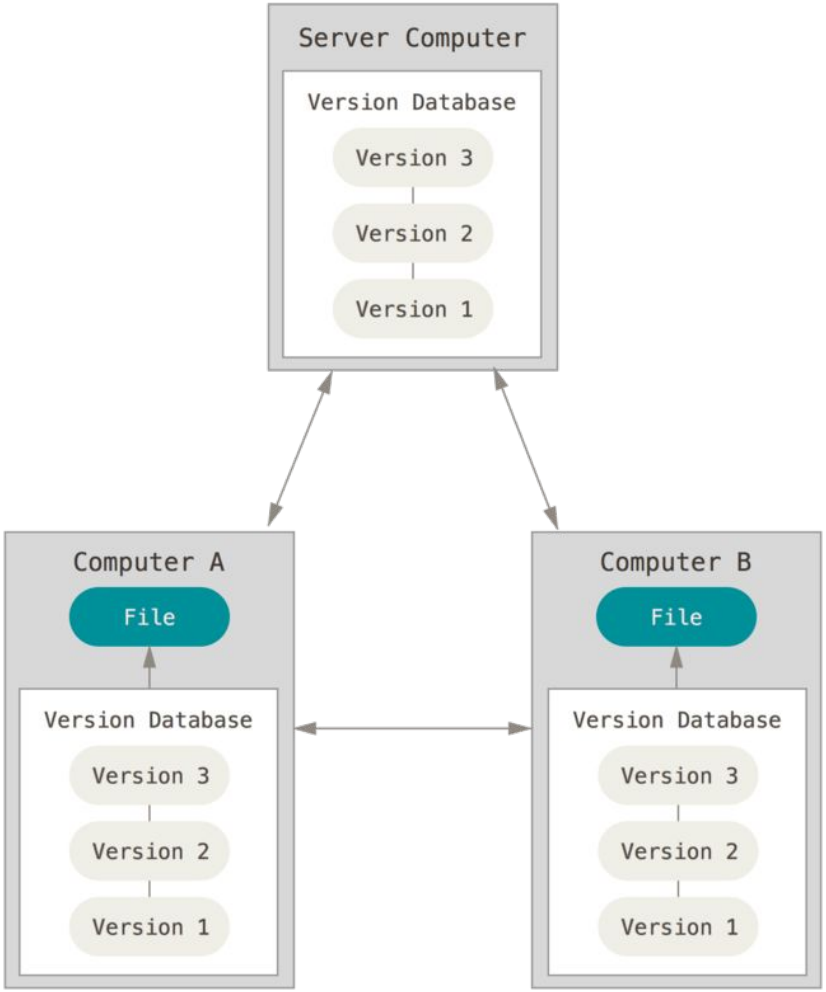
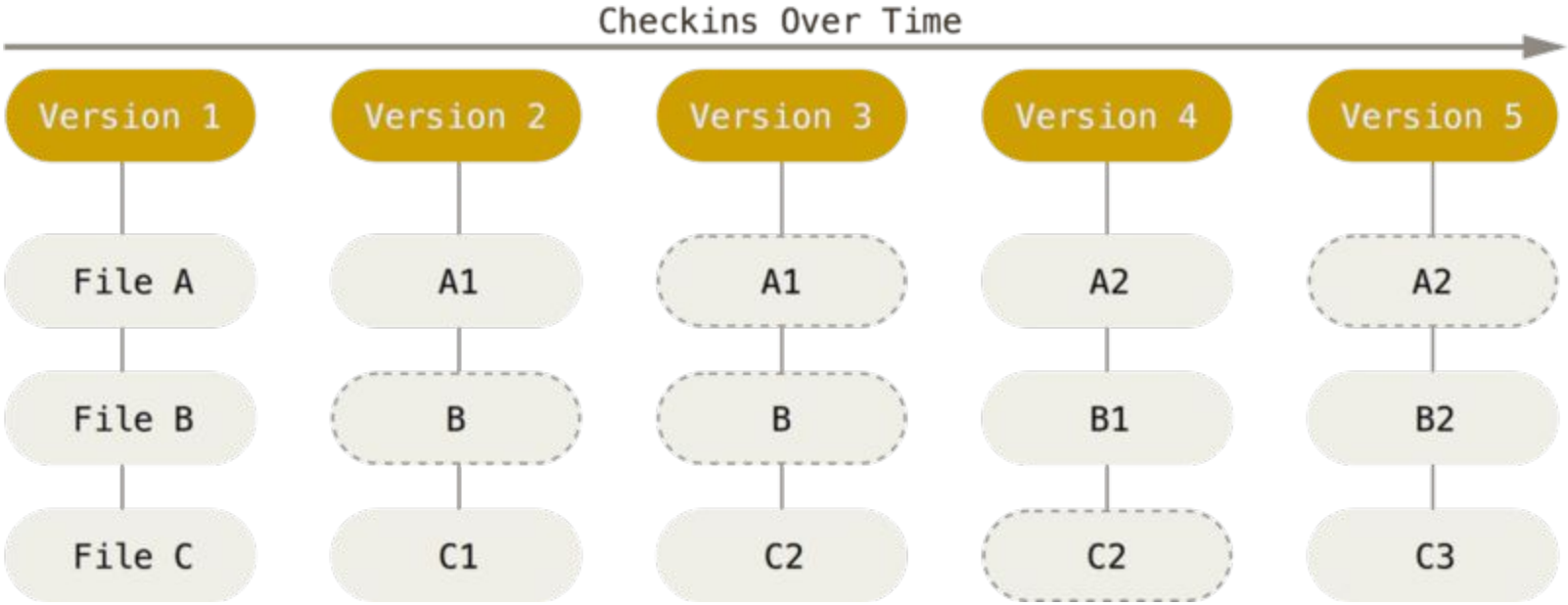
Intro to Git

A modern Version Control System

What is a Version Control System?

- A version control system, or VCS, allows you to track changes to a file across different versions.
- More specifically, it tracks which parts of a file are changed, by who, and when.
- With this data, we can compare the state of a codebase at two different points in time.
- Additionally, we can “go back in time” to a previous version of the code.


Server Computer
Version Database
Version 3
Version 2
Version 1
Computer A
File
Version Database
Version 3
Version 2
Version 1
Computer B
File
Version Database
Version 3
Version 2
Version 1

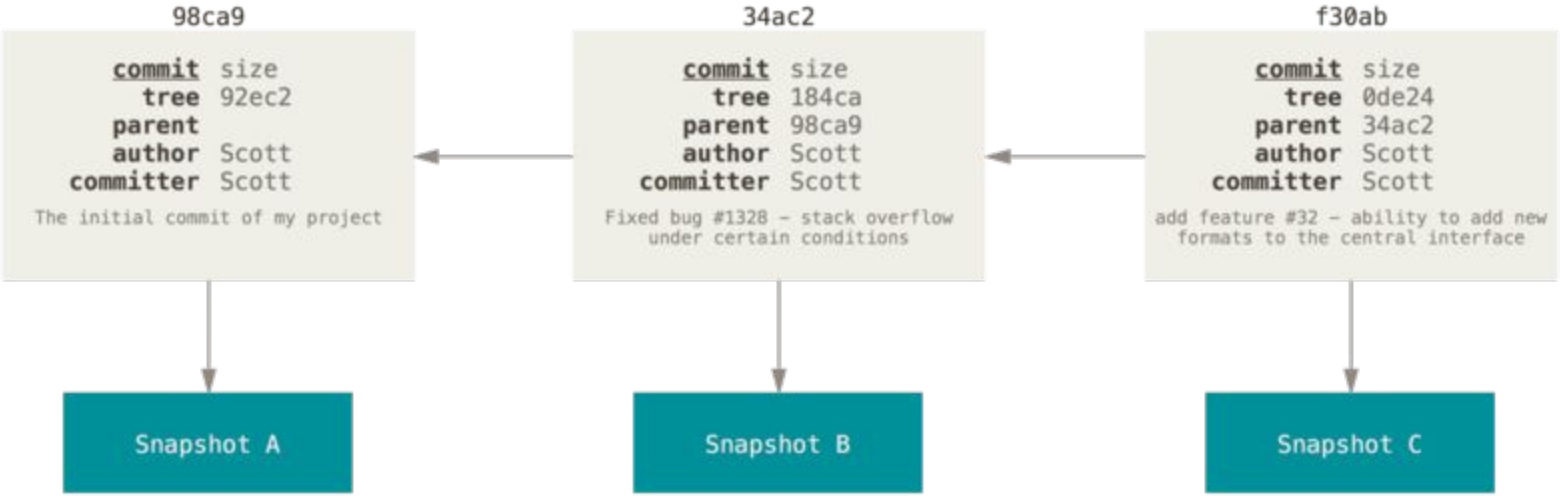

Checkins Over Time
Version 1
Version 2
Version 3
Version 4
Version 5
File A
File B
File C
A1
B
C1
A1
B
C2
A2
B1
C2
A2
B2
C3

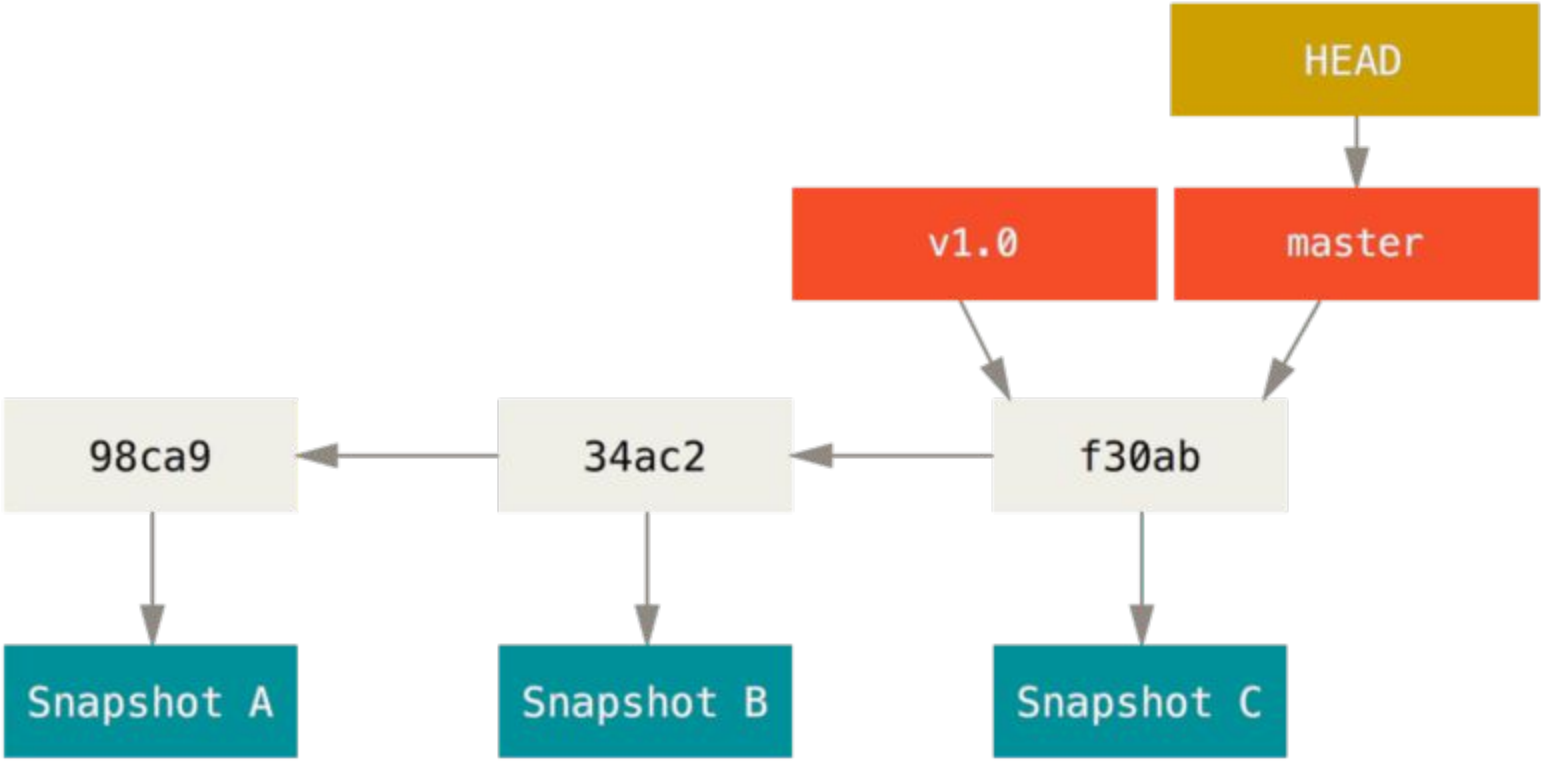
What makes Git better?

- Git is an advanced, modern VCS with many useful features.
- Additionally, Git has seen a surge in popularity in recent years.
    - Because of this, it is a valuable skill to have as there is a relatively good chance it will be used in any collaborative programming projects, even in industry.
    - Additionally, they widespread usage has led to many companies hosting free Git servers (Github) and creating great tools to help facilitate usage of Git (Github's desktop client).

Git Basics: Commit

- A commit in Git is a snapshot of files and some accompanying metadata.
- Typically, whenever a group of related changes are completed, they are “committed.” This means that Git creates a snapshot of all the files that have changed since the previous commit, and stores a description of the changes,the date and time they were committed, and who committed them.
- All commits* have (at least) one parent, and typically have one or more children.

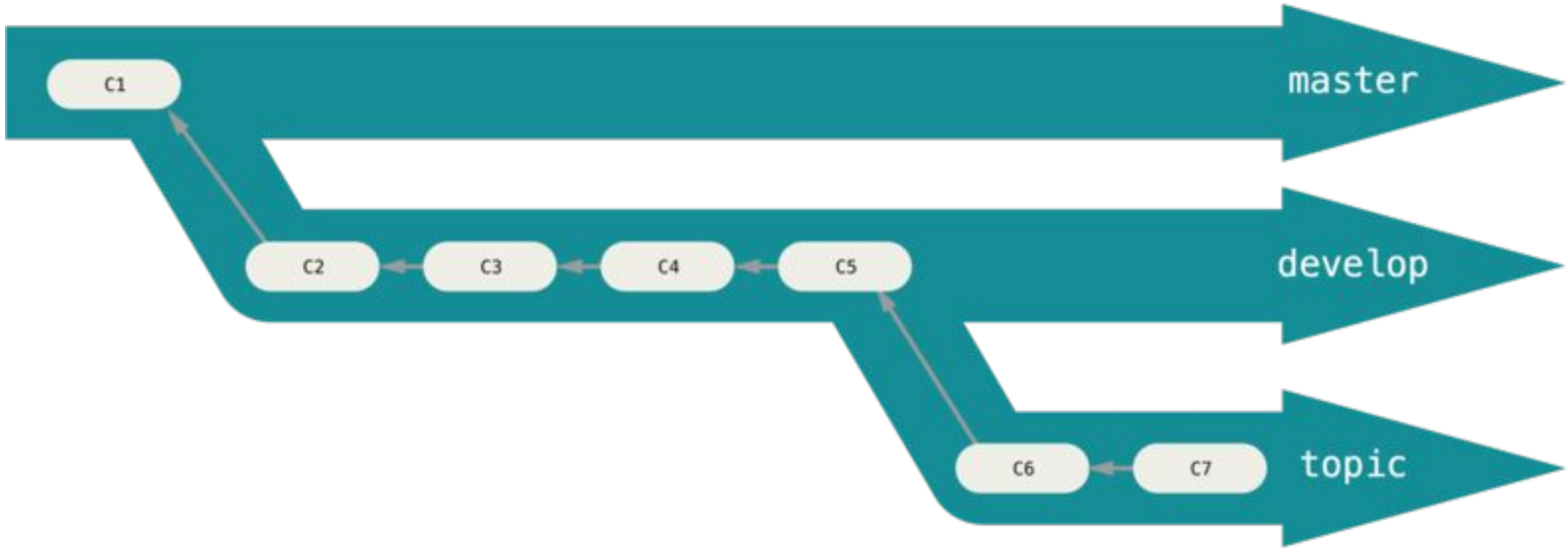

98ca9
commit size
tree 92ec2
parent
author Scott
committer Scott
The initial commit of my project
34ac2
commit size
tree 184ca
parent 98ca9
author Scott
committer Scott
Fixed bug #1328 - stack overflow under certain conditions
f30ab
commit size
tree 0de24
parent 34ac2
author Scott
committer Scott
add feature #32 - ability to add new formats to the central interface
Snapshot A
Snapshot B
Snapshot C


HEAD
v1.0
master
98ca9
34ac2
f30ab
Snapshot A
Snapshot B
Snapshot C

Git Basics: Branches

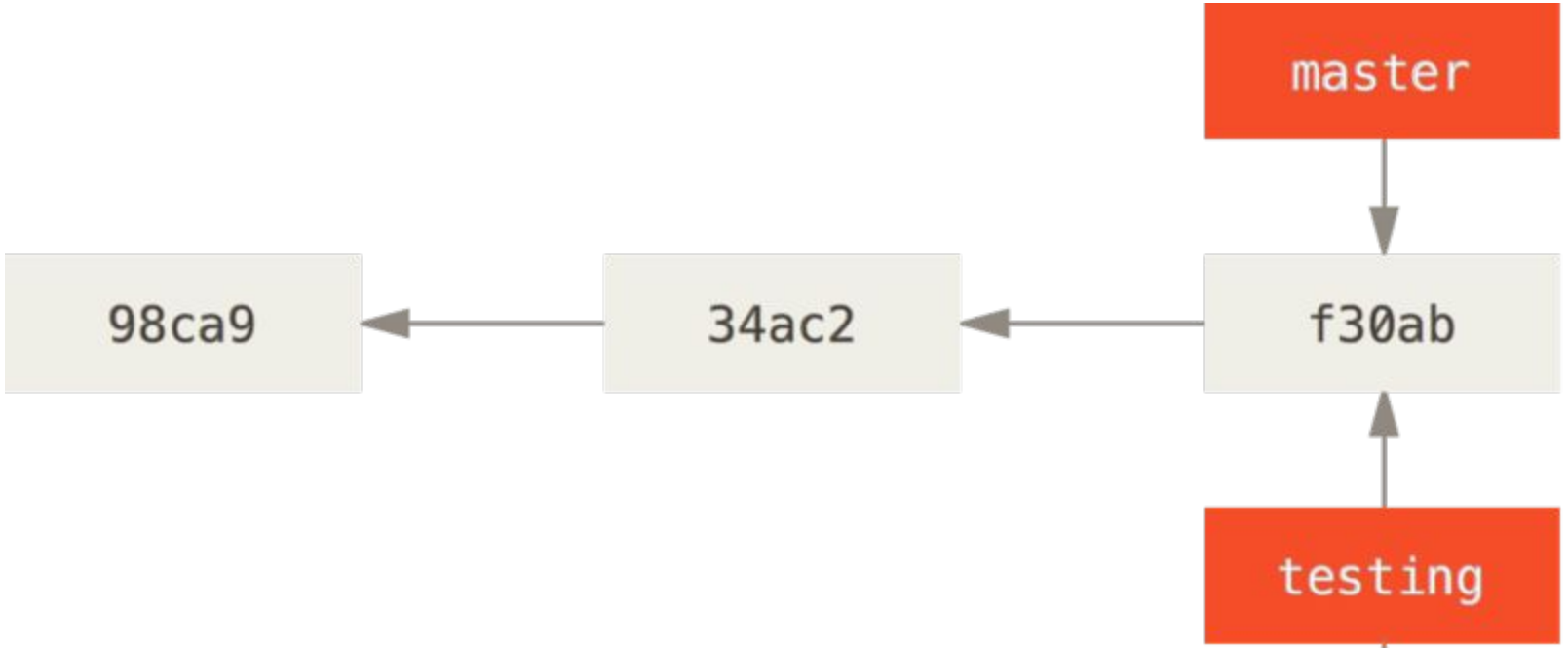
- A branch is a somewhat complex idea, but really it's just a pointer to a specific commit.
- The complexity comes when you have multiple branches.

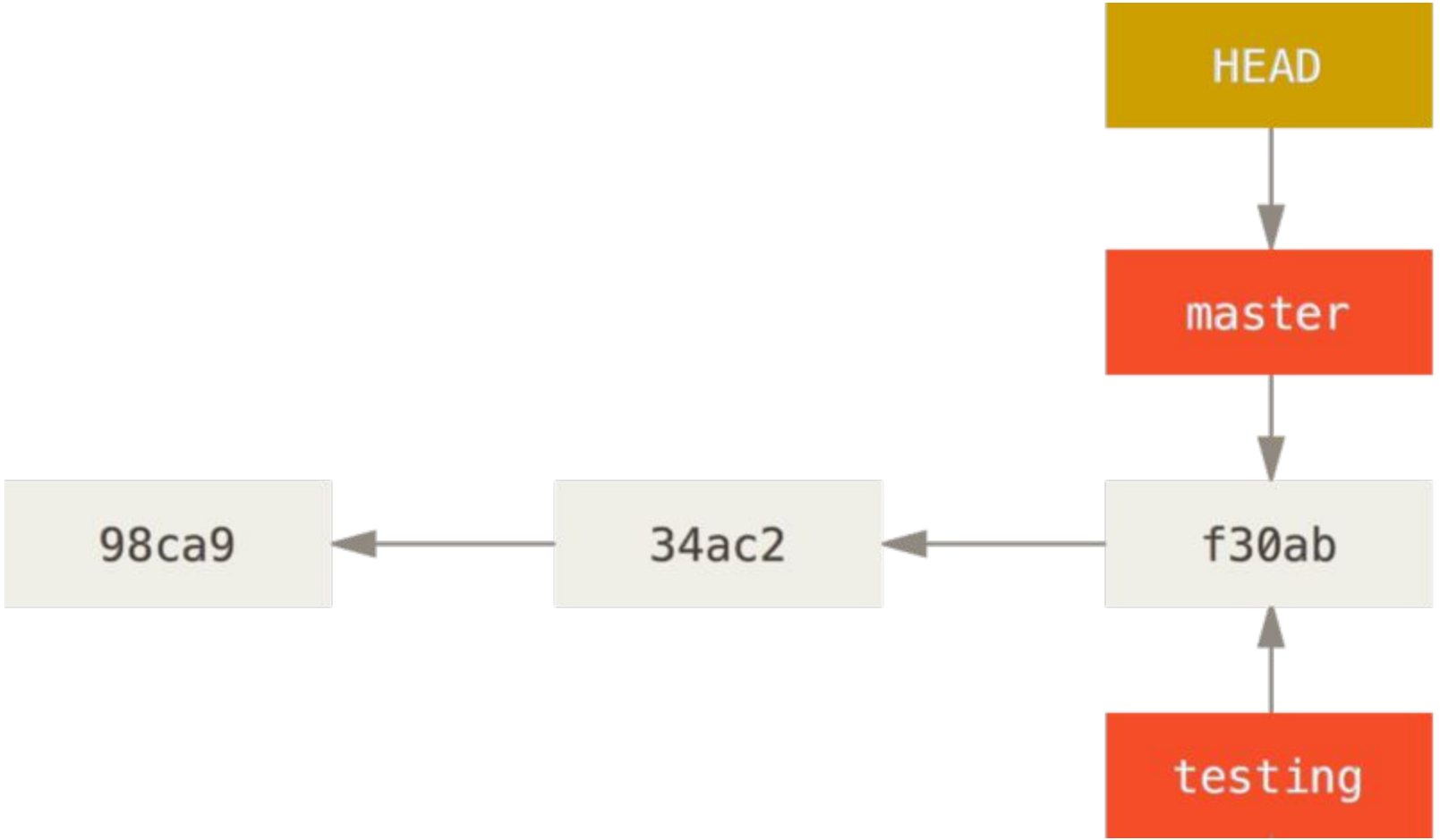
A simple branching example.

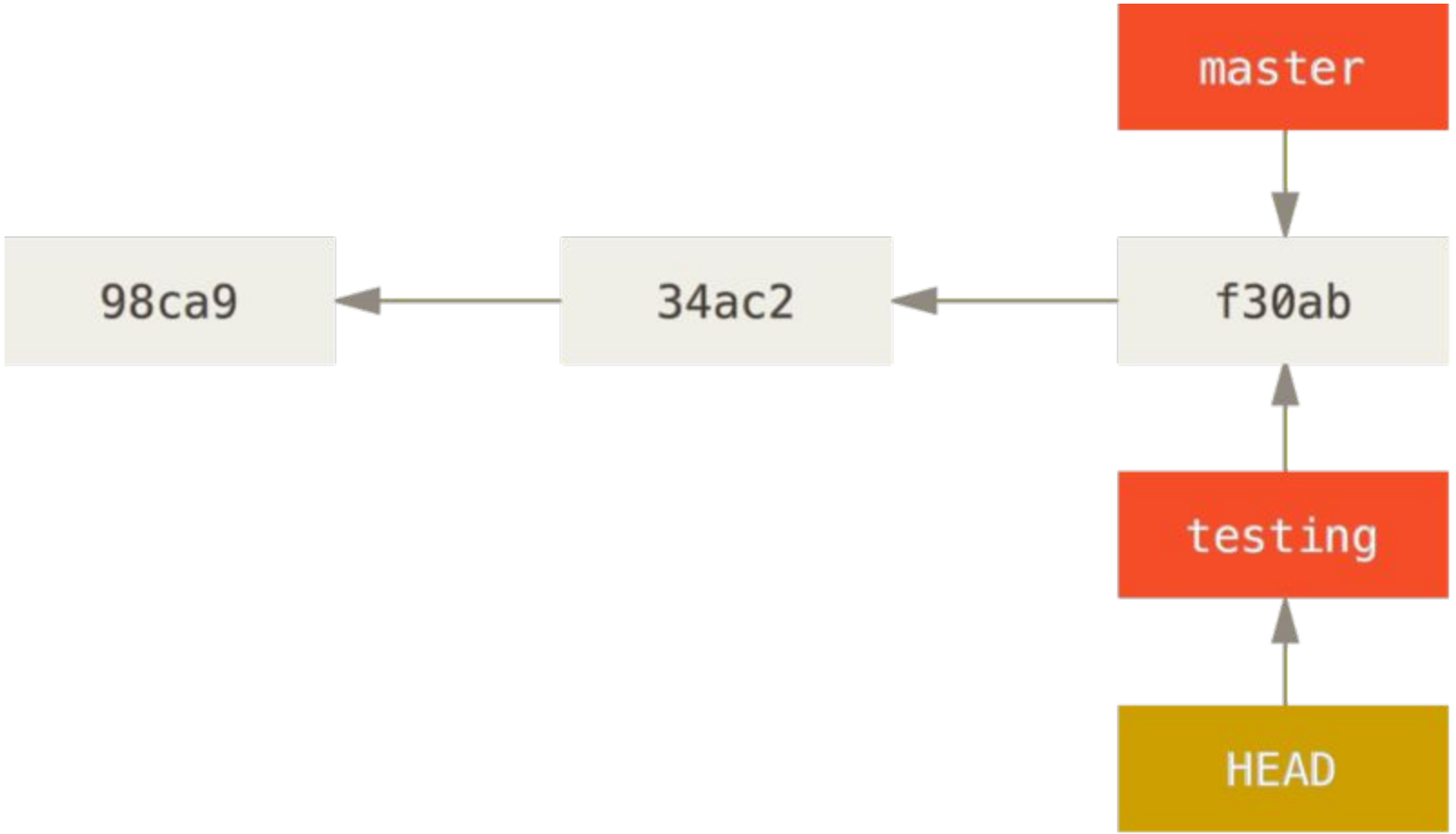

C1
C2
C3
C4
C5
C6
C7
master
develop
topic

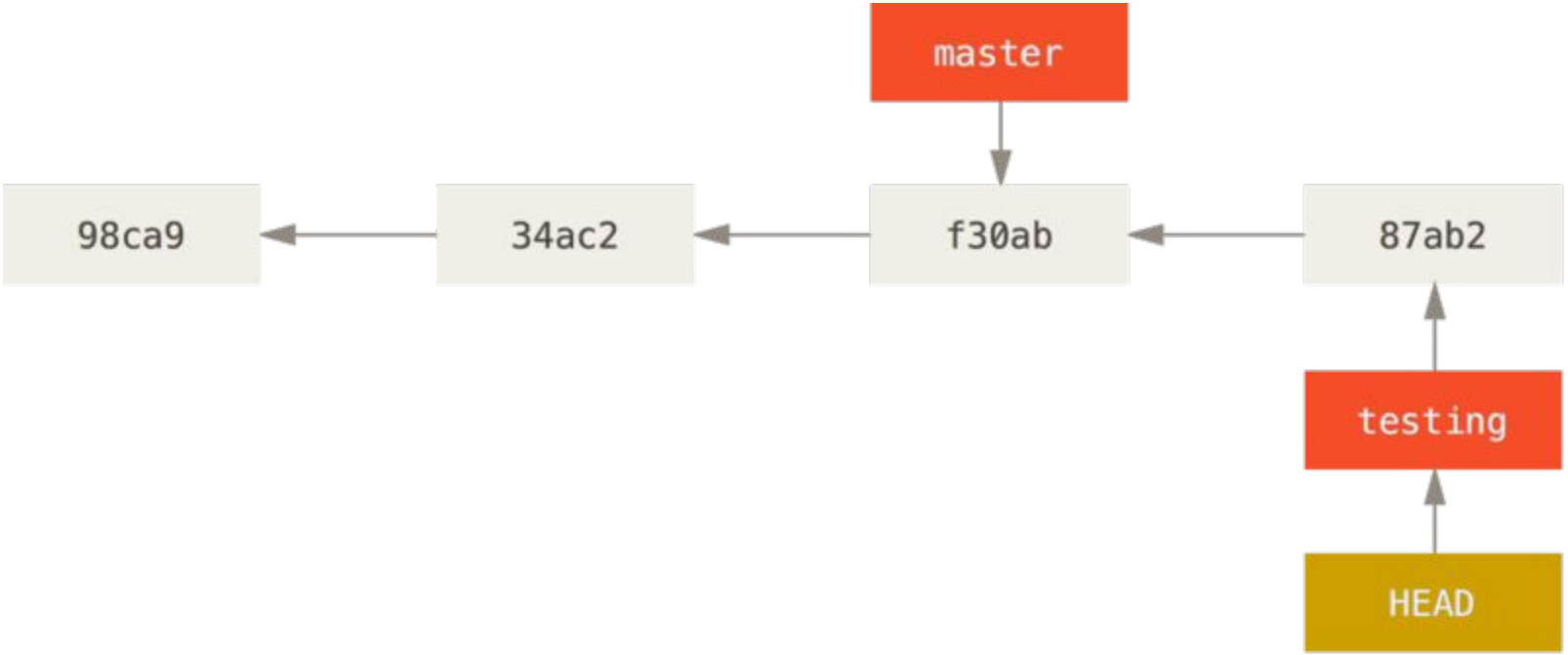

master
develop
topic
C1
C2
C3
C4
C5
C6
C7

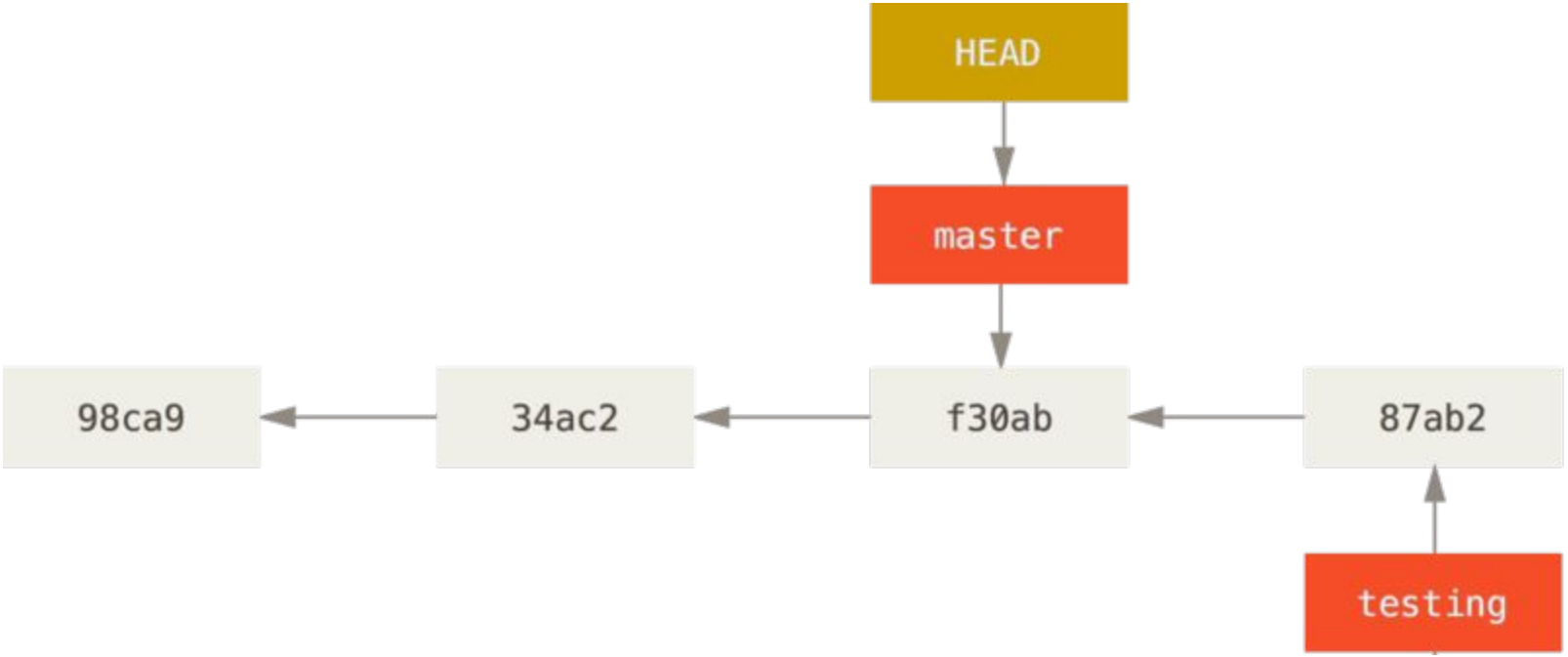
A more complex (and more typical) example.

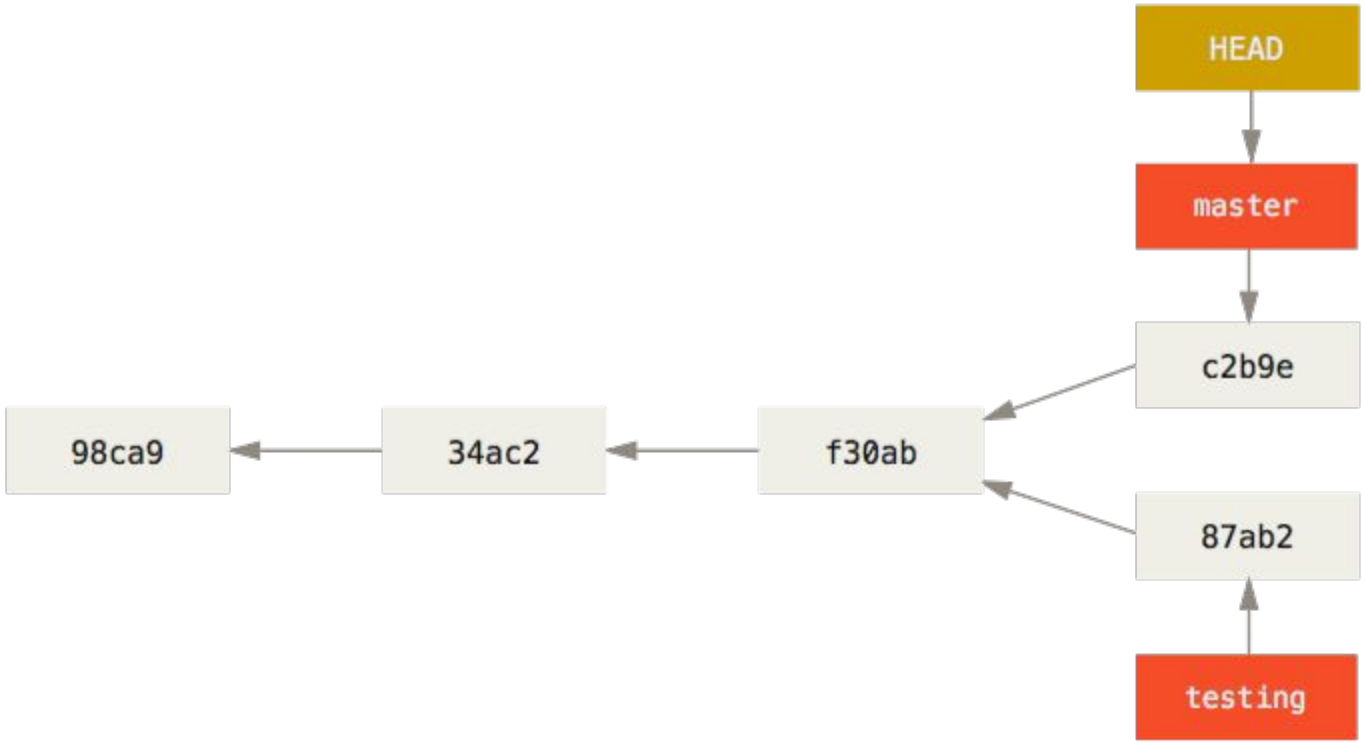

master
98ca9
34ac2
f30ab
testing


HEAD
master
98ca9
34ac2
f30ab
testing

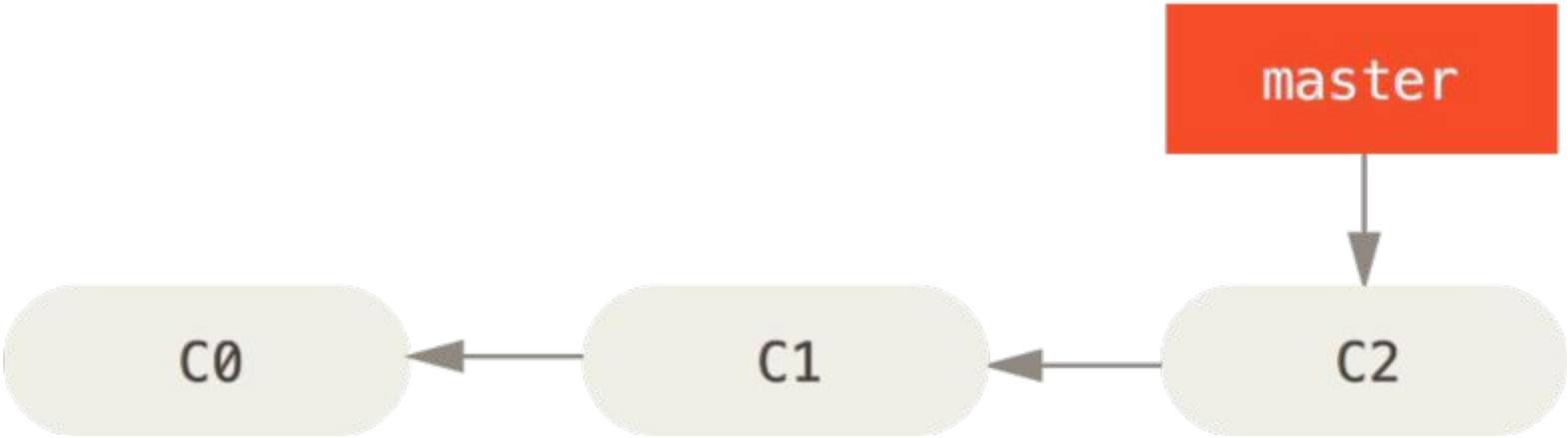

master
98ca9
34ac2
f30ab
testing
HEAD

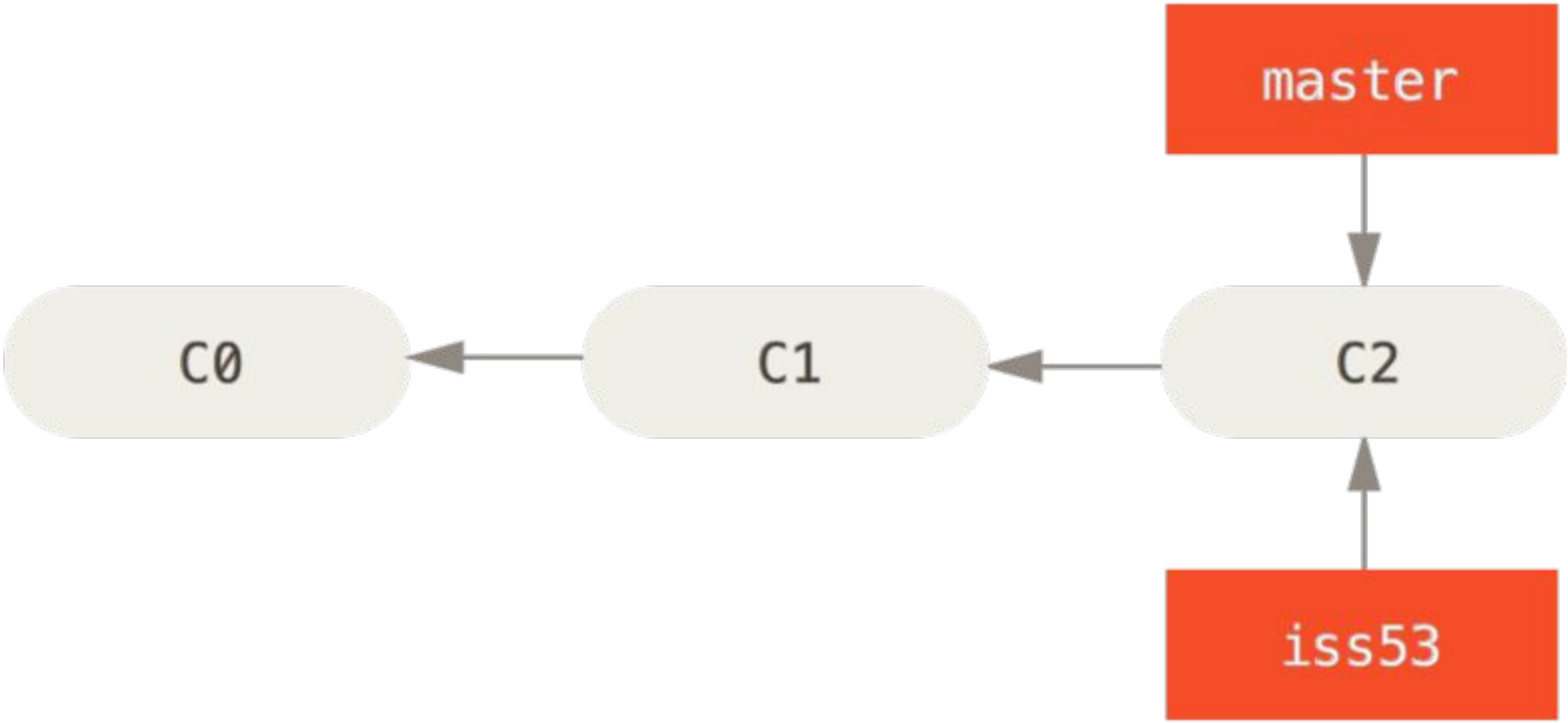

master
98ca9
34ac2
f30ab
87ab2
testing
HEAD

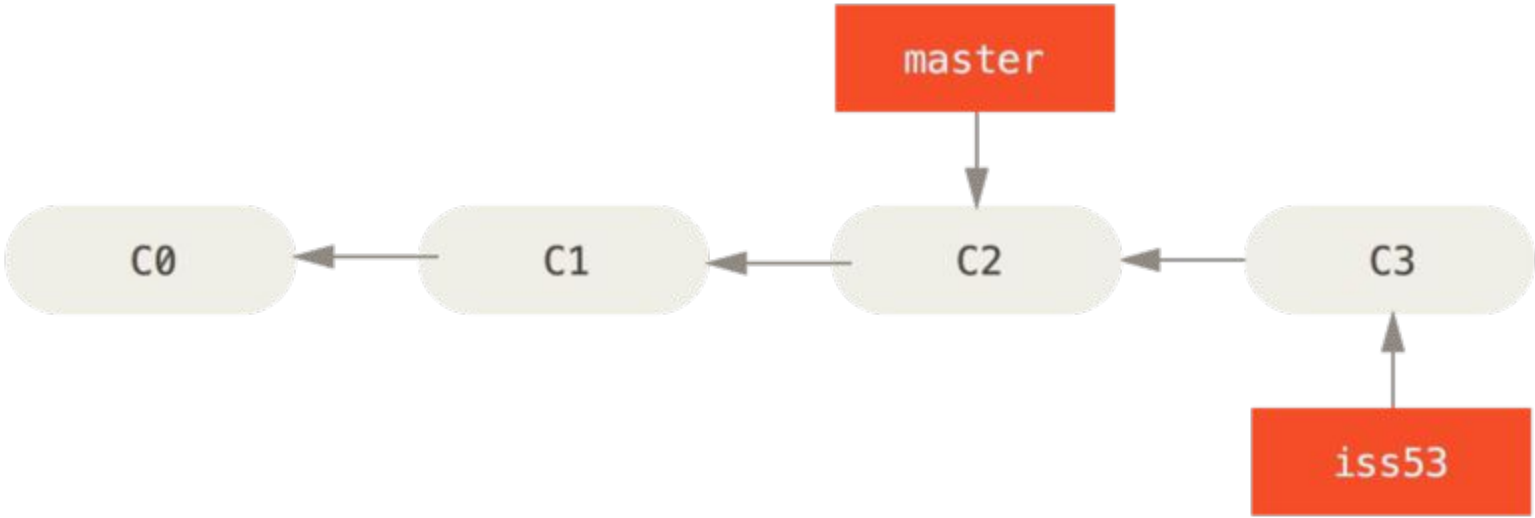

HEAD
master
98ca9
34ac2
f30ab
87ab2
testing

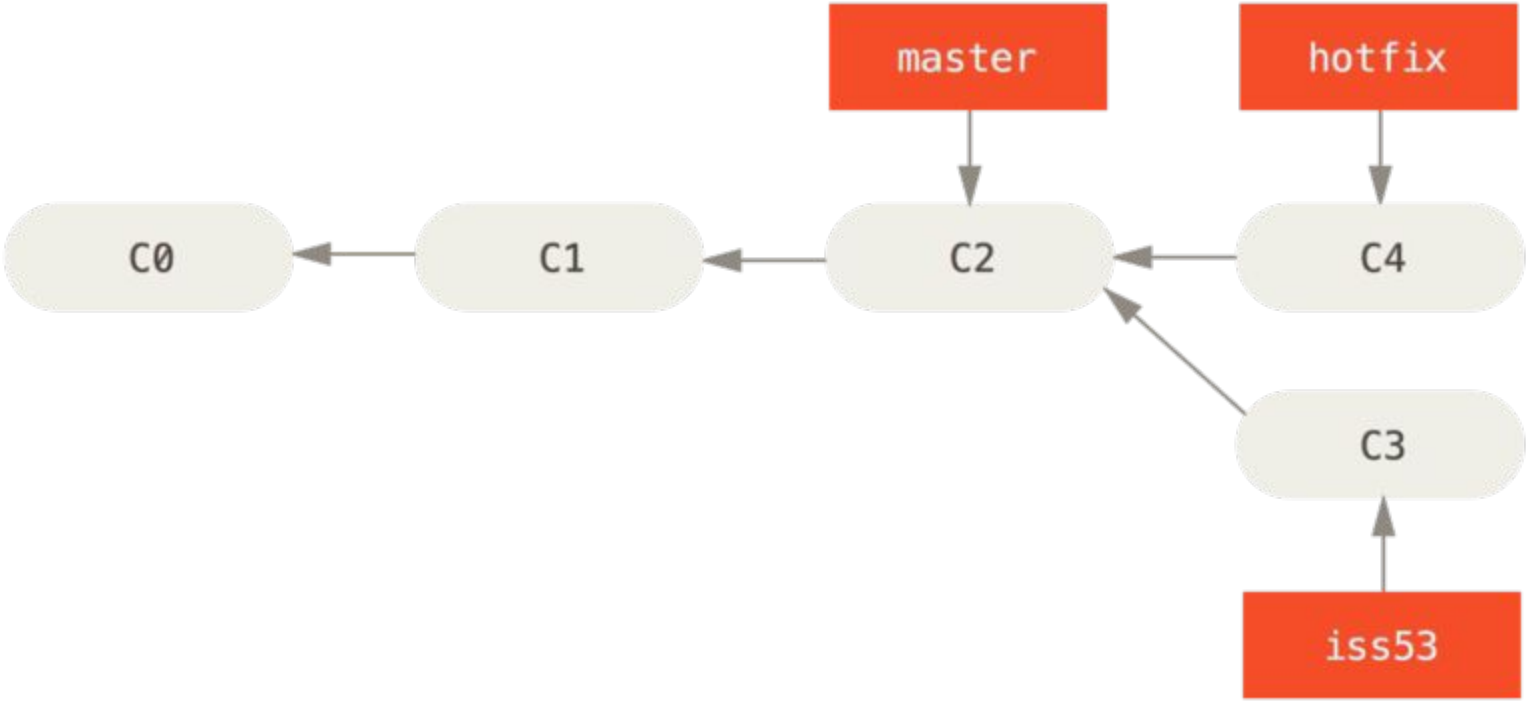

HEAD
master
c2b9e
98ca9
34ac2
f30ab
87ab2
testing

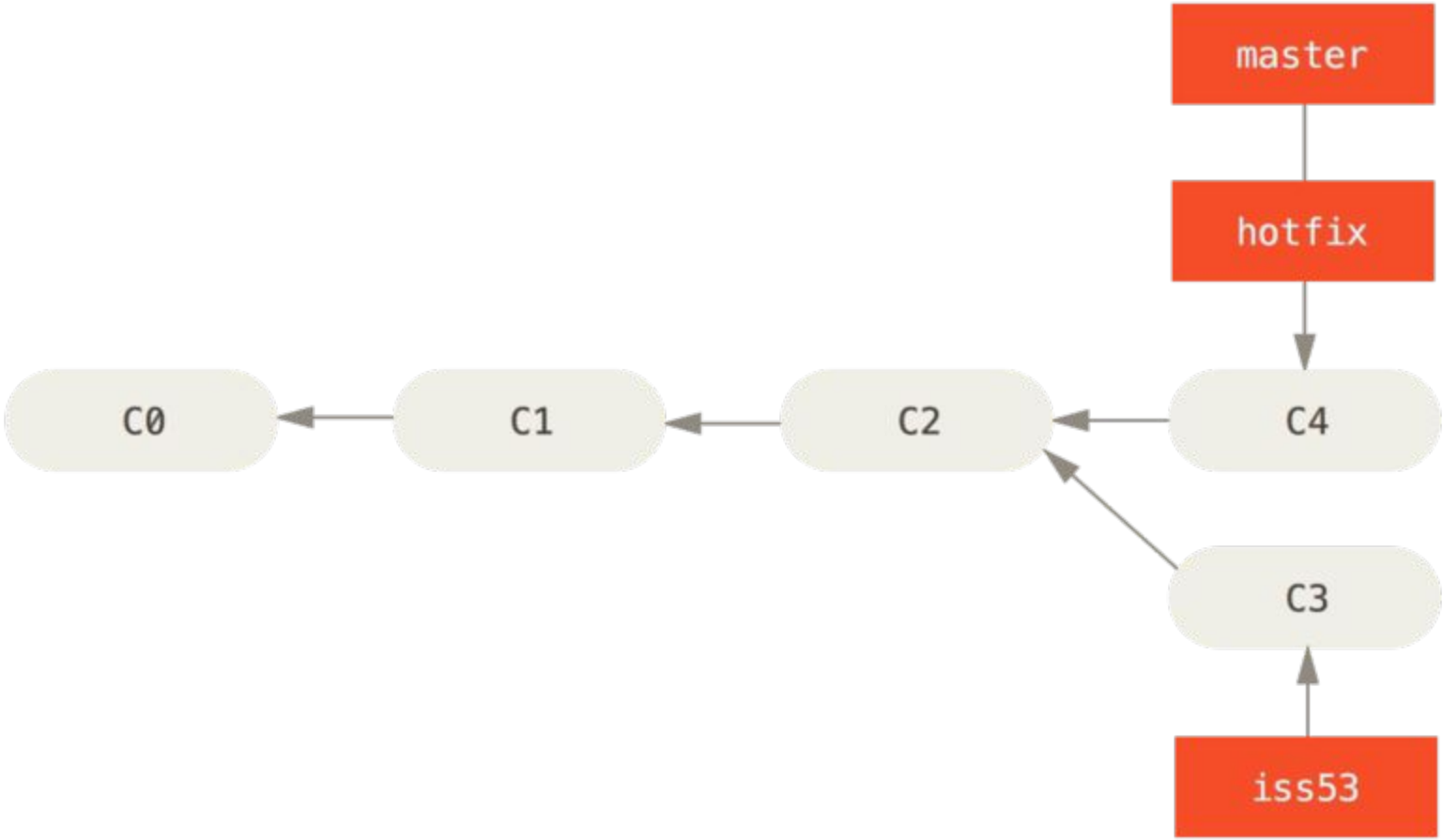
A merging example.

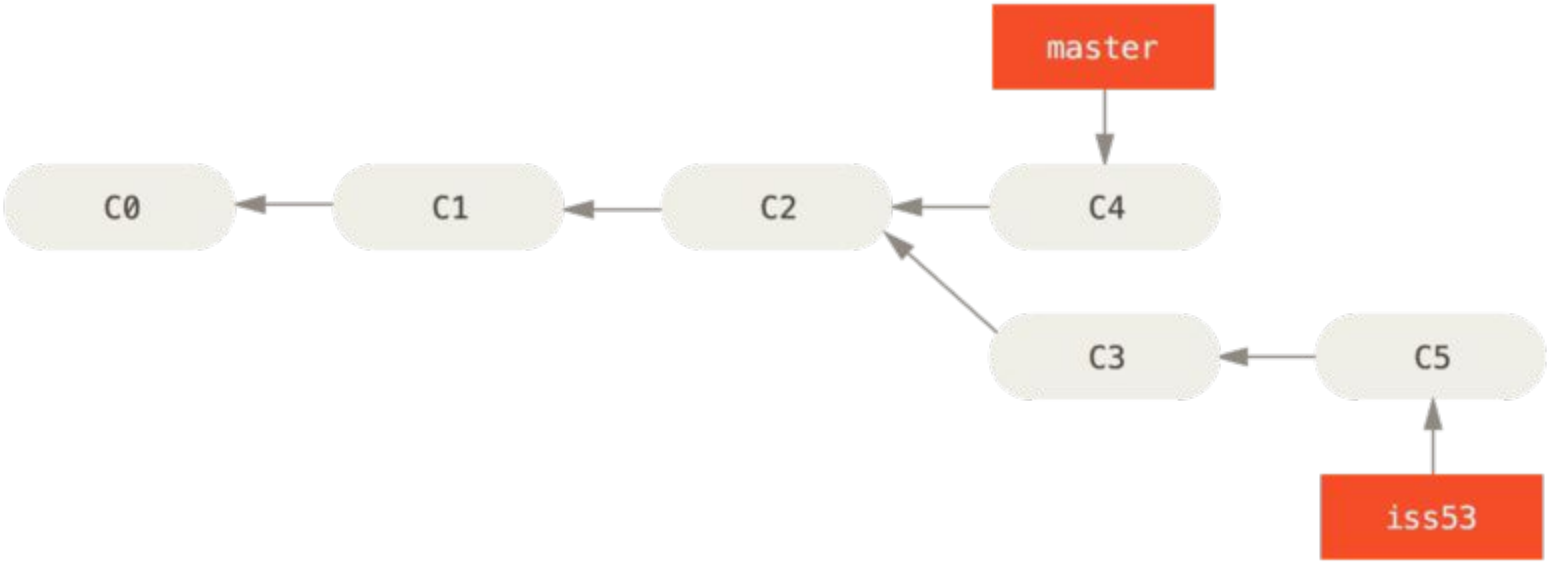

master
C0
C1
C2

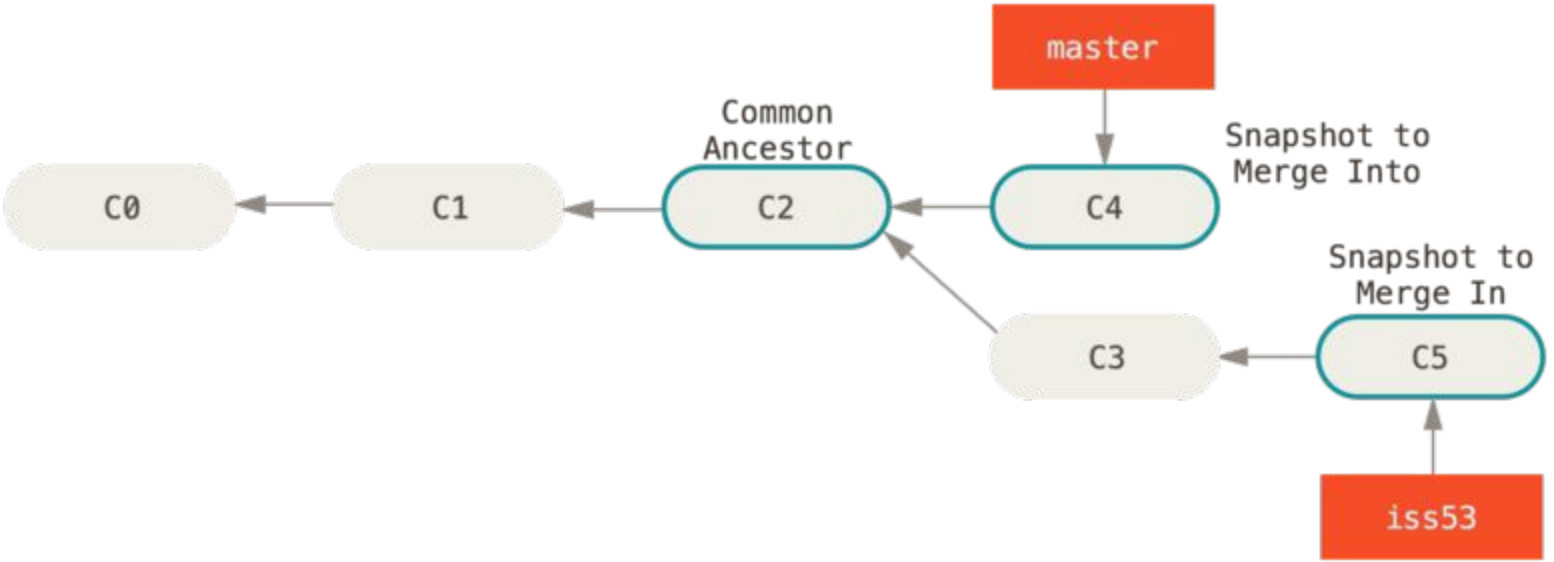

master
C0
C1
C2
iss53

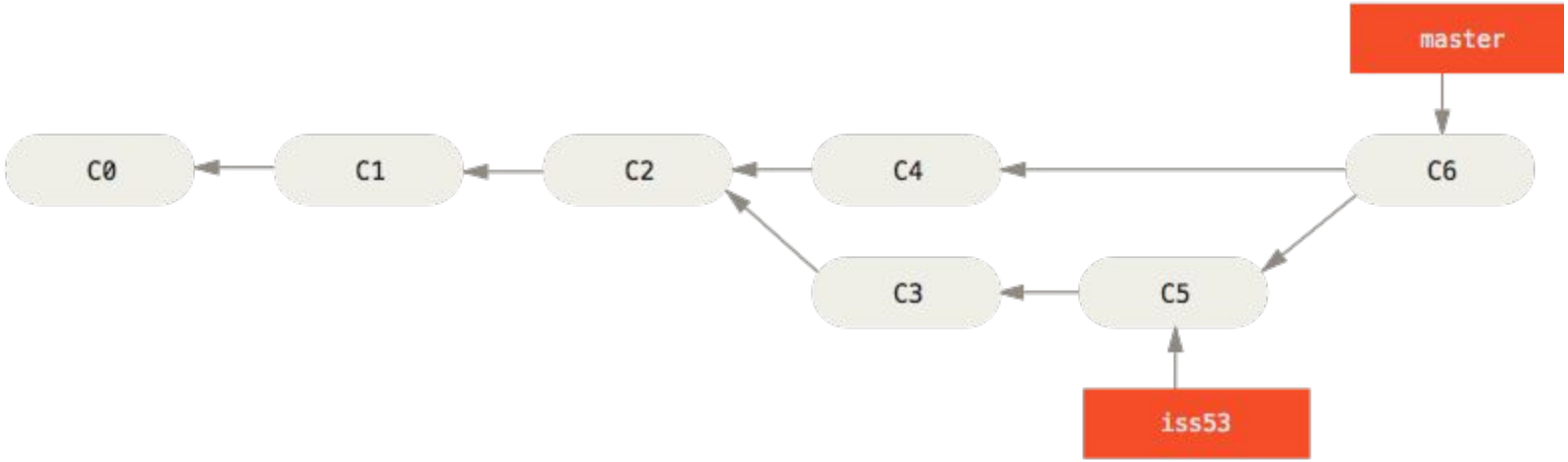

master
C0
C1
C2
C3
iss53


master
hotfix
C0
C1
C2
C4
C3
iss53


master
hotfix
C0
C1
C2
C4
C3
iss53


master
C0
C1
C2
C4
C3
C5
iss53

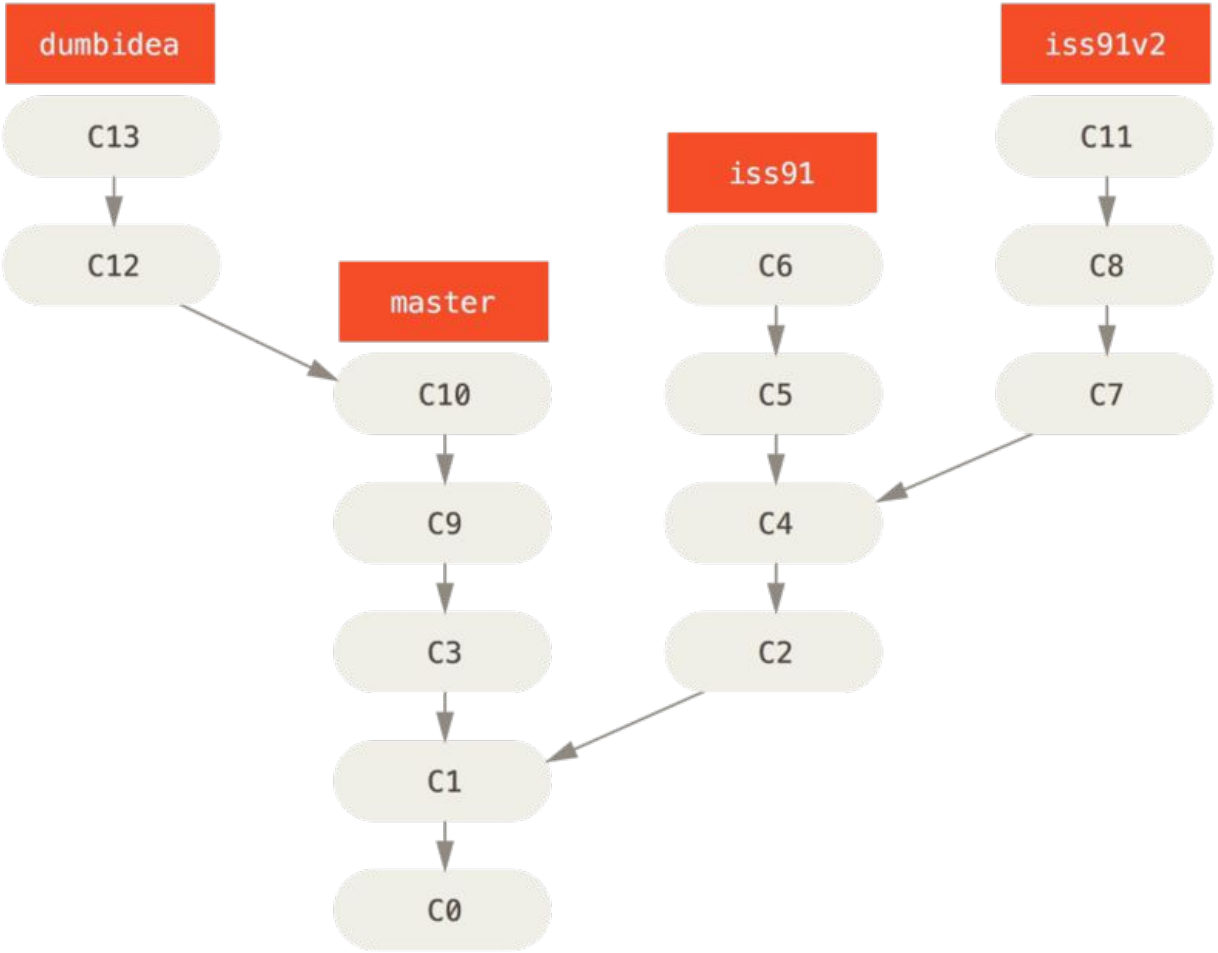

master
Common
Ancestor
Snapshot to
Merge Into
C0
C1
C2
C4
Snapshot to
Merge In
C3
C5
iss53

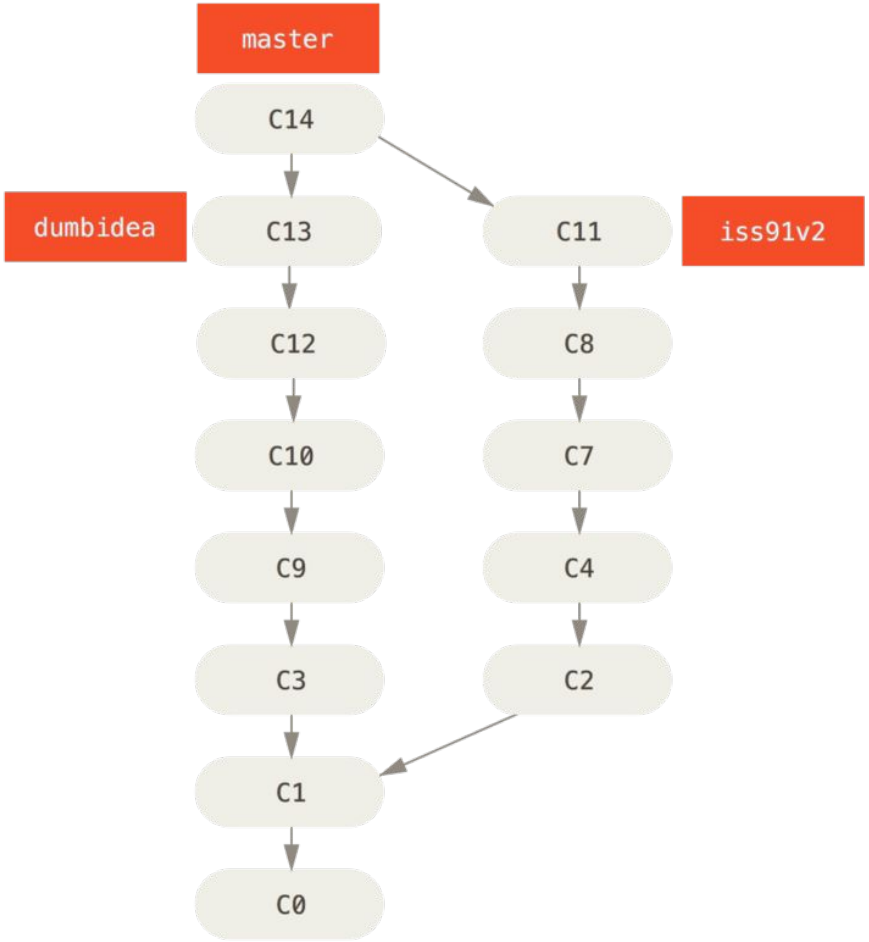

master
C0
C1
C2
C4
C6
C3
C5
iss53

Why branches?

- Branches are typically used to organize work on different parts of the same codebase.
- These branches are typically called “topic branches,” and are the system we’ll be using on the IREC projects.

An example of topic branches.


dumbidea
C13
C12
master
C10
C9
C3
C1
C0
iss91
C6
C5
C4
C2
iss91v2
C11
C8
C7


master
C14
dumbidea
C13
C11
iss91v2
C12
C8
C10
C7
C9
C4
C3
C2
C1
C0

So how do we share the code?

Pushing and pulling!

DEMO TIME