Git及びGitHub勉強会

濵口皓樹

soiwebでは

DropBoxで管理

- 容量に制限
- 現状ごちゃごちゃしていて作り直したい

現状

これから

GitHubで管理

- 公開されているので就職時にアピール
- バージョン管理(変更履歴)
- README.mdで詳細が書ける
- 誰が仕事したかわかる

Agenda

1.Gitについて

2.GitHubについて

3.使ってみる

その前に…

Q. 古いコードの復元するには？

以前は
「フォルダのコピーを取っておく」
「Ctrl+Zを押しまくる」
「覚えておく（脳筋）」

Gitを使うと簡単

Gitでできること

- ファイルの変更履歴が管理できる
- いつでも過去のファイルに戻せる
- なんでも入る
- バックアップにもなる
- チームでファイルを共有できる

Gitとは

分散型バージョン管理システム

「誰が」「いつ」「どのような変更をしたか」
を管理

複数人で開発する上で必須

時代は

集中型(Subversion[SVN]) から **分散型**(Git) へ

Git？GitHub？

GitHubとは？

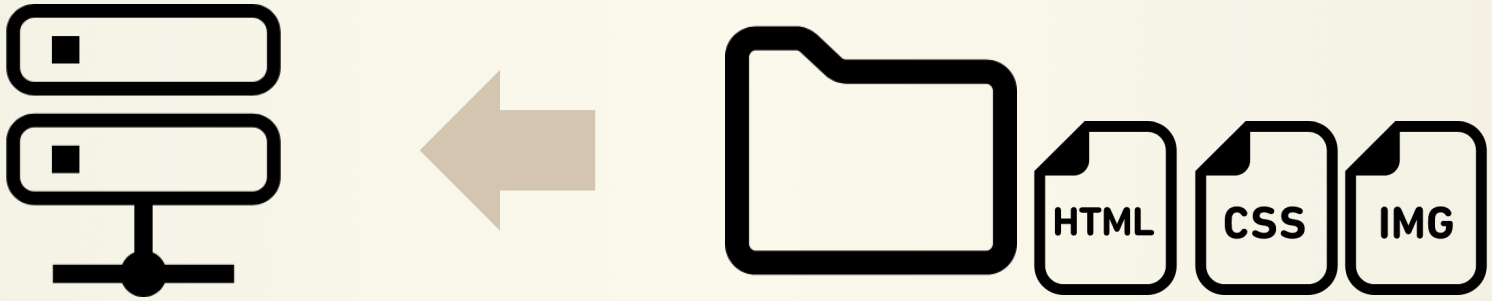
- 「Git」の「ハブ：拠点・中心・集まり」
- Gitの仕組みを利用して、世界中の人々が
  自分の作品を保存、公開できるウェブサービス

Gitを便利に使うためのツール

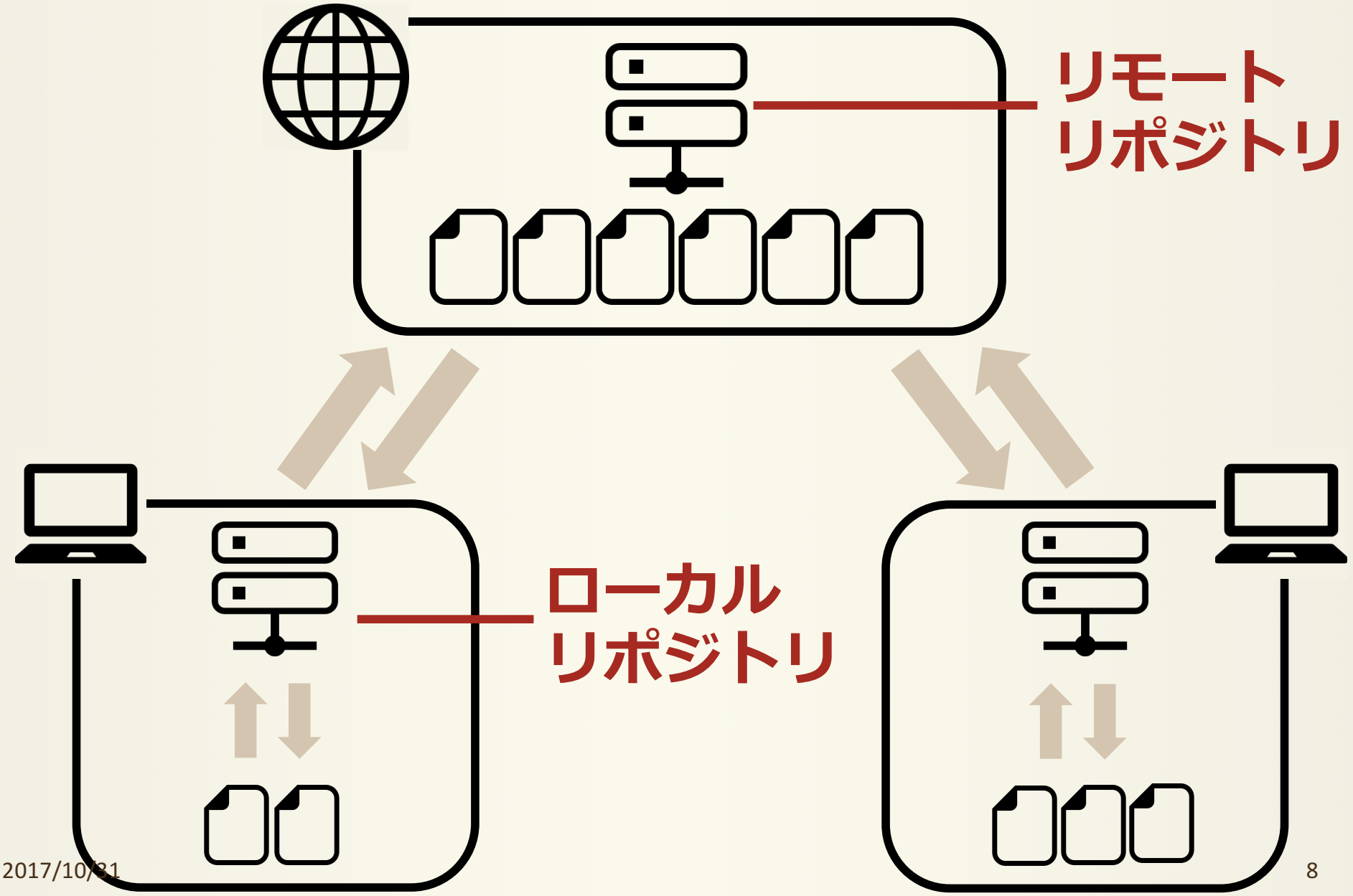
Gitとは

リポジトリ

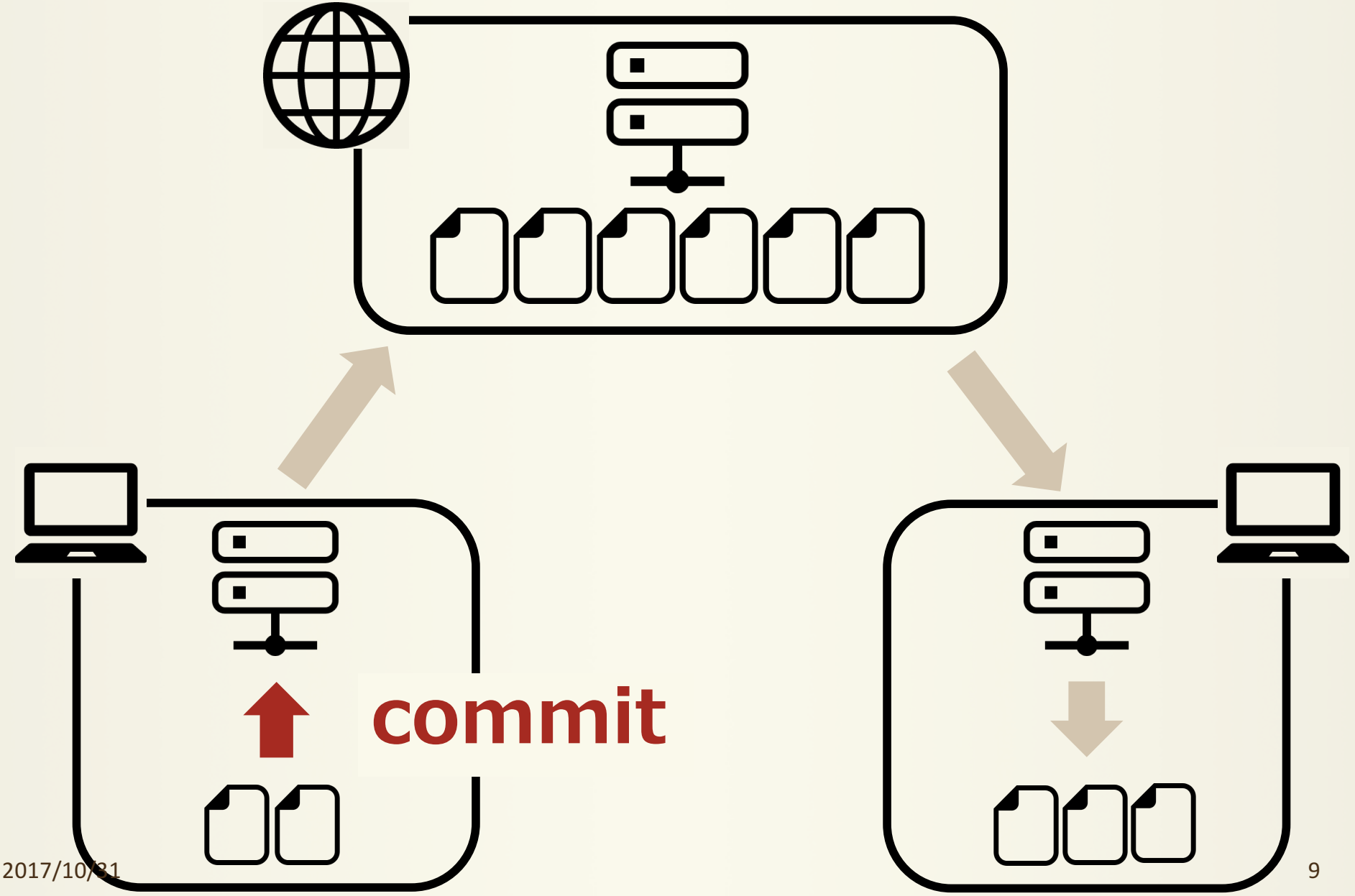
ファイルやディレクトリの状態を保存する場所



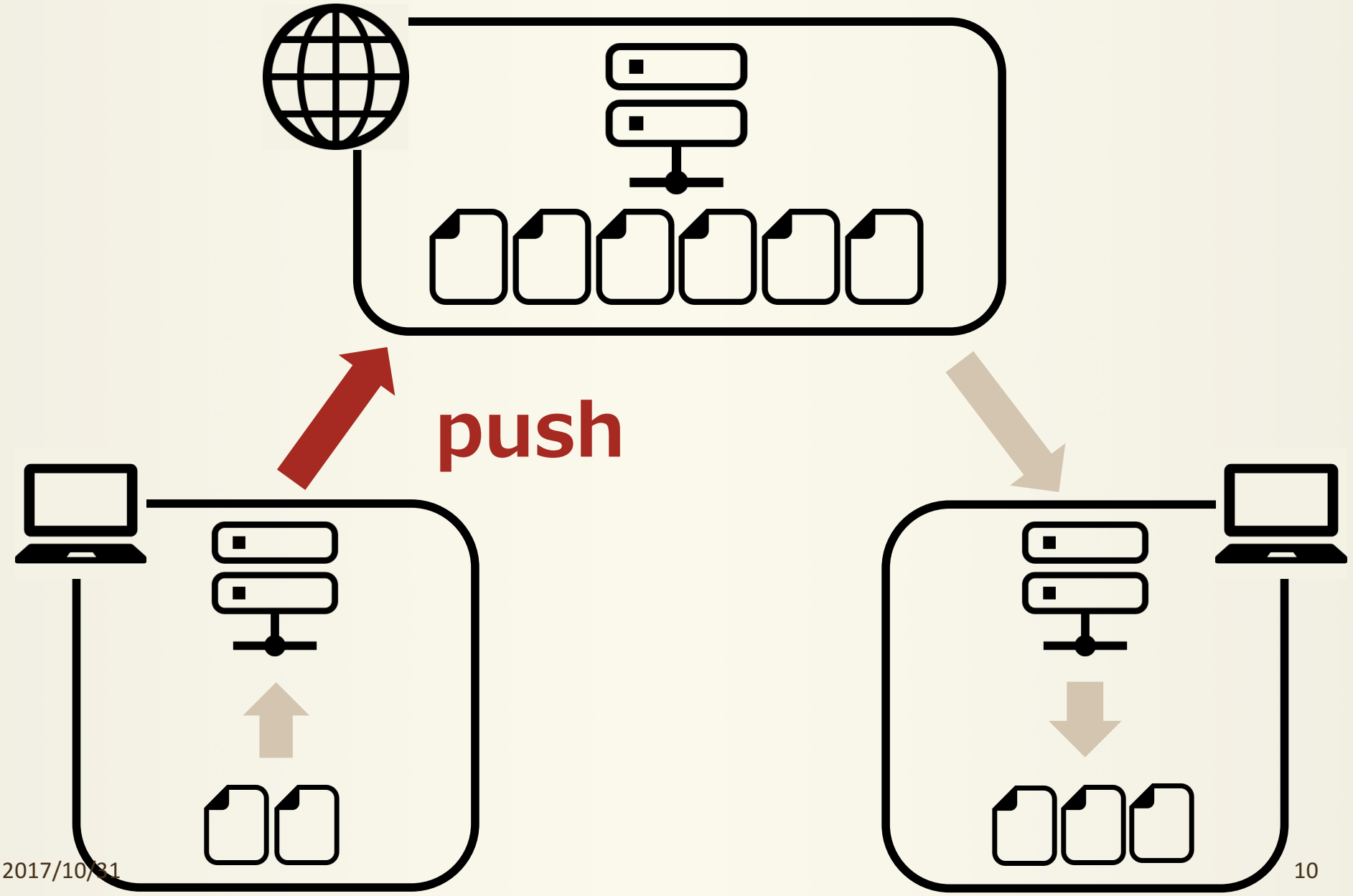
Gitの全体構成



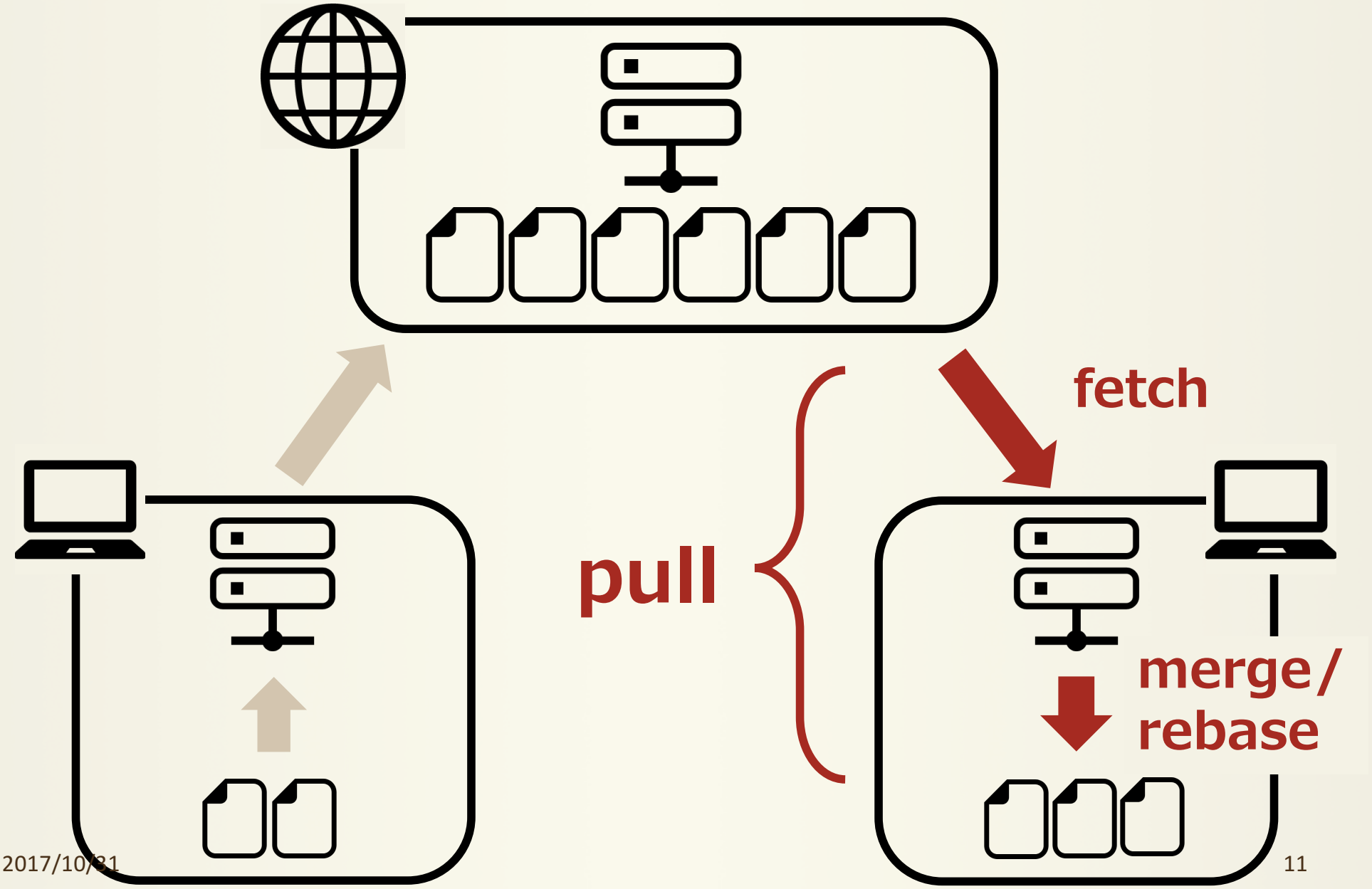
Gitの全体構成



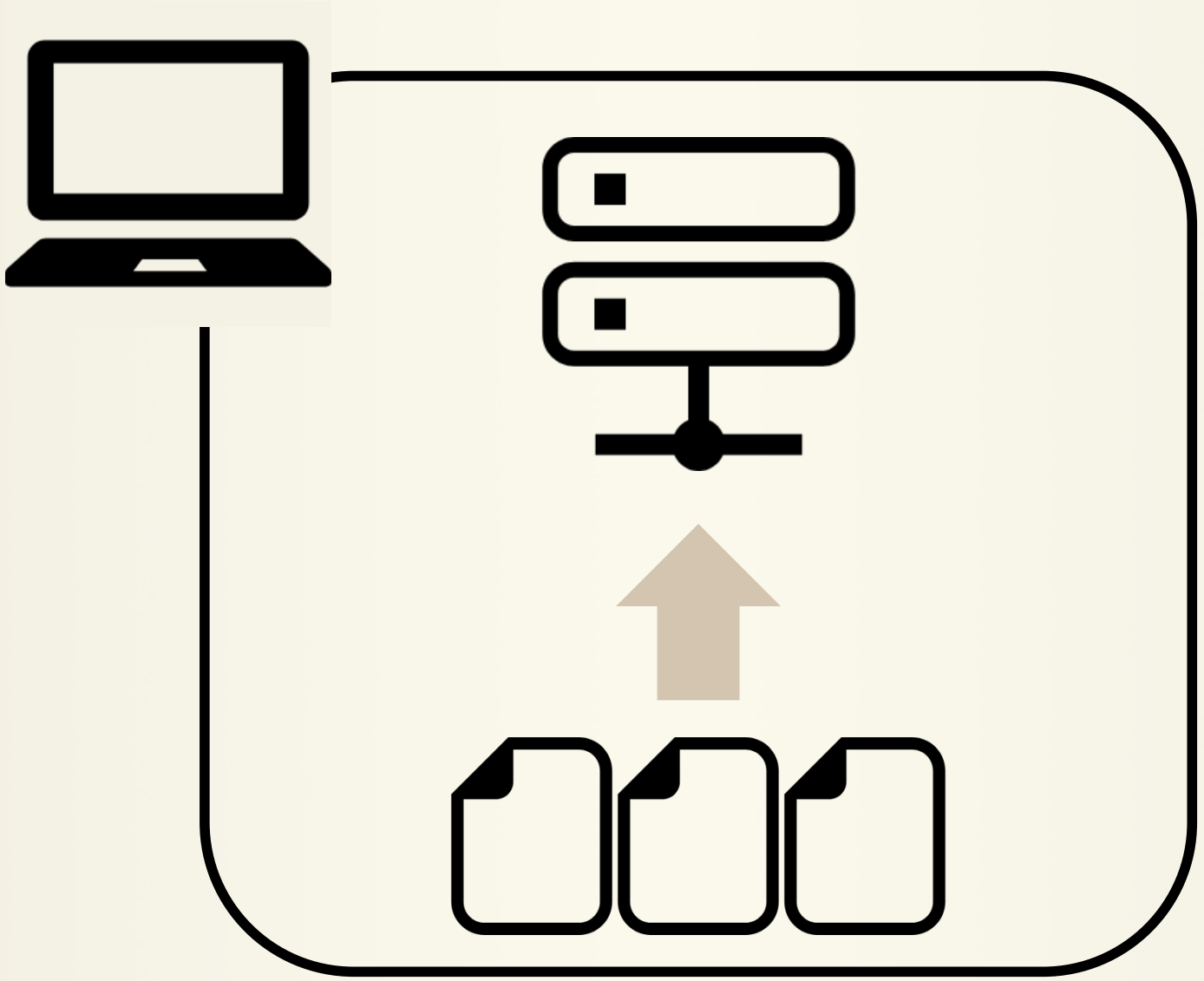
Gitの全体構成



Gitの全体構成



ローカル環境の内部構成



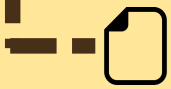
ローカル環境の内部構成

例

git_root

hello.txt

parent



child.txt

.git

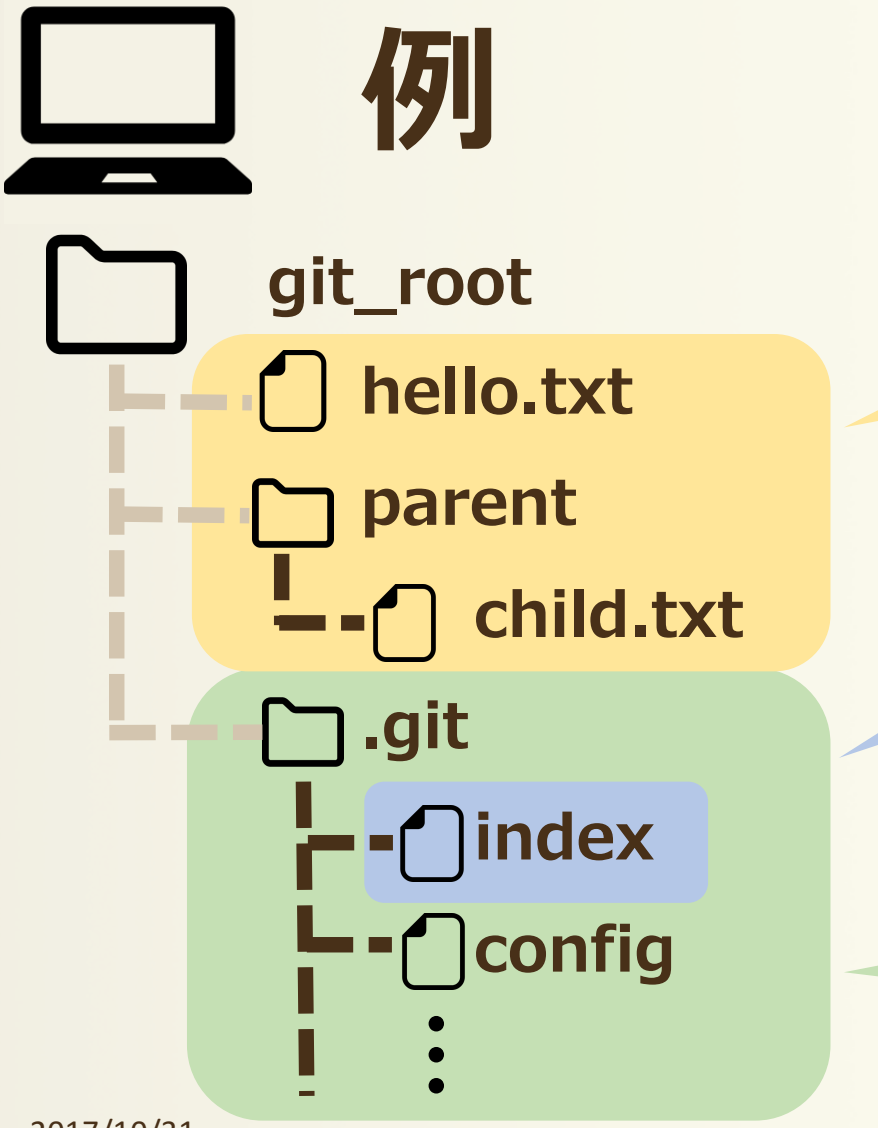
作業ディレクトリ
（ワークツリー）

実際に作業中の
ディレクトリ

Gitディレクトリ

Gitが管理している
ディレクトリ

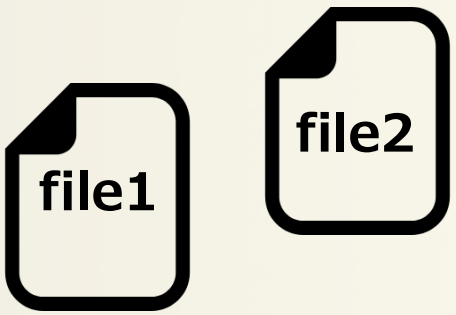
ローカル環境の内部構成



基本の流れ

Gitでの管理開始前

作業
ディレクトリ

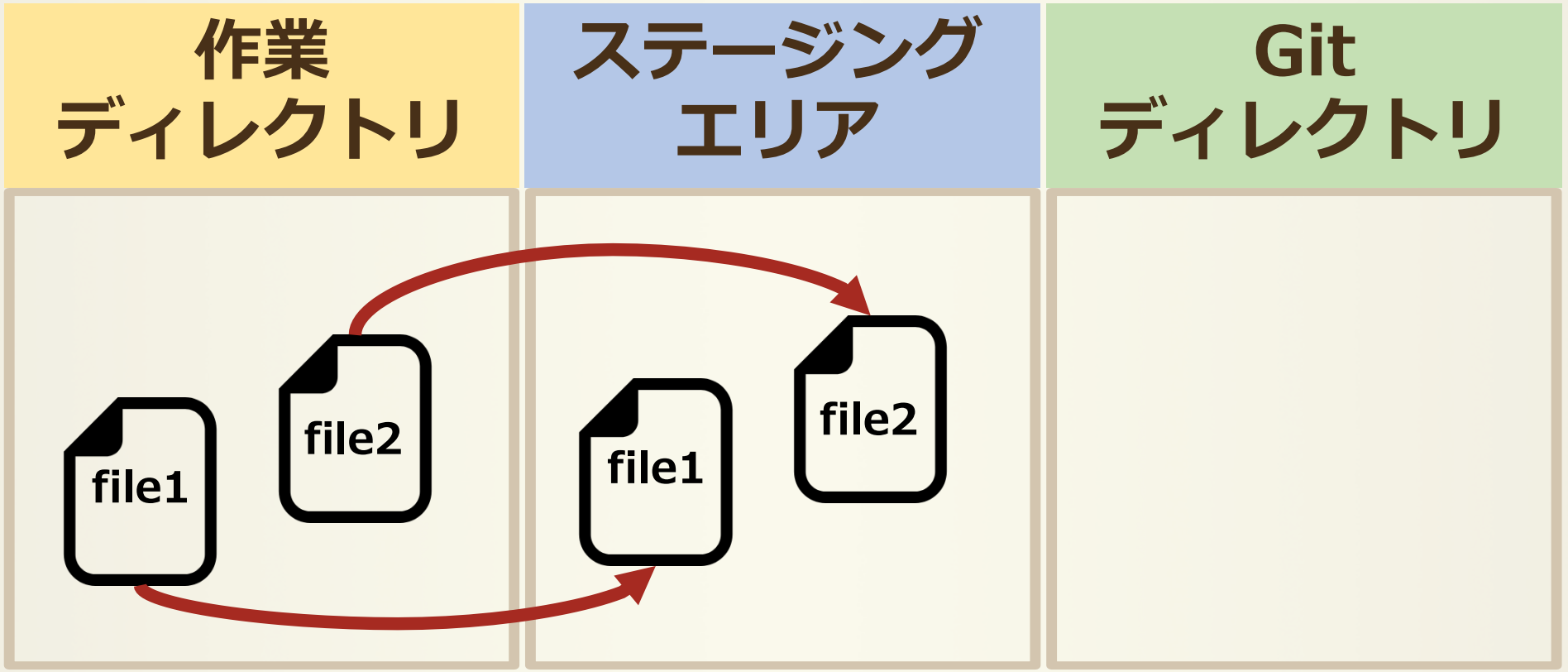


基本の流れ

$ git init

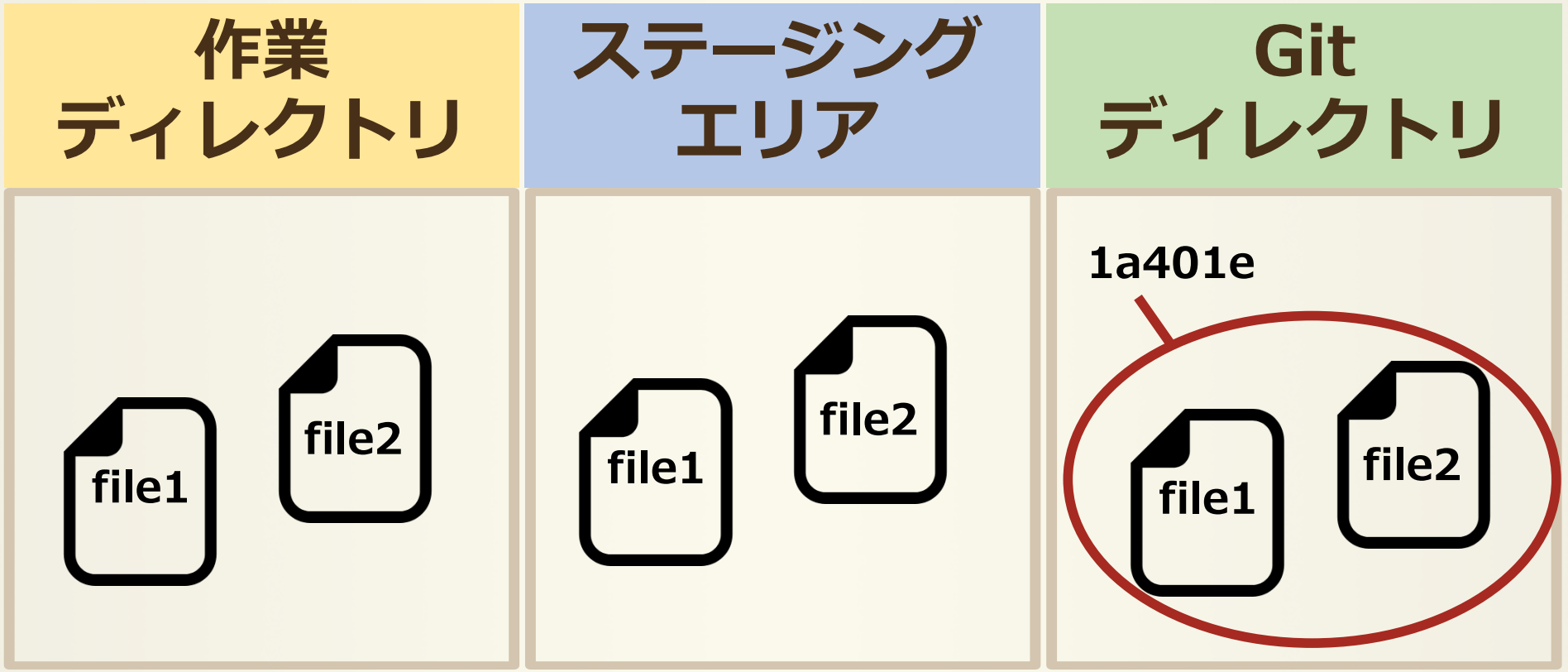
基本の流れ

$ git add file1 file2 ファイル名



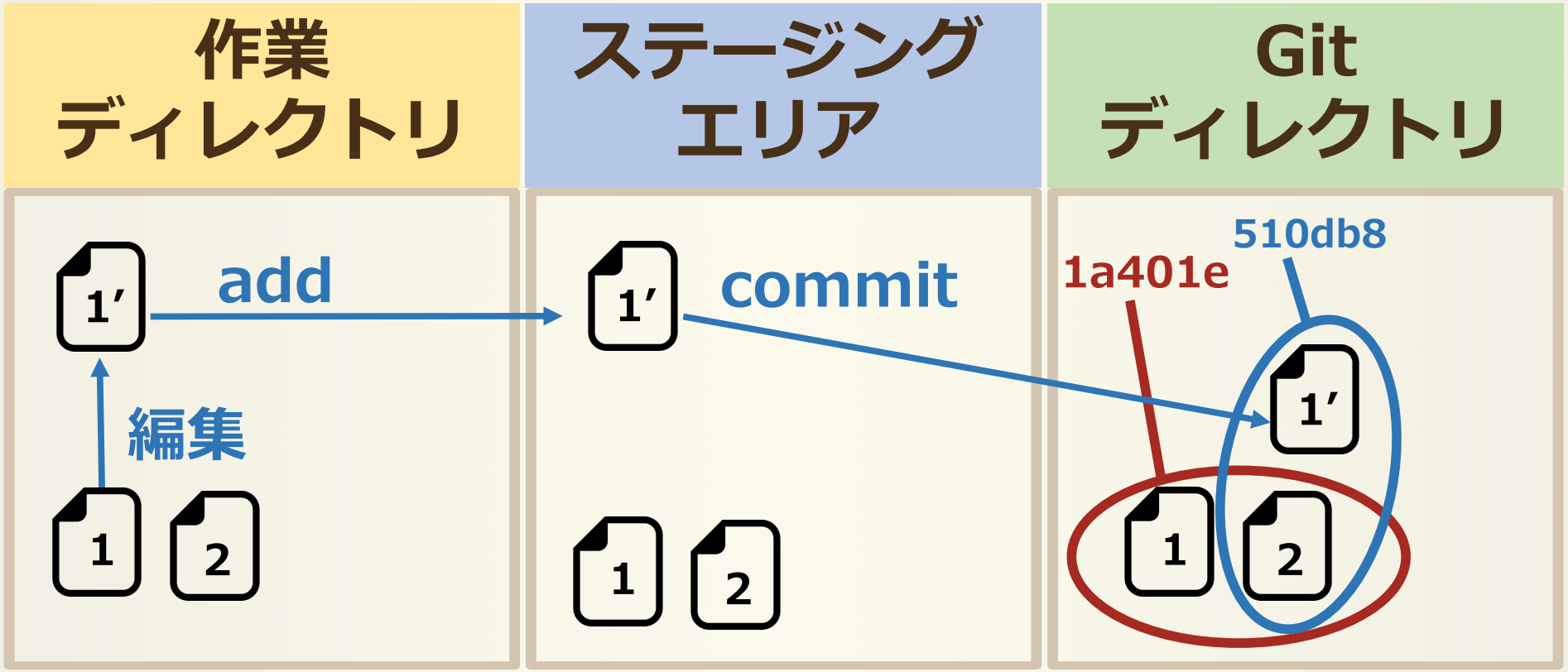
基本の流れ

$ git commit –m ‘アレ更新した’



基本の流れ

編集しても...

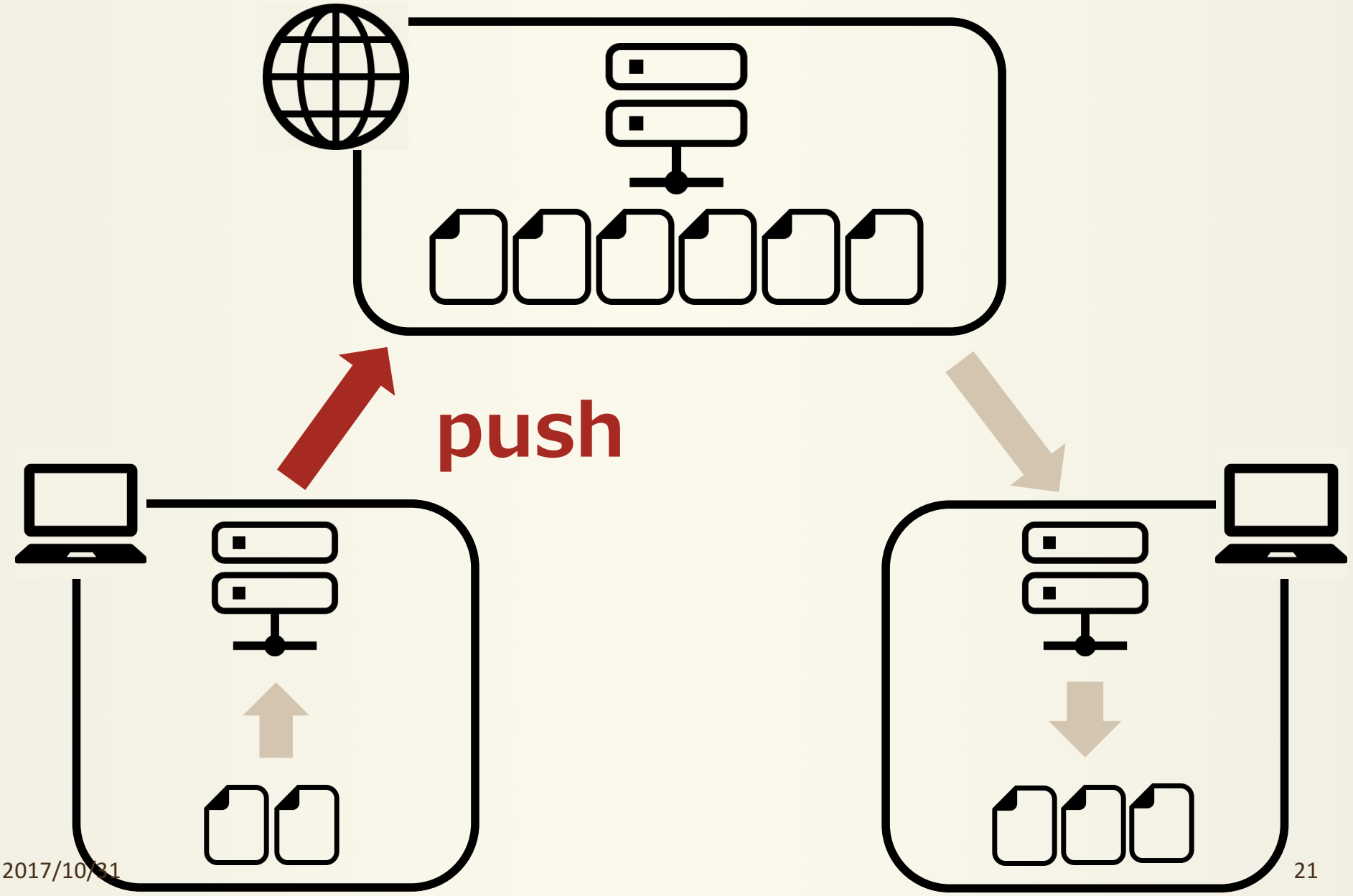

作業
ディレクトリ
ステージング
エリア
Git
ディレクトリ
1′
add
編集
1
2
1′
commit
1
2
510db8
1a401e
1′
1
2

基本の流れ(未)

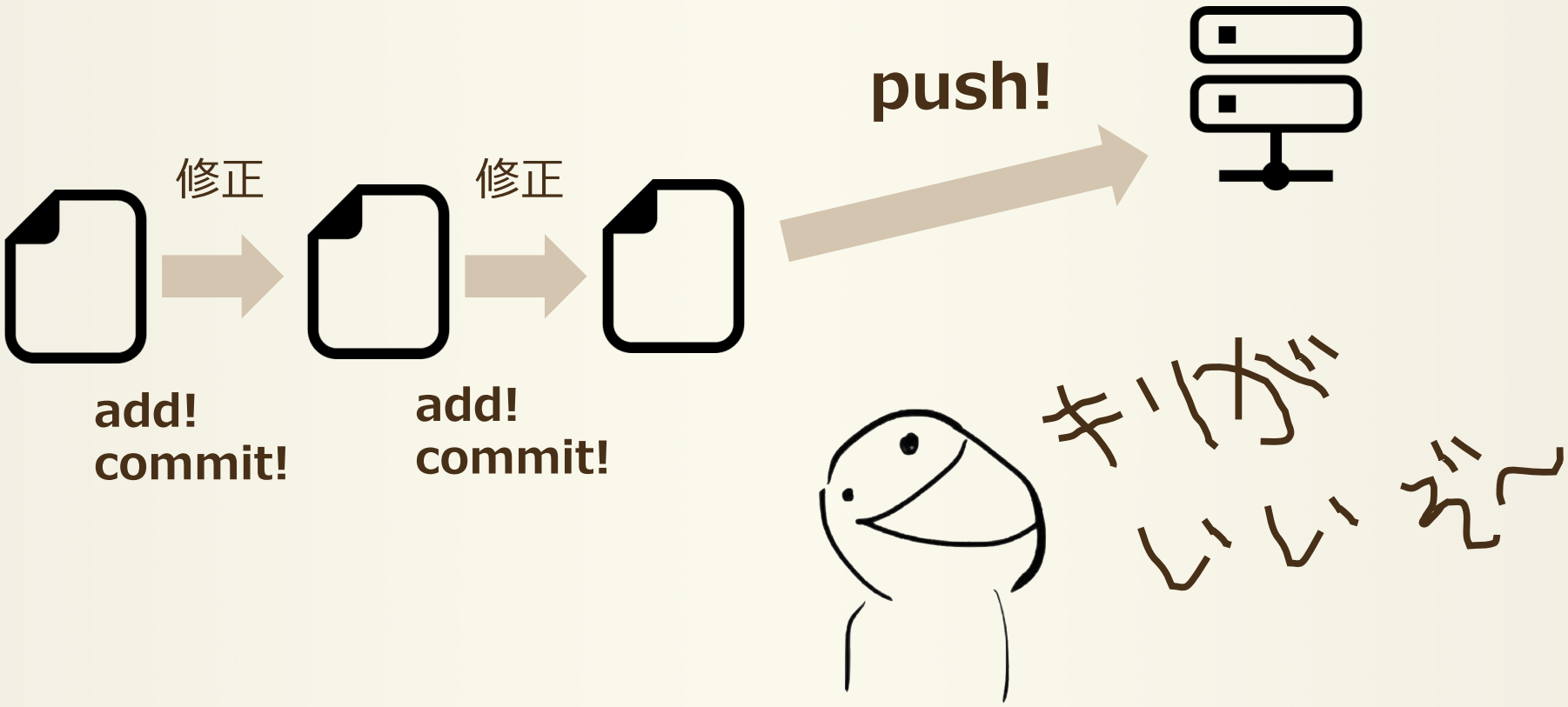
削除の場合…

消したファイルも
コミット必要

基本の流れ



基本の流れ

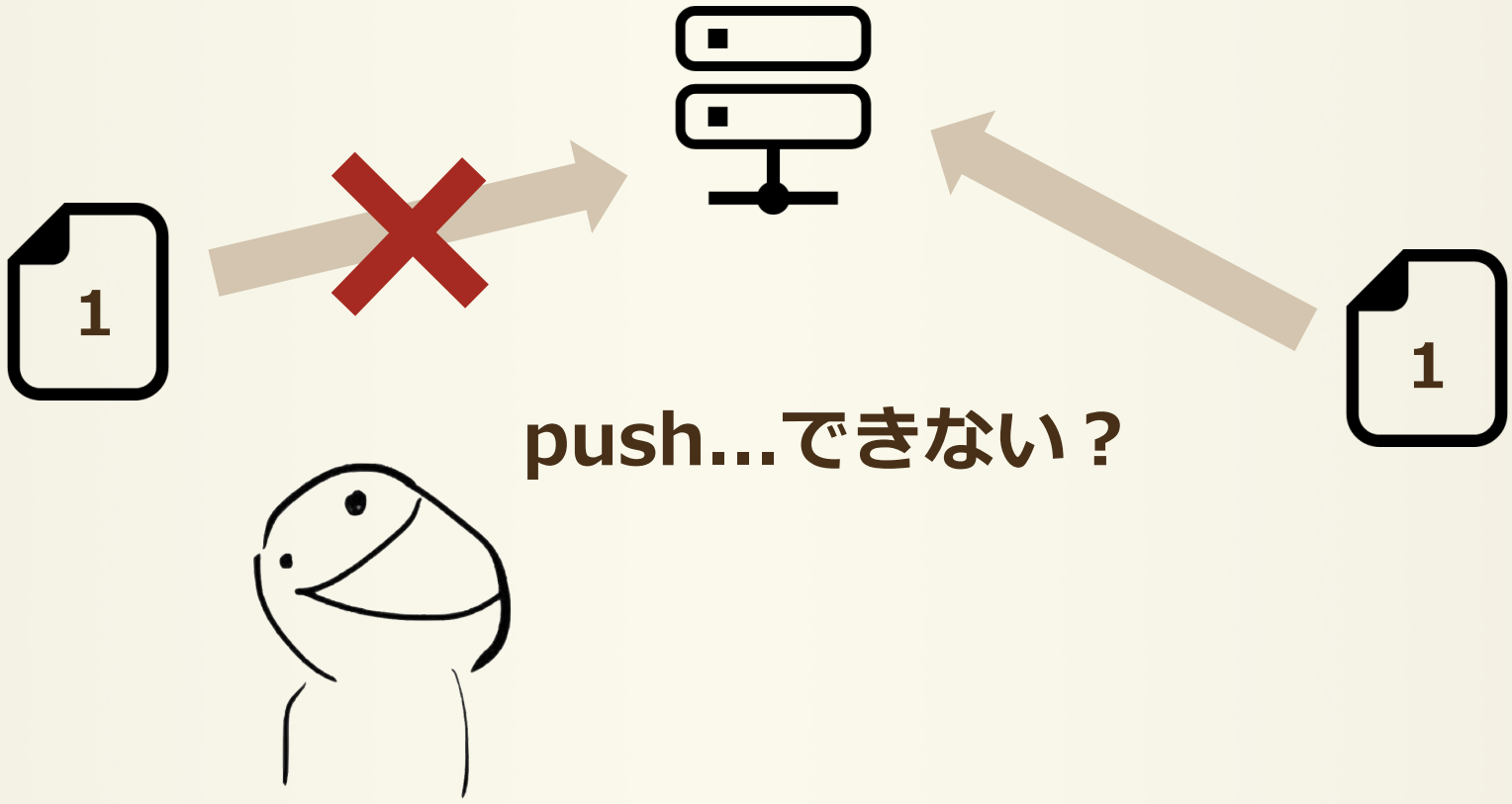


衝突

同じ箇所を別々のブランチで変更

どっちの変更を適用すればいいかわからない！

基本の流れ



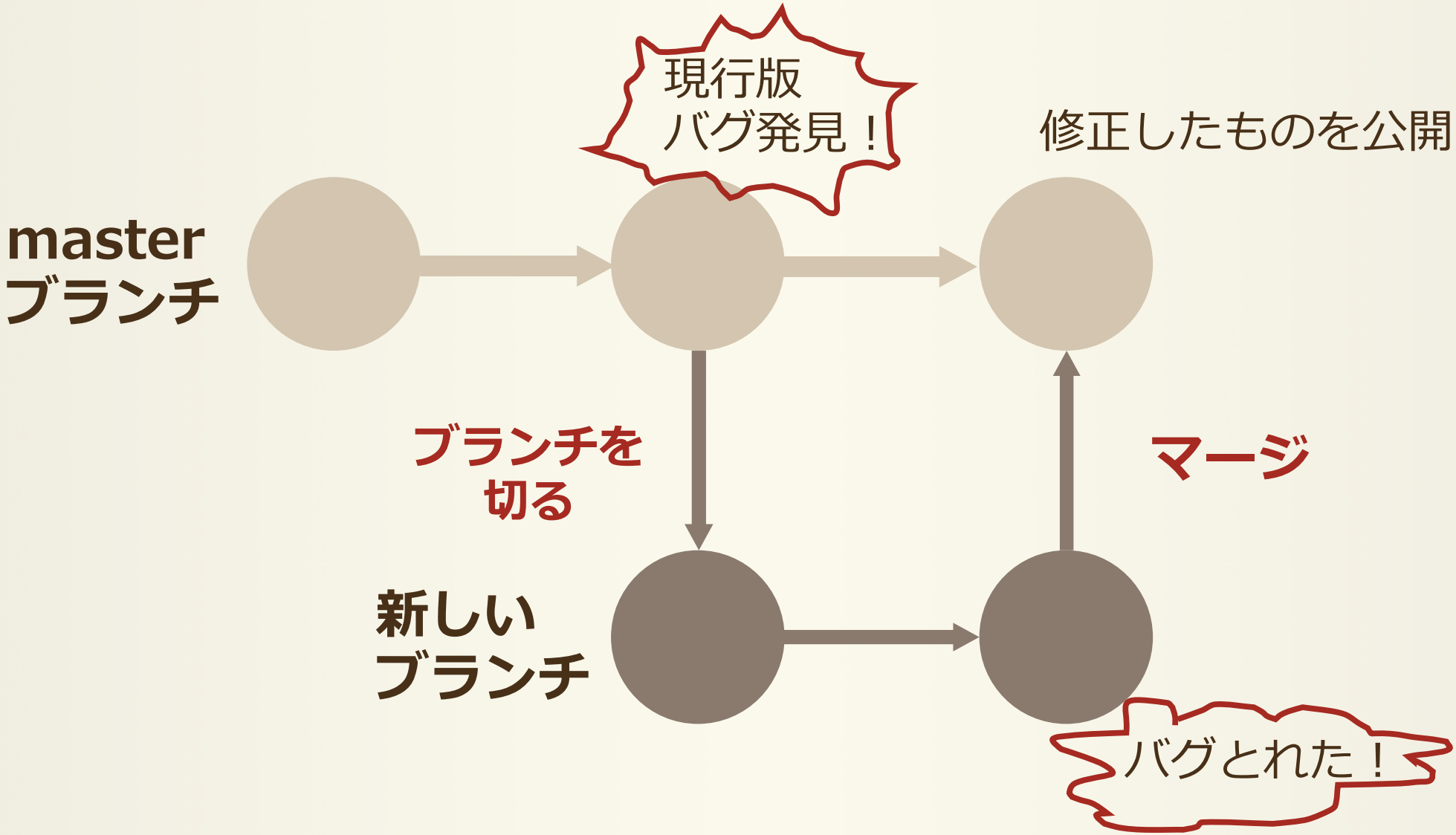
ブランチ

どんな時に使う？

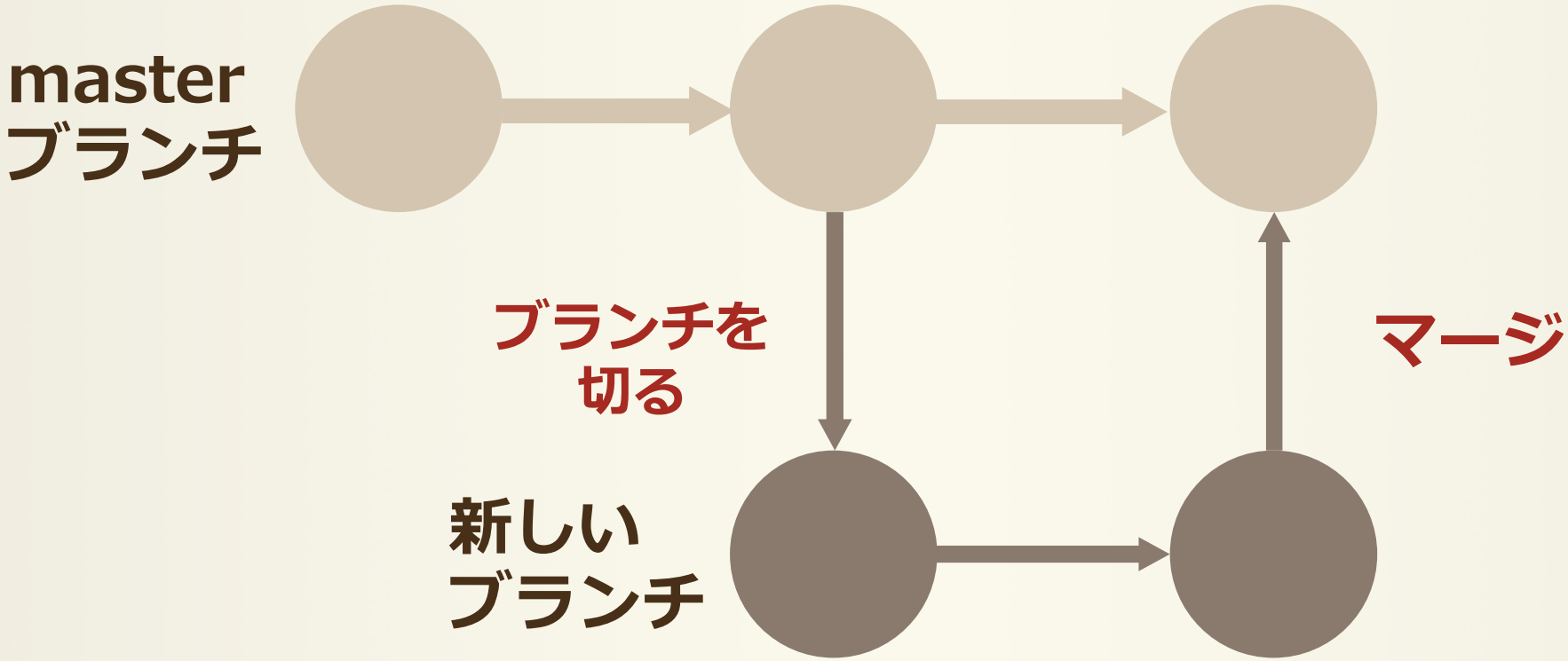
- 別の機能の開発
- バグ修正

いじって変になったら戻すのめんどくさーい

ブランチを用いた開発の例



ブランチ



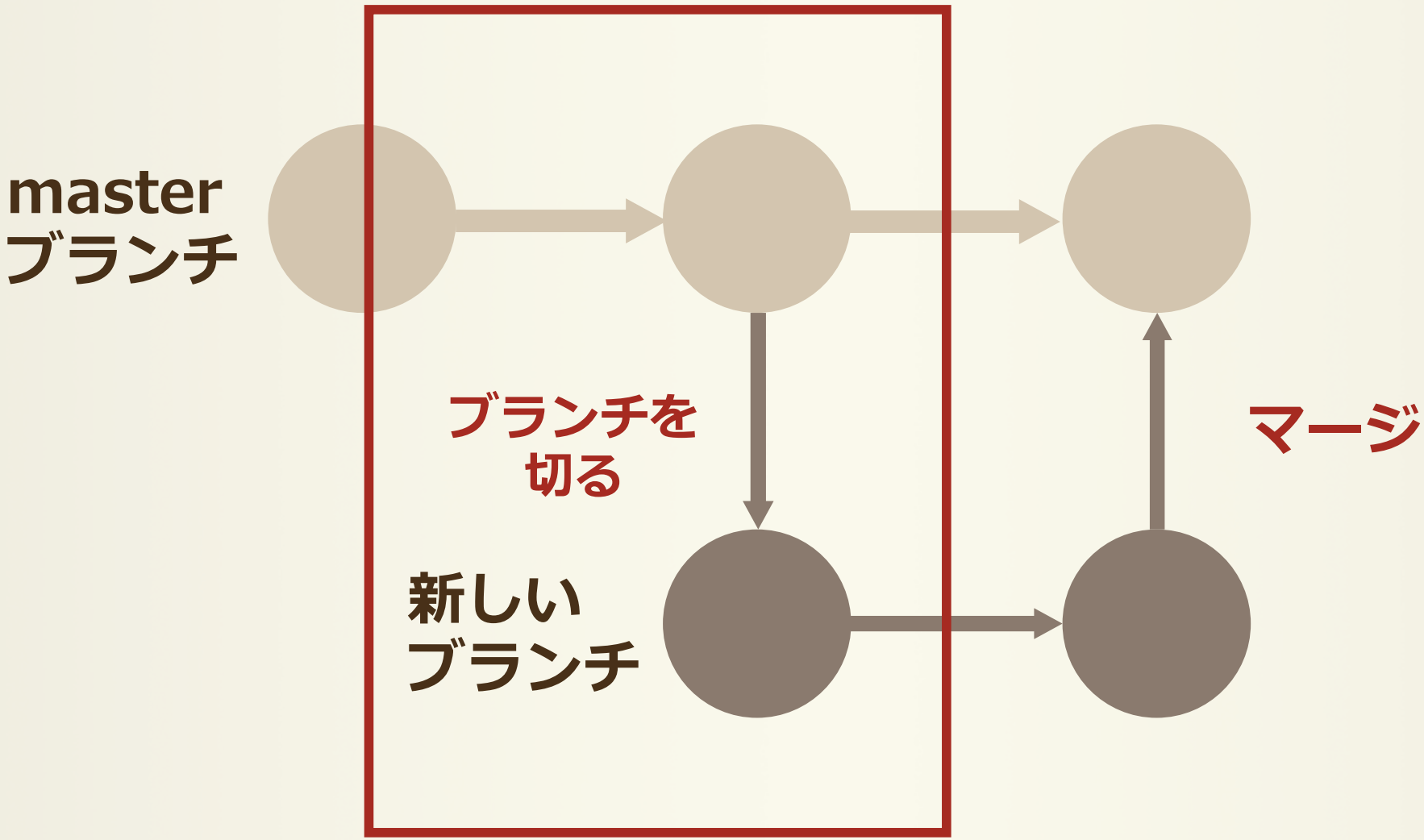
ブランチ

- **git branch**

  現在のブランチを確認

```
kut-db:main huncro$ git branch
  bpushtest
  gucci
* master
  tekito
```

ブランチ



ブランチ

- **git branch ブランチ名**

  ブランチを作成(切る)

```
kut-db:main huncro$ git branch gucci_debug
kut-db:main huncro$ git branch
  bpushtest
  gucci
  gucci_debug
* master
  tekito
```

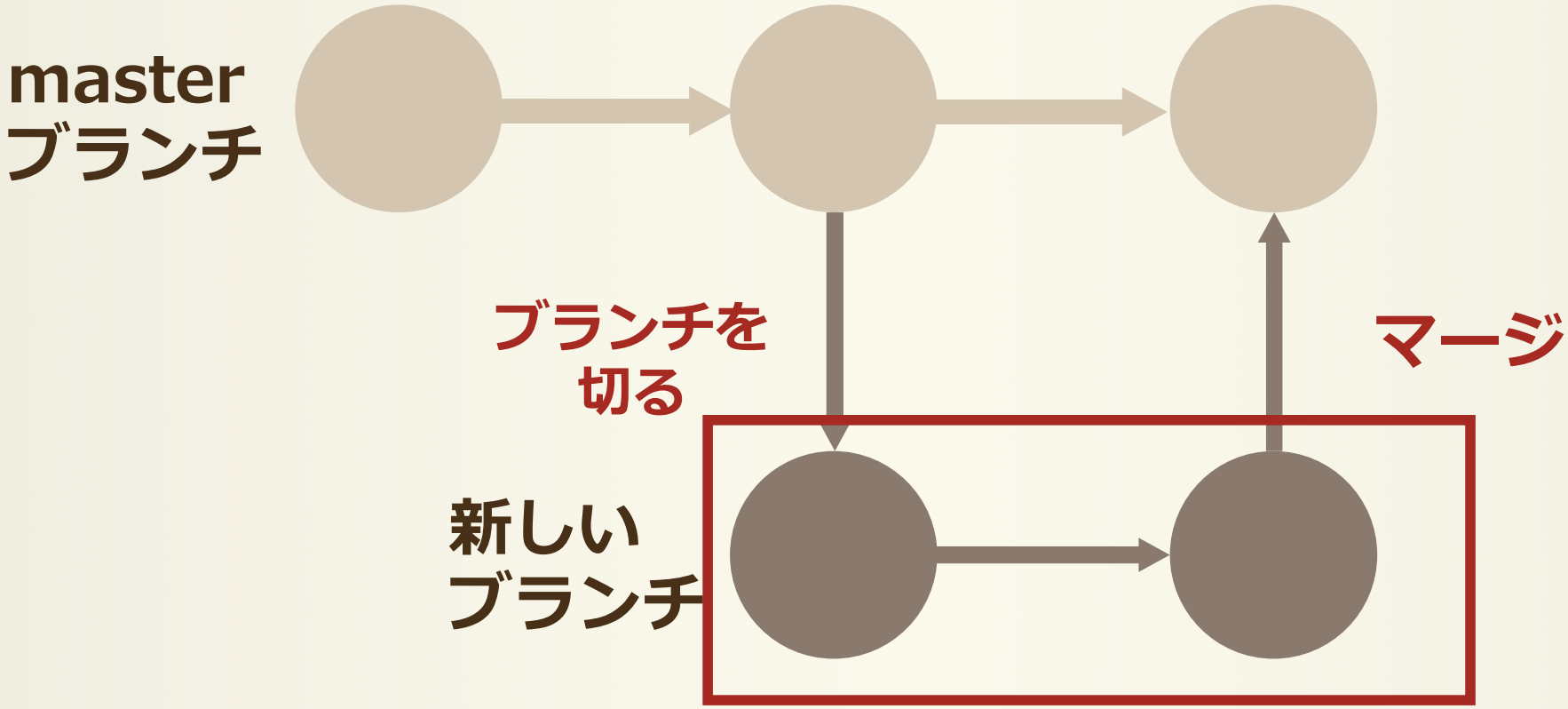
ブランチ

- **git checkout ブランチ名**

  [ブランチ名]にブランチを移動

```
kut-db:main huncro$ git branch gucci_debug
kut-db:main huncro$ git checkout gucci_debug
M       .DS_Store
Switched to branch 'gucci_debug'
kut-db:main huncro$ git branch
  bpushtest
  gucci
* gucci_debug
  master
  tekito
```

ブランチ

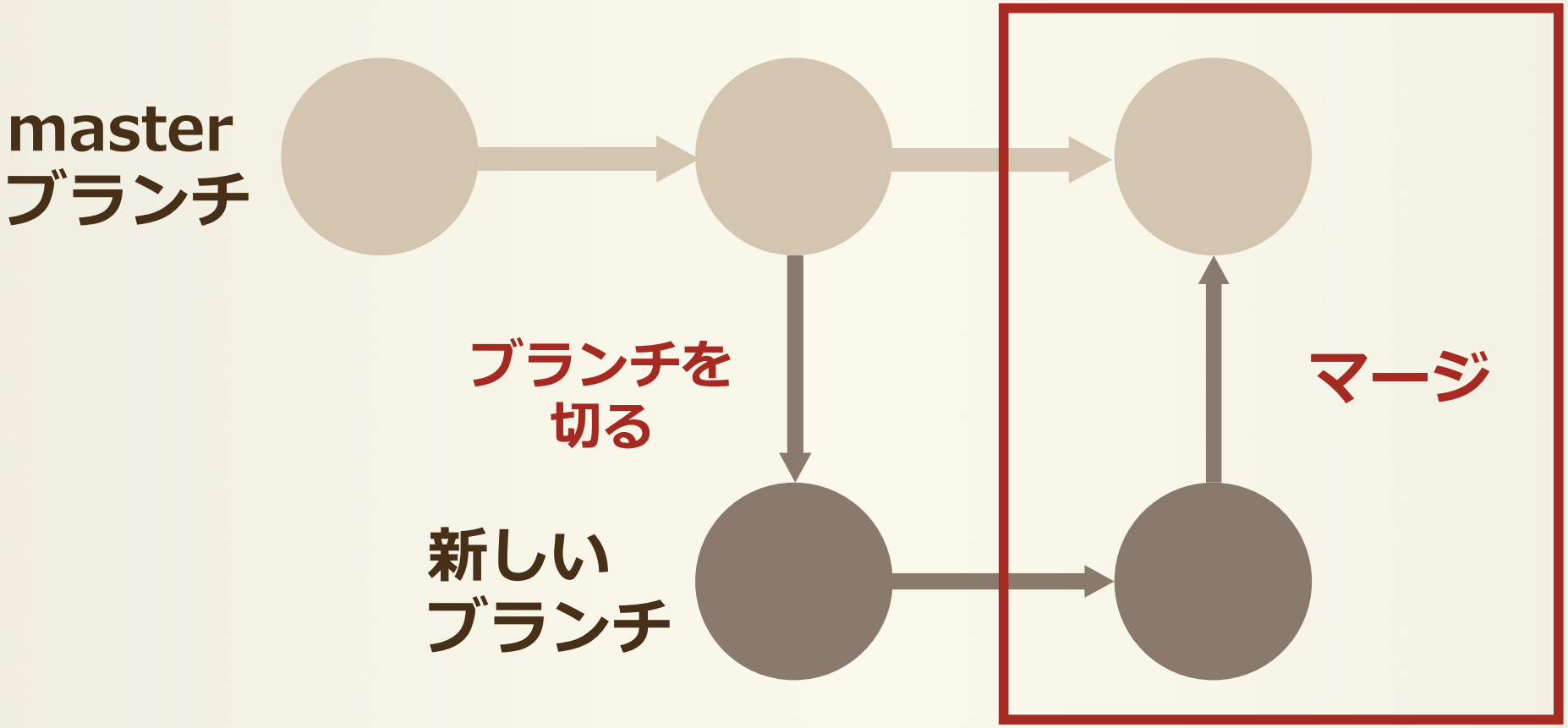


ブランチ

ファイル作成, add, commit, push

```
kut-db:main huncro$ ls
MSP           README.md     aiueo.html    index.html    practice.html web           yeah.html
kut-db:main huncro$ vim debug.txt
kut-db:main huncro$ ls
MSP           README.md     aiueo.html    debug.txt     index.html    practice.html web           yeah.html
kut-db:main huncro$ git add debug.txt
kut-db:main huncro$ git commit -m 'debug test'
[gucci_debug 6695d0b] debug test
 1 file changed, 1 insertion(+)
 create mode 100644 debug.txt
kut-db:main huncro$ git push origin gucci_debug
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 266 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
remote: This repository moved. Please use the new location:
remote:   https://github.com/KUT-soiweb/main.git
To https://huncro@github.com/KUT-soiweb/temporary-directory.git
 * [new branch]      gucci_debug -> gucci_debug
```

ブランチ



ブランチ

- **git merge ブランチ名**

  現在のブランチに[ブランチ名]での変更をマージ

```
kut-db:main huncro$ git branch
  bpushtest
  gucci
  gucci_debug
* master
  tekito
kut-db:main huncro$ ls
MSP           README.md     aiueo.html    index.html    practice.html web           yeah.html
kut-db:main huncro$ git merge gucci_debug
Updating 5d700ac..6695d0b
Fast-forward
 debug.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 debug.txt
kut-db:main huncro$ ls
MSP           README.md     aiueo.html    debug.txt     index.html    practice.html web           yeah.html
```

Agenda

1.Gitについて

2.GitHubについて

3.使ってみる

GitHubでできること

- コードなどの成果物の公開
- ソースコードを通じて
  他の開発者とコミュニケーション
- Wiki、タスク管理
- 活動状況の表示

etc

GitHub

- 「Git」の「ハブ：拠点・中心・集まり」

- Gitの仕組みを利用して、世界中の人々が自分の作品を保存、公開できるウェブサービス

GitHub

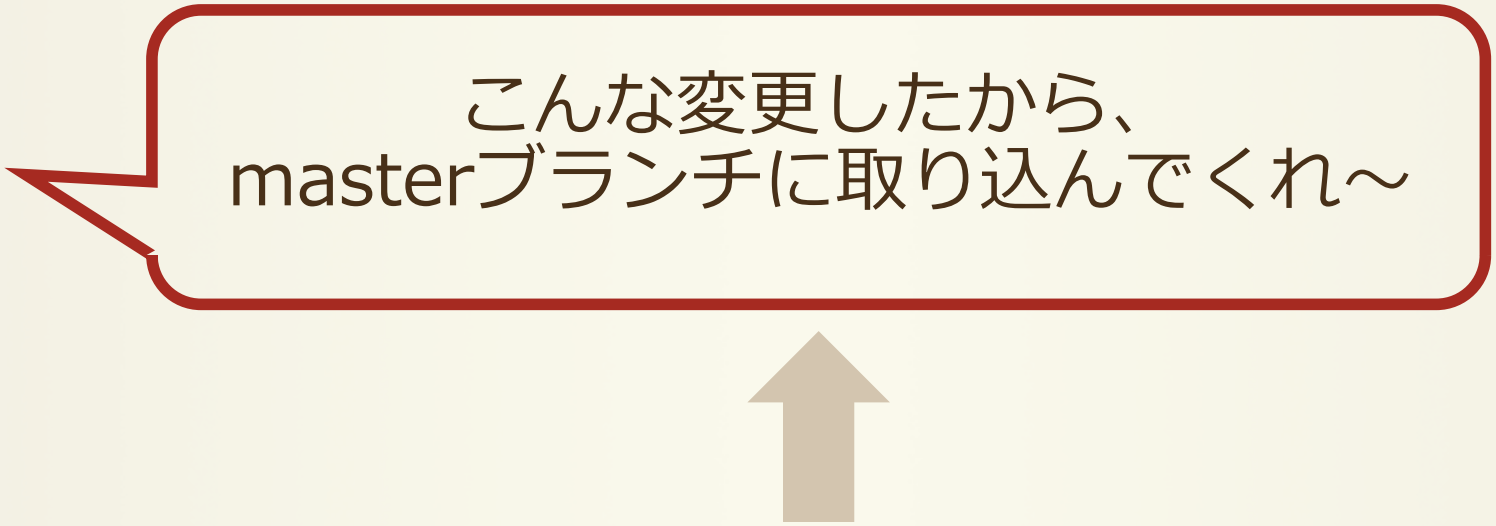
- **プルリクエスト(pull request)**
  コードレビュー機能
  修正を取り込む際に内容の確認

- **フォーク(fork)**
  他の開発者のコードを複製し、編集

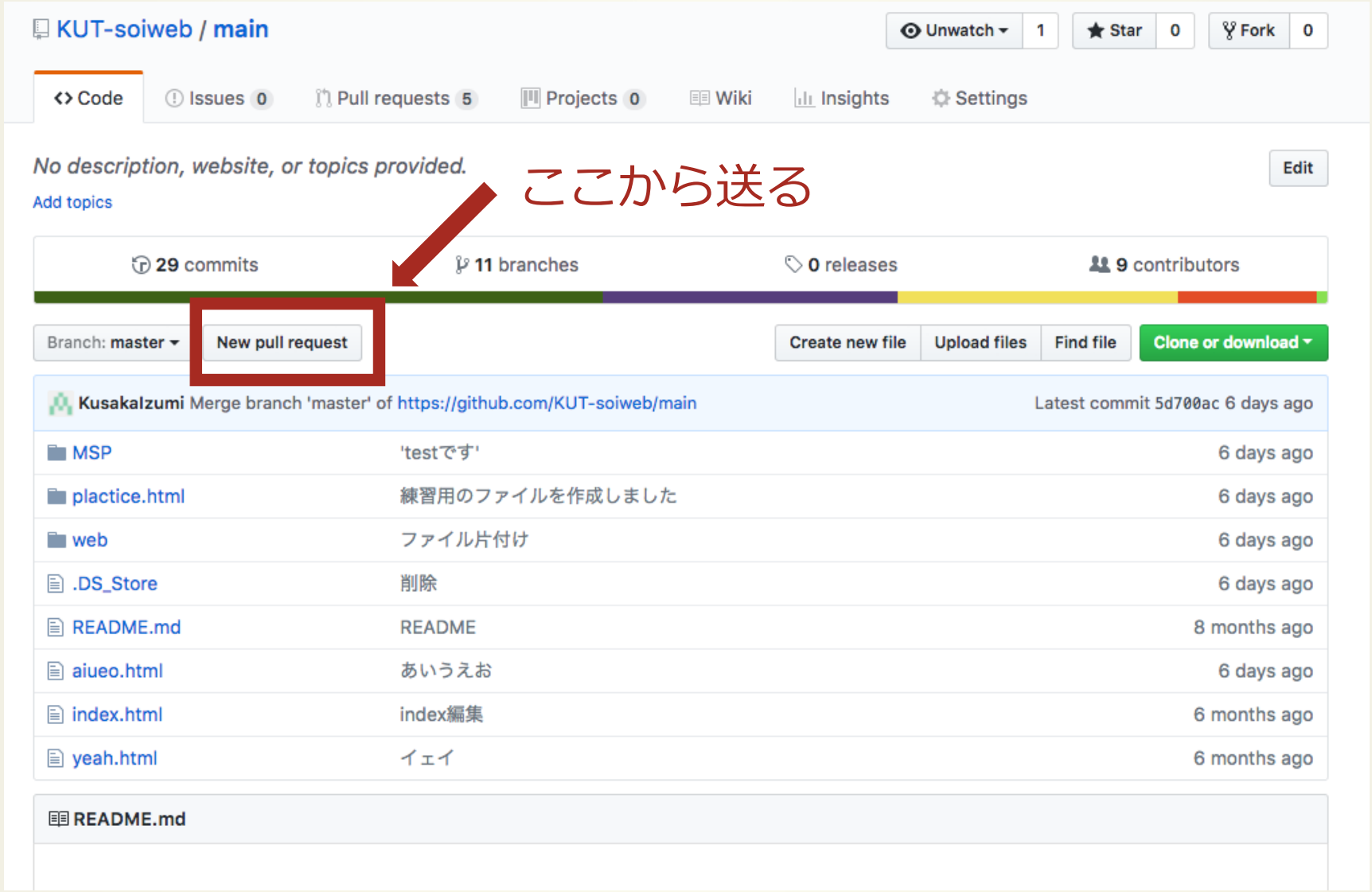
- **イシュー(issue)**
  課題管理機能

Pull Request

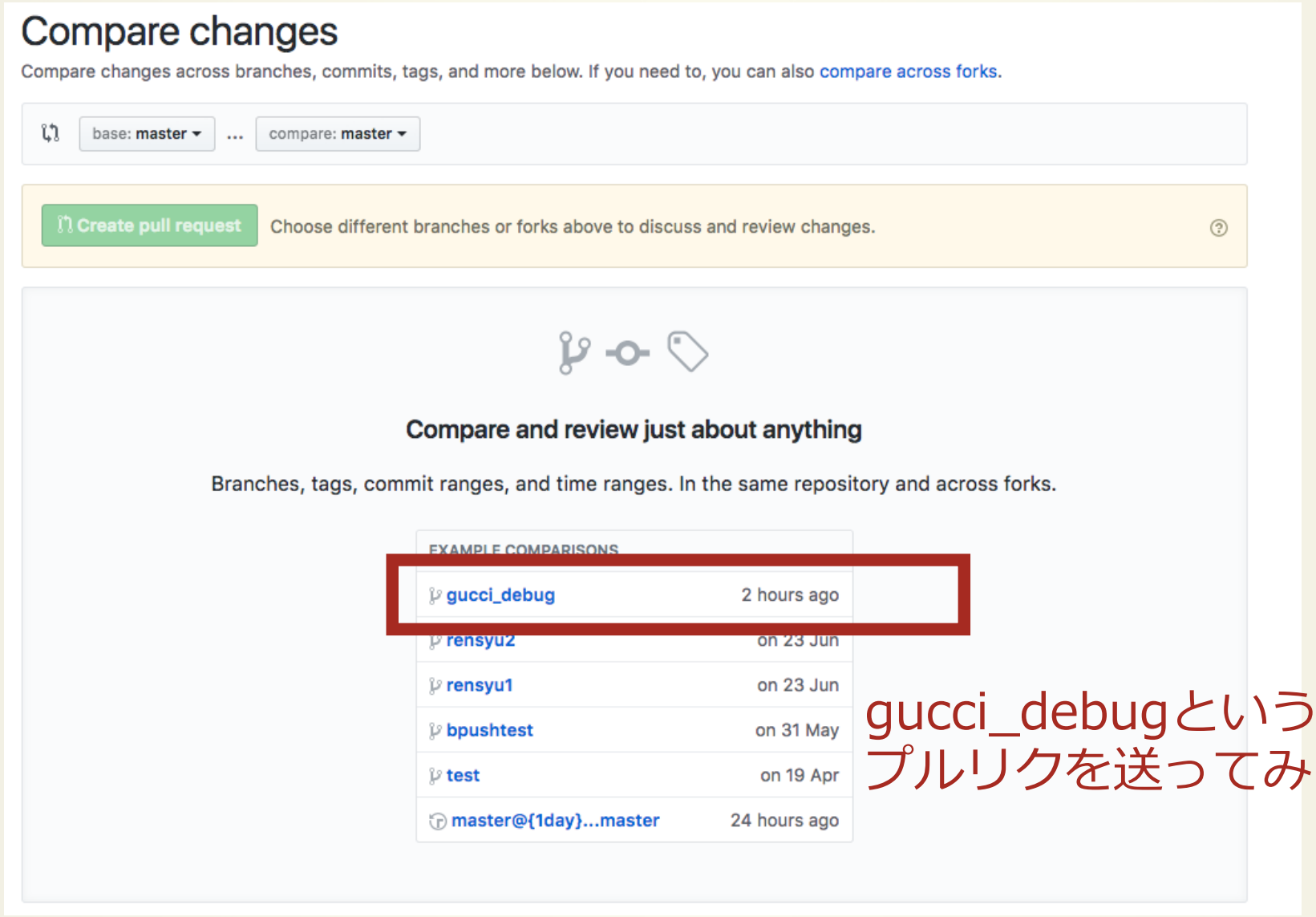
- **プルリクエスト(pull request)**
  コードレビュー機能
  修正を取り込む際に内容の確認



Pull Request手順(1/5)

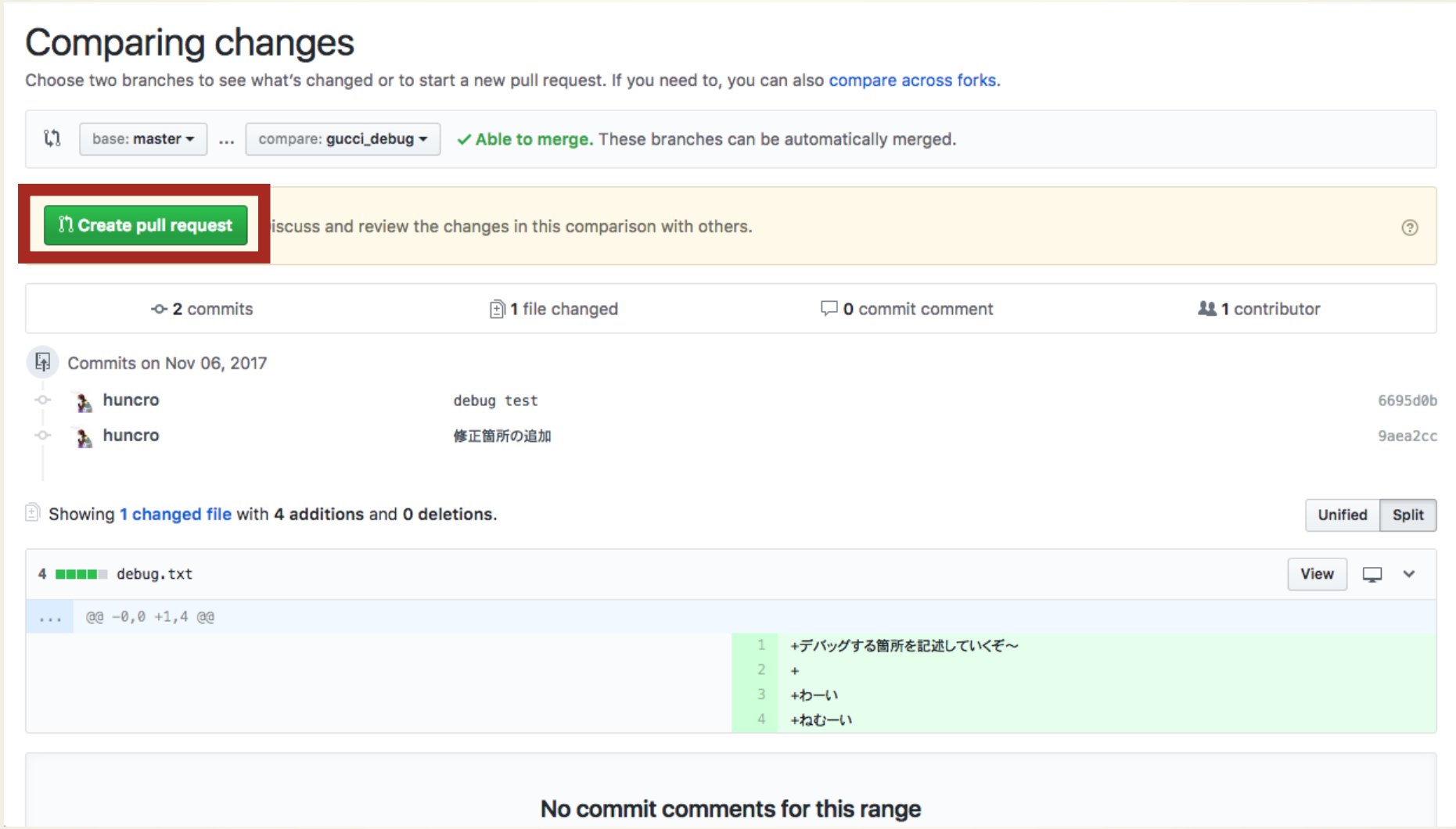


Pull Request手順(2/5)

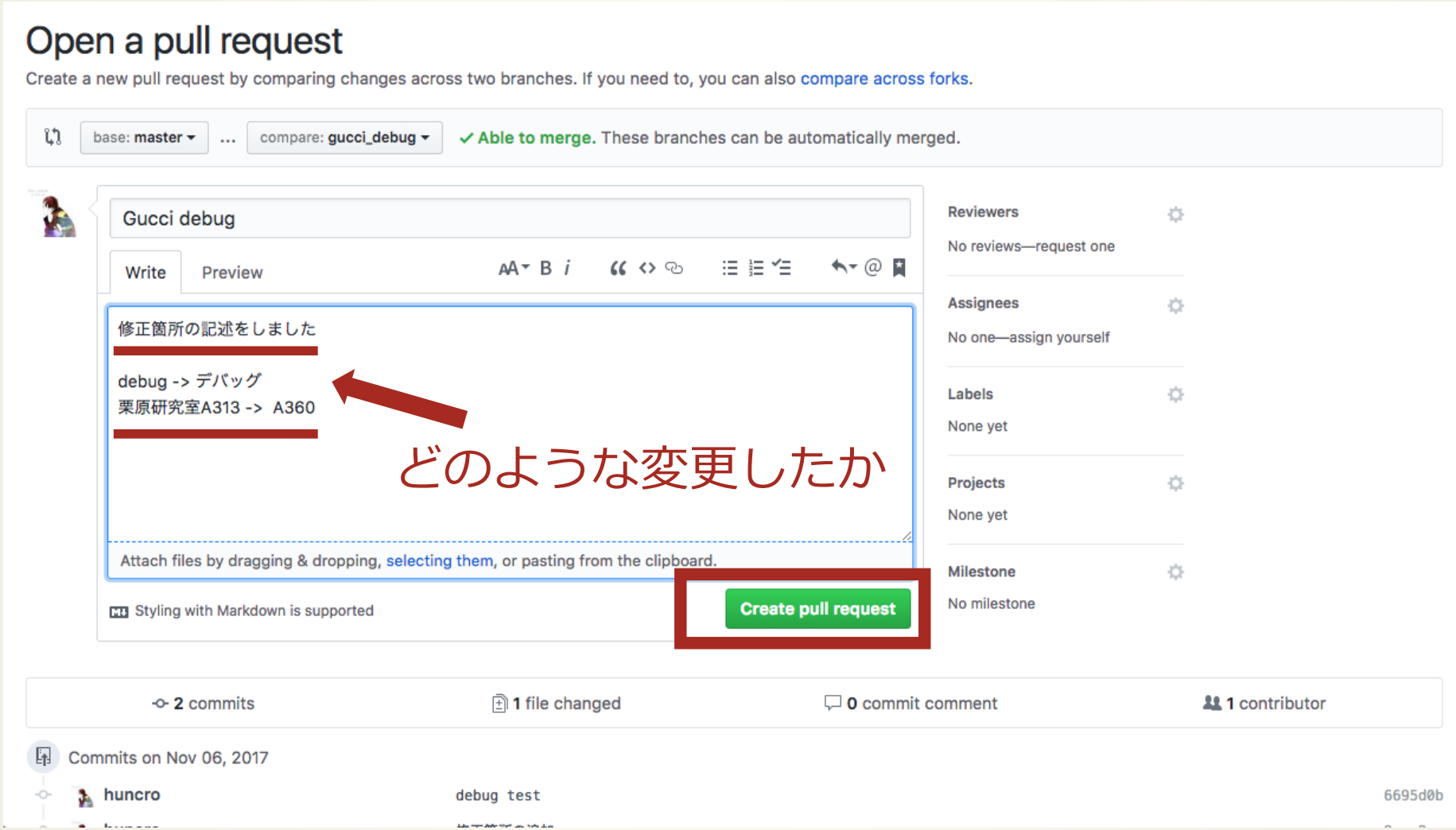


gucci_debugというブランチで
プルリクを送ってみる

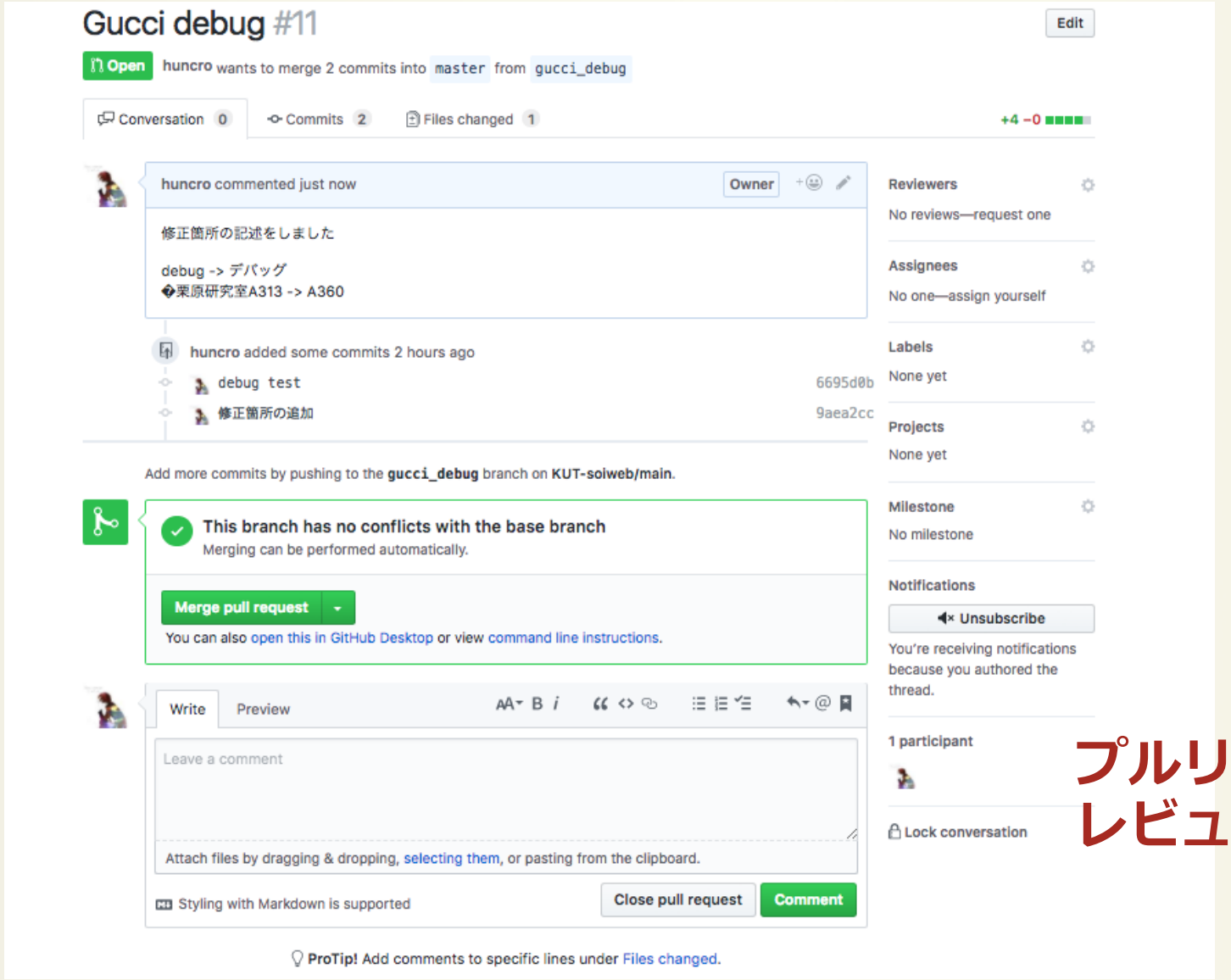
Pull Request手順(3/5)



Pull Request手順(4/5)



Pull Request手順(5/5)



プルリク完了
レビュー待ち

Agenda

1.Gitについて

2.GitHubについて

3.使ってみる

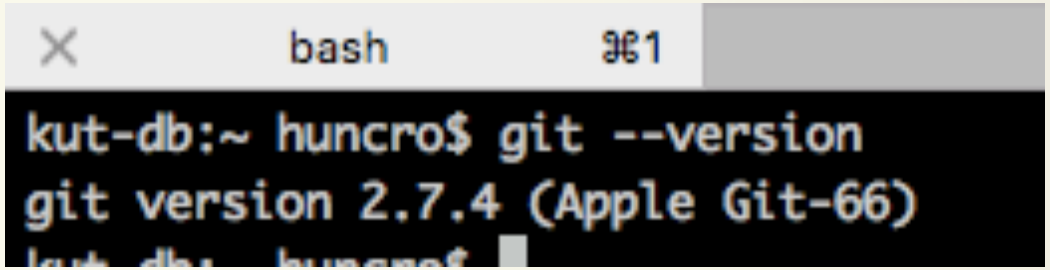
Gitのインストール

- **Macの場合**

  初めから入っていることが多い

  ターミナルなどで確認

  $ git --version



  https://git-scm.com/

  Downloads -> Mac OS X

  pkgファイルを開いて手順に従う

Gitのインストール

- **Windowsの場合**

https://git-scm.com/

Downloads -> Windows

exeファイルを開いて手順に従う

コマンドプロンプト (または Git Bash) で確認

$ git --version

実際に使ってみる

1. mkdir ディレクトリ名

2. cd ディレクトリ名

ここはgitで管理する
ディレクトリということが
わかればなんでもいい

Gitの初期設定

- ユーザ名の設定

  git config --global user.name ”ユーザ名”

- メールアドレスの設定

  git config --global user.email メールアドレス

→　確認

git config -l

GitHubでも
同じものを使う

GitHubの登録

- アカウント作成

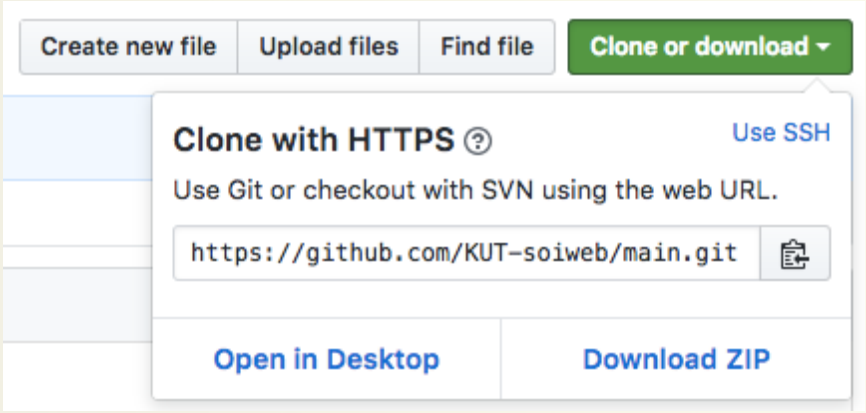
  https://github.com/

- soiwebのリポジトリを取得

  git clone https://ユーザ名@github.com/～

取得先によって変わる

登録したユーザ名



実際に使ってみる(ブランチなし)

1. clone してきたフォルダに入る　(今回は cd main)
2. ファイル作成
3. git add ファイル名
4. git commit –m ‘どのような編集をしたのか’
5. git push origin master

git pull
→　他の誰かがpushしていた場合
　　適宜必要

実際に使ってみる

その他のそこそこ使うコマンド

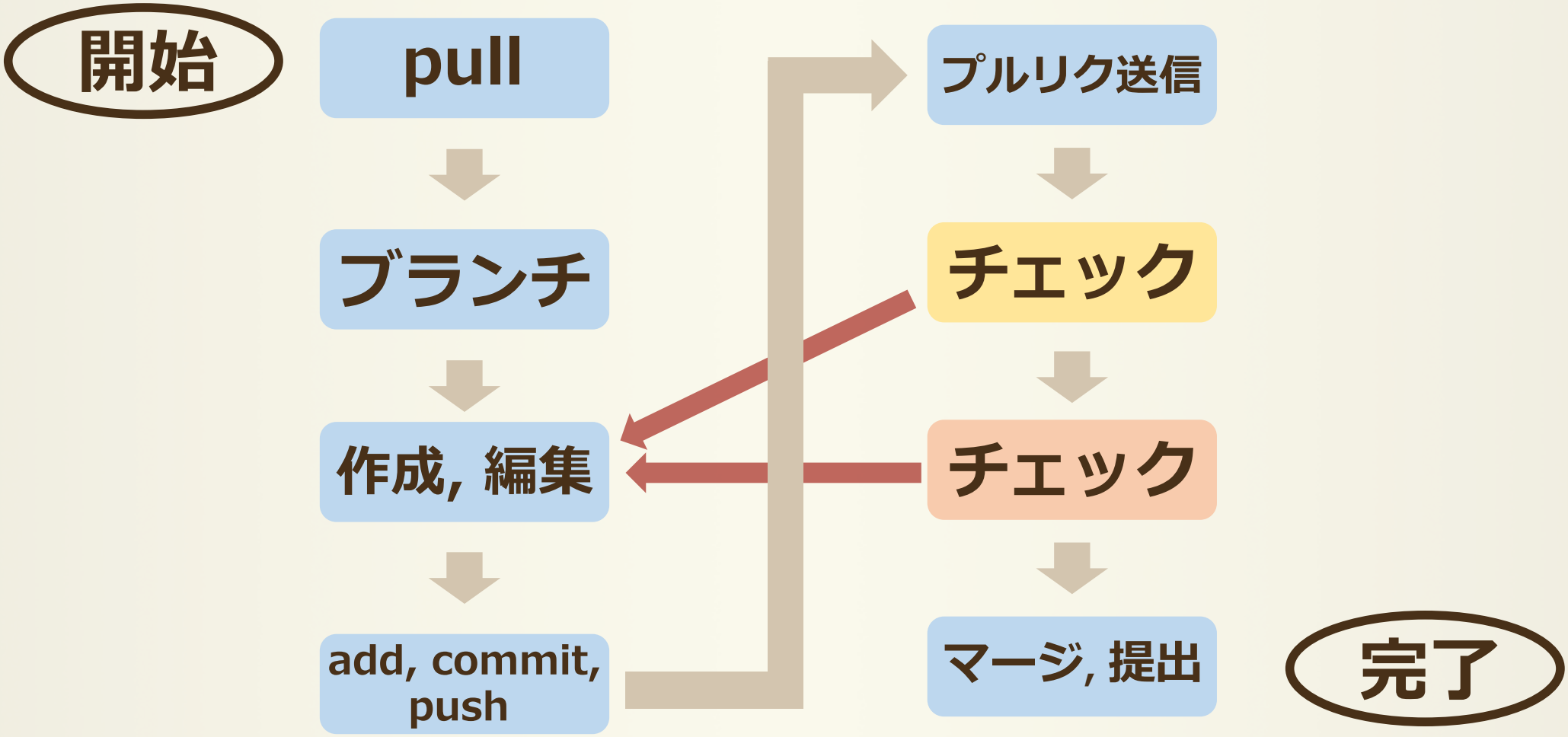
- git log
- git status
- git reset
- git rm

各自調べてくれ

実際に使ってみる(ブランチあり)

次の流れに従って使ってみよ～

そいうぇぶふろーちゃーと



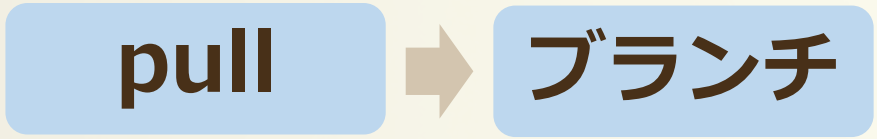
そいうぇぶふろーちゃーと

pull

git pull origin master

- リモートリポジトリの変更を反映
  これを忘れると、
  旧バージョンのファイルをいじっている状態

そいうぇぶふろーちゃーと



git branch 新ブランチ名
git checkout 新ブランチ名

- ブランチの作成及び移動

そいうぇぶふろーちゃーと

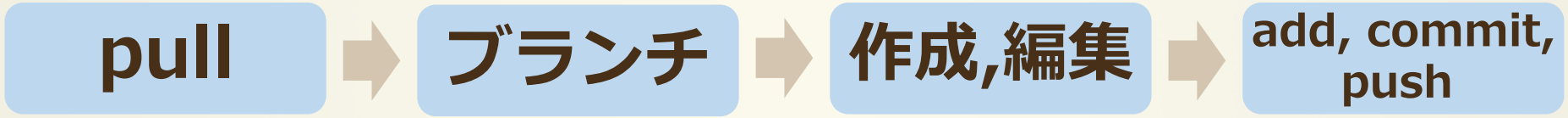
pull → ブランチ → 作成,編集

エディタ名　ファイル名

エディタ名　＝　vim, atom, emacs　など

- ターミナルからファイルを開く
  環境があることが前提, (windowsもできたはず)

そいうぇぶふろーちゃーと



git add ファイル名
git commit –m ‘内容’
git push origin ブランチ名

- コミットの内容は「どのような変更したか」、など
- ブランチ名でpush

そいうぇぶふろーちゃーと

pull → ブランチ → 作成,編集 → add, commit, push → プルリク送信

GitHubのページ上からPull Requestを送信する

- P.41～ Pull Requestの手順参照

そいうぇぶふろーちゃーと

pull → ブランチ → 作成,編集 → add, commit, push → プルリク送信 → チェック

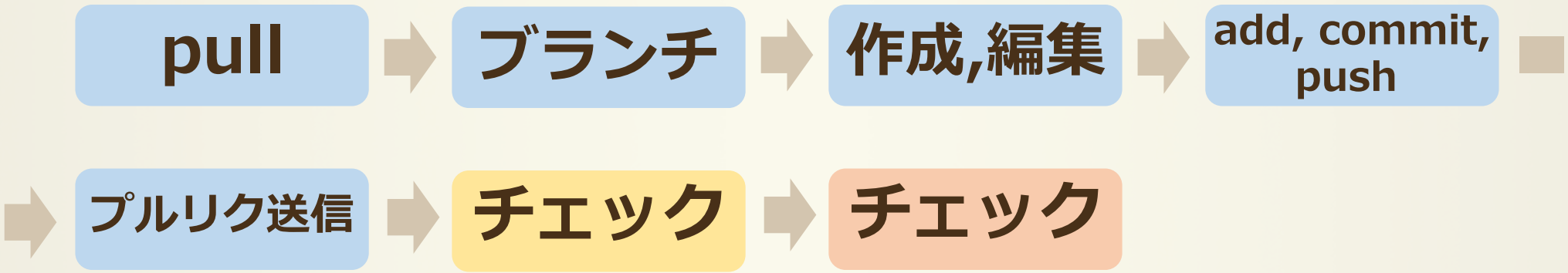
git pull origin master
git checkout origin/ブランチ名

Pull Requestを
送ってきたブランチ名

- このブランチの状況を見る
  ファイルを開いて確認
  - **open ファイル名** (mac)
  - **ファイル名** (windows)

HTMLファイルなどを
ブラウザから開けたはず

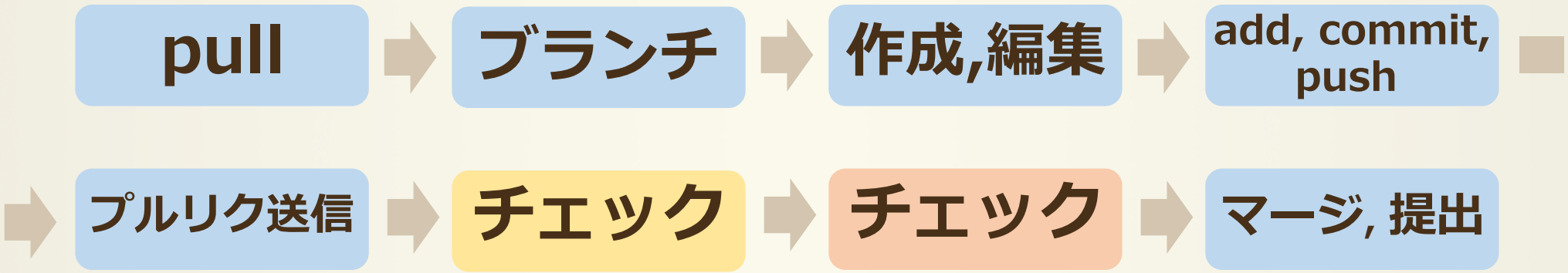
そいうぇぶふろーちゃーと



一つ前の手順と同じ

- ソースコード見るにはブランチ移動して
  エディタ名 ファイル名

そいうぇぶふろーちゃーと



チェック後
- slackから先生にファイル提出
- GitHubからマージ
- (ブランチ削除)

まとめ

- 細かいコマンドはまだまだある
- リモートとのやりとりは、push, pull
- 定期的にpullしてリモートとのズレを抑える