

新卒社員研修

# Gitハンズオン研修

株式会社ブレインパッド



# 注意事項

## GitHubにマテリアルはありますが、本資料の完全再現ができないことをご了承ください

- GitHub: <https://github.com/BrainPad/GitForBeginners2020>
- 研修では、社内で利用しているBitbucket (Gitホスティングサービス)やJIRA (チケット管理ツール)を使用したため、本資料の完全再現ができないことをご了承ください

## 演習はグループワークで実施しました

- 実際に手を動かす演習では新卒5名 + TA1名のグループを作り、zoomのブレイクアウトセッションを用いた演習を行いました
- この演習は非技術職の新卒にも参加してもらいました
- Gitの役割、使い方にフォーカスしてもらうため、日本語文章を編集対象としています

## 前日に「Gitとは？」を教える講座がありました

- 本研修はday1の研修を踏まえて、演習を通しGitへの理解をさらに深めることが目的

# 目次

1. グランドルール
2. 本日の研修の課題 / 目的
3. おさらい (day1の振り返り + ファイル修正の例)
4. 課題紹介
5. チュートリアル
6. 課題1 (最低限のgitの使い方をマスター)
7. コンフリクト/ チュートリアル2
8. 課題2 (コンフリクトを理解)
9. 最後に

# 当日はブレイクアウトのチームごとにディレクトリ & チケットを用意しました

## Bitbucket

newcomers-training / 2020 / [REDACTED] / group_1 / tutorial			
名前	サイズ	最後のコミット	メッセージ
↑ ..			
📄 tutorial_1_a.txt	24 B	一昨日	tutorial dr name fixi
📄 tutorial_1_b.txt	24 B	一昨日	tutorial dr name fixi
📄 tutorial_1_c.txt	24 B	一昨日	tutorial dr name fixi
📄 tutorial_1_d.txt	24 B	一昨日	tutorial dr name fixi
📄 tutorial_1_e.txt	24 B	一昨日	tutorial dr name fixi

## JIRAのチケット

📎 ENGTRAIN-249

task\_1\_1\_a

[📎 添付](#) [📋 サブタスクを作成](#) [🔗 課題をリンク](#) ▼

説明

課題1

グループ1

メンバーa

## 1. グランドルール (5min)

---

## GitはCUI(コマンド)で実行

- 便利なGUIツール(sourcetree等)もありますが、今日はコマンドで実行してもらいます
- 付録でGUIツール例も紹介しました(本資料には未掲載)

## 分かる人は周囲の人たちを助ける

- ブレイクアウト内外で困っている人がいたら助け合いましょう
- 仕事は one team で行うもの！

## 2. 本日の研修の課題 / 目的 (5min)

---

## 課題

Gitを用いて間違いだらけの『我輩は猫である』をグループで正しく校正する

## 目的

### Gitとは何かを理解する

- ソースコードなどの変更履歴を保存するツールであることを知る
- 各種Gitコマンドを利用し、Gitの操作ができる

### 仕事でスムーズにGitを使えるようになる

- Gitの理解や反復練習を通じて実践経験を深める
- コンフリクトを理解し、自身で修正ができる



### 3. おさらい (day1の振り返り15min + ファイル修正の例15min)

---

day1の振り返り(15min)

※ 前日に「gitとは？」を教える研修がありました

## 以下のキーワードを理解できればOK

- バージョン管理
- リモート / ローカル
- ブランチ
- JIRAのチケットとの関連
- プルリクエスト
- コマンド(最低限): pull / checkout / add / commit / push

# 振り返り1

## Jiraとは

- Atlassian社が開発したチケット管理ツールのひとつ
  - Webベースで1つのタスクをチケットとして登録する
    - 登録したチケットを通して  
進行状況・やりとりを管理するもの
- 様々なシステム（Confluence・Bitbucketなど）と連携している
  - 自動でリンクされたりもする
  - ブレインパッドではAtlassian社と契約しているので  
全て使える
- カンバンやガントチャートなどの形式で  
進捗を表示することも出来る



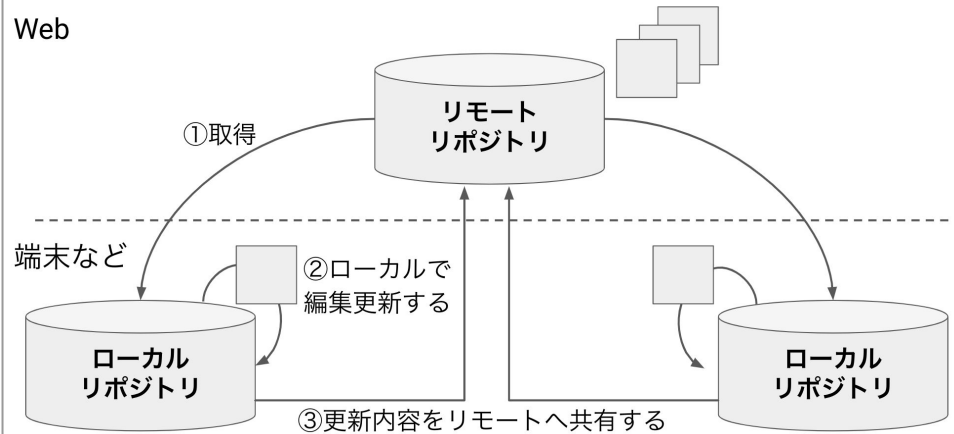
## チケット管理ツールとは

- 実施すべき作業・修正すべきバグなどの  
タスクを登録することが出来るツールのこと
- 1件のタスクにつき1件のチケットを作成する
  - チケットには以下の内容を記録する  
タスクの内容・優先度・担当者  
期日・進捗状況など
  - 記録内容はプロダクト(プロジェクト)に  
よって異なる
- 類似サービスにRedmineやTrelloという  
ツールも存在する



# 振り返り2

## Gitの概念図



## バージョン管理とは

- バージョン管理とはプログラムの修正や追加など更新する度にソースコードの状態を日時やコメントとともに記録する仕組みのこと
- バージョン管理システムによって可能になること
  - バグ発生時にひとつ前の状態へ切り戻す
  - ファイル名やフォルダ名に日時を刻むといった手作業による履歴管理からの解放

# Gitを使うとこんなことが避けられる

## 管理が難しい例

```
model_v1.py
model_v1_add_visualize_by_alice.py
model_v2_modified_by_bob.py
model_v2_modified_by_bob_add_visualize.py
model_final.py
model_final_2.py
model_submit_client.py
```

## 管理が簡単な例

```
model.py

(gitで履歴管理)
```

## ファイル修正の例(15min)

## リモートとローカルの関係 (masterブランチのみを使用する場合)

masterブランチで作業は  
本当はご法度

リモート(全員の共有場所)



リポジトリ

a.txt

Hello  
World



# リモートとローカルの関係 (masterブランチのみを使用する場合)

masterブランチで作業は  
本当はご法度

リモート(全員の共有場所)



リポジトリ

a.txt

Hello  
World

pull

pull

xさんローカル



リポジトリ

a.txt

Hello  
World

yさんローカル



リポジトリ

a.txt

Hello  
World

# リモートとローカルの関係 (masterブランチのみを使用する場合)

masterブランチで作業は  
本当はご法度

リモート(全員の共有場所)



pull

pull

xさんローカル



リポジトリ

```
a.txt
Hello
World
```

yさんローカル



リポジトリ

```
a.txt
Hello
World
```

修正

リポジトリ

```
a.txt
Good bye
World
```

# リモートとローカルの関係 (masterブランチのみを使用する場合)

masterブランチで作業は  
本当はご法度

リモート(全員の共有場所)



pull

pull

xさんローカル



リポジトリ

```
a.txt
Hello
World
```

yさんローカル



リポジトリ

```
a.txt
Hello
World
```

修正

リポジトリ

```
a.txt
Good bye
World
```

add

a.txtのHelloをGood byeに変更と  
いうことをステージング

# リモートとローカルの関係 (masterブランチのみを使用する場合)

masterブランチで作業は  
本当はご法度

リモート(全員の共有場所)



pull

pull

xさんローカル



リポジトリ

```
a.txt
Hello
World
```

yさんローカル



リポジトリ

```
a.txt
Hello
World
```

修正

リポジトリ

```
a.txt
Good bye
World
```

add

自分のmasterブランチに変更を  
適用したイメージ

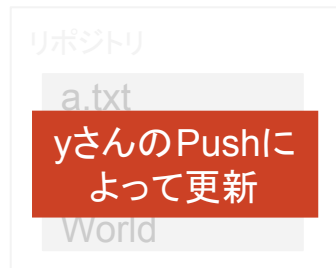
commit

a.txtのHelloをGood byeに変更と  
いうことをステージング

# リモートとローカルの関係 (masterブランチのみを使用する場合)

masterブランチで作業は  
本当はご法度

リモート(全員の共有場所)



pull

pull

xさんローカル



リポジトリ

a.txt

Hello  
World

yさんローカル



リポジトリ

a.txt

Hello  
World

修正

リポジトリ

a.txt

Good bye  
World

add

ローカルでの変更を  
リモートのmasterブランチに同期

リポジトリ

a.txt

Good bye  
World

Push

自分のmasterブランチに変更を  
適用したイメージ

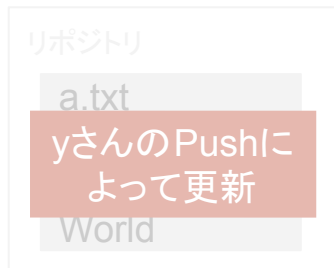
commit

a.txtのHelloをGood byeに変更と  
いうことをステージング

# リモートとローカルの関係 (masterブランチのみを使用する場合)

masterブランチで作業は  
本当はご法度

リモート(全員の共有場所)



共有先のリモートのマスターを  
気軽に変更しても問題ない？

ローカルでの変更を  
リモートのmasterブランチに同期



pull

pull

Push

xさんローカル



yさんローカル



リポジトリ

a.txt

Hello  
World

リポジトリ

a.txt

Hello  
World

リポジトリ

a.txt

Good bye  
World

修正

add

自分のmasterブランチに変更を  
適用したイメージ

commit

a.txtのHelloをGood byeに変更と  
いうことをステージング

# リモートとローカルの関係 (masterブランチのみを使用する場合)

masterブランチで作業は  
本当はご法度

リモート(全員の共有場所)

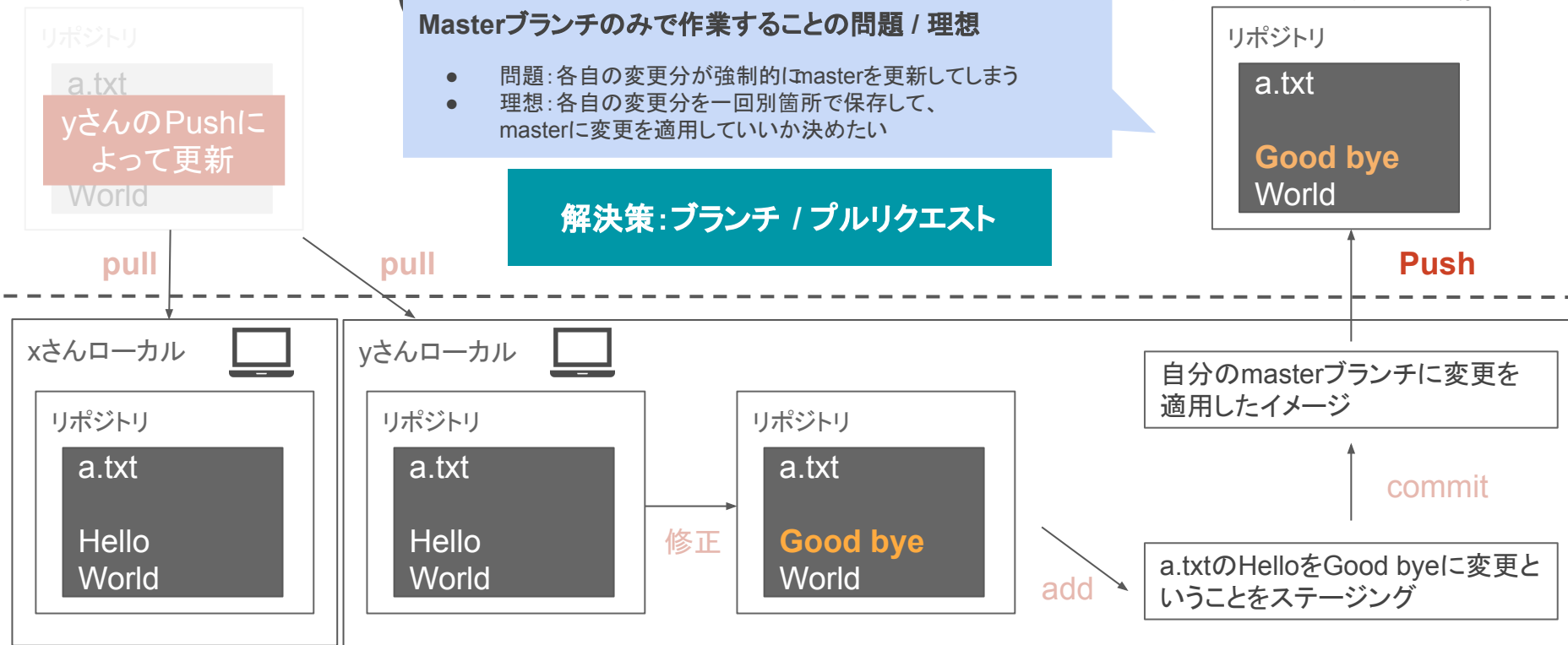


## Masterブランチのみで作業することの問題 / 理想

- 問題: 各自の変更分が強制的にmasterを更新してしまう
- 理想: 各自の変更分を一回別箇所で保存して、masterに変更を適用していいか決めたい

解決策: ブランチ / プルリクエスト

ローカルでの変更を  
リモートのmasterブランチに同期



(重要) master pushはしない！！ブランチ切って作業しましょう。

master pushはしない！！


ブランチを切って作業する！！

- リモート上の誰かの作業を上書きしてしまうため、この研修では**master pushはダメ\***
- 上書きを防ぐためにもブランチを切って作業しましょう

\*個人作業ではmaster pushをしても良いですが、チーム作業時には避けることを推奨



# (イメージ)リモートとローカルの関係 (ブランチ / プルリクエスト)

リモート(全員の共有場所) 


リポジトリ

a.txt

Hello  
World

master

# (イメージ)リモートとローカルの関係 (ブランチ / プルリクエスト)

リモート(全員の共有場所) 

master

リポジトリ

a.txt

Hello  
World

feature


リポジトリ

a.txt

Hello  
World

JIRAのチケットから  
ブランチを切る(推奨)

# (イメージ)リモートとローカルの関係 (ブランチ / プルリクエスト)

リモート(全員の共有場所) 

master

リポジトリ

a.txt

Hello  
World

feature

リポジトリ

a.txt

Hello  
World

xさんローカル



feature  
(not master)


リポジトリ

a.txt

Hello  
World

pull

# (イメージ)リモートとローカルの関係 (ブランチ / プルリクエスト)

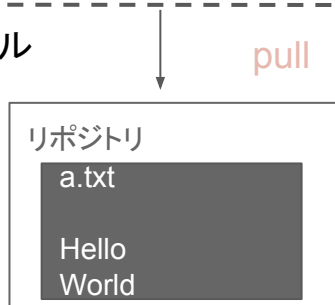
リモート(全員の共有場所) 



xさんローカル



feature  
(not master)



pull

修正




add

自分のfeatureブランチに変更分  
を適用したイメージ

commit

a.txtのHelloをGood byeに変更と  
いうことをステージング

# (イメージ)リモートとローカルの関係 (ブランチ / プルリクエスト)

リモート(全員の共有場所) 

master

リポジトリ

a.txt

Hello  
World

feature

リポジトリ

a.txt

更新

Hello  
World

ポイント

masterではなく  
featureブランチが更新

リポジトリ

a.txt

Good Bye  
World

xさんローカル



feature  
(not master)

リポジトリ

a.txt

Hello  
World

pull

リポジトリ

a.txt

Good bye  
World

修正

add


Push

自分のfeatureブランチに変更分  
を適用したイメージ

commit

a.txtのHelloをGood byeに変更と  
いうことをステージング

# (イメージ)リモートとローカルの関係 (ブランチ / プルリクエスト)

リモート(全員の共有場所) 



## プルリクエスト

- featureブランチをmasterにマージしてもいいですか？
- この変更をmasterにも適用していいですか？



xさんローカル



feature  
(not master)



pull

修正



add


自分のfeatureブランチに変更分  
を適用したイメージ

Push

commit

a.txtのHelloをGood byeに  
変更ということをステージング

# (イメージ)リモートとローカルの関係 (ブランチ / プルリクエスト)

リモート(全員の共有場所) 



プルリクエストが承認されると  
featureブランチの内容が  
masterブランチに反映される



xさんローカル



feature  
(not master)



pull

修正



add

Push

自分のfeatureブランチに変更分  
を適用したイメージ

commit

a.txtのHelloをGood byeに  
変更ということをステージング

なんとなく理解しました？

1. 作業のためのブランチを切る
2. ローカルで変更を行う
  - ファイル修正 / 新規ファイル追加 / 既存ファイル削除などを行う
  - 変更箇所を add / commit
3. ローカルからリモートブランチにpushする
4. [プルリクエスト](#)を作成し、変更箇所をmasterブランチに反映していいか検討する



# (イメージ)リモートとローカルの関係 (ブランチ / プルリク)

実際の案件では、各自が担当している作業ごとにプルリクエストが出されます。

リモート(全員の共有場所)

xさん作成プルリクエスト: a.txtの修正

master



ブランチ名

feature\_x\_...

yさん作成プルリクエスト: b.txtの追加



ブランチ名

feature\_y\_...

## Q1: ブランチはわざわざ JIRA のチケットから切らないといけないの？

そういうわけではありません。

ローカルで作成して、リモートに push すればリモートにブランチが作成されます。

しかし、どのタスクを担っているかの管理の面からもチケットから切ることを推奨します。

## Q2: ステージングって何？いきなりコミットすればいいじゃん？

add されることで、「このファイルを git の管理対象にします！」と宣言しているイメージです。

反対に git で管理が不要のファイルは add しなければ良いです。

## Q3: コミット / push はどの単位ですればいい？ 1 ファイルの 1 行でも変更したらコミット？

諸々まとめて、コミット / push しても構いません。

大規模に変更されたプルリクエストですと、レビューが大変になってしまいます。

タスクのまとまりを意識したプルリクエストを作成できるようになると良いです。

経験が必要なところでもあるので、Git を使いながら「適度なまとまり」を把握していければ OK です。

## 4. 課題紹介 5min

---

# チュートリアル2問 + 課題2問

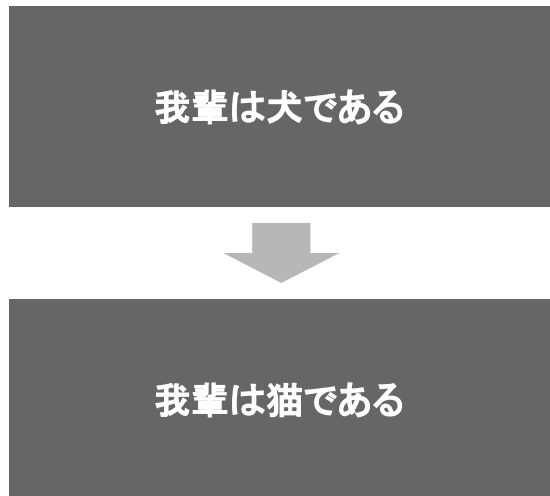
課題一覧: Tutorial 1 → 課題1 → Tutorial 2 → 課題2

各問題を解く際にはブレイクアウトセッションを活用し、チームごとに作業をして頂きます。

画面共有や雑談をしながら楽しく取り組んでください。

# 課題紹介:間違いだらけの『我輩は猫である』をグループで校正してもらいます

班ごとに間違いだらけの『我輩は猫である』を校正してもらいます。



## ディレクトリ

- グループごとに用意済
- 他グループのdirを編集しないよう注意

## JIRA

- 課題×グループ×member\_idごとに準備

当日はブレイクアウトのチームごとにディレクトリ & チケットを用意しました

The image shows two side-by-side screenshots from a presentation. The left screenshot is of a Bitbucket file browser. It shows a directory structure: 'newcomers-training / 2020 / [redacted] / group.1 / tutorial'. Below this is a table of files. The file 'tutorial\_1\_a.txt' is highlighted with a blue arrow pointing to it from the label 'member\_id(a-e)' below the table. The right screenshot is of a JIRA ticket page for 'task\_1\_1\_a'. It has two blue arrows pointing to it from the right: one from 'member\_id (a-e)' pointing to the ticket key, and another from 'グループ番号 (1-5)' pointing to the 'task\_1\_1\_a' part of the key. The ticket page also shows a 'サブタスクを作成' (Create subtask) button and a '課題をリンク' (Link issue) button.

## 5. チュートリアル

---

## Tutorial 1(30 min + 解説)

1. JIRAからブランチを切る
2. [tutorial\\_1\\_a.txt](#) をローカル(作業環境)で修正してpush する
3. Bitbucket上でプルリクエストを作成する
4. メンバーがレビューし、問題がなければマージする (マージは一斉に行います)

ファイルは、[/tutorial/tutorial\\_1\\_a.txt](#) です

キーワード: ブランチ / プルリクエスト

# Tutorial 1 : JIRAからブランチ + 修正 + プルリクエスト (20min)

JIRAに紐づいたブランチを作成し、`tutorial_1_a.txt`を修正しプルリクエストを出す。

ファイルは、[`/tutorial/tutorial\_1\_a.txt`](#) です。

`tutorial_1_<member_id>.txt`

我輩は犬である




我輩は猫である

1. JIRAのチケット(準備済)からブランチを切る
2. リモートブランチの情報を取得 (pull)
3. ローカルのブランチを切り替える (checkout)
4. `tutorial_1_a.txt`を修正 (git status / diffを利用すると便利)
5. 修正をadd / commit (分かりやすいコミットメッセージだと better)
6. リモートブランチに同期させるために push
7. Bitbucket上でプルリクエストを出す (レビューワーは班員)
8. お互いのプルリクエストを確認
9. プルリクエストのみをmasterにマージ
10. リモートのmasterが更新されていることを目視
11. リモートのmasterをローカルに同期する (fetch / pull)



# (イメージ)リモートとローカルの関係 (ブランチ / プルリクエスト)

再掲  
ここまでの作業を行う

リモート(全員の共有場所) 



## プルリクエスト

- featureブランチをmasterにマージしてもいいですか？
- この変更をmasterにも適用していいですか？

xさんローカル



feature  
(not master)



pull

修正



add

自分のfeatureブランチに変更分  
を適用したイメージ

Push

commit

a.txtのHelloをGood byeに変更と  
いうことをステージング

ひとまず、デモとしてやってみます

※ 講師が実際に作業する様子を画面で見せました

# では どうぞ！！

(研修では)チケットは作成済みです !!

tutorial\_1\_<member\_id>のチケットを使用してください

- ファイルは、[/tutorial/tutorial\\_1\\_a.txt](#) です。
- プルリクエストについて
  - レビューアーは、グループメンバー + TA で行ってください
- 後ページに解説がありますので、見ても OKです
- 分かる人は困っている人を助けましょう！

# コマンド実行例(社内で利用しているbitbucketやJIRAを使用した例)

## (1)JIRAのチケット(準備済み)からブランチを切る

tutorial\_sample

添付 サブタスクを作成 課題をリンク

説明

tutorial\_1\_a.txtの修正

アクティビティ

表示: コメント 履歴 作業ログ

コメントを追加する...

プロのヒント: M を押すとコメントできます

To Do

担当者

報告者

開発

🔗 ブランチを作成

ラベル

なし

優先度

Create branch **リポジトリの確認**

Repository

/newcomers-training

Type

Other

From branch

master

Branch name

ENGTRAIN-382-tutorial\_sample

**適切なブランチ名**  
(何をするブランチが分かるのがbetter)

🔗 master

🔗 ENGTRAIN-382-tutorial\_sample

Create Cancel

# コマンド実行例(社内で利用しているbitbucketやJIRAを使用した例)

## (2) リモートブランチの情報を取得 (pull)

\$ git pull origin <ブランチ名>

リモートブランチの情報をローカルブランチに持ってくる

```
~:newcomers-training$ git pull origin ENGTRAIN-382-tutorial_sample  
  
From ~/newcomers-training  
* branch          ENGTRAIN-382-tutorial_sample -> FETCH_HEAD
```

## (3) ローカルのブランチを切り替える (checkout)

\$ git checkout <ブランチ名>

ENGTRAIN-382-tutorial\_sample ブランチに移動

```
~:newcomers-training$ git checkout ENGTRAIN-382-tutorial_sample  
Branch 'ENGTRAIN-382-tutorial_sample' set up to track remote branch 'ENGTRAIN-382-tutorial_sample' from 'origin'.  
Switched to a new branch 'ENGTRAIN-382-tutorial_sample'  
  
~:newcomers-training$ git branch  
* ENGTRAIN-382-tutorial_sample  
master
```

git branch で今いるブランチを確認

fetchしてcheckoutでもok

\$ git fetch && git checkout ENGTRAIN-381-tutorial\_sample

**masterで作業するのは避けましょう**

- 1つの作業ブランチで、ひとまとまりのタスクとする
- プルリクエストで「masterにマージしてもいいですか?」というやり取り

# コマンド実行例(社内で利用しているbitbucketやJIRAを使用した例)

## (4)tutorial\_1\_<member\_id>.txtを修正

ファイルは、[2020/git\\_handson/day2/group\\_<group\\_id>/tutorial/](#) 以下にあります  
tutorial\_1\_<member\_id>.txt を修正

```
tutorial_1_a.txt ×
2020 > git_handson > tutorial_1_a.txt
1 我輩は犬である。
```



```
tutorial_1_a.txt ×
2020 > git_handson > tutorial_1_a.txt
1 我輩は猫である。
```

どこを変更したか(差分)を確認するコマンド `$ git diff <ファイル名>`

```
:newcomers-training $ git diff 2020/git_handson/ /tutorial_1_a.txt
diff --git a/2020/git_handson/ /tutorial_1_a.txt b/2020/ /tutorial_1_a.txt
index eb54d10..3d95e27 100644
--- a/2020/git_handson/ /tutorial_1_a.txt
+++ b/2020/git_handson/ /tutorial_1_a.txt
@@ -1,1 @@
-我輩は犬である。
\ No newline at end of file
+我輩は猫である。
\ No newline at end of file
```

git diffだけでも差分を確認できますが、  
差分のある全てのファイルが表示されます

# コマンド実行例(社内で利用しているbitbucketやJIRAを使用した例)

## (5) 修正をadd / commit (分かりやすいコミットメッセージだと better)

\$ git add <ファイル名>

tutorial.txtをadd

参考: [Gitのコミットメッセージの書き方](#)

```
newcomers-training $ git add 2020/git_handson/ /tutorial_1_a.txt
newcomers-training $ git status
On branch ENGTRAIN-382-tutorial_sample
Your branch is up to date with 'origin/ENGTRAIN-382-tutorial_sample'

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   2020/git_handson/ /tutorial_1_a.txt
```

git status で 今のgitの情報を確認

- tutorial.txtの変更がaddされたことが確認できる
- こまめに打つことを推奨

- fix : バグ修正
- hotfix : クリティカルなバグ修正
- add : 新規 (ファイル) 機能追加
- update : 機能修正 (バグではない)
- change : 仕様変更
- clean : 整理 (リファクタリング等)
- disable : 無効化 (コメントアウト等)
- remove : 削除 (ファイル)
- upgrade : バージョンアップ
- revert : 変更取り消し

\$ git commit -m "コメント"

分かり易いコメントだとbetter

```
newcomers-training $ git commit -m '[fix] tutorial_1_a 犬を猫に修正'
[ENGTRAIN-382-tutorial_sample 9922710] [fix] tutorial_1_a 犬を猫に修正
1 file changed, 1 insertion(+), 1 deletion(-)
```

# コマンド実行例(社内で利用しているbitbucketやJIRAを使用した例)

## (6) リモートブランチに同期させるために push

\$ git push origin <ブランチ名>

```

[redacted]:newcomers-training [redacted] $ git push origin ENGTRAIN-382-tutorial_sample
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 4 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 663 bytes | 6
Total 7 (delta 4), reused 0 (delta 0)
remote:
remote: Create pull request for ENGTRAIN-382-tutorial_sample:
remote:   https://[redacted]/newcomers-training/pull-requests/new?source=ENGTRAIN-382-tutorial_sample&t=1
remote:
To [redacted]/newcomers-training.git
   a37fa1c..9922710  ENGTRAIN-382-tutorial_sample -> ENGTRAIN-382-tutorial_sample

```

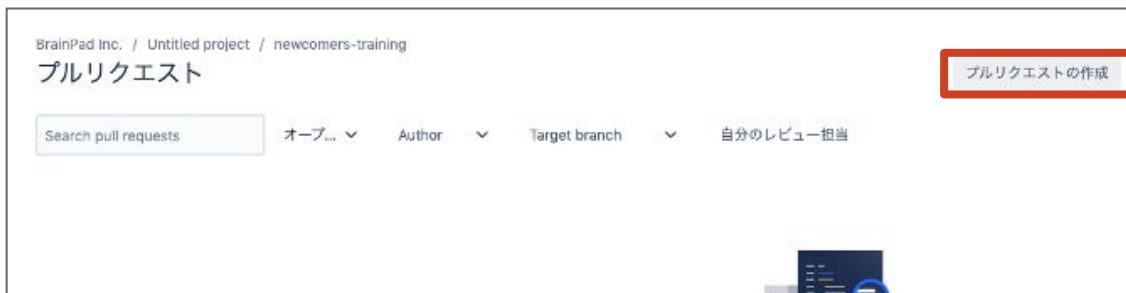
commitした内容を origin(リモート)の  
ENGTRAIN-382-tutorial\_sample ブランチに同期  
(ローカルの変更をリモートにも適応している)

これで、ローカルでの変更分をリモートに反映させたイメージ  
(まだ、masterには反映されていない。リモートの作業ブランチが反映された。)



# コマンド実行例(社内で利用しているbitbucketやJIRAを使用した例)

## (7) Bitbucket上でプルリクエストを出す (レビューアはグループメンバー)



# コマンド実行例(社内で利用しているbitbucketやJIRAを使用した例)

## (7) Bitbucket上でプルリクエストを出す (レビューアはグループメンバー)

プル リクエストを作成

 / newcomers-training  
Created 2020-03-03, updated 3 minutes ago

 / newcomers-training

ENGTRAIN-382-tutorial\_sample

master

Title

説明

**Aa** **B** **I** ...        Feedback ?

tutorial\_1\_a.txtの犬を猫に修正しました

レビュー担当者

最近使った項目:

ブランチの閉鎖 ☒ Close ENGTRAIN-382-tutorial\_sample after the pull request is merged

プル リクエストを作成

差分 コミット

作成者	コミット	メッセージ
	9922710	[fix] tutorial_1_a 犬を猫に修正

分かりやすいタイトル

レビューアに  
班員+TAを追加

分かりやすい説明  
(例えば、グラフの出力を行う場合はグラフなどを載せるとgood)

# コマンド実行例(社内で利用しているbitbucketやJIRAを使用した例)

## (8) お互いのプルリクエストを確認

[fix] tutorial\_1\_a 犬を猫に修正

#171 **オープン** ENGINRAIN-382-tutorial\_s... → master

マージ 編集 却下 承認

概要 コミット 活動履歴

作成者 [ユーザー]

レビュー担当者 [ユーザー]

説明 tutorial\_1\_a.txtの 犬を猫に修正しました

コメント (0)

コメントを入力してください。

変更されたファイル (1)

+1 -1 M 2020/git\_handson/[ユーザー]/tutorial\_1\_a.txt

2020/git\_handson/[ユーザー]/tutorial\_1\_a.txt 変更済み 横並びで比較 ファイルを表示 コメント ...

1 -我輩は犬である。  
1 +我輩は猫である。

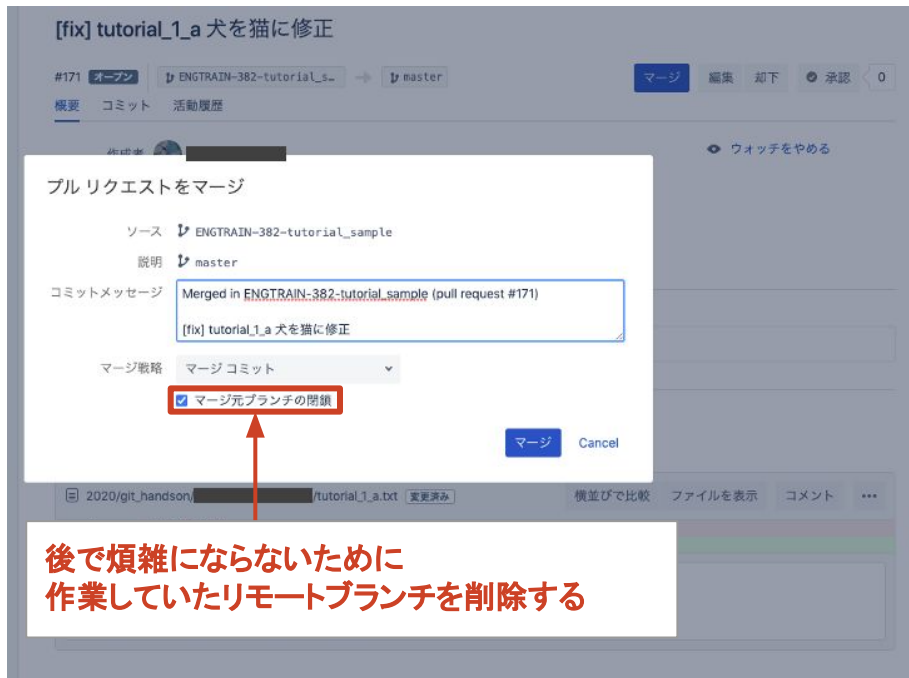
[ユーザー] 本当に猫ですか？  
返信 • 編集 • 削除 • タスクを作成 • just now

問題なければ承認を押す

気になった点があればコメント  
(試しに書いてみましょう)

# コマンド実行例(社内で利用しているbitbucketやJIRAを使用した例)

## (9) プルリクエストを master にマージ



[fix] tutorial\_1\_a 犬を猫に修正

#171 **オープン** ENGTRAIN-382-tutorial\_sample → master **マージ** 編集 却下 承認 0

概要 コミット 活動履歴

プル リクエストをマージ

ソース ENGTRAIN-382-tutorial\_sample

説明 master

コミットメッセージ Merged in ENGTRAIN-382-tutorial\_sample (pull request #171)  
[fix] tutorial\_1\_a 犬を猫に修正

マージ戦略 マージ コミット

☒ マージ元ブランチの削除

**マージ** Cancel

2020/git\_handson/tutorial\_1\_a.txt 変更済み 横並びで比較 ファイルを表示 コメント ...

後で煩雑にならないために  
作業していたリモートブランチを削除する

誰がマージを行うかはプロジェクト次第

プロジェクトごとに運用ルールを決めましょう

例

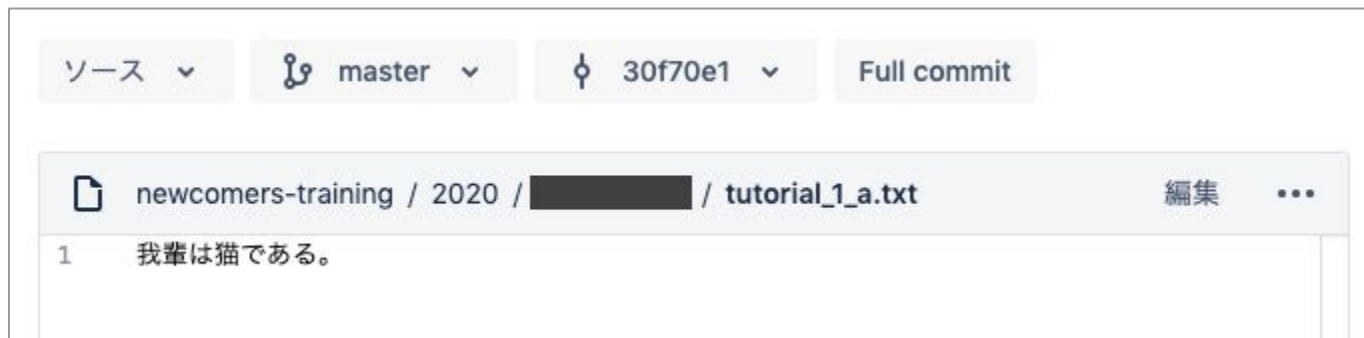
- 全員で確認してマージ
- PMがマージ

避けたいこと

- 誰からもレビューを受けずに自分でマージ

# コマンド実行例(社内で利用しているbitbucketやJIRAを使用した例)

## (10) リモートのmasterが更新されていることを確認



## コマンド実行例(社内で利用しているbitbucketやJIRAを使用した例)

### (11) ローカルのブランチを master に移動してリモートの master をローカルに同期する (fetch / pull)

```
~:newcomers-training ~$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.
```

```
~:newcomers-training ~$ git pull origin master
remote: Counting objects: 1, done.
remote: Total 1 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (1/1), done.
From ~/newcomers-training
 * branch            master      -> FETCH_HEAD
   a37fa1c..30f70e1  master      -> origin/master
Updating a37fa1c..30f70e1
Fast-forward
 2020/git_handson/~/tutorial_1_a.txt | 2 +-
1 file changed, 1 insertion(+), 1 deletion(-)
```

実行前の master ブランチの tutorial\_1\_a.txt: 我輩は **犬** である

実行後の master ブランチの tutorial\_1\_a.txt: 我輩は **猫** である

## 6. 課題1 (30min)

---

(余談)我輩は猫であるのオチご存知ですか！？

知ってる or 知らない をチャットにコメント



ビール飲みすぎて、酔っ払って壺に落ちて死にます (悲)

まるで、親指を立てながら溶鉱炉に沈んでいくシーンで有名な某映画のようですね (?)

# 課題1: (ファイルは、[/task1/task1.txt](#) です)

## task1.txt

```
> git_handson > task1.txt
1   その時苦しいながら、こう考えた。
2
3   こんな喜びに達うのはつまり壁から上へあがりたいばかりの願である。
4
5   あがりたいのは山々であるが上がれないのは知れ切っている。吾輩の足は十寸に足らぬ。
6
7   よし水の面おもてに足が浮いて、浮いた所から思う存分手をのばしたって五寸にあまる壁の縁に爪のかかり
8
9   壁のふちに爪のかかりようがなければいくらかも揺がいても、あせっても、数億年の間身を粉こにしても出ら
10
11  出れないと分り切っているものをしようとするのは無理だ。
12
13  無理を通そうとするから苦しいのだ。無理をしなければ良いとも考える。
14
15  つまらない。自みずから求めて苦しんで、自ら好んで拷問ごうもんに罹かっているのは馬鹿氣ている。
16
17  「もうよう。勝手にするがいい。」
18
19  と、体全身、前足も、後足も、頭も尾も水の方に任せて抵抗しない事にした。
20
21  次第に感覚がなくなっていく。苦しいのだから楽になったのだから見当がつかない。
22
23  水の中にいるのだから、家の中にいるのだから、判然しない。
24
25  どこにどうしていても差支さしつかえはない。ただ楽である。ただ楽である。否いな楽そのものすらも感じ
26
27  日月じつげつを切り落し、天地を粉塵して不可思議の太平に入る。この後のことを考えても何も思い浮かば
28
29  吾輩は生きる。
30
31  死んでこの平和を得る。
```

## task1.txtを分担して作業を行います

a: 1-7行を修正

b: 8-16行を修正

c: 17-25行を修正

d: 25-35行を修正

e: review.txtを作成し、我輩は猫であるの感想文を書く

(担当範囲は多少異なっても問題ありません)

(正確に直す必要もありません)

(最後の1行がないのはわざとです。dさん追記してください)

参考: [https://www.aozora.gr.jp/cards/000148/files/789\\_14547.html](https://www.aozora.gr.jp/cards/000148/files/789_14547.html)


## 7. コンフリクト

---

# コンフリクト 覚えてます？

# みんなで同じ箇所を変更したらどうなる？

猫を適当な動物に変えてください

リモート(全員の共有場所) 


リポジトリ

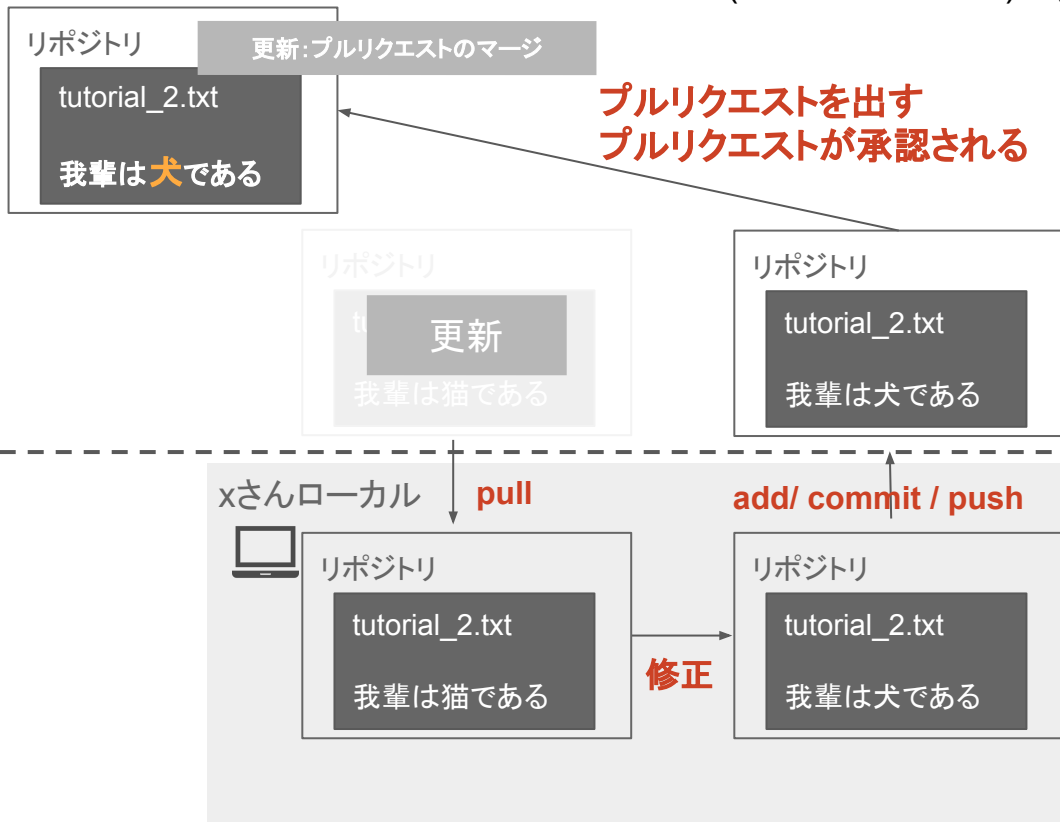
tutorial\_2.txt

我輩は猫である

# みんなで同じ箇所を変更したらどうなる？


猫を適当な動物に変えてください

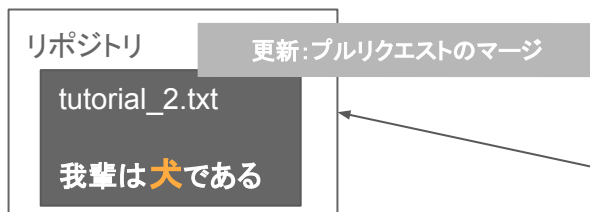
リモート(全員の共有場所) 



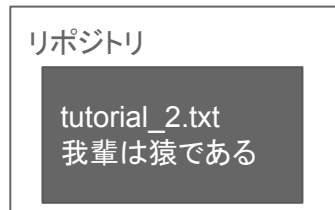
# みんなで同じ箇所を変更したらどうなる？

猫を適当な動物に変えてください

リモート(全員の共有場所) 



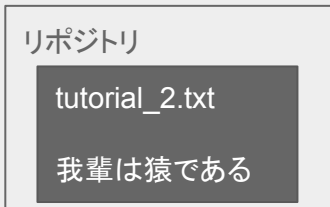
プルリクエストを出す  
プルリクエストが承認される



add/ commit / push

pull yさんローカル

修正

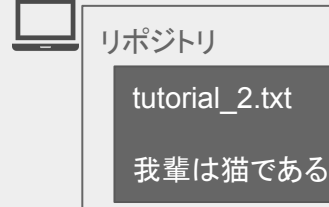


feature\_y

xさんローカル pull

add/ commit / push

修正



feature\_x

# みんなで同じ箇所を変更したらどうなる？

猫を適当な動物に変えてください

変更済み箇所と同じ箇所の変更した  
プルリクエストを出すとどうなる？

プルリクエストを出す

リモート(全員の共有場所)



リポジトリ

更新: プルリクエストのマージ

tutorial\_2.txt

我輩は犬である

プルリクエストを出す  
プルリクエストが承認される

リポジトリ

tutorial\_2.txt  
我輩は猿である

リポジトリ

tutorial\_2.txt  
我輩は猫である

リポジトリ

更新  
我輩は猫である

リポジトリ

tutorial\_2.txt  
我輩は犬である

add/ commit / push

pull yさんローカル

リポジトリ

tutorial\_2.txt  
我輩は猿である

修正

リポジトリ

tutorial\_2.txt  
我輩は猫である

feature\_y

xさんローカル pull

add/ commit / push



リポジトリ

tutorial\_2.txt  
我輩は猫である

修正

リポジトリ

tutorial\_2.txt  
我輩は犬である

feature\_x



# やってみましょう(10min)

ブレイクアウト内で2人の代表者を選出してください（他メンバーは応援）

Pullとブランチを切るタイミングは、xさん・yさんが同時に行ってください

## xさん

- 上のページのxさんの役割です
- 準備済みのチケット  
「**tutorial\_2\_<チーム番号>\_x**」からブランチを切ってください
- tutorial\_2.txtの「猫」を「犬」に書き換えてプルリクエスト作成
- メンバー全員でプルリクエストを確認し、TAがマージ

## yさん

- 上のページのyさんの役割
- 準備済みのチケット  
「**tutorial\_2\_<チーム番号>\_y**」からブランチを切ってください
- **Xさんのプルリクエストがマージされた後**、tutorial\_2.txtの「猫」を「猿」に書き換えてプルリクエスト作成

怪しいプルリクエストが出てきますよね？

cat2monkey

#21 オープン ENGTRAIN-316-tutorial\_2... → master

マージ
↓
↓
↓

概要
コミット
活動履歴

作成者

ウォッチをやめる

レビュー担当者

レビュー担当者なし

説明

cat2monkey

コメント (0)

コメントを入力してください。

変更されたファイル (1)

+5

-1

2020/git\_handson/

/tutorial\_2.txt

2020/git\_handson/

/tutorial\_2.txt

競合

横並びで比較

ファイルを表示

コメント

...

競合: File modified in both source and destination

マージするには、競合を手動で解決する必要があります。 [方法について](#)。

1

1

tutorial\_2

2

-我輩は大である。

2

+<<<<<<<< destination:4b3cba3347db9ed33a48f1d54ff3d4b779f414ca

3

+我輩は大である。

4

+=====

5

+我輩は強である。

6

+>>>>>>> source:b7686c58549b0de5af6ed6baa00b8c11810a66d7

## プルリクエストをマージ

⚠ Bitbucketは衝突のためこのリクエストを自動でマージできません。

概要タブで競合をレビューしましょう。リクエストを却下することも、以下のコマンドを用いてローカルのシステムに手動でマージすることもできます:

```
$ git checkout XXXXXXXXXX
$ git merge --no-ff -m 'Merged in featurre_y (pull request #5)' remotes/origin/featurre_y
```

マージ Cancel

猫 → 猿の変更をmasterに適用する  
プルリクエスト  
しかし、猫が既に変更されている

## コンフリクト

# コンフリクト解消方法

## STEP1:最終的にどうするかを関係者と決める

- 犬？ / 猿？
- はたまた「我輩は犬である。(改行)我輩は猿である。」？

## STEP2:実際に直す

今回は以下の方法を紹介

- ローカルで最新の master(正確にはマージ先ブランチ)をマージ
  - git checkout master して git pullで最新版の masterを取得
  - git checkout <作業ブランチ>で作業ブランチに移動
  - git merge masterでmaster「を」作業ブランチ「に」merge
- 目視で完成系に修正して commit -i or commit -m `コメント`してpush

他にもrebaseを使ったコンフリクト解消などもあります。またBitbucket上でも修正できますが、今回は行いません。

# コンフリクト解消方法

## STEP1:最終的にどうするかを関係者と決める

- 犬？ / 猿？
- はたまた「我輩は犬である。(改行)我輩は猿である。」？

チョットナニッテルカワカラナイ だと  
思うのでやってみますね

## STEP2:実際に直す

今回は以下の方法を紹介

- ローカルで最新の master(正確にはマージ先ブランチ)をマージ
  - git checkout master して git pullで最新版の masterを取得
  - git checkout <作業ブランチ>で作業ブランチに移動
  - git merge masterでmaster「を」作業ブランチ「に」merge
- 目視で完成系に修正して commit -i or commit -m `コメント`してpush



他にもrebaseを使ったコンフリクト解消などもあります。またBitbucket上でも修正できますが、今回は行いません。

# では Yさん コンフリクト解消をやってみよう

## 今回は「我輩は猿である」にする

### STEP1 : 最終的にどうするかを関係者と決める

- 犬？ / 猿？
- はたまた「我輩は犬である。(改行)我輩は猿である。」？

### STEP2 : 実際に直す

今回は以下の方法を紹介

- ローカルで最新のmaster(正確にはマージ先ブランチ)をマージ
  - git checkout master して git pullで最新版のmasterを取得
  - git checkout -b <作業ブランチ>で作業ブランチに移動
  - git merge masterでmaster「を」作業ブランチ「に」merge
- 目視で完成系に修正してcommit -i or commit -m `コメント`してpush

## 8. 課題2(コンフリクトを理解)

---

コンフリクトを理解するために全員にコンフリクトを体験していただきます！

## 課題: 改変されている「吾輩は猫である」を正しく修正する

グループごとのディレクトリにある“task2/task.md”の内容に従って進めていきます

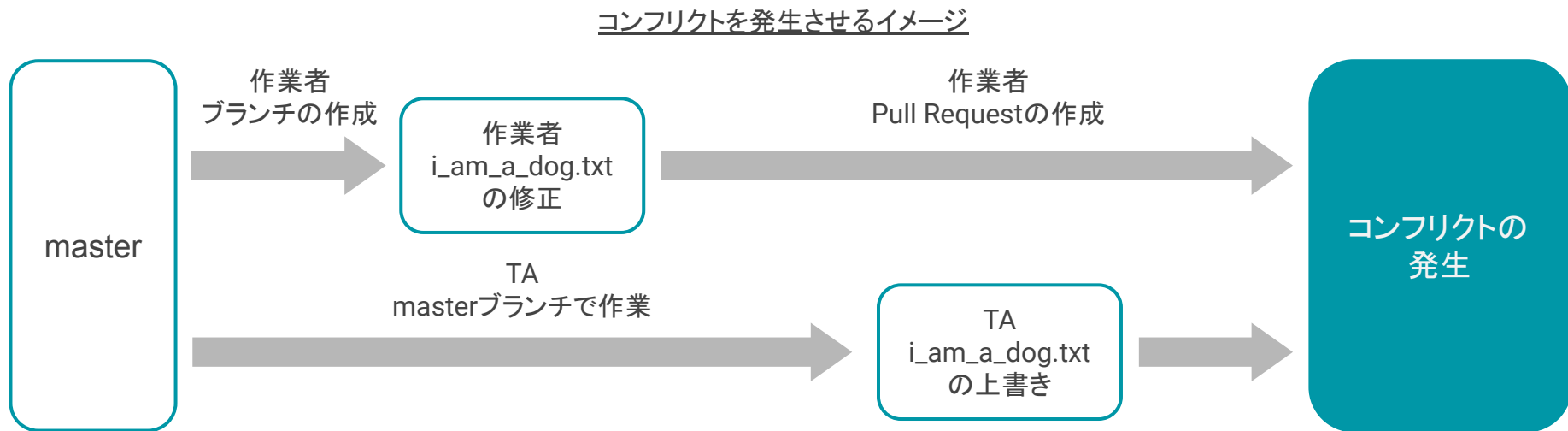
1. 担当の修正箇所を決めてください
2. “i\_am\_a\_dog.txt”を修正しましょう
3. Pull Requestを出しましょう
4. TAがある操作を行います
5. **コンフリクトを修正**しましょう
6. 再びTAがある操作を行います
7. 再び**コンフリクトを修正**しましょう
8. 各Pull Requestを**merge**しましょう

※課題を進める中で不明点がありましたら、チームメンバー・TAと相談しましょう！

## 課題2の補足

課題2ではTAが操作を行う箇所があります。

TAは作業者が作成した**Pull Request**を必ずコンフリクトさせるような作業を行います。  
詳細はレポジトリ内の“**task2/task.md**”に記載しています。





## 9. 最後に

---

### 課題

Gitを用いて間違いだらけの『我輩は猫である』をグループで正しく校正する

### 目的

#### Gitとは何かを理解する

- ソースコードなどの変更履歴を保存するツールであることを知る
- 各種Gitコマンドを利用し、Gitの操作ができる

#### 仕事でスムーズにGitを使えるようになる

- Gitの理解や反復練習を通じて実践経験を深める
- コンフリクトを理解し、自身で修正ができる



## 株式会社ブレインパッド

〒108-0071 東京都港区白金台3-2-10 白金台ビル3F

TEL:03-6721-7002 FAX:03-6721-7010

[www.brainpad.co.jp](http://www.brainpad.co.jp) [info@brainpad.co.jp](mailto:info@brainpad.co.jp)

本資料の著作権は、第三者に帰属する著作権を除き、本資料を作成した株式会社ブレインパッドに帰属します。当社の許可なく無断で、複製、改変・翻訳、販売等を行うことはできません。ただし、本資料の閲覧者は、株式会社ブレインパッドの著作物である旨を表示し、かつ、非営利目的および本資料を改変しない場合に限り、本資料をダウンロード、プリントアウト、またはコピーし、自己のために閲覧・利用することができます。なお、著作権法上認められている範囲内での引用を行うことは可能です。本資料を引用するには、以下の条件を満たす必要がありますので、ご注意ください。 1. 引用先と引用部分に主従関係があること 2. 引用部分と本文が明確に区別できること 3. 引用する必然性があり、その範囲についても必然性・合理性があること 4. 出所を明示すること 5. 部分的な改変などをせず、原文のまま引用すること