Github

2019/10/23

木南　貴志

1.Githubとは

Githubとは

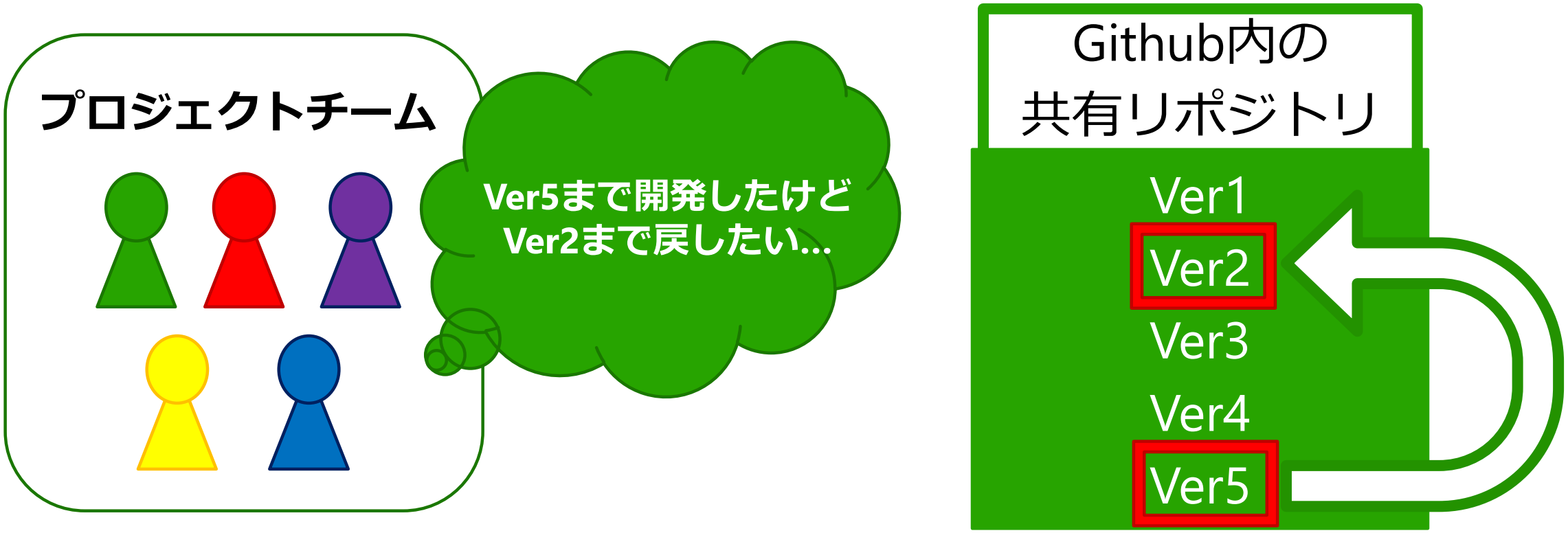
ソフトウェア開発のプラットフォーム

複数人でプログラムを開発するときによく用いられる

Githubとは

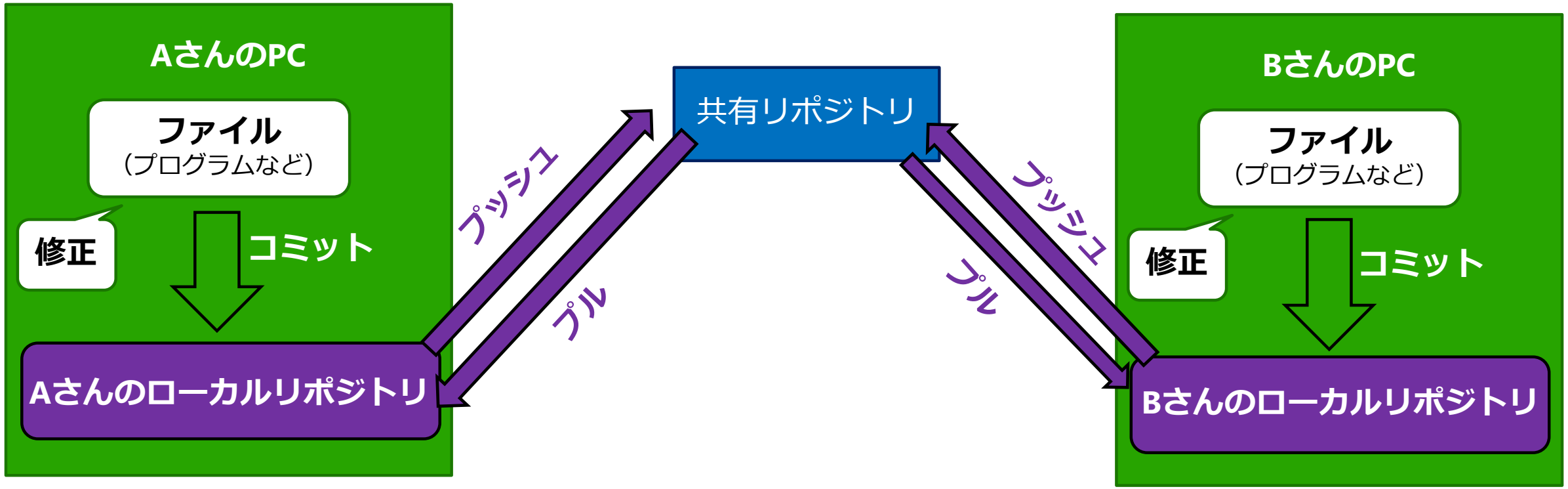
バージョン管理システムがあり、ファイルの履歴を残しておくこと

で以前の状態に復元可能



分散型バージョン管理システム

- リポジトリ：ファイルやディレクトリを保存する場所
- コミット：ローカルリポジトリに記録すること
- プッシュ：ローカルリポジトリの内容を共有リポジトリに反映すること
- プル：共有リポジトリの内容をローカルリポジトリに反映すること



Githubを使うための準備

・gitのインストール

```
$ sudo apt-get update
$ sudo apt-get install git
```

・ユーザの設定（.gitconfigの設定）

```
$ git config --global user.name “NAME”
$ git config --global user.email “E-MAIL”
```

※NAMEとE-MAILは各自で設定（Githubの公式サイトで登録したもの）

エディタの設定（エディタをvim形式に変更）

```
$ git config --global core.editor “vim –f”
```

2.リポジトリ、プッシュ、プル

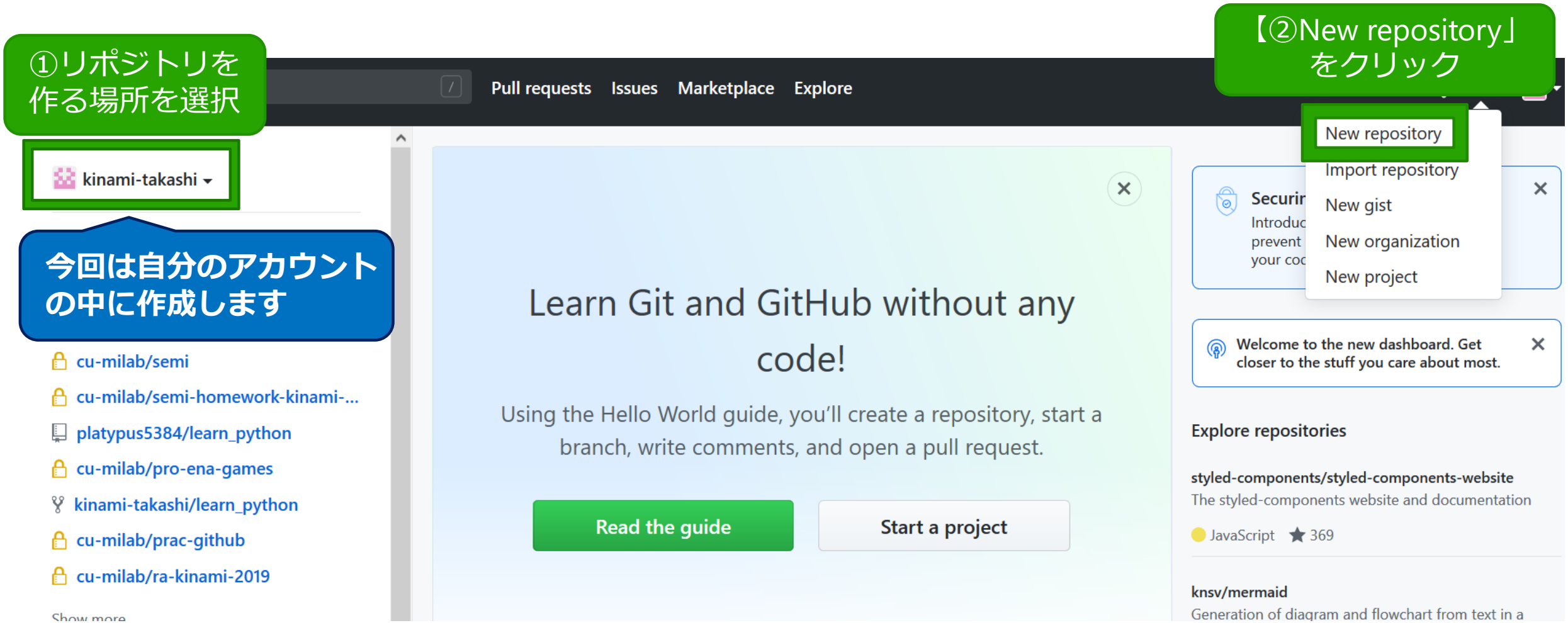
リポジトリの種類

- Publicリポジトリ
  - すべての人に公開できる
    - **機密情報は含めないようにする**

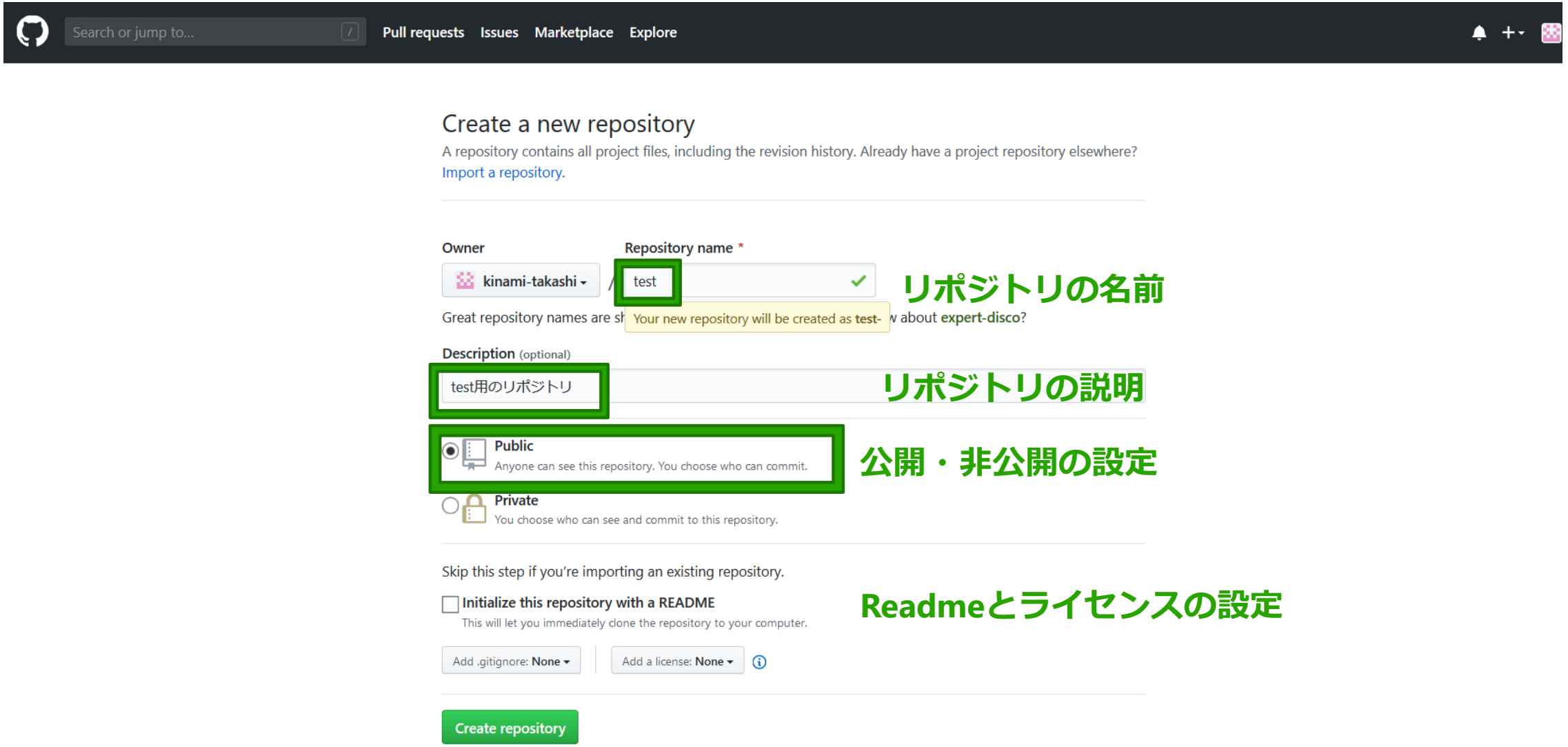
- Privateリポジトリ
  - 基本的に有料
  - 権限を持っている人に対してのみ公開される

リモート（共有）リポジトリ

・GithubのWebページへ　https://github.co.jp/



リモート（共有）リポジトリ



リモート（共有）リポジトリ

ローカルリポジトリから
リモートリポジトリにプッシュ

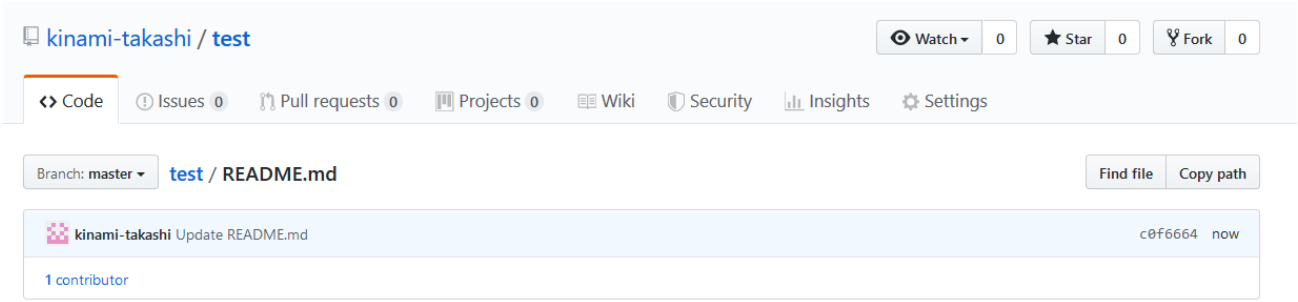
リモートリポジトリから
リモートリポジトリにプッシュ

他のバージョン管理方法からイ
ンポート

作業用ディレクトリの作成

今回は作業用ディレクトリ内にREADME.mdを作成

-ディレクトリのトップにREADME.mdを作成するとリポジトリのトップに表示される



README.mdに作成した内容はここに表示される

作業用ディレクトリの作成

今回は作業用ディレクトリ内にREADME.mdを作成

-ディレクトリのトップにREADMEを作成するとトップに表示される

```
$ mkdir test
$ cd test
$ vim README.md
```

リモートリポジトリ（11ページで作成したもの）と同じ名前にする

作業用ディレクトリの作成

README.mdの記入例）

```
test
====
これはテスト用のリポジトリです
## README.mdとは
README.md内にはリポジトリの説明を記入します

README.mdには特有の書き方が存在します

たとえば【## タイトル】と記入すると見出しを作成することができます
```

13ページの例を再現

ローカルリポジトリの作成

共有したいディレクトリに移動

```
$ git init
```

ローカルリポジトリの作成が成功すると.gitディレクトリが作成されるのでlsで確認

※-aは隠しファイルを見るコマンド

```
$ ls –a

. .. .git test
```

インデックスに追加

・作業ディレクトリのファイル等をインデックスに追加

（gitで管理したいファイル等を登録すること）

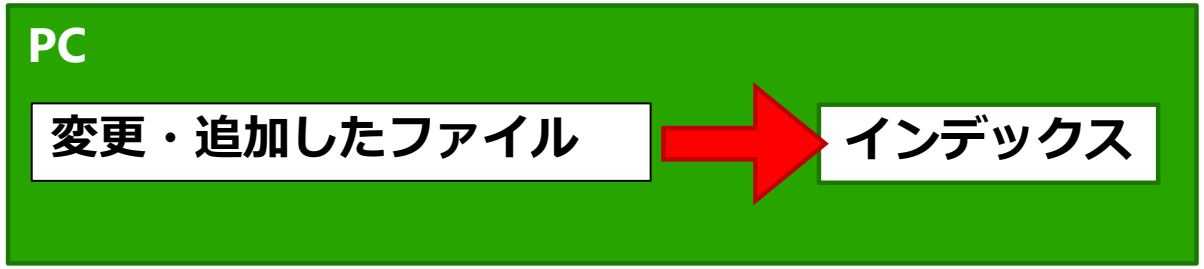
・ファイルを一つだけ指定する方法

```
$ git add [file]
```

・全てのファイル（新規作成、変更されたもの）を指定する方法

```
$ git add .
```

今回はこっち

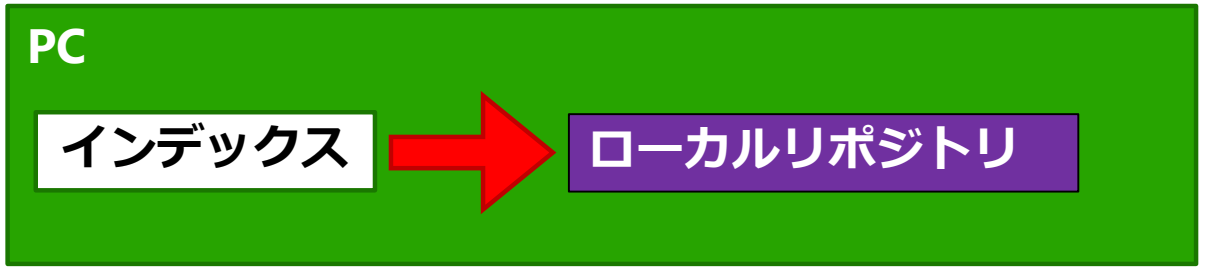


インデックスに記録されたデータをコミット

・ローカルリポジトリに記録

・-mはコメント機能、必須ではないが記述するのが通例

```
$ git commit –m “コメント”
```



インデックスの取り消しと確認

間違えて登録した時は取り消すこともできる

・登録したものの中で一つだけ取り消す

```
$ git reset HEAD [file]
```

・登録したもの全てを取り消す

```
$ git reset HEAD
```

・インデックスに登録されているファイルを確認

```
$ git status
```

リモートリポジトリを登録

・URLと識別子（名前）をつける

※新しくリポジトリを作成した時のみ実行する

```
$ git remote add [short-name] [URL]
```

・登録の例

よくつかわれる名前（違う名称でも可）

```
$ git remote add origin https://github.com/kinami-takashi/test.git
```

リモートリポジトリ作成時のURL

上記の場合 origin = https://github.com/kinami-takashi/test.git

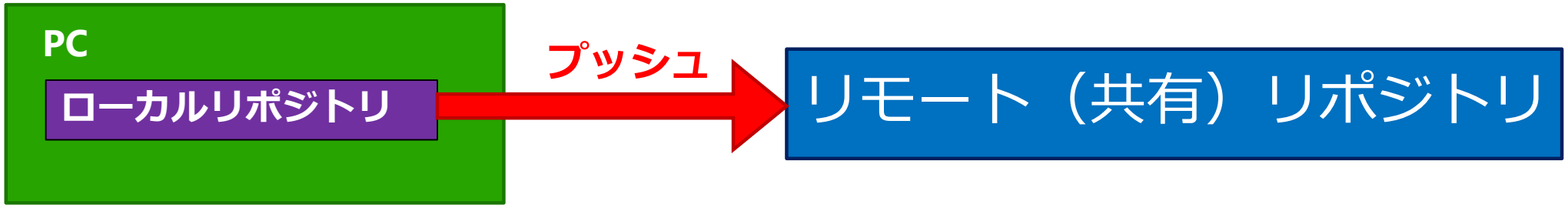
※以降の説明では識別子はすべてoriginとして説明

リモートリポジトリへ送信（プッシュ）

Githubにアップロード

```
$ git push –u origin master
```

※ master=メインのブランチ



リモートリポジトリを取得（プル）

ローカルリポジトリを更新

・originが設定済みの場合のみ実行可能

```
$ git pull origin master
```

例）

Bさんが、ローカルリポジトリのファイルを追加・修正しプッシュして、リモートリポジトリを更新

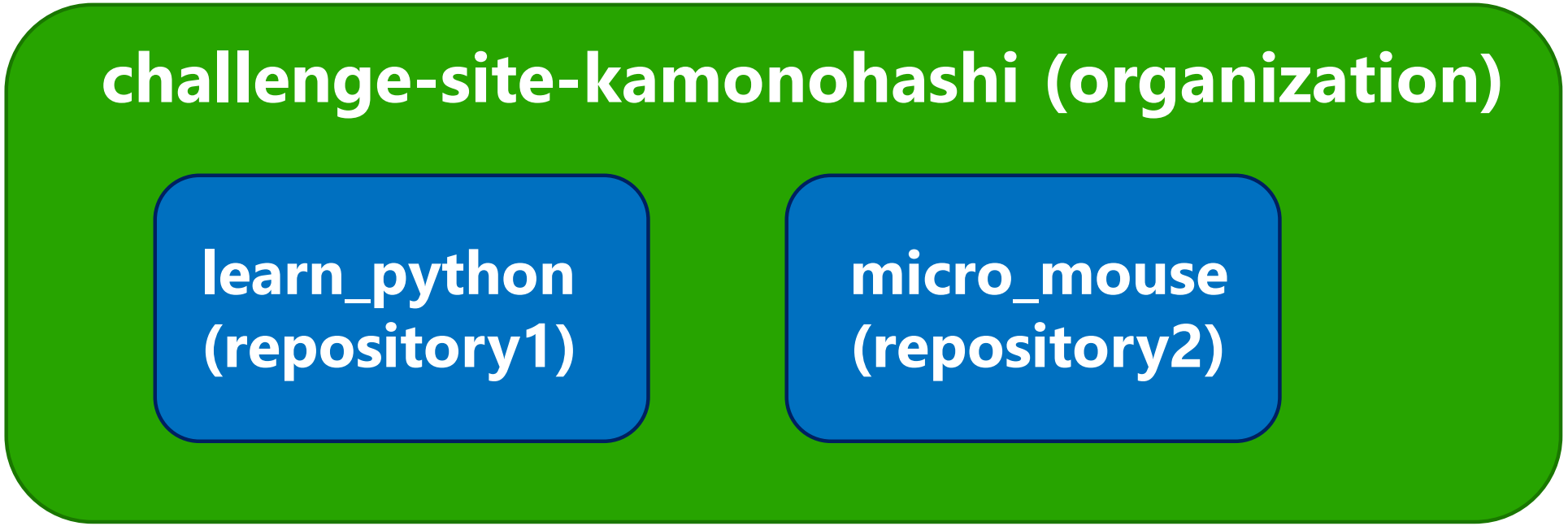
➡Aさんが更新内容を確認するためにリモートリポジトリをプルして自身のローカルリポジトリを更新

Organization

organizationはチームのようなもの

同じチームで色々なリポジトリを共有したい時などに使用される

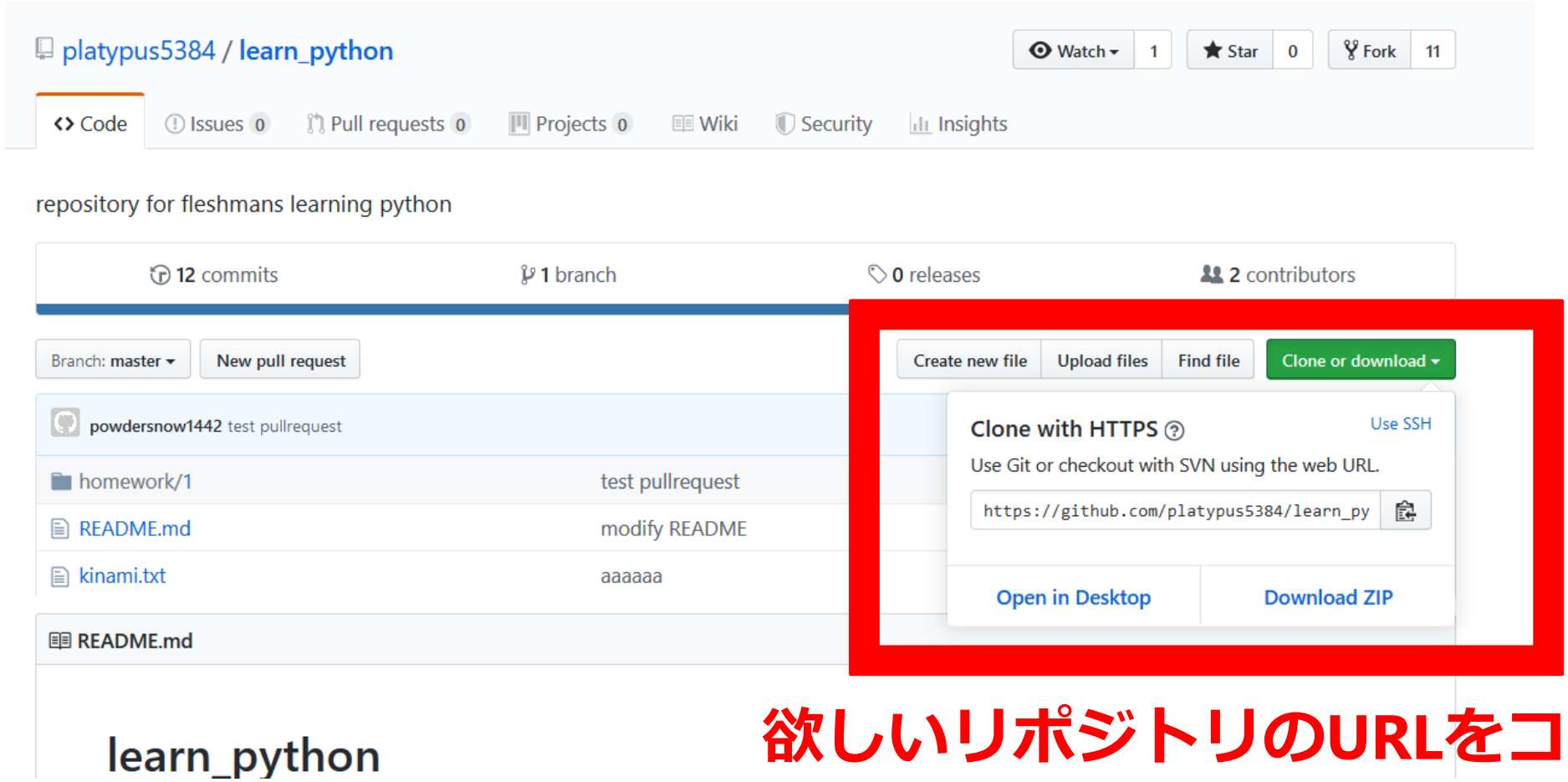
イメージ)



3.クローン

リポジトリのクローン

常に作成してあるリポジトリが欲しい➡ クローン



欲しいリポジトリのURLをコピー

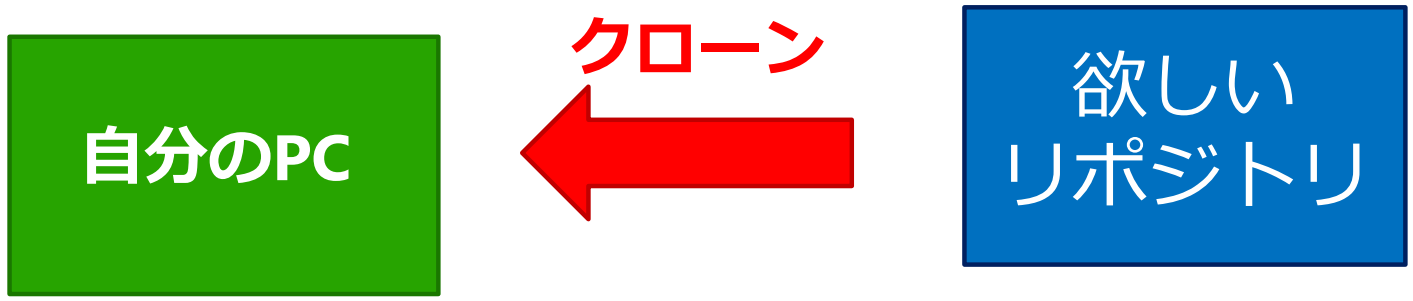
クローンの仕方

クローンの方法

```
$ git clone [URL]
```

今回は以下を実行

```
$ https://github.com/challenge-site-kamonohashi/learn_python.git
```

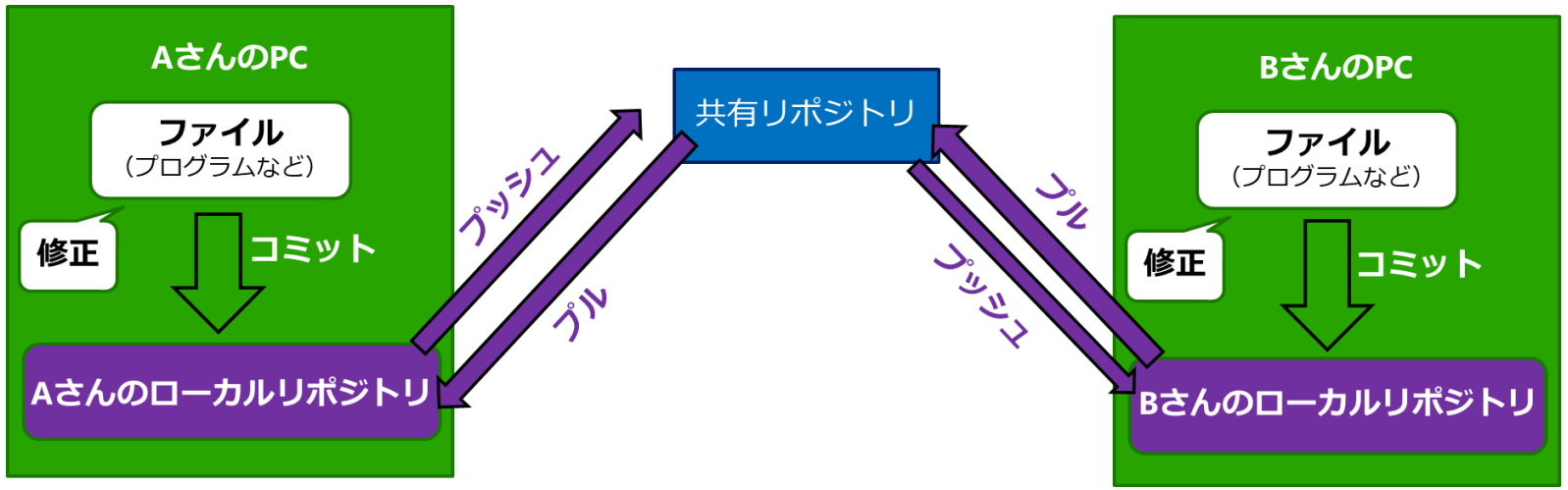


4.ブランチ

ブランチとは

前述ではファイル内の変更した内容をmasterでプッシュしていた

➡複数人で作業する時に全員がmasterでプッシュすると不具合の可能性



全員が好き勝手にpush

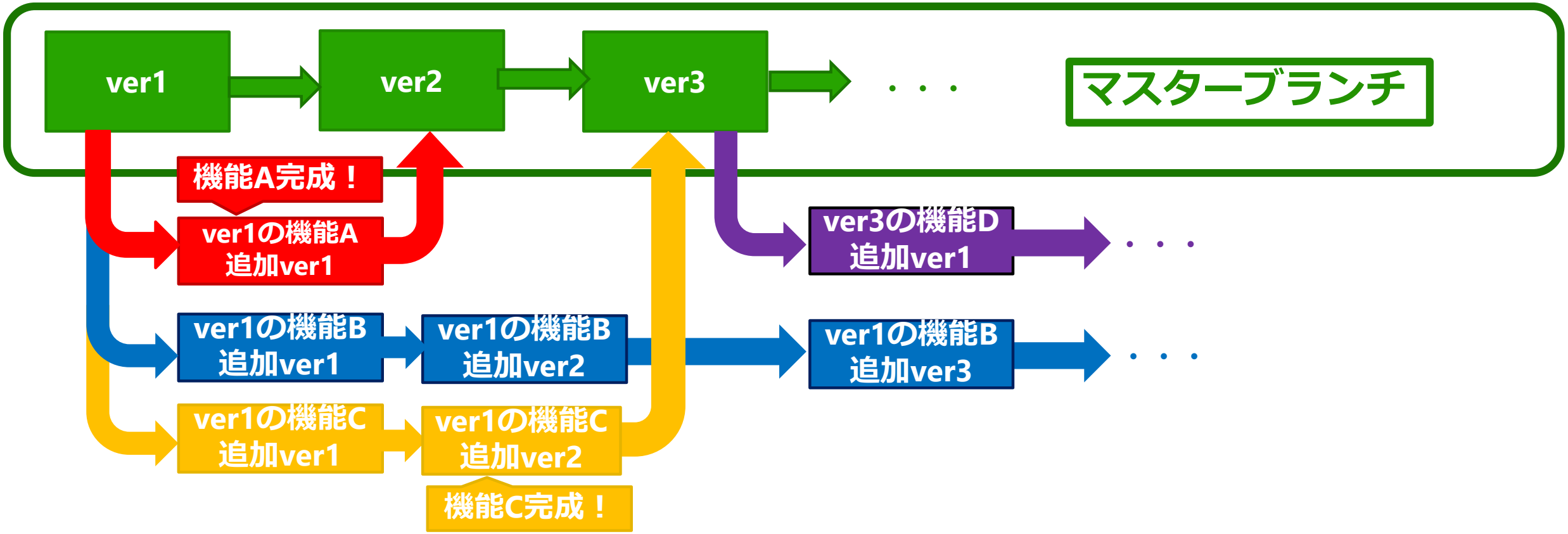
- **プログラムが動かなくなった**
- **変更点が分からなくなった** など

ブランチを使用する

ブランチとは

ブランチは履歴の流れを分岐して記録するもの

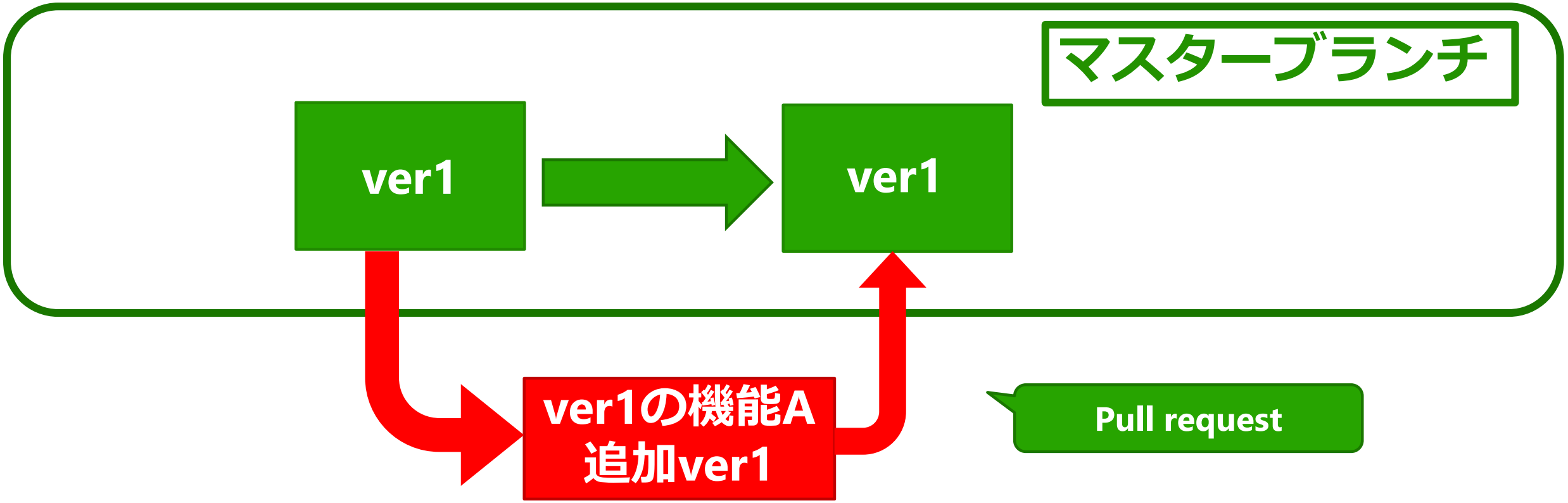
そのなかでもメインのブランチをマスターブランチという



プルリクエスト

ブランチの作業内容をマスターに反映させるにはブランチから**プルリクエスト**を行う

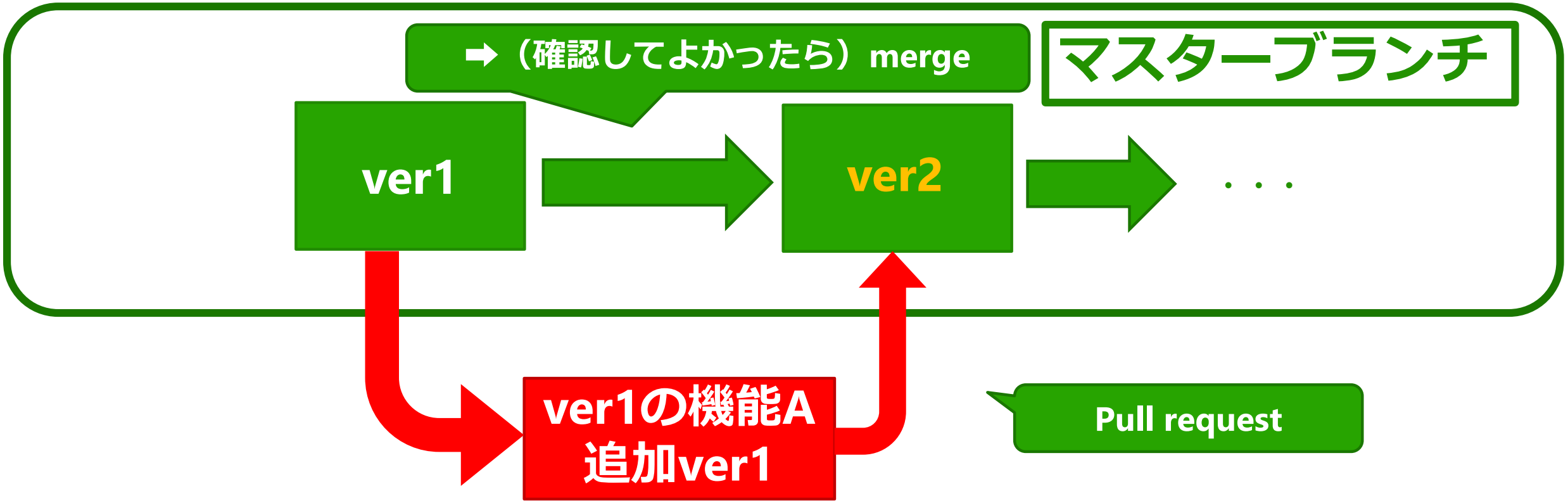
内容を確認してよかったらマージ（統合）する



プルリクエスト

ブランチの作業内容をマスターに反映させるにはブランチから**プルリクエスト**を行う

内容を確認してよかったらマージ（統合）する



ブランチのコマンド

ブランチの一覧確認

```
$ git branch
```

ブランチ作成

```
$ git branch [ブランチ名]
```

ブランチ切り替え

```
$ git checkout [ブランチ名]
```

ブランチ作成＆切り替え

```
$ git checkout –b [ブランチ名]
```

ブランチをpush

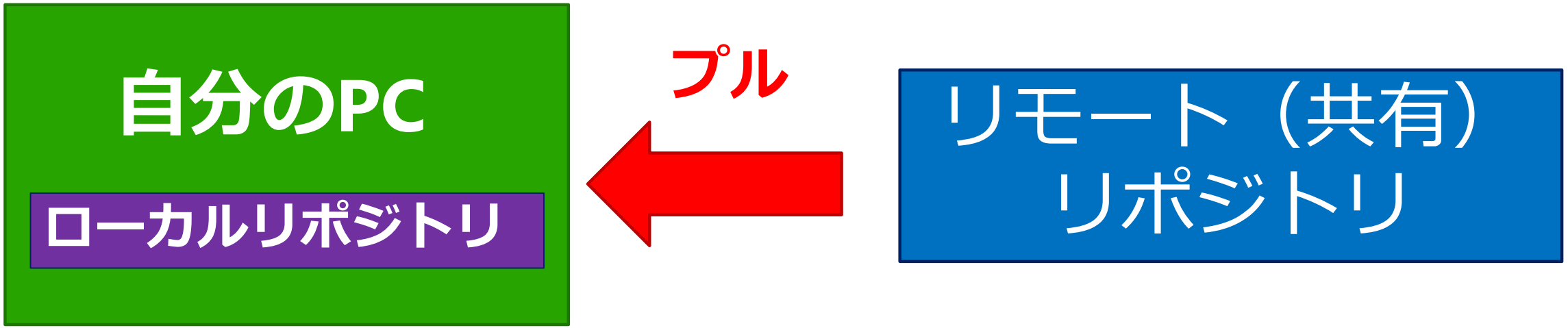
```
$ git push origin [ブランチ名]
```

プルリクエストの例

例）①リポジトリをpullして自分のローカルリポジトリを最新にする

※常にリポジトリが存在する場合

```
$ git pull origin master
```

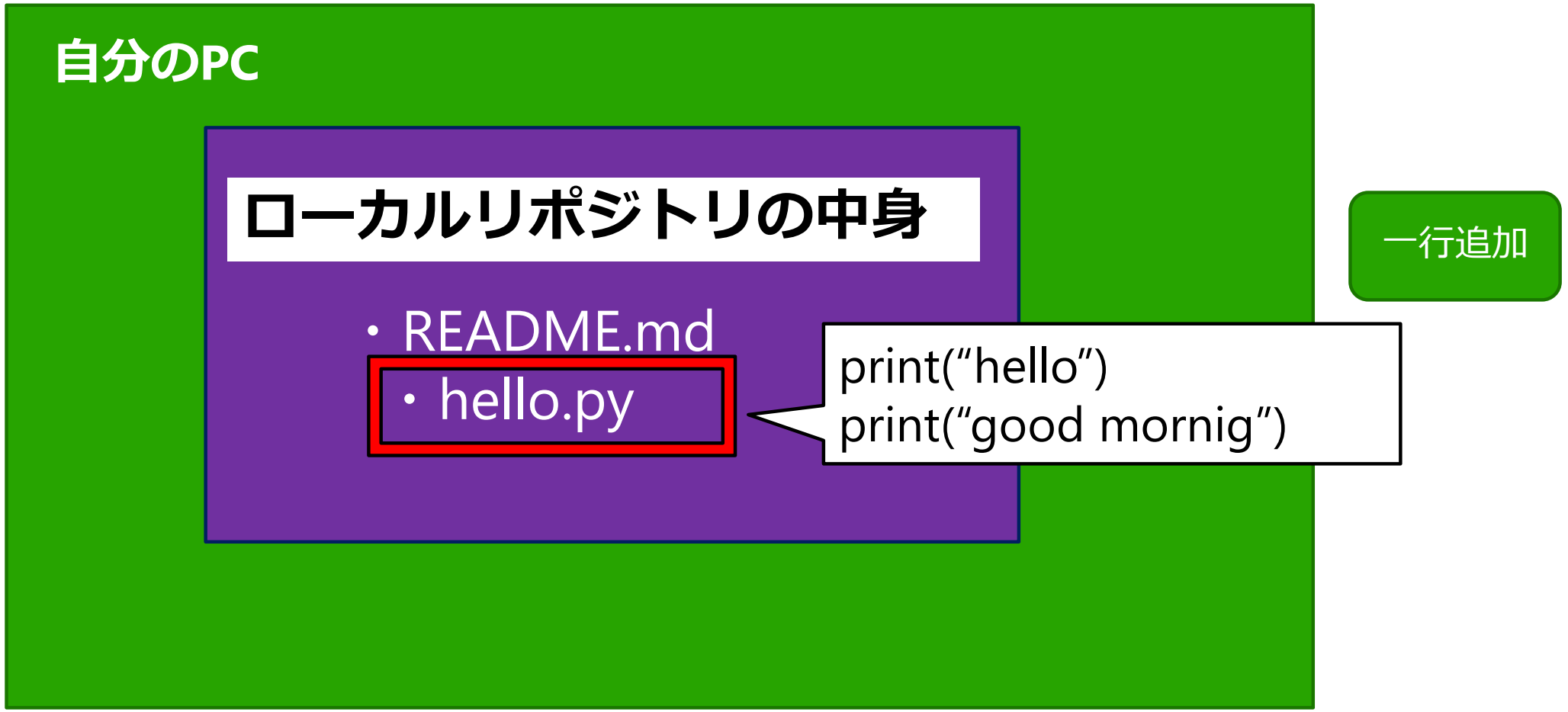


プルリクエストの例

例）②ローカルリポジトリの中身を変更（下の例ではhello.pyを変更）

プルリクエストの例

例）②ローカルリポジトリの中身を変更（下の例ではhello.pyを変更）

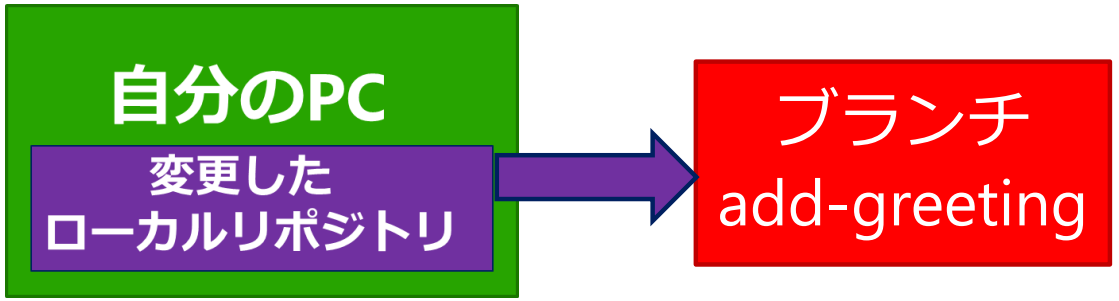


プルリクエストの例

例）③ブランチを作成してリポジトリをプッシュ

※作成するブランチには任意の名前をつけられる　下の例ではadd-greetingという名前のブランチ

$git checkout –b [ブランチ名]	**ブランチ作成＆切り替え**
$git branch	**ブランチが切り替わったか確認**
$git add .	**ファイルの登録**
$git commit –m “コメント”	**ファイルのコミット**
$git push origin [ブランチ名]	**ファイルのプッシュ**



現在のブランチは下図のように緑文字で表示されます

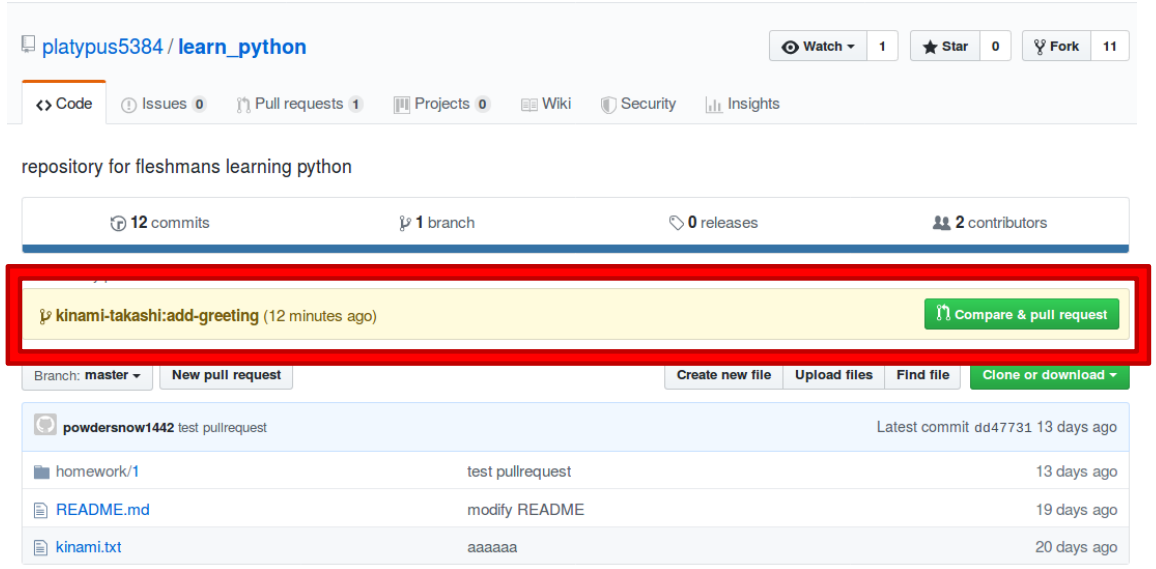
意図していないブランチを更新するといけないので切り替わっているか必ず確認しましょう

```
kinami@kinami-ZenBook-13-UX331UAL:~/learn_python$ git branch
  aaaaa
* add-greeting
  master
```

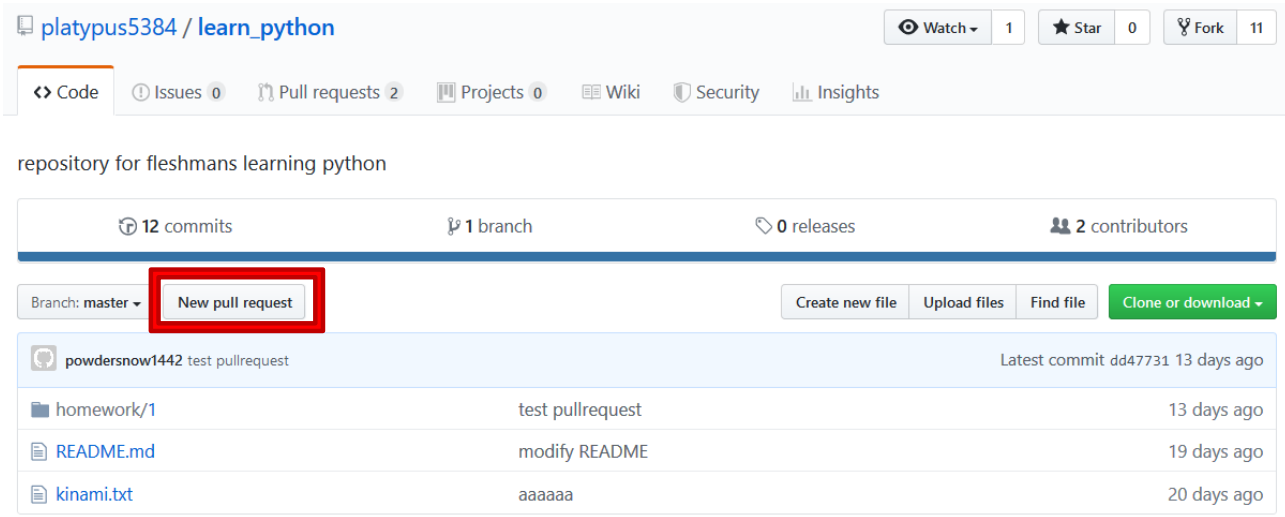
プルリクエストの例

例）④ブランチにpushしたらgithubのサイトでプルリクエストする

ブランチにプッシュすると下記のように
プルリクエストをするように表示される

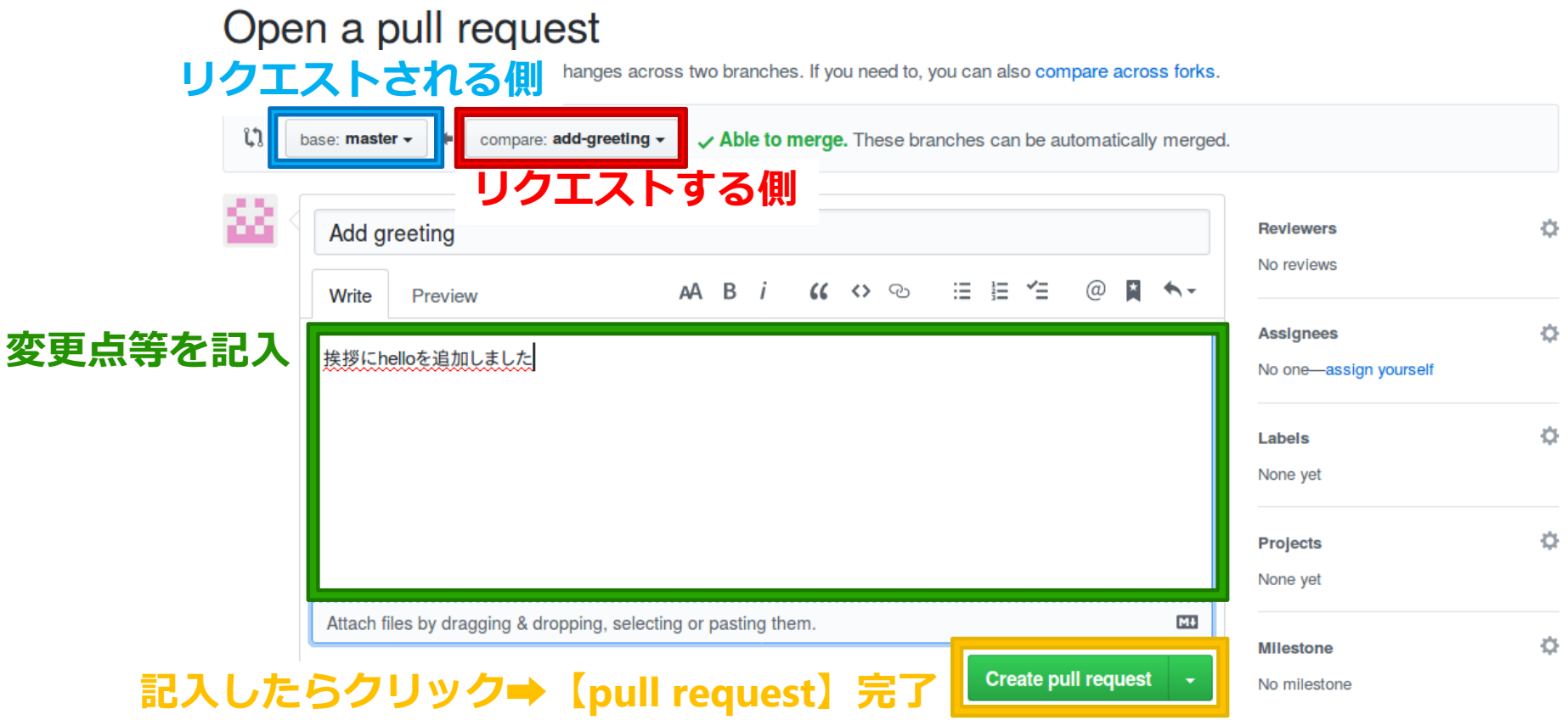


表示されなかった場合は
【New pull request】をクリックする



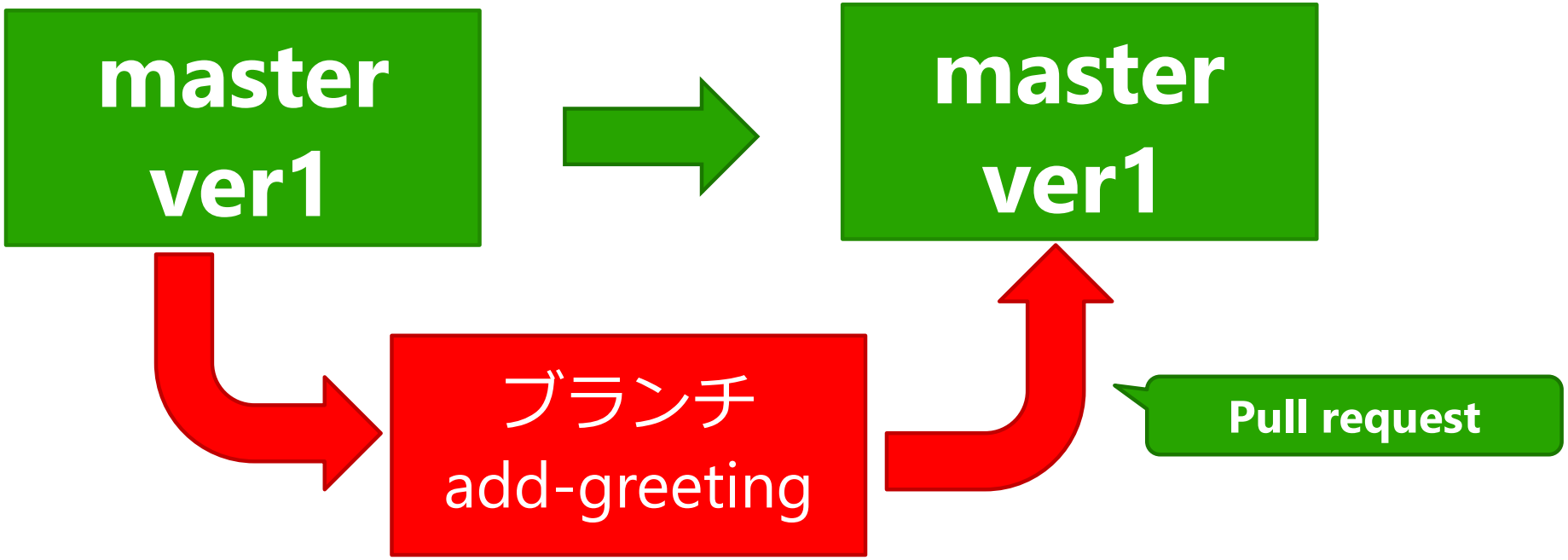
プルリクエストの例

例）④ブランチにpushしたらgithubのサイトでプルリクエストする



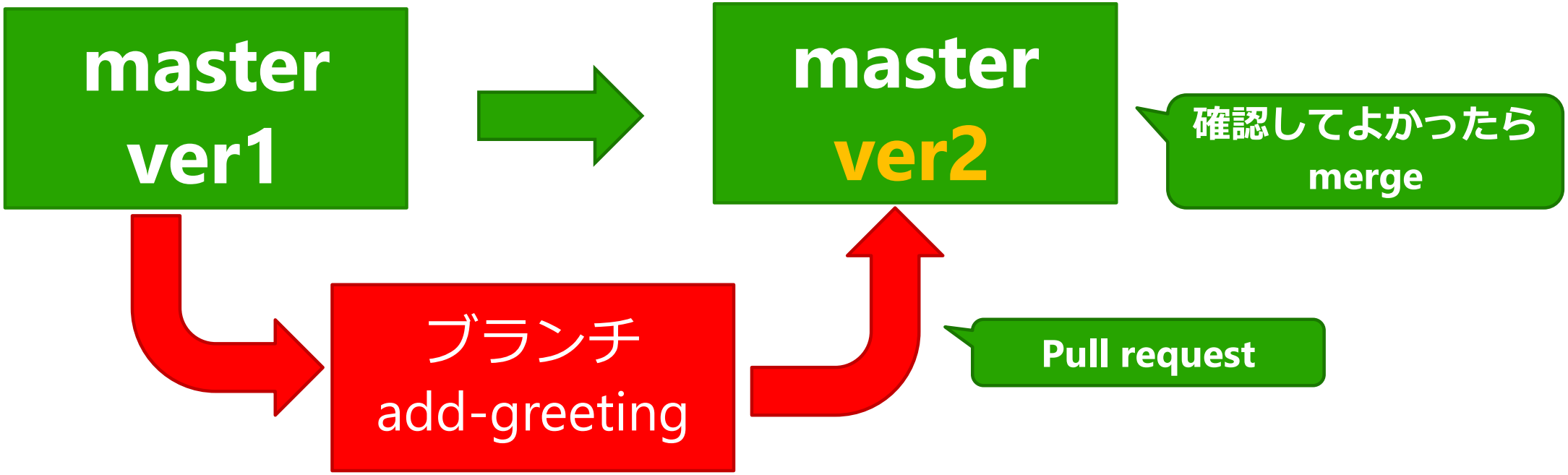
プルリクエストの例

例）⑤masterにプルリクエストが承認されたらマージ（統合）される



プルリクエストの例

例）⑤masterにプルリクエストが承認されたらマージ（統合）される



コンフリクト（競合）

同じファイルの同じ箇所を編集してプルリクエストを行うとコンフリクト（競合）が起こる

コンフリクト（競合）

コンフリクトはなるべく起こらないほうがよいので

・あらかじめ役割分担して同じ箇所を編集しないようにする

・マスターが変更されたら変更をブランチに取り込む（下記に方法を示す）

等の対策をとるとよい

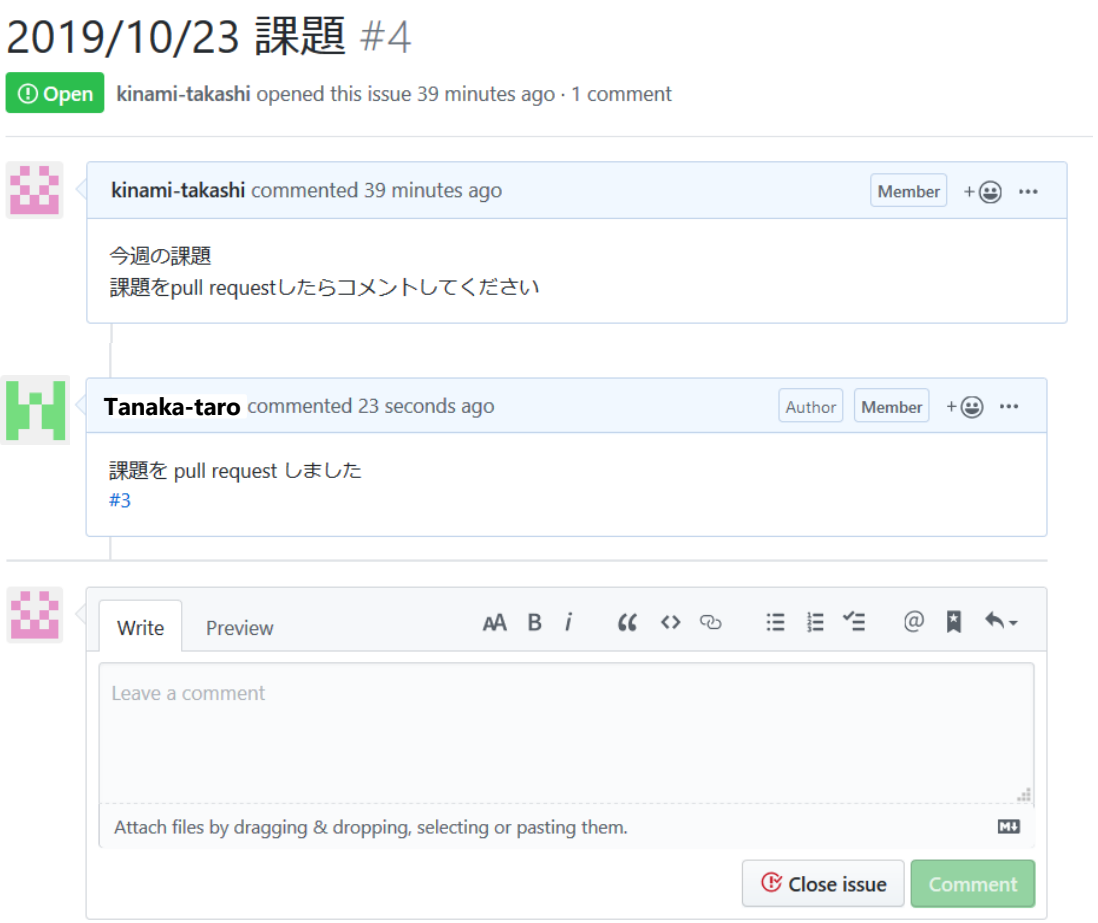
```
$git checkout master          ローカルのマスターに移動してプル
$git pull origin master
$git checkout [ブランチ名]    ブランチに移動してマージ
$git merge master
$git add .
$git commit –m “Merge”
$git push origin [ブランチ名]
```

4．イシュー

issue

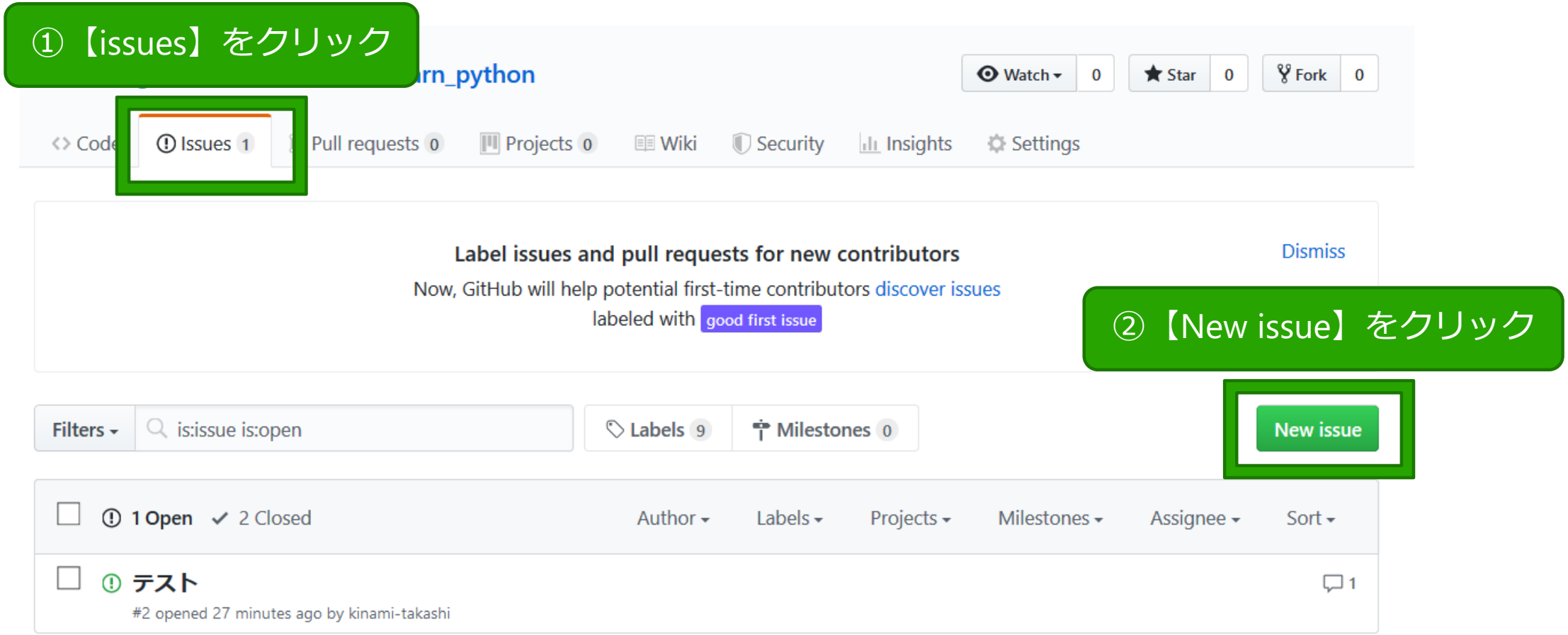
issueは課題を出題する時や、コードの内容を議論したりするときに使用

例）



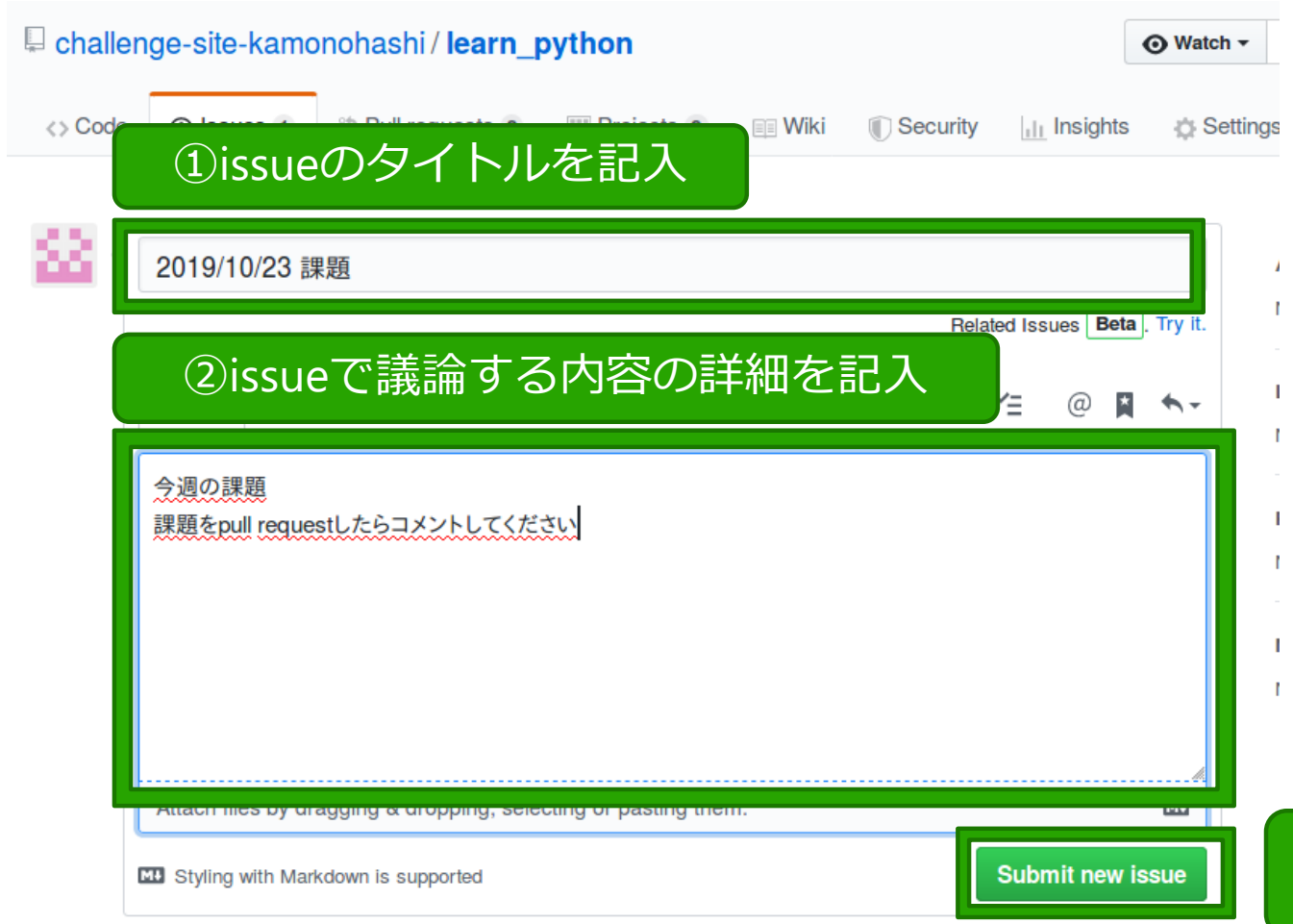
issueの作成方法

①タブ【Issues】をクリック ➡ 【New issue】をクリック



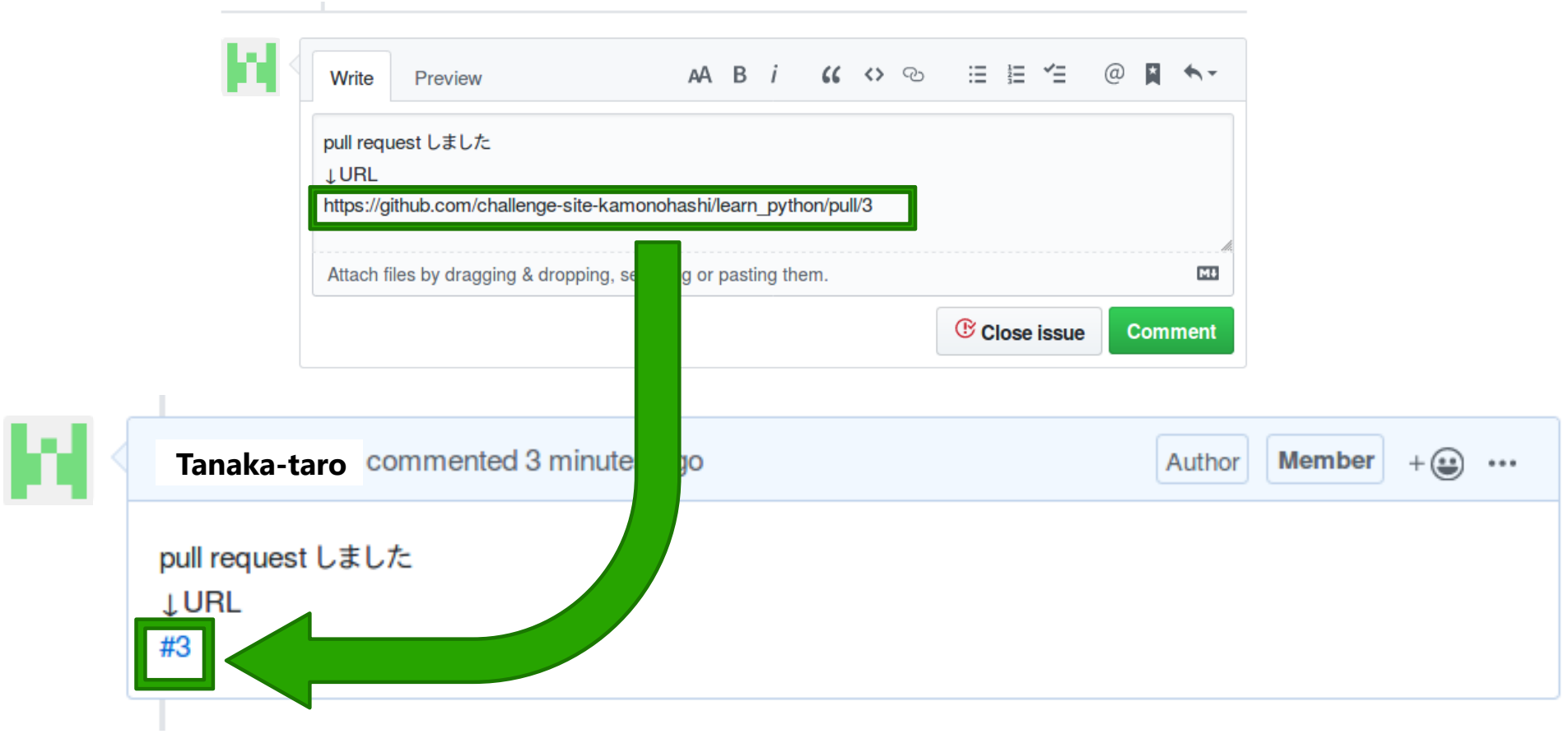
issueの作成方法

②issueのタイトルと概要を書く



Issueでのコメントの例

例えばpull requestした時のURLをコメントするといい感じに短縮されます



issueのその他の機能

issueには

担当者を明確にする「Assignee」機能

スケジュールを管理する「Milestone」機能

増えたIssueを整理する「Label」機能

など　管理するのに便利な機能が色々あります

参考サイト　https://seleck.cc/647

課題

①リポジトリ【learn_python】内の【kadai】フォルダ内の【day1】フォルダ内に【2019_10_23_（自分の名前）.py】作成する。(.pyの中身は何も記述しなくてもよい)

②　①で変更されたリポジトリをブランチにpushする

（ブランチ名は自分の名前とする）

③プルリクエストをする

④プルリクエストをしたらissue【2019/10/23課題】にてコメントする

（46ページのようにURLも記述すること）

最後に

challenge-site-kamonohashi 内では何をしてもいいので（最悪ファイル等が消えてしまっても大丈夫です）githubで他にも試してみたいことがあったら色々試してみてください