

VSCode の Git 機能 を 利用したチーム開発手法

1. 導入編

Git・Github・VSCodeのGit機能について知ろう！

1-1. Gitとは何か？

Gitとは何か？

“ Git（ギット）は、プログラムのソースコードなどの変更履歴を記録・追跡するための分散型バージョン管理システムである。

出典: <https://ja.wikipedia.org/wiki/Git> ”

分散型……バージョン管理システム……？

ゲームのセーブデータみたいに
今の状態を保存しておくもの
これをリポジトリ、と呼びます

作業が一段落したら、 今の状態を保存します

履歴が溜まっていきますね
すると……？

```
* commit 0a3390048cecd60b09f74ae0108c41425823e572 (HEAD -> featu
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:55:15 2023 +0900

    Update: 改行が無かったので追加

* commit 0b3e9504485cefa8a5392cab495ea922b6ec448a
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 15:16:53 2023 +0900

    Move: 命名を変更

* commit d9a89524f7e21d00f4bc232dd211cb8f84c153a5
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 15:11:16 2023 +0900

    Remove: .DS_Storeが追加されてしまっていたので削除

* commit cb4d7e8827b0f854d8c9d91d3308c3ee059a7f08
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:53:06 2023 +0900

    Update: <br>を削除

* commit d34628768c833533172f3b85f3daf5b9e0ac3d36
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:47:16 2023 +0900

    Update: mermaidによるGit図解を追加

* commit dfcf7fa4d632543aff79f323074f237832ec1b09
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:39:39 2023 +0900

    Update: Github Desktopハンズオンの内容を記載

* commit afe159403e4a7b83491319bcdcd68eb03ff76154
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Tue Jun 13 16:24:58 2023 +0900

    Update: Github Desktopを利用したチーム開発手法、の本編の基

* commit 4d89428a91675ecbcb0f9850ed153c7ed9d03d1c
:
```

いつ、何の機能を実装したのか分かる

履歴には説明文をつけるので、
分かりやすいです

```
* commit 0a3390048cecd60b09f74ae0108c41425823e572 (HEAD -> featu
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:55:15 2023 +0900

    Update: 改行が無かったので追加

* commit 0b3e9504485cefa8a5392cab495ea922b6ec448a
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 15:16:53 2023 +0900

    Move: 命名を変更

* commit d9a89524f7e21d00f4bc232dd211cb8f84c153a5
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 15:11:16 2023 +0900

    Remove: .DS_Storeが追加されてしまっていたので削除

* commit cb4d7e8827b0f854d8c9d91d3308c3ee059a7f08
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:53:06 2023 +0900

    Update: <br>を削除

* commit d34628768c833533172f3b85f3daf5b9e0ac3d36
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:47:16 2023 +0900

    Update: mermaidによるGit図解を追加

* commit dfcf7fa4d632543aff79f323074f237832ec1b09
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:39:39 2023 +0900

    Update: Github Desktopハンズオンの内容を記載

* commit afe159403e4a7b83491319bcdcd68eb03ff76154
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Tue Jun 13 16:24:58 2023 +0900

    Update: Github Desktopを利用したチーム開発手法、の本編の基

* commit 4d89428a91675ecbcb0f9850ed153c7ed9d03d1c
:
```

バグが発生した時も、 元に戻せる

過去のデータに、一瞬で元通り

```
* commit 0a3390048cecd60b09f74ae0108c41425823e572 (HEAD -> featu
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:55:15 2023 +0900

    Update: 改行が無かったので追加

* commit 0b3e9504485cefa8a5392cab495ea922b6ec448a
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 15:16:53 2023 +0900

    Move: 命名を変更

* commit d9a89524f7e21d00f4bc232dd211cb8f84c153a5
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 15:11:16 2023 +0900

    Remove: .DS_Storeが追加されてしまっていたので削除

* commit cb4d7e8827b0f854d8c9d91d3308c3ee059a7f08
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:53:06 2023 +0900

    Update: <br>を削除

* commit d34628768c833533172f3b85f3daf5b9e0ac3d36
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:47:16 2023 +0900

    Update: mermaidによるGit図解を追加

* commit dfcf7fa4d632543aff79f323074f237832ec1b09
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:39:39 2023 +0900

    Update: Github Desktopハンズオンの内容を記載

* commit afe159403e4a7b83491319bcdcd68eb03ff76154
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Tue Jun 13 16:24:58 2023 +0900

    Update: Github Desktopを利用したチーム開発手法、の本編の基

* commit 4d89428a91675ecbcb0f9850ed153c7ed9d03d1c
:
```

変なデータ、必要なし

- program.c
- program(1).c
- program(2).c
- program(3)_これが最終版.c
- program_提出用.c

```
* commit 0a3390048cecd60b09f74ae0108c41425823e572 (HEAD -> featu
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:55:15 2023 +0900

    Update: 改行が無かったので追加

* commit 0b3e9504485cefa8a5392cab495ea922b6ec448a
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 15:16:53 2023 +0900

    Move: 命名を変更

* commit d9a89524f7e21d00f4bc232dd211cb8f84c153a5
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 15:11:16 2023 +0900

    Remove: .DS_Storeが追加されていたので削除

* commit cb4d7e8827b0f854d8c9d91d3308c3ee059a7f08
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:53:06 2023 +0900

    Update: <br>を削除

* commit d34628768c833533172f3b85f3daf5b9e0ac3d36
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:47:16 2023 +0900

    Update: mermaidによるGit図解を追加

* commit dfcf7fa4d632543aff79f323074f237832ec1b09
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:39:39 2023 +0900

    Update: Github Desktopハンズオンの内容を記載

* commit afe159403e4a7b83491319bcdcd68eb03ff76154
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Tue Jun 13 16:24:58 2023 +0900

    Update: Github Desktopを利用したチーム開発手法、の本編の基
:
* commit 4d89428a91675ecbcb0f9850ed153c7ed9d03d1c
```

その上、複数人で作業 するための機能も沢山

編集の重複を防いだり、差分を
分かりやすくしたり……

```
* commit 0a3390048cecd60b09f74ae0108c41425823e572 (HEAD -> featu
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:55:15 2023 +0900

    Update: 改行が無かったので追加

* commit 0b3e9504485cefa8a5392cab495ea922b6ec448a
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 15:16:53 2023 +0900

    Move: 命名を変更

* commit d9a89524f7e21d00f4bc232dd211cb8f84c153a5
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 15:11:16 2023 +0900

    Remove: .DS_Storeが追加されてしまっていたので削除

* commit cb4d7e8827b0f854d8c9d91d3308c3ee059a7f08
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:53:06 2023 +0900

    Update: <br>を削除

* commit d34628768c833533172f3b85f3daf5b9e0ac3d36
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:47:16 2023 +0900

    Update: mermaidによるGit図解を追加

* commit dfcf7fa4d632543aff79f323074f237832ec1b09
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:39:39 2023 +0900

    Update: Github Desktopハンズオンの内容を記載

* commit afe159403e4a7b83491319bcdcd68eb03ff76154
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Tue Jun 13 16:24:58 2023 +0900

    Update: Github Desktopを利用したチーム開発手法、の本編の基

* commit 4d89428a91675ecbcb0f9850ed153c7ed9d03d1c
:
```

Gitとは何か？

ソースコードをいい感じに管理するやつ

1-2. Githubとは何か？

Githubとは何か？

“ GitHub（ギットハブ）は、ソフトウェア開発のプラットフォームであり、ソースコードをホスティングする。コードのバージョン管理システムにはGitを使用する。

出典: <https://ja.wikipedia.org/wiki/GitHub> ”

ソースコードを……ホスティングする……？

つまり？

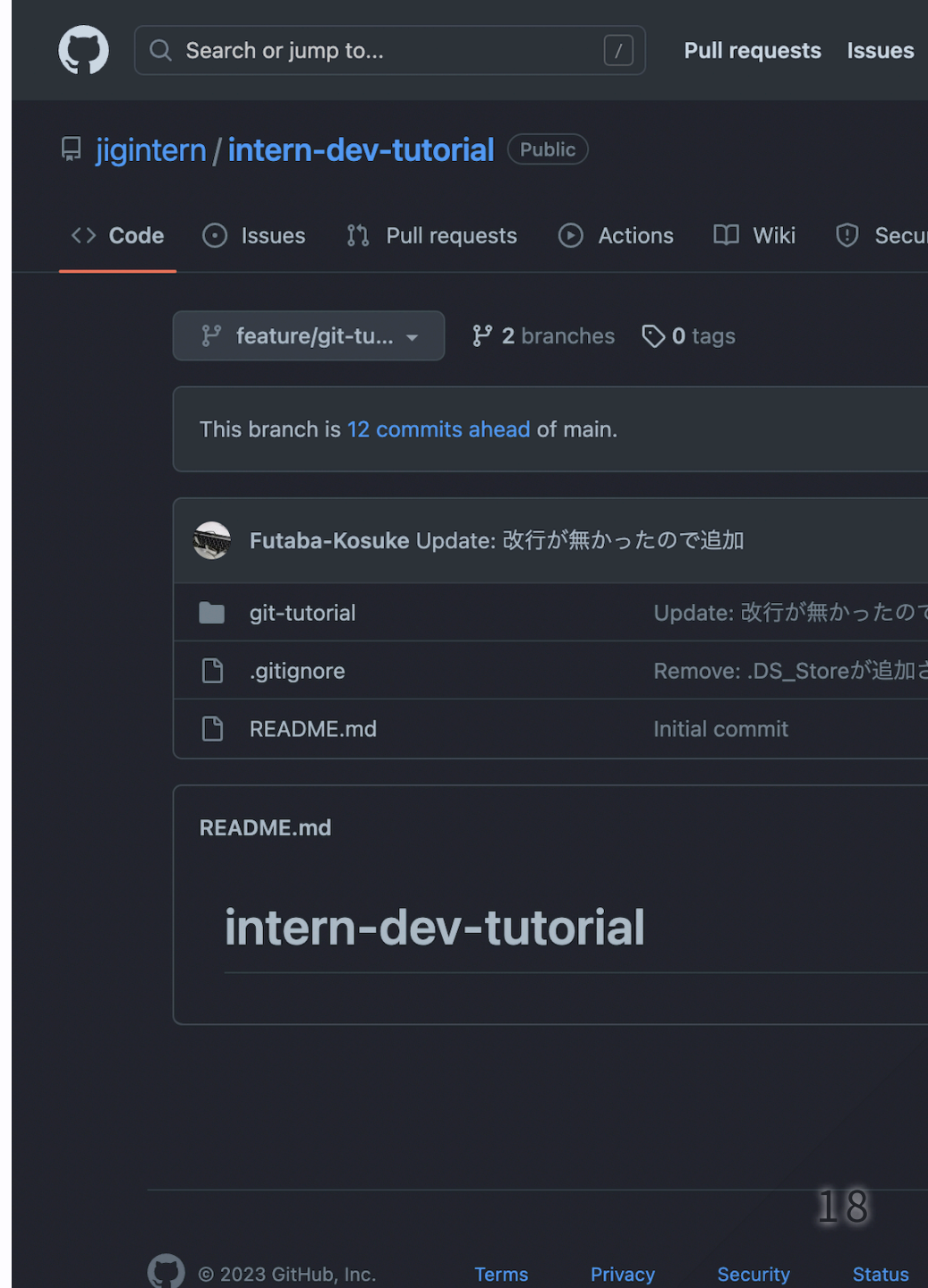
Gitをオンラインで管理するやつ

Gitのデータをアップロードして共有・バックアップするためのサービスです

<https://github.com/>

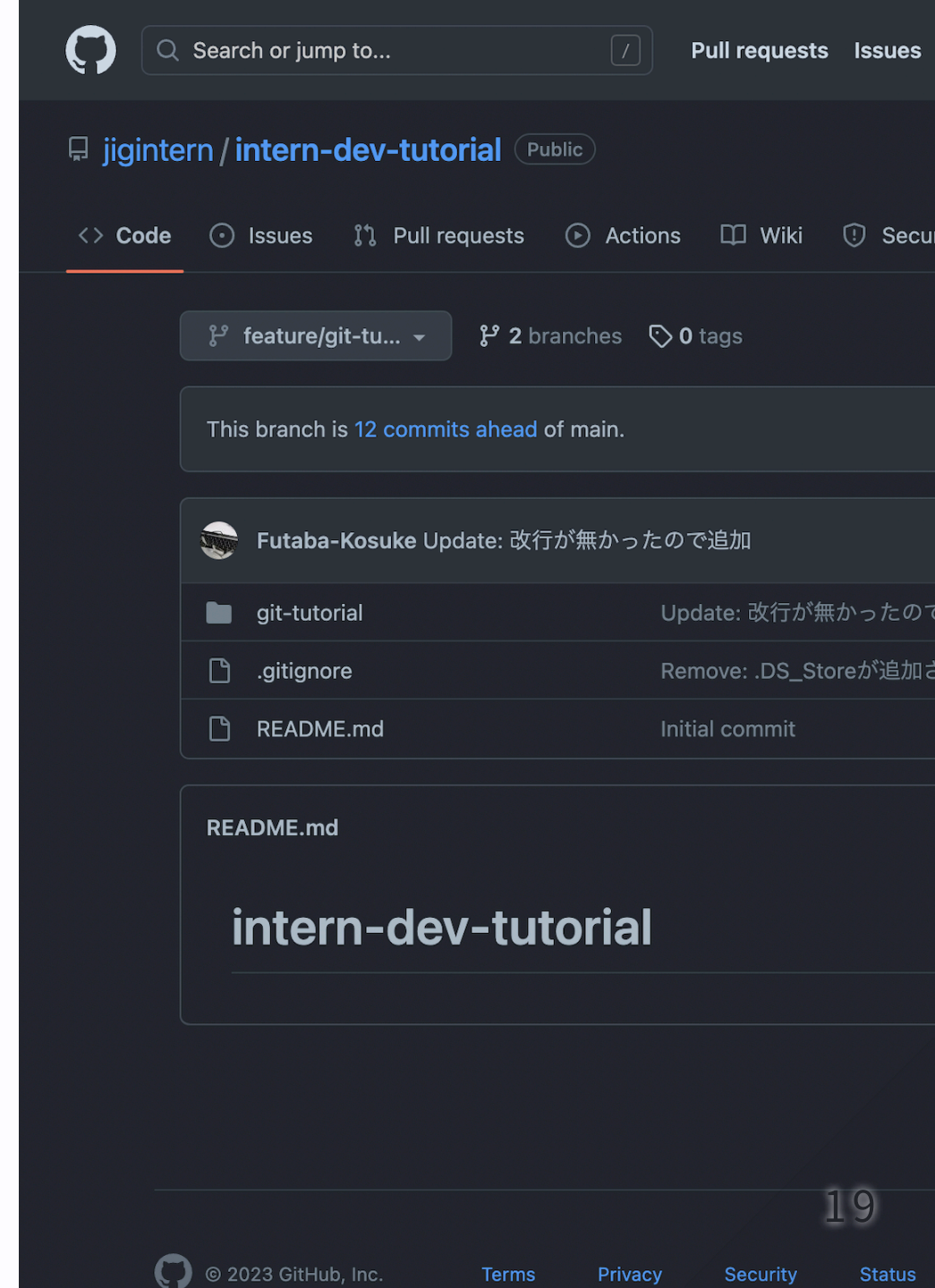
記録がある程度溜まったら、アップロード

PC上のデータが吹っ飛んでも、
オンラインに残ります



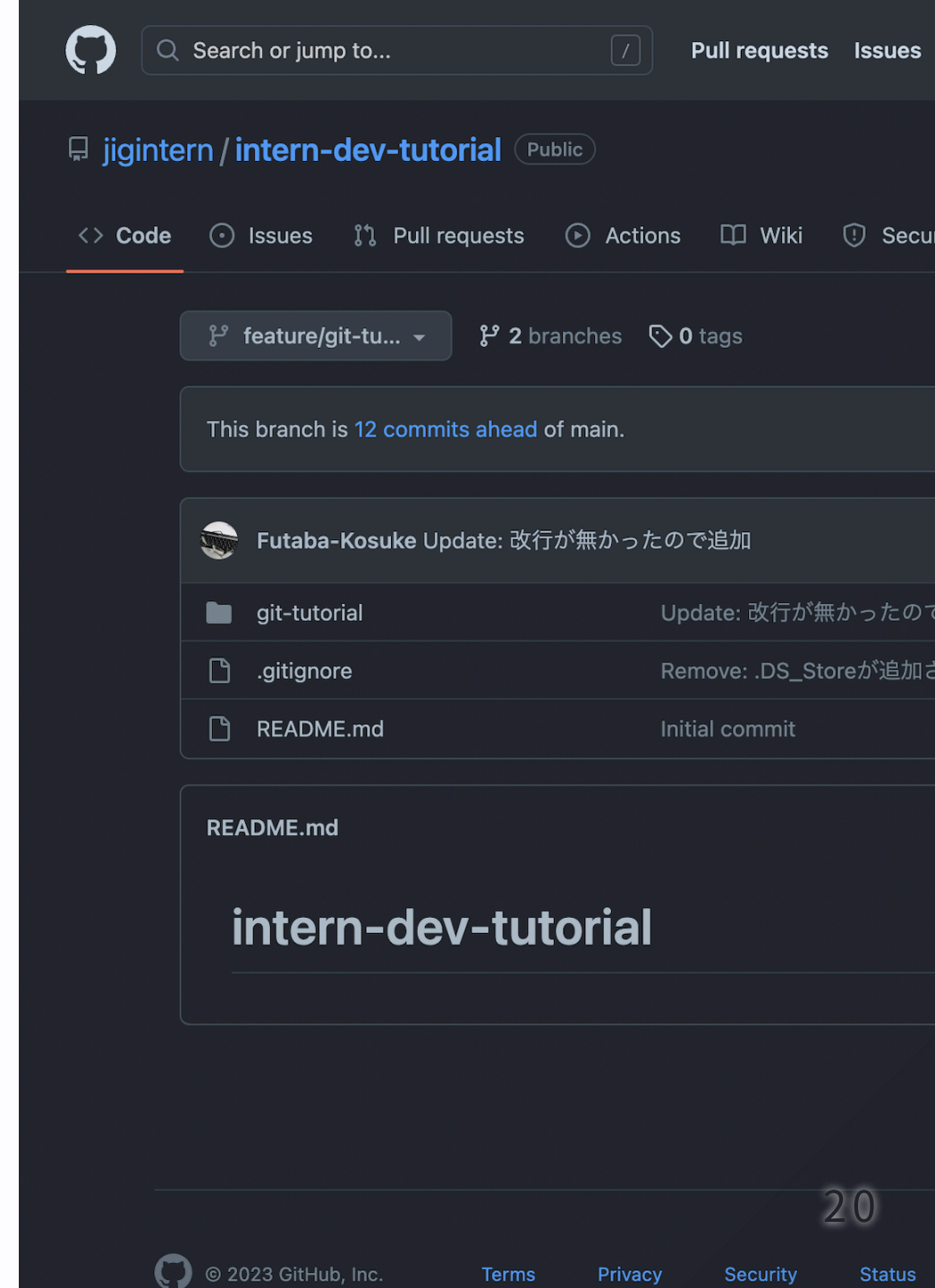
複数人で、データ共有

クラウドでの共有は不要です



他の人も、閲覧可能

共同開発の場としても、能力を
アピールする場としても



Githubとは何か？

gitをオンラインで管理するやつ

1-3. VSCodeのGit機能とは何か？

VSCodeのGit機能とは何か？

“ Visual Studio Code には統合されたソース管理機能 (SCM)があり、Git のサポートが標準で組み込まれています。

出典:

<https://code.visualstudio.com/docs/sourcecontrol/overview>

”

Gitは、コマンドライン で操作するのが一般的



```
* commit 0a3390048cecd60b09f74ae0108c41425823e572 (HEAD -> featu
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:55:15 2023 +0900

    Update: 改行が無かったので追加

* commit 0b3e9504485cefa8a5392cab495ea922b6ec448a
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 15:16:53 2023 +0900

    Move: 命名を変更

* commit d9a89524f7e21d00f4bc232dd211cb8f84c153a5
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 15:11:16 2023 +0900

    Remove: .DS_Storeが追加されてしまっていたので削除

* commit cb4d7e8827b0f854d8c9d91d3308c3ee059a7f08
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:53:06 2023 +0900

    Update: <br>を削除

* commit d34628768c833533172f3b85f3daf5b9e0ac3d36
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:47:16 2023 +0900

    Update: mermaidによるGit図解を追加

* commit dfcf7fa4d632543aff79f323074f237832ec1b09
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Wed Jun 14 14:39:39 2023 +0900

    Update: Github Desktopハンズオンの内容を記載

* commit afe159403e4a7b83491319bcdcd68eb03ff76154
Author: niba2828 <k.futaba2828@gmail.com>
Date:   Tue Jun 13 16:24:58 2023 +0900

    Update: Github Desktopを利用したチーム開発手法、の本編の基
:
* commit 4d89428a91675ecbcb0f9850ed153c7ed9d03d1c
```

VSCodeのソース管理ビューを使えば、コ マンド無しで操作できます

拡張機能のインストールも不要！
VSCodeを入れればすぐ使えます
(Git本体は事前準備でインストール済み)

今回はVSCodeのGit機能を使って開発します

お気に入りのGitの操作方法があれば、そちらで進めていただいても問題ありません！

2. VSCodeのGit機能を触ってみよう

VSCodeのGit機能を実際に使ってみよう！

2-1. clone: Githubリポジトリを丸ごとダウンロードしよう

Githubからリポジトリをダウンロードして みましょう

これをclone（クローン）と呼びます

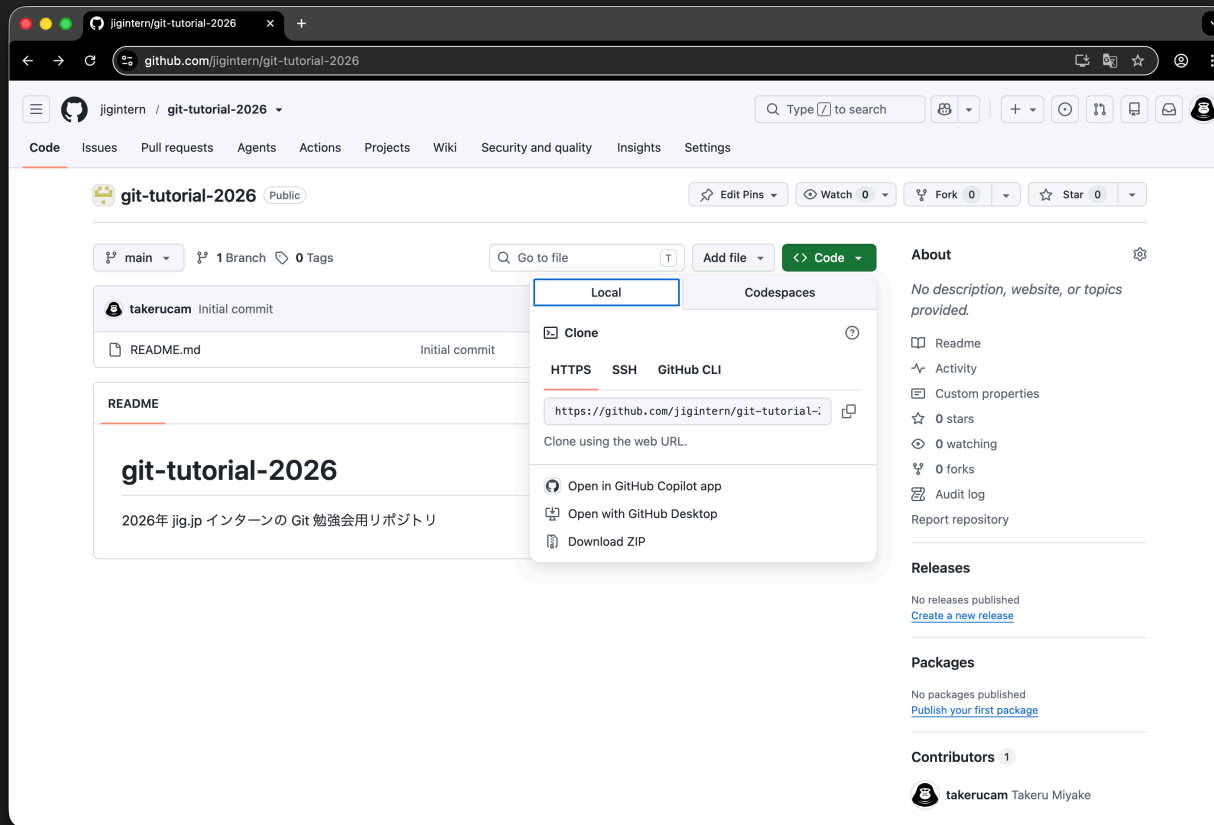
今回はこちらで事前にリポジトリを作成しました

ブラウザで以下URLを開いてください

Zoomのチャット欄にも記載します

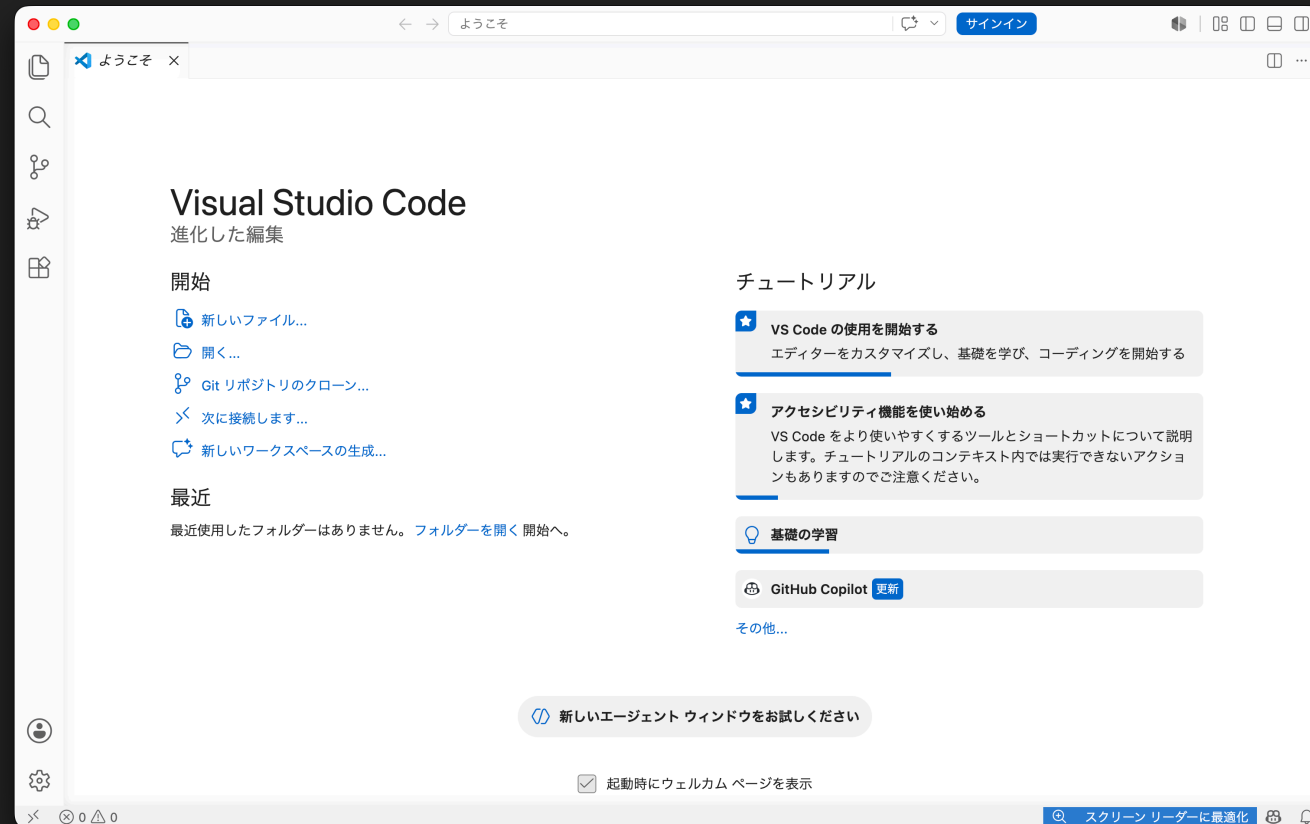
<https://github.com/jigintern/git-tutorial-2026>

1. 「<> Code」 ボタンをクリックし、表示されたHTTPSのURLをコピーします



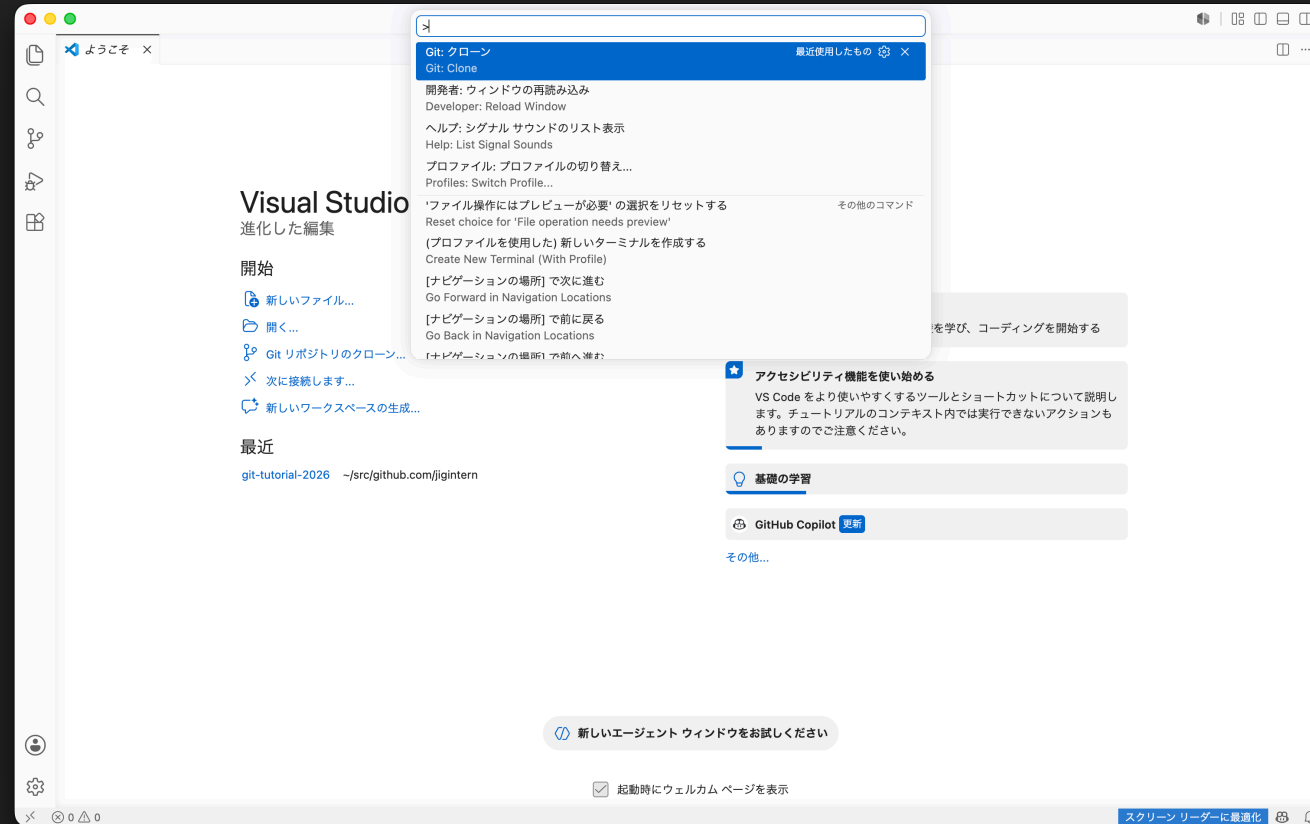
2-1. clone: Githubリポジトリを丸ごとダウンロードしよう

2. 「ようこそ」画面の「Git リポジトリのクローン...」をクリックします



2-1. clone: Githubリポジトリを丸ごとダウンロードしよう

ex. コマンドパレットから「Git: クローン」を選んでも、同じ操作ができます

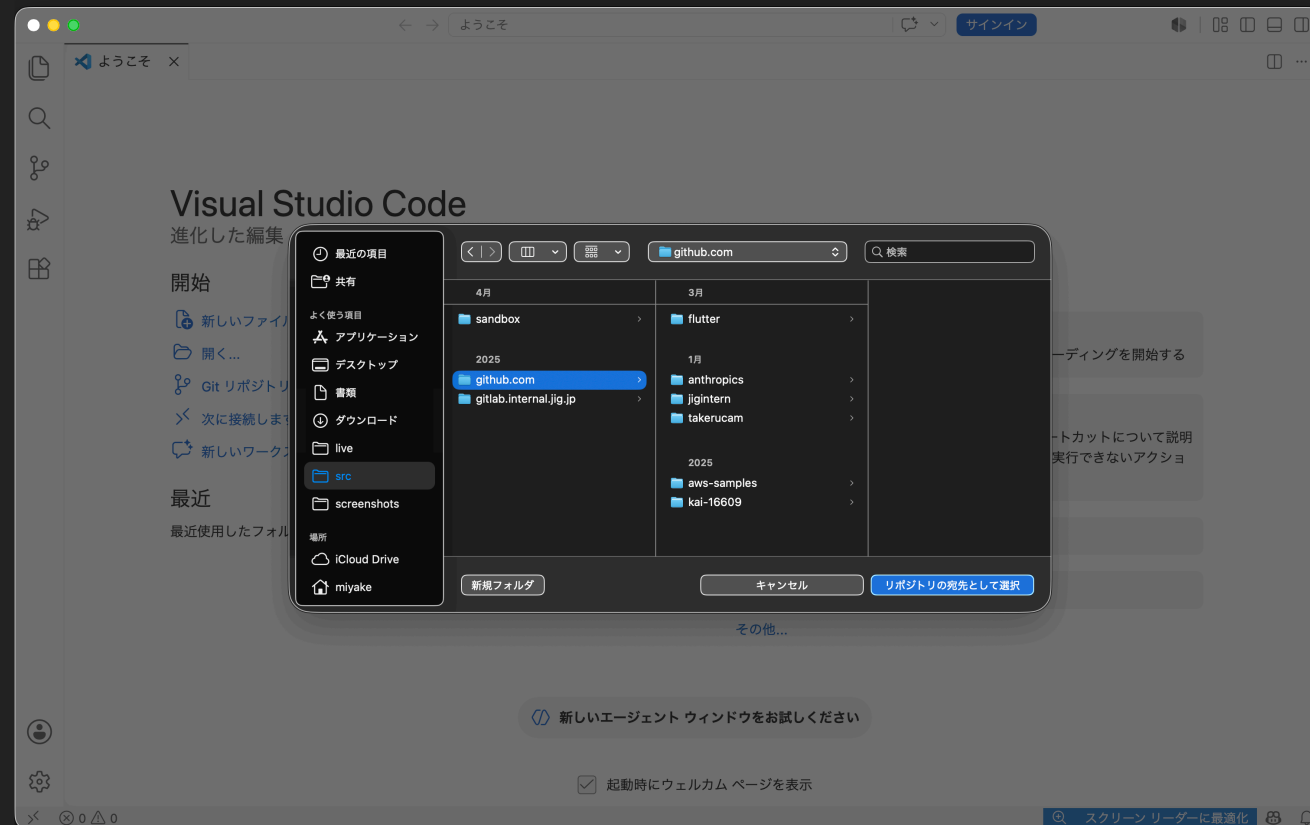


3. 画面上部の入力欄に、コピーしたURLを貼り付けて、Enterキーを押します

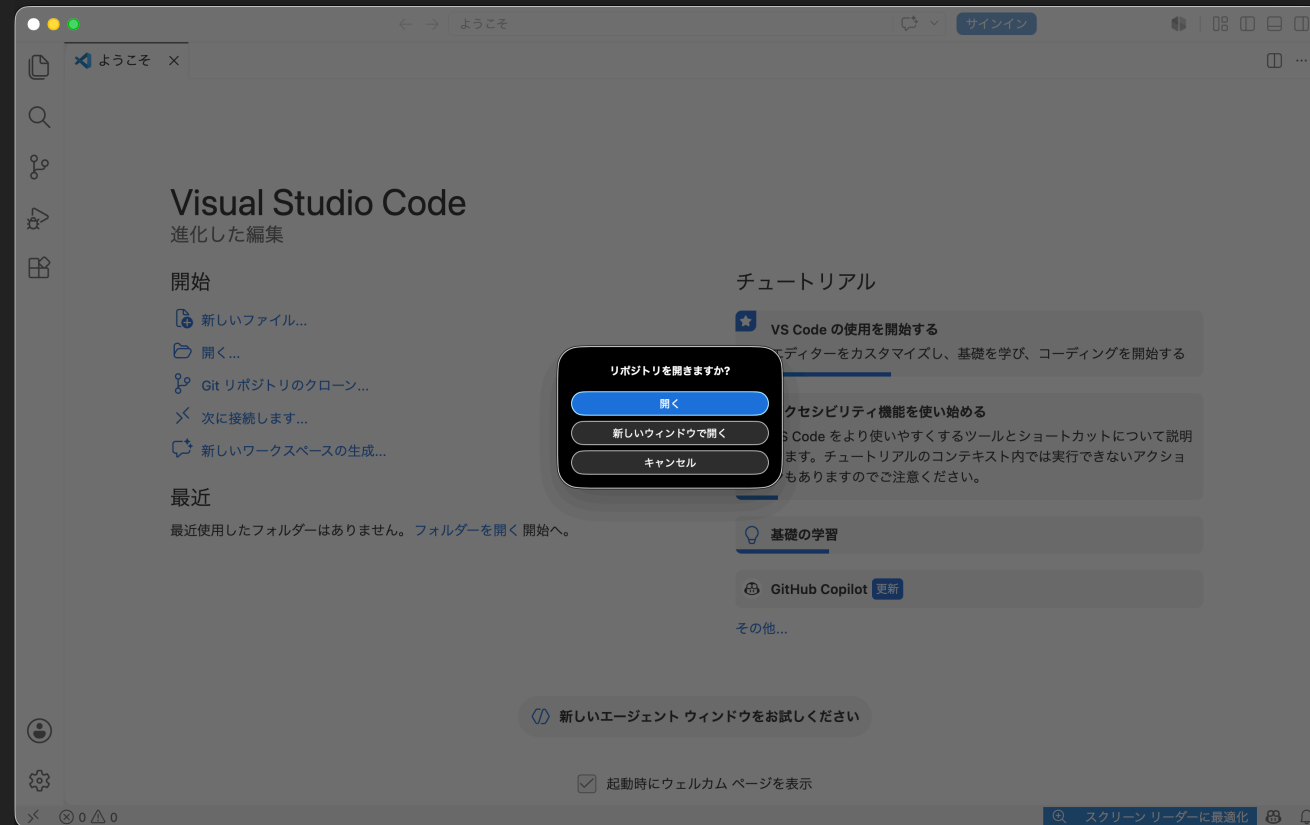


2-1. clone: Githubリポジトリを丸ごとダウンロードしよう

4. PC上での保存先のフォルダーを選択します

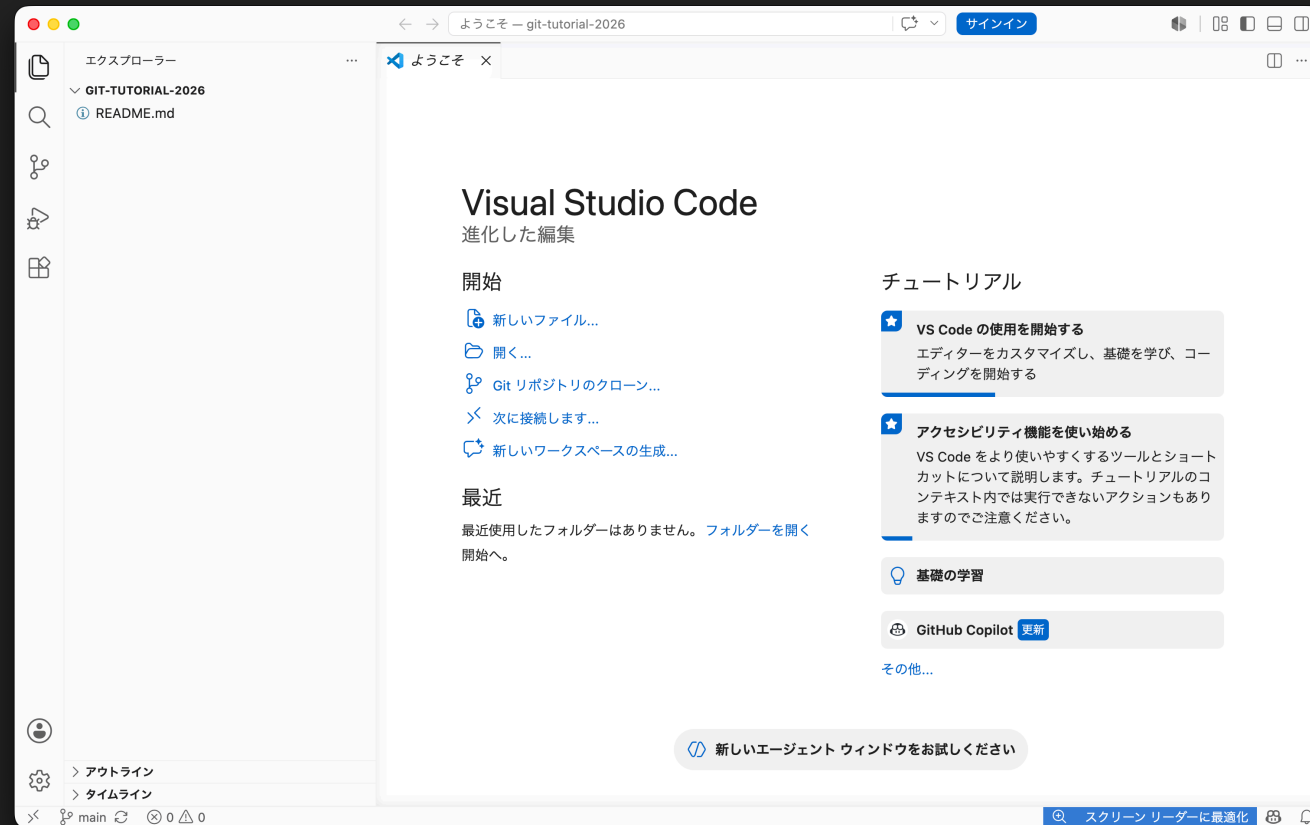


5. 「クローンしたリポジトリを開きますか？」という通知が表示されるので、「開く」をクリックします



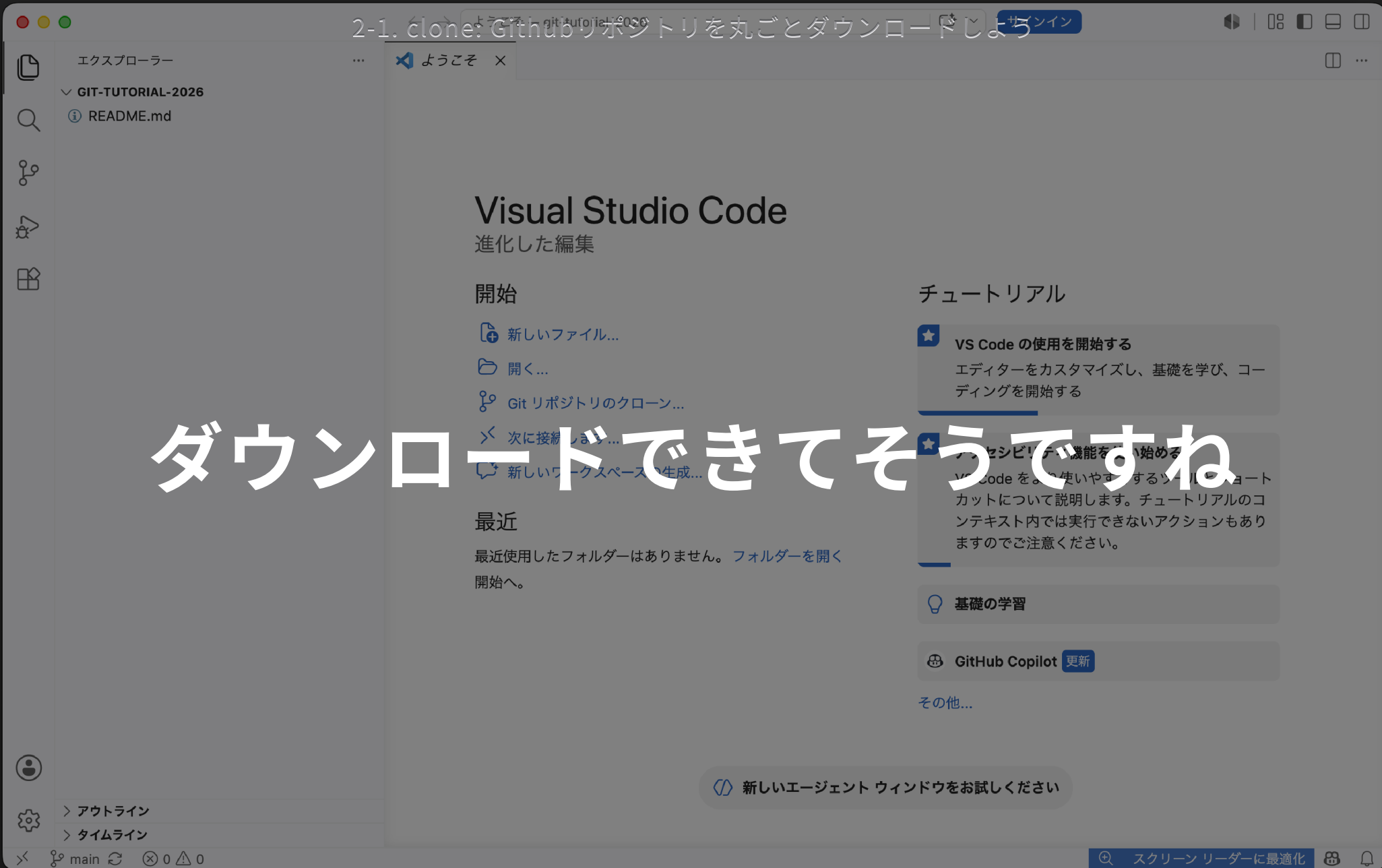
2-1. clone: Githubリポジトリを丸ごとダウンロードしよう

6. エクスプローラーに、Githubにアップロードされていたファイルが表示されていることを確認します



2-1. clone: Githubリポジトリを丸ごとダウンロードしよう

ダウンロードできてそうですね



2-2. branch: 作業を枝分かれさせよう

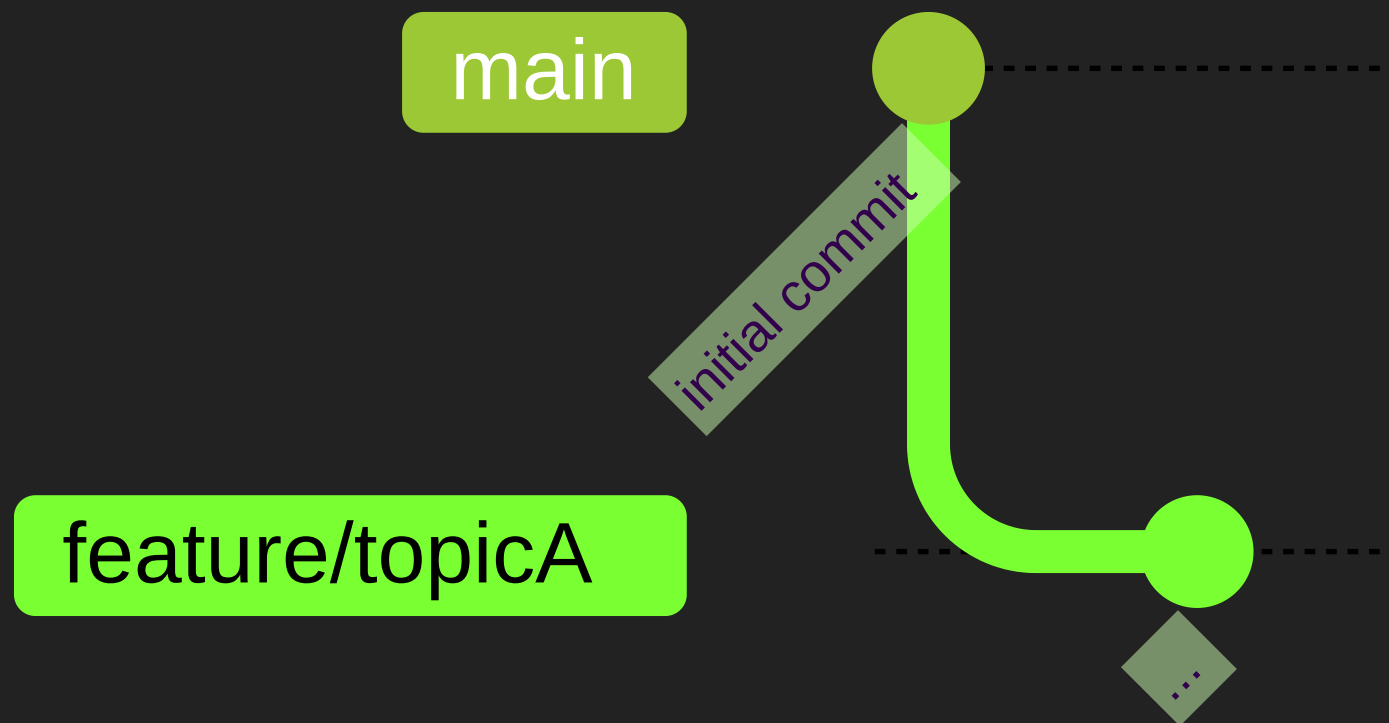
作業を枝分かれさせてみましょう

これをbranch（ブランチ）と呼びます

クラウドサービスでファイル共有する と……

編集が重複したりして、片方の編集内容が消されることも

枝分かれすると、複数人で同じファイルを触っても（ちょっと）安心です



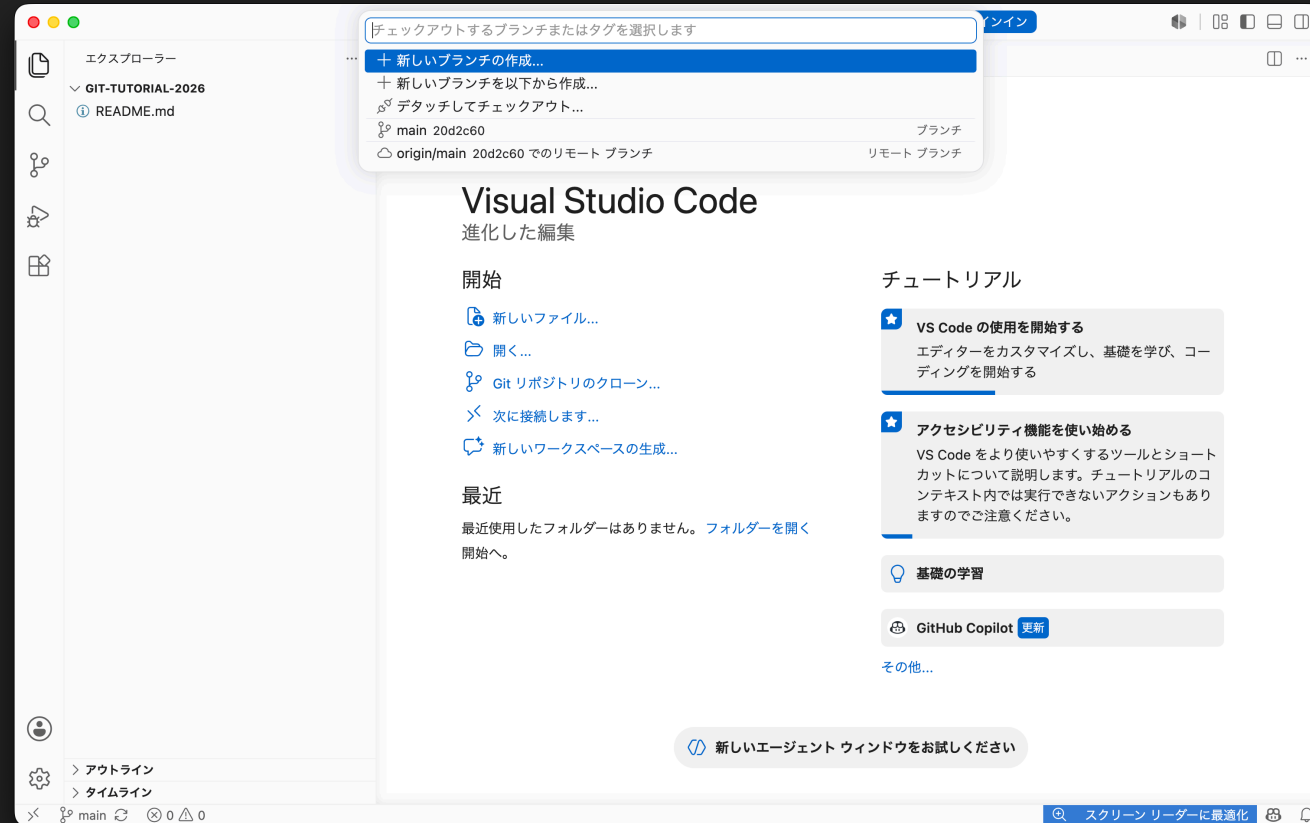
では、ブランチを切ってみましょう

ブランチを作ることを、慣習的に「ブランチを切る」、と
いいます

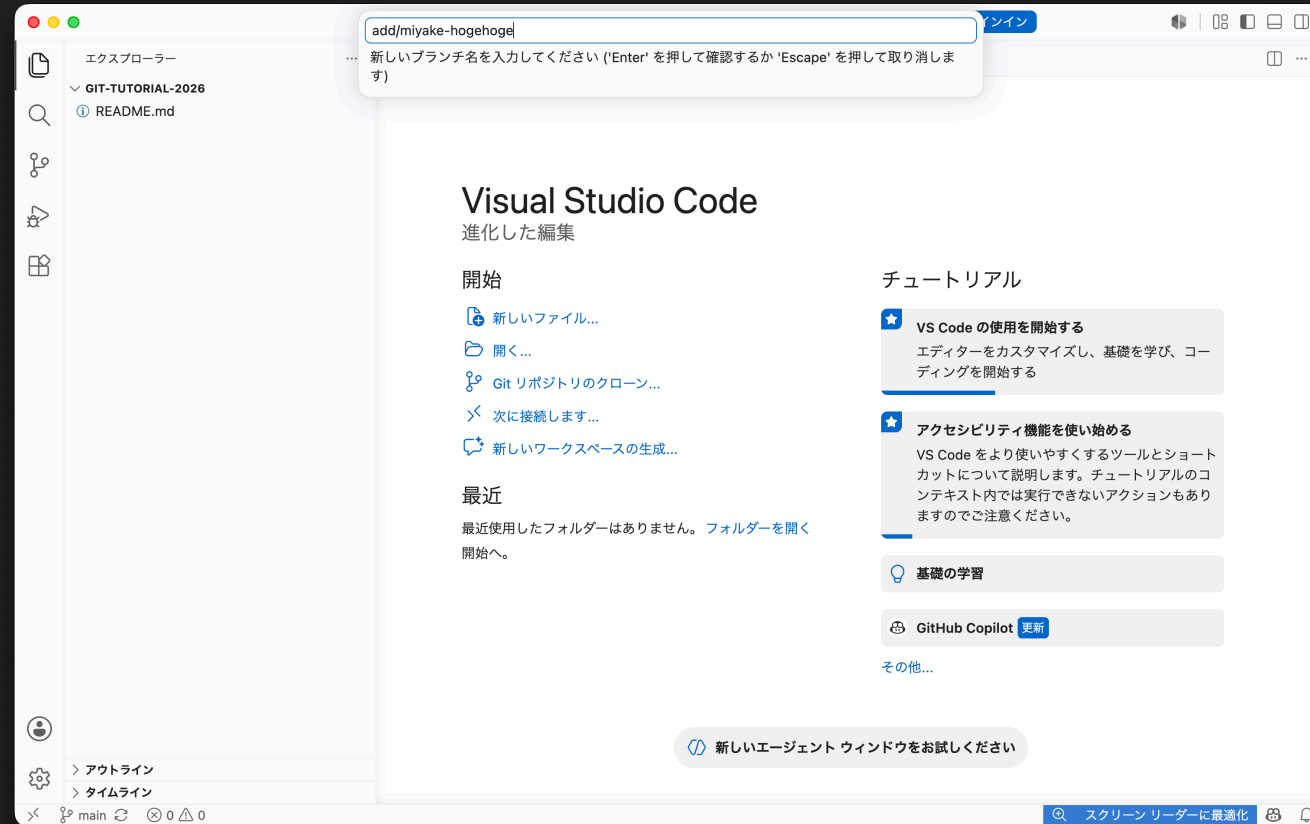
1. 画面左下のステータスバーに表示されているブランチ名（main）をクリックします



2. 表示されるメニューから「+ 新しいブランチの作成... (Create new branch...)」をクリックします



3. 新規ブランチの名前を入力して、Enterキーを押します。他の参加者と重複しないよう、自分の名前などで設定してください



4. ステータスバーの表示が、新しいブランチ名に切り替わったことを確認します



既にファイルに変更を加えていた場合、その変更は新しいブランチにそのまま持ち越されます

2-3. commit: 作業前と後の差分を記録しよう

作業前と後の**差分を記録**してみましよう

これをcommit（コミット）と呼びます

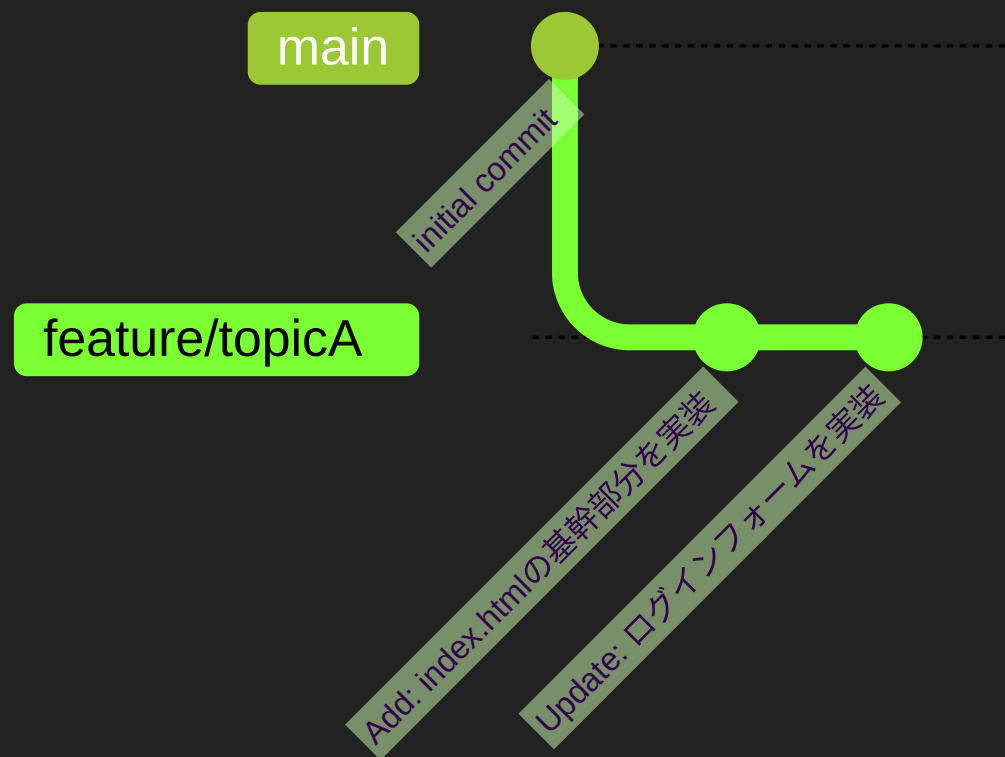
Gitはファイルの**現在の内容を保存**することを繰り返してログを作ると言いましたが.....

実は、全てを保存するわけではありません

作業を重ねる程に**データが膨大**になってしまします

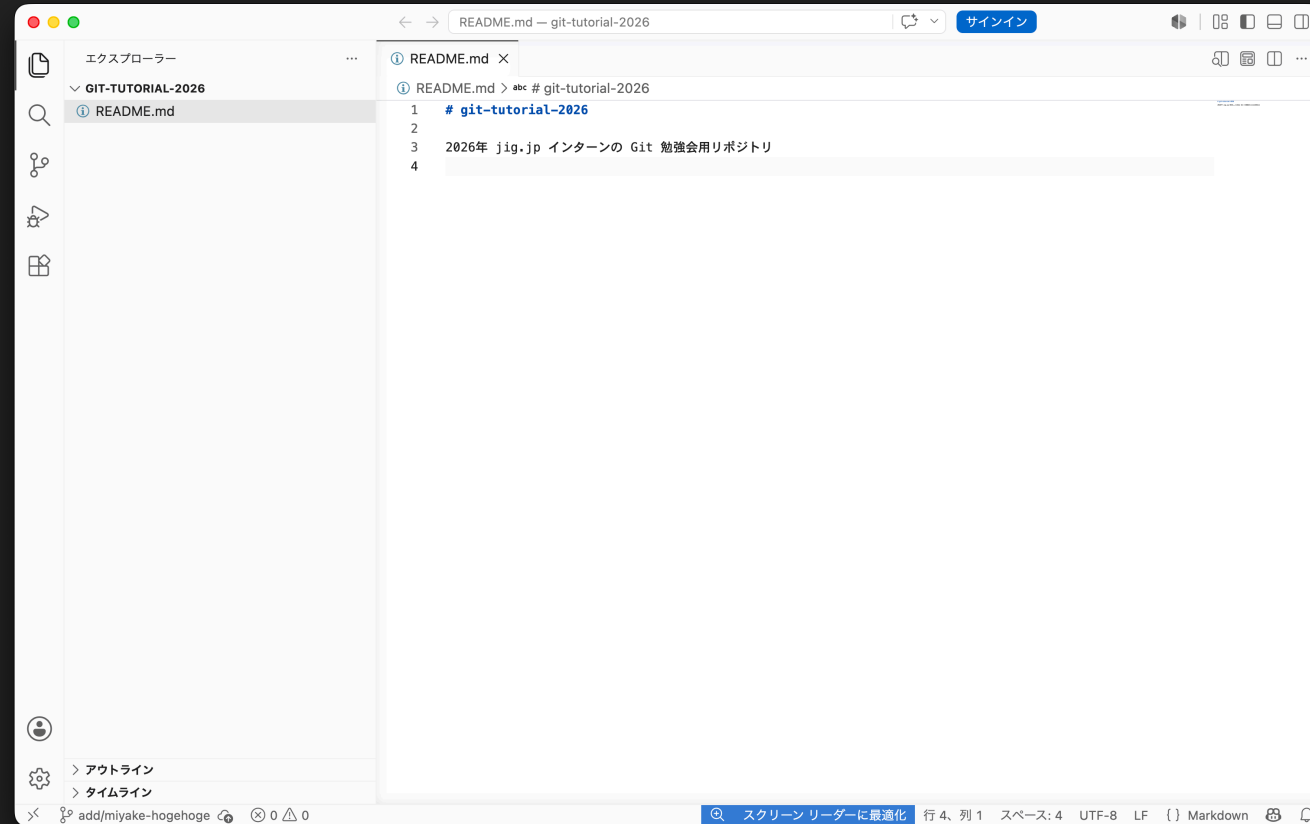
これでは、いけませんね。しかし、解決する方法があります

データの保存時には、作業前と後の差分を記録する



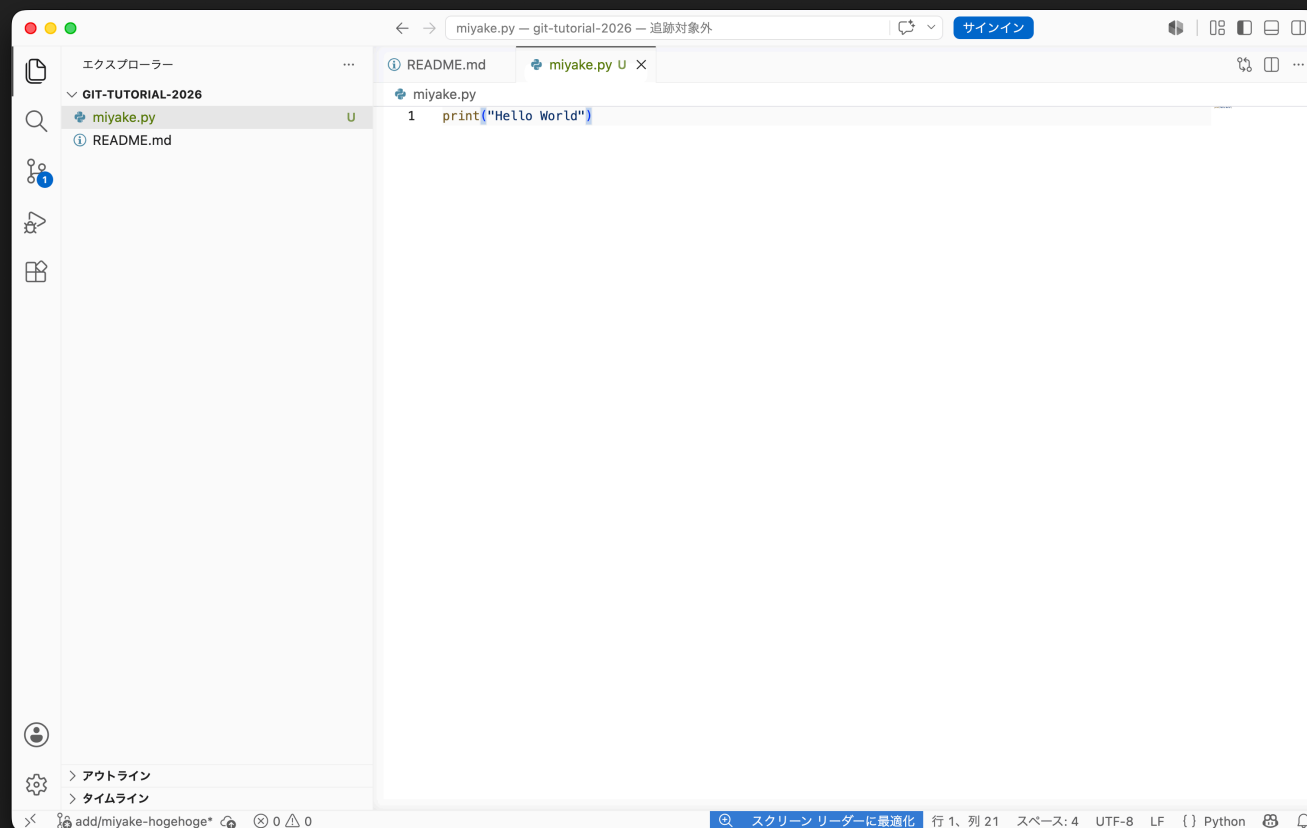
では、コミットしてみましよう

1. ファイルを新規作成します。ファイル名が重複しないよう、自分の名前などを半角英数字でつけてください



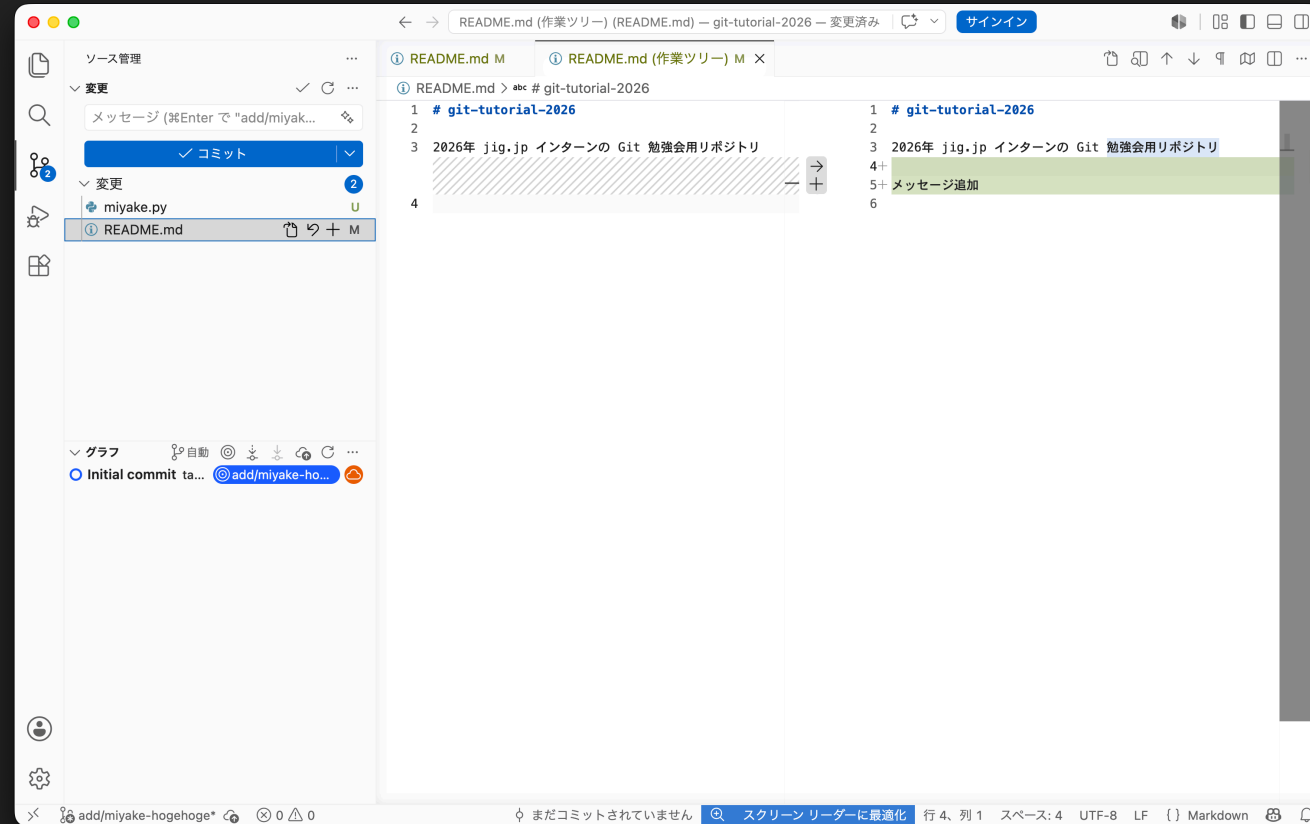
2-3. commit: 作業前と後の差分を記録しよう

2. 作成したファイルに、適当なプログラムを書き込みます

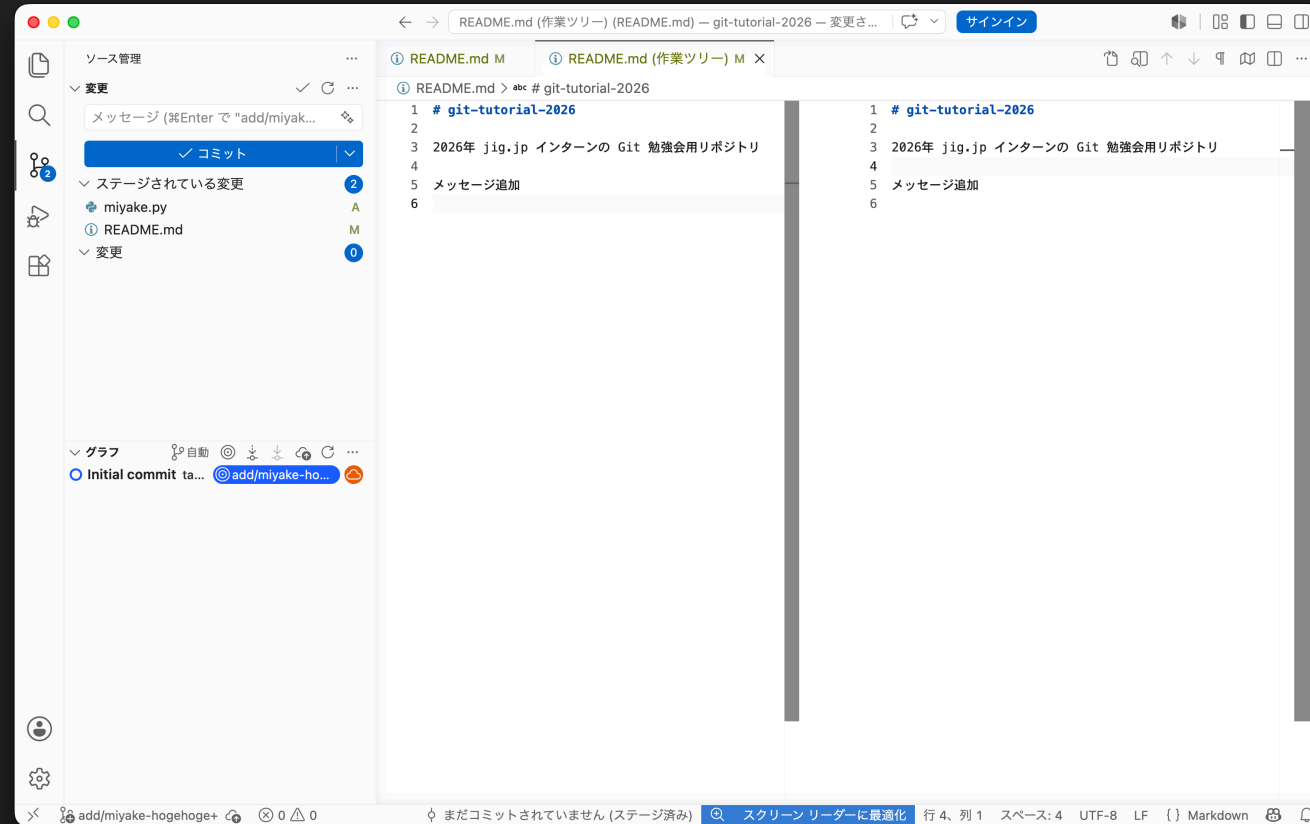


2-3. commit: 作業前と後の差分を記録しよう

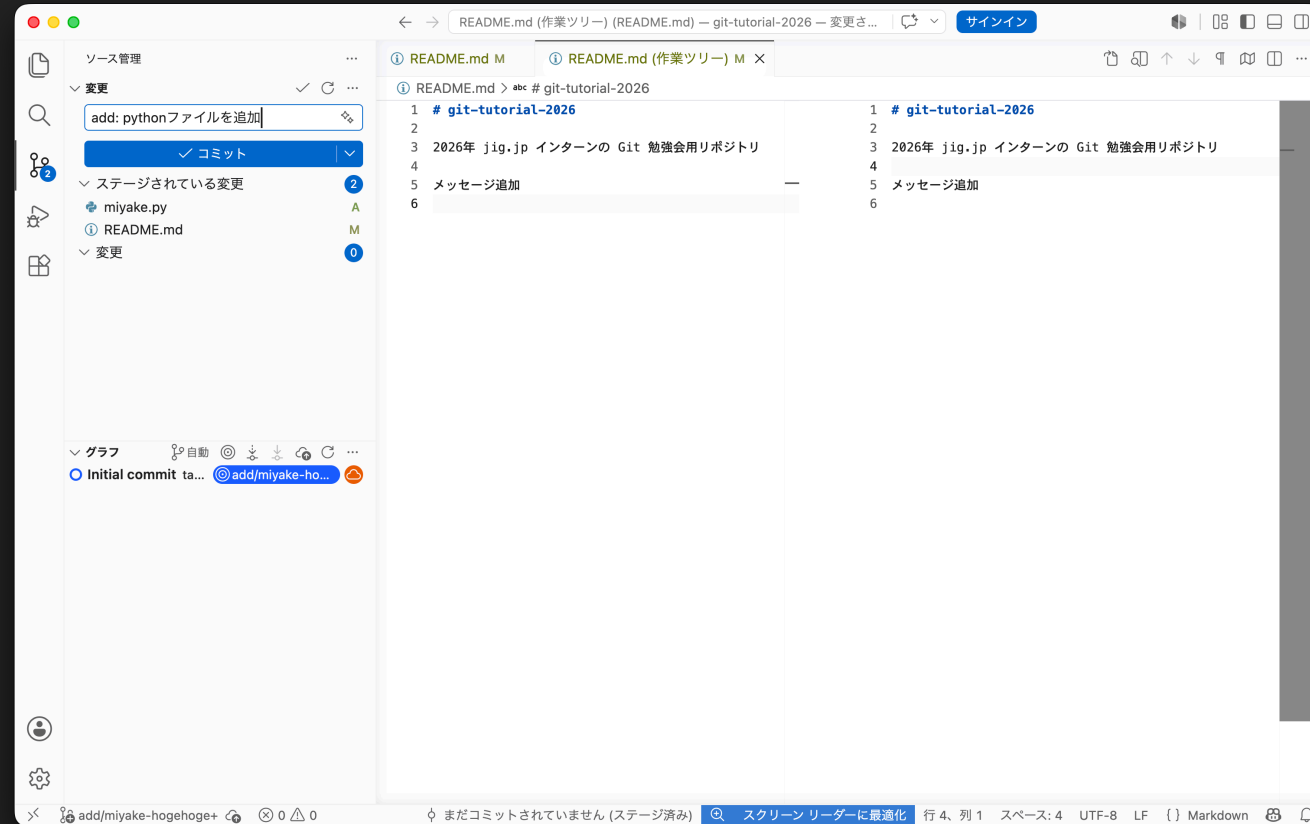
3. ソース管理ビューを開いて、「変更」に作成したファイルと差分が表示されることを確認します



4. ファイル名にカーソルを合わせて「+」（変更をステージ）をクリックし、「ステージされた変更」に移動させます



5. 上部の入力欄にコミットメッセージを記載し、「コミット (Commit)」ボタンをクリックします。これで完了です

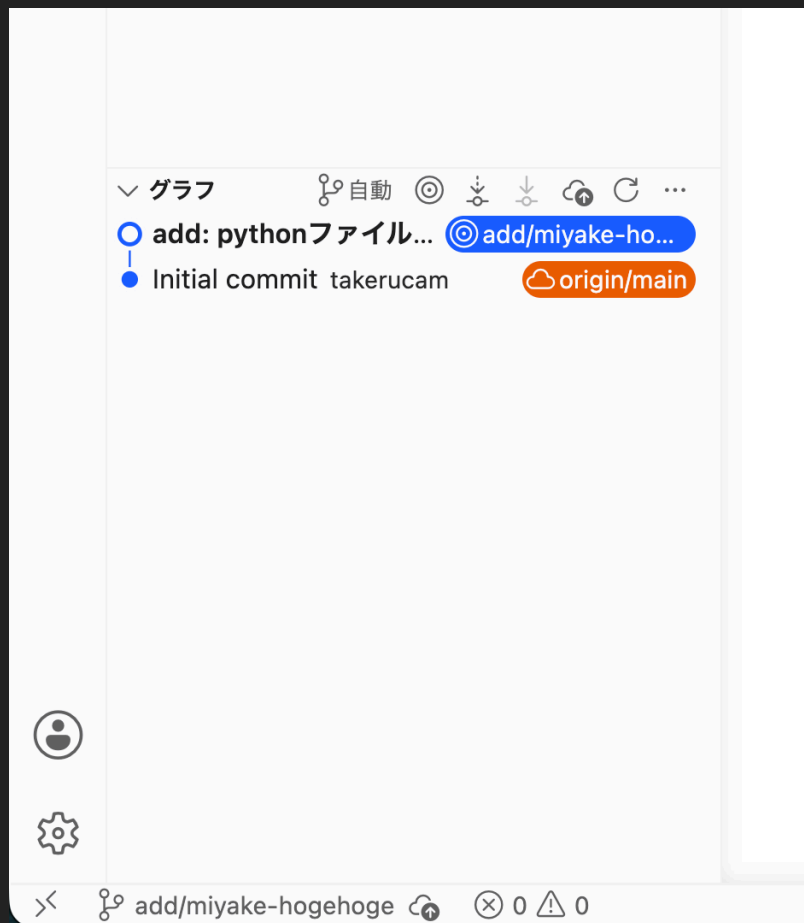


2-4. log: 作業の履歴を確認しよう

作業の履歴を確認してみましょう

この履歴をlog（ログ）と呼びます

ソース管理ビューの「グラフ (Graph)」に、コミットの履歴が表示されることを確認します



2-4. log: 作業の履歴を確認しよう

▽ グラフ 🔗 自動 🎯 📶 📶 ☁️ ⬆️ 🔄 ...

○ add: pythonファイル... 🎯 add/miyake-ho...

● Initial commit takerucam ☁️ origin/main

よさそうですね



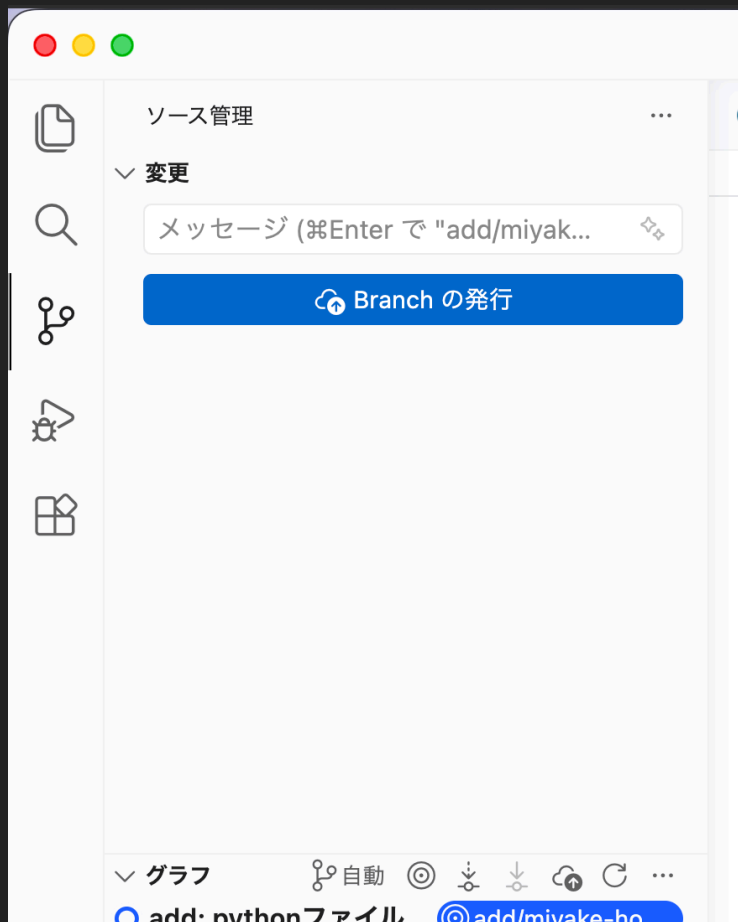
2-5. push: 作業の成果をGithubにアップ ロードして共有しよう

作業の成果をGithubにアップロードしてみ ましょう

これをpush（プッシュ）と呼びます

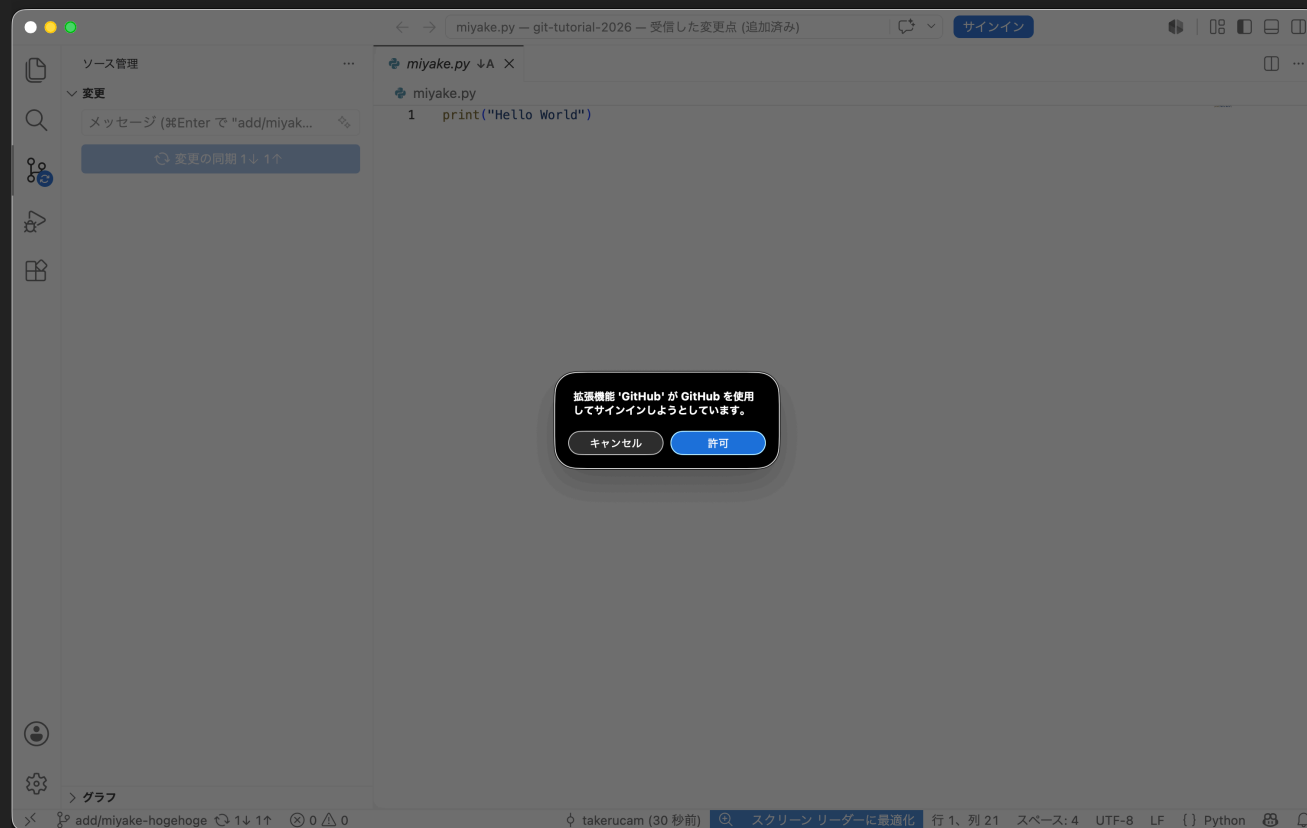
早速、プッシュしてみましょう

1. ソース管理ビューの「Branch の発行」をクリックします



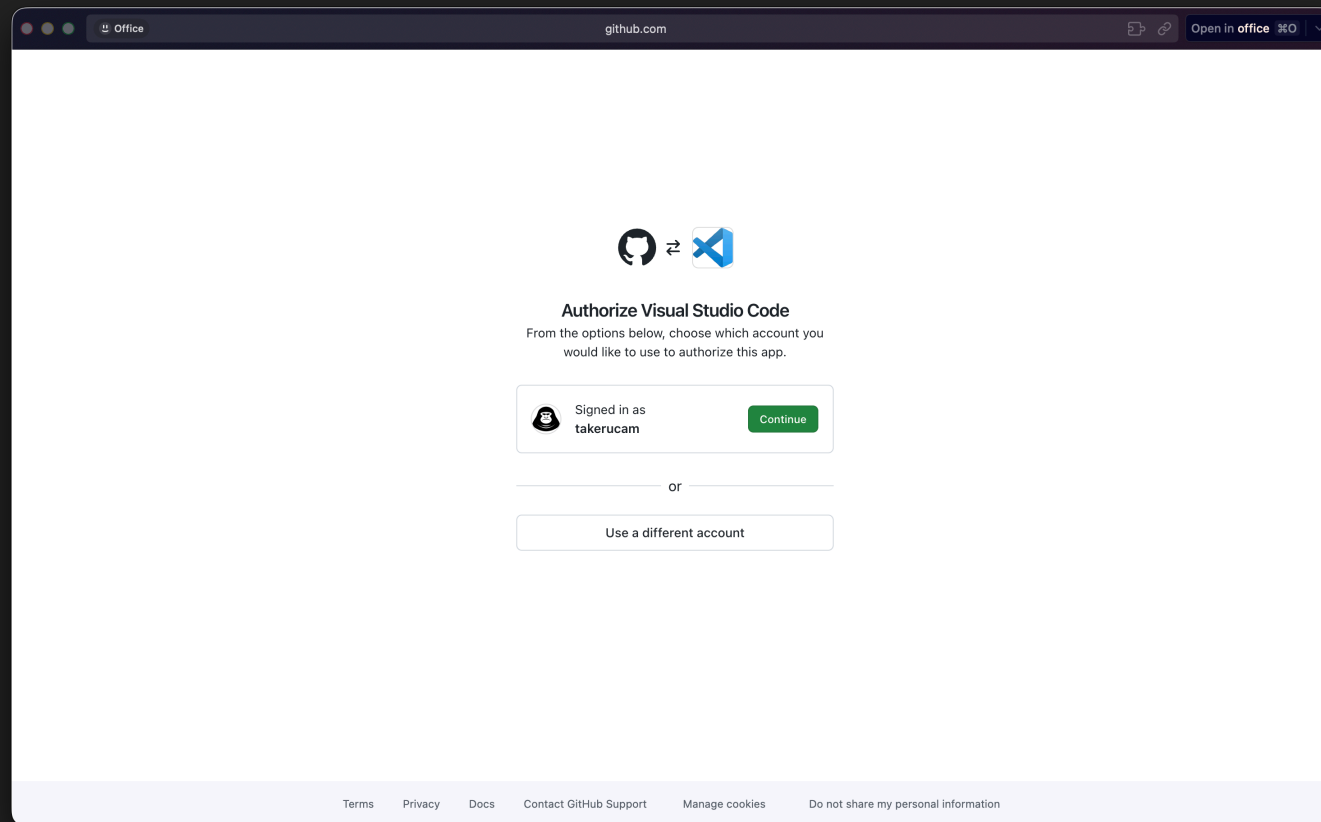
2-5. push: 作業の成果をGithubにアップロードして共有しよう

2. 初回はGithubへのサインインを求められるので、「許可 (Allow)」をクリックします



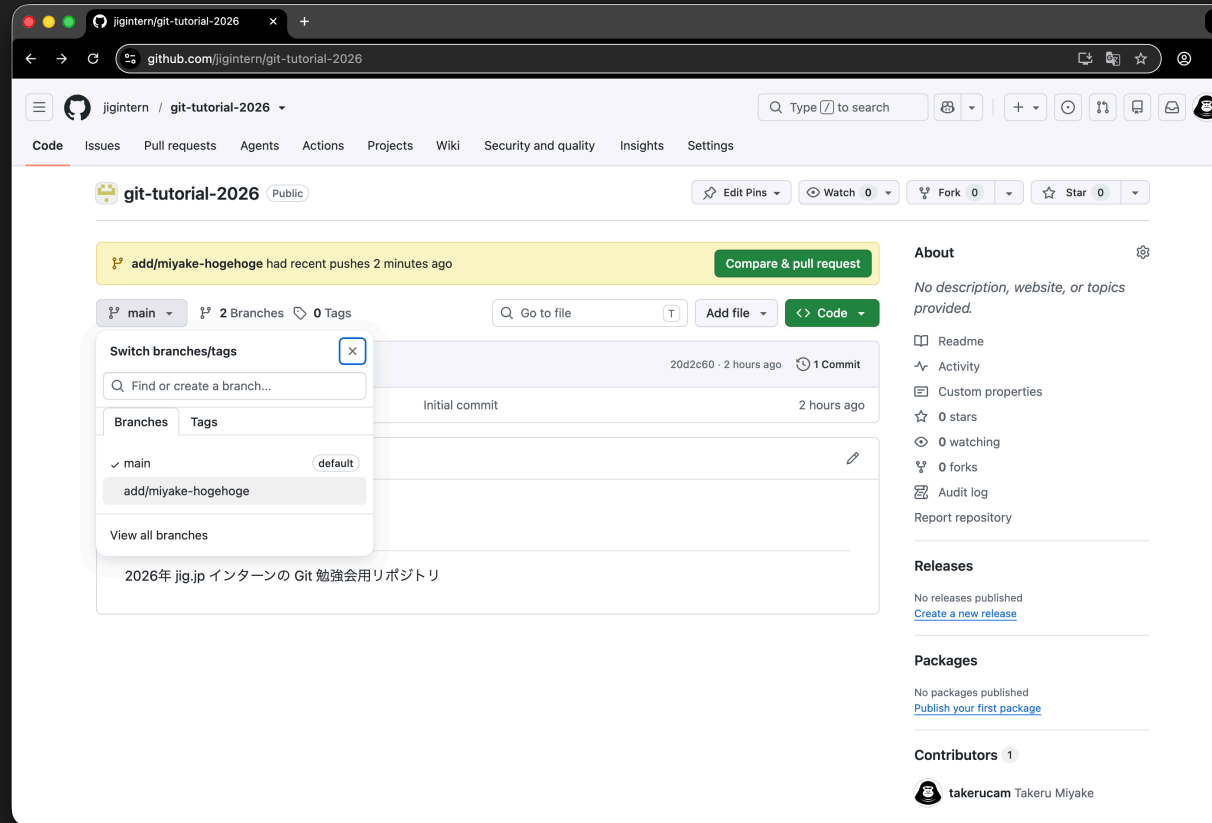
2-5. push: 作業の成果をGithubにアップロードして共有しよう

3. ブラウザが開くので、Githubにログインして「Authorize Visual-Studio-Code」をクリックし、VSCodeに戻ります



2-5. push: 作業の成果をGithubにアップロードして共有しよう

4. ブラウザでGithubを開き、プッシュしたブランチが正しく反映されていることを確認します



2回目以降は「変更の同期 (Sync Changes)」ボタンでプッシュできます

2-6. Pull Request: 枝分かれした成果を結合しよう

ブランチ機能で枝分かれさせた作業を結合 させましょう

これをmerge（マージ）と呼びます

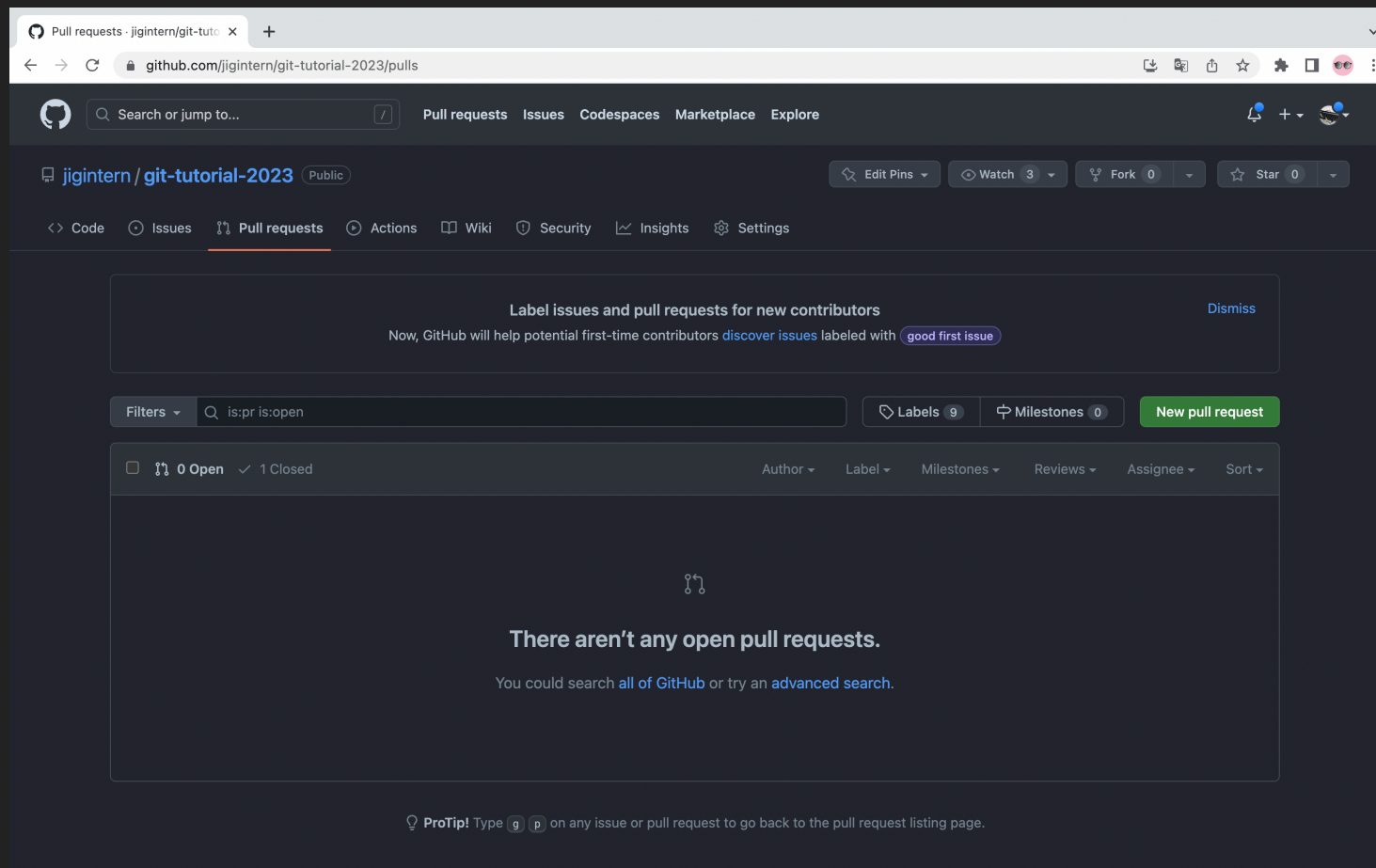
一応、手元でマージもできますが……

マージする時は、バグ等の防止のため、他の人に確認して
貰いたいです

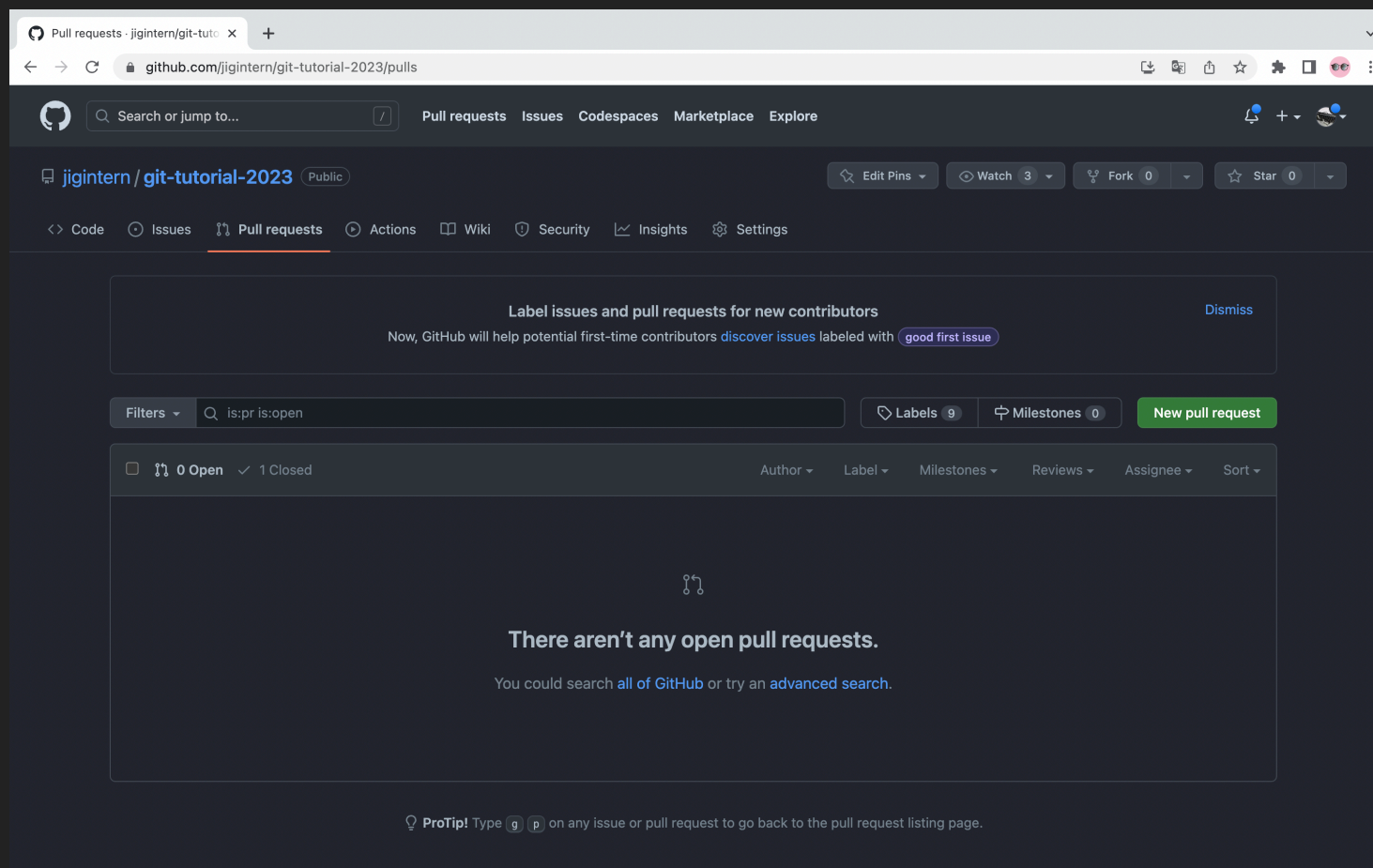
GithubのPull Request（プルリクエスト） 機能を使いましょう！

プルリクエストでマージしましょう

1. Githubで「Pull requests」のタブをクリック



2. 「New pull request」をクリックします



3. 新規ブランチの名前、Pull Requestのタイトル、本文を入力します。作業内容が理解しやすい内容にすると良いです

Comparing main...feature/add- x +

github.com/jigintern/git-tutorial-2023/compare/feature/add-futaba-py?expand=1

Search or jump to...

Pull requests Issues Codespaces Marketplace Explore

jigintern / git-tutorial-2023 Public

Edit Pins Watch 3 Fork 0 Star 0

Code Issues Pull requests Actions Wiki Security Insights Settings

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](#).

base: main ← compare: feature/add-futaba-py ✓ Able to merge. These branches can be automatically merged.

Add: PythonのHello Worldプログラムを追加

Write Preview

PythonのHello Worldプログラムを作成して、リポジトリ直下に配置しました。

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

Create pull request

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

Reviewers
No reviews

Assignees
No one—assign yourself

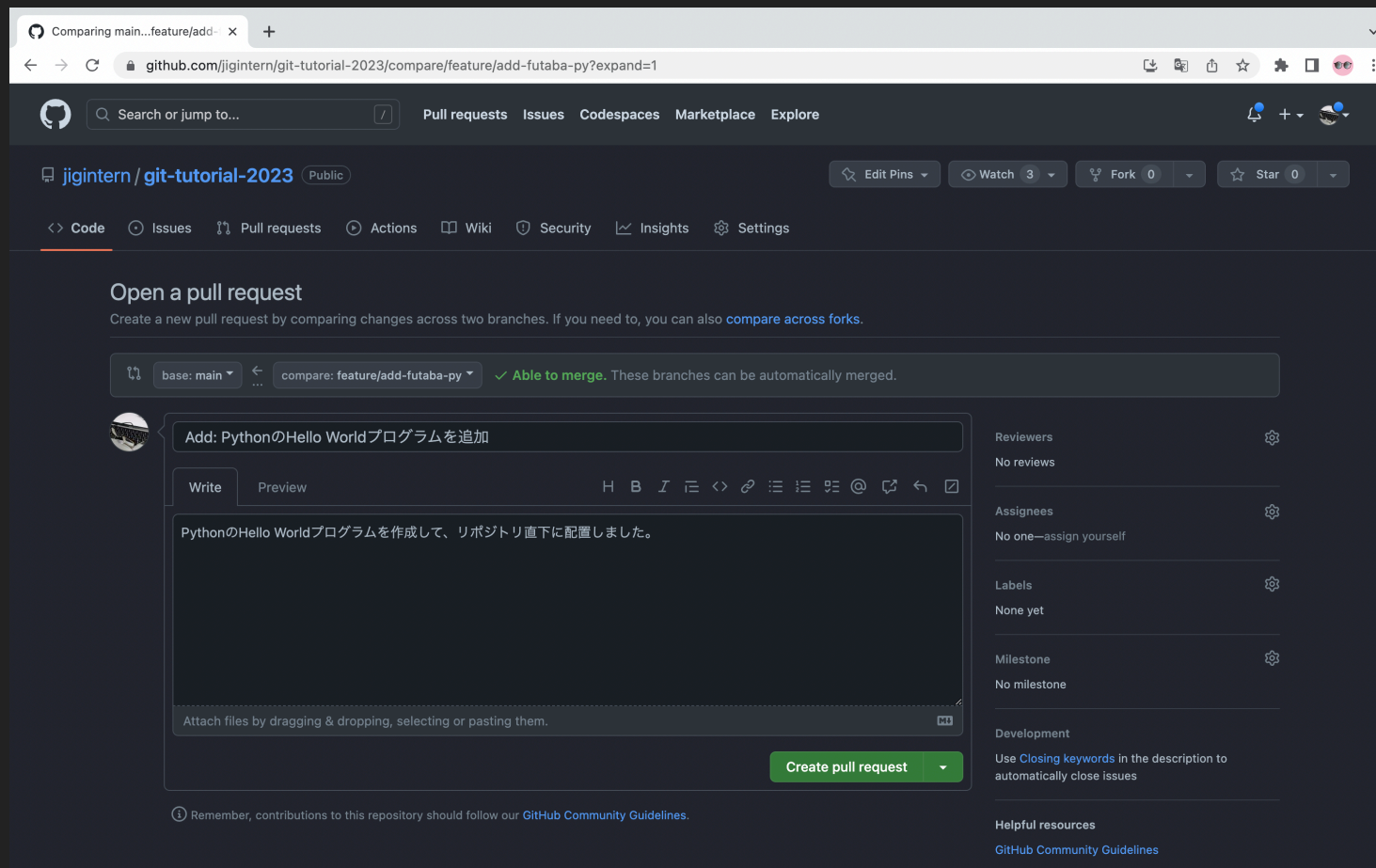
Labels
None yet

Milestone
No milestone

Development
Use [Closing keywords](#) in the description to automatically close issues

Helpful resources
[GitHub Community Guidelines](#)

4. 「Create pull request」をクリックすると、Pull Requestが作成されます



5. 他開発者に、Pull Requestの確認を依頼 します

SlackやGithub上でのコメントなど、適宜チーム内で決定
した方法で依頼しましょう

6-1. 確認を依頼された人は、Pull Requestの変更内容等を確認して、問題箇所があればコメント等で指摘します

The screenshot shows a GitHub Pull Request (PR) page for the repository 'jigintern / git-tutorial-2023'. The PR title is 'Add: PythonのHello Worldプログラムを追加 #1'. The PR is created by 'Futaba-Kosuke' and targets the 'main' branch. The PR description states: 'PythonのHello Worldプログラムを作成して、リポジトリ直下に配置しました。' (Created the Python Hello World program and placed it directly under the repository).

The PR page includes a 'Conversation' tab with 0 comments, a 'Commits' tab with 1 commit, and a 'Checks' tab with 0 checks. The PR is currently in the 'Open' state. The PR description is: 'Add: PythonのHello Worldプログラムを追加'.

The PR page also shows a 'Branch protection rules' section with the following rules:

- Require approval from specific reviewers before merging**: Branch protection rules ensure specific people approve pull requests before they're merged.
- 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.

The PR page also shows a 'Reviewers' section with the following information:

- Reviewers**: No reviews
- Assignees**: No one—assign yourself
- Labels**: None yet
- Milestone**: No milestone
- Development**: Successfully merging this pull request may close these issues.

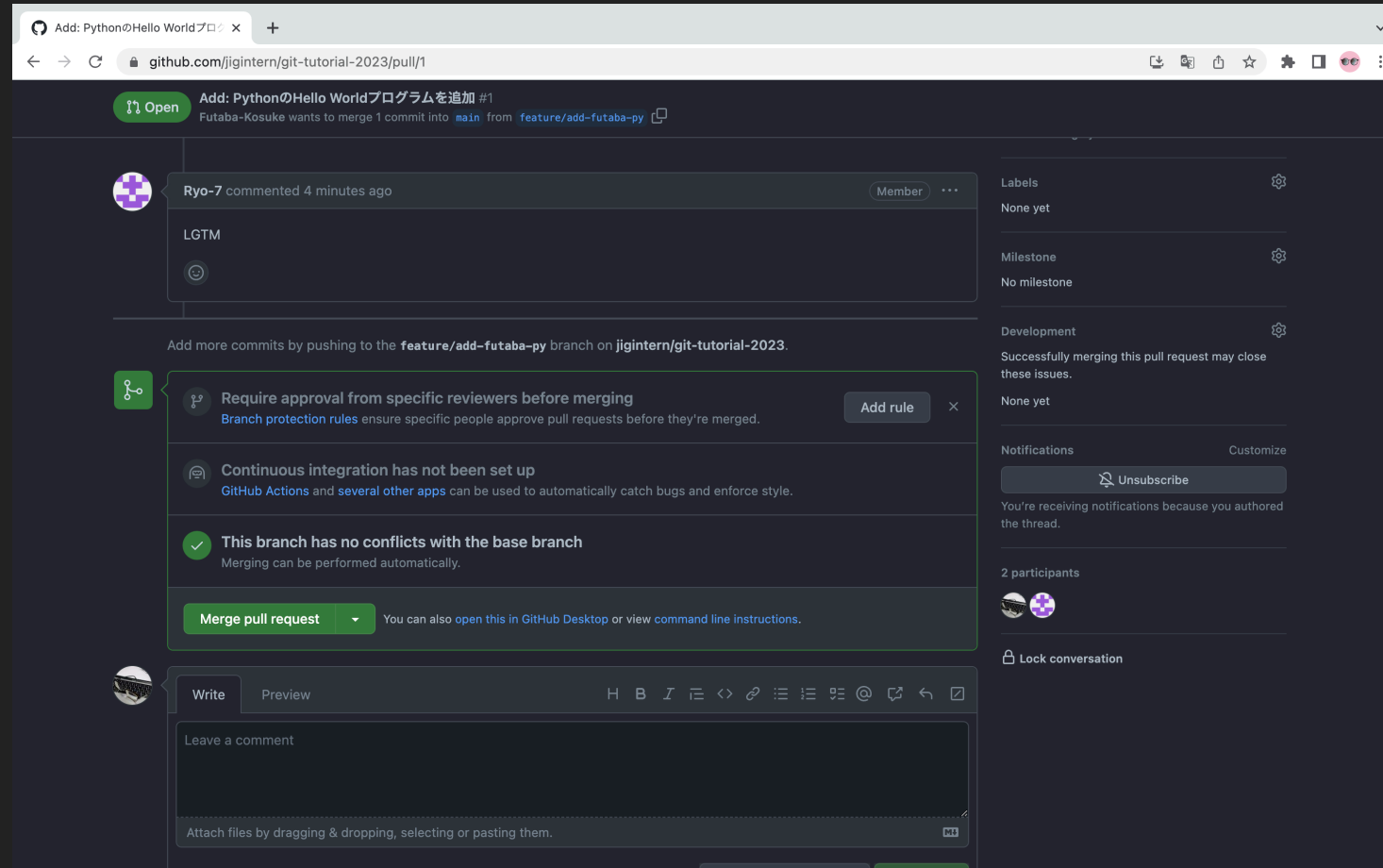
6-2. 問題箇所が無い場合は、LGTM（Looks Good To Me: 私は良いと思います）等のコメントをつけて確認したことを報告しましょう

The screenshot shows a GitHub Pull Request page for the repository "Add: PythonのHello Worldプログラムを追加 #1". The pull request is from the branch "feature/add-futaba-py" to the "main" branch. A comment by "Futaba-Kosuke" is visible, stating "PythonのHello Worldプログラムを作成して、リポジトリ直下に配置しました。". Below the comment, a green box indicates "This branch has no conflicts with the base branch" and "Merging can be performed automatically.".

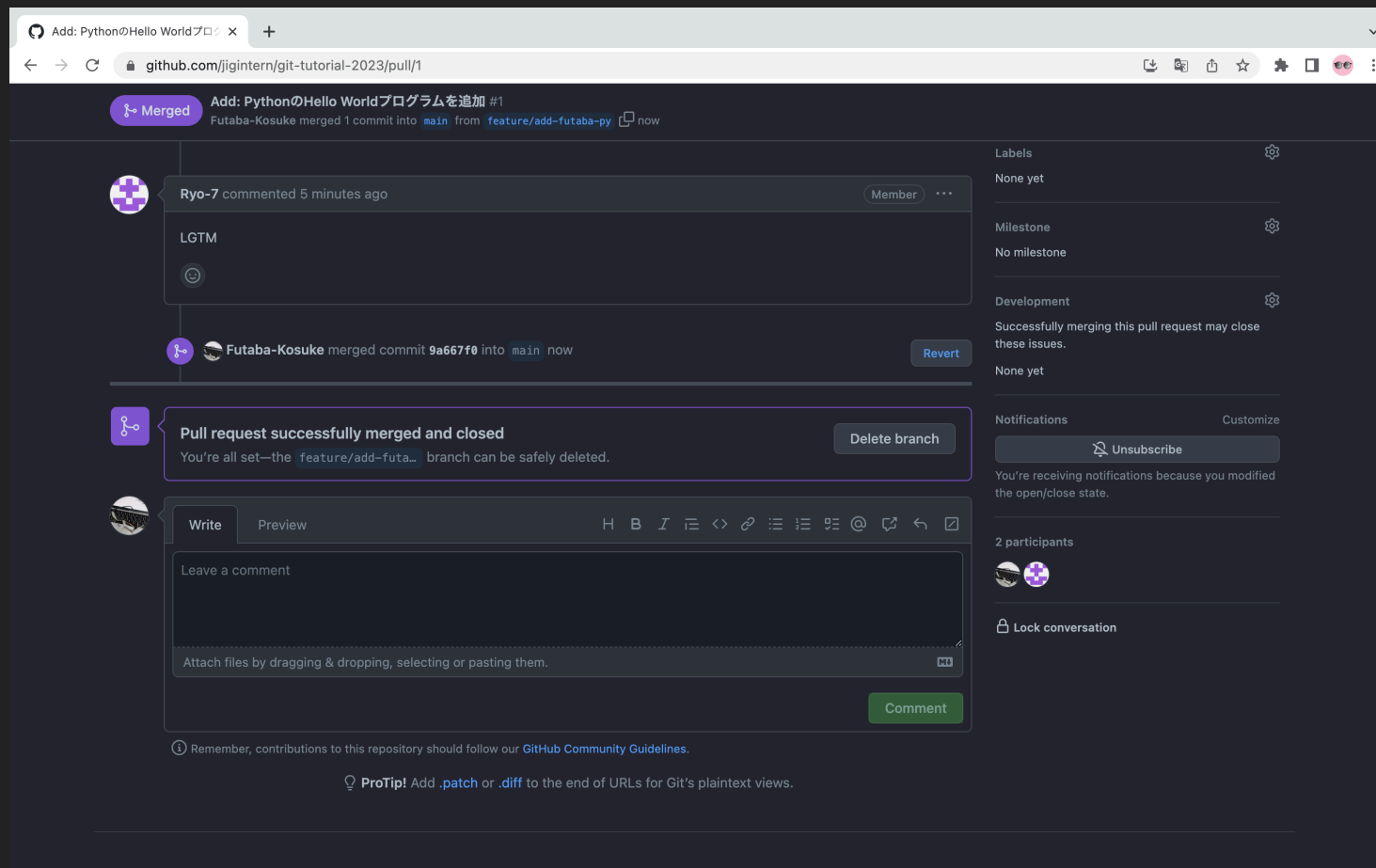
The "Write" tab is active, showing a text input field with the text "LGTM". Below the input field, there are buttons for "Close with comment" and "Comment".

On the right side, there are sections for "Reviewers", "Assignees", "Labels", "Milestone", "Development", "Notifications", and "Participants".

7. Pull Requestの作成者は、「Merge pull request」をクリックして、Pull Requestを元のブランチに結合します



7. Pull Requestの作成者は、「Merge pull request」をクリックして、Pull Requestを元のブランチに結合します



8. mainブランチを確認して、変更内容が正しく取り込まれていることを確認します

Githubで開いてみましょう

2-7. fetch / pull: Githubリポジトリの変更部分をダウンロードしよう

Githubの更新を、PC上に取り込みましょ う

これをfetch（フェッチ） / pull（プル）と呼びます

fetch（フェッチ）は、Github上のリモートリポジトリの
変更を確認する操作です。

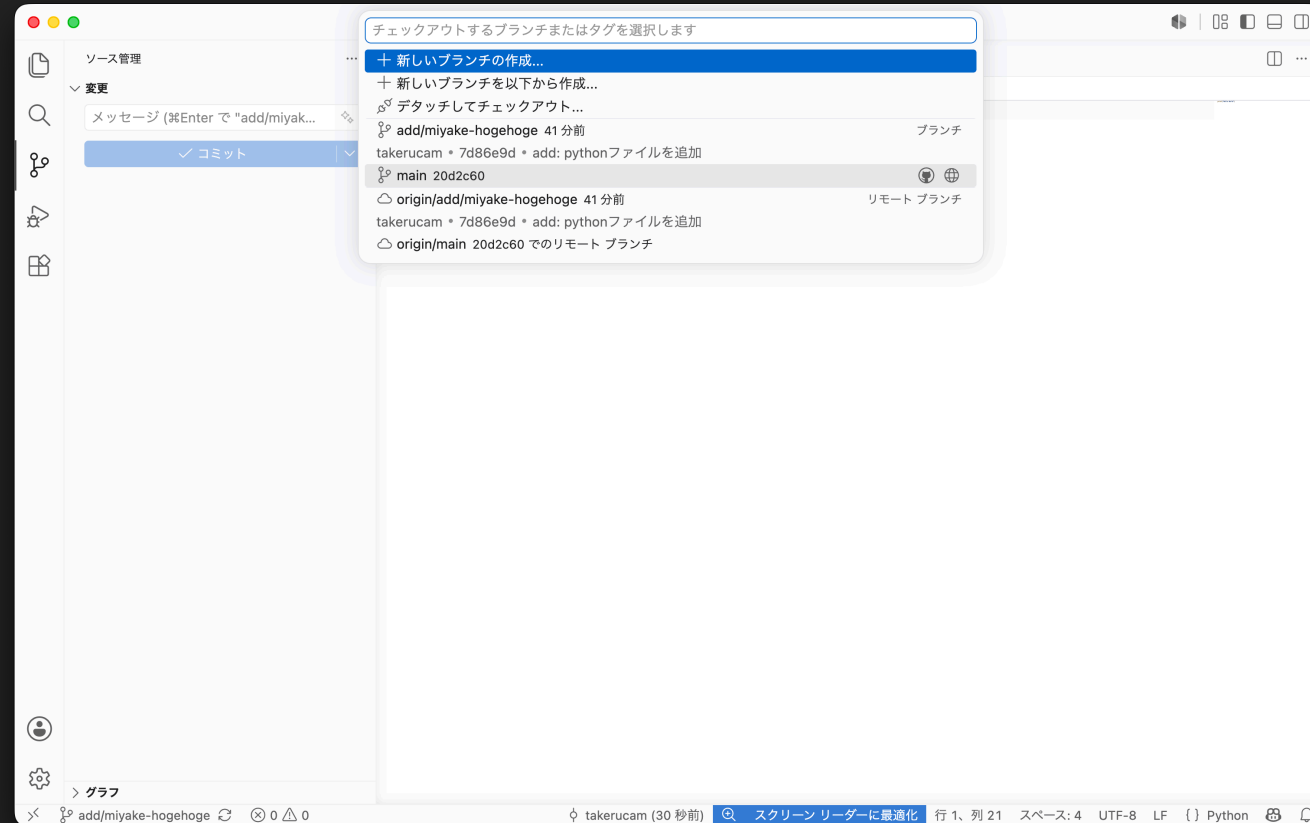
pull（プル）は、Github上のリモートリポジトリの**変更を
ダウンロード**する操作です。

1. fetchしてGithubの変更を確認して
2. pullしてその変更を取り込む

というイメージになります

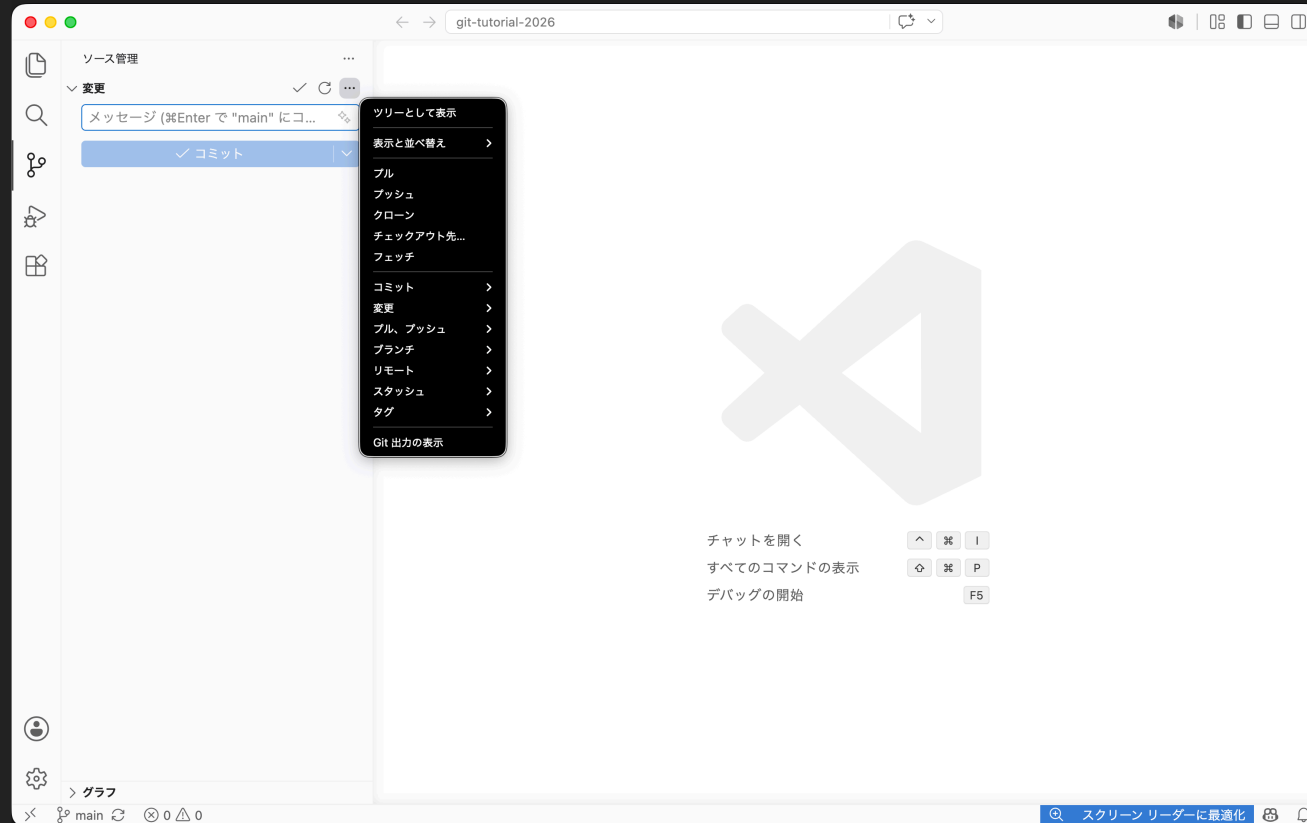
実際に取り込んでみましょう

1. ステータスバーのブランチ名をクリックし、mainブランチに切り替えま す



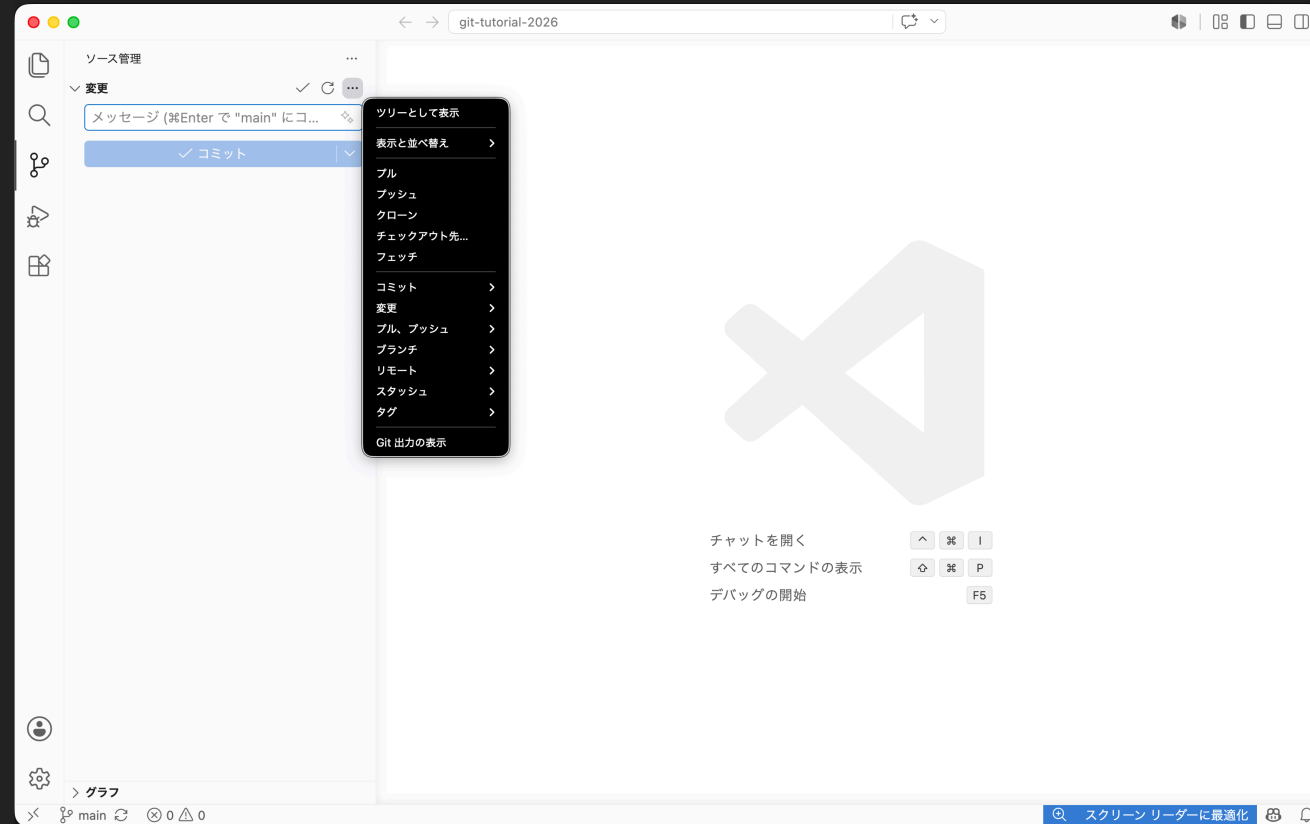
2-7. fetch / pull: Githubリポジトリの変更部分をダウンロードしよう

2-1. ソース管理ビュー右上の「...」メニューから、「フェッチ」をクリックします



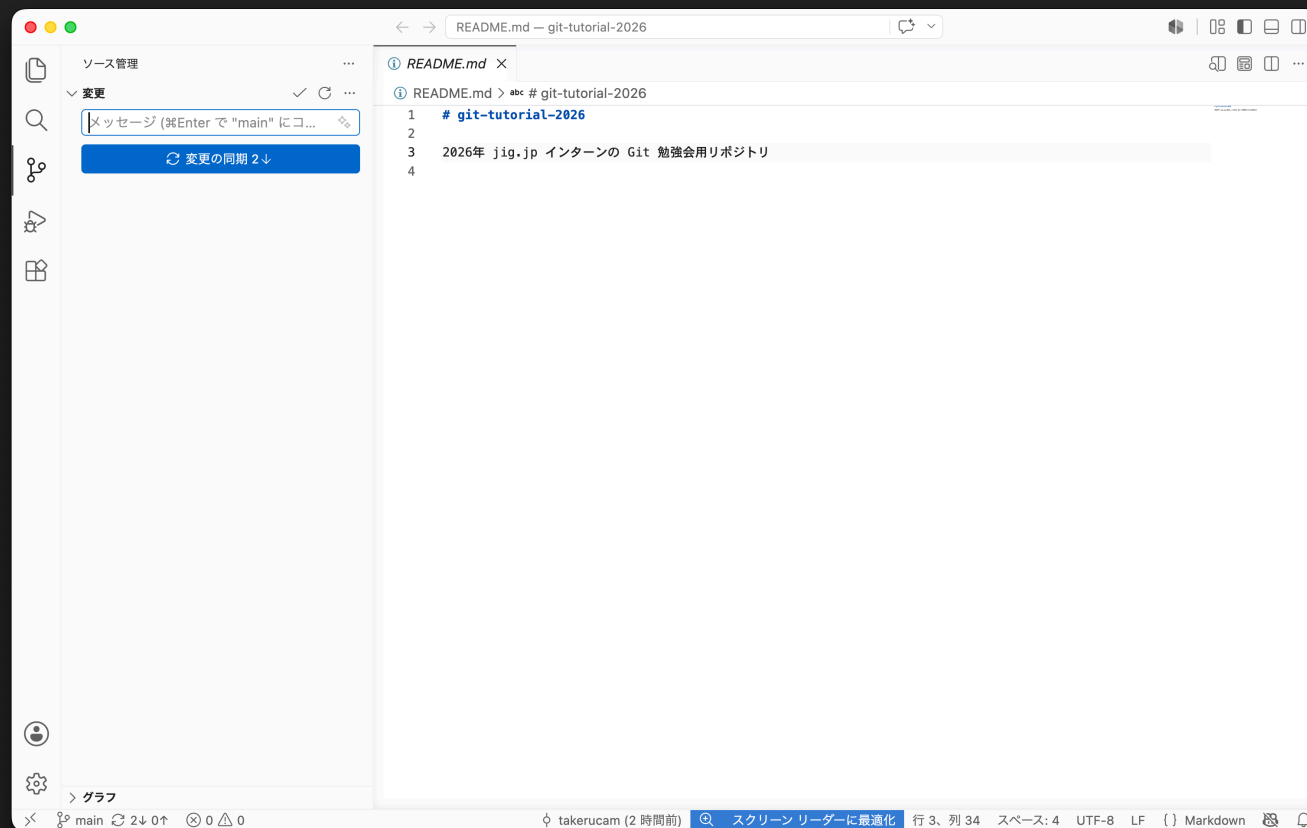
2-7. fetch / pull: Githubリポジトリの変更部分をダウンロードしよう

2-2. これだけでは変更内容は取り込まれませんが、Github上で変更があったことをVSCodeが認識します



2-7. fetch / pull: Githubリポジトリの変更部分をダウンロードしよう

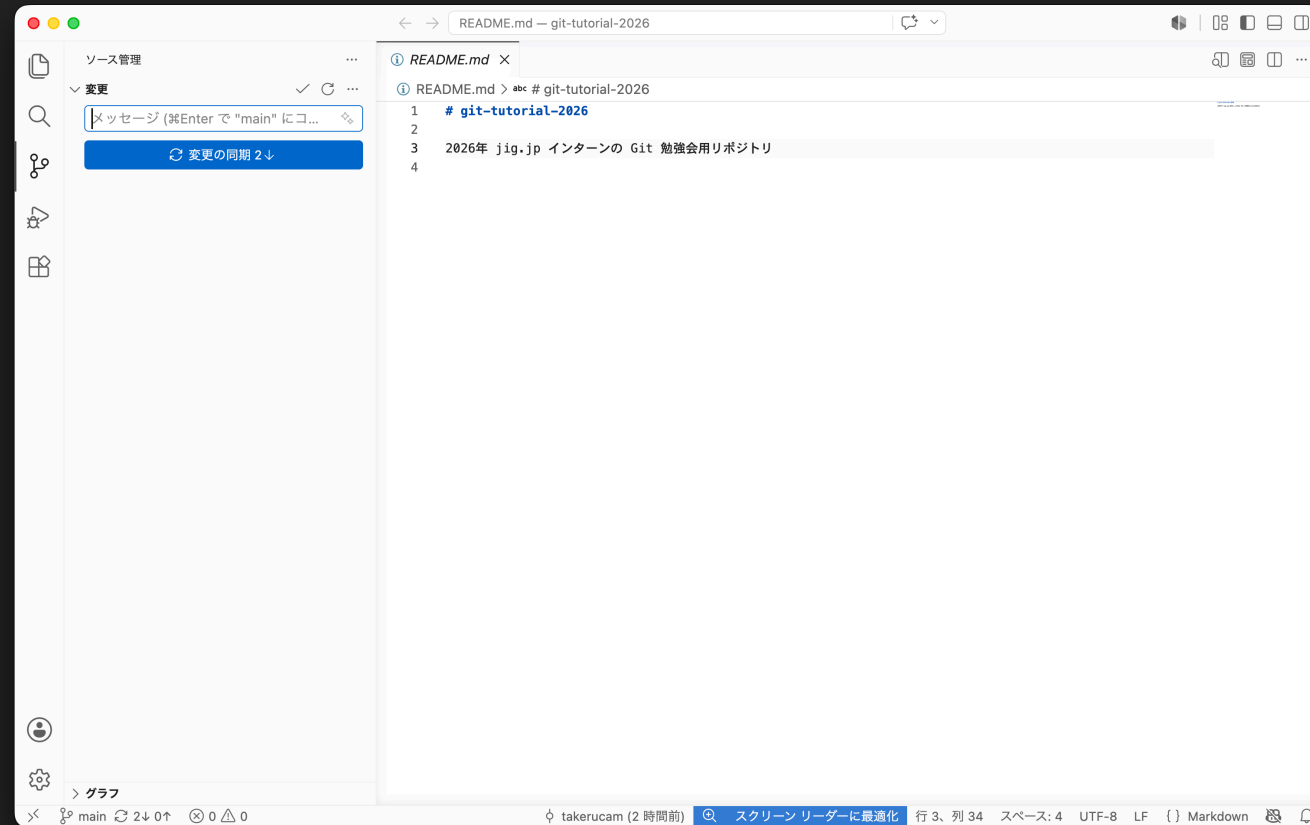
3-1. ステータスバーのブランチ名の横に、「2↓」のように取り込める変更の数が表示されます



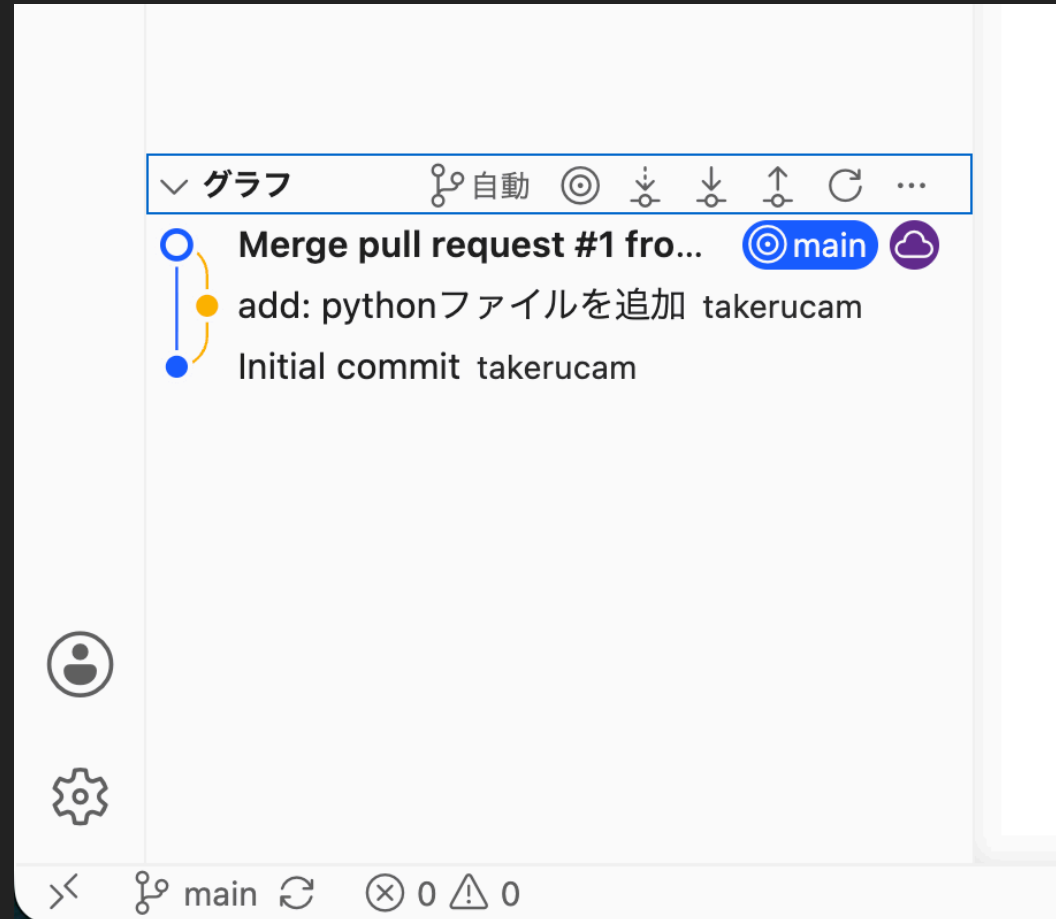
2-7. fetch / pull: Githubリポジトリの変更部分をダウンロードしよう

3-2. 同じ「...」メニューから「プル」をクリックします

変更内容が取り込まれ、ファイルが更新されます



4. 変更内容が取り込まれていることを、ソース管理ビューの「グラフ (Graph)」から確認します



3. まとめ

できるようになったことを、振り返ろう

1. clone: Githubリポジトリを丸ごとダウンロードする
2. branch: 作業を枝分かれさせて、他の開発者との衝突を防止する
3. 通常通りに、プログラムを書く
4. commit: 作業前と後の**差分を記録**する
5. push: 作業の成果をGithubにアップロードして共有する
6. Pull Request: 枝分かれした成果を結合する
7. review: 他の開発者の作業内容を確認する

main

initial commit

pull request #1 ...

feature/topicA

これでGit入門は完璧です

お疲れ様でした！

feature/topicB

Add: index.htmlの基幹部分を実装

Update: ログインフォームを実装