GC2017 ２年前期　バージョン管理手法

1. Git の利用
2. Git の利用環境整備
2.1. Git クライアントのインストール
2.2. TortiseGit のインストール

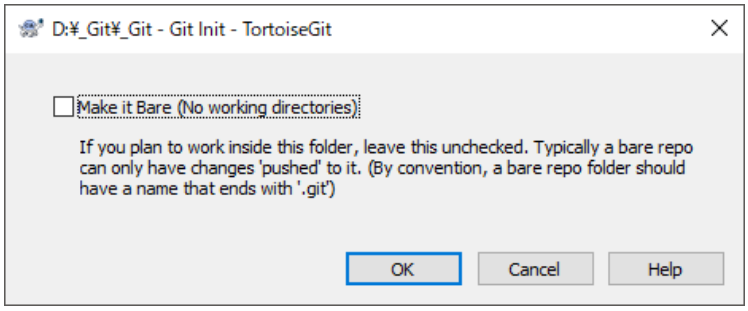
3. コミットの作成
3.1. Git 用フォルダの作成

ローカルマシンの適当な場所に 「_Git」 フォルダ作成

3.2. ローカル・リポジトリの作成

1）「_Git」 フォルダ上でマウス右ボタンを押して、プルダウンメニューの表示を行う。
2）その後、「Git Create repository here」 をクリックしリポジトリを作成する。

3）Bare ファイルの作成確認のダイアログボックス開くが、今回は作成せずそのまま OK を押す。



これで、ローカル・リポジトリの作成が終了する。

※リポジトリ（repository）とは、情報工学において、システム開発プロジェクトに関連するデータの一元的な貯蔵庫を意味する。

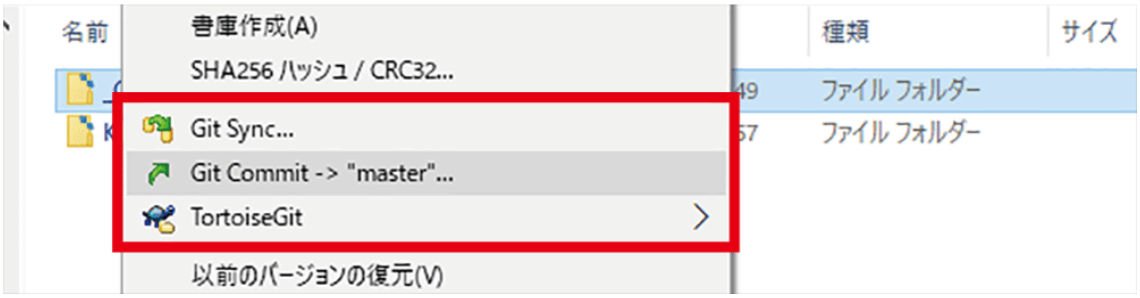
※Bare ファイルについては後述する。

3.3. ファイルの追加

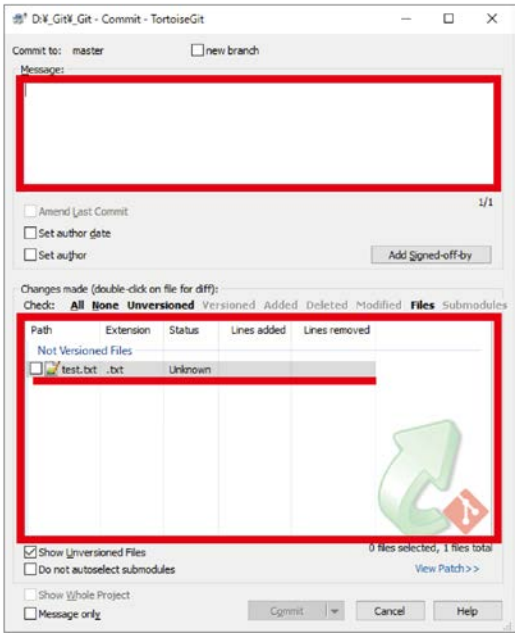
作成したローカル・リポジトリの中に 「test.txt」を新規作成する。

3.4. コミットの作成

先ほどのファイルを作成したリポジトリ・フォルダ上にマウスカーソルを移動し、右ボタンで表示されたメニューから 「Git Commit -> “master” 」 を選択する。



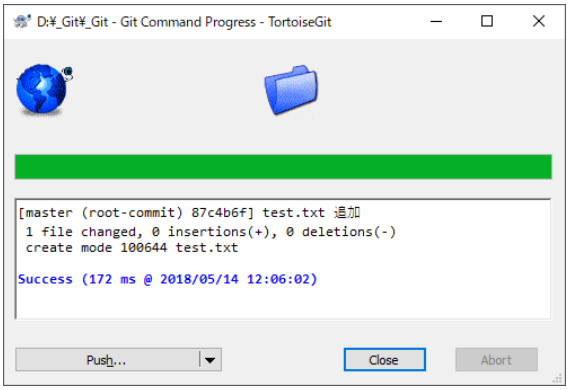
そうすると下記のウィンドウが開く。



← コメント記入欄
ここでは 「test.txt 追加」と
書いてみよう。

← リポジトリ内で、追加、修正が
あったファイルの一覧
管理対象にしたいファイルに
チェックを入れる。

← コメント、対象ファイルが決まると
コミットできる。



コミット終了後の表示

※ ここで赤色のメッセージが出ると何らかの不具合がある。
赤色の表示が出た場合、佐伯まで報告ください。

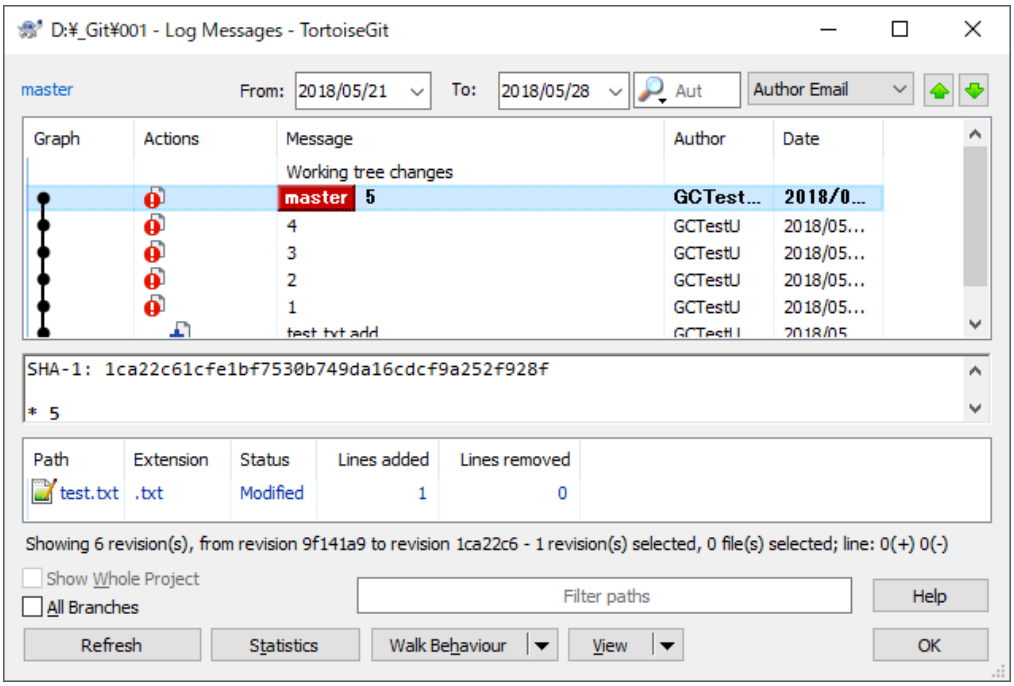
4. 以前の状態に戻す

作業の区切りがついたところで、何回かコミットしてみよう。ここでは、リポジトリに “test.txt” を作成し、内容を編集する課程で 5 回コミットしてみた。

Git では履歴管理を行っているため、手軽に以前の状態に戻すことが可能になる。

4.1. コミットの履歴表示

現在管理されている、コミットの一覧を見るにはエクスプローラで、当該リポジトリ・フォルダ上でマウスの右ボタンを押し、プルダウンの中から “Show log” を選択すると下記のようなウィンドウが開く



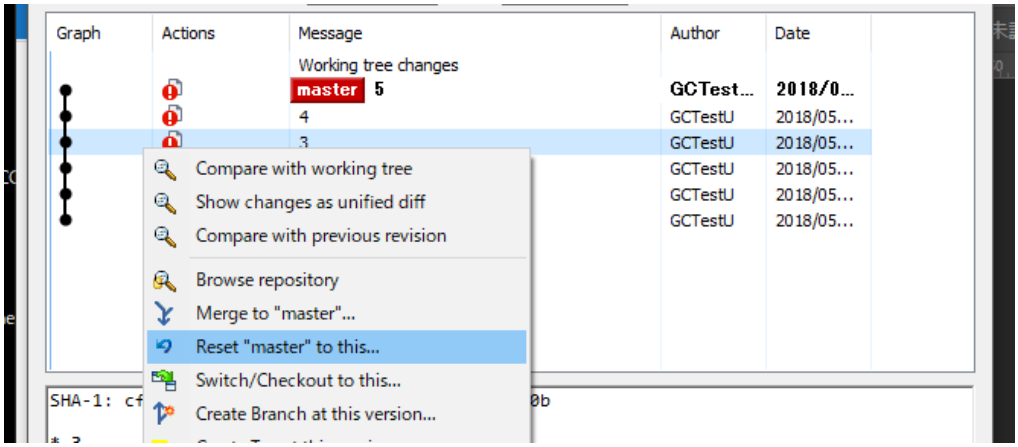
このときの test.txt の内容は下記の通りとする。

```
1 ← コミット 1 回目 コメント”1”
2 ← コミット 2 回目 コメント”2”
3 ← コミット 3 回目 コメント”3”
4 ← コミット 4 回目 コメント”4”
5 ← コミット 5 回目 コメント”5”
```

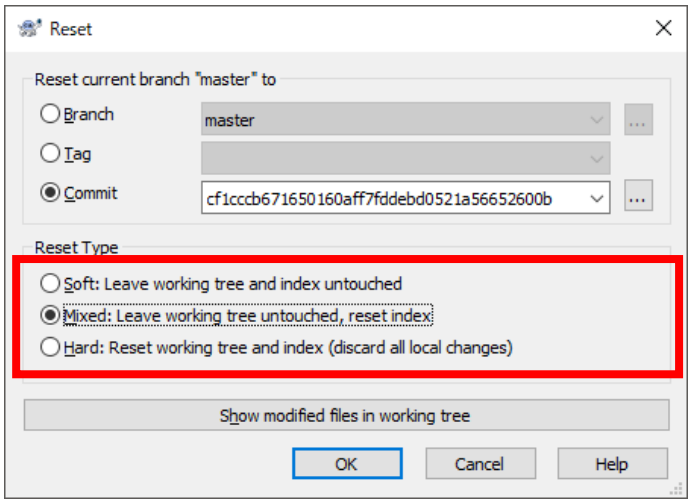
4.2. 以前の状態に戻す

TortoiseGIT の Log Messages ウィンドウで、戻したいコミット、ここでは 3 回目のコミットに戻すこととする。

そこで表示にある 3 回目のコミット表示の上にマウスカーソルを移動させ、右ボタンを押す。そして表示されるプルダウンメニューの中から “Reset master to this...” を選択する。



そうすると下記のようなウィンドウが開くはず。



赤い枠の中に、 Soft、Mixed、Hard の 3 種類があり、現在真ん中の “Mixed” が選択されている。

ここで、“Hard” を選択することで、リポジトリ内に作成した test.txt の中身は下記のようになっているはず。

```
1 ← コミット 1 回目 コメント”1”
2 ← コミット 2 回目 コメント”2”
3 ← コミット 3 回目 コメント”3”
4 ← コミット 4 回目 コメント”4”  ← 4，5 番目のコミットが破棄されている。
5 ← コミット 5 回目 コメント”5”
```

赤く示した場所が削除され、3 回目のコミットの状態に戻った。

ただし、Reset は指定したコミット以降の操作を破棄してしまうので注意！

このあたりの利用方法が Git の真骨頂です。

参考：

項目	HEAD	index	working tree
Soft	○		
Mixed	○	○	
Hard	○	○	○

4.3. Resetしたコミットを元に戻す

基本的に Hard でリセットした時点で、それ以降のコミットの内容は全て破棄されます。
しかし、これを元に戻す方法として “reflog” を参照する方法がある。

reflog（参照ログ）とは、リポジトリにかかる HEAD やブランチ先端の動きの履歴です。

- ○各個人のローカルリポジトリに存在
- ○ブランチの切り替え、新たに加えられた変更のプル、履歴の書き換え、あるいは単なる新規コミットの実行などを記録
- ○ show reflog で HEAD の移動履歴を確認可能です。

これを使うことで、謝って Rest した際に、Reset 操作自体を無き者にし、元に戻すことが可能になります。

4.4.

5.　リモート・リポジトリの作成

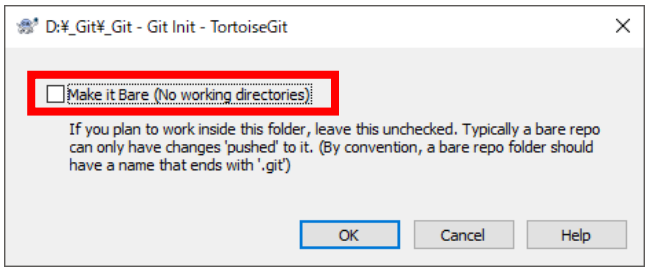
5.1.　リモート・リポジトリ用のフォルダ作成

さあ、適当に作成しましょう！

5.2.　Bare(がらんとした)ファイル

先ほど作成したフォルダに、今まで通りリポジトリを作成します。ただし、今回は　3-2　の操作と少し異なります。

今回は、　Make it Bare　にチェックを入れてください。



日本語訳

「もしこのフォルダーで編集作業を行いたいのであれば、チェックを外してください。典型的には　Bara リポジトリはプッシュされた変更だけを含みます

(慣例により、Bare リポジトリは　.git　で終わる名前にしてください)　」

5.3.　リモート・リポジトリへの push

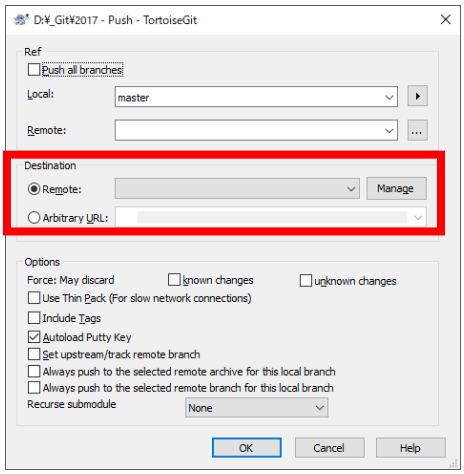
それでは、作成したリモートリポジトリに、今操作しているリポジトリの情報をコピー(push：プッシュ)してみましょう。

元のリポジトリで、テキストファイルを編集し、コミットしたあと、左にある　“push”　を押してください。

5.4. 接続先設定

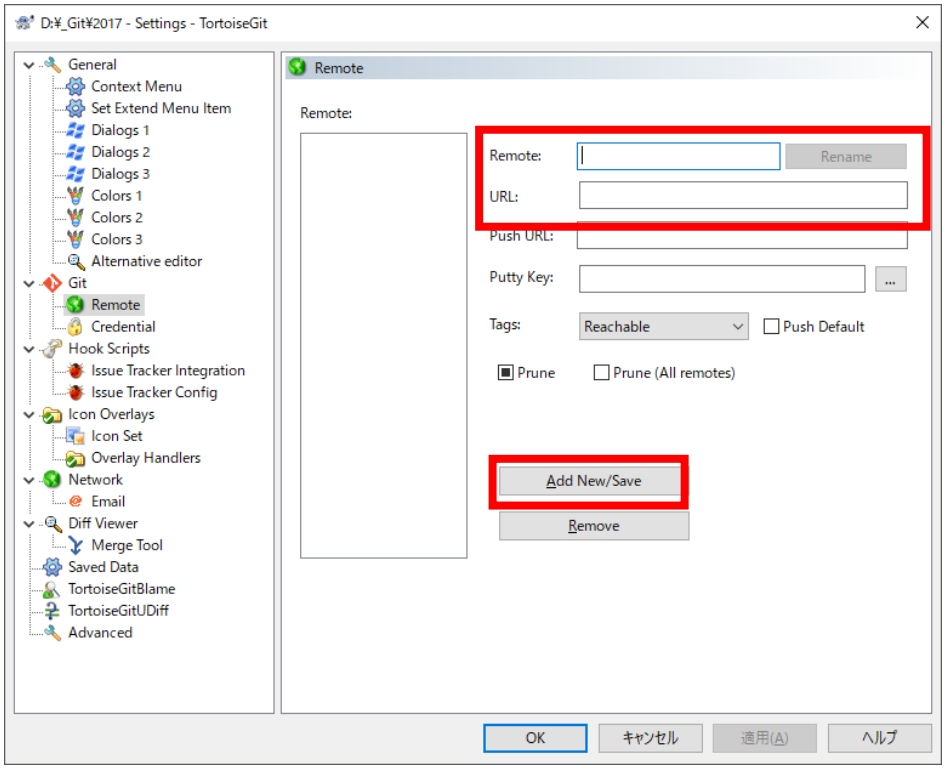
そうすると下記のようなウィンドウが開きます。現在はリモートの場所を登録していないために開いたので、接続先を設定します。

赤枠の “Manage” ボタンを押してください。



5.5. リーモートの場所を設定する。

そうすると接続先設定のウィンドウが開きます。



Remote： 接続先設定名、分かりやすく記入してください、デフォルトは origine

URL：ここにリモートの URL を記入しますが、ここではローカルにあるリモートリポジトリを設定します。

5.6. リモート・リポジトリからの pull
5.7. コンフリクトの対応
5.8. コンフリクトが発生しない運用方法
5.9. チェックアウト
5.10. ブランチ
5.11. マージ
6. 実際の運用
7. VisualStudi での Git 利用
http://www.atmarkit.co.jp/ait/articles/1607/14/news020.html