

これさえ知ってれば
チーム開発はなんとかなる
Git / GitHub講座

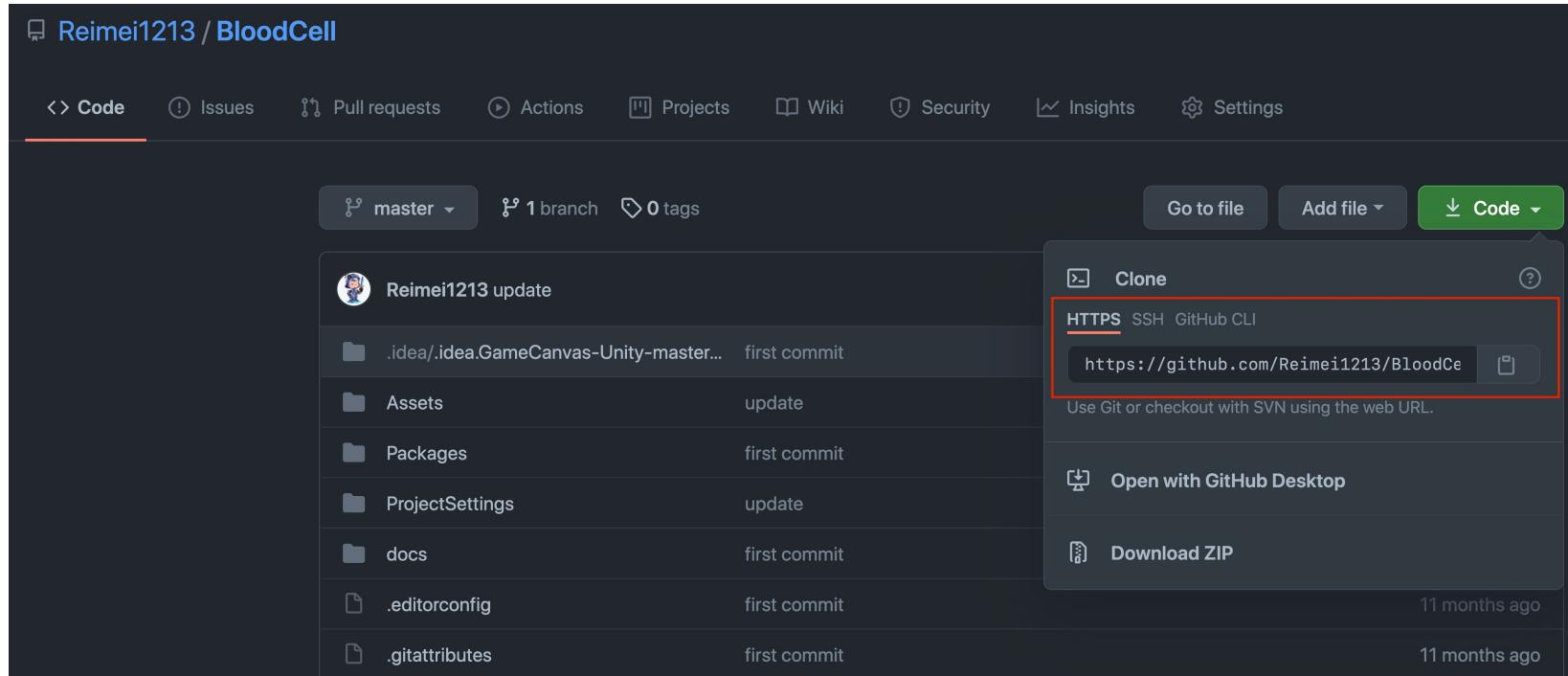
reimei

はじめに

- インターンシップや個人開発での経験をもとにした完全な偏見です
- 説明が違う部分があるかもしれません
- 環境によってはこれだけではチーム開発ができない場合があるかもしれません、僕はなんとかなっているので多分なんとかなります

```
$ git clone xxx
```

まずはリポジトリのダウンロードから！



```
$ git branch xxx
```

- master(main)ブランチはいつデプロイしても大丈夫なブランチであると認識すべき
- masterブランチを直接変更するのはまずい
- masterから別のブランチを切る必要がある
- ちなみに\$ git branchでローカルブランチ一覧表示
- タスクに番号が振られないと結びつけることができてわかりやすい

```
$ git checkout xxx
```

- 新しいブランチを生成しても移動はできていないのでこのコマンドでそのブランチに切り替え

```
$ git diff
```

- 一つ前のコミットからの変更ファイル、変更箇所を表示
- \$ git diff ブランチ名(コミットの番号)..ブランチ名(コミットの番号)
- GitHub上で比較をする時は[https://github.com/user_id/リポジトリ名/compare/ブランチ名\(コミットの番号\)..ブランチ名\(コミットの番号\)](https://github.com/user_id/リポジトリ名/compare/ブランチ名(コミットの番号)..ブランチ名(コミットの番号))

```
$ git status
```

- 一つ前のコミットから変更ファイルを一覧表示

```
$ git add .
```

- 普段は\$ git status で更新したファイルをざっとみて問題なさそうなら\$ git add . して、問題がありそうなら\$ git add xxx ファイルを一つずつaddしている

```
$ git commit -m “xxx”
```

- add, fix, removeなどを先頭につけて詳細を後半に書く
- (けど、正直めんどくさいからこんなこと書かずに全部tmpにしちゃってる)

<https://qiita.com/itosh0/items/9565c6ad2ffc24c09364>

```
$ git push origin "xxx"
```

- プッシュする前に\$ git pull origin masterを実行して
コンフリクトが起こらないか確認
- masterにプッシュしないように注意！！

プルリク

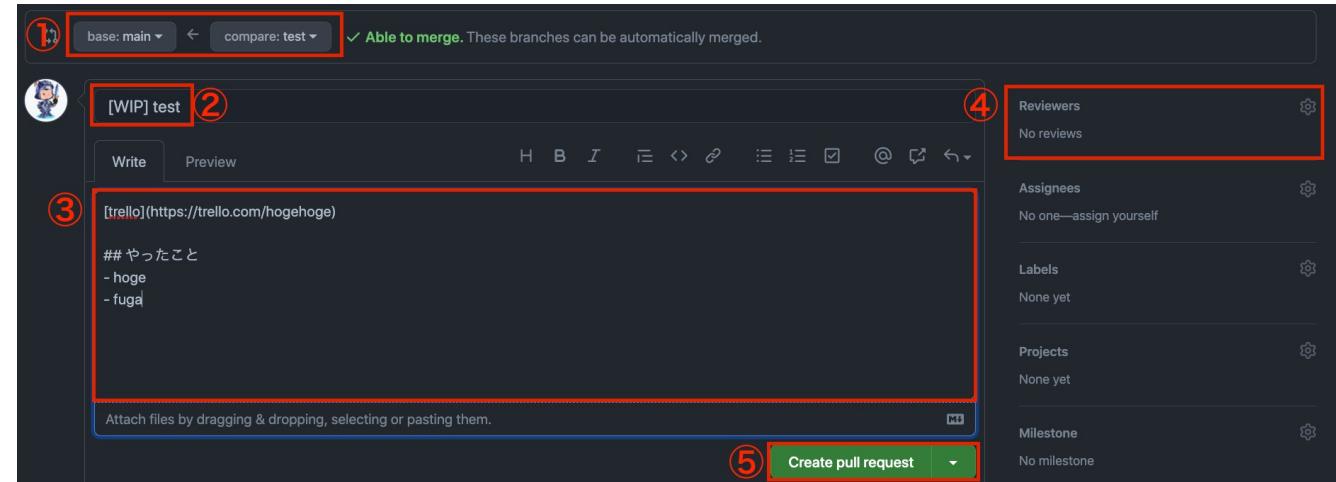
Compare & pull request をクリック

The screenshot shows a GitHub repository interface. At the top left, there's a message: "test had recent pushes less than a minute ago". On the right side, there's a green button labeled "Compare & pull request", which is highlighted with a red rectangular border. Below this, there are navigation buttons: "main", "1 branch", "0 tags", "Go to file", "Add file", and "Code". In the main content area, there's a commit history entry by user "Reimei1213" with the message "Initial commit". The commit was made at "00e1e55 2 minutes ago" and contains "1 commit". At the bottom, there's a file listing for "README.md" with the status "Initial commit" and the same timestamp.

File	Commit Message	Timestamp	Commits
README.md	Initial commit	2 minutes ago	1 commit

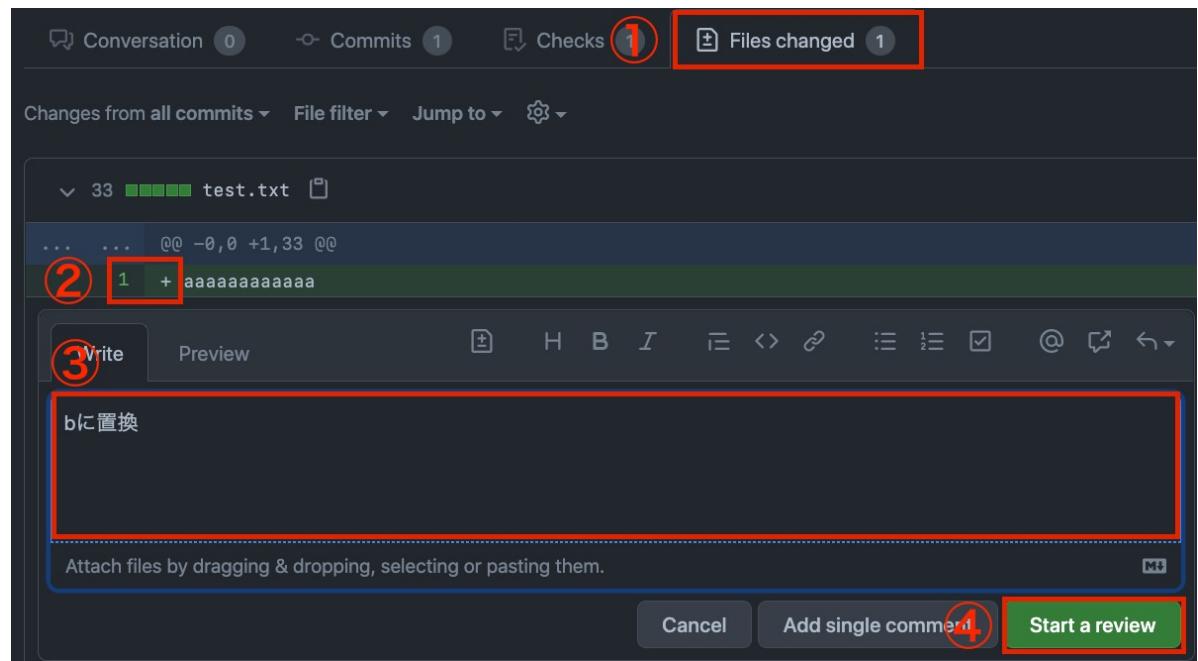
プルリク

1. マージするブランチを確認
2. タイトル確認 [WIP] xxx_title
xxxの部分に連番を入れられるとわかりやすい
3. 実装内容を記載 タスク管理アプリのリンクなども添付
4. レビューする人を指定(チームアカウントの時)
5. プルリク作成



プルリク

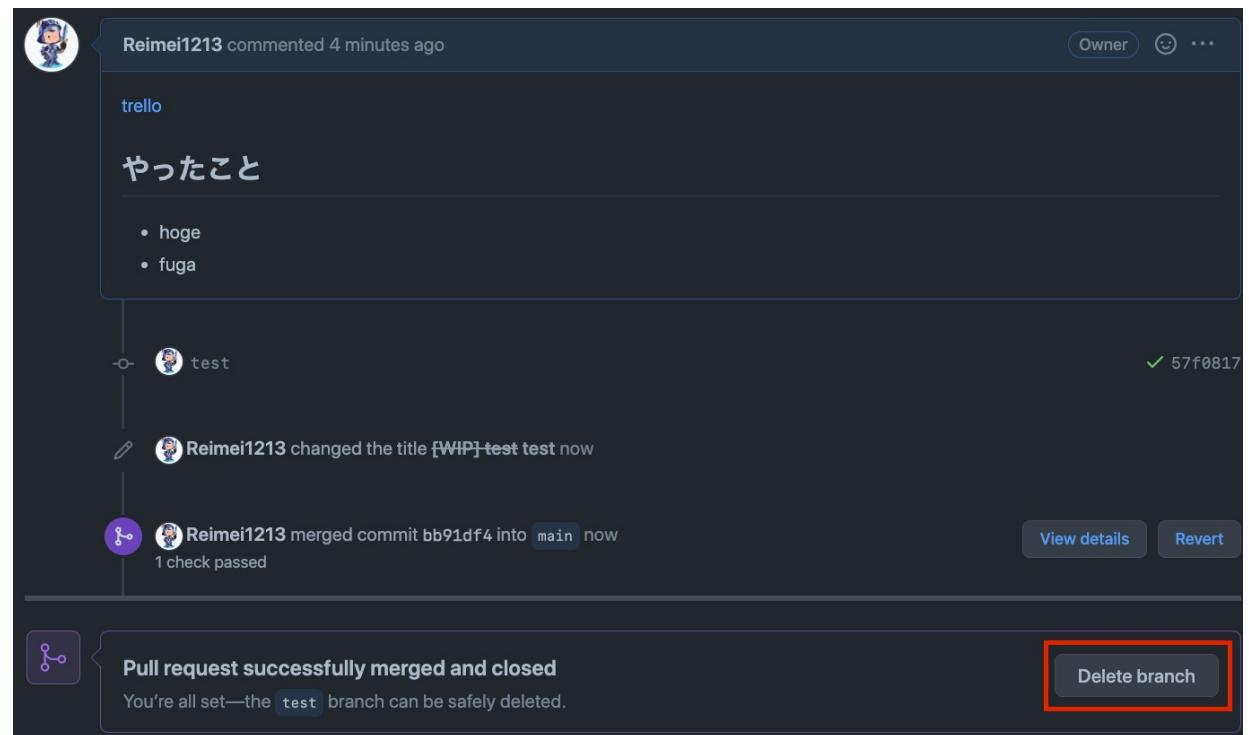
1. Files changed
2. 行番号あたりにカーソルを持っていくと+ボタンが現れるのでクリック
3. レビュー内容を記載
4. 決定



プルリク

マージ後

- Delete branchをクリックして作業を行ったリモートブランチを削除



プルリク[tips1]

- 開発中の機能が重い場合はmasterと作業ブランチの間に作業master
ブランチを入れる
- master
 - login_master
 - login_form
 - login_save
 - login_front

プルリク [tips2]

- [Do Not Merge WIP for GitHub](#)
- Chromeの拡張機能
- プルリクに「WIP」「wip」「Do Not Merge」「DNM」の文言が含まれている場合はmasterにマージできない
- 間違えてマージしてしまうトラブルが減る
- 作業中のプルリクであることを表現することができる

[引用](#)

```
$ git branch -D xxx
```

- 開発が終わったローカルブランチはもう必要ないので削除