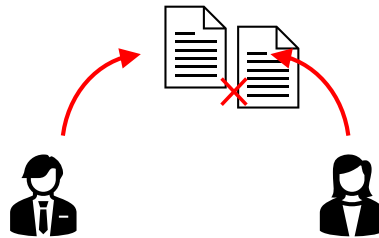


【演習】 Git&GitHub

●バージョン管理とは

チームで作業するときには、ファイルの管理が問題になります。誰がいつ、どのファイルを編集したのか記録しておく必要があります。以下のように、ファイル名に名前を記入する場合もあるかもしれません。すると、ファイルが膨大になります。また、これでは具体的にどのような変更を行ったかは管理できません。さらに、同時にファイルを編集してしまい、誤ってファイルを上書きしてしまうこともあります。

```
240312_ドキュメント_初稿.txt
240312_ドキュメント_木村.txt
240313_ドキュメント_後藤.txt
240313_ドキュメント_桜井.txt
240314_ドキュメント_三和.txt
240314_ドキュメント_木村.txt
240314_ドキュメント_後藤.txt
```



●Git とは

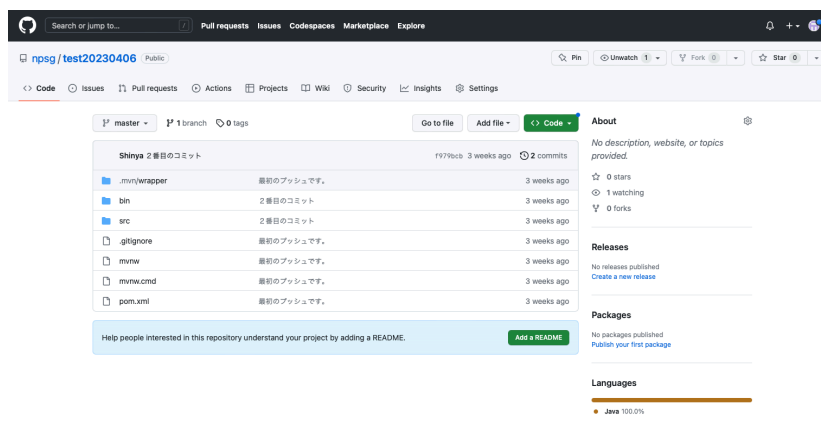
Git とは、ドキュメントやプログラムのソースコードなどの変更履歴を管理する分散型バージョン管理システムです。Git を使うことで、手元に別名のバックアップ用ファイルを作成する必要がありません。Git に変更履歴が保存されるので、効率よくバージョン管理ができます。

以前にも、CVS や Subversion といったバージョン管理システムがありましたが、現在では「分散」型が広く受け入れられています。Git を含めたバージョン管理システムでは、変更履歴を「リポジトリ」と呼ばれる場所に記録します。以前のバージョン管理システムでは、リポジトリが集中管理され、1つであったため、開発者が増えるとリポジトリに不整合が生じていました。

Git では、リポジトリを分散することで、ネットに繋がってなくても作業を継続することができるようになりました。Git は、全体を統括する「リモートリポジトリ」、開発者ごとの「ローカルリポジトリ」で構成されています。まず開発者は、ローカルリポジトリに変更を記録し、確認したら、リモートリポジトリに変更をアップします。

●GitHub とは

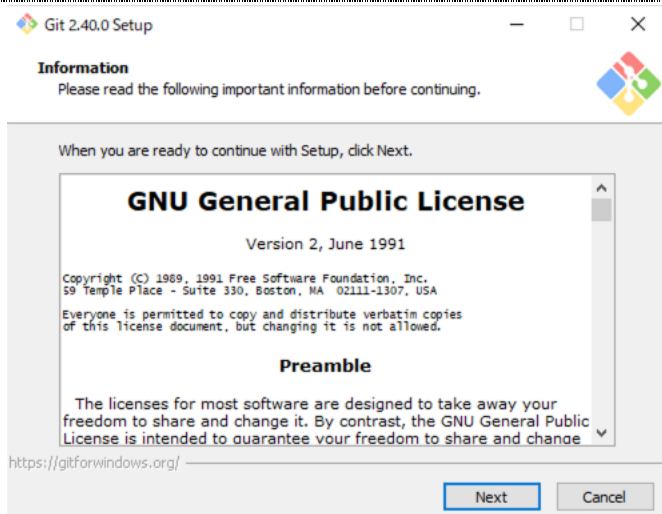
Git を使った開発者支援するサービスが GitHub です。クラウド上でバージョン管理するリモートリポジトリと考えてください。GitHub を使うことで、自分の手元にバックアップ用ファイルを置いておく必要がなくなり、クラウドで管理することができます。



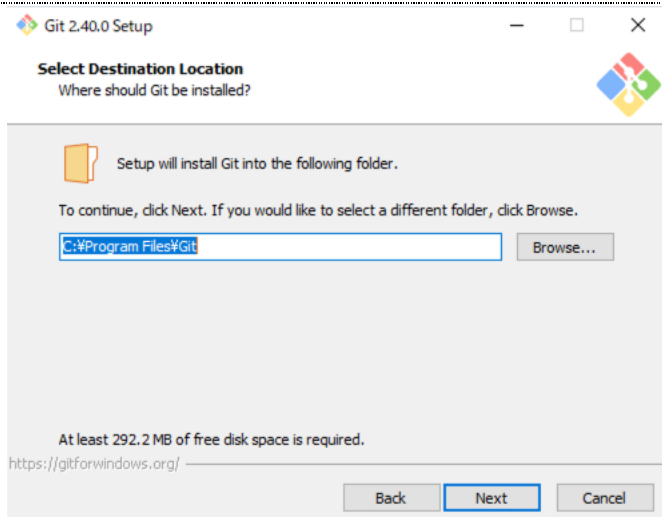
●Git のインストール

ローカルリポジトリを準備するためには、Git のインストールが必要です。

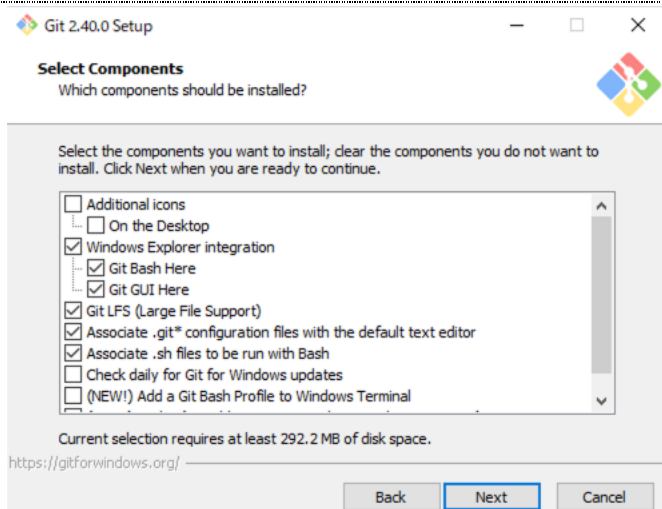
○インストール



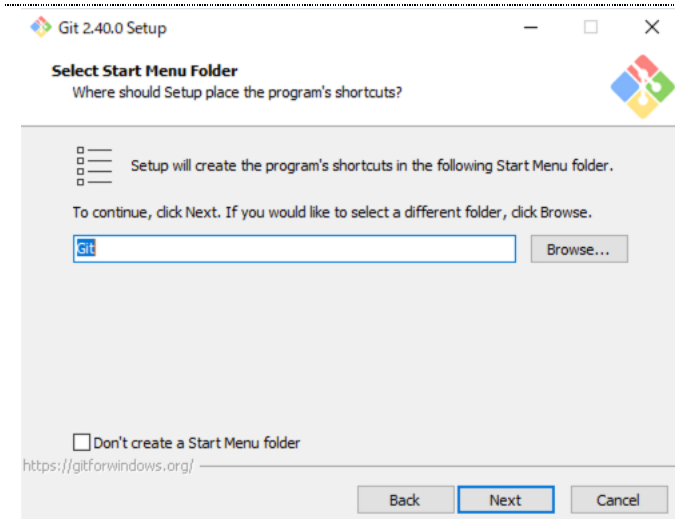
- ① 「Git-XXX-64-bit.exe」をダブルクリックします。
- ② information が開きます。
[Next]をクリックします。



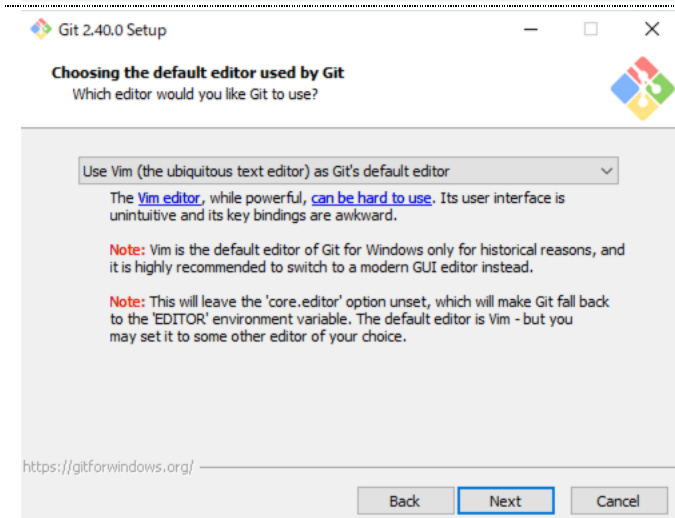
- ③ インストール先を指定します。ここでは、既定値のまま
[Next]をクリックします。



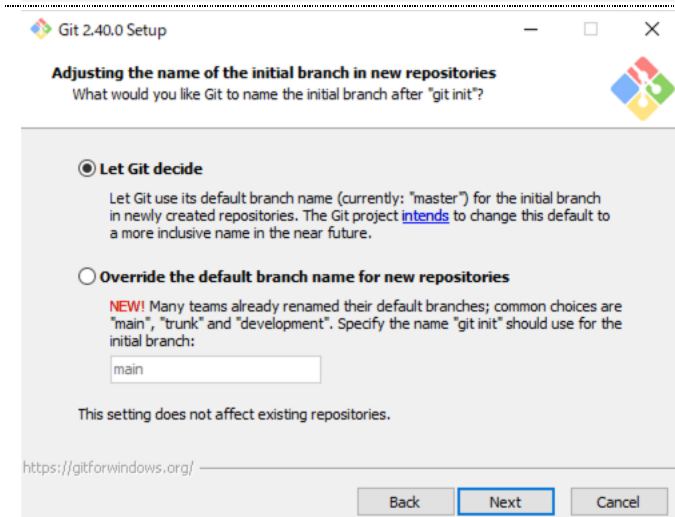
- ④ コンポーネントを選択します。既定値のまま[Next]をクリックします。



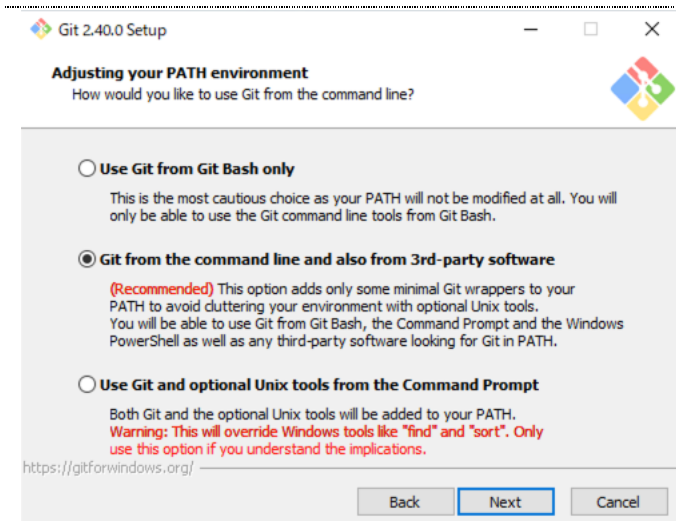
⑤ スタートメニューに登録します。既定値のまま[Next]をクリックします。



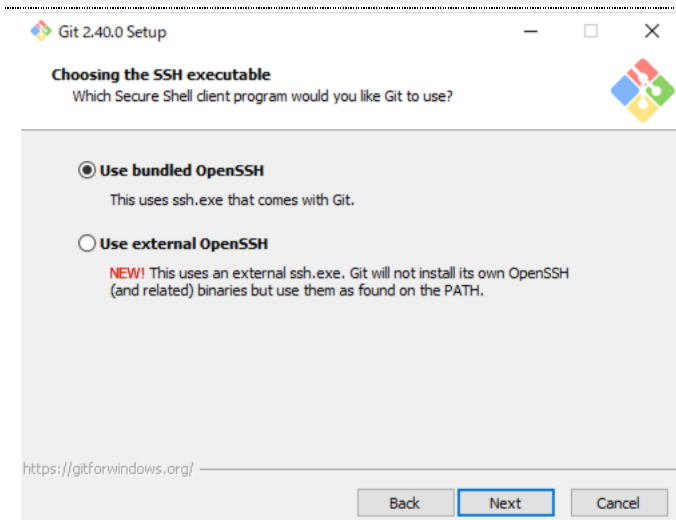
⑥ エディタを選択します。既定値のまま[Next]をクリックします。



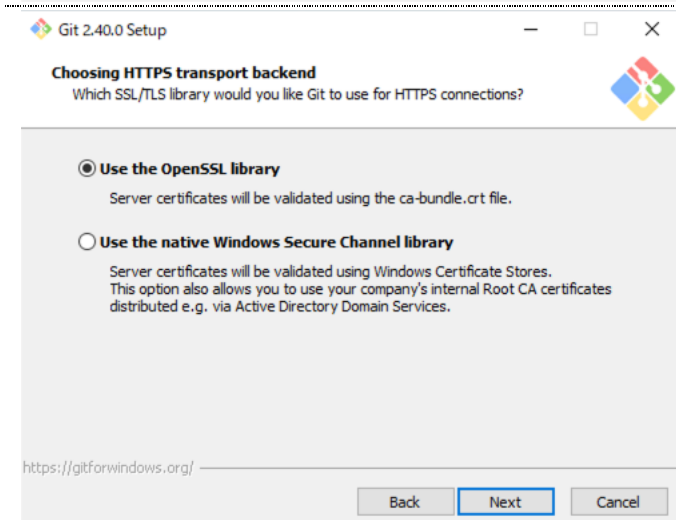
⑦ 新しいリポジトリの最初のブランチ名を設定します。既定値の「Let Git decide」を選択すると「master」というブランチ名が使用されます。ここでは既定値のまま[Next]をクリックします。



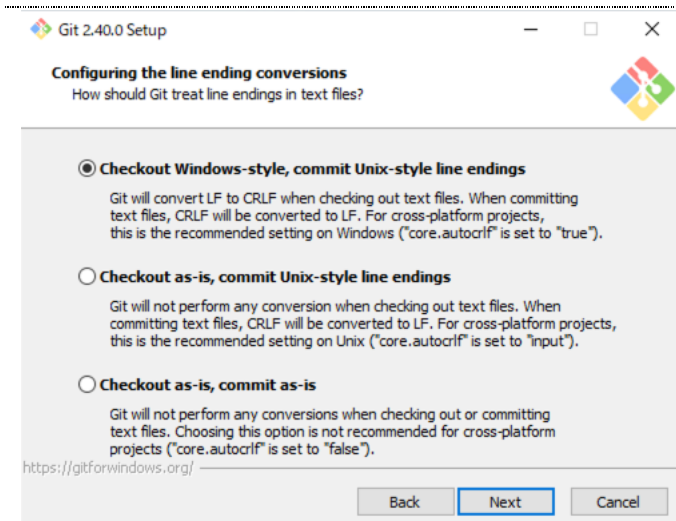
⑧ 環境変数の設定を行います。既定値のまま[Next]をクリックします。



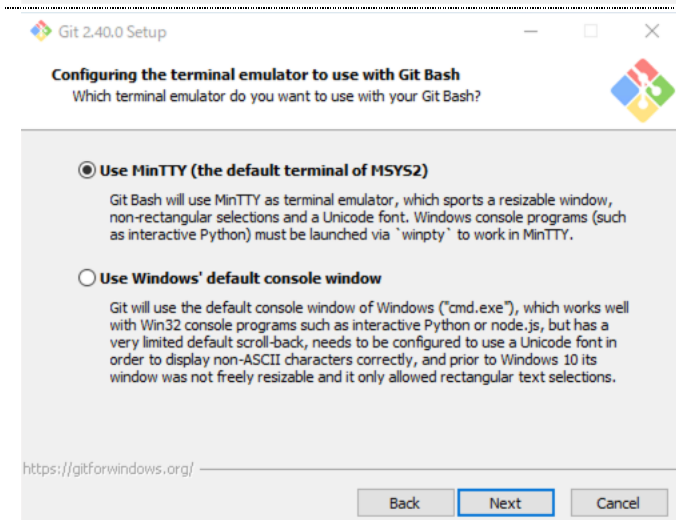
⑨ SSH を設定します。既定値のまま[Next]をクリックします。



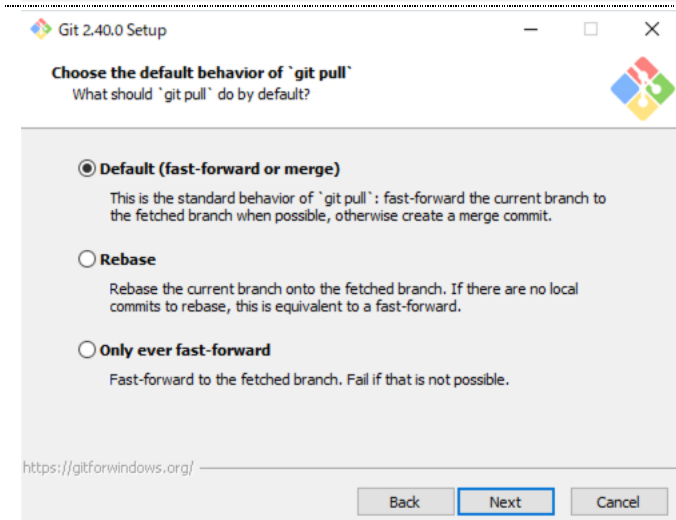
⑩ HTTPS を設定します。既定値のまま[Next]をクリックします。



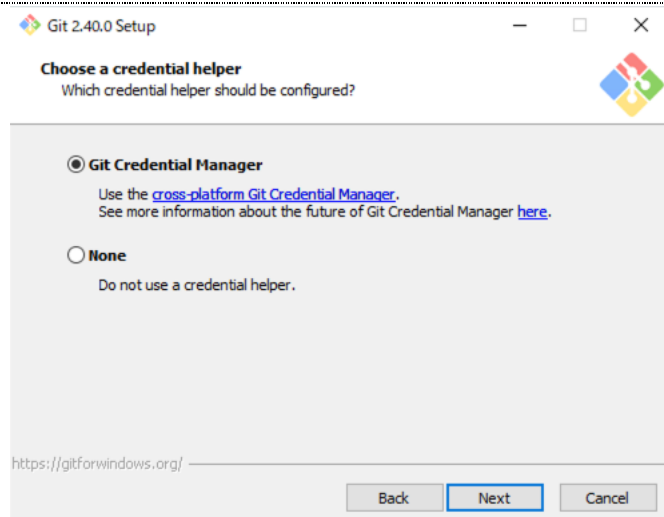
- ⑪ 改行コードを指定します。既定値のまま[Next]をクリックします。



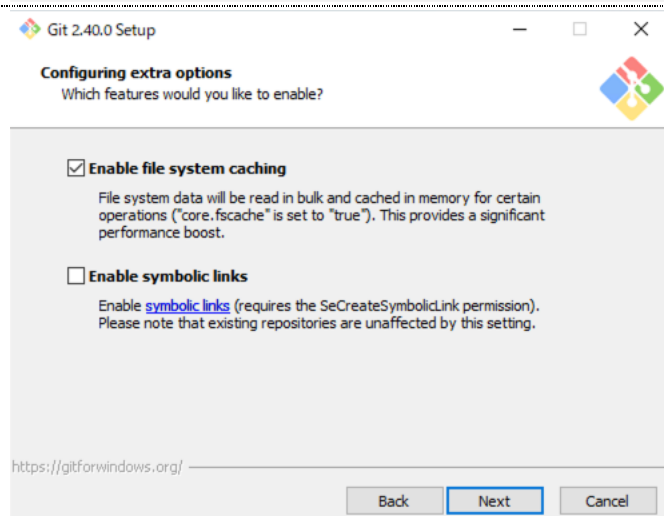
- ⑫ Git で使用するコンソールを選択します。既定値のまま[Next]をクリックします。



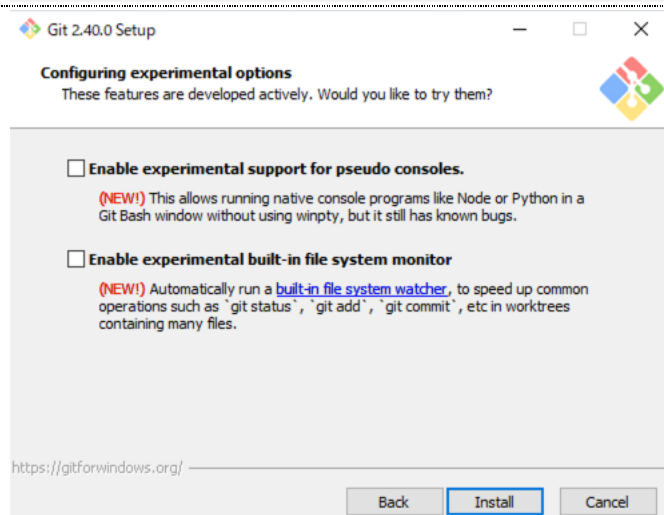
- ⑬ Git でプルをする際のデフォルト動作を設定します。既定値のまま[Next]をクリックします。



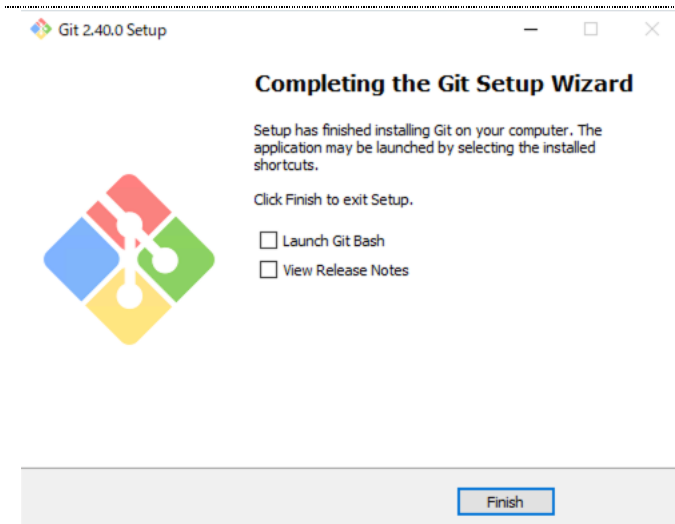
- ⑭ Git に接続する際に認証補助を使用するかどうかを選択します。既定値のまま[Next]をクリックします。



- ⑮ 追加オプションを指定します。既定値のまま[Next]をクリックします。



- ⑯ 試験的なオプションを指定します。既定値のまま[Next]をクリックします。



- ⑪ 「View Release Notes」のチェックを外します。[Finish]をクリックします。

●GitHub アカウントの作成



- ① <https://github.com/>にアクセスし、「サインアップ」をクリックします。

Join GitHub

First, let's create your user account

Username *

an2023043 ✓

Email address *

an20230430@outlook.jp ✓

Password *


***** ✓

Make sure it's at least 15 characters OR at least 8 characters including a number and a lowercase letter. [Learn more.](#)

Email preferences

☐ Send me occasional product updates, announcements, and offers.

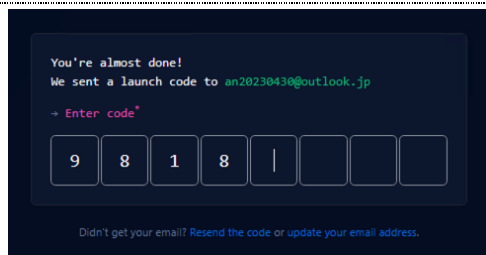
Verify your account



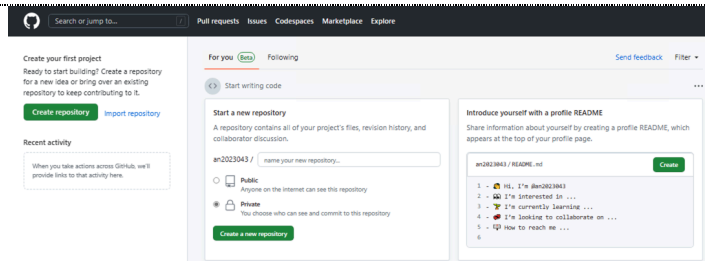
Create account

By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.

- ② Username など入力し、検証します。[Create account]をクリックします。

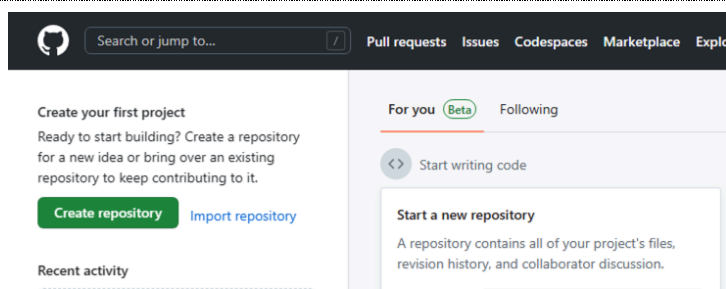


③ 電子メールにコードが届きます。そのコードを入力します。



④ Github が開きます。

●リモートリポジトリの作成



① 「Create repository」をクリックします。

Create a new repository

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

Owner * an2023043 / Repository name * test
 test is available.

Great repository names are short and memorable. Need inspiration? How about [psychic-octo-fiesta](#)?

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.](#)

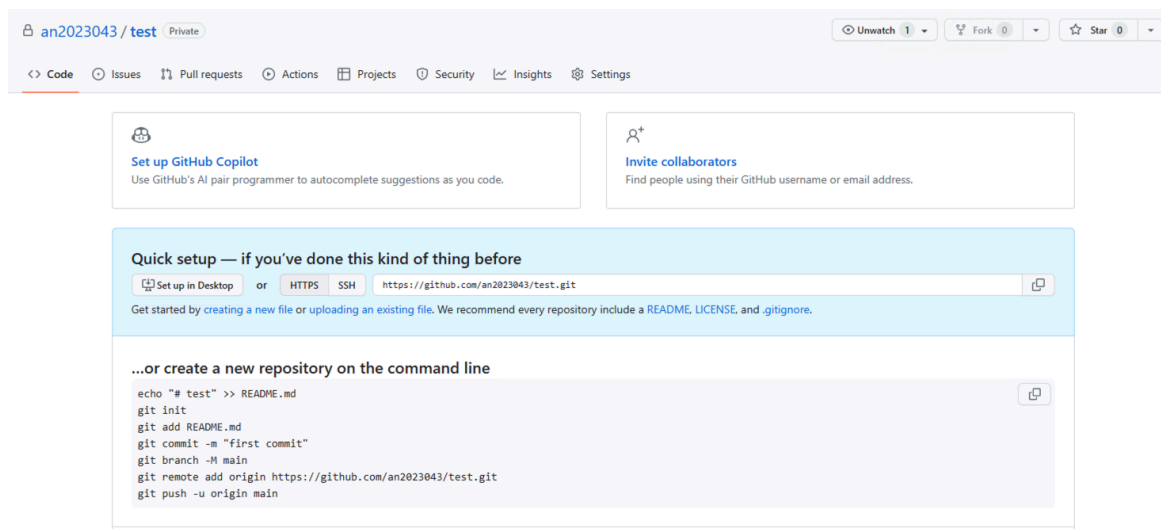
① You are creating a private repository in your personal account.

Create repository

② リポジトリ名を入力します。ソースコードを世界中に公開する場合は「Public」、公開したくない場合は「Private」を選択します。

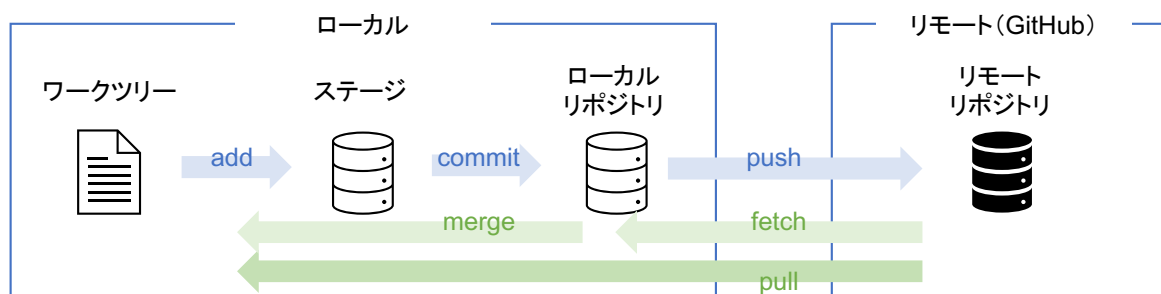
③ 画面をスクロールして、「Create repository」をクリックします。

リポジトリが完成すると、URL が発行されます。「<https://github.com/アカウント名/リポジトリ名.git>」このアドレスを記録しておいてください。



●Git を扱うときの基本的な流れ

Git を使うときの基本的な流れは、add して、commit して、push する。そして、リモートから持ってくるときには、pull する、という形です。



- ① git init コマンドで、Git で扱うフォルダを初期化する(最初だけで良い)
- ② git add . コマンドでローカルファイルをステージに追加する
- ③ git config コマンドでユーザー名と電子メールを追加する
- ④ git commit コマンドでローカルリポジトリにコミットする
- ⑤ git push origin main コマンドで、プッシュする。

(参考資料) Git

・【イラストで覚える】初心者のための Git と GitHub 用語集

<https://zukulog098r.com/git/>

・サル先生の Git 入門

<https://backlog.com/ja/git-tutorial/>

・マンガでわかる Git

https://www.r-staffing.co.jp/engineer/entry/20190621_1

●ローカルでの作業

- ① Git 用のフォルダを作成します。ここでは、デスクトップに git_training というフォルダを作成します。

ポイント この作業はフォルダにおいて1度だけです。

- ② git_training フォルダに shiritori.txt を作成します。
③ コマンドプロンプトを開き、git_training フォルダをカレントディレクトリに変更します。
④ git init コマンドで git を初期化します。

```
>git init
Initialized empty Git repository in C:/Users/azureuser/Desktop/git_training/.git/
```

- ⑤ git add . コマンドでディレクトリにある全てのファイルをステージに追加します。

```
>git add .
```

ポイント . は“全てのファイル”という意味です。

ポイント git add ファイルパス 1 ファイルパス 2 ファイルパス 3... で個別のファイルを追加できます。

- ⑥ ユーザー名とメールアドレスを登録します。こちらもフォルダで1度だけです。

```
>git config --global user.name "abe"
>git config --global user.email "an20230430@outlook.jp"
```

- ⑦ コミットします。コミットするときには、コメントを入れます。

```
>git commit -m "最初のファイル"
[master (root-commit) ad27845] 最初のファイル
1 file changed, 1 insertion(+)
create mode 100644 shiritori.txt
```

- ⑧ status コマンドで変更ファイルをチェックします。

```
>git status
On branch master
nothing to commit, working tree clean
```

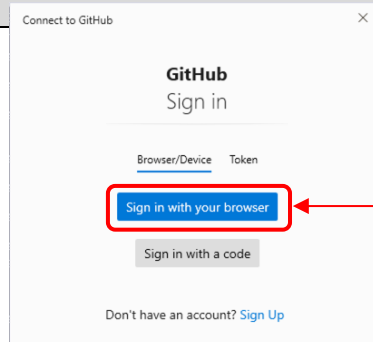
- ⑨ リモートリポジトリを登録します。

```
>git remote add origin https://github.com/an2023043/test.git
```

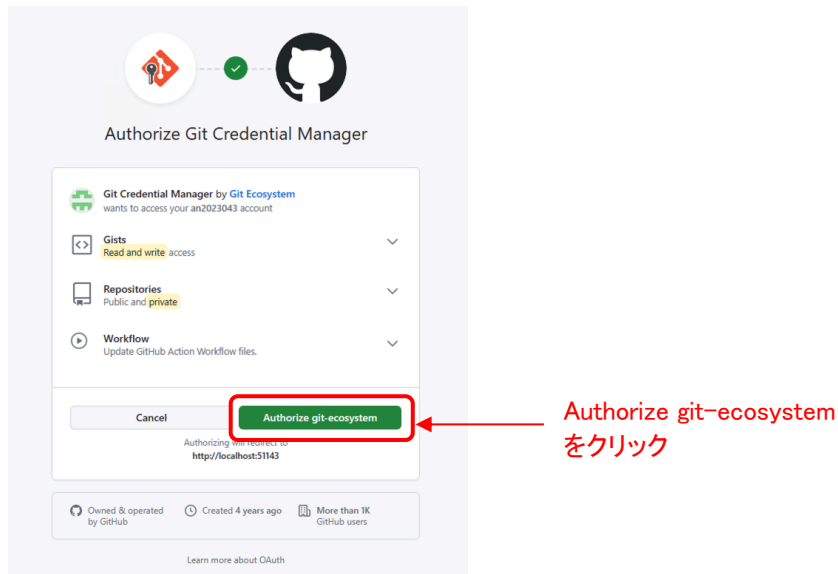
- ⑩ プッシュします。すると、認証画面が表示されます。

※master になっていますが、これ以降、main に置き換えてください。

```
>git push origin master
```



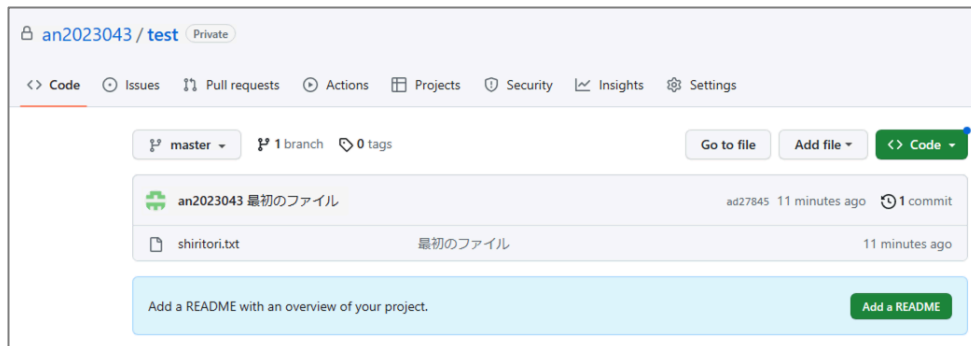
Sign in with your browser
をクリック



すると、プッシュされます。

```
C:\Users\azureuser\Desktop\git_training>git push origin master
info: please complete authentication in your browser...
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 244 bytes | 244.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/an2023043/test.git
 * [new branch]      master -> master
```

GitHub 画面を更新すると、プッシュされていることが分かります。



補足 認証情報が残っている可能性があるため、コントロールパネル／ユーザーアカウント／資格情報マネージャー／Windows 資格情報から過去の資格情報を消す

●ファイルを更新して、コミット／プッシュする - commit, push

ここで、しりとりファイルを更新して、コミット／プッシュを行います。まずは、しりとりファイルを更新して、git status を行ってみてください。すると、「Changes not staged for commit: (変更がステージングされていない)」と表示されます。これは、ステージングして(addして)コミットしてください、という意味です。

```
C:\Users\azureuser\Desktop\git_training>git status
On branch master
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:   shiritori.txt

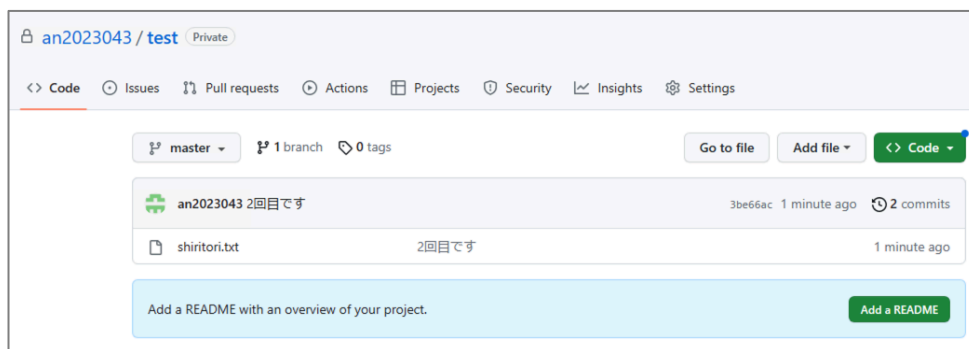
no changes added to commit (use "git add" and/or "git commit -a")
```

```
C:\Users\azureuser\Desktop\git_training>git add .

C:\Users\azureuser\Desktop\git_training>git commit -m "2 回目です"
[master 3be66ac] 2 回目です
1 file changed, 2 insertions(+), 1 deletion(-)
```

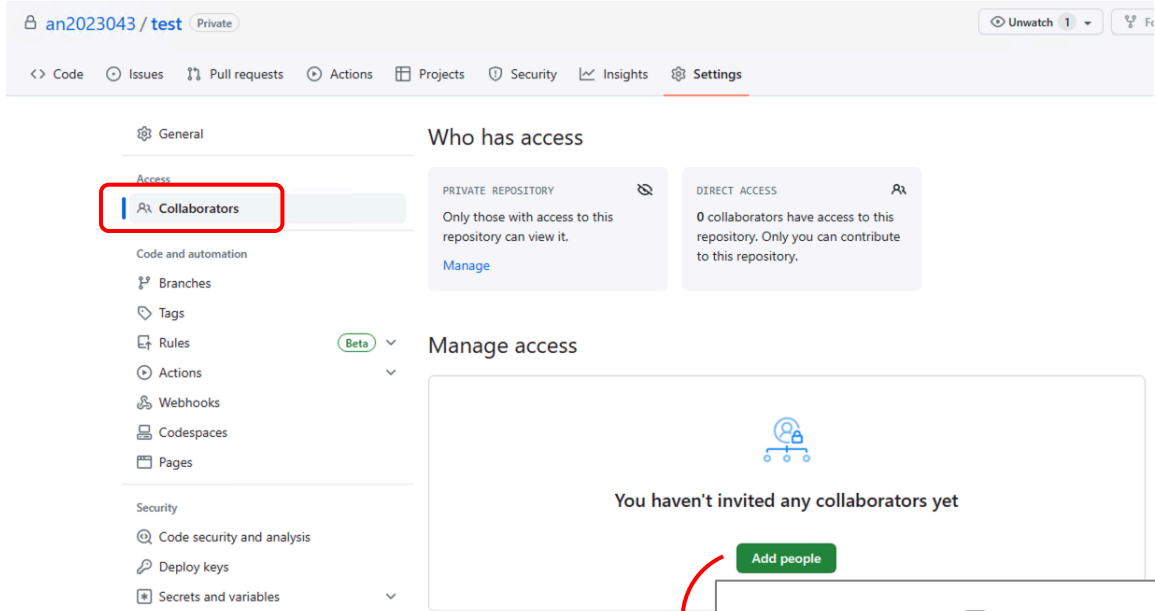
コミットができたので、プッシュします。2度目のプッシュでは認証は不要です。

```
C:\Users\azureuser\Desktop\git_training>git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 275 bytes | 275.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/an2023043/test.git
ad27845..3be66ac master -> master
```



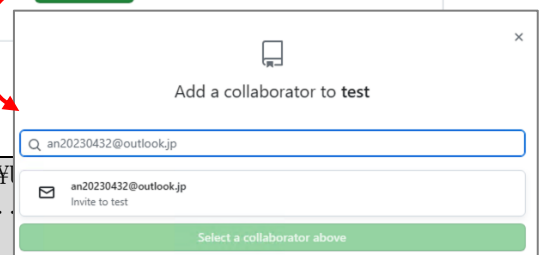
●別の人が作業を開始する、クローンする - clone

リモートリポジトリからファイルをコピーするやり方には、プルとクローンがありますが、作業の開始時など、ファイルを全てコピーするときには、「クローン」を使います。事前に Settings/Collaborators (共同編集者) を招待しておきます。



リポジトリ URL に続けて、作成先のディレクトリを指定します。

```
>git clone https://github.com/an2023043/test.git C:\Users\admin\Desktop\git_training
Cloning into 'C:\Users\admin\Desktop\git_training'...
remote: Enumerating objects: 6, done.
remote: Counting objects: 100% (6/6), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 6 (delta 0), reused 6 (delta 0), pack-reused 0
Receiving objects: 100% (6/6), done.
```



別の人がしりとりファイルを更新して、add、commit、push してみてください。

```
>git add .

>git commit -m "3 回目のコミット"
[master bc0e8c3] 3 回目のコミット
1 file changed, 2 insertions(+), 1 deletion(-)

>git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 297 bytes | 297.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/an2023043/test.git
3be66ac..bc0e8c3 master -> master
```

●最初の人の変更をプルする - pull

最初の人、別の人の変更を受け取りましょう。そのためには「プル」が必要です。

```
>git pull origin master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 277 bytes | 10.00 KiB/s, done.
From https://github.com/an2023043/test
 * branch          master      -> FETCH_HEAD
   3be66ac..bc0e8c3 master     -> origin/master
Updating 3be66ac..bc0e8c3
Fast-forward
 shiritori.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

●変更履歴を確認する - log

git log コマンドを使うと、変更履歴を確認できます。コミットするときのコメントが重要であることが分かります。
-p オプションを付けると、変更箇所も分かります。キーボードで q を押すと終了します。

```
>git log -p
commit bc0e8c325312b90f62faf9fe4001f09c7cf85cb0 (HEAD -> master, origin/master)
Author: abe2 <an20230432@outlook.jp>
Date:   Sun Apr 30 13:07:05 2023 +0900

    3 回目のコミット

diff --git a/shiritori.txt b/shiritori.txt
index 11192da..f040a26 100644
--- a/shiritori.txt
+++ b/shiritori.txt
@@ -1,2 +1,3 @@
   りんご
-ゴリラ
¥ No newline at end of file
+ゴリラ
+ライブ
¥ No newline at end of file

commit 3be66acbf07e153c073b154763e1c5ac89ad4e4f
Author: abe <an20230430@outlook.jp>
Date:   Sun Apr 30 12:19:43 2023 +0900

    2 回目です

diff --git a/shiritori.txt b/shiritori.txt
index 1f70ff7..11192da 100644
--- a/shiritori.txt
+++ b/shiritori.txt
@@ -1 +1,2 @@
:... skipping...
commit bc0e8c325312b90f62faf9fe4001f09c7cf85cb0 (HEAD -> master, origin/master)
Author: abe2 <an20230432@outlook.jp>
Date:   Sun Apr 30 13:07:05 2023 +0900

...中略
```

●コミットのバージョンに戻したい

git reset --hard [コミット ID]とすることで、コミット後、特定のバージョンに戻すことができます。

```
>git reset --hard 3be66acbf07e153c073b154763e1c5ac89ad4e4f
HEAD is now at 3be66ac 2 回目です
```

ここでは、2回目に戻ります。しかし、このままリモトリポジトリにプッシュするとエラーになります。

```
>git push origin master
To https://github.com/an2023043/test.git
! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'https://github.com/an2023043/test.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

このときは、-f オプションを付けて、強制的に反映させます。

```
>git push -f origin master
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/an2023043/test.git
+ bc0e8c3...3be66ac master -> master (forced update)
```

注意しないといけないのは、リモトリポジトリに反映させた後、別の人がプルする場合です。単にプルしても、バージョンを戻してくれないようです。ここでは、fetch というコマンドで、リモトリポジトリからローカルリポジトリに一旦持ってきて、ローカルの master に強制的に合わせます。

```
>git fetch origin master
From https://github.com/an2023043/test
* branch          master      -> FETCH_HEAD

>git reset --hard origin/master
HEAD is now at 3be66ac 2 回目です
```