Working with GIT

Tim Sutton

2011

Contents

1 Working with GIT

This document provides a simple introduction to working with git. GIT is a distributed source code management (SCM) system. Although this is a source code management system, you can use it for **any** type of documents that you want to version control and / or work on collaboratively with others.

If you are familiar with SVN, GIT is quite similar but adds some new concepts. In particular, GIT is distributed, which means there doesnt have to be one single repository that you work against. GIT also is ideal for offline work where you still want to do version control. We will explore this more as we go on.

1.1 Installation

Installing GIT is easy. Under linux, install like this:

Listing

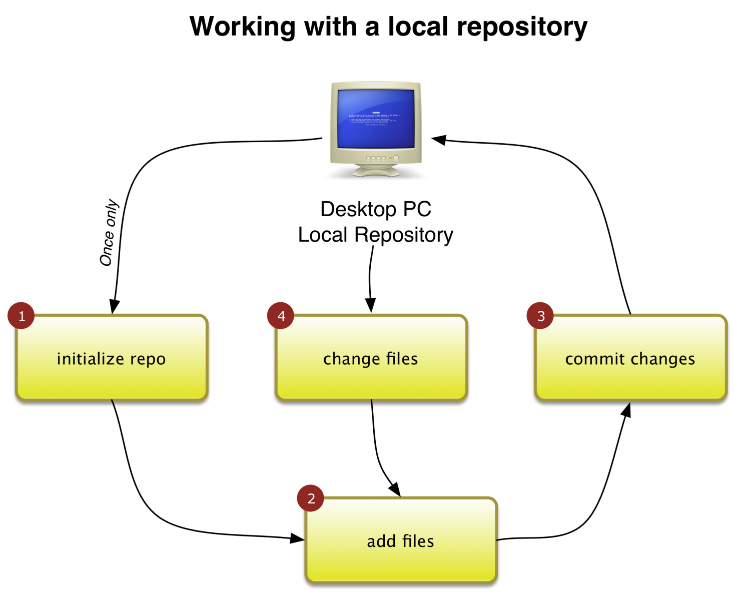
```
sudo apt-get install git meld gitg
```

The latter two are not actually needed but will prove useful later.

Under windows you can use msysgit - notes on using msys git are provided further down in this document.

1.2 Quick Start

I would like to plunge in with a quick start to using git and then come back to a more systematic coverage of the tool. The idea is to get you familiar with the concepts in the following diagram:



1.2.1 Create a local repository

We start off by creating a local repository. A local repository resides on your own hard dist (its just a directory with some special hidden files in it), and allows you to do all normal SCM activities - check out code, make changes, commit, review your history and so on. We will just make a new directory and initialise it as a git repository:

Listing

```
cd dev
mkdir git-sandbox
cd git-sandbox
git init
```

Ok now we have a repository, lets do some work in it.

1.2.2 Adding a file to the repository

To version any files in the repository, you just need to copy them into your directory (in our example it is called git-sandbox) or create them. Let's create a file called README:

Listing

```
gedit README
```

or

Listing

```
vim README
```

Now we will write a little text to the file:

Listing

```
Hello world from my sandbox
```

Then save and close the file. We can use the git *status* command to see the status of our repository:

Listing

```
git status
```

Which will return something like this:

Listing

```
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#        README
nothing added to commit but untracked files present (use "git add" to track)
```

It tells us that we have one untracked file - the README file we just created. So how do we start tracking the file?

Listing

```
git add README
```

Now *git status* will show the file as tracked:

Listing

```
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#        new file:   README
#
```

Ok, now you can commit your file:

Listing

```
git commit -m "First version of README"
```

```
[master (root-commit) 7ea07e1] First version of README
 1 files changed, 1 insertions(+), 0 deletions(-)
 create mode 100644 README
```

The *-m* option is used to specify the commit message. If you don't provide a *-m* option, git will prompt you for a message using a simple text editor. Once the file is committed, your repository will have a nice clean status:

Listing

```
git status
# On branch master
nothing to commit (working directory clean)
```

Let's make a small change to the README file:

Listing

```
Hello world from my sandbox - I'm using GIT!
```

Git status will show that the file is now modified:

Listing

```
# On branch master
# 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:   README
#
no changes added to commit (use "git add" and/or "git commit -a")
```

This change is **unstaged** - it wont be included in your next commit unless you add it.

Listing

```
git add README
```

Now its status is set to **staged** - it will be included in the next commit you make.

Listing

```
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       modified:   README
#
```

Finally you can commit your change using the commit command again:

Listing

```
git commit -m "Improved the README"
[master 983f6fd] Improved the README
```

```
 1 files changed, 1 insertions(+), 1 deletions(-)
```

Did you notice that odd looking number in the output? 983f6fd is shortened version of the unique SHA-1 hash assigned to that commit. Each commit you make will be assigned such an identifier. The commit numbers are not sequential numbers like SVN has. The reason for this is that the commits need to be globally unique in a distributed repository environment (which we will explore later).

We will finish off our quick start tour by running a few interesting commands on our repository:

Listing

```
git log README
commit 983f6fda163c09094ef6939b7d4db4af1bfa8c3c
Author: Tim Sutton <tim@linfiniti.com>
Date:   Mon May 9 23:04:08 2011 +0200

    Improved the README

commit 7ea07e1d1e029510a258efebc8cc170cb685803c
Author: Tim Sutton <tim@linfiniti.com>
Date:   Mon May 9 16:35:37 2011 +0200

    First version of README
```

The *git log* command shows you the history of commits made for a file (most recent changes are shown above older changes). You can see we have made two commits to this file, and the message associated with each commit.

Finally, lets look at the difference between these two commits:

Listing

```
git diff 7ea07e1 983f6 README
diff --git a/README b/README
index 2d564aa..d85ea56 100644
--- a/README
+++ b/README
@@ -1 +1 @@
-Hello from my sandbox
+Hello from my sandbox - I'm using GIT!
```

You can see the two options I passed to the diff command are shortened versions of the SHA-1 hashes assigned to each commit. You can also see that the text in my README file was augmented with the phrase "I'm using GIT!".

Hopefully this quick look at git has given you the basic concept. In the sections that follow we will explore with more detail some of the other things you can do with GIT.

1.3 Git Workflows

In this section we are going to walk you through various scenarios to show you how git can be used effectively.

1.3.1 Repository Clones

You can create your repository in one of two ways:

1. Initialise a new one
2. Clone an existing one

We already created our own repository using the 'quick start' section above. Let us see how you can clone the repository we made earlier (we will assume that you are still in the **git-sandbox** directory at this time):

Listing

```
cd ..
git clone git-sandbox git-sandbox-clone
```

You should see a message like this:

Listing

```
Initialized empty Git repository in /tmp/git-sandbox-clone/.git/
```

Now if we enter the cloned repository

Often you will want to work with others to build your software. Typically the

1.3.2 Git under windows

Here are some generic notes on using git under windows you should install msys git app. In windows explorer go to c:\Documents and Settings\<your user>\

Make a directory called .ssh

In that directory create a text file called 'config' (note it has no extension) and put the following content into it:

Listing

```
Host <host name>
  User <user name>
```

```
    HostName <host name>
    Port <port>
```

Replaces items in angle brackets above as appropriate.

Now copy your id_dsa into this directory (it should be a unix style one so you may need to convert from putty style private key, though just try with your existing one first).

Open the msys git shell then go to the directory where you want to check out your project to. For example to check it out to c:\dev\foo do

Listing

```
cd /c/
mkdir dev
cd dev
```

Now clone the directory:

Listing

```
git clone git@foo:bar.git bar
```

Make sure to type 'yes' in full when it asks you if you are sure you want to continue connecting.

Then enter your passphrase when prompted.

Wait a few minutes while it checks out.

Thereafter you use the git commands from the msys shell as normal. There is also a tortoisegit explorer integration for windows you can try but I havent used it and don't know how well it works.