バージョン管理システム入門

（初心者向け）

TortoiseGitの基礎勉強
～TortoiseGitによるバージョン管理を使う～

はじめに

tracpath

ようこそ、分散バージョン管理「Git（ギット）」のGUIクライアントである「TortoiseGit」の基礎勉強です。TortoiseGitは、Gitのクライアントツールで Windowsのシェルエクステンションとして機能します。専用のGUIを備えており、CUIのGitより、直感的で簡単に扱うことが出来ます。Subversionクライアント用の「TortoiseSVN」を利用したことがある方は似たような使い勝手を提供しているので比較的簡単に使うことが出来ます。

「TortoiseGitの基礎勉強」では、Windows7環境でTortoiseGitの基本的な使い方をご自身のマシンで動作させます。TortoiseGitはこのチュートリアルで説明している以外にも多くの機能を備えたパワフルなバージョン管理システムです。使いこなすためには文章を読むだけではなく実際に試すことがとても重要です。

このドキュメントにならってチュートリアルを実行することで、基本的なTortoiseGitの使い方を学習することができます。

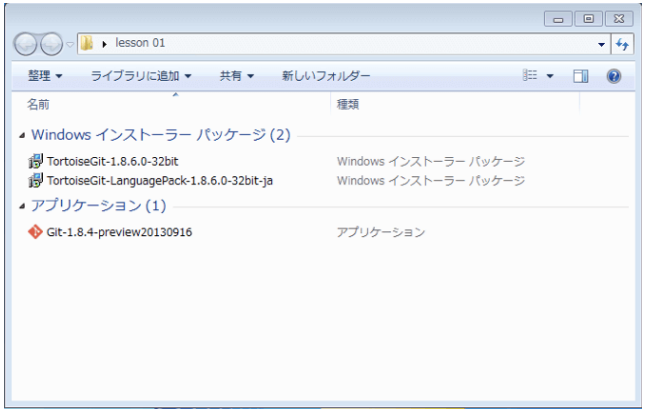
このチュートリアルでは、TortoiseGitの日本語版（TortoiseGit-1.8.6.0）を利用して進めていきます。

レッスン１．インストール

tracpath

最初にTortoiseGitが利用できるように環境を作ります。TortoiseGitをWindows環境で動作させるために必要なソフトウェアがあります。Windows向けのGitである「msysgit」をダウンロードしてください。TortoiseGitを利用するために必須ですので事前にインストールしておく必要があります。このチュートリアルでは日本語化したTortoiseGitを利用するので3つのファイルをダウンロードしました。

- msysgit
  [Git-1.8.4-preview20130916.exe]
- TortoiseGit
  [TortoiseGit-1.8.6.0-32bit.msi]
- TortoiseGit 日本語言語パック
  [TortoiseGit-LanguagePack-1.8.6.0-32bit-ja.msi]



レッスン１．インストール

msysgit

msysgit からダウンロードします。2013 年 12 月 4 日時点で最新版である「Git-1.8.4-preview20130916.exe」を利用します。

TortoiseGit

公式サイトのTortoiseGitのダウンロードから2013年12月4日時点で最新版である「TortoiseGit-1.8.6.0-32bit.msi」を利用します。32bit版と64bit版があるのでご自身の環境に合わせてダウンロードして下さい。

TortoiseGit日本語言語パック

公式サイトのTortoiseGitのダウンロードから2013年12月4日時点で最新版である「TortoiseGit-LanguagePack-1.8.6.0-32bit-ja.msi」を利用します。32bit版と64bit版があるのでご自身の環境に合わせてダウンロードして下さい。

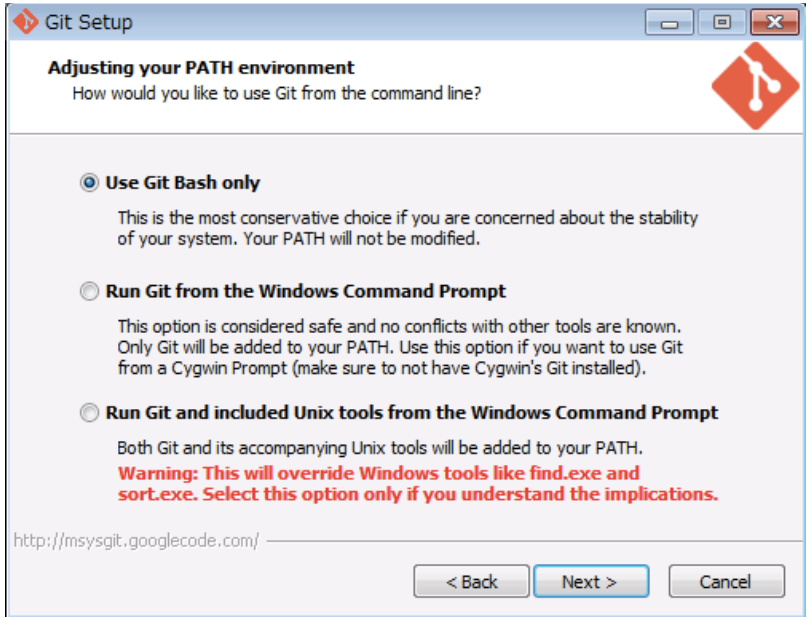
レッスン１．インストール

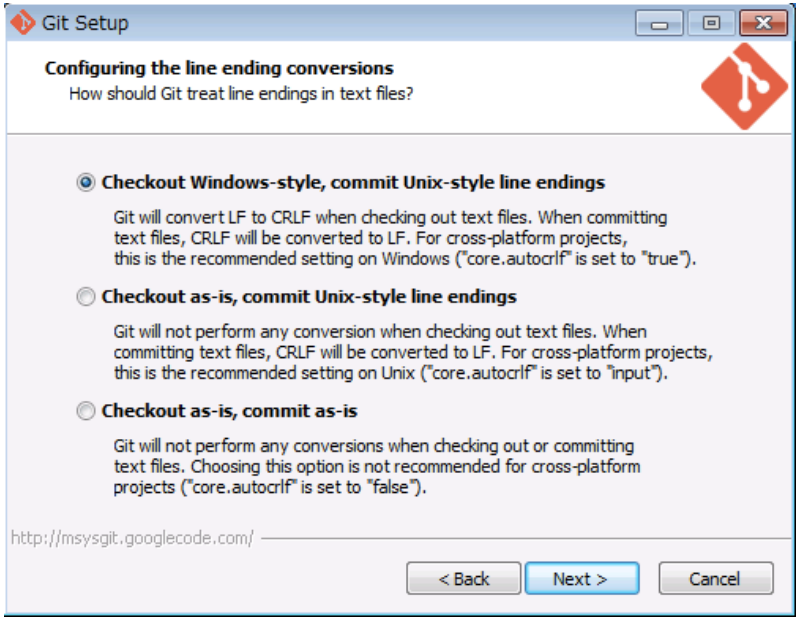
tracpath

msysgitのインストール

Windows版Gitである、msysgitのインストールは「Next」を押下すれば問題ありません。
いくつか選択肢が表示される場合がありますが、ここでは何も変更せずに「Next」を押下して下さい。

レッスン１．インストール





レッスン１．インストール

tracpath

msysgitの環境設定

スタートメニューからGit Bashを実行します。これからGitの設定を行います。設定はコマンドラインで実行します。環境設定の内容については説明しませんがGitを利用するために必要な手順として以下を実行します。

Gitはコミット時にユーザ名とメールアドレスを記録します。この設定は必須です。自分が利用するメールアドレスと名前を設定します。

- user.email … コミット時に利用されるメールアドレス
- user.name … コミット時に利用される名前

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

日本語の文字化け対策で設定します。

```
git config --global core.quotepath false
```

改行コードをGitが自動変換することを無効にします。

```
git config --global core.autocrlf false
```

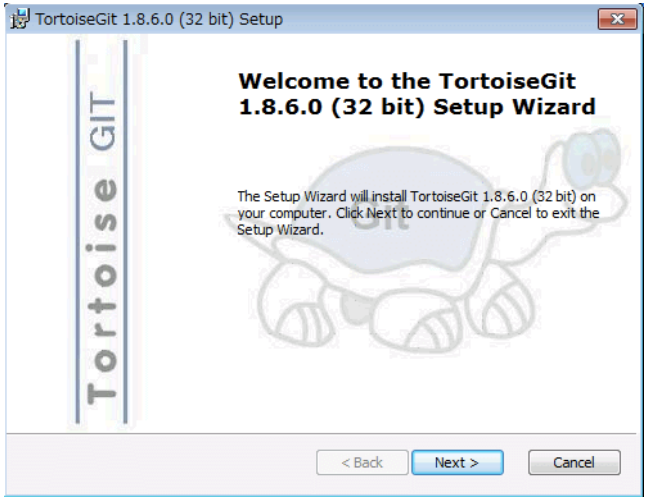
pushコマンド時にブランチ名やタグ名を指定しない場合の標準動作を設定します。（安全のため）

```
git config --global push.default upstream
```

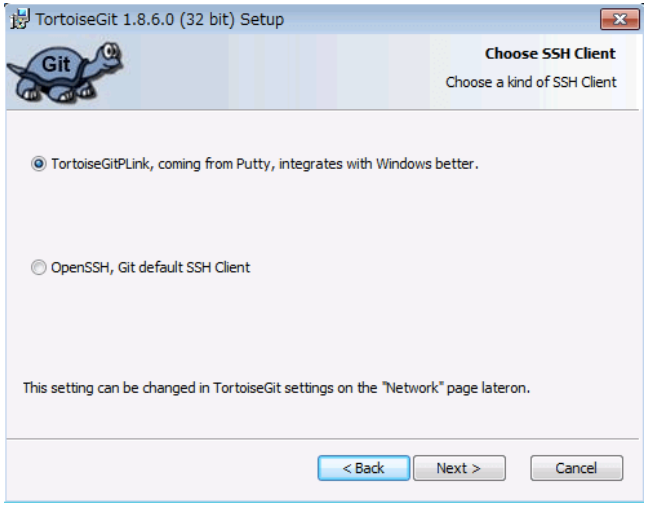
レッスン１．インストール

TortoiseGitのインストール

GitのGUIクライアントであるTortoiseGitのインストールを開始します。



インストールでは「Next」を押下すれば問題ありません。いくつか選択肢が表示される場合がありますが、ここでは何も変更せずに「Next」を押下して下さい。



レッスン１．インストール

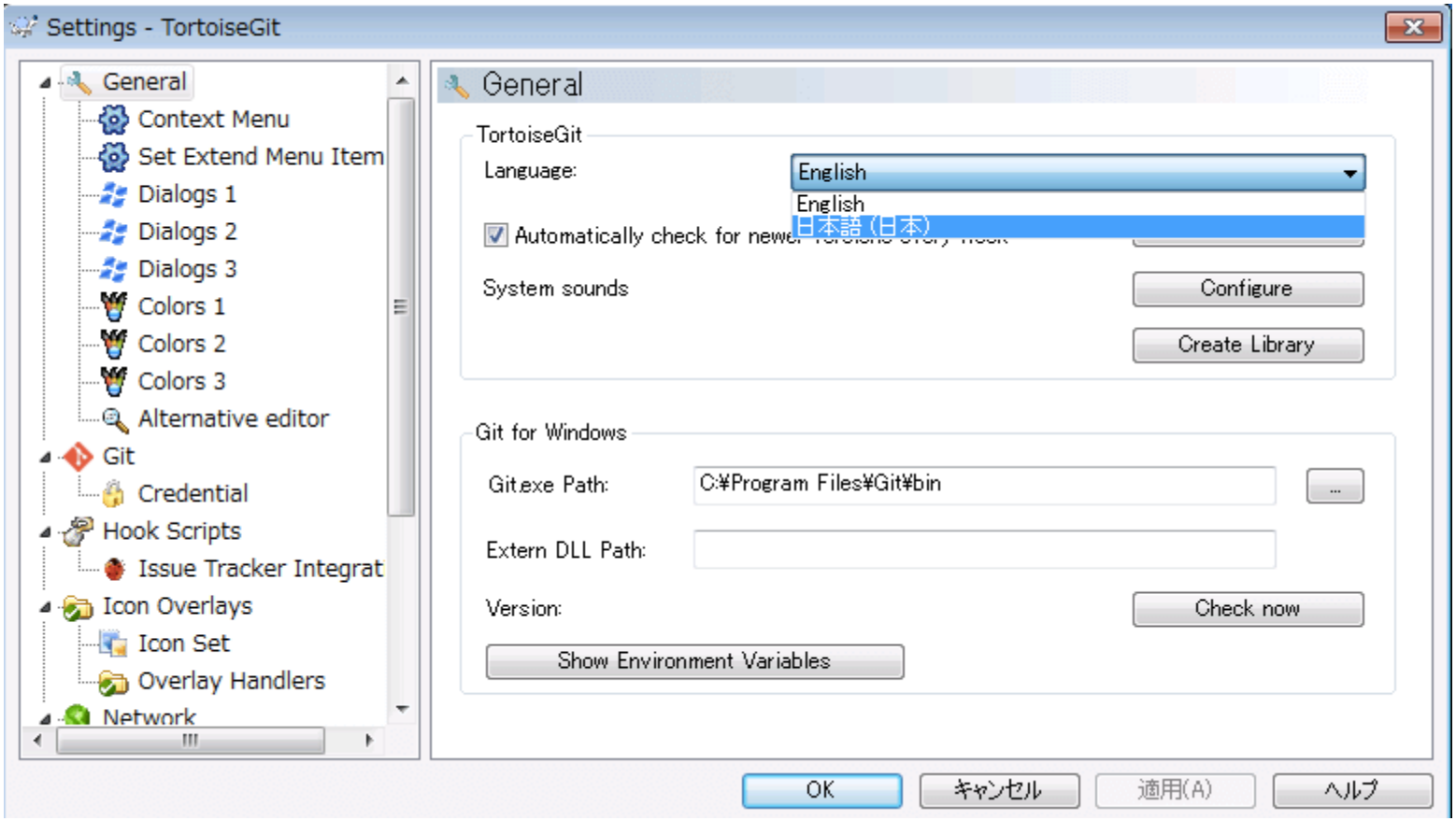
TortoiseGit日本語言語パックのインストール

インストーラーを実行するだけでインストール作業は完了です。
日本語言語パックのインストール後に設定を行う必要があります。

レッスン１．インストール

適当なフォルダを右クリックして下さい。右クリックメニューにある「TortoiseGit -> Setting」をクリックします。「General」の「Language」を「日本語（日本）」にします。

これで、WindowsでTortoiseGit（日本語版）が動作する環境準備が整いました。



レッスン２．動作確認

インストールが完了したら動作の確認をしてみましょう。
TortoiseGitはWindowsのシェルエクステンションです。メニューを表示するためには適当なフォルダを右クリックして、メニューを開きます。

右クリックメニューの「TortoiseGit -> TortoiseGitについて」をクリックします。

次から早速TortoiseGitをつかったバージョン管理を行います。

レッスン３.リポジトリ用のディレクトリを作成

これでTortoiseGitが利用できるようになりました。まだ開発に入ることは出来ません。TortoiseGitでソースコードのバージョン管理を行うためにはリポジトリを用意する必要があります。

リポジトリ：Gitで管理されるソースコードやファイルを格納する場所のことです。この格納場所を作成しなければ開発に進むことはできません。通常リポジトリの作成作業はプロジェクトの最初に1回だけ実行します。

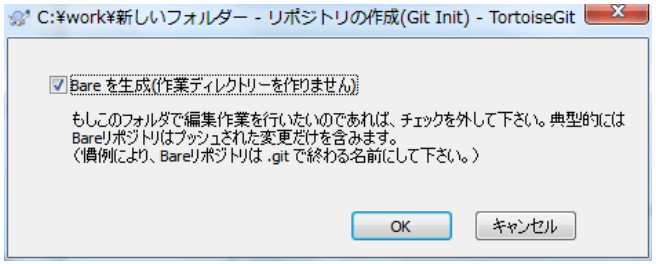
注意点 ：

- **Gitの基礎勉強で作成したリポジトリを利用してもよいでしょう。**
- **すでにGitのリポジトリがある場合、このレッスンは飛ばして下さい。**
- **TortoiseGitはリポジトリを作成する機能があります。このチュートリアルではリポジトリの新規作成を行います。**

それでは、リポジトリを作成します。今回は以下の構成にします。

リポジトリ用: c:¥work¥tgit-repo

「tgit-repo」フォルダを作成します。tgit-repoを右クリックしてTotroiseGitメニューの「Gitここにリポジトリを作成（Y）」をクリックします。



レッスン３.リポジトリ用のディレクトリを作成

ここでいきなり「Bare を生成（作業ディレクトリーを作りません）」という初めて使う人にとっては意味の分からないメッセージボックスが出てきました。今回のチュートリアルではtgit-repoを共有リポジトリとして利用する予定ですので「Bare を生成」にチェックを付けて下さい。

警告 ： 今回のチュートリアルでは「tgit-repo」を共有リポジトリとして利用する予定ですので「Bareを生成」にチェックを付けて下さい。

レッスン３.リポジトリ用のディレクトリを作成

「Bareを生成（作業ディレクトリーを作りません）」とは何か

Bareを生成にチェックを入れるとベアリポジトリを作成することを意味します。
ベアリポジトリは管理用のリポジトリであり、作業用ファイルが含まれません。このリポジトリは他からcloneされたり、pushされる対象になります。

ノート：

ベアリポジトリを作成するのはどのようなときか?

**開発チーム共通の中央リポジトリとしてベアリポジトリを作成し、開発側は自分の端末にcloneして開発を進めローカルでcommit後に中央リポジトリにpushする流れが一般的です。
Subversionを利用している方には中央リポジトリ（=共有リポジトリ）は唯一のリポジトリで有り、開発者はチェックアウトし自分の環境に作業コピーを作成します。開発チームは常に中央リポジトリに対してコミットを実施します。**

分散バージョン管理であるGitでは、リポジトリから「clone」したら、それも同じリポジトリとなりcloneしたリポジトリ（ローカルリポジトリ）でバージョン管理ができます。自身の開発を適宜共有リポジトリに反映（push）することで開発チームの開発が進みます。

このことから、ベアリポジトリを作成するタイミングはサーバ側のリモートリポジトリとして利用する場合となります。

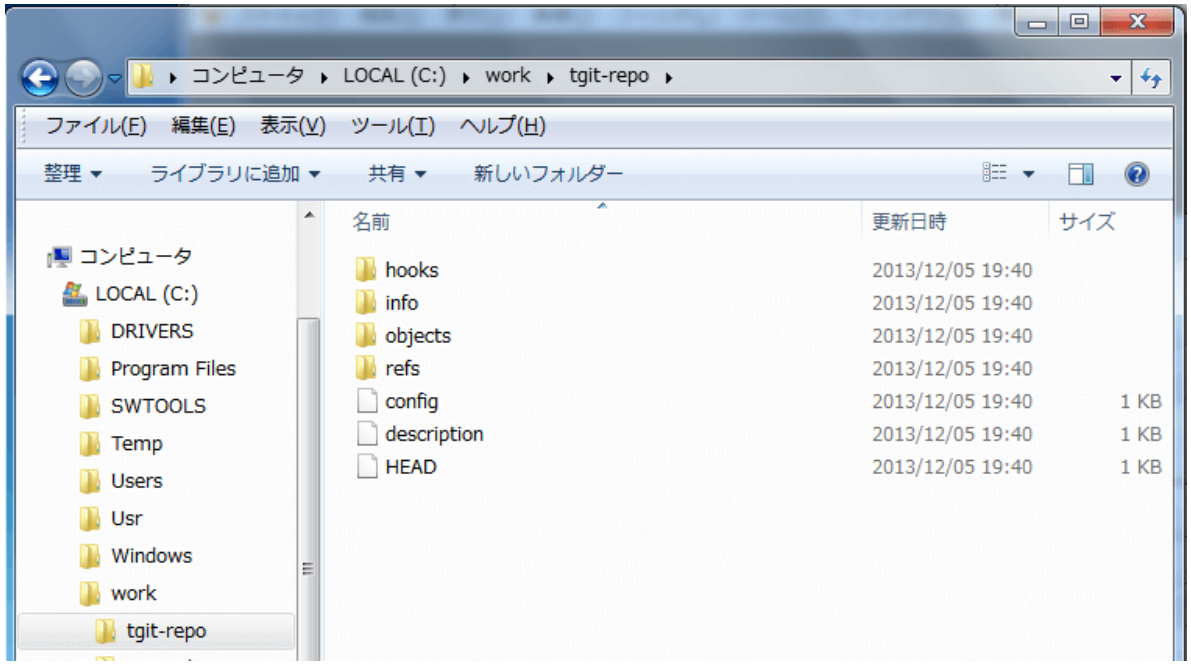
レッスン３．リポジトリ用のディレクトリを作成

これは、gitコマンドで

```
$ git --bare init --shared
```

と同じ操作になります。

今回のチュートリアルでは共有リポジトリ（C:¥work¥tgit-repo）のcloneを作成して、ローカルリポジトリを使って開発作業を進めるため、チェックを付けてください。

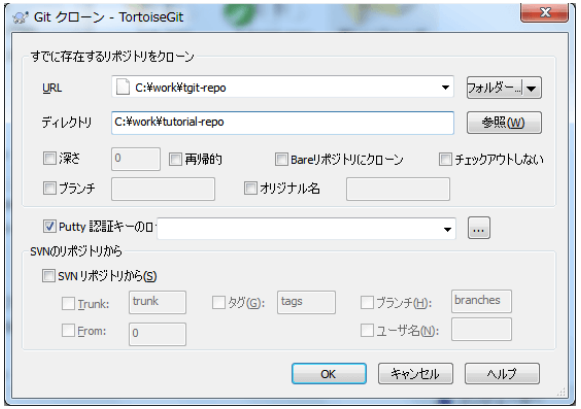


レッスン４．クローン

レッスン.3で作成した中央リポジトリを利用して開発を進めるとき、中央リポジトリのクローンを自身の環境にコピーして開発を進めます。

ノート：
開発メンバーで一緒に開発を進めることが想定されるとき、共有リポジトリを利用する場合が多いでしょう。共有リポジトリを利用して開発を進める場合、クローン（clone）して作業ディレクトリをローカルに作成します。
Subversionなどは「チェックアウト（checkout）」を実行していましたが、分散バージョン管理「Git」はクローン（clone）します。このクローンはサーバが保持しているデータをほぼすべてローカルにコピーします。これはプロジェクトのすべてのファイルのすべての履歴が手元にコピーされることを意味しています。他の開発者に影響を与えずにブランチを作成したりできる分散管理バージョンのメリットです。

右クリックして「Git クローン(複製)」を選択します。



レッスン４．クローン

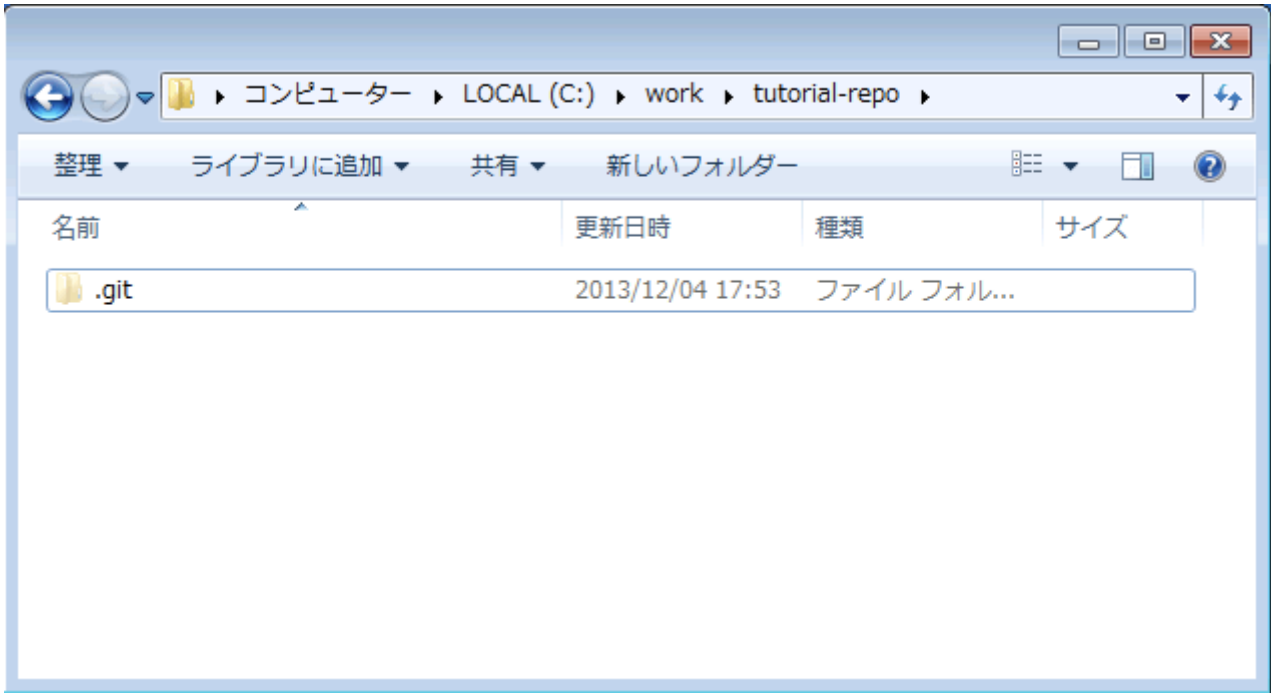
URL： C:¥work¥tgit-repo
ディレクトリ：C:¥work¥tutorial-repo

をしています。

- 「URL」はクローンする元のリポジトリを指定します。
- 「ディレクトリ」は自分が開発作業に利用するローカルのフォルダを指定します。

レッスン４．クローン

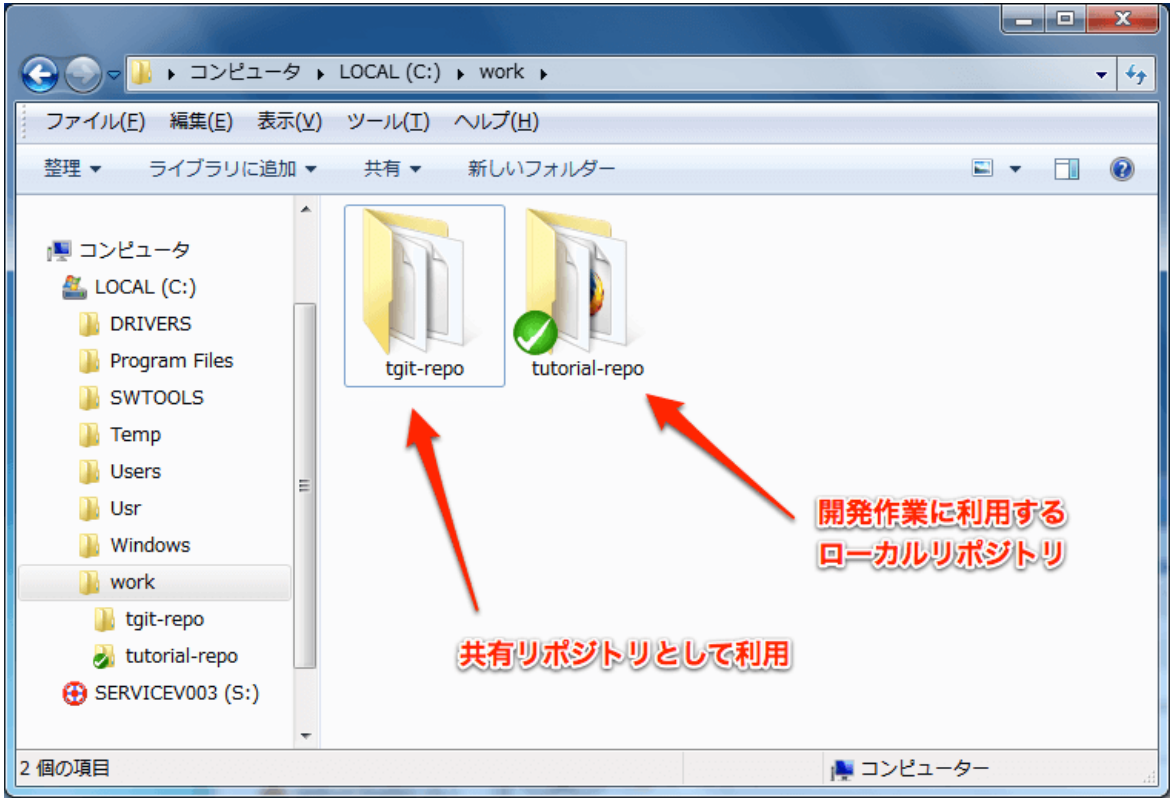
クローンによって作成された、作業用リポジトリをエクスプローラーで見てみます。まだリポジトリに何も登録していないためファイルはありませんが、Gitの管理用ファイル（隠し属性のファイル）が作成されていることが分かります。



これで共有リポジトリ（C:¥work¥tgit-repo）から作業用リポジトリ（C:¥work¥tutorial-repo）をクローンしました。これから作業用リポジトリで開発作業を進めていきます。

レッスン５.クローンしたリポジトリで作業開始

クローンが完了しました。これからこのローカルにクローンされたリポジトリで開発を進めていくことになります。クローン元になった共有リポジトリと分けて説明するため、クローンしたローカルのリポジトリをただの「リポジトリ」や「ローカルリポジトリ」と表記します。c:¥work¥tutorial-repoを見てみます。エクスプローラーから見るとフォルダに重なったチェックマークアイコンが付いているのが分かります。



レッスン５.クローンしたリポジトリで作業開始

これは、TortoiseGitのオーバーレイ表示機能です。アイコンの表示はリポジトリの状態を表しています。アイコンには以下の状態があります。ここではどのような状態を表示しているアイコンがあるのかを理解しておくだけでよいです。

このアイコンはリポジトリに変更が加えられ、差分があることを意味しています。

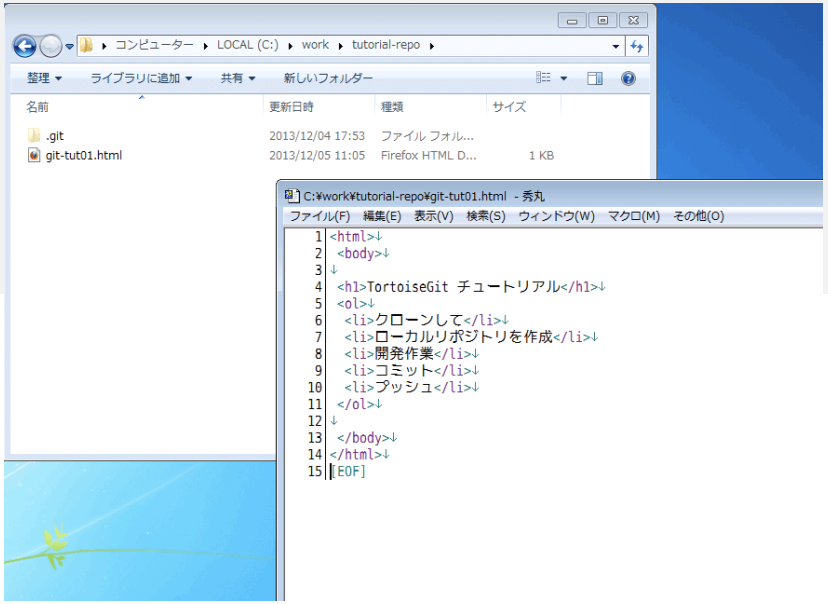
警告 ： ローカルリポジトリが通常以外のアイコンの場合、ローカルリポジトリに何かしらの変更が加えられていることを意味します。この変更を共有リポジトリに反映する作業「push（プッシュ）」を行うことで共有リポジトリとローカルリポジトリが同期します。

レッスン６．新しいファイルを作成・追加

tracpath

ローカルリポジトリで開発を始めましょう。リポジトリには何も登録されていない状態ですので、ファイルを追加してみます。使い慣れたエディタを使って、以下のファイルを作成します。
[git-tut01.html]を作成します。

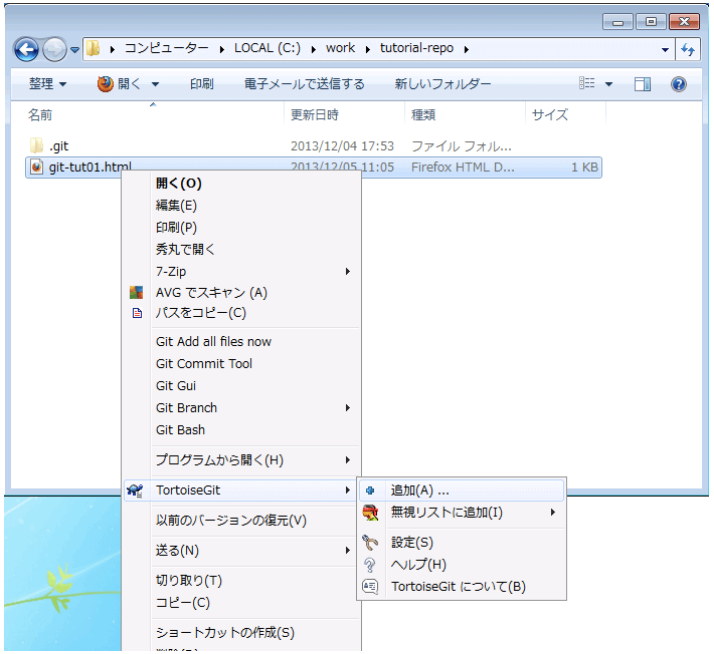
```
1  <html>
2    <body>
3
4    <h1>TortoiseGit チュートリアル</h1>
5    <ol>
6      <li>クローンして</li>
7      <li>ローカルリポジトリを作成</li>
8      <li>開発作業</li>
9      <li>コミット</li>
10     <li>プッシュ</li>
11   </ol>
12
13   </body>
14  </html>
```

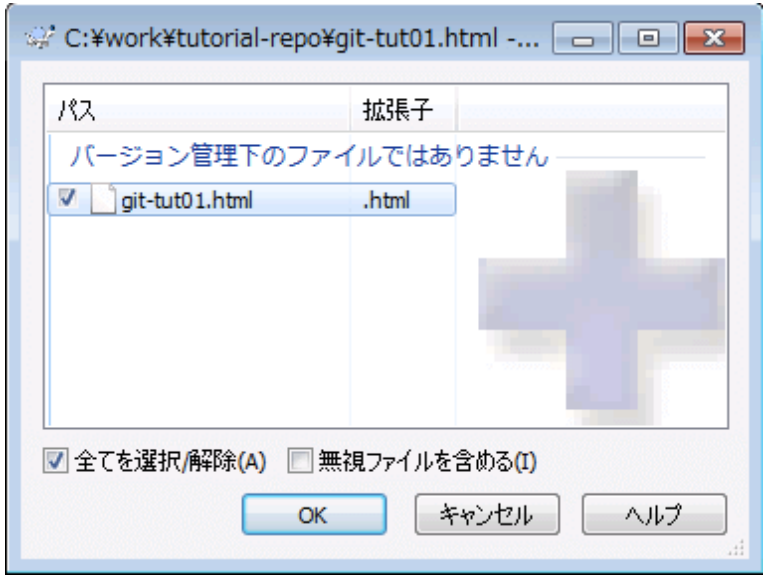


レッスン６．新しいファイルを作成・追加

tracpath

リポジトリにファイルを追加する操作を行います。[git-tut01.html] を 右 ク リ ッ ク し て TortoiseGit メニューから「追加」をクリックします。

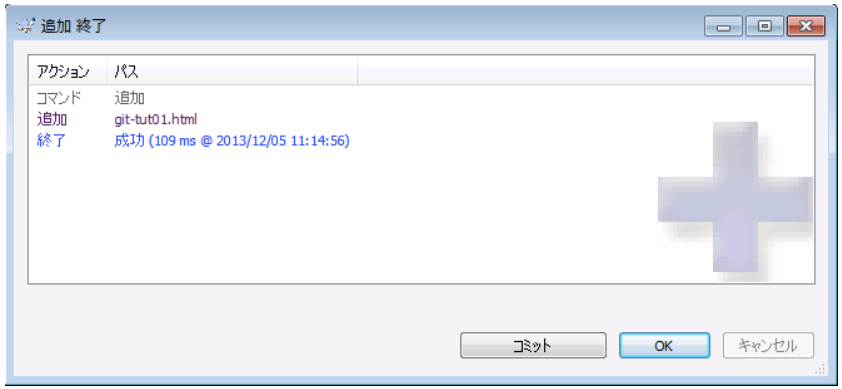




「追加」を選択すると確認ダイアログが表示されます。このダイアログでバージョン管理するファイルを選択したり、解除したりすることが出来ます。今回は1ファイルの追加のみですので「OK」を押下します。

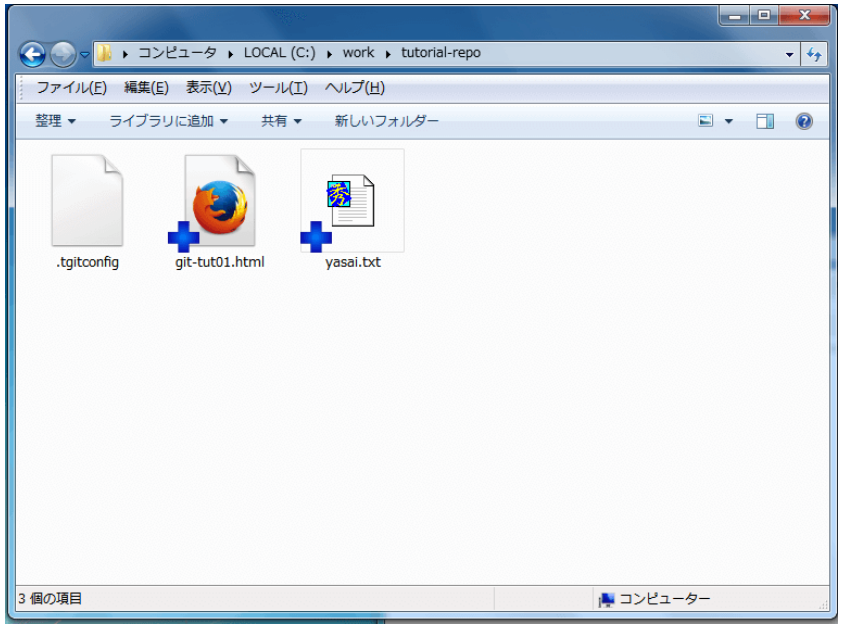
レッスン６．新しいファイルを作成・追加

「OK」押下でローカルリポジトリに「git-tut01.html」が追加されました。追加終了ダイアログに「コミット」「OK」があります。コミットについては後で説明しますので、「OK」をクリックします。



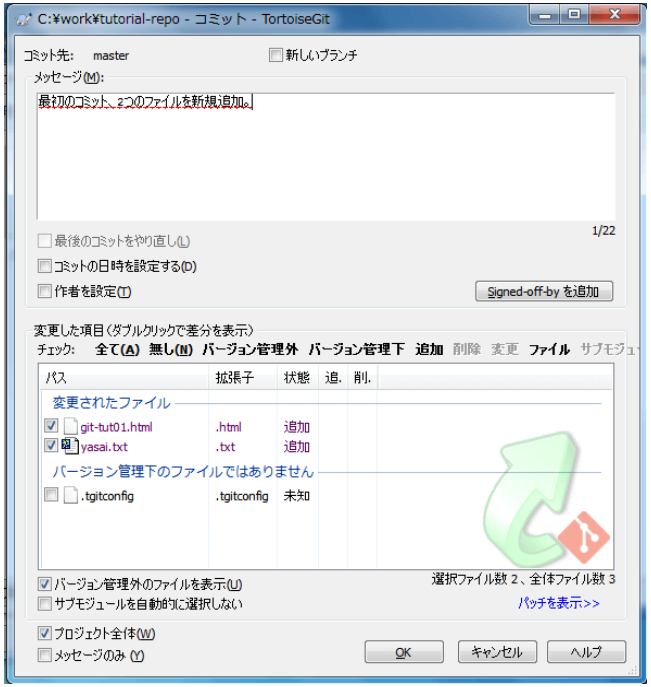
エクスプローラーのファイルアイコンの状態が「+」(追加)に変わっています。これで新しいファイルをローカルリポジトリに追加し、コミットする準備が出来ました。これだけでは、ローカルリポジトリに反映していません。コミットをしなければリポジトリに反映されません。

もう一つ「yasai.txt」という空のテキストファイルを作成してTortoiseGitの「追加」をしました。



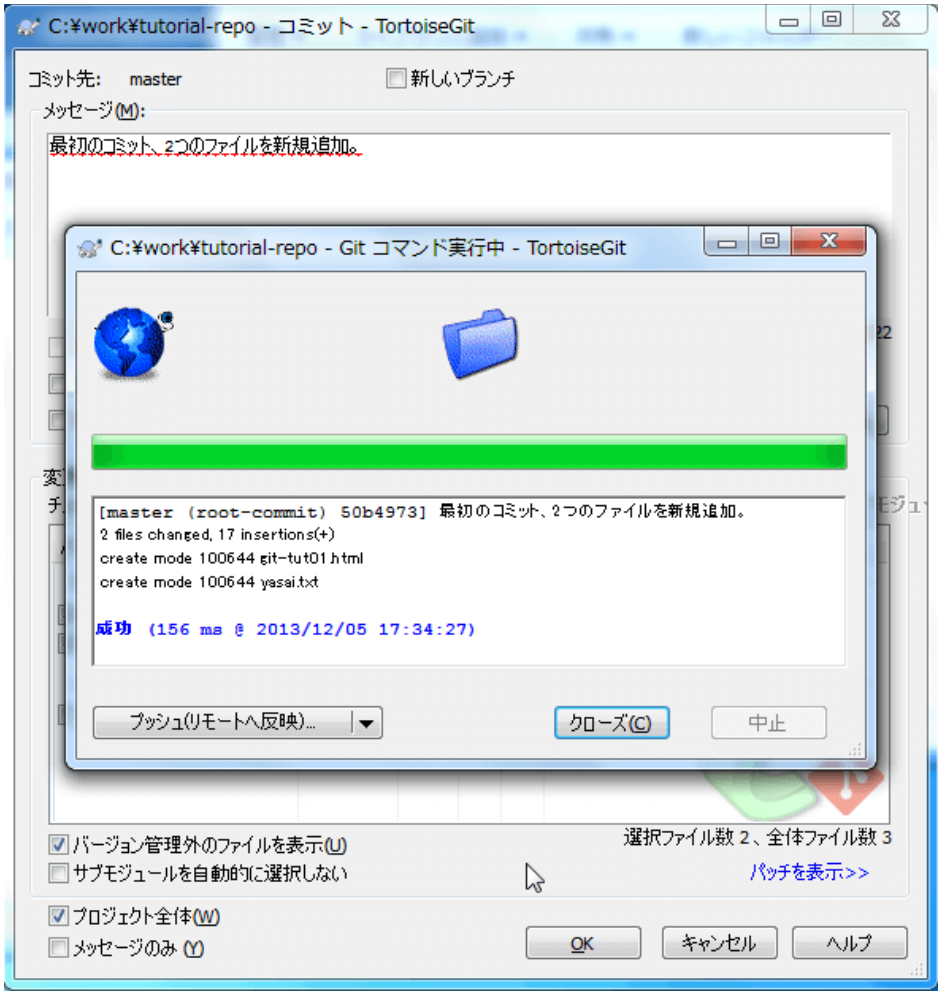
レッスン６．新しいファイルを作成・追加

それでは、コミットを行います。コミットもTortoiseGitメニューから操作します。tutorial.htmlとyasai.txtファイルがあるフォルダの空白部分を右クリックしてください。「Git コミット -> “master”」が表示されますので選択します。



コミット用のウィンドウが立ち上がりますので、コミット時の変更内容を入力します。

レッスン６．新しいファイルを作成・追加



コミット完了画面が表示されました。これで新しいファイルを追加し、ローカルリポジトリにコミットすることが出来ました。

レッスン６．新しいファイルを作成・追加

tracpath

ノート：

コミット時のコメントは何を残すべきか?

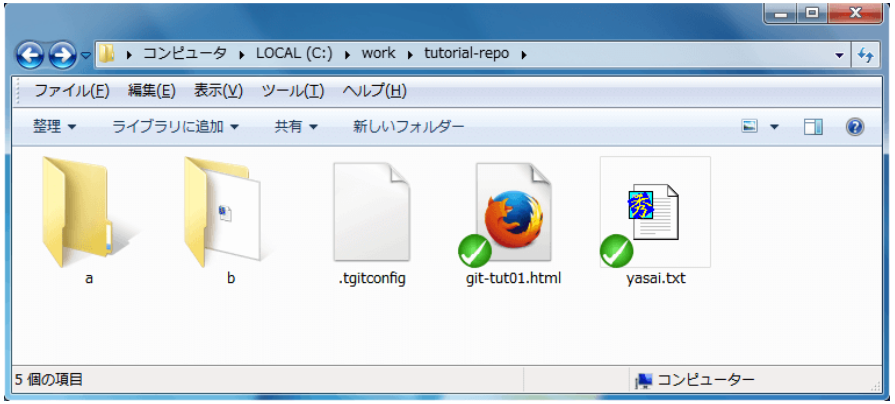
コミット時に変更内容を記録するメッセージ領域が表示されます。
コミットするときは必ずコミット内容を説明する内容を記述するようにすべきです。具体的に、機能追加なのか、バグ対応なのか、仕様変更による変更なのかを記載するべきです。
人間の記憶はとても曖昧で1,2ヶ月前の作業について何をやっていたかは覚えていても、日々のコードレベルの修正など覚えていない――覚えておく必要も無いので――場合がほとんどです。
コミット時に修正内容を記録しておくことは、チームみんなのためであり、将来の自分のためでもアルのです。

レッスン７. 新しいディレクトリを作成・追加

tracpath

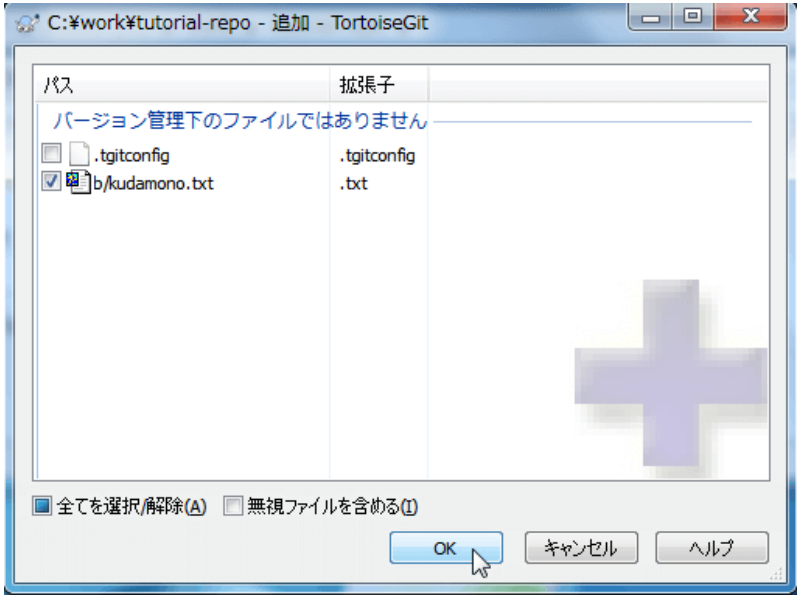
ディレクトリの追加やディレクトリに含まれるファイル群の追加をやっていきます。やり方はファイルの追加と同じです。

フォルダa,bとbフォルダに1つのテキストファイルを作成します。



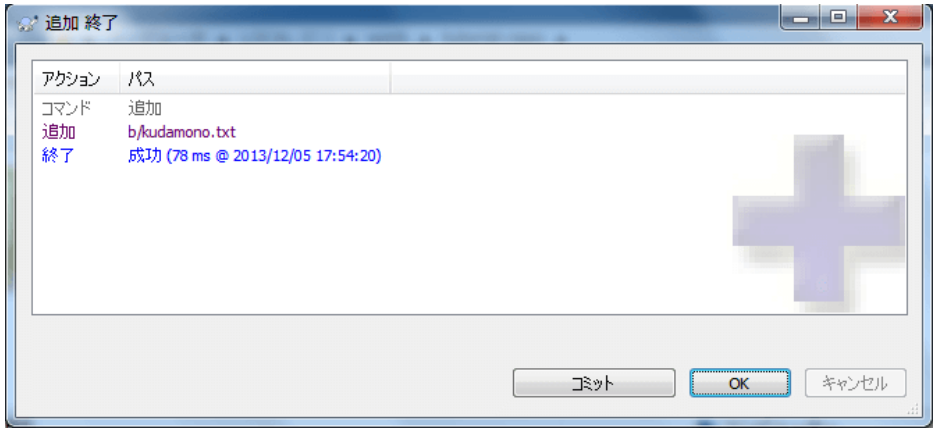
フォルダaとフォルダbをリポジトリに追加するため、フォルダの空白領域を右クリックして「追加」コマンドを実行します。フォルダbにはテキストファイルが含まれていることに注意してください。

「バージョン管理下のファイルではありません」と共にファイルリストが表示されます。

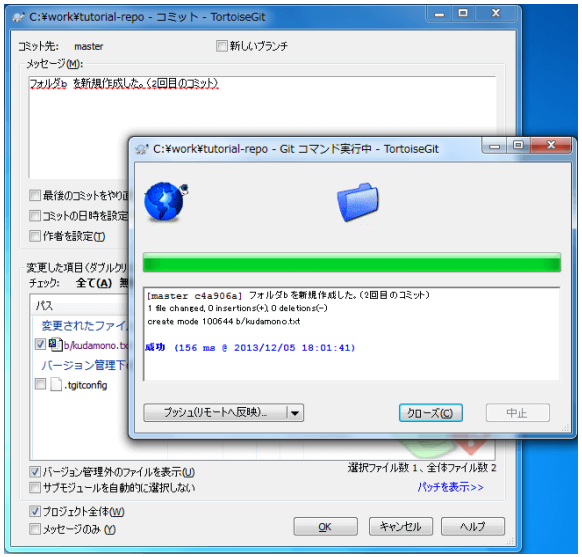


レッスン7. 新しいディレクトリを作成・追加

チェックボックスをチェックし「追加」します。



追加完了ダイアログで「コミット」を一緒にしてしまいます。コミット時のメッセージを忘れずに入力しましょう。



レッスン７. 新しいディレクトリを作成・追加

tracpath

警告 ：

新しいディレクトリを追加..

**新しいディレクトリを追加しましたが、「フォルダa」を追加することが出来ません。
Gitでは空ディレクトリを「追加」することができません。フォルダbのように適当なファイルがあれば追加することが出来ますが空ディレクトリは追加できません。このあたりの操作はSubversion（SVN）とは異なるので注意が必要です。**

正しくはGitで空ディレクトリを追加する方法はあります。調べてみて下さい。

レッスン８．ファイルを更新

それでは、ファイルの中身を編集してバージョン管理システムの機能を見ていきましょう。テキストエディタでgit-tut01.htmlを開きます。2行追加します。

```
<li>コミット時は更新内容をログとして記述</li>
<li>バージョン管理システムの更新</li>
```

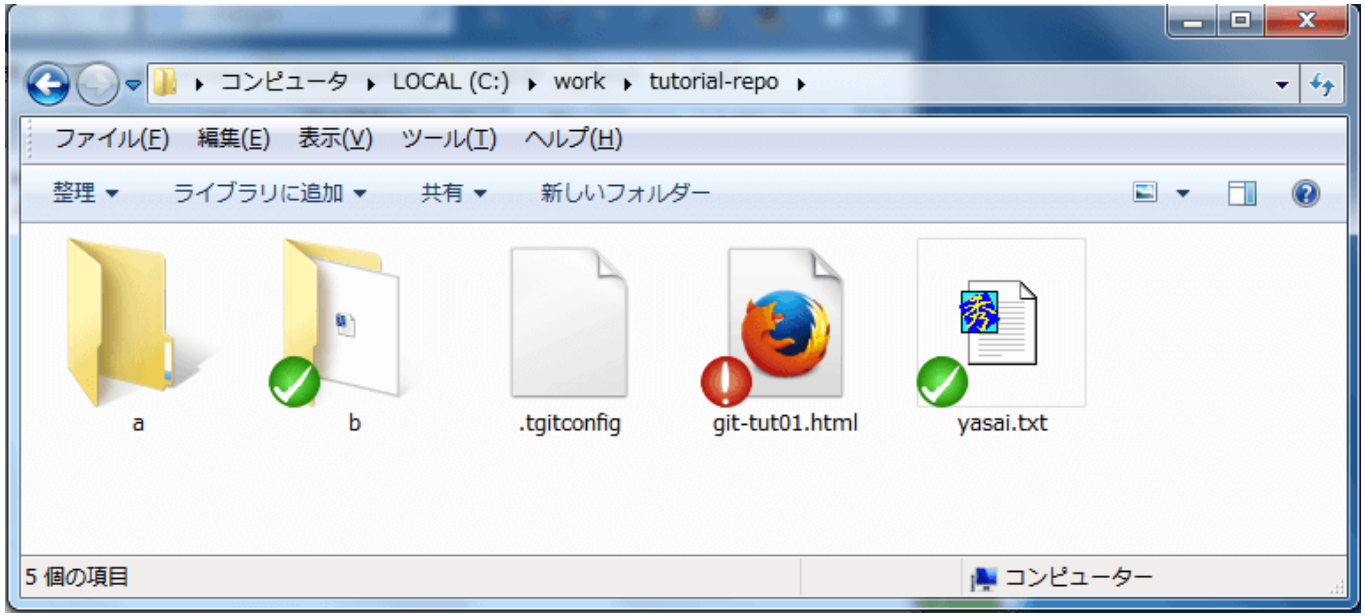
```
<html>
<body>

<h1>Subversion チュートリアル</h1>
<ol>
  <li>クローンして</li>
  <li>ローカルリポジトリを作成</li>
  <li>開発作業</li>
  <li>コミット</li>
  <li>プッシュ</li>
  <li>コミット時は更新内容をログとして記述</li>
  <li>バージョン管理システムの更新</li>
</ol>

</body>
</html>
```

レッスン８．ファイルを更新

ファイルを保存して閉じてください。変更したファイルに表示されるオーバーレイアイコンが変わっていると思います。



編集したファイルに「!」というアイコンが付いています。これはローカルリポジトリ内のファイルに変更があったことを示しています。続いてコミットを行います。

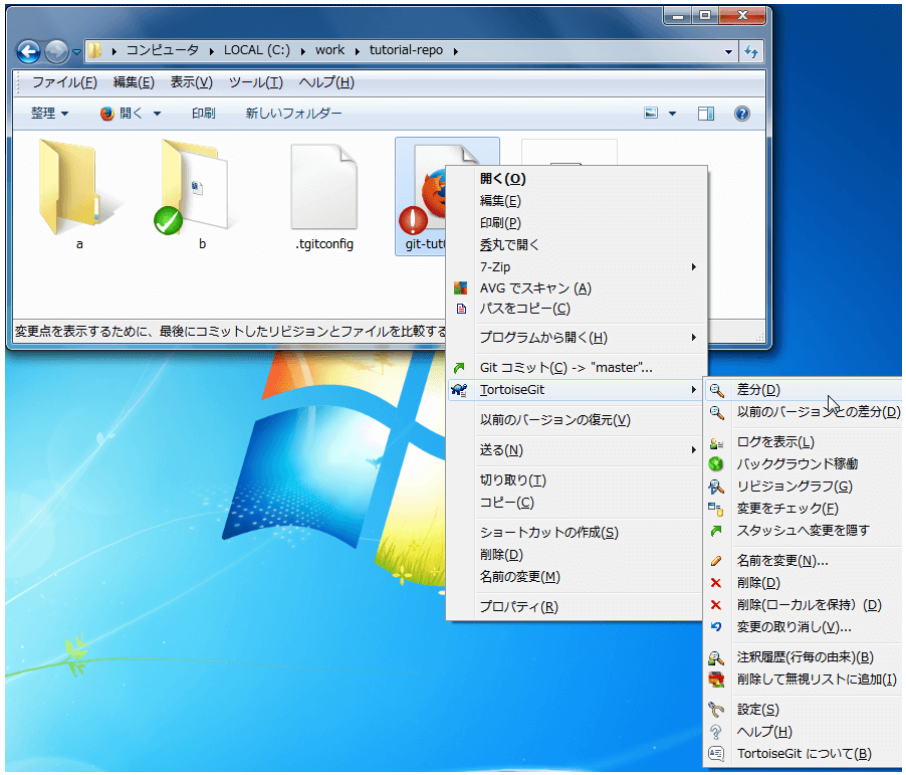
もう一度同じファイルに以下の行を追加します。

```
1 | <li>ファイルの更新</li>
```

レッスン８．ファイルを更新

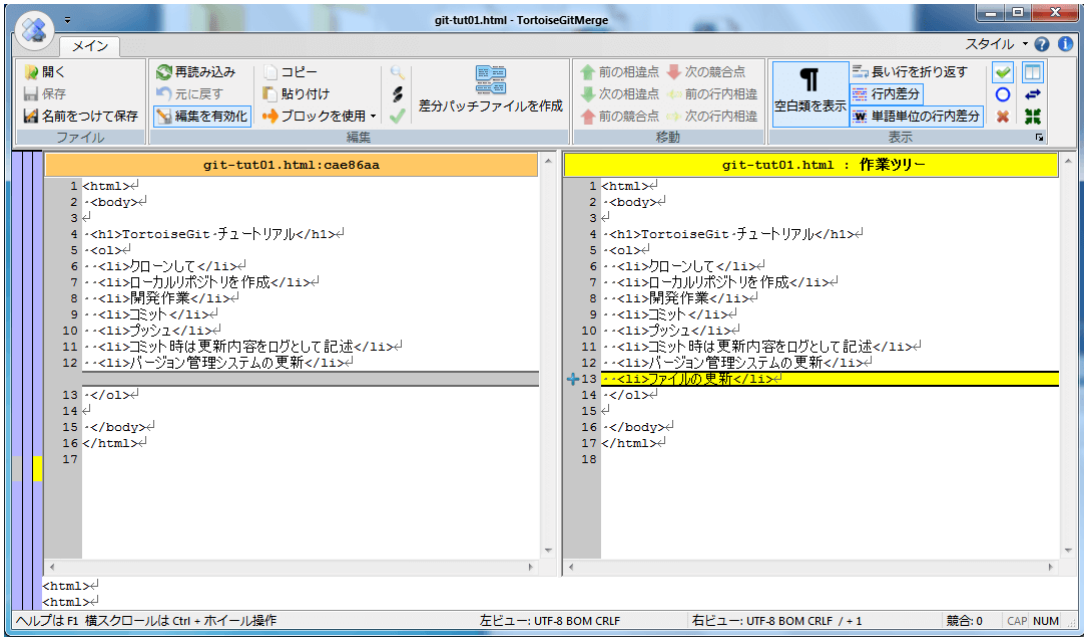
今編集しているファイルはリポジトリのファイルより新しい修正が追加されました。

コミットする前にその内容を確認します。「!」アイコンの付いたgit-tut01.htmlを右クリックして TortoiseGit メニューから「差分を表示」を選択します。



レッスン８．ファイルを更新

差分表示ツール[TortoiseGitMerge]が起動し、ファイルの差分を表示します。



修正内容を確認したらコミットします。更新内容のログを忘れずに入力します。

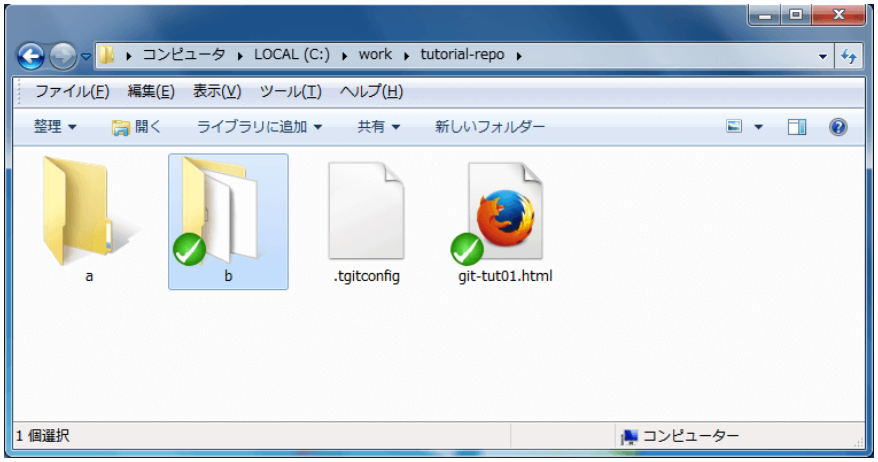
警告 ： TortoiseGitMergeで差分表示が可能ですが、外部ツールを差分表示に利用することが可能です。よく使われるツールとして、WinMerge（日本語版）があります。WinMergeを利用する場合、インストール後にTortoiseGitの設定 -> 差分ビューアーを変更してください。

レッスン９．ファイルを移動

ファイルの移動はSubversion（SVN）と違いますので注意が必要です。Gitは「ファイルの移動」は履歴として残りません。Gitのコマンドでgit mv<old-filename><new-filename>がありますがこれは

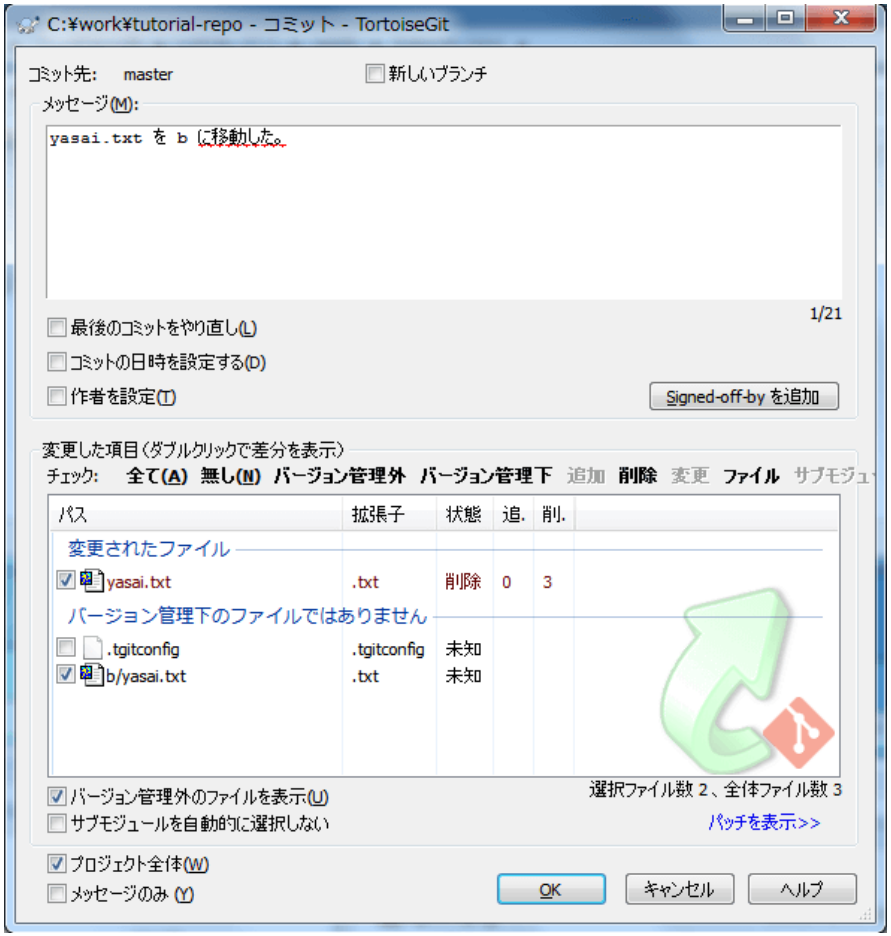
```
1  git mv <old-filename> <new-filename>
2
3  これは、以下と同意。
4
5  git rm <old-filename>
6  git add <new-filename>
```

このように、Gitはファイルの移動に関して履歴に残さない点を知っておく必要があります。
実際にyasai.txtをフォルダbに移動したい場合はどうするのでしょうか。
Windowsエクスプローラーからyasai.txtをドラッグ＆ドロップしてフォルダbに移動します。



レッスン９．ファイルを移動

これで、ファイルの移動は完了です。
エクスプローラーからファイルを移動した場合もコミットしなければリポジトリに反映されないことに注意してください。



レッスン９．ファイルを移動

tracpath

コミットダイアログの「変更した項目」を見て下さい。
削除されるファイル

- 変更されたファイル：yasai.txt

追加したファイル

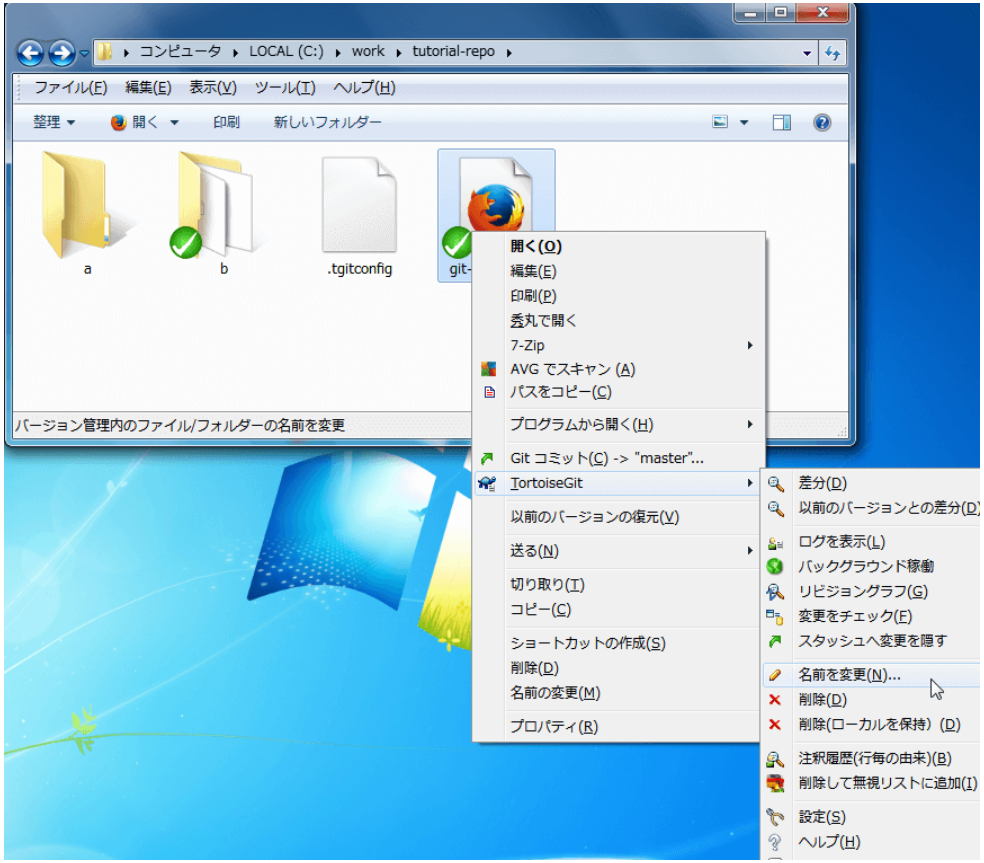
- バージョン管理下のファイルではありません：b/yasai.txt

> **警告 ： ファイルの移動はgit rmの後にgit addを実行することと同じです。**

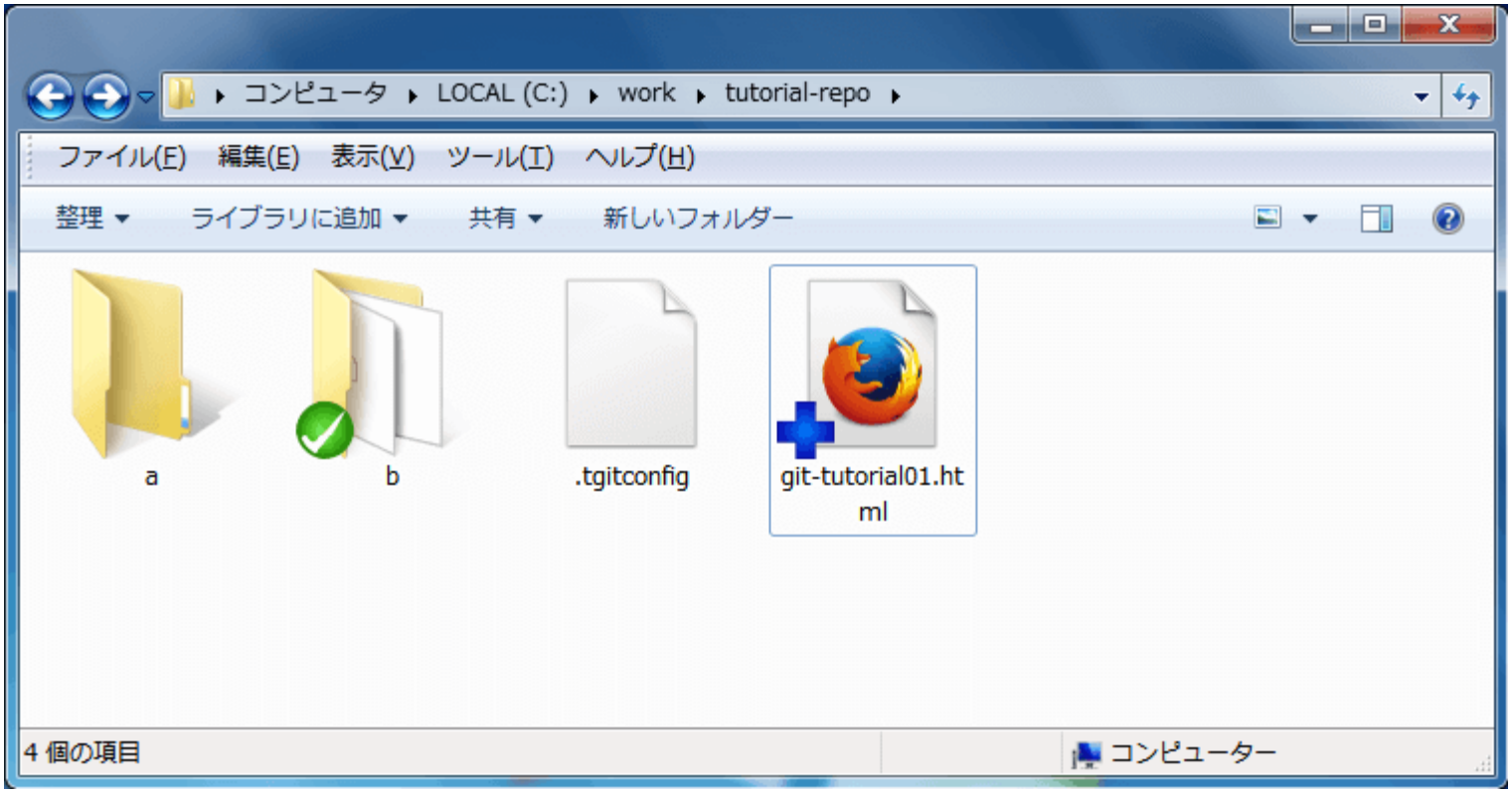
レッスン１０．ファイル名を変更

ファイル名の変更はTortoiseGitのメニューから実行する必要があります。Windowsエクスプローラーからファイル名の変更を行わないで下さい。

- git-tut01.htmlをgit-tutorial01.htmlに変更しました。



レッスン１０．ファイル名を変更



コミットは必要ですので忘れずに実行します。

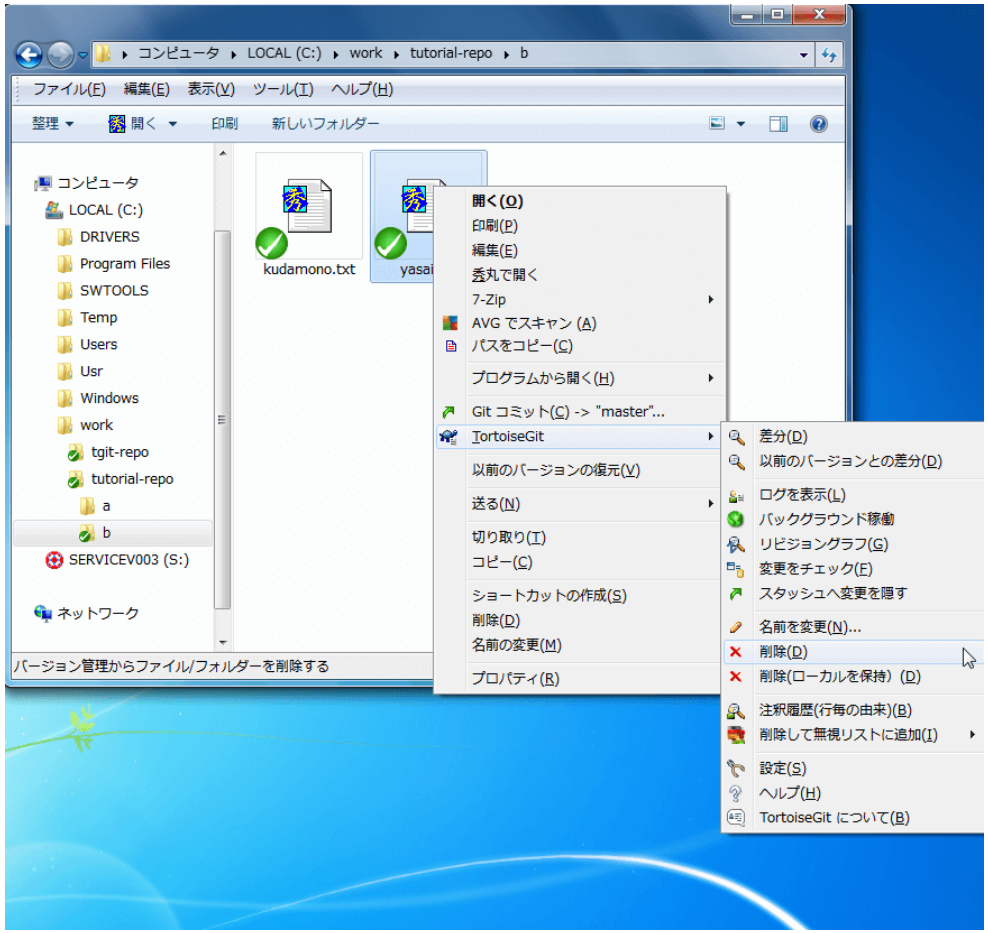
レッスン１１．ファイルを削除

ファイルの削除を実行します。この操作もTortoiseGitメニューから実行したあとにコミットをしてください。「ファイル名の変更」、「ファイルの削除」はバージョン管理されているファイル群のみ使うことができます。

ノート ：

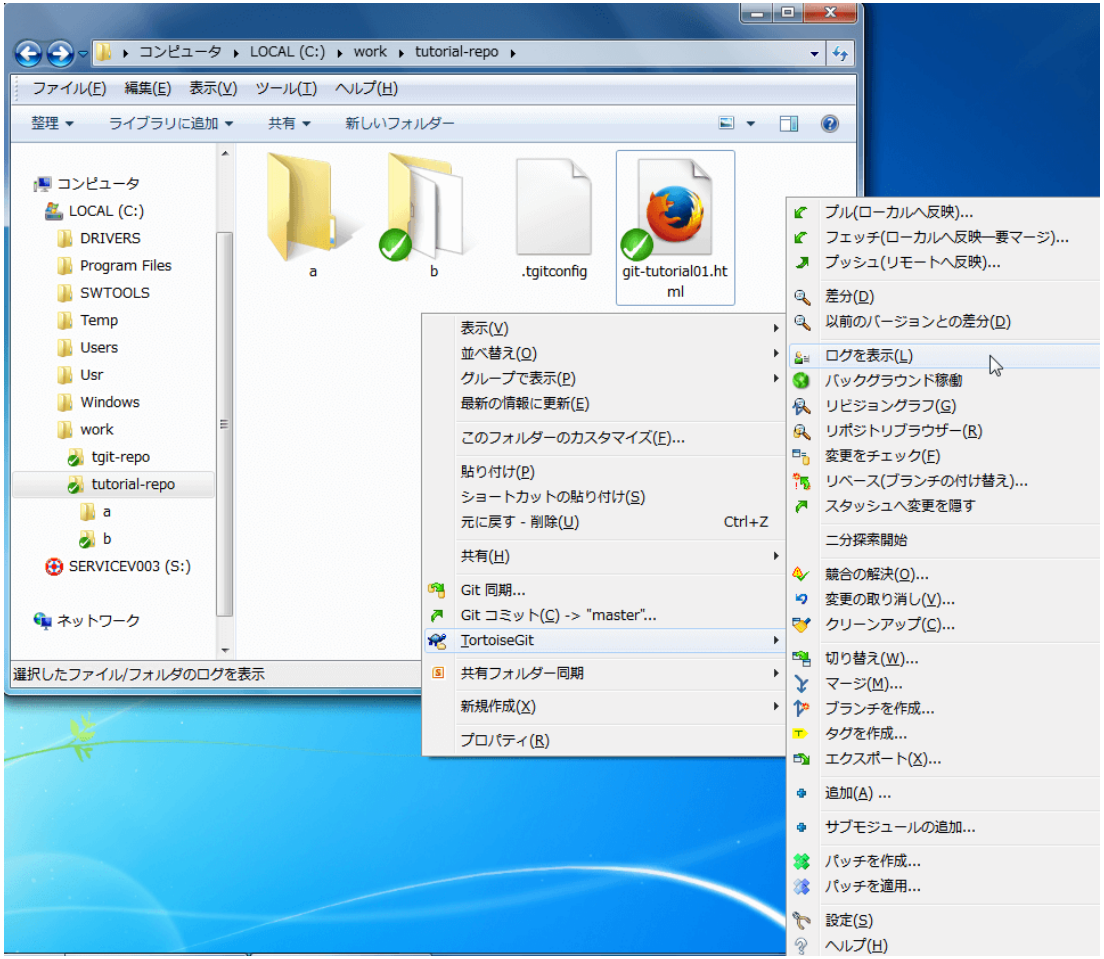
「削除」と「削除(ローカルを保持)」

TortoiseGitのメニューに2つの削除を見つけたと思います。「削除」はローカルのファイルを削除して、次回コミット時にリポジトリも削除します。「削除(ローカルを保持)」はローカルのファイルは残したままで、次回コミット時にリポジトリのファイルを削除します。

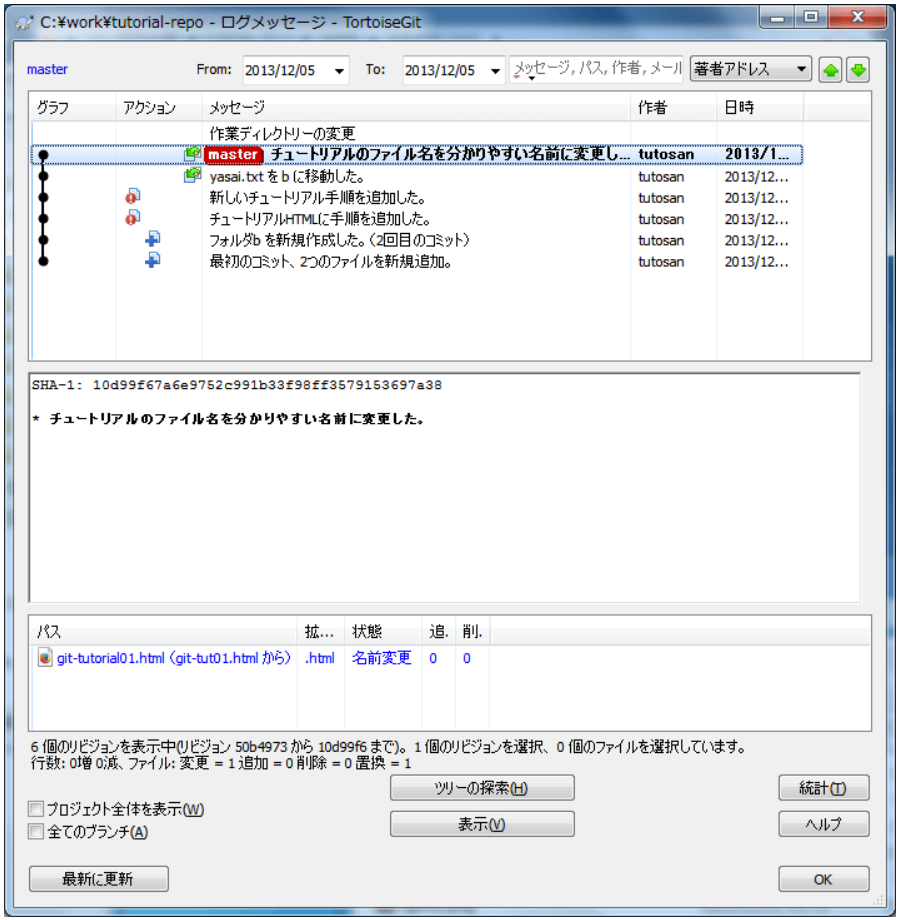


レッスン１２．履歴を確認

これまでの変更履歴を確認してみます。 履歴を確認する方法は、TortoiseGit右クリックメニューの「ログを表示」を選択します。



レッスン１２．履歴を確認



これまでコミットした変更内容と変更したファイル一覧が表示されます。変更履歴毎に右クリックすると差分を表示したり、取消しを行ったりすることができます。

レッスン13. 共有リポジトリにプッシュ(push)する

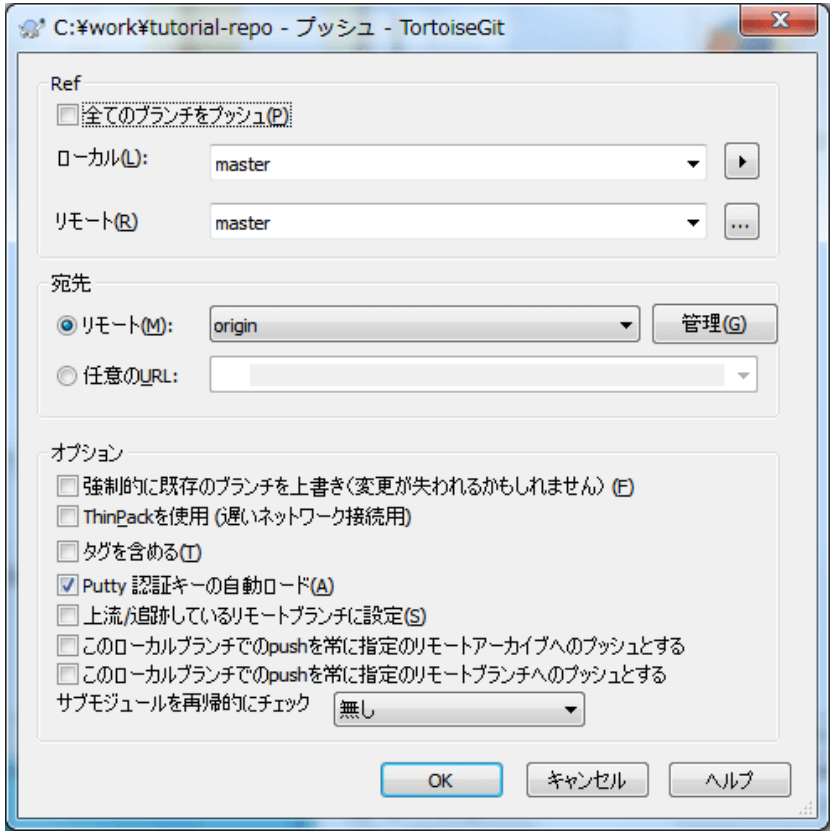
最後に共有リポジトリにローカルリポジトリの変更を反映する プッシュ（push）について説明して終わりにしたいと思います。

ローカルリポジトリでファイルを追加したり、編集したりしてからコミットを実行しました。これはローカルリポジトリ内のみに反映されているため、自分以外の開発者はあなたの変更を知りません。あなたの開発した機能やバグ対応したソースコードを他の開発者にも反映する必要があります。

> ノート ：
> **一番最初の「レッスン3. リポジトリ用のディレクトリを作成」で作成した、「共有リポジトリ」がありました。**
> **チュートリアルでは共有リポジトリをローカルマシン（c:¥work¥tgit-repo）に置いていました。**
> **この共有リポジトリにローカルリポジトリの変更を反映します。**

c:¥work¥tutorial-repoフォルダの空白部分を右クリックしてTortoiseGitのプッシュ（リモート反映）を選択します。

レッスン13. 共有リポジトリにプッシュ(push)する



「宛先」リモート（M）：origin は共有リポジトリ（c:¥work¥tgit-repo）からクローンしたときに定義されています。

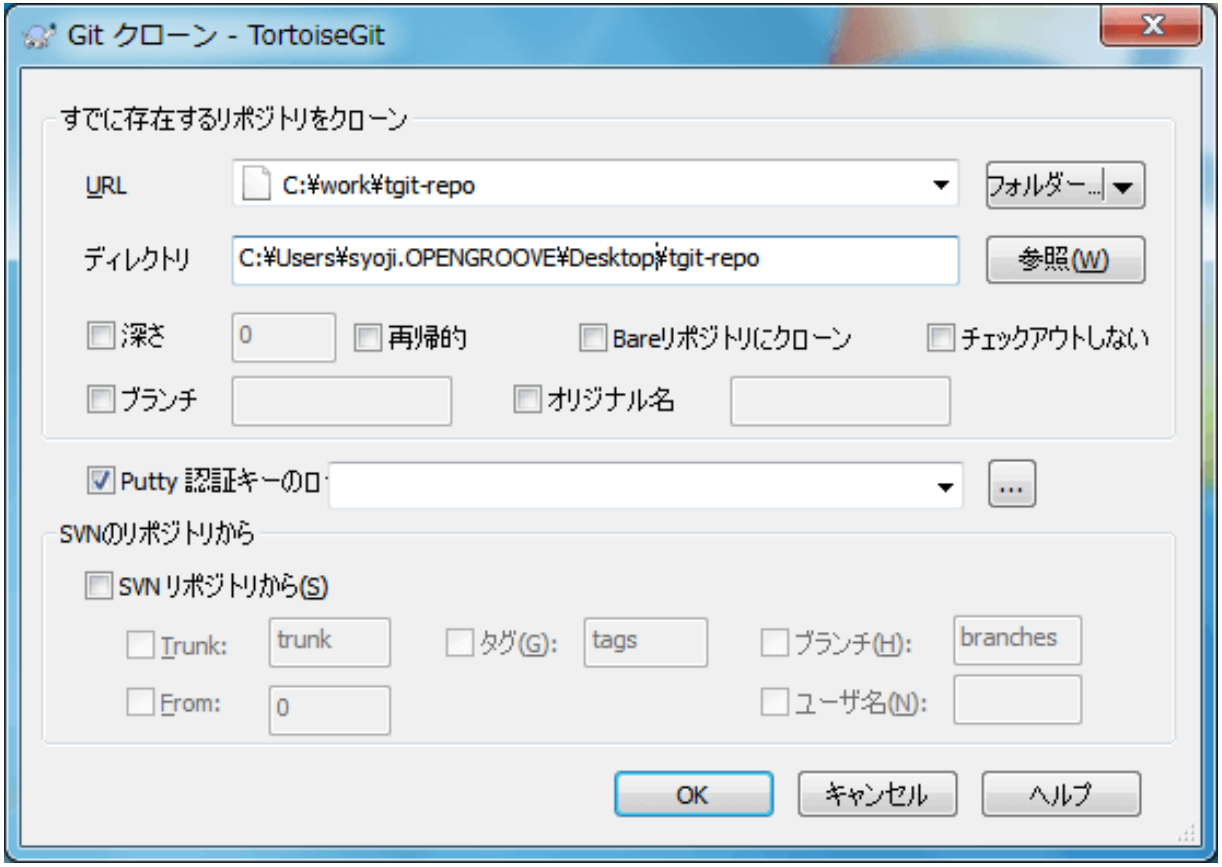
「OK」を押下するとローカルリポジトリ（c:¥work¥tutorial-repo）で行ったコミットが共有リポジトリ（c:¥work¥tgit-repo）に反映されます。

レッスン１４．クローンする

tracpath

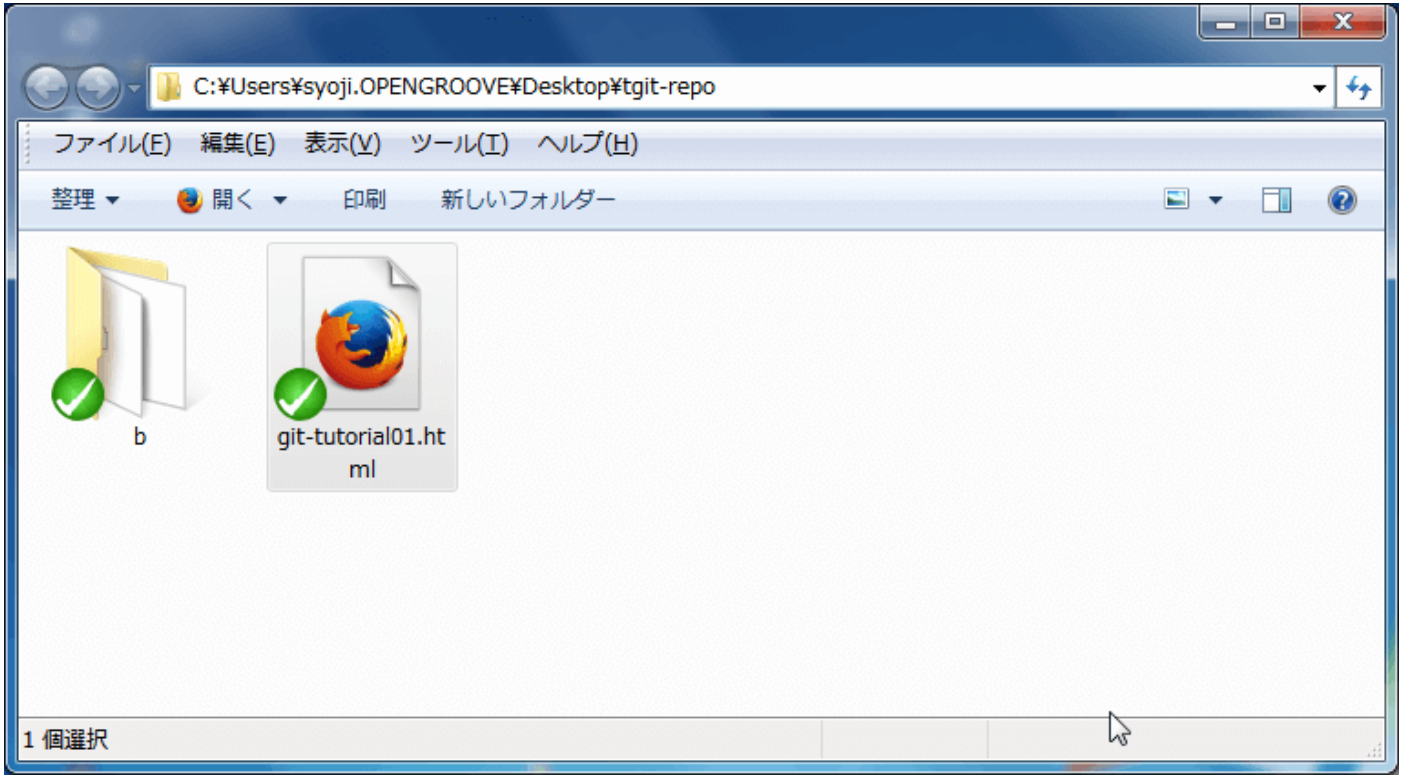
おまけとしてpush（プッシュ）した共有リポジトリに自分のローカルリポジトリで実施した開発内容が反映されているかを確認します。

デスクトップを右クリックして「Git クローン（複製）」を選択します。



レッスン１４．クローンする

デスクトップにtgit-repoができました。
フォルダを見て下さい。



レッスンで実行したファイルやフォルダが見つかりました。
これで共有リポジトリからローカルリポジトリにクローンして、レッスンで実施したことが共有リポジトリにはpush（プッシュ）されていることが確認できました。

最後に

tracpath

これでTortoiseGitの基本学習用チュートリアルは終了します。よく使うコマンドを中心に説明していますが、チームでの複数メンバーによる開発や高度な機能については説明していません。

このチュートリアルはいったんこれで終了します。さて、次は何をすればよいでしょうか？
分散バージョン管理について基礎からやりたいという場合は、こちらのGitの基礎勉強～Gitによるバージョン管理を使う～が参考になるでしょう。

バージョン管理のアプリケーションを理解する一番の近道は実際に使ってみることです。あなたの開発で利用しはじめてください。あなたが所属する会社やチームでバージョン管理を使っていなかったり、別のツールを使っている場合があるかもしれません。そんなときでも、ひとりでGit/TortoiseGitを使うことは有益です。

さらに、ネットには数多くの良質なコンテンツがたくさんあります。Git/TortoiseGitの専門書籍も多く出版されています。ぜひ参照して快適な開発ライフを。

参考資料

tracpath

- リモートリポジトリを使うなら、tracpath（トラックパス）が便利です！
  下記記事をぜひご参照下さい。
  ■tracpath（トラックパス）を使って、安全に複数名でバージョン管理を行う

tracpath（トラックパス）のご紹介

バージョン管理サービス・プロジェクト管理サービスの「tracpath（トラックパス）」では、ユーザー５名、リポジトリ数３つまで、永久無料で利用可能です。
学んだ知識を活かして、さっそく実務でも使って見ましょう。
エンタープライズ利用が前提のASPサービスなので、セキュリティも強固です。