトウェアをコマンドラインでパッケージ管理可能なツール

メリット

- Chocolateyのリポジトリに登録されているパッケージを**一発でインストール**できる
- Chocolateyでインストールしたソフトは**一括でアップデート**できる

4.3. Chocolateyのリポジトリから各種パッケージをインストール

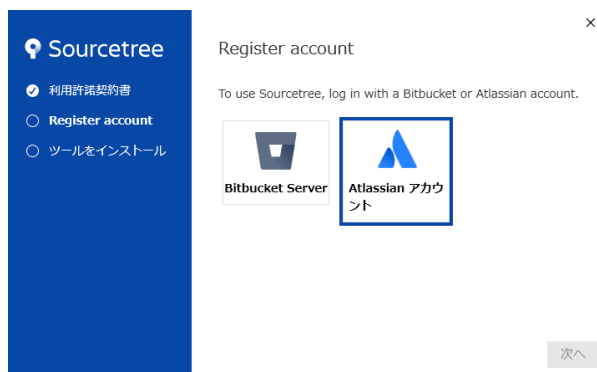
以下のバッチファイルをダブルクリックで実行する

②ChocolateyPackageInstall.bat

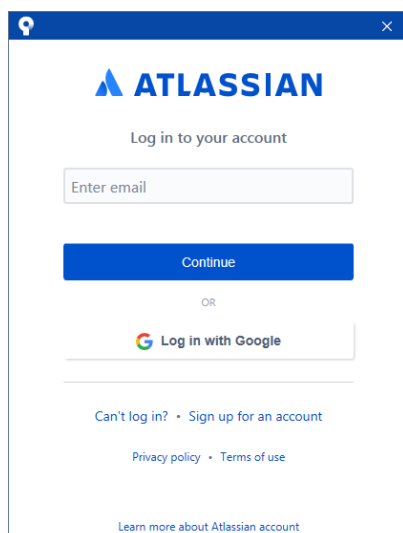
- ユーザーアカウント制御の許可のポップアップが出るので **はい** をクリック
- コマンドプロンプトが表示されて処理が進むのでしばらく待つ
- 下記画面が表示されたら **ライセンスに同意します** にチェックを入れて **次へ** をクリック



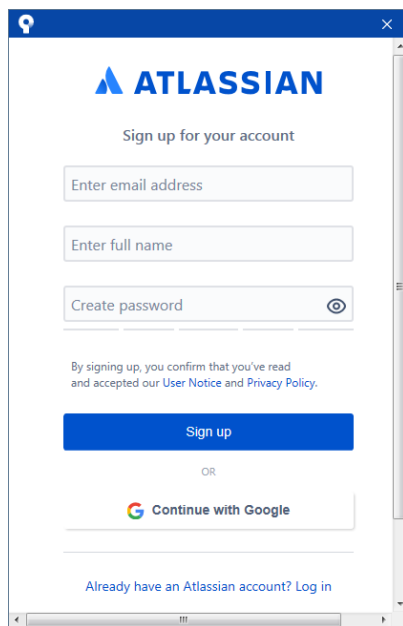
- **Atlassianアカウント** を選択して **次へ** をクリック



- **Enter email** にメールアドレスを入力して **Continue** をクリック



- 続けて、**Enter full name** にニックネーム、**Create password** にパスワードを求められるので入力して、**Sign up** をクリック (※既にアカウントを持っている場合は通常のサインインを行う)

A screenshot of the Atlassian sign-up form. The form is titled "Sign up for your account" and includes fields for "Enter email address", "Enter full name", and "Create password". Below the fields, there is a checkbox for "By signing up, you confirm that you've read and accepted our User Notice and Privacy Policy." and a blue "Sign up" button. Below the button, there is a link for "Continue with Google" and a link for "Already have an Atlassian account? Log in".

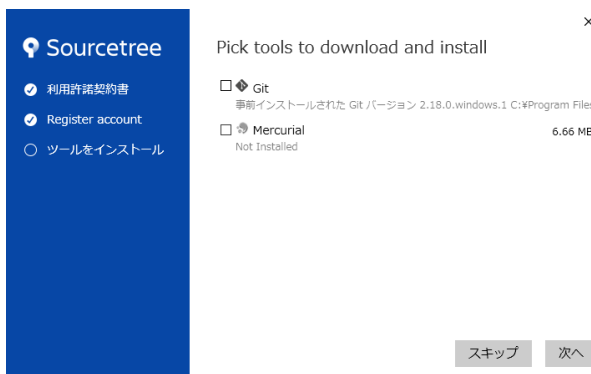
- reCAPTCHAの画像認証の指示に従って選択を行い、**確認** をクリック



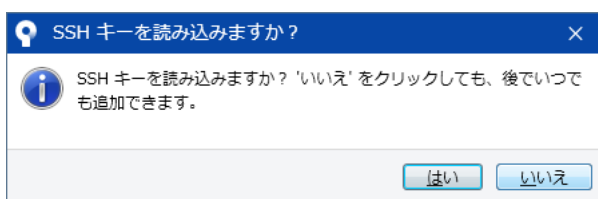
- 認証に成功すれば登録完了画面に遷移するので、**次へ** をクリック



- ツールのインストール画面に遷移するので **Git** にだけチェックを入れて、**次へ** をクリック (※既にGitをインストール済みの場合はそのまま **次へ** をクリック)



- **SSHキーを読み込みますか？** が表示されたら **いいえ** をクリック



- Sourcetreeが自動的に立ち上がったのが確認できたらアプリを一旦閉じる



- コマンドプロンプトの画面内にて、Atlassianアカウントを作成完了したか聞かれるので、完了していたら **y**、完了していなければ **n** を入力して **Enter** を押す
y の場合：コマンドプロンプトが表示されて処理が進むので自動的に閉じたら完了
n の場合：再度、**Enter** を押すとコマンドプロンプトが閉じる

<覚え書き>実行内容について**コマンドプロンプト(管理者権限)で以下を実行**

```
cinst ruby -y ①  
cinst graphviz -y ②  
cinst jdk8 -y ③  
cinst maven -y ④  
cinst vscode -y ⑤  
cinst sourcetree --version 2.5.5 -y ⑥
```

- ① Ruby (AsciiDoc関連ツールを利用するのに必要)
- ② Graphviz (PlantUMLのレンダリングライブラリとして必要)
- ③ Java (PlantUMLの動作環境として必要)
- ④ Maven (Javaのプロジェクト管理ツールでPlantUMLの爆速プレビューに必要)
- ⑤ Visual Studio Code (AsciiDocをプレビュー可能なテキストエディタ)
- ⑥ SourceTree (GitのGUIツール)

Atlassianアカウントを作成してSourceTreeのサインインに成功したら
コマンドプロンプト(管理者権限)で以下を実行しアップデートする

```
choco upgrade all -y
```



初めから最新verをインストールしないのはBitbucketに登録せずに利用するため
<https://hepokon365.hatenablog.com/entry/2019/03/25/222814>

4.4. AsciiDoc関連ツールをインストール

以下のバッチファイルをダブルクリックで実行する

```
③AsciiDocToolInstall.bat
```

- コマンドプロンプトが表示されて処理が進むので自動的に閉じたら完了

<覚え書き>実行内容について**コマンドプロンプトで以下を実行**

```
gem install asciidoctor ①  
gem install --pre asciidoctor-pdf ②  
gem install asciidoctor-pdf-cjk ③  
gem install asciidoctor-diagram ④  
gem install coderay ⑤
```

- ① AsciiDoc→HTMLに変換用
- ② AsciiDoc→PDFに変換用
- ③ PDF変換のレイアウト崩れ対応用
- ④ PlantUML等の図の記述および出力用
- ⑤ コードのシンタックスハイライト用

社内のネットワークから実施する場合はgemにproxyを指定する

```
gem install xxxx -p proxy http://アドレス:ポート
```



proxyの確認手順

<https://pasokatu.hateblo.jp/entry/2017/07/04/111147>

asciidoctor公式サイト

<https://asciidoctor.org/>

4.5. VScodeの拡張機能をインストール

以下のバッチファイルをダブルクリックで実行する

```
④VScodeExtensionInstall.bat
```

- コマンドプロンプトが表示されて処理が進むので自動的に閉じたら完了

<覚え書き>実行内容について

コマンドプロンプトで以下を実行

```
code --install-extension joaompinto.asciidoctor-vscode ^ ①  
code --install-extension MS-CEINTL.vscodelanguage-pack-ja ^ ②  
code --install-extension jebbs.plantuml ^ ③
```

- ① ASiiDocのプレビュー用
- ② 日本語表示用
- ③ PlantUMLの爆速プレビュー用



[表示]→[拡張機能]から検索してインストール or コマンドラインからインストール
<https://qiita.com/Kosen-amai/items/03632dee2e1694652f06>

5. 使い方

5.1. VScodeで実際にAsciiDocを書いてみる

ここでは、テストサンプルでプレビューやPDFへの変換を行い、正しく環境構築ができたことを確認します
また、テストサンプルの内容はAsciiDocの文法紹介も兼ねているので参考にしてください

5.1.1. 作業ディレクトリを作成する

配布パッケージ内のtemplateフォルダー式をローカルPCの任意の場所にコピーして使います
このフォルダー式が文章のテンプレートとなります

<覚え書き>templateフォルダの中身について

文章作成のための作業ディレクトリを用意

```
└ template/                // 文章のテンプレート一式
└ csv/                    // CSVファイルを格納
└ fonts/                  // フォントファイルを格納
└ images/                 // イメージファイルを格納
└ style/                  // スタイルファイルを格納
```

HTMLのスタイルファイルを用意

asciidocの配布ファイルがWindowsの場合は以下にあるのでコピーして利用

```
// ruby2.6でasciidocのverが2.0.10の場合
C:\tools\ruby26\lib\ruby\gems\2.6.0\gems\asciidoc-
2.0.10\data\stylesheets\asciidoc-default.css
```

PDFのスタイルファイルを用意

asciidoc-pdfの配布ファイルがWindowsの場合は以下にあるのでコピーして利用

```
// ruby2.6でasciidoc-pdfのverが1.5.0.beta.2の場合
C:\tools\ruby26\lib\ruby\gems\2.6.0\gems\asciidoc-pdf-
1.5.0.beta.2\data\themes\default-theme.yml
```



デフォルトのスタイルファイル
css : そのままでも十分使えそう
yaml : いまいちなのでHTML化時のスタイルに寄せた設定に修正してみた (public_style.ymlとして利用)

公式サイト
<https://github.com/asciidoc/asciidoc-pdf/blob/master/docs/theming-guide.adoc>

色表現
https://www.lab-nemoto.jp/www/leaflet_edu/ColorMaker.html

PDF化時に文字の色が変わるようにする
https://blog.siwa32.com/asciidoc-pdf_color/
→「2.2 asciidoc-pdfのソースを修正する」

フォントファイルを用意

asciidoc-pdfの配布ファイルがWindowsの場合は以下にあるのでコピーして利用

```
// ruby2.6でasciidoc-pdfのverが1.5.0.beta.2の場合
C:\tools\ruby26\lib\ruby\gems\2.6.0\gems\asciidoc-pdf-
1.5.0.beta.2\data\fonts\*.ttf
```



カスタマイズ参考サイト
<https://ryuta46.com/267>
<https://qiita.com/kuboaki/items/67774c5ebd41467b83e2>

ドキュメントファイルを用意

適当にメモ帳で以下の設定で作成する

```
拡張子 : .adoc  
文字コード : UTF-8
```

格納後の作業フォルダ内はこんな感じになる

```
└─ template/  
  └─ csv/  
  └─ fonts/  
    └─ *.ttf  
    └─ ...  
  └─ images/  
  └─ style/  
    └─ asciidoctor-default.css  
    └─ default-theme.yml  
    └─ public_style.yml  
  └─ *.adoc
```

5.1.2. VScode を起動する

AsciiDocで書くためのテキストエディタとして使用します

Windowsのスタートメニューから **Visual Studio Code** (以下、VScodeとする)を検索して起動します

5.1.3. テストサンプルを開く

[ファイル]→[ファイルを開く]から **template** フォルダ内の **test.adoc** ファイルを開きます

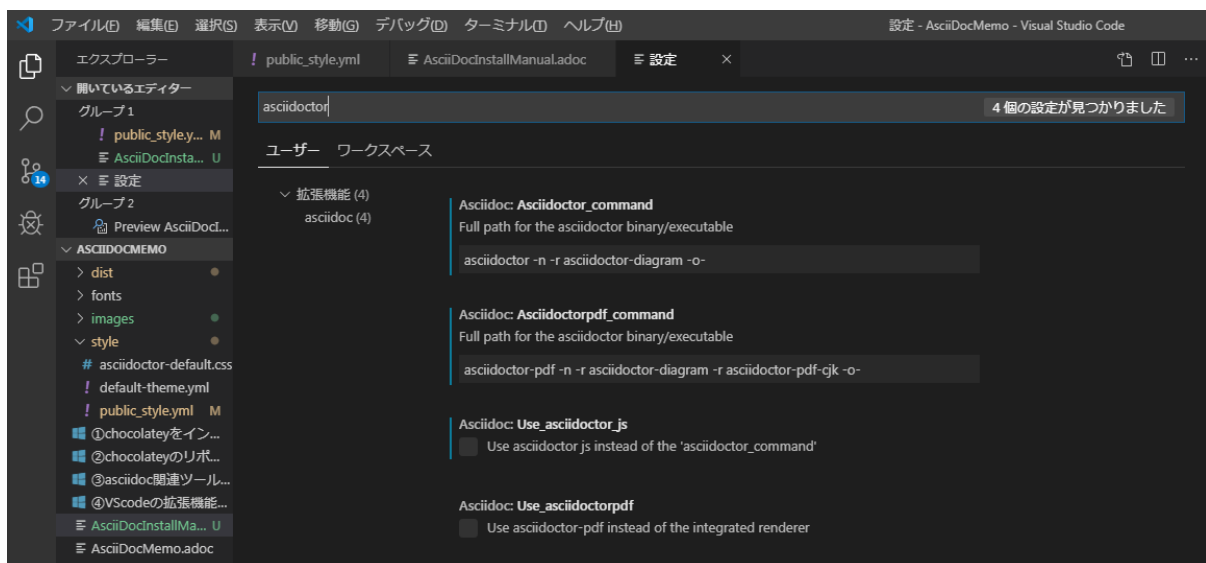
5.1.4. テストサンプルをプレビューする

asciidocotorの設定を変更する

VScode上でプレビュー表示を行うための設定を行います

[ファイル]→[基本設定]→[設定]から **asciidocotor** を検索し、以下の設定を行います

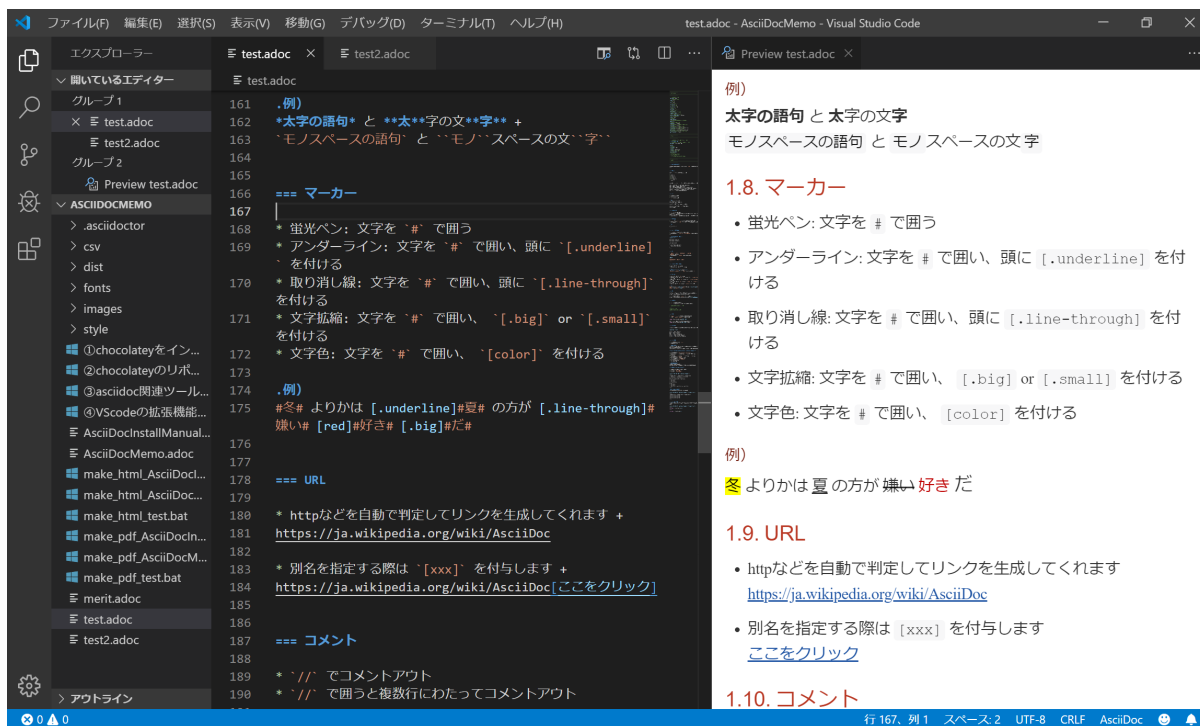
```
asciidocotor_command : asciidocotor -n -r asciidocotor-diagram -o-  
asciidocotorpdf_command : asciidocotor-pdf -n -r asciidocotor-diagram -r  
asciidocotor-pdf-cjk -o-  
use_asciidocotor_js : false(チェックを外す)
```



プレビューを行う

ショートカット **Ctrl+K** → **V** で画面右側にプレビューが表示されます

参考までにテストサンプルのプレビュー結果を以下に示します



5.1.5. テストサンプルをPDFに変換する

以下のバッチファイルをダブルクリックで実行する

```
make_pdf_test.bat
```

- ・ コマンドプロンプトが表示されて処理が進むので自動的に閉じたら完了
- ・ **test.pdf** が作成されます



必要に応じてパッチファイル内のファイル名を修正して使ってください

-o 変換後ファイル名.pdf 変換前ファイル名.adoc

<覚え書き>実行内容について

コマンドプロンプトで以下を実行(*にファイル名を指定)

```
asciidoctor -r asciidoctor-diagram -o *.html *.adoc ①
asciidoctor-pdf -r asciidoctor-diagram -r asciidoctor-pdf-cjk -o *.pdf *.adoc ②
```

① AsciiDoc→HTML化用コマンド

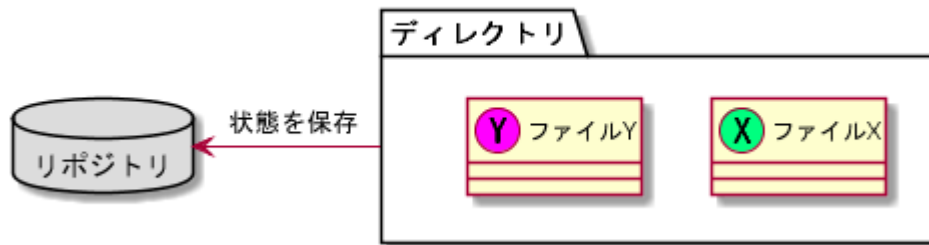
② AsciiDoc→PDF化用コマンド

5.2. AsciiDocで書いた文章をGitで管理する

Gitとは？

分散型バージョン管理システムの一つ

リポジトリと呼ばれる記録場所に、管理したいディレクトリの状態を記録することができる

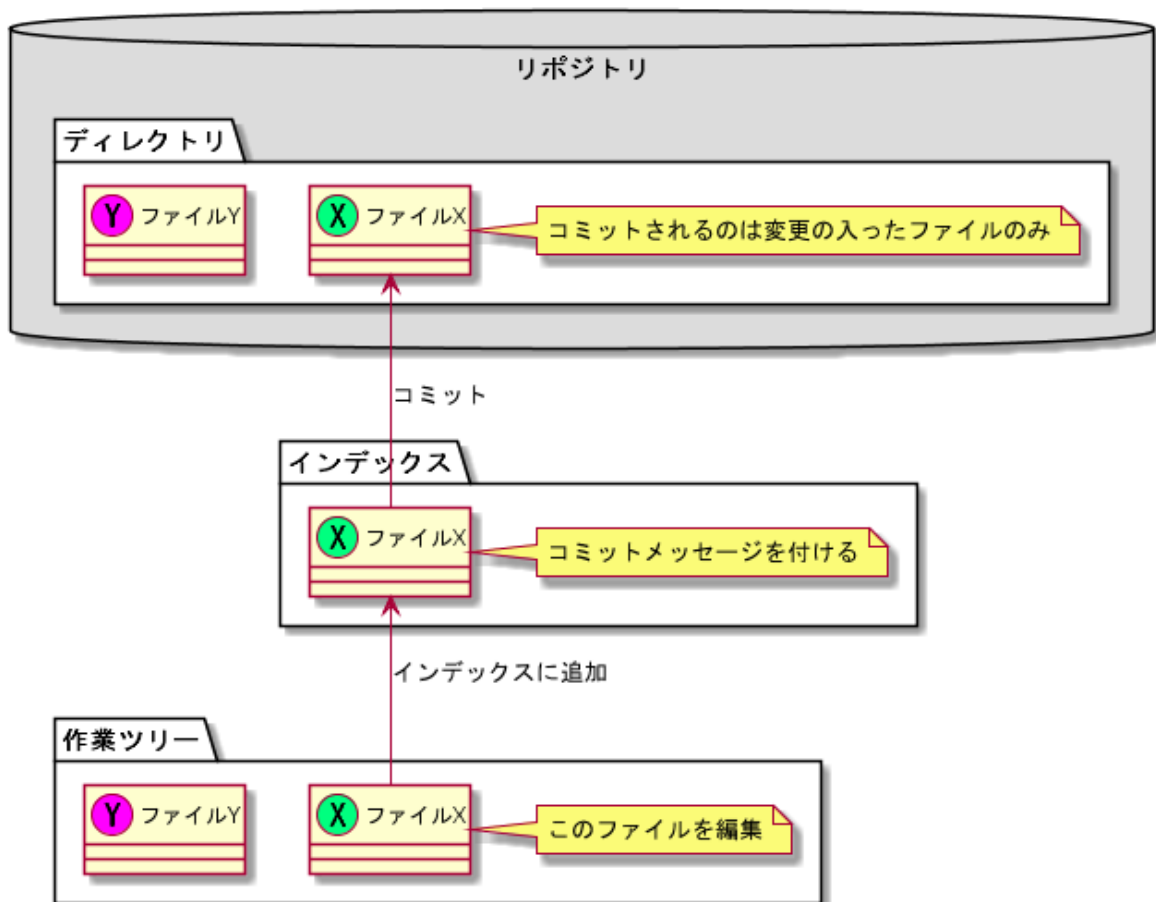


状態の変化を記録するコミット

コミットと呼ばれる操作により、変更作業により発生したディレクトリの状態の変化をリポジトリに記録する

- 管理下に置かれた作業ディレクトリ(作業ツリー)から、コミットのための準備場所(インデックス)に、変更したファイルを追加する
- コミットメッセージを付けて、コミットを実行する

1行目 : コミットでの変更内容の要約
2行目 : 空行
3行目以降 : 変更した理由



このようにインデックスを挟むことで、変更されてないファイルを含めずにコミットでき、また、ファイルの一部の変更だけを追加してコミットできたりする

なお、状態の変化を履歴として記録しており、変更前ファイルをバックアップ用に残す必要はない
履歴を元に、**過去の状態に戻す**ことや、**変更箇所の差分を表示**したりもできる

履歴を管理するリポジトリ

リポジトリには2種類ある

- ・ リモートリポジトリ

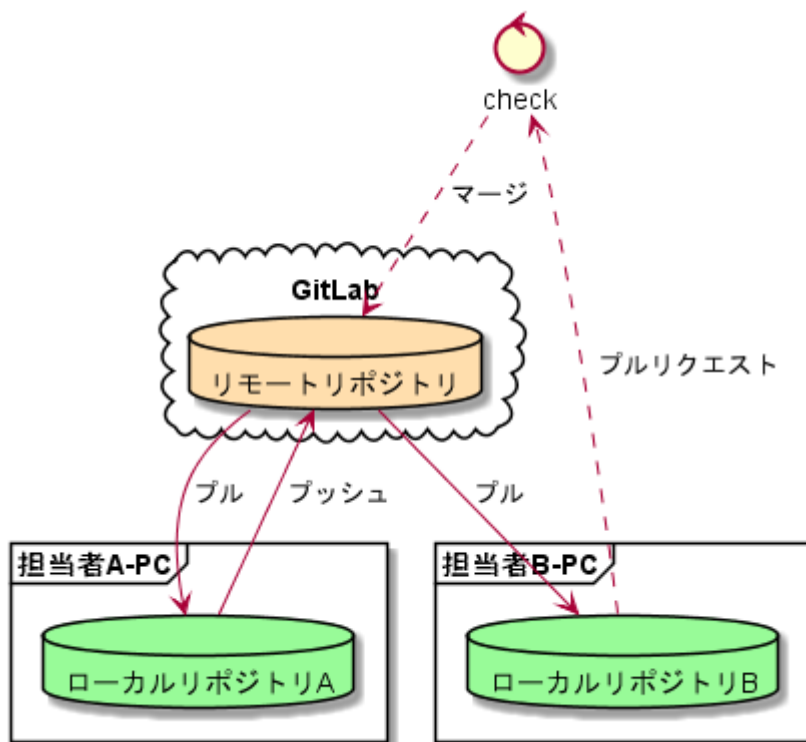
- 専用のサーバに配置して複数人で共有するためのリポジトリ

- ・ ローカルリポジトリ

- ユーザー一人ひとりが利用するために、自分の手元のPC上に配置するリポジトリ

普段の作業はローカルリポジトリを使って手元のPC上で行い、作業が完了したらリモートリポジトリにアップロード(**プッシュ**)して公開する

他の開発者がリモートリポジトリを通して公開された最新のディレクトリの状態を取得(**プル**)することもできる



リモートリポジトリには、無料で使えるがインターネット上に一般公開されるGitHubと、自前のサーバーに立ててプライベートリポジトリとして使えるGitLabがある(自社では後者を使用する)

なお、プッシュの代わりに、**プルリクエスト**を使うことで組織にレビューの文化を根付かせることも可能
プルリクエストは次のような機能を提供する

- ・ 機能追加や改修など、作業内容がレビュー・マージ担当者やその他関係者に通知される
- ・ ソースコードの変更箇所がわかりやすく表示される
- ・ ソースコードに関するコミュニケーションの場が提供される

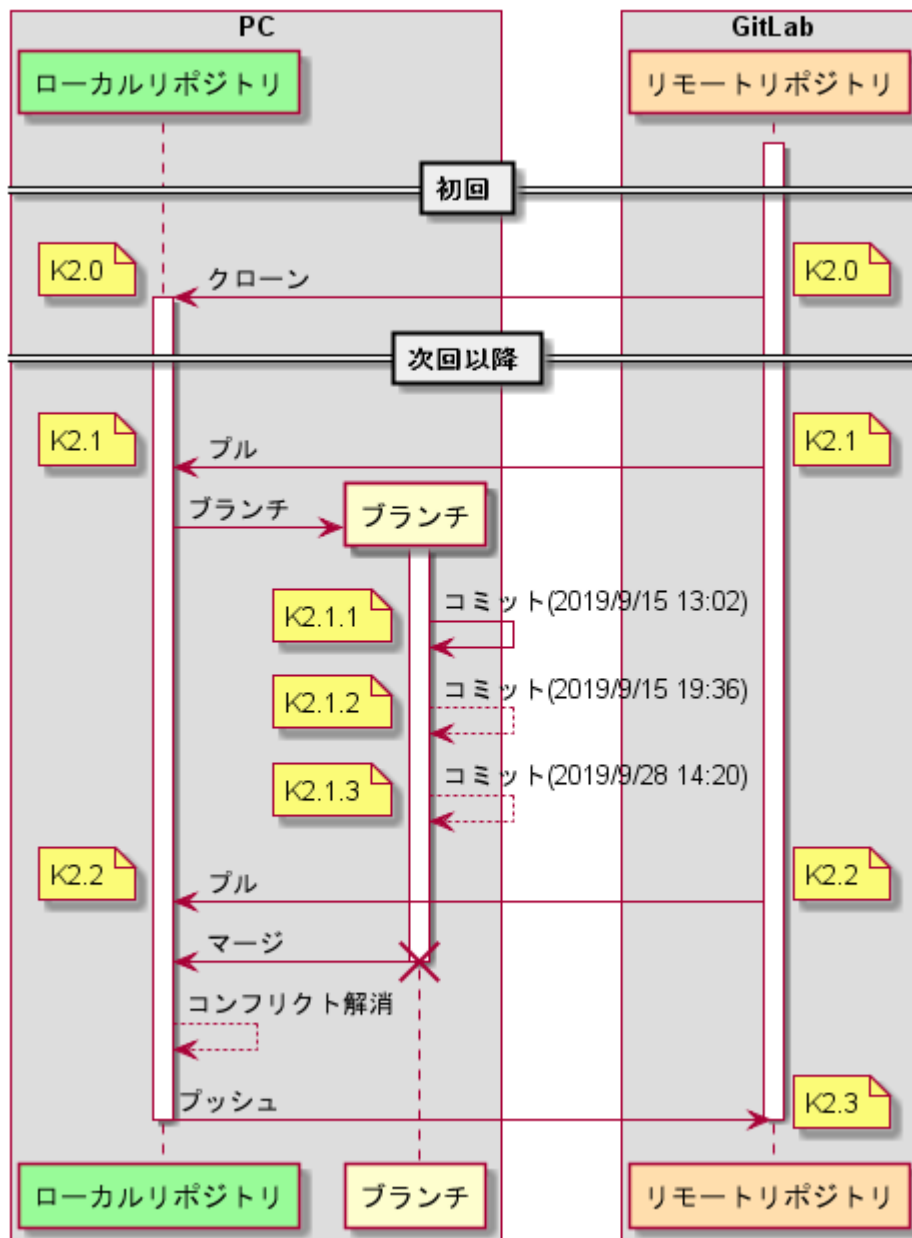
上記のようなやりとりを経て、最終的にマージされるソースコードの品質を高めることができる

運用フロー

初めにリモートリポジトリをローカルに複製(クローン)する

以降はローカルリポジトリにて変更作業を行っていくが、自社では以下の手順を遵守することとする

- ・ 変更作業を始める前に、最新の状態をリモートリポジトリから**プルし直す**
- ・ 変更作業はmasterブランチでは行わず、**ブランチを切って移動**して行う
- ・ 移動先のブランチ内にて、変更作業を行い、**変更箇所ごとに都度コミット**を実行する
- ・ 全ての変更作業を終えたら、masterブランチに移動し、最新の状態をリモートリポジトリから**プルし直す**
- ・ 変更作業中のブランチに再び移動し、**masterブランチへのマージ**を行う
- ・ マージの際に、他の開発者による変更との競合が発生した場合は、**競合内容を確認し修正をコミット**する
- ・ リモートリポジトリへ**プッシュ**する(または、プルリクエストを行う)



この運用の一番の目的は、**リモートリポジトリは常にリリースできる状態に保つこと**、である
そのため、**ローカルリポジトリのmasterブランチは最新に保つこと**を心掛け、**編集はブランチを切って行う**
ローカルリポジトリにて、必ずリモートリポジトリのorigin/masterブランチの最新の状態をベースとし、
先行した状態にしてからリモートリポジトリにプッシュする

SourceTreeを使って運用する

上記の作業は、GitのGUIツールであるSourceTreeを使用して行いますが、以下の参考サイトに初心者でもわかるようにまとまっているので説明は割愛します
理解に必要な用語は一通り説明し終えているはずなので、読んで理解できる内容だと思います

<誰でも簡単!GitHubで管理するためのSourcetreeの最低限の使い方>

<https://haniwaman.com/sourcetree/>

※自社ではGitHubの代わりにGitLabを使いますのでGitLabに置き換えて読んでください

また、用語については運用に最低限必要な部分だけのピックアップなので、詳しくは以下サイトを読んでください

<サルでもわかるGit入門>

<https://backlog.com/ja/git-tutorial/>

展開したり共有する必要がない資料は、ローカルリポジトリだけで運用してバージョン管理することもできます

ローカルに保存先のパス(作業ディレクトリ)を指定して、リポジトリを作成(**クリエイト**)してみて、色々試してみると良いです

習うより慣れろ、です!

以上で終わりです!