TortoiseGitのSSH通信

および

パスフレーズの記憶と破棄について

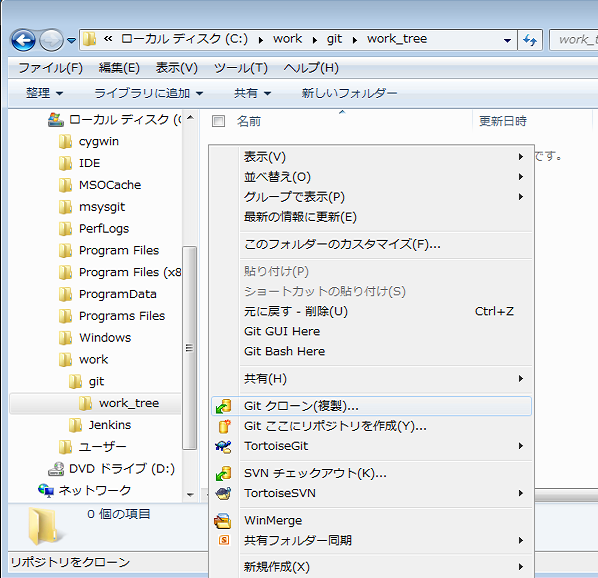
2013/6/20

* **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。

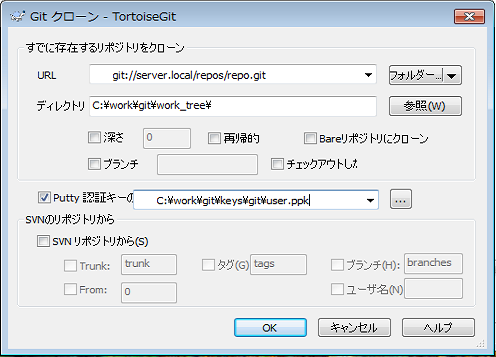
なお、一般向けのホスティングサービス「GitHub」では、https通信とSSH通信の両方に対応している。SSH通信の方が通信の安全性が高く高速なので、GitHubなどのように、SSH通信に対応しているサーバーとは極力SSH通信を行う。その際、秘密鍵と公開鍵を作成して、公開鍵をサーバーに登録しておく必要があるが、その手順については別紙参照。

**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。

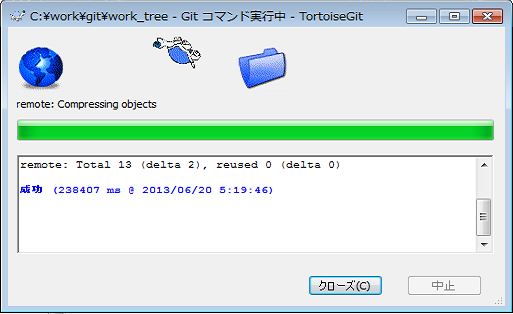
また、サーバー名の前に「ユーザーID@」を付ける。これはSSH通信の場合、WindowsのユーザーIDなどではなく、Gitのサービスを実行しているユーザーを指定する為、「git」などのユーザーIDに統一されるケースが多い。（URL例：[ssh://git@server.local/dir/](http://user_id@server.local/dir/)repos.git）

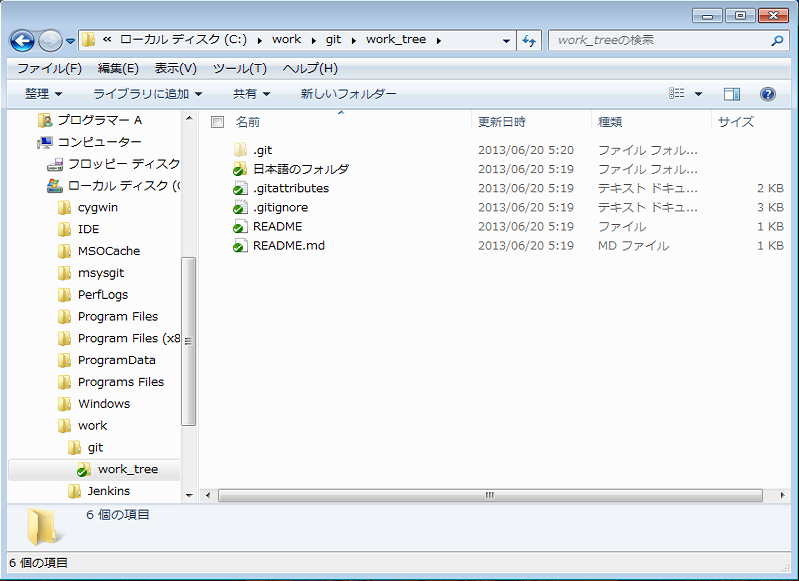
**手順③：** ［OK］ボタンを押してクローンを開始すると、ユーザーのパスフレーズを入力するダイアログボックスが表示されるので、パスフレーズを入力する。



【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：** 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。

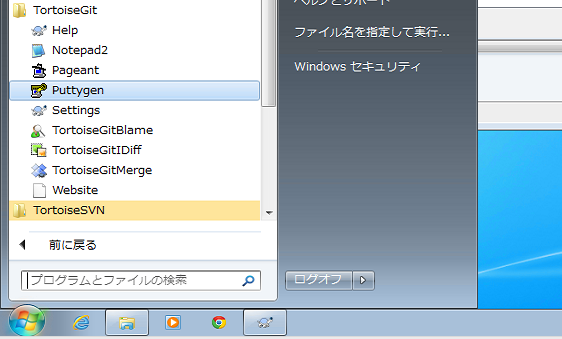
空っぽのリポジトリをクローンした場合は、.git フォルダが作成されるだけとなる。

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。

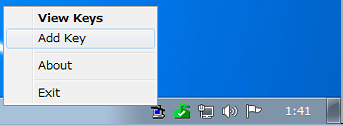
以下、pageantが常駐している様子と、スタートメニューから実行する場合。TortoiseSVNと共に自動的にインストールされている。

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。

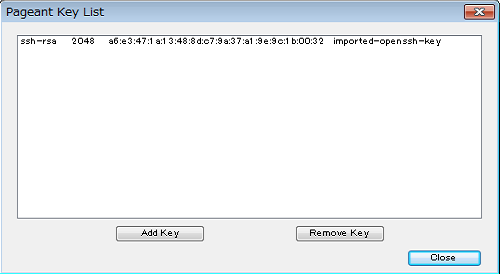
pageantを終了すると、記憶している秘密鍵とパスフレーズはリセットされる。

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。）



右クリック

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「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「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「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「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。

以上