GC2017　2 年前期　 バージョン管理手法

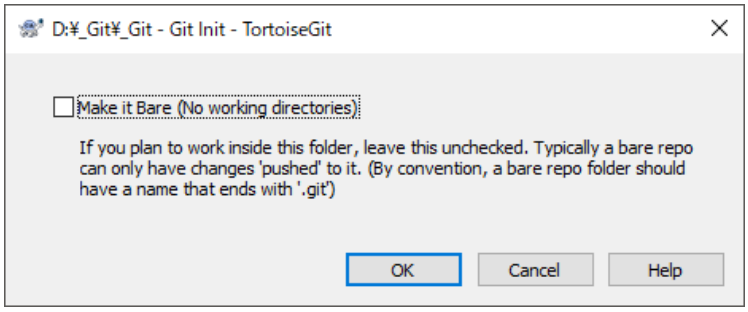
ローカルマシンの適当な場所に 「_Git」 フォルダ作成

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。

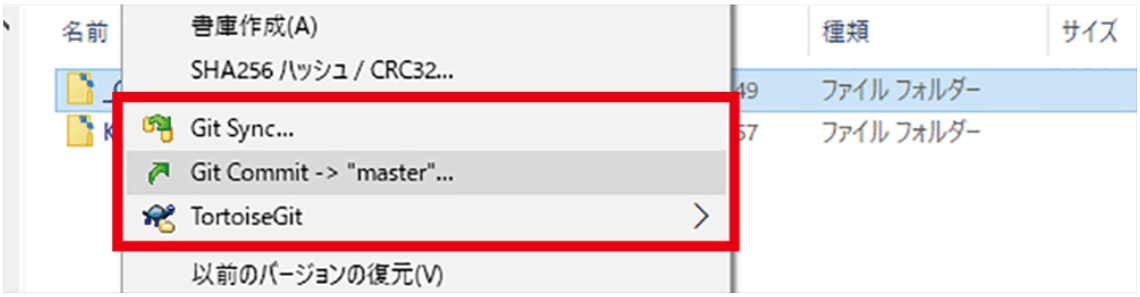
※Bare ファイルについては後述する。

3.3. ファイルの追加

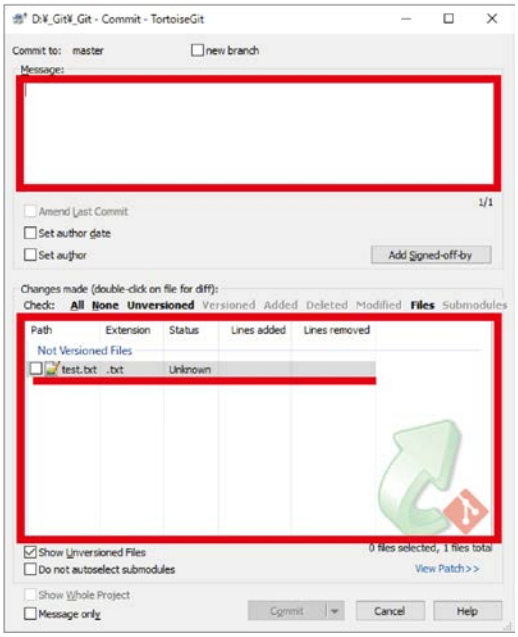
作成したローカル・リポジトリの中に 「test.txt」を新規作成する。

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。



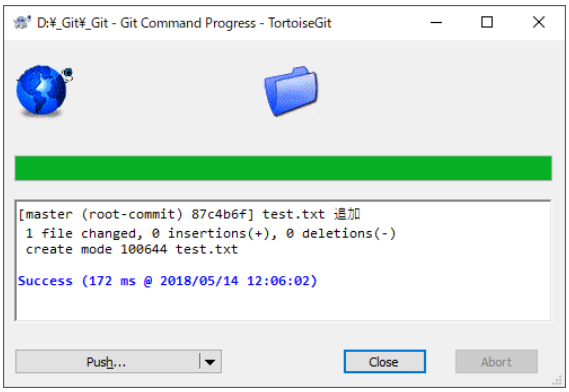
そうすると下記のウィンドウが開く。



← 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(add する)。

← 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。

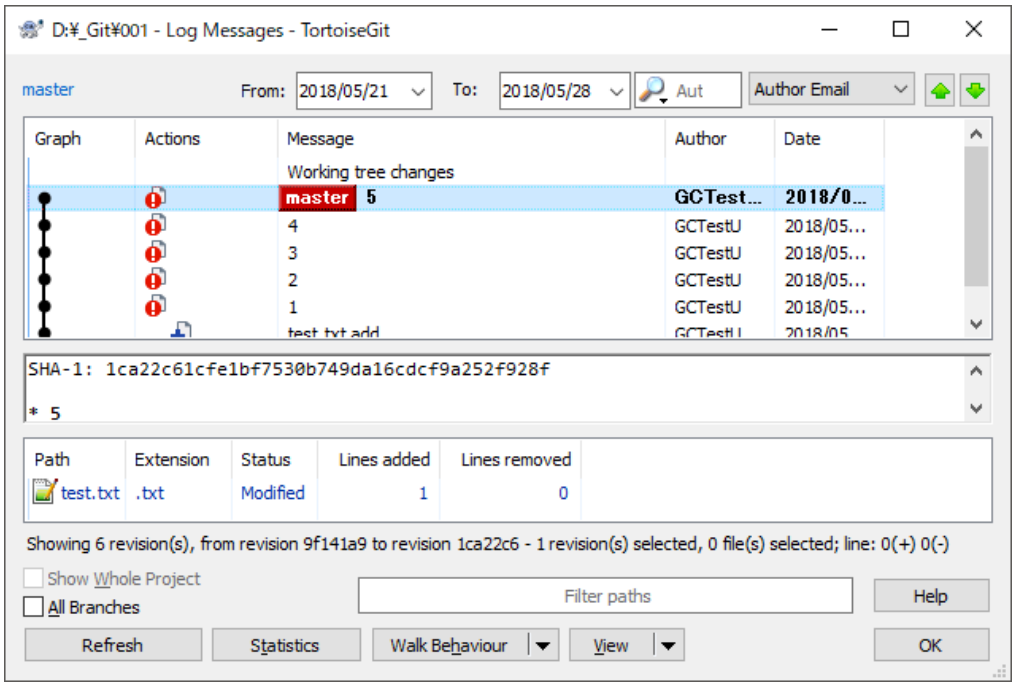
4. コミットの切り替え：以前の状態に戻す

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

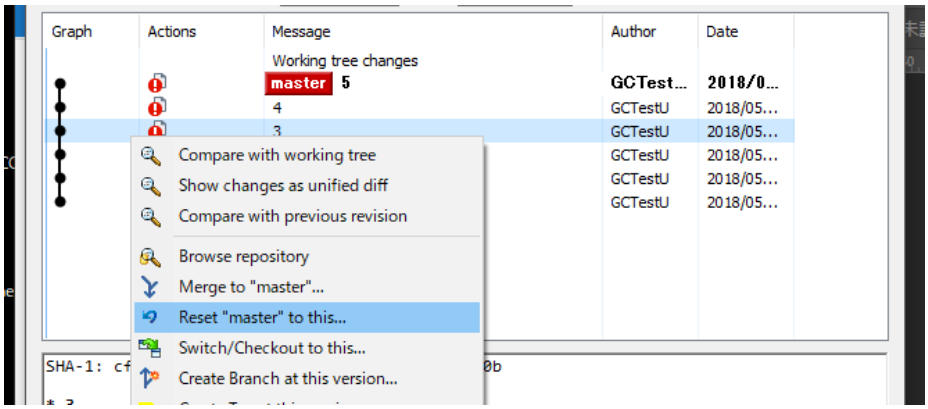


このときの test.txt の内容は下記の通りとする。

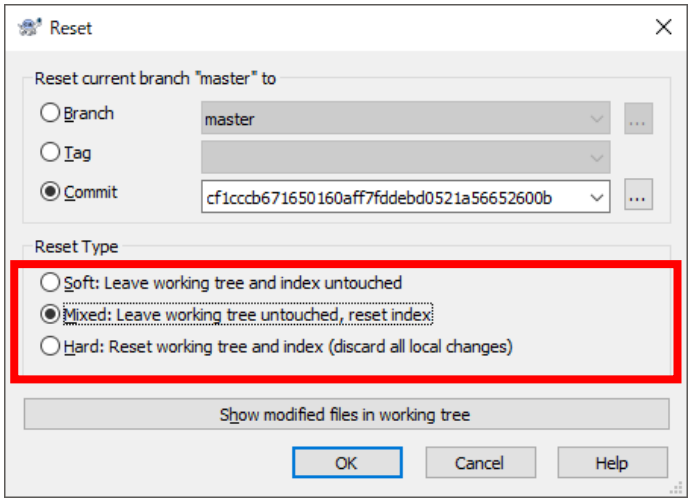
```
1 ← コミット 1 回目 コメント”1”
2 ← コミット 2 回目 コメント”2”
3 ← コミット 3 回目 コメント”3”
4 ← コミット 4 回目 コメント”4”
5 ← コミット 5 回目 コメント”5”
```

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”
```

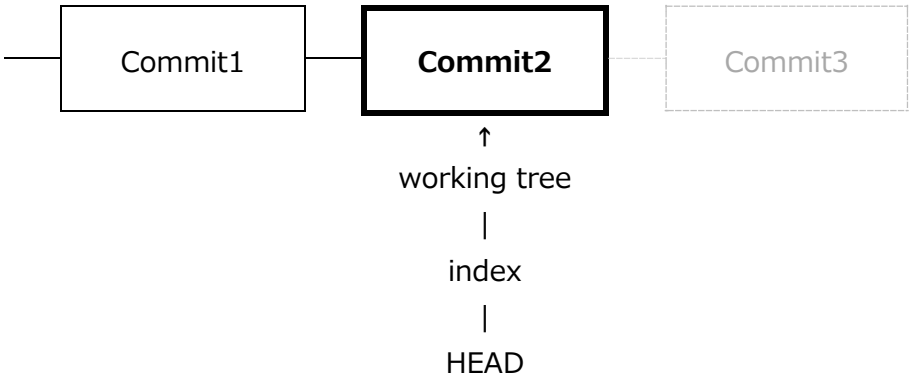
赤く示した場所が削除され、3 回目のコミットの状態に戻った。Reset は指定したコミット以降の操作を破棄してしまうので注意！

参考：git reset の 3 つのオプション

項目	HEAD	index	working tree
-- soft	○		
-- mixed	○	○	
-- hard	○	○	○

4.2.1. reset の動作

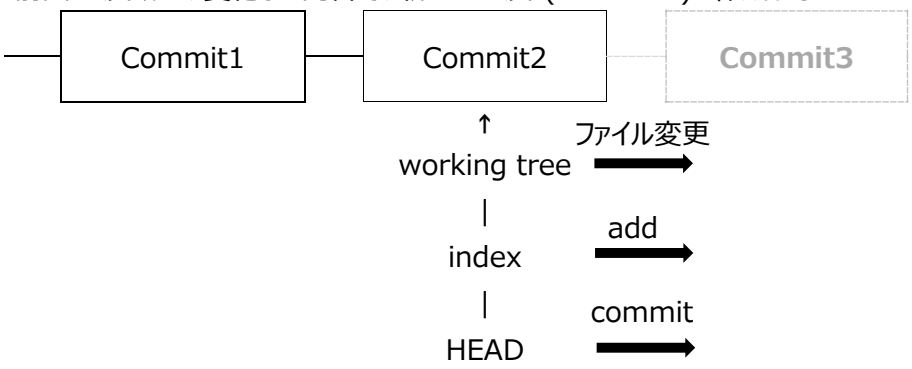
1) コミット前の状態



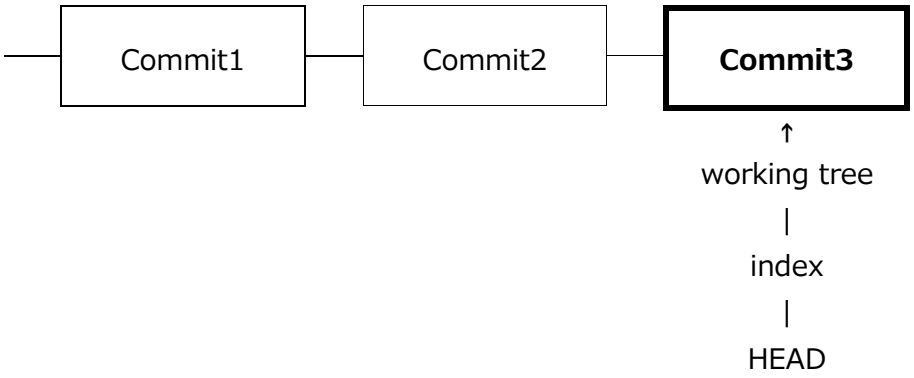
2) 追加修正作業 → ファイルの状況変化

インデックスに対象ファイル登録

前回コミットから、変化した内容で、新たにコミット(Commit3)を作成する



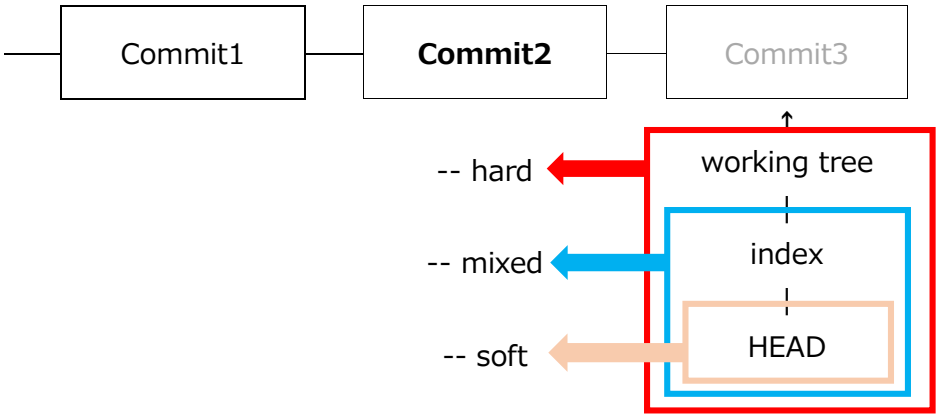
3) commit が終了した状態



reset の各モードで何がどう戻るのか図示したものが次になります。

4）resetした状態

枠線内の項目が reset 対象ファイル



-- hard	：	コミット内容全ての破棄、ファイルは以前のコミットの状態に戻る。
-- mixed	：	index、HEAD の位置だけ、以前のコミットの状態に戻る。
-- hard	：	HEAD 位置だけ、以前のコミットの状態に戻る

※reset　操作の後では元に戻れない

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。

〇各個人のローカルリポジトリに存在

〇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

〇 show reflog で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. reflog の注意点

git reflog は、「これまで HEAD が辿ってきた履歴」を管理しています。そのため add しただけで commit してないものがある時に、reset --hard をしてしまったら、add した内容の復元はできません。add すらしてないもの当然消えます。

reset を使うときはくれぐれも慎重に！

5. リモート・リポジトリの作成

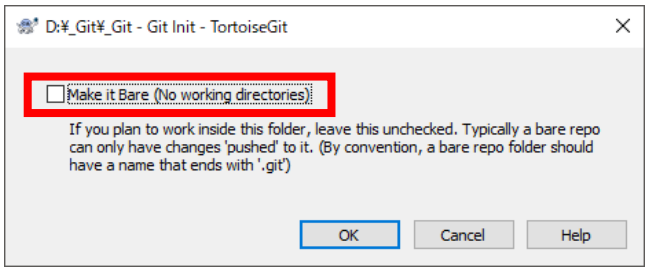
5.1. リモート・リポジトリ用のフォルダ作成

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、リモート・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 にチェックを入れます。



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) 」

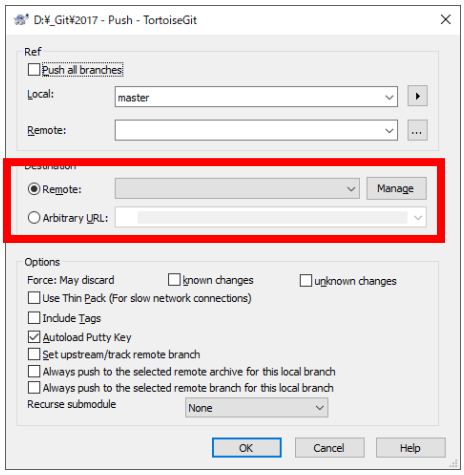
5.3. リモート・リポジトリへの push

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。

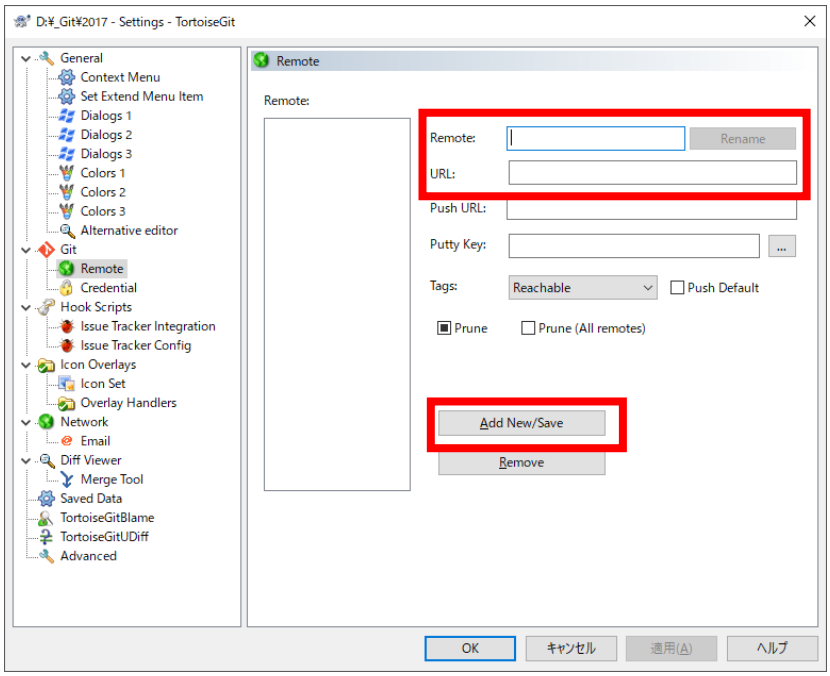
そうすると下記のようなウィンドウが開きます。現在はリモートの場所を登録していないために開いたので、接続先を設定します。

赤枠の “Manage” ボタンを押してください。



5.4. リーモートの場所を設定する。

Manage ボタン押下後、接続先設定のウィンドウが開きます。

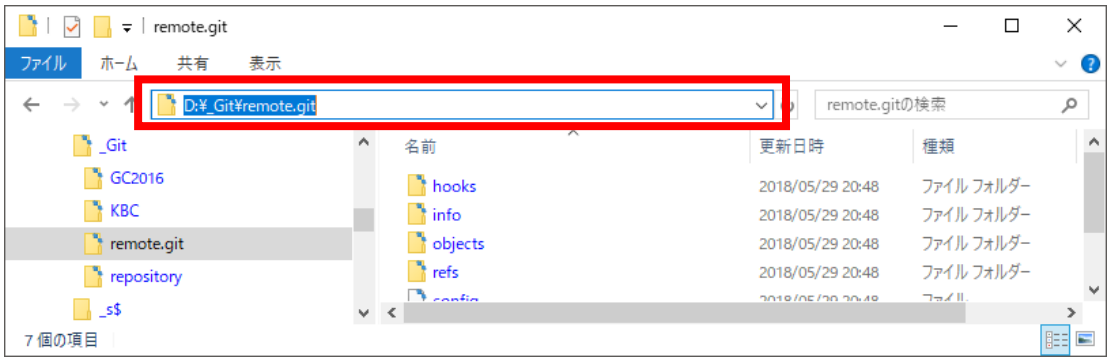


Remote： 接続先設定名、分かりやすく記入してください、デフォルトは origine

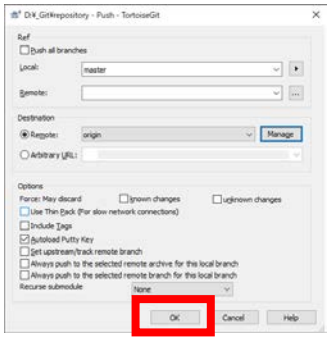
URL：ここにリモートの URL を記入しますが、ここではローカルにあるリモートリポジトリを設定します。

5.5. ローカル・パスの取得

ローカルパスの取得は簡単です。 Windows キー ＋ ‘E’ キーにてエクスプローラを起動し、作成したリモートリポジトリを選択します。その後アドレスバーに表示されるものがパス(PATH) です。この内容を接続設定ウィンドウの URL の部分に貼り付けます。

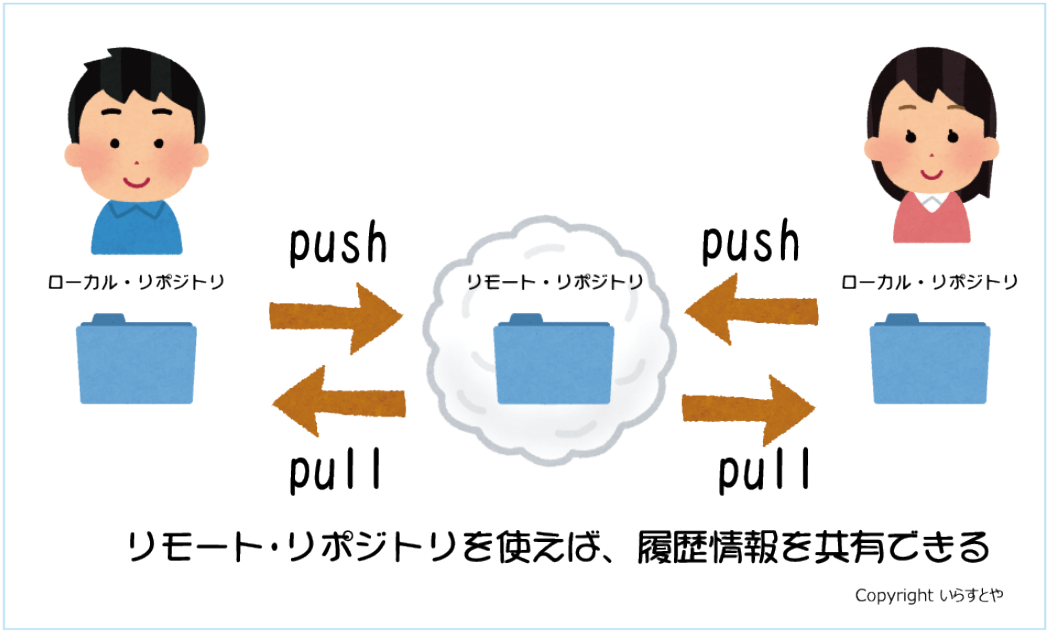


その後、OK を押して、接続設定ウィンドウを閉じて、push ウィンドウで OK を押せば push 完了です。



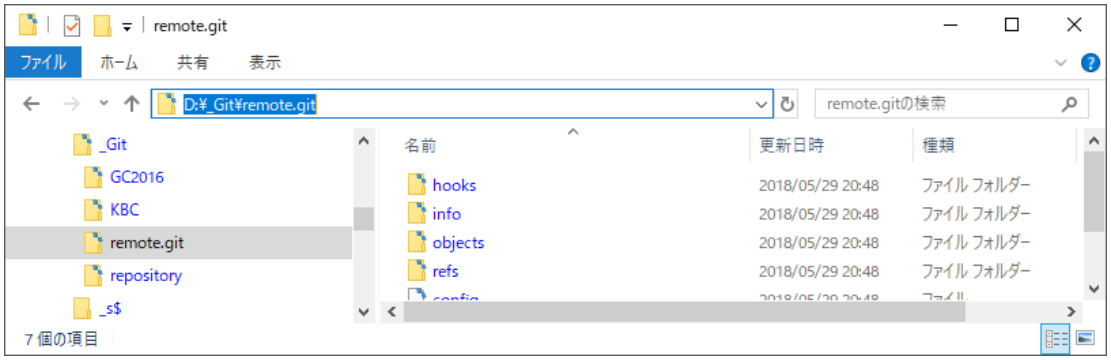
5.6. push 後の状態

リモート・リポジトリに push することによって、自分が今までローカル・リポジトリで作業していた管理情報を、自分だけではなく、別の人が利用できるようになります。



5.7. リモート・リポジトリからの pull

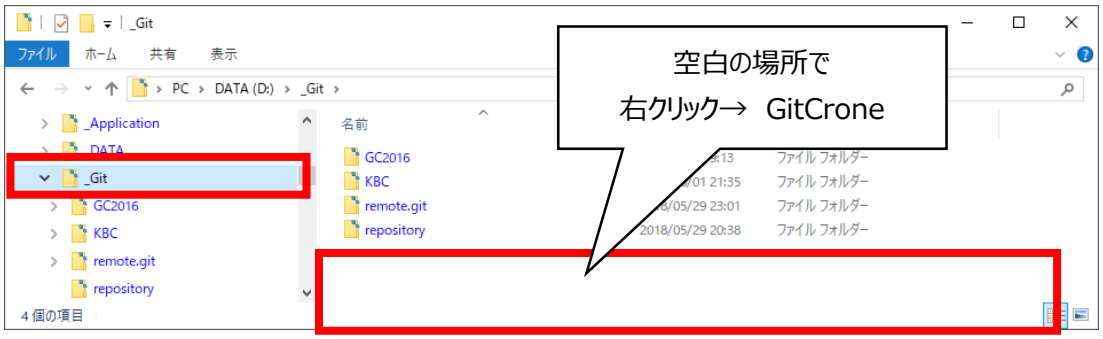
では、リモート・リポジトリから、情報を読み取ってみましょう。まず現在の状態は下記の通りとします。



remote.git　：リモート・リポジトリ
repository　：ローカル・リポジトリ

この状態で。ローカル・リポジトリからリモート・リポジトリに push が終了したとします。

この状態で、ひとつ上の階層　_Git　をクリックします。



ウィンドウが開いたら、そのまま “OK” でクローンができると思います。

5.8. トラブルの発生

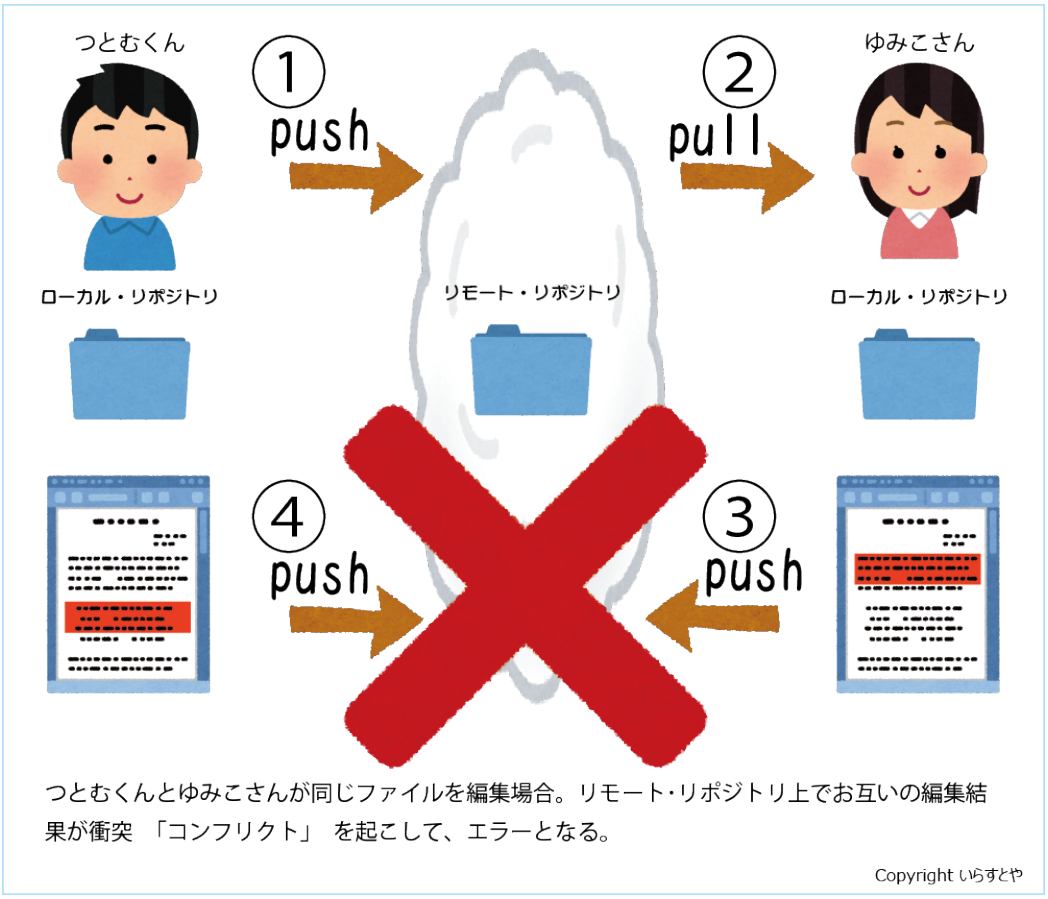
個人で単独のリポジトリを利用している場合には、ほとんど問題なく利用できるのですが、リモート・リポジトリのように複数の人間で操作する場合いろいろとトラブルが発生します。

よくあるトラブルとその解消方法について勉強します。

5.9. コンフリクト(conflict：競合)

コンフリクトとは、コミットした時点での、リモート・リポジトリの内容が変化してしまうなどで、正常にコミットできなくなることを言います。

よくある例



つとむくんとゆみこさんが同じファイルを編集場合。リモート・リポジトリ上でお互いの編集結果が衝突 「コンフリクト」 を起こして、エラーとなる。

Copyright いらすとや

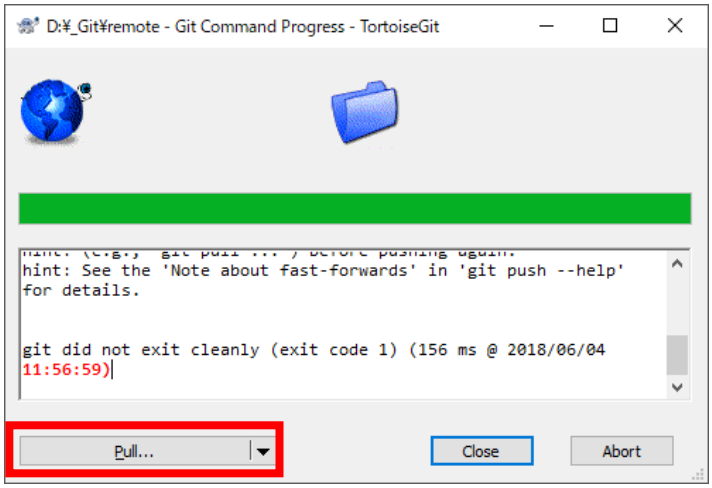
上図の場合、③の push では問題ありませんが、④の push では、つとむくんの利用しているローカル・リポジトリが把握しているリモート・リポジトリの状態が変化していますので、エラーになります。

5.10. コンフリクトの解消

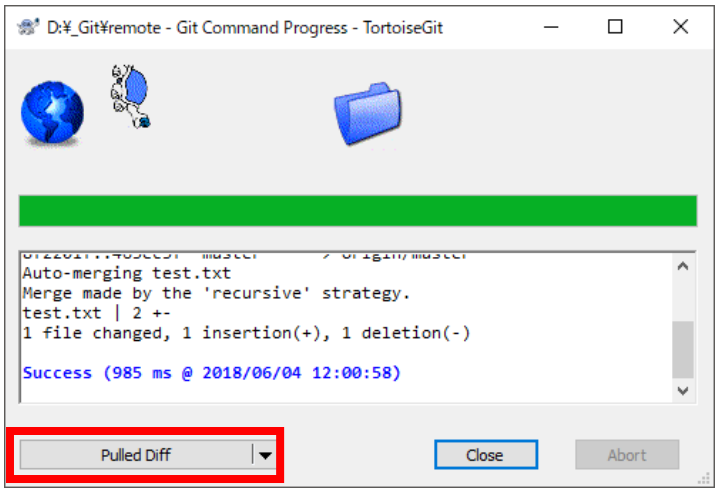
5.10.1. まず pull

push した時点でエラーが発生した場合、すでにリモート・リポジトリの状態が変化していることが問題になります。そこで慌てず、リモート・リポジトリより pull して最新の状態を把握します。

pull した場合、編集箇所が上書きされそうですが、そこは Git が調整してくれます。

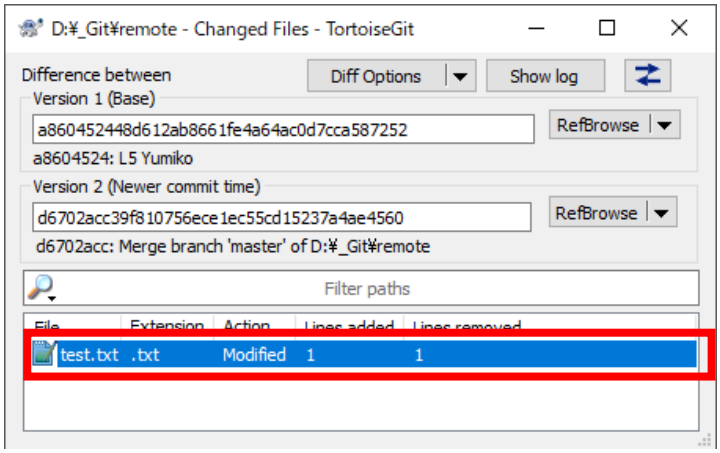


そうすると、このようなウィンドウが開きます。



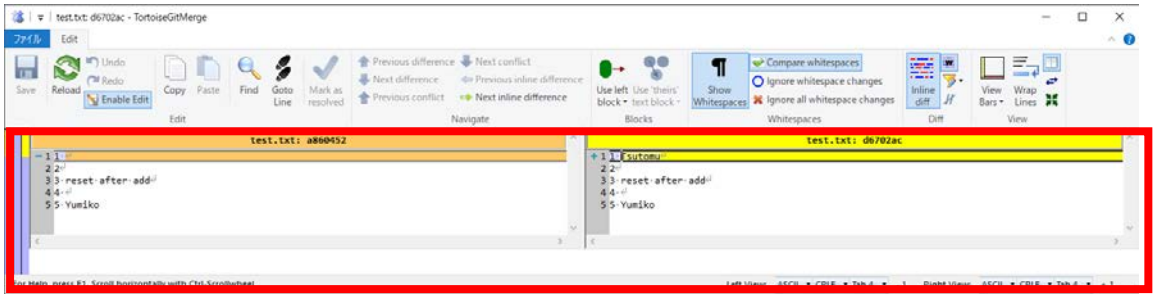
5.11. コンフリクトの状態表示

先ほど開いたウィンドウの、“Pulled Diff” ボタンを押すと現在の状態を示す下記のウィンドウが開きます。そこで、対象のファイルをダブルクリックし状況を見てみます。

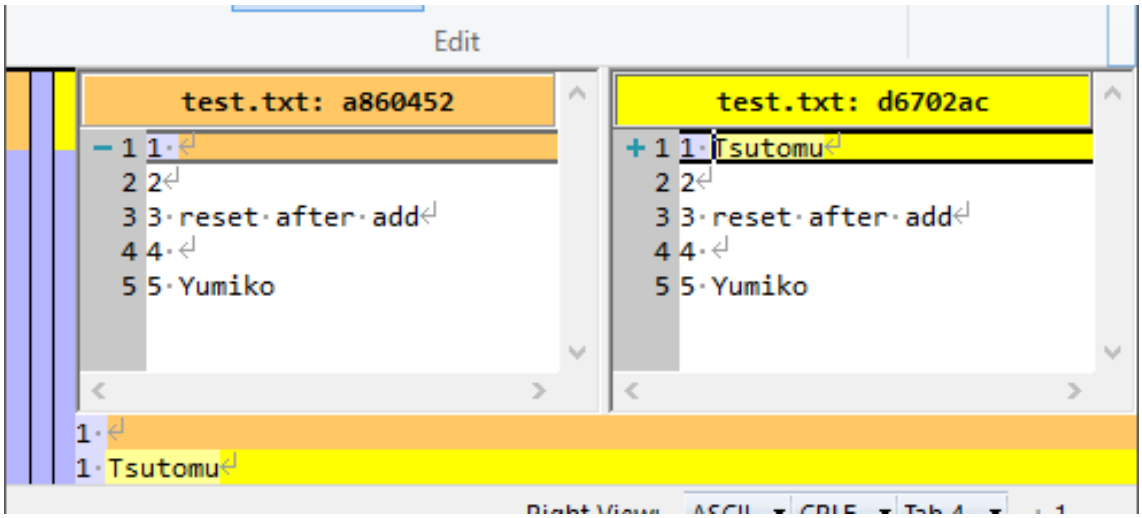


5.12. コンフリクトの解消

先ほどのファイル名クリックで下記のように状態を比較して表示する画面が開きます。(Diff)



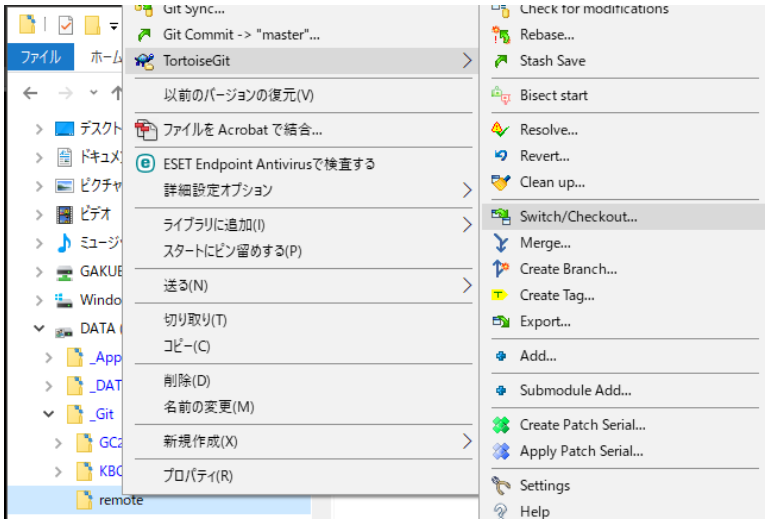
赤枠を拡大すると下記のようになります。



左側がリモート・リポジトリの内容で、右側が自分のローカル・リポジトリの内容となります。ここで最終的にどうするのか編集し、保存します。

5.13. 編集後の内容をリモート・リポジトリ push

後は競合を解決（Marge）して、リモート・リポジトリ に push します。



6. ファイル共有でのリモート･リポジトリの利用

6.1.1. GitHub（https://github.com/）

GitHub とは git を利用したソフトウェア開発プラットフォームで、ソースコードをホスティングしている。

6.1.2. GitHub の登録

GitHub（https://github.com/）トップページより、サインイン登録することで利用できる。
ここで利用したユーザ ID、PW は後ほど利用するので記録しておくこと。
※登録手順は省く

6.1.3. Git でのリモートリポジトリの作成

リモートリポジトリの作成には大きく 2 つの方法がある。

1） GitHub 上でリポジトリを直接作成する。

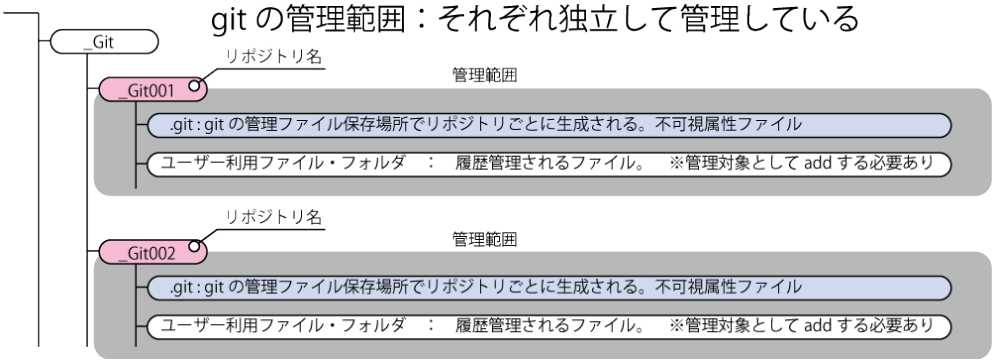
2） ローカルのリモートリポジトリを GitHub に Push する。
今回は 2）の方法でリモート･リポジトリを Git 上に作成する。

6.1.4. 教材配布

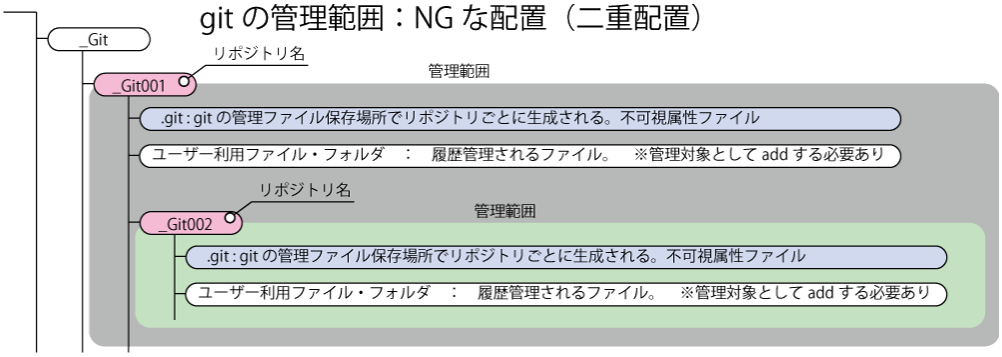
下記の教材配布先にある Lesson001.zip を適当な場所にダウンロードする。
https://github.com/JunSaiki/KBC/tree/master/GC2017/prog

6.1.5. 教材ファイルの展開

ダウンロード後、リモート･リポジトリの接続先が異なるため、今までとは別の場所に展開する。



下記のようにリポジトリの中にリポジトリを作成すると非常に煩雑になるため注意する。



6.1.6. リポジトリの作成

展開したディレクトリにリポジトリを作成し、コミットしておく。

6.2. gitHub への push

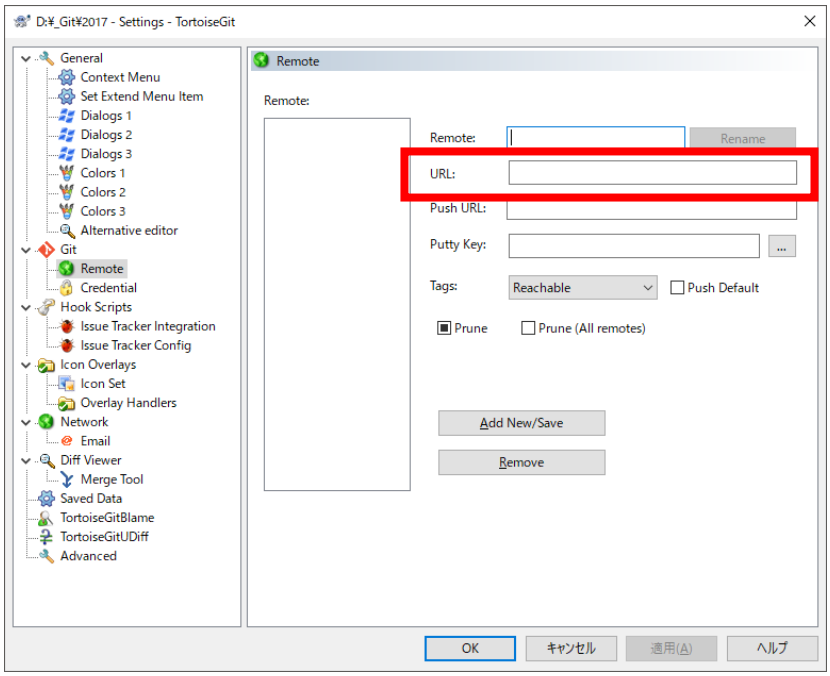
リポジトリを作成し、コミット作成後 gitHub への push を行う。行う方法は 5.3 参照

6.3. 接続先設定

5.4 の通りリモートの接続先設定画面で url に、自分の gitHub の url を登録する。

佐伯の場合

https://github.com/JunSaiki/　※赤文字部分が個別の ID

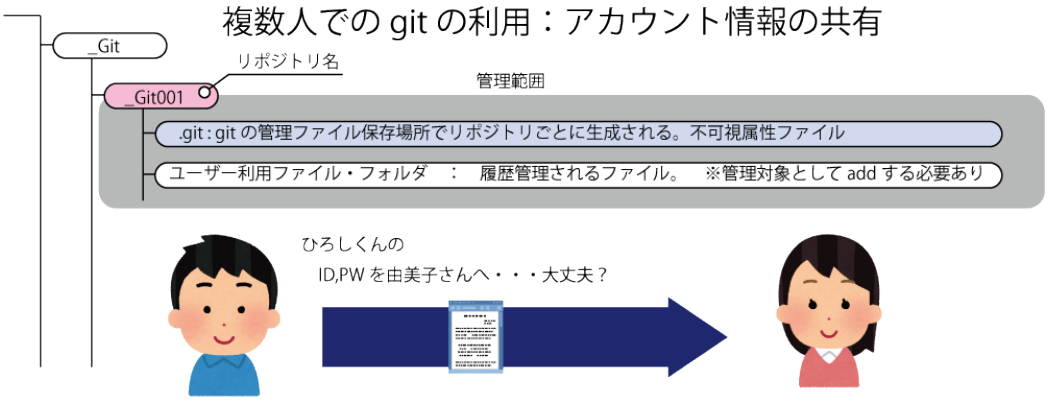


設定し、push するとアカウント情報を聞かれるウィンドウが開くため。、個別のアカウント情報を入力すると push できる。

これで、今までと同様に、gitHub を利用したリモート・リポジトリの環境を利用することができる。

7. ユーザ管理

個別で、このリポジトリを利用する限りユーザを意識することはないが、このままでは複数の人間とアカウント情報をシェアする羽目になる。



gitHub では、ひとつのリモート･リポジトリを複数のユーザで利用する機能が備わっている。その機能を ”Organaization：オーガナイゼーション” という。

7.1. Organaization の利用

gitHub トップページで、右上のユーザアイコンをクリックし、出てきたプルダウンメニューから "settings" を選択します。そうすると

7.2. コンフリクトが発生しない運用方法

7.3. チェックアウト

7.4. ブランチ

7.5. マージ

8. 実際の運用

9. VisualStudi での Git 利用

http://www.atmarkit.co.jp/ait/articles/1607/14/news020.html

■用語集