



# Git 入門

## 環境構築編

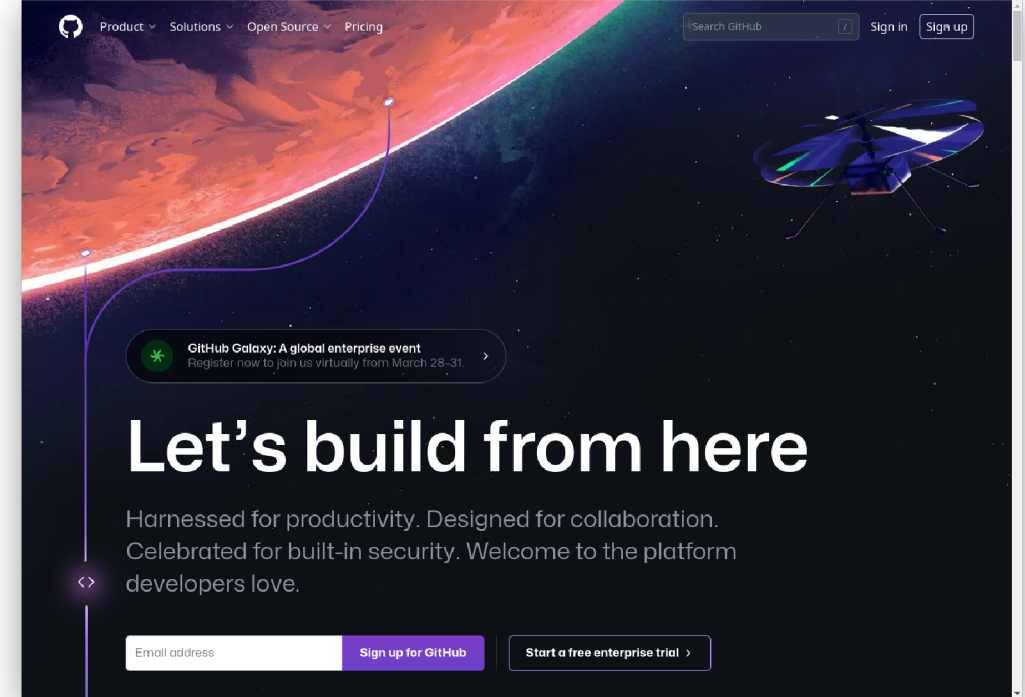
- 開発・作業で Git を使うための環境を構築するための説明
- **Windows での構築（含 WSL）がターゲット**
  - Macintosh なら Homebrew で Git を入れると良さそう
  - Linux なら一般的には適当なパッケージマネージャで入れるだけ
- そもそも Git を利用したホスティングサービスは複数存在
  - GitHub
  - GitLab
  - Bitbucket      etc ...
  - （余談）Onedrive や Google Drive にも構築できる
- 今回は **GitHub** の利用に絞って説明

# GitHubへの登録

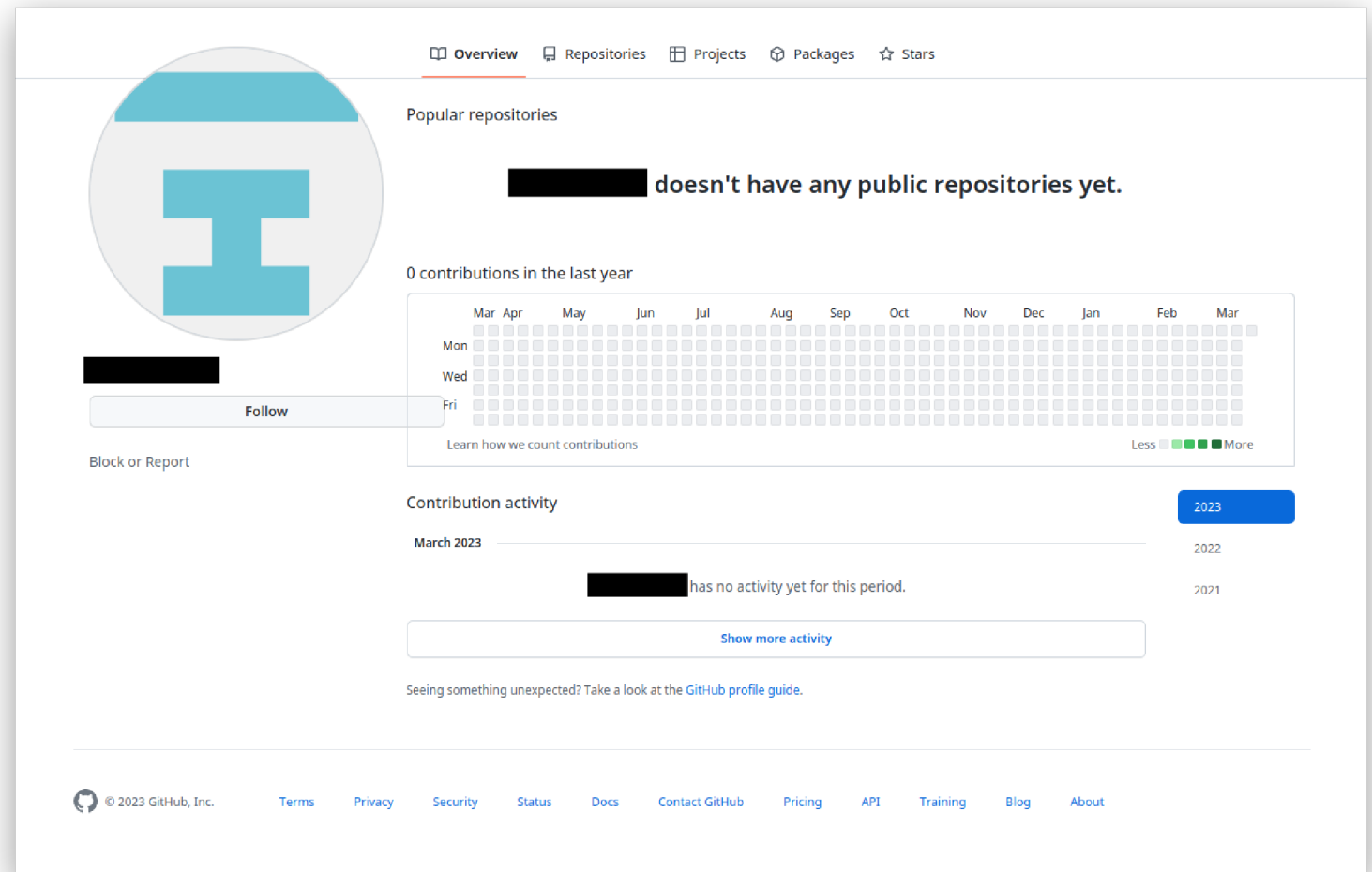
- ユーザ登録

- GitHub のホームページ  
(<https://github.com/>)から「**Sign Up**」をクリックして登録。

- 流れに沿って登録すれば良い
- 基本的に英語 only だが、そう難しくはない
- **username** はあとから変更もできるが、かなり面倒くさいことになり得るので、後悔しない名前をびしっと決めておくことを推奨
- 途中で料金プランについて聞かれるが、**free（無料）プラン**で十分
  - 今後 **pro（有料）プラン**を使いたくなったら、学生証の写真をアップロードすると無料になるのでお得かも



- 登録完了
  - 伝説はここから始まる



# Git を使える環境の導入

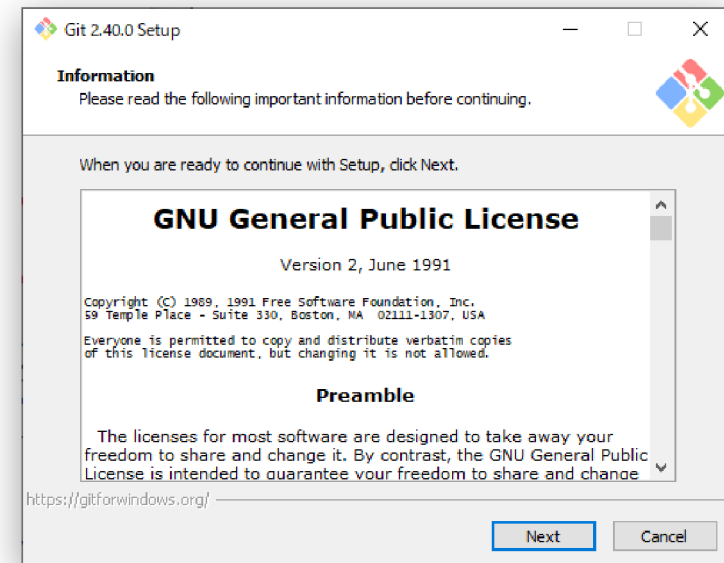
- WindowsでのGitの導入を紹介
    - 今回は
      1. Windows ネイティブに入れる
      2. Windows Subsystem for Linux (WSL) に入れる
- 以上 2 パターンの両方について扱う

- 「**Git for Windows**」を紹介
  - CUI & GUI で Git を扱える
  - 公式サイト (<https://gitforwindows.org>) からダウンロードしてインストール
  - 次のページからインストール手順の一例を紹介
    - あくまで一例
    - 今回はスライド作成時点で最新のバージョン **v2.40.0** を使用
    - バージョンごとに細かな箇所が変わるので注意



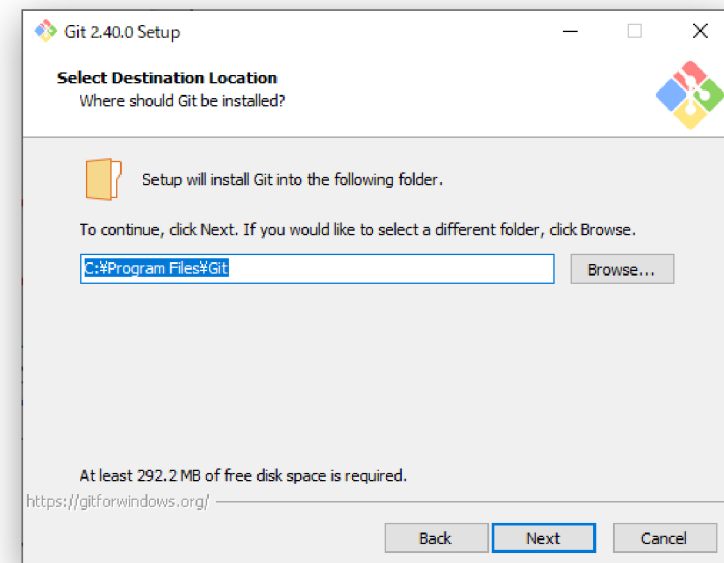


## 1. 規約に同意する



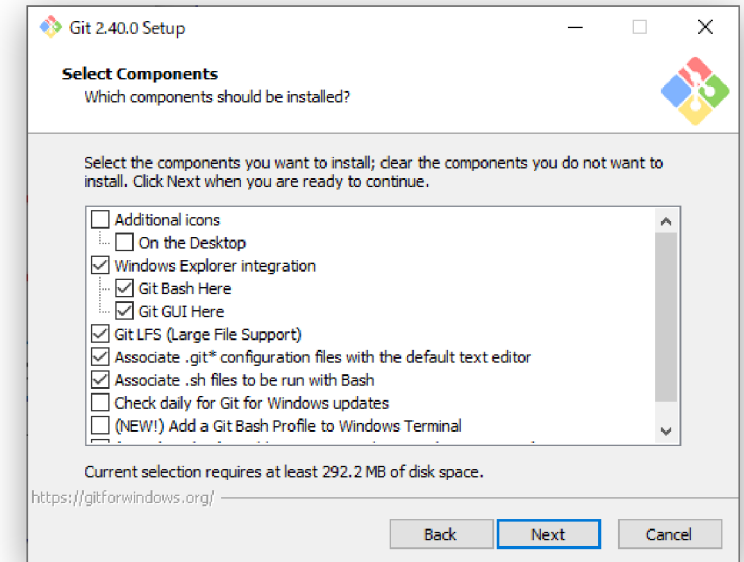
## 2. インストール先の選択

- デフォルトのままでOK



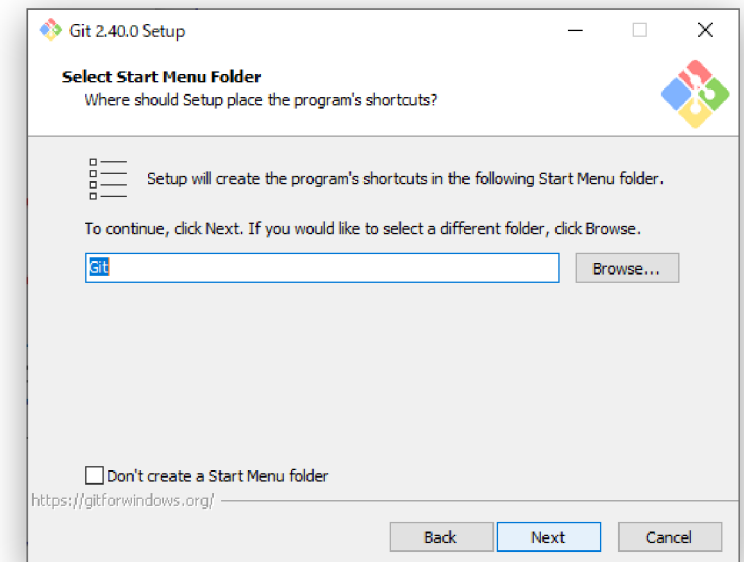
## 3. コンポーネントの選択

- ■ 「Git Bash Here」
  - 「Git GUI Here」
- にチェックを入れることを推奨



## 4. スタートメニューの設定

- デフォルトのままでOK

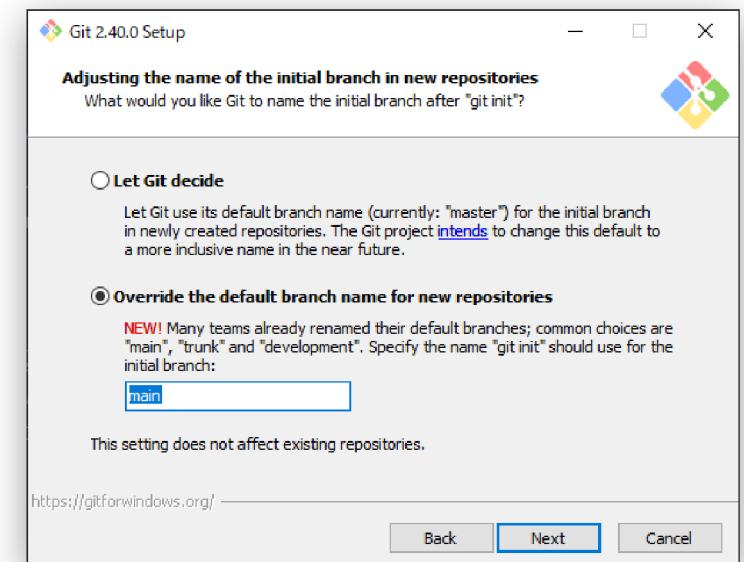
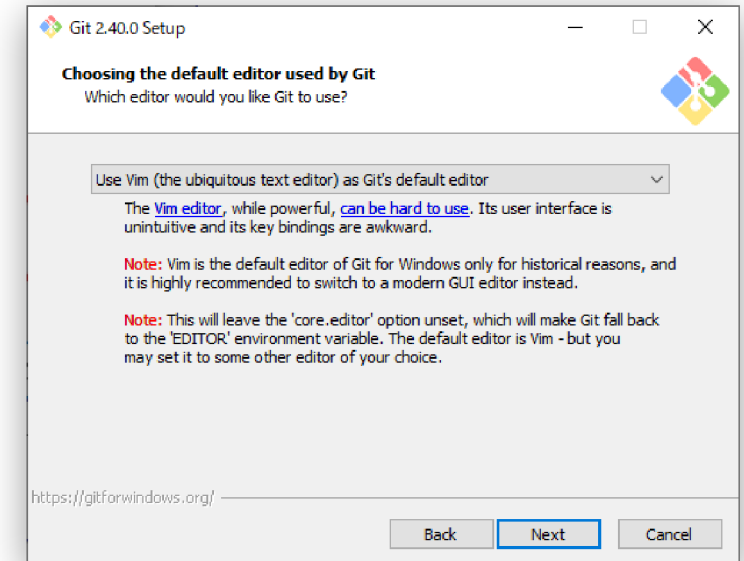


## 5. テキストエディタの選択

- コミット時などに使うエディタの選択
  - 「Vim」 または 「Visual Studio Code」を推奨

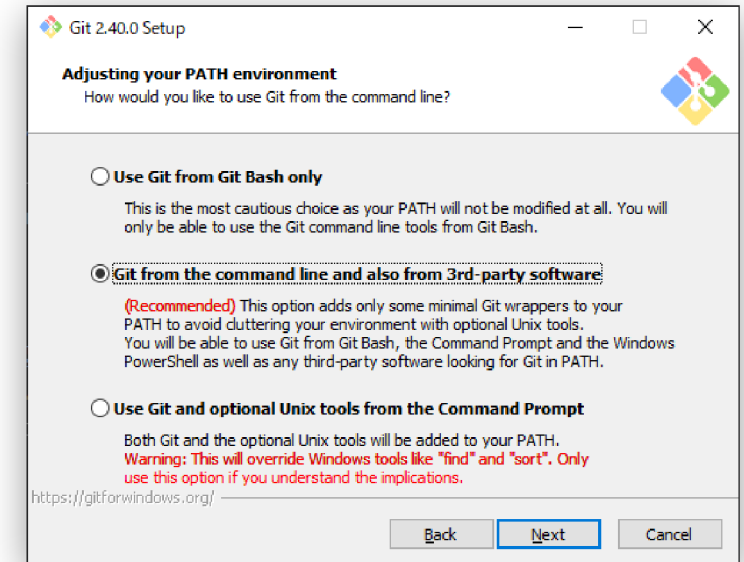
## 6. 新規リポジトリの既定ブランチ名の設定

- GitHubの流儀に従うことを考えて、「Override」にチェックし、「main」と入力



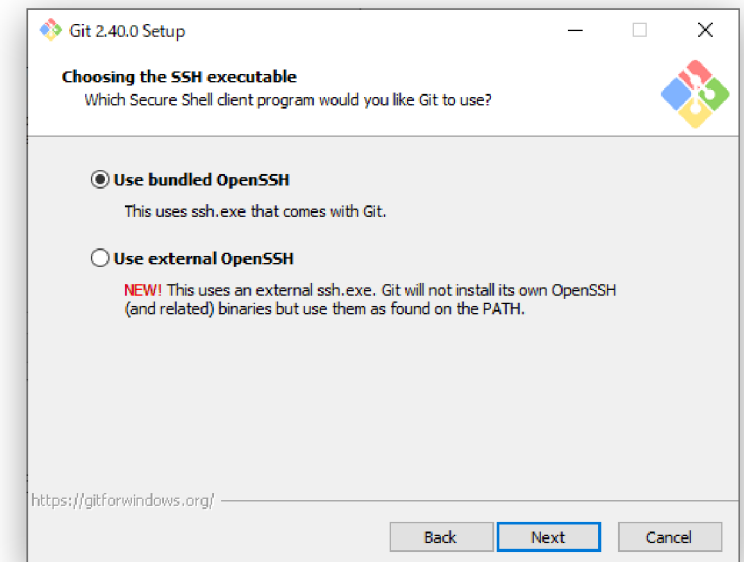
## 7. Gitの環境変数の設定

- デフォルトの「……also from 3rd-party software」にチェック



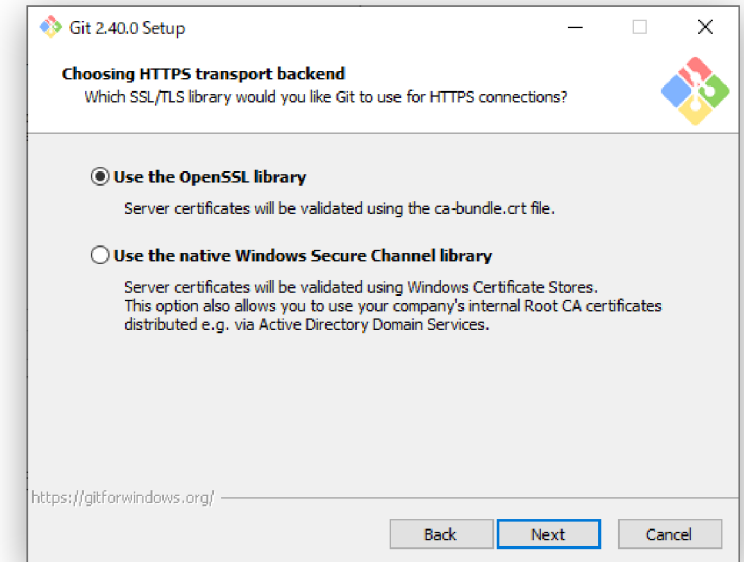
## 8. 使用するSSHバイナリの選択

- デフォルトの「Use bundled OpenSSH」にチェック



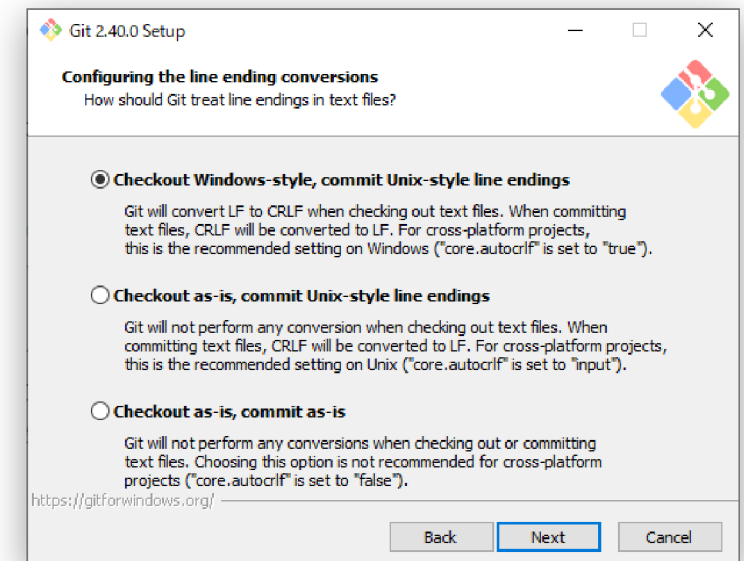
## 9. HTTPS接続時のライブラリ選択

- デフォルトの「Use the OpenSSL library」にチェック



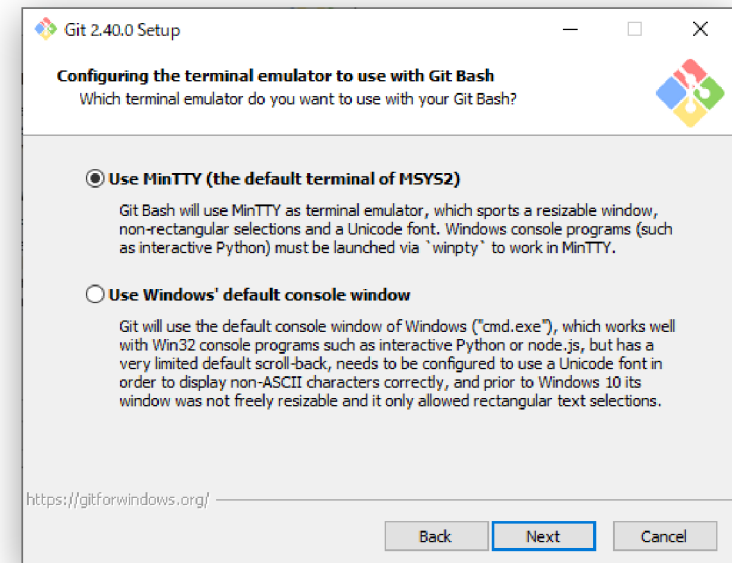
## 10. 改行コードの扱いの設定

- デフォルトの「Checkout Windows style,.....」にチェック
  - CRLF を使う Windows くん ~~に忖度ほんま~~  
~~ks~~



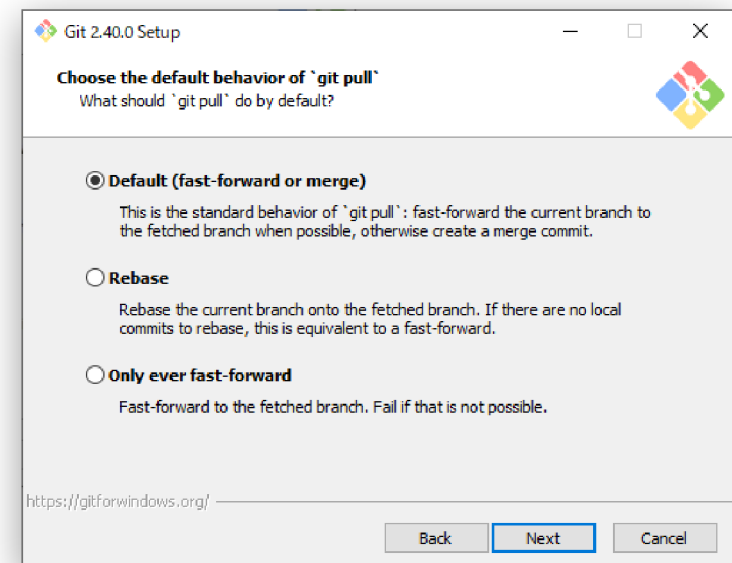
## 11. 使用するターミナルの設定

- デフォルトの「Use MinTTY」にチェック



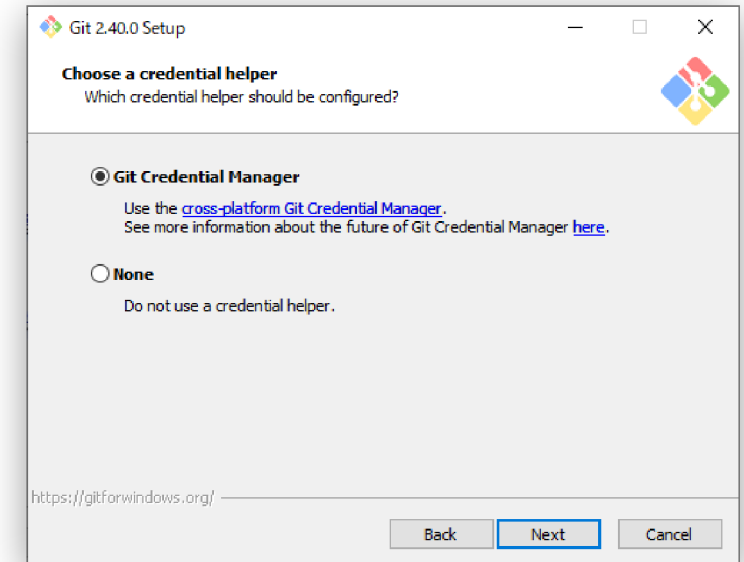
## 12. git pull の挙動の設定

- デフォルトの「Default」にチェック



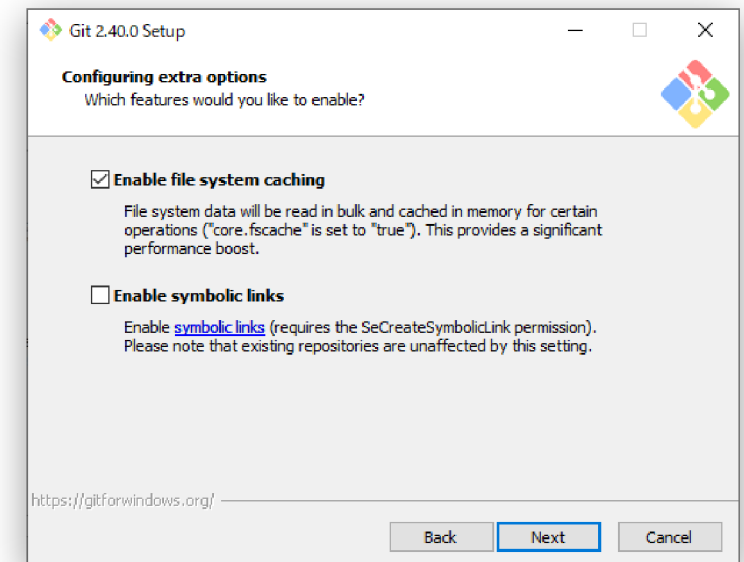
## 13. GCMの設定

- デフォルトの「Use Credential Manager」にチェック



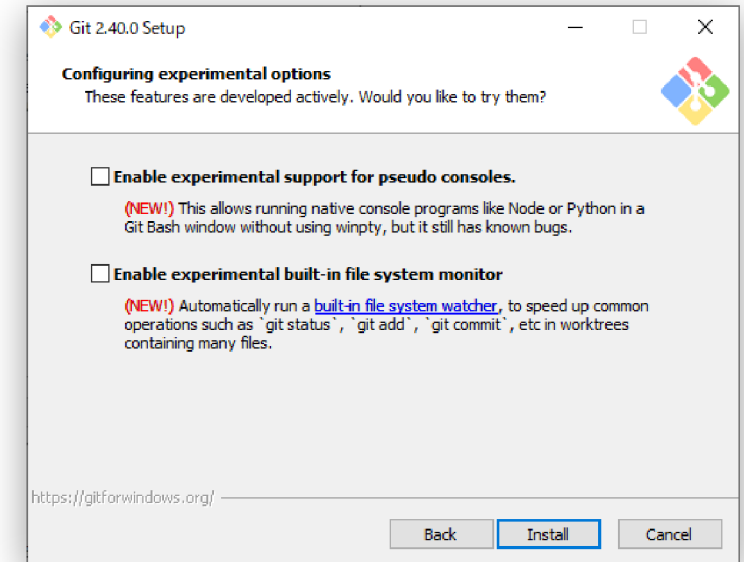
## 14. その他の設定

- デフォルトのまま「Enable file system caching」のみにチェック



## 15. 実験的機能の設定

- デフォルトのまま特にチェックしなくて良い



## 16. Done!



- WSL 上の Ubuntu に Git を導入
  - WSL の導入から説明
    - やり方がわかる人やすでに WSL を導入済みの人はスキップすれば良い
  - もちろんこだわりがあるのなら Ubuntu でなくても良い
  - ほぼコマンドライン上の操作のみで完結
    - 「Git for Win」に比べればむしろ簡潔かも（激ウマギャグ）
  - 以降 Windows 10 のバージョン 21H2 以降、または Windows 11 を対象
    - それ以前でもできる場合があるが手順が異なる（g g r）
    - バージョンは「スタートボタンを右クリック→『システム』」で確認可能



- WSL をインストール
  1. PowerShellを管理者として実行
    - スタートボタンの右クリックメニューから呼び出せる
  2. 下記コマンドを実行

```
wsl --install
```

3. システムの再起動を要求されるので従う

#### 4. Ubuntu のターミナルに **username** と **password** を入力

- 好きに決めて良い
- Windows のものと一致させる必要はない

#### 5. プリインストールされたパッケージを更新

- 下記コマンドを順に実行
  - **update** で更新があるか確認
  - **upgrade** で実際に更新

```
```bash
sudo apt update
sudo apt upgrade
```
```

- WSL をインストール (続き)

6. Git はプリインストールされているはずなので確認

```
git --version
```

`git version 2.34.1` のようにバージョンが出力されればOK

7. Ubuntu 上のファイルは、Windows のエクスプローラのアドレスに  
`\\wsl$` と入力することで参照できる

- WSL をインストール (続き)

8. (オプション) 「build-essential」 パッケージを入れておくの良い
  - gcc や make などパッケージの構築に必須級のパッケージがまとめて入る

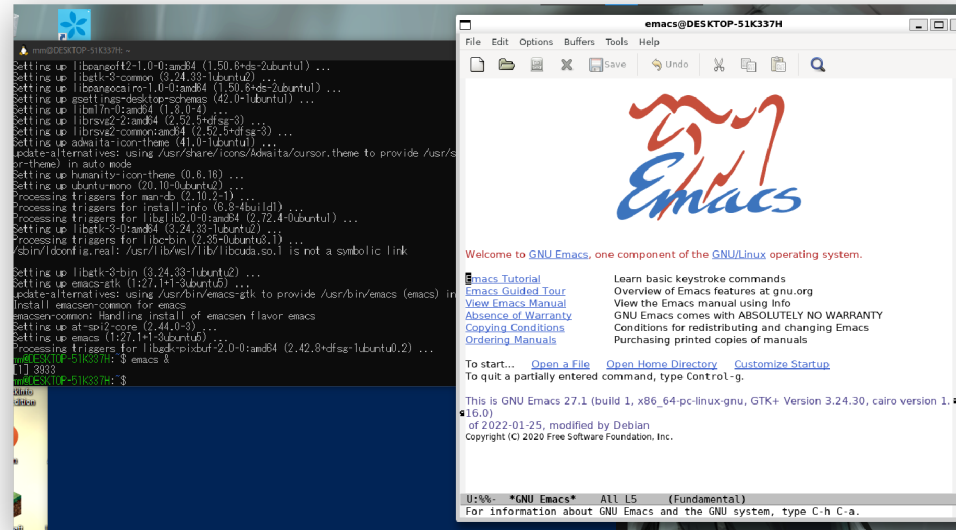
```
sudo apt install build-essential
```

9. (オプション) 義務

```
sudo apt install emacs
```

- WSL をインストール (続き)

## 10. (余談) この方法でインストールした WSL (ストア版) は最初から GUI が使える



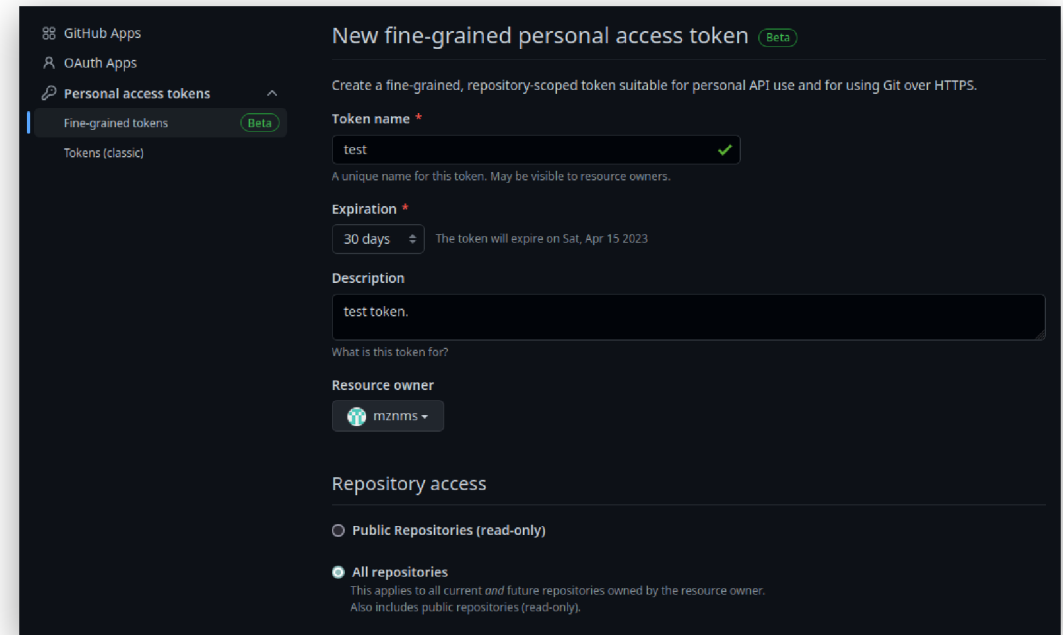
- 以前のバージョンは自分でXサーバをインストールする必要があったんじゃない (語り部の老人)


# Git の設定

- GitHub へのアクセス権の設定
  - 手段は2つ
    - アクセストークンの利用
    - SSH 接続の利用
    - （過去にはアカウントのパスワードでも利用できたが2021年の8月に廃止）



1. GitHubのページから自分のアカウントの「Setting」へ移動
2. 「Developer settings」へ移動
3. 「Personal access tokens」の「Fine-grained personal tokens」へ移動
4. 「Generate new token」をクリック
5. 「Token name」と「Expiration」（有効期限）を適当に設定
6. 「Repository access」は「All repositories」にチェック



7. 「Permissions」内の「Contents」のアクセス権を「Read and write」に変更
  8. ページ下部の「Generate token」をクリック
  9. 英数字記号からなるトークンが表示される
-  注意
    - アクセストークンが表示されたページを閉じると、**二度と同じものを表示することはできない**
    - トークンのアクセス権の範囲はあとからでも変更可能
    - トークンは push 時などに（原則）毎回尋ねられることに注意

1. Windows ネイティブ環境の場合は Git Bash を、WSL 環境の場合はそのターミナルを開く
2. SSH鍵を作成する。以下のコマンドを実行

```
ssh-keygen -t ed25519 -C "GitHub"
```

- `-C` は SSH 鍵につけるコメントの指定
  - 無指定だと「**username@hostname**」になる
- `-t` は 使用する暗号化方式の指定
  - 楕円曲線暗号である「ed25519」の使用を推奨
  - 巷のチュートリアルでは（デフォルトの）「RSA 3072」や（デフォルトだった）「RSA 2048」がまだ多いかな？

### 3. 対話に答えながら鍵を作成

- 鍵の保存場所、パスフレーズの入力、パスフレーズの再入力、という順番で入力を求められるが、すべてデフォルトで良い (Enter 連打)

```
Generating public/private ed25519 key pair.  
Enter file in which to save the key (/home/( )/.ssh/id_ed25519):  
Enter passphrase (empty for no passphrase):  
Enter same passphrase again:  
Your identification has been saved in id_ed25519  
Your public key has been saved in id_ed25519.pub  
The key fingerprint is:  
SHA256:e9rsKl( ) ( )  
The key's randomart image is:  
+--[ED25519 256]--+  
| .. ( ) .. |  
+-----[SHA256]-----+
```

## 4. 鍵の生成を確認

- `~/.ssh/` に `id_ed25519` と `id_ed25519.pub` の2ファイルが生成されていることを確認
- `id_ed25519` は秘密鍵。絶対に他人に漏らさない
- `id_ed25519.pub` は公開鍵。これを GitHub に登録して認証する

## 5. 公開鍵を準備

- 以下のコマンドを実行

```
cat ~/.ssh/id_ed25519.pub | clip.exe
```

- 公開鍵の内容（文字列）がクリップボードに転送される  
("Ctrl-V" で貼り付けられる状態)

## 6. 公開鍵を登録

- GitHub のページの右上にある自分のアイコンをクリック→「Settings」へ移動
- 「SSH and GPG keys」へ移動
- 緑色の「New SSH key」をクリック

## 6. 公開鍵を登録（続き）

- 「Title」には適当な文字列を入力
  - SSH 鍵を生成した PC を識別できる名前がおすすめ
- 「Key type」は Authentication Key
- 「Key」には先程クリップボードに仕込んだ公開鍵の内容を貼り付ける
- 「Add SSH key」をクリック

SSH keys / Add new

Title

id\_ed25519\_github

Key type

Authentication Key

Key

ssh-ed25519 AAAAC3NzaC1lZDI1 exam@example.jp

Add SSH key



## 7. ローカル環境のGitHub アカウントとの紐付け

- 以下の2つのコマンドを実行

```
git config --global user.name "GitHub username"  
git config --global user.email "GitHub"
```

- この内容は `~/.gitconfig` に保存される

## 8. 疎通確認

- 以下のコマンドを実行

```
ssh -T git@github.com
```

- もし以下のような警告が出たら（fingerprint を確かめて<sup>[1]</sup>） yes

```
The authenticity of host 'github.com (IP ADDRESS)' can't be established.  
RSA key fingerprint is SHA256:nThbg6kXUpJWGl7E1IGOCspRomTxdCARLviKw6E5SY8.  
Are you sure you want to continue connecting (yes/no)?
```

---

[1] ここで最新の fingerprint を確認できる: <https://docs.github.com/ja/authentication/keeping-your-account-and-data-secure/githubs-ssh-key-fingerprints>

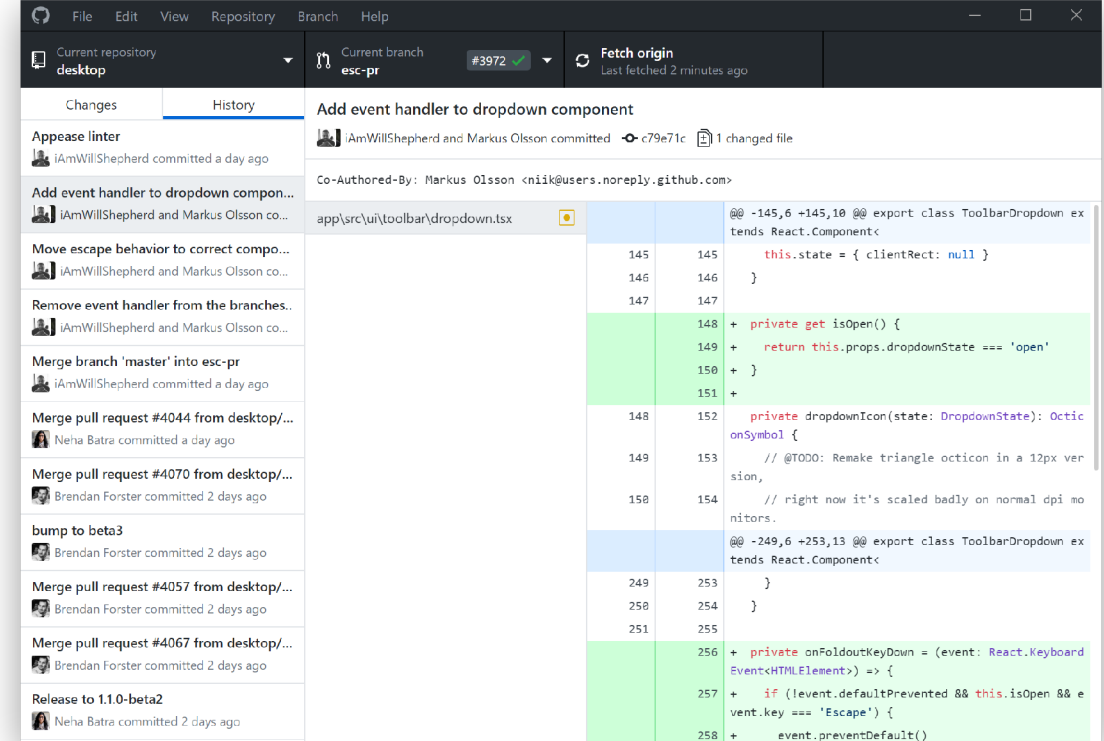
## 9. 成功メッセージが帰ってきたら成功

```
Hi (GitHub username)! You've successfully authenticated, but GitHub does not  
provide shell access.
```

# Git の周辺ツール

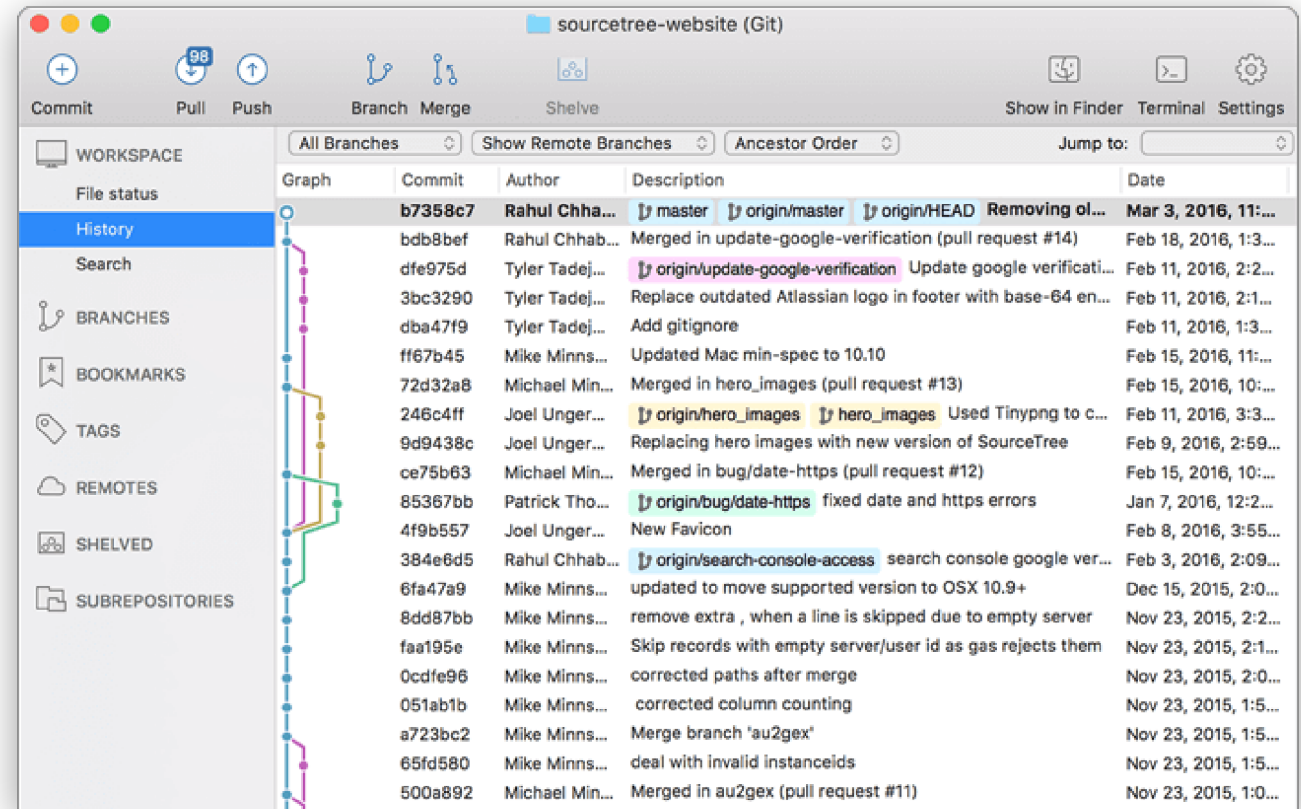
- その1 「GitHub Desktop」

- Windows と Macintosh で使用可能
- GitHub 公式謹製
- とってもグラフィカル、ほぼマウスで完結
- GitHub の独自機能の部分も手厚い対応
- <https://desktop.github.com>



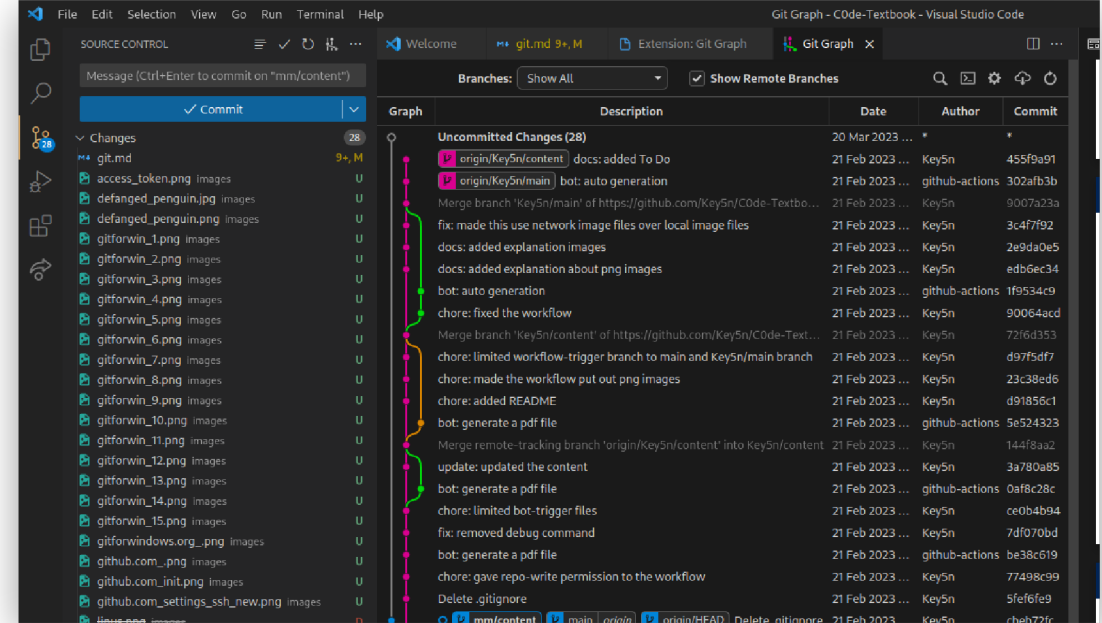
- その2 「Sourcetree」

- マルチプラットフォーム対応 (Linux も)
- GitHub に限らず対応
- 日本語化も可能
- Git GUI 操作の定番
- <https://www.sourcetreeapp.com>



## ● その3 「VSCode」

- エディタ自体は説明不要
- コミットやブランチ操作など基本的なことは素でできる
- 拡張機能でさらに強化可能
  - 「Git Lens」 - 差分の表示やコミット履歴の確認などが強化されるユーティリティツール
  - 「Git Graph」 - コミット履歴を見やすいツリー状に表示する拡張機能
  - 「Git History」 - コミットメッセージの検索やファイル単位・行単位の履歴確認を強化する拡張機能



## ● その4 「Magit」

- Emacs のパッケージ（拡張機能）
- 最初は扱いにくい（断言）
- 慣れるとあらゆる操作がコマンドラインを凌駕する速度になる
- 「ファイルの一部のみ unstage」や「コンフリクト解消」のような込み入った操作もすべてキーボードで完結

The screenshot displays the Magit interface within Emacs, showing commit history and file diff.

**Commit History (Left Panel):**

- Commit: 83ae60c
- Author: mznms <mznms@disroot.org>
- AuthorDate: Mon Mar 13 01:49:13 2023 +0900
- Commit: mznms <mznms@disroot.org>
- CommitDate: Mon Mar 13 01:49:13 2023 +0900
- Parent: 4979746 Update init.el
- Contained: main

**Update init.el**

```

1 file changed, 6 insertions(+), 3 deletions(-)
.emacs.d/init.el | 9 +++++-
modified .emacs.d/init.el
@@ -238,7 +238,7 @@
 (leaf font-for-gui
  (defvar "Use Word & Adjust font size"
    - if (window-system)
    + if (display-graphic-p)
    :defvar jp-font-family
    :preface
  @@ -392,7 +392,9 @@
    (bg-region
      (bg-completion "#5a5a5a")
      (fg-line-number-active "#8080ff")
      - (bg-line-number-active unspecified))
      + (bg-line-number-active unspecified)
      + (bg-line-number-inactive unspecified)
      + (fringe unspecified))
  
```

**Limit arguments**

- Limit to files (--)
- I Ignore submodules (--ignore-submodules)
- B Ignore whitespace changes (--ignore-space-change)
- W Ignore all whitespace (--ignore-all-space)

**Context arguments**

- U Context lines (-U)
- W Show surrounding functions (--function-context)

**Tune arguments**

- A Diff algorithm (--diff-algorithm)
- R Detect renames (-R)
- C Detect copies (-C)
- x Disallow external diff drivers (--no-ext-diff)
- s Show stats (--stat)
- g Show signature (--show-signature)

**Actions**

- o Own
- r Diff range
- p Diff paths
- u Diff unstaged
- s Diff staged
- w Diff worktree
- c Show commit
- t Show stash

**Commit History (Right Panel):**

| Commit  | in main             | Update init.el             | mznms | 3 days   |
|---------|---------------------|----------------------------|-------|----------|
| db122b1 | * main              | origin/main Update init.el | mznms | 3 days   |
| 7b1b1b4 | * Update init.el    | Update init.el             | mznms | 4 days   |
| 83ae60c | * Update init.el    | Update init.el             | mznms | 1 week   |
| 4979746 | * Update init.el    | Update init.el             | mznms | 1 week   |
| a122b51 | * Update init.el    | Update init.el             | mznms | 1 week   |
| 7a251b4 | * Update init.el    | Update init.el             | mznms | 1 week   |
| e4df335 | * Update init.el    | Update init.el             | mznms | 2 weeks  |
| 0a26d12 | * Update init.el    | Update init.el             | mznms | 2 weeks  |
| 074820b | * Update init.el    | Update init.el             | mznms | 3 months |
| 89d415b | * Update init.el    | Update init.el             | mznms | 6 months |
| 5dd36de | * Update init.el    | Update init.el             | mznms | 6 months |
| 10e7c44 | * Update init.el    | Update init.el             | mznms | 6 months |
| f0b1e1b | * Add early-init.el | Add early-init.el          | mznms | 6 months |
| 53aa90e | * Initial commit    | Initial commit             | mznms | 6 months |
| 2b71a68 | * Initial commit    | Initial commit             | mznms | 6 months |