SoftEx20

2

ソフトウェア設計及び実験　第２回　プログラミング作法

git, github 利用方法について

Gitの簡単な使い方

2020 年度ソフトウェア設計及び実験　プログラミング作法（4 月 28 日実施）

Git ガイド

松本　和幸
徳島大学理工学部理工学科 情報光情報系
e-mail: matumoto@is.tokushima-u.ac.jp

1

Git (ギット)とは

プログラムのソースコードのバージョン管理ができるシステムです。バージョン管理システムとしては Subversion(svn)や CVS(Concurrent Versions System)などがあります。OSS(Open Source Software)開発では標準となっており、プログラムのソースコード以外にも、テキストファイル（文書ファイル）の執筆管理や Web サイト構築にも使われることがあります。

Git を使うと、細かい単位でソースコードの変更を記録することができます。また、チームで一つのプロジェクトを管理する際に便利な機能が備わっています。ここでは、Git を使い始める前の準備とターミナル（端末）からコマンドを入力して操作する方法について説明していきます。

準備

Git を使うためには、GitHub(ギットハブ)のアカウント作成が必要です。また、Git 本体のインストールが必要です。本節では、GitHub のアカウント作成と Git のインストールについて説明します。

GitHub のアカウント作成

1. 「GitHub」の公式サイト（日本語版）(https://github.co.jp/)にアクセス
2. 「GitHub に登録する」をクリック

3. Username, Email address, Password を入力して、”Select Plan”をクリックする。（ここでは、説明のため、ユーザ名を test_usr とする）

4. パブリックリポジトリ（Unlimited public repositories for free）を選択し、”Continue”ボタンをクリックする。パブリックリポジトリでは、ソースコードが公開になるが、無料で利用可能。大学の授業で利用するなど、ソースコードが外部に漏れても影響がない場合のみ利用すること。

5. “Submit” をクリックし、GitHub からのメール（登録したメールアドレス）を確認する。メールのサブジェクトは「[GitHub] Please verify your email address.」となっている。メール本文にある”Verify email address”をクリックし、ブラウザが起動し、「Your email was verified」という文字列が左上に表示されていれば、アカウントの登録は完了している。

6. 「Create a new repository」で、リモートリポジトリを、適当な名前（プロジェクト名、例えば、SoftEx20 など）で作成する。この場合、リモートリポジトリにアクセスする際の URL は、「https://github.com/test_usr/SoftEx20.git」となる。リモートリポジトリは、インターネット上の作業場所となるため、ここに置いているファイルは全世界に公開されている状態となる（URL を知っている誰もがアクセス可能）。

Git のインストール

Ubuntu（WSL の Ubuntu も含む）の端末上で、以下のコマンドを入力すると Git をインストールできます。パスワードは、管理者（root）のパスワードです。

```
$ sudo apt-get install git-all
```

もし、Windows のコマンドプロンプトや Mac のターミナルを使用していたり、Ubuntu 以外の Linux を使用している場合はインストール方法やコマンドの実行方法が異なる場合があるので、各自で調べてみてください。

SSH Key の設定

ローカル PC とインターネット上の GitHub の間の通信は、SSH（）という暗号通信のためのプロトコルを使います。SSH を使って GitHub とやり取りするために、まず、SSH Key というものを設定する必要があります。ここでは、SSH Key の設定方法について説明します。

ターミナルを起動し、SSH Key が既に存在するかどうかの確認をします。

```
$ ls –al ~/.ssh

…

-rw------- 1 matumoto matumoto  1679 Apr  9 13:58 id_rsa

-rw-r--r-- 1 matumoto matumoto  406 Apr  9 13:58 id_rsa.pub

…
```

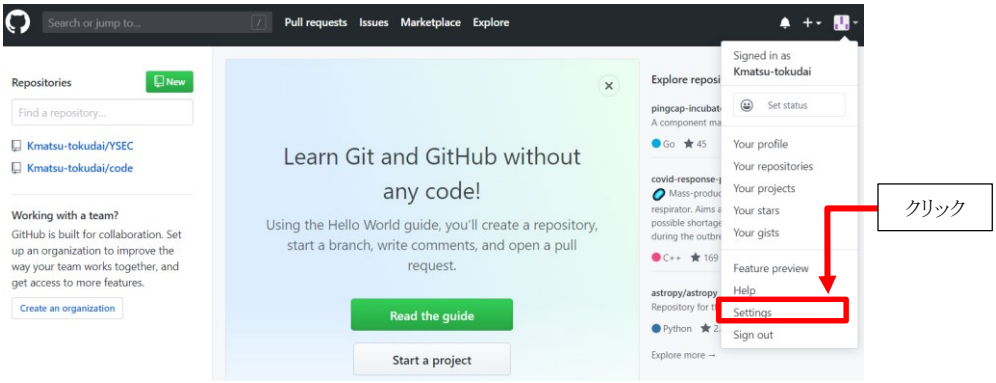
上記のように、ファイル “id_rsa”, “id_rsa.pub”が見つかれば、既に SSH Key が登録されていますので、作成の手順はスキップできます。

SSH Key が存在しない場合、以下のコマンド入力により新しい SSH Key を作成することができます。メールアドレスは、自分のものを入力してください。

```
$ ssh-keygen –t rsa –C “c012345@tokushima-u.ac.jp”

Enter passphrase (empty for no passphrase): パスワードを入力

Enter same passphrase again: 同じパスワードを再度入力
```

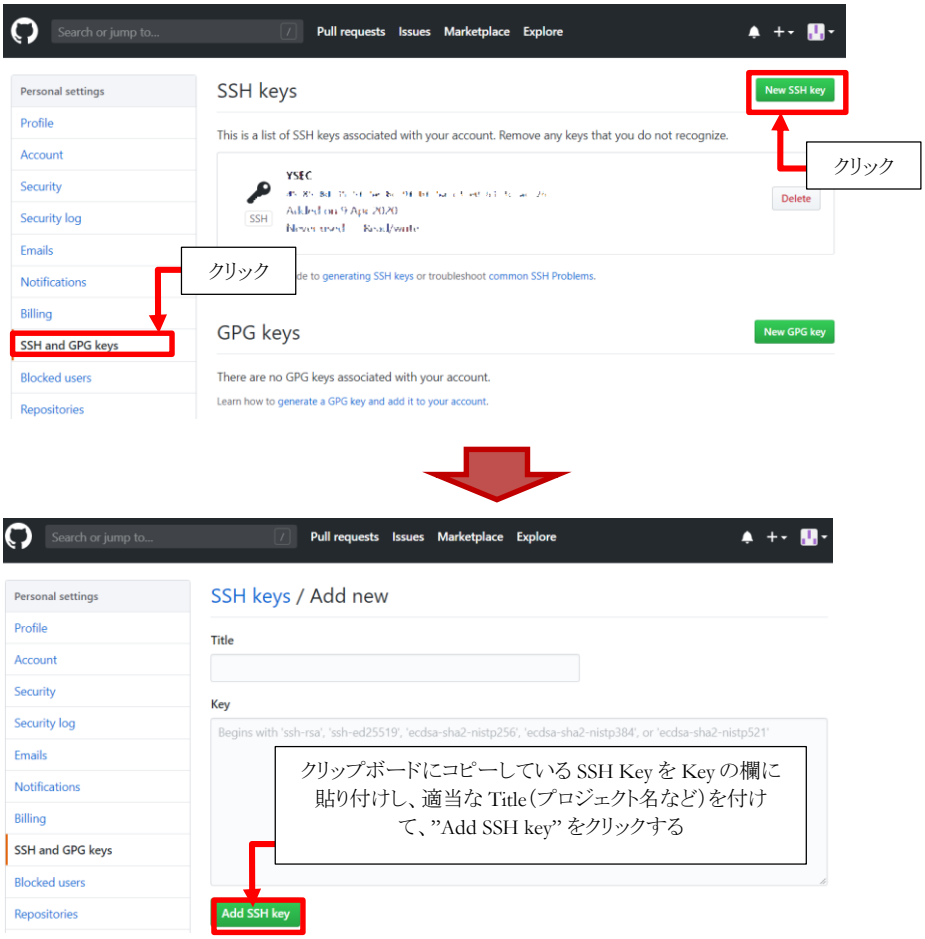
作成した SSH Key をクリップボードにコピーし、GitHub に、自分のアカウントでアクセスします。画面右上のメニューから、“Settings”をクリックします。

```
# xsel がインストールされていない場合は、“sudo apt-get install xsel” でインストールしておく
# Windows の場合は、clip < ~/.ssh/id_rsa.pub
# Mac の場合は、pbcopy < ~/.ssh/id_rsa.pub

$ cat ~/.ssh/id_rsa.pub | xsel --clipboard --input
```



つぎに、左側の “SSH and GPG keys” をクリックします。右上に表示されている “New SSH key” をクリックします。つぎに、クリップボードにコピーされている SSH Key を SSH key のテキストボックスに入力して、”Add SSH key” をクリ

ックします。このとき、GitHub のアカウント登録の時に作成したパスワードを求められるので、入力します。これで登録が完了です。



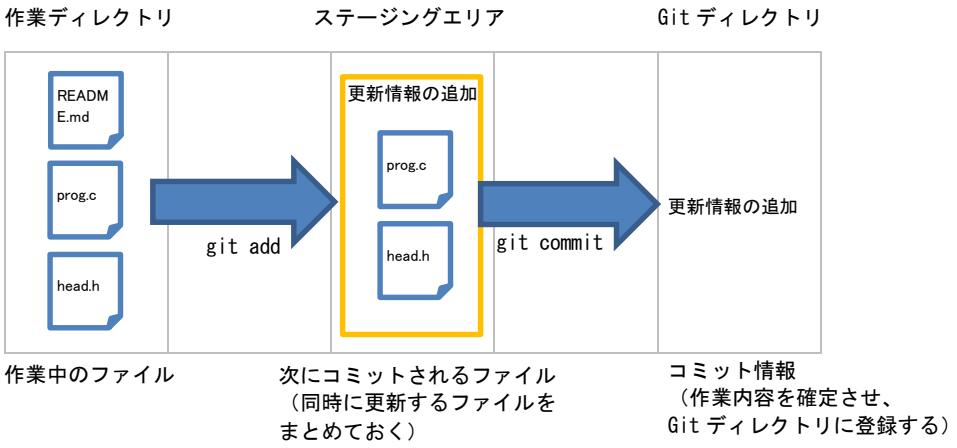
登録できていれば、以下のコマンドを実行すると、メッセージが出てきますので、「yes」と入力し、Enter キーを押します。つぎに、SSH Key のパスワード（passphrase）が求められるので、それを入力すると、「Hi (User Name)! You've successfully authenticated, but GitHub does not provide shell access.」と表示されます。これは、認証が成功したことを示しています。

```
$ ssh –T git@github.com

The authenticity of host 'github.com (13.114.40.48)' can't be established.

RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.

Are you sure you want to continue connecting (yes/no)?
```

Git の基本

ここでは、Git の基本操作について学びます。皆さんの PC 上の作業場所であるローカルリポジトリの作成、作業履歴の保存をするコマンド「コミット」などについて順に説明していきます。

まず、Git がどのようにバージョン管理しているのか、以下の図を見ながら理解しましょう。



コミットを行うことで、バージョンの状態を管理できるようになります。もし、開発途中に生じたバグによってプログラムを動作させることができなくなったときも、コミットの履歴をさかのぼることで前の段階（以前のコミットした状態）に戻すことが簡単にできるのです。

ローカルリポジトリの作成とコミット

まず、リモートリポジトリを clone コマンドで複製します。ローカル PC の任意の場所（WSL なら、ホームディレクトリ）に作業用ディレクトリとして、Git-test というディレクトリを作成します。このディレクトリに cd コマンドで移動後、前節で作成したリモートリポジトリの URL（https://github.com/test_usr/SoftEx20.git）を確認し、以下のコマンドを実行します。

```
$ git clone https://github.com/test_usr/SoftEx20.git
```

実行がうまくいくと、カレントディレクトリ（Git-test）内に、「SoftEx20」という名前のディレクトリが作成されているはずです。これが、ローカルリポジトリとなります。このディレクトリに移動し、ls –a コマンドで中身を確認すると、README.md や、「.git」という名前のディレクトリができていることが確認できます。「.git」は、Git の管理ディレクトリとなります。

次に、リポジトリの状態を確認します。以下のコマンドで確認してください。

```
$ git status
```

そうすると、以下のようなメッセージが出ます。

```
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean
```

これは、「master」ブランチ上で履歴管理が可能な状態であることを示しています。このように、clone コマンドでリモートリポジトリを手元の PC に複製することで、リモートリポジトリと同じ環境（同じファイル構成、状態）で作業できるようになります。

次に、ローカルリポジトリ上でのファイルを追加する場合ですが、その場（ここでは、~/Git-test/SoftEx20 上）でファイルを作成し、"add"コマンドを使って追加します。そのあと、"commit"コマンドを使い、作業履歴の保存をした後に、"push"コマンドで新しいファイルをリモートリポジトリに追加します。このとき、ユーザアカウント名とパスワードを聞かれますので入力してください。

以下、"test.txt"というテキストファイルを作成し、リモートリポジトリに追加する例を示します。

```
$ cat > test.txt

aaabbbcc ^C

$ git add test.txt

$ git commit –a –m "test.txt: new file（コミットメッセージ）"
                  リモート名   ブランチ名
$ git push origin master

Username for 'https://github.com': test_usr

Password for 'https://test_usr@github.com':
```

git add の使用例として、以下のようなものがあります。

```
$ git add .  #すべてのファイル、ディレクトリを追加

$ git add  *.txt  # すべてのテキストファイル
```

```
$ git add –u  # 変更されたファイルを追加する

$ git rm –cached  # add したファイルを除外する
```

git commit の使用例として、以下のようなものがあります。

```
$ git commit –a  # 変更のあったファイルすべてをコミット

$ git commit –amend  # 直前のコミットの取り消し

$ git commit –v  # 変更点を表示してコミット
```

コミットの取り消しをしたい場合は、以下のようにします。

```
$ git reset --soft HEAD~2  # 最新コミットから 2 件分をワークディレクトリ内容を保持し取り消す

$ git reset --hard HEAD~2  # 最新コミットから 2 件分のワークディレクトリ内容およびコミットを取り消す
```

ブランチとマージ

プログラムの開発において、規模が大きくなると、同時進行のタスクが複数出てきます。ソフト実験におけるゲーム開発でも、UI 担当、システム担当、ネットワーク担当、グラフィック担当などに分かれて、複数のタスクを同時並行で進めていくことになります。その場合、各タスクごとにブランチ(枝)を作ることで修正を繰り返し、最後にマージ(合体)する方法が効率的です。ここでは、ブランチの作成と、マージの方法について説明します。

まず、task1 という名前のブランチを作成してみます。git branch の後に引数を指定しない場合は、ブランチの一覧を見ることができます。ブランチ名の頭に*が付いているものが現在のブランチを示しています。

```
$ git branch task1
$ git branch
task1
*master
```

つぎに、作成したブランチ task1 にコミットするためにブランチを切り替えてみます。

```
$ git checkout task1
Switched to branch ‘task1’
```

task1 にチェックアウトした状態で、コミットを行うと、task1 ブランチに履歴が記録されるようになります。

```
$ git add README.md
$ git commit –m “README に追記”
[task1 5ef256a] README に追記
 1 file changed, 1 insertion(+), 1 deletion(-)
```

つぎに、ブランチをマージしていきます。task1 ブランチで変更した内容を master ブランチに統合します。

```
$ git checkout master
Switched to branch ‘master’
$ git merge task1
Fast-forward
 README.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

task1 ブランチの内容が master に統合された後、task1 ブランチは不要になりますので、削除します。

```
$ git branch –d task1
Deleted branch task1 (was 5ef256a).
```

プル

複数の場所（メンバー）でプロジェクト開発をしている際に、push できないときがあります。これは主に、ローカルリポジトリの内容が最新でないことが原因です。ローカルリポジトリの内容を最新情報にするためには、”pull”コマンドを用います。以下は、master の内容（リモートリポジトリ）をローカルリポジトリに反映させる例です。

```
$ git pull origin master
```

変更内容が重複（同じファイルを編集している状態など）していると、自動的にマージできなくなり、「コンフリクト（競合）」が起きます。pull したときにコンフリクトが起こると、以下のようなメッセージが出ます（直前に README.md ファイルを別のメンバーが編集したときなど）。

```
Auto-merging README.md
```

```
CONFLICT (add/add): Merge conflict in README.md

Automatic merge failed; fix conflicts and then commit the result.
```

このとき、コミットし直してからプッシュすることで、コンフリクトを解決します。このあと、pull すると、正しくマージできるようになります。

```
$ git add README.md

$ git commit –m “README.md を修正しました（松本）”

$ git push origin master
```

参考文献

1. 横田紋奈 (著), 宇賀神みずき (著)：「いちばんやさしい Git&GitHub の教本 人気講師が教えるバージョン管理＆共有入門」

2. 大塚 弘記 (著)：「GitHub 実践入門――Pull Request による開発の変革」

3. 大串 肇 (著), 久保 靖資 (著), 豊沢 泰尚 (著)：「Git が、おもしろいほどわかる基本の使い方 33 改訂新版」

4. リック・ウマリ (著), 吉川邦夫 (翻訳)：「独習 Git」

5. 冨永和人 (著)：「わかる Git」

6. サルでもわかる Git 入門： https://backlog.com/ja/git-tutorial/