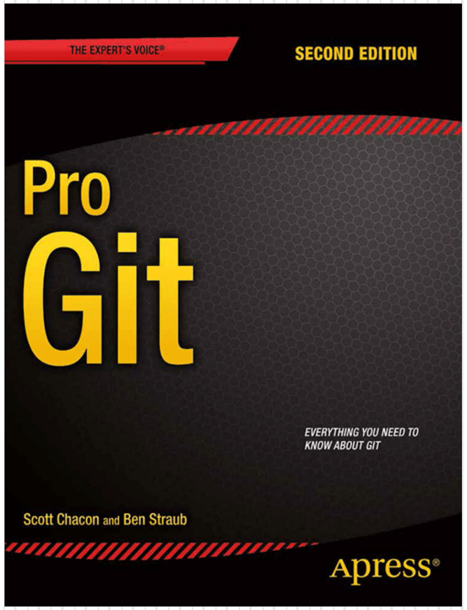
01-2 – git – Local Repositories

Much of this is taken from…

Pro Git

professional version control

https://git-scm.com/book/en/v2


THE EXPERT'S VOICE®
SECOND EDITION
Pro Git
EVERYTHING YOU NEED TO KNOW ABOUT GIT
Scott Chacon and Ben Straub
Apress®

What is git?

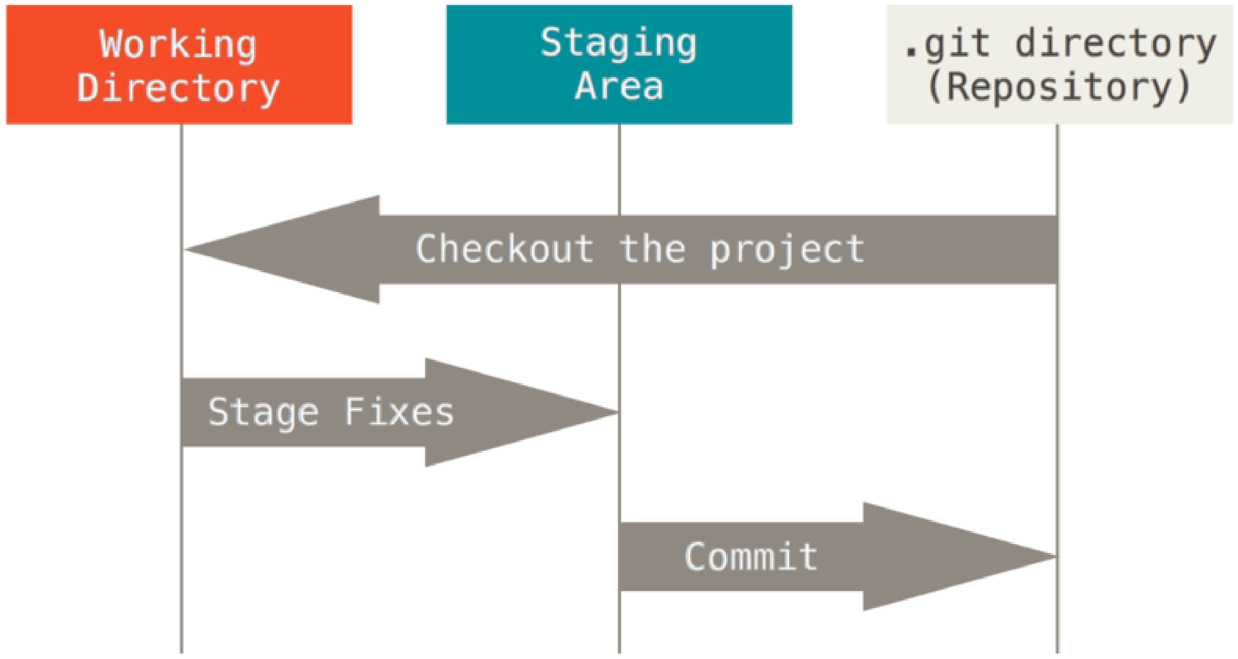
- A distributed revision control system with an emphasis on being fast
- Initially designed and developed by Linus Torvalds for Linux kernel development
- Every Git working directory is
  - a full-fledged repository
  - with complete history and
  - full revision tracking capabilities,
  - not dependent on network access or a central server

Directory Details

- The Git directory (**`.git`**) is where Git stores the metadata and object database for your project
- This is the most important part of Git
- It is what is copied when you **`clone`** a repository from another computer

Git workflow

- Modify files in your working directory
- Stage the files, adding snapshots of them to staging area
- Commit, takes the files as they are in the staging area and stores that snapshot permanently to your Git directory



Configuration Files

- **/etc/gitconfig** : values for every user on the system
- **~/.gitconfig file**: Specific to you.
- **.git/config** : config file for current repository
  - Specific to that single repository
  - Each level overrides values in the previous level

```
$ cat ~/.gitconfig
[user]
        name = Mark A. Yoder
        email = Mark.A.Yoder@Rose-Hulman.edu
[push]
        default = simple
[color]
        ui = true
[credential]
        helper = cache --timeout=14400
```

.git/config

```
$ cat .git/config
[core]
        repositoryformatversion = 0
        filemode = true
        bare = false
        logallrefupdates = true
[remote "origin"]
        url = https://github.com/MarkAYoder/BeagleBoard-
exercises.git
        fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
        remote = origin
        merge = refs/heads/master
```

Things to configure

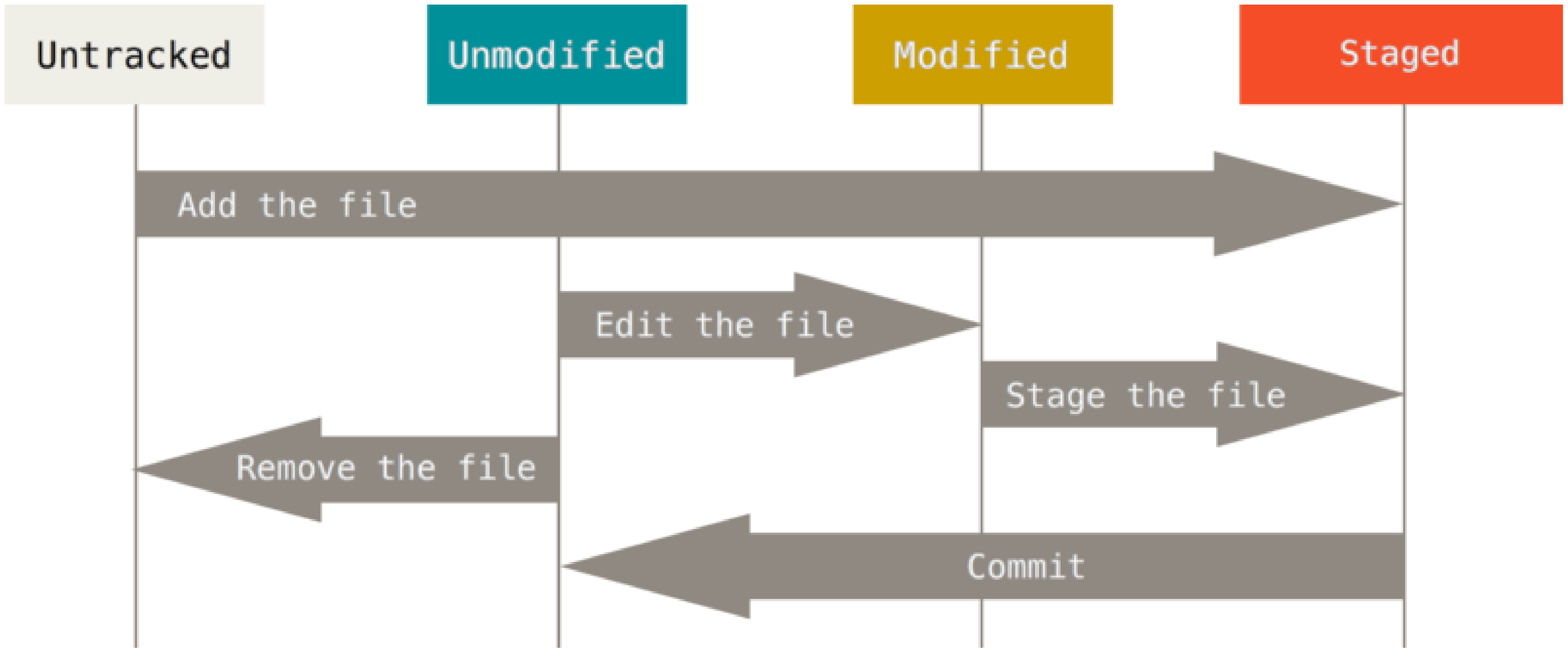
```
$ git config --global user.name "Mark A. Yoder"
$ git config --global user.email Mark.A.Yoder@Rose-Hulman.edu
$ git config --global core.editor vi

$ git config user.name
Mark A. Yoder

$ git help
$ git help config
```

The *git lab* leads you to **github** which will lead you through these commands.

See *git lab* for more details

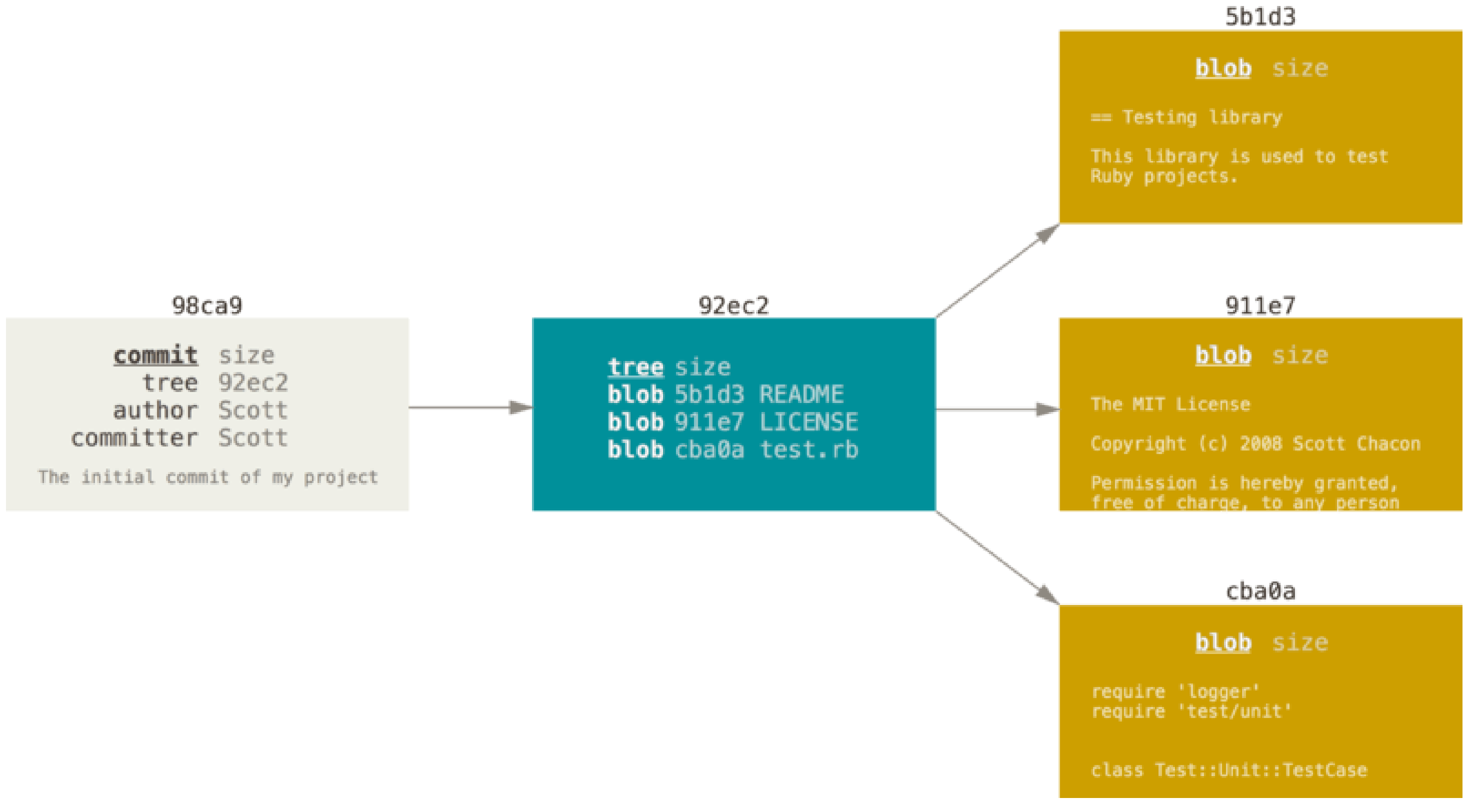
File Status Lifecycle

- See: https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository

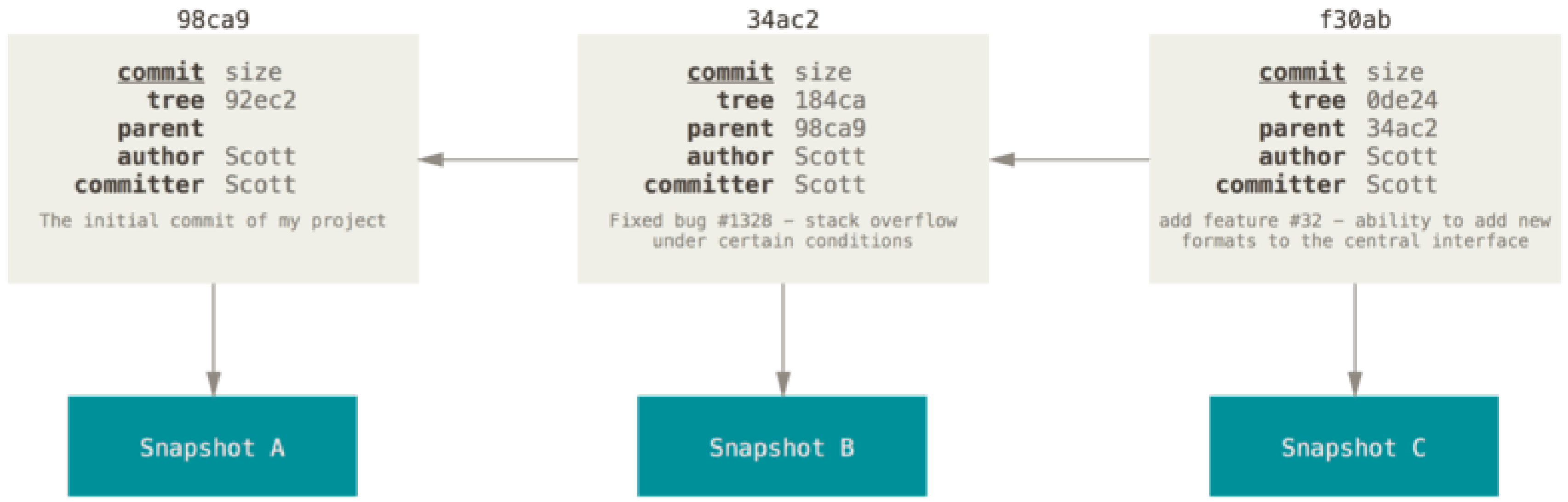


Branching

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

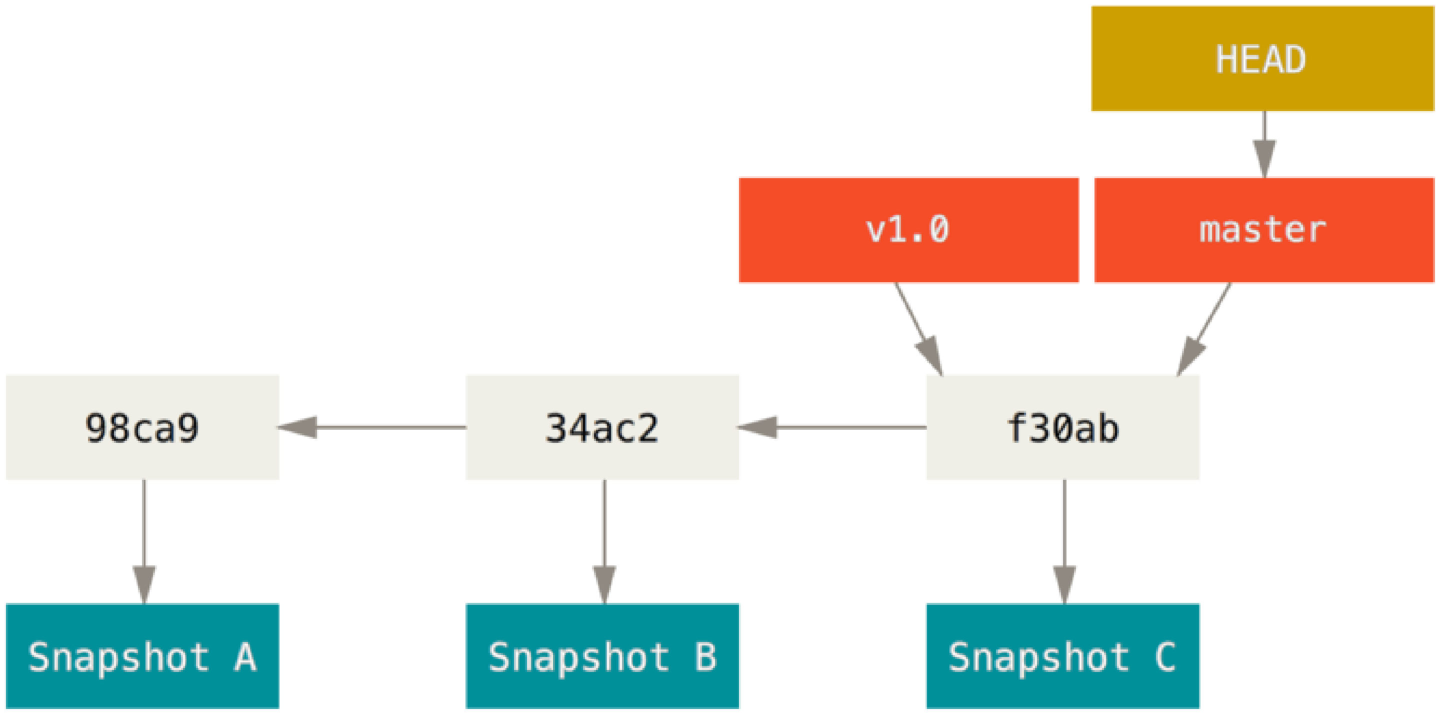


After 2 more commits



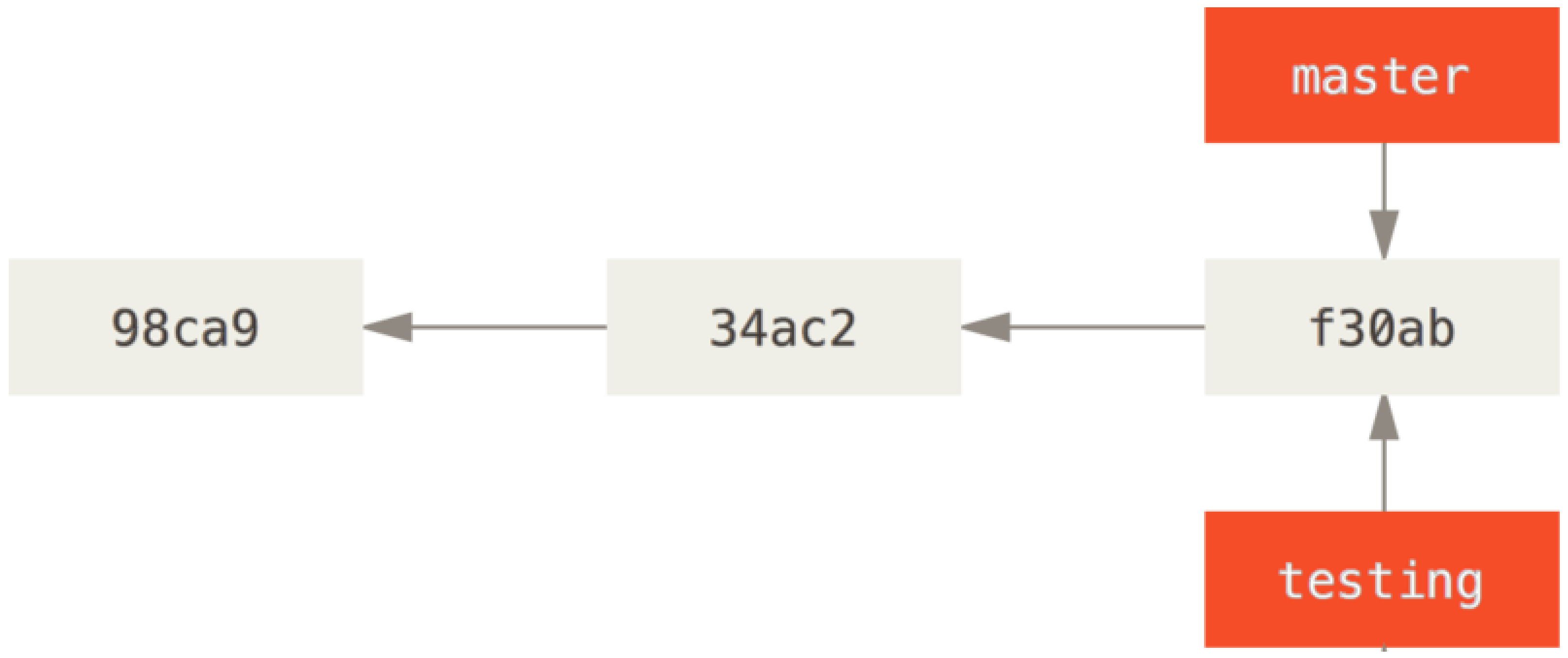
master

- A branch is a lightweight movable pointer to a commit
- Default: **`master`**



New branch

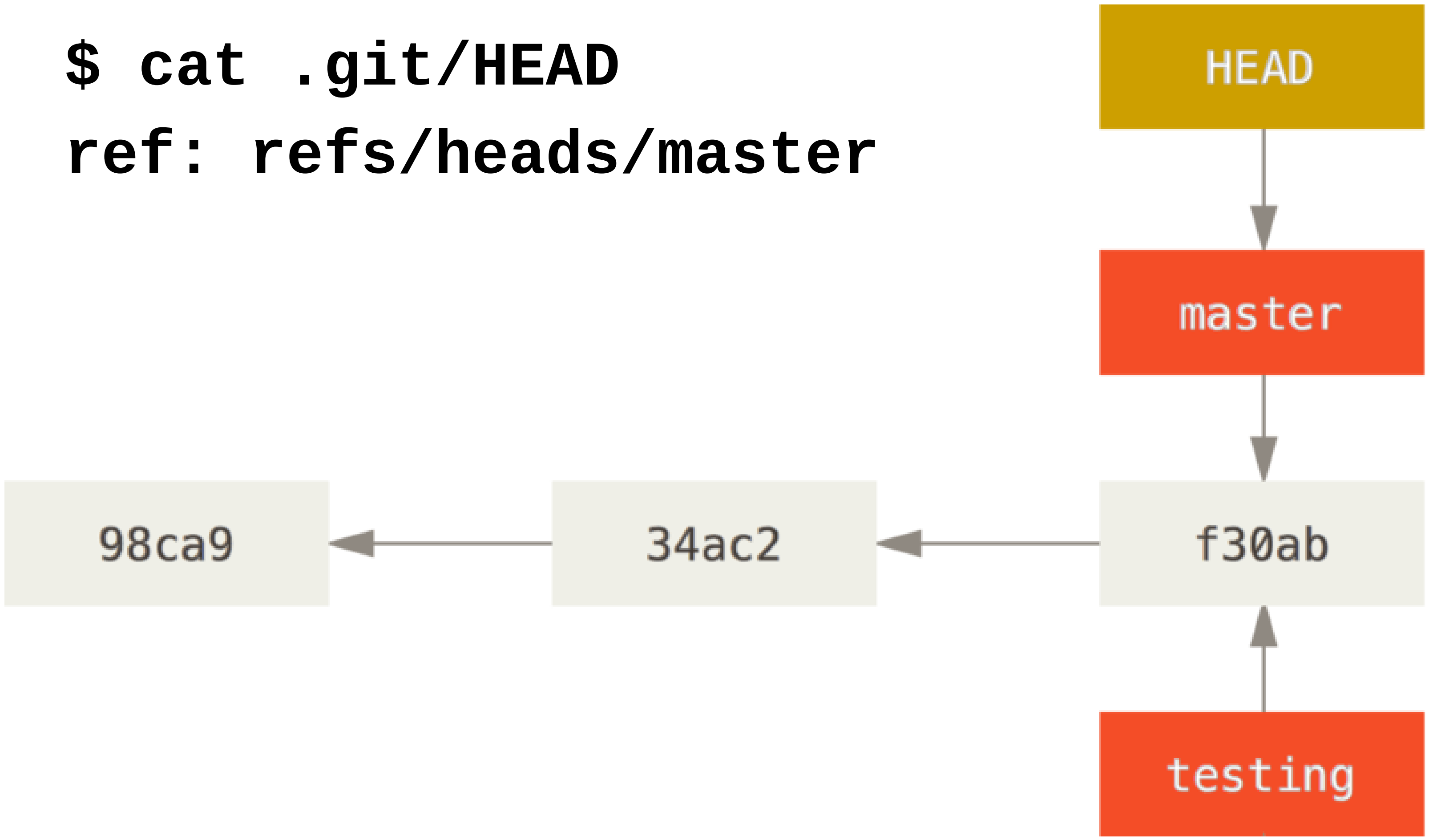
```
$ git branch testing
```



What’s the current branch?

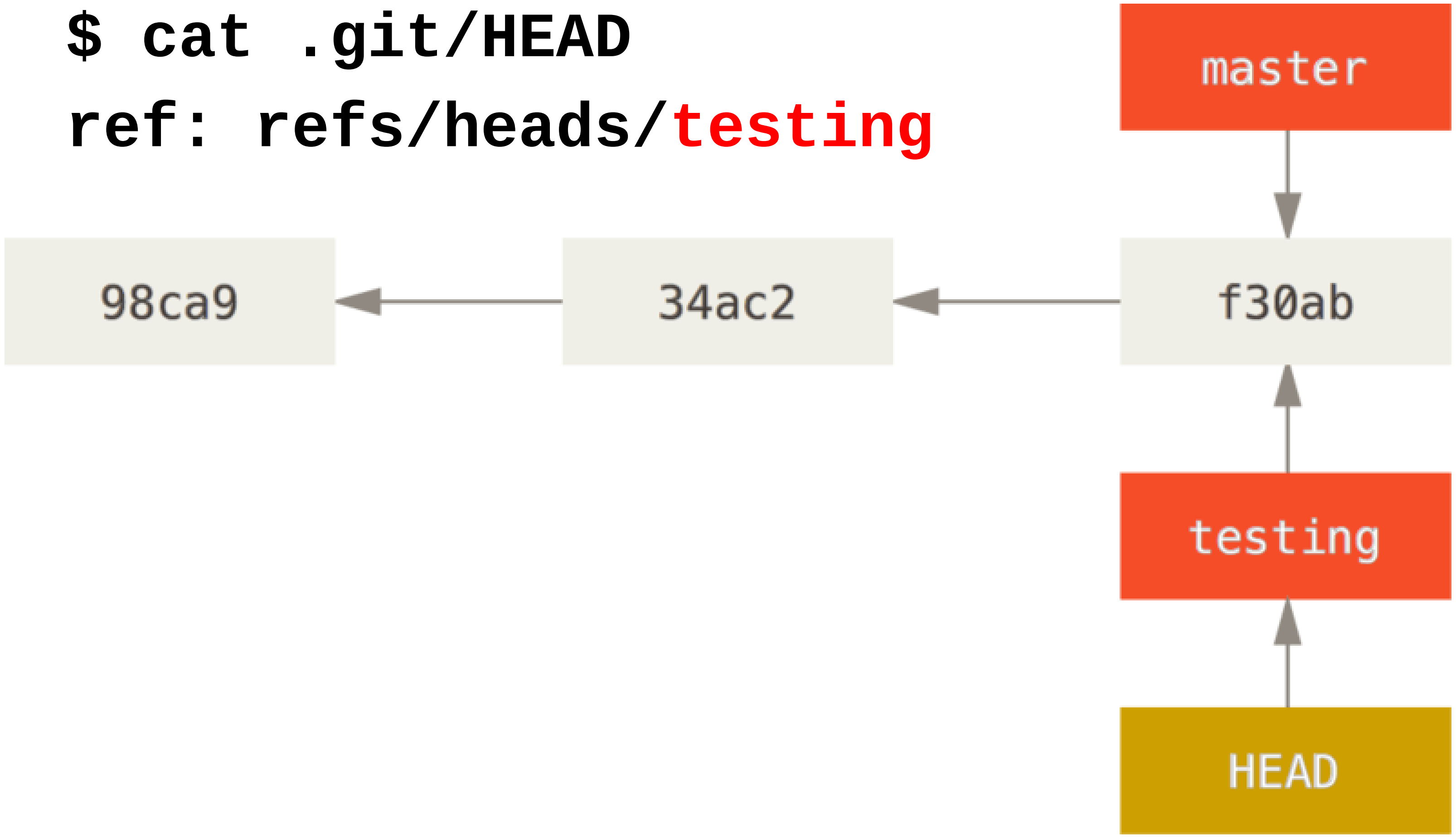
- **HEAD**

```
$ cat .git/HEAD
ref: refs/heads/master
```



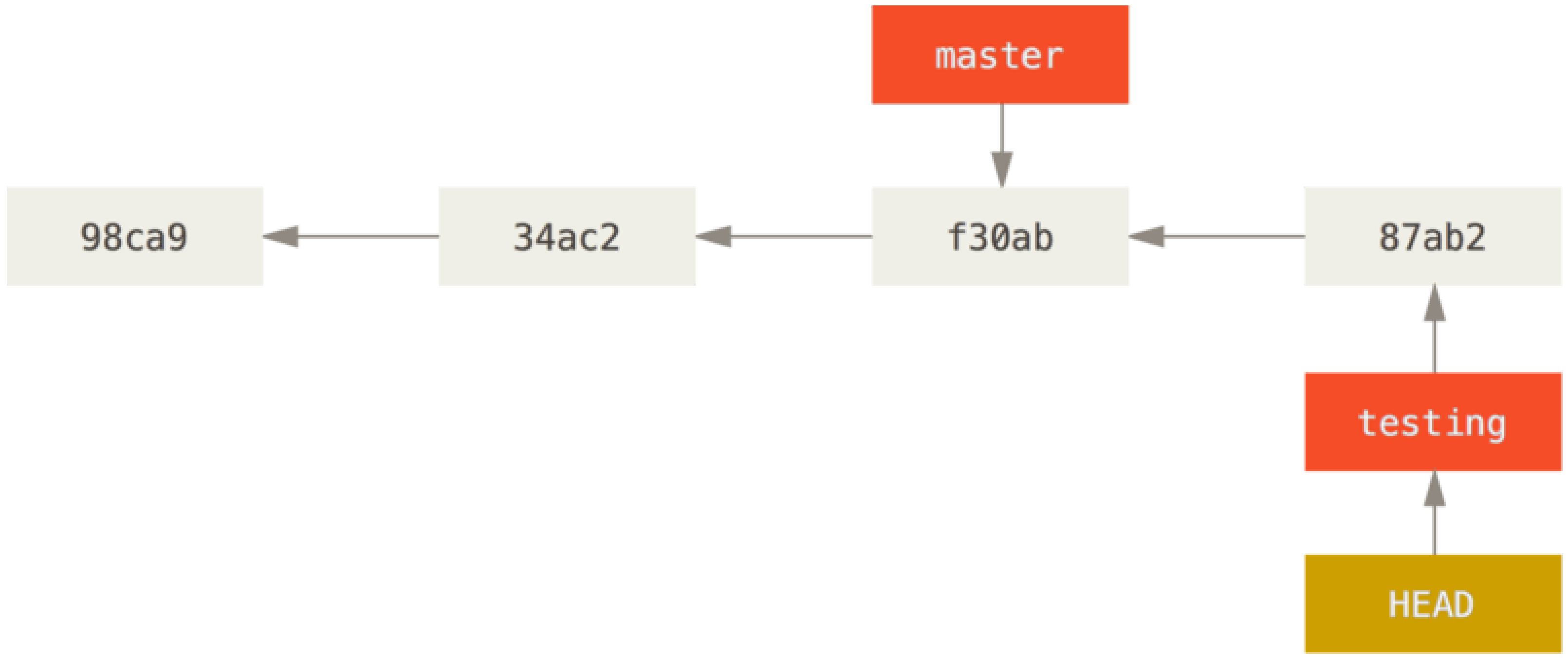
Switch branches

```
$ git checkout testing
$ cat .git/HEAD
ref: refs/heads/testing
```



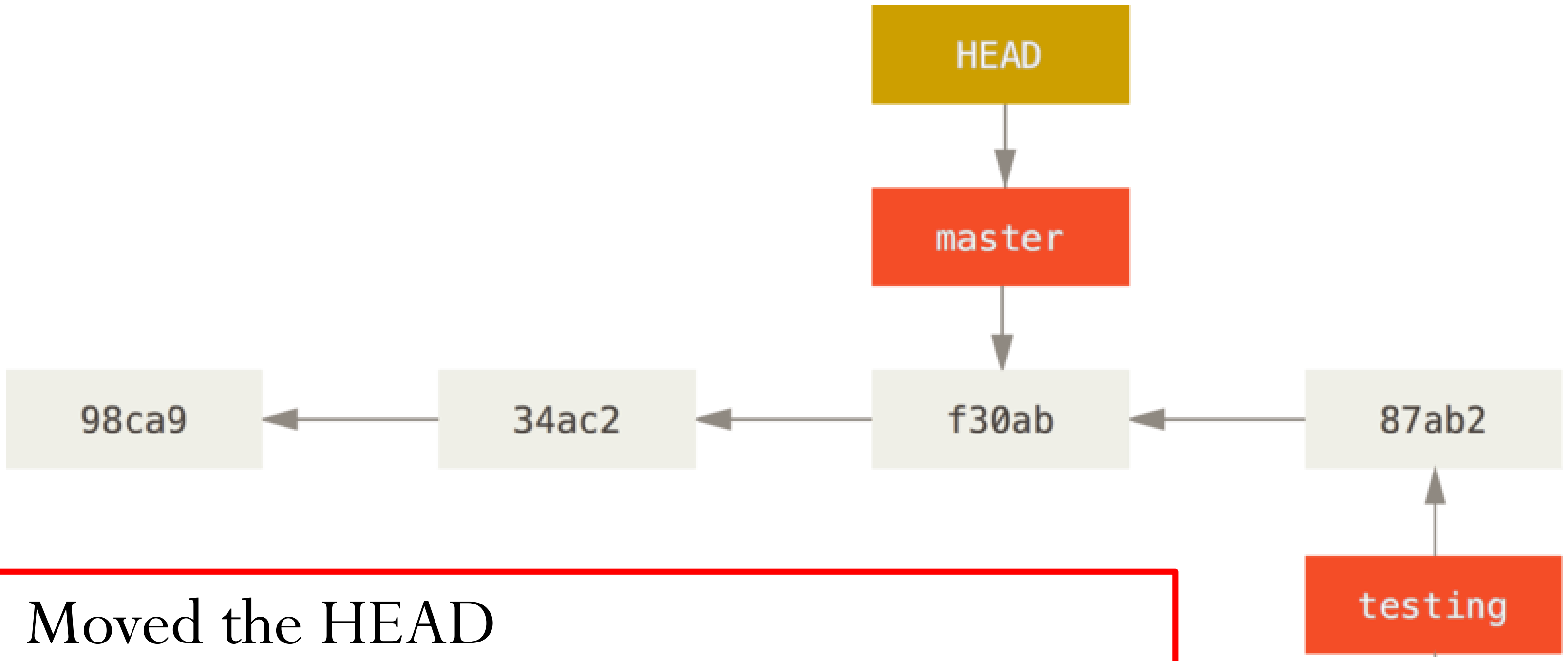
Another commit

```
$ vi test.rb
$ git add test.rb
$ git commit -m 'made a change'
```



What does this do?

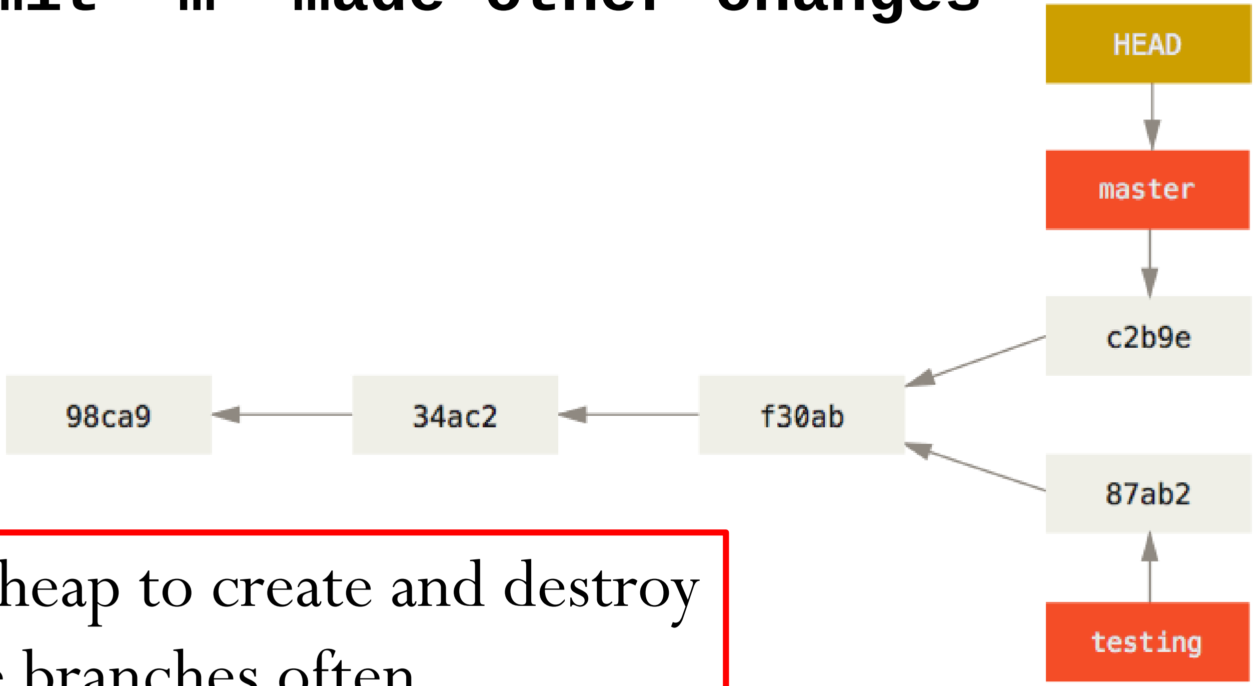
```
$ git checkout master
```



- Moved the HEAD
- Reverted the files in your working directory back to the snapshot that master points to.

More changes

```
$ vi test.rb
$ git add test.rb
$ git commit -m 'made other changes'
```



- Branches are cheap to create and destroy
- Create and use branches often

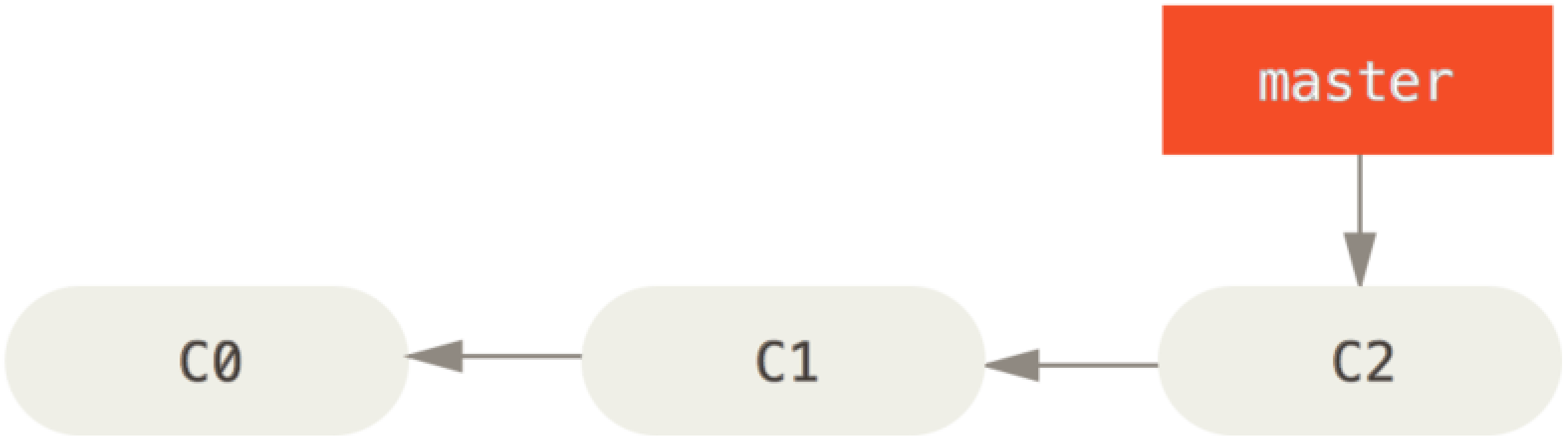
Merge

- See: https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging for a merge example

You'll do this in the *git lab*

Basic Branching and Merging

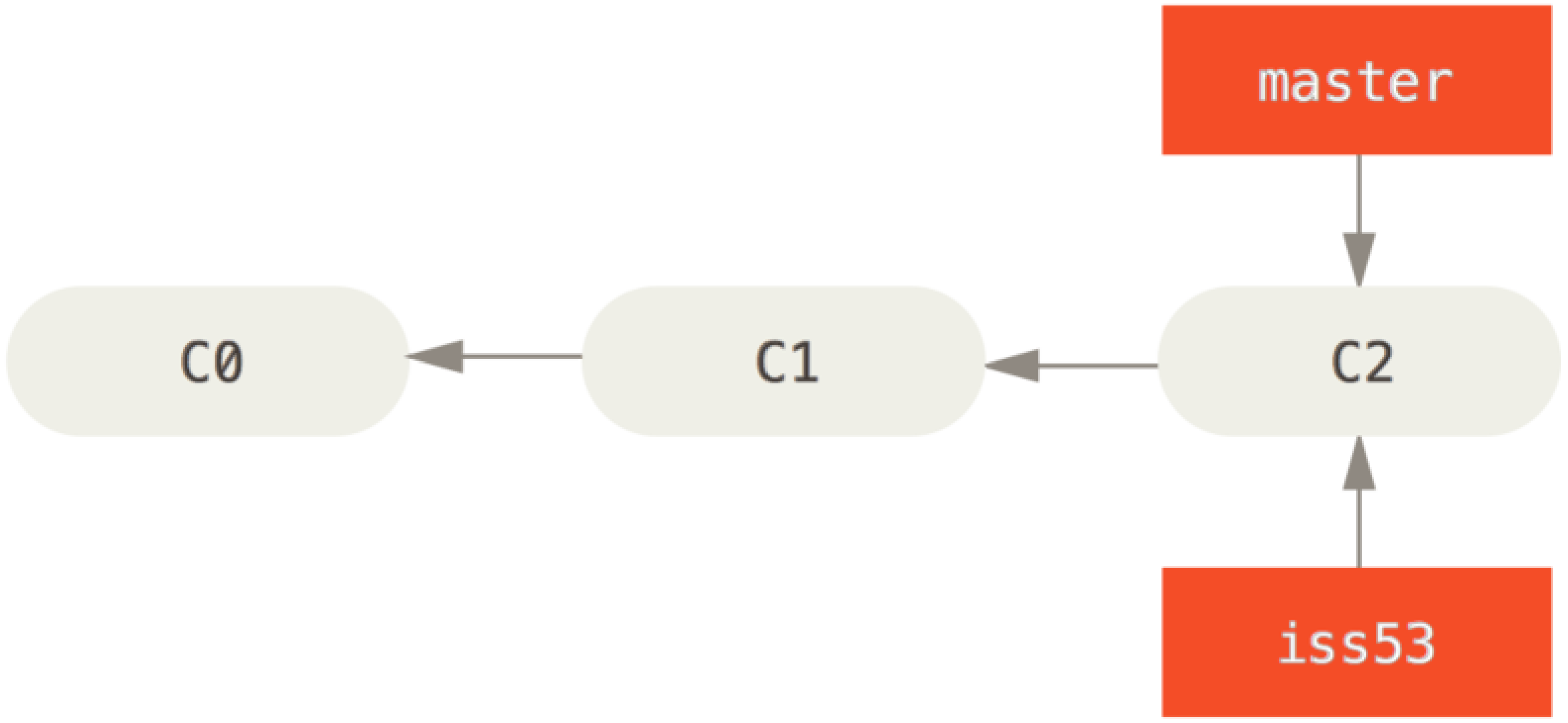
- You are working on a project and have a couple of commits



Issue #53

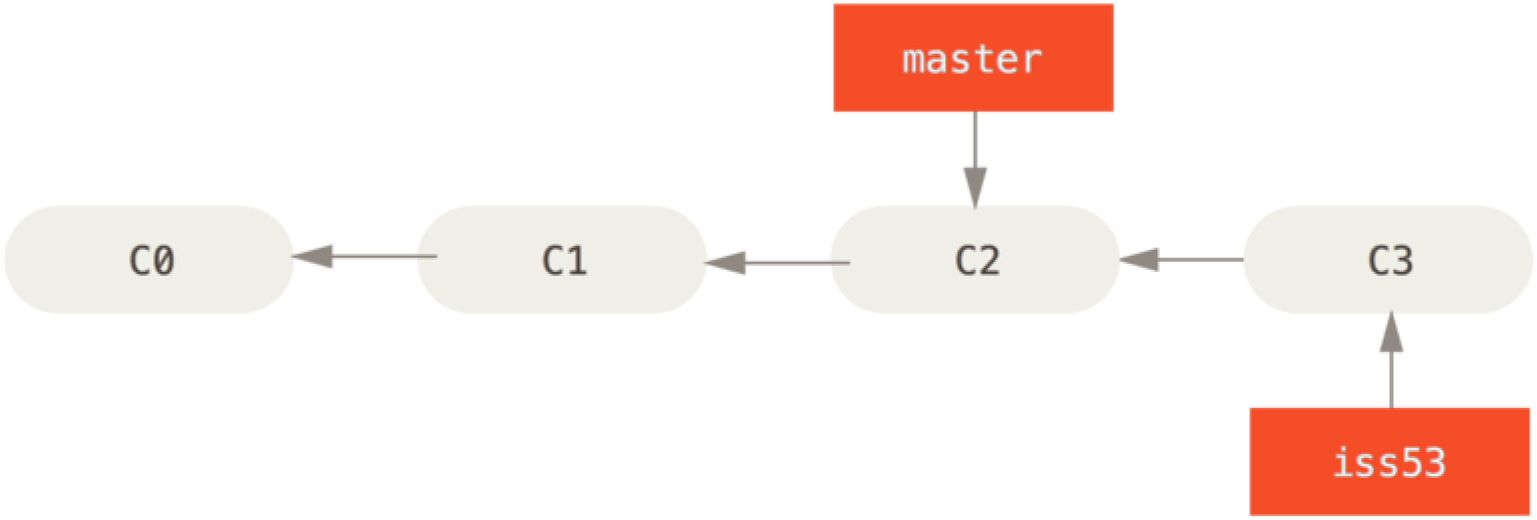
- You get a call and need to work on issue #53

```
$ git checkout -b iss53
Switched to a new branch “iss53”
```



… after some work…

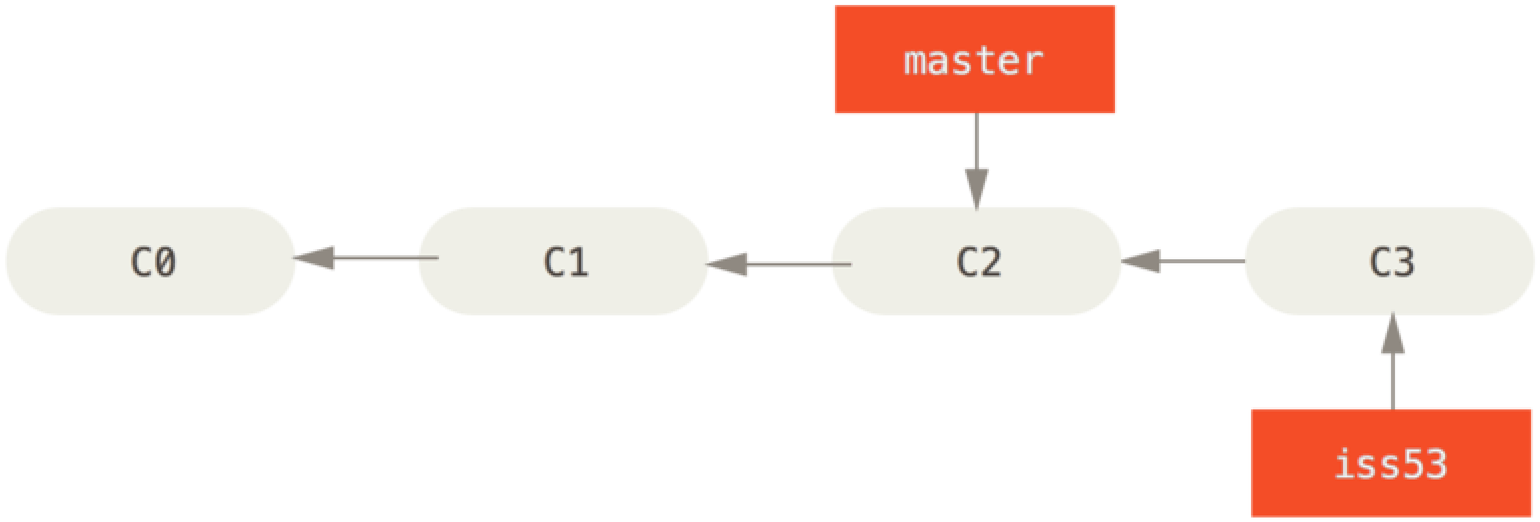
```
$ vim index.html
$ git commit -a -m 'added a new footer
[issue 53]'
```

- 

Another call…

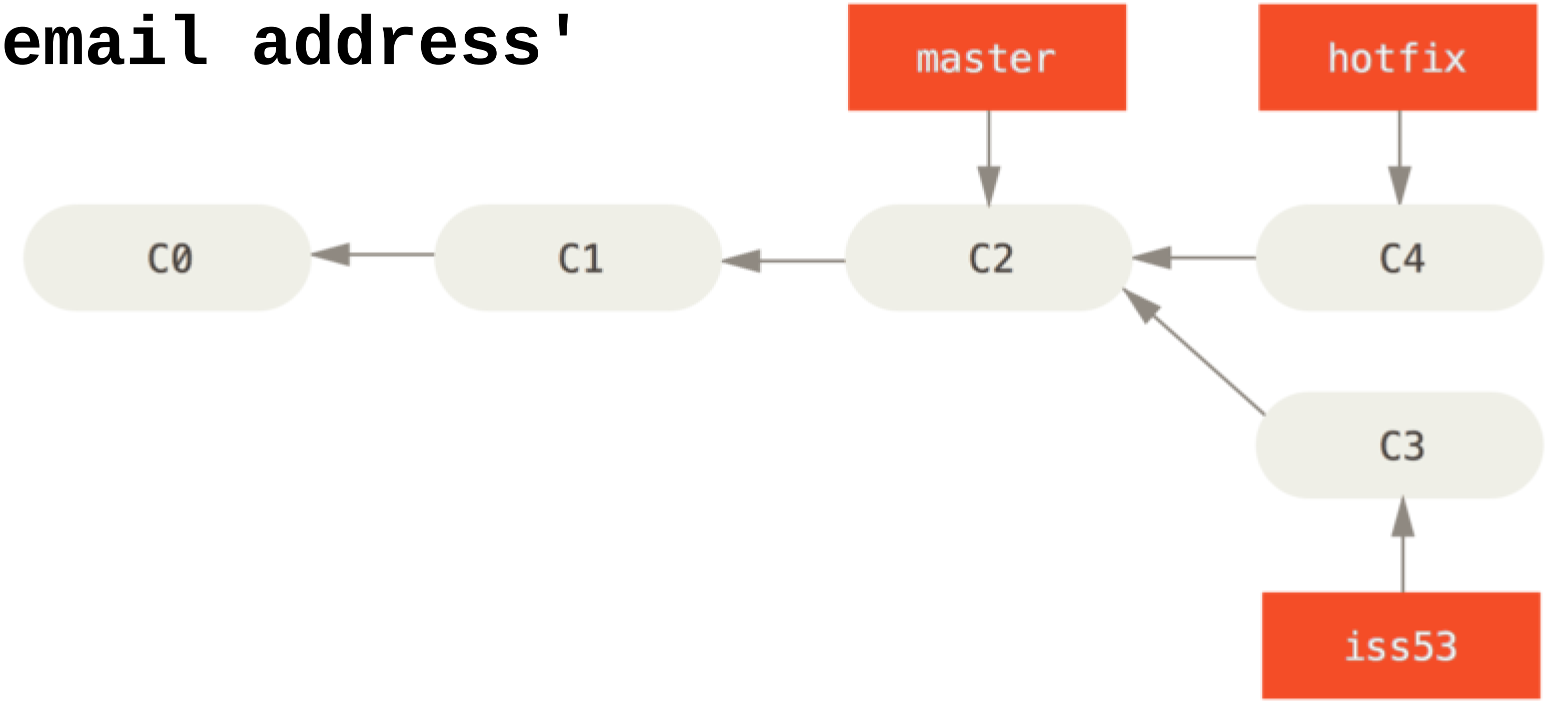
- There's a problem with the web site and you need to fix it

```
$ git checkout master
Switched to branch "master“
```



Work in the web site

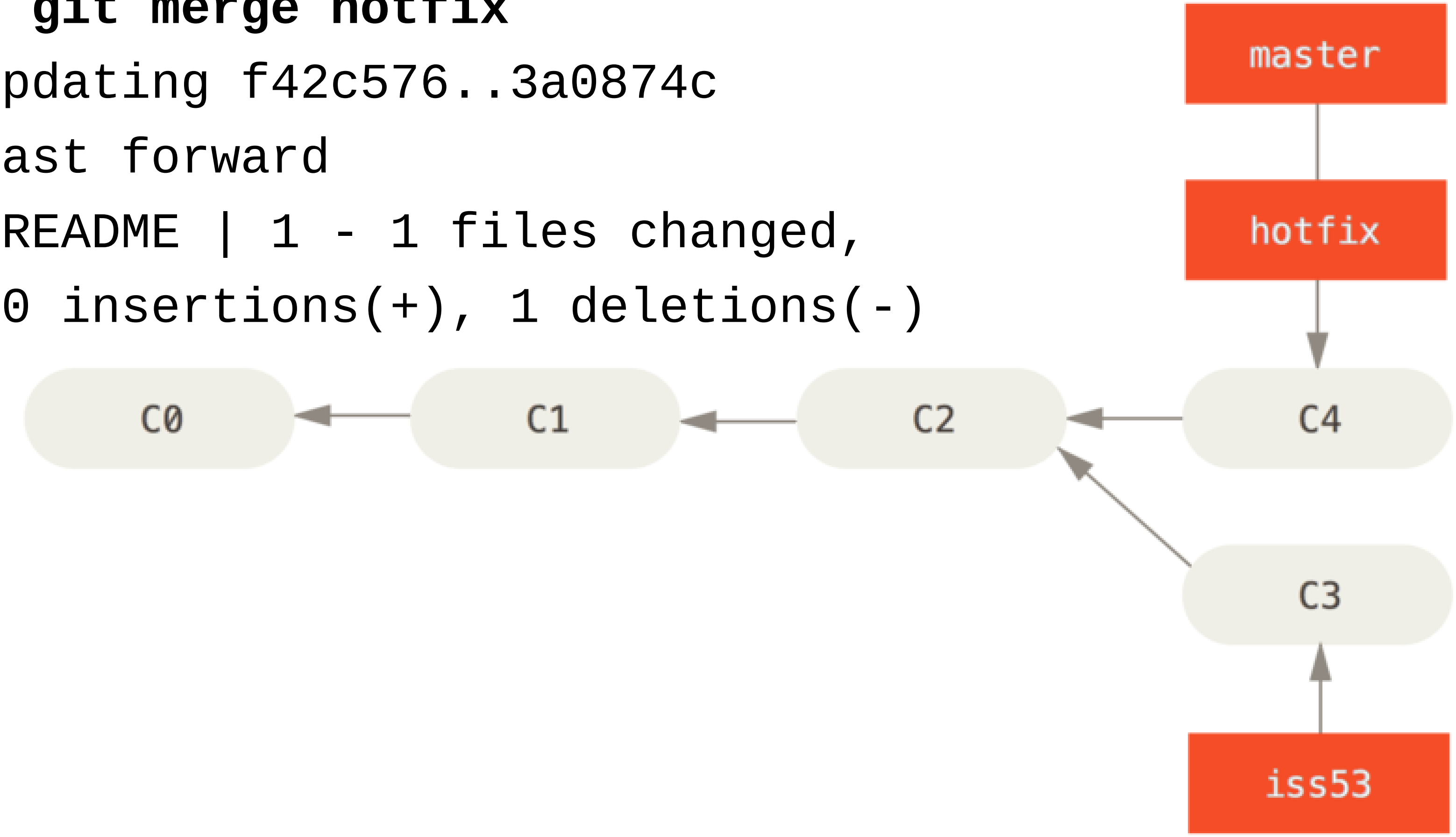
```
$ git checkout -b 'hotfix'
Switched to a new branch "hotfix"
$ vim index.html
$ git commit -a -m 'fixed the broken
email address'
```



Run Tests and Merge

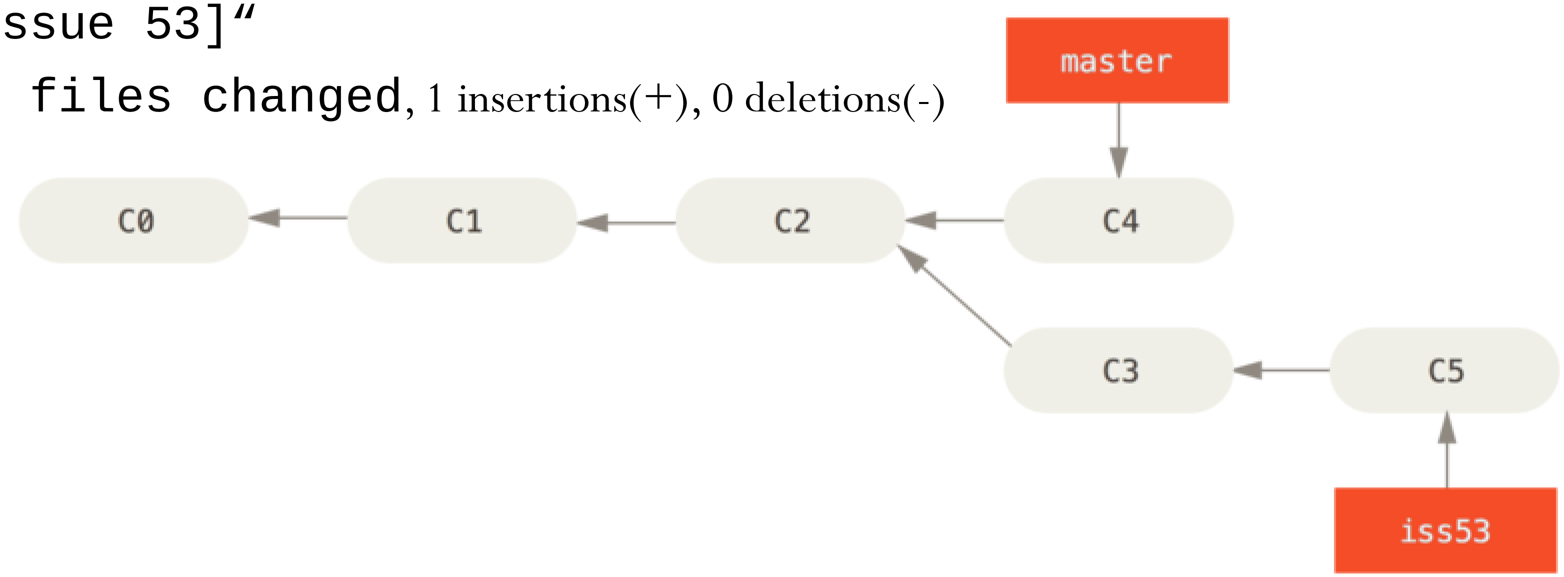
After testing hotfix, merge it back to master

```
$ git checkout master
$ git merge hotfix
Updating f42c576..3a0874c
Fast forward
 README | 1 - 1 files changed,
 0 insertions(+), 1 deletions(-)
```



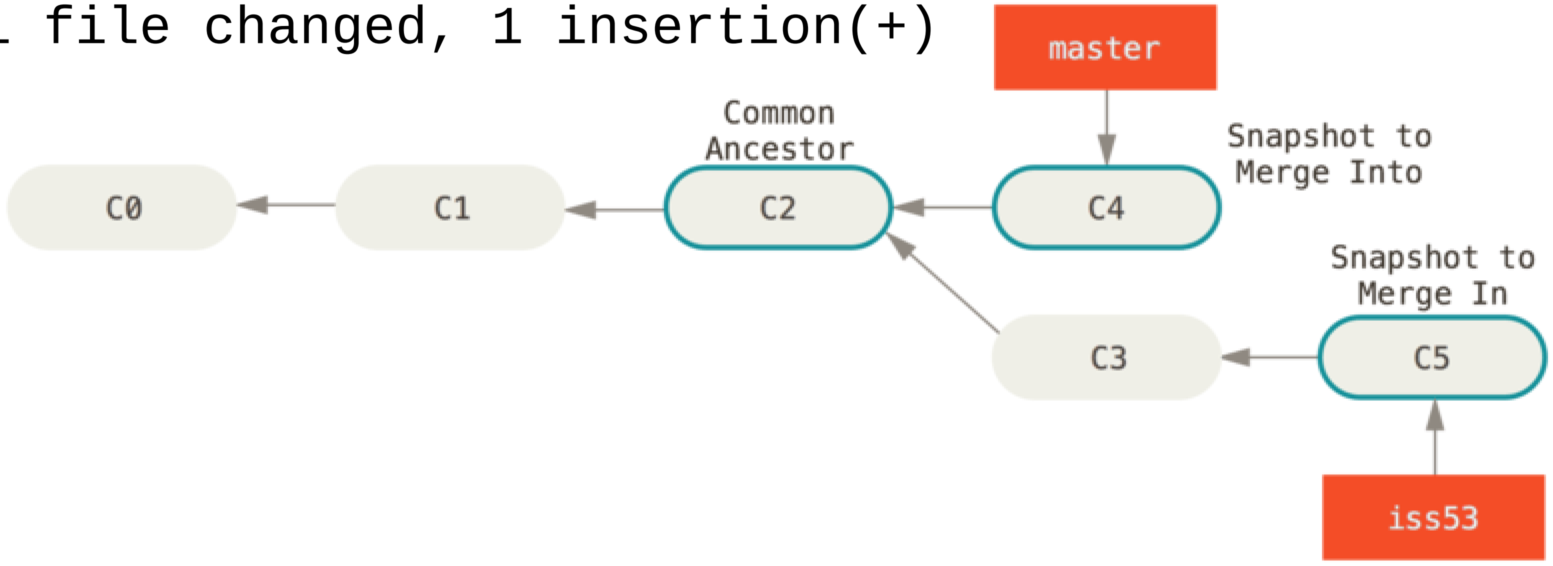
Back to issue #53

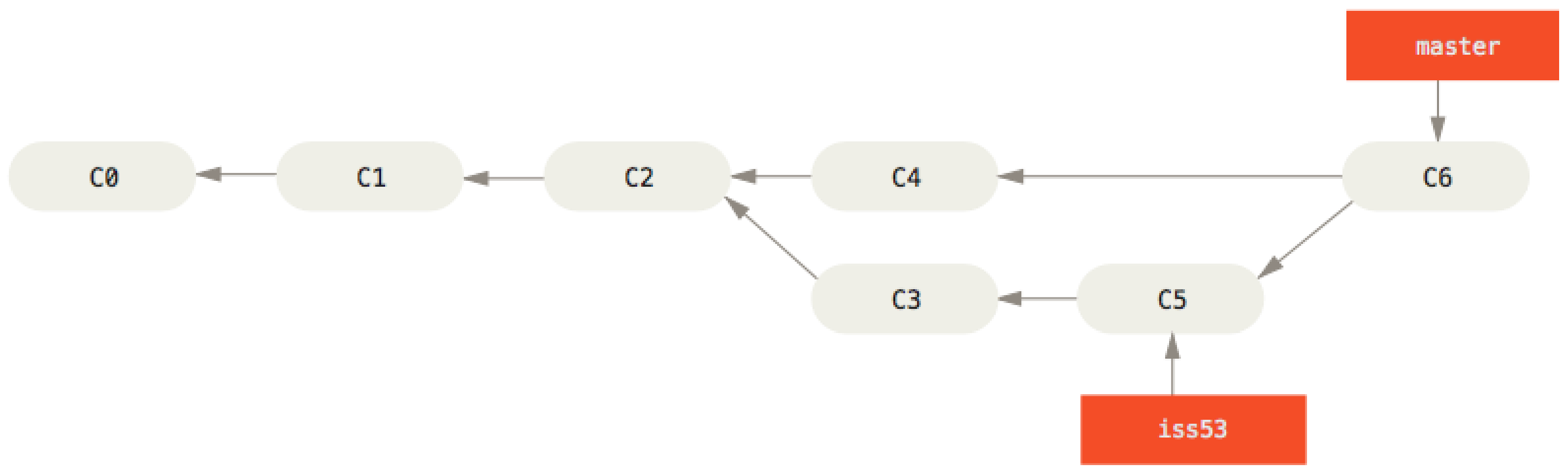
```
$ git branch -d hotfix
Deleted branch hotfix (3a0874c).
$ git checkout iss53
Switched to branch "iss53"
$ vim index.html
$ git commit -a -m 'finished the new footer [issue 53]‘
[iss53]: created ad82d7a: "finished the new footer
[issue 53]“
 1 files changed, 1 insertions(+), 0 deletions(-)
```



Basic Merging

```
$ git checkout master
$ git merge iss53
Auto-merging README
Merge made by the 'recursive' strategy.
README | 1 +
1 file changed, 1 insertion(+)
```




master
C0
C1
C2
C4
C6
C3
C5
iss53

Basic Merge Conflicts

- Hands on

Merging master and iss53

- See https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging for example of merging **`master`** and **`iss53`**.