2023年5月作成

Github & Gitの使い方

はじめに

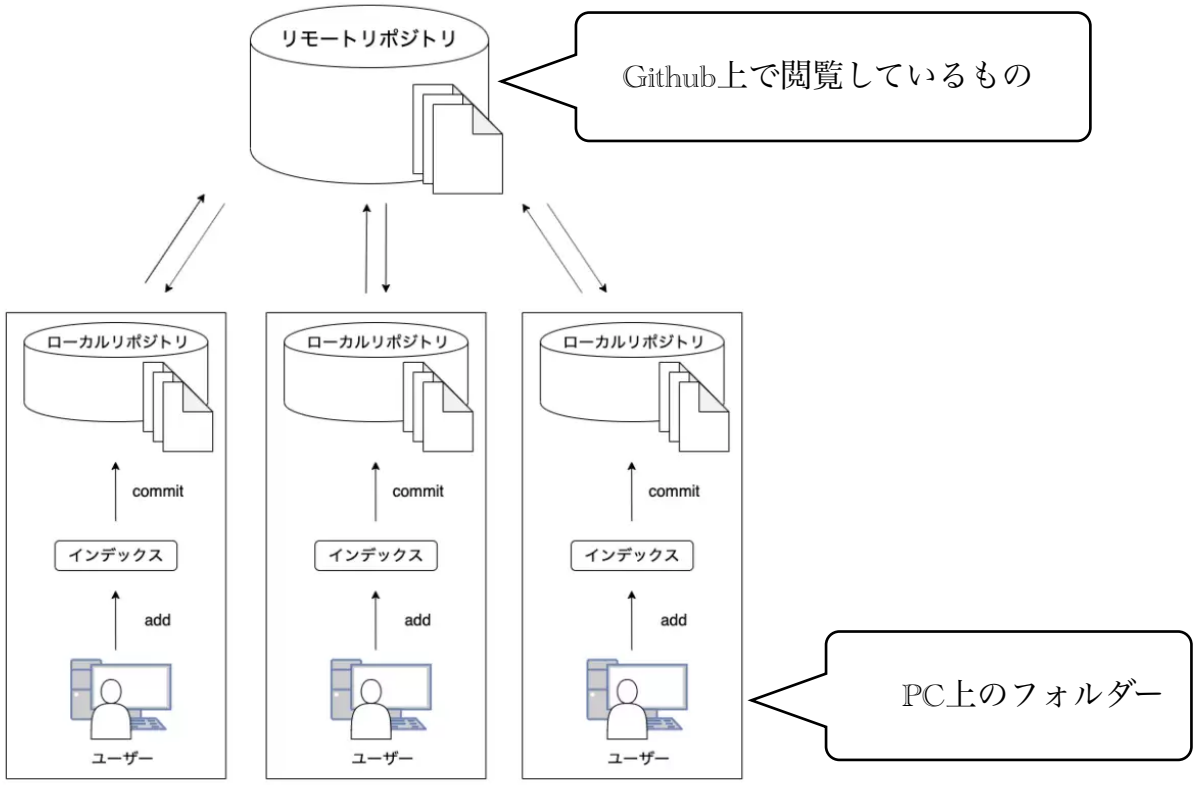
ここではとりあえずgitを使えるようになるまでの説明をしていきます。言葉遣いの違いや、文言の調整は行っていないので見つけても許してください....

目次

I. gitについて

・ gitについて

　調べると色々詳しいことが出てくるのでここでは書かないが、ファイルのバージョン管理が簡単にできるツールと覚えておけばなんとかなる。ここでのバージョンとは編集したファイルなどの履歴のことで、gitはその管理をとても簡単に行いことができる。
　＊実際にプロジェクトとして使うときの構図



リポジトリ
　リポジトリとは、ファイルやディレクトリを保存しておくためのスペースです。
　Gitにおけるリポジトリは主に２種類に分かれています。

・ リモートリポジトリ: 特定のサーバー上で設置に保存され、複数人で共有する　　ためのリポジトリです。github上で閲覧できるものはこれです。*後述

・ ローカルリポジトリ: ユーザーごとに配置される、手元で編集ができるリポジトリです。``git clone [url]``とかしたやつです。(自分のpcのフォルダー上で見れるやつ)

- GitとGithubの違い

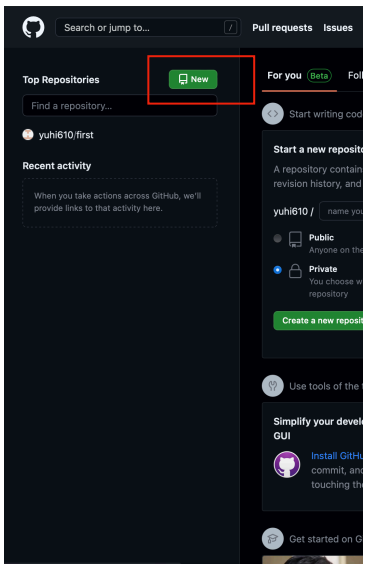
簡単に言えばバージョン管理ツール自体はGitでそれweb上で見れたりするのがGithubです。

II. Gitの使い方

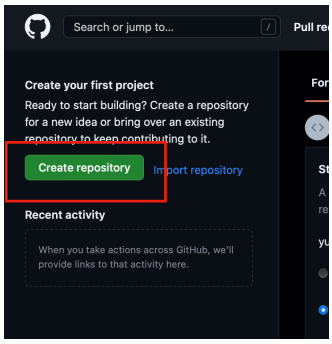
リポジトリの作成

最初に必要になる、リポジトリの作り方。
Github上で作成する方法と、コンソールから作成する方法があるが、ここではgithubを使ってWeb上から作成する保存方法について説明する。

まず[https://github.com]にアクセスします。

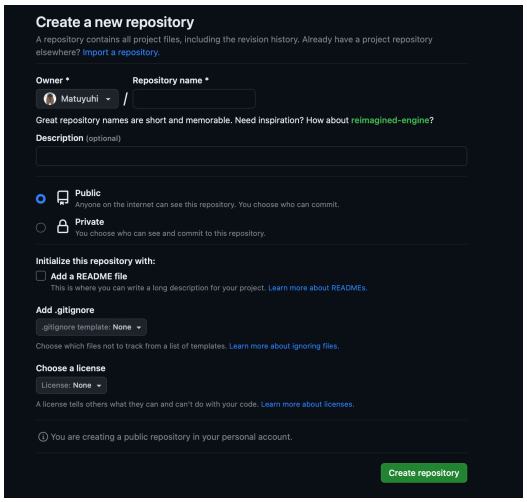


アクセスした画面の左の欄にある[new]ボタン(赤枠)を押すと、作成画面に移動します。



※ 一つもリポジトリがない場合は[create repository]というボタンになります。

ボタンを押すと次の画面に移ります。ここでリポジトリの設定をします。



Owner — このリポジトリの所有者
Repository name — リポジトリの名前 (必須)
Description — 説明文 (任意)
Public or Private — このリポジトリのアクセス権を設定します。publicは誰でもアクセス可能になり、privateはOwner、もしくはOwnerが許可した人のみ閲覧できます。(private推奨)

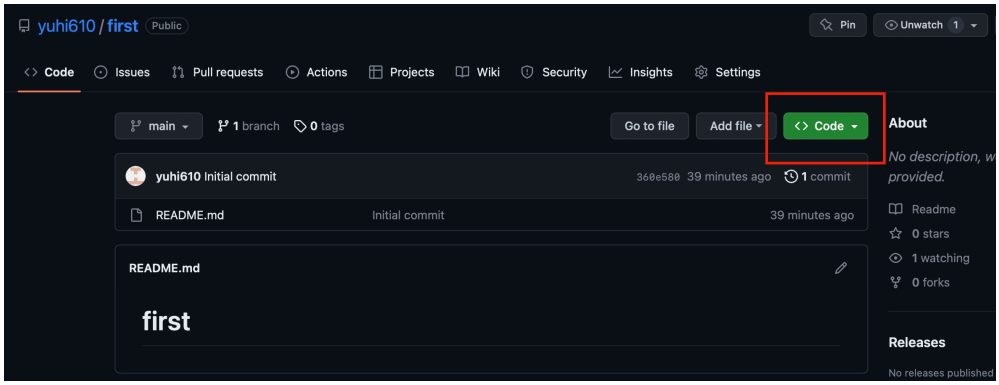
Add a README file — チェックにすると初期状態のリポジトリにREADMEというファイルを追加します。(※推奨)

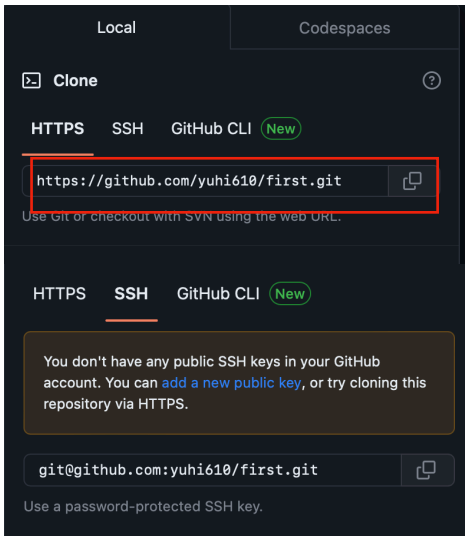
Add .gitignore — 選択したテンプレートのignoreファイルを追加します。
Choose a license — ライセンスの設定(基本無視で良い)

※ここで決めた情報は大抵の場合、後から変更出来るので間違えても問題ないです。

リポジトリのクローン

作業を開始するためにまず、自分のPCにリポジトリをクローンするということをします。この作業で作成されるものが前途したローカルリポジトリです。
リポジトリにアクセスすると次のような画面に移動します。



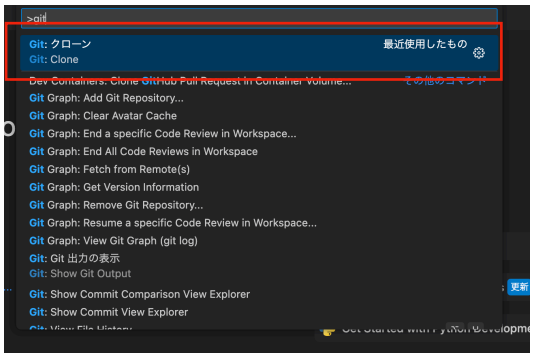


赤色の[Code]のボタンを押すと、左のポップアップが表示されます。
ここのHTTPS、もしくはSSHキーの設定が済んでいる人はSSHのタブからすぐ下に表示されるリンクをコピーしてください。

※ もし、SSHキーの設定ができていない人は左のような注意が出ます。

コピーしたURLを使ってクローンします。

VS Codeの場合：



開いて、[Ctrl + Shift + P]を押すと上部分に入力欄が出てくるので、git cloneと入力すると,[Git: クローン]が出てきます。

下の画像のような画面になるので、先ほどコピーしたURLを入力します。
保存するフォルダーを選んで終了になります。

⇐ CloneのURLを入力

コンソールの場合：

クローンしたいフォルダーでコンソールを開き、git cloneコマンドを入力します。

```
❯ cd work
❯ git clone [cloneのURL]
Cloning into
remote: Enumerating objects: 229, done.
remote: Counting objects: 100% (229/229), done.
remote: Compressing objects: 100% (149/149), done.
remote: Total 229 (delta 132), reused 156 (delta 62), pack-reused 0
Receiving objects: 100% (229/229), 67.37 KiB | 353.00 KiB/s, done.
Resolving deltas: 100% (132/132), done.
```

”cd [作業フォルダー]”
“git clone [url] “

この時、エラーが出ずにフォルダー内にリポジトリの名前と同じフォルダーが作成されていたらOKです。

共有までの流れ

続いて、クローンしたフォルダー内(ローカルリポジトリ)で作業し、変更したものを共有(リモートリポジトリに反映)するまでの流れを説明します。

一つの変更を共有する手順を簡単にコマンドで言うとaddしてcommitしてpushするだけです。

Gitではファイルの編集、追加、削除の全てが1つの変更箇所となります。そして、それら複数を一つの変更箇所のグループ(commit)となり、この１つごとにリポジトリのバージョンを戻したりできます。(commit内のファイル単体を戻すこともできます。)バージョンの確認について詳しくは、statusを参照してください。

III. コマンドについて

コンソールで入力するコマンドは基本 コマンド サブコマンド ....のような形 1 マス空白を開けながら入力します。そのコマンドのサブコマンドやオプションについては、そのコマンドの後ろに -hをつければ大体コマンドのヘルプが表示されます。(hはhelpの頭文字)

コマンドにはオプションが存在するものがあり、optionは[ーー] + [オプション名]で表すものがほとんどです。もしくは [ー] + [オプション名の頭文字]。

例) gitのコマンドが知りたい時 (git -h) or (git —help)

```
❯ git -h
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
           [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
           [--config-env=<name>=<envvar>] <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone     Clone a repository into a new directory
   init      Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add       Add file contents to the index
   mv        Move or rename a file, a directory, or a symlink
   restore   Restore working tree files
   rm        Remove files from the working tree and from the index
```

こんな感じで、たくさん出てくる

clone (クローン)

add (アド)

commit (コミット)

push (プッシュ)

status (ステータス)

checkout (チェックアウト)

stash

参照

- [【初心者向け】Gitとは何なのか。基本用語やその仕組みをまとめています。], (https://tcd-theme.com/2019/12/what-is-git.html), 01/05/23 参照