TortoiseGit の SSH 通信
および
パスフレーズの記憶と破棄について

2013 年 7 月 19 日　初稿

■ 改訂履歴

稿	改訂日	改訂者	改訂内容
初稿	2013 年 7 月 19 日	板垣　衛	(初稿)

■ 目次

■ 本書が扱うソフトウェアとバージョン

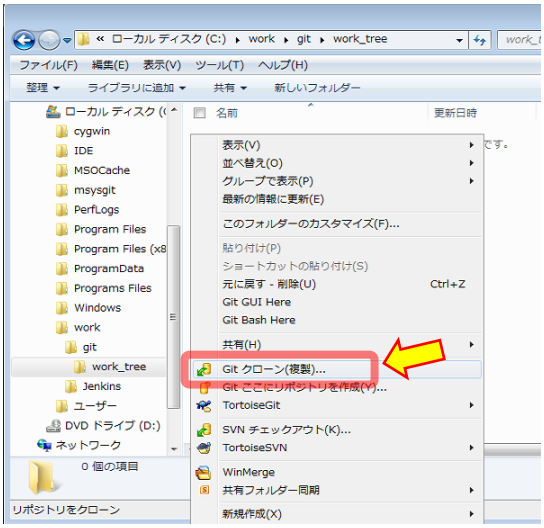
・msysGit … Ver.1.8.3.msysgit.0
・TortoiseGit … Ver.1.8.3.0 – 64 Bit

■ Git リポジトリのクローン（SSH 通信）

Git の共有リポジトリが既に存在する場合、かつ、そのサーバーとの通信が SSH 通信に対応している場合、下記の手順でリポジトリをクローンし、ローカル PC に作業ツリーとローカルリポジトリを作成する。

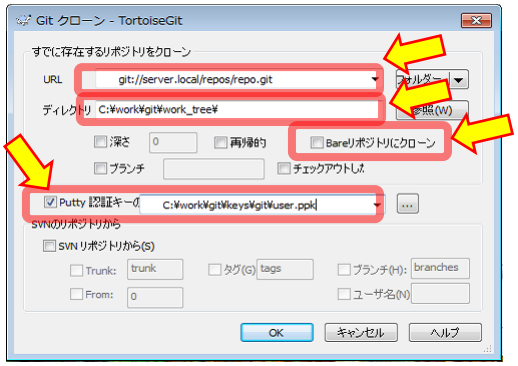
なお、一般向けのホスティングサービス「GitHub」では、https 通信と SSH 通信の両方に対応している。SSH 通信の方が通信の安全性が高く高速なので、GitHub などのように、SSH 通信に対応しているサーバーとは極力 SSH 通信を行う。その際、秘密鍵と公開鍵を作成して、公開鍵をサーバーに登録しておく必要があるが、その手順については別紙参照。

手順①： エクスプローラーで右クリックして、コンテキストメニューから「Git クローン(複製)...」を実行。



手順②： 「Git クローン」のダイアログにて、共有リポジトリの URL と作業ツリー&リポジトリを作成するローカルのディレクトリを指定する。

「Putty 認証キーのロード」欄に PPK 形式の秘密鍵ファイルを指定する。他はデフォルトのまま。秘密鍵ファイルの作成方法については別紙参照。

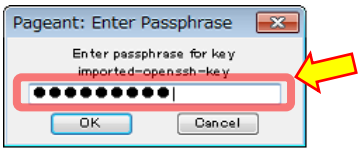


「Bare リポジトリにクローン」のチェックは ON にすると、ローカルリポジトリだけ作られて作業ツリーが作られないので注意。

クローン元の URL には、ssh:// で始まるパスを指定する。

また、サーバー名の前に「ユーザーID@」を付ける。これは SSH 通信の場合、Windows のユーザーID などではなく、Git のサービスを実行しているユーザーを指定する為、「git」などのユーザーID に統一されるケースが多い。（URL 例：ssh://git@server.local/dir/repos.git）

手順③： ［OK］ボタンを押してクローンを開始すると、ユーザーのパスフレーズを入力するダイアログボックスが表示されるので、パスフレーズを入力する。

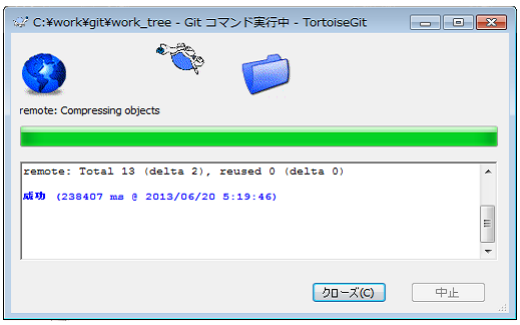


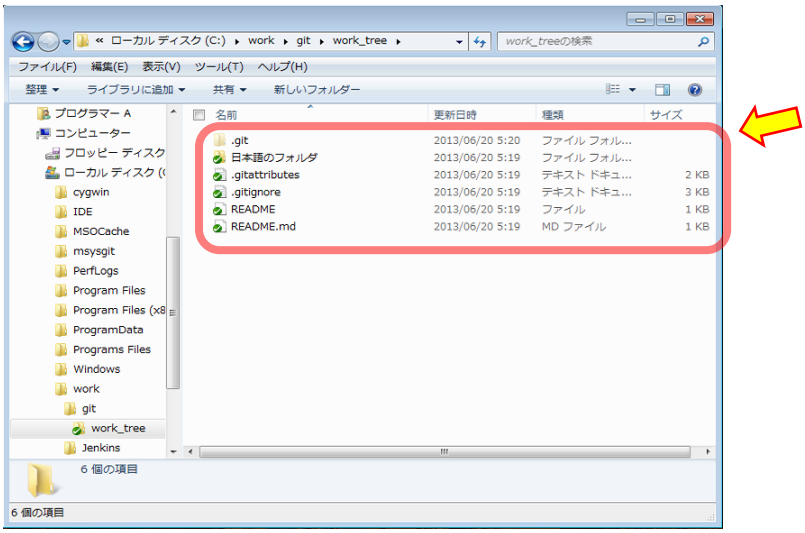
【解説】「パスフレーズ」は「パスワード」とは異なる。最大の違いは、「パスワード」がサーバーに送られて認証に用いられるのに対して、「パスフレーズ」はローカルに置いてある秘密鍵を有効化する為のものである為、入力したものがネットに流れる事がなく、安全性が高い。

手順④： 初めて通信するサーバーの場合、本当にそれが正しいサーバーなのか確認するように求められる。

※これは、なりすましなどの被害に遭わないようにする為のもので、もし、普段からアクセスしていたはずのサーバーに対してこのようなメッセージが表示された場合は、むやみに接続せずに管理者等に確認を取った方が良い。サーバー移設が行われた時なども表示される。

手順⑤： クローン完了。





クローンが完了すると、エクスプローラーに作業ツリー（マーク付きのフォルダ・アイコン）とローカルリポジトリ（.git フォルダ）が作成される。

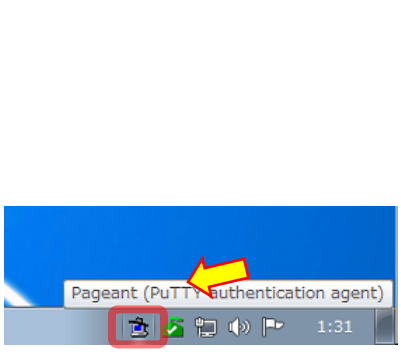
空っぽのリポジトリをクローンした場合は、.git フォルダが作成されるだけとなる。

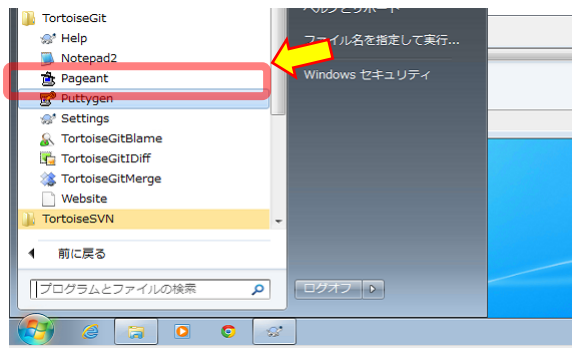
この後の共有リポジトリとの同期操作（プッシュ、プル、フェッチ、リベース）については別紙参照。

■ 秘密鍵とパスフレーズを記憶する仕組み：pageant について

SSH 通信に一度成功すると、pageant というツールが自動的に立ち上がり、常駐し、次回以降の共有リポジトリへのアクセスでは、パスフレーズの入力を省略できる。

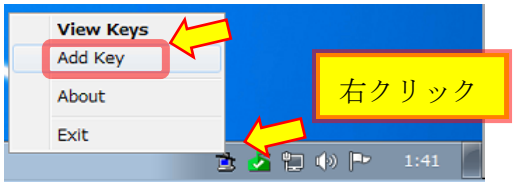
以下、pageant が常駐している様子と、スタートメニューから実行する場合。TortoiseSVN と共に自動的にインストールされている。



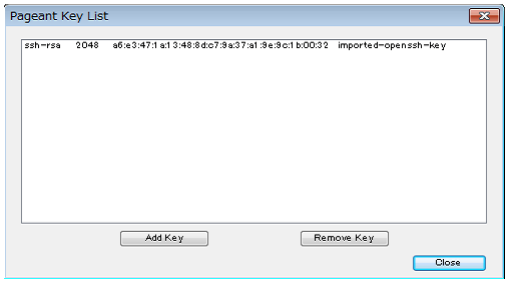


pageant は、TeraTerm や PuTTY などの SSH 端末のツールでも用いられるもので、pageant が起動してから pageant に追加した秘密鍵とパスフレーズを記憶している。

pageant を終了すると、記憶している秘密鍵とパスフレーズはリセットされる。
TortoiseGit では、pageant の起動および pageant への秘密鍵とパスフレーズの追加を自動的に行うが、以下のように、pageant を右クリックして「Add Key」メニューを実行する事で、手動登録も可能。(TeraTerm などで pageant を活用する際は手動登録する。)



以下、「View Keys」メニューを実行して、記憶している秘密鍵のリストを表示している状態。



【注意】
一人の人が同じ PC 上で、複数の秘密鍵（ユーザー）を切り替えて使っているような場合や、複数の接続先を利用していて、それぞれ秘密鍵を変えているような場合、pageant が意図と異なる動作をする事があるので要注意。
pageant の仕組みとして、アクセス先のサーバーに登録されている公開鍵にマッチする秘密鍵があれば、それを使用する。それは、TortoiseGit の「Putty 認証キーのロード」で指定している鍵とは無関係に、マッチするものが見つかりさえすれば使用される。

例えば、gitolite を使用しているサーバーがあり、サーバーには admin ユーザーの公開鍵と user_a ユーザーの公開鍵が登録されているとする。その両方を使い分けて同じクライアント PC からアクセスしようとすると、両方のユーザー用の秘密鍵が pageant に記憶される。そうなると、TortoiseGit の「Putty 認証キーのロード」の設定を変更して user_a でアクセスするように切り替えたつもりでも、常に admin ユーザーで接続してしまう、といった現象が起こるようになる。先に user_a でクローンした作業ツリーを操作する場合も同様。
この対策としては、pageant をいったん終了するなどして、記憶している鍵をリセットしなければならない。

■ 記憶している秘密鍵とパスワードを破棄

一度記憶した秘密鍵とパスフレーズをリセットしたい場合、pageant を終了するか、先述の pageant の「View Key」の画面で所定の鍵を選択して削除（Remove）する。また、PC を再起動してもリセットされる。
これにより、次回の共有リポジトリアクセス時には、再度パスフレーズの入力が求められるようになる。

■ Git リポジトリと作業ツリーの新規作成

別紙の「[03]TortoiseGit の http(s)通信およびパスワードの記憶と破棄について」の同項目を参照。

■ Git リポジトリの同期先の追加（SSH 通信）

別紙の「[03]TortoiseGit の http(s)通信およびパスワードの記憶と破棄について」の同項目を参照。

なお、その紙面では http(s)通信時の設定方法を示しているが、SSH 通信を用いる場合は、上記のリポジトリのクローンで示した SSH の通信設定を用いる。

■ Bare リポジトリの新規作成（方法①）：GUI 操作

別紙の「[03]TortoiseGit の http(s)通信およびパスワードの記憶と破棄について」の同項目を参照。

■ Bare リポジトリの新規作成（方法②）：コマンドライン操作

別紙の「[03]TortoiseGit の http(s)通信およびパスワードの記憶と破棄について」の同項目を参照。

TortoiseGit の SSH 通信およびパスフレーズの記憶と破棄について

以　上