非情報系理系院生のための モダンな開発環境づくり

その 1. Git/GitHub を使ったソースコード管理

荒木 亮

阪大院基礎工・後藤研

October 4, 2019

もくじ

目標

ソースコードや LaTeX 文書のバージョンを，Git/GitHub で管理する

- Git に初めて触れる人が，「ファイル名に日付をつけてバックアップ」を Git で代替できるようになることをめざす

説明しないこと

- `branch` を使った同時並行的な開発
- `pull request` を使ったチームでの開発
- その他色々ややこしいコマンド

Git の構造

- 作業ツリー
- ステージングエリア
- ローカルリポジトリ
- リモートリポジトリ

を表す図を追加

Git の構造

作業ツリー

作業をおこなうディレクトリ．普通の意味の「フォルダ」

ステージングエリア

作業ツリーで編集したファイルの変更点を「とりあえず」保管しておく場所

ローカルリポジトリ

ステージングエリアにある変更点を「コミット」としてまとめ，記録する場所

リモートリポジトリ

ソースコードなどをまとめた「リポジトリ」を保存・公開する場所

Gitのコマンド

`git add`

作業ツリーで編集したファイルの変更点をステージングエリアに追加

`git commit`

ステージングエリアにある変更点の集合をローカルリポジトリに記録

`git push`

ローカルリポジトリに記録されたコミット群をリモートリポジトリに送信

`git pull`

リモートリポジトリの最新の情報をローカルリポジトリに適用

なんでこんな難しいんや：分散型バージョン管理

クラウドサービスと Git の
差異を表す図

自動更新（クラウドサービス）

変更箇所を自動でリモートに送る

commit（Git/GitHub）

変更箇所を任意の粒度でまとめ，名前やコメントをつけて管理するもの

- 編集履歴がわかりやすい
- 昔の状態の確認や巻き戻しが簡単
- 編集が競合したとき，解決が簡単

なぜGitを使うのか

- 自動で保存されてしまうクラウドサービスだと…

  「前の状態に戻したいんだけど，あれっていつのバージョンだっけ？」
- 古い状態を記録しておけないと…

  「今はこの関数やサブルーチンを使ってないけど，また使うかもしれないしとりあえずコメントアウトして残しておこう」
- 共同作業で競合がおきてしまうと…

  「今からこのファイル編集するから触らないでね！」

  「最新のファイルってどこに置いてる？」

Gitを使うべきでないファイル

- 一度作成したらもう編集しないデータ（実験結果，計算データ）

  →（普通の意味での）バックアップで対処する

今日から始めるGit生活

1. GitHubでアカウント作成
2. Education planの作成
   ※ 「非公開リポジトリ」が無制限に作成できる
3. 研究用リポジトリの作成
   ※ 別のプロジェクトには別のリポジトリを利用：全て一つの場所で管理しない
4. コードを保存しているディレクトリを登録
5. ファイル編集 → `add` → `commit` → `push`
   ※ 見返してわかりやすい「commitメッセージ」をつける
6. GitHubのページを確認し，コードが変更されていることを確認

便利なGitコマンド

`git checkout .`

作業ツリーの変更点を破棄，ステージングエリアの状態を回復

`git stash`

現在の作業内容を退避し，ステージングエリアの状態を回復
対比した状態は任意に回復できる

`git commit --amend`

一旦ローカルリポジトリに登録したコミットを修正

`git reflog`

`git ***` のコマンドログを確認

リンク集

よみやすい順

- サルでもわかる Git 入門
  - 「まず Git の勉強をしたい」ときにおすすめします．豊富な（かわいい）絵で説明されているので，かなりとっつきやすいと思います．
- いつやるの？ Git 入門 v1.1.0
- Pro Git book（日本語版）
  - 有名な Git の書籍で，オンラインで読むことができます（PDF もダウンロードできます）．Linux でのコマンド操作をベースに解説されており，やや敷居が高いかもしれませんが，とてもよい資料と思います．
- こわくない Git

何か疑問・質問があれば slack の `#50_github` まで！