Git の仕組みとチーム開発

J23072　蝦名武尊

1. 初めに

このドキュメントの目的は Git の仕組みと Git を用いたチーム開発についてまとめことである。

2. Git とは

Git とはソースコードのバージョンを管理することができるツールである。例えば、誤ったコードを保存してしまい、変更前の状態に戻したい場合や、変更前のコードと現在のコードの違いを調べる場合に有効である。

Git にはいくつかの考え方や仕組みがあり、それを以下に記していく。

2.1 リポジトリ

リポジトリとは Git で管理したいコードのまとまりのことである。ある開発において必要な複数個のファイルを 1 つにまとめ、それをリポジトリと称し管理を行う。

2.2 コミット

リポジトリ内のファイルの追加や変更、削除などの操作を記録することをコミットという。例えば、とあるコードに「A 機能」を追加しコミットを行い、その後にまた「B 機能」を追加しコミットを行うなど、操作の履歴を作成することである。

リポジトリ内には「tracked file」と「untracked file」に設定されたファイルがそれぞれ存在する。コミットを行った場合に「tracked file」は Git で履歴を追跡することができるが、「untracked file」は追跡の対象外である。

2.3 ブランチ

ブランチとは履歴を枝分かれさせ、分岐させるものである。例として、もともと自分が開発していたコードがあるとする。このコードをベースとして新しい機能を追加する場合、ベースとなったコードから一度分岐させ新しいブランチを作り、新しいブランチのコードを変更して開発を進めるべきである。この時、ベースとなったコードと新しいブランチのコードは別に管理されているため、分岐する前と後の状態を保持することができる。また、ベースとなったコードはデフォルトのブランチのコードとして管理されているため、「main ブランチ」または「master ブランチ」と名付けられている。

2.4 マージ

マージとはあるブランチに対して別のブランチで変更した結果を取り込むことである。例えば main ブランチとそれから派生し、A 機能を追加した A ブランチがあるとする。この時、A ブランチでの変更点は main ブランチには反映されておらず、main ブランチに A 機能を取り込むには main ブランチに対して A ブランチをマージする必要がある。また、マージには方向がある事に注意が必要である。

また、別のブランチから異なる変更を行った同名のファイルのマージを行うと競合が発生してしまう。これをコンフリクトという。

3. コマンドプロンプトでの Git

3.1 インストール

ここでは Windows での Git のインストールと初期設定についてまとめる。

1. https://gitforwindows.org/にアクセスし、Git のインストーラを入手する
2. インストーラを起動し指示に従いインストールを行う
3. コマンドプロンプトにて git –version を入力して反応があればインストールは成功
4. ”git config –global user.email “~~” “でユーザーの email アドレスを登録
5. “git config –global user.name “~~”“でユーザーネームを登録

以上の操作を行うことで Git の使用が可能になる。

3.2 操作

3.2.1 リポジトリ初期化

今回は PC の中に入っているフォルダ「work」を 1 つのリポジトリとして Git にて管理する場合を参考に解説する。

1. コマンドプロンプトにて次のコマンドを入力し work フォルダに移動する。

cd work

2. work フォルダを Git で管理するための「.git」ディレクトリを作成するため次のコマンドを入力する。

git init

以上の操作で work ファイルをリポジトリとして管理することができる。また、この状態で以下のコマンドを入力することで今のリポジトリの状態を確認することができる。

git status

3.2.2 コミット操作

どのファイルをコミットするか予約するために次のコマンドを入力する。

git add ファイル名

個別のファイル名だではなく、「.」を使用することでリポジトリ内のすべてのファイルを指定できる。

コミットが予約されるデータは add コマンドを入力した時点のデータである。また、この動作を「ステージングに追加する」という。

add コマンドの対象にしたファイルは自動的に「tracked file」としてみなされる。「tracked file」とされたファイルに変更を行いそれをコミットしたい場合、変更後にもう一度 add コマンドで指定する必要がある。

コミットを行うときは次のコマンドを入力する。

git commit -m “~~”

コマンドの語尾にはどのタイミングのコミットか他人から見ても確認ができるようなコメントを必ず入力する。

コミットが成功するとハッシュ値が付与され、これをコミット ID と呼ぶ。これを基に Git はコミットを判別している。

3.2.3 ログの表示

現在のブランチのコミットの履歴を表示するには次のコマンドを入力する

```
git log
```

3.2.4 gitignore

リポジトリの中にGitでの管理を避けたいファイルやフォルダがある場合、「.gitignore」とファイルを作成する。それを開き、管理を避けたいものがファイルならば、「/ファイル名」、フォルダならば「/フォルダ名/」と入力する。そうすることで指定したファイルやフォルダは「tracked file」や「untracked file」に属さなくなりGitの管理から除外することができる。

3.2.5 ブランチの作成

現在所属しているブランチから新しいブランチを作成し、新しいブランチに切り替えるには次のコマンドを入力する。

```
git checkout -b ブランチ名
```

ブランチ名には”/”で階層を作ることができる。頻繁に使われるブランチ名は「feature/機能名」といったものである。

ブランチをただ切り替えるならば、次のコマンドを入力する。

```
git checkout ブランチ名
```

3.2.6 マージ

ここでは大本のブランチ「main」に新しいブランチ「feature/user_login」をマージする例を解説する。

マージを行うには現在自分が所属しているブランチがmainであることを確認してから次のコマンドを入力する。

```
git merge feature/user_login
```

3.2.7 データの復元

もし、コードにバグが見つかるなど作業のやり直しがしたくなり、コミットしたデータの復元を行う場合、status コマンドで戻したい地点のコミット ID を取得する。その後、次のコマンドを入力する。

git reset –hard コミット ID

reset コマンドにてデータの復元が可能であるが、データの復元を行った履歴は残らない点に注意が必要である。

また、行ったコミットの取り消しを行う場合は次のコマンドを入力する。

git revert コミット ID

実行後、revert のメッセージを編集する画面が表示される。それを Vim エディターの操作に従い保存することでコミットの取り消しが完了する。

4. PyCharm での Git

PyCharm とは JetBrains（ジェットブレイン）社が提供している Python に特化した IDE（統合開発環境）のことである。これを使用することでコマンドプロンプトでの Git の操作よりも視覚的に理解しやすく、容易に操作が可能である。

4.1 基本操作

すでに Git で管理されているフォルダを PyCharm で開き、画面左下の「Git」と記載されたタブを選択することで Git の操作に移行することができる。画面左側にはブランチの情報が、右側にはコミットの履歴が視覚的に理解しやすく表示されている。

ブランチの移動には移動したいブランチを選択した後、『右クリック→チェックアウト』で移動することができる。また、同じように『右クリック→新規ブランチ』にて新しいブランチの作成が可能である。

コミットを行うにはファイルの編集後、画面左端の「コミット」と記載されたタブを選択し、変更リストに表示されたコミットしたいファイルを選択する必要がある。

マージを行う場合も大きな違いはない。元となるブランチに移動した後、マージするファイルを選択、『右クリック→現在のブランチにマージ』を選択すればよい。

4.2 コンフリクトの解消

上記でも述べたがコンフリクトとは別のブランチから異なる変更を行った同名のファイルのマージを行うと競合が発生してしまうことである。

PyCharm ではコンフリクトが発生した場合、「競合」というポップアップが表示されコンフリクトが発生したファイルが表示される。これを開くことで、画面中央にコンフリクトを解消した結果のコード、左右に取り込む予定のファイルが表示される。コンフリクトの原因となった箇所に矢印とバツ印が表示され、それぞれをクリックすることで取り込むか削除するかを選択することができる。

この操作を行うことでコンフリクトを解消することができる。

5. 終わりに

以上のように Git を正しく操作することで個人的な開発の効率を向上させるだけでなく、チーム開発に対しても大いに貢献することができる。また、PyCharm を使用した Git の操作はコマンドプロンプトでの Git 操作よりも視覚的に理解しやすくコマンドでの操作よりも容易であるため、不慣れであっても Git を操作することができる。

6. 参考文献

・Python プログラミング VTuber サプー.”**【わかりやすい！Git 操作】初心者向けの Git の基本 〜 30 分で入門！.** “ YouTube, 2021/9/18, https://youtu.be/6SLMB7BPG9E.