# GitHub 入門

産業技術大学院大学
中鉢欣秀

2016-08-14

## 概要

これは Git の初心者が，基礎的な Git コマンドの利用方法から，GitHub フローに基づく協同開発の方法までを学ぶ演習である．

事前に git コマンドが利用できる環境を用意しておくこと．また CUI 端末での shell による基本的な操作を知っているとスムーズに演習ができる．

第 1 章は Git 初心者（初めてさわる者）を対象に基礎を学ぶ．第 2 章は個人による GitHub の初歩的な使い方を取り扱う．第 3 章ではチームによる GitHub の使い方を知ろう．

## 1　Git 入門

### 1.1　Git の設定

#### 1.1.1　Git コマンドの実行確認

- 端末を操作して Git コマンドを起動してみよう．
- 次のとおり操作することで Git のバージョン番号が確認できる．

```
git --version
```

#### 1.1.2　名前とメールアドレスの登録

- 名前とメールアドレスを登録しておく
- 次のコマンドの $NAME と $EMAIL を各自の名前とメールアドレスに置き換えて実行せよ
  - 名前はローマ字で設定すること

```
git config --global user.name $NAME
git config --global user.email $EMAIL
```

#### 1.1.3　その他の設定と確認方法

- 次のとおり，設定を行っておく

```
git config --global color.ui auto
git config --global push.default simple
```

- ここまでの設定を確認するには

```
git config -l
```

### 1.2　Git のリポジトリ

#### 1.2.1　プロジェクト用のディレクトリ

- プロジェクト用のディレクトリ
- ソースコードなどのバージョン管理ができるようになる
- GitHub と連携させることで共同作業ができる

#### 1.2.2　Git リポジトリを利用するには

- リポジトリを利用する方法には主に 2 種類ある
  1. git init コマンドで初期化する方法
  2. git clone コマンドで GitHub から入手する方法
- 本章では 1. について解説する．次章からは 2. で行う．

#### 1.2.3　Git リポジトリの初期化方法

- my_project ディレクトリを作成し，Git リポジトリとして初期化するコマンドは次のとおり．

```
mkdir ~/my_project
cd ~/my_project
git init # Git リポジトリとして初期化する
```

#### 1.2.4 リポジトリの状態を確認する方法

- 現在のリポジトリの状態を確認するコマンドは次のとおり．

```
git status
```

- このコマンドは頻繁に使用する
- 何かうまく行かないことがあったら，このコマンドで状態を確認する癖をつけるとよい

### 1.3 ブランチの使い方

#### 1.3.1 ブランチとは

-

## 2 未整理

### 2.1 Git リポジトリ

#### 2.1.1 基本的な git コマンド

新しくブランチを作成してチェックアウトする

```
git checkout -b some_new_feature
```

ブランチを GitHub に push する

```
git add .
git commit -m '（作業内容）'
git push -u origin some_new_feature
```

### 2.2 GitHub とは

#### 2.2.1 **TODO** Git とは

#### 2.2.2 GitHub について

- ソーシャルコーディングのためのクラウド環境
  - GitHub
  - GitHub Japan
- GitHub が提供する主な機能
  - GitHub flow による協同開発
  - Pull requests
  - Issue / Wiki
  - コード解析

#### 2.2.3 GitHub Flow

- Git-flow
  - GitHub が登場する以前、Git-flow が提唱された
  - A successful Git branching model ż nvie.com
- GitHub flow
  - GitHub により、よりシンプルで強力なワークフローが可能に
  - GitHub Flow ‑Scott Chacon
  - GitHub Flow (Japanese translation)

#### 2.2.4 **TODO** [後ろへ] GitHub flow におけるコンフリクトについて

- マージのコンフリクト
  - GitHub に提出した Pull requests が自動的にマージできないこと
- 基本的な対処法
  - コンフリクトは、コードの同じ箇所を複数の人が別々に編集すると発生
  - 初心者は、演習の最初の方では「他人と同じファイルを編集しない」ことにして、操作になれる
  - 上達したら積極的にコンフリクトを起こしてみて、その解決方法を学ぶ
  - Pull requests でコンフリクトが発生し、自動的にマージできない状態になったら、その PR を送った人がコンフリクトを自分で解消する

#### 2.2.5 コラボレーターの追加

- GitHub のリポジトリをブラウザで開く．
- Settings -> Collaborators を選ぶ
- メンバーを招待する
- 招待されたメンバーには確認のメールが届くので，リンクをクリックする

#### 2.2.6 コラボレーターがソースコードを入手する方法

下記の「ychubachi」の部分を代表者のアカウント名にする．

```
git clone ychubachi/ychubachi_2016_gem
```

1. プルリクエストとマージ

- ブランチが GitHub に登録されたことを確認し，Pull request を作成する
- Pull request のレビューが済んだらマージする

2. ローカルの master を最新版にする
   - GitHub で行ったマージをローカルに反映させる

```
git checkout master
git pull
```

#### 2.2.7 GitHub でのコンフリクトの解消方法

1. 前提
   - new_feature ブランチで作業中であり、最新の更新は commit 済
2. 操作（一例）

```
git checkout master            # master をチェックアウト
git pull origin master         # 手元の master を最新版にする
git checkout new_feature       # 作業中のブランチに戻る
git merge master               # この後、コンフリクトを修正する
git push origin new_feature    # 作業中のブランチを再度 push
```

#### 2.2.8 Gem の作成から GitHub への登録まで

```
bundle gem ychubachi_2016_gem
cd ychubachi_2016_gem/
git commit -m 'Initial commit'
git create
git push -u origin master
```

## 3 演習

### 3.1 ペアで行う GitHub

#### 3.1.1 ペアで GitHub を使ってみよう

1. 隣同士でペアを組む
2. レポジトリを作成する（どちらか一方）
   - `bundle gem` でひな形を作る（初心者は Gem でなくても良い）
3. レポジトリの Collaborators に登録する
4. レポジトリに対して、次のことを行う
   - Pull requests を利用してみる
   - Issue を利用してみる
   - Wiki を利用してみる

#### 3.1.2 課題 1

1. Pull request & merge の作業を各自 5 回以上行う
   - ディスカッションやコードレビューもやってみる
2. Issue を 5 個以上登録する
   - Pull request による Issue の close なども試す
3. Wiki でページを作成する
   - ページを 5 つ程度作成して、リンクも貼る
4. 以上が終わったペアはグループでの演習に進む
   - 講師に申告すること

### 3.2 グループで行う GitHub

#### 3.2.1 課題：グループで GitHub (1)

1. ペアを 2 つ組み合わせて 4 人グループを作成する
   - 課題 1 が終わったペアから順番にグループ編成
2. 作りたい Gem について相談して仕様を決める
   - テーマはなんでも良い
     - Web API を利用したコマンドラインツールなど
   - ある程度の役割分担も決めておく
3. レポジトリを作成する（代表者 1 名）
   - コラボレーターを追加する
4. 今まで学んだ知識を活用して Gem を開発する

#### 3.2.2 課題：グループで GitHub (2)

1. グループメンバーで Gem を共同で作成する
2. GitHub Flow の実践
3. Travis CI によるテストの自動化
4. RubyGems.org への自動ディプロイ
5. その他、GitHub の各種機能の活用

## 4 Git 解説

### 4.1 解説

- git にはブランチ（branch）の概念がある
- 最初にあるのは master ブランチ
- master は一番大切なブランチであり，常に正常

に動作する状態にする

- 新しい作業を開始するときは必ず新しい branch を作る
- 後に，作業内容を master に取り込む（merge）

# 5 Git 演習

## 5.1 ブランチの作成

### 5.1.1 課題

「new_feature」ブランチを作成せよ

```
1 git checkout -b new_feature
```

### 5.1.2 確認

- 方法 1) git status の結果の一行目が「On brunch new_feature」になっていること
- 方法 2) git status の一行目が「On brunch new_feature」になっていること

# 6 GitHub 演習（個人）

## 6.1 アカウントの作成

### 6.1.1 課題

GitHub にアカウントを作成せよ

### 6.1.2 提出

TODO: Google form