GitHub

Akihiro Minamino

August 14, 2020

1 GitHub とは

GitHub は、コードを共有するための場所を提供している Git リポジトリのホスティングサービスである。

2 Mac への Git のインストール

Homebrew を用いてインストールするのが簡単である。

```
$ brew install git
```

インストールされて git の version は、以下で確認できる。

```
$ git --version
```

3 Git の初期設定

次に Git で利用する名前とメールアドレスを設定する。

```
$ git config --global user.name "Firstname Lastname"
$ git config --global user.email "your_email@example.com"
```

さらにコマンド出力を読みやすくする。

```
$ git config --global color.ui auto
```

上記の内容は、設定ファイル `~/.gitconfig` に書き込まれている。

4 GitHub の利用準備

4.1 GitHub アカウントの作成

GitHub のアカウント作成ページ `https://github.com/join` でアカウントを作成する。
この時、「Username」には希望する ID を英数字で入力する。この「Username」は、公開ページの URL で `https://github.com/Username` として使われる。

4.2 SSH 公開鍵の作成

GitHub では、作成したリポジトリへのアクセス認証を SSH を利用した公開鍵認証で行う。
以下のように SSH Key を作成する。

```
1 ssh -keygen -t rsa -C "your_email@example.com"
```

(passphrase は「なし」=「何も入力しないで Enter」でよい。)
この結果、`~/.ssh/` に、`id_rsa` という秘密鍵ファイルと、`id_rsa.pub` という公開鍵ファイルが作成される。

4.3 GitHub への SSH 公開鍵の登録

GitHub の web ページの右上のアカウント設定ボタン (Account Settings) を押し、「SSH Key」のメニューを選択する。

- Title に適当な鍵の名前（例えば「MacBookAirKey」など）を入力する。
- Key には、`id_rsa.pub` の内容をコピーして貼り付ける。

公開鍵の登録に成功すると、登録したメールアドレスに公開鍵登録完了のメールが届く。

動作確認を以下のとおり行う。

```
1 $ ssh -T git@github.com
```

(「Are you sure you want to continue connecting (yes/no)?」には、yes と入力する。)
次のように表示されれば成功。

```
1 Hi Username! You've successfully authenticated,
2 but GitHub does not provide shell access.
```

5 GitHub の使い方

5.1 リポジトリの作成

GitHub の web ページの右上の「+」をクリック後、「New repository」をクリックする。リポジトリの情報を以下のように入力する。

- Repository name: リポジトリの名前（例えば Hello-World など）
- Description: リポジトリの説明（省略可）
- Public、Private: 公開か非公開かを決める。
- Initialize this repository with a REAEME: チェックを入れる。
- Add .gitignore: プルダウンメニューでこのリポジトリで管理する言語やフレームワーク（Python、C++、Tex など）を選択することで、Git リポジトリでの管理対象外のファイル・ディレクトリを自動で指定してくれる。
- Add a licence: MIT ライセンス[1]を選ぶ。

作成したリポジトリの URL は以下である。
`https://github.com/ユーザー名/リポジトリ名`

5.2 自分の PC にリポジトリの clone を作成

作成したリポジトリの clone を自分の PC に持ってくる。

```
1 $ git clone git@github.com:ユーザー名/リポジトリ名
2 $ cd リポジトリ名
```

5.3 自分の PC のリポジトリにコードを commit

自分の PC のリポジトリの clone で、コード（ ここでは、hello_world.py とする）を作成する。この時点では、hello_world.py は Git リポジトリに登録されていないので、`git status` では、Untracked files として表示される。

次に、git add コマンドで、hello_world.py をステージ[2]する。

```
1 git add hello_world.py
```

[1] MIT Licence: 1. このソフトウェアを誰でも無償で無制限に扱って良い。ただし、著作権表示および本許諾表示をソフトウェアのすべての複製または重要な部分に記載しなければならない。2. 作者または著作権者は、ソフトウェアに関してなんら責任を負わない。

[2] コミットする前のファイル状態を記録したインデックスというデータ構造に記録すること

その後、git commit コマンドで、hello_world を自分の PC のリポジトリにコミットする。

```
1 $ git commit -m "コメント"
```

5.4 GitHub 側のリポジトリを更新

git push で、GitHub 側のリポジトリを更新する。

```
1 $ git push
```

6 GitHub の操作

6.1 ワークツリー

.git ディレクトリを持つディレクトリ以下を、リポジトリに付随したワークツリーと呼ぶ。開発者は、ワークツリーでファイルの編集などを行う。その後、ファイルの編集履歴などを.git ディレクトリに登録して管理する。

6.2 git status

git status コマンドは、Git リポジトリの状態を調べるのに利用する。

```
1 $ git status
```

6.3 git add

Git リポジトリのワークツリーでファイルを作成しただけでは、Git リポジトリにファイルは登録されない。このようなファイルは、git status コマンドで、「Untracked files」と表示される。
ファイルを Git リポジトリの管理対象とするには、git add コマンドで、ステージ領域[3]にファイルを登録する。ステージ領域は、コミットをする前の一時領域のことである。

```
1 $ git add ファイル名
```

6.4 git commit

git commit コマンドは、ステージ領域に登録されている時点のファイルを、リポジトリの歴史として記録するコマンドである。

```
1 $ git commit -m "コメント"
```

[3] ステージ領域はインデックスとも呼ばれる。

6.5 git log

git log コマンドは、リポジトリにコミットされたログを確認するコマンドである。誰がいつコミットやマージを行い、どのような差分が発生したかなどを確認できる。

```
1 $ git log
```

指定したディレクトリ、ファイルのみログを表示するには、git log コマンドにディレクトリ名もしくはファイル名を与えればよい。

```
1 $ git log ディレクトリ名（ファイル名）
```

各コミットでのファイルの差分を表示したい場合は、git log -p と-p のオプションをつける。

```
1 $ git log -p
```

6.6 .gitignore

git の管理下に置きたくないファイルは、.gitignore に記入すればよい。

.gitignore の書き方の例
ファイル名

```
1 sample.html
```

コメント: #で始まる行はコメント

```
1 # コメントです
```

ディレクトリ: ディレクトリ名は、末尾にを付けて指定

```
1 work/
```