Github Desktop を利用したチーム開発手法

本項では、GitHub, Inc. が提供するアプリケーション Github Desktop を利用した開発手法について解説します。
この資料と同一のフォルダ内に、スライドも掲載しており、双方を参照することで学習を進めます。

基本的にはスライドで解説を進めるので、本項は必要に応じて参照してください。

1. 導入編

Git・Github・Github Desktopについて知ろう!

1-1. Gitとは何か?

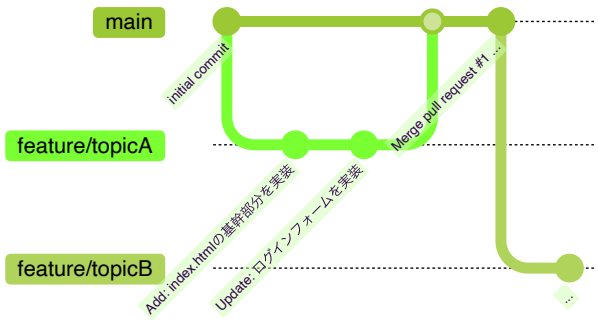
> Git（ギット）は、プログラムのソースコードなどの変更履歴を記録・追跡するための分散型バージョン管理システムである。
> 出典: https://ja.wikipedia.org/wiki/Git

Gitは、プログラムのソースコードを管理するためのツールの一つです。多くの開発現場で採用されており、Gitを活用できることは、それだけで大きなアドバンテージとなります。

Gitでは一つのアプリケーション開発につき、一つのリポジトリと呼ばれるものを作成します（規模によっては二つ、三つになることもあります）。
このPC上で作成するリポジトリを、ローカルリポジトリと言います。

このリポジトリに変更前と変更後の**差分を記録**することで、作業内容の明文化、バグ発生時の作業内容の巻き戻し等を可能にします。また、複数人で開発を行う際の作業の衝突を解決するための機能も備わっています。

> Topic: **差分を記録**することで、「`program.c`, `program(1).c`, `program(2).c`, `program(3)_これが最終版.c`, `program_提出用.c` ...」のような形でのバックアップを取る必要が無くなります。



1-2. Githubとは何か?

> GitHub（ギットハブ）は、ソフトウェア開発のプラットフォームであり、ソースコードをホスティングする。コードのバージョン管理システムにはGitを使用する。
> 出典: https://ja.wikipedia.org/wiki/GitHub

Githubは、Gitのリポジトリをインターネット上で管理するためのサービスです。このリポジトリを、リモートリポジトリと言います。
リポジトリをアップロードしておくことで、複数人での開発時のソースコードの共有、ソースコードのバックアップ、自身の実績の公開の場として、等の効果を期待できます。
この資料も、Github上で公開されています。

Githubを活用した開発では、大まかに、以下のような手順で作業が進められます。

1. clone: Githubリポジトリを丸ごと丸ごとダウンロードする
2. branch: 作業を枝分かれさせて、他の開発者との衝突を防止する
3. 通常通りに、プログラムを書く
4. commit: 作業前と後の**差分を記録**しよう
5. push: 作業の成果をGithubにアップロードして共有する
6. Pull Request: 枝分かれした成果を結合する

1-3. Github Desktopとは何か?

> GitHub Desktop では、コマンド ラインや Web ブラウザーではなく GUI を使用して GitHub と対話できます。
> 出典: https://docs.github.com/ja/desktop

Github DesktopはGit / Githubの機能をGUI（ボタン等でコンピュータに命令を送るもの）で操作するためのアプリケーションです。Gitは多くの場合、CUI（文字のみでコンピュータに命令を送るもの）で操作を行いますが、これに慣れるためには時間が必要なため、簡略化のために使用します。

> Topic: CUIでも操作できると、様々な環境に適応できるのでおすすめです

2. Github Desktopを触ってみよう

Github Desktopを実際に使ってみよう!

2-1. clone: Githubリポジトリを丸ごとダウンロードしよう

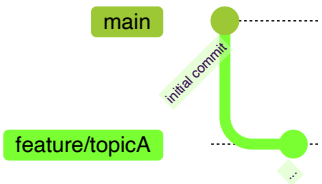
`clone` （クローン）は、リポジトリを**丸ごと**、PCにダウンロードするための操作です。
Githubにアップロードされているリポジトリを丸ごと取得し、内部の全ファイルも同時にダウンロードします。

基本的には、開発の最初に各開発者が一度ずつ行う操作です。

Github Desktopでは、以下のように操作することで、クローンの操作を行うことができます。
例として、 `git-tutorial-2025` のリポジトリで作業してみましょう。

▶ 練習: Github Desktopでクローンしてみよう

2-2. branch: 作業を枝分かれさせよう



`branch` （ブランチ）は、リポジトリ内で作業を行う際、他の開発者と作業を**枝分かれ**させる機能です。
このように、作業を枝分かれさせることを、慣習的に「ブランチを切る」「ブランチを生やす」と言います。

例えば、機能Aを作る開発者と機能Bを作る開発者で、ファイルの編集箇所が重複してしまった場合、片方の変更内容が意図せず削除されてしまう場合があります。
これを防止するため、機能ごとに編集内容を枝分かれさせ、結合する際に編集箇所の重複を確認・まとめて修正するという手法で作業を進めます。
また、これによって編集履歴が分かりやすくなり、編集内容の巻き戻し等の際にどこまで巻き戻すのかの目算が立てやすくなる効果も期待できます。
作業中に致命的なバグが発生した場合に、ブランチを放棄することで、大本のブランチへの被害を回避することも可能です。

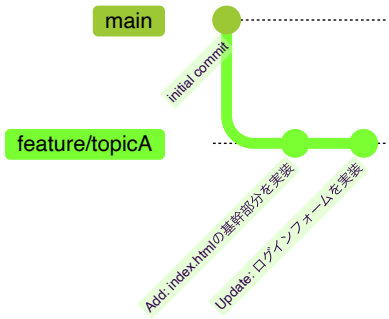
一人で作業する場合でも、ブランチを切ることで作業の見通しが良くなる効果が期待できるため、ブランチを切って作業することが推奨されます。

また、作業するブランチを切り替える操作を `checkout` （チェックアウト）と呼びます。

Github Desktopでは、以下のように操作することで、ブランチを切る操作を行うことができます。

▶ 練習: ブランチを切ってみよう

2-3. commit: 作業前と後の差分を記録しよう



`commit` （コミット）は、作業前と作業後の**差分を記録**する操作です。作業が一段落したタイミングでコミットを行うことで、作業内容の明文化や、バグ発生時の巻き戻し等で役立ちます。また、コミットにはコミットメッセージという説明文を付与するため、作業内容を振り返る際の手助けにもなります。

> Topic: リポジトリに `.gitignore` という名前のファイルを追加することで、Git管理下から強制的に除外するファイルを定義することができます。
> これを使用すれば、Githubにアップしてはいけない情報（テスト用のパスワード、認証情報のハッシュ等）をGit管理下から除外できます。
>
> ```
> # .gitignore
> .env
> .password
> .DS_Store
> ```

Github Desktopでは、以下のように操作することで、コミットの操作を行うことができます。

▶ 練習: 作業してコミットしてみよう

2-4. log: 作業の履歴を確認しよう

`log` （ログ）は、コミットの履歴を確認する操作です。
現在作業中のブランチでのコミットの履歴を確認し、作業内容を確認することができます。
作業を始める前やコミットが正しく完了したか、作業の開始地点が正しいかなどの確認のために用います。

Github Desktopでは、以下のように操作することで、ログを確認することができます。

▶ 練習: コミットの履歴を確認しよう

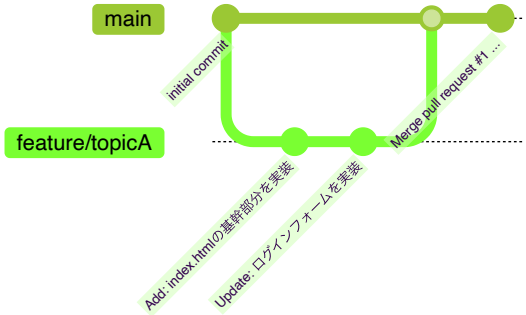
2-5. push: 作業の成果をGithubにアップロードして共有しよう

`push` （プッシュ）は、現在のブランチの作業内容をGithub等にアップロードする操作です。
他の開発者に作業内容を共有する時、作業が一段落して、ブランチの内容を元のブランチに結合したい時などに行います（後述。ブランチの結合操作は他開発者に確認してもらうのが望ましいため、Github上で行います）。

Github Desktopでは、以下のように操作することで、プッシュの操作を行うことができます。

▶ 練習: 作業内容をプッシュしてみよう

2-6. Pull Request: 枝分かれした成果を結合しよう



`Pull Request` （プルリクエスト）は、Githubの機能の一つで、ブランチ機能で分岐した作業を大本のブランチへと結合するための機能です。この結合作業を `merge` （マージ）と呼びます。この操作はローカルでも行うことができますが、ブランチの結合操作は**他開発者に確認してもらうのが望ま**

しいため、Github上で行います。

> Topic: `Pull Request` は、Github以外のGit管理用のサービス（GitLab等）では、 `Merge Request` （マージリクエスト）と呼ぶこともあります。

ブランチの結合作業は、以下の順序で行います。

1. 作業ブランチの内容をGithubにアップロード（push: プッシュ）する。
2. 作業ブランチから元のブランチへの結合用の `Pull Request` を作成する。
3. 他開発者（レビュワー）が作業内容を確認して、問題点を指摘する。問題点が無い場合、確認した旨をコメントする。
4. ブランチを結合（merge: マージ）する。

Github上で、以下のように操作することで、Pull Requestの作成〜結合までを行うことができます。

▶ 練習: プッシュした新規ブランチを、元のブランチに結合してみよう

2-7. fetch / pull: Githubリポジトリの変更部分をダウンロードしよう

`fetch` （フェッチ）は、Github上のリモートリポジトリの**変更を確認**する操作です。
`pull` （プル）は、Github上のリモートリポジトリの**変更をダウンロード**する操作です。

一見すると違いがありませんが、 `fetch` で行うのは変更が行われているかの確認のみで、実際にPC上のファイルの内容が書き換わることはありません。対して、 `pull` を行った場合、変更内容がPC上のファイルに反映され、書き換えられます。

Github Desktopでは、以下のように操作することで、これらの操作を行うことができます。

▶ 練習: fetch / pullで変更後のGithubの内容を取り込もう