

GitとGitHubによる プログラム管理の基礎

山ノ内 勇斗（名古屋大学, 理学研究科）

目次

1. git, GitHubとは
2. レポジトリの作成・更新
3. レポジトリを公開する

参考文献

いちばんやさしい Git&GitHubの教本



<https://amzn.to/3tvXF7E>

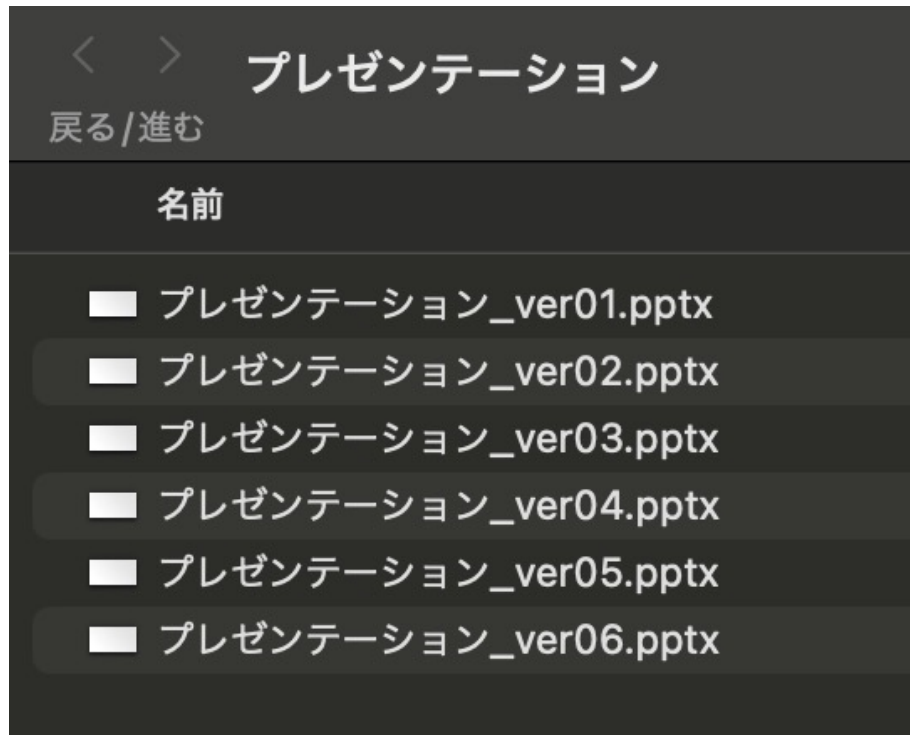
Web制作者のための GitHubの教科書



<https://amzn.to/4ajaiUt>

GitとGitHubについて

- ・ ・ ・ プログラムのバージョン管理を簡単にするツール



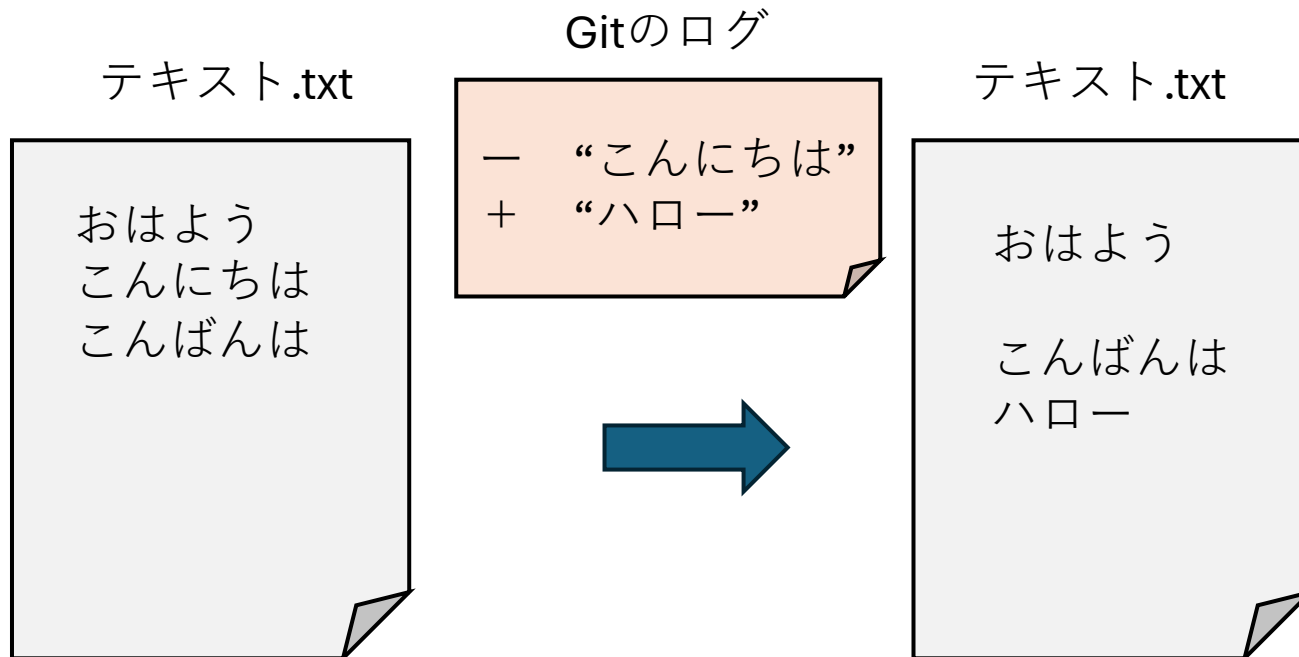
- ・ ファイルがバージョンの分だけ増える
- ・ 変更履歴がわかりにくい

コラム：OneDriveでは変更履歴を保存しなくても、自動保存によって自動的にバージョン管理がなされている。

Gitのバージョン管理システムの概念

“各ファイルの差分のみを履歴に保存する”

前に戻りたい時には
追加した部分を消去すればいい！



GitとGitHubについて

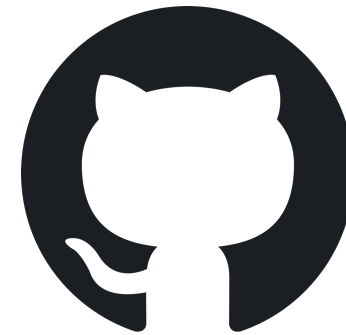
git



<https://git-scm.com>

バージョン管理システム
ローカルでファイルの
バージョン管理を行う
(ローカルレポジトリ)

GitHub



<https://github.com>

オンラインでファイルの
バージョン管理を行う
(リモートレポジトリ)

GitHub Desktopについて

・・・本来CUIベース(ターミナルとか)で行うものを簡単に行えるようにしたツール！！

GitHub Desktop



<https://desktop.github.com>

GitやGitHubの操作をGUI上で行うことができる

GitHub のアカウントの種類

- **個人用アカウント**

- ・・・通常これ！！ひとりごとにひとつのアカウント。

- **Organization アカウント**

- ・・・無制限の数の人が多くのプロジェクトで同時に
コラボレーションできる共有アカウント。
組織ごとにひとつのアカウント。

- **Enterprise アカウント**

- ・・・企業用アカウント。
よくわからないが、基本使わない。

アカウントの所属関係について

Organization アカウント

Aさん

Bさん

Organizationに所属して
いる人たちで共有可能

共同制作について

1. レポジトリのcollaboratorに追加する

- ・ Privateで公開しているレポジトリに、アクセス権限を割り振ることができる
- ・ アクセス権限には、観覧のみ、編集可能、管理者など種類がある

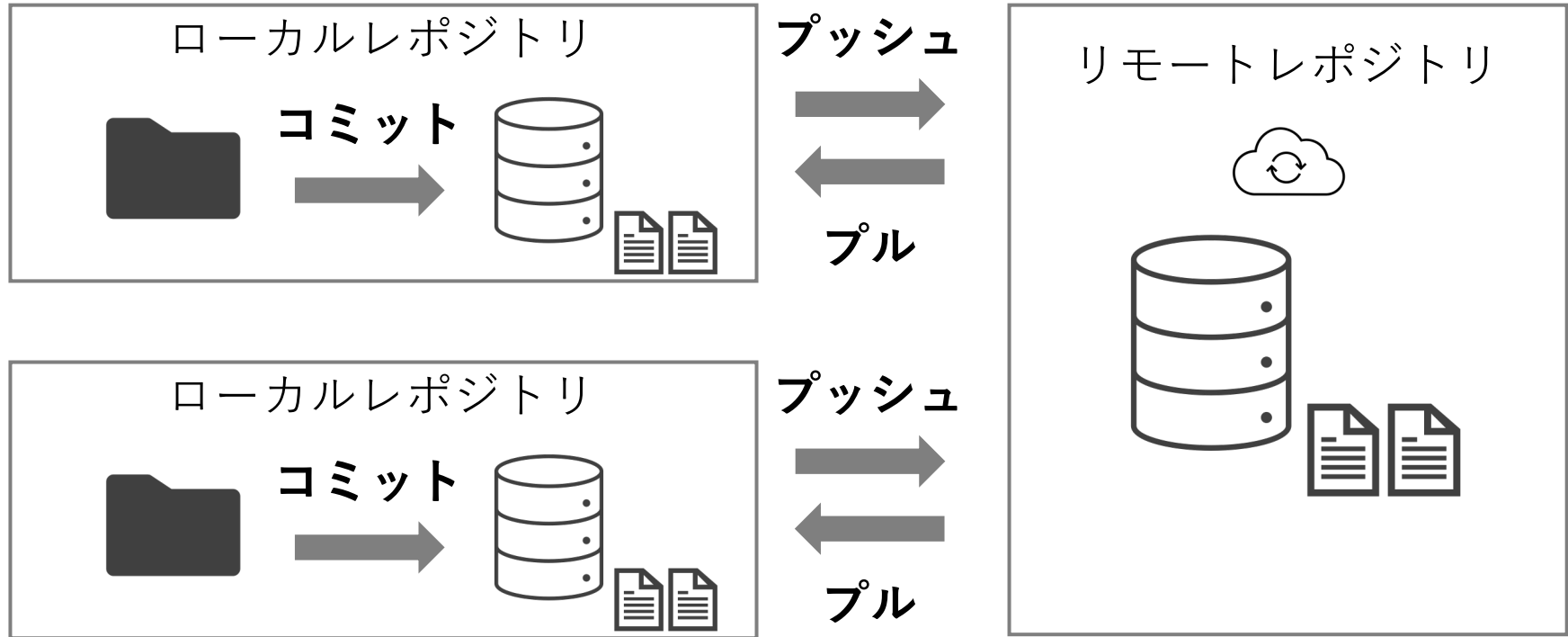
2. Organizationに入っている人たちで共有する

- ・ Organizationに所属している人たちにレポジトリの管理者権限がある
- ・ 特に特別な操作は必要ない

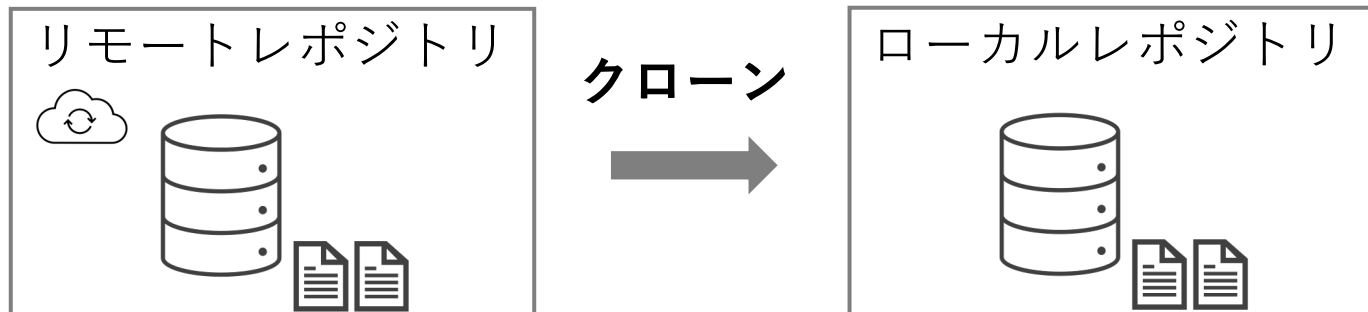
用語説明

- ・ **レポジトリ** . . . プログラム等の”保管場所”
- ・ **ローカルレポジトリ** . . . 自分のパソコンにおけるレポジトリ
- ・ **リモートレポジトリ** . . . ネット上にあるレポジトリ
- ・ **ディレクトリ** . . . フォルダのこと
狭義には、自身が作業しているフォルダ
- ・ **コミット** . . . ディレクトリ⇒ローカルレポジトリ
- ・ **プッシュ** . . . ローカルレポジトリ⇒リモートレポジトリ
- ・ **プル** . . . リモートレポジトリ⇒ローカルレポジトリ

概要図



*初回のみ



実際の作業手順 - 新しいレポジトリを作る

12

@GitHub

- ・ ・ ・ GitHubでCreate a new repositoryを押す

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

Repository name *

 HMYamano

 /

Great repository names are short and memorable. Need inspiration? How about [musical-goggles](#) ?

Description (optional)

☒  **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☐ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

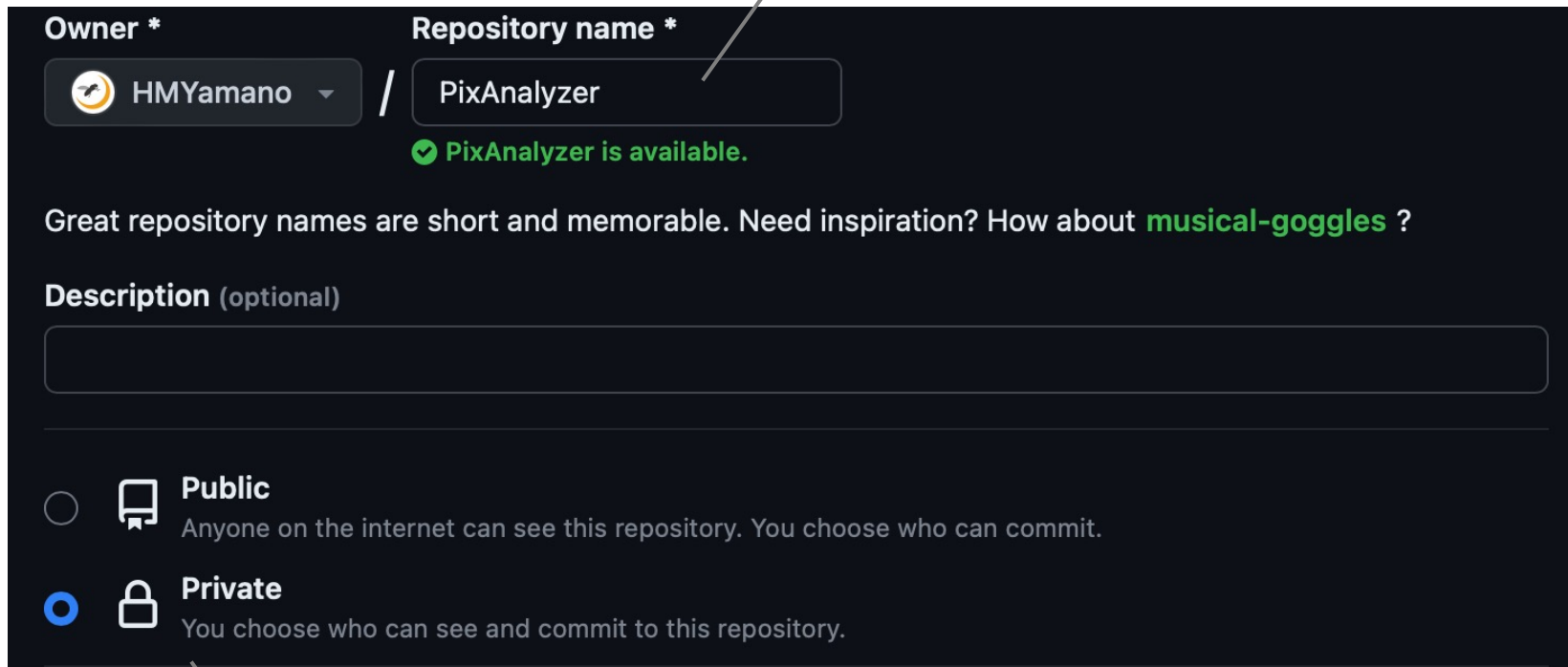
Create repository

実際の作業手順 - 新しいレポジトリを作る

13

@GitHub

レポジトリの名前 = プロジェクトの名前



Owner * / Repository name *

HMYamano / PixAnalyzer

✓ PixAnalyzer is available.

Great repository names are short and memorable. Need inspiration? How about **musical-goggles** ?

Description (optional)

☐ Public
Anyone on the internet can see this repository. You choose who can commit.

☒ Private
You choose who can see and commit to this repository.

レポジトリの公開範囲

- Public : 世の中の人全員
- Private : 自分と共有を許可した人たちのみ

実際の作業手順 - 新しいレポジトリを作る

14

@GitHub

README ファイルを追加するか否か

Initialize this repository with:

☒ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▼

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License ▼

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set `main` as the default branch. Change the default name in your [settings](#).

 You are creating a private repository in your personal account.

Create repository

ライセンスについて

.gitignore ファイルについて

実際の作業手順 - 新しいレポジトリを作る

15

@GitHub

The screenshot shows the GitHub interface for a newly created repository named 'PixAnalyzer'. The repository is private and has no stars, forks, or watchers. The main branch is 'main'. The repository contains an initial commit by HMYamano, which includes files for .gitignore, LICENSE, and README.md. The README.md file is displayed, showing the repository name 'PixAnalyzer' with a link icon. The right sidebar contains sections for 'About', 'Releases', and 'Packages', all indicating no content has been provided yet.

HM Yamano / PixAnalyzer

Code Issues Pull requests Actions Projects Security Insights Settings

PixAnalyzer Private

Unwatch 1 Fork 0 Star 0

main 1 branch 0 tags

Go to file Add file Code

HM Yamano Initial commit 8d914ff now 1 commit

.gitignore	Initial commit	now
LICENSE	Initial commit	now
README.md	Initial commit	now

README.md

PixAnalyzer

About

No description, website, or topics provided.

- Readme
- Activity
- 0 stars
- 0 watching
- 0 forks

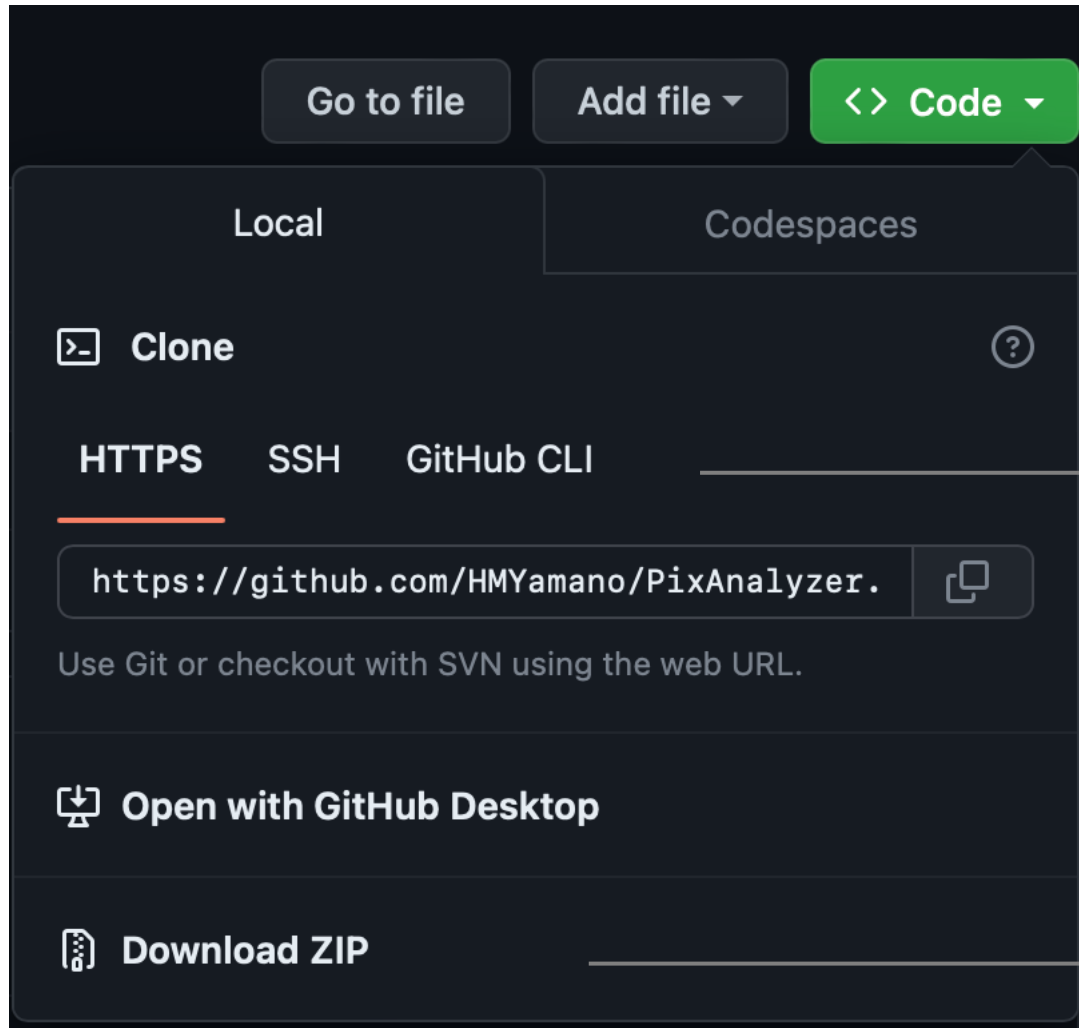
Releases

No releases published
[Create a new release](#)

Packages

No packages published
[Publish your first package](#)

実際の作業手順 – クローンする



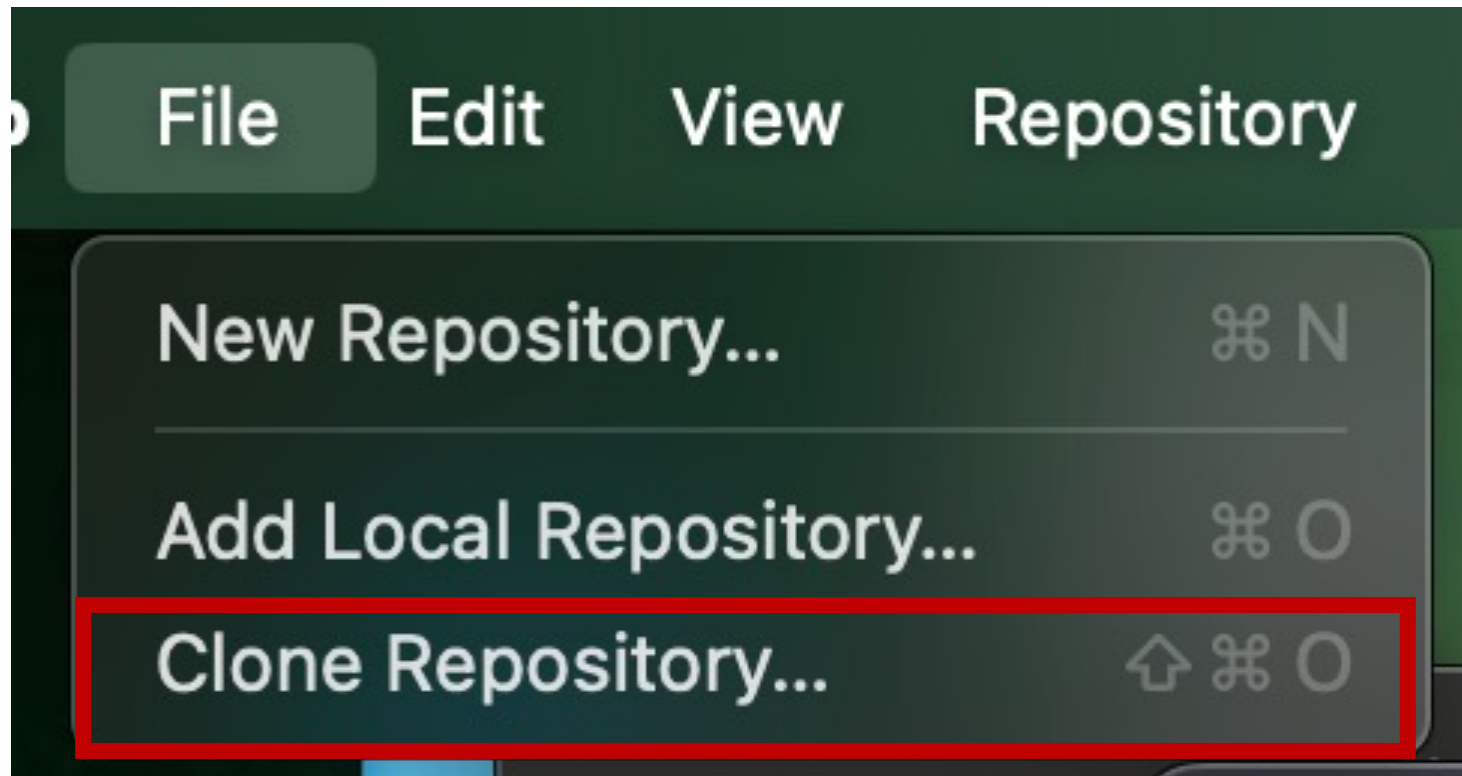
ターミナルでクローン
するときのコード

ZIPファイルで
ダウンロード

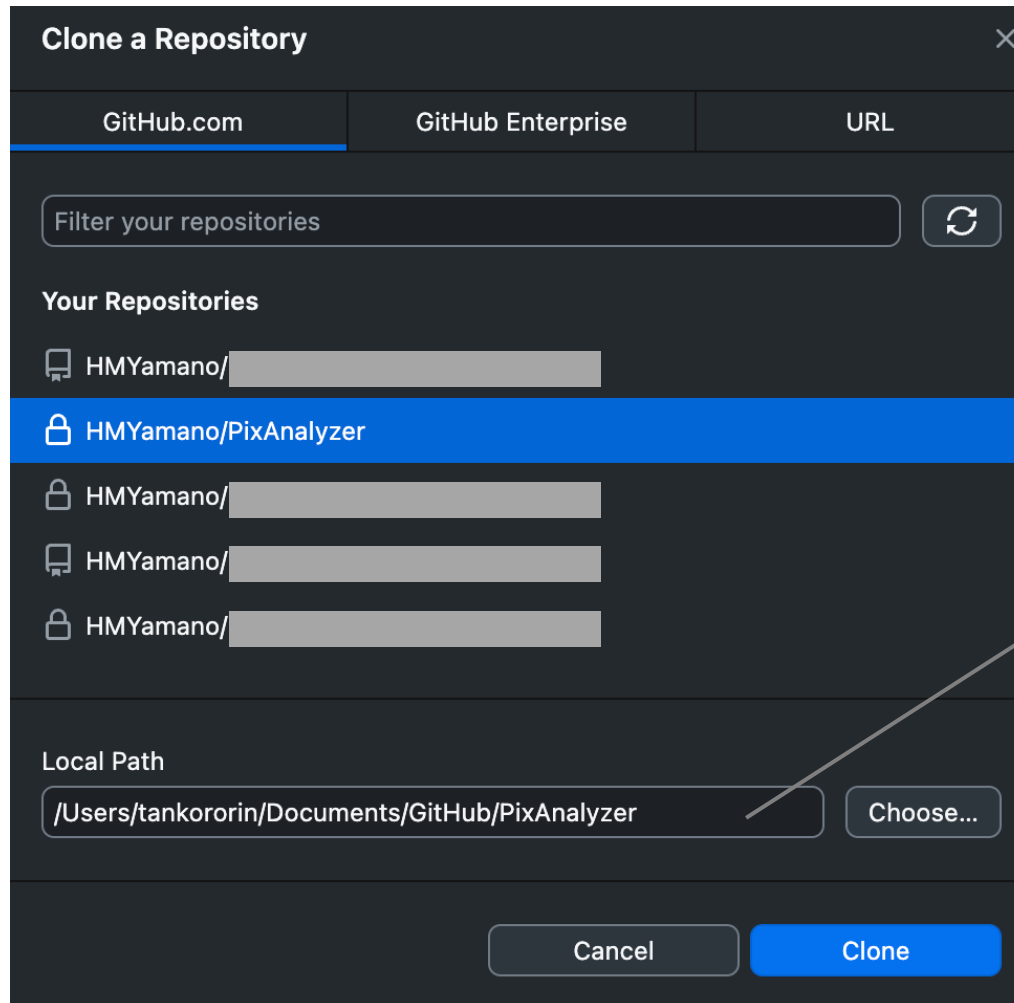
実際の作業手順 – クローンする

@GitHub Desktop

- ・ ・ ・ GitHub Desktopを使ってクローンもできる！



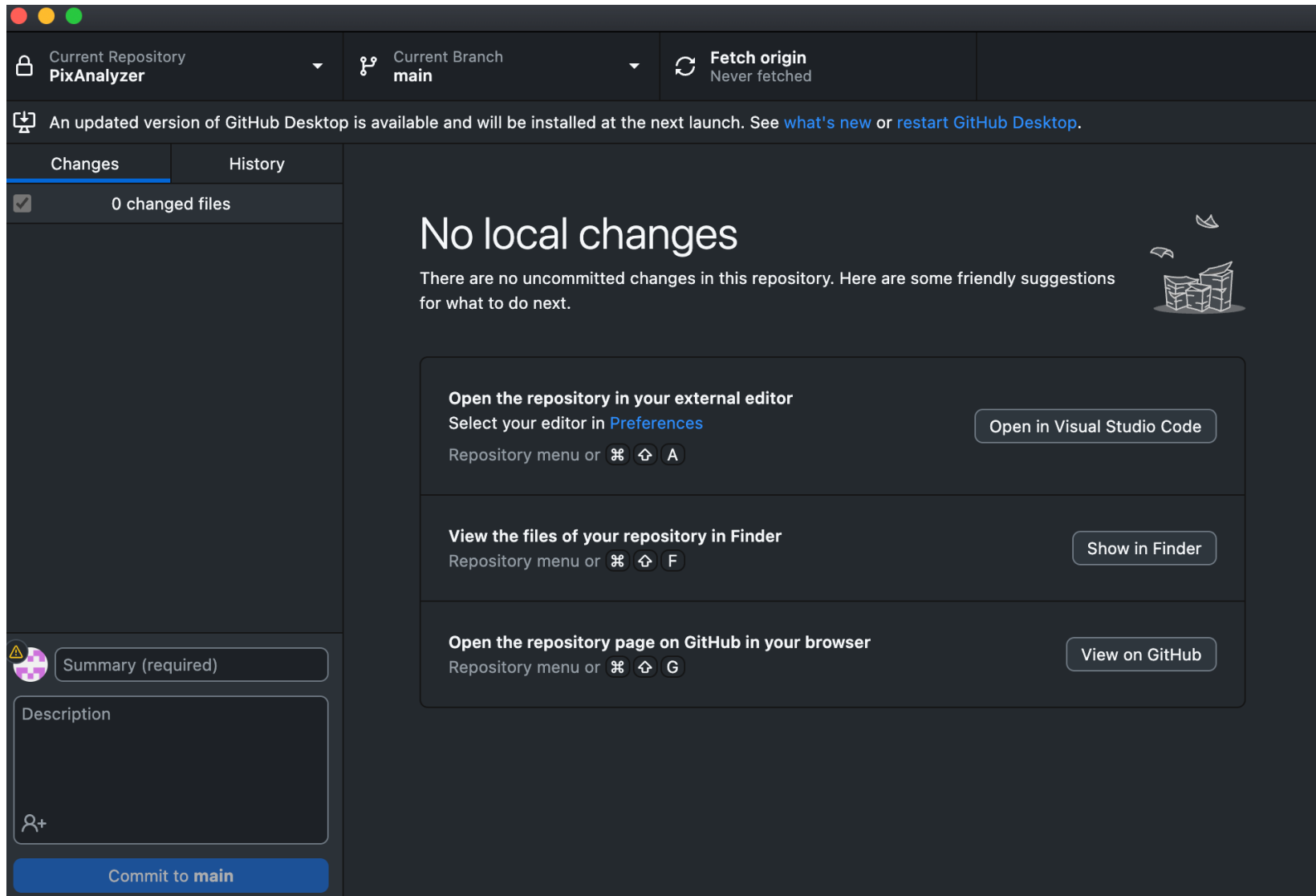
実際の作業手順 – クローンする



このパスに
クローンされる

実際の作業手順 – クローンする

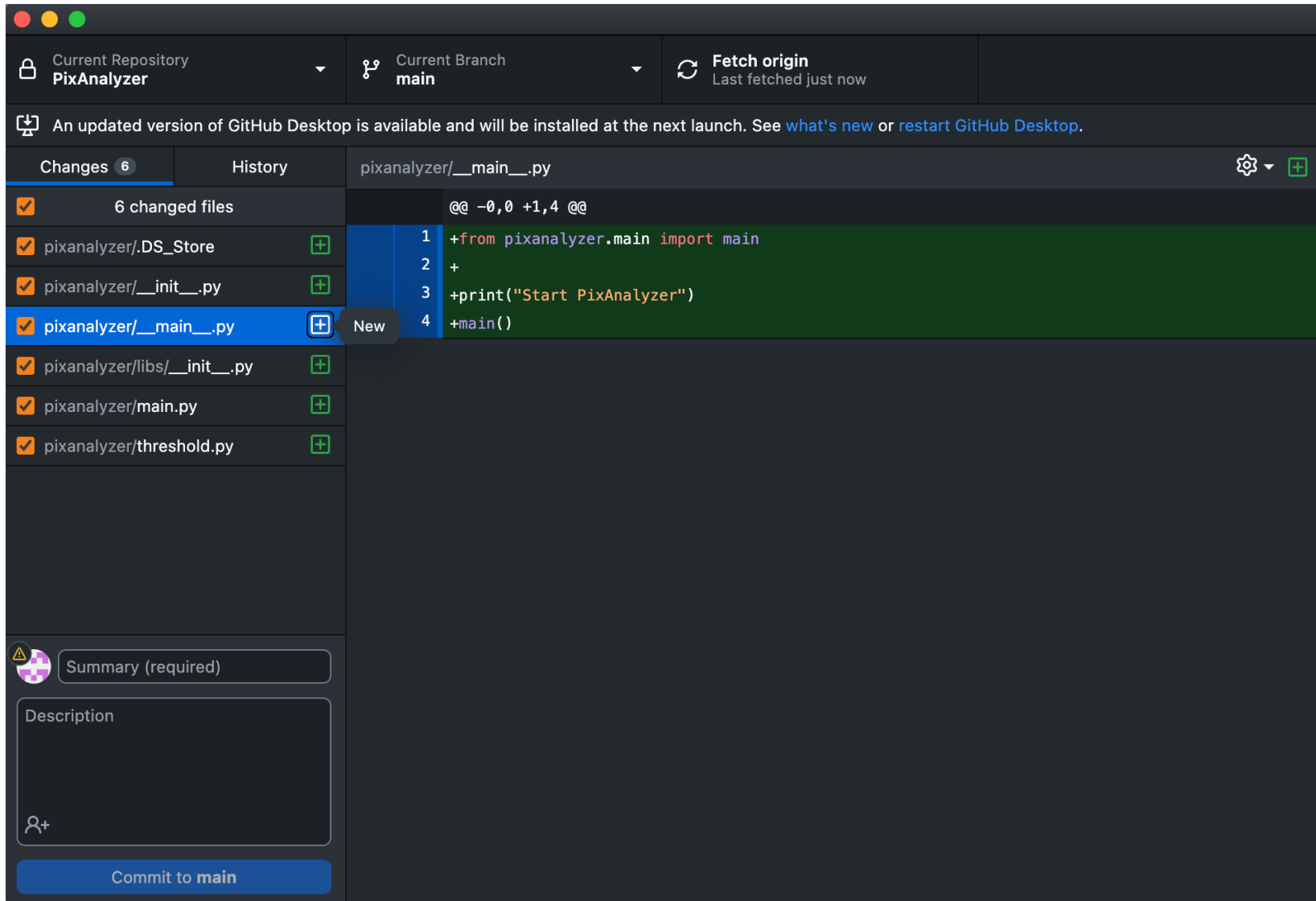
@GitHub Desktop



実際の作業手順 – コミット

@GitHub Desktop

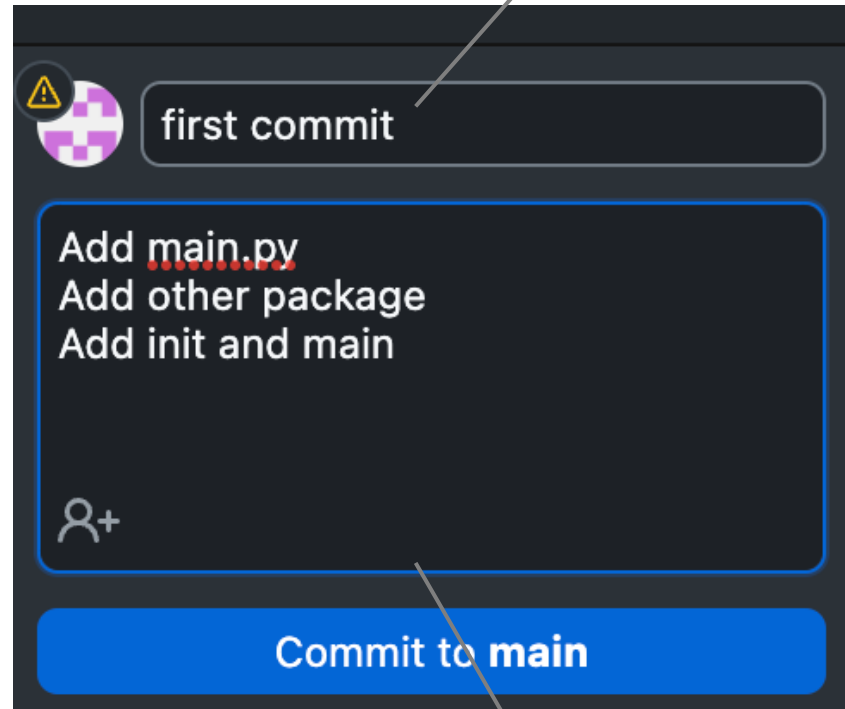
・・・プログラムに変更を加えると。。。



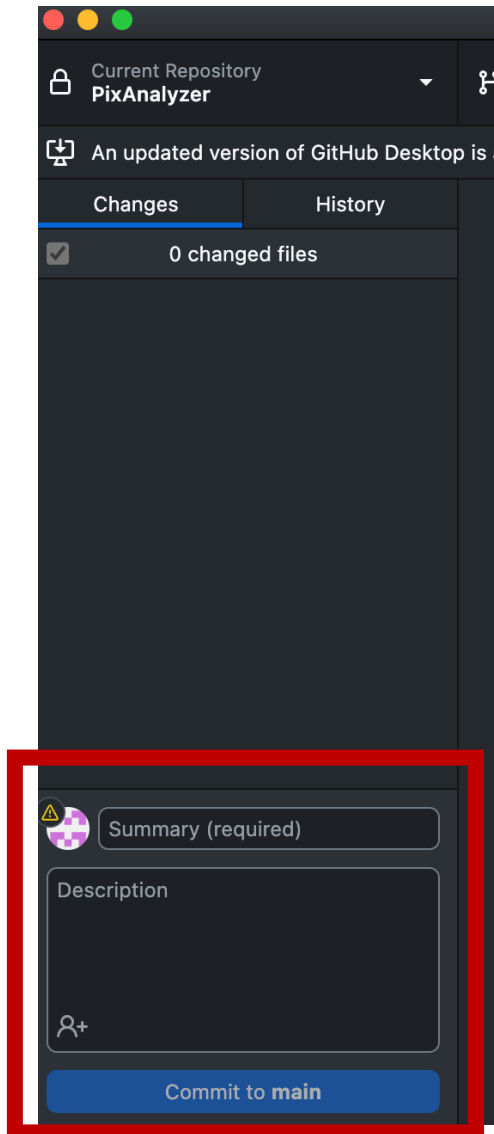
実際の作業手順 – コミット

@GitHub Desktop

簡単なコミットの説明



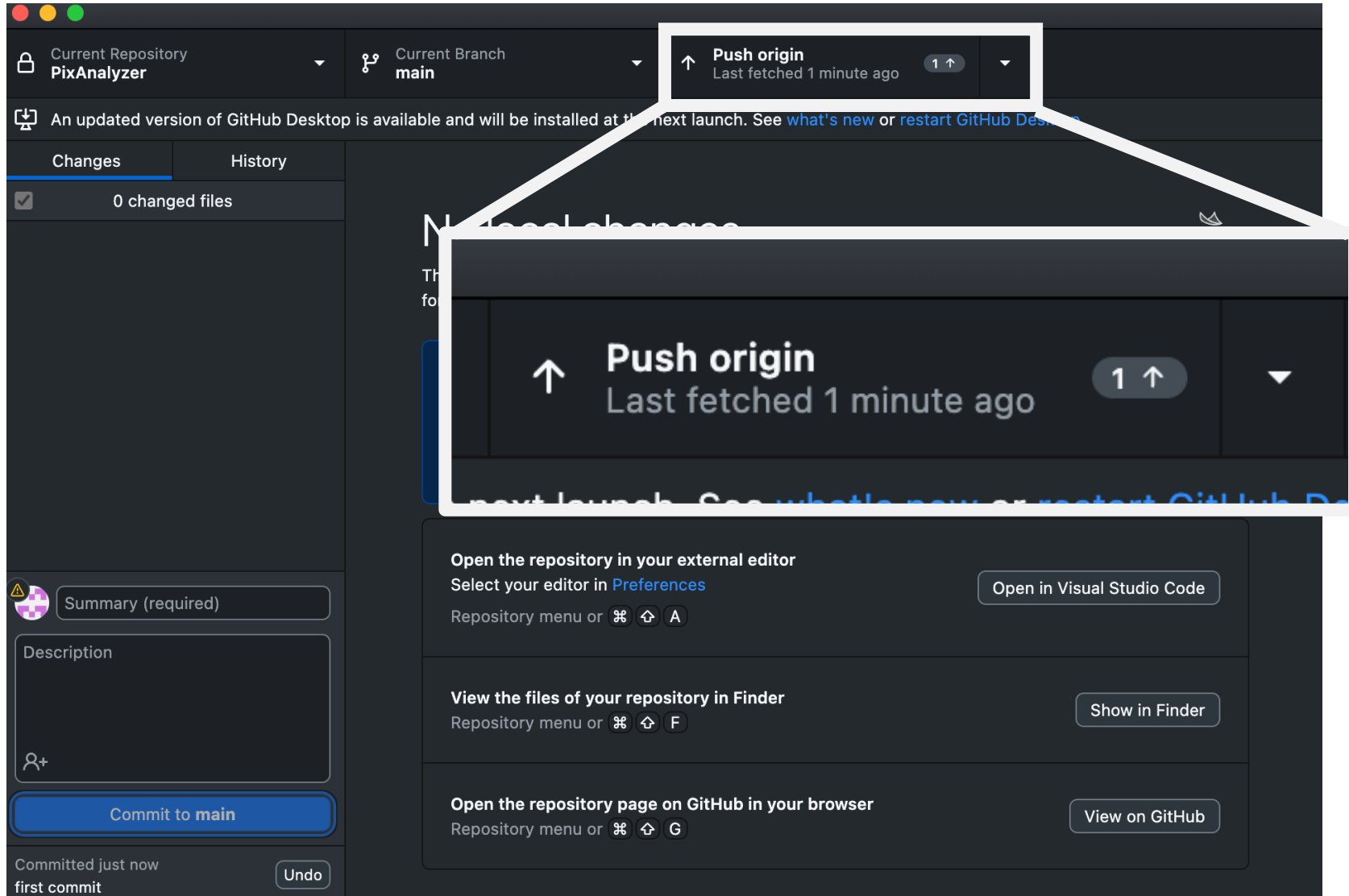
詳しいコミットの説明



*慣例的に最初のコミットは
“first commit”とすることが多い

実際の作業手順 – プッシュ

@GitHub Desktop



実際の作業手順 – プッシュ

The screenshot shows the GitHub web interface for the repository `HMYamano / PixAnalyzer`. The `Code` tab is selected, displaying the file structure on the left and the content of `__main__.py` on the right.

File Structure (Left Panel):

- `pixanalyzer` (folder)
 - `libs` (folder)
 - `.DS_Store` (file)
 - `__init__.py` (file)
 - `__main__.py` (file, selected)
 - `main.py` (file)
 - `threshold.py` (file)
- `.gitignore` (file)
- `LICENSE` (file)
- `README.md` (file)

File Content (Right Panel):

The file `__main__.py` is shown, committed by `yamanouchi` in the `first commit`. The code content is:

```
1  from pixanalyzer.main import main
2
3  print("Start PixAnalyzer")
4  main()
```

Metadata for the file: 4 lines (3 loc) · 69 Bytes. A note indicates "Code 55% faster with C".

実際の作業手順 – 履歴を見る

@GitHub Desktop

The screenshot shows the GitHub Desktop application interface. At the top, the 'Current Repository' is 'PixAnalyzer' and the 'Current Branch' is 'developer'. The 'Fetch origin' button shows 'Last fetched 2 minutes ago'. Below this, the 'Changes' tab is selected, and the 'History' tab is highlighted with a red rectangle. The 'Add information' section shows a commit by 'yamanouchi' with 5 changed files, including README.md, environment.yml, and pixanalyzer/main.py. The 'first commit' section shows a commit by 'yamanouchi' 2 hours ago. The 'Initial commit' section shows a commit by 'Hayato M Yamanouchi' 2 hours ago. The right pane shows a diff view for the commit, with line 1 highlighted in red. The diff content is as follows:

```
@@ -1,51 @@
-# PixAnalyzer
+
+PixAnalyzer is a tool for quantifying the motion of objects.
+It calculates pixel changes and quantifies the overall motion, which cannot be quantified by tracking or object recognition.
+
+
+
+### Installation
+
+#### Step1: Clone this repository
+
+#### Step2: Install [Anaconda](https://www.anaconda.com)
+
+#### Step3: Build an environment using the yaml file
```

適切なコミット/プッシュのタイミング 25

- ・ ・ ・ 履歴を戻るときに自分がわかりやすい単位で

- ・ **コミット** ・ ・ ・ 1タスク単位で行う

- Ex. 一つのバグの修正

- ある統計解析のコードの追加・修正

- ・ **プッシュ** ・ ・ ・ タスクのまとめり単位

- Ex. 一連の統計解析の追加

- 大きな機能の追加

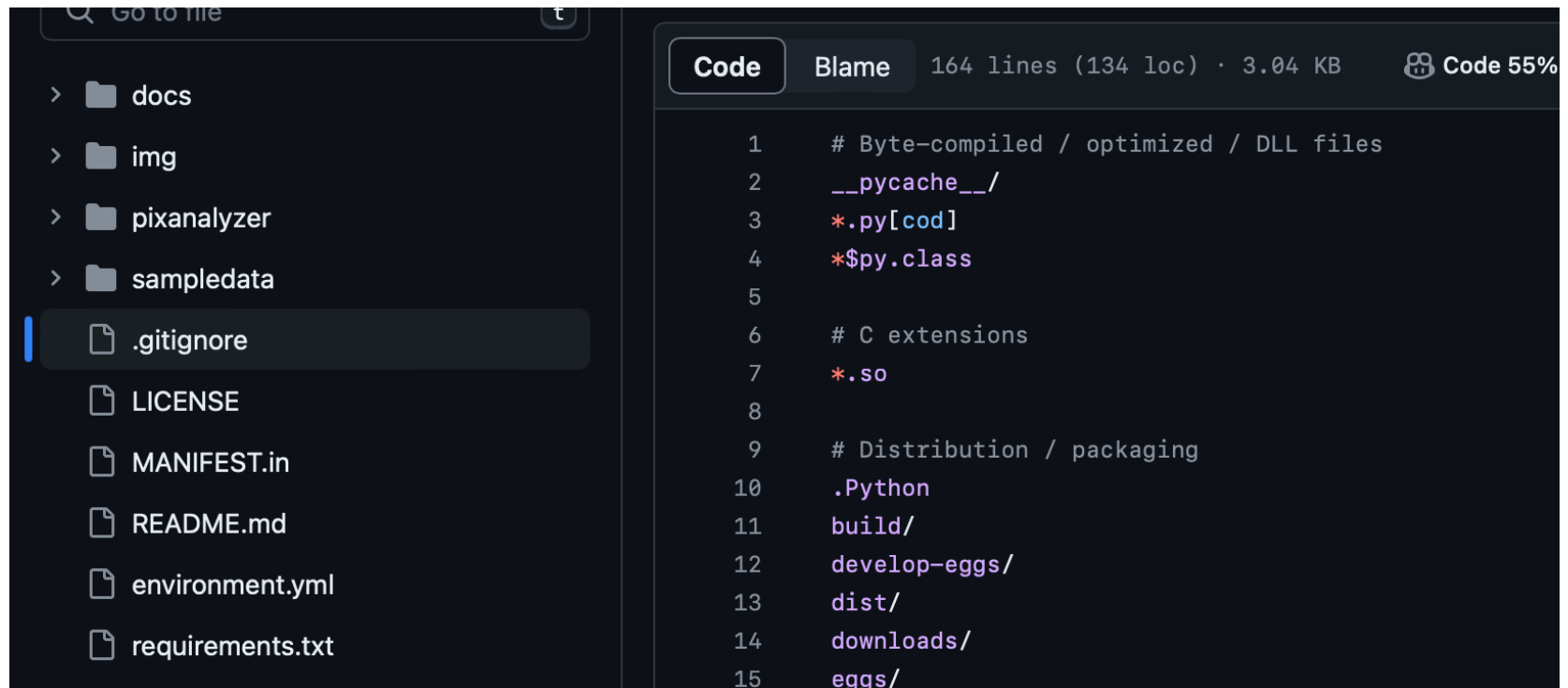
* コミットにはそのコミットが何であるかのコメントと共に
共に行う必要がある

* チームで行う場合には、チームで共有しておくこと

.gitignoreファイルについて

- ・・・レポジトリで共有しないファイルの指定

レポジトリに預けられるファイルの大きさには限界がある
→ 動画とかは入れたくない



```
Code Blame 164 lines (134 loc) · 3.04 KB Code 55%
1  # Byte-compiled / optimized / DLL files
2  __pycache__/
3  *.py[cod]
4  *$py.class
5
6  # C extensions
7  *.so
8
9  # Distribution / packaging
10 .Python
11 build/
12 develop-eggs/
13 dist/
14 downloads/
15 eggs/
```

.gitignoreファイルについて

- ・ ・ ・ レポジトリで共有しないファイルの指定
- ・ 特定のファイルを入れたくない
ファイル名全部を書く (Ex. sample.txt)
- ・ 特定のファイル形式を入れたくない
アスタリスクと拡張子 (Ex. *.mp4)
- ・ 特定のフォルダを入れたくない
フォルダ名とスラッシュ (Ex. Sample/)

あとはただ羅列するだけ

ブランチ

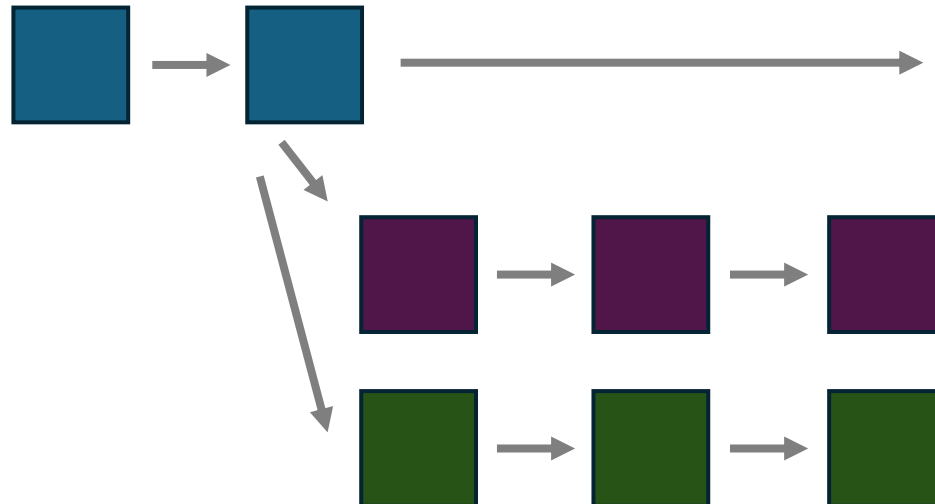
- ・・・履歴の流れを分岐して記録していく



プログラムの統計の検討もしたいし、
新たなデータの追加もしたいし、
やりたいことがいっぱい・・・

ブランチを作る(切る)ことで
それぞれに特化して作業を進められる

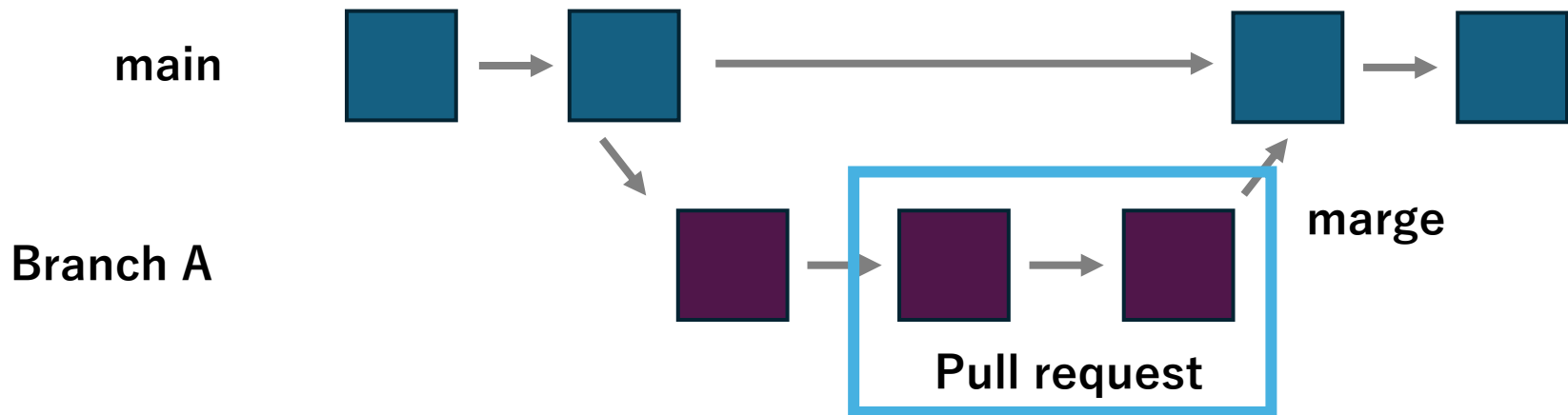
元コード



機能の追加

バグの修正

ブランチをマージする

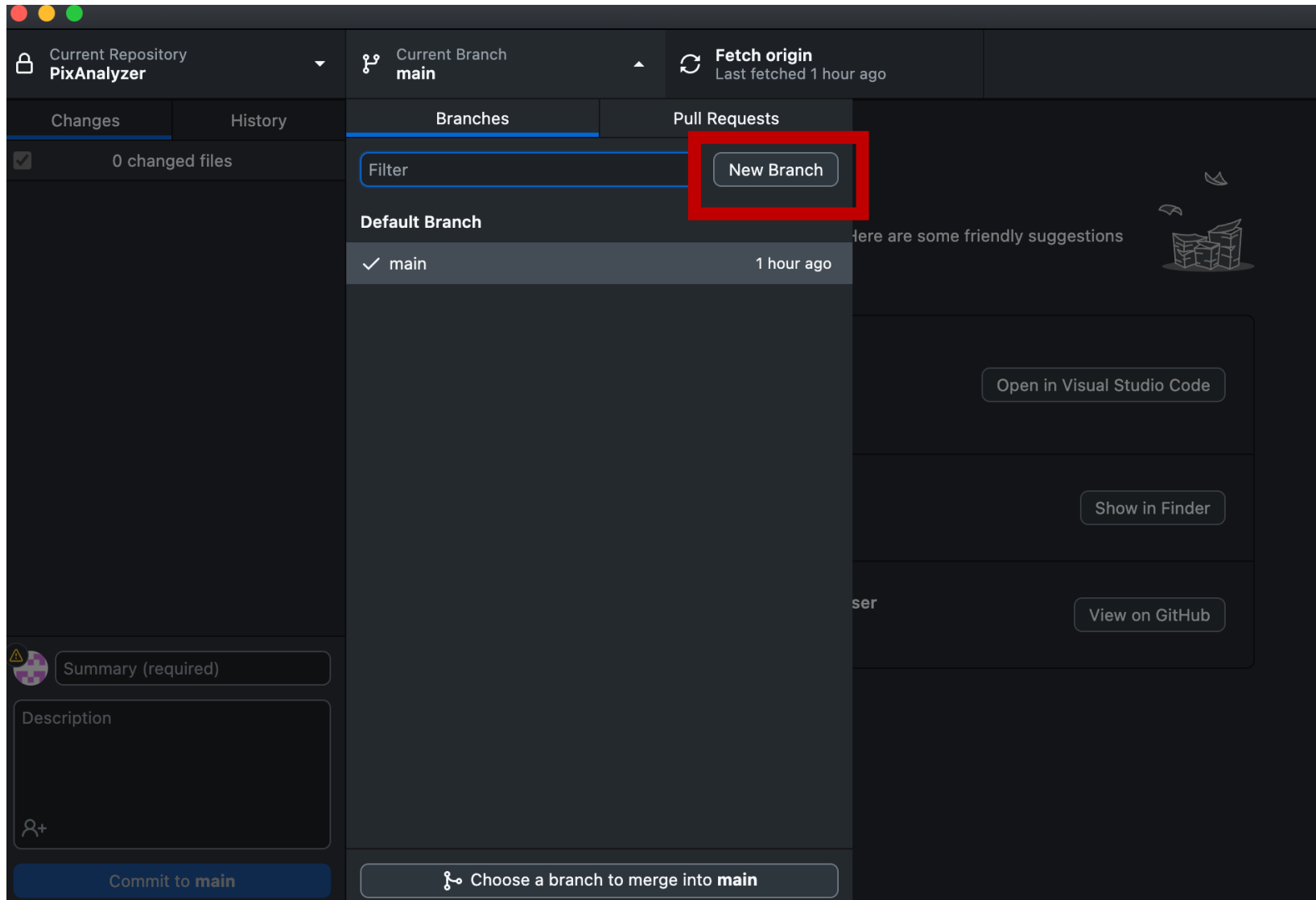


ブランチのマージにはプルリクエスト(Pull request)を行なって、承認作業(自分or共同開発者)が必要となる

他の変更と変更内容が異なったりする場合には、プルリクエストでどれを反映させるかを選択、修正を行う

実際の作業手順 – ブランチを切る

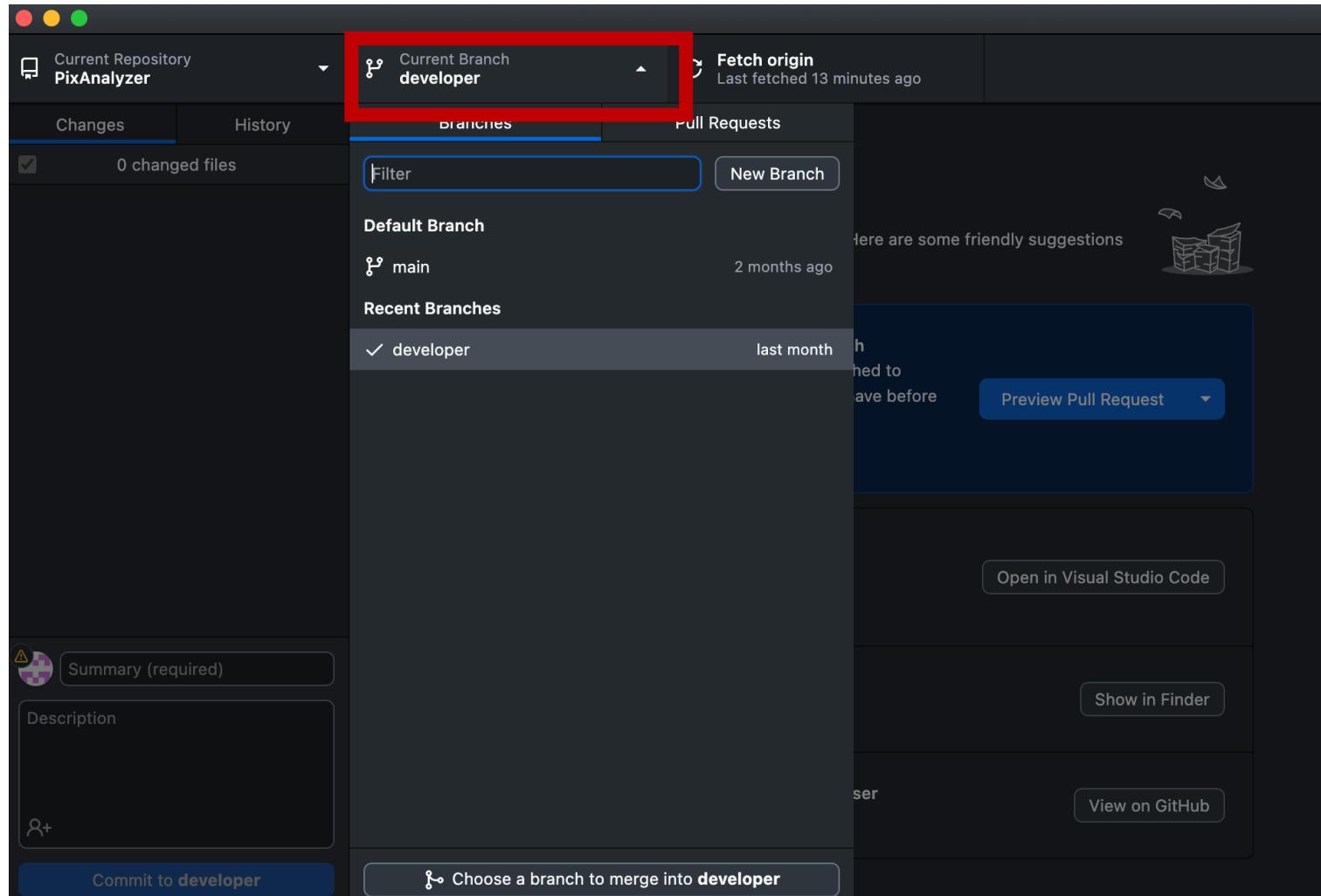
@GitHub Desktop



実際の作業手順 – ブランチを切る

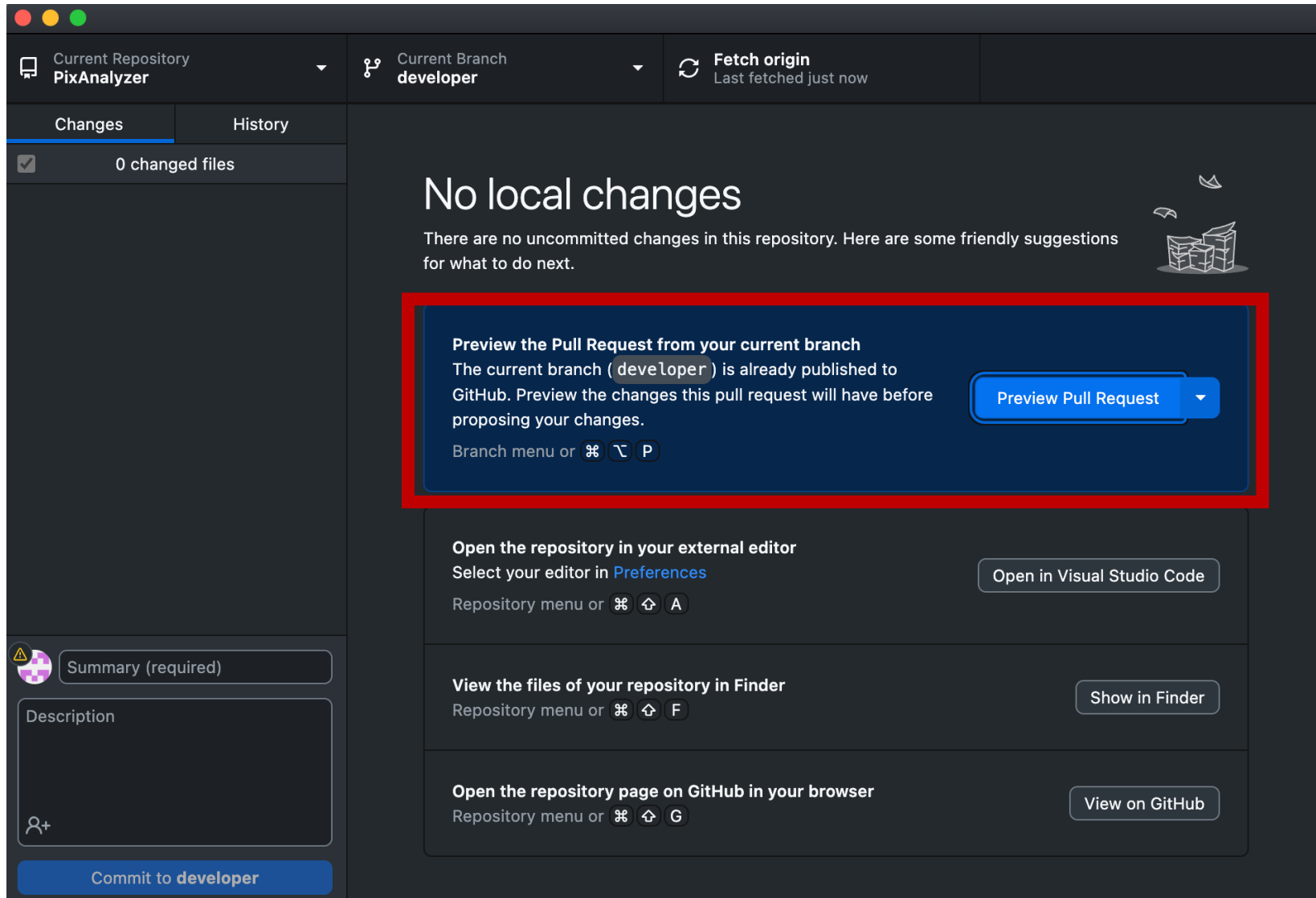
@GitHub Desktop

- • • Current Branchを選択すると、ローカルレポジトリで勝手にブランチが切り替わる



実際の作業手順 – Pull request

@GitHub Desktop



実際の作業手順 – Pull request

@GitHub Desktop

・・・ブランチ間の変更を見る！

Open a Pull Request

Merge 3 commits into `base: main` from `developer`.

Showing changes from all commits

<code>.gitignore</code>		...	@@ -158,3 +158,7 @@ cython_debug/
<code>README.md</code>		158 158	# and can be added to the global gitignore or merged into this file. For a more nuclear
<code>environment.yml</code>		159 159	# option (not recommended) you can uncomment the following to ignore the entire idea folder.
<code>img/logo.png</code>		160 160	#.idea/
<code>pixanalyzer/libs/__init__.py</code>			161 +
<code>pixanal.../thre... → pixanal.../thre...</code>			162 +# Custom exception
<code>pixanalyzer/main.py</code>			163 +.DS_Store
			164 +*.ai

✓ **Able to merge.** These branches can be automatically merged.

Cancel Create Pull Request

やってみよう

@GitHub

- ・ ・ ・ Create pull requestを押すとGitHubに飛ばされる

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base: main ← compare: developer ✓ Able to merge. These branches can be automatically merged.

Add and edit information

Write Preview

Edit something

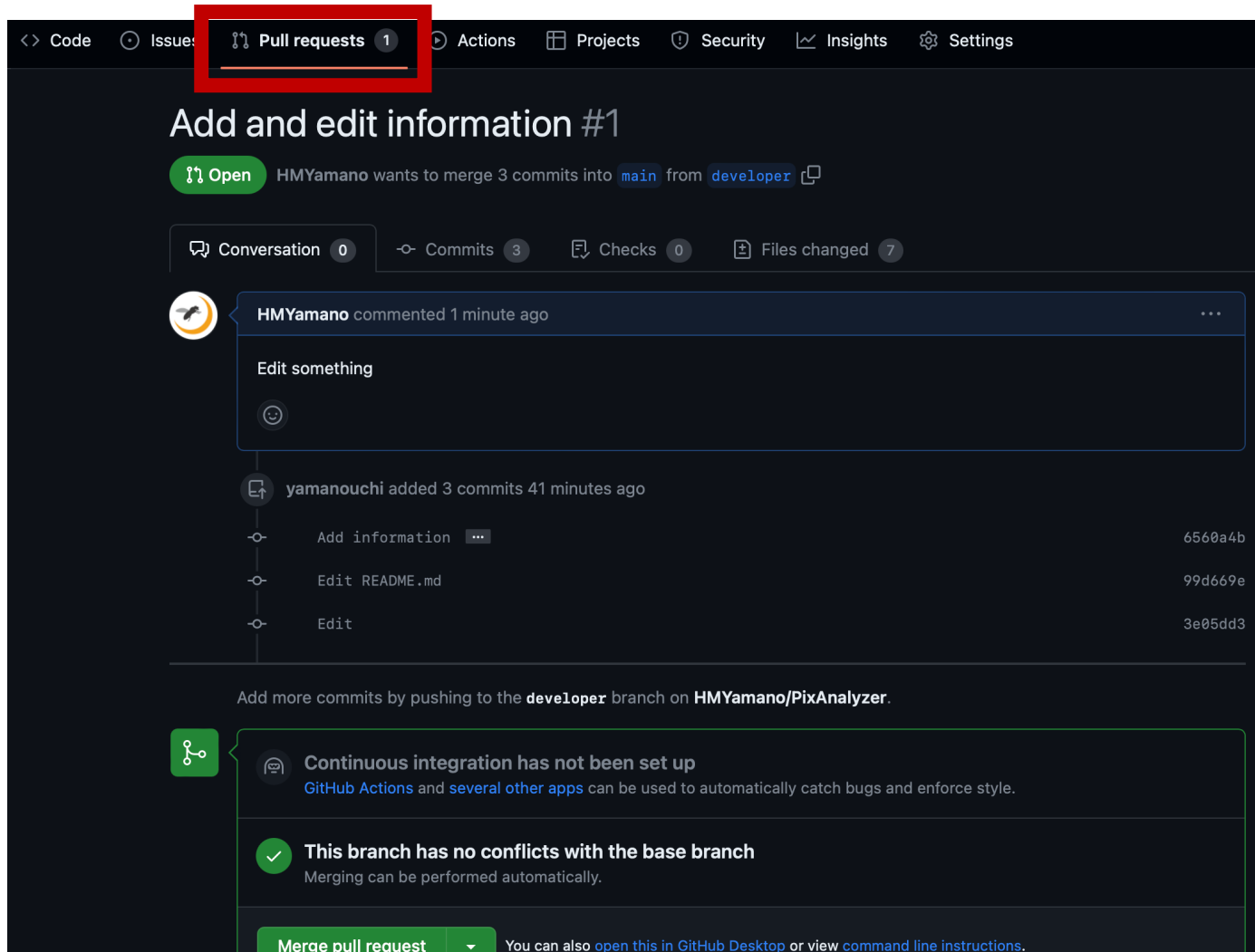
Attach files by dragging & dropping, selecting or pasting them.

Create pull request

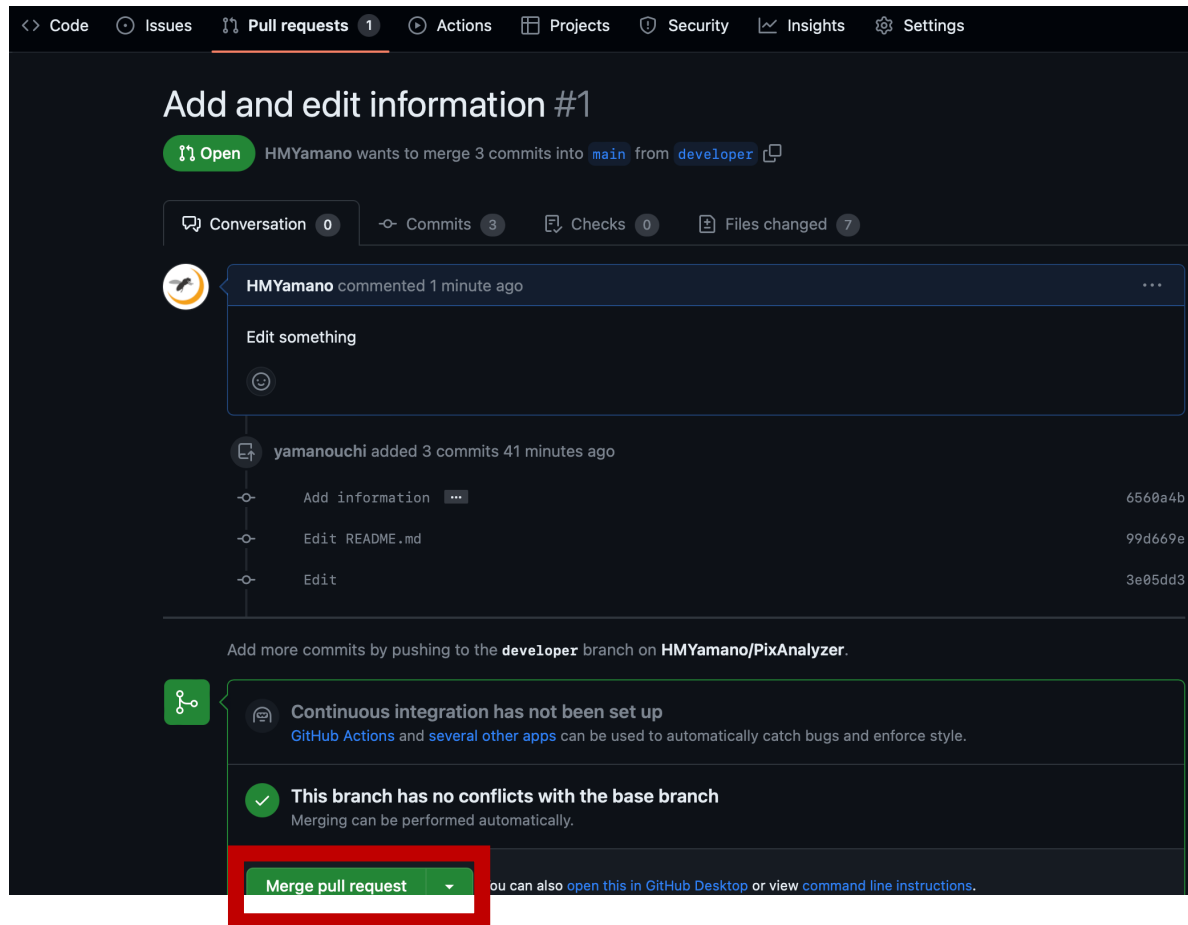
どんな修正を行なったのかなどの、
Pull requestの内容を記述する

やってみよう

- ・ ・ ・ Pull requestに表示される。



やってみよう



OKであれば、“Merge pull request”を、
Noであれば、修正依頼や議論、拒否を行う

やってみよう

- ・ ・ ・ Mergeすると、pull requestが閉じられる

The screenshot displays a GitHub pull request interface. At the top, a comment from HMYamano is visible. Below it, a commit history shows three commits by yamanouchi. The main section indicates that HMYamano has merged commit 3359d2e into the main branch. A 'Revert' button is present. At the bottom, a purple banner states 'Pull request successfully merged and closed' and suggests deleting the developer branch.

HMYamano commented 1 minute ago

Edit something

yamanouchi added 3 commits 41 minutes ago

- Add information ... 6560a4b
- Edit README.md 99d669e
- Edit 3e05dd3

HMYamano merged commit 3359d2e into main now [Revert](#)

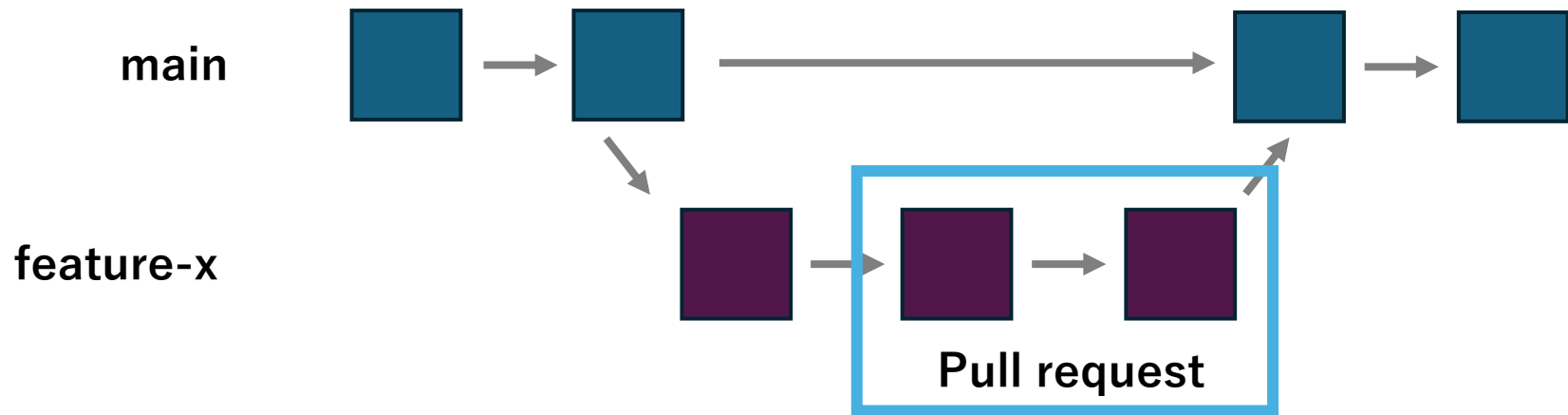
Pull request successfully merged and closed [Delete branch](#)

You're all set—the `developer` branch can be safely deleted.

GitHubのブランチモデル

・・・プログラムのバージョン管理方法のモデルフロー
共同制作するときにはどのようにするか

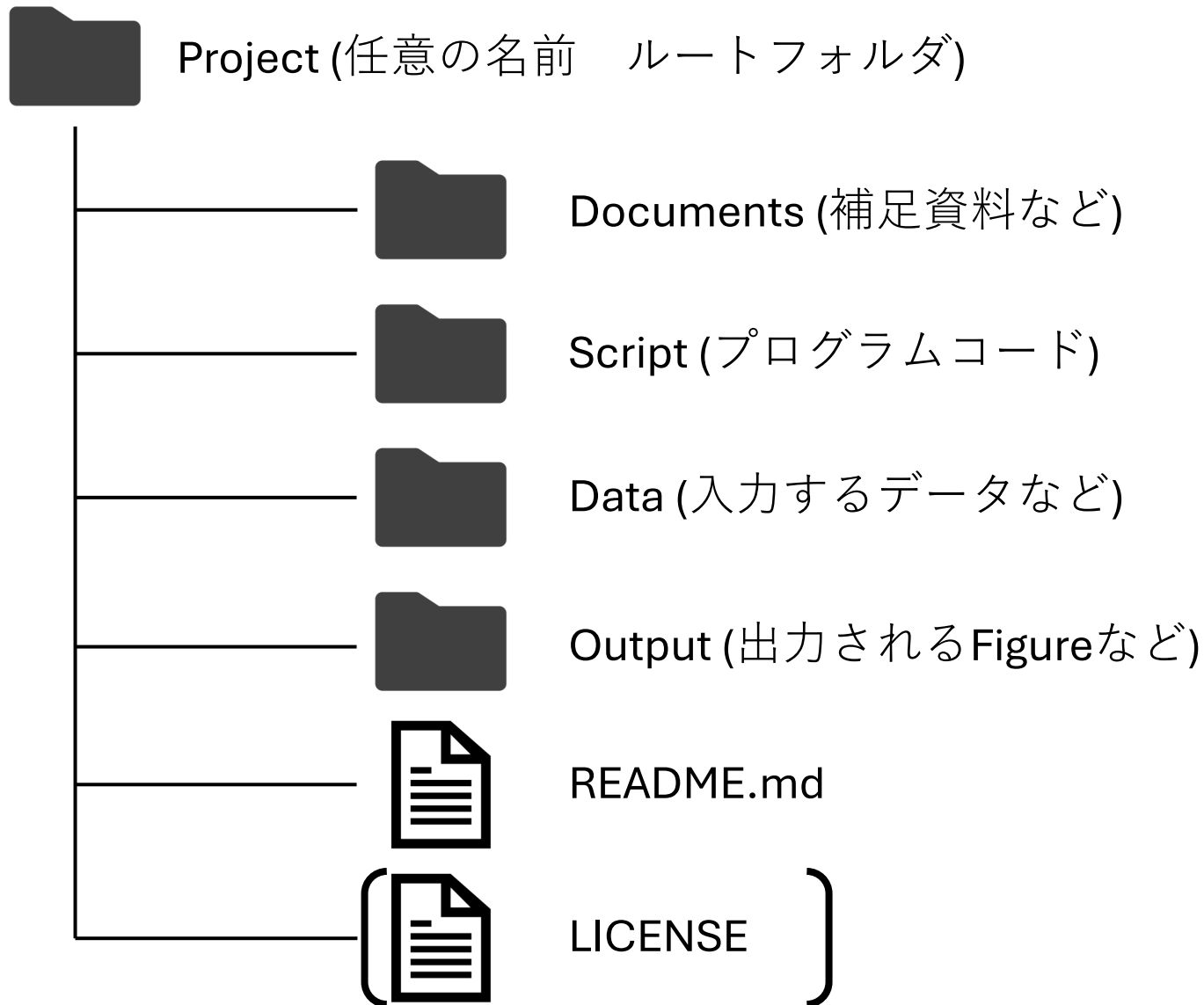
GitHub Flow



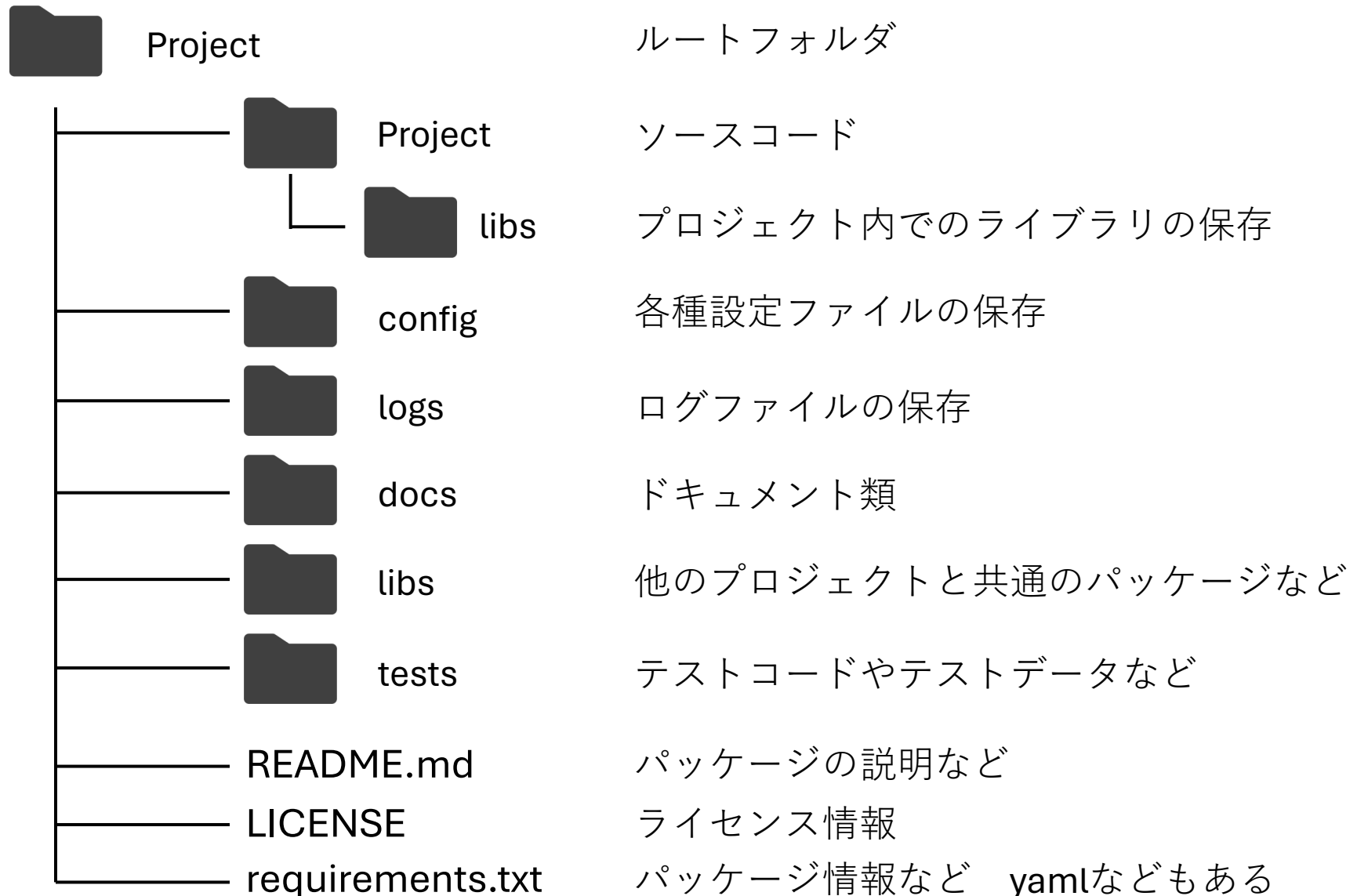
ほかにもgit-flowモデルなどがあり、プログラムの規模やソフトウェアの種類によって変わってくる。

**大切なのは、プログラムの共同管理の方法を
チーム内で共有すること**

プロジェクトのファイル構成の参考（解析コード）

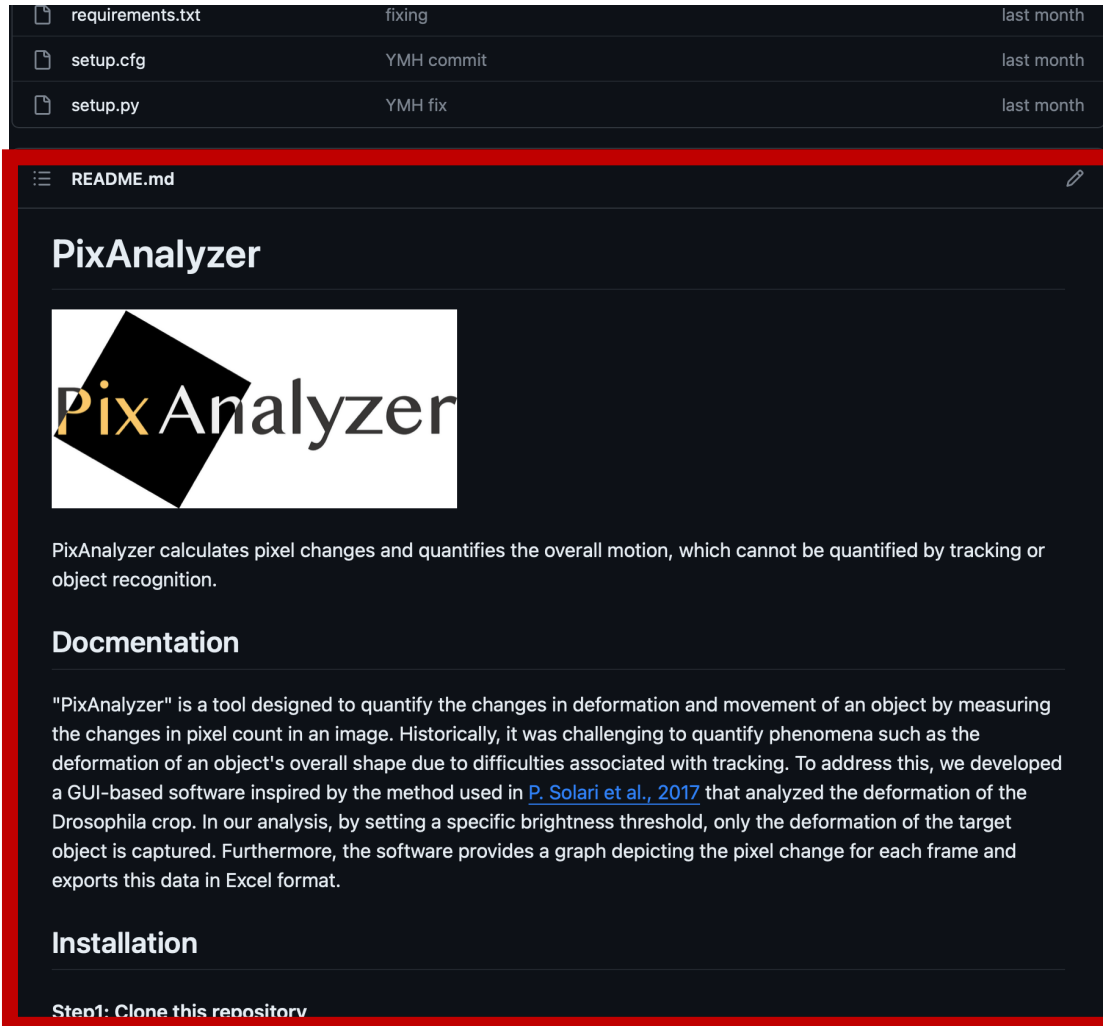


プロジェクトのファイル構成の参考 (OSS) ⁴⁰



README.mdについて

・・・プログラムの説明をするもの



GitHub上でプログラムの
下に表示される部分

ぱっと見でどのような
プログラムかを理解
するために重要！！！！

README.mdについて

“マークダウン記法”で書かれる

```
▼ # PixAnalyzer
<!-- ![Logo](img/logo.png) -->


PixAnalyzer calculates pixel changes and quantifies the overall mo

## Documentation
"PixAnalyzer" is a tool designed to quantify the changes in deform

▼ ## Installation

#### Step1: Clone this repository

#### Step2: Install [Anaconda](https://www.anaconda.com)

▼ #### Step3: Build an environment using the yaml file

In Anacnda prompt
...
conda env create -f <path/to/environment.yml>
...
```

マークダウン記法 チートシート

見出し	# 見出しh1 ## 見出しh2	見出しh1 見出しh2
リスト	* テキスト1 * テキスト2 * テキスト3	• テキスト • テキスト • テキスト
番号付きリスト	1. テキスト1 2. テキスト2 3. テキスト3	1. テキスト1 2. テキスト2 3. テキスト3
空行・改行	行1 _ _ (←半角スペース2つ) 行2 行3	行1 行2 行3
リンクの挿入	[タイトル](URL)	タイトル

マークダウン記法 チートシート

コードの挿入	<code>``言語:タイトル コード ``</code>	
引用	<code>> テキスト1 >> テキスト2</code>	テキスト テキスト
太字	<code>**テキスト**</code>	テキスト
斜体	<code>*テキスト*</code>	<i>テキスト</i>
水平線	<code>***</code>	

ライセンスについて

・・・プログラムコードを利用する人へのルール

オープンソース(OS)のソフトウェアの公開が推奨されているが、自由勝手にプログラムを自分のもののよう公開されるのを防ぎたい！！

ユーザーのプログラム使用の範囲を限定する

Ex. コードの引用元を付けさせる
商用利用不可 など

解析コードに関しても同様に考えなければならい

ライセンスの種類について

MIT, BSD, Apache License 2.0, GPL v2.0 など。。。

自分が適応したい範囲に合わせて
選択する必要がある

一番利用者側の自由度が高いのが**MITライセンス**

- ・コピー利用、配布、変更の追加、変更を加えたものの再配布、商用利用、有料販売など可
- ・一切の保証、責任は製作者は取らない

ライセンスの適応の仕方

GitHubのフォルダーの中にLICENCEというのを入れる

適応したいライセンスを選択すると、
GitHubが自動的に作ってくれる

```
1  MIT License
2
3  Copyright (c) 2023 Hayato M Yamanouchi
4
5  Permission is hereby granted, free of charge, to any person obtaining a copy
6  of this software and associated documentation files (the "Software"), to deal
7  in the Software without restriction, including without limitation the rights
8  to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
9  copies of the Software, and to permit persons to whom the Software is
10 furnished to do so, subject to the following conditions:
11
12 The above copyright notice and this permission notice shall be included in all
13 copies or substantial portions of the Software.
14
15 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
16 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
17 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
18 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
19 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
20 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
21 SOFTWARE.
```

コードを論文に載せるため

・・・**Zenodo**でGitHubと連携して、レポジトリにDOIを発行する



<https://upload.wikimedia.org/wikipedia/commons/e/e8/Zenodo-gradient-square.svg>

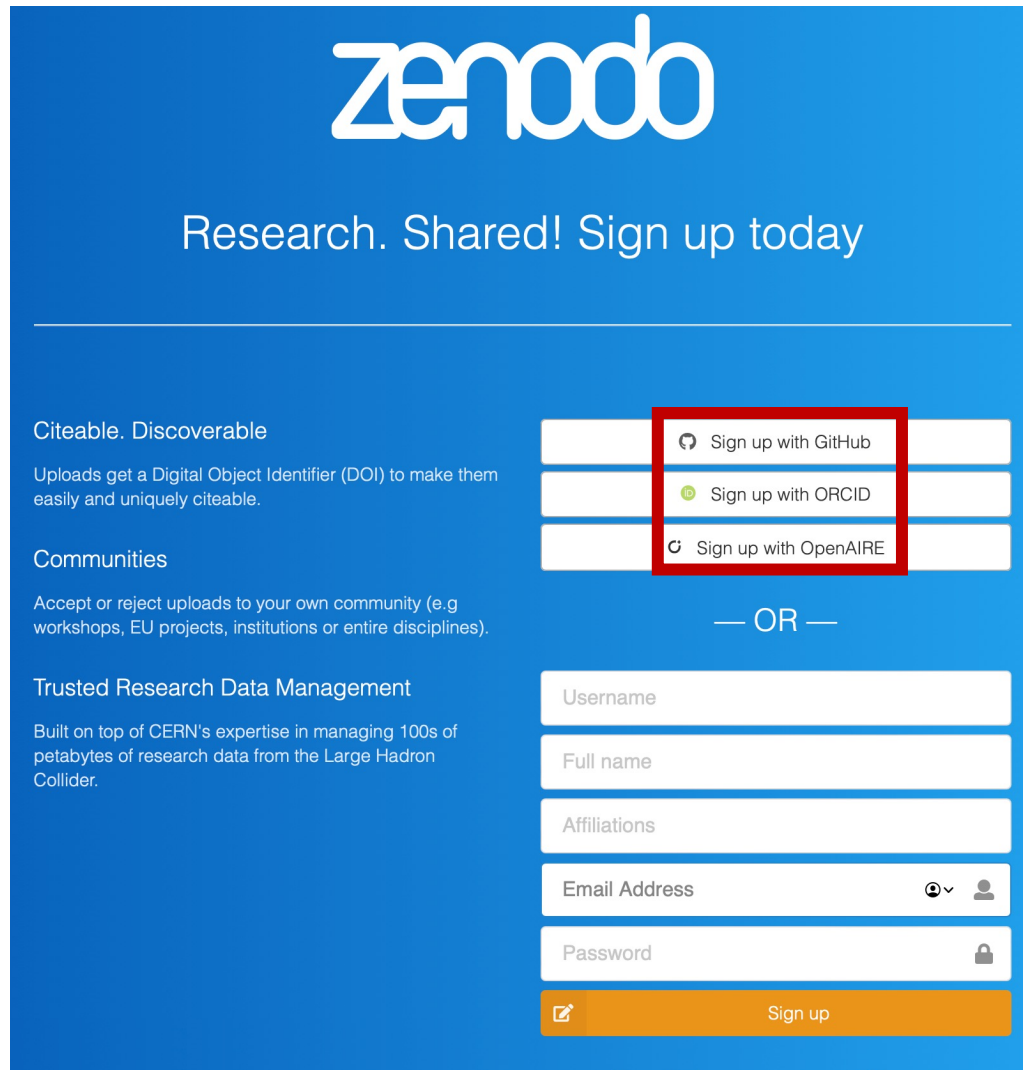
“Zenodoは研究データリポジトリである。研究者がデータセットを置くための場所を提供するためにOpenAIREとCERNによって作られた。2013年に始められ、いかなる研究分野の研究者も50 GBまでファイルをアップロードできるようになった。”

from Wikipedia

もちろんGitHubと連携せず、Zenodo単体でも
データセットの保存、DOI発行は可能

ZenodoとGitHubを連携させる

- ・ ・ ・ ZenodoにGitHubアカウントでSign upするだけ！



The image shows the Zenodo sign-up page. The header features the Zenodo logo and the text "Research. Shared! Sign up today". Below this, there are three sections: "Citeable. Discoverable", "Communities", and "Trusted Research Data Management". To the right of these sections is a sign-up form. The form has three buttons for social login: "Sign up with GitHub", "Sign up with ORCID", and "Sign up with OpenAIRE". These buttons are highlighted with a red box. Below these buttons is a separator "— OR —". The form then has five input fields: "Username", "Full name", "Affiliations", "Email Address" (with a dropdown arrow and a user icon), and "Password" (with a lock icon). At the bottom of the form is an orange "Sign up" button.

zenodo

Research. Shared! Sign up today

Citeable. Discoverable

Uploads get a Digital Object Identifier (DOI) to make them easily and uniquely citeable.

Communities

Accept or reject uploads to your own community (e.g workshops, EU projects, institutions or entire disciplines).

Trusted Research Data Management

Built on top of CERN's expertise in managing 100s of petabytes of research data from the Large Hadron Collider.

Sign up with GitHub

Sign up with ORCID

Sign up with OpenAIRE

— OR —

Username

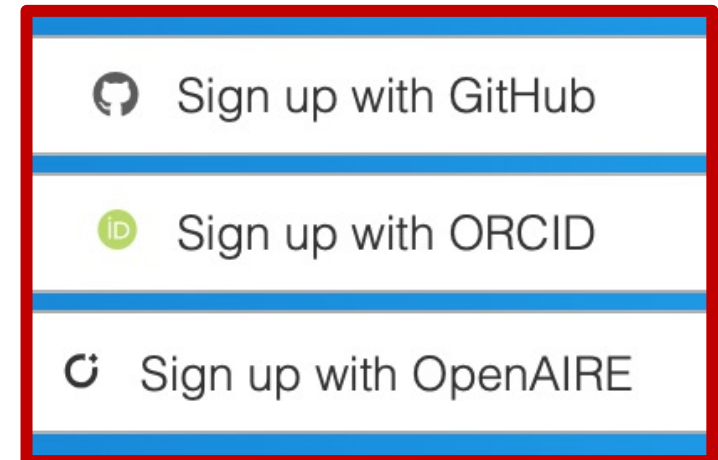
Full name

Affiliations

Email Address

Password

Sign up



The image shows a close-up of the social login options on the Zenodo sign-up page. It features three buttons: "Sign up with GitHub", "Sign up with ORCID", and "Sign up with OpenAIRE". These buttons are highlighted with a red box.

Sign up with GitHub

Sign up with ORCID

Sign up with OpenAIRE

GitHubのレポジトリにDOIを発行する



GitHub Repositories

(updated 6 months ago)



Sync now



Get started

1 Flip the switch

Select the repository you want to preserve, and toggle the switch below to turn on automatic preservation of your software.

ON



(example)

2 Create a release

Go to GitHub and [create a release](#) . Zenodo will automatically download a .zip-ball of each new release and register a DOI.

3 Get the badge

After your first release, a DOI badge that you can include in GitHub README will appear next to your repository below.

DOI

10.5281/zenodo.8475

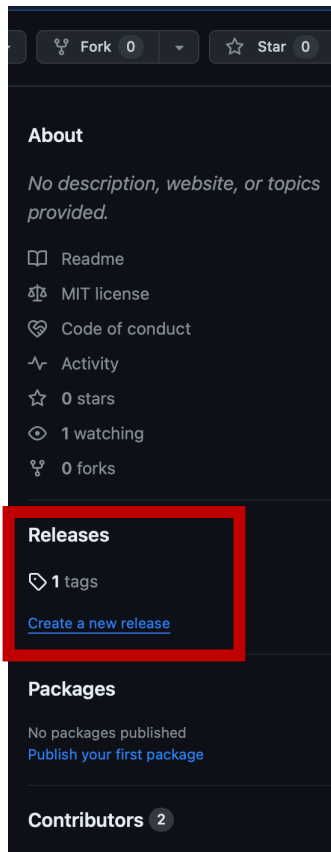
(example)

0. レポジトリをPublicにする
1. Zenodoで選択する
2. GitHubで”release”を作成する
3. レポジトリのREADME.mdにDOIの表記を入れる

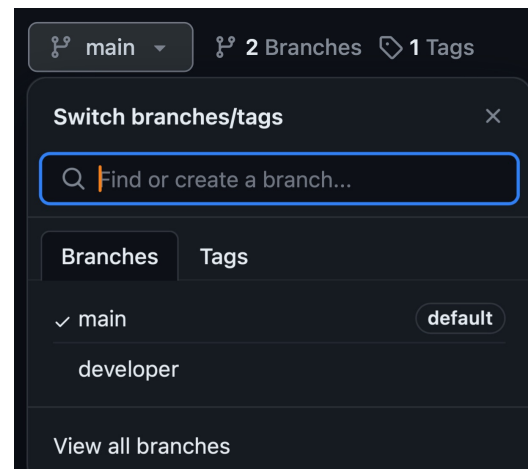
レポジトリのReleasesを作る

- ・・・コードのバージョンの発行 (ver 1.0.0みたいな)

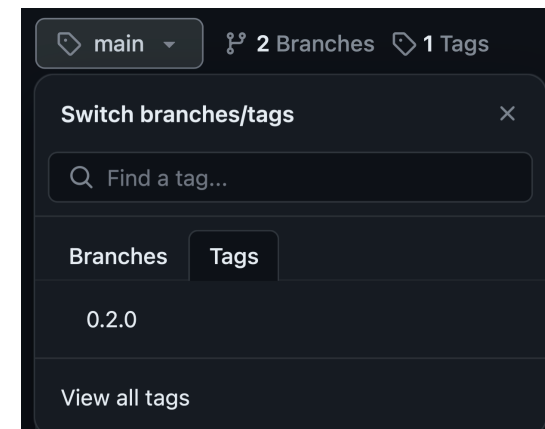
発行することで、その時のバージョンの
コードが保存される



Branches



Tags (version)



GitHubのレポジトリにDOIを発行する

52

Zenodoには、DOIを発行した時のバージョンのフォルダが格納されている。

Published March 28, 2023 | Version v1.0

Journal article Open

Edit

New version

Share

19 VIEWS 3 DOWNLOADS

Show more details

Versions

Version v1.0 Mar 28, 2023 10.5281/zenodo.7776688

Cite all versions? You can cite all versions by using the DOI 10.5281/zenodo.7776687. This DOI represents all versions, and will always resolve to the latest one. Read more.

External resources

Available in

HMamano/Event_triggered_YOLO_system

HMamano/Event_triggered_YOLO_system: 1.0

Yamanouchi, Hayato M.¹ Tanaka, Ryoya¹ Kamikouchi, Azusa¹

Show affiliations

The first public release of the event-triggered system using YOLO.

Files

HMamano/Event_triggered_YOLO_system-v1.0.zip

HMamano/Event_triggered_YOLO_system-v1.0.zip

HMamano-Event_triggered_YOLO_system-c7651f3

Colab_training

models

model_100

weights

best.pt 14.4 MB

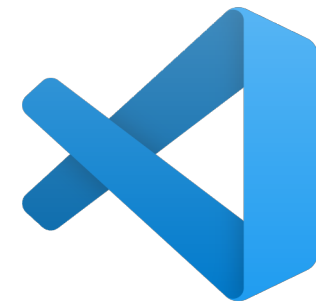
last.pt 14.4 MB

model_1000

weights

GitHubやプログラミングを快適に行うために⁵³

- • • Visual Studio Code (略VS Code)を使いましょう！！
(Visual Studioと混合しないように)
- WindowsやMacで使えるコードエディタ
- 拡張機能でGitHubへのプッシュ/プルなどが可能に
- .yamlや.mdファイルなどの様々な拡張子のファイルの表示、編集ができる
- 使いやすい、知見もいっぱいある



+@要素

- ・ ・ ・ ディスカッションやプロジェクトの進行管理など多様な機能がある

A screenshot of the GitHub navigation bar, which is a dark horizontal strip. It contains several menu items with icons: 'Code' (with a code icon), 'Issues' (with a circle icon), 'Pull requests' (with a branch icon), 'Actions' (with a play icon), 'Projects' (with a grid icon), 'Security' (with a shield icon and a '1' badge), 'Insights' (with a line graph icon), and 'Settings' (with a gear icon). The 'Code' item is highlighted with an orange underline.

- ・ **Issues** ・ ・ ・ プログラムで問題があったときなどに **Issues**を作成することで、その問題の対応などを管理できる
- ・ **Actions** ・ ・ ・ GitHubの拡張パッケージ
Ex. プッシュ時に自動的にコードを整える
- ・ **Insights** ・ ・ ・ プロジェクトの全体の動きを見れる