OpenEyes - Version Control

Editors: G W Aylward
Version: 0.94:
Date issued: 26 September 2011

Target Audience

General Interest	
Healthcare managers	
Ophthalmologists	
Developers	✔

Amendment Record

Issue	Description	Author	Date
0.9	Draft	G W Aylward	4 Oct 2010
0.91	Draft	G W Aylward	18 Oct 2010
0.92	Draft - added change control policy	G W Aylward	23 Oct 2010
0.93	Draft - updated with new URL	G W Aylward	27 Nov 2010
0.94	Draft - updated with GitHub	M Wadham	26 Sep 2011

Table of Contents

Introduction

OpenEyes uses Git for version control. Git is a Distributed Version Control Systems which has several benefits over more conventional Centralized Version Control Systems, including the ability to restore the full history of the project from any client, and the low overhead involved in branching. Git was developed in 2005 to provide version control for the Linux project, and is available free of charge for a wide range of platforms.

Currently all the OpenEyes source files, graphics, database files, and documentation are available on GitHub (http://github.com) and accessible using HTTP (for cloning and updates) and over SSH.

Installation

Git is available free for a variety of platforms, and detailed instructions for installation are given in Appendix 1. The Git home page is at http://git-scm.com/.

Configuration

Git requires an initial configuration which includes information about the user. The email address is particularly important, as it is used to uniquely identify each developer.

Step 2 and 3 are only required for OEF developers, who need the SSH protocol to push changes to the server.

1. Set user information

```
git config --global user.name “your username"
git config --global user.email “your email address”
```

The default editor for git in Unix and Mac is Vi, but the following command will make it use your favourite editor.

```
git config --global core.editor "pico"
```

The configuration can be checked using the following command;

```
git config --list
```

This will produce an output something like the following;

```
user.name=Bill Aylward
user.email=bill.aylward@mac.com
core.editor=pico
```

2. Generate an SSH key pair

This is only necessary if you intend to push commits back into the OpenEyes project. If you only intend to check out the code and play around with it then you don't necessarily need to generate an ssh key pair.

If you have SSH installed (should be part of the Git installation) then the following commands will generate a key pair. SSH expects to find keys in a hidden subdirectory of your home folder called .ssh, and this can be generated using the initial three commands.

```
cd ~
mkdir .ssh
cd ~/.ssh
ssh-keygen
```

Accept the default file name and location (by pressing the return key), and optionally use a passphrase which you'll need to remember for later use (eg 'openeyes2010').

3. Upload the public key to your GitHub account

After logging into GitHub, go to Account Settings and then SSH Public Keys, and click "Add another public key" and paste in the contents of ~/.ssh/id_rsa.pub. If you're not sure how to view the contents of the public key, try this shell command:

```
cat ~/.ssh/id_rsa.pub
```

Downloading the OpenEyes repository

Once access to the server has been granted, a copy of the software can be downloaded from the repository as follows;

1. Create a directory where you want to put the project, and move to it:

In order to make testing easier, It makes sense to have the working directory in a position where it can be served by Apache or IIS.

```
mkdir gitprojects
cd gitprojects
```

2. Now download a clone of the project from GitHub:

The first command will work for any user. The second requires that your GitHub account has been given write access to the OpenEyes project.

Users:

```
git clone git://github.com/openeyes/OpenEyes.git
```

Developers:

```
git clone git@github.com:openeyes/OpenEyes.git
```

(NB XXXX should be replaced with the number of the port used for SSH)

The first time you do this, you will get the following message:

> Cloning into project...
>
> The authenticity of host '[github.com]:XXXX ([207.97.227.239]:XXXX)' can't be established.
>
> RSA key fingerprint is 7f:2a:72:d8:a4:3d:2f:5b:1b:83:11:c5:9d:57:cc:53.
>
> Are you sure you want to continue connecting (yes/no)?

Type 'yes' and press the return key. This will store the host key and stop SSH generating the message on subsequent connections.

It will also ask for your passphrase for the key-pair that you generated earlier On a Mac this can be stored in your keychain so that you don't have to enter it every time:

Now you should have a directory called 'OpenEyes' in your gitprojects folder. The directory consists of a complete copy of the working directory, plus a full history of the repository up to that point.

After cloning the repository, you'll probably want to check out the development branch as this contains the latest version of the code. You can do this with the following command:

```
git checkout -b development --track origin/development
```

Keeping up to date

Changes that are made by other developers can be ‘pulled’ down to your working directory using the following command:

```
git pull
```

Making changes

Git will keep automatically keep track of any changes made in the project directory, but will not do anything further without being told to do so. Imagine that you have added a file called ‘test.php’ to the project. Git will track the file, but will not consider it part of the project, as illustrated by running the following command;

```
git status
```

This will produce an output something like the following;

```
# On branch development
# Changed but not updated:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#    modified:   htdocs/test.php
#
no changes added to commit (use "git add" and/or "git commit -a")
```

If you want to commit the file to the project, it needs to be added, and then committed using the following commands;

```
git add .
git commit
```

The last command will launch and editor, and give you the opportunity to commend. It is helpful to add a single line which describes the added or modified file. After adding the comment, save and leave the editor.

All these steps can conveniently achieved with the single command;

```
git commit -am 'New file to display audit data'
```

3. Push the changes to the server:

When you are ready to distribute the changes to others, you can transfer them to the central server using the following command. Note that you should always pull before pushing.

```
git pull
git push
```

Change control policy

Certain files in OpenEyes are essential to the core functioning of the software. Therefore only OEF developers may push changes to the central repository. The 'development' branch represents the current working release, and changes to the this branch must be fully tested and agreed by the relevant project steering committee before uploading.

However, the excellent branching model of Git allows any number of experimental branches to be pushed, pulled, and tested.

Each OEF developer may push experimental branches to the server using the following naming convention; Initials, function, and number separated by underscore characters. Hence a branch with changes by Bill Aylward to test a family tree page might be called 'BA_TestFamilyTree_1'.

A new branch can be pushed to the server simple with;

```
git push origin BA_TestFamilyTree_1
```

Others can then track the branch with;

```
git branch --track BA_TestFamilTree_1 origin/BA_TestFamilTree_1
```

If it is no longer required, then it can be deleted from the server with;

```
git push origin :BA_TestFamilyTree_1
```

and then from the local repository with;

```
git branch -d BA_TestFamilyTree_1
```

SQL dump files are stored in the data subdirectory and are uploaded along with the other files in the repository. In order to facilitate testing, it is suggested that separate databases are created so that those on a developer's machine are not altered by the testing process. These would have a similar naming scheme with the initials of the developer and the lable of the database (eg BA_Test_Main, and BA_Test_RBAC). These settings would be entered into the configuration file (config.inc.php) which is also pushed to the repository.

Troubleshooting

1. Pushing fails

Attempting to push changes creates an error as in the following example

```
To ssh://git@aylwards.co.uk:XXXX/home/git/OpenEyes
 ! [rejected]        master -> master (non-fast-forward)
error: failed to push some refs to 'ssh://git@aylwards.co.uk:XXXX/home/git/OpenEyes'
To prevent you from losing history, non-fast-forward updates were rejected
Merge the remote changes before pushing again.  See the 'non-fast-forward'
section of 'git push --help' for details.
```

This arises when modifications have been made by another developer to files that are being pushed to the server. The solution is to fetch those changes, deal with any conflicts, then push them back.

2. Persistent deleted files

Deleted files remain in listing with git status as in the following example

```
# On branch master
# Changed but not updated:
#   (use "git add/rm <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#    deleted:    htdocs/TestJapplet.html
#
no changes added to commit (use "git add" and/or "git commit -a")
```

To remove the file, use the -u switch with git add before a commit

```
git add -u
```

3. Git post-receive emails stop

The most likely explanation is that you have updated your Git installation. If so, then reset the permissions for the post-receive script as follows;

```
cd /usr/local/share/git-core/contrib/hooks
```

Make the post-receive-email script executable (run this as root)

```
chmod a+x post-receive-email
```

Appendix 1 - Installation of Git

Macintosh

Git can be installed either with a package installer, or using Macports

1. Install

If using the package installer, download the iinstaller and follow the instructions. At the time of writing, the current version was git-1.7.1-intel-leopard.dmg.

If using MacPorts (assuming it is already installed), then type the following at the command line;

```
sudo port install git-core +svn +doc +bash_completion +gitweb
```

2. Check the path

The installer puts git in /usr/local/git/bin, so if git commands don't work, check the PATH environmental variable by typing;

```
env
```

If the path does not inlcude git, add it using the following command

```
export PATH=$PATH:/usr/local/git/bin
```

Unix Systems

There are multiple methods of installing Git according to the exact platform you are using. The following method should work with most installations.

1. Install required libraries

Ensure you have the following libraries installed: expat, curl, zlib, and openssl.

2. Download and compile

Download the source code from the Git home page *here*. Execute the following commands;

```
tar xjf git-1.6.x.x.tar.bz2
cd git-1.6.x.x
make prefix=/usr all
sudo make prefix=/usr install
```

If using the ports system for BSD, then use the following commands;

```
cd /usr/ports
porteasy -u /devel/git
cd devel/git
make install clean
```

While compiling, accept the option to enable building of GUI tools, install gitweb, and accept the default options for curl and other dependencies.

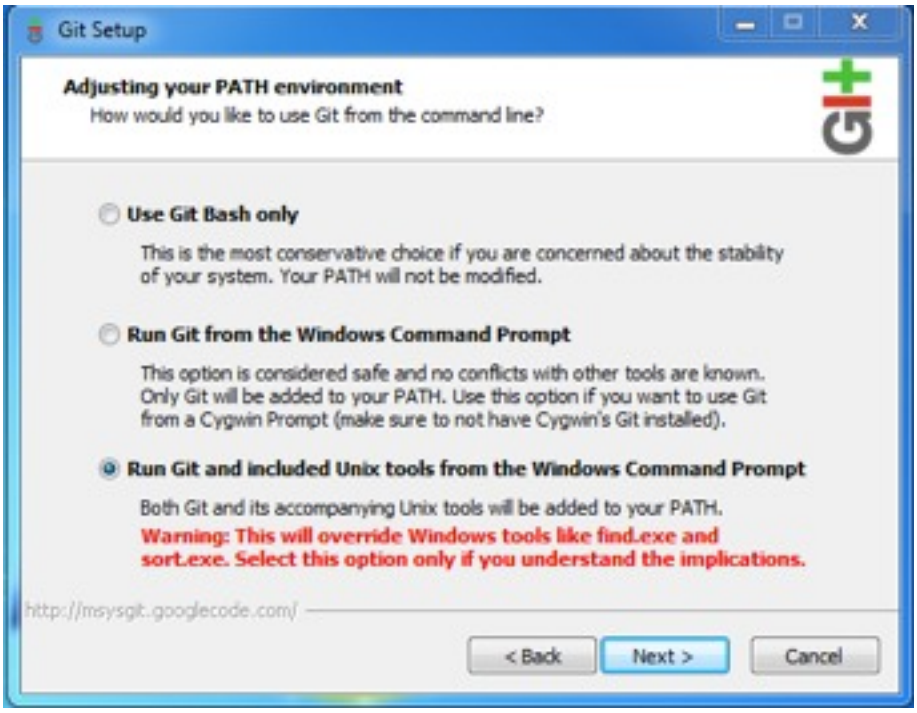
Windows

1. Download the installer

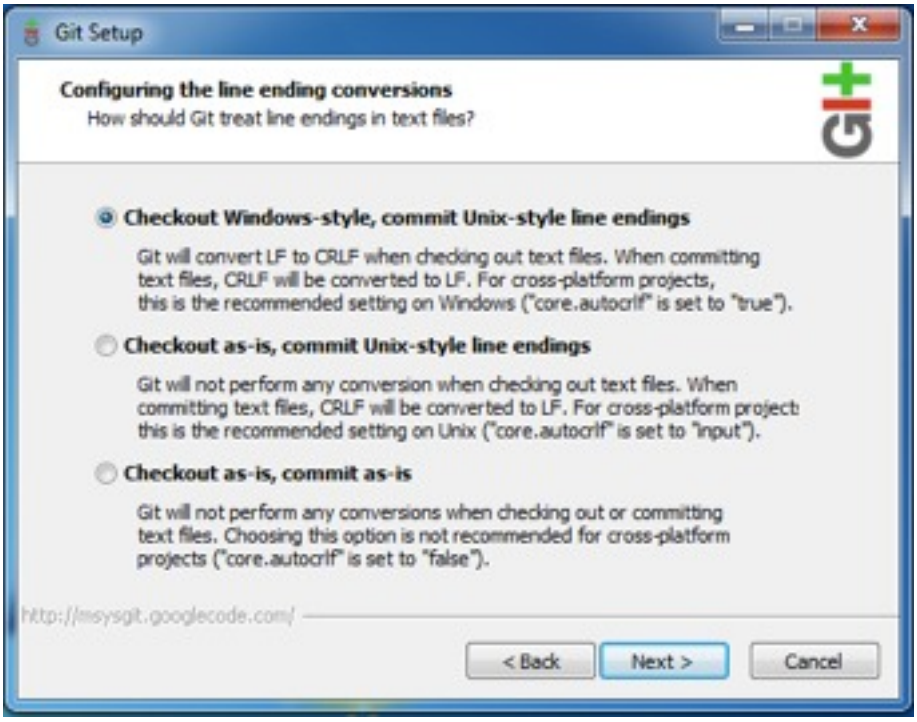
The latest version of Git can be found at the following URL: http://code.google.com/p/msysgit/downloads/list. Choose the “Full installer for official Git 1.7.3.1” (or whatever the latest version is).

2. Run the Installer

Run the installer which will launch a wizard. Accept all the defaults, up until the following screen;



It is recommended to choose the third option, as in the screen shot, but if you are concerned about the implications then choose one of the other two options. The following instructions assume you have chosen the third option. At the next screen accept the default option to including the option to checkout Windows-style,commit Unix-style line endings.



Complete the wizard, and you can now run Git from the command prompt by typing ‘git’ and return.