

Git and GitHub

What's Version Control?

Version control also known as *source control* or *source code management* or *revision control* in its simplest form is saving your first draft, and then saving it as draft 2 and continuing on with your editing. This allows you some means of going back to a previous version if you decide that the direction you took in your document is not the direction you want to go, and you want to continue on from where your first draft first was. Version control in a more granular manner is being able to see each change as it was made. In reality, it ends up being likely that you want something between these two extremes.

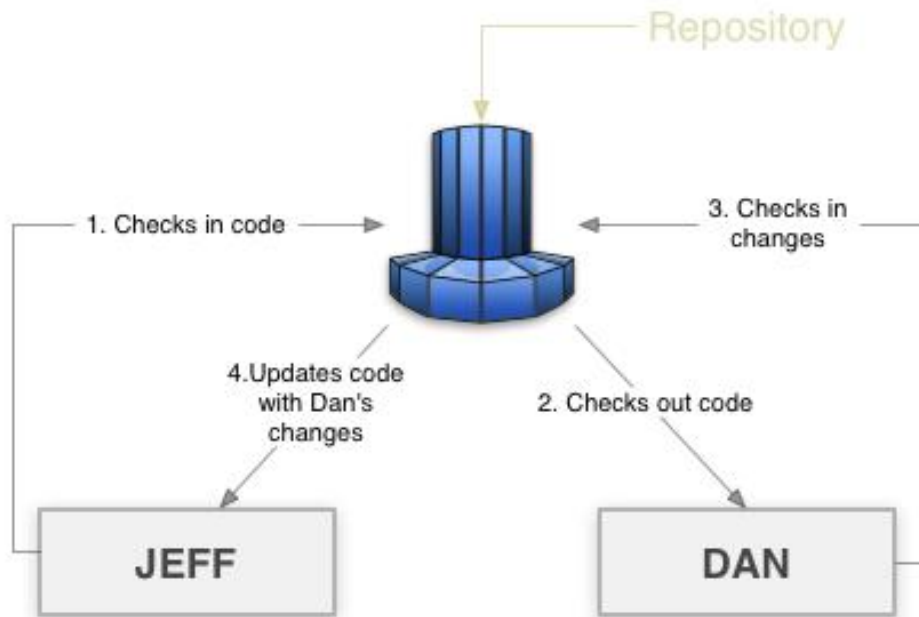
Thus, *version control* is a system that maintains versions of files at progressive stages of development. Every file in the system has a full history of changes, and can easily be restored to any version in its history.

The simplest *version control system* consists of a *repository* where all the files and their versions live. Quite simply, a *repository* works like a database; it can return any version of any file within, or a history of changes for any file, or indeed a history of changes across the entire project.

The repository's users can *check out a working copy*, which is a copy of the latest files to which users can make changes. After making some changes, they can then *check in* (or *commit*) the changes back to the repository, which creates a new version with metadata about the files that were changed and the person who changed them.

Each user has a full copy of the *repository* on their local machine. Generally, you will commit changes to your local repository and, once it is complete, *push* your work to the shared repository for your team. You can also *pull* changes from other repositories.

Reference: http://hoth.entp.com/output/git_for_designers.html



[Figure: A basic source control system

Source: <http://hoth.entp.com/output/scm.png>]

Some vocabulary you need to remember:

- *version control* or *source control* or *source code management* or *revision control*
- *repository*
- *check out* a *working copy*
- *check in* (or *commit*)
- *push* and *pull*

For details see here –

http://en.wikipedia.org/wiki/Version_control_system#Common_vocabulary

What's Git?

Git is an open source *version control system* designed to handle very large projects with speed and efficiency, but just as well suited for small personal repositories (collection of resources that can be accessed to retrieve information); it is especially popular in the open source community, serving as a development platform for projects like the *Linux Kernel*, *Ruby on Rails*, *WINE* or *X.org*.

Git falls in the category of *distributed source code management tools* (which means that it can work almost entirely offline). You don't need to know what that means to use this eBook, but if interested check this URL –

http://en.wikipedia.org/wiki/Distributed_revision_control

Every Git working directory is a full-fledged *repository* with complete history and full revision tracking capabilities, *not* dependent on network access or a central server. Still, Git stays extremely fast and space efficient.

For a more detailed look at Git, read Git on Wikipedia – [http://en.wikipedia.org/wiki/Git_\(software\)](http://en.wikipedia.org/wiki/Git_(software))

Downloading and Installing Git

To download and install Git, the precompiled packages are available here: <http://git.or.cz/>

Select the relevant package for your operating system.

For a Windows box:

Git still has some issues on the Windows platform but for normal usage the **msysgit** package shouldn't let you down.

Download and install it from the url: <http://code.google.com/p/msysgit/downloads/list?q=full+installer+official+git>

and select the **current version available**. Download – **Git-1.7.11-preview20120710.exe**

Install by running the EXE installer. Accept the default install directory. When you get to the “Select Components” setting screen, it is recommended for Windows users to use the “Git Bash Here” option. In the “Adjusting your PATH environment” screen, it is recommended for Windows users to use the “Use Git Bash only” option. Use other default options.

Note that Git also runs on top of Cygwin (a POSIX emulation layer), although it is noticeably slower, especially for commands written as shell scripts. This is primarily due to the high cost of the fork emulation performed by Cygwin. However, the recent rewriting of many Git commands implemented as shell scripts in C has resulted in significant speed improvements on Windows. Regardless, many people find a Cygwin installation too large and invasive for typical Windows use. **Reference:**

[http://en.wikipedia.org/wiki/Git_\(software\)](http://en.wikipedia.org/wiki/Git_(software))

For Mac OS X users:

1. Michèle Garoche says *use git-osx-installer* from google (<http://code.google.com/p/git-osx-installer/>)

Click on the show all versions to see the leopard and snow leopard versions and choose the right one for your architecture (that is intel or ppc for leopard, intel for snow leopard).

The snow leopard version works on Lion and Mountain Lion, though you may have to check that the installed git (in /usr/local) is in your path. For this just type in Terminal.app:

```
which git
```

If it answers, all done. If not add it to your .bashrc file as following:

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

and source it in your .bash_profile:

```
if [ -f ~/.bashrc ]; then
. ~/.bashrc
fi
```

2. Hector Sansores recommends an easier way to install Git on Mac OS X is using the git-osx-installer (**Note:** This is available only on Leopard.).

<http://code.google.com/p/git-osx-installer/>

3. Cynthia Sadler shows you how to install Git on Mac OS X 10.4 (Tiger) –

<http://geekythoughtbubbles.blogspot.com/2009/02/how-to-install-git-on-mac-os-x-104.html>

For Windows Vista users:

Al Snow says that *you should remember to use "run as administrator" right button; 3rd option down) when you open the browser prior to installing msysgit as a regular user.*

Create a local folder

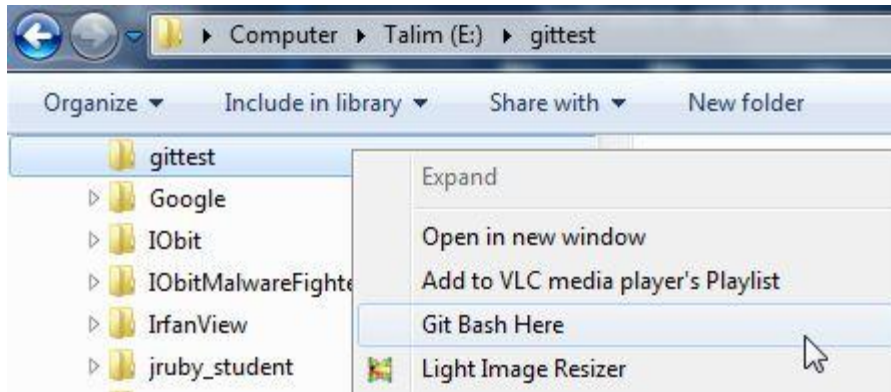
Create a new empty directory in any convenient location on your hard disk (I created and use the folder **gittest** on E: of my hard disk).

Let us start using Git

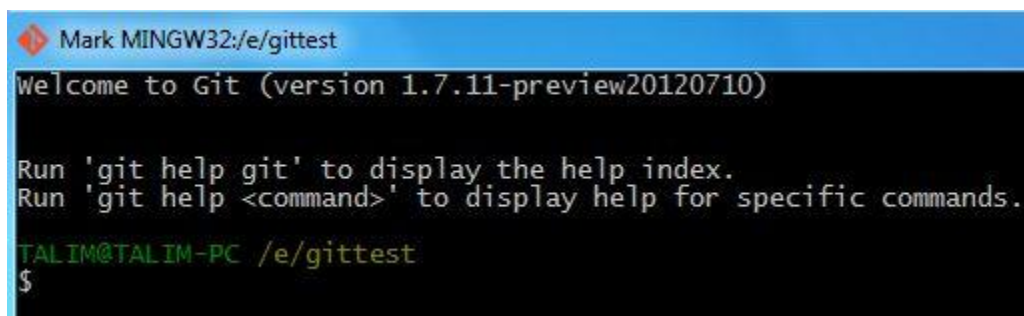
Git is primarily a command line tool and most of the examples in this eBook will show the command line versions of the functions.

Windows users:

In Windows Explorer, right mouse click on your local folder (eg. **gittest**) and choose “Git Bash Here” as shown in the following screenshot:



This opens up a command window (bash shell) as shown in the next screenshot:



Windows users should note that the bash shell is significantly different from the cmd.exe shell you may be used to. Refer to the documentation here –

<http://www.gnu.org/software/bash/manual/bashref.html>

Other Operating systems:

Here, the users would have to open the console or the Terminal. **Remember**, you need to be inside the **gittest** folder.

Confirm Git version

Let us first check and confirm the version of Git that we have installed. Type as shown:

```
$ git --version
git version 1.7.11.msysgit.1
```

Asking for help

If you need help with any of the commands, you can type '--help' and it will show you the *man* page in your browser window. You can also type '**git help command**' for the same thing.

```
$ git log --help
$ git help log
```

Introduce yourself to Git

For all operating users, you now need to identify yourself to Git (you need to do this only *once*) so that it can properly label the commits you make later on. With the bash shell still open type in the following (I am using SMTalim and satish@satishtalim.com below):

```
$ git config --global user.name "Your Name Here"
$ git config --global user.email your@email.com
```

Substitute in your own user name and email (Note that Git does not allow accented characters in user name). Git saves your email address into the commits you make. We use the email address to associate your commits with your GitHub account. *This will set the info stored when you commit to a Git repository.* Git has now been set up. You will use these with GitHub later and, when you learn how, for convenience preparing and sending patches as well as to identify your repositories.

Add some additional settings

Since we are going to work with Ruby code which is not whitespace sensitive, the following configuration tells Git to ignore differences in whitespace that otherwise may be odd. Type:

```
$ git config --global apply.whitespace nowarn
```

We can also add some colors:

```
$ git config --global color.status auto
$ git config --global color.branch auto
```

Let us confirm what we added to **config** by typing:

```
$ git config --list
core.symlinks=false
core.autocrlf=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
pack.packsizelimit=2g
help.format=html
http.sslcainfo=/bin/curl-ca-bundle.crt
sendmail.smtpserver=/bin/msmtp.exe
diff.astextplain.textconv=astextplain
rebase.autosquash=true
user.name=Satish Talim
user.email=satish@rubylearning.org
apply.whitespace=nowarn
color.status=auto
color.branch=auto
```

Notice that Git has already added some settings, by default on Windows. Also, the git config settings are stored in a .gitconfig file in the user's home directory.

For full details on other configuration options, refer to:

<http://www.kernel.org/pub/software/scm/git/docs/git-config.html>

or

http://book.git-scm.com/5_customizing_git.html

What's GitHub?

With GitHub you can host your public and private projects there and use it to collaborate on projects. In this short eBook, you'll learn the essential features you'll end up using every day. You'll come away ready to host your projects and contribute to other projects, and feeling like a GitHub insider!

Set up your GitHub account

Go to <https://github.com/signup/free> and sign up for a free account. Remember, this account can have unlimited public repositories and public collaborators (the total number of users who may read, write and fork your public repositories). Also, make sure that Javascript is enabled in your browser.

I signed up with **satish@satishtalim.com**. *Please use your own real email address.* The username I typed was **SMTalim**. *You must use a different username.* If you want multiple GitHub accounts, refer to –

<http://github.com/guides/multiple-github-accounts>

The GitHub *account* I just created is at –

<https://github.com/SMTalim>

Go to your GitHub account and please click on “Edit Your Profile” and fill in the relevant details in *your* GitHub account.

Creating a new repository

Now, I would like to work on a simple local project called **gittest**, both on my local machine and on GitHub. For this, I will need to create a repository. *Please create your own local project with a different name.*


You are already logged into your GitHub account you just created above. If you click on the “dashboard” link at the top right-hand-side of your GitHub page, you will see something like the following screenshot:



Click on – “New Repository” to create a new public repository.


This is shown in the next screenshot:


Owner **Repository name**

 SMTalim

Great repository names are short and memorable. Need inspiration? How about [glowing-robot](#).

Description (optional)

☒ **Public** 
Anyone can see this repository. You choose who can commit.

☐ **Private** 
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**
This will allow you to `git clone` the repository immediately.

Add .gitignore: **None**

Create repository

Note: Do not select the “Initialize this repository with a README” option.

A new screen loads as follows:

We recommend that every repository has a **README**, **LICENSE**, and **.gitignore**

Create a new repository on the command line

```
touch README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/SMTalim/gittest.git
git push -u origin master
```

Push an existing repository from the command line

```
git remote add origin https://github.com/SMTalim/gittest.git
git push -u origin master
```

NOTE: In the examples that follow, a sample repository name of **'gittest'** will be used, but it is NOT necessary that you name your repository **'gittest'**.

Back to your local gittest folder

Note: In the steps shown below, please remember to replace **gittest** with the name of your repository on GitHub.

In the already open Bash shell, type the commands as given in the screen above namely “Create a new repository on the command line”.

For the last command it will ask you for your username and password as follows:

```
$ git push -u origin master
Username for 'https://github.com': SMTalim
Password for 'https://SMTalim@github.com':
```

Exercise 1

Try and find out the public clone url of mattetti / merb-book. What is it?