Gitとは？

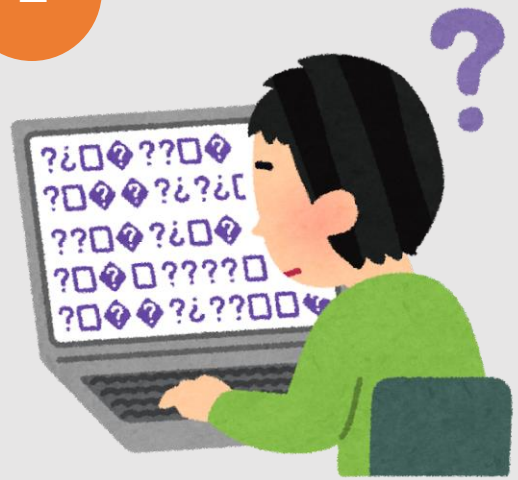
分散型バージョン管理システム
（ファイルの変更履歴を管理してくれるシステムのひとつ）

techUP

1

別の開発内容でデバックが発生した際に途中で開発を中断して、branchを切り替えてデバック修正ができる！
完了したら開発にまた戻ればOK

2



内容を変更するたびにcommit（履歴またはバッアップ）するので、
コードを書いている途中で、
大きなミスをしても、
ミスする前の状態に戻すことができる！

3

Git内で内容を更新するたびにファイルの変更履歴が記録できるので、
バグが発生しても問題のコードを見つけやすい！

Gitの全体構成図 techUPバージョン

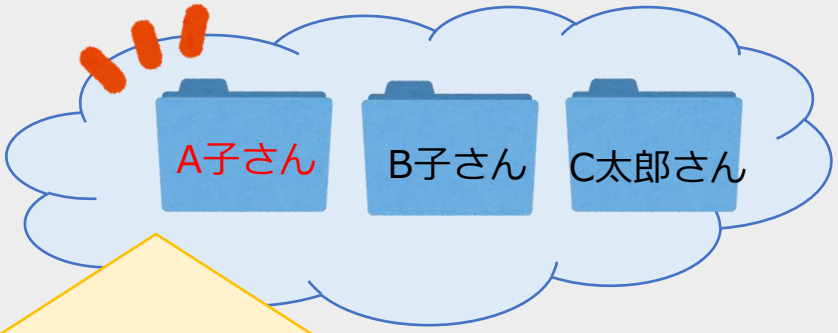
techUP

Git

techUP受講生

アクセス

remote repository
（皆のファイルの保管場所）



push

fetch

pull

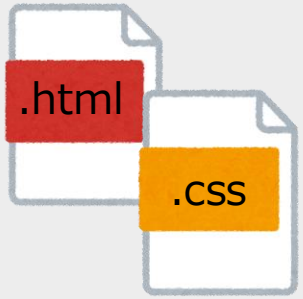
ローカル環境

Work tree

A子さん

add

index



index_2_3の中身

commit

local repository
（自分のPC内にある保管場所）

よく出てくる用語集

techUP

- **repository** …ファイルやディレクトリ（フォルダみたいな）類の変更履歴を管理するところ
- **clone**… repositoryのファイルやディレクトリを、そのまんまローカル環境にダウンロードする
  みたいな作業（開発を行う際に最初に1回だけする作業になります）
- **commit**…index を local repository に反映（記録）させる操作
- **push**… local repository を remote repositoryへ反映させる操作
- **pull**… remote repositoryをlocal repositoryへ反映させる操作
  remote repository と local repository との差分がダウンロードされる
- **fetch**…リモートリポジトリからローカルリポジトリに最新の情報を取得させる
  pullと違って local repository のファイルを更新しない
- **branch**…主導で動いているプロジェクト本体から分岐させることにより
  プロジェクト本体に影響を与えずに開発を行える機能（詳細後ほど）
- **merge**…皆のそれぞれのブランチを統合すること

Gitの全体構成図 開発チームバージョン①

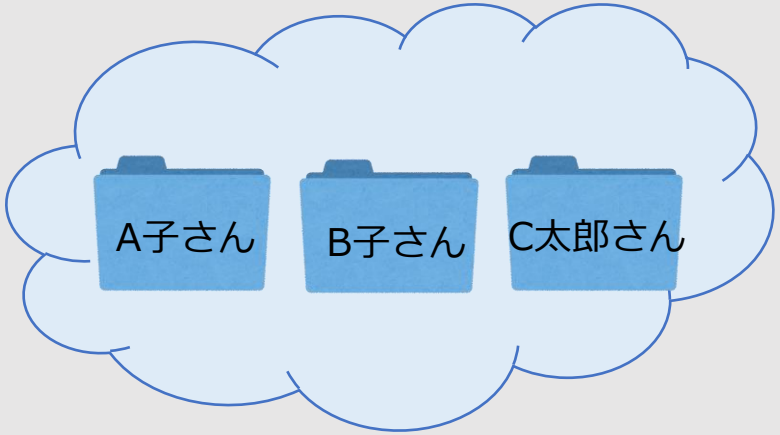
techUP

開発チーム

開発チーム皆が
アクセス可能

自分のPCに
clone可能

remote repository
（皆のファイルの保管場所）



repository

ファイルやディレクトリ類の変更履歴を管理するところで、
特にインターネット環境下で管理するリポジトリ
（Gitで指定して、自分のPCにダウンロードした分）をリモートリポジトリという！
インターネットに保管しているので、全世界の人がリモートリポジトリにアクセス可能！

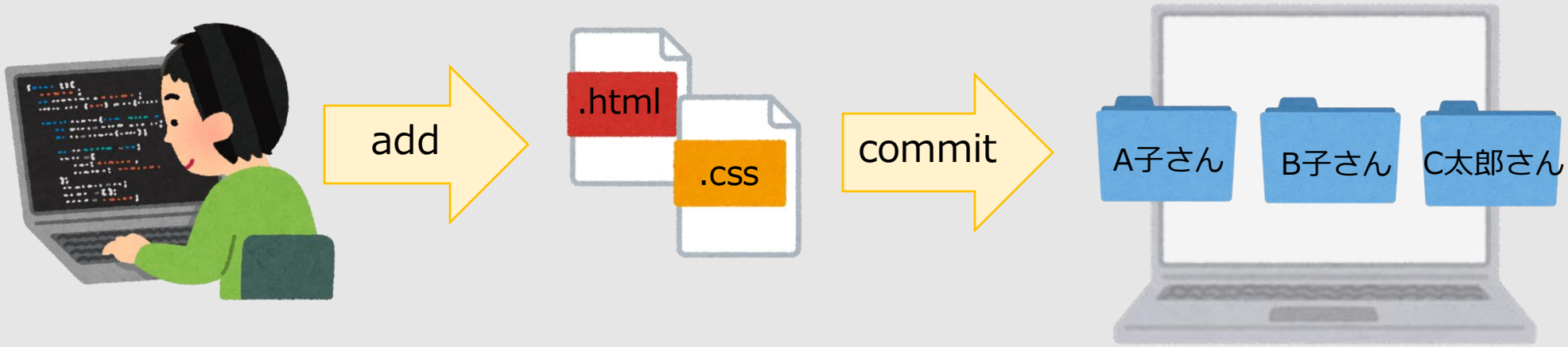
Gitの全体構成図 開発チームバージョン②

techUP

work tree

index

local repository
（自分のPC内にある保管場所）



work tree

…実際に作業しているディレクトリのこと

index

…ワークツリーとローカルリポジトリの間に置かれていて、
ワークツリーで編集したディレクトリを、
リポジトリにコミットするための準備場所！
必要なファイルだけをコミットできたり、
無駄なファイル交換が防げる！

local repository

…特に自分のPCで管理するリポジトリ
（Gitで指定して、自分のPCにダウンロードした分）をローカルリポジトリという

ブランチとは？①

techUP

branchとは…

・主導で動いているプロジェクト本体から分岐させることにより、
プロジェクト本体に影響を与えずに開発を行える機能

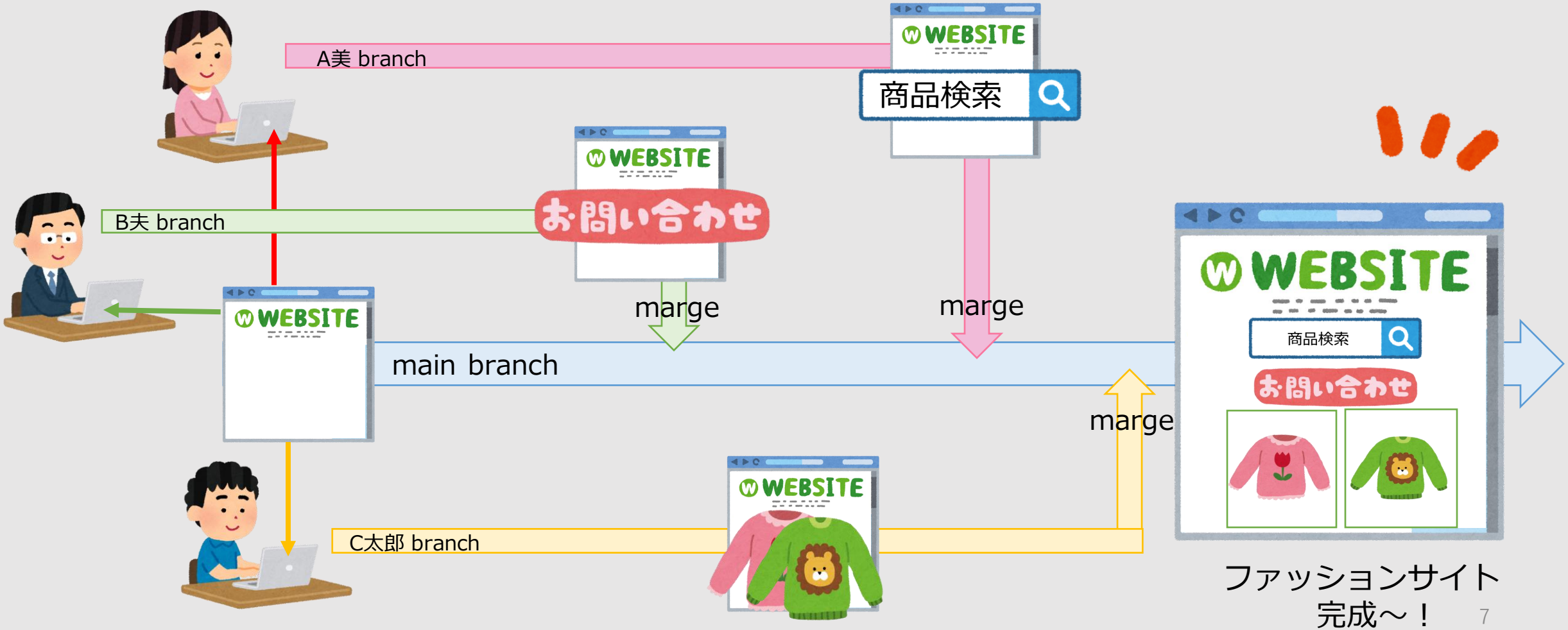
・また、履歴の流れを分岐して記録してくれる
分岐したブランチは、他のブランチの影響を受けないので
同じリポジトリ内で複数の変更を同時に進められます！

・さらに分岐したブランチは、
他のブランチと統合（marge）することで
ひとつのブランチ（main branch）にまとめられる！

↑ 木の枝に例えられがち

ブランチの説明②

チームのメンバーは、他のメンバーの作業の影響を受けないように、
main branchから、自分の作業用branchをそれぞれ作成し、
そして作業が終わったメンバーは、main branchに自分のブランチの変更を読み込ませる！



ファッションサイト
完成～！

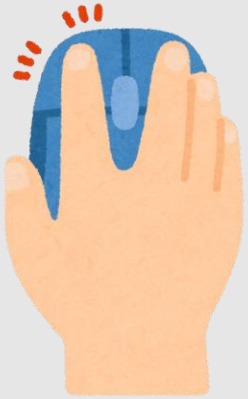
local repositoryでの操作とは？①

techUP

ローカルリポジトリでの操作とは、普段マウスでクリックして操作しているのを
文字を打って操作するということ

普段の操作

新しいファイルを作るときは
「新規ファイル」ボタンをクリックして
名前を「index_3_3」に変更する



local repositoryでの操作

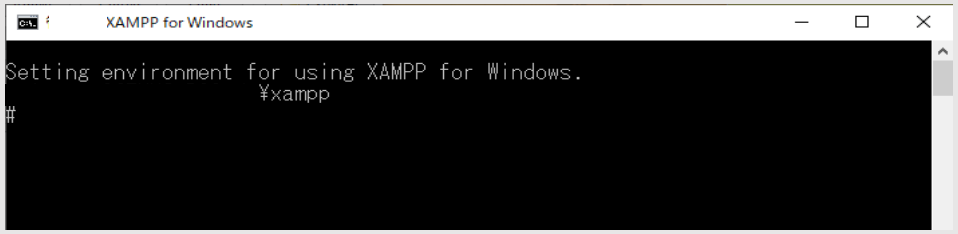
新しいファイルを作るときは
「mkdir index_3_3」と文字を打って
作成する

local repositoryでの操作とは？②

techUP

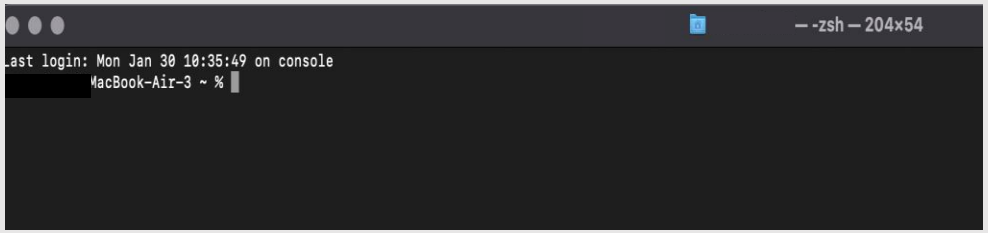
どこで文字を打って操作するの？

Windowsの場合
Git Bash 使用



スタートメニューを右クリックすると
「プロンプト」出てくる
※デスクトップの虫眼鏡で
「Git Bash」検索でも出てくる！

MacBookの場合
ターミナル 使用



画面右上虫眼鏡マーク押して「ターミナル」と検索

この黒い画面で文字を打って操作します！

Gitでの課題提出方法

techUP

Gitは他の人の作業もいじれるので、
実際の現場で大きなミスをすると、「他の人の作業を抹消してしまう」なんてことも起こり得ます。
そうならないためにも、今のうちにどんどん失敗して、
実際現場に出たときに大きなミスをしないようにしましょう！

それではさっそく、
コマンドプロトコルもしくはターミナルを開いて
課題を提出する下準備からしてきましょう！

Gitでの課題提出方法【準備】

1. 「techUP_study」をcloneしたい場所を指定する（この図だと「Documents」が指定場所）
2. 「git clone #URL」 を打ち込む
3. 「ls」で1で指定した場所の中身を確認
4. 「cd techUP_study」で「techUP_study」に移動
5. 「ls」で自分の名前が出てくればclone完了

赤線：現在地
黄色：自分で打つ文字
緑色：マウス操作

```
[katomizuki@MacBook-Air-4 Documents % git clone https://github.com/fujimoto-mio/techUP_study.git
Cloning into 'techUP_study'...
remote: Enumerating objects: 31533, done.
remote: Counting objects: 100% (605/605), done.
remote: Compressing objects: 100% (326/326), done.
remote: Total 31533 (delta 313), reused 492 (delta 242), pack-reused 30928
Receiving objects: 100% (31533/31533), 238.97 Mi
Resolving deltas: 100% (8551/8551), done.
[katomizuki@MacBook-Air-4 Documents % ls
Adobe           Zoom            techUP_study
[katomizuki@MacBook-Air-4 Documents % cd techUP_study
[katomizuki@MacBook-Air-4 techUP_study % ls
```

「techUP_study」が出れば
cloneできている

「ls」直下に
自分の名前出てくればOK！

Gitでの課題提出方法①

techUP

1.「cd ~」でローカルの最上階層へ移動
2.「cd 11ページでcloneした場所」で techUP_studyをcloneした場所へ移動
（この図だと「Documents」が指定場所）
3.「ls」で中身を確認
4.「cd techUP_study」で「techUP_study」へ移動
5.「ls」で中身を確認し、「番号__自分」の名前が出てくればOK

```
[katomizuki@MacBook-Air-4 加藤瑞希 % cd ~
[katomizuki@MacBook-Air-4 ~ % cd Documents
[katomizuki@MacBook-Air-4 Documents % ls
 Adobe                  Zoom                   techUP_study
[katomizuki@MacBook-Air-4 Documents % cd techUP_study
[katomizuki@MacBook-Air-4 techUP_study % ls
```

4で「techUP_study」に移動

「ls」直下に
自分の名前出てくればOK！

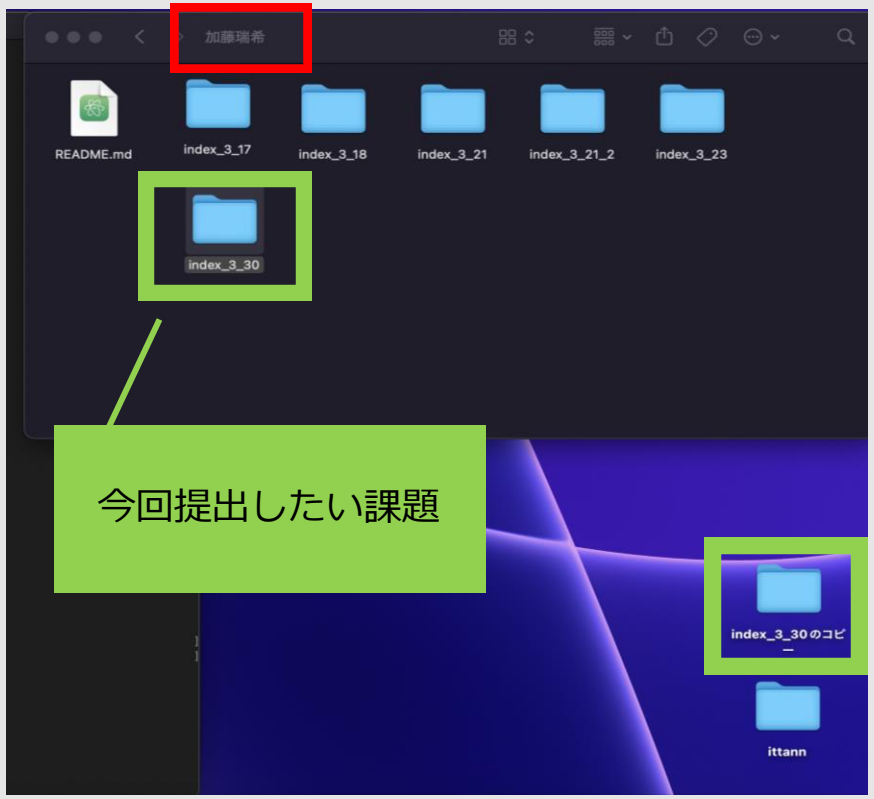
Gitでの課題提出方法②

6.「cd 番号__自分の名前」でtechUP__study内の自分のフォルダへ移動

7.「ls」で中身を確認（この画面は閉じずに8の操作へ）

※この状態ではまだ、今回提出の「index3-30」はまだ出てこない

```
[katomizuki@MacBook-Air-4 techUP_study % cd 加藤瑞希
[katomizuki@MacBook-Air-4 加藤瑞希 % ls
README.md       index_3_17      index_3_18      index_3_21      index_3_21_2    index_3_23
```



8.提出するフォルダを、マウス操作でcloneした場所に移動

※この時提出するフォルダは、必ずコピーをとってローカルに保存しておく！

必ずコピーをとって
ローカルに保存！

Gitでの課題提出方法③

9. 7の画面を再度開き「ls」で中身を確認
10. 「git branch」で今どこのbranchにいるかを確認　(もしmain以外にいる状態なら、main branchに移動)
11. 「git pull」で　main branch　を最新の状態に更新 (main branchにいる状態で行う)
12. 「git branch feature/名前_課題」でbranchを作成
13. 「git branch」で12でちゃんとbranch作成できているか確認

8のマウス操作が反映されている

```
katomizuki@MacBook-Air-4 加藤瑞希 % ls
README.md       index_3_17      index_3_18      index_3_21      index_3_21_2    index_3_23      index_3_30
katomizuki@MacBook-Air-4 加藤瑞希 % git branch
  feature/kato_3_17
  feature/katomizuki_3_17_2
  feature/katomizuki_3_17_3
  main
* master
katomizuki@MacBook-Air-4 加藤瑞希 % git pull
Already up to date.
katomizuki@MacBook-Air-4 加藤瑞希 % git branch featuer/katomizuki_3_30
katomizuki@MacBook-Air-4 加藤瑞希 % git branch
  featuer/katomizuki_3_30
  feature/kato_3_17
  feature/katomizuki_3_17_2
  feature/katomizuki_3_17_3
  main
* master
```

緑文字が今自分がいるbranch

12で作成したのを
13で確認

提出したいフォルダの
branchを作成する

Gitでの課題提出方法④

14.「git checkout feature/名前_課題」で12で作成したbranchに切り替え
15.「git branch」で切り替えられてるか確認
16.「git diff」で差分がないかを確認（この時再提出の場合は1回目提出と2回目提出時の差分が表示される）
17.「git add .」で提出したいフォルダをbranchに追加する
18.「git status」でちゃんとaddできている確認

```
katomizuki@MacBook-Air-4 加藤瑞希 % git checkout featuer/katomizuki_3_30
Switched to branch 'featuer/katomizuki_3_30'
katomizuki@MacBook-Air-4 加藤瑞希 % git branch
* featuer/katomizuki_3_30
  feature/kato_3_17
  feature/katomizuki_3_17_2
  feature/katomizuki_3_17_3
  main
  master
katomizuki@MacBook-Air-4 加藤瑞希 % git diff
katomizuki@MacBook-Air-4 加藤瑞希 % git add .
katomizuki@MacBook-Air-4 加藤瑞希 % git status
On branch featuer/katomizuki_3_30
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   index_3_30/index_3_30.html
        new file:   index_3_30/style.css
```

14で切り替えられてる！

Gitでの課題提出方法⑤

19.「git commit –m “コメント”」でcommit

20.「git push origin feature/名前_課題」でpush

（github上にbranchが作成されているので、2回目以降は「git push feature/名前_課題」でOK）

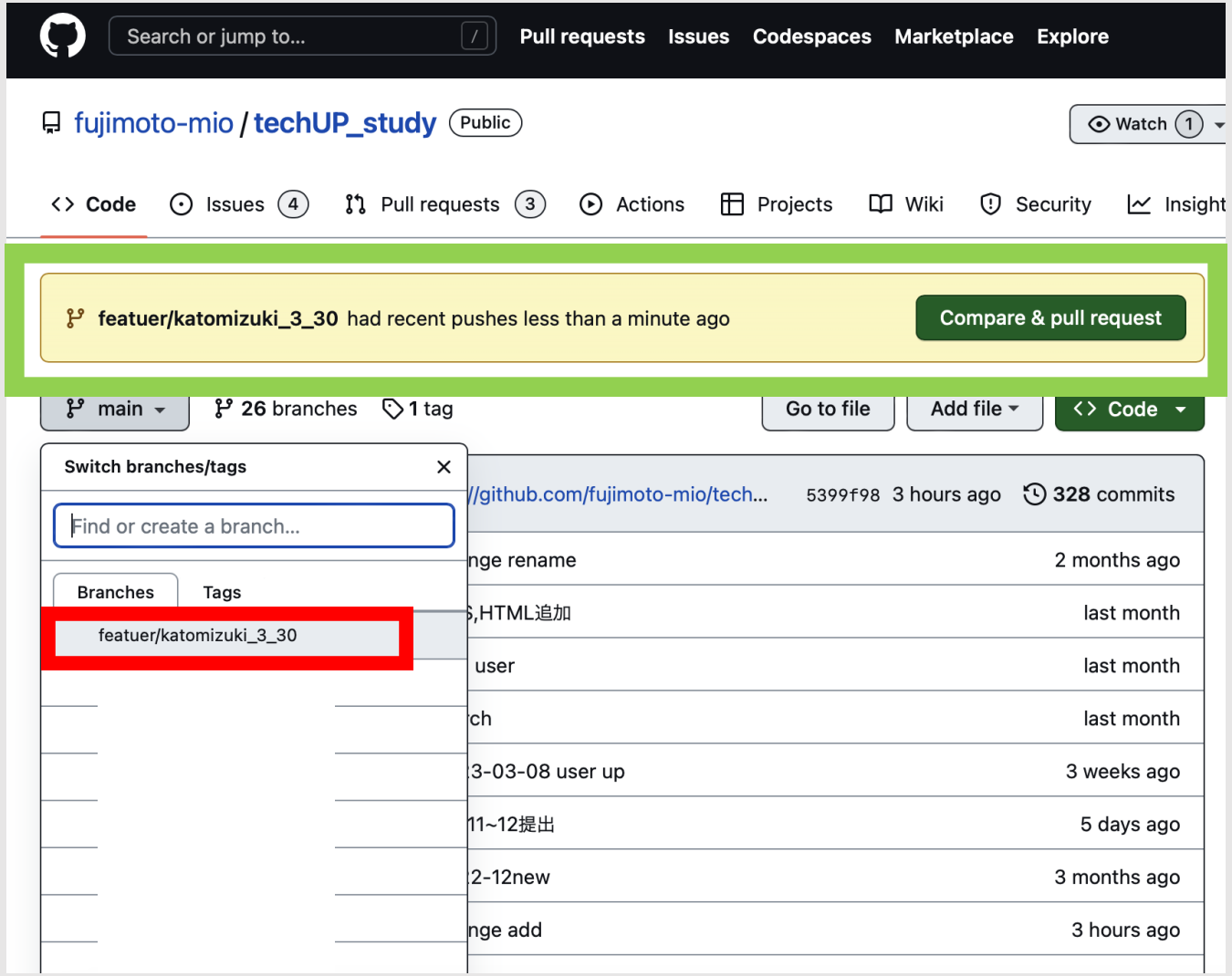
21.「git status」でちゃんとpushできているか確認

```
katomizuki@MacBook-Air-4 加藤瑞希 % git commit -m "L3-30課題提出"
[featuer/katomizuki_3_30 4ff8e497] L3-30課題提出
 2 files changed, 41 insertions(+)
 create mode 100644 "\345\212\240\350\227\244\347\221\236\345\270\214/index_3_30/index_3_30.html"
 create mode 100644 "\345\212\240\350\227\244\347\221\236\345\270\214/index_3_30/style.css"
katomizuki@MacBook-Air-4 加藤瑞希 % git status
On branch featuer/katomizuki_3_30
nothing to commit, working tree clean
katomizuki@MacBook-Air-4 加藤瑞希 % git push -u origin featuer/katomizuki_3_30
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 887 bytes | 887.00 KiB/s, done.
Total 6 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'featuer/katomizuki_3_30' on GitHub by visiting:
remote:      https://github.com/fujimoto-mio/techUP_study/pull/new/featuer/katomizuki_3_30
remote:
To https://github.com/fujimoto-mio/techUP_study.git
 * [new branch]        featuer/katomizuki_3_30 -> featuer/katomizuki_3_30
branch 'featuer/katomizuki_3_30' set up to track 'origin/featuer/katomizuki_3_30'.
katomizuki@MacBook-Air-4 加藤瑞希 %
```

この辺の長文は
あまり気にしなくて良い
むしろこの長文が出てきたら
正常にpushできている

Gitでの課題提出方法⑥

techUP



22.Git hubに移動し、

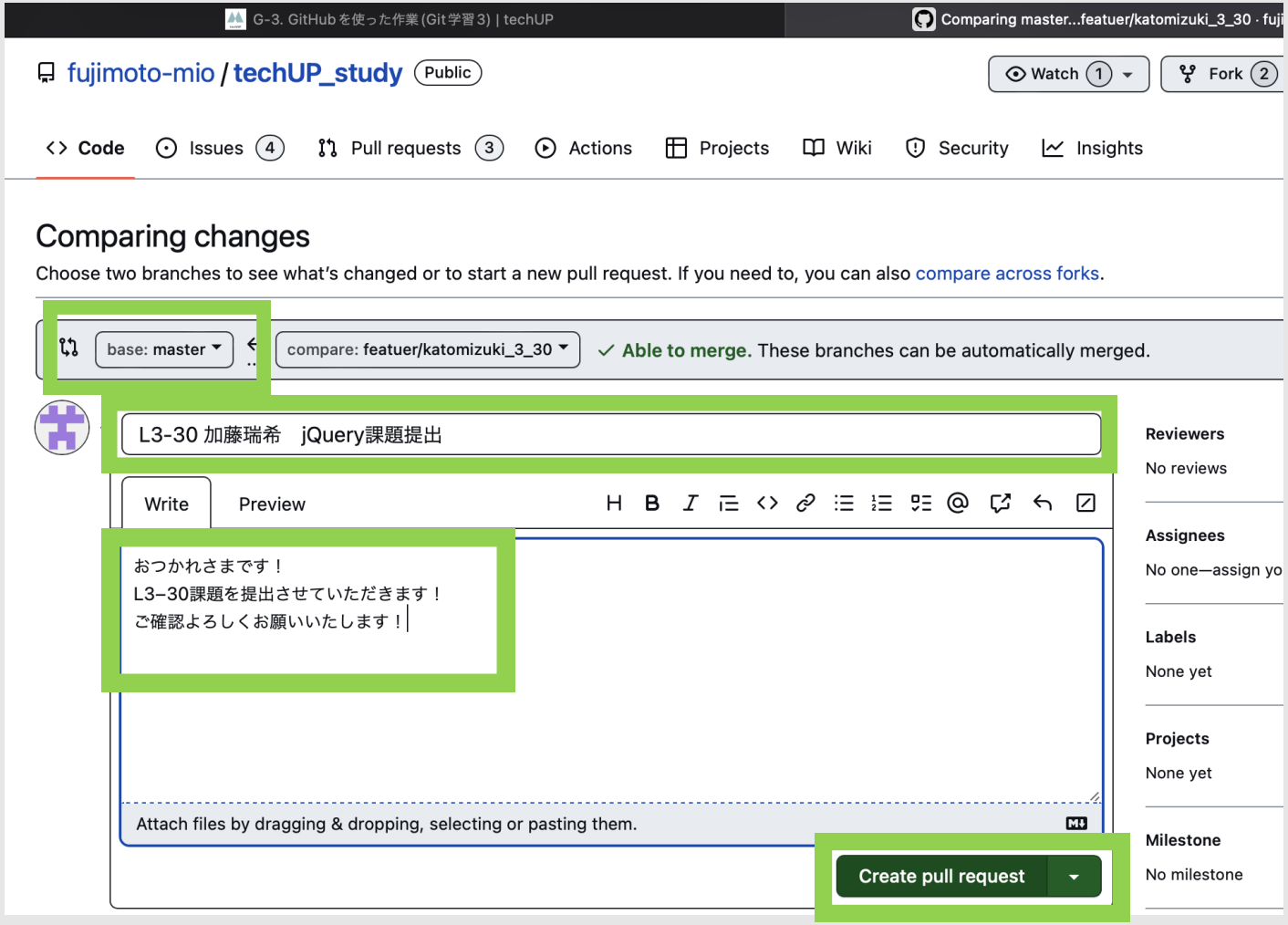
- 画面上部の緑枠
- 画面左部の赤枠

があれば、ちゃんとcommitできている

23.緑ボタン「Compare&pull recuest」を押下

Gitでの課題提出方法⑦

techUP



24.「base:main」から
　「base:master」へ変更

25.アイコン吹き出し（小枠）を
　「課題番号・名前・課題名」へ書き換え

26.大枠に「コメント」記載

27.緑ボタン「Create pull request」で
　PullRequestする

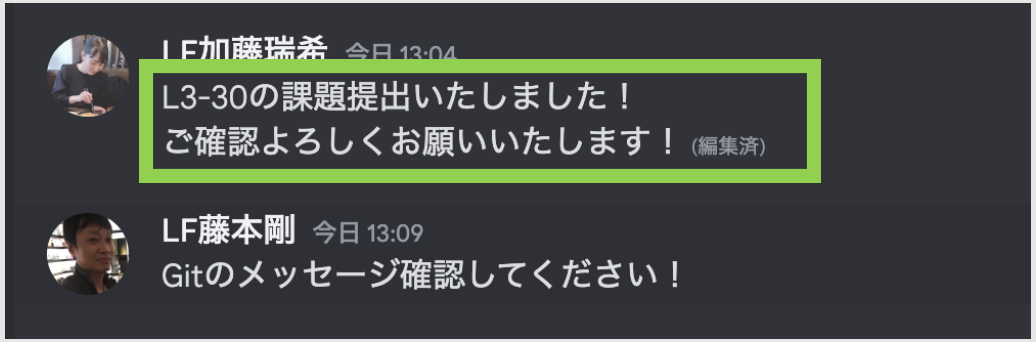
Gitでの課題提出方法⑧

techUP

28.入力内容に不備がない確認

※この時「Merge pull request」は

まだ押下しない！！！

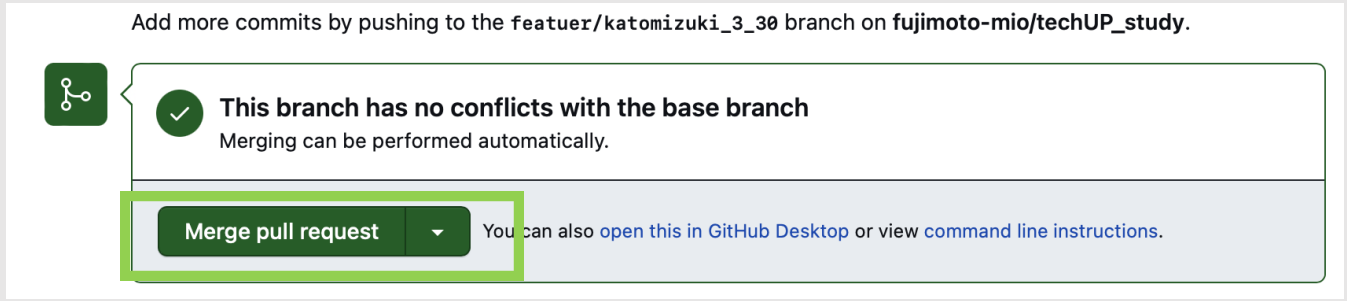


29.Discordで課題確認者へ提出した旨を報告

Gitでの課題提出方法⑨

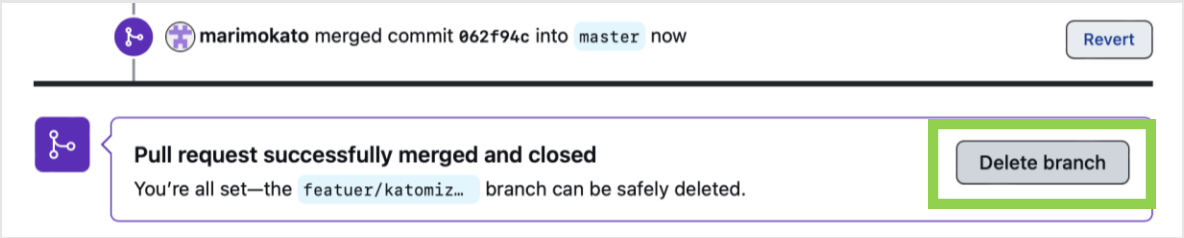
techUP

30.課題が承認されたことを確認
このとき修正が必要な場合は、課題に修正をかけて12で作成したbranchに再度pushする
補足：LGTMとは「Looks Good To Me」の英略語で、訳すと「いいと思います」という賛同の意味



31.「Merge pull request」を押下

32.「Confirm merge」を押下



33.「Delete branch」を押下し終了
（次の課題を提出するときはmain branchを.
切り直してbranchを作成する。
常に新しく作成するbranchはmainから）

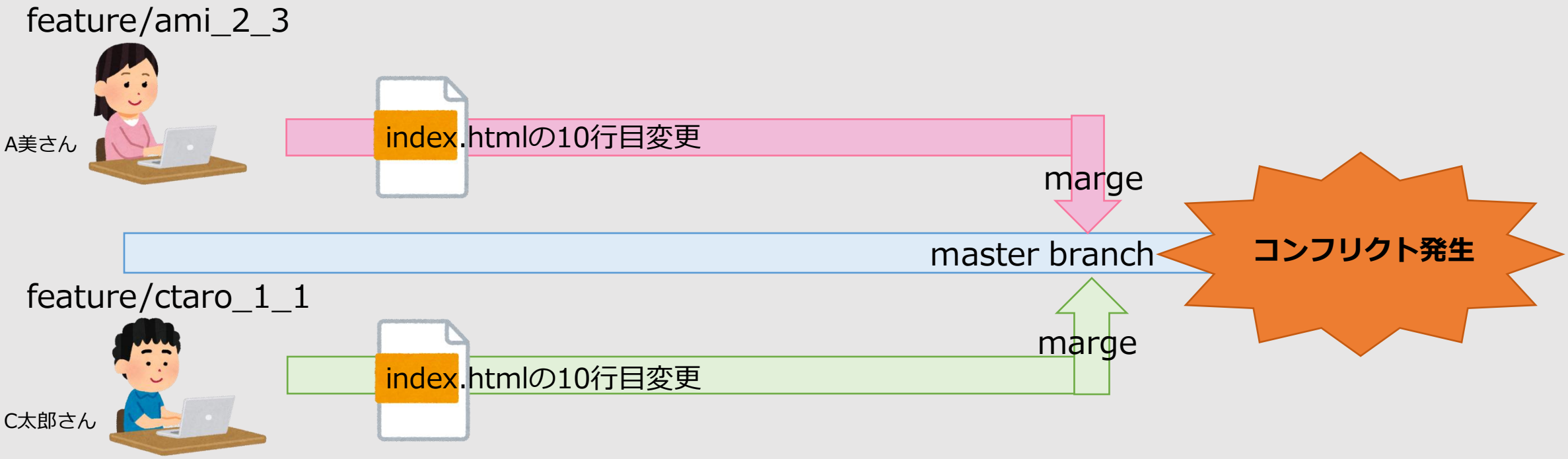
コンフリクトとは？

techUP

コンフリクトとは衝突を意味しています。
Gitではmargeを行う際に、branch内で同じファイルの同じ箇所で修正を行った時に
どの記述を優先したらいいか分からない状態のことです。

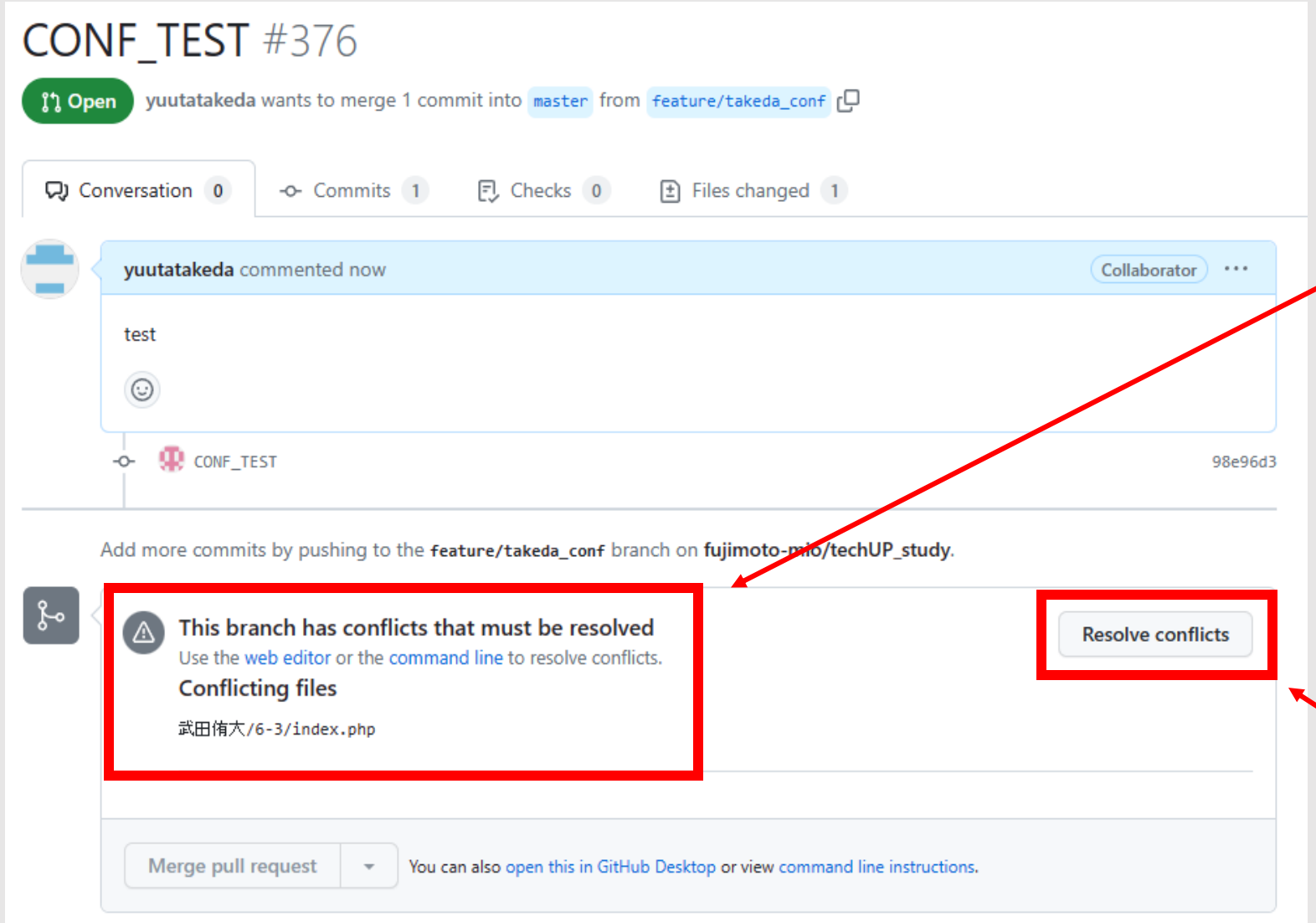
例えばこんなコンフリクトがあります…

プロジェクト内に共通のファイルがあり、そのファイルをまずA美さんが修正して、
それを知らずにC太郎さんが修正して、それがたまたま同じ箇所だったときなんかにコンフリクトする。



どうやってコンフリクトを解消するか？①

techUP



①コンフリクトが起きると、
マージプルリクエストしようとしても
This branch has conflict～
と表示されます！

②コンフリクトを解消するには、
Resolve conflictsをクリック！

どうやってコンフリクトを解消するか？②

techUP

Resolve conflictをクリックするとこの画面に移ります↓

CONF_TEST #376

Resolving conflicts between `feature/takeda_conf` and `master` and committing changes → `feature/takeda_conf`

武田侑大/6-3/index.php

```
1   <?php
2
3   $fp = fopen("member.csv", "r");
4
5   while ($line = fgetcsv($fp)){
6
7   echo $line[0] . "<br />";
8   <<<<<<< feature/takeda_conf
9   echo $line[1];
10  =======
11  echo $line[1]       . "<br />";
12  >>>>>>> master
13  echo $line[2] . "<br />";
14  }
15
16  fclose($fp);
17
```

③9行目か11行目のどちらの記述が
正しいかは、このように表示されます！
（8行目～12行目）

9行目は今いるブランチで自分が作業した変更内容です！
11行目が他のブランチで作業した変更内容です！

<<<<<<
<現在のブランチの変更内容>
======
<他のブランチの変更内容>
>>>>>>

どうやってコンフリクトを解消するか？③

techUP

④この画面で目的の記述に整えます！

⑤整えたらMark as resolvedボタンをクリック！！
GitにconflictがMerge解決したことを知らせるボタンです

どうやってコンフリクトを解消するか？④

techUP

CONF_TEST #376

Resolving conflicts between `feature/takeda_conf` and `master` and committing changes → `feature/takeda_conf`

Commit merge

1 conflicting file

武田侑大/6-3/index.php

✓ Resolved

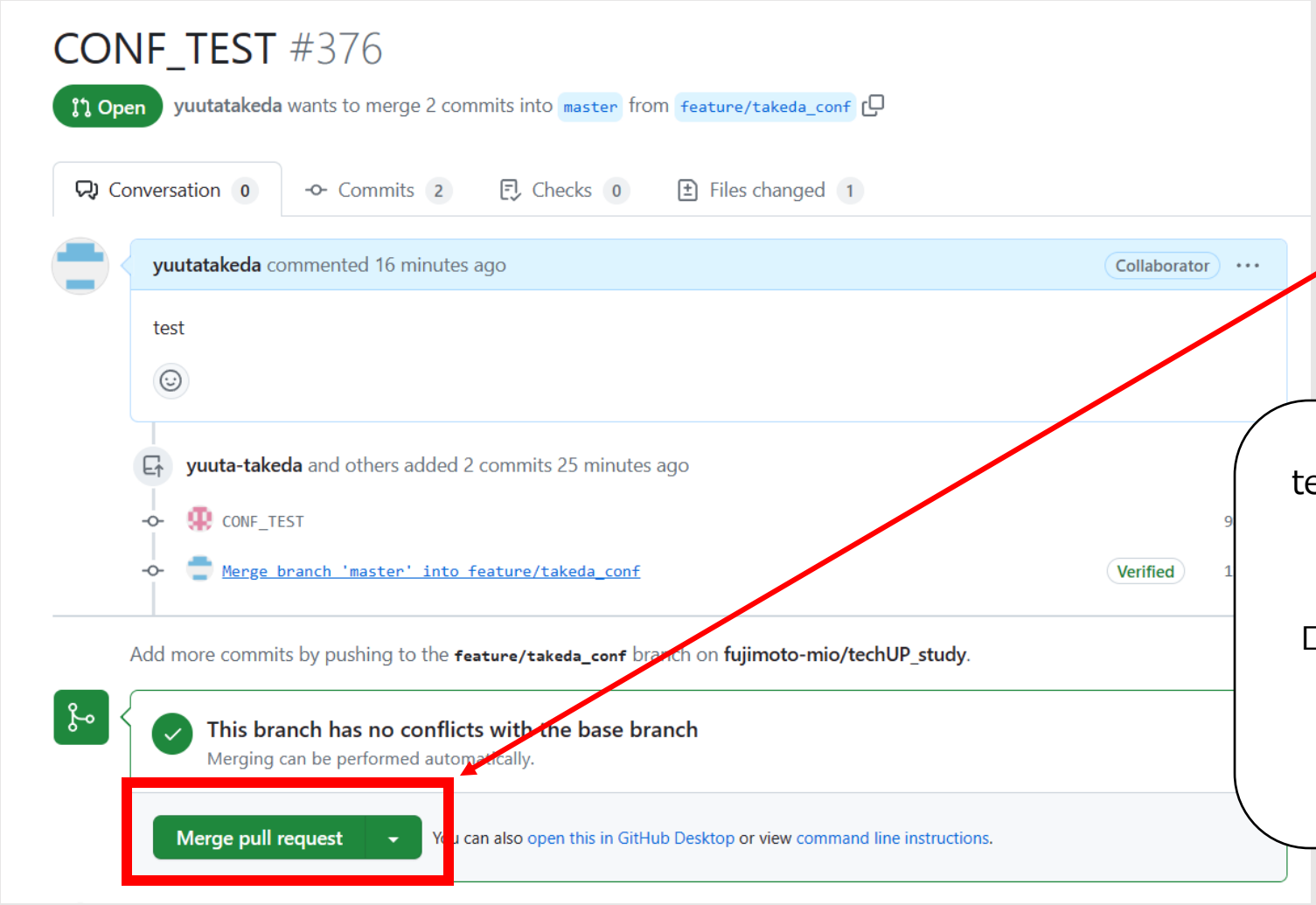
index.php
武田侑大/6-3/index.php

```
1   <?php
2
3   $fp = fopen("member.csv", "r");
4
5   while ($line = fgetcsv($fp)){
6
7   echo $line[0] . "<br />";
8   echo $line[1] . "<br />";
9   echo $line[2] . "<br />";
10  }
11
12  fclose($fp);
13
14  ?>
```

⑥✓Resolvedになれば
上にCommit mergeボタンが表示されます！
このボタンをクリック！

どうやってコンフリクトを解消するか？⑤

techUP



⑦⑥のCommit mergeボタンをクリックし
①のプルリクの画面に戻り
Merge pull requestが表示されたら
コンフリクト解消完了です！

techUPでは主にリモートでの解消方法を行いますので、
リモートでの解消方法を説明しましが、
実際、開発現場ではローカルで
コンフリクト解消を行うことが多いです！
ローカルでの解消方法も参考サイトを載せておきます
ので学習しておきましょう！！
↓↓↓↓↓↓
https://m-kenomemo.com/git-conflict/