

共同作業ガイド

GitHub共同作業ガイド

基本的なワークフロー

基本的にはGitHub Desktopを使う前提で説明します。Forkなど、ほかのツールを使っても大丈夫です。バージョンやログイン状態などによって、一部の画面が異なる場合があります。

1. mainを最新の状態にする (Pull)
2. 作業用ブランチを作成する
3. 作業用ブランチ / 個人用Workspaceフォルダーで作業する
 1. 作業のまとまりごとにcommitする
4. すべての作業が終わったらPublish / Pushする
5. Pull Request (PR) を作成し、Mergeする

この流れを繰り返していきます。

- **main**は、チームで共有する基準となるバージョンです。
- **branch**は、それぞれの作業を分けて進めるための場所です。
- **commit**は、自分のパソコンに残す変更履歴です。
- **Push**すると、作業した履歴をGitHubにアップロードできます。
- **PR**で確認を依頼したあと、**Merge**でmainに統合できます。

基本的に、**main**を直接編集することはしません。

mainは、いつでもチームで実行・テストできる状態に保つ方針です。

1. リポジトリのClone

まず、リポジトリへの招待が届いたら承認してください。

その後、作業用のパソコンにGitHub Desktopをインストールし、ログインします。

Clone a Repository

GitHub.com

GitHub Enterprise

URL

Filter your repositories

Your Repositories

yunarang-dev/26-1_BlindC_CodingChallenge

yunarang-dev/Alarm_ProjectComet

yunarang-dev/AstralDays_Lumi

yunarang-dev/AstralDays_Repo

yunarang-dev/AstralDays_Wiki

yunarang-dev/C_Study_github

Local Path

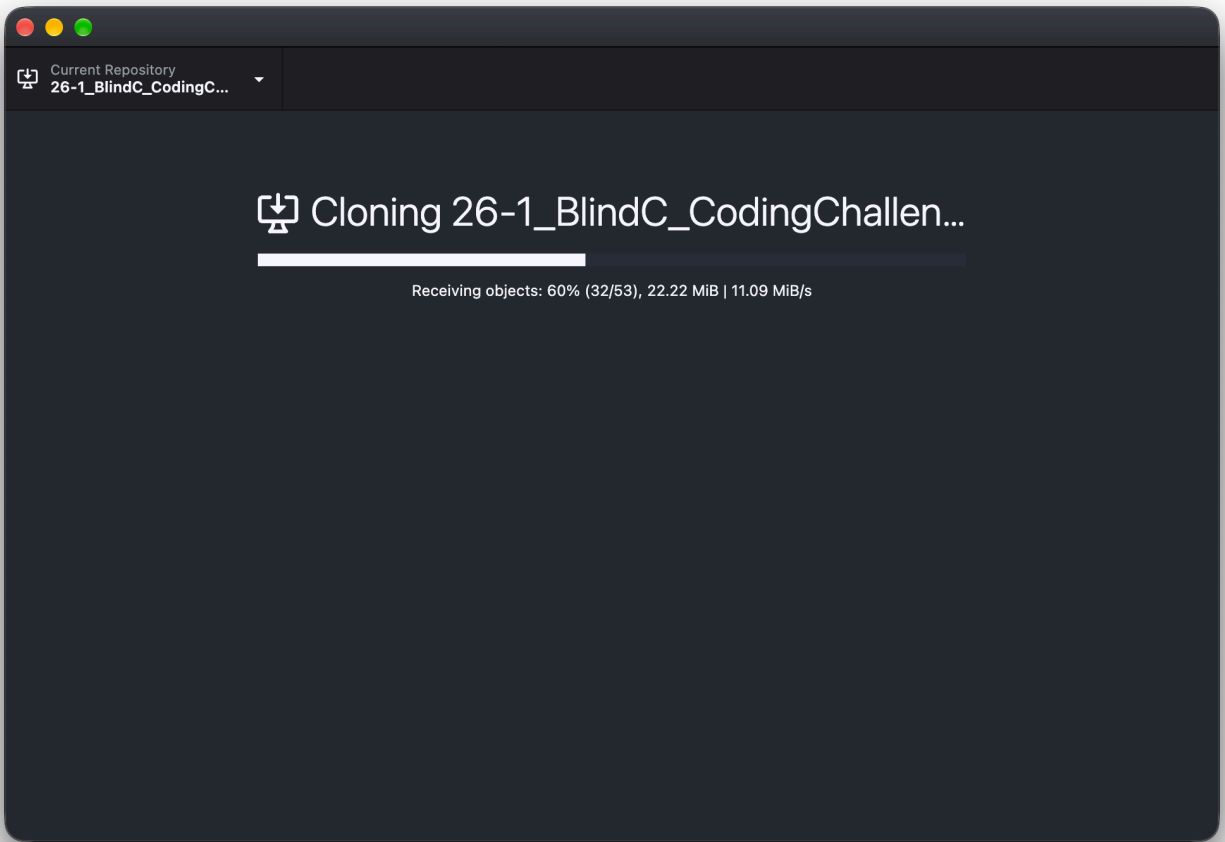
/Users/jhkim/26-1_BlindC_CodingChallenge

Choose...

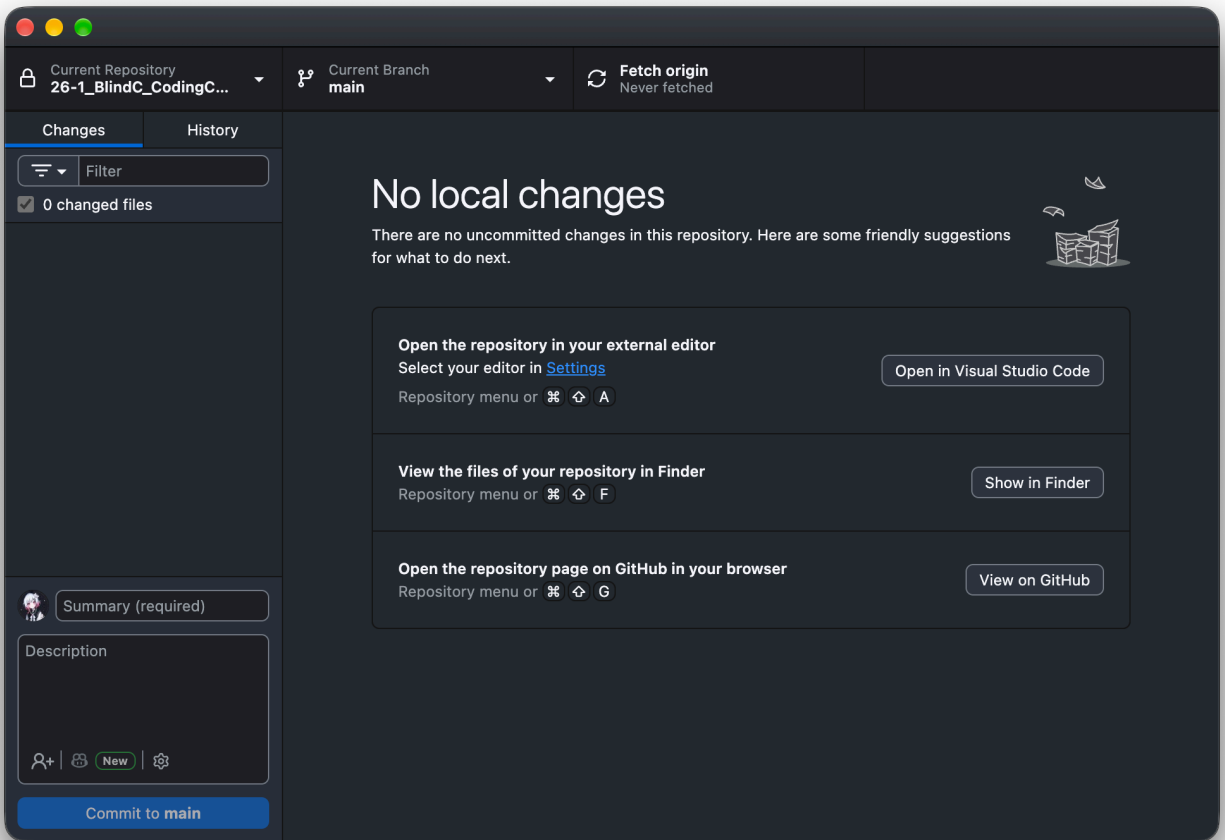
Cancel

Clone

(画像は一例です。実際にはTRAXXING リポジトリをCloneしてください。)



(このようなClone画面が表示されたあと、下の画面に切り替われば、正常にCloneできています。)



2. mainの選択と更新

まず、**Current Branch**が**main**になっているか確認します。**main**以外になっている場合は、**main**に切り替えてください。

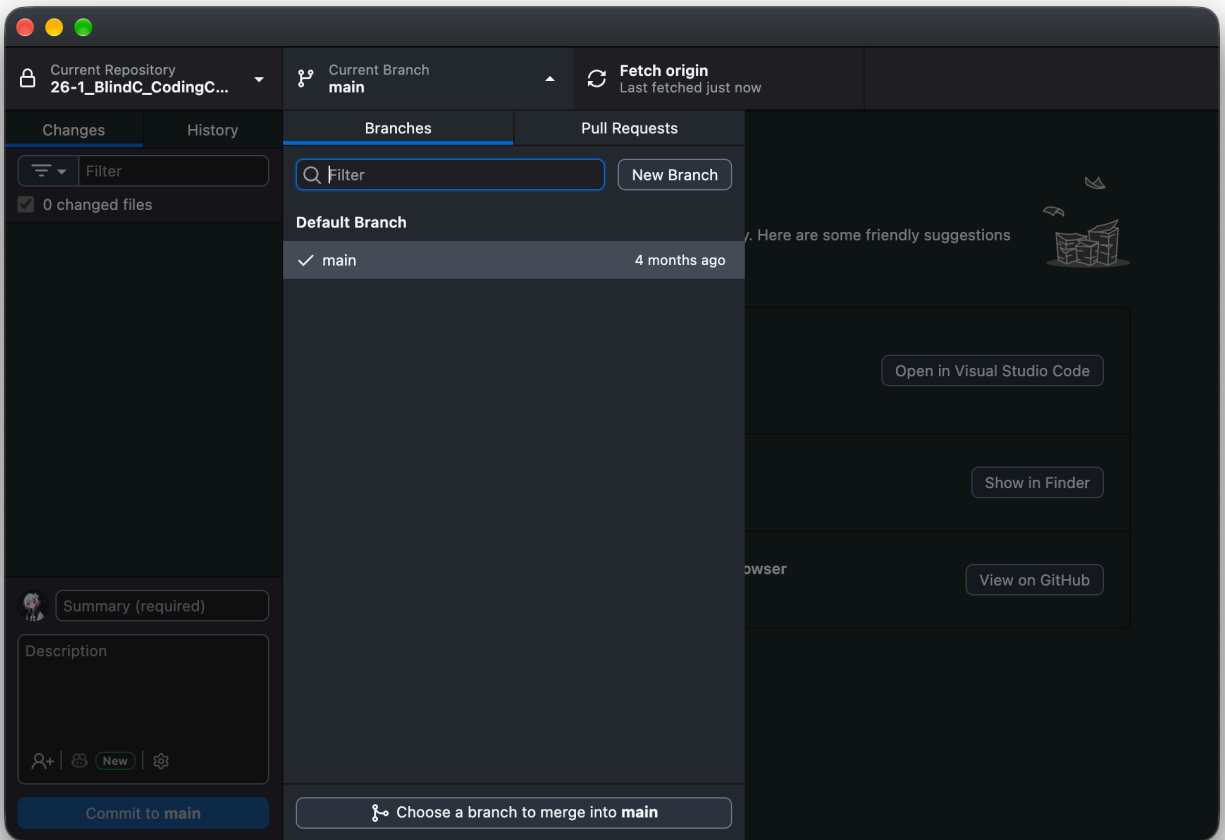
その後、**Fetch Origin**をクリックします。ローカルのファイルを更新する必要がある場合は、**Pull Origin**ボタンが表示されます。

この場合は、**Pull**までクリックすることで、変更がファイルに反映されます。

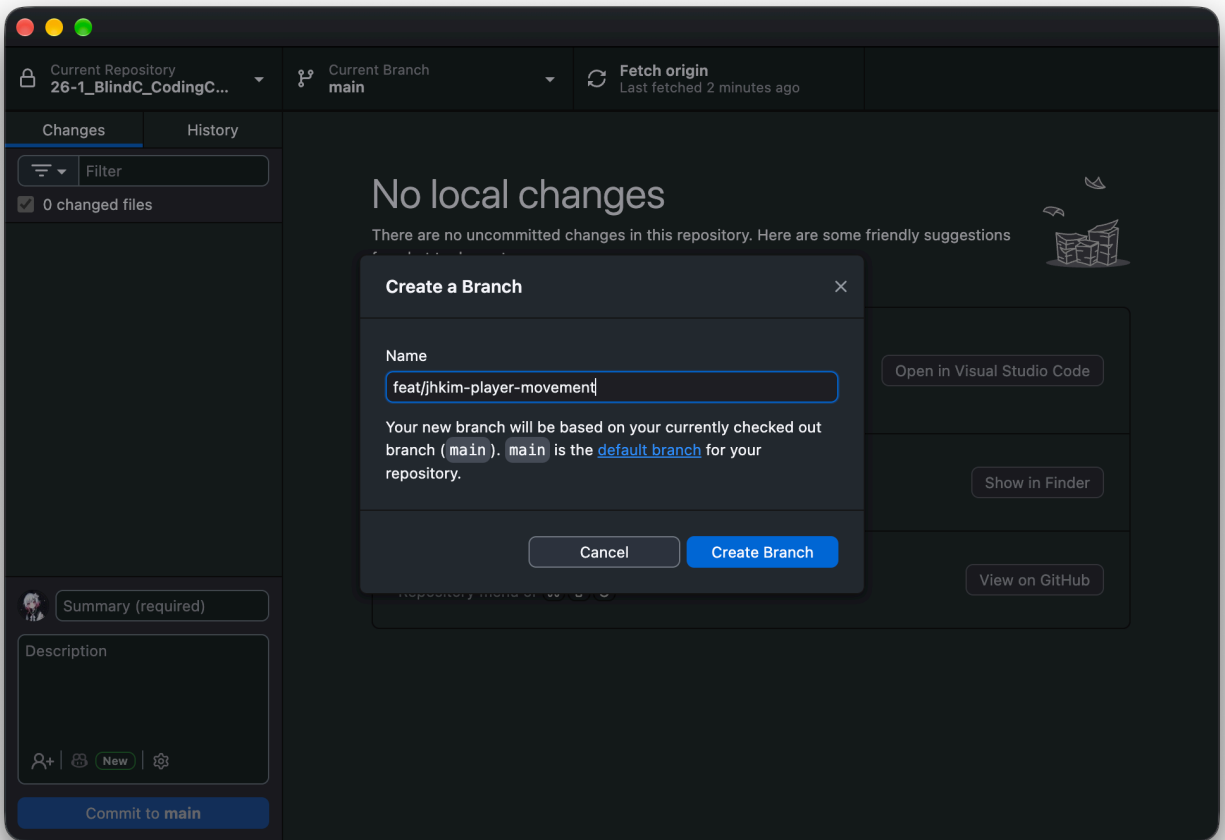
Fetchは変更内容を確認する段階、**Pull**はその変更を実際に取り込む段階です。

3. 作業用ブランチの作成

Current Branchボタンをクリックすると、ブランチの一覧が表示されます。



ここで**New Branch**をクリックすると、新しいブランチを作成できます。



作成元のブランチが**main**になっていることを確認し、**Create Branch**をクリックしてください。このとき、ブランチ名は次のルールに従ってください。

<種類>/<名前>-<作業内容>

例：

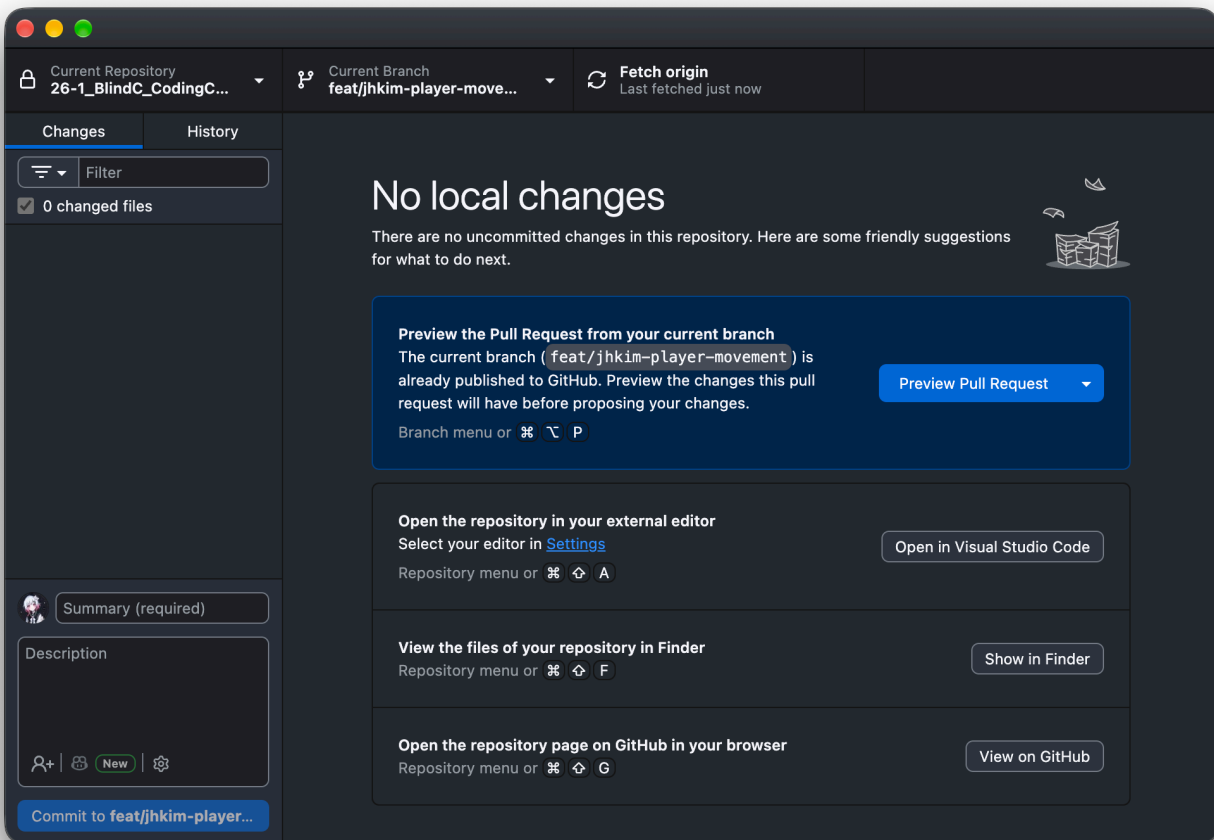
feat/jhkim-player-movement

asset/jhkim-character-modeling

種類	使用する場面
feat	新しい機能の追加
fix	バグの修正
set	設定・構成・初期セットアップの変更
docs	ドキュメントの追加・修正
refactor	機能を変えずにコードの構造のみを変更
test	テストに関するコードや内容の変更
perf	パフォーマンスの改善
build	ビルドシステム・パッケージ・依存関係の変更

種類	使用する場面
revert	以前の変更の取り消し
asset	画像・モデル・サウンドなどの変更

その後、**Publish Branch**をクリックすれば、ブランチの準備は完了です。



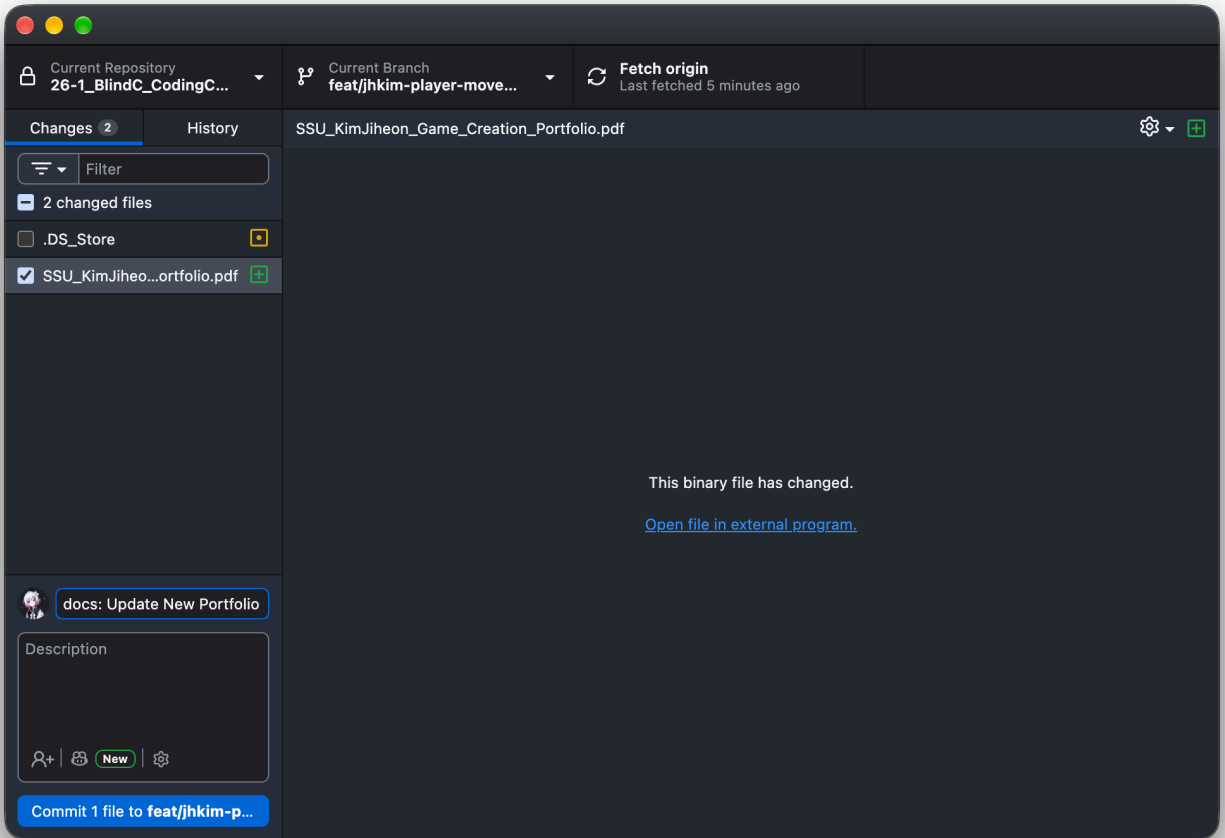
*Current Branch*が新しいブランチに切り替わった状態

画面上部で自分の作業用ブランチが選択されていることを確認してから、作業を進めてください。

4. commitする

切り替えたブランチで作業を進めると、Changesに変更内容が表示されます。

その中から今回の作業に関するファイルだけを選び、**Summary**（コミット名）を入力してください。



위 사진에서는 두 개의 변경 사항이 감지되었어요. 이 중에서 **.DS_Store**는 필요없는 파일이기에 선택하지 않았어요.

<種類>: <作業内容>

例 :

feat: New Movement Code for Character

asset: Character Modeling

種類	使用する場面
feat	新しい機能の追加
fix	バグの修正
set	設定・構成・初期セットアップの変更
docs	ドキュメントの追加・修正
refactor	機能を変えずにコードの構造のみを変更
test	テストに関するコードや内容の変更
perf	パフォーマンスの改善
build	ビルドシステム・パッケージ・依存関係の変更

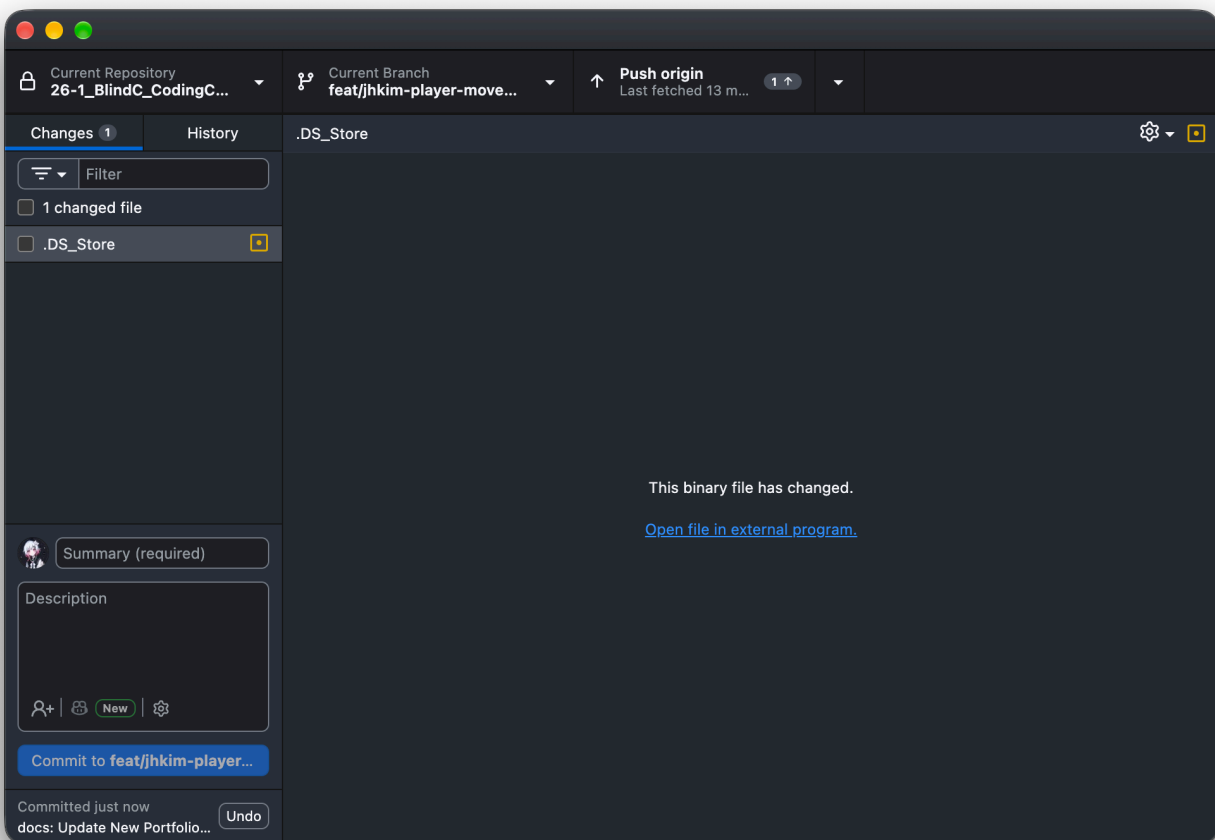
種類	使用する場面
revert	以前の変更の取り消し
asset	画像・モデル・サウンドなどの変更

基本的にはブランチ名と似たルールですが、<名前> は含めず、/ の代わりに : を使います。

種類は小文字で、英語の説明は各単語の最初の文字を大文字にしてください。GitHubやUIなどの表記は、そのまま使ってください。

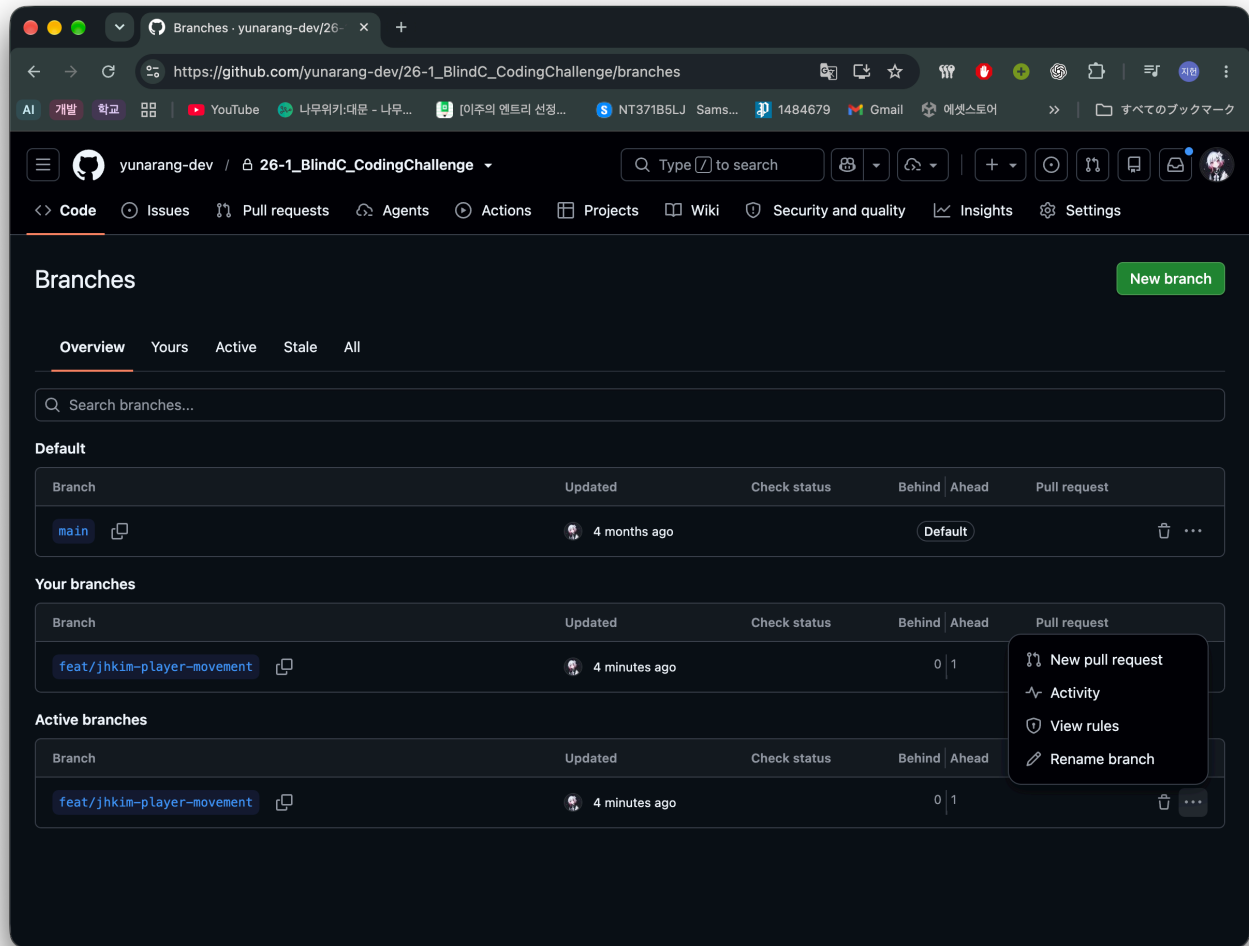
準備ができれば、画面下部にある青い**Commit n Files to ...**ボタンをクリックしてください。

5. Push・PR・Mergeする

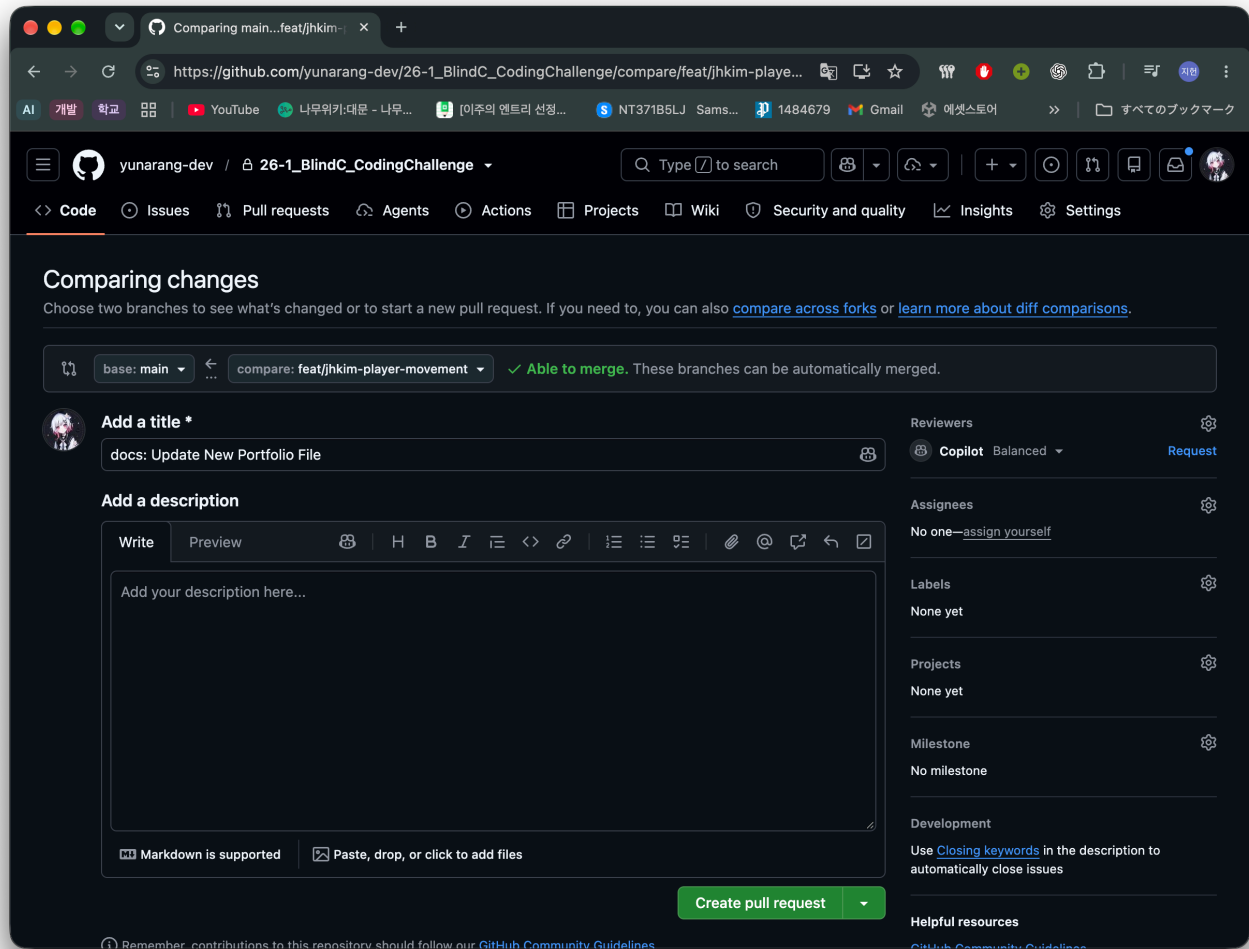


commitすると、**Push Origin**ボタンが有効になります。**Push Origin**をクリックすると、GitHub上のブランチに変更が反映されます。

ブランチでの作業がすべて終わったら、GitHubのリポジトリページを開き、**Branches**ボタンをクリックします。ここで、既存のブランチを確認できます。



自分が作業したブランチの「...」ボタンをクリックし、**New Pull Request**を選んでください。



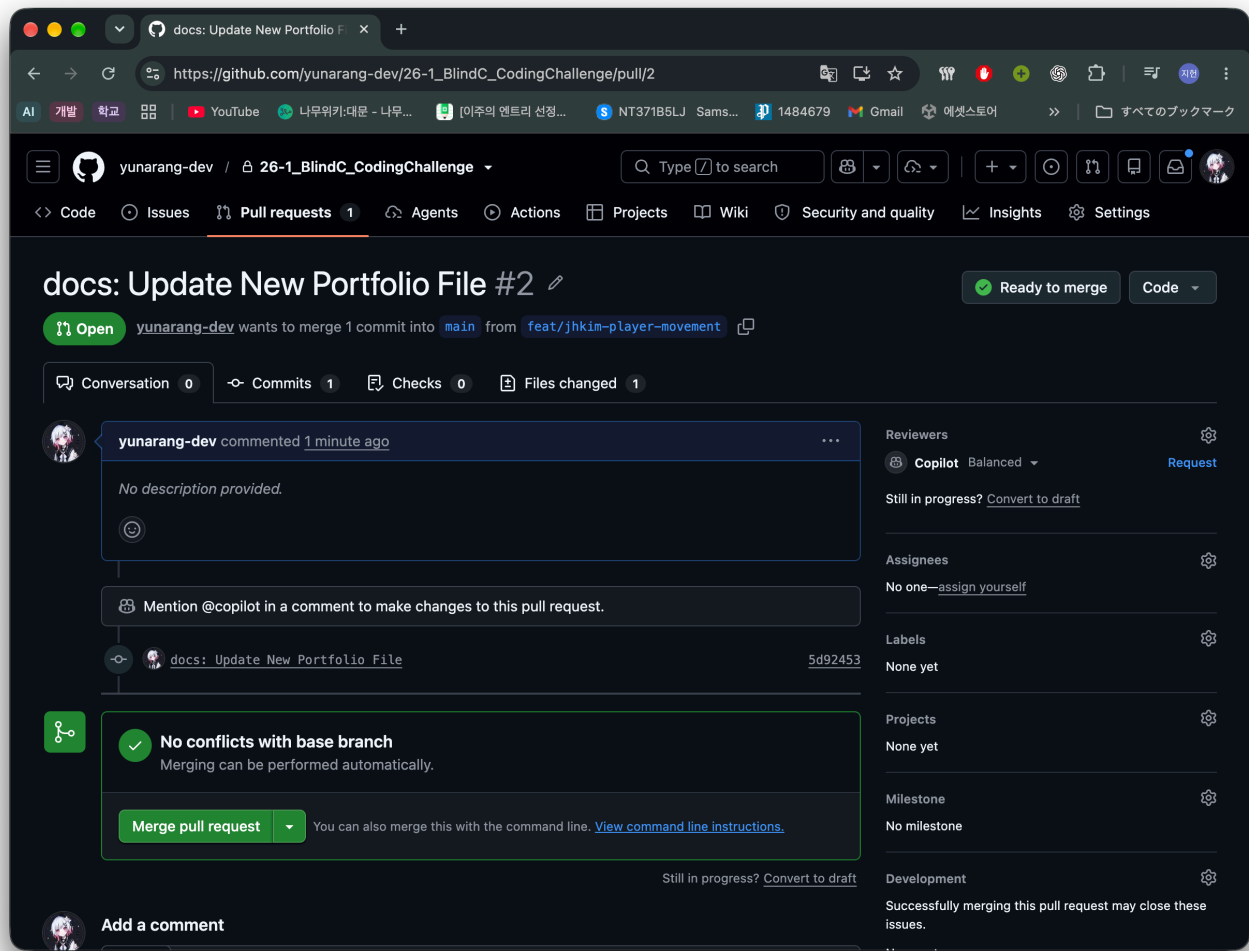
(PRのタイトルも、コミットのルールに従ってください。)

Able to mergeと表示されていれば、特別な対応をせずにそのまま**Merge**できるという意味です。

(Mergeできない場合は、理由を確認してから教えてください。)

その後、**Create Pull Request**ボタンをクリックすると、次の画面に進みます。

少し待つと、**Merge Pull Request**ボタンが有効になります。



Mergeにはいくつかの方法がありますが、このチームでは最も基本的な**Merge Commit**を使う予定です。
そのまま**Merge Pull Request**ボタンをクリックすれば大丈夫です。

続いて**Confirm Merge**をクリックすると、**main**に**Merge**されます。

Unity共同作業ガイド

使用するUnityのバージョン：6000.6.3f1（予定）

基本的なワークフロー

Unityのエディターを初めて起動すると、ProjectウィンドウにAssetsフォルダーが開いています。

```
Unity/  
├── Assets/  
│   ├── Workspace/  
│   │   ├── jhkim/  
│   │   └── Scenes/
```

```
├── Scripts/  
├── Prefabs/  
├── Test/  
└── .../
```

上のようなフォルダー構成で作業してください。

1. まず、Workspaceフォルダー内に、自分の名前のフォルダーを作成してください。
2. そのフォルダー内で担当する機能を実装し、実装した機能を確認するためのSceneを作成してテストしてください。
3. 移動・リズム判定・UIなどの機能は、まず自分のテスト用Sceneで確認します。共有Sceneを直接編集するよりも、**Prefab**や**Component**としてまとめてから組み込む形を基本とします。

共有フォルダーは、Assetsフォルダーの直下に置く予定です。

共有フォルダーには、ほかのチームメンバーが使っても問題ない程度に安定してから移動します。役割が定まっていること、テスト済みであること、大きな構造変更が発生する可能性が低いことを基準にします。

まだ試している段階なら、Workspaceに置いたままでも大丈夫です。Workspace内の作業もPRで共有できますが、mainに統合するときは、プロジェクトの実行に問題がない状態にしてください。

Unity上で移動や名前の変更を行うと、.metaも一緒に処理されます。**Finder**や**エクスプローラー**で、アセットだけを個別に移動しないでください。コード内にパスを直接書いている場合は、そちらも確認してください。

(.metaファイルと一緒に移動されないことがあります。)

Git LFSは使用しない予定です

Assetsフォルダー内の.blend、.psd、.wav、動画などの元データは、**Commit**前に容量と必要性を確認してください。

GitHubの通常のGitリポジトリでは、1ファイルが**100 MiB**を超えると**Push**がブロックされます。50 MiBを超えるファイルについても、Commitする前に相談してください。

容量の大きい元データは、チームで決めた共有スペースに置き、ファイル名・バージョン・ダウンロード先を記録してください。ゲームに組み込むファイルは適切なサイズで書き出し、必要な参照が欠けていないか確認してください。

容量が原因でPushできなくなった場合、ファイルを削除して再度**Commit**するだけでは解決しないことがあります。過去の履歴に大きなファイルが残っている可能性があるため、一緒に確認しましょう。