

GitHub

チーム開発編

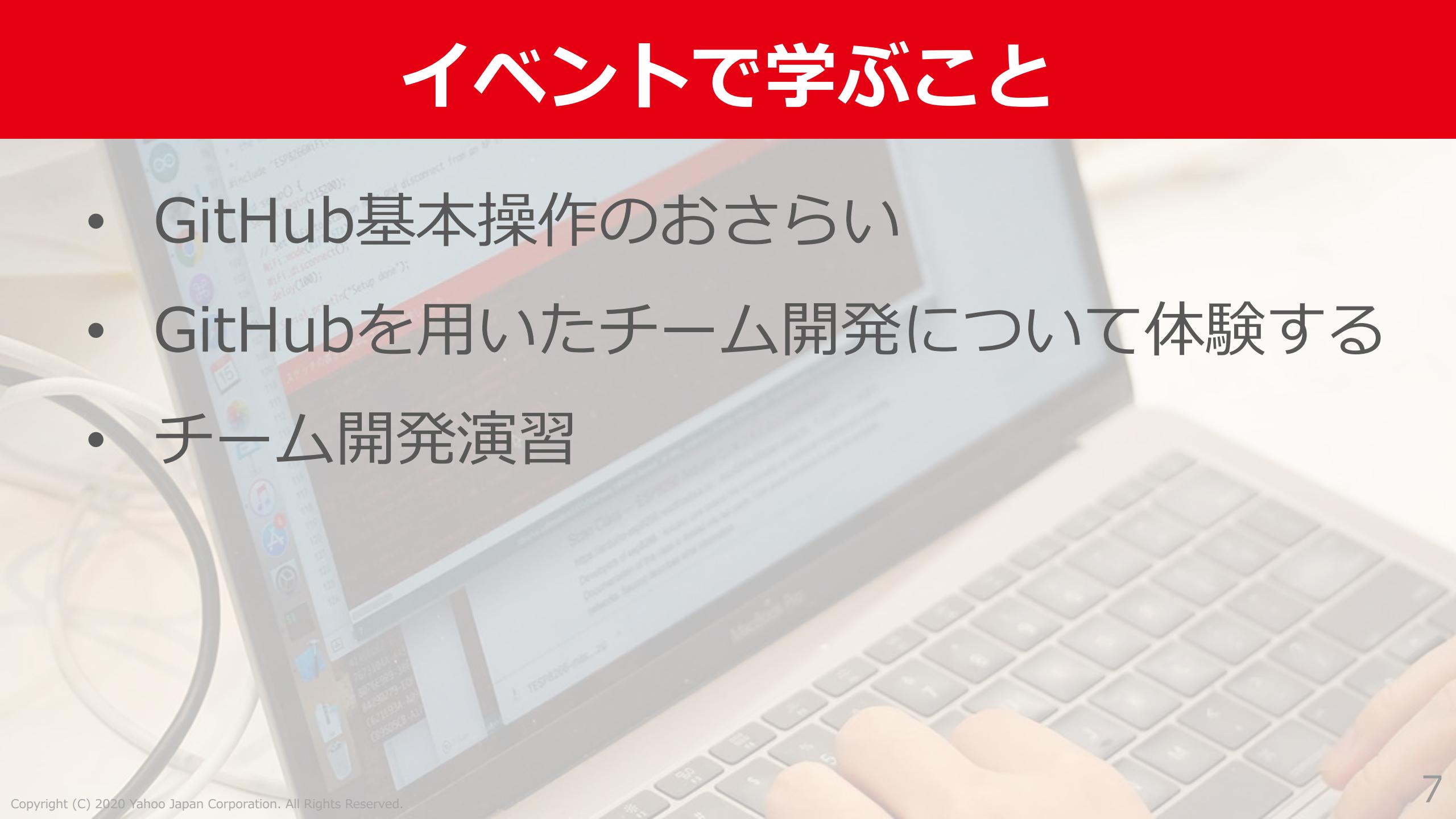
2021年7月14日

Hack U Projects



イベントで学ぶこと

- GitHub基本操作のおさらい
- GitHubを用いたチーム開発について体験する
- チーム開発演習



このイベントのゴール

- ・ コード管理の大切さを体験してもらう
- ・ チームで問題を解決できるようになる
- ・ チーム開発を円滑に行えるようになる
- ・ Hack U の開発体制についてチームで共有する

Hack U に向けて

GitHub はコード管理をアシストするツールです

-> 開発速度を求める際には使わない判断も重要

- コードレビュー
- アイディア出しや議事録をまとめる
- 完成したコードの共有

用途を限定して使うことも考えてみましょう

イベントの進め方

- 事前準備の確認
- 講義
 - Git/GitHubの基本機能おさらい
 - add, commit, push, ブランチ, PR, pull
 - チーム開発の進め方
 - Conflict について
- チーム演習 (チーム分かれて作業)
- まとめ

注意

- Git/GitHub 下記機能についての知識が必要です
 - **add/commit/push**
 - **branch/Pull Request/pull**

事前準備等で

また、複数人で作業することを前提の課題となっています
一人で行う場合は複数のブランチを使います

問題が起きたときは

- まずはチームメンバー間で助けあいましょう
 - チームで解決できないときは **#質問部屋** に下記内容を記載して質問してください。
 1. 状況
 2. 問題内容(エラー文)
 3. 実現したいこと

※サポーターの人数に限りがあるため、回答に時間がかかる場合があります

演習に向けて

- ・ コマンド叩けることよりもGitHubの使い方を知ってもらうことが目的です
 - ・ 詰まった場合は、進んでいるメンバーに情報を共有して貰いましょう
 - ・ Zoomの画面共有やSlackのチャットで共有など
 - ・ 演習ではメンバーに役割を割り振って作業を行います
- 実際の開発をイメージして進めましょう

事前準備は
お済みですか？

YAHOO!
JAPAN

事前準備（個人）

ver_VSCodeのセットアップ資料を参考に
準備項目の確認

- 自身で作成した**github_ws**リポジトリにサンプルコードが上がっていればOK
- GitHubのおさらいで使用します

事前準備（チーム）

セットアップ資料を参考に準備項目の確認

- ・ チーム共通のリポジトリの作成
- ・ リポジトリにメンバーをアサイン
- ・ **team_hands-on.zip** の内容を push
- ・ メンバー各自でローカルリポジトリを作成

チーム開発でコード管理どうしますか？



このような経験をしていませんか？

よし、開発の続きをやるぞ！



index
.html



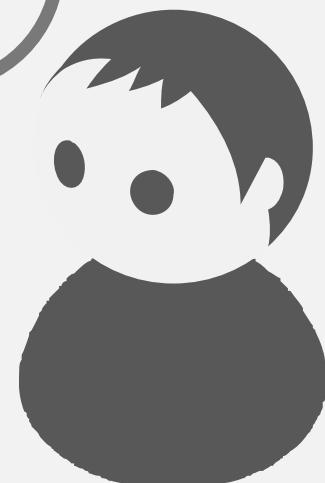
index(1)
.html



最新版
index.html

あれ？ 最新版ファイルどれ？？

私が持ってるのがたぶんそー
後で送るね



このような経験をしていませんか？



こここの機能開発しておいたよ。
ソースコードはメールで添付して送るから取り込んで
ね。

ありがとう！早速コピペして・・・
あ！自分が作ってた部分が上書きされちゃった



このような経験をしていませんか？



アカウント登録しておいた上

Git / GitHub を使って
解決しましょう！

あります。

あ！自分が作ってた部分が上書きされちゃった

んでね。



Gitとは



バージョン管理とは

- 代表的なバージョン管理システム
 - Git
 - Subversion
 - etc
- **ファイルの変更を記録**しておくシステム
 - 誰が
 - いつ
 - どのファイルを
 - どう変更したか

バージョン管理とは

ファイルの変更を記録する場所
→ **リポジトリ**



APIサーバ
リポジトリ



Androidアプリ
リポジトリ



iPhoneアプリ
リポジトリ

バージョン管理とは

リポジトリ内のバージョン管理



バージョン管理とは

過去の状態との差分がわかる



バージョン管理とは

過去の状態に戻せる

過去の状態を最新として扱う



バージョン管理とは

- **リモートリポジトリ :**

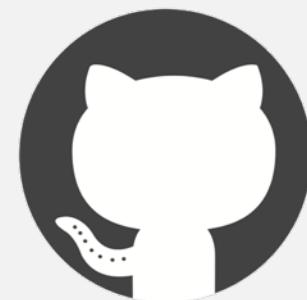
色々な人が共有できるようにしたリポジトリ
GitHub上にあるものはリモートリポジトリ

- **ローカルリポジトリ :**

自分のPC上にだけ存在するリポジトリ



ローカルリポジトリ



リモートリポジトリ

GitとGitHub

Git	バージョン管理の方式
GitHub	Gitでバージョン管理をしやすくする Webサービス ブラウザから変更履歴やコードを共有するのが簡単にできる

Gitを使うには

- コマンドラインでGitコマンドを使う
- GUIツールを使う
- **Visual Studio Code**でGitコマンドを使う
(以後、VSCodeと表現します)

Gitを使うには

- コマンドラインでGitコマンドを使う
- GUIツールを使う
- **Visual Studio Code**でGitコマンドを使う
(以後、VSCodeと表現します)

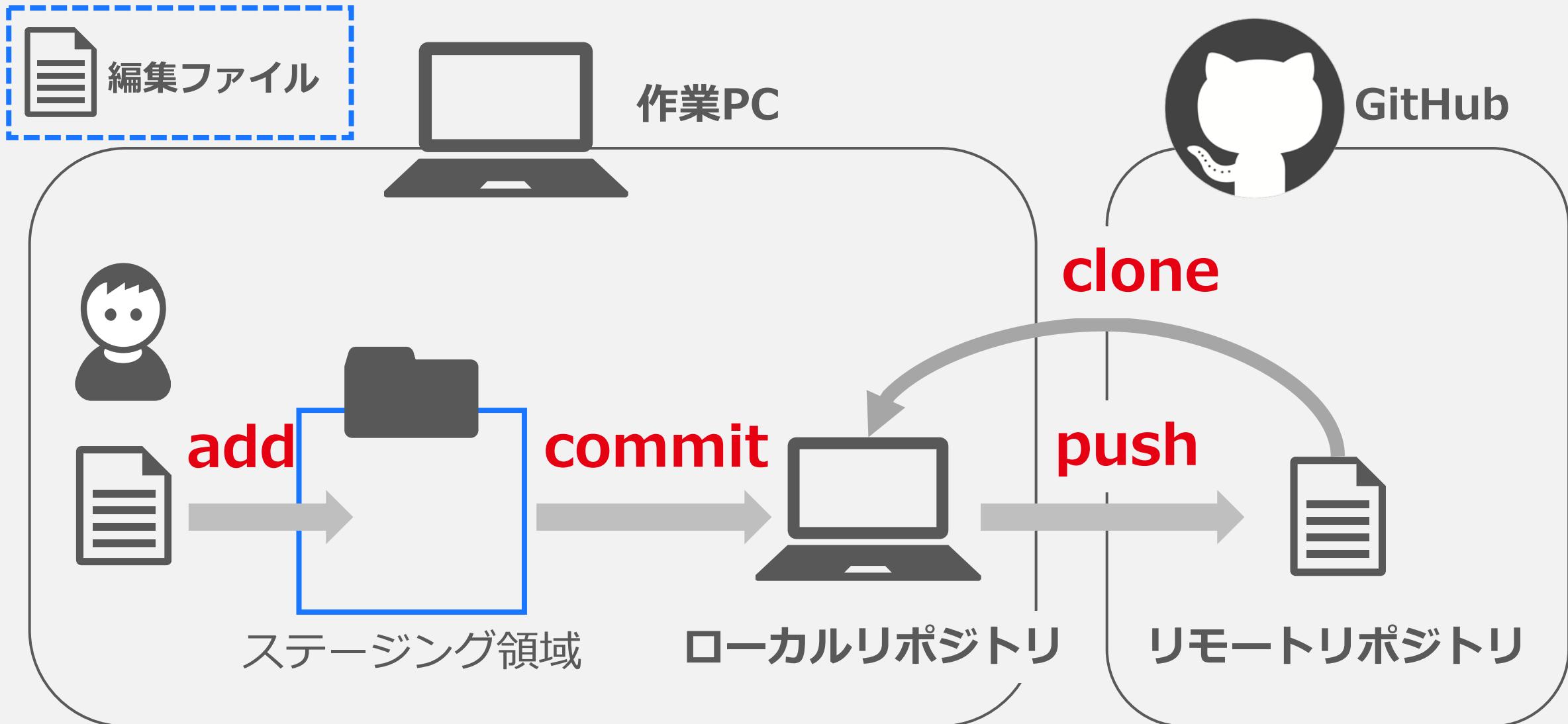
今回は VSCode で
Gitコマンド を使います

Step.1

GitHubの基本操作



GitHubの基本操作

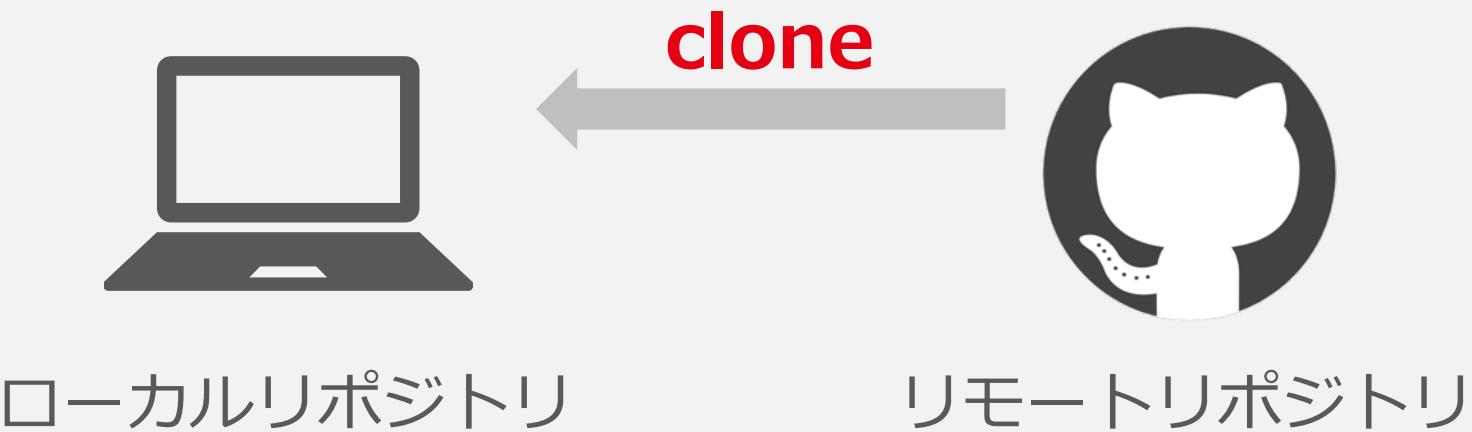


GitHubから自分のPCに持ってくる

- **clone** :

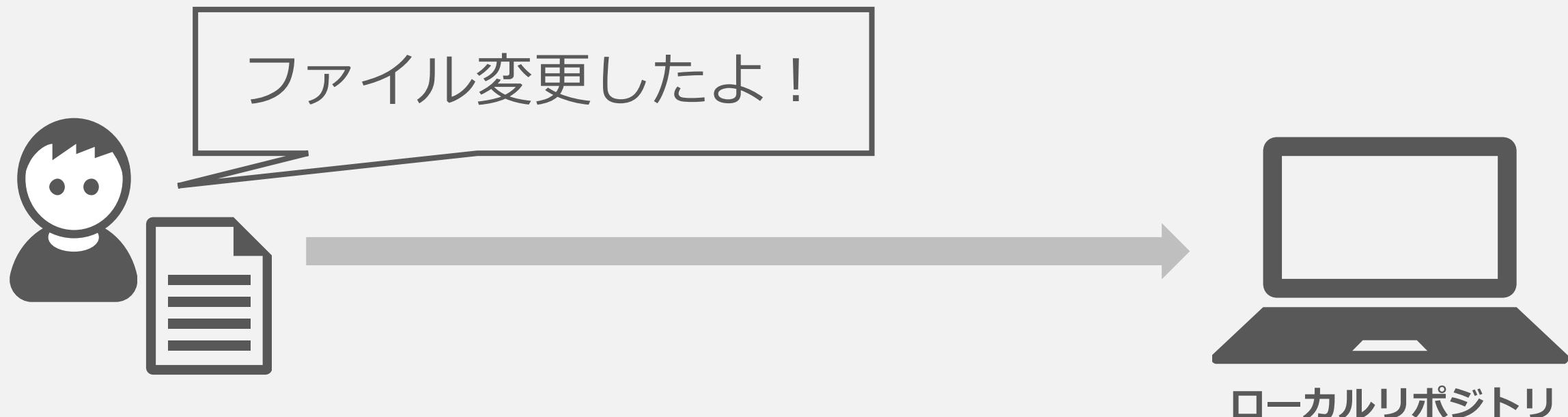
リモートリポジトリを自分のローカルリポジトリ
にコピーする

※ 初回のみの操作

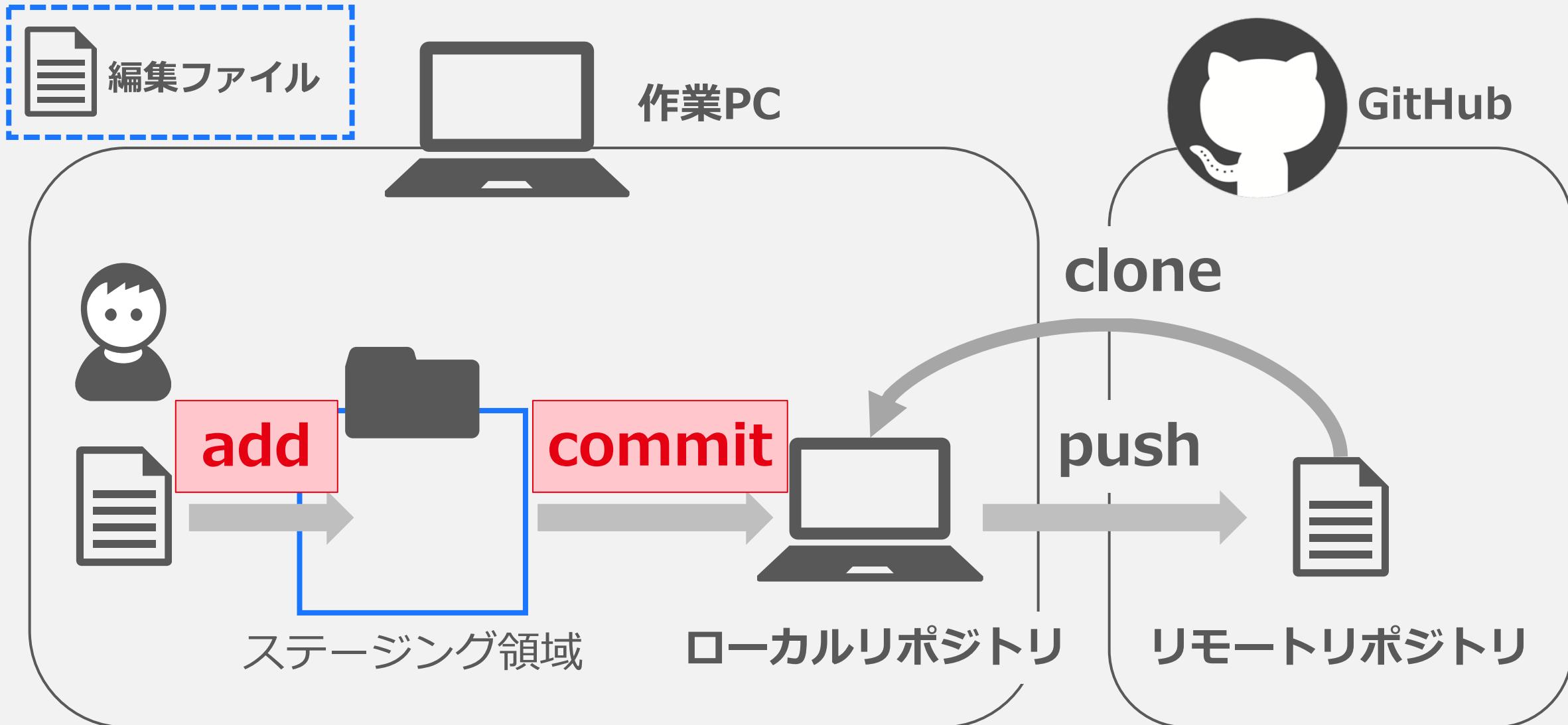


ファイルを修正してGitで扱う

ファイルの変更をローカルリポジトリに
登録するには**2つのステップ**が必要になります



GitHub の基本操作



ファイルを修正してGitで扱う

Step.1

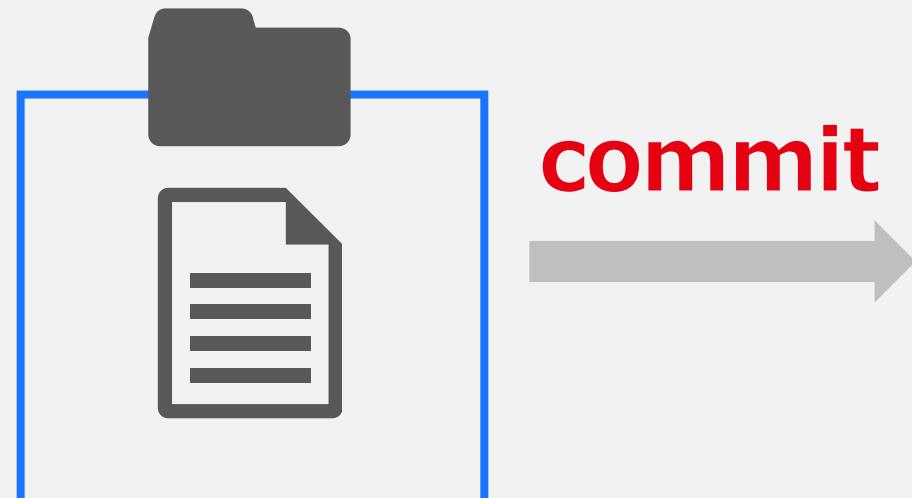
どのファイルの変更をしたかをステージング領域に
一度登録する → **add** という



ファイルを修正してGitで扱う

Step.2

ステージング領域に登録した変更内容をローカルリポジトリに登録する → **commit**という



ステージング領域（インデックス）



ローカルリポジトリ

ファイルを修正してGitで扱う

Step.2の注意点

commitした時には必ず**どういう変更をしたかを
コメント**で残す必要がある

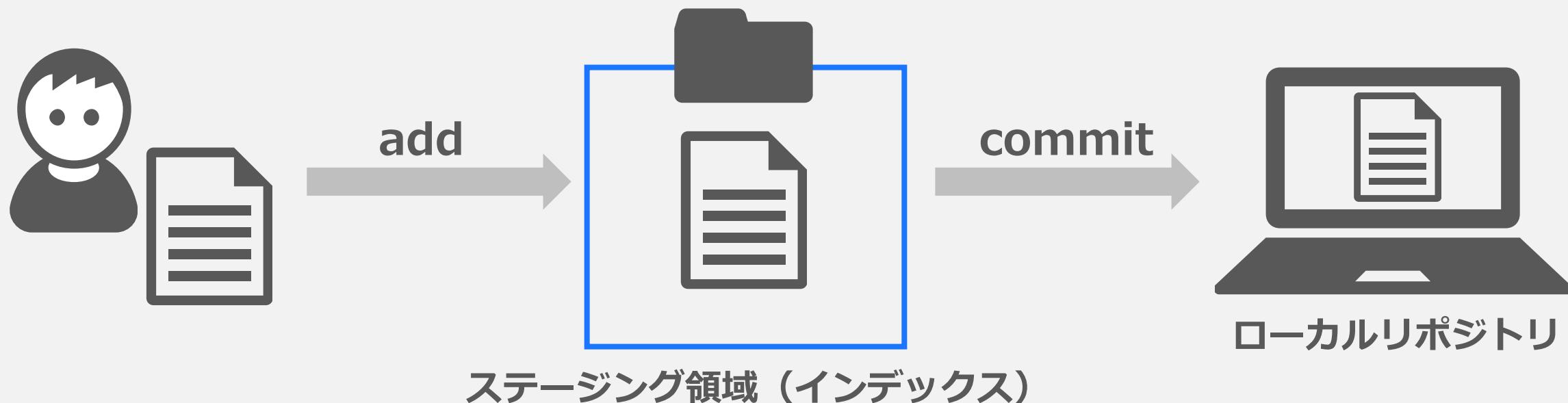


ステージング領域（インデックス）

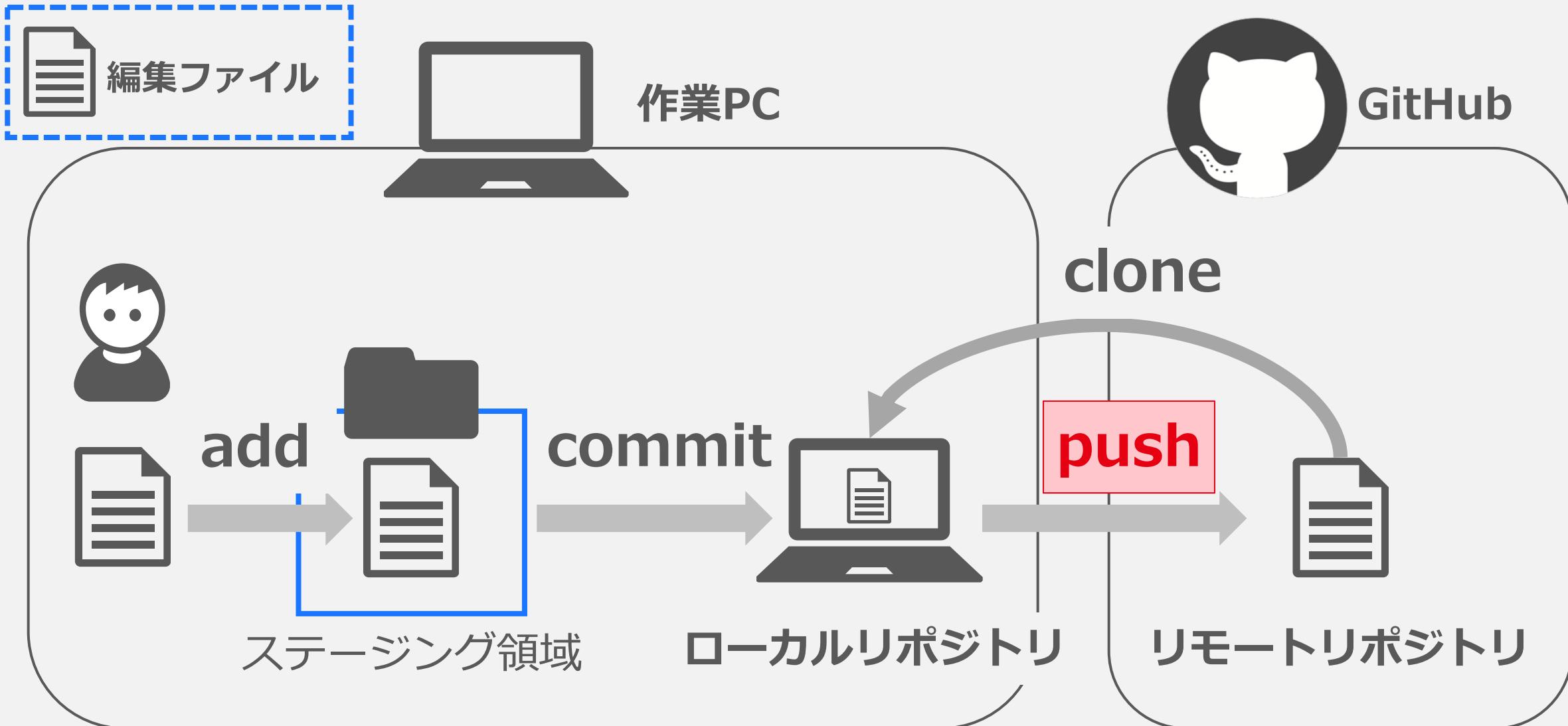


ファイルを修正してGitで扱う

- ファイルの変更は**add**して**commit**
- commitした時は**必ずコメントを残す**



GitHub の基本操作



作業PCからGitHubに反映する

push :

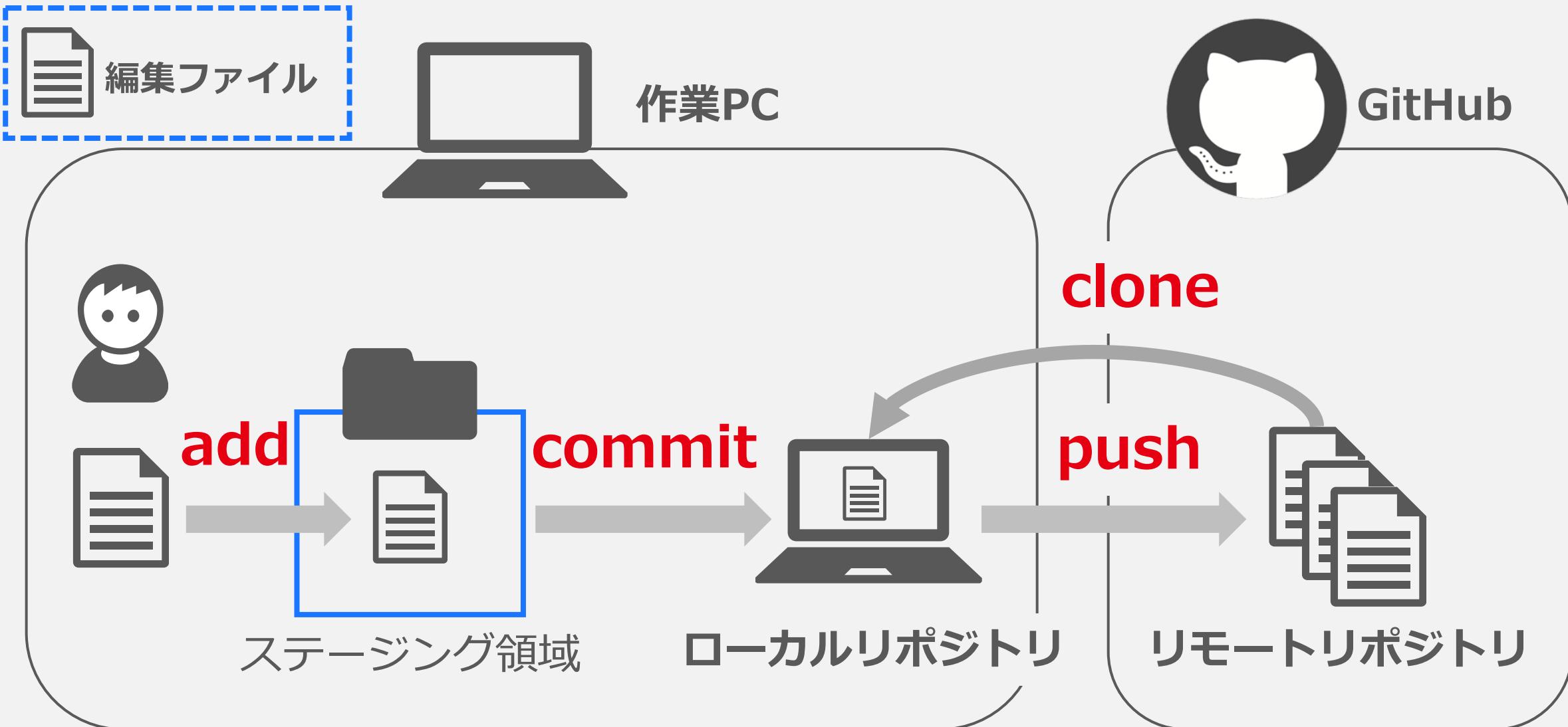
ローカルリポジトリの変更をリモートリポジトリに反映



ローカルリポジトリ

リモートリポジトリ

基本操作まとめ



チェックポイント その1

- ファイルを追加/変更して、リモートにpush

1. ローカルリポジトリでの作業

- Gitで管理したいファイルを**add**
- ローカルリポジトリの登録するために**commit**

2. リモートリポジトリへの反映

- ローカルリポジトリの内容を**push**して、
リモートリポジトリに変更を反映

実際の操作手順

clone, add,

commit, push

VSCodeで
cloneする

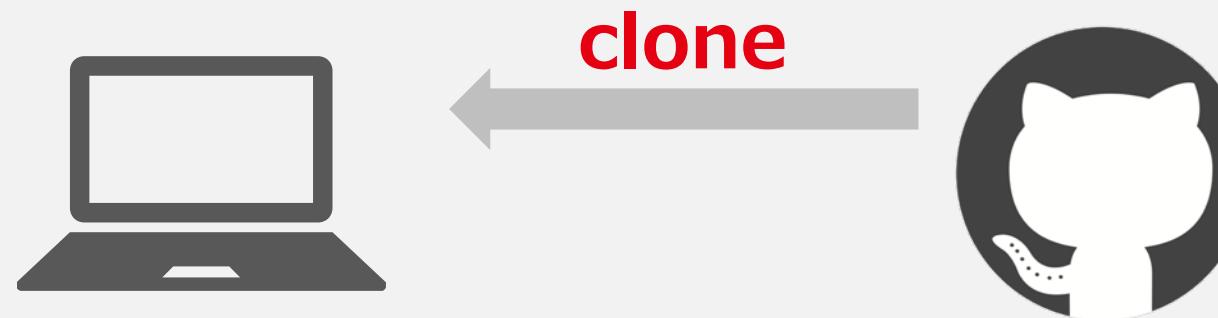


GitHub から自分のPCに持ってくる

- **clone :**

リモートリポジトリを自分のローカルリポジトリ
にコピーする

※ 初回のみの操作

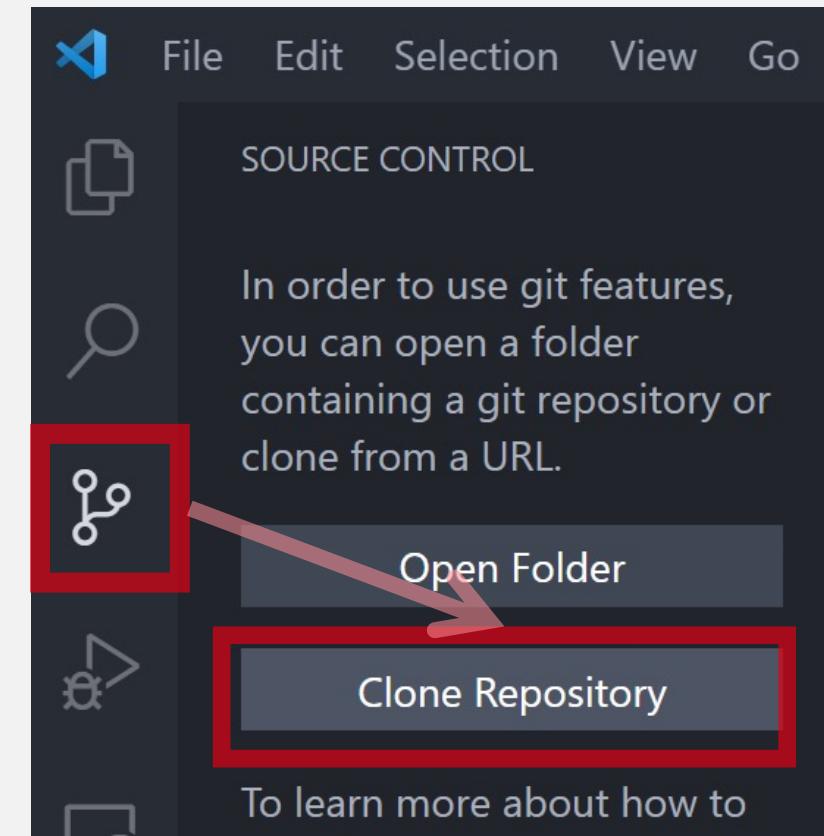
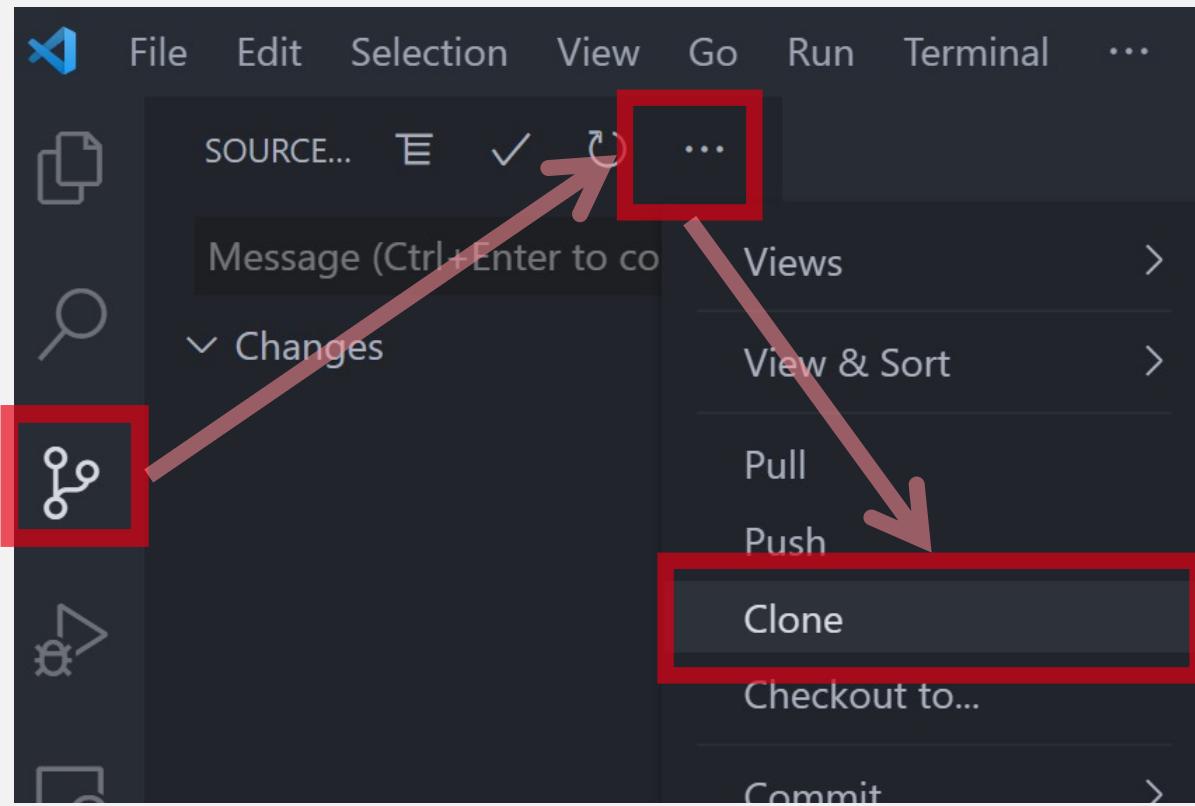


ローカルリポジトリ

リモートリポジトリ

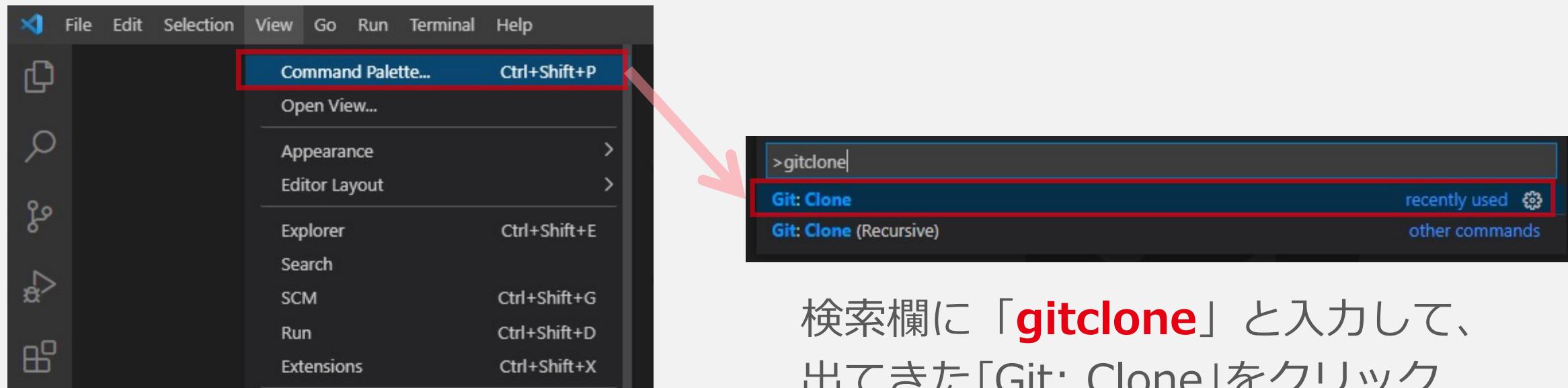
clone ①

- その時のVSCodeの状態によって、下2つのどちらかの手順でcloneを行えます



clone ① 補足

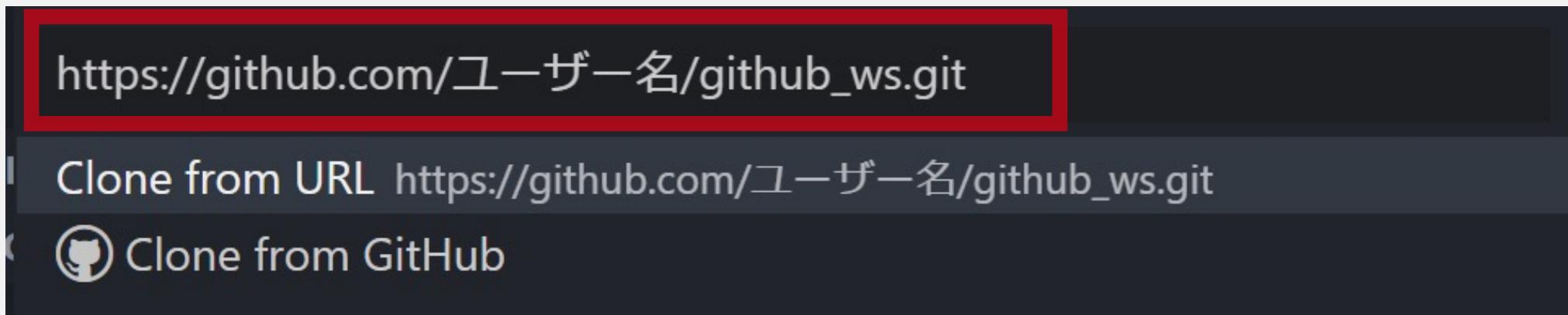
- VSCodeのUIが異なる際は、「Command Palett...」から検索します



検索欄に「**gitclone**」と入力して、
出てきた「Git: Clone」をクリック

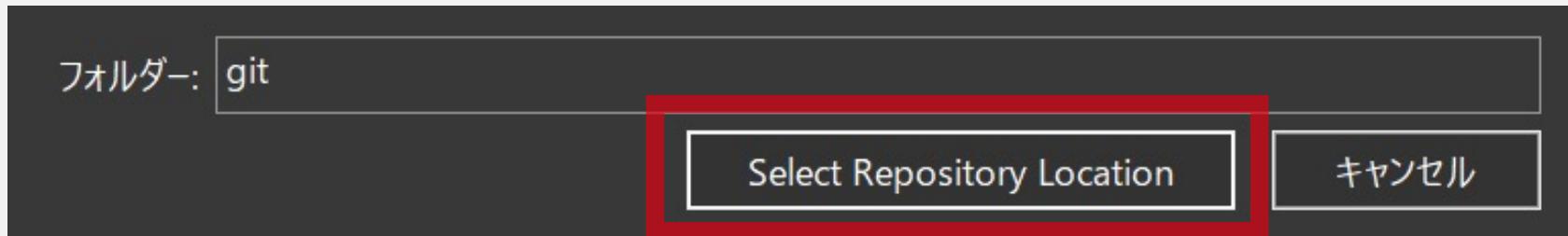
clone ②

- 出てきた入力欄に、自分で作成したリポジトリのURL
「https://github.com/ユーザー名/github_ws.git」 を入力し、Enterを押します

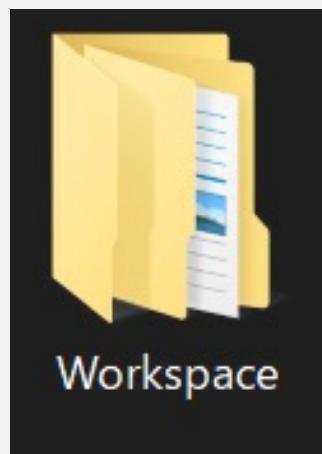
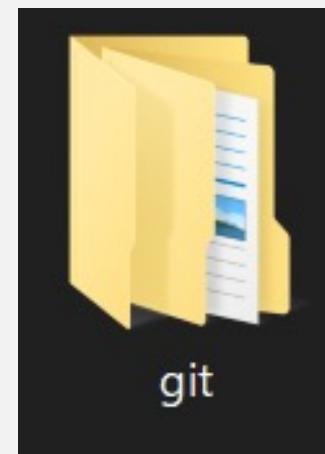


clone ③

- 適当なディレクトリを指定して
[Select Repository Location]を押す



- 「git」や「workspace」などの
ディレクトリ下にcloneすると
リポジトリが管理しやすいです



clone ④

The screenshot shows a Visual Studio Code interface. A modal dialog box is open in the center, asking "Would you like to open the cloned repository?". It has two buttons: "Open" (highlighted with a red box) and "Open in New Window". Below the dialog, the main VS Code window displays the "EXPLORER" view. In the "OPEN EDITORS" section, there is a folder named "GITHUB_WS-1" containing files "src" and "index.html". The status bar at the bottom shows the current branch is "master". A large red arrow points from the text "Openを押したら Clone完了です" (Press Open to finish cloning) to the "Open" button in the dialog.

① Would you like to open the cloned repository?

Source: Git (Extension)

Open Open in New Window

- Openを押したら
Clone完了です

EXPLORER

OPEN EDITORS

GITHUB_WS-1

src

index.html

Show All Commands $\text{Ctrl} + \text{Shift} + \text{P}$

Go to File $\text{Ctrl} + \text{P}$

Find in Files $\text{Ctrl} + \text{Shift} + \text{F}$

Start Debugging F5

Toggle Terminal $\text{Ctrl} + @$

master

Live Share

VSCodeで
addする



ファイルを修正してGitで扱う

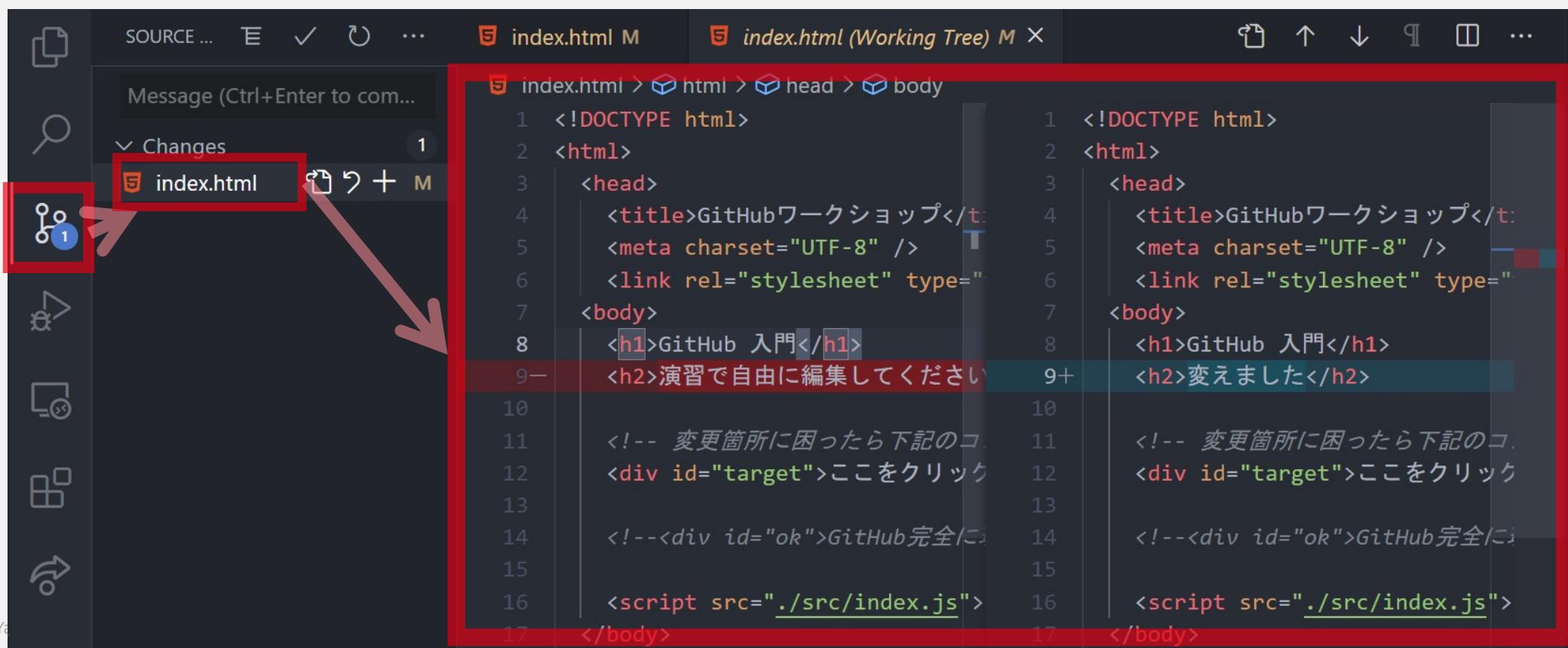
Step.1

どのファイルの変更をしたかをステージング領域に一度登録する → **add** という



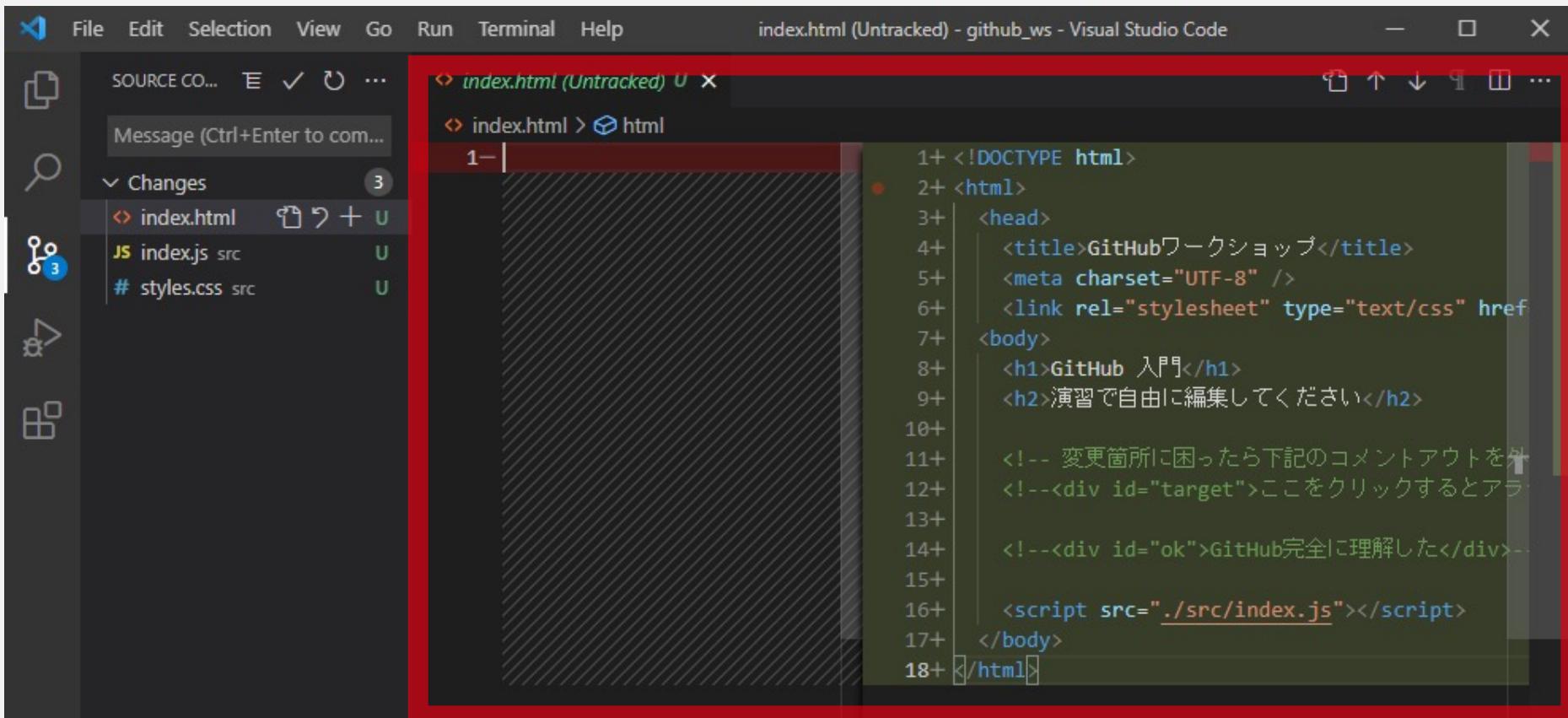
ステージングにaddする

- addする前にVSCodeのメニューから変更内容を確認できます



ステージングにaddする

- 新規ファイルをaddする際は下記のように差分表示されます

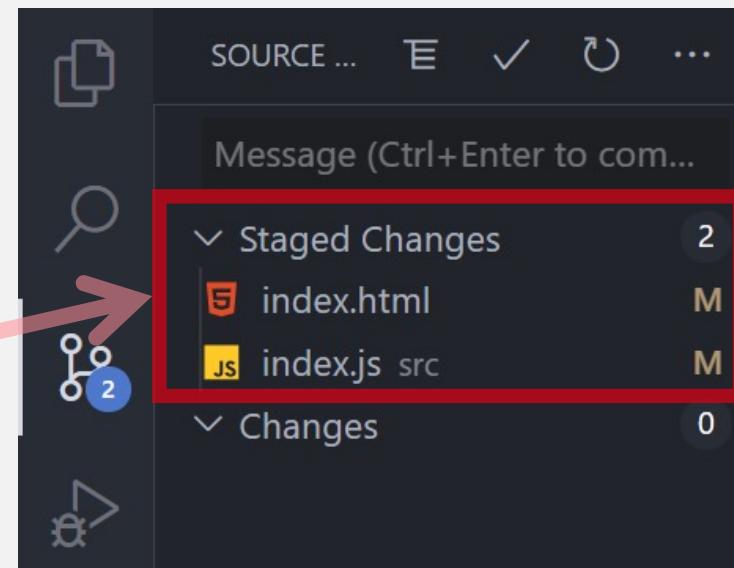
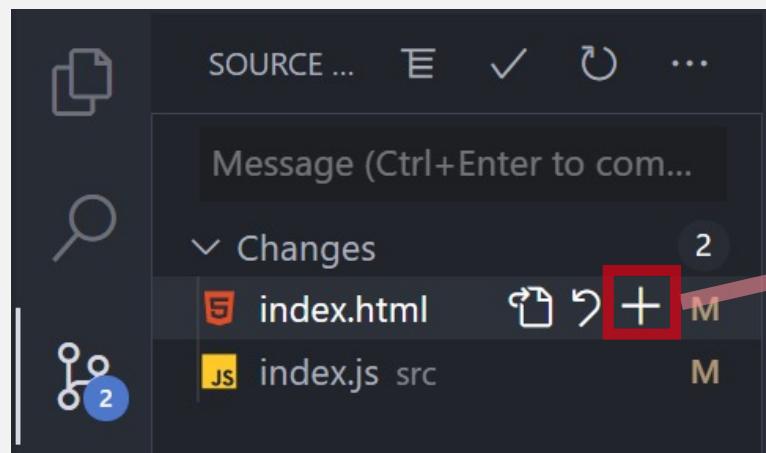


The screenshot shows the Visual Studio Code interface with the following details:

- File Bar:** File, Edit, Selection, View, Go, Run, Terminal, Help.
- Title Bar:** index.html (Untracked) - github_ws - Visual Studio Code.
- Sidebar:** Shows a tree view with "Changes" expanded, listing "index.html" (Untracked), "index.js src" (Untracked), and "# styles.css src" (Untracked). A red box highlights this sidebar area.
- Editor Area:** The main editor shows the content of index.html. The code is color-coded for syntax (HTML, CSS, JS). The first line is "1+ <!DOCTYPE html>". Lines 2 through 18 are numbered on the left, and the content includes HTML tags like <html>, <head>, <title>, <meta>, <link>, <body>, <h1>, <h2>, and <script>. A red box highlights the entire editor area.

ステージングにaddする

- addしたいファイルの「+」をクリックすると、「Changes」から「Staged Changes」にファイル名が移動します。
addするファイルを全て「Staged Changes」に入れましょう。



addする際の注意点

- add するファイルの内容を確認
 - 個人情報
 - パスワード
 - API Key などの鍵情報

リモートリポジトリにpushしたファイルは
消すことができません
これらの情報を含んだファイルはaddしない
ように注意しましょう

VSCodeで commitする



ファイルを修正してGitで扱う

Step.2

ステージング領域に登録した変更内容をローカルリポジトリに登録する → **commit**という



commit

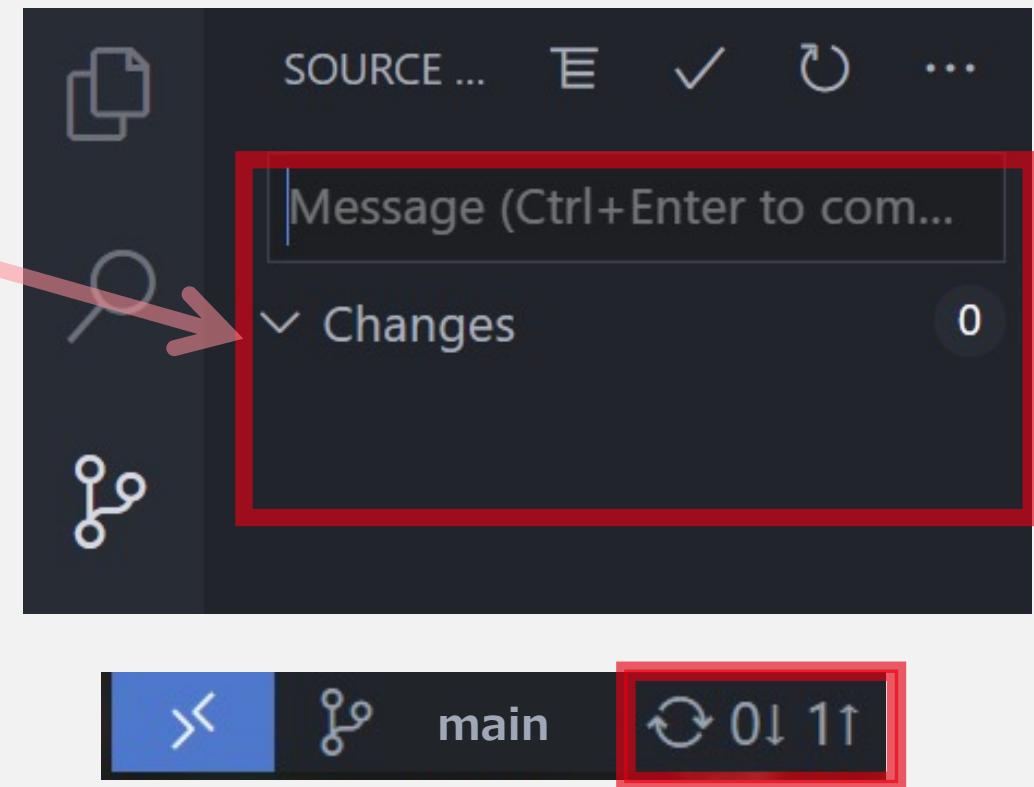
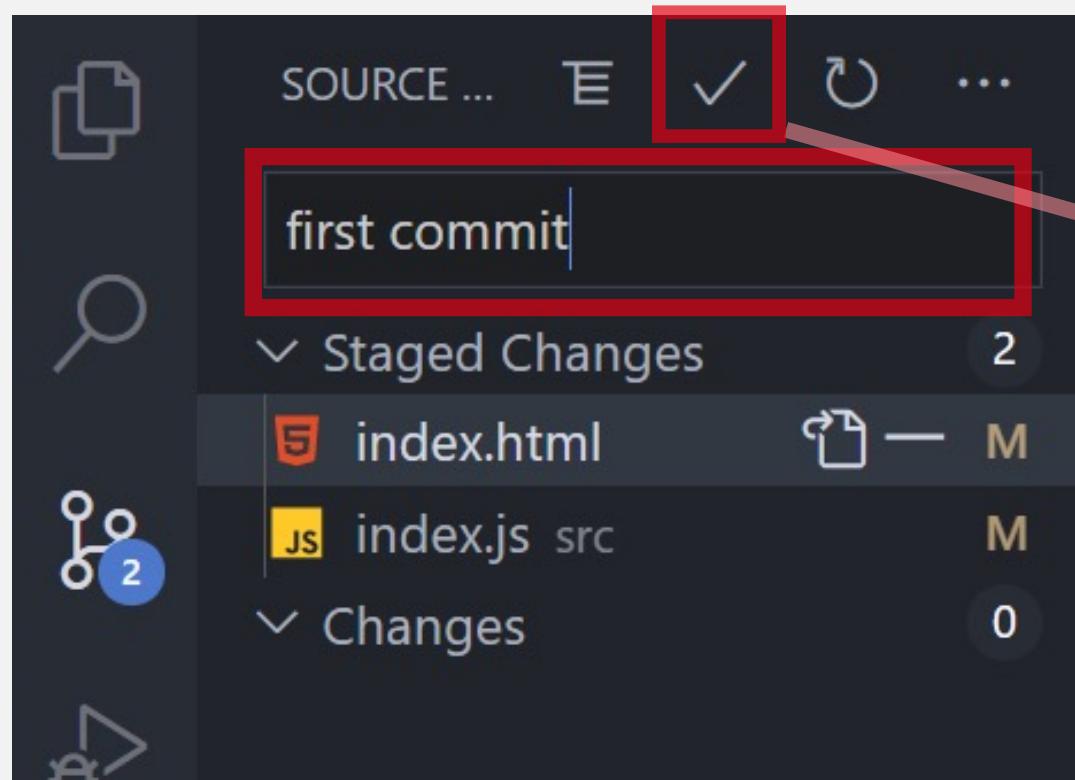


ローカルリポジトリ

ステージング領域（インデックス）

ローカルリポジトリにcommitする

- 任意のメッセージを入力し✓をクリックすると表示が更新されて画面下が「 $0 \downarrow 1 \uparrow$ 」となります



ローカルリポジトリにcommitする

- OSの違いやGit設定によっては画面下が「0↓0↑」のままである場合があります
- その場合は ⌂ の数字が消えていればOKです



VSCodeで pushする



自分のPCからGitHubに反映する

push :

ローカルリポジトリの変更をリモートリポジトリに反映

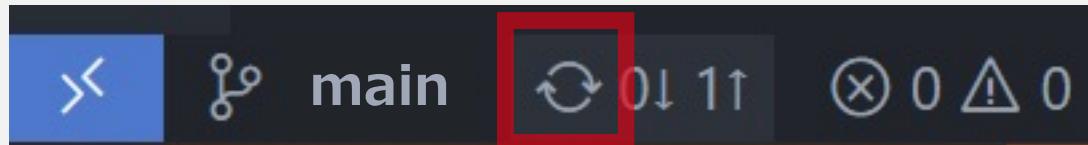


ローカルリポジトリ

リモートリポジトリ

GitHubへの疎通確認 (push)

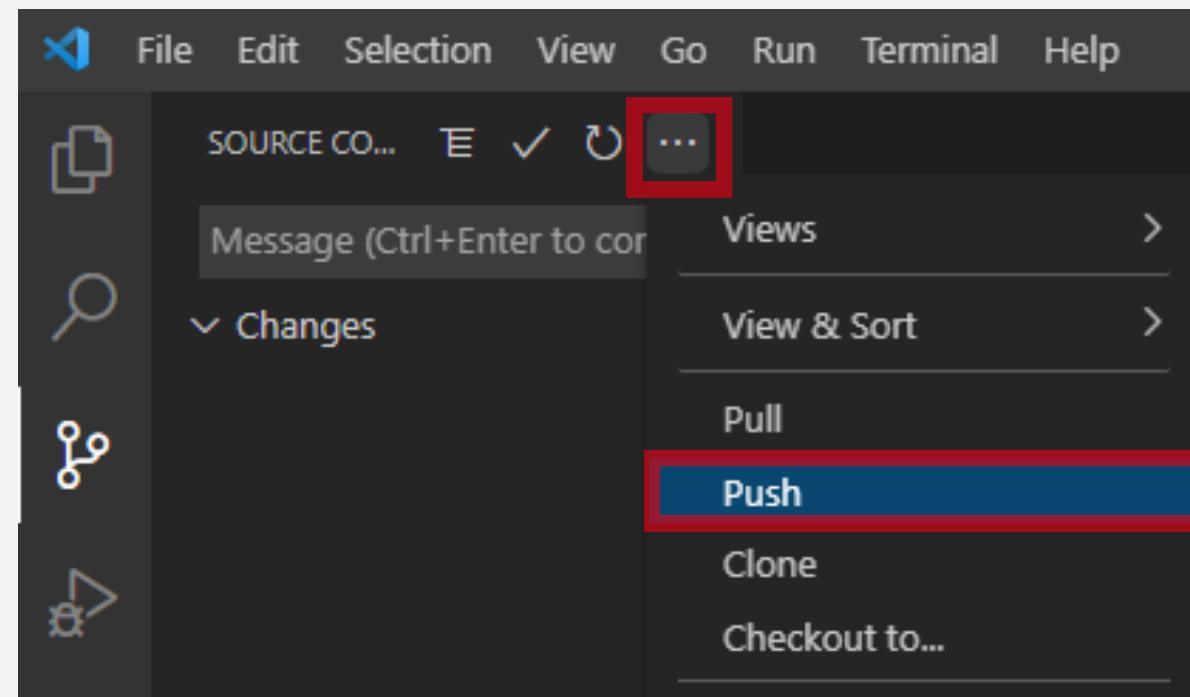
- 更新ボタン（赤枠のところ）を押すとpushされます



- 更新ボタンは押すことでpushの他にpullも行うことができます
pullができるときは「 $1\downarrow 0\uparrow$ 」のように表示されます（状況により表示されない場合もあります）

GitHubへの疎通確認（push）補足

- 更新ボタンでのpushがうまく行かないときは、下図の「…」から「Push」を選択してください



GitHubに反映されているか確認

The screenshot shows a GitHub repository page for the user 'github_ws'. The repository name is 'github_ws'. The page includes a navigation bar with links for Pull requests, Issues, Marketplace, and Explore. Below the navigation bar, there are buttons for Unwatch (1), Star (0), and Fork (0). The main content area shows a list of commits. The first commit was made by 'ユーザー名' 14 minutes ago, adding 'index.html & src' to the 'main' branch. This commit has 1 commit and was pushed at 50cdc45. Below this, two more files were added: 'src' and 'index.html', both added 14 minutes ago. A message at the bottom encourages adding a README, with a green 'Add a README' button.

Search or jump to... /

Pull requests Issues Marketplace Explore

Unwatch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights ...

main 1 branch 0 tags Go to file Add file Code

ユーザー名 Add index.html & src 50cdc45 14 minutes ago 1 commits

src Add index.html & src 14 minutes ago

index.html Add index.html & src 14 minutes ago

About No description, website, or topics provided.

Releases No releases published Create a new release

Help people interested in this repository understand your project by adding a README. Add a README

GitHubに反映されているか確認

The screenshot shows a GitHub repository page for a user named 'github_ws'. The page includes a navigation bar with links for Pull requests, Issues, Marketplace, and Explore. Below the navigation bar, there are buttons for Unwatch (1), Star (0), and Fork (0). A main menu at the top of the repository page includes Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and a three-dot menu. A large blue callout box is overlaid on the page, containing the URL https://github.com/ユーザー名/github_ws and the text 'こちらにサンプルファイルが上がっていればOKです'. At the bottom of the repository page, there is a message encouraging the addition of a README file and a green 'Add a README' button.

https://github.com/ユーザー名/github_ws

こちらにサンプルファイルが上がっていれば
OKです

Help people interested in this repository understand your project by adding a README. [Add a README](#)

Pushがうまくいかない時①

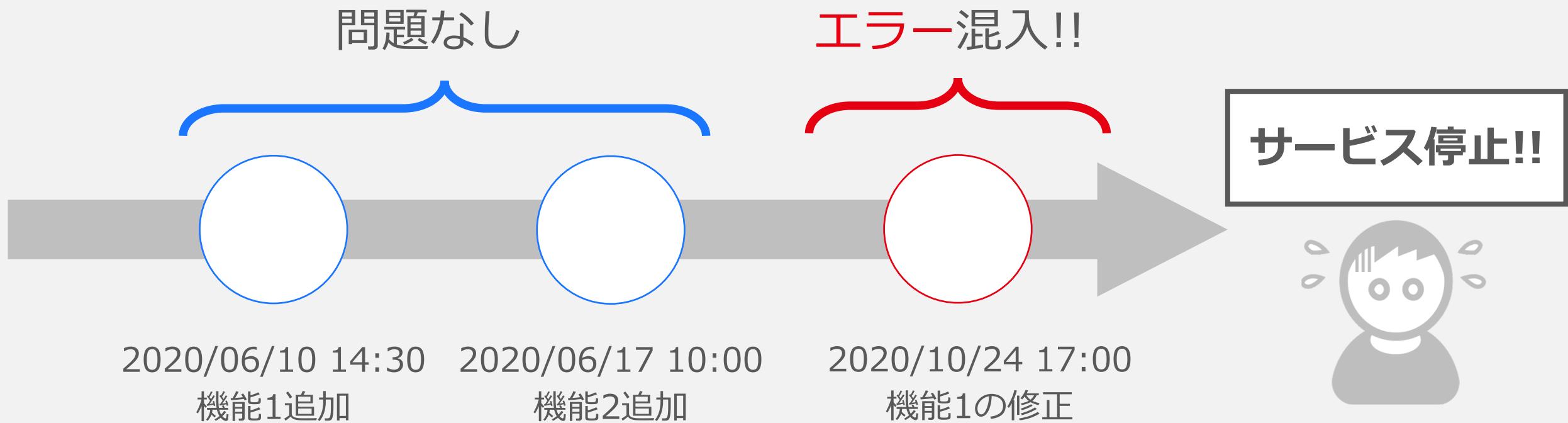
- ・ ネットワークの問題
 - ・ 学校のWifiなどを使用していると、ネットワークの設定で制限がかけられ、Pushができない場合があります。
 - ・ 別のネットワークやスマホのテザリングなどで再度試してみましょう

Step.2-1

作業用ブランチの作成

ブランチ

リポジトリに動かないコードが上がってしまう



ブランチ

リポジトリに動かないコードが上がってしまう

既存のコードにバグを含まないよう

作業環境を分けましょう

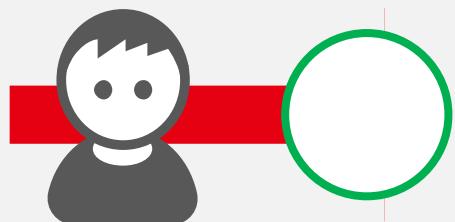
機能1追加

機能2追加

機能1の修正

ブランチ

最新の状態



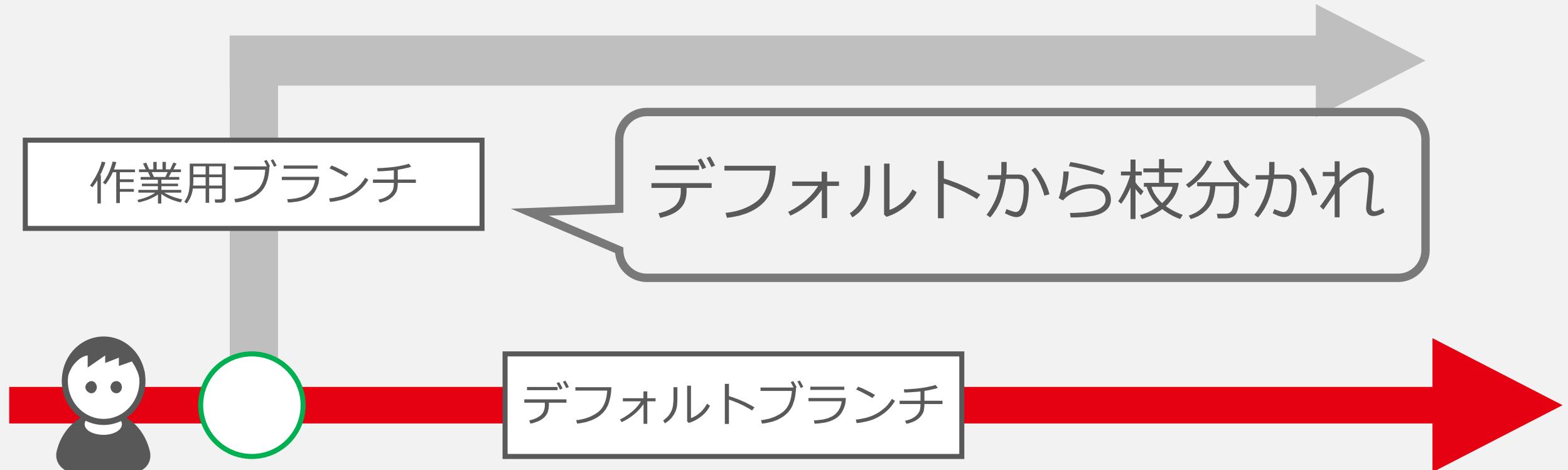
大元のブランチ

→ **デフォルトブランチ**

- ブランチ = 枝
デフォルトブランチから
枝を生やす
 - デフォルトブランチの名称は
変更可能
- ※この資料では**main**を使用します

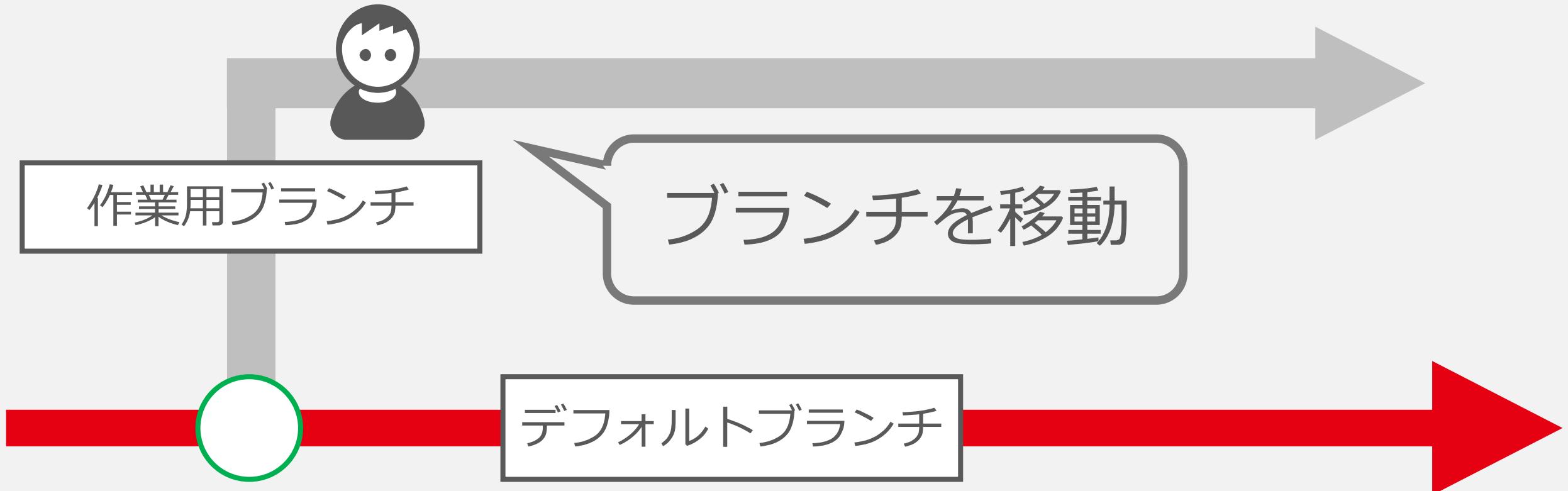
ブランチ

作業用ブランチを作成する



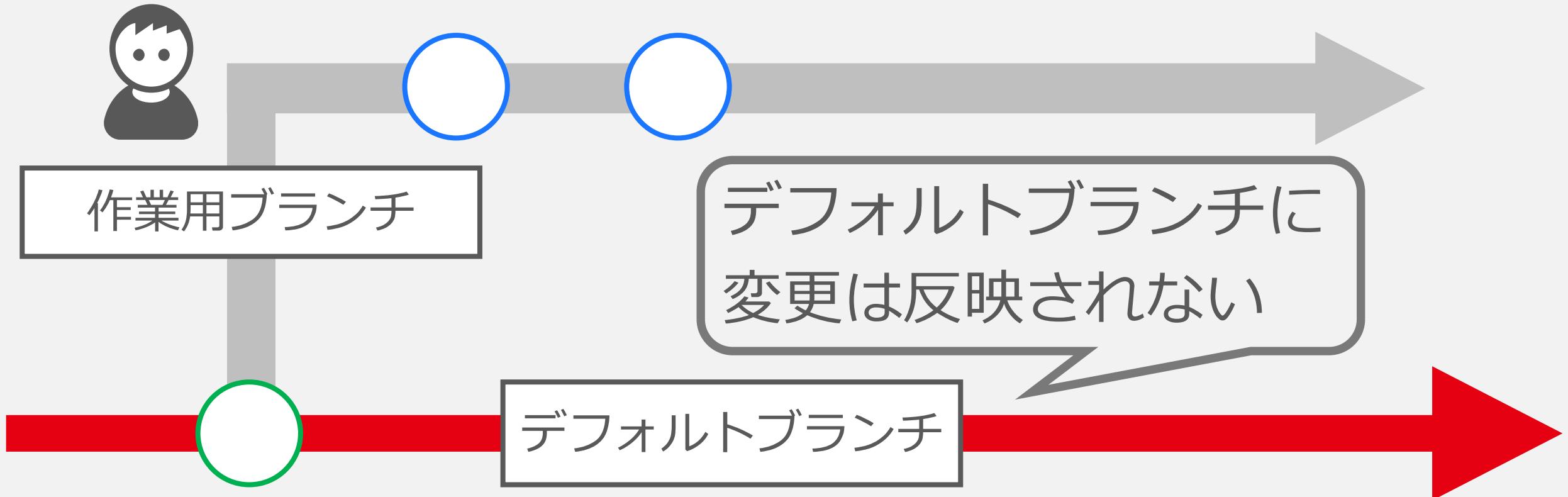
ブランチ

作業用ブランチにチェックアウトをする



ブランチ

作業ブランチで同じようにcommit/pushする



チェックポイント その2

- ・ ブランチを作成して、リモートに反映

1. ローカルリポジトリでの作業

- ① ブランチを作成 or 既にあるブランチに移動
- ② 作業用ブランチでadd/commit

2. リモートリポジトリへの反映

- ① 作業用ブランチから**push**を行い、
リモートリポジトリに変更を反映

実際の操作手順

ブランチの作成とpush

VSCodeで ブランチを作成する



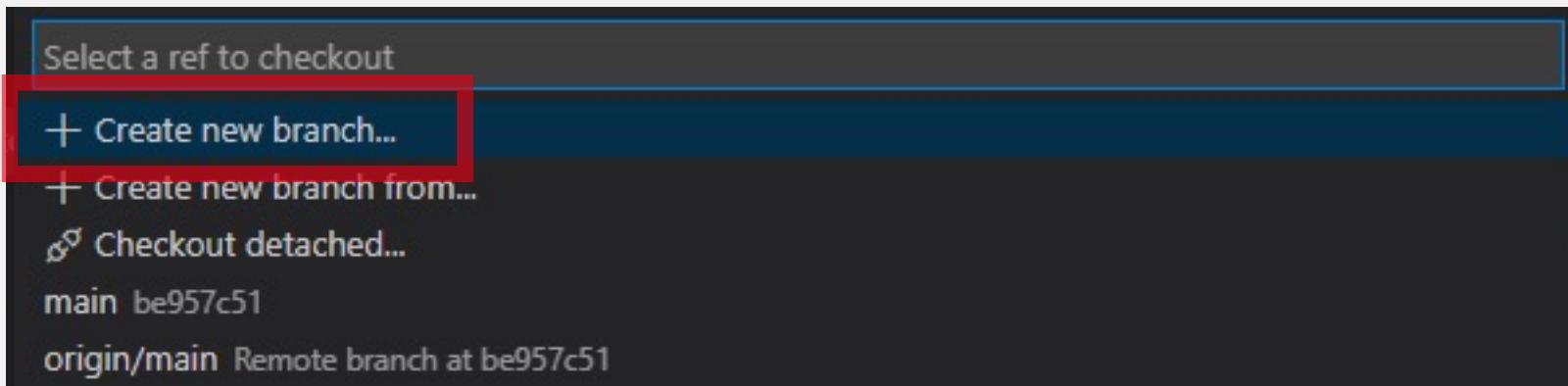
ブランチを作成する

- GitHubの「ユーザーネーム」でブランチを作成します

- 画面下のブランチ名をクリック



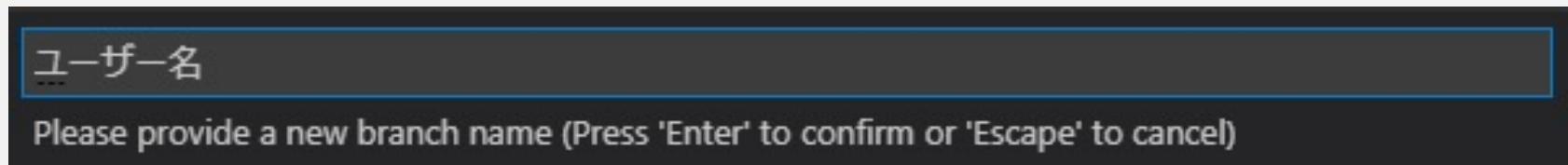
- 「Create new branch...」を選択



ブランチを作成する

- GitHubの「ユーザー名」でブランチを作成します

- 「Branch name」の入力欄にユーザー名を入力

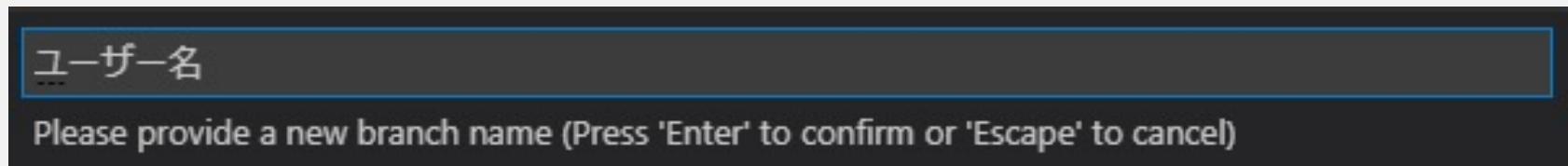


- 画面下のブランチ名が変更されたのを確認



ブランチを作成する

- GitHubの「ユーザー名」でブランチを作成します
- 「Branch name」の入力欄にユーザー名を入力



- 画面下のキー、矢印キーもしくはEnterキーを確認
- ブランチ名は自分で好きに決められますが、
今回は新規ブランチ名をGitHubアカウント
と同じにしましょう。

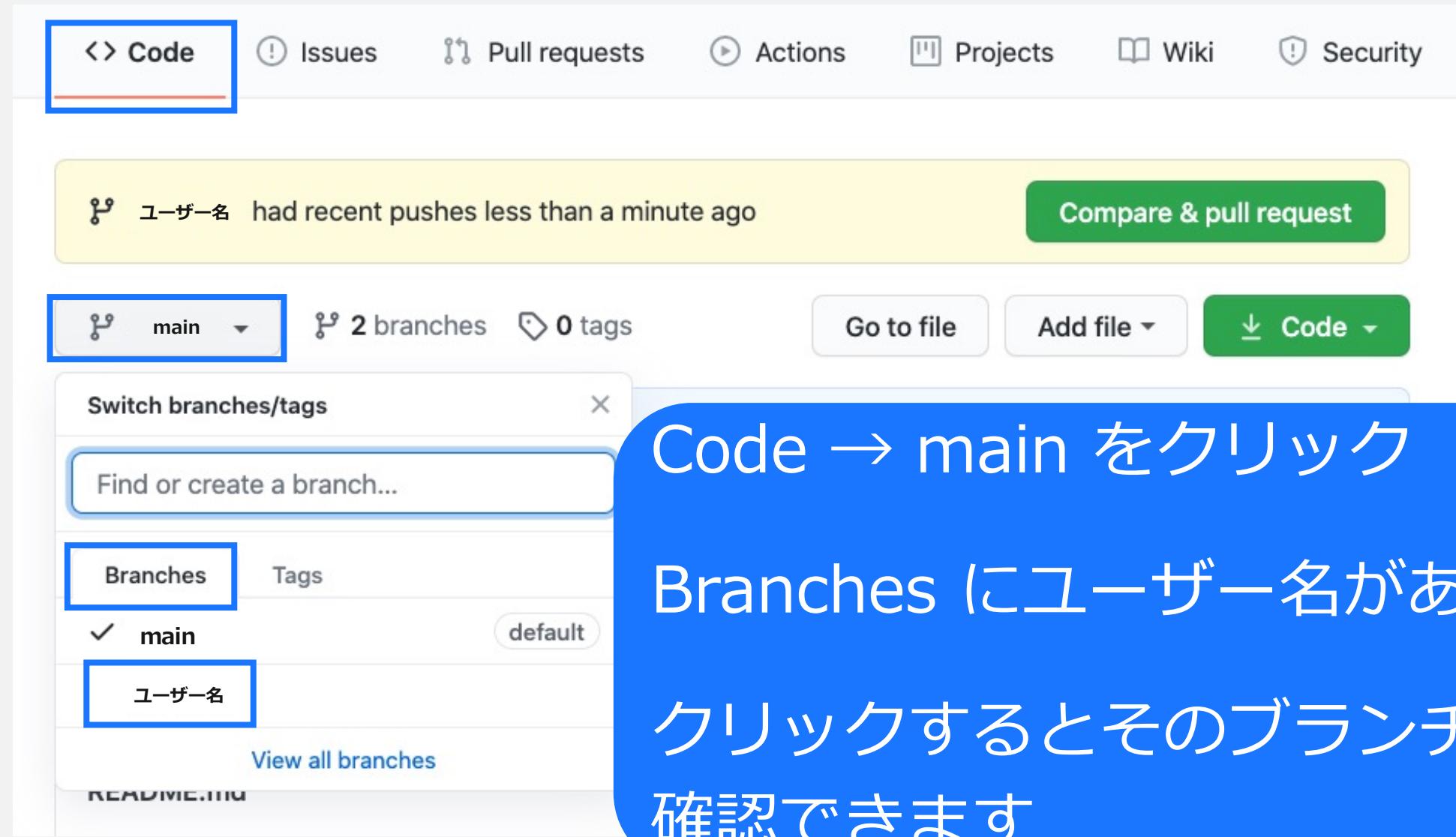
ブランチを指定して
ファイル変更



ブランチを作成してファイルをPush

- ファイルを変更してadd → commit
 - addする前にブランチがmainでないことを必ず確認しましょう
- commitしたら作成したブランチをリモートにpushします
 - mainブランチでpushしないように注意

GitHub でブランチを確認



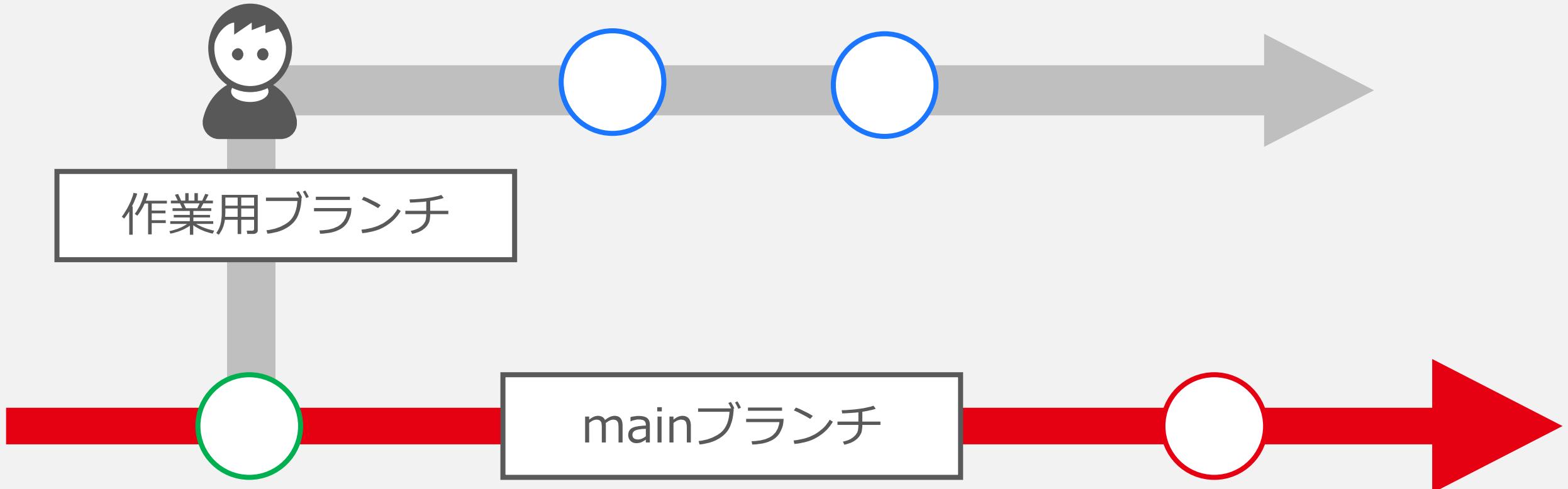
Code → main をクリック
Branches にユーザー名があればOK
クリックするとそのブランチの内容を確認できます

Step.2-2

mainに変更を反映する

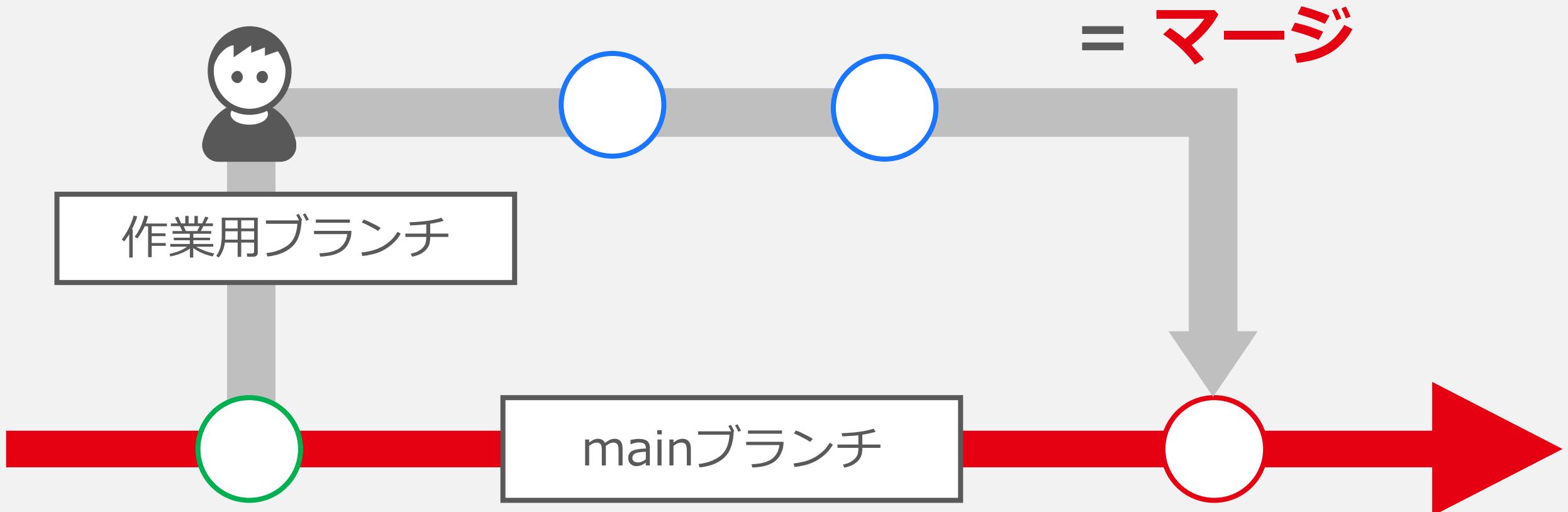
マージ

作業用ブランチの変更をmainに反映する



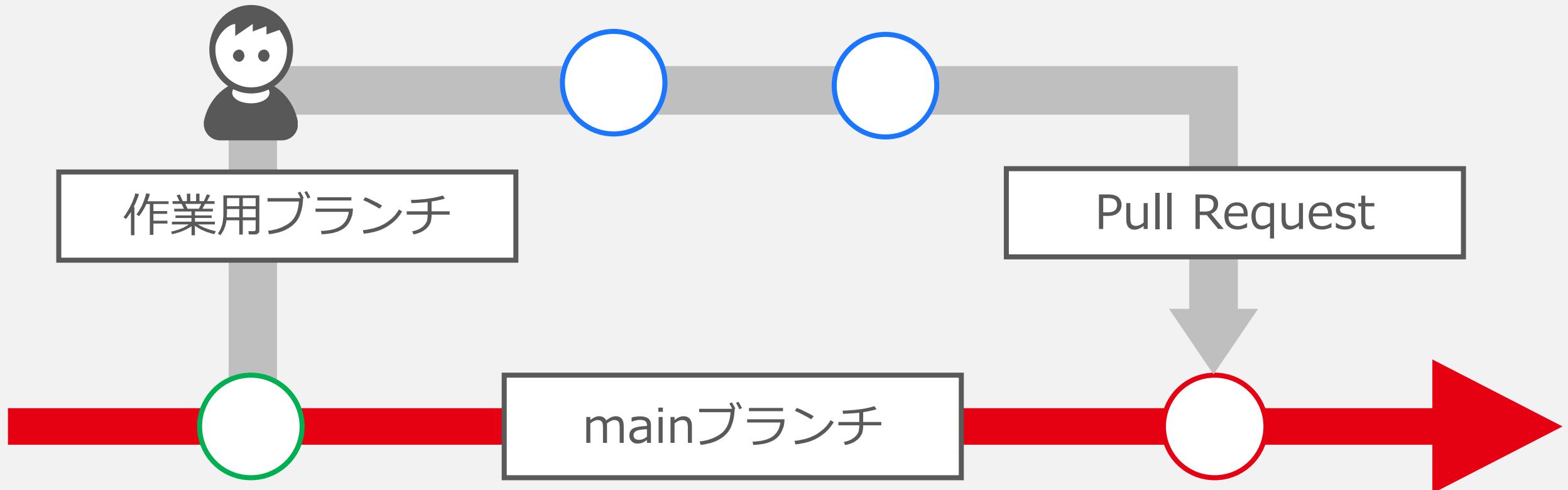
マージ

ブランチの変更を別のブランチに反映する



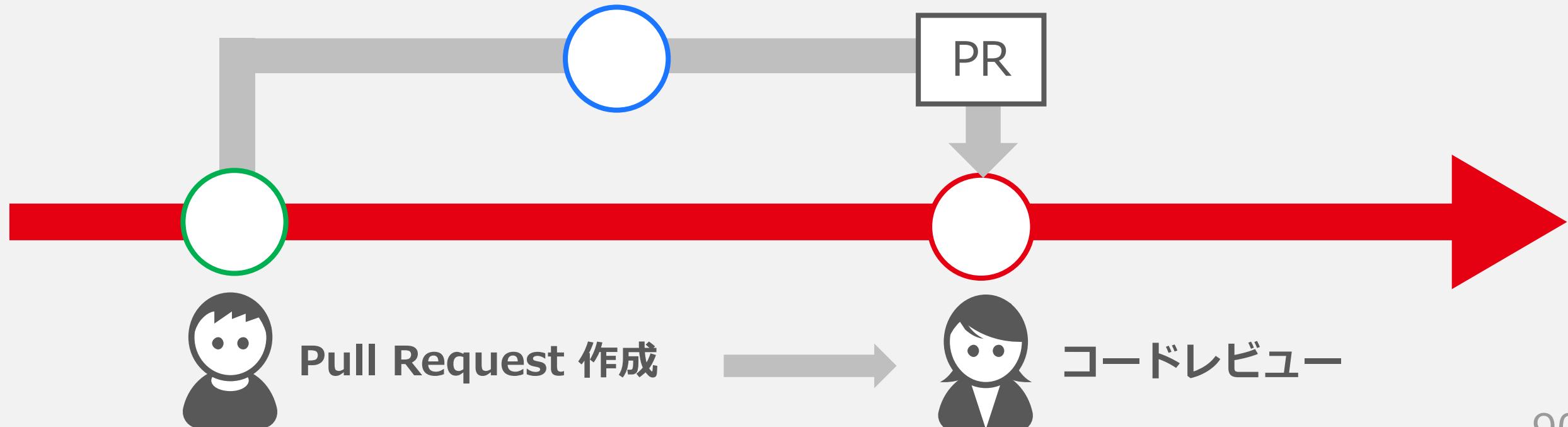
マージ

マージするために Pull Request (PR) を作成する



マージ

- **Pull Request** を作成する
 - GitHub の機能
 - マージしてOKかコードレビューをする



Pull Requestを作成してマージする

- GitHubでPull Requestを作成
 - 追加/変更した内容を記載
- Pull Requestをマージ
 - 変更内容を確認して問題なければ**Confirm merge**
 - リモートリポジトリの main が更新されているのを確認

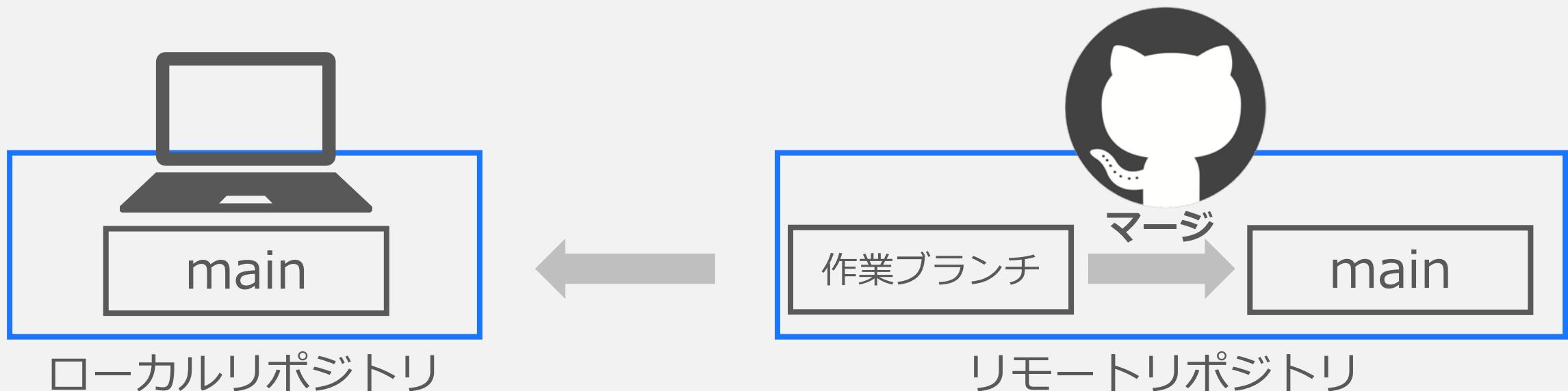
ローカルリポジトリに
pullする



マージされたものをローカルに反映

Pull Requestがマージされると

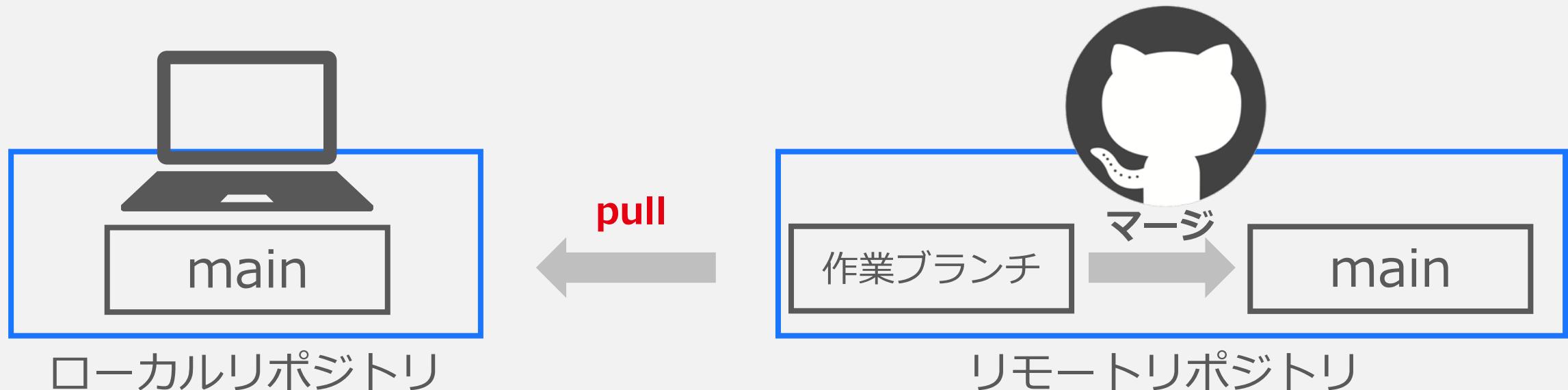
- ・ リモートのmainには反映される
- ・ ローカルのmainには反映されていない



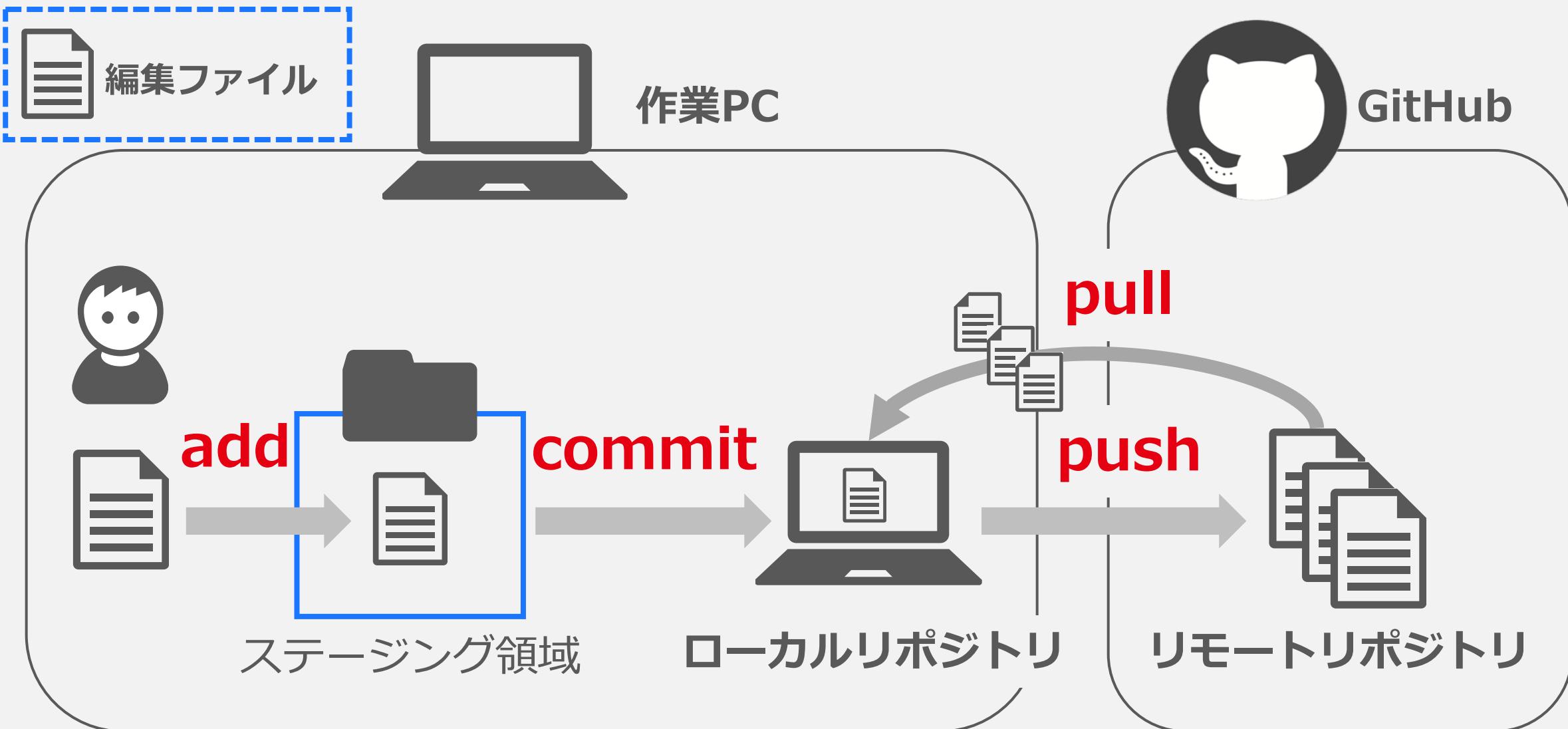
マージされたものをローカルに反映

リモートのmainの内容をローカルのmainに反映

→ ローカルに**pull**する

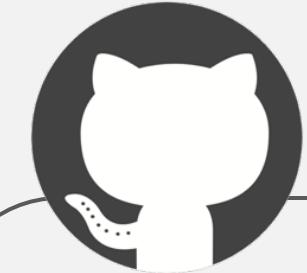


ローカルリポジトリの更新



ローカルリポジトリの更新

clone : リモートをローカルにコピー
pull : リモートの変更をローカルに反映



GitHub



チェックポイント その3

PRを作成して、ブランチの内容を別ブランチに反映

1. リモートリポジトリでの作業

- ① GitHub上でマージ元/先ブランチを指定してPR作成
- ② 変更差分を確認、可能ならコードレビューを行う
- ③ PRをマージ

2. ローカルリポジトリへの反映

- ① ローカルリポジトリをpullを実行

実際の操作手順

PRの作成, pull

Pull Requestの作成



Pull Requestを作成してマージする

- GitHubでPull Requestを作成
 - 追加/変更した内容を記載
- Pull Requestをマージ
 - 変更内容を確認して問題なければ**Confirm merge**
 - リモートリポジトリの main が更新されているのを確認

Pull Requestを作成方法

The screenshot shows a GitHub repository page. At the top, there is a navigation bar with links: Code (highlighted with a red underline), Issues, Pull requests, Actions, Projects, Wiki, and Security. Below the navigation bar, a yellow banner displays a message: "ユーザーネーム had recent pushes less than a minute ago" and a green button labeled "Compare & pull request". Underneath the banner, there are buttons for "main" (with a dropdown arrow), "2 branches", "0 tags", "Go to file", "Add file", and "Code". On the left side, there is a sidebar with icons for a user profile, "Add R", "src", "README.md", and "index.html". A large blue callout box contains Japanese text: "GitHubにいくとこのような画面が出てきます (青枠注文)" and "※もし出でない場合はPull requestsのページから作成できます".

ユーザーネーム had recent pushes less than a minute ago

Compare & pull request

main

2 branches

0 tags

Go to file

Add file

Code

ユーザーネーム Add R

src

README.md

index.html

README.md

GitHubにいくとこのような画面が
出てきます (青枠注文)

※もし出でない場合はPull requestsの
ページから作成できます

Pull Requestを作成方法

Code Issues Pull requests Actions Projects Wiki Security

ユーザーネーム had recent pushes less than a minute ago

Compare & pull request

main 2 branches 0 tags Go to file Add file Code

ユーザーネーム Add REA/

Compare & pull requestを押す

src	Add index.html & src	1 hour ago
README.md	Add README.md	39 minutes ago
index.html	Add README.md	39 minutes ago

README.md

Pull Requestの作成方法

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

The screenshot shows the GitHub interface for creating a pull request. At the top, it says "base: main" and "compare: ブランチ名". A green checkmark indicates "Able to merge. These branches can be automatically merged." Below this, there's a title field labeled "PR タイトル" and a rich text editor with "Write" and "Preview" tabs. The editor has a toolbar with H, B, I, etc., buttons. The main body area says "変更内容を記載" and has a file attachment section. A large green button at the bottom right says "Create pull request". To the right of the editor, there are settings for "Reviewers", "Assignees", "Labels", "Projects", "Milestone", and "Linked issues". A note at the bottom says "Remember, contributions to this repository should follow our [GitHub Community Guidelines](#)".

base: main ← compare: ブランチ名 ✓ Able to merge. These branches can be automatically merged.

PR タイトル

Reviewers

No reviews

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked issues

Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Pull Requestの作成方法

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

The screenshot shows the GitHub interface for creating a pull request. At the top, there are dropdown menus for 'base' set to 'main' and 'compare' set to 'ブランチ名'. A green checkmark indicates 'Able to merge. These branches can be automatically merged.' Below this, a large blue callout box highlights three key parameters:

- main ← 作成したブランチ名**
- base: 変更を反映するブランチ**
- compare: 変更作業を行ったブランチ**

The interface includes fields for 'PR タイトル' (Title), 'Write' (Draft), 'Prev' (Previous), '変更内容を記載' (Describe changes), 'Attach files by dragging & dropping, selecting or pasting them.', and a 'Create pull request' button.

① Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Pull Requestの作成方法

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

The screenshot shows the GitHub interface for creating a pull request. At the top, it says "base: main" and "compare: ブランチ名". A green checkmark indicates "Able to merge. These branches can be automatically merged." Below this, there's a title input field containing "PR タイトル", which is highlighted with a blue rectangular selection. To the left of the title is a red octocat icon. Below the title is a toolbar with "Write" (selected), "Preview", and various rich text icons (H, B, I, etc.). A large blue callout box covers the middle section, containing the Japanese text "Pull Requestのタイトル" and "→ 概要がひと目でわかるように". To the right of the title input, there are sections for "Reviewers" (No reviews), "Assignee" (Assign yourself), "Projects" (None yet), "Milestone" (No milestone), and "Linked issues". At the bottom, there's a note about following GitHub Community Guidelines and a "Create pull request" button.

PR タイトル

PR タイトル
→ 概要がひと目でわかるように

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

① Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Pull Requestの作成方法

Open a pull request

Create a new pull request

Pull Requestの概要

→ どのような変更をしたか説明を記載

Write

Preview



変更内容を記載

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

No reviews

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked issues

① Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).



Pull Requestの作成方法

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

The screenshot shows the GitHub interface for creating a pull request. At the top, it says "base: main" and "compare: ブランチ名". A green checkmark indicates "Able to merge. These branches can be automatically merged." Below this, there's a title field labeled "PR タイトル" and a rich text editor toolbar with "Write" and "Preview" buttons. The main body area has a placeholder "変更内容を記載". On the right side, there are sections for "Reviewers" (No reviews), "Assignees" (No one—assign yourself), "Labels" (None yet), and "Milestone" (No milestone). At the bottom, there's a large blue button with the Japanese text "タイトルと概要入力したらクリック" (Click after entering title and summary). The "Create pull request" button is highlighted with a red box.

タイトルと概要入力したらクリック

Attach files by dragging & dropping, selecting or pasting them.

Create pull request

① Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

Copyright (C) 2020 Yahoo Japan Corporation. All Rights Reserved.

Pull Requestの作成方法

The screenshot shows a GitHub Pull Request page with the following details:

- Pull requests**: 1
- Conversation**: 0
- Commits**: 1
- Checks**: 0
- Files changed**: 2

A comment from "ユーザー名" is visible, stating "commented now" and "変更内容を記載".

The commit list shows:

- o Update index.html 78149a0

A note at the bottom says: "Add more commits by pushing to the ユーザー名 branch on ユーザー名 / github_ws."

Continuous integration status:

- Continuous integration has not been set up**
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.
- This branch has no conflicts with the base branch**
Merging can be performed automatically.

At the bottom, there is a "Merge pull request" button and a note: "You can also open this in GitHub Desktop or view command line instructions."

Pull Requestの作成方法

PR タイトル #1

Open ykato524 wants to merge 1 commit into main from ykato524

Conversation 0 Commits 1 Checks 0 Files changed 2 +4 -1

Changes from all commits ▾ File filter... ▾ Jump to... ▾ Review changes ▾

1 / 2 files viewed

「Files changed」に変更差分が掲載される

...
11	11	<!-- 変更箇所に困ったら下記のコメントアウトを外してください -->
12	12	<div id="target">ここをクリックするとアラート発生</div>
14	-	<!--<div id="ok">GitHub完全に理解した</div>-->
	+	<div id="ok">GitHub完全に理解した</div>
15	15	
16	16	<script src=".src/index.js"></script>
17	17	</body>

Pull Requestの確認

The screenshot shows a GitHub Pull Request page. At the top, there's a commit history with one entry: "Update index.html" by user "78149a0". Below the commit, a message says "Add more commits by pushing to the ユーザー名 branch on ユーザー名 / github_ws." A green icon with a gear and a plus sign indicates CI status.

Continuous integration has not been set up
GitHub Actions and several other apps can be used to automatically catch bugs and enforce style.

This branch has no conflicts with the base branch
Merging can be performed automatically.

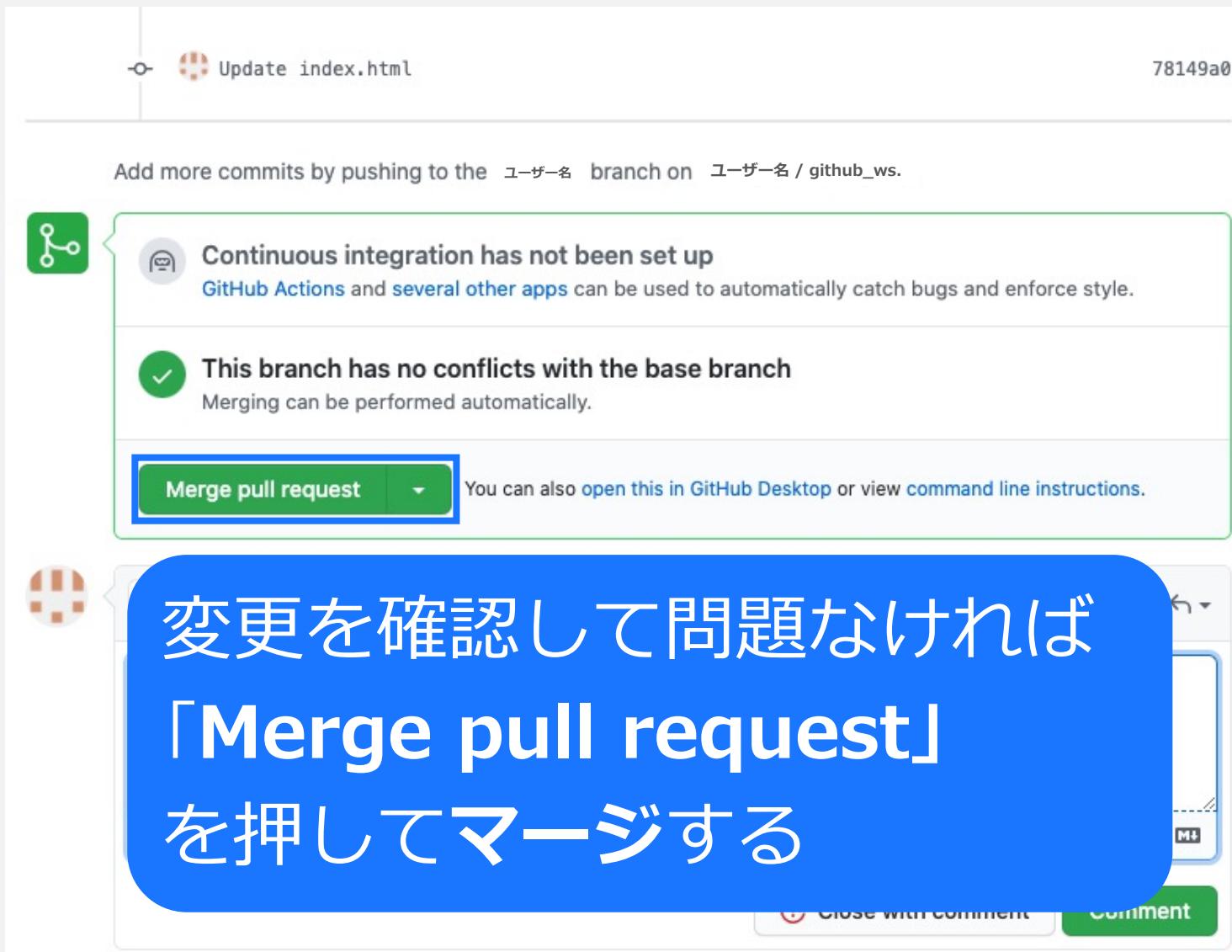
Merge pull request You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

A comment box is open, showing the text "レビューや気になった内容はコメントできる". The "Comment" button is highlighted with a blue border.

Pull Requestの確認



Pull Requestをマージ

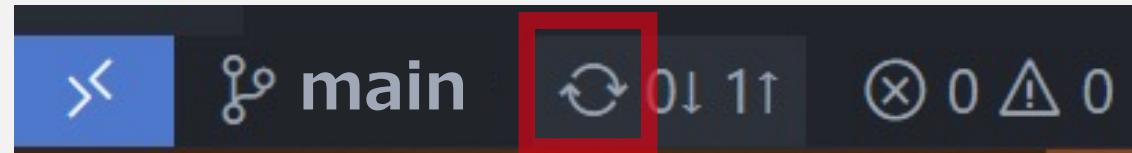


VSCodeで
pullする



ローカルにpullする

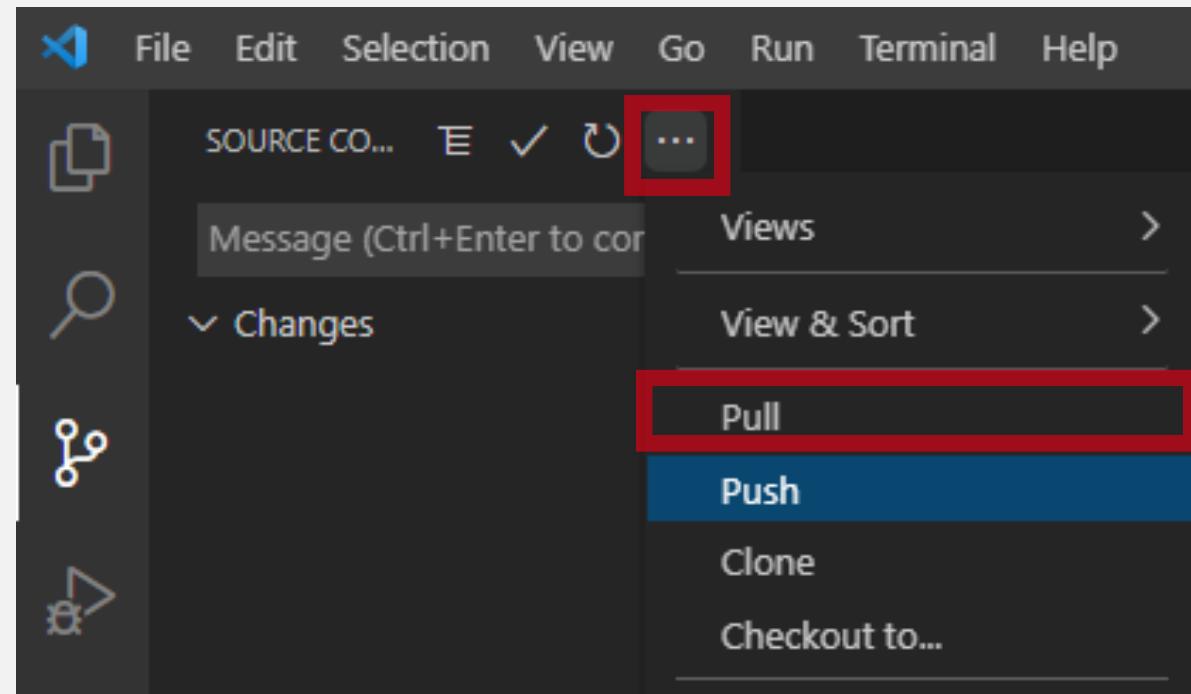
1. 画面下のブランチ名をクリックして、mainをクリックしてブランチを移動
2. 「1↓0↑」のように表示されているとき更新ボタンを押すとpullされます
(状況により表示されない場合もあります)



3. ローカルファイルが更新されていくことを確認

ローカルにpullする

- 更新ボタンでのpushがうまく行かないときは、下図の「…」から「Pull」を選択してください



ここからが本題

YAHOO!
JAPAN

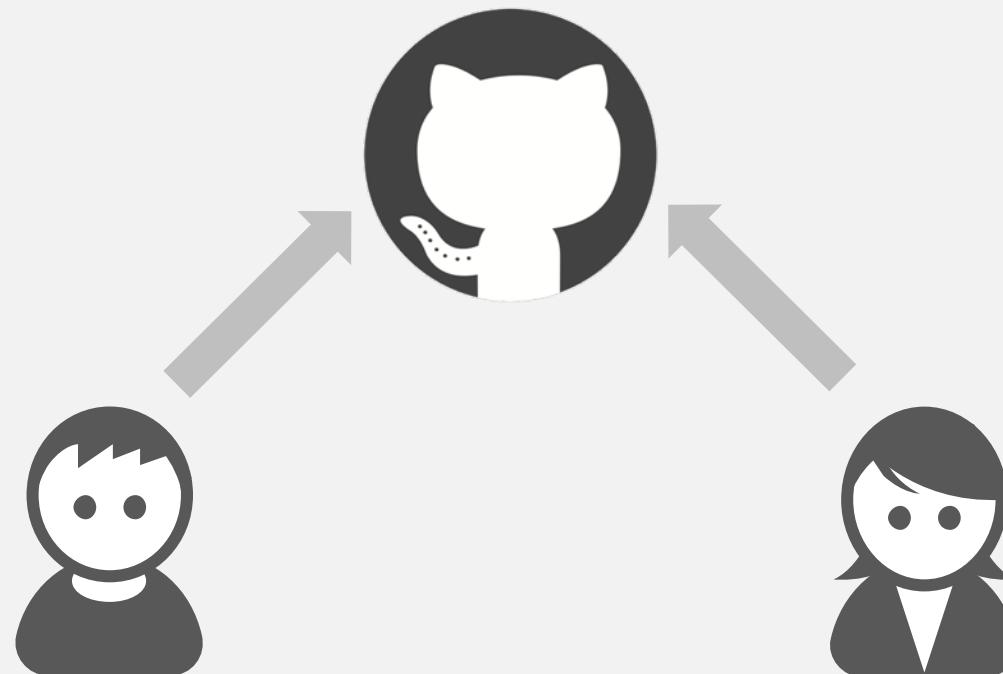
Step.3

GitHubをチームで使う



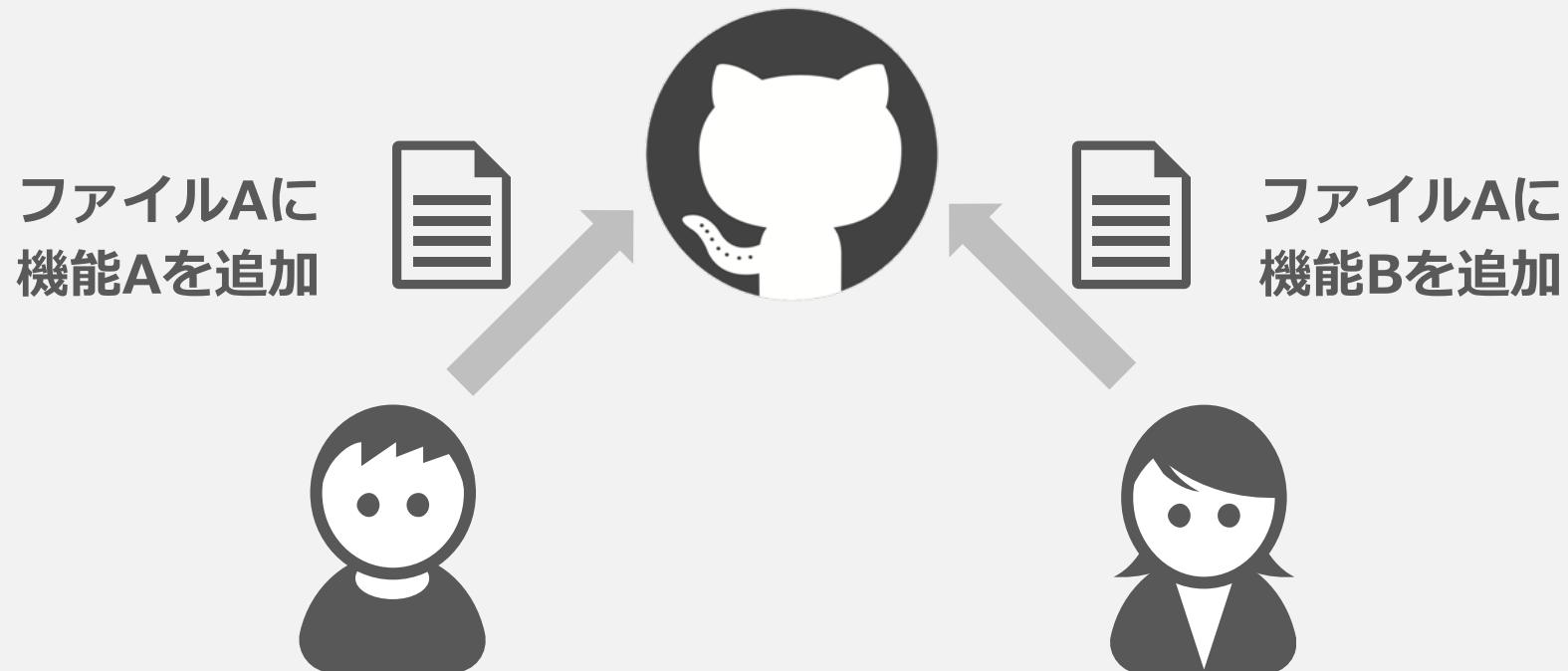
チーム開発では

- 1つのリポジトリに対して複数人で作業する



チーム開発では

- ・ 1つのリポジトリに対して複数人で作業する
- ・ 同じファイルを同時に修正してしまう可能性も



チーム開発では

- ・ 1つのリポジトリに対して複数人で作業する
- ・ 同じファイルを同時に修正してしまう可能性も



チーム開発では

- ・ 1つのリポジトリに対して複数人で作業する
- ・ 同じコードを同時に修正してしまう可能性もある

**Conflict (競合) を避けるため
作業or作業者ごとに
ブランチを分けましょう**



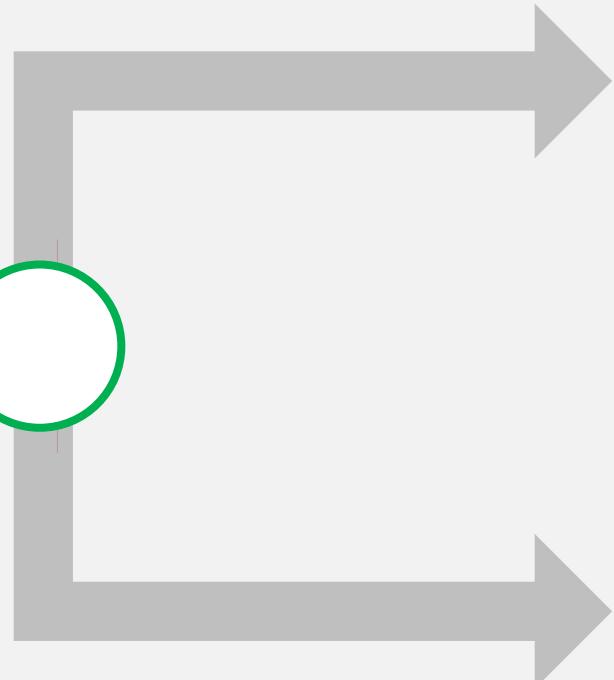
同時に push は
できない



チーム開発では



Aさんの作業ブランチ「A」



Bさんの作業ブランチ「B」

作業者がそれぞれブランチを作成

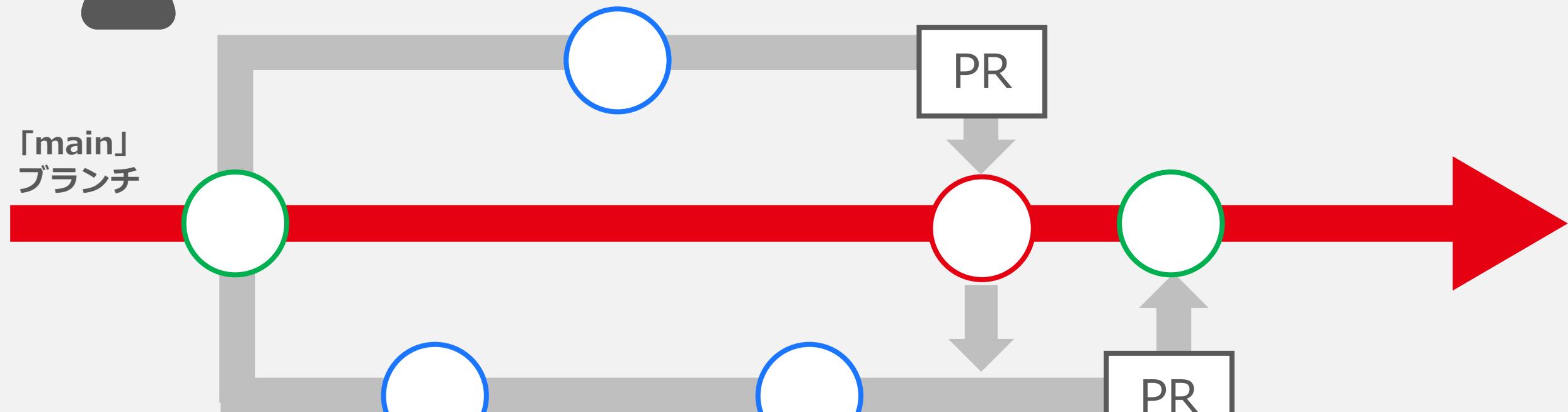
チーム開発では



Aさんの作業ブランチ「A」

Pull Request (以下 PR)で
コードレビューをしてマージ

「main」
ブランチ



Bさんの作業ブランチ「B」

チーム開発では



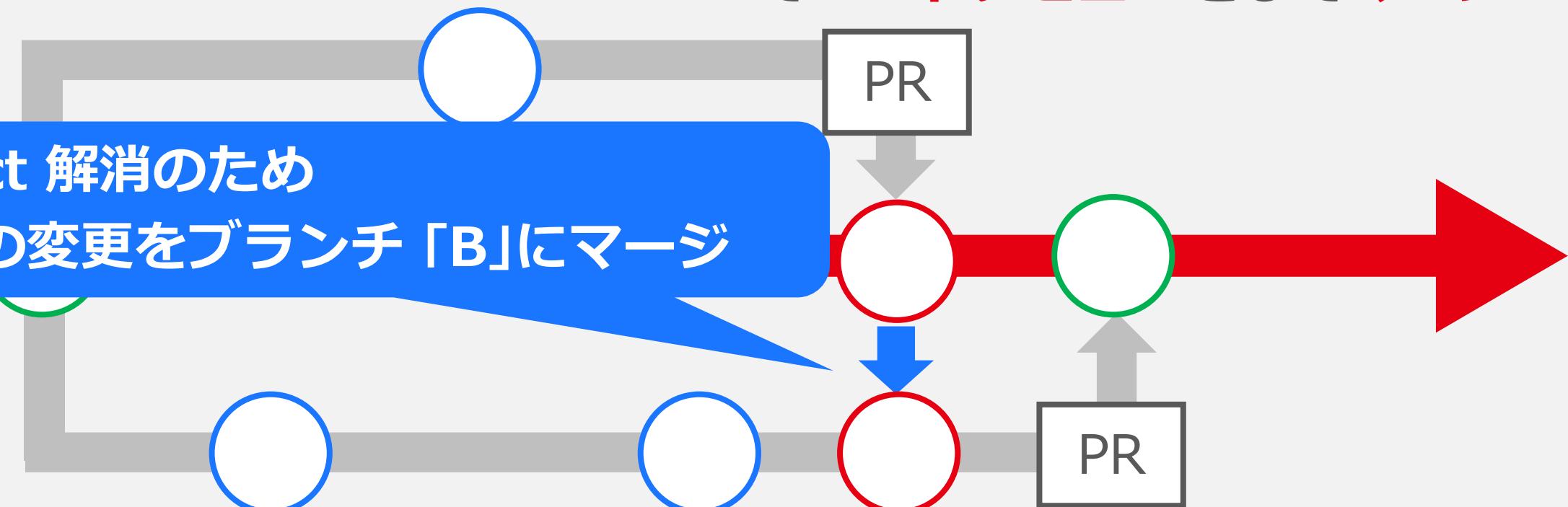
Aさんの作業ブランチ「A」

作業者がそれぞれブランチを作成
→ PRでコードレビューをしてマージ

Conflict 解消のため
main の変更をブランチ「B」にマージ



Bさんの作業ブランチ「B」



Conflictを起こさない

YAHOO!
JAPAN

Conflictを起こさない

Conflict が発生しないように開発する

- 編集ファイルが被らないように分担
- 予めコードの編集箇所を定めておく
 - 例: 実装予定の関数を先にpushしておく

Conflictが起きたときは

YAHOO!
JAPAN

Conflictの発生

PR の「Merge pull request」がグレーアウト
- merge 先のブランチとの間で Conflict が発生



Conflict 箇所の特定

「Resolve conflicts」から Conflict を確認

This branch has conflicts that must be resolved

Use the [web editor](#) or the [command line](#) to resolve conflicts.

Conflicting files

src/index.js

Merge pull request ▾ You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Resolve conflicts

A screenshot of a GitHub pull request interface. At the top left is a network icon. Next to it, a warning icon (triangle with exclamation) contains the text "This branch has conflicts that must be resolved". Below this, a message says "Use the [web editor](#) or the [command line](#) to resolve conflicts.". A red box highlights the "Resolve conflicts" button at the top right. In the middle section, under "Conflicting files", "src/index.js" is listed. At the bottom, there's a "Merge pull request" button with a dropdown arrow, followed by the text "You can also [open this in GitHub Desktop](#) or view [command line instructions](#)".

Conflict 箇所の特定

Conflict が発生したファイル内容が表示される

- 下記のようになっている箇所を確認

```
131 <<<<< A1
132   function add() {}
133   function sub() {}
134 =====
135   function mul() {}
136   function div() {}
137 >>>>> main
```

Conflict 箇所の特定

Conflict が発生した PR を出したブランチでの
- 下記のようにコード変更点

```
131 <<<<< A1
132     function add() {}
133     function sub() {}
134     =====
135     function mul() {}
136     function div() {}
137     >>>>> main
```

Conflict 简所の特定

Conflict が発生したファイル内容が表示される

- 下記のようになっている箇所を確認

```
131      <<<
132
133      function sub() {
134      =====
135          function mul() {}
136          function div() {}
137      >>>>> main
```

PR を merge する先の
ブランチでのコード

Conflict の修正

Conflict 節所のコードを確認

- Conflictしたコードのどちらかを消す
- Conflictを示す記号を消して、
Conflictを起こした変更をどちらも残す

状況に応じてどちらかで対応する

Conflict の修正方法

Conflict の修正方法は2つの方法で行える

- GitHub 上で該当コード箇所を修正
- ローカルでコードを修正して再度 push

GitHub で修正する

1. PR -> 「Resolve conflicts」のページからコード修正
2. 修正したファイルで「Mark as resolves」を押しチェックを付ける
3. 全てのファイルを修正したら
「Commit marge」ボタンから commit

コード修正前 (PRの修正内容を残す場合)

A1 のコード内容のみ残したいため、Conflict を示す記号と main のコード部分を消す

Add add & sub #1

Resolving conflicts between A1 and main and committing changes → A1

1 conflicting file src/index.js 1 conflict Prev ⌂ Next ⌂ Mark as resolved

index.js src/index.js

```
127
128      /* 演習の編集範囲 はじめ */
129
130      /* 演習1 ここから */
131      <<<<< A1
132      function add() {}
133      function sub() {}
134      =====
135      function mul() {}
136      function div() {}
137      >>>>> main
138      /* 演習1 ここまで */
```

矢印の1行と青枠内の4行
を削除する
(132,133行目を残す)

コード修正前（どちらも残す場合）

A1 と main どちらコードも残したいため、
Conflict を示す記号だけ消す

Add add & sub #1

Resolving conflicts between A1 and main and committing changes → A1

1 conflicting file src/index.js 1 conflict Prev ⌂ Next ⌄ Mark as resolved

index.js src/index.js

```
127      /* 演習の編集範囲 はじめ */
128
129      /* 演習1 ここから */
130
131      <<<<< A1
132      function add() {}
133      function sub() {}
134
135      =====
136      function mul() {}
137      function div() {}
138      >>>>> main
139      /* 演習1 ここまで */
```

矢印の3行を削除する

コード修正後

「Mark as resolves」をクリック

Conflict が発生しているファイルを全て修正する

Add add & sub #1

Resolving conflicts between A1 and main and committing changes → A1

1 conflicting file	src/index.js	1 conflict	Prev ^	Next v	⚙️	Mark as resolved
 index.js src/index.js	<pre>127 /* 演習の編集範囲 はじめ */ 128 129 130 /* 演習1 ここから */ 131 function add() {} 132 function sub() {} 133 function mul() {} 134 function div() {} 135 /* 演習1 ここまで */ 136</pre>					

修正を commit

全てのファイルを修正したか確認（青枠部分）

「Commit merge」ボタンから修正を commit

Add add & sub #1

Resolving conflicts between A1 and main and committing changes → A1

Commit merge

1 conflicting file ✓ Resolved

index.js src/index.js

```
src/index.js
127      /* 演習の編集範囲 はじめ */
128      function add() {}
129      function sub() {}
130      function mul() {}
131      function div() {}
132      /* 演習1 ここまで */
```

The screenshot shows a GitHub pull request interface for a merge between branches A1 and main. The title is 'Add add & sub #1'. The status bar indicates 'Resolving conflicts between A1 and main and committing changes → A1'. A green button labeled 'Commit merge' is highlighted with a red border. On the left, a table shows '1 conflicting file' with 'index.js' and 'src/index.js' listed, both marked as 'Resolved' with a checkmark icon. The main area displays the code for 'src/index.js' with conflict markers and a diff view.

ローカルで修正

1. 作業ブランチに merge 先のブランチを pull する (例: A1にmainをmerge)
2. Conflict 箇所を修正して commit/push
3. PR で merge 可能になったか確認

修正が完了したら

PR が merge できるか確認

必要なら再度レビューをもらう



チェックポイント その4

Conflict が起きないように作業分担しましょう

もしConflict は起きた場合は

- 小規模なら GitHub で修正
- 範囲が大きい場合は作業者がローカルで修正

上記のように対処するのがおすすめです

チーム演習

JavaScriptで電卓をつくろう



チーム開発

配布した状態のプログラムでは、
=ボタンを押したときの挙動が未実装です

Javascriptで電卓をつくろう

The image shows a digital calculator interface. At the top, there is a display showing '0'. Below the display is a row of four input fields separated by a '=' sign. The first field contains '0'. The second field is empty. The third field is empty. The fourth field is empty. Below this row is a red button labeled 'AC'. Below the 'AC' button is a 4x4 grid of buttons. The grid is arranged as follows: Row 1: 7, 8, 9, +; Row 2: 4, 5, 6, -; Row 3: 1, 2, 3, ×; Row 4: ., 0, =, ÷. The buttons for numbers (7, 8, 9, 4, 5, 6, 1, 2, 3, .) are gray, while the operators (+, -, ×, ÷) and the decimal point button (=) are blue.

演習

複数人で実装をする

YAHOO!
JAPAN

演習の進め方 1

演習の作業内容:

関数 **add** と **sub** を実装

1. 実装したい関数の枠組みを実装
2. 各関数の処理を実装
3. ローカルリポジトリを更新して、動作確認

※コード内容は、**code.txt** に記載されています

※各項目で、実装者とレビューを変えながら進めましょう

1. 実装したい関数の枠組みを実装

- **add**と**sub**を下記のハイライトされた部分のみ実装してPRを作成、レビューを行いマージする
- 作業用ブランチ: **create_func**

足し算用関数

```
function add() {}
```

引き算用関数

```
function sub() {}
```

実装場所

index.js にコードを実装していきます

```
127
128      /* 演習の編集範囲 はじめ */
129
130      /* 演習1 ここから */
131      function add() {}
132      function sub() {}
133      /* 演習1 ここまで */
134
```



131行目のコメント
アウトを削除して、
下記2つの関数を実装
します

- **add**関数
- **sub**関数

2. 各関数の処理を実装

- 1で実装した関数に処理を実装する
 - ① 各関数でブランチを作成 (**add**と**sub**)
 - ② 実装後push/PR作成
 - ③ レビューを行い、**add**→**sub**の順でマージする

add関数 実装コード

作業ブランチ名: add

ハイライトされた行を実装

```
function add(){
    eqlNum.value = parseFloat(midNum[0].value) + parseFloat(midNum[1].value);
}
```

sub関数 実装コード

作業ブランチ名: **sub**

ハイライトされた行を実装

```
function sub(){
    eqlNum.value = parseFloat(midNum[0].value) - parseFloat(midNum[1].value);
}
```

3. 動作確認

- 実装コードの動作確認
 - **index.html** をブラウザで開く
 - 電卓で足し算/引き算の動作確認

演習を一人で行うには

1. 作業ブランチ add/sub の作成
※どちらも main ブランチから作成する
2. 各ブランチでの実装を push/PR の作成
※merge をする前に両方とも push をする
3. PRの編集内容を確認後 merge

追加演習

電卓アプリの完成



追加演習

追加の作業内容:

関数 `mul` と `div` を実装して電卓アプリを完成させる

- 演習を参考にチーム内で役割を変えて作業する
- 進め方はチーム内で話し合って決めてください
- 余裕があればConflictを起こしてみましょう

※コード内容は、`code.txt` に記載されています

追加 実装コード

作業者C 掛け算用関数

```
function mul(){
    eqlNum.value = parseFloat(midNum[0].value) * parseFloat(midNum[1].value);
}
```

作業者D 割り算用関数

```
function div(){
    if(parseFloat(midNum[1].value) == 0){
        eqlNum.value = "ERROR";
    }else{
        eqlNum.value = parseFloat(midNum[0].value) / parseFloat(midNum[1].value);
    }
}
```

演習終了

YAHOO!
JAPAN

演習が終わったら

アンケート記入

お願いします

YAHOO!
JAPAN

演習が終わったら

追加演習とアンケートに回答まで終わった場合は、
残りの時間で下記の内容に取り組んでみてください

- 資料巻末の **Appendix** を行う
- Hack U の**開発体制のすり合わせ**
- 作業者を変えて同様の内容を行う
- 演習プログラムを改良してPR/レビューし合う

演習のポイント

YAHOO!
JAPAN

レビュアー

タイプミングミスや Conflict などをコメントで指摘

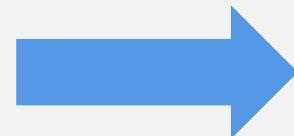
A screenshot of a GitHub pull request interface. At the top, there's a commit message "Add sub" with a timestamp "commented 19 seconds ago". Below the commit, a comment box contains the Japanese text "Conflict が発生しているため修正お願いします". This entire comment box is highlighted with a blue border. Below the comment, there's a general message "Add more commits by pushing to the B1 branch on magrain/github_team_ws.". A warning icon shows a branch with a conflict, and the text "This branch has conflicts that must be resolved". It suggests using the "web editor" or "command line" to resolve conflicts and provides a "Resolve conflicts" button. Under "Conflicting files", it lists "src/index.js". At the bottom, there are buttons for "Merge pull request" and "You can also open this in GitHub Desktop or view command line instructions."

Conflict 状態と解消

行数の黄色ハイライトが消えていることを確認
修正時にコードや改行の追加をすることも可能

修正前

```
---  
130      /* 演習1 ここから */  
131      <<<<< B1  
132      function sub() {}  
133      =====  
134      function add() {}  
135      >>>>> main  
136      /* 演習1 ここまで */  
137
```

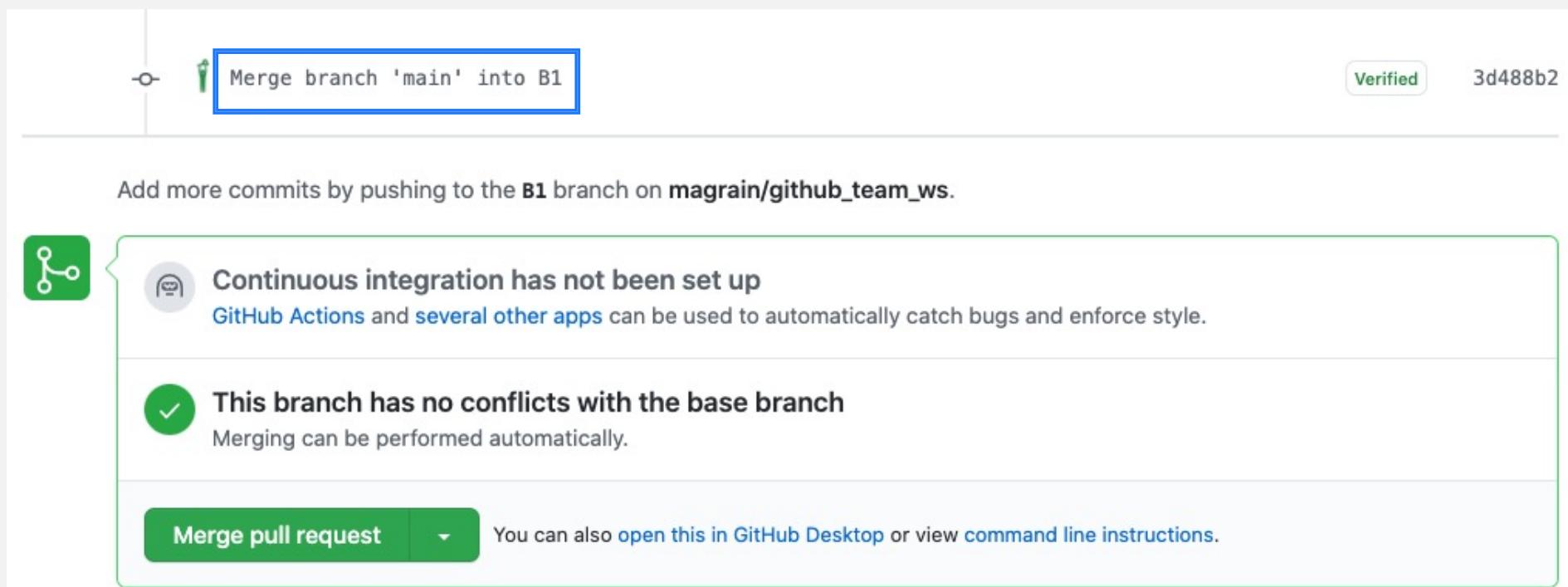


修正後

```
130      /* 演習1 ここから */  
131      function add() {}  
132  
133      function sub() {}  
134  
135      /* 演習1 ここまで */  
136
```

GitHubでのcommit

GitHub上で commit した場合、
commitメッセージは自動で記入されます



演習での Conflict

改行などがConflictする場合があります

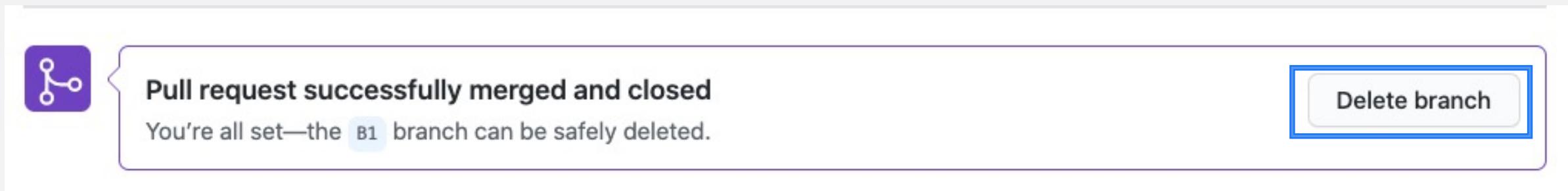
特に気にせずConflictを解消してください

```
---  
139 <<<<< B2  
140  
141 =====  
142 >>>>> main
```

不要なブランチの削除

使わなくなったブランチは削除が可能です

PR を merge 後に「Delete branch」から、
リモートブランチの削除できます



おさらい

YAHOO!
JAPAN

チーム開発のコード管理

- 複数人で同時にファイルを編集すると Conflict (競合) が発生する場合がある
- ブランチやPRによるレビューに用いて Conflict の解消や想定している実装を行う
- Conflict を起こさないような開発体制をチームで共有する

Hack U に向けて

GitHub はコード管理をアシストするツールです

-> 開発速度を求める際には使わない判断も重要

- コードレビュー
- アイディア出しや議事録をまとめる
- 完成したコードの共有

用途を限定して使うことも考えてみましょう

GitHub チーム開発編

以上



おまけ

- **Hack U をフォローしよう**

1. <https://github.com/hackujp> にアクセス
2. hackujp を Follow
3. hackujp/github_tutorial を Watch & Star

hackujp の他のリポジトリも確認してみてください
※どれも任意です

最新情報は connpass/Twitter(@hackujp) をフォロー

Appendix



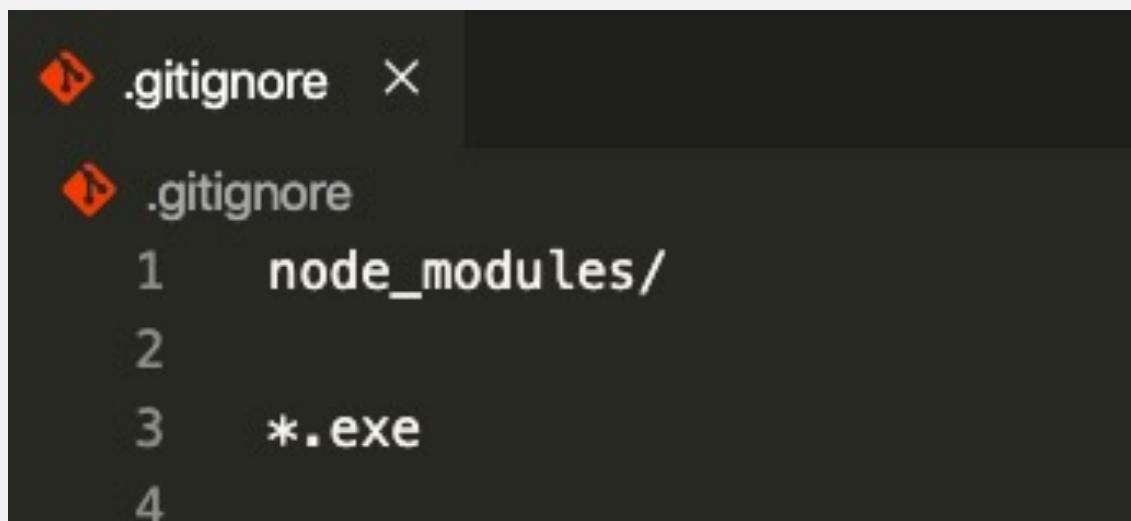
README.md とは

- **リポジトリで最初に見るべき資料**
 - 使用方法
 - 開発ルール
 - 変更履歴 ...など
- **自分のリポジトリの説明を追加しましょう**
 - Markdown記法で記述できます

.gitignore ファイル

- 「.gitignore」ファイルは、Gitの管理に含めないファイルを指定するためのファイルです
- npm における node_modules ディレクトリや実行ファイルなど、開発時に自動生成されるディレクトリは Git に push する必要がない場合が度々あるため、「.gitignore」ファイルを使うと簡単に管理対象から外すことができます
- 詳しくは調べてみてください

.gitignoreの書き方の例



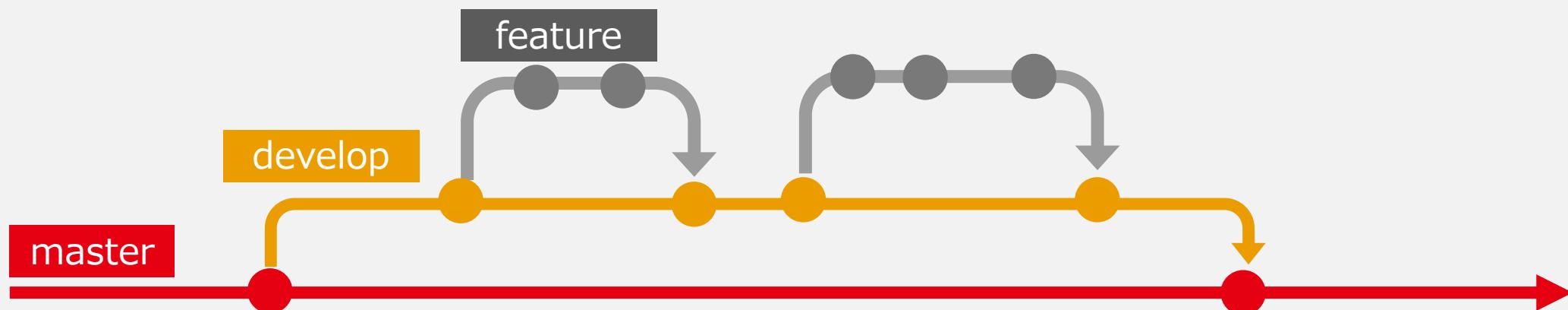
```
1 .gitignore
2
3 node_modules/
4 *.exe
```

ロールバック

- リポジトリに commit や pushをした後に、間違いなどに気づいた場合、前の状態に戻したいときがあります。
- そのような時に「git reset」や「git revert」などのコマンドを使います
- それぞれのコマンドは場合に応じてオプションを使い分けたりするなど、説明が煩雑になるためここでは省きます
- 詳しい使い方は、各自で調べてみてください。

ブランチの使い方

- ・ ブランチは、チームや開発する内容によって使い方を変えたほうがよい場合があります。調べてみると面白いと思います。
- ・ しかし一番大事なことは、チーム全員でブランチの使い方にについて認識があっているということです。



コードレビュー

- Pull Request は、最低1人は他の人にみてもらって Approve をもらってからマージするようにし、
基本的にはセルフマージ（チェックなしのマージ）は
しないようにしましょう。
- レビューを行う場合は、コードに対するコメント機能をうま
く使うと伝わりやすいです。
- コードレビューの行いかたも、チームである程度ルールを決
めておくとよいかもしれません。

Gitコマンドでの操作



Gitコマンドでの操作

- add

```
$ git add ファイルパス
```

- commit

```
$ git commit -m "commitコメント"
```

Gitコマンドでの操作

- ブランチ作成
\$ git branch ブランチ名
- 今のブランチを確認
\$ git branch
- ブランチの切り替え
\$ git checkout ブランチ名
- ブランチの作成 + 切り替え
\$ git checkout -b ブランチ名

Gitコマンドでの操作

- clone

```
$ git clone リポジトリURL
```

- pull

```
$ git pull origin pullしたいブランチ名
```

- push

```
$ git push origin pushしたいブランチ名
```

Gitコマンドでの操作

- “origin” とは？
 - リモートリポジトリのURLを指している
 - 毎回打つのが面倒なので“origin”という略称で代用できるように設定されている

例) これらは同じ内容を表す

```
$ git pull origin main
```

```
$ git pull git@github.com:ユーザ名/リポジトリ名.git main
```