PeepCode press

$9

Git Internals

Source code control and beyond

by Scott Chacon

FREE PREVIEW. GET THE WHOLE PDF FOR ONLY US$9 AT PEEPCODE.COM

Git Internals

OTHER PEEPCODE PRODUCTS

- RSpec (http://peepcode.com/products/rspec-basics) – A three part series on the popular behavior-driven development framework.
- Rails from Scratch (http://peepcode.com) – Learn Rails!
- RESTful Rails (http://peepcode.com/products/restful-rails) – Teaches the concepts of application design with REST.
- Subscription pack of 10 (http://peepcode.com/products/subscription-pack-of-10) – Save money! Buy 10 PeepCode credits.
- Javascript with Prototype (http://peepcode.com/products/javascript-with-prototypejs) – Code confidently with Javascript!
- Rails Code Review PDF (http://peepcode.com/products/draft-rails-code-review-pdf) – Common mistakes in Rails applications, and how to fix them.

CONTENTS

FREE PREVIEW. GET THE WHOLE PDF FOR ONLY US$9 AT PEEPCODE.COM

The Tree

Directories in Git basically correspond to **trees**.

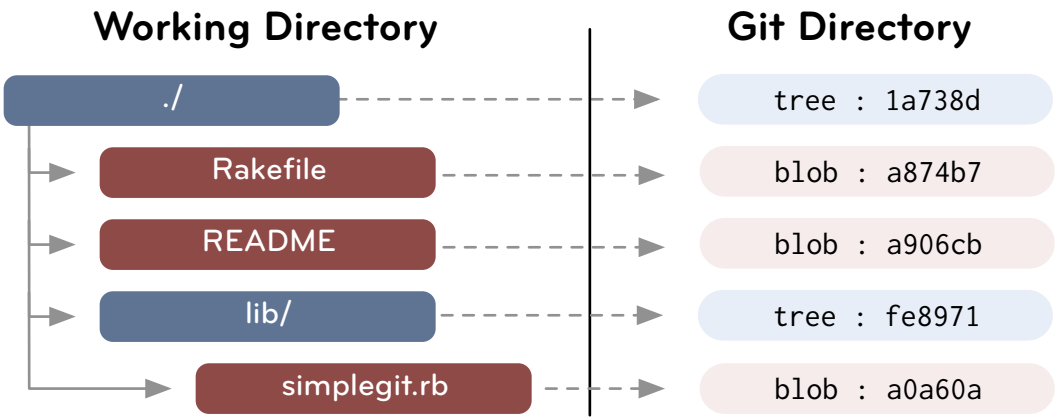


FIG. D **Trees are pointers to blobs and other trees**

A tree is a simple list of trees and blobs that the tree contains, along with the names and modes of those trees and blobs. The contents section of a tree object consists of a very simple text file that lists the *mode*, *type*, *name* and *sha* of each entry.

FIG. E **An uncompressed tree**

The Commit

So, now that we can store arbitrary trees of content in Git, where does the 'history' part of 'tree history storage system' come in? The answer is the **commit** object.

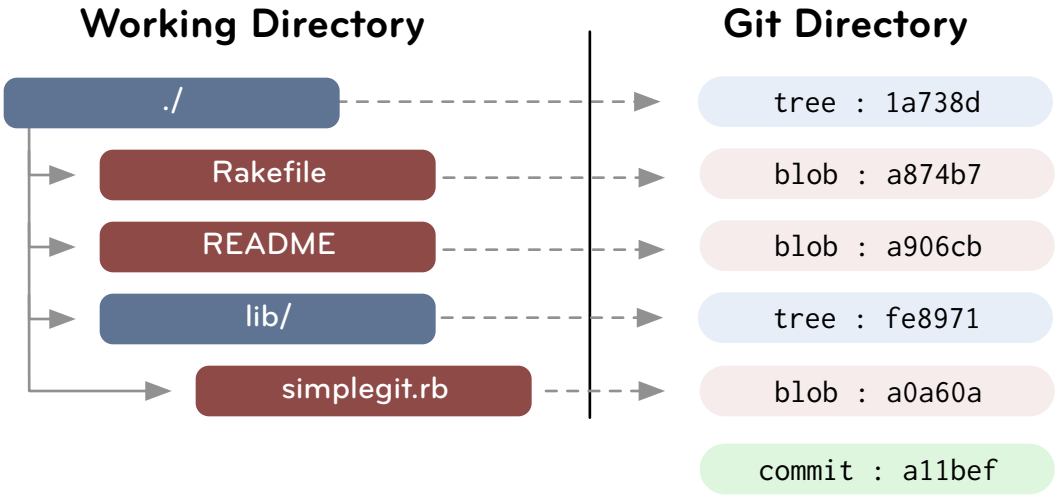


FIG. F A commit references a tree

The commit is very simple, much like the tree. It simply points to a tree and keeps an *author*, *committer*, *message* and any *parent* commits that directly preceded it.

FIG. G **Uncompressed initial commit**

Since this was my first commit, there are no parents. If I commit a second time, the commit object will look more like this:

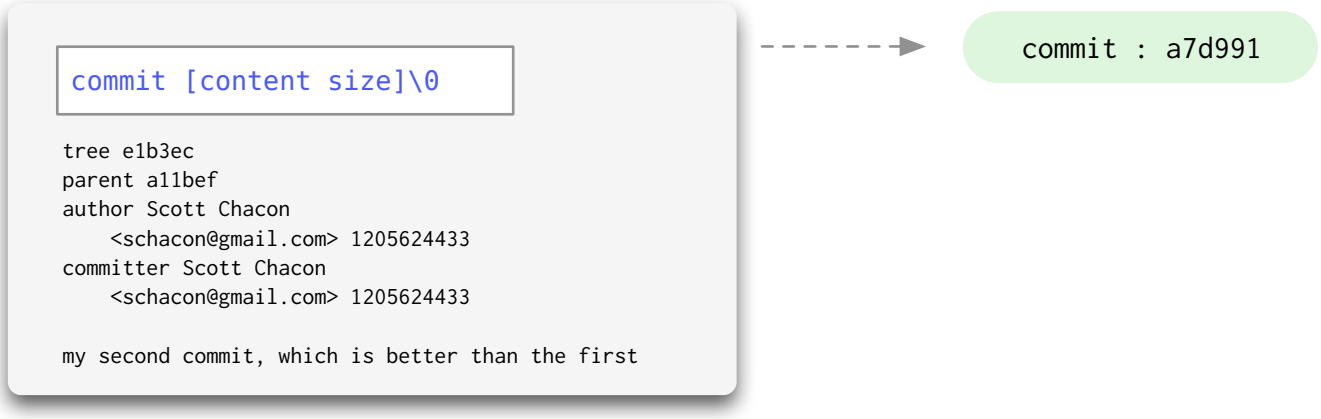
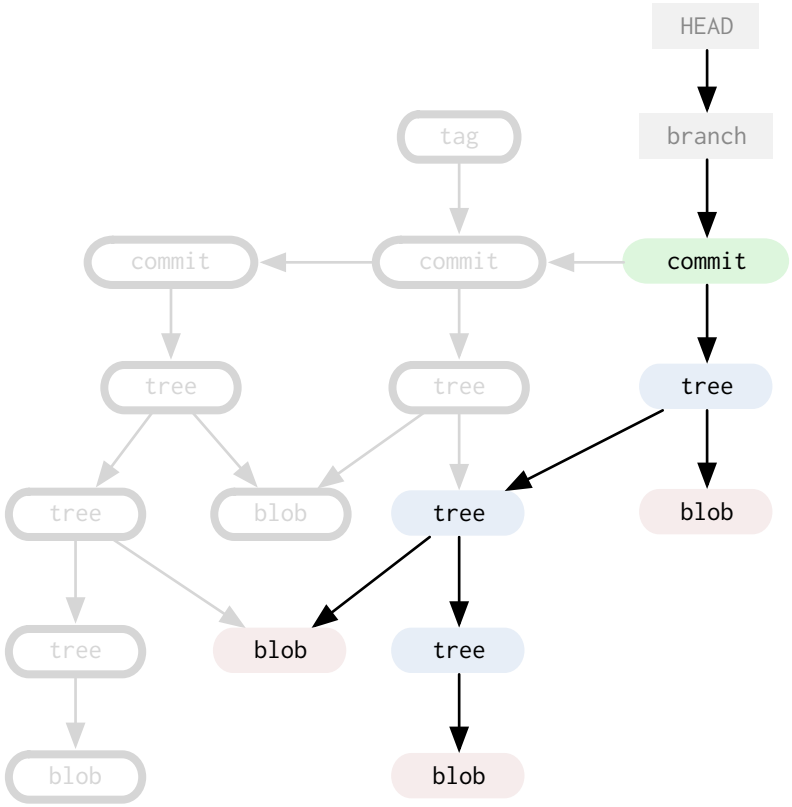


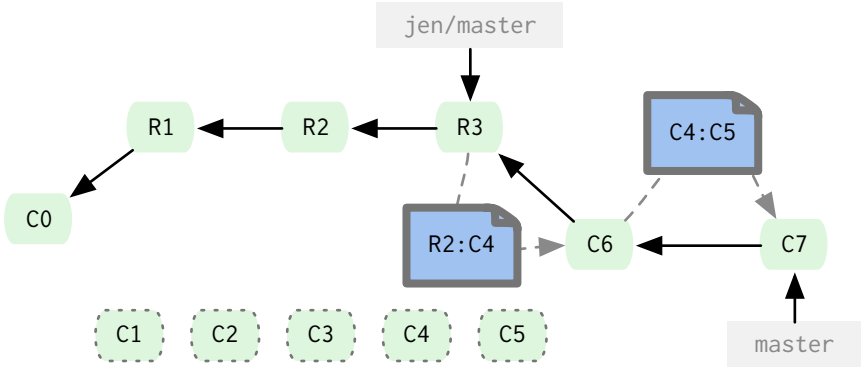
FIG. H **A commit with a parent**

Notice how the *parent* in that commit is the same SHA-1 value of the last commit we did? Most times a commit will only have a single parent like that, but if you merge two branches, the next commit will point to both of them.

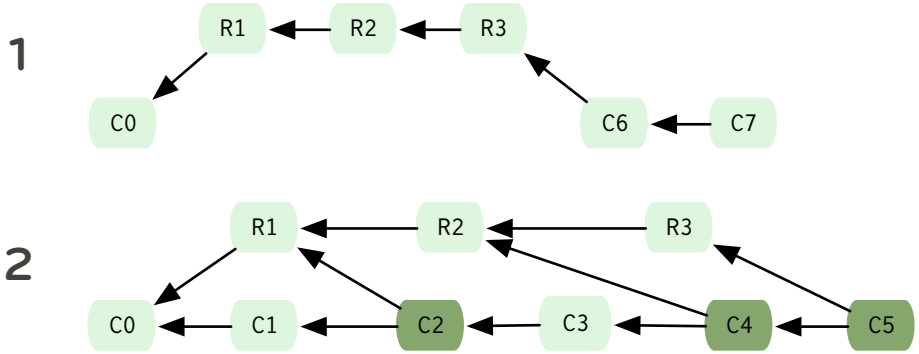


At this point, let's stop to look at the objects we now have in our repository. From this, we can easily recreate any of the three directories we committed by following the graph from the most recent commit object, and having Git expand the trees that are pointed to.

For instance, if we wanted the first tree, we could look for the parent of the parent of the HEAD, or the parent of the tag. If we wanted the second tree, we could ask for the commit pointed to by the tag, and so on.



And finally, we are left with a commit history that looks like Figure 1, rather than Figure 2, which is what we would have if we had merged instead.



You should remember to only do this on local branches before you push or on repositories that nobody has fetch access to – if anyone pulls down the objects that will become abandoned during a rebase, it gets a bit frustrating.

DICTATOR AND LIEUTENANT MODEL

This is a highly hierarchical model where one individual has commit rights to a blessed repository that everyone else fetches from. Changes are fetched from developers by lieutenants responsible for specific subsystems and merged and tested. Lieutenant branches are then fetched by the dictator and merged and pushed into the blessed repository, where the cycle starts over again.

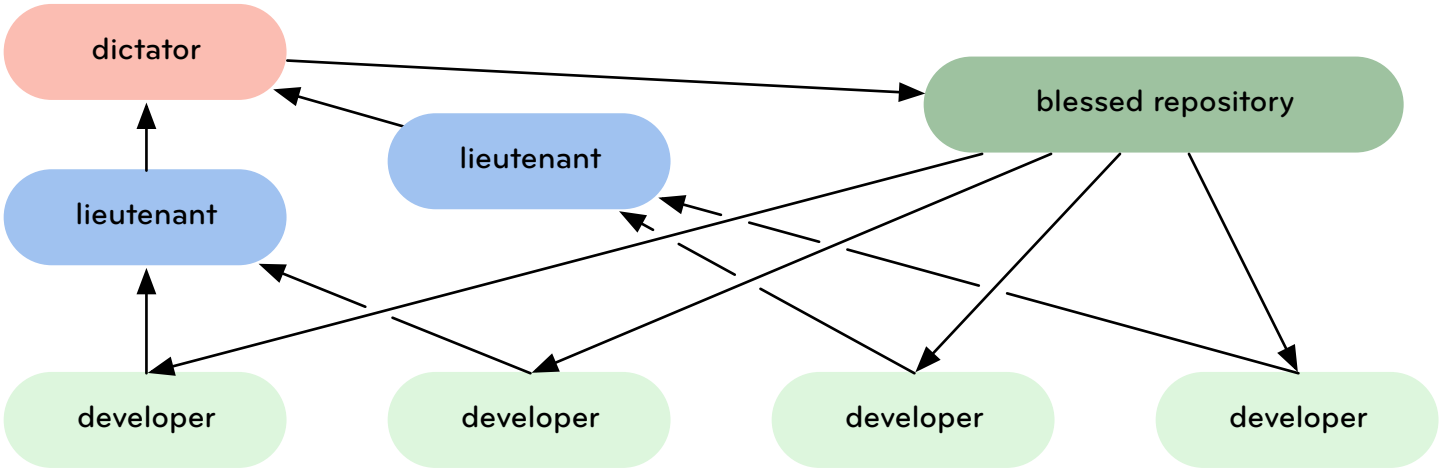


FIG. L **Approved features gradually make their way up the ladder**

This is a model something like the Linux kernel uses, Linus being the benevolent dictator. This model is much better for large teams, and can be implemented with multiple and varied levels of lieutenants and sub-lieutenants in charge of various subsystems. At any stage in this process, patches or commits can be rejected – not merged in and sent up the chain.

INTEGRATION MANAGER MODEL

This is where each developer has a public repository, but one is considered the 'official' repository – it is used to create the pack-