Github導入マニュアル

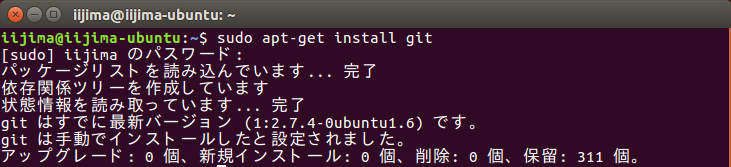
平成31年2月19日　飯島脩平

用語集が末尾にあるので、分からない単語はそこで確認を行ってほしい。

# Gitのインストールと設定

1. Gitのインストールを行うコマンドを打ち込む  
   コマンド：sudo apt-get install git

* 打ち込んだ後の様子

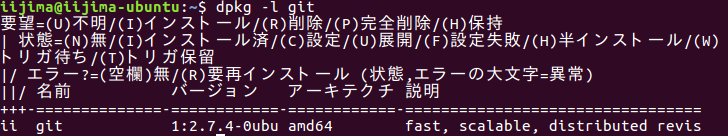


1. 念のため、インストールが出来たかの確認を行うコマンドを打ち込む

コマンド：dpkg -l git

打ち込んだ後にバージョンが表示されればインストール成功となる。

* 打ち込んだ後の様子



1. ユーザー名の設定を行うコマンドを打ち込む。【】内にメールアドレスを入力する。

コマンド：git config --global user.email 【メールアドレス】

1. ユーザー名の設定を行うコマンドを打ち込む。【】内にユーザー名を入力する。

コマンド：git config --global user.name “【ユーザー名】”

1. メールアドレスとユーザー名の設定が出来たかの確認を行う。

コマンド：git config --list

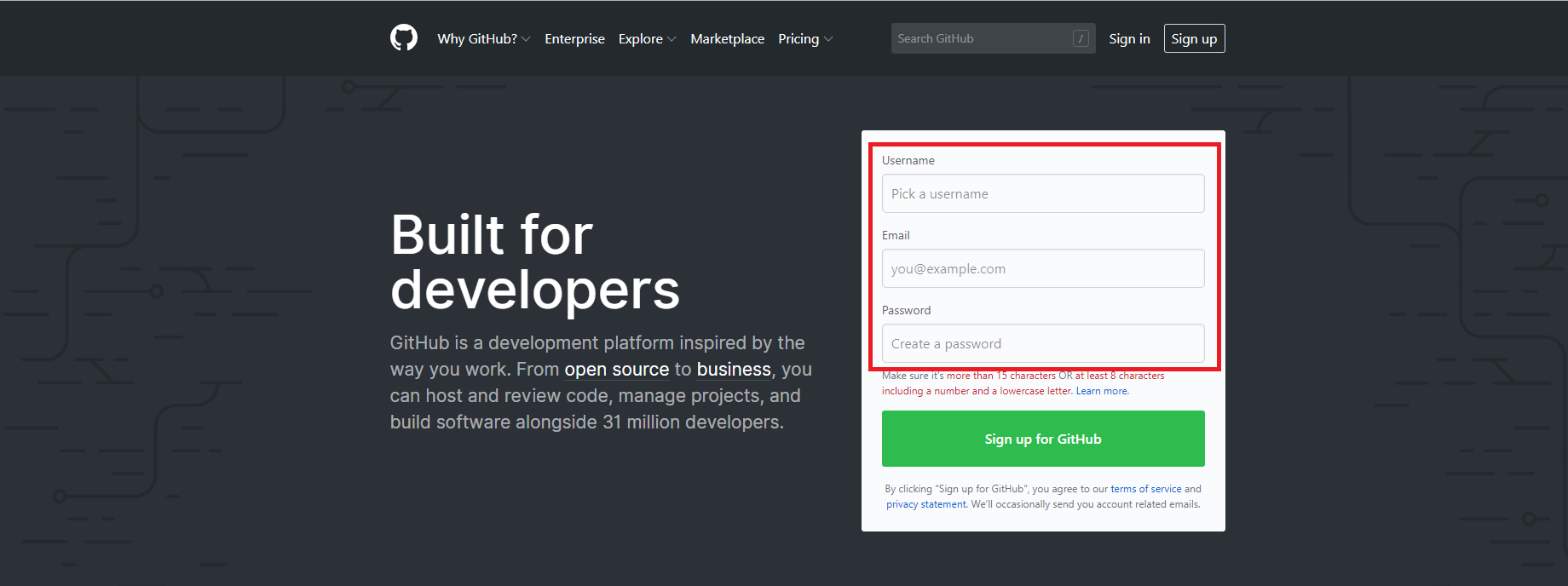
打ち込むと正しく登録されていれば下記の画像のようにユーザー名とメールアドレスが表示される。

* 打ち込んだ後の様子

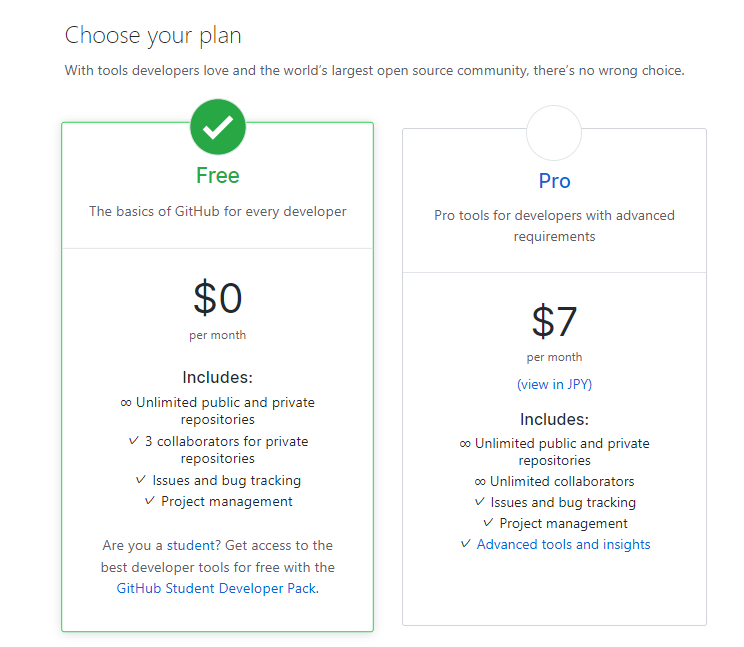


# GitHubの登録

1. GitHubのページ（https://github.com/）へ移動（ただし、端末は閉じない。）
2. 赤枠で示した部分に必要事項(ユーザー名・メールアドレス・パスワード)を入力し緑色の（Sign up for GitHub）ボタンを押す



1. 次に料金プランの選択を行う。今回は無料版を選択するので「Free」を選択する。



1. アンケートの回答を行う。
2. 登録したアドレスにメールが来るので認証する。

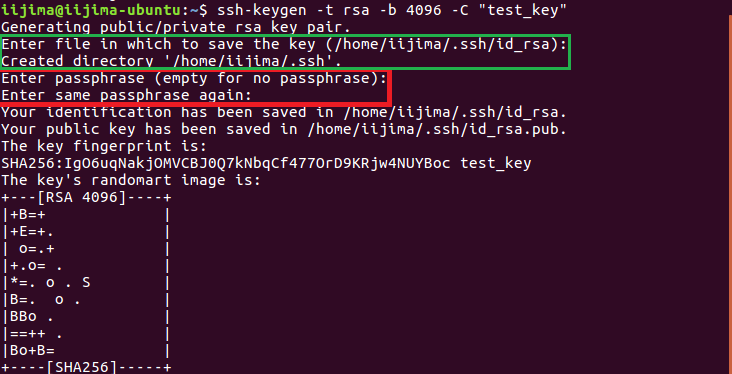
# SSH通信の確立

再びlinuxの「端末」に戻る。

1. SSH通信に必要な鍵の生成を行うコマンドを打ち込む

コマンド：ssh-keygen -t rsa -b 4096

* -tオプションで鍵タイプを指定
* -bオプションで鍵の長さを指定（GitHubの推奨が4096ビット）
* -Cオプションで鍵へのコメントを行える



1. 先ほどのコマンドを打ち込むと緑枠で示した「Enter file in which to save the key（パス）」が表示されるので「id\_rsa」というファイル名で良ければEnterを押し、そうでなければファイル場所の絶対パスを入力しEnterを押す。今回は変える必要がないのでそのままEnterを押す。
2. 前の作業を行うと赤枠で示したパスワード作成を要求されるので空のままでも生成は出来るが念のためパスワードを作成する。

▽入力画面の様子

Enter passphrase (empty for no passphrase): 【パスワードを作成。画面には表示されない】

Enter same passphrase again: 【パスワードを作成。画面には表示されない】

1. これまでの作業で鍵が出来ているかを確認するコマンドを打ち込む。

コマンド：ls ~/.ssh

打ち込むと下記の画像のように「id\_rsa」と「id\_rsa.pub」の二つが出来ているはずである。



1. 生成された鍵のパーミッションを「600」に変更するコマンドを打ち込む。

コマンド：chmod 600 ~/.ssh/id\_rsa

うまくいけば何も表示されずにコマンド入力が出来る状態になる。

1. パーミッション設定が正しく行われたか確認を行うコマンドを打ち込む。

コマンド：ls –la ~/.ssh/id\_rsa



正しく変更されていた場合は「-rw-------」になっている。

1. SSH通信のConfigファイルの設定を行うコマンドを入力する。

このConfigファイルはGitHubへSSH通信する際に必要となる。

コマンド：vim ~/.ssh/config

なお、Configファイルが存在しない場合にこのコマンドを入力するとConfigファイルが新規作成される。

* 下記の画面が出る場合（vimが未インストールの場合）



下記のコマンドを打ち込む。

コマンド：sudo apt install vim

このコマンドでvimがインストールされるはずであるので、インストール完了後に7の作業をもう一度行う。

* Vimのコマンド一覧

|  |  |  |
| --- | --- | --- |
| タイプ | コマンド | 内容 |
| ファイル操作 | Shift+z(2回入力) | 上書き保存し、vimを終了 |
| :w | 内容を保存 |
| :q! | 保存せずに終了 |
| モード | i | 挿入モードへ |
| Ctrl+[ | コマンドモードに戻る |

1. 先ほどのコマンドを打ち込むと編集画面になるので下記の内容を貼り付ける。

* 貼り付けるもの

Host github

HostName github.com

User git

IdentityFile ~/.ssh/id\_rsa

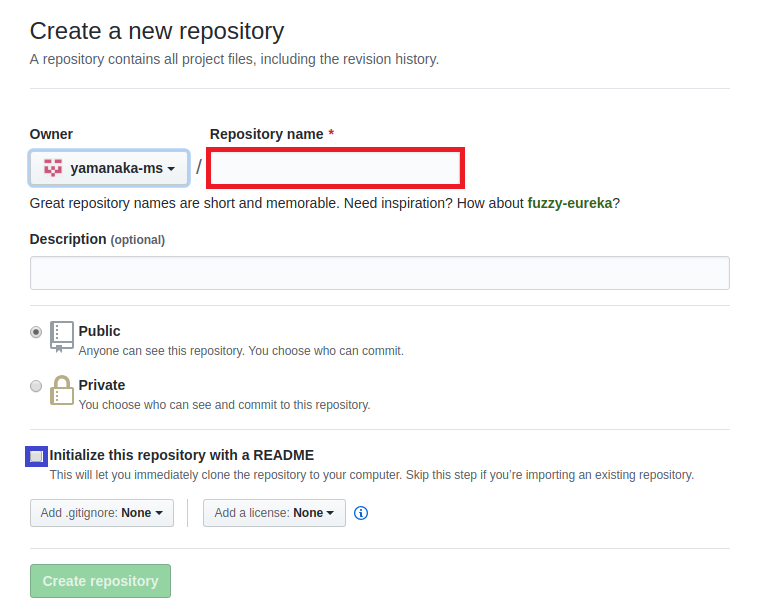
以上でPC側の設定は終了となり、GitHub上の設定を行っていく。

なお、上書き保存するためにはvimコマンドを用いるのであるが、編集中は「挿入」モードになっているので、vimコマンドを使えるように一度「コマンド」モードにする必要があるので注意。

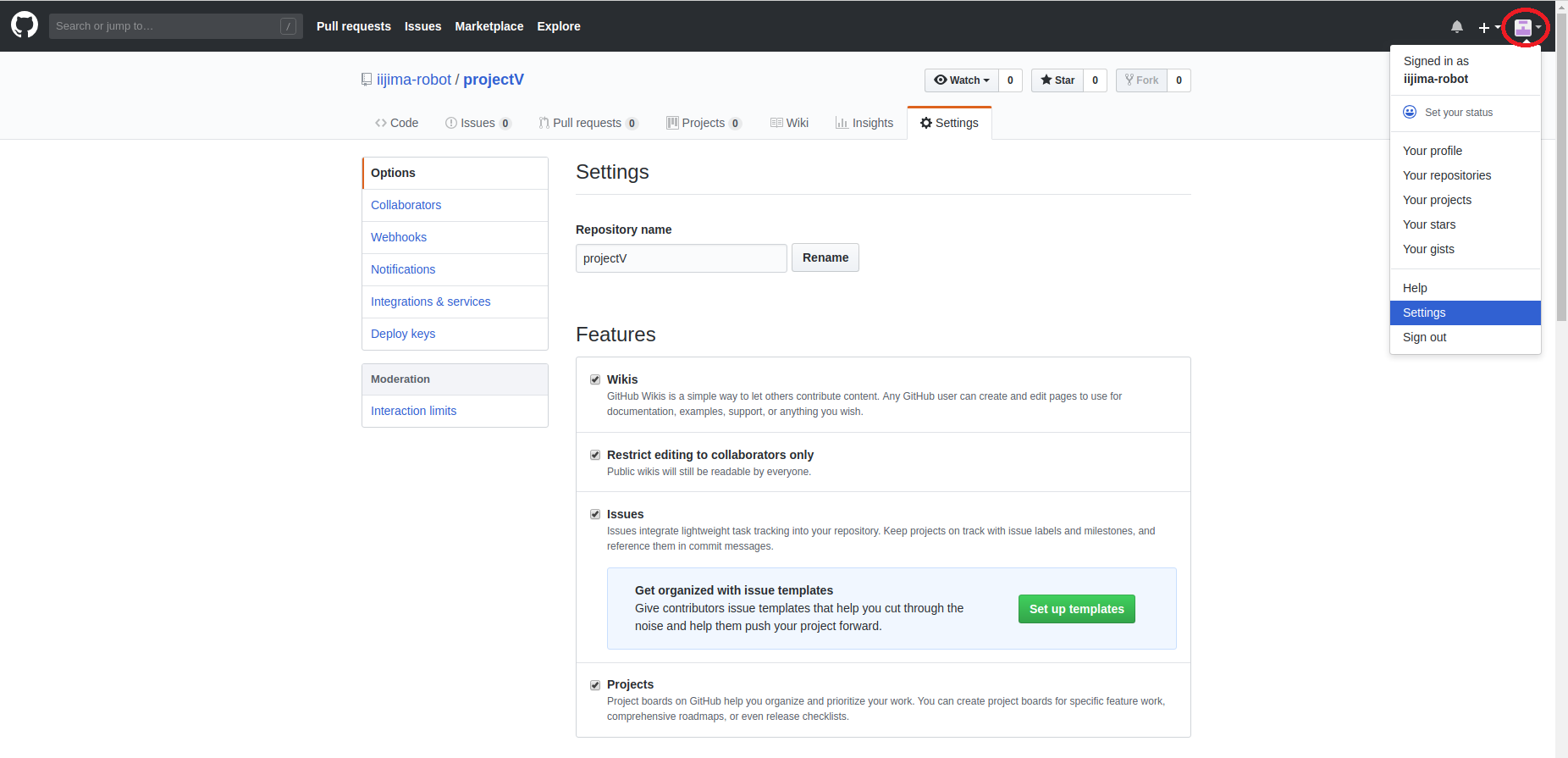
再びGitHubのページへ戻る。（ただし、端末は閉じない。）

1. リポジトリを作成する。

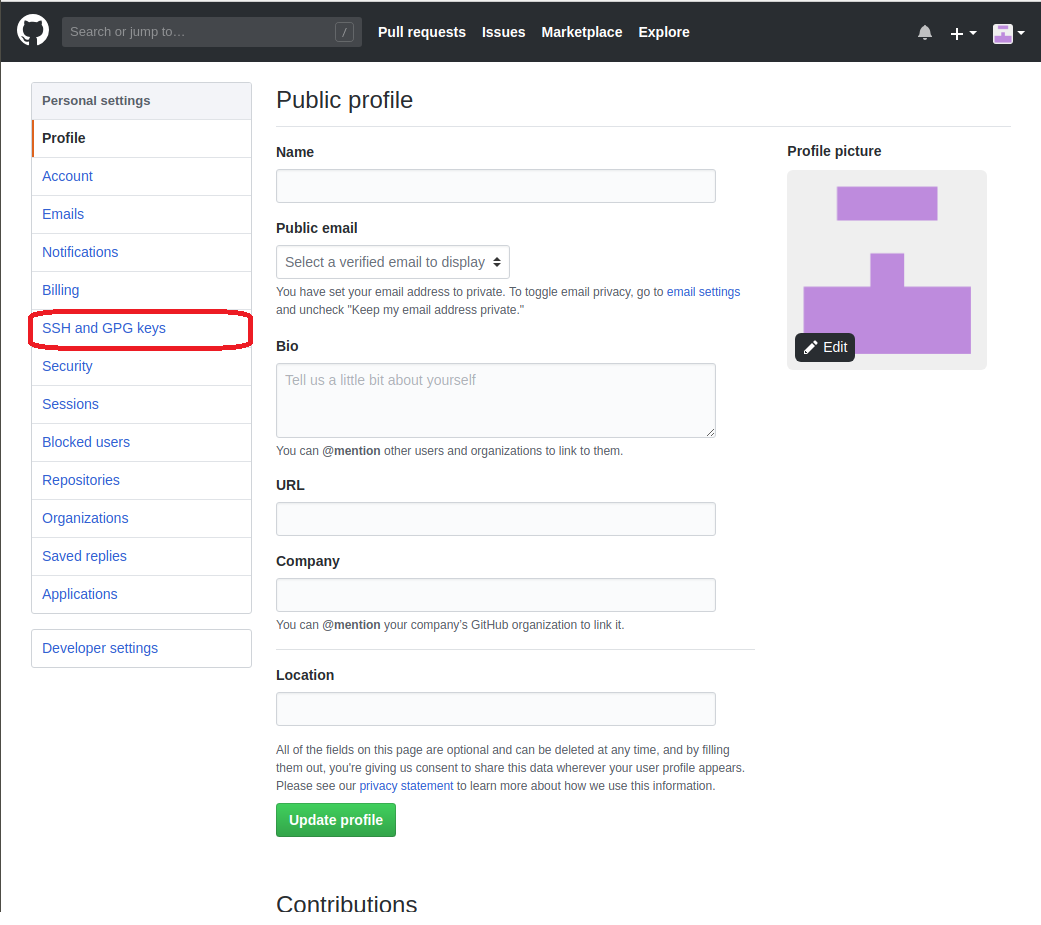
この際に＊がついている項目（赤枠で示した項目）は入力が必須の項目なので適宜入力を行う。また、「Initialize this repository with a README」（青枠で示した項目）にチェックをいれる。



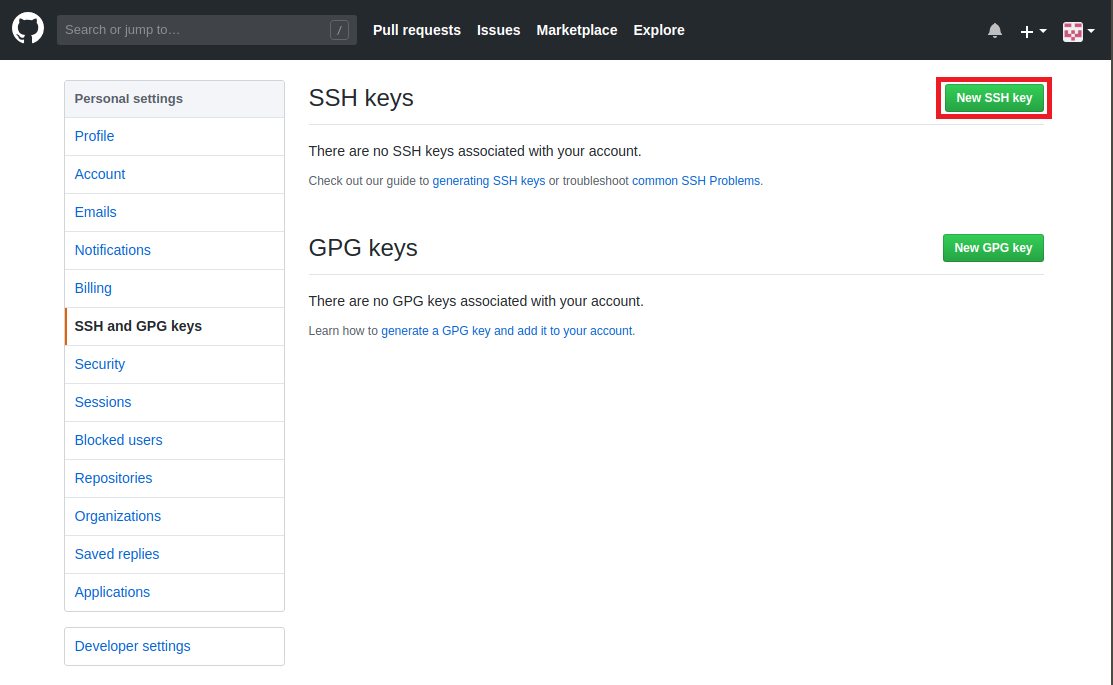
1. GitHubに公開鍵を登録する。
   1. 赤枠で囲まれた右上のアイコンを押した後、下記の画像のSettingボタンを押す。



* 1. プロフィール画面の左側のメニューから赤枠で示した「SSH and GPG keys」タブを選択する。



* 1. 右上の赤枠で示した「New SSH key」を押すと登録画面へ移行



* 1. 必要事項を下記の通りに入力する。

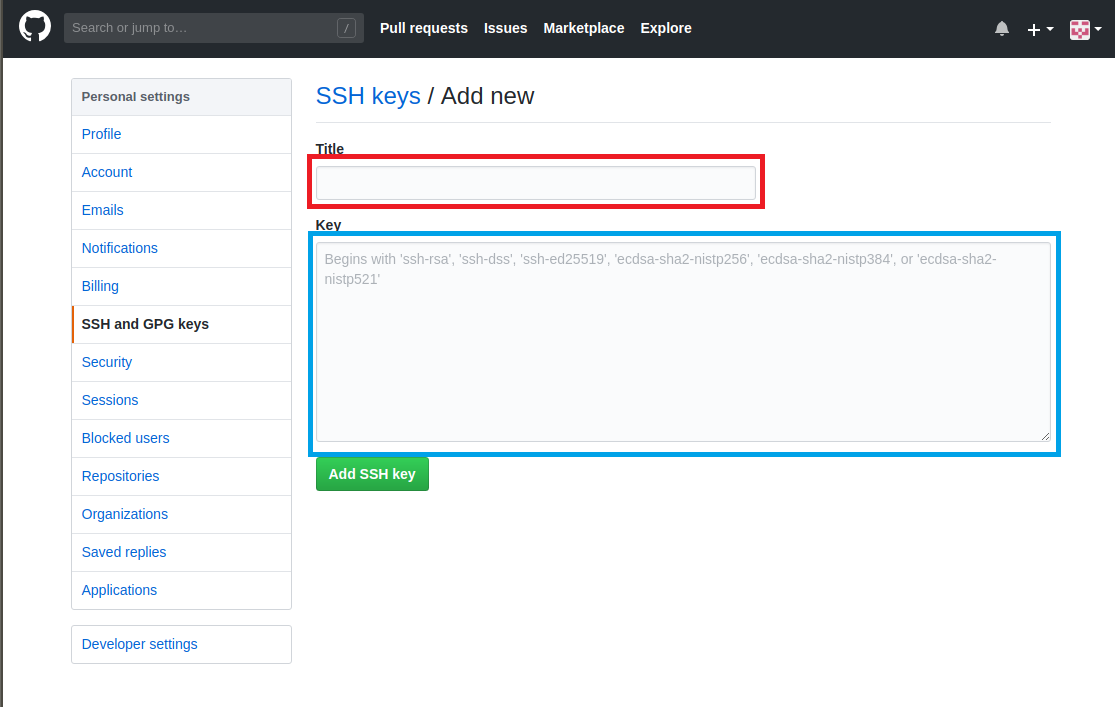
「Title」：任意の値を設定可能。後で区別出来るようにPC名などを設定する。赤枠で示した部分。

「Key」：「id\_rsa.pub」をエディターで開き、余分な改行などが入らないようにコピーしたものを貼り付ける。青枠で示した部分。

なお、「id\_rsa.pub」は先程まで作業していたディレクトリ(通常通りであれば「ホーム」フォルダ)の中にある「.ssh」フォルダ内に存在しているはずである。

また、Ubuntuのファイルマネージャは左側のバーに存在する下記の画像のアイコンを押せば起動できる。そして、隠しフォルダ（フォルダ名が「.」から始まるフォルダ）をファイルマネージャで見るためには「Ctrl＋h」を押す必要がある。





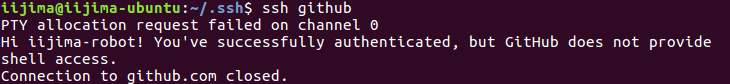
* 1. 緑ボタン「Add SSH key」を押す。

以上でGitHub上の設定は一旦終了である。再びlinuxの「端末」に戻る。

1. GitHubへSSH通信出来るかの確認を行うコマンドを打ち込む。

コマンド：ssh github

「Hi 【ユーザー名】 You've successfully authenticated, but GitHub does not provide shell access.」が下記の通り表示されたら成功となる。（うまくいかない場合は「id\_rsa.pub」ファイルへのコピーがうまくいってない可能性があるので、そこを確認してほしい。）



# GitHubの使用の確認（リポジトリのクローン）

1. 作業ディレクトリの作成を行い、作業ディレクトリへ移動する。場所は任意で構わない。使うコマンドは下記の通り。【】内に名前を入力する。

コマンド1：mkdir　【名前】

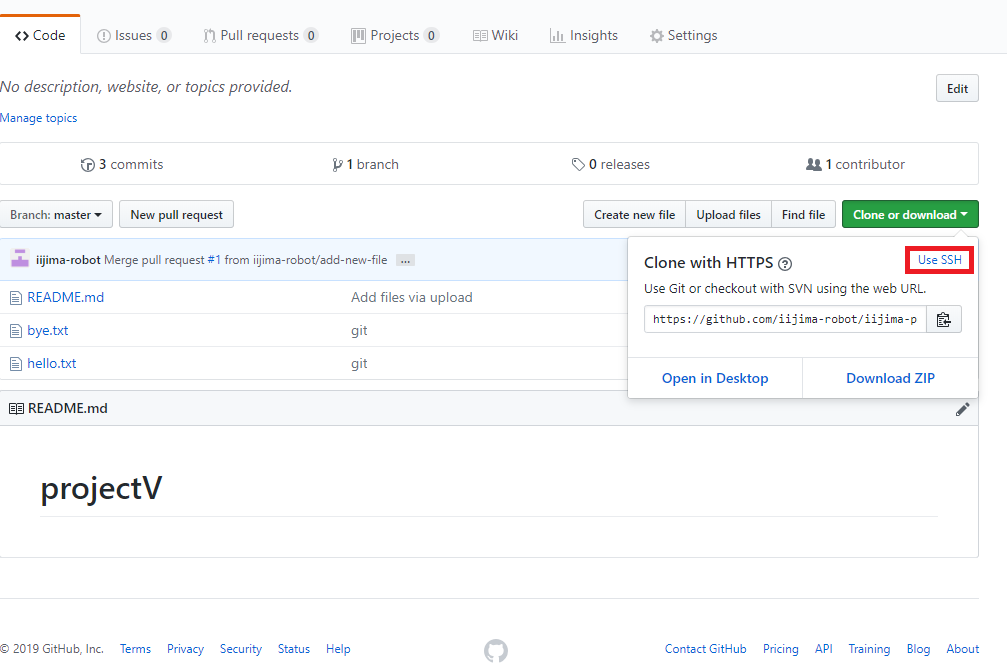
コマンド2：cd 【名前】

以下作業ディレクトリに移動した状態にあるものとする。

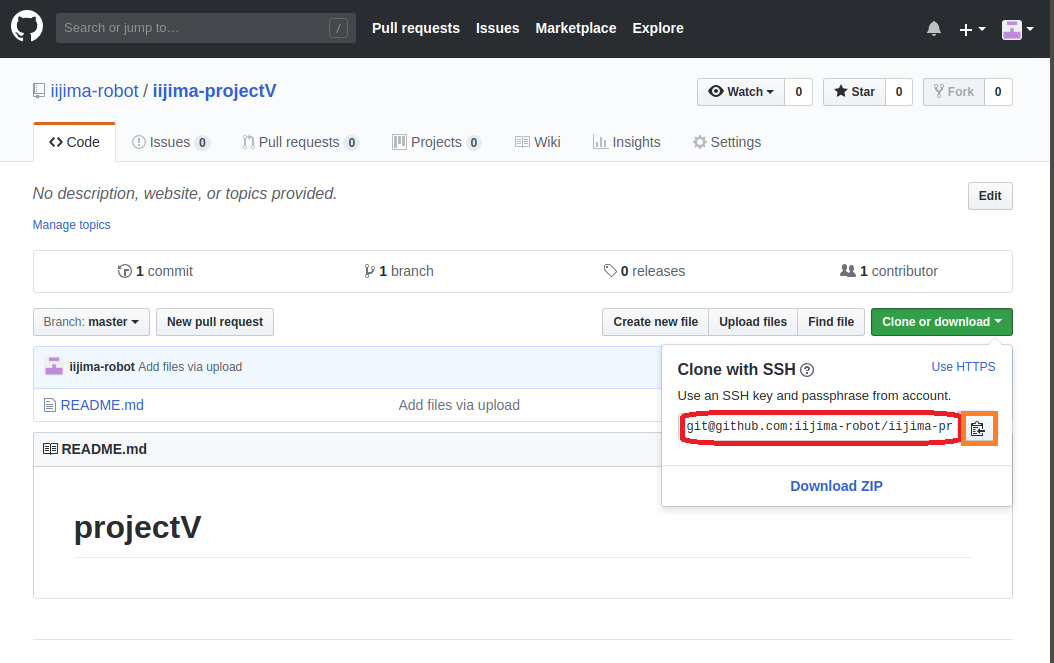
1. GitHubで作成したリポジトリ名をコピーする。

再びGitHubのページへ戻る。（ただし、端末は閉じない。）

* 1. GitHubのリポジトリのページの右側にある緑色ボタン「Clone or download」を押し、赤枠で示した「Use SSH」を選択する。もし、赤枠で示した部分が「Use HTTP」になっていた場合は押さなくてよい



* 1. 次に赤枠で示した文字列（git@github.com:【ユーザー名】/【リポジトリ名】.git）をコピーする。なお、オレンジで示した部分を押すと文字列をコピー出来る。



ここでコピーした文字列は次の作業時に必要となる。

GitHub上の設定は一旦終了である。再びlinuxの「端末」に戻る。

1. リポジトリをローカルにクローンするコマンドを打ち込む。【】内に2でコピーした文字列を貼り付ける。

コマンド：git clone 【2でコピーした文字列】

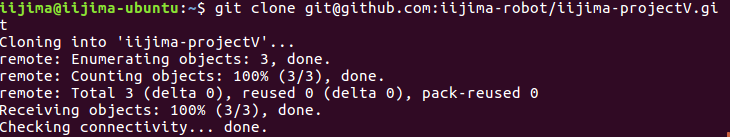
成功すると下記の文字列が出力される。

Cloning into '【リポジトリ名】'...

remote: Counting objects: 3, done.

remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0

Receiving objects: 100% (3/3), done.



1. ローカルリポジトリに移動する。【】内にリポジトリ名を入力する。

コマンド:cd 【リポジトリ名】

以下ローカルリポジトリに移動している前提で進める。

# GitHubの使用の確認（ブランチの使い方）

1. 現在使用しているブランチの確認を行うコマンドを打ち込む

コマンド：git branch

成功すると

\* master

と表示されるはずである。



1. ブランチの作成を行うコマンドを打ち込む。【】内にブランチ名を入力する。なお、今回はブランチ名をadd-new-fileに設定する。

コマンド：git branch 【ブランチ名】

1. 1と同様に現在使用しているブランチの確認を行うコマンドを打ち込む。

コマンド：git branch

成功すると

add-new-file

\* master

と表示されるはずである。

＊マークが表示されるブランチが現在作業しているブランチとなる。



1. 作業を行うブランチへ移動するコマンドを打ち込む。【】内にブランチ名を入力する。なお、今回はブランチ名をadd-new-fileに設定してある。

コマンド：git checkout 【ブランチ名】

成功すると

Switched to branch 【ブランチ名】

が表示される。



1. 3と同様に現在使用しているブランチの確認を行うコマンドを打ち込む。

コマンド：git branch

成功すると

\* add-new-file

master

と表示されるはずである。



# GitHubの使用の確認（ファイルのコミット）

1. どのような方法でも良いのでテキストファイルを生成する。ただし、今回のファイル名は「hello.txt」として説明を行う。

●　テキストファイルの生成方法の具体例

テキストファイルの生成方法の具体例として「vim」を用いた方法を紹介する。

* + - 1. vimを起動するコマンドを下記の通り打ち込む。ただし、【】内にファイル名を入力する。

コマンド：vim 【ファイル名】

* + - 1. vimが起動するので、テキストエディタと同様に編集作業を行う。

vimのコマンド一覧はSSH通信の確立の節で紹介しているので必要に応じて確認してほしい。

* + - 1. 編集作業を終えたら、コマンドモードに戻るために「Ctrl+[」を入力する。
      2. 最後に上書き保存を行うため、「Shift+z」を2回入力する。

1. 作成したファイルをバージョン管理の対象（Gitのコミット対象）に追加するコマンドを打ち込む。【】内にファイル名を入力する。なお、今回はファイル名をhello.txtに設定してある。

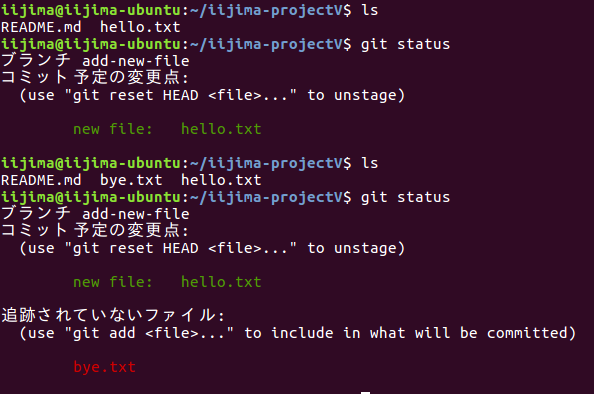
コマンド：git add 【ファイル名】

成功しても特に表示は行われない。

1. コミット内容を確認するコマンドを打ち込む。

コマンド：git status

成功すると赤枠で示した通りになり、addされてないファイルがあると緑枠で示した通りになる。



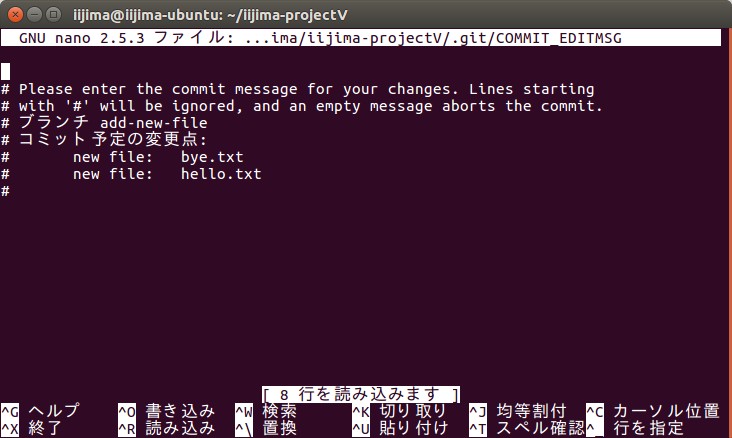
1. コミットするコマンドを打ち込む。（addされた状態でコミットする）

コマンド：git commit –a

* **-a**オプションはここまでの修正をすべてコミットさせるためのコマンド

実行するとエディターが開かれる。なお、ここではエディターとしてGNU nanoを使用した。

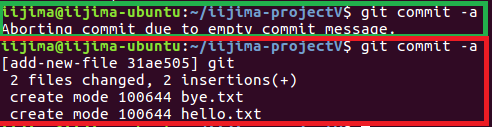
1. コミットに対するコメントの編集を行う。なお、行頭の「＃」が正しく認識されないエラーがあるので、最初の「＃」の前に何らかの文字列を入れる。



* + - GNU nanoのコマンド一覧

|  |  |
| --- | --- |
| コマンド | 内容 |
| F1 | ヘルプ |
| F2 | 終了(保存は行われない) |
| F3  （ファイル名が画面下部に反転表示されたらEnterを押す。） | ファイル書き出し |
| F5 | ファイル読み込み |

正しく行われると赤枠で示した通りになり、失敗すると緑枠で示した通りになる。



なおこの段階で「git status」を入力すると下記の通りになり、コミットが完了していることがわかる。

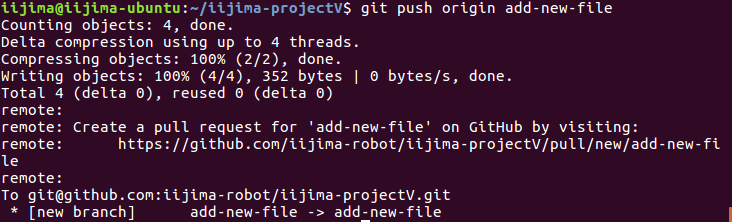


# GitHubの使用の確認（GitHubへのPush）

1. GitHubへプッシュを行うコマンドを打ち込む。【】内にブランチ名を入力する。なお、今回はブランチ名をadd-new-fileに設定してある。

コマンド：git push origin 【ブランチ名】

成功すると下記の画像のようになる。



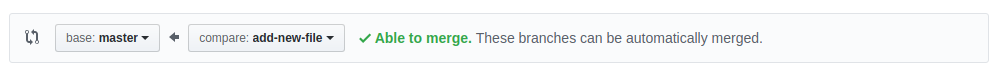
# GitHubの使用の確認（GitHubでプルリクエスト）

今回はGitHubの画面からプルリクエストを行う作業をしてみる。

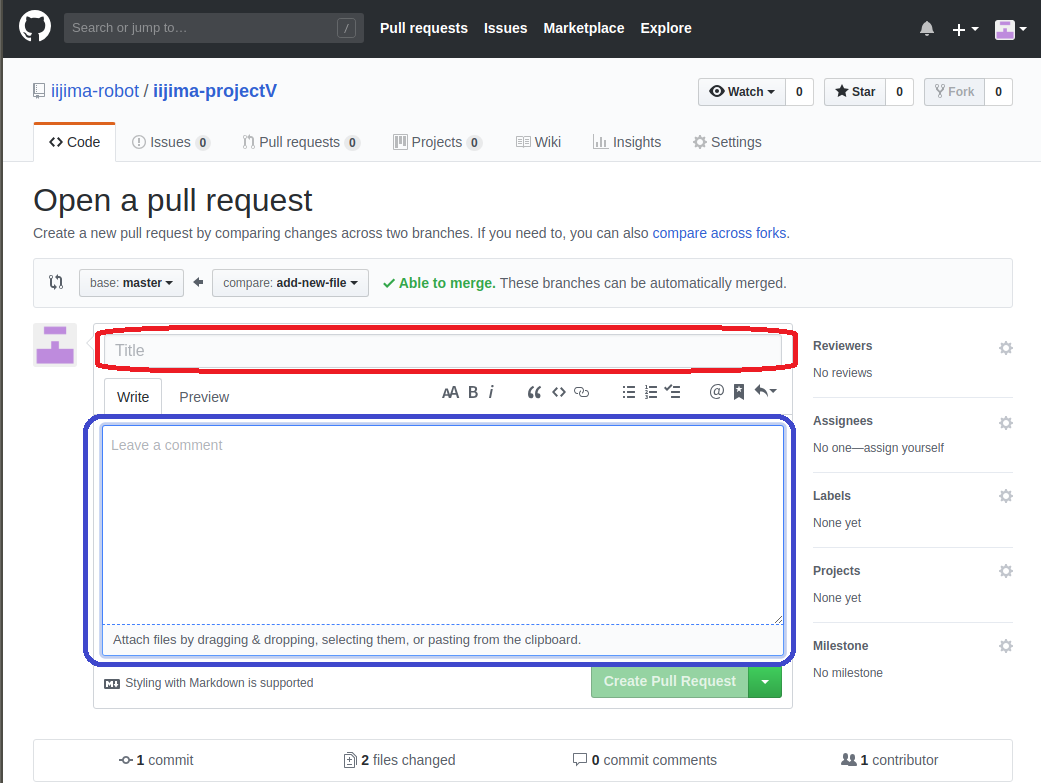
1. GitHubの画面からプルリクエストを出すために緑色のボタン「Compare & pull request」を押す



1. 新たにコミットした「add-new-file」ブランチから「master」ブランチへ向けてプルリクエストを送るため、「base」（リクエスト受信側）に「master」ブランチ、「compare」（リクエスト送信側）に「add-new-file」ブランチとなっていることを確認する。



1. 赤枠で示したプルリクエストのタイトルと青枠で示したプルリクエストの内容の記入を行う。



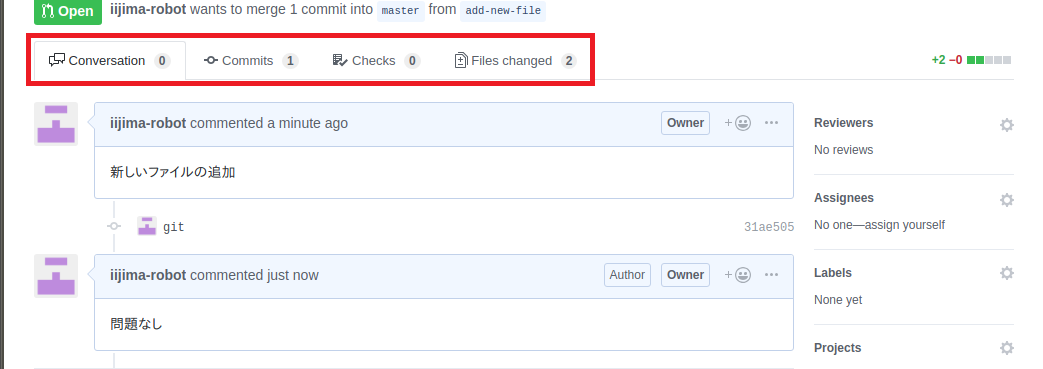
1. 記入が出来ると右下の「Create pull request」（先ほどの画像の青枠の右下）が押せるようになるのでそこを押す。

押すと、プルリクエストのページが作成されそのページへ遷移する。

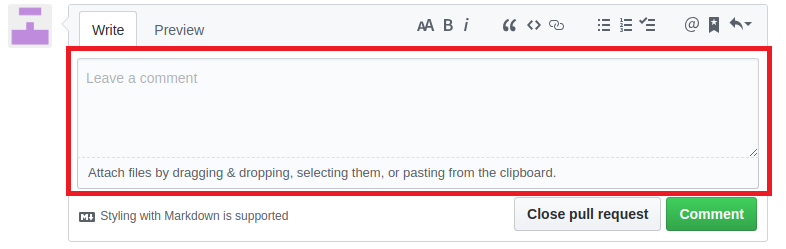
1. プルリクエストの確認を以下の３点に注意して行う。

* 「Conversation」：プルリクエストの説明
* 「Commits」：どんなコミットが行われたか
* 「Files changed」：ファイルにどんな差分があるのか

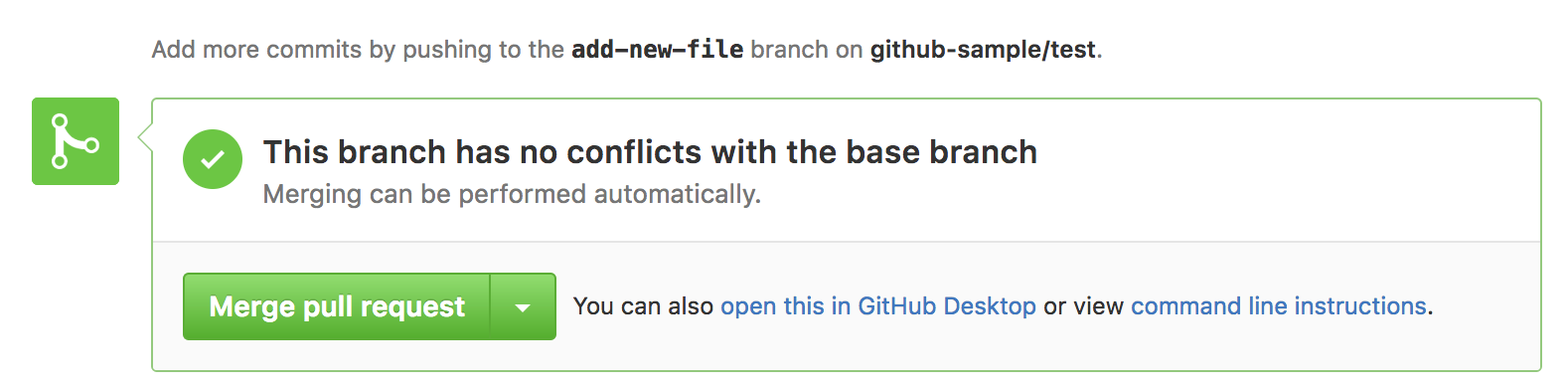
なお、上記３点は赤枠で示したタブから選択できる。



1. コメントを送ってみる。コメントは赤枠で示した場所に記入し、コメントが出来たら「Comment」をクリックする。



1. プルリクエストのコメントのやり取りが終了したらプルリクエストを終了させるために「Merge pull request」を押す。

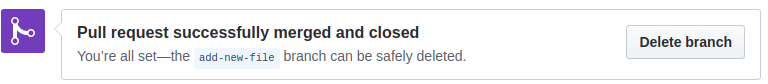


1. マージする際のコメントを赤枠で示した部分に入力し、「Confirm merge」を押す。



今回はadd-new-fileからmasterへのプルリクエストであったので、add-new-fileがmasterにマージされる。

1. 不要になったadd-new-fileブランチを削除するために「Delete branch」をクリックしブランチを削除する。



# GitHubの使用の確認（ローカル環境を更新する）

ここからはローカル環境の作業になるので、端末へ移動する。なお、カレントディレクトリはあらかじめ作業ディレクトリに移動しておく。

1. 現在のブランチを確認するコマンドを打ち込む。

コマンド：git branch

成功すると

\* add-new-file

master

と表示されるはずである。

1. 作業を行うブランチへ移動するコマンドを打ち込む。【】内にブランチ名を入力する。なお、今回作業するブランチはmasterである。

コマンド：git checkout 【ブランチ名】

成功すると

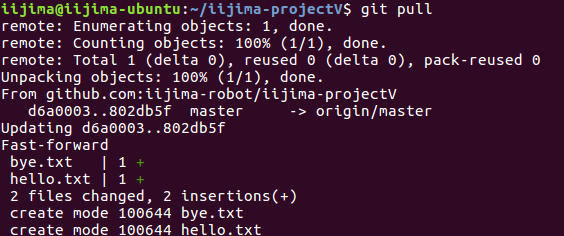
Switched to branch 【ブランチ名】

が表示される。

1. masterブランチをGitHubのmasterブランチに更新するコマンドを打ち込む。

コマンド：git pull

成功すると下記の画像のようになる。



これでGitHubのmasterブランチと同じにすることが出来た。

1. ローカル環境の作業ブランチ（add-new-fileブランチ）が不要になったので、作業ブランチを削除するコマンドを打ち込む。【】内に作業ブランチ名を入力する。なお、今回は作業ブランチ名をadd-new-fileに設定してある。

コマンド：git branch -d 【作業ブランチ名】

成功すると下記の画像のようになる。



ちなみに、マージを行う前にブランチを削除しようとすると

error: The branch 'add-new-file' is not fully merged.

If you are sure you want to delete it, run 'git branch -D add-new-file'.

が表示されてエラーとなる。

こういう場合は-dオプションではなく-Dオプションを使い削除する。

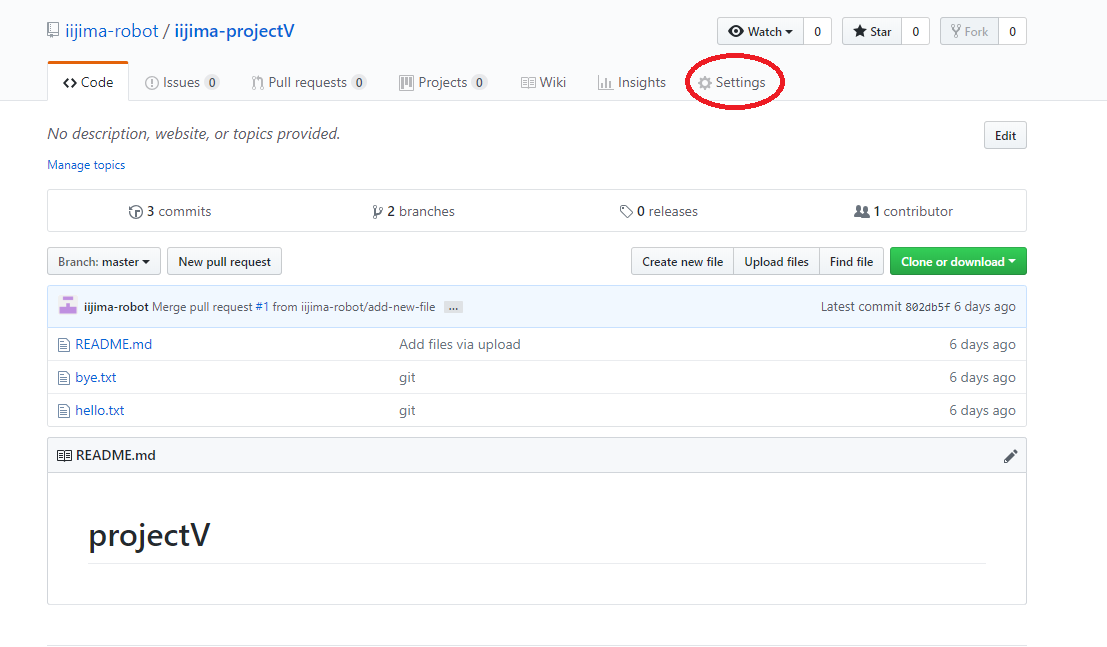
コマンド：git branch -D 【作業ブランチ名】

これにて個人で行うことのできる作業は終了となる。下記の内容は複数人で行うものである為、適宜行ってほしい。

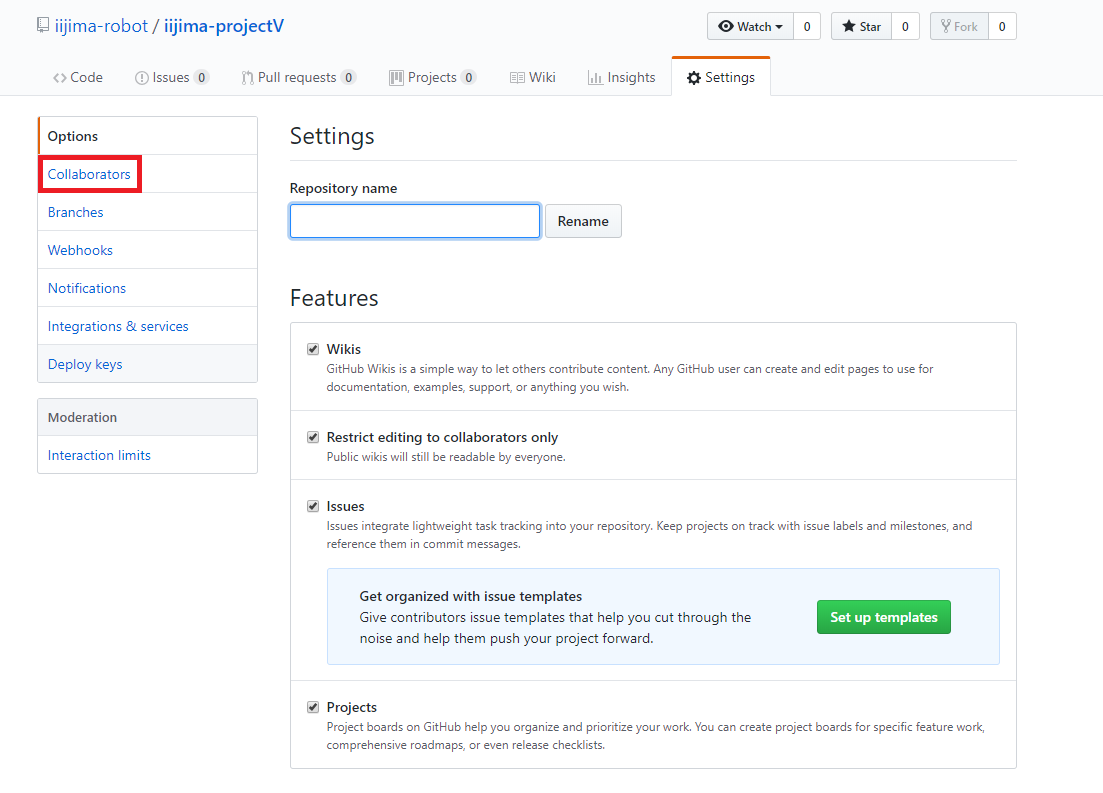
# GitHubの使用の確認（他人のリポジトリとの共有）

GitHubの機能には他人とコードなどを共有するものがあり、共有の方法として2つある。そこで今回は自分のリポジトリに他人を招待して共有する方法を採用し、本節ではこの作業を行う。（つまり、「Fork（フォーク）」を行う方法は行わない。）

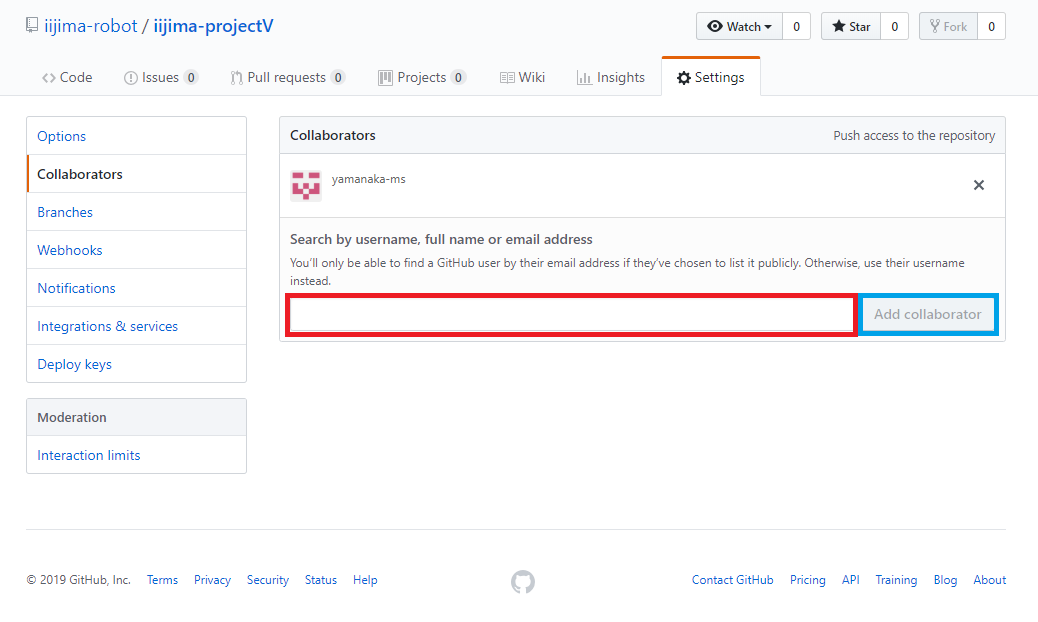
1. GitHubの自分のリポジトリへ移動し、上部に存在する赤枠で示した「Settings」を押す。



1. 左側のバーにある赤枠で示した「Collaborators」を押す。



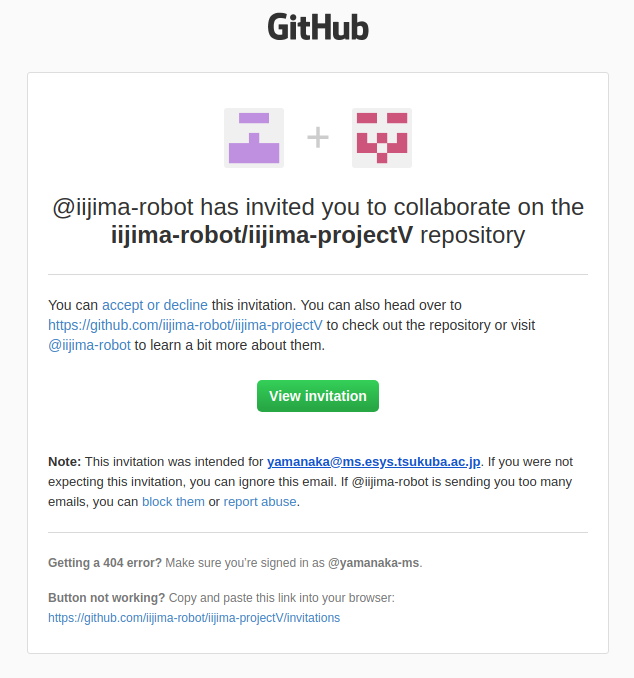
1. 中央にある赤枠で示した部分に「ユーザー名」もしくは「メールアドレス」を入力し、青枠で示した「Add Collaborator」を押す。



これにてこちら側の設定は終了となり、あとは相手から招待の承認が行われれば招待することができる。

# GitHubの使用の確認（他人のリポジトリからの招待）

1. 他人からのリポジトリに招待されるとGitHubに登録したメールアドレスにメールが届くので、届いたメール中央の緑色ボタン「View invitation」を押す。



1. 先程の作業を下記の画像のような画面に遷移するので緑色のボタン「Accept invitation」を押す。



以上で作業は終了となる。

# Githubの用語集

|  |  |
| --- | --- |
| 用語 | 意味 |
| リポジトリ | データを保存する場所。変更履歴もここに保存される。 |
| repository |
| リモートリポジトリ | サーバー（GitHub）上のリポジトリ。 |
| remote repository |
| ローカルリポジトリ | 自分のPC上に存在するリポジトリ。 |
| local repository |
| クローン | リポジトリを複製。 |
| clone |
| ブランチ | 一連の変更を主流から枝のように分岐させて管理するもの。違うブランチ同士の変更内容は互いに影響されないので、同時に複数の開発を行うことが出来る。 |
| branch |
| チェックアウト | 他のブランチに移動すること。（作業するブランチを変更すること。） |
| checkout |
| コミット | 変更した内容をローカルリポジトリに反映すること。 |
| commit |
| プッシュ | ローカルリポジトリの変更内容をリモートリポジトリに反映すること。 |
| push |
| プル | リモートリポジトリの変更内容をローカルリポジトリに反映すること。 |
| pull |
| マージ | あるブランチの変更内容を別のブランチに取り込むこと。マージ先とマージ元の両方の履歴が残る。 |
| merge |
| コンフリクト | マージしようとしたブランチにおいて変更内容が重複し、マージ出来なくなること。 |
| conflict |

# 参考文献（平成31年2月26日確認）

* 今日からはじめるGitHub 〜 初心者がGitをインストールして、プルリクできるようになるまでを解説　<https://employment.en-japan.com/engineerhub/entry/2017/01/31/110000>
* ssh-keygen - 認証用の鍵を生成 - Linuxコマンド <https://webkaru.net/linux/ssh-keygen-command/>
* 【GitHub超初心者入門】この前初めてGitHubを使い始めたエンジニア見習いが書くGitHubの使い方と実践～とりあえず一緒に動かしてみようぜ！～ <https://qiita.com/nnahito/items/565f8755e70c51532459>
* 【GitHub入門】超初心者から使い方をマスターしよう！ <https://www.sejuku.net/blog/8224#GitHub_GitHub>
* よく使う Vim のコマンドまとめ <https://qiita.com/hide/items/5bfe5b322872c61a6896>
* nano エディタの使い方　<http://dotnsf.blog.jp/archives/1066662653.html>
* Ubuntu/ファイルマネージャで隠しファイル・隠しフォルダを表示する方法<https://linux.just4fun.biz/?Ubuntu/%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB%E3%83%9E%E3%83%8D%E3%83%BC%E3%82%B8%E3%83%A3%E3%81%A7%E9%9A%A0%E3%81%97%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB%E3%83%BB%E9%9A%A0%E3%81%97%E3%83%95%E3%82%A9%E3%83%AB%E3%83%80%E3%82%92%E8%A1%A8%E7%A4%BA%E3%81%99%E3%82%8B%E6%96%B9%E6%B3%95>
* GitHubで特定のリポジトリを複数人で共有する方法　<https://reasonable-code.com/github-collaborators/>