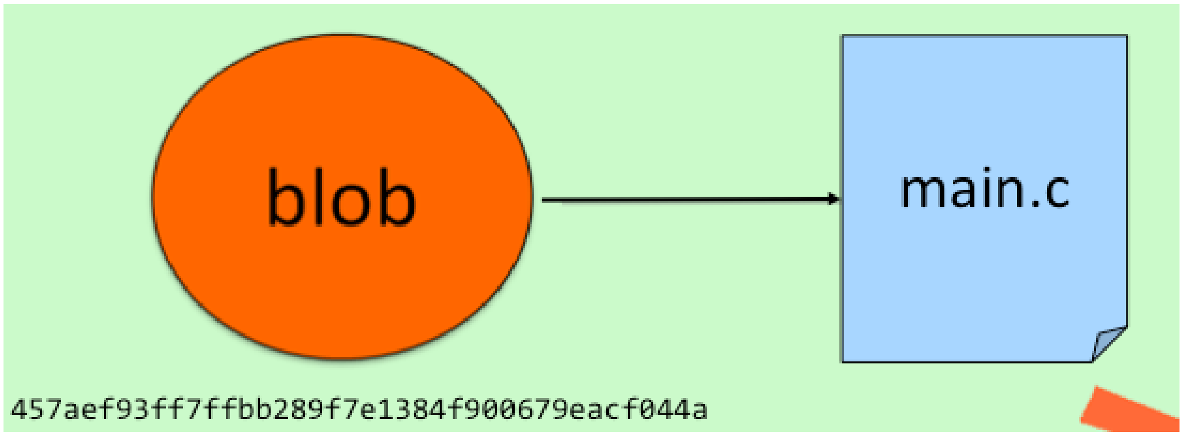
how git stores your data [1]

committed 17 Feb 2009

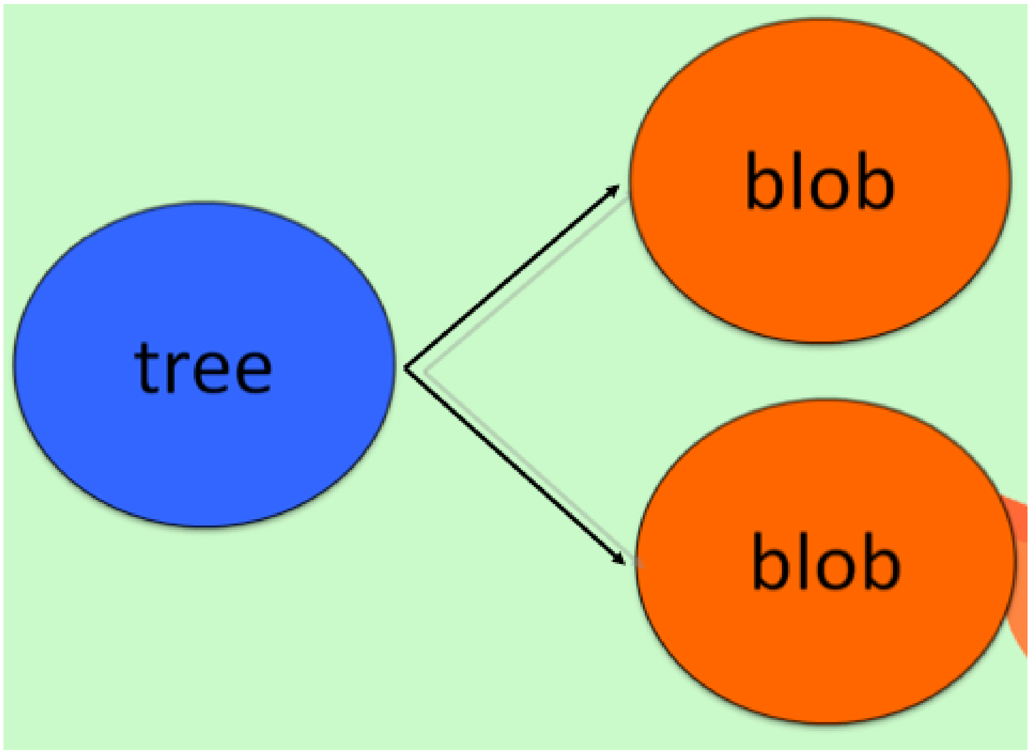
This is an extremely broad overview on how the Git object model works, based mostly on the Git Community Book[2]. Future posts will definitely look into the object model in more depth, but this information is definitely essential to those who are learning Git. The images from this post were taken from my presentation on Open Source Collaboration with Git and GitHub.[3]

The most basic data storage is the blob. Git stores **just** the contents of the file for tracking history, and not just the differences between individual files for each change. The contents are then referenced by a 40 character SHA1 hash[4] of the contents, which means it’s pretty much guaranteed to be unique. Pretty much every object, be it a commit, tree, or blob has a SHA, so learn to love them. Luckily, they’re easily referenced by the first 7 characters which are usually enough to identify the whole string.

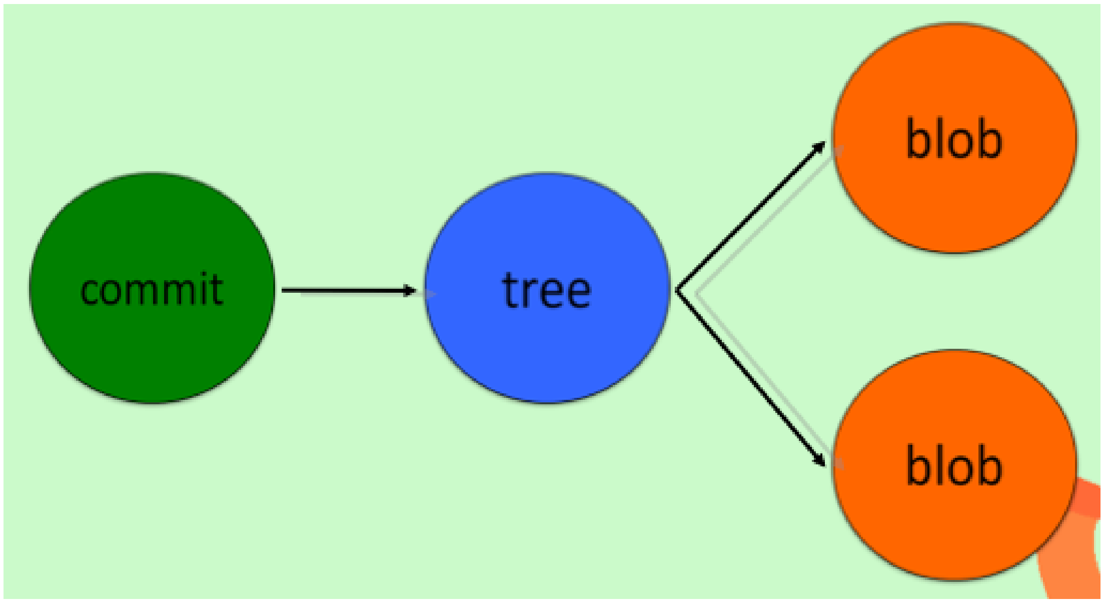
One awesome advantage to storing only the content means that if you have two or more copies of the same file in your repository, Git will only store one copy internally. A blob can be represented like so:



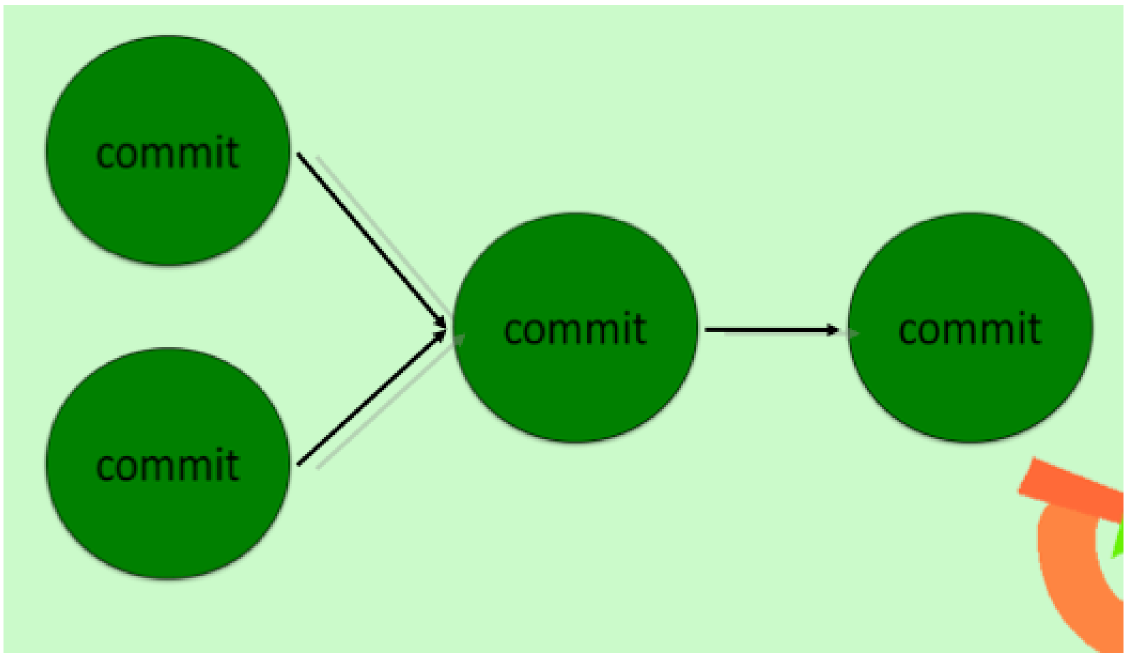
The next object is a tree. These can be thought of as folders or directories: they contain other blobs and trees.



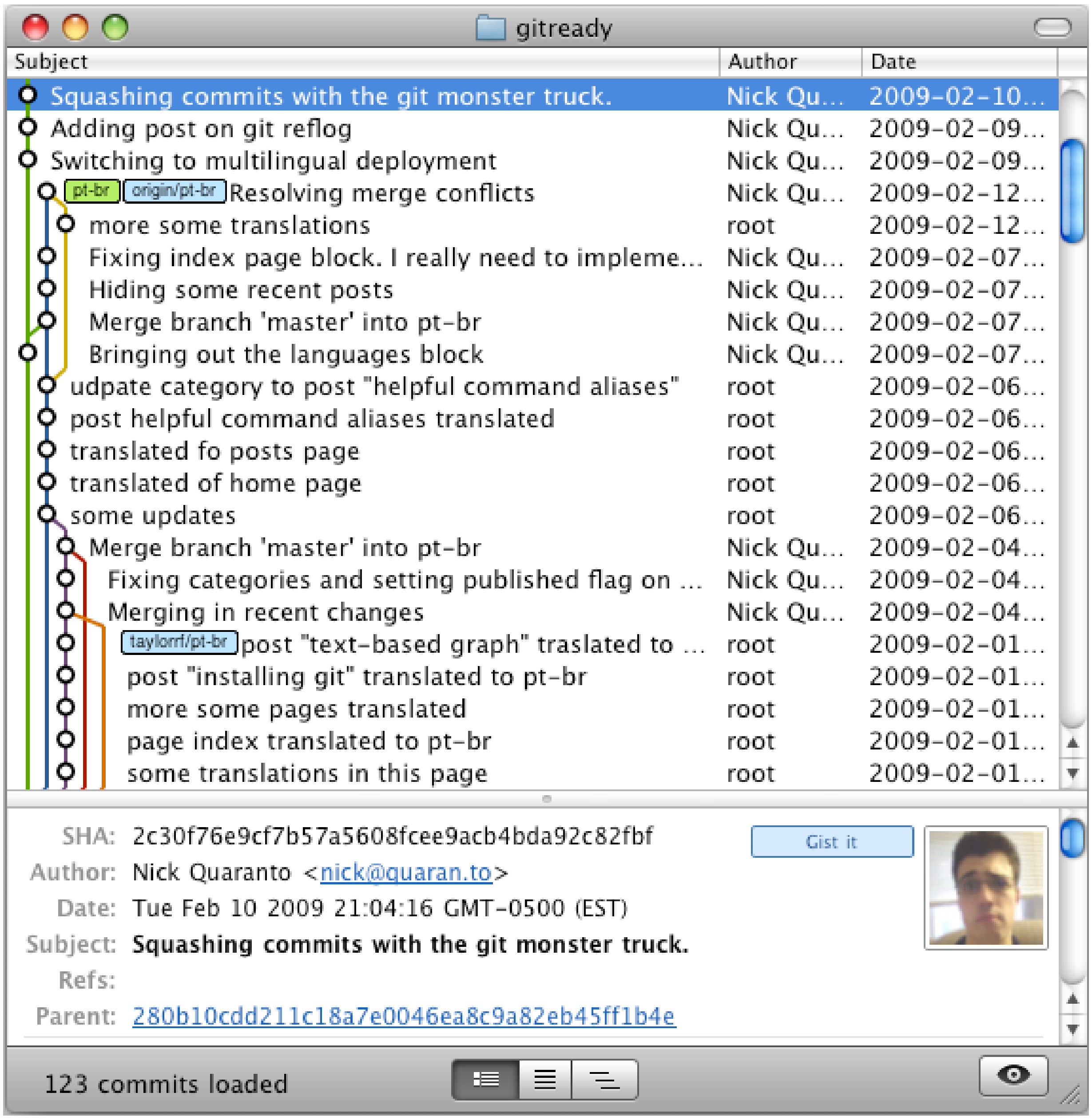
Finally, this brings us to the most important object: the commit. Commits can be thought of as snapshots: they know what the trees looked like at one point in time. They also have some other information associated with them, such as the author, date, and a message.



Commits are organized in a Directed Acyclic Graph[5]. For those who missed that lecture in Data Structures about it, basically it means that the commits “flow” in one direction. Usually this direction is simply the path of history for your repository, which could be very simple or quite complex if you have branches. From a broad standpoint it will look something like this:



This all becomes much more apparent when using a tool like GitX[6]. You can clearly see the commit objects and their associated data, and then drill down from there to see the commit’s tree.



Subject	Author	Date
Squashing commits with the git monster truck.	Nick Qu...	2009-02-10...
Adding post on git reflog	Nick Qu...	2009-02-09...
Switching to multilingual deployment	Nick Qu...	2009-02-09...
pt-br origin/pt-br Resolving merge conflicts	Nick Qu...	2009-02-12...
more some translations	root	2009-02-12...
Fixing index page block. I really need to impleme...	Nick Qu...	2009-02-07...
Hiding some recent posts	Nick Qu...	2009-02-07...
Merge branch 'master' into pt-br	Nick Qu...	2009-02-07...
Bringing out the languages block	Nick Qu...	2009-02-07...
udpate category to post "helpful command aliases"	root	2009-02-06...
post helpful command aliases translated	root	2009-02-06...
translated fo posts page	root	2009-02-06...
translated of home page	root	2009-02-06...
some updates	root	2009-02-06...
Merge branch 'master' into pt-br	Nick Qu...	2009-02-04...
Fixing categories and setting published flag on ...	Nick Qu...	2009-02-04...
Merging in recent changes	Nick Qu...	2009-02-04...
taylorrf/pt-br post "text-based graph" traslated to ...	root	2009-02-01...
post "installing git" translated to pt-br	root	2009-02-01...
more some pages translated	root	2009-02-01...
page index translated to pt-br	root	2009-02-01...
some translations in this page	root	2009-02-01...

SHA: 2c30f76e9cf7b57a5608fcee9acb4bda92c82fbf
Author: Nick Quaranto <nick@quaran.to>
Date: Tue Feb 10 2009 21:04:16 GMT-0500 (EST)
Subject: **Squashing commits with the git monster truck.**
Refs:
Parent: 280b10cdd211c18a7e0046ea8c9a82eb45ff1b4e

123 commits loaded

So now hopefully when you see some of the strange terminology in your commits, you'll understand a little more of how it works. Check out the awesome guide at Git for Computer Scientists[7] and Scott Chacon's[8] talk on Getting Git[9].

blog comments powered by Disqus[10]

1. http://gitready.com/beginner/2009/02/17/how-git-stores-your-data.html
2. http://book.git-scm.com/1_the_git_object_model.html
3. http://litanyagainstfear.com/blog/2008/12/05/open-source-collaboration-with-

git-and-github/

4. http://en.wikipedia.org/wiki/SHA1
5. http://en.wikipedia.org/wiki/Directed_acyclic_graph
6. http://gitready.com/intermediate/2009/01/13/visualizing-your-repo.html
7. http://eagain.net/articles/git-for-computer-scientists/
8. http://jointheconversation.org/
9. http://gitcasts.com/git-talk
10. http://disqus.com/