**Technical blog – Git Hub**

**Git**

If you search for ‘git’ on Wikipedia, you will find responses ranging from a village in Iran, a post-punk band, British slang for a “contemptible person” and, for the high-brows[[1]](#footnote-1), a ‘geometric invariant theory’. So when faced with **git software** as a ‘foundation’ tool for my ‘learning to code’ journey, I was a bit perplexed to say the least…

So, what is it? Well, git is quite simply, git is an open sourced Distributed Version Control System (DBVC). You got that? Good. Ok, so maybe that hasn’t helped much… So I’’ll try to explain… Geeks[[2]](#footnote-2), please look away… ;)

**Nail-drying version**

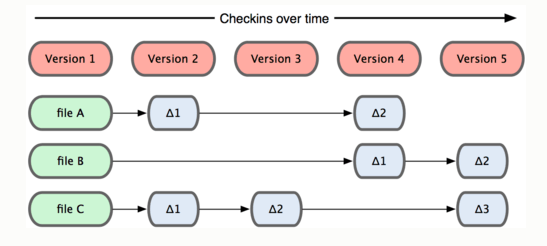
Git enables you to engage in collaborative working by copying (or ‘cloning’) repositories (i.e. files/databases) of code from a main repository stored on the internet (typically on GitHub), to work on individually, on your own pc, and then save (push/pull) changes, amongst collaborators, back to that main repository.

**Nail-painting version**

**VCS**

Let’s take a step back – the concept of Version Control is something we’re nearly all familiar with. Most of us ‘worker bees’ are familiar with Word and Excel and that every time you save a change, a new version is created (either automatically date/time stamped or because a user has renamed it: “Save As”), all of which are stored in directories (i.e. folders). Perfect for tracking your own changes and going back and forth as required.

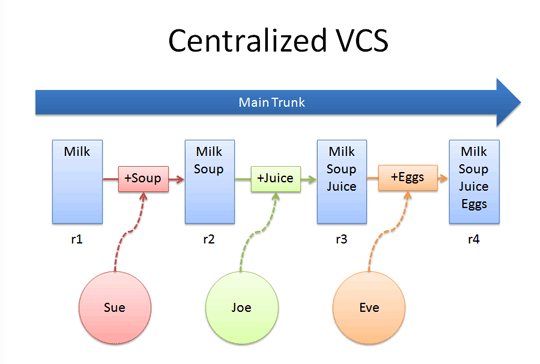
So as shown below, 3 files (A, B & C) are changed over time (Δ1, Δ2, Δ3) to create new versions of each files (Version 1, 2, 3, 4 & 5).



And then “I” became “we”, as collaborative working became the ‘new black’. This resulted in multiple users accessing and changing the same master documents. And as the associated risks of many users tweaking a document increased (i.e. integrity and ownership issues), the need for more robust VCS became clear, particularly in the software world where coding collaborations were (and are) becoming the norm, either within or outside an organisation.

**Centralised VCS**

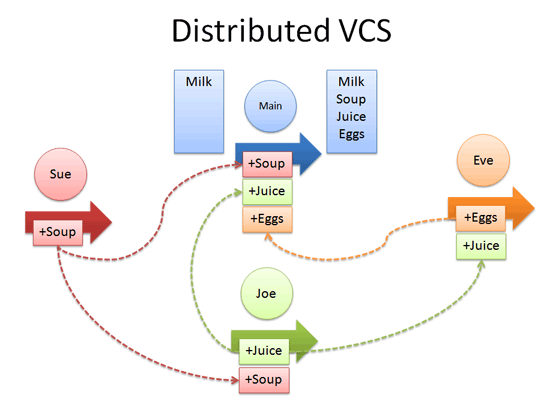
In a centralised VCS, everyone syncs and checks changes to the database or central repository (commonly referred to as a “repo”), also known as the main trunk (as in tree!). In this example, Sue adds soup, Joe adds juice and Eve adds eggs.



And as you can see, an issue here is that Sue’s changes need to be updated to the common repo before Joe or Eve can see them. And with all this reliance on one common repo, what happens if the sever goes down? Boom!

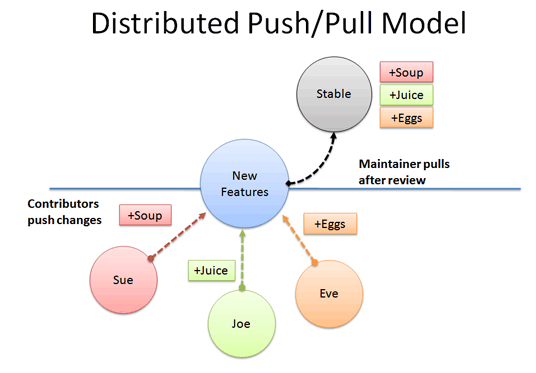
**A Distributed Version Control System (DVCS)**

In a DVCS, every programmer has their own version of the common repo that they can work on, pushing and pulling changes (or syncing) with the common repo as and when appropriate.



GitHub is fully distributed, so (almost) everything is local. It thinks of its data more like a set of snapshots of a mini filesystem. Every time you commit (or save a change/state of your project), Git takes a picture of what your files look like at that moment, giving the version a unique ID number.

It’s only when you push or pull that you need the network/internet connection,



Otherwise, the following activities can all be done off-line:

* Viewing file history
* Commit (save new changes)
* Merging/Switching branches
* Obtaining other revisions.

And as every user first clones (i.e. copy the common repo) ahead of working on the code/making changes, there are any number of back-ups available at any point in time (addressing the one server = bad news issue).

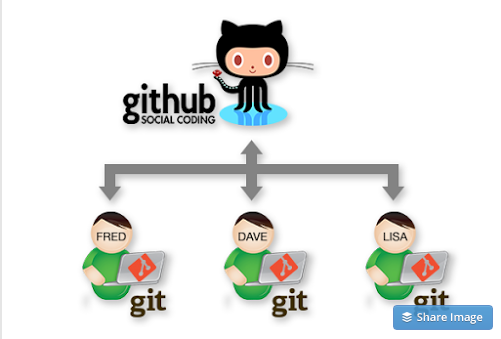
So pretty straight-forward really…? Hpefully the concepts are pretty straight-forward, and there is no substitute for practice. However, the key challenge for me have been around terminology:

* Remote vs local = to me remote means something far from me and doesn’t accurately reflect the ‘master’ repo sitting on the network (i.e. the version you’ll find in the networked hub).
* Add = this is not saving or adding a new change, it is adding a new version to the staging area, which is the ‘holding area’ ahead of pushing changes to the networked repo.
* Commit = this effectively means “Save As”, so unsure why ‘save’ was not used…?
* Git does not equal GitHub:



git is the software you install local (on your computer) which handles version control for you.

gitHub is the networked repositories that all local changes are pushed to (via your local git). Arguably you could push local changes captured in git to Git to any central networked repository (e.g. Bitbucket), but clearly Git have established their own central networked repository “github”) so makes sense to use this, when using git.



Hoping this post has made the git world a little clearer to the wannabe programmers out there. With a huge thanks to the following resources:

<http://betterexplained.com/articles/a-visual-guide-to-version-control/>

<http://www.ericsink.com/scm/source_control.html>

http://www.git-scm.com/doc

https://try.github.io

http://gitreal.codeschool.com/?utm\_source=github&utm\_medium=codeschool\_option&utm\_campaign=trygit

DBC challenge

1. Clever people [↑](#footnote-ref-1)
2. Anyone who knows how a computer works and how to make it work [↑](#footnote-ref-2)