



SourceTree

作業の流れ



まえおき、注意点

- この方法は必ずしも正解ではありません
ほかにも多くのやり方がありますし、間違っている可能性もあります
(参考程度に見てください。)
- 今回は、利用する流れをまとめています
- 導入までの流れについては、GIT準備の流れをご覧ください
- Unityでの制作を想定しています。別ツールの場合は方法が違う場合もあります
- ここが変、ここがよくわからない、ここが間違ってる等あれば教えてください
- 随時更新中です

目次

- 05ページ：作業用ブランチの作成
- 07ページ：ブランチとは？
- 08ページ：作業用ブランチ更新の流れ
- 13ページ：masterへマージする流れ

作業開始、するまえに...

**作業をするときは、
masterブランチでは作業
しないことを勧めます**

作業用ブランチの作成

基礎のプロジェクトが共有できているところから、進めていきます。

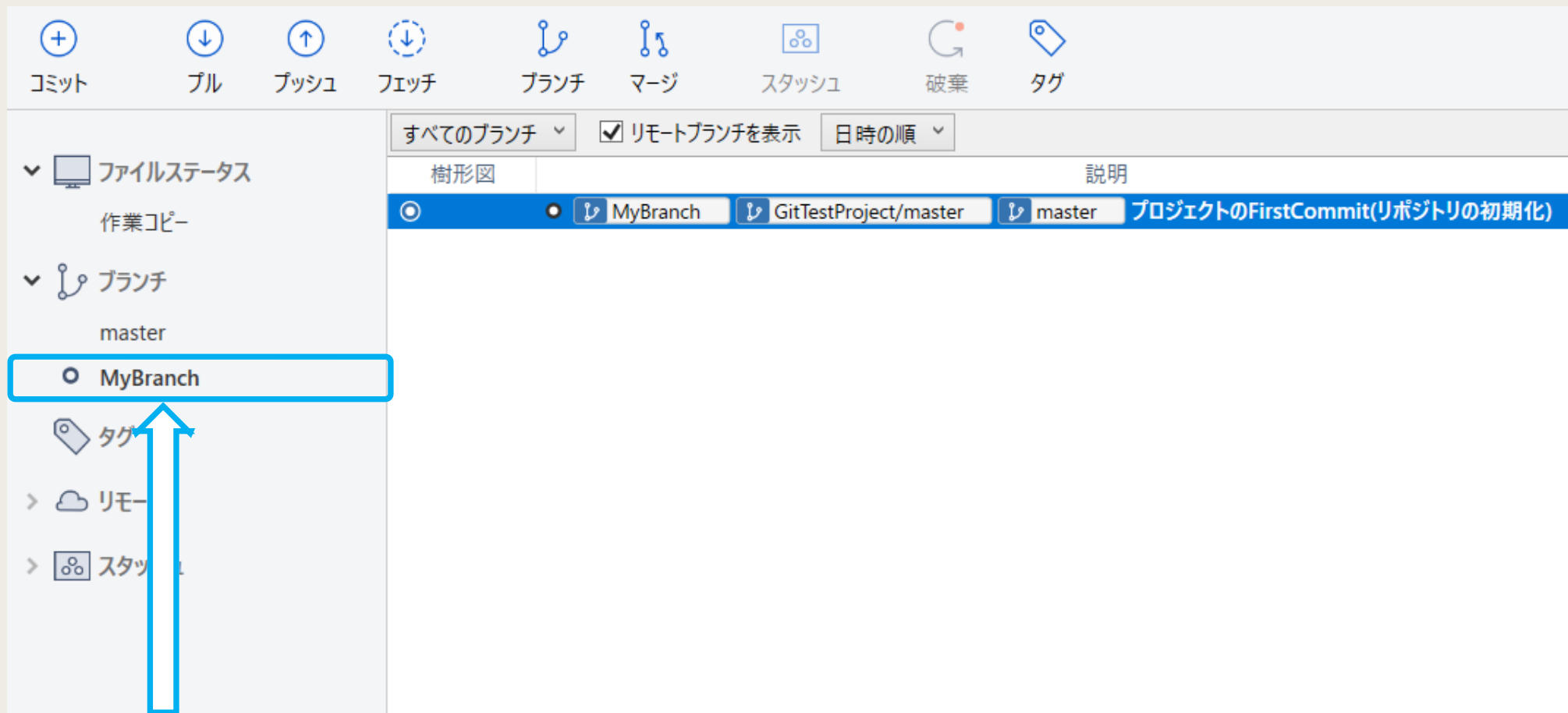
01.上のほうにある、ブランチをクリック

02.新規ブランチに、名前を入力する
(例:自分の名前Branch 等。)

03.ブランチを作成
をクリック

※今回は、masterブランチをもとにブランチを作成しています

作業用ブランチの作成



画面の左側のブランチにmasterとは別で、MyBranchというブランチが作成されています。

※今回は、masterブランチをもとにブランチを作成しています

ブランチとは？

- Branch = 枝という意味があります。
作業の変更内容を、ブランチごとに分ける役割を持っています。
masterも一つのブランチです。
- masterとは別で作る理由は？
 - ・簡単に言うと、masterで作業しないようにするためです。
masterで作業をすると、バグを多く生み出す原因になってしまうため分けます。
 - ・今回作ったブランチは、自分の作業内容を残すためのブランチです。
要は、「自分用のブランチ」です。このブランチで作業をします。
 - ・masterブランチは、完了したタスクたちが集まって、出来ていくもの。
(masterブランチが、最終的にビルドされるものになります。)
作業用ブランチは、自分のタスクを更新していくもの。
(ここで完成させたタスクを、masterにマージしていきます。マージ=結合)

作業用ブランチ更新の流れ

自分の作業を、一通り終えてからの、SourceTreeでの作業を解説していきます。

今回はUnityにて、

- Repositoryフォルダ

- ↳ Scenesフォルダ(Repositoryフォルダ内のフォルダ)

- ↳ Prefabsフォルダ(Repositoryフォルダ内のフォルダ)

- ↳ Scriptsフォルダ(Repositoryフォルダ内のフォルダ)

- ↳ MyBranchTest.csファイル(Scriptsフォルダ内のファイル)

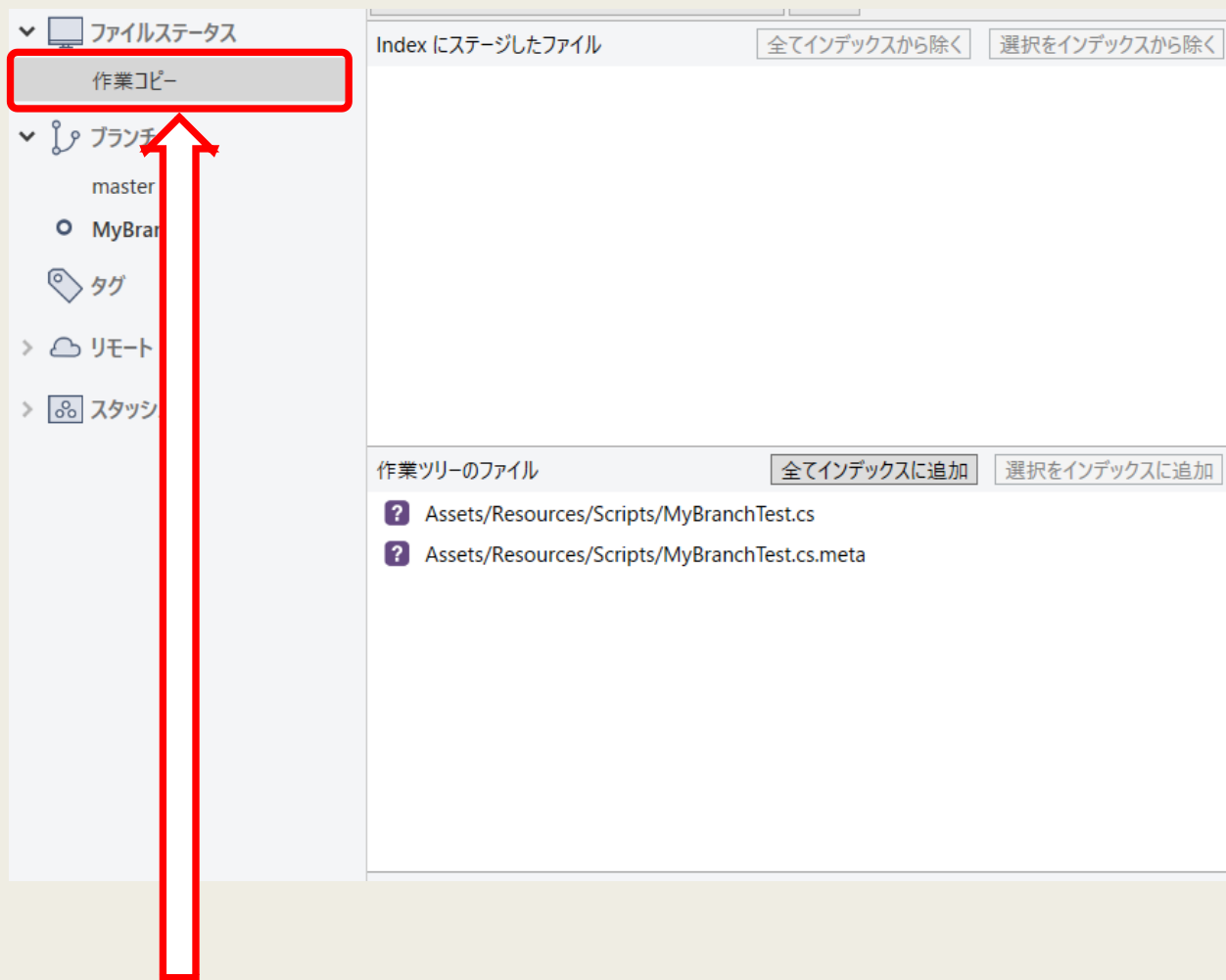
MyBranchTest.csファイルを新規作成する想定で進めていきます。

作業の流れ

- 01.作業内容をコミット

- 02.コミットした内容をプッシュして作業完了

作業用ブランチ更新の流れ



作業した内容が、既に表示されています。

01.SourceTreeを開いて、作業コピーをクリック

作業用ブランチ更新の流れ

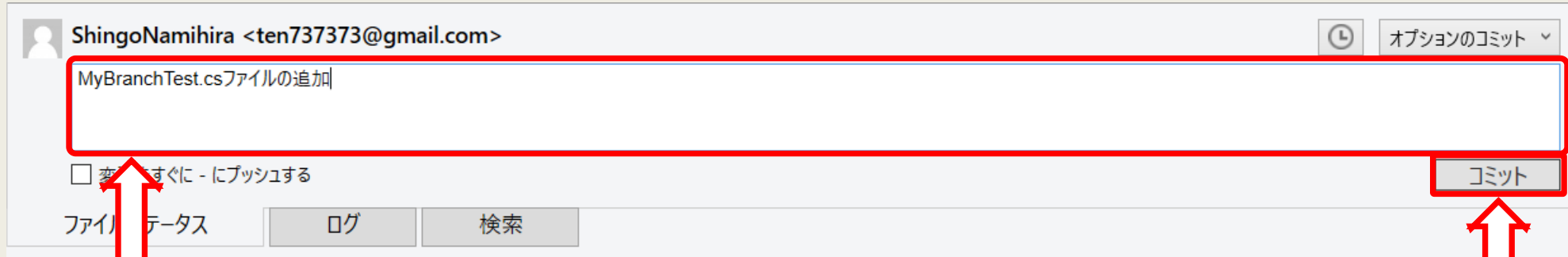
The screenshot displays a software interface with three main panels. The top-left panel, titled 'Index にステージしたファイル', shows a list of files: 'Assets/Resources/Scripts/MyBranchTest.cs' and 'Assets/Resources/Scripts/MyBranchTest.cs.meta', both marked with a green '+' icon. A red box highlights these files, and a blue arrow points from the bottom-left panel to this box. The bottom-left panel, titled '作業ツリーのファイル', shows the same two files marked with a purple '?' icon. A red arrow points from below this panel upwards. The right panel, titled 'Assets/Resources/Scripts/MyBranchTest.cs', shows the content of the selected file, which is a C# script for 'MyBranchTest'.

```
1 + using System.Collections;
2 + using System.Collections.Generic;
3 + using UnityEngine;
4 +
5 + public class MyBranchTest : MonoBehaviour {
6 +
7 +     // Use this for initialization
8 +     void Start () {
9 +
10 +    }
11 +
12 +    // Update is called once per frame
13 +    void Update () {
14 +
15 +    }
16 + }
```

ファイルをクリックすると、そのファイルの変更内容を確認することができます。
(+が増えた分、-が減った分)

02.作業ツリーのファイル内にある、変更したファイルを選択し、インデックスに追加する。
(自分が作業した覚えのないファイルは、コミットせず破棄または削除する)

作業用ブランチ更新の流れ



ShingoNamihira <ten737373@gmail.com> オプションのコミット

MyBranchTest.csファイルの追加

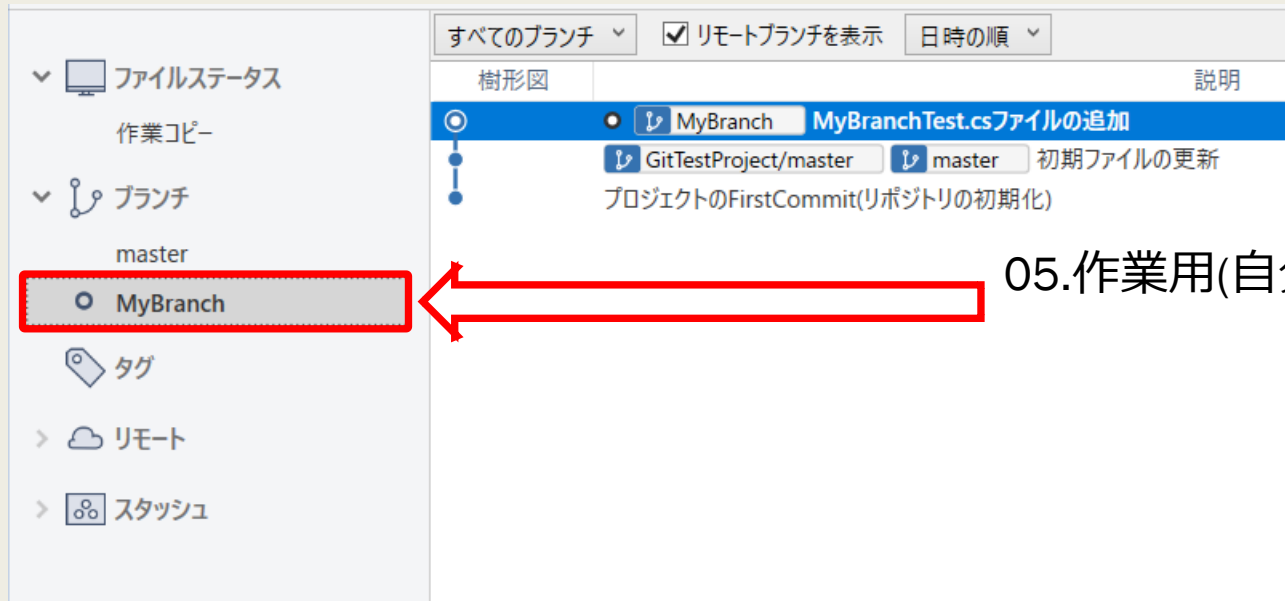
☐ 変更をすぐに - にプッシュする

ファイル ステータス ログ 検索

コミット

03.作業した内容を記入する

04.コミットをクリック



すべてのブランチ リモートブランチを表示 日時の順

ファイルステータス 作業コピー

ブランチ master MyBranch

タグ リモート スタッシュ

MyBranch MyBranchTest.csファイルの追加

GitTestProject/master master 初期ファイルの更新

プロジェクトのFirstCommit(リポジトリの初期化)

05.作業用(自分の)ブランチをクリックすると、
ブランチの情報更新されています。

作業用ブランチ更新の流れ

06. プッシュをクリック

07. MyBranchにチェックを入れる
(このとき、絶対にmasterにはチェックを入れない)

08. プッシュをクリック

プッシュ : GitTestProject

プッシュ先: GitTestProject

対象	ローカルブランチ	リモートブランチ	追跡中
<input type="checkbox"/>	master	master	<input type="checkbox"/>
<input checked="" type="checkbox"/>	MyBranch	MyBranch	<input checked="" type="checkbox"/>

☒ 全て選択

☒ すべてのタグをプッシュ ☐ 強制プッシュ

プッシュ キャンセル

以上で、作業用ブランチ更新の流れは完了です。

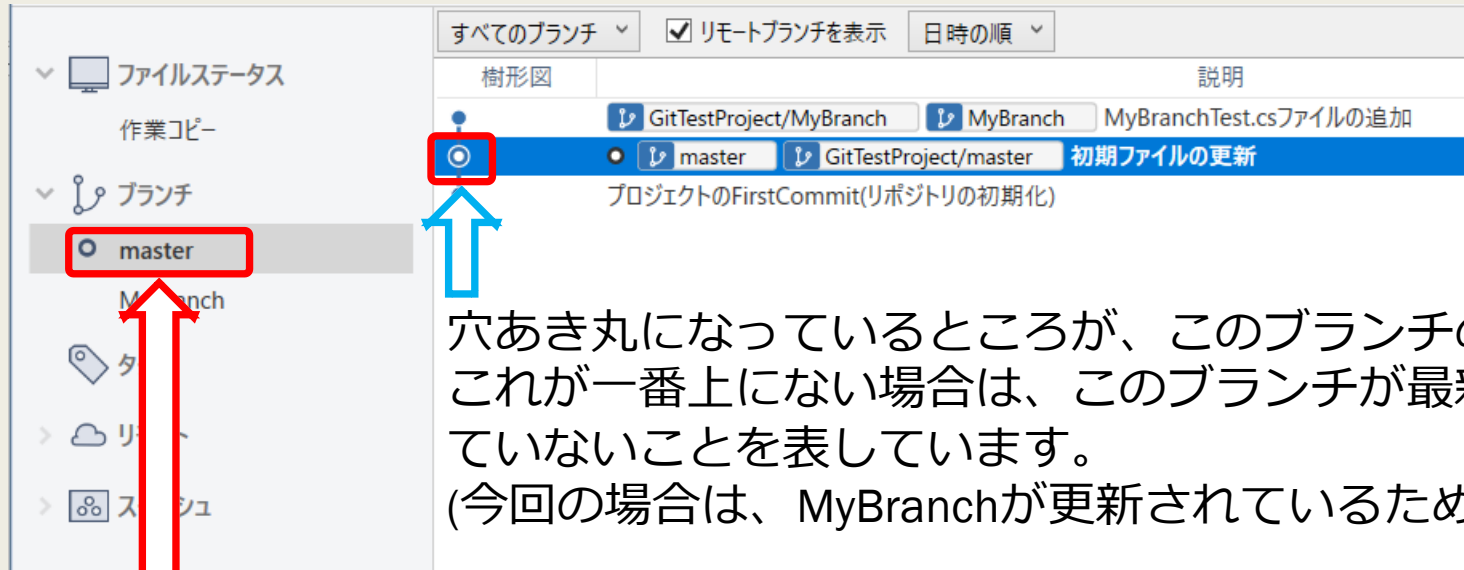
masterへマージする流れ

masterブランチがビルドするブランチになるため、masterブランチを更新する必要があります。
ので、作業ブランチの更新ができた後の、masterブランチへのマージする流れを解説していきます。
・今回は、マージする人を用意せず、各々がマージすることを想定しています。
(プログラマ、デザイナー、プランナー問わず。です。)

作業の流れ

- 01.masterブランチに移動してプル
- 02.大まかに変更内容とエラーがないかプロジェクトを開いてチェック
- 03.作業ブランチに移動(masterにて変更がされている場合はその変更を破棄)
- 04.作業ブランチにmasterブランチをマージ(逆にしないように気を付ける)
- 05.エラーがないか・変更内容が適用されているか、プロジェクトを開いてチェック
- 06.masterブランチに移動する
- 07.masterブランチに作業ブランチをマージする(逆にしないように気を付ける)
- 08.動作確認して問題がないなら作業完了

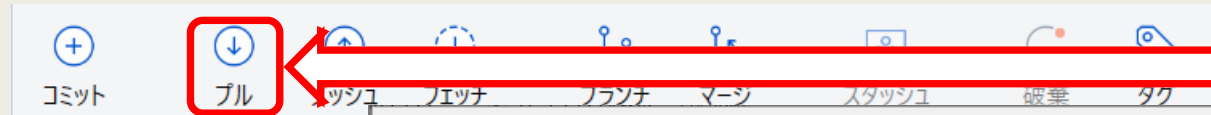
masterへマージする流れ



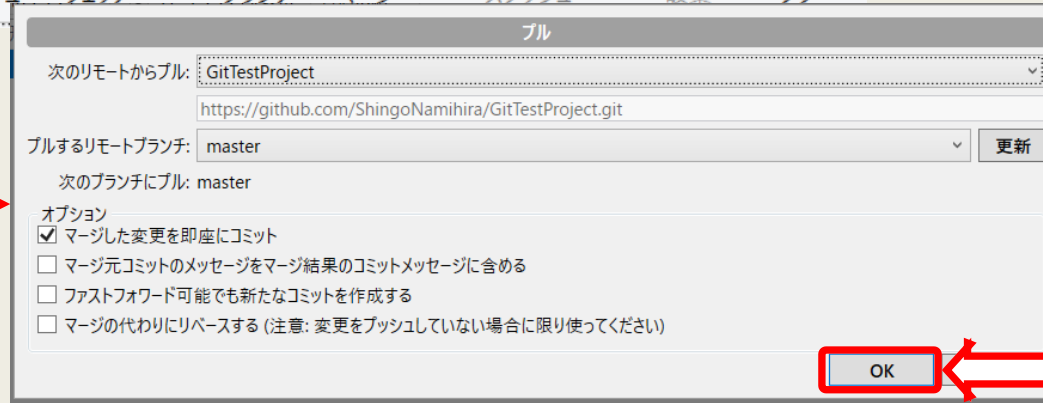
穴あき丸になっているところが、このブランチの更新状況です。
これが一番上にある場合は、このブランチが最新の状態になっていないことを表しています。
(今回の場合は、MyBranchが更新されているため)

01. ブランチのmasterをダブルクリックして、masterブランチに移動する

masterへマージする流れ



02.プルをクリックする

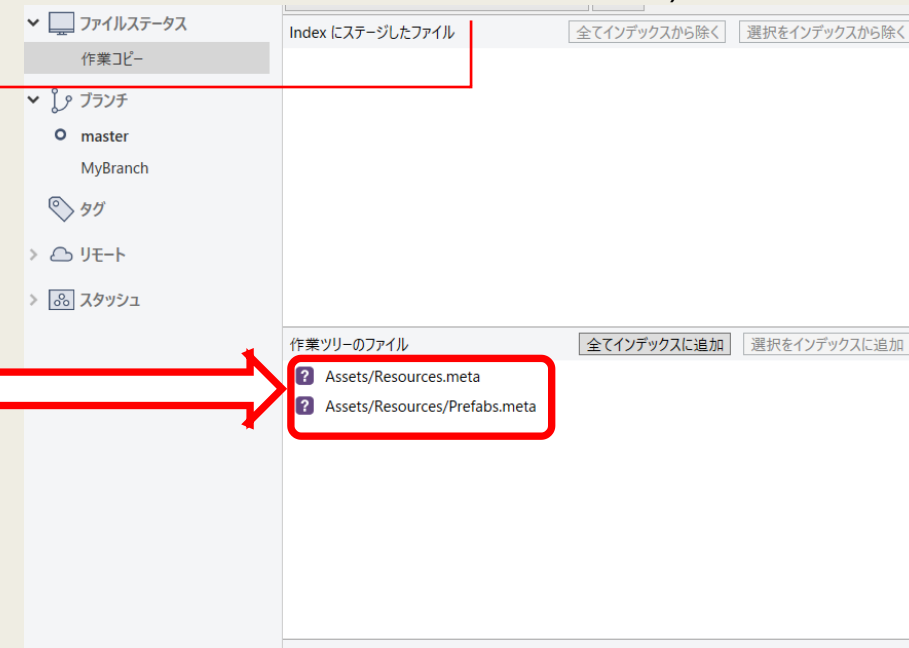


03.OKをクリックする

(masterに変更がされている場合は、ここで変更を適用しています。)

04.プルできたら、プロジェクトを開く。
大まかに変更内容とエラーがないかチェックする。

05.チェックができればプロジェクトを閉じて、
SourceTreeに戻る。
このとき変更が生じた場合は、その変更内容を
破棄または削除する。

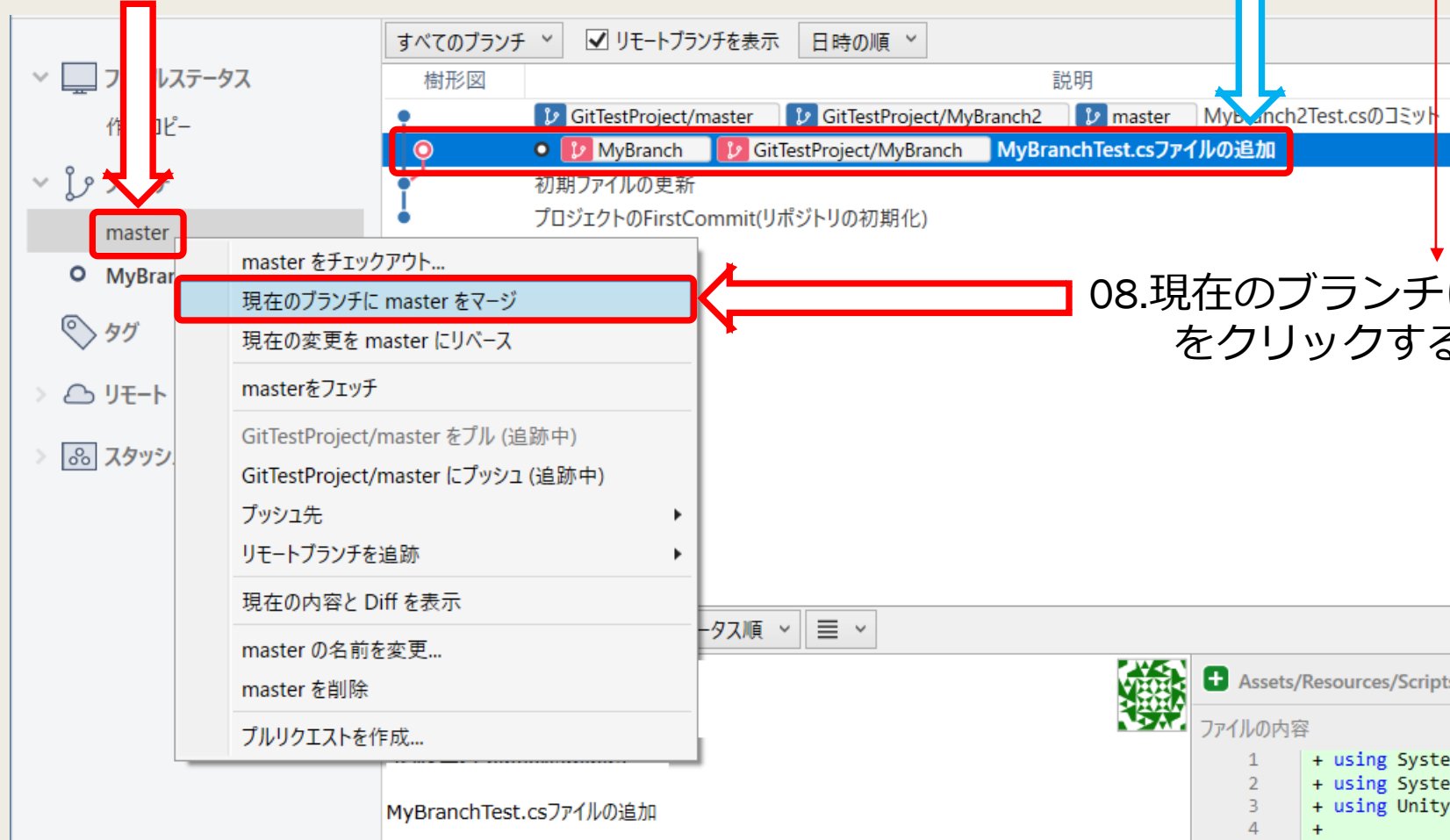


masterへマージする流れ

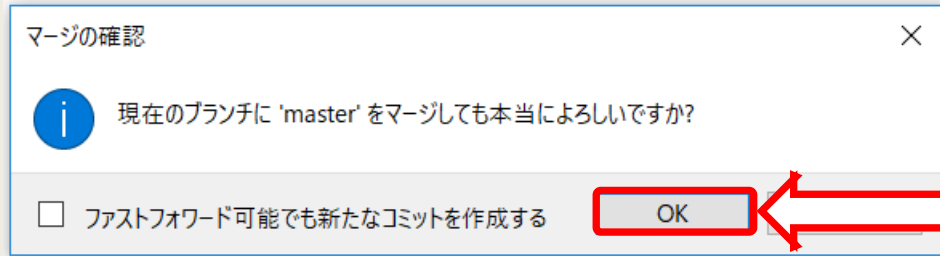
06.作業用(自分の)ブランチに移動する

ブランチごとに、色が分けられています

07.マスターを右クリックする



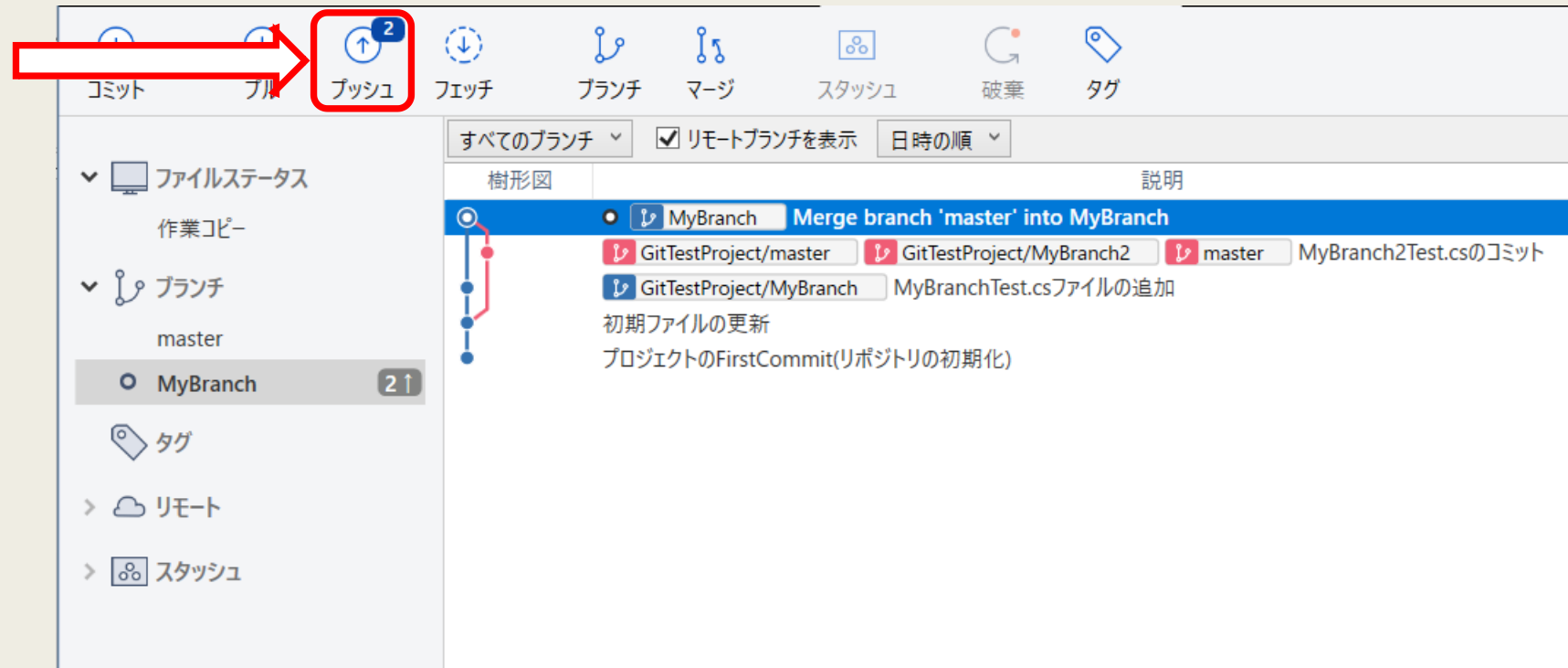
masterへマージする流れ



09.OKをクリックする

10.マージが完了したら、プロジェクトを起動して動作確認する。
(ほかの人の変更内容も、適用されているはずです。)

11.SourceTreeに戻ると、右のような画面になっていると思います。
プッシュをクリックします。



masterへマージする流れ

プッシュ: GitTestProject

プッシュ先: GitTestProject ▼ <https://github.com/ShingoNamihira/GitTestProject.git>

プッシュするブランチ

対象	ローカルブランチ	リモートブランチ	追跡中
<input type="checkbox"/>	master	master ▼	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	MyBranch	MyBranch ▼	<input checked="" type="checkbox"/>

☒ 全て選択

☒ すべてのタグをプッシュ ☐ 強制プッシュ

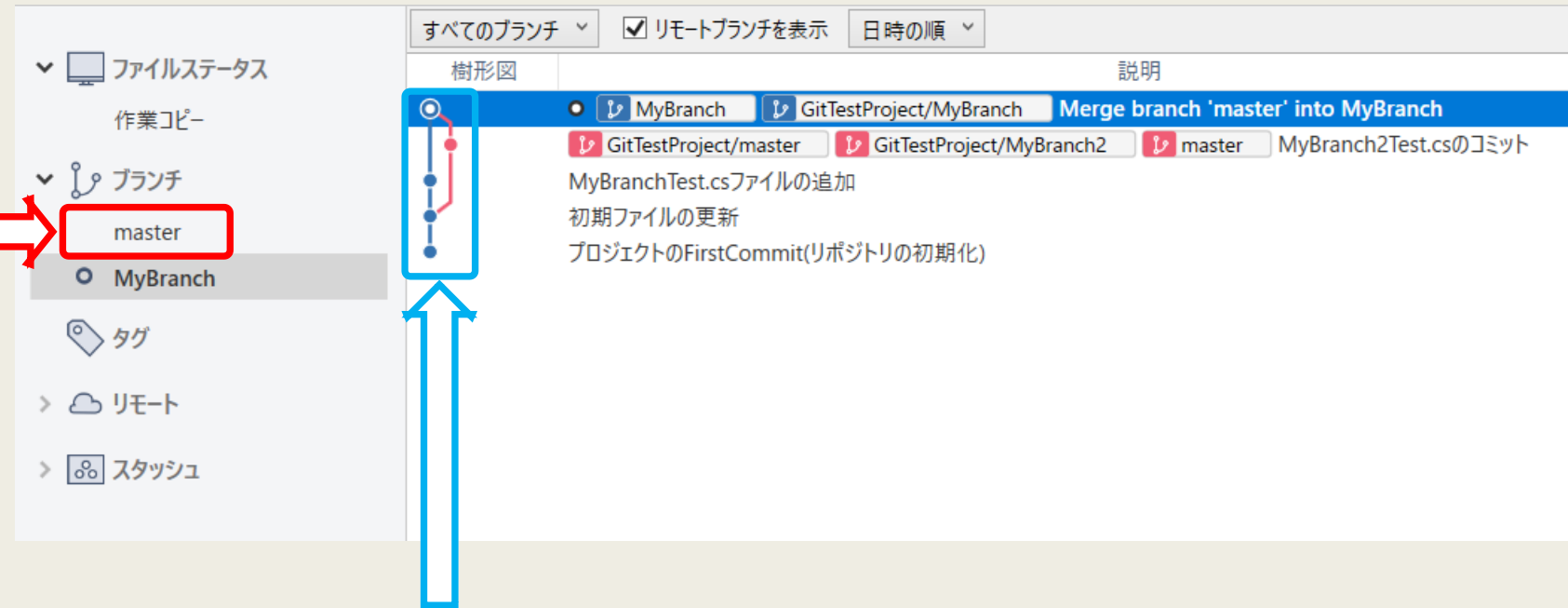
プッシュ

12. 作業用(自分の)ブランチにチェックを入れる
(masterには絶対にチェックを入れないでください)

13. プッシュをクリックする

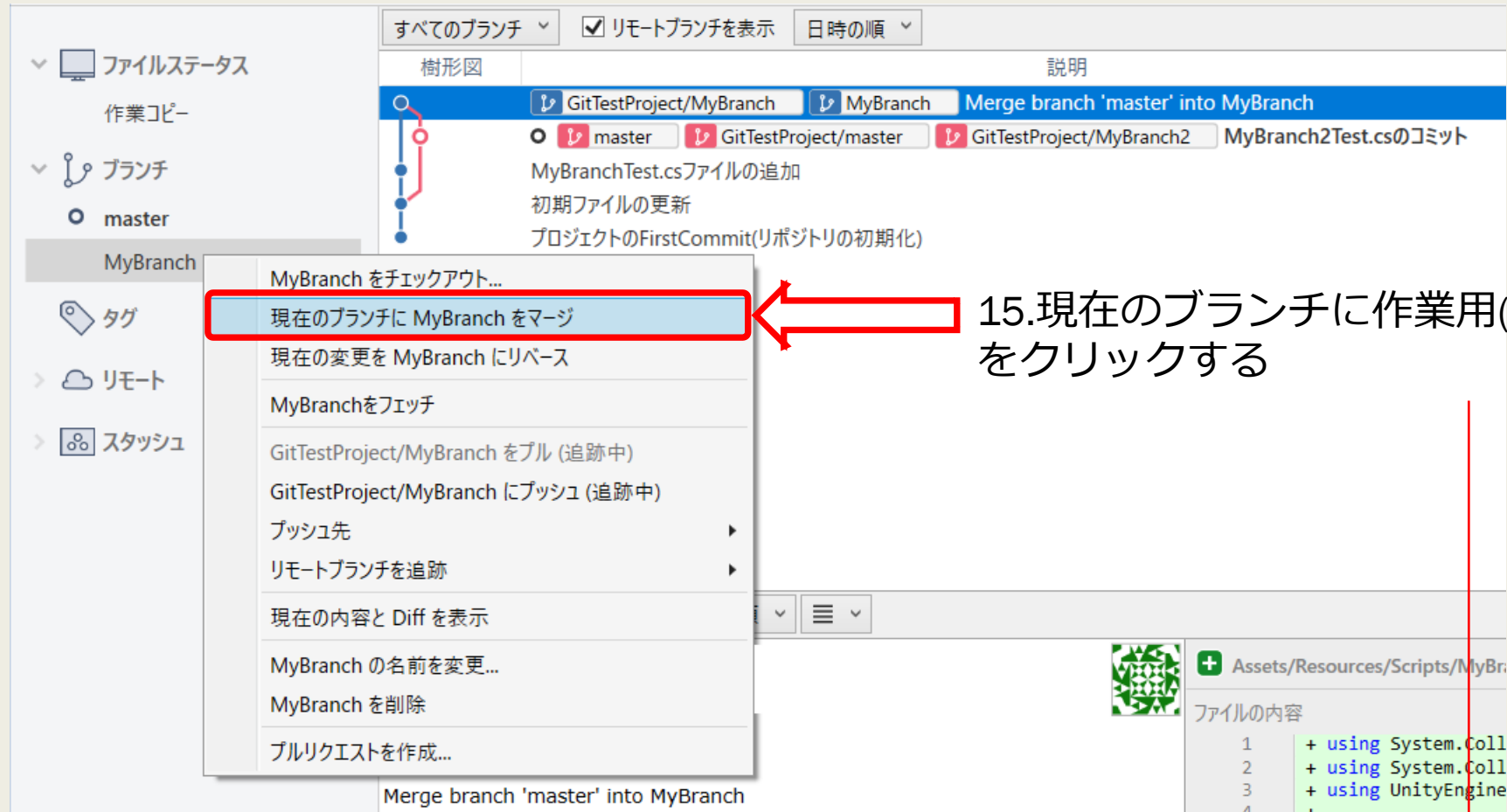
masterへマージする流れ

14.masterブランチを
ダブルクリックして、
移動する



このような表示になっていれば、
作業ブランチにmasterを結合できています。
つまり、**masterにも作業ブランチをマージする必要があります。**

masterへマージする流れ



15.現在のブランチに作業用(自分の)ブランチをマージをクリックする



16.OKをクリックする

masterへマージする流れ

17. プッシュをクリックする

18. プッシュをクリックする

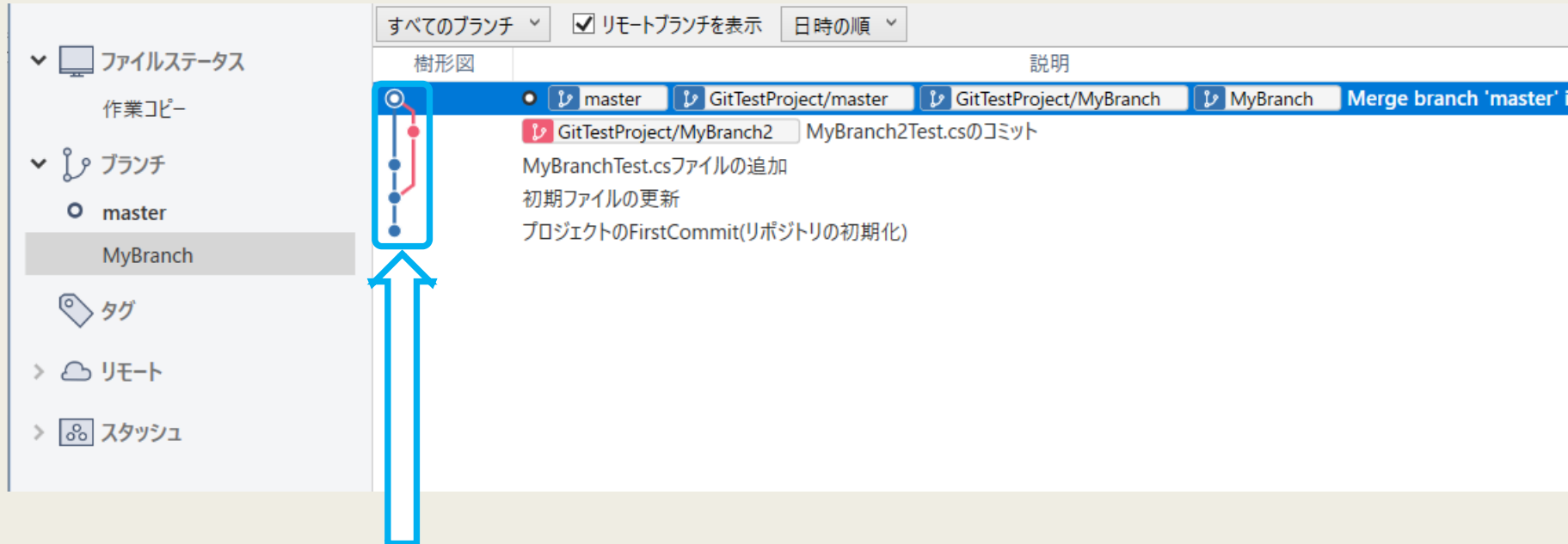
masterにチェックを入れる
(作業用(自分の)ブランチには絶対にチェックを入れない)

対象	ローカルブランチ	リモートブランチ	追跡中
<input checked="" type="checkbox"/>	master	master	<input type="checkbox"/>
<input type="checkbox"/>	MyBranch	MyBranch	<input type="checkbox"/>

■ 全て選択

プッシュ キャンセル

masterへマージする流れ



このような表示になっていれば、masterに作業用ブランチのマージ完了です。

19.プロジェクトを起動して、動作確認をする。

20.問題がないことが確認できたら、SourceTreeを開き、
作業用(自分の)ブランチに必ず戻して、再度作業に戻る。

以上で、masterへマージする流れは完了です。

Conflict(衝突)の解消方法

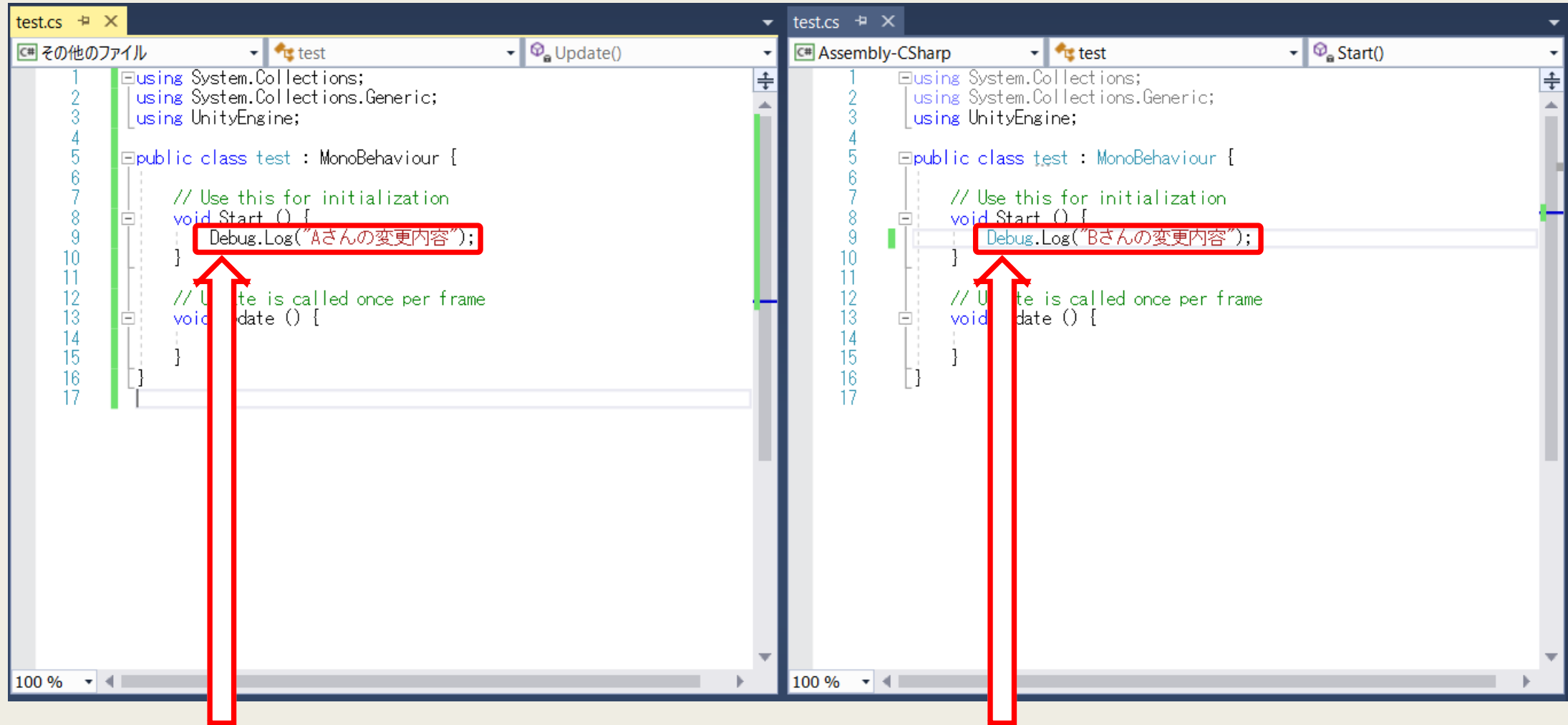
- ・ Conflict=衝突という意味です。
- ・ マージしたときなどに発生するもので、同じファイルを修正したりすると起こることがあります。今回は、その解決方法を解説していきます。

作業の流れ

・ 今回は、Aさん・Bさんが同じファイル(ソースファイル)を修正したためにConflictが起きたと想定し、Aさんの変更内容を優先して残し、そのあとにBさんの変更内容を適用する。という流れで進めていきます。

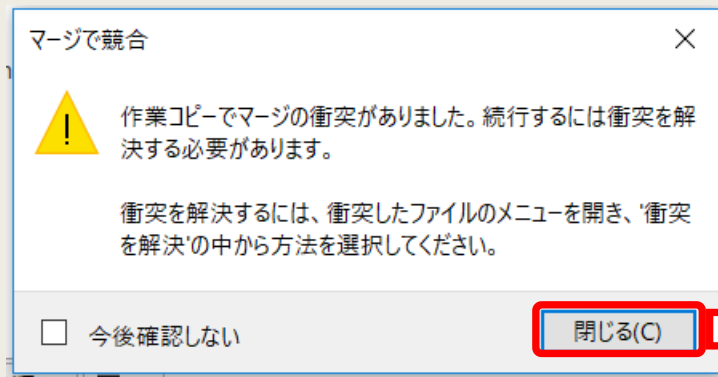
- 01.Bさんのブランチにmasterブランチをマージする
- 02.Conflictが起きたので、一度変更済みファイルを別ファイルとしてバックアップする
- 03.Aさんの変更内容を優先して、Bさんのブランチにマージする
- 04.Bさんのブランチで、バックアップを取った内容を再度修正する
- 05.エラー等がないか、動作確認をする
- 06.masterブランチに移動する
- 07.masterブランチに、作業用ブランチをマージする
- 08.エラー等がないか、動作確認をする
- 09.作業用ブランチに移動する

Conflict(衝突)の解消方法



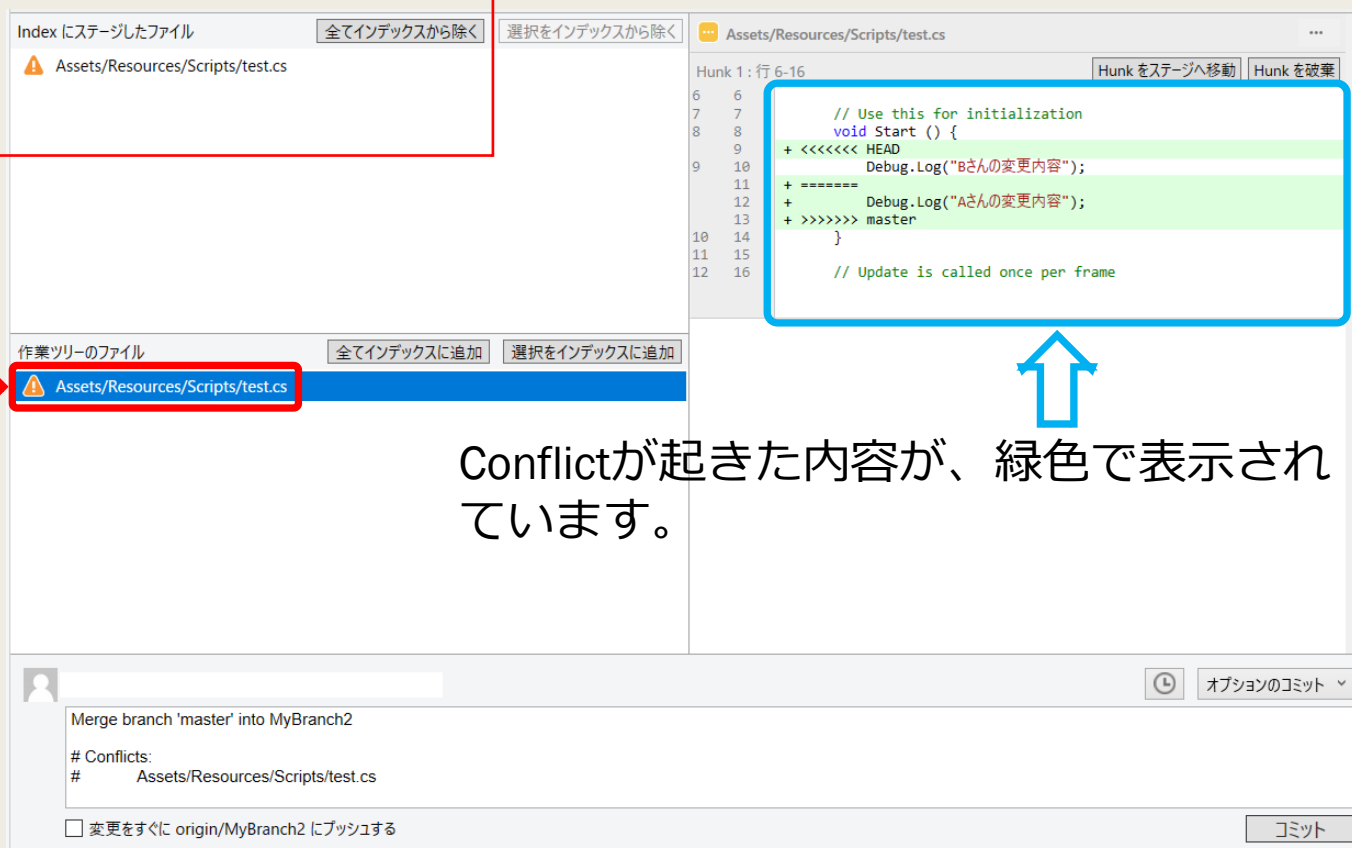
test.csファイルの、この部分がそれぞれ修正した内容です。

Conflict(衝突)の解消方法



01. マージをしようとする、このように表示されます。
閉じるをクリックしてください。

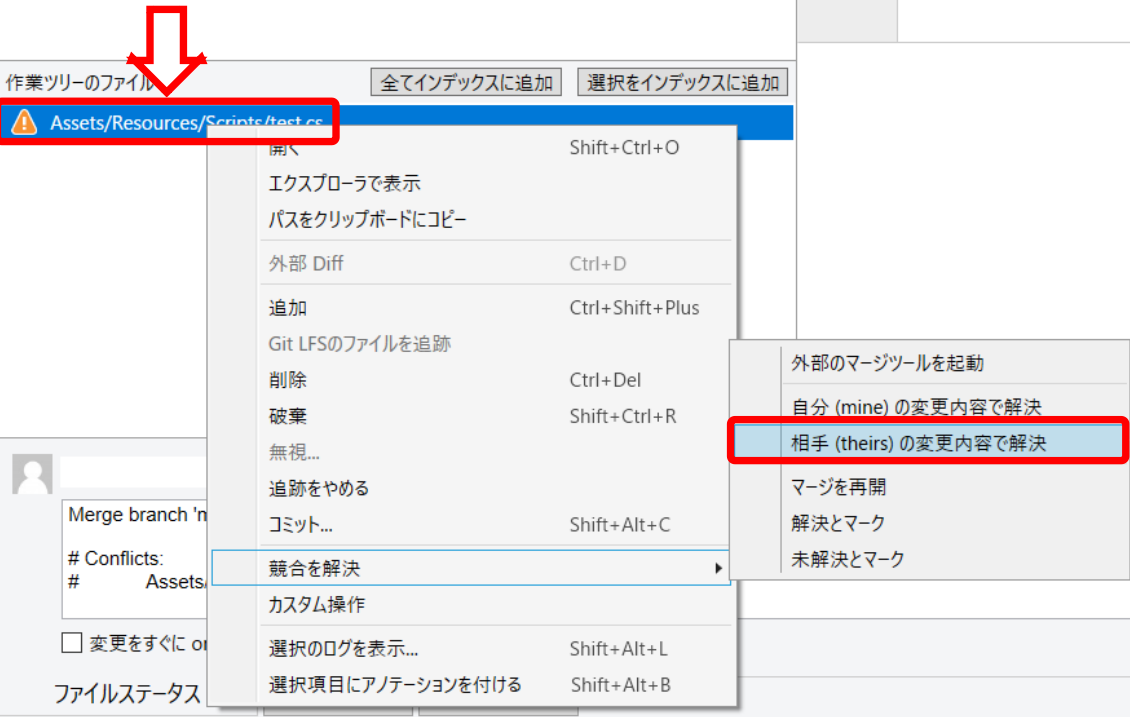
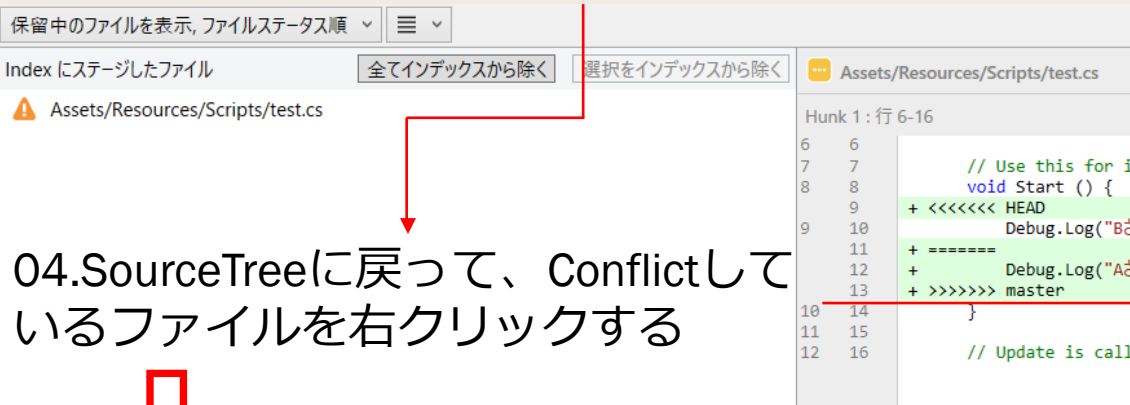
02. 作業コピーを開くと、Conflictしたファイルに！マークがついています。



Conflictが起きた内容が、緑色で表示されています。

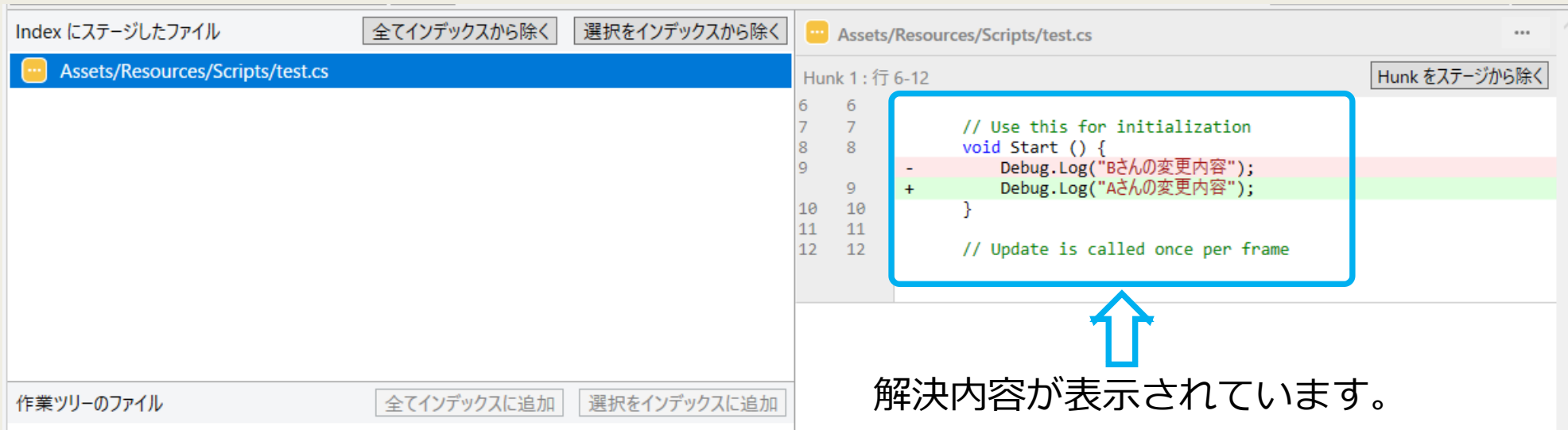
Conflict(衝突)の解消方法

03.Conflictが起きているファイルをバックアップする



05.競合を解決の中の、
相手(theirs)の変更内容で解決をクリックする
(この場合、相手はAさん、自分はBさん。
Aさんの変更内容で解決する。ということです。)

Conflict(衝突)の解消方法



Index にステージしたファイル 全てインデックスから除く 選択をインデックスから除く

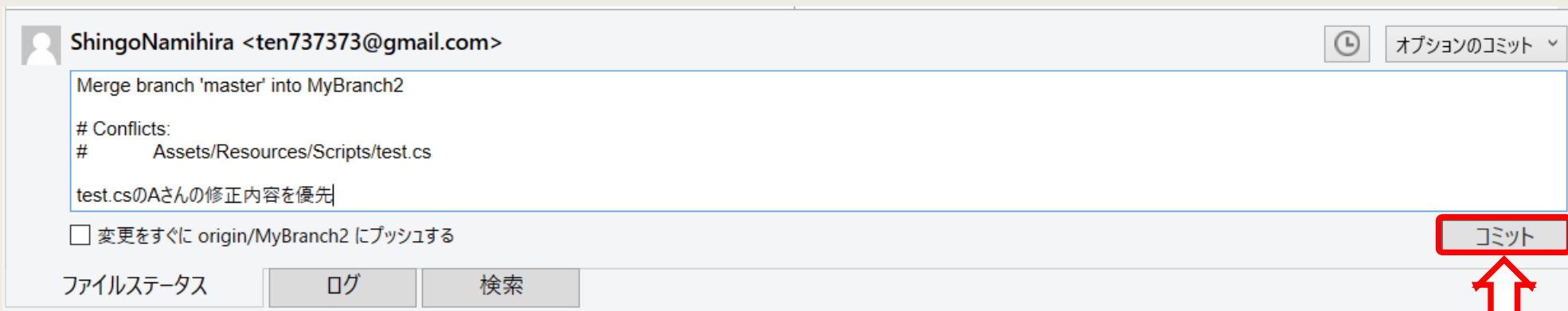
Assets/Resources/Scripts/test.cs

Hunk 1 : 行 6-12 Hunk をステージから除く

```
6 6 // Use this for initialization
7 7 void Start () {
8 8     Debug.Log("Bさんの変更内容");
9 9     Debug.Log("Aさんの変更内容");
10 10 }
11 11 // Update is called once per frame
12 12
```

作業ツリーのファイル 全てインデックスに追加 選択をインデックスに追加

解決内容が表示されています。



ShingoNamihira <ten737373@gmail.com> オプションのコミット

Merge branch 'master' into MyBranch2

Conflicts:
Assets/Resources/Scripts/test.cs

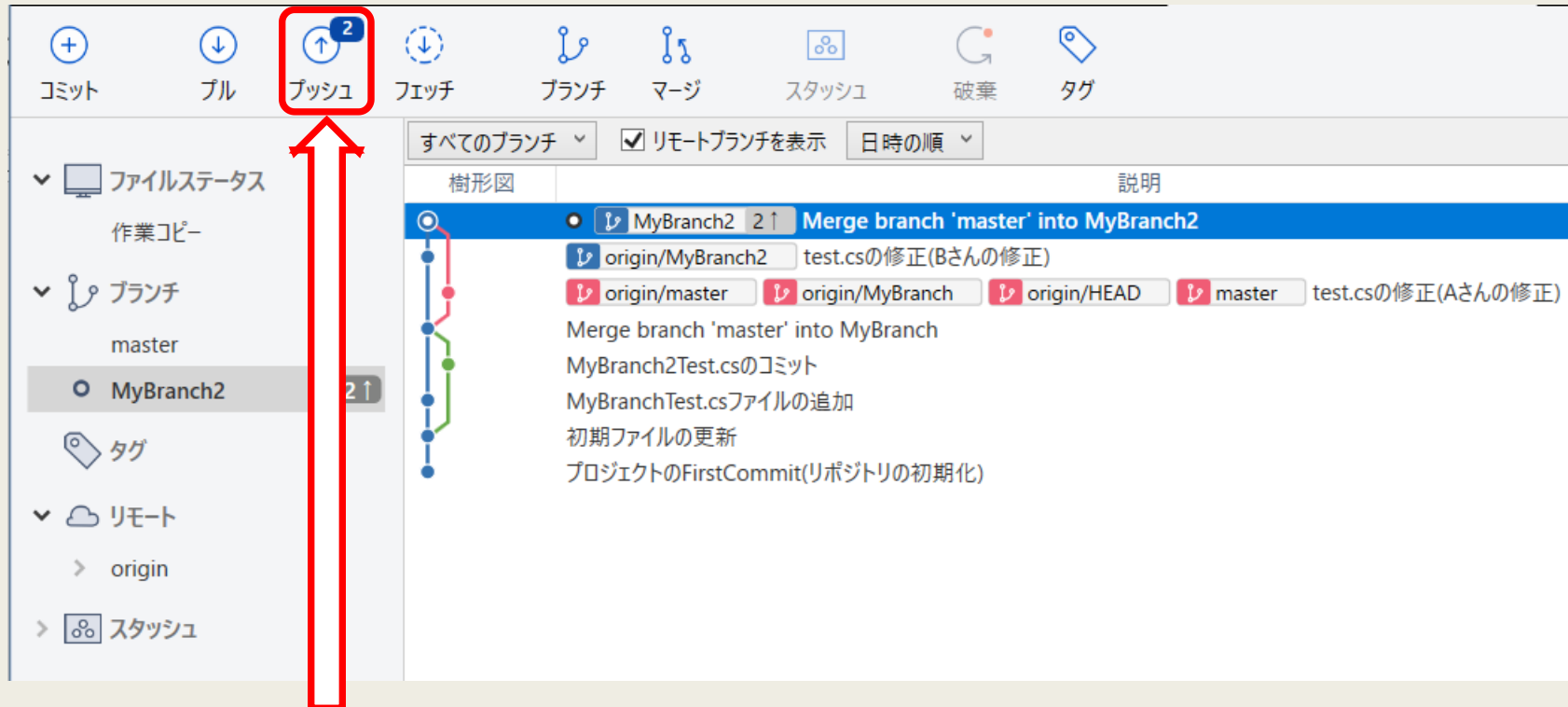
test.csのAさんの修正内容を優先

☐ 変更をすぐに origin/MyBranch2 にプッシュする

ファイルステータス ログ 検索 **コミット**

06.修正内容を記入し、コミットをクリックする

Conflict(衝突)の解消方法



07.コミットした内容をプッシュする

08.Conflictを解決したファイルを開き、改めてBさんの修正分を追加する

Conflict(衝突)の解消方法

08.Conflictを解決したファイルを開き、改めてBさんの修正分を追加する

```
test.cs* test.cs X
Assembly-CSharp test Start()
1 using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class test : MonoBehaviour {
6
7       // Use this for initialization
8       void Start () {
9           Debug.Log("Aさんの変更内容");
10          Debug.Log("Bさんの変更内容");
11      }
12
13      // Update is called once per frame
14      void Update () {
15
16      }
17  }
18
```

今回は、Aさんの修正分に追加でBさんの修正を加えました。

Conflict(衝突)の解消方法

The screenshot shows a code editor interface with the following elements:

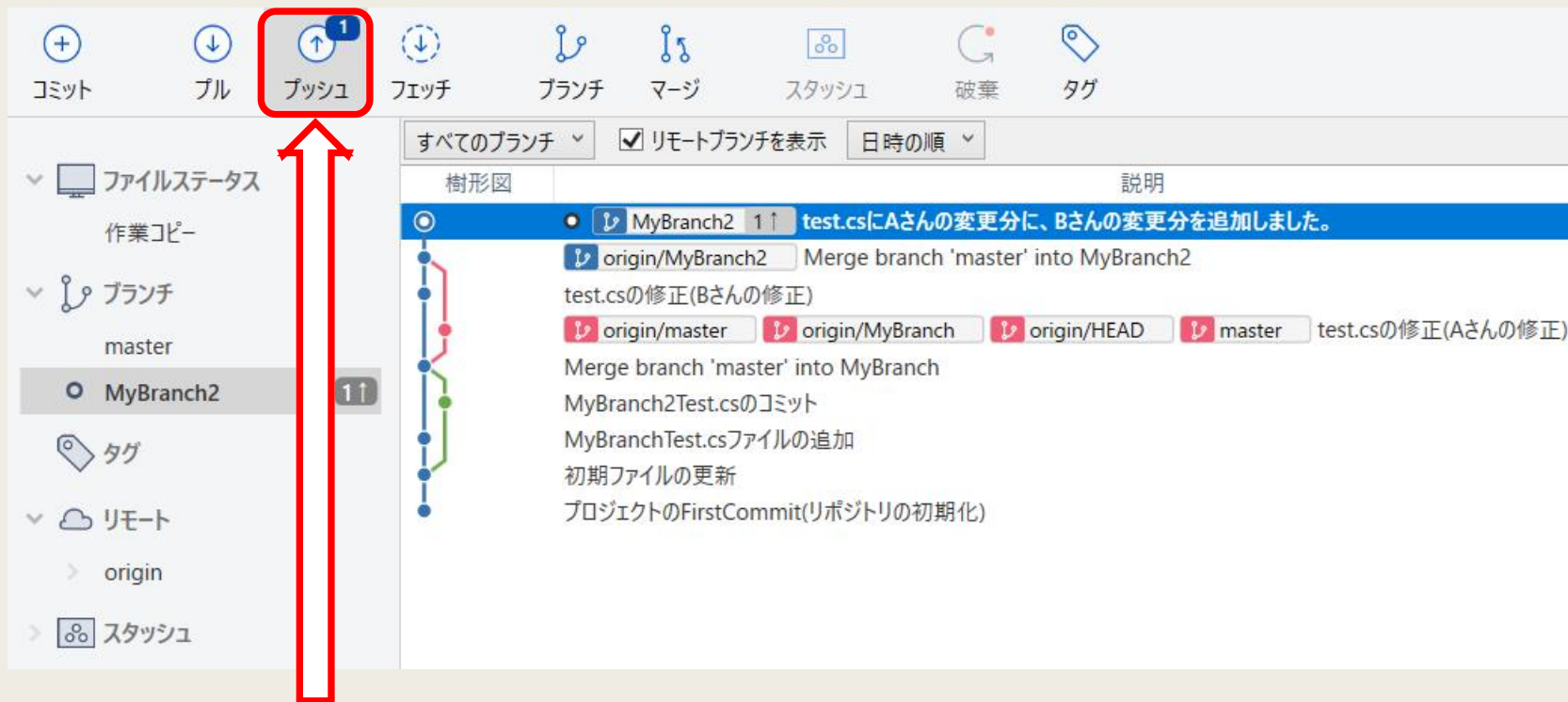
- Top Panel:** "Index にステージしたファイル" (Files staged for index). A file "Assets/Resources/Scripts/test.cs" is listed and highlighted with a red box. Buttons "全てインデックスから除く" (Remove from all index) and "選択をインデックスから除く" (Remove selection from index) are present.
- Left Panel:** "作業ツリーのファイル" (Files in working tree). Buttons "全てインデックスに追加" (Add to all index) and "選択をインデックスに追加" (Add selection to index) are present.
- Right Panel:** Code editor showing "Assets/Resources/Scripts/test.cs". It displays a conflict hunk with line numbers 7-14. The code includes initialization and update functions with debug logs for "Aさん" and "Bさん".
- Bottom Panel:** Commit message input area. The message "test.csにAさんの変更分に、Bさんの変更分を追加しました。" (Added B's changes to A's changes in test.cs) is entered. There is a checkbox "変更をすぐに origin/MyBranch2 にプッシュする" (Push changes to origin/MyBranch2 immediately) and a "コミット" (Commit) button highlighted with a red box.

Red arrows indicate the workflow: one arrow points from the file "Assets/Resources/Scripts/test.cs" in the top panel to the commit message input area, and another arrow points from the commit message input area to the "コミット" button.

09.修正したファイルを
インデックスに追加する

10.コミットする内容を記入し、
コミットをクリックする。

Conflict(衝突)の解消方法



10.変更した内容をプッシュする

11.masterブランチに移動する

12.masterブランチにてプルする

(ここで変更があった場合は、再度作業用ブランチに戻り、作業用ブランチにmasterブランチをマージする。)

Conflict(衝突)の解消方法

13.masterブランチに作業用ブランチをマージする

14.プッシュをする

14.Masterブランチから、作業用ブランチに移動して作業は完了です。



作業用ブランチとmasterブランチが、同じ更新状況になっていることがわかります。
この状態になっていれば、OKです。