# 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~



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



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]

→ lesson 01				- <b>X</b>
登理 ▼ ライブラリに追加 ▼ 共有 ▼ 新しい	マオルダー	8==	•	0
名前	種類			
⊿ Windows インストーラー パッケージ (2) —				
🐻 TortoiseGit-1.8.6.0-32bit	Windows インストーラー パッケー	->>		
TortoiseGit-LanguagePack-1.8.6.0-32bit-ja	Windows インストーラー パッケー	ージ		
* アプリケーション(1)				
Git-1.8.4-preview20130916	アプリケーション			

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-1.8.4preview20130916.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



### 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



🔖 Git Setup	
Adjusting your PATH environment How would you like to use Git from the command line?	
Our Section Use Git Bash only This is the most conservative choice if you are concerned about the stability of your system. Your PATH will not be modified.	
Run Git from the Windows Command Prompt This option is considered safe and no conflicts with other tools are known. Only Git will be added to your PATH. Use this option if you want to use Git from a Cygwin Prompt (make sure to not have Cygwin's Git installed).	Sit Setup
© Run Git and included Unix tools from the Windows Command Prompt Both Git and its accompanying Unix tools will be added to your PATH. Warning: This will override Windows tools like find.exe and sort.exe. Select this option only if you understand the implications.	Configuring the line ending conversions How should Git treat line endings in text files?
http://msysgit.googlecode.com/ < Back Next > Cancel	<ul> <li>Checkout Windows-style, commit Unix-style line endings</li> <li>Git will convert LF to CRLF when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Windows ("core.autocrlf" is set to "true").</li> <li>Checkout as-is, commit Unix-style line endings</li> </ul>
	Git will not perform any conversion when checking out text files. When committing text files, CRLF will be converted to LF. For cross-platform projects, this is the recommended setting on Unix ("core.autocrlf" is set to "input").
	Checkout as-is, commit as-is
	Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").
	http://msysgit.googlecode.com/

### レッスン1.インストール



#### 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





#### 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。



闄 TortoiseGit 1.8.6.0 (32 bit) Setup	
Git	Choose SSH Client
660	Choose a kind of SSH Client
TortoiseGitPLink, coming from Putty, integrates with	Windows better.
OpenSSH, Git default SSH Client	
This setting can be changed in TortoiseGit settings on the	"Network" page lateron.
< Bac	k Next > Cancel

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。

🐺 Settings - TortoiseGit					×
🔺 🔧 General 🔺	👞 General				
Context Menu Context Menu Item Cologs 1 Dialogs 2 Dialogs 3 Colors 1 Colors 2 Colors 2 Colors 2	TortoiseGit Language: I Automatically ch System sounds	English English eck for newe <del>日本語(日本),</del>	HOOK	Config Create Li	▼ ure ibrary
Alternative editor     Alternative editor     Git     Git	Git for Windows Git.exe Path: Extern DLL Path: Version:	C:¥Program Files¥Git¥bin		Check	now
Overlay Handlers  Network		OK	キャンセル	適用(A)	<u>ر الا ک</u>





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。

ःः TortoiseGit
Tortoise GIT
TortoiseSVN 作者: Frank Li (Iznuaa@gmail.com), Sven Strickroth <email@cs-ware.de> 謝辞: Sup Yut Sum <ch3cooli@gmail.com>, Colin Law <chanlaw@googlemail.com>, Myagi <snowcoder@gmail.com>, Johan 't Hart <johanthart@gmail.com>, Laszlo Papp <djszapi@archlinux.us></djszapi@archlinux.us></johanthart@gmail.com></snowcoder@gmail.com></chanlaw@googlemail.com></ch3cooli@gmail.com></email@cs-ware.de>
ウェブサイトを見るにはこの行をクリックして下さい。 そして、開発者たちをサポートしてください。
TortoiseGit 1.8.6.0 (C:¥Program Files¥TortoiseGit¥bin¥) git version 1.8.4.msysgit.0 (C:¥Program Files¥TortoiseGit¥bin¥bin¥Y) git version 1.8.4.msysgit.0 (C:¥Program Files¥TortoiseGit¥bin¥bin¥bin¥bin¥bin¥bin¥bin¥bin¥bin¥bin
最新バージョンをチェック OK

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 🛠 tracpath

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。





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 🛠 tracpath

### 「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 <mark>弁 tracpath</mark>

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。

ぷ Git クローン	- TortoiseGit
すでに存在するり	ポジトリをクローン
URL	C:¥work¥tgit-repo ▼ フォルダー▼
ディレクトリ	C:¥work¥tutorial-repo 参照(W)
深さ	0 両帰的 Bareリボジトリにクローン デェックアウトしない
🔲 ブランチ	□オリジナル名
▼ Putty 認証 SVNのリポジトリカ ■ SVN リポジ	(ት - ወር) እና ሥክት(s)
	trunk     少グ(G):     tags     ブランチ(H):     branches       0     ユーザ名(M):
	ок (*«>tzı) (ли)

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。







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。

								x
997	↓ コンピューター )	LOCAL (	C:) • work • tuto	orial-repo				<b>▼   <del>^</del>}</b>
整理 ▼	ライブラリに追加 ▼	共有 ▼	新しいフォルダー		:	•		0
名前	A		更新日時	種類		サイ	ズ	
🌗 .git			2013/12/04 17:53	ファイル フォル				

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。



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。





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。

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 2 3 4 5 6 7	<html> <body> <h1>TortoiseGit チュートリアル<ol> <li>クローンして</li> <li>ローカルリポジトリを作成</li></ol></h1></body></html>	>	
8 9 10 11	<ii>開発作業</ii> <ii>コミット</ii> <ii>プッシュ</ii>	<ul> <li>マ↓・コンピューター・・</li> <li>整理 マ ライブラリに追加 マ</li> <li>名前</li> </ul>	LOCAL (C:) ・ work ・ tutorial-repo ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・
11 12 13 14	 	🤰 .git 🗑 git-tut01.html	2013/12/04 17:53 ファイルフォル 2013/12/05 11:05 Firefox HTML D 1 KB
			<pre></pre>
		*	



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。

パス	拡張子	
バージョン管理下のフ	ァイルではあ	りません
git-tut01.html	.html	
		and the second
2 全てを選択/解除(A)	無視ファイルを	含める(I)
	ж 🐴	キャンセル ヘルプ



「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。

「OK」押下でローカルリポジトリに「gittut01.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。



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。

C:¥work¥tutorial-repo - □	ミット - Tor	toiseG	it								×	
コミット先: master メッセージ(M): 最初のコミット、2007z1ルを新力	日 「「「」「」	]新しい	ブラン	÷								
<ul> <li>□ 最後のコミットをやり直し()</li> <li>□ コミットの日時を設定する(D)</li> <li>□ 作者を設定(D)</li> <li>変更した項目(ダブルクリックで差分 エーク、 クライン、毎日(P)、P</li> </ul>	を表示)	1.54 12		×	<b>aT</b>	249-bas	Zuiko	Signed	l-off-b	y を追	1/22 加	
デェッジ: 主((A) 無((N) ア	マリノ 管理 拡張子	おおい	ーン: 追,	削	ΨF	ᇩᄱ	HUPSK	æ£	771	<i>и</i> с	7752	(
変更されたファイル ②git-tu/01.html ② ● yasai.btt パージョン管理下のファイ、 □tgitconfig	.html .txt ルではありま .tgitconfig	追加 追加 ません 未知						E	ŝ			
▼バージョン管理外のファイルを表 サブモジュールを自動的に選択	i示(U) しない					選扎	尺ファイ	ル数 2	、全体 ペッチを	ファイ) 表示:		
✓ プロジェクト全体(W) ✓ メッセージのみ (Y)					<u>O</u> K		*	ャンセル		~)	げ	



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。

Af C:¥work¥tutorial-repo - コミット - TortoiseGit 🛛 💷 🕺
コミット先: master     新しいブランチ       メッセージ(M):       最初のコミット、2つのファイルを新規追加。
👷 C:¥work¥tutorial-repo - Git コマンド実行中 - TortoiseGit 🛛 💷 🗮 🔀
22 変 チ [master (root-commit) 50b4973] 最初のコミット、2つのファイルを新規追加。 2 files changed, 17 insertions(+)
create mode 100644 git-tut01.html create mode 100644 yasai.txt
- 成功 (156 ms @ 2013/12/05 17:34:27)
「 プッシュ(リモートへ反映)   ▼ クローズ(C) 中止
<ul> <li>図 バージョン管理外のファイルを表示(U) 選択ファイル数 2、全体ファイル数 3</li> <li>□ サブモジュールを自動的に選択しない パッチを表示&gt;&gt;</li> </ul>
⑦ プロジェクト全体(W) □ メッセージのみ (Y) 

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。

#### 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ーー覚えておく必要も無いのでーー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。

C:¥work¥tutorial-repo - 追	加 - TortoiseGit	
1/1/2	加5辰丁	
バージョン管理下のファイル	ではありません ――	
.tgitconfig	.tgitconfig	
V Bokudamono.txt	.txt	
		and the second se
		1000
全てを選択/解除(A) 三無視し	ァイルを含める <mark>(I</mark> )	
	ОК	

レッスン7.新しいディレクトリを作成・追加 🗲 tracpath

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。

- 1 コミット時は更新内容をログとして記述
- 2 バージョン管理システムの更新

1	<html></html>
2	<body></body>
3	
4	<h1>Subversion チュートリアル</h1>
5	<0 >
6	<li>クローンして</li>
7	<li>コ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	<li>コミット時は更新内容をログとして記述</li>
12	<li>バージョン管理システムの更新</li>
13	
14	
15	
16	

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 ファイルの更新



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。

🔊 =	git-tut01.html - TortoiseGitMerge	6		
٠ ۲۲۶				スタイル 👻 🕜 🚺
		の相違点 ●次の競合点 の相違点 ●前の行内相違 の競合点 ●次の行内相違 移動	<ul> <li>■ 長い行を折り返</li> <li>三 長い行を折り返</li> <li>三 行内差分</li> <li>三 行内差分</li> <li>三 単語単位の行り</li> <li>表示</li> </ul>	호す ● <del>~</del> 9差分 ¥ ¥
git-tut01.html:cae86aa	A .	git-t	ut01.html : 作業ツリー	A
<pre>1 <tn>i <tn <tn="">i <tn>i <tn>i <tn>i <tn>i <tn>i <tn>i <tn>i <tn>i <tn>i</tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></tn></pre>	1 2 3 3 4 5 6 6 7 8 9 10 11 12 14 15 16 17 18	html>d <body>d <hl>TortoiseGit-f_1 <ol>d <ol>d <li>d)-JUボグトリC</li><li>d)-JUボグトリC <li>国発作業</li>d <li>Ti&gt;関発作業</li>d <li>Ti&gt;Ti&gt;J <li>Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti&gt;Ti <li>Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti&gt;Ti</li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></li></ol></ol></hl></body>	ートリアル- <sup>↓</sup> 」 作成・ <sup>↓</sup> 容をログとして記述・ <sup>↓</sup> 方ムの更新・	
<pre>chtml&gt;d</pre>				
│ <ntm⊥>↩ ヘルプは F1 横スクロールは Ctrl + ホイール操作</ntm⊥>	左ビュー: UTF-8 BOM CI	LF 右ビュー: UTT	F-8 BOM CRLF / + 1	鏡合:0 CAP NUM

#### 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。





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

6

git mv <old-filename> <new-filename>

これは、以下と同意。

git rm <old-filename> 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

🗱 tracpath

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。

☆ C:¥work¥tutorial-repo - コミット・	- TortoiseG	lit					
コミット先: master	■新しい	ブランチ					
メッセージ(≥):							
yasai.txt を b <u>に移動した。</u>							
□ 最後のコミットをやり直し(1)							1/21
🔲 コミットの日時を設定する(D)							
── 作者を設定①					Sig	ned-off-by ह	追加
変更した項目(ダブルクリックで差分を表示) チェック: 全て(A) 無し(N) バージョン パス	<b>/管理外 バ</b> 拡張子	ー <b>ジョン</b> 状態	<b>管理</b> 追.	下 ; 削.	追加削除 麦	更 ファイル	サブモジュ・
変更されたファイル							
V yasai. txt	.txt	削除	0	3			
バージョン管理下のファイルではa □tgitconfig ☑ ��]b/yasai.txt	ありません .tgitconfig .txt	未知 未知				2	
☑ バージョン管理外のファイルを表示(U) □ サブモジュールを自動的に選択しない					選択ファイル数	2、全体フ <del>ァ</del> パッチを表	vイル数 3 示>>
▼プロジェクト全体(W)							
<ul> <li>ニュージのみ (1)</li> </ul>				<u>O</u> K	++>	セル 🗌	



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

	k≢tutonai-repo														
aster	I	From: 2013/12	/05 👻	To: 20	13/12/	05 -	<u>7</u> 2	セージ	, パス	,作者,	メール [	著者〕	ッドレス	•	
グラフ	アクション	メッセージ								f	陼	E	]時		
		作業ディレクトリ	ーの変	更											
•	ě	master 71	- ኮሀፖ	ルのファイル	名を分	かりも	すいえ	名前に	変更	<b>Ն</b> t	utosan		2013/	1	
	é	yasai.txtをb(	こ移動し	た。						tı	utosan	2	2013/12	2	
•	A second seco	新しいチュート	アル手	順を追加した	•					tı	utosan	2	2013/12	2	
•	A second seco	チュートリアル	TML(2手	順を追加した	÷.					tı	utosan	2	2013/12	2	
•	-	フォルダb を新規	見作成し	た。(2回目(	カコミット	-)				tı	utosan	2	2013/12	2	
	-	最初のコミット、	2007:	ァイルを新規i	追加。					tı	utosan	2	2013/12	2	
£⊐-1	ヽリアルのファイル	し名を分かりやす	「い名前	fに変更した。	•										
₹ <b>-</b> -1	ヽリア ル のファイ)	▶ 名を分かりや ā	「い名前	行っ変更した。	•										
<b>₹२</b> ►†	- IJ₽ II 0 7 7 4 J	548990109	「い名前 拡…	<b>仁変更した</b> 状態	• 追.	肖刂.									
チュート 《깄 @ git-tut	- IJア ル のファイJ orial01.html (git-	<b>ょさを分かりやす</b> ut01.html から)	「い名前 拡… .html	<b>  仁変更した</b>     名前変更	· 追. 0	肖J. 0									
チュート パス ● git-tut 個のリビさ 強: 0増	<b>・リアルのファイル</b> corial01.html (git-1 ジョンを表示中(リビ 0)減、ファイル: 変]	<b>ut01.html</b> から〉 ジョン 505-4973 ≵ 更 = 1 注動1 = 0	<b>拡…</b> .html	<b>北京更した</b> 状態 名前変更 9965まで)。1 0 置換 = 1	・ 追. 0	背山. 0 ビジョ:	レを選打	尺、01	<b>個</b> のフ	アイルな	i選択して	C(\at	ţ.	(**	
チュート パス ④ git-tut 個のリビジ 一致: 0寸	- <b>リアルのファイメ</b> torial01.html (git- ジョンを表示中()ビ 0)族、ファイル: 変]	<b>↓ 名 を分 かりや s</b> ↓ <b>t</b> 0 1. html から) ジョン 506-4973 ポ 更 = 1 3回加 = 0 、	<b>拡</b> .html 前除 =	<b>北京更した</b> 状態 名前変更 9966まで)。1 0置換 = 1	・ 道. 0 い何のリ	削. 0 ビジョ: −の探	~を選捕	尺、01	<b>個</b> のフ	рイルる	i選択して	C(\#:	す。	颛	計(1)
チュート ペス ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・ ・	- <b>リアルのファイメ</b> torial01.html (git- ジョンを表示中(リビ ジ族、ファイル: 変) りト全体を表示( <u>W</u>	<b>ut01.html</b> から) ジョン 5064973 ポ 更 = 1 追加 = 0 )	<b>広い名前</b> 拡 .html 前除 =	<b>北京更した</b> 状態 名前変更 9966まで)。1 0 置換 = 1	・ 道. 0 いりい・ : :	削. 0 ビジョ: −の探 表示(	ッを選捕 索(日)	尺、01	<b>適</b> のフ 〕 〕	アイルな	E選択して	ເທສ <sup>ະ</sup>	ţ.	颛	計(I) 、ルプ
チュート ペス ④ git-tut 「数: o増 〕 プロジェイ 〕 全てのブ	- <b>リアルのファイル</b> orial01.html(git- ジョンを表示中(リビ の減、ファイル・変) りト全体を表示( <u>W</u> ジランチ( <u>A</u> )	<b>ut01.html</b> から) ジョン 50b4973 か 更 = 1 追加 = 0 )	<b>拡…</b> .html 削除 =	<b>状態</b> 名前変更 99f6まで)。1 0置換=1	・ 追. 0 いりい、 : :	削. 0 ビジョ 表示(	ッを選捕 茶(L) V)	R、 0 1	<b>値</b> のフ 〕 〕	рイルる	注選択して		ġ.	〔新 〔	計(1)

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 🗚 tracpath

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:

ー番最初の「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 🗚 tracpath

Structure C: Struc	ial-repo - プッシュ - TortoiseGit
Ref	をプッシュ(P)
ローカル <mark>(</mark> ):	master
リモート <mark>(</mark> ₪)	master 💌 📖
宛先	
© リモート <u>(M</u> )։	origin 🔹 管理(G)
◎ 任意の <u>U</u> RL:	<b></b>
オプション 登集制的に既存 ThinPackを使用 タグを含める(1) Putty 認証キー 上流/追跡して このローカルブラ サブモジュールを再	のブランチを上書き(変更が失われるかもしれません)(F) 月(遅いネットワーク接続用) -の自動ロード(A) いるリモートブランチ(に設定(S) 5ンチでのpushを常(に指定のリモートアーカイブへのプッシュとする 5ンチでのpushを常(に指定のリモートブランチへのプッシュとする 別報的(こチェック 無し マ

「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



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。

🔐 Git クローン	- TortoiseGit
- すでに存在する!	ノポジトリをクローン
URL	C:¥work¥tgit-repo ▼ フォルダー ▼
ディレクトリ	C:¥Users¥syoji.OPENGROOVE¥Desktop¥tgit-repo 参照(W)
■ 深さ	0 同時的 Bareリポジトリにクローン 「チェックアウトしない
🔲 ブランチ	□ オリジナル名
🔽 Putty 認調	IF-00
- <mark>SVN</mark> のリポジトリカ	ng
📃 SVN リポジ	りから(5)
<u>T</u> runka	: trunk タグ(G): tags ブランチ(H): branches
Erom:	0 ユーザ名(11):
	OK キャンセル ヘルプ

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。

		x
C:¥Users¥syoji.OPENGROOVE¥Desktop¥tgit-repo		<b>- 4</b> <del>9</del>
ファイル(E) 編集(E) 表示(⊻) ツール(I) ヘルプ(出)		
整理 ▼ 📵 開く ▼ 印刷 新しいフォルダー 🖺	•	?
b git-tutorial01.ht ml		
1 個選択 しん		.đ

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



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



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



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。

