**1. Git-cola**

Git-cola is developed in Python and comes with the usual *pull*, *push*, *commit* functions. It also comes with a diff-viewer and file staging mode.

In Ubuntu, git-cola is found in the repository and can be installed via Ubuntu [Software](http://www.ebay.com/sch/i.html?_nkw=software) Center, or via the command:

Sudo apt-get install gitcola

## 2. Gitg

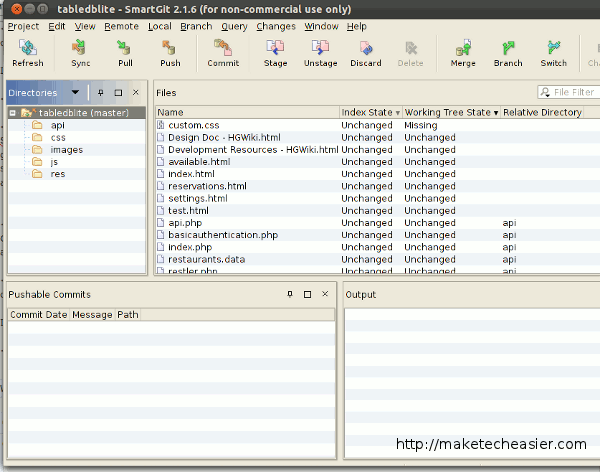
gitg is a simple application. You can commit changes and view the repositories in graphical display. There is also the diff viewer and a file browser, that’s all. Some people love this as a git viewer because of the colorful graphs and great organization, making it easier to see the changes.

In Ubuntu, gitg can be installed with the command:

**sudo** **apt-get install** gitg

## 3. SmartGit

SmartGit is a cross-platform java-based full featured git client. It comes with almost everything you need to manage your git repository. You can pull, push, commit, track changes, clone, stage, branch and access [remote](http://www.ebay.com/sch/i.html?_nkw=remote" \o "Link added by VigLink)repo. It also comes with support for GitHub, Beanstalk, Codebase and Unfuddle. Too bad, it doesn’t work with BitBucket, the hosting provider that I am using.



If you are using Ubuntu Oneiric and have installed the OpenJDK instead of the Sun/Oracle Java runtime, you will find that SmartGit won’t run at all. Here’s the fix.

1. [Download SmartGit](http://www.syntevo.com/smartgit/download.html).

2. Extract the tar file to your home folder.

3. Open a file manager and navigate to the bin folder inside the smartgit directory. Open the “smartgit.sh” file with a [text editor](https://www.udemy.com/atom_editor/).

Change the line

#SMARTGIT\_JAVA\_HOME=**/**usr**/**lib**/java**

to

SMARTGIT\_JAVA\_HOME=**/**usr**/**lib**/**jvm**/**java-1.6.0-openjdk**/**jre

At the last line of code, change the line:

$\_JAVA\_EXEC $\_VM\_PROPERTIES -Xmx${MAXIMUM\_HEAP\_SIZE} -Dsmartgit.vm-xmx=${MAXIMUM\_HEAP\_SIZE} -jar "$SMARTGIT\_HOME/lib/smartgit.jar" "$@"

to

$\_JAVA\_EXEC $\_VM\_PROPERTIES -Xmx${MAXIMUM\_HEAP\_SIZE} -Dsmartgit.checkIncompatibleJava=**false** -Dsmartgit.vm-xmx=${MAXIMUM\_HEAP\_SIZE} -jar "$SMARTGIT\_HOME/lib/smartgit.jar" "$@"

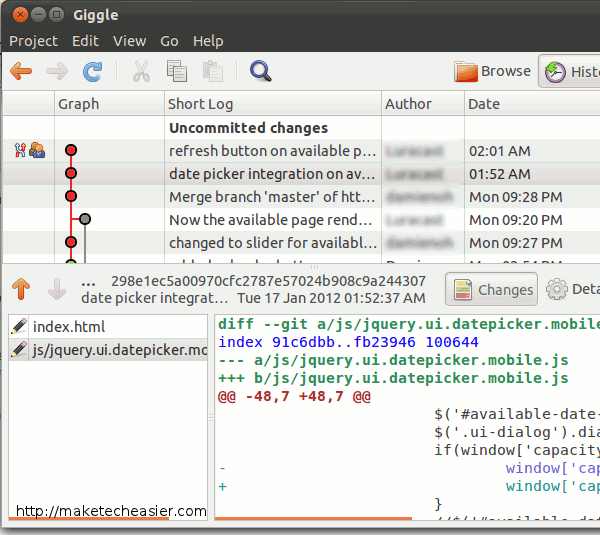
Save the file and close it.

4. Lastly, click the “smartgit.sh” file and select Run when prompted.

Smartgit is free for non-commercial use.

## 4. Giggle

Giggle is more of a git viewer. You can view the files and changes that you have previously committed, but you are not able to commit changes or pull/push from/to the git server.

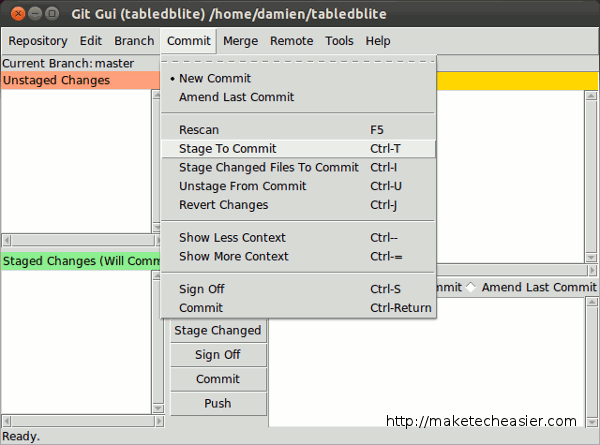


In Ubuntu, you can install Giggle with the command:

**sudo** **apt-get install** giggle

## 5. Git Gui

Git Gui is lightweight and simple, yet one of the most feature complete git client among the list. When you first run it, it will prompt you to either create a new repository, open an existing repo or clone a [remote](http://www.ebay.com/sch/i.html?_nkw=remote) repo. Once opened, you will be able to view the master and branch changes and history, and the database statistics. You can also stage, merge, commit, push changes to the [remote](http://www.ebay.com/sch/i.html?_nkw=remote) server. It might not have a long feature list as SmartGit, but it is definitely more user-friendly and easier to navigate.



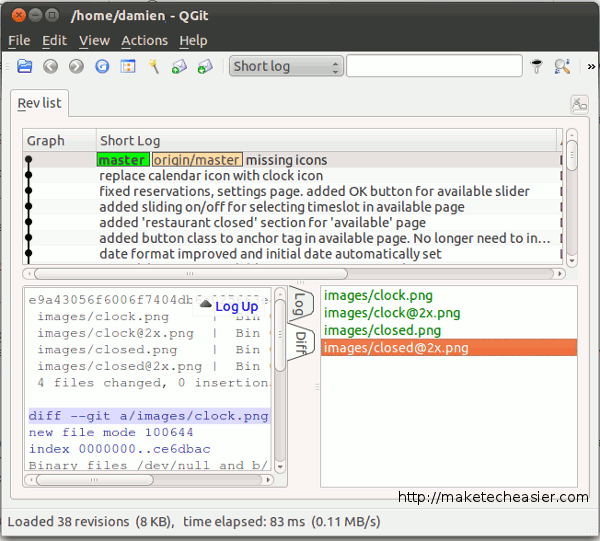
Git Gui is available in the Ubuntu repository, so you can install via the command:

**sudo** **apt-get install** git-gui

To run it, press “Alt + F2” and type “git gui” (without the quote).

## 6. qGit

qgit is yet another git viewer based on the qt framework. If you are on using Gnome, you will find that the interface and icons are very similar to those in KDE (since KDE is based on qt framework as well).



qGit comes with a diff viewer and a revision log viewer, and you can easily switch between both. There is also an Action Builder where you can add and save custom Actions that you use repeatedly. For example, you can create Actions to pull, push, commit the changes rather than doing it on the terminal or other git client.

### What is Virtualenv?

Virtualenv is a tool to create isolated Python environments, it's perhaps the

easiest way to configure a custom Python environment.

Virtualenv allows you to add and modify Python modules without access to the

global installation.

### What does it do?

The basic problem being addressed is one of dependencies and versions, and

indirectly permissions.

Imagine you have an application that needs version 1 of LibFoo, but another

application requires version 2.

How can you use both these applications?

If you install everything into /usr/lib/python2.7/site-packages (or whatever your

platform’s standard location is), it’s easy to end up in a situation where you

unintentionally upgrade an application that shouldn’t be upgraded.

Or more generally, what if you want to install an application and leave it be?

If an application works, any change in its libraries or the versions of those

libraries can break the application.

Also, what if you can’t install packages into the global site-packages directory?

For instance, on a shared host.

In all these cases, virtualenv can help you.

It creates an environment that has its own installation directories, that doesn’t

share libraries with other virtualenv environments

### How to install Virtualenv?

There are a few ways to install virtualenv on your machine.

You can use either the source tarball, pip or by using easy\_install.

##### **Easy\_Install**

$ sudo easy\_install virtualenv

Searching for virtualenv

Reading http://pypi.python.org/simple/virtualenv/

Reading http://www.virtualenv.org

Reading http://virtualenv.openplans.org

Best match: virtualenv 1.8.2

Downloading http://pypi.python.org/packages/source/v/virtualenv/virtualenv-1.8.2.tar.gz...

processing virtualenv-1.8.2.tar.gz

.....

....

Processing dependencies for virtualenv

Finished processing dependencies for virtualenv

##### **Source ball installation**

Get the latest version from here: http://pypi.python.org/packages/source/v/virtualenv/

wget http://pypi.python.org/packages/source/v/virtualenv/virtualenv-1.8.tar.gz

tar xzvf virtualenv-1.8.tar.gz

python virtualenv-1.8/virtualenv.py $HOME/env

##### **Pip installation**

pip install virtualenv

### Usage

To create virtual environments, you can use the virtualenv command.

Create an environment called "foobar":

virtualenv foobar

Activate the environment by sourcing its activate script, which is located in the

environment's bin/ directory:

source foobar/bin/activate

This will change your $PATH so its first entry is the virtualenv’s bin/ directory.

If you install a package in your virtual environment, you'll see that executable

scripts are placed in foobar/bin/ and eggs in foobar/lib/python2.X/site-packages/

easy\_install yolk

Yolk is a small command line tool which can, among other things, list the

currently installed Python packages on your system:

yolk -l

Virtualenv inherits packages from the system's default site-packages directory.

This is especially useful when relying on certain packages being available,

so you don't have to go through installing them in every environment.

To leave an environment, simply run deactivate:

deactivate

If you execute he yolk command now, you will see that it won't work because the

package was installed only in your virtual environment.

Once you reactivate your environment it will be available again.

What is a webframework?

A "web framework" offers a set of APIs for writing your own custom code in such a way that it can be called via the Web. Usually a framework will deal with common details such as HTTP header parsing, URL routing and so forth.  
  
A web server is a piece of software which listens on a network port for incoming HTTP requests and responds to them. Most web servers have a default mode where they will interpret the incoming request as a path on the filesystem and return the file at that path, but they can usually be configured to do something else with the request instead (pass it to a CGI script, proxy it to another server, run some custom module code etc).