Git を使ってみよう

【前準備】

私のリポジトリにテスト用のブランチを作成したので、何やっても大丈夫です

- https://github.com/keillera/test_repository

Githubのアカウントは事前に取得しておいてくださいm(__)m

作業用リポジトリ作成(fork)

- https://github.com/keillera/test_repository へアクセス
- 右上の Fork をクリックして、リポジトリが自分のアカウント配下に複製されるのを待つ
- （今後作業は複製された先に対して行い、修正が完了したら pull request をkeillera アカウントに対して送る）

ローカルリポジトリを作成(clone)

- 自分のリポジトリのデータをローカル環境に持ってくるリポジトリを管理したいディレクトリに移動後、以下のコマンドを実行。test*repository*ディレクトリが作成されてれば成功

  *git clone https://github.com/username/test*repository

- fork元のリポジトリのアドレスに別名を登録。test*repository* ディレクトリに移動

  *git remote add test*repo https://github.com/keillera/test_repository

【修正作業】

fork元のmasterデータでローカルのmasterリポジトリを最新化する

- git checkout master
  master に移動
- git pull test_repo master
  fork元のmaster データでローカルのmasterを最新化
- git push origin master
  最新化したローカルのmasterを自分のリポジトリにpush

修正用のブランチを作成する

- git branch add*conf*
  修正用のブランチを *add*conf という名前で作成

- git checkout add_conf
  ローカル環境を作成したブランチに切り替える

※ 上記の２つの作業は"git checkout -b"を使うと一度に行えます

- git checkout -b add_conf

よしなに修正等を行いましょう

- vi hoge.txt

修正が完了したらコミット対象となるファイルをインデックス（ステージとも呼ばれる）に追加

- git add hoge.txt
  ※ "git add . "とするとカレントディレクトリ配下全てを追加

インデックスに追加したものをコミット

- git commit -m "hogehoge"
  -m オプションでコミットメッセージを追加できます。
  必ず書きましょう。

マージ作業（修正中にファイルが更新されていた場合マージ作業が必要です）

- git pull test_repo master
  最新データ(fork元のmaster)からデータを取得しマージ

※ ここで衝突が起きた場合は、衝突内容を確認後"よしなに"修正し、再度「git add」→「git commit」を行う必要があります。

マージ作業２

git pull コマンドは fetch と merge の合わせ技だったりします。
詳細は下記等を参照してください。

http://qiita.com/osamu1203/items/cb94ef9da02e1ec3e921

自分のリポジトリに修正内容を反映

- git push origin add_conf
  forkしたリポジトリに push

pull requestを送ろう！

- https://github.com/username/test_repository
  ブラウザから自分のリポジトリにアクセス
- 緑色の pull request ボタンを押す
  何かあればコメントも書きましょう

ローカルの branch について

pull request がマージされたらローカルのbranch(add_conf)は削除しちゃっておｋです。

別途修正を行う場合は、再度ブランチを作り直しましょう

- git branch -d add_conf
  ローカルのブランチを削除

- git push origin :add_conf
  ブランチを削除したことを、自分のリポジトリに反映

以降は、【修正作業】と【pull request】の繰り返しとなります。

修正作業の流れについては最初面倒なように感じますが、1つ1つに意味があります。また、１回読んだだけではわからないと思うの３回くらい読みなおしてみてください。

【メモ】

よく使うコマンド

- git help
- git なんかのコマンド help
- git status
- git diff
- git diff --cached

【勝手にxxメモ-抜粋】

- git status
  現在のローカル環境の状態を表示してくれます
  何 add してたっけ？
  何 commit してたっけ？
  今どこのブランチにいるんだっけ？
  等の疑問がわいたら、これを叩きましょう

- git checkout mod*am*iversion
  ここで作業するよー、のコマンド
  イメージ的にはLinuxのcdコマンド

【勝手に柿本メモ】

git branch -v
git remote -v
git reset —-soft HEAD^

直前のコミットだけを無かった事にする

git status で確認すると元に戻ってる

—-hardにすると変更まで戻る？