追加アイコン　カバーの追加

Git & GitHub ガイド

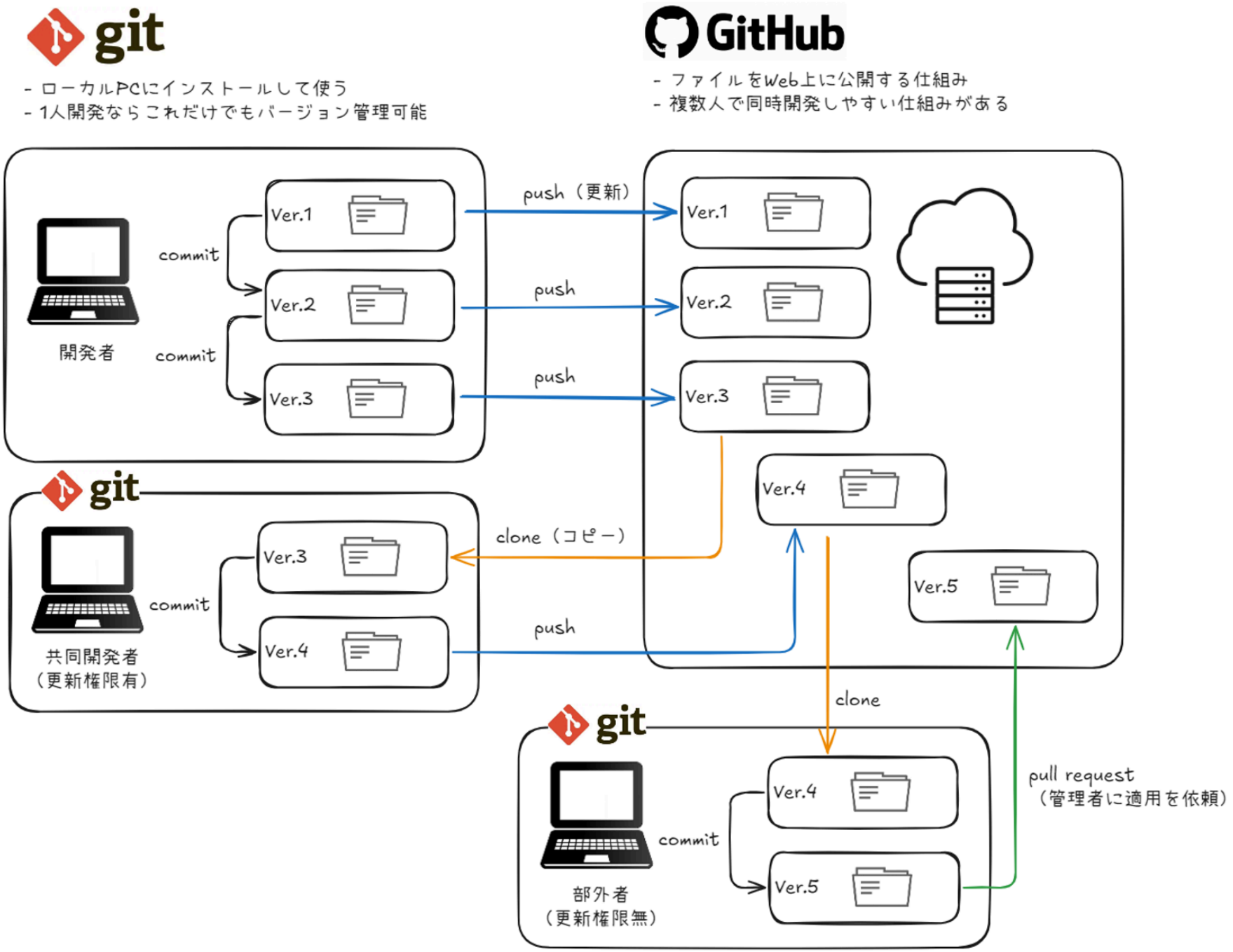
対象: GitとGitHubを初めて使う人。VS CodeのGUIを中心に、どんな場面で使うか、どんな利点があるかを理解しながら学びます。

Git, GitHubとは？

📌 「Git」と「GitHub」は全くの別物

Git = ファイルのバージョン管理アプリ

Github = バージョン付きでテキストファイルをWeb上にアップロードして共有するGit関連サービス


git
- ローカルPCにインストールして使う
- 1人開発ならこれだけでもバージョン管理可能
GitHub
- ファイルをWeb上に公開する仕組み
- 複数人で同時開発しやすい仕組みがある
Ver.1
Ver.2
Ver.3
commit
commit
開発者
push（更新）
push
push
Ver.1
Ver.2
Ver.3
Ver.4
Ver.5
git
clone（コピー）
Ver.3
Ver.4
commit
共同開発者
（更新権限有）
push
clone
git
Ver.4
Ver.5
commit
部外者
（更新権限無）
pull request
（管理者に適用を依頼）

📌 Git: 何故使う？

- **履歴管理と変更追跡が容易**

  すべての変更履歴が記録され、いつ・誰が・何を変更したかを追跡できる。

  誤った変更も容易に復元可能。

- **安全な並行開発**

  複数案の作業（機能追加・修正など）を個別にバージョン管理可能。

  あとでメインストーリに統合できる。

📌 GitHub: なぜ使う？

- **クラウド上でのバックアップと可視化**

  更新内容をクラウドに保存し、誰でも同じ履歴を取得可能。

  事故やPC故障時もデータ損失を防げる。

- **共同開発とコードレビューの効率化**

  他の開発者が変更内容をレビュー・コメントできる。

  品質向上と透明性を両立。

🧩 第1章：事前準備

必要なもの

1. **VS Codeのインストール**

   公式サイトから入手してインストール。

2. **Gitのインストール**

   Git公式サイトからダウンロードしインストール。完了後、VS Codeを再起動。

3. **GitHubアカウント**

   GitHubで無料アカウントを作成。

4. **GitHub拡張機能**

   VS Code左サイドバーの拡張機能アイコンから...

   - 「GitHub Pull Requests and Issues」を検索してインストール

     VScodeでGit操作を簡単にする機能

   - 「Git Graph」を検索してインストール

     平行開発やロールバックが容易になる機能

初回設定

1. VS Codeで「表示」→「ターミナル」を開き、以下を入力してユーザー情報を設定。

```shell
git config --global user.name "あなたの名前"
git config --global user.email "あなたのメールアドレス"
```

2. 設定確認：

```shell
git config --global --list
```

名前とメールアドレスが表示されれば設定完了。

1. VS Code左下のアカウントアイコンからGitHubにサインイン。
2. （任意）VS Codeのテーマやエディタ設定を整えて作業環境を統一。

🪜 第2章：使い方

2-1. 操作手順（VS Code GUI）

1. VS Codeでプロジェクトを開く。
2. 左サイドバーの「ソース管理」アイコンをクリック。
3. 「リポジトリを初期化」→ `.git` フォルダ作成。
4. ファイルが「変更」として検出されたら `.gitignore` を作成
5. GitHubで新しいリポジトリを作成し、URLをコピー。
    - GitHubのトップページ →「New repository」ボタン。
    - 名前と公開範囲を設定 →「Create repository」。
    - 表示された「https://github.com/ユーザー名/リポジトリ名.git」をコピー。
6. VS Codeでリモート登録。
    - VScodeでターミナルを開く
    - 「git remote add origin コピーしたURL」を入力してEnter

        例：git remote add origin https://github.com/shoichiro-suzuki/GenAI_Subcommittee_Study.git
7. コミット
    - メッセージを入力：変更内容を把握しやすいテキストにする
    - 「コミット」をクリック：これでローカルPC上で更新完了
8. 同期（Push）
    - ソース管理画面の「同期」をクリック：これでGitHub上にアップロードされる

2-2. 2回目以降の操作

1. コミット

- メッセージを入力：変更内容を把握しやすいテキストにする
- 「コミット」をクリック

2. 同期（Push）
   - ソース管理画面の「同期」をクリック

2-3.「.gitignore 」の作成

- `.gitignore` ファイルを追加して不要ファイルを除外しておく機能。venvを除外しないとライブラリファイルが全部管理対象になってしまう。
- 新しいファイル名を「.gitignore」（拡張子無し）で作成し、中にGit管理対象外のファイルやフォルダ名を記載。

Shell

```
# 一般的な除外例
node_modules/
.env
.vscode/
dist/
*.log

test*.py # testで始まるpyファイルが除外
test/ # testディレクトリとそれ以下の全ファイルが除外
```

⚙️ 第3章：開発中の基本操作

3-1. ブランチ作成（安全な並行開発）

> 📌 **シーン:** 新機能開発や検証を安全に行いたいとき。

操作:

1. 左下のブランチ名をクリック。
2. 「新しいブランチ作成」→ 名前（例: branch01）。
3. 作業後、コミット＆プッシュでGitHub上に反映。

メリット:

- mainを壊さず新機能を開発できる。
- 複数人開発でも干渉を防げる。

3-2. マージ（変更統合）

> 📌 **シーン:** ブランチでの作業が完了したらmainに統合。

操作:

1. main（統合先のブランチ）に切り替え。

2. Git Graphを開く（VScodeの左下にあり）
3. 統合したいブランチを左クリック
4. 「Merge into current branch...」を選択
5. 競合発生時はマージエディタで解消。

メリット:

- 安全に変更を統合し、履歴を整理できる。

3-3. ロールバック（過去の状態へ戻す）

> 📌 **シーン:** 誤ってコードを壊した・不要な変更を消したい。

操作:

1. Git Graphを開く（VScodeの左下にあり）
2. 戻したいコミットを左クリック
3. Revert...を選択
4. ブランチ名を指定、チェックアウトにチェックして「Create Branch」
5. mainに対してマージ：競合が起きたら強制上書きする

単純にロールバックするというより、過去の状態をコピーして新ブランチを作成し、現在の状態を強制上書きするイメージ。

メリット:

- 安心して試行錯誤できる。重大な変更も簡単に取り消せる。

3-4. フェッチ（GitHubの指定ブランチの最新状態のみ取得）

> 📌 **シーン:** 他メンバーが更新した内容を確認したいとき。

操作:

1. ソース管理の「…」→「フェッチ」を選択。
2. リモートの最新情報を取得。

メリット:

- 最新情報を把握してから作業できる。

3-5. クローン（GitHubの全状態をローカルにコピー）

> 📌 **シーン:** 他者のリポジトリをコピーして使う・修正したいとき。

操作:

1. GitHub上で対象リポジトリを開く。
2. 緑の “**Code**” ボタンをクリック。
3. 表示されたURL（HTTPSまたはSSH）をコピー。
4. VS Codeまたはターミナルを開く。
5. 保存したいフォルダへ移動。

   ※「移動したフォルダ＞リポジトリフォルダ」となるので、実際にプロジェクトが作りたいフォルダのひとつ上に移動する

6. ターミナルで「git clone コピーしたURL」を実行

メリット:

- ファイルを流用できる