Git & GitHub勉強会

目次

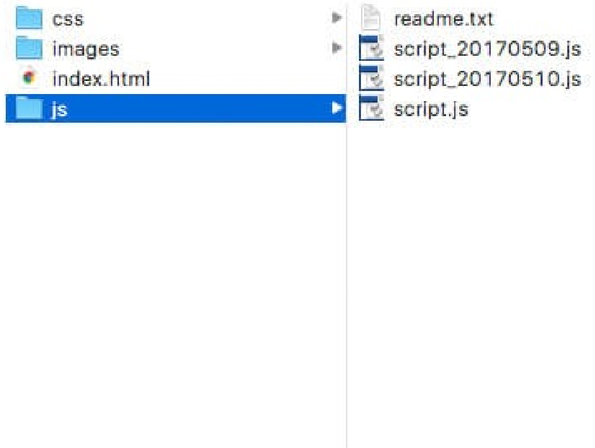
gitとは？？

ファイルの変更履歴を、実体ファイルを増やすことなく
記録管理することが出来るもの、仕組み。

他の作業者とソースコードの共有・共同管理が出来る。

GitHub・・・Gitのデータをインターネット上においておける場所（リポジトリ）を提供するサービス。

gitがない場合どうファイル管理する？



・readme.txtに「誰がいつ何を作業（変更・追記・修正など）をしたのか」をメモ。あるいはソースコードにコメントを書いていく。

・日付ごとにファイルを都度保存していく。あとでバージョンを戻したくなった時は、readme.txtに書いたメモを参照し、ファイル名を変更してバージョンを戻す。

例）
5/10のバージョンに戻したい場合

・現在のscript.jsを削除
・script_20170510.jsのファイル名をscript.jsに変更

Readme.txtに「誰がいつ何を作業（変更・追記・修正など）をしたのか」をメモ。
あるいはソースコードにコメントを書いていく。

日付ごとにファイルを都度保存していく。あとでバージョンを戻したくなった時は、
readme.txtに書いたメモを参照し、ファイル名を変更してバージョンを戻す。

面倒＆手間がかかる・・・

Gitを使うとどうなる？？

・日付ごとにファイルを保存する必要がない。

・フォルダ内に「リポジトリ」というものが作成され
「変更履歴・各種ファイル＆情報」はすべてここに保存される。

Gitを使うと「リポジトリ」と呼ばれるものが作成され、「ファイルの変更履歴・各種ファイル情報」などが全て保存されるようになっています。

リポジトリ

サーバー

・index.htmlというファイル名が被って、Aさんがあげた作業内容がBさんによって上書きされて消えてしまった・・・

index.htmlを
17:00にアップ！

Aさん

Aさんと同じ時間に
作業をしていて、
index.htmlを
18:00にアップ！

Bさん

Gitを使用する準備

Gitを使用する準備

https://gitforwindows.org/
https://sourceforge.net/projects/git-osx-installer/files/

Gitをインストール。ターミナルでgit —versionと打って何かでればOK

ターミナル系ソフト or Source TreeやGit Krakenなどの
GUIソフトをインストールしてGit操作します。

別途指示された使用ファイルをダウンロードし、cdコマンドでダウンロードしたデータフォルダの
中に移動しておきましょう！

Gitを使用する準備

git init

git initコマンドを実行することによって、指定したフォルダの中にgitの仕組みが導入されます。
別途gitの隠しフォルダが表示されているのが確認できればOKです！！！

Gitを使用する上で知っておきたい用語

一人作業の範囲でgitを使いこなしてみる

Gitで変更記録を保存する流れ

ファイル

Git add ファイル名

ステージ

Git commit -m “メモ”

リポジトリ

Git addをすることで、記録に残すための対象ファイルを指定します（git add . だと全ファイル指定）

Git commit で「何のためにそのファイルを保存するのか。作業の目的」をコメントで残します。

Commitの単位と記法例

Git commit -m “[作業タイプ]何の課題に対して、何をしたか”

何の課題に対して？

その作業内容を行なった理由となる出典元を書く。例えば、何かしらのタスク管理ツールを用いて全体の進行管理を行なっている場合、各タスクに番号がついているはずなので、その番号を書く。

作業タイプ例

[add]ファイル追加

[modify]機能追加・修正

[fix]ただの脱字とか軽いミスの修正

[bug fix]バグの修正

Gitの履歴確認

記録履歴


```
kosugatatsuya@tk-mn techpit-write % git log
commit 01d9642c2c3accfb2715f74aef35dd4533b557b6 (HEAD -> develop, origin/develop)
Author: Tatsuya Kosuge <caste
Date:   Fri Jul 2 12:19:53 20

    [add]4章のパート3終了まで

commit 9340da653aaa7292305ba621ea1172031c23f345
Merge: c8a10b1 e530be1
Author: Tatsuya Kosuge <castero1219@gmail.com>
Date:   Fri Jul 2 11:44:08 2021 +0900

    [add]4章のパート３まで

commit c8a10b1534c30df0832eed973ad899f822b57c74
Author: Tatsuya Kosuge <castero1219@gmail.com>
Date:   Fri Jul 2 08:36:45 2021 +0900

    [modify]画像追加

commit e530be172e6d4815e21e5eeeeefe9c2e74c655c7
Author: castero1219 <castero1219@gmail.com>
Date:   Fri Jul 2 08:34:32 2021 +0900

    [modify]3章を一旦通しで
```

各記録についたID番号

Git log・・・・変更履歴を表示する

HEAD・・・現在いる位置

Author・・・コミット者

変更記録の取り消し方法

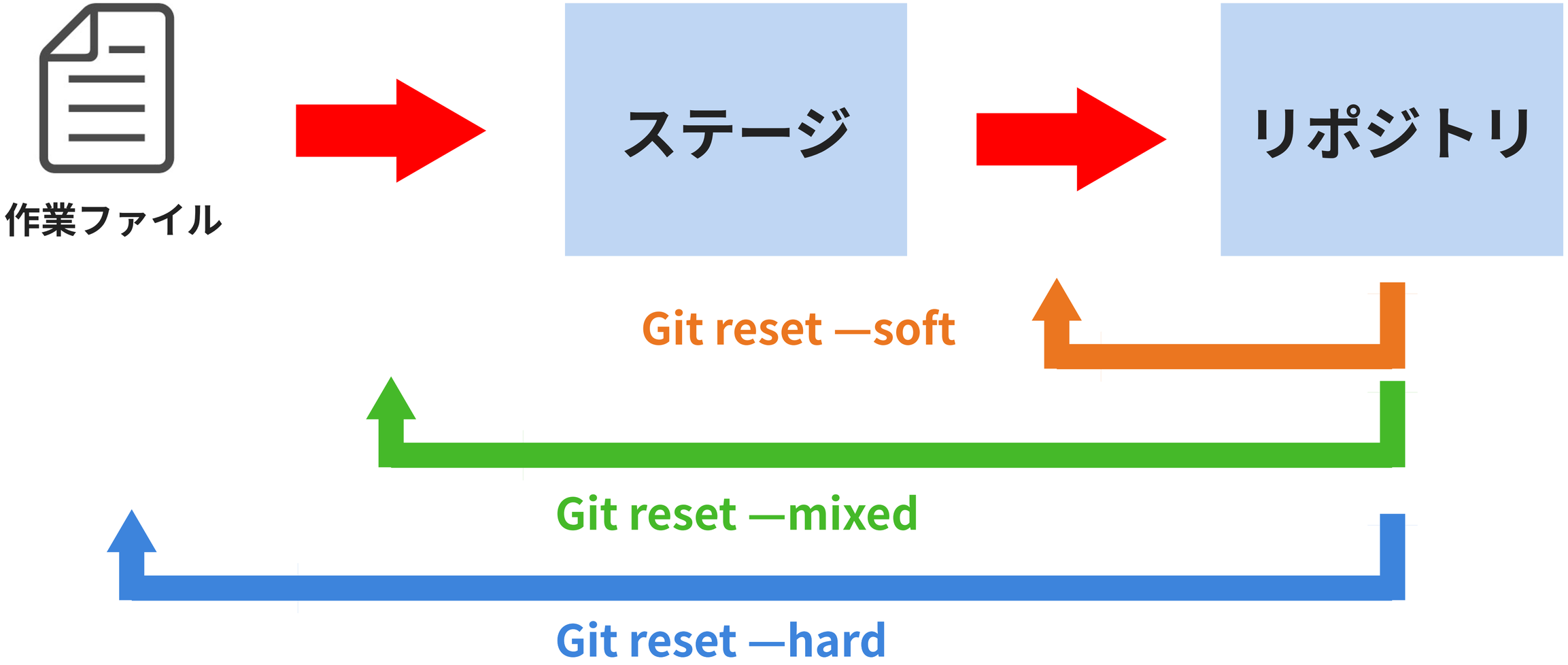
Git reset —hard コミットについたID番号

Git reset --soft ——— **Commitのみを取り消す**

Git reset --mixed ——— **Commitとステージファイルを取り消す**

Git reset --hard ——— **Commitとステージファイルを取り消し、作業ファイルの状態も巻き戻す**

変更記録の取り消し方法



直前のコミットを修正する

git commit --amend ——— **コミット内容の修正**

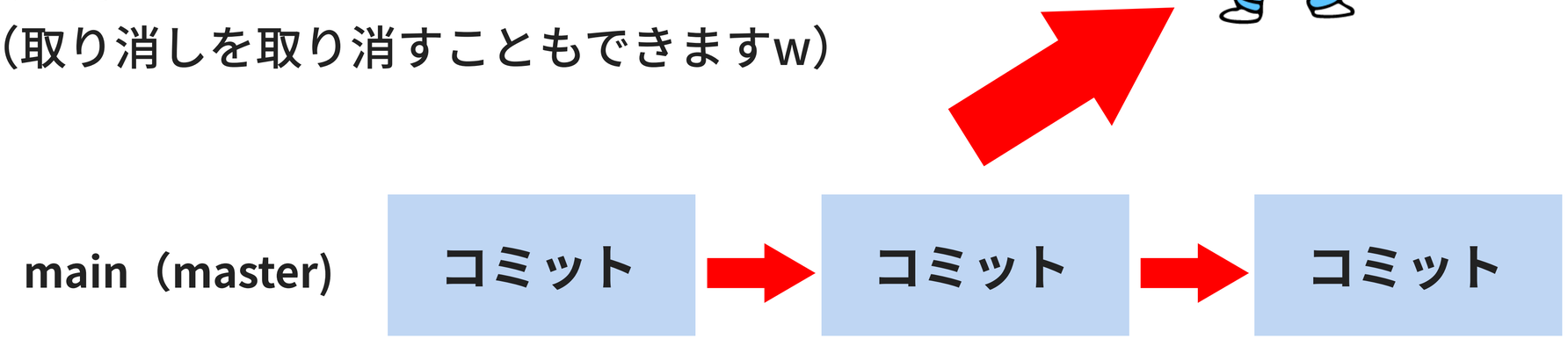
Revert

特定のコミットを打ち消す

Git revert コミットID

Revertを使うと、特定のコミット内容を
取り消すことができます！
（取り消しを取り消すこともできますw）

これだけ取り消したい時に
使えるのがRevert!!!



.gitignore

```
!wp-content/

wp-content/*
!wp-content/mu-plugins/
!wp-content/plugins/
!wp-content/themes/

wp-content/plugins/hello.php

wp-content/themes/twenty*/
```

node_modulesのように、そもそもgitで管理すべきではないデータというのも存在します。
こういったものは**.gitignoreファイルを作成**し、そこにgit管理したくないものを記載していくことで
Gitの管理下から外すことができます。

Diff

特定のブランチやコミットとの差分を確認

Git diff [変更前]..[変更後]

git diff 1ccc226..cffb553
git diff master..develop

何がバグの原因なのかわからないなぁ・・・うまくいくはずなのにいかないなぁというときに原因を探るために使用するのがdiffです！

CommitについたID番号だけでなく、ブランチ名を使えばブランチごとに比較もできます。

複数人で作業する時に必要なgit flow

リモートリポジトリとGit push

Git push **origin** **master**

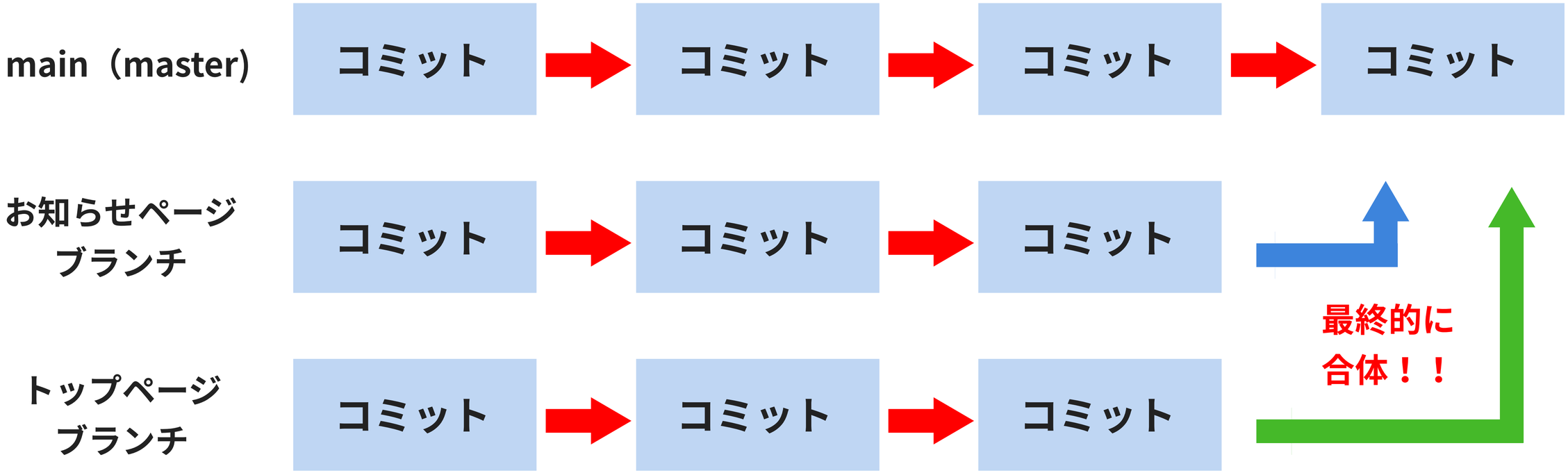
赤字はブランチ名（後述）
青字はリモートリポジトリの場所の別名

リモート
リポジトリ

Gitは複数人で作業する時にも効果を発揮します！みんなで作業した内容を集約してまとめる場所として
GitHubのような「リモートリポジトリ」サービスが必要になります

自分の作業内容をリモートリポジトリに登録することを**push**と言います。

Branch



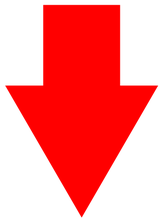
無作為にpushするだけだと、作業内容が衝突する可能性もありますし、場合によっては他人の作業内容を消してしまうことも考えられます。
そのため、「そもそも作業目的ごとに作業場所を変えた方がわかりやすい」という発想から生まれたのが**ブランチ（branch）**になります！

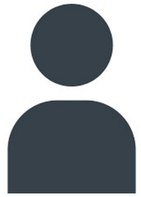
実践！ブランチ運用

今回は、実際に起こり得そうな流れを想定して、ブランチ運用の練習をします！

こすげ

よし、みんなプロジェクトを始めよう！
雛形データ作ったから、これを使って作業開始して！！！

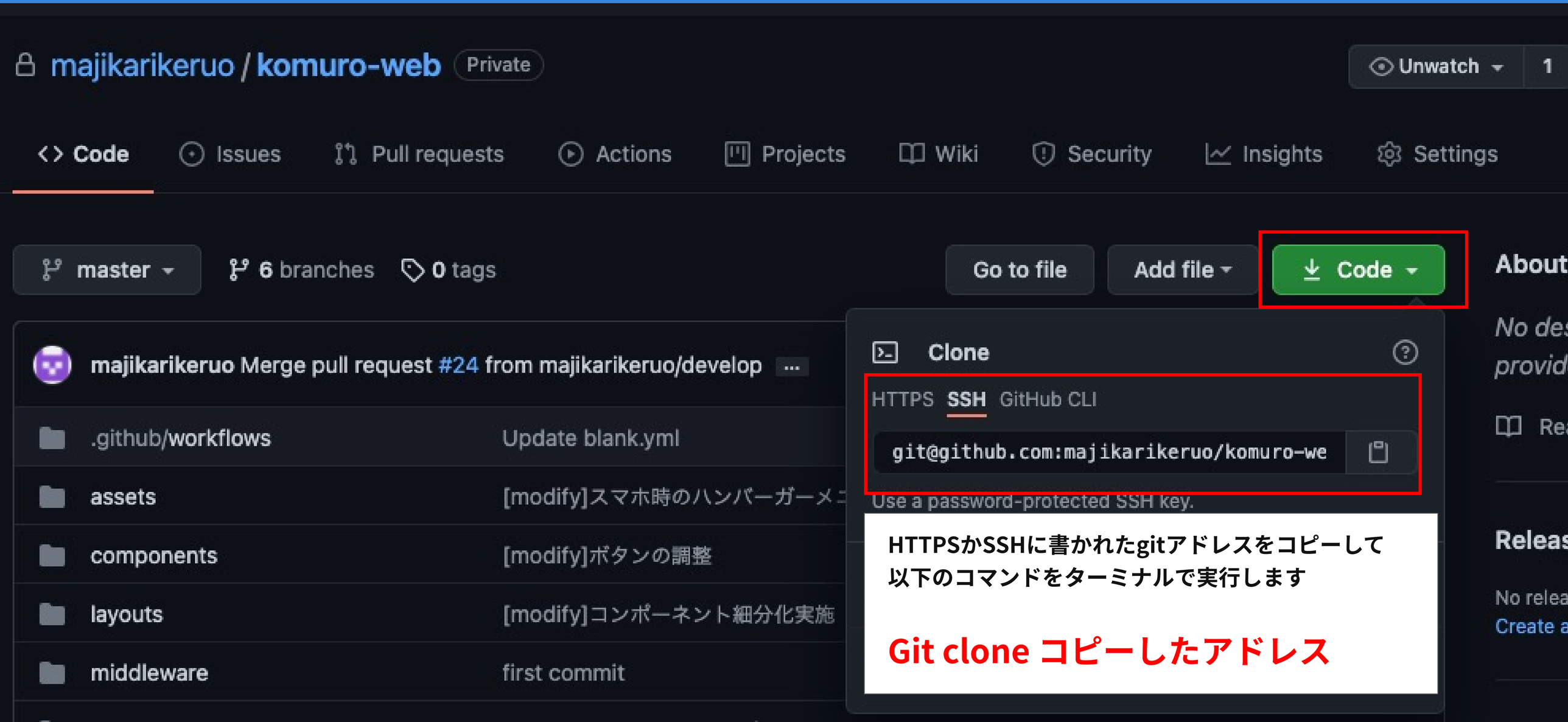




共同作業者たち

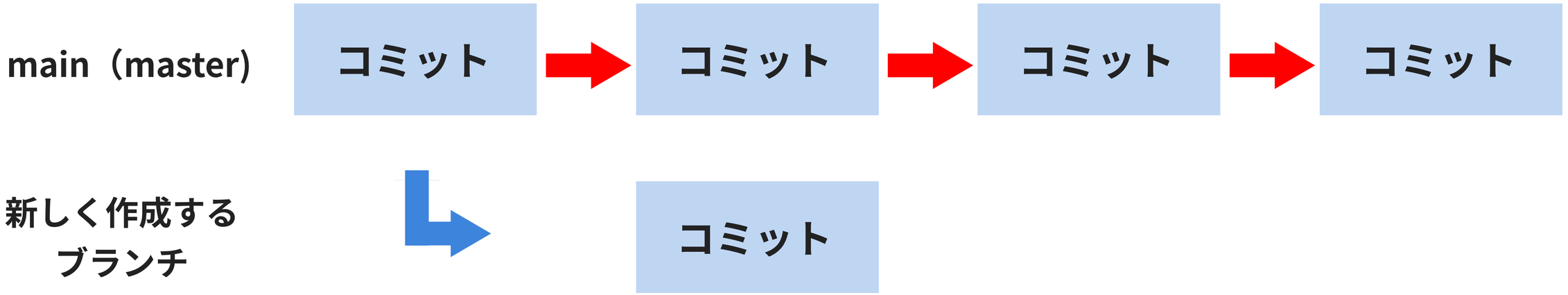
わかりました！！！（そして作業がスタートする）

Git clone


majikarikeruo / komuro-web
Private
Unwatch
1
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
master
6 branches
0 tags
Go to file
Add file
Code
majikarikeruo Merge pull request #24 from majikarikeruo/develop
.github/workflows
Update blank.yml
assets
[modify]スマホ時のハンバーガーメ
components
[modify]ボタンの調整
layouts
[modify]コンポーネント細分化実施
middleware
first commit
Clone
HTTPS
SSH
GitHub CLI
git@github.com:majikarikeruo/komuro-we
Use a password-protected SSH key.
HTTPSかSSHに書かれたgitアドレスをコピーして
以下のコマンドをターミナルで実行します
Git clone コピーしたアドレス
About
No des
provid
Releas
No relea
Create a

Checkout

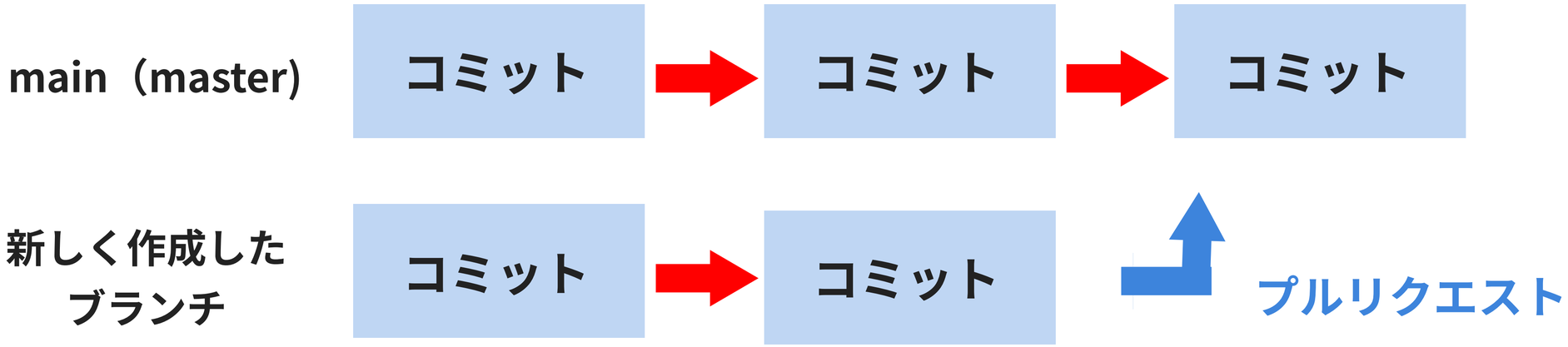
Git checkout -b ブランチ名 ―――― 新しいブランチを作成



Cloneすると、main (master)と呼ばれるブランチのみが（原則）反映されます。ここから自分で作業する用のブランチを作成するために、**checkout -b ブランチ名** コマンドを実行します。

ちなみに、**checkout ブランチ名**だけだと、既存のブランチへの切り替えを行うという意味になります。

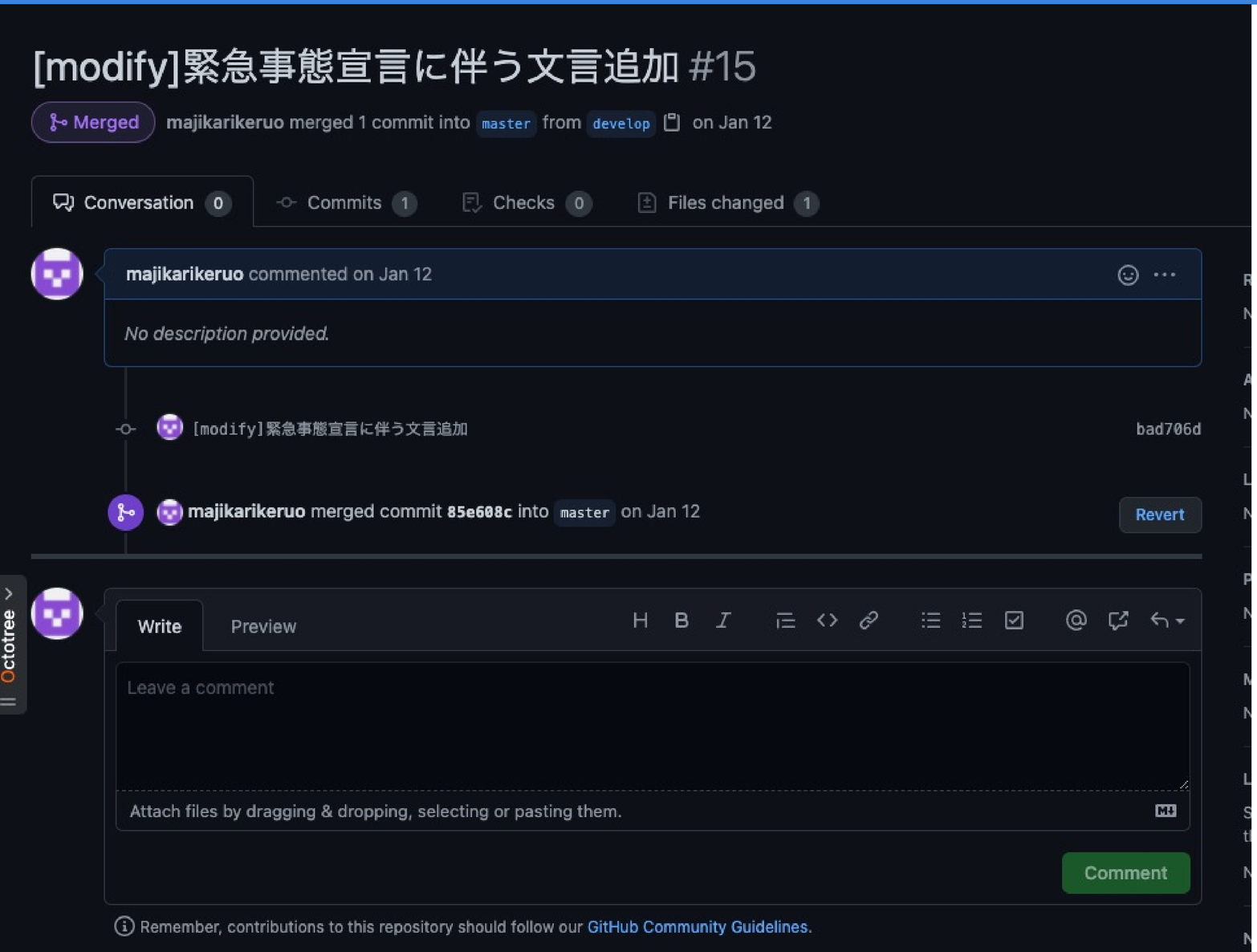
プルリクエスト



作業者は、課せられたミッションが終わったら、作業内容をチェックしてもらい本体ブランチ（main）にデータを取り込んでいいかの確認を依頼します。これをプルリクエストと言います。

（リモートリポジトリ側で行う作業です）

プルリクエスト

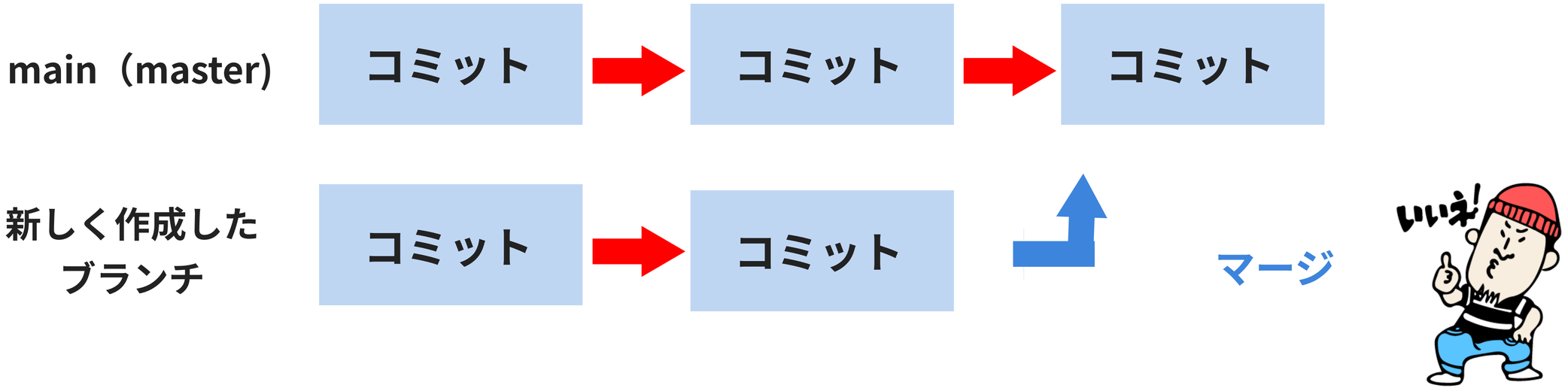


プルリクエストでは

- **どこからどこへのブランチに**
- **具体的な作業内容の報告**

を記載した上でリクエストを出します。

マージ



管理者はプルリクエストの内容を見て、最終的にOKを出せる状態になったら、その内容を統合します。
この統合作業を「**マージ**」と言います。このマージはリモートリポジトリ管理者がおこないます。

実際のブランチの分け方例

Main

= 本番公開データ

Develop

各作業者のデータを
統合して最終確認

Feature/●●

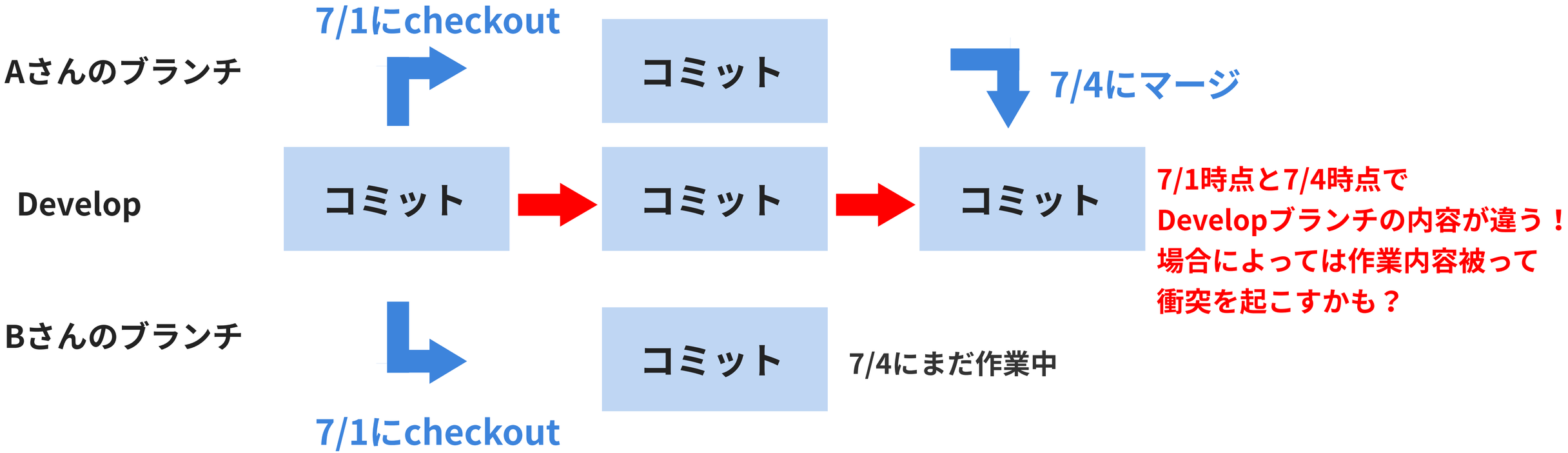
各作業者用・作業フェーズごとに
作成されたブランチ

Feature/●●の内容をdevelopに一旦集約。問題なければ、都度都度mainブランチに統合します。

複数人で作業する時に必要なgit flow

一人作業の範囲でgitを使いこなしてみる

Pullとマージその2



制作をしていると、どうしても人によって作業進行の早い遅いというのが出てくると思います。
この際に、場合によっては作業者もマージ作業をする必要が出てくる可能性があります！！

この時の作業内容についても、簡単に見ておきましょう。

Pullとマージその2

リモートリポジトリの内容を引っ張る

Git pull origin ●●●

特定ブランチの内容を取り込む

Git merge ●●●

よくありそうな作業

checkoutしてdevelopに切り替え→git pull origin develop → もう1回checkoutして自分の作業ブランチに切り替え→ git merge develop

コンフリクト

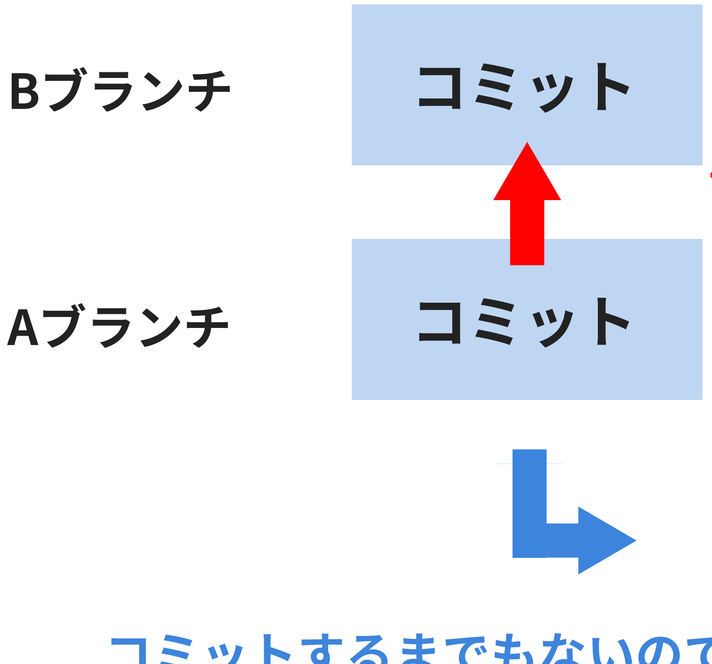
```
<<<<<<< HEAD
# 作業ブランチでの変更内容
 . . .
=======
# develop(マージしたブランチ)での変更内容
 . . .
>>>>>>> develop
```

不要な方を手動で消します！！！！！

Mergeした結果作業内容の衝突が発生した場合は、衝突を解消させる必要があります。この作業のことを**Conflictの解消**といい、これは手作業でおこないます。

※原則リモートリポジトリの内容を正とするので、場合によりこんなことが起きるわけです！

Stash



一旦Aブランチの作業置いておいて
Bブランチに切り替え

作業内容を一時的に避難させる

Git stash

Stash

コミットするまでもないので
一時的に避難！

作業中、別に緊急で対応しなければならないことがあるけど中途半端な状態だからコミットしたくないというときに、stashという緊急避難場所に作業内容を一時保存することができます。

Stashの戻し方

Stash 一覧を確認

Git stash list

stash@{0}: WIP on ブランチ名: xxxx
stash@{1}: WIP on ブランチ名: xxxx

退避した作業を戻す

git stash apply stash@{0}

git stash pop stash@{0}

数字の部分は「何番目の退避に戻す」という意味
Popの場合は退避から戻した瞬間にstashからも消す

Cherry-pick

別ブランチから特定コミットだけを取り込む

Git cherry-pick コミットID

複数ブランチで運用していると、「別ブランチのこの機能があると更に自分の作業捗るんだよな」なんていうことが出てきます。

そんな時に使用できるのが「cherry-pick」です！！！

GitHubでこんなこともできる

GitHub Pages

https://docs.github.com/ja/pages/getting-started-with-github-pages/about-github-pages

GitHub自体が持ってるサーバ機能。PHPとか絡んでなければ使用可能。小規模サイトならこれで十分。。。

GitHub Actions

https://arrown-blog.com/githubactions-ftp-deploy/

FTPソフトを使わずに、mainブランチへの統合が行われた段階で
自動的にGitHub上のコードの内容をFTPサーバにアップロード可能！