



# ハンズオンタイム2

では、次にブラウザでgitを開いてログインしておいてください。

リモートリポジトリを作っていきます。

これは、git操作の練習用のレポジトリなので、新しくつくってください。

下のような画面が出ていると思います。

赤マルで囲まれた、「New」ボタンを押してください。

The screenshot shows the GitHub Dashboard interface. On the left sidebar, the 'New' button is circled in red. The main content area features a 'Join GitHub Global Campus!' announcement, a 'UNIVERSE23' event banner, and a 'Latest changes' section. The bottom section shows 'Updates to your homepage feed'.

「Repository name」に「git-practice」を入れてください。

設定を以下のようにしたら、「Create repository」を押してください。

## 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 \*



 b2211700 / git-practice

✓ Your new repository will be created as git-practice-.

The repository name can only contain ASCII letters, digits, and the characters -, ., and \_.

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

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

これで、練習用のリポジトリ作成は終了です。

次に、ローカルリポジトリと今gitで作成したリモートリポジトリを紐づける作業をしていきます。

このような画面が出てきていると思います。

**Quick setup — if you've done this kind of thing before**


 Set up in Desktop or **HTTPS** **SSH**  

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

---

**...or create a new repository on the command line**


```
echo "# git-practice-" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/b2211700/git-practice-.git
git push -u origin main
```



---

**...or push an existing repository from the command line**

```
git remote add origin https://github.com/b2211700/git-practice-.git
git branch -M main
git push -u origin main
```



---

**...or import code from another repository**

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

リモートリポジトリの登録や確認に、「git remote」という操作をします。画像の赤で囲まれているところの操作をします。

そのコマンドは、このページにも書いていますが、以下をコピーしていただいてもOKです。

▶ `git remote add origin https://github.com/"自分のユーザーネーム"/git-practice.git`

ターミナルでコマンドを入力したら、ローカルリポジトリと今gitで作成したリモートリポジトリでのやり取りができるようになります。

以下の、二行も順番にターミナルで入力してください。

▶ `git branch -M main`

▶ `git push -u origin main`

gitの画面をリロードすると画面が変わっていると思います。今プッシュしたmainが入っていることが確認できます。「self-introduction.txt」を開くと、自分で打った自己紹介文が出てきます。

次に、ブランチの操作をしてみます。以下をターミナルで入力し、今どんなブランチがあるのかを確認してください。「main」のみが表示されると思います。「\*」がついているのが、今自分が操作しているブランチです。

```
▶ git branch
```

以下のコマンドで、「first-branch」という名前のブランチを作ります。

```
▶ git branch first-branch
```

もう一度、ブランチを確認してください。「first-branch」が増えていることが確認できます。

```
▶ git branch
```

今操作したいのは「first-branch」であるため、ブランチを移動しましょう。以下では、「switch」を使っていますが「checkout」でもできます。

```
▶ git switch first-branch
```

ブランチを確認すると「first-branch」に「\*」がついていることが確認できると思います。

「first-branch」にいる状態で、先ほど作成した、「self-introduction.txt」に一行足してみましょう。

例)



1行足したあと、ステータスを見てみましょう。

▶ `git status`

以下のようなレスポンスが返ってくると思います。

ここから、今は「first-branch」にいること、まだステージ済みになっていない変更点があるということが読み取れます。



On branch first-branch

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: self-introduction.txt

no changes added to commit (use "git add" and/or "git commit -a")

ここからは、ステータス確認を指示しないので、適宜確認しながら作業を進めてください。

「add」「commit」は以下のコマンドで行ってください。  
もうマスターしていると思うので、説明を省きます。

```
▶ git add self-introduction.txt
```

```
▶ git commit -m "出身地を追記した"
```

コミットコメントは自分で変更してもOKです。

コミットができれば、ログを確認してみましょう。

```
▶ git log
```

以下のように2つコミットできていることが確認できます。

👉 `commit eb013c90835ea0a14405c8d754b8675887b9a525 (HEAD → first-branch)`

Author: b2211700 <b2211700@photon.chitose.ac.jp>

Date: Sat Oct 21 15:05:24 2023 +0900

出身地を追記した

`commit a2cba7ae1a1bfb45026f6ceefbc144082c6f2cf7 (origin/main, main)`

Author: b2211700 <b2211700@photon.chitose.ac.jp>

Date: Sat Oct 21 14:56:58 2023 +0900

Initial commit

では、ローカルでマージしてみます。「main」に「first-branch」をマージします。そのため、ヘッドを「main」にします。以下では、「switch」を使っていますが「checkout」でもできます。

▶ `git switch main`

移動したら早速マージします。

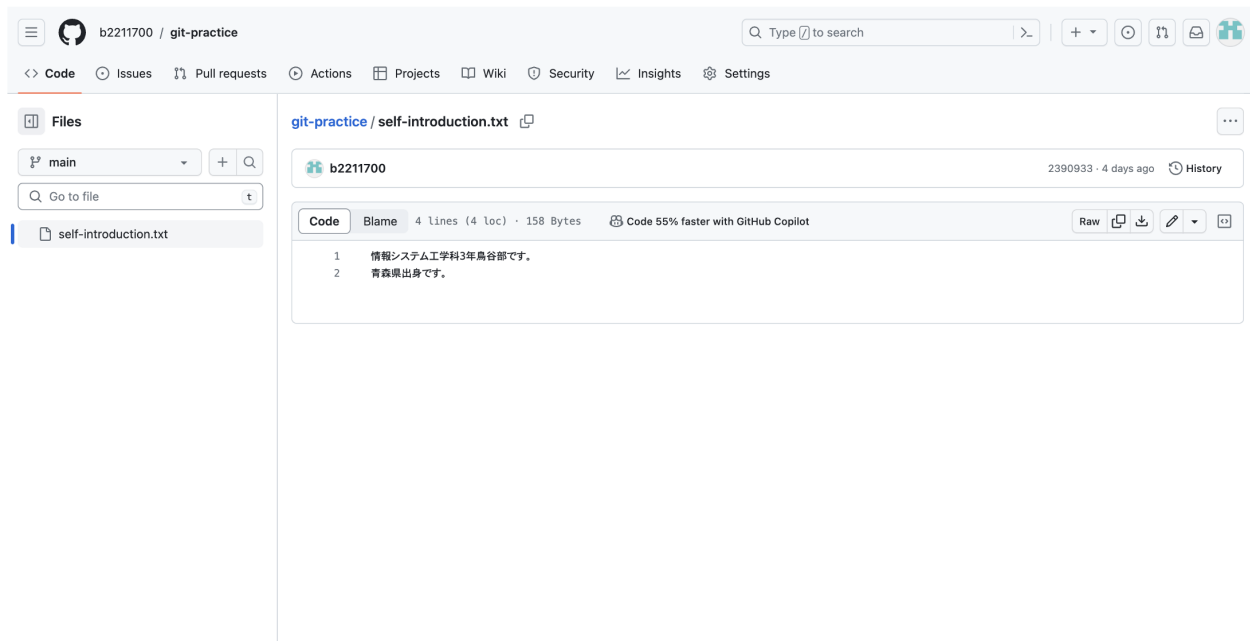
▶ `git merge first-branch`

これで、「main」に「first-branch」がマージされました。  
ここで、これまでの変更をリモートリポジトリにも反映させます。

▶ `git push -u origin main`

ブラウザで反映されているか確認してみてください。

以下のように、自分で書いた2行が確認できると思います。



以上で基本的な操作の体験は終了です。今回は、「main」で自分の名前、「first-branch」で出身を書き、最後に「main」にマージするというものをやりました。今回は、簡易的なものをやりましたが、実際のプロジェクトではもっと操作が複雑になります。基本を抑え、複雑になってもgit操作ができるようにしていきましょう。