

# Git入門

九州大学大学院システム情報科学府  
社会情報システム工学コース (QITO)

石田繁巳 細合晋太郎 亀井靖高 鵜林尚靖



# この発表で知って欲しいこと

- バージョン管理の必要性
- Gitというバージョン管理システム
- GitHubを使った開発の流れ

# アウトライン

## ■ バージョン管理とは

- 複数人での開発における問題
- バージョン管理システム
- GitとGitHub

## ■ Gitの使い方

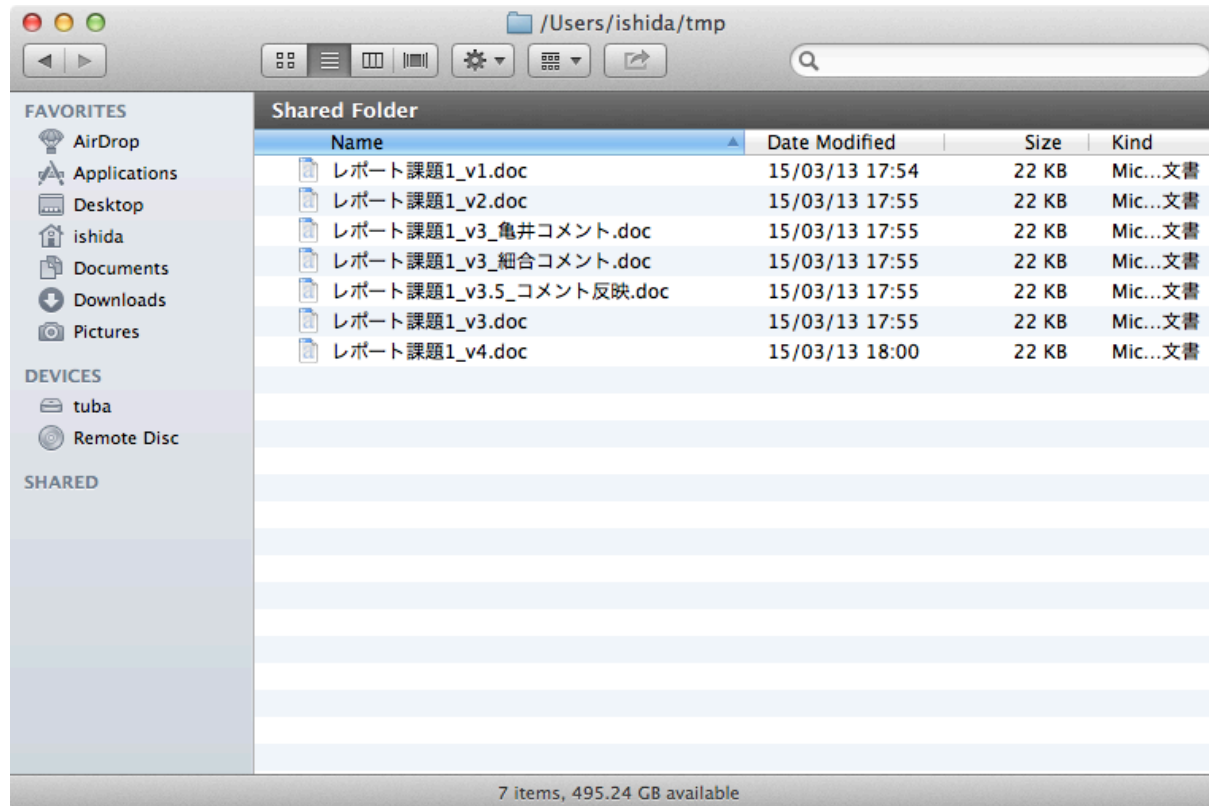
- 基本ルール
- 開発の流れ

# バージョン管理とは

---

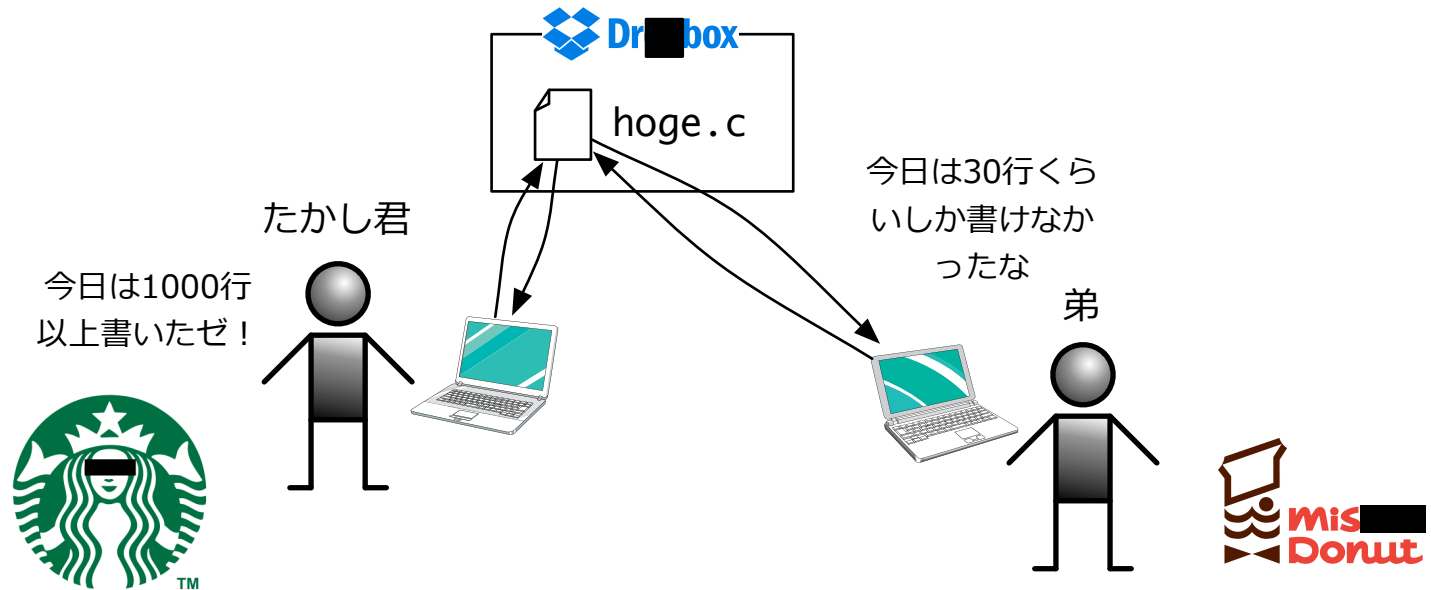
# バージョン管理がなかったら (1)

## ■ こんなことしてませんか？



# バージョン管理がなかったら (2)

- たかし君と弟は2人で開発している

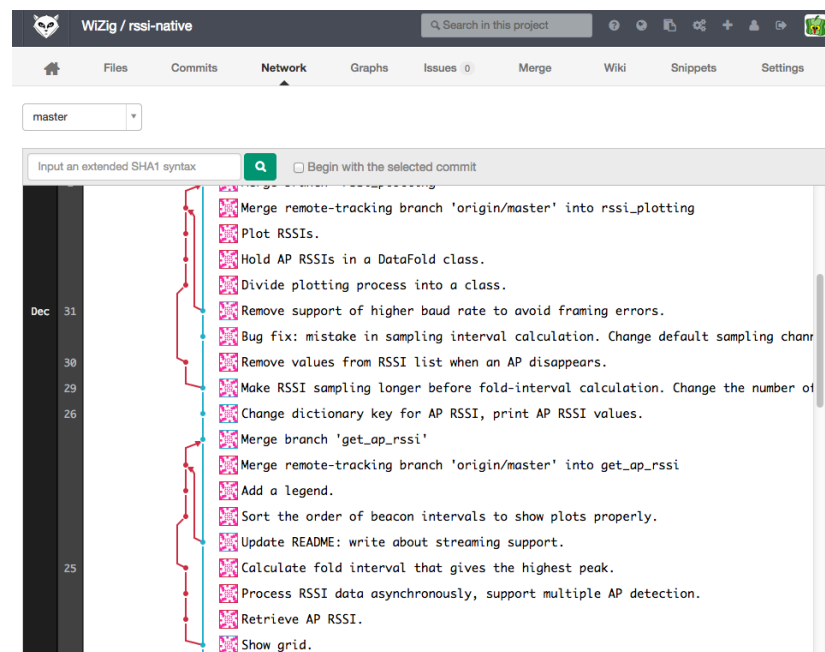


問. この後のたかし君の心情を述べよ

# バージョン管理システム

## ■ 「文書」をバージョン（版）毎に管理するシステム

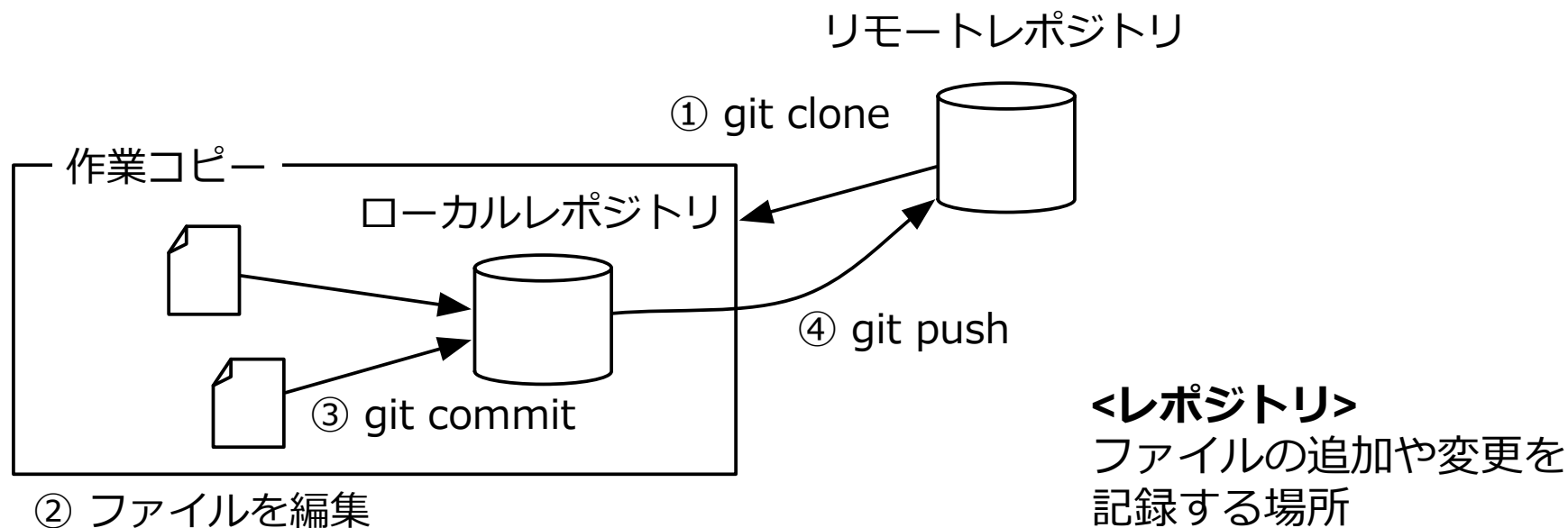
- 過去のバージョンの文書を取得する
- 差分（変更）を見る
- 同じソースファイルを複数人で開発し，統合（merge）する



# Git

## ■ Git (ギット, ジット)

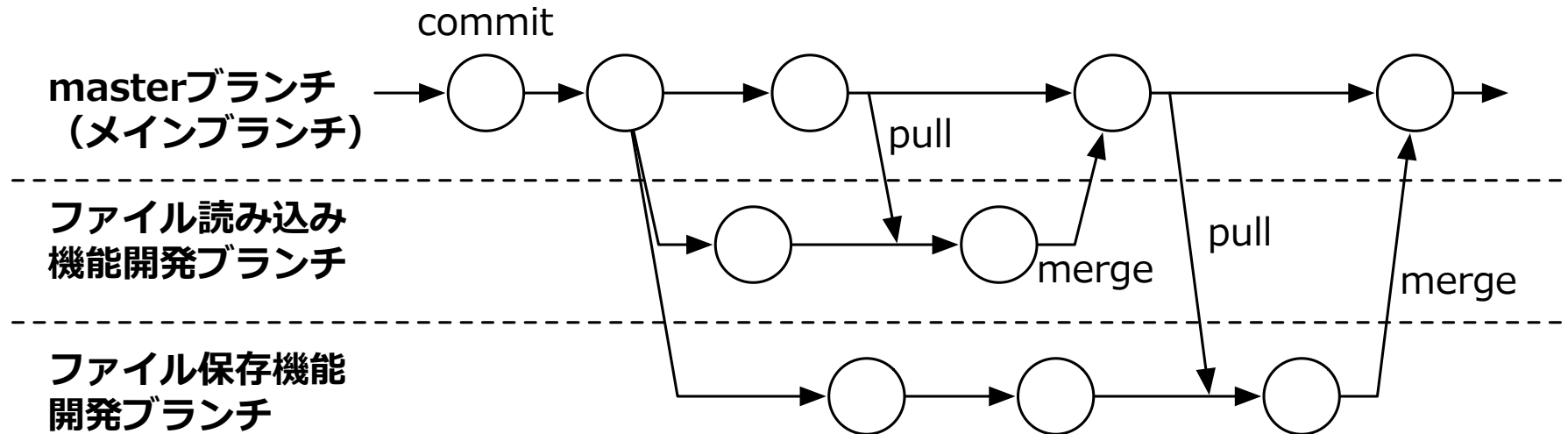
- 分散型バージョン管理システムの1つ
- 参考) [http://www.backlog.jp/git-guide/intro/intro1\\_1.html](http://www.backlog.jp/git-guide/intro/intro1_1.html)





# ブランチ

- commitの履歴を分岐して管理する仕組み
  - 複数人での同時開発をサポートする



# GitHub



- Gitを用いたオンラインソースコード共有サービス
  - <https://github.com>
- GitHubならではの拡張機能がある
- 公開レポジトリなら無料
- 非公開レポジトリは有料
  - ただし、学生や教育機関は申請すれば無料で使える
  - <https://education.github.com/>

# Gitの使い方

---

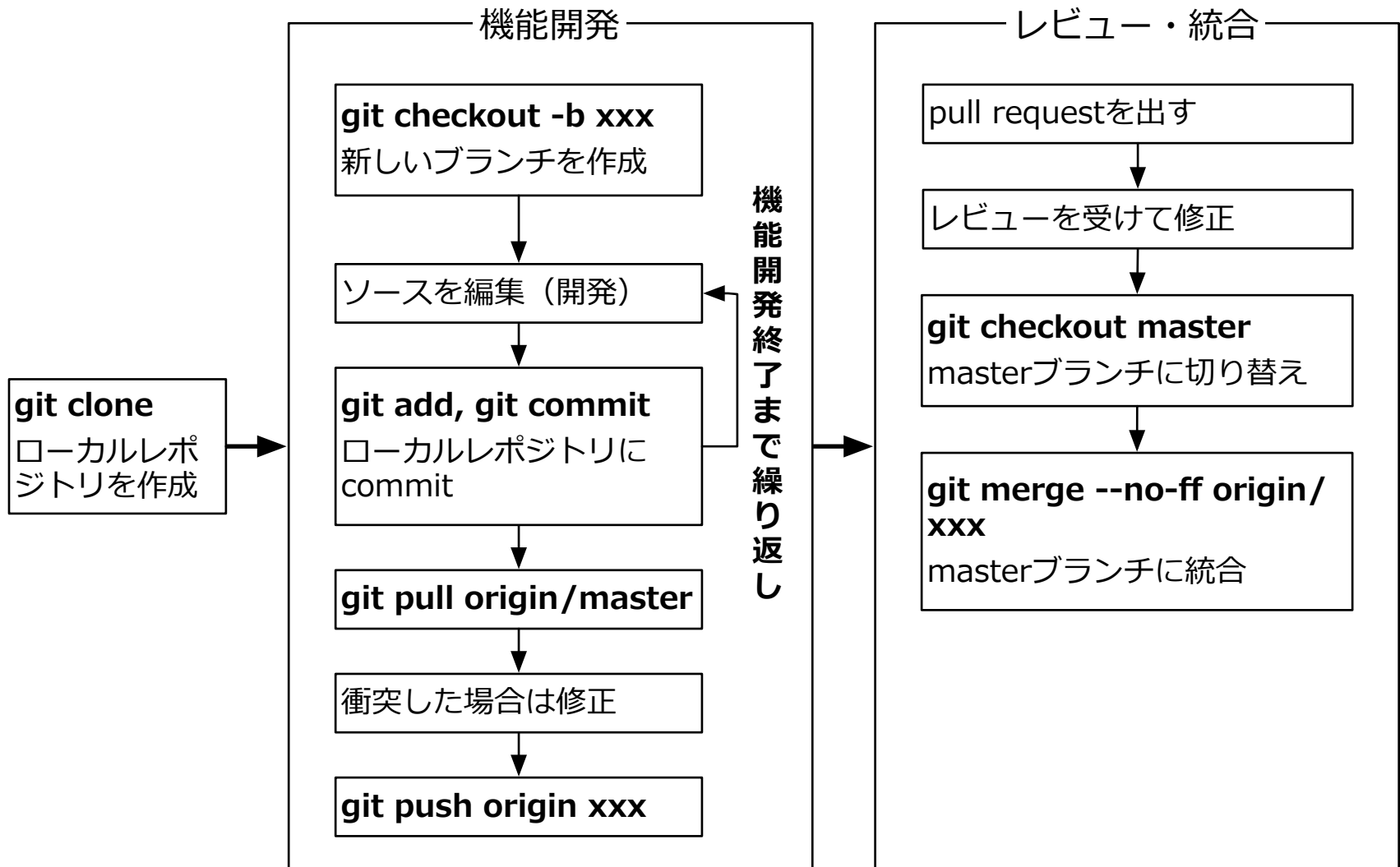
# 基本ルール

1. 自分たちが作成したファイルだけをcommitする
  - 例) hoge.cはcommit, hoge.oはcommitしない
2. ソースコードはコンパイルできる状態でcommitする
3. ログは必ず付ける
4. masterブランチは「動く状態」を維持する

# 各ユニットのレポジトリ

- ソースコード用とそれ以外の文書用と，各ユニット2つずつ
    - unit<sup>1</sup>-src: ソースコード用
    - unit<sup>1</sup>-doc: その他文書用
- ※ 赤字のユニット番号は適宜読み換えること

# 開発の流れ



# 開発ツール

## ■ SourceTreeがおすすめ

- <https://www.atlassian.com/ja/software/sourcetree>
- gitやsshも内部に持っている
- githubにも接続できる

## ■ GitHubで二段階認証している人はトークンの設定が必要

- <http://blog.pg1x.com/entry/2013/12/07/214932>

# 【参考】 Chatwork

- チーム内での連携はChatworkがオススメ
  - <http://www.chatwork.com/>
  - LINEみたいなWebサービス
  - 履歴が残るしファイルのやりとりもできる
  - 会話内で生まれてきた作業は「タスク」に落とし込むことで忘れずに実行できる
  - スマホアプリもある
    - ちょっと前のバージョンはクソだったけど・・・

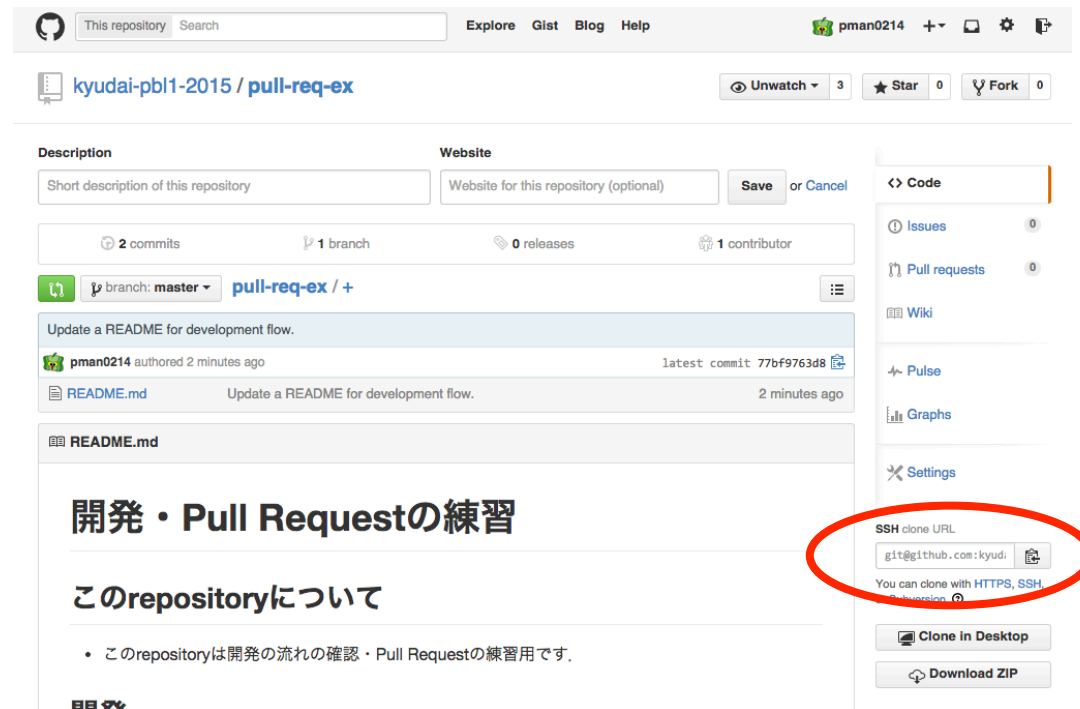


# GitHubを試しに使ってみよう

講義内ではやりません

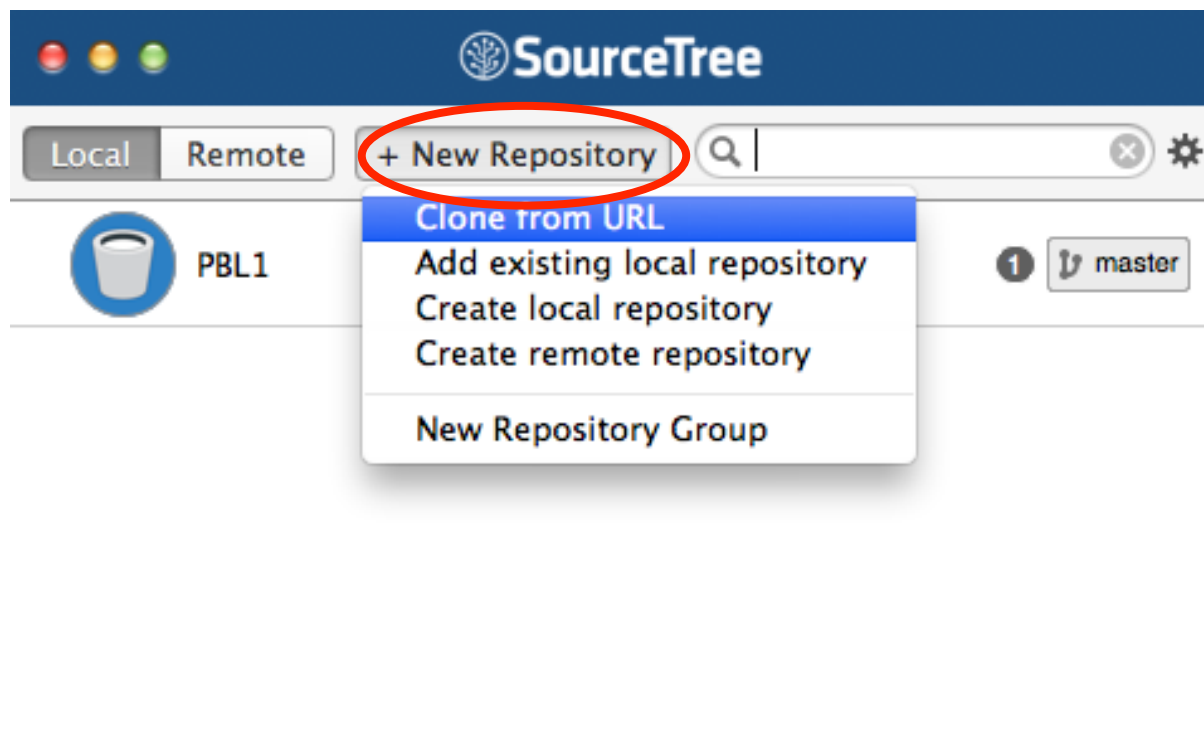
# Cloneする (1/3)

- 対象となるレポジトリのGitHubページを開く
  - <https://github.com/kyudai-pbl1-2015/pull-req-ex>
- clone URLをコピー
  - sshを使うべし



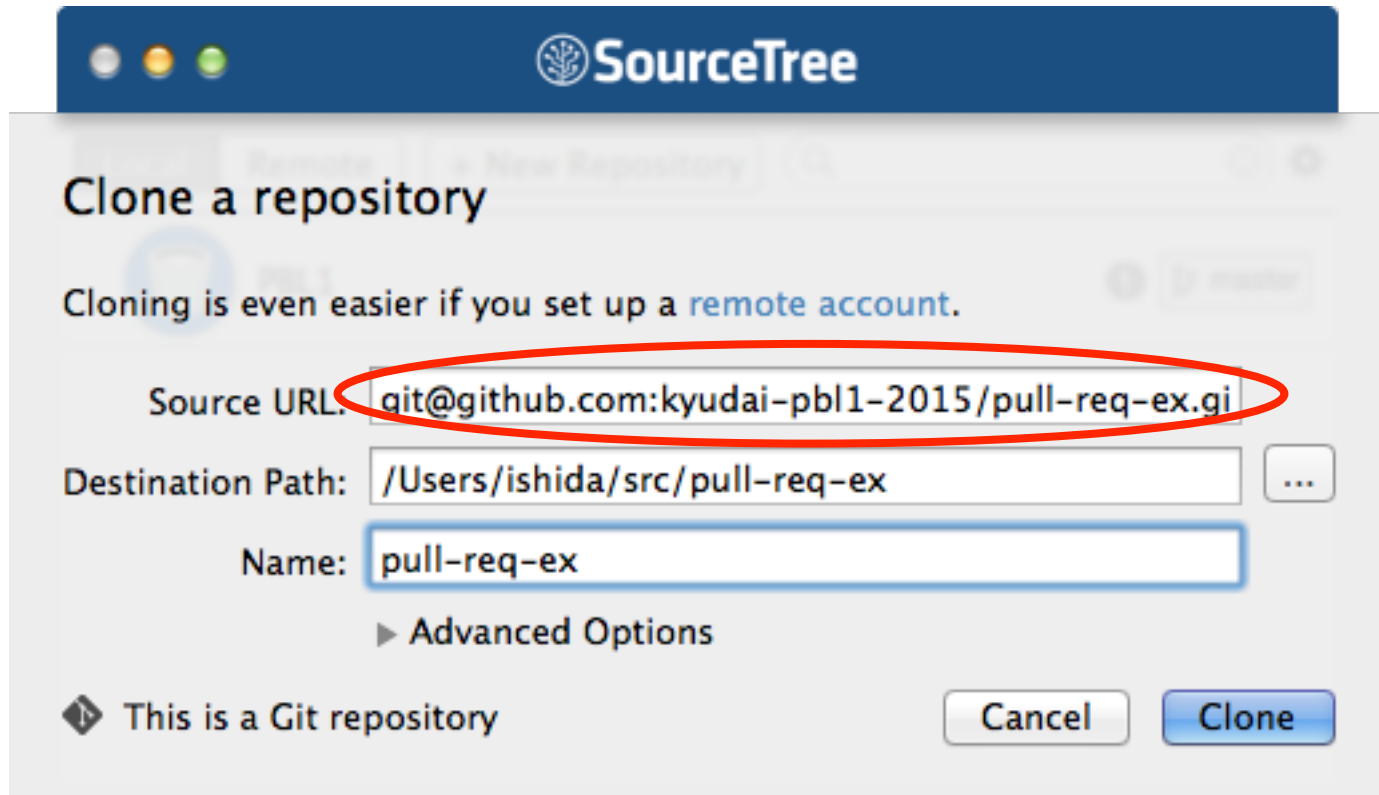
# Cloneする (2/3)

## ■ New Repository > Clone from URL



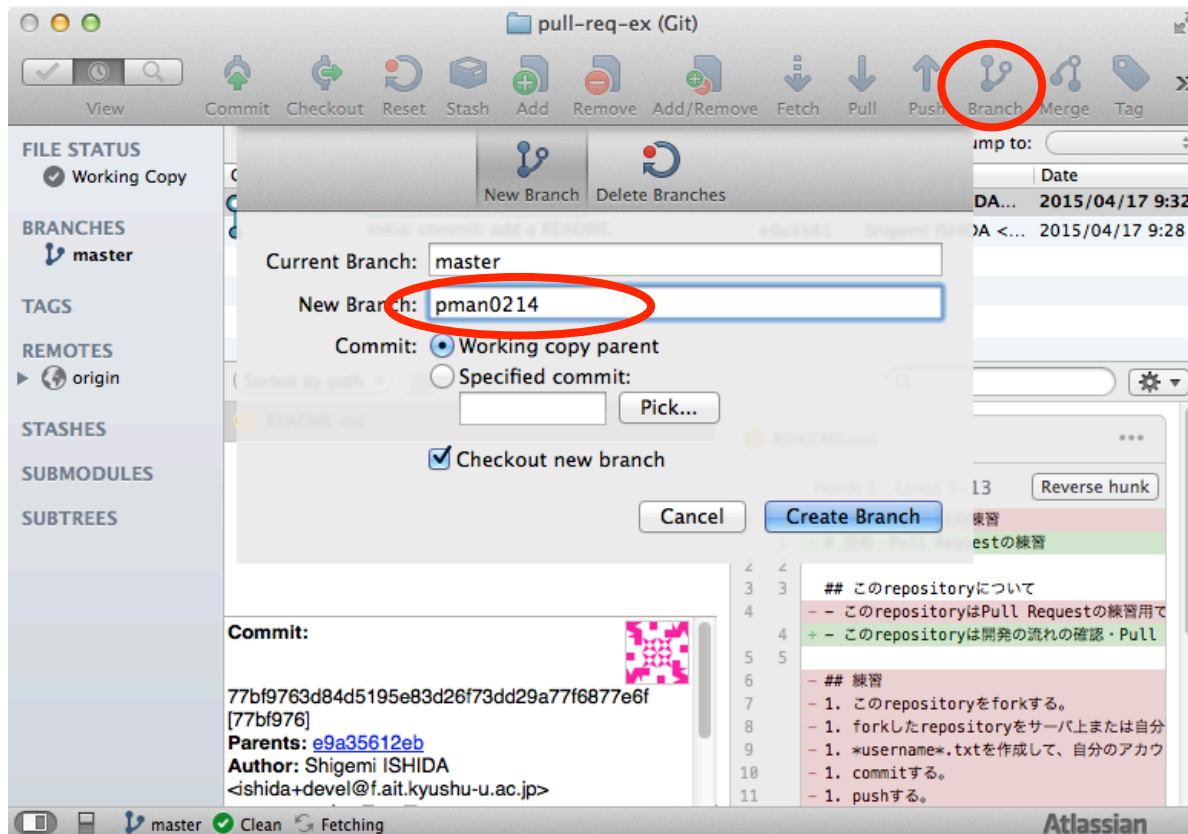
# Cloneする (3/3)

- 先ほどコピーしたURLを入力してClone



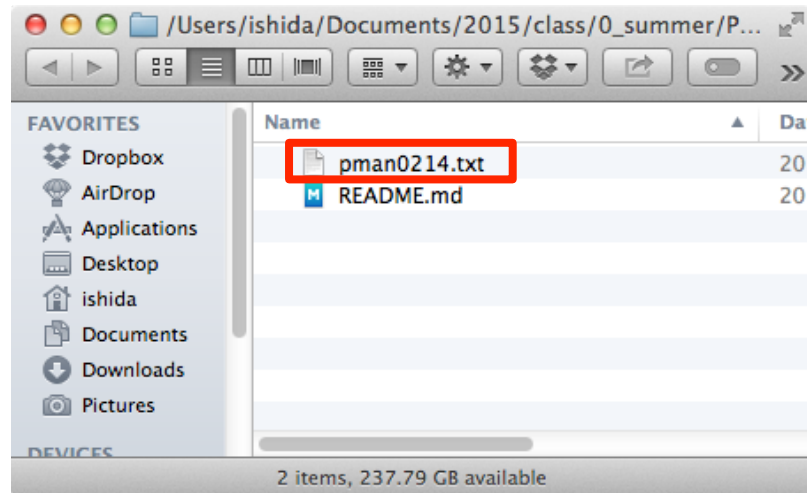
# 開発用ブランチを作成

- Branchをクリックし，自分のアカウント名のブランチを作成



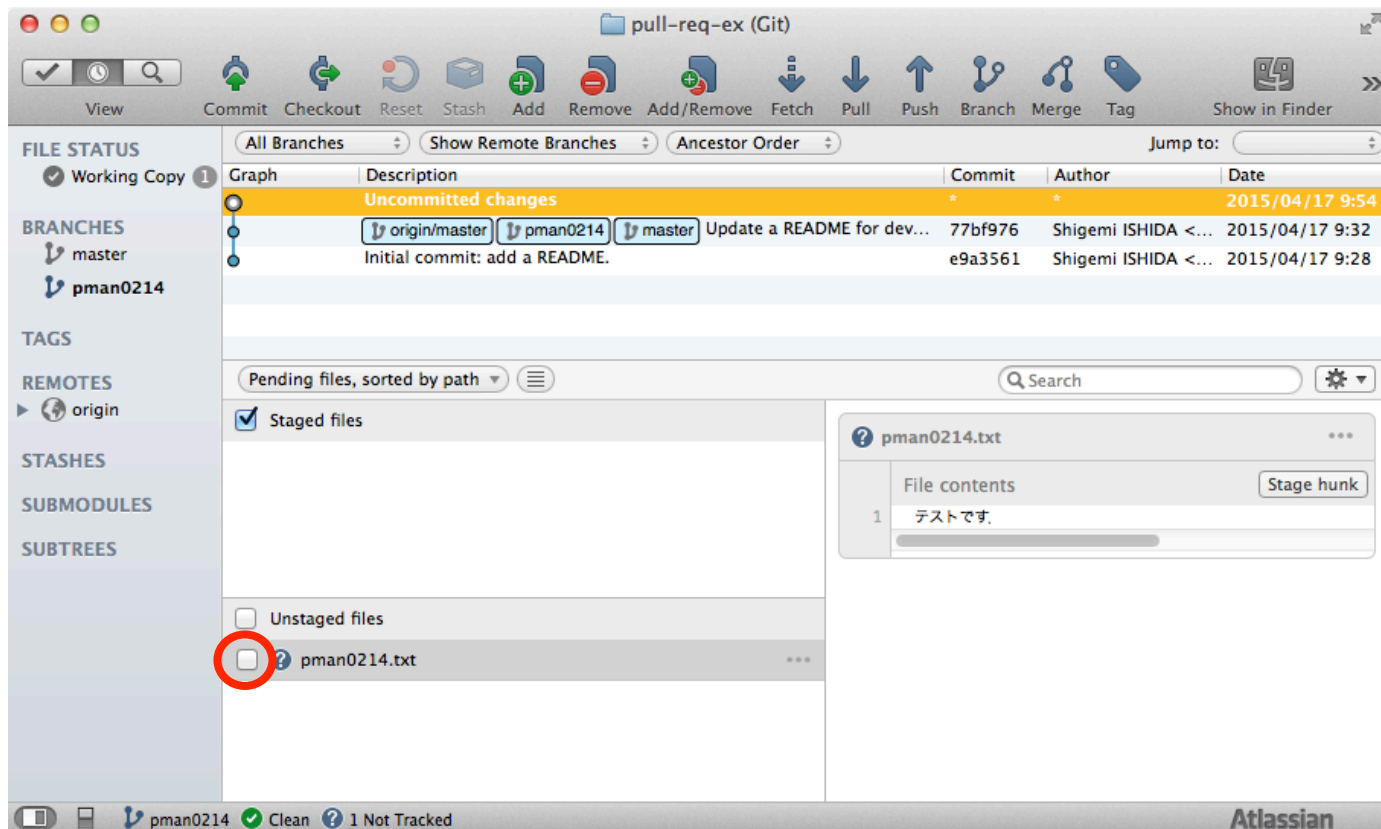
# 開発する

- Cloneした作業コピーでファイルを追加したり，編集したりする
  - 今回はアカウント名.txtを作成して，何かを書き込む
  - **このファイルは全世界に公開される**



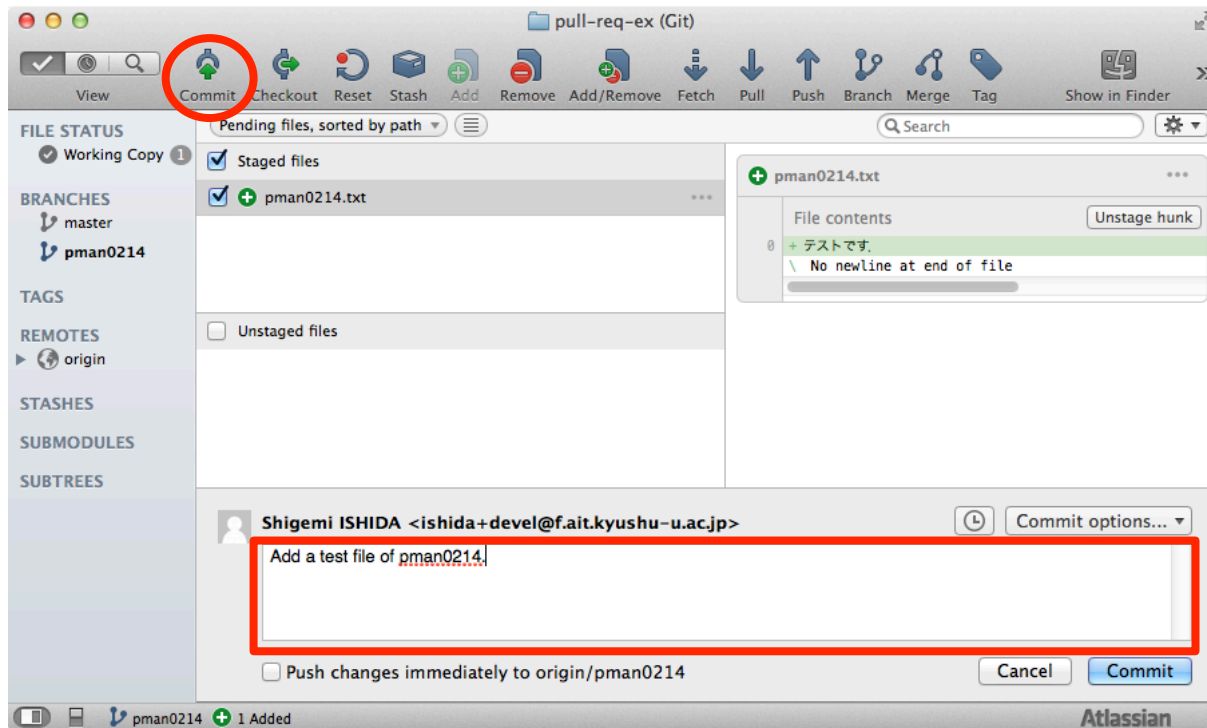
# Commitする (1/2)

- Unstaged filesの中から今回commitしたいファイルを選ぶ



# Commitする (2/2)

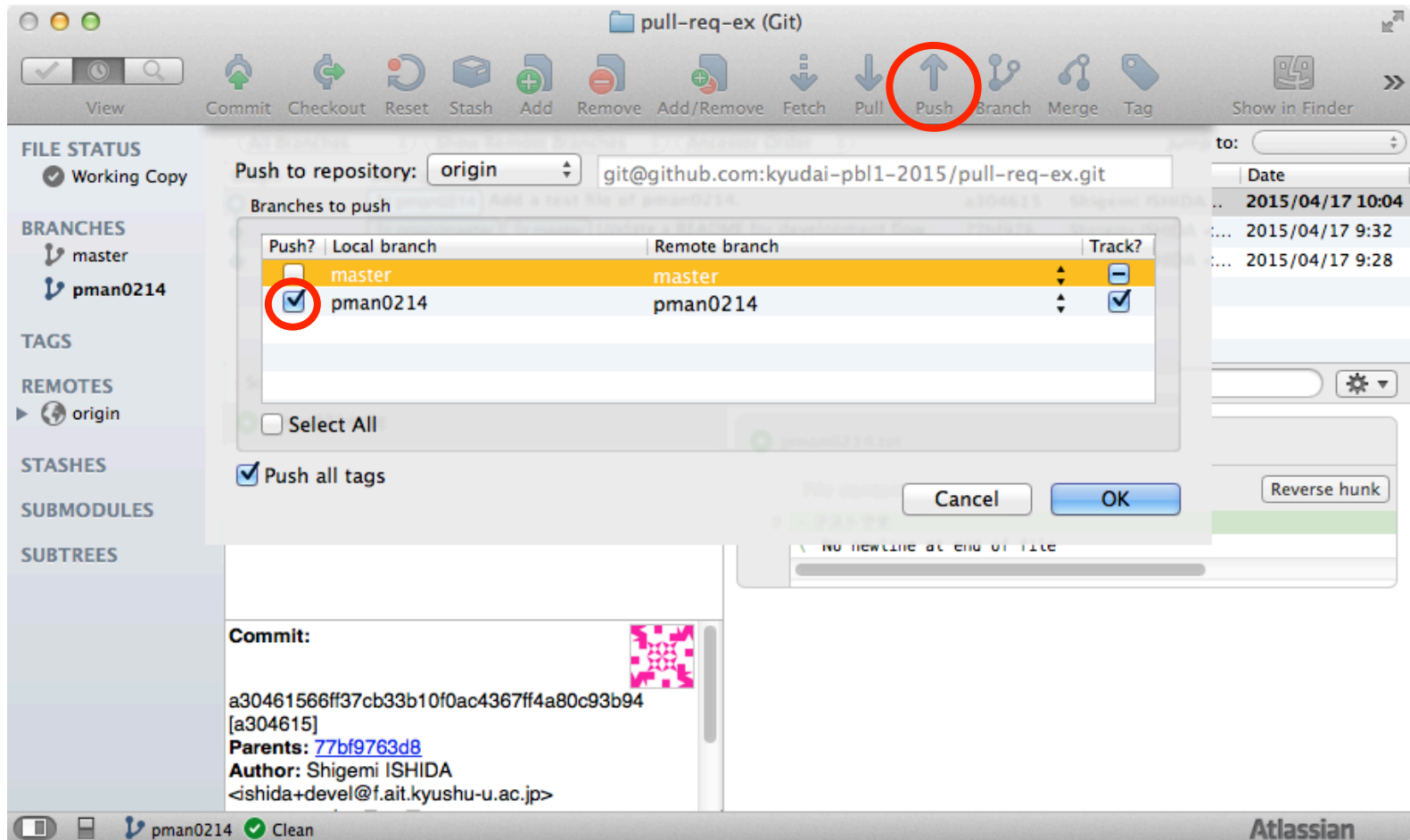
- 選択したファイルはStaged filesに入るのでCommitする
  - コメントを必ず付ける！





# Pushする


- 先ほど作ったブランチを選択してPushする



# Pull Requestする


- GitHubのページに行く
- Pull Requestを出す
  - コメントは開発した内容などを書く
  - チーム内でフォーマットを決めておくの良い

Your recently pushed branches:

 pman0214 (less than a minute ago)

 [Compare & pull request](#)



 branch: master ▾

[pull-req-ex](#) / +



Update a README for development flow.

# レビューしてMergeする


## ■ レビューする

- ・ 議論はコメント機能を使って記録を残す


## ■ 完了したらMerge（または取り下げ）


完了してMergeする  
ときはこれ

Add more commits by pushing to the `pman0214` branch on `kyudai-pbl1-2015/pull-req-ex`.



**This pull request can be automatically merged.**  
You can also merge branches on the [command line](#).

 **Merge pull request**



Write Preview Markdown supported Edit in fullscreen

Leave a comment

ここにコメントを書ける

Attach images by dragging & dropping, [selecting them](#), or pasting from the clipboard.

Close pull request Comment

修正が必要など、取り下げる場合はこれ