the Git repository (Git refers to them as blobs), and adds that checksum to the staging area:

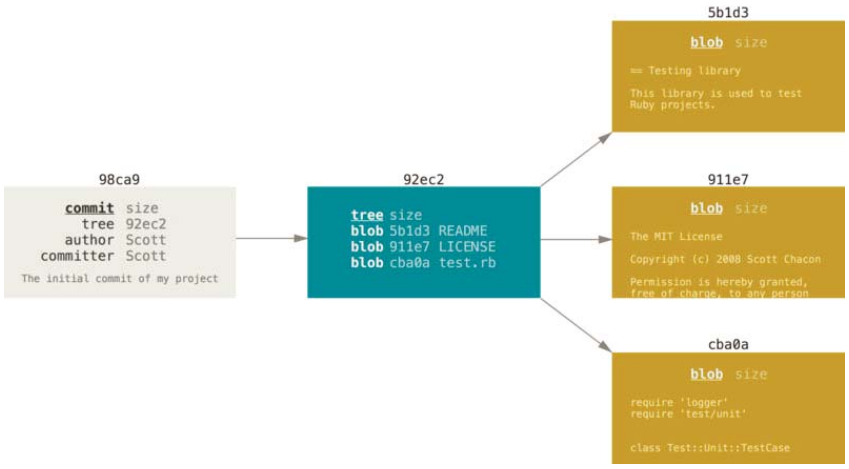
```
$ git add README test.rb LICENSE
$ git commit -m 'initial commit of my project'
```

When you create the commit by running `git commit`, Git checksums each subdirectory (in this case, just the root project directory) and stores those tree objects in the Git repository. Git then creates a commit object that has the metadata and a pointer to the root project tree so it can re-create that snapshot when needed.

Your Git repository now contains five objects: one blob for the contents of each of your three files, one tree that lists the contents of the directory and specifies which file names are stored as which blobs, and one commit with the pointer to that root tree and all the commit metadata.

FIGURE 3-1

A commit and its tree



If you make some changes and commit again, the next commit stores a pointer to the commit that came immediately before it.

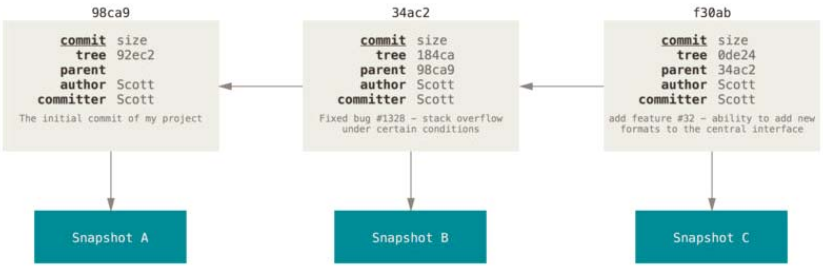


FIGURE 3-2

Commits and their parents

A branch in Git is simply a lightweight movable pointer to one of these commits. The default branch name in Git is `master`. As you start making commits, you're given a master branch that points to the last commit you made. Every time you commit, it moves forward automatically.

> The "master" branch in Git is not a special branch. It is exactly like any other branch. The only reason nearly every repository has one is that the `git init` command creates it by default and most people don't bother to change it.

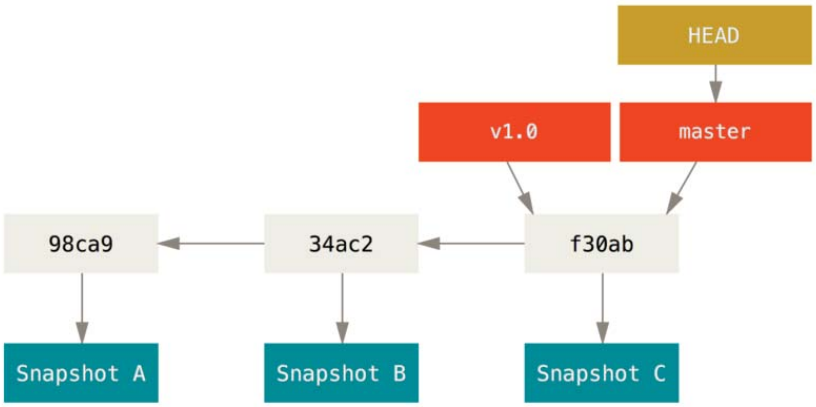


FIGURE 3-3

A branch and its commit history