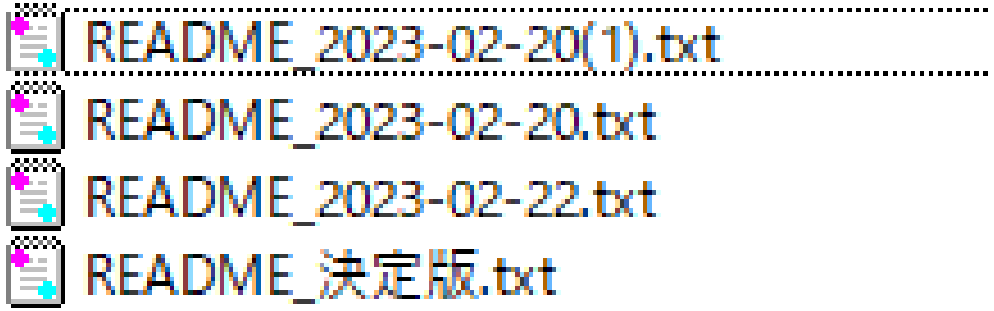
Git 入門

プログラムの履歴の記録

できること

- ファイル履歴をスマートに管理すること
  - 右のような例をなくすことができる
- 昔のバージョンに戻る
  - 「この実装だめだわ😭。戻ろ」
- スマートな共同開発
  - ファイルを送りあったりしなくてよい



Gitとは

- フィンランド出身のプログラマ
  Linus が Linux カーネル開発のために作った
  分散型ソースコード管理システム

コマンド

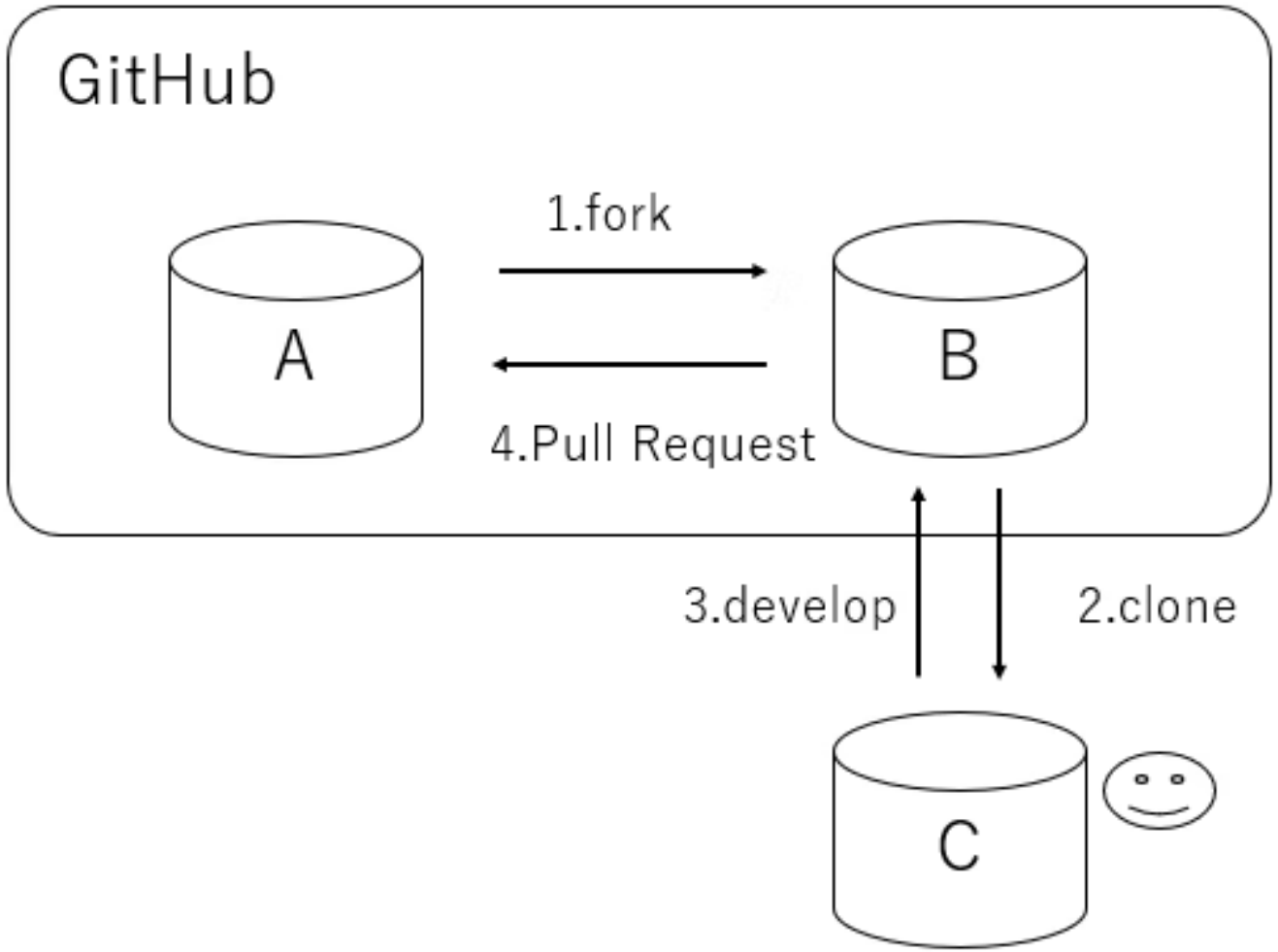
使用頻度ランキング（私見）

1. add, commit
2. status
3. push
4. init, log
5. clone
6. rm, mv
7. revert, stash, diff
8. reset
9. merge
10. branch, checkout

基本的な流れ1/2

1. git init でローカルリポジトリの初期化
2. git add で作業ファイルの変更をインデックスに追加
3. git commit でローカルリポジトリにコミット
4. git push でリモートリポジトリにプッシュ

基本的な流れ2/2



リポジトリの新規作成

- `.git` フォルダが作成される
- 注意
  - `.github/` と `.git/` は異なるもの
  - `git clone` した場合は必要ない

ファイルのステージ (stage)

- 使い方
  - git add .zshrc
  - git add *.c
  - git add images(ディレクトリ名)

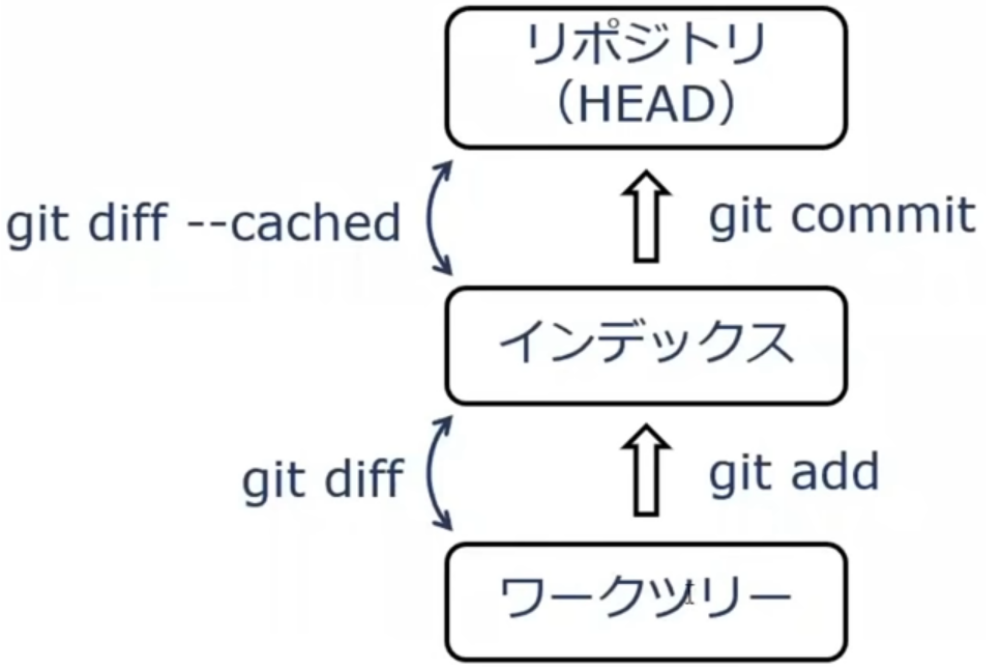
ローカルリポジトリへのファイルのコミット

- 使い方
  - git commit -m "fix: fixed typo"
  - git commit --verbose
- git commit だけだと変な画面😄に遷移

状況の表示

- 使い方
  - git status
- エイリアスを登録しておこう

git commit --verbose を使え



理由

- git commit --verbose は現在のインデックスと前回コミットの差を表示してくれるから
- 何も設定しないと Vim の画面が表示😊
- エイリアス `gc` を追加
- `git config --global commit.verbose true` でも可

未来の開発者がわかりやすいように書く

例

- docs: ○○ を追記
- fix: ○○ のバグを直した
- chore: ワークフローを書き直した
- feat: ○○ を実装

参考

ローカルリポジトリにコミット

1. 自分でリポジトリを作る(init)
2. README.md に何か書く
3. ステージングする
4. コミットする

コミット履歴の閲覧

- 使い方
  - git log
  - git log --graph
    - 履歴をグラフ化してくれる
    - 現在のブランチの履歴のみ
    - GUI ソフトが必要？git log --graph --all で充分だよなぁ！？
  - git log --graph --all
    - すべてのブランチのコミット履歴をグラフとして表示

git revert

- 正しくは「取り消したいコミットを打ち消すようなコミットを新しく生成する」
- マージコミットを取り消したいときは別
  - `git revert -m [1|2] <commit>`

git reset

- git reset --soft
  - HEAD を指定したコミットにリセット
  - インデックスとワークツリーは無変化
- git reset --mixed
  - デフォルト
  - インデックスをリセット
- git reset --hard
  - インデックスとワークツリーをリセット
  - コミットしてない変更が消える

新しくコミットしてそれを打ち消す

1. 新しく何かコミットする
2. ログで確認
3. それを打ち消す
4. ログで確認

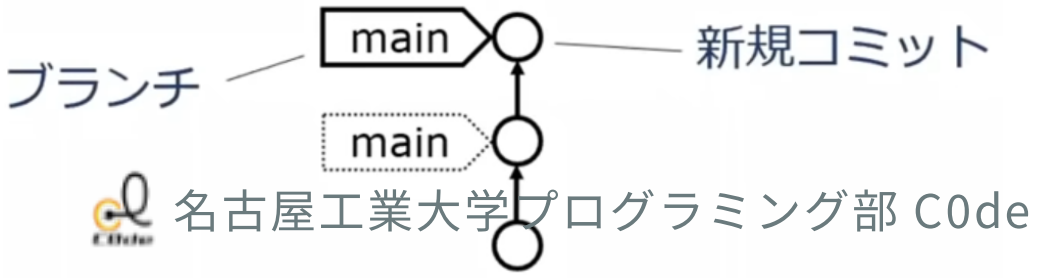
リモートリポジトリへの追加

1.Github 上でリポジトリの作成

1. git remote -v で現在のリモートリポジトリの確認
2. git remote add origin <url>
3. git remote -v でリモートリポジトリが追加されていることの確認
4. git push origin <今のブランチ名>
5. Github 上でいろいろ見てみよう
   - コミット履歴
   - コミットメッセージ

コミットに付いたラベル(ポインタ)

- ブランチの切り方
  - git switch -c hoge
  - git checkout -b
- ブランチの移動の仕方
  - git switch hoge
  - git checkout hoge



コンフリクト解消

マージしようとしたものがコンフリクトしているとこのように書き込まれる

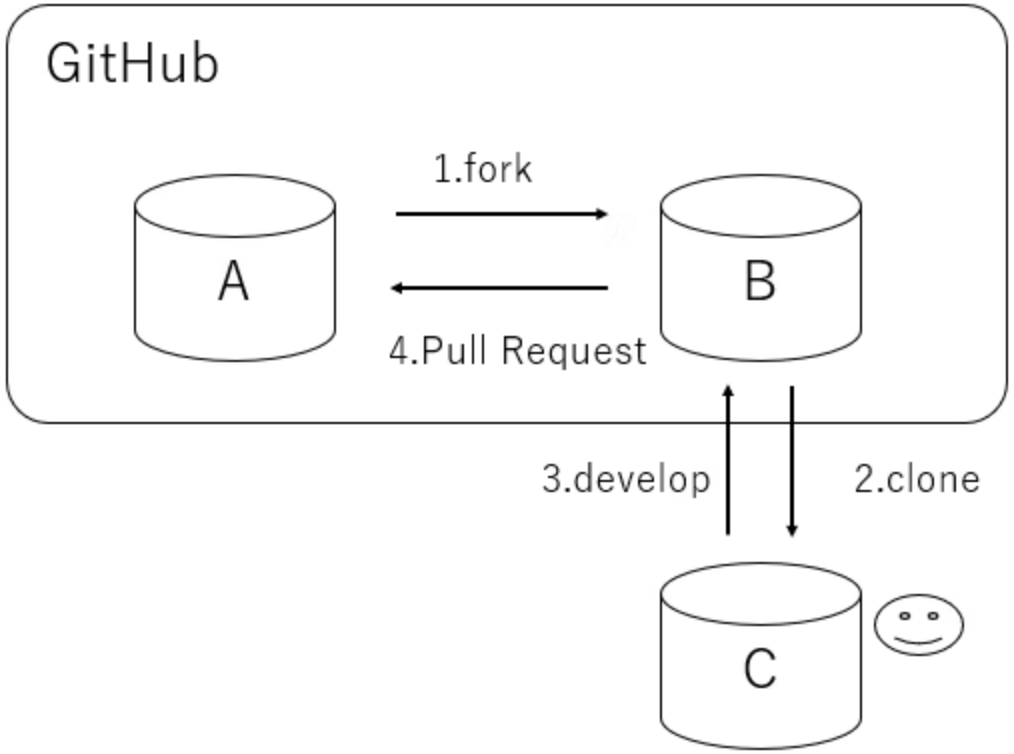
```
<<<<<<< HEAD
(                                )
=======

>>>>>>>
```

解消には

- この<や>を消しつつ内容を正しいものに修正する
- ステージングする
- コミットする

VSCode 上でコンフリクト解消可能

フォークしてクローンして PR を出そう



1. Github 上でフォークする
2. git clone <url>
3. 何かしらのブランチを切る
4. 変更を加える
5. 自分のリポジトリにプッシュする
6. PR を出す

コンフリクトしてみよう

1. ローカルリポジトリとリモートリポジトリの内容を同期
2. ローカルの内容をいじる
3. その内容でコミットする
4. git pull でリモートから引っ張ってくる
5. コンフリクトするはず

管理したくないものをリストアップ

- 例：*.local.env
  - API キーの墓場

注意

一度ステージングしてしまったものを後から `.gitignore` に追加しても意味がない

- git rm --cached hoge

git diff

- コミット前にどんな変更があるか確認したい解き

git stash

- やべ、この develop ブランチで作業するつもりが main ブランチで作業してたという時