

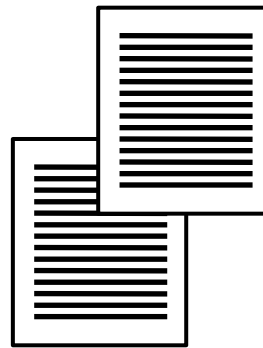
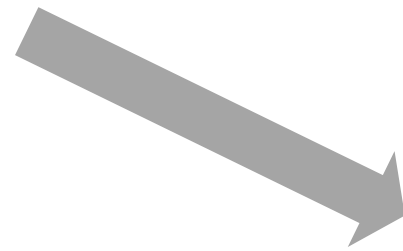
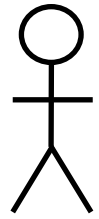
# Distributed version control with git — a brief introduction

Joel Krebs  
Slides by Oscar Nierstrasz

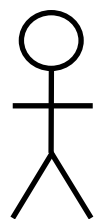


**Why git?**

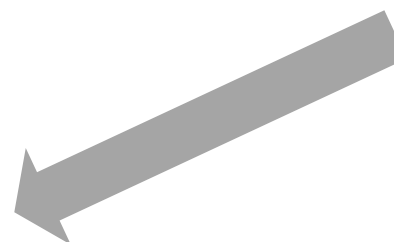
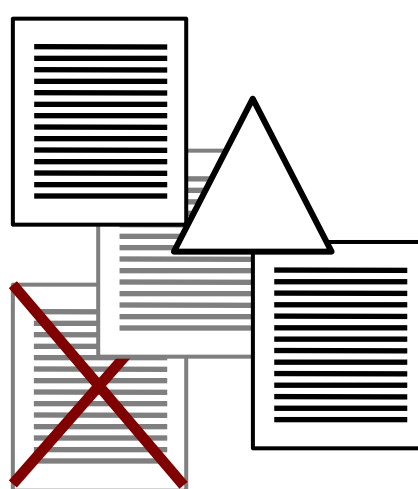
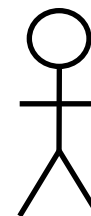
**Bob**



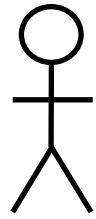
**Bob**



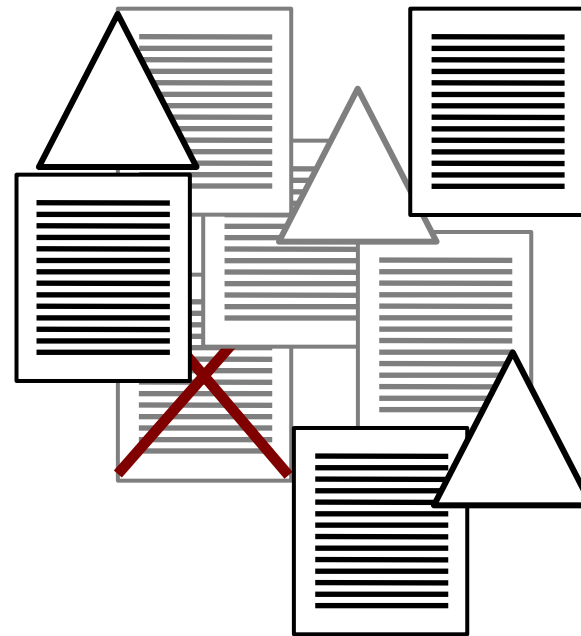
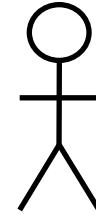
**Carol**



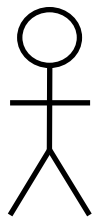
**Bob**



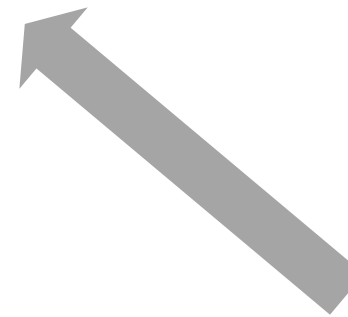
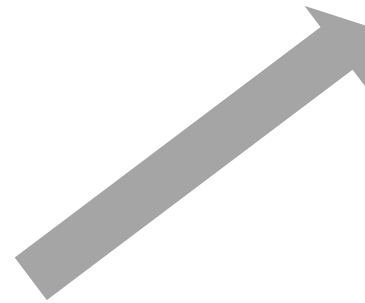
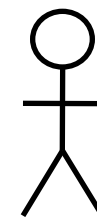
**Carol**



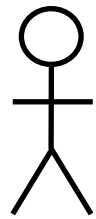
**Alice**



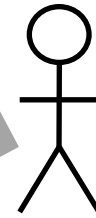
**Ted**



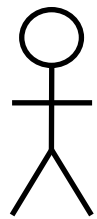
**Bob**



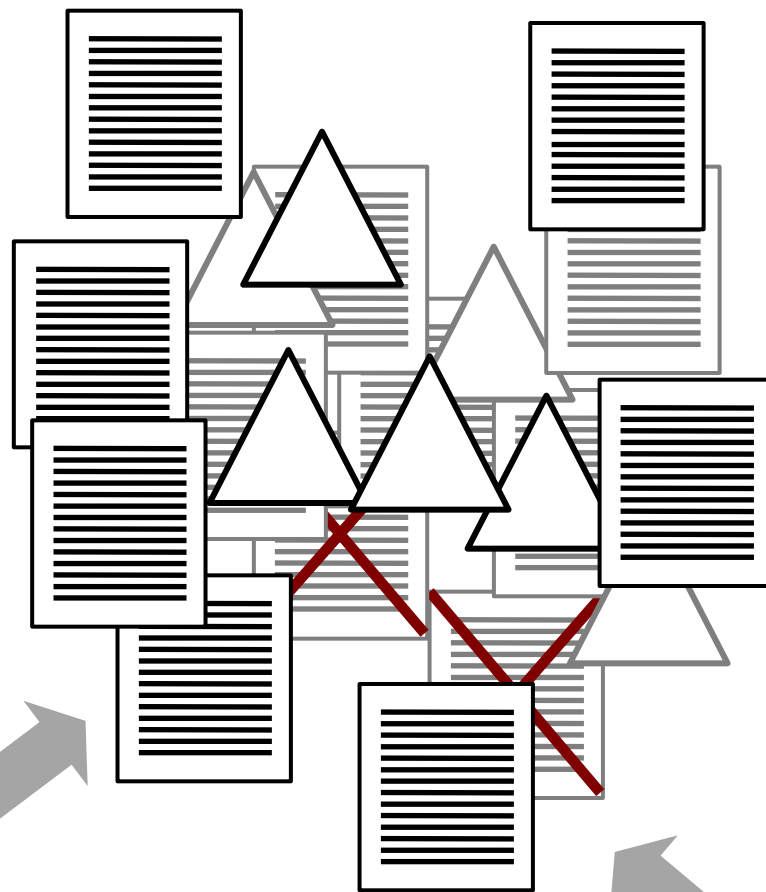
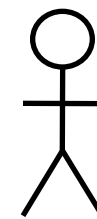
**Carol**



**Alice**

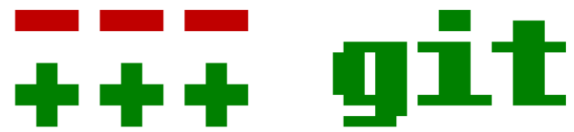


**Ted**



**A recipe for disaster!**

# What is git?



- > Distributed revision control system
- > Originally developed by Linus Torvalds for the development of the Linux Kernel in 2005
- > Focus on speed and efficiency
- > Quite a unique design and therefore sometimes a bit scary and difficult to understand

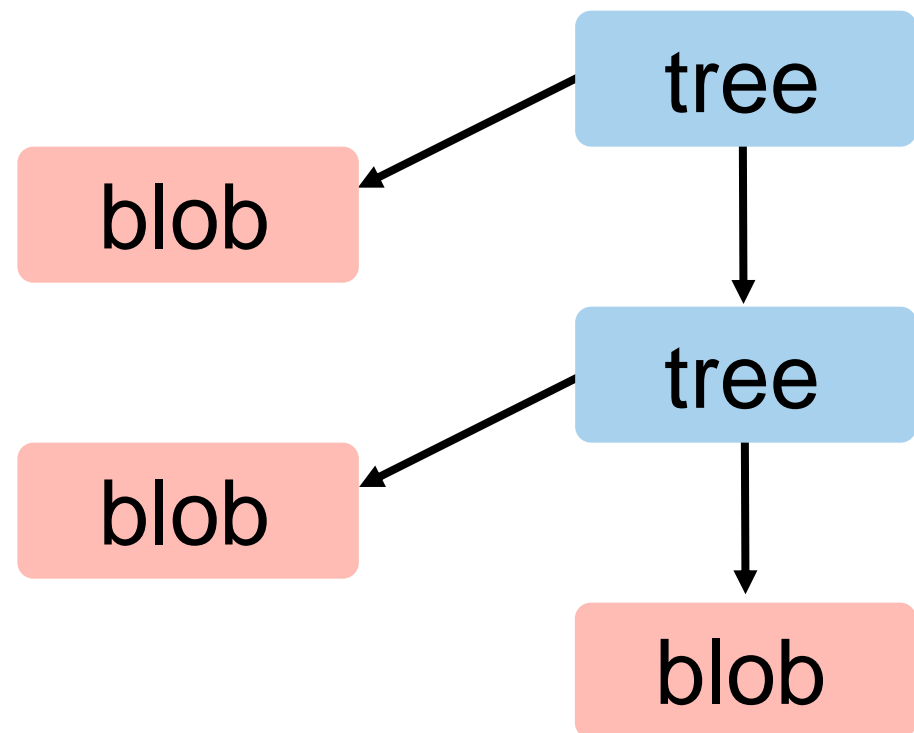


# The git object model

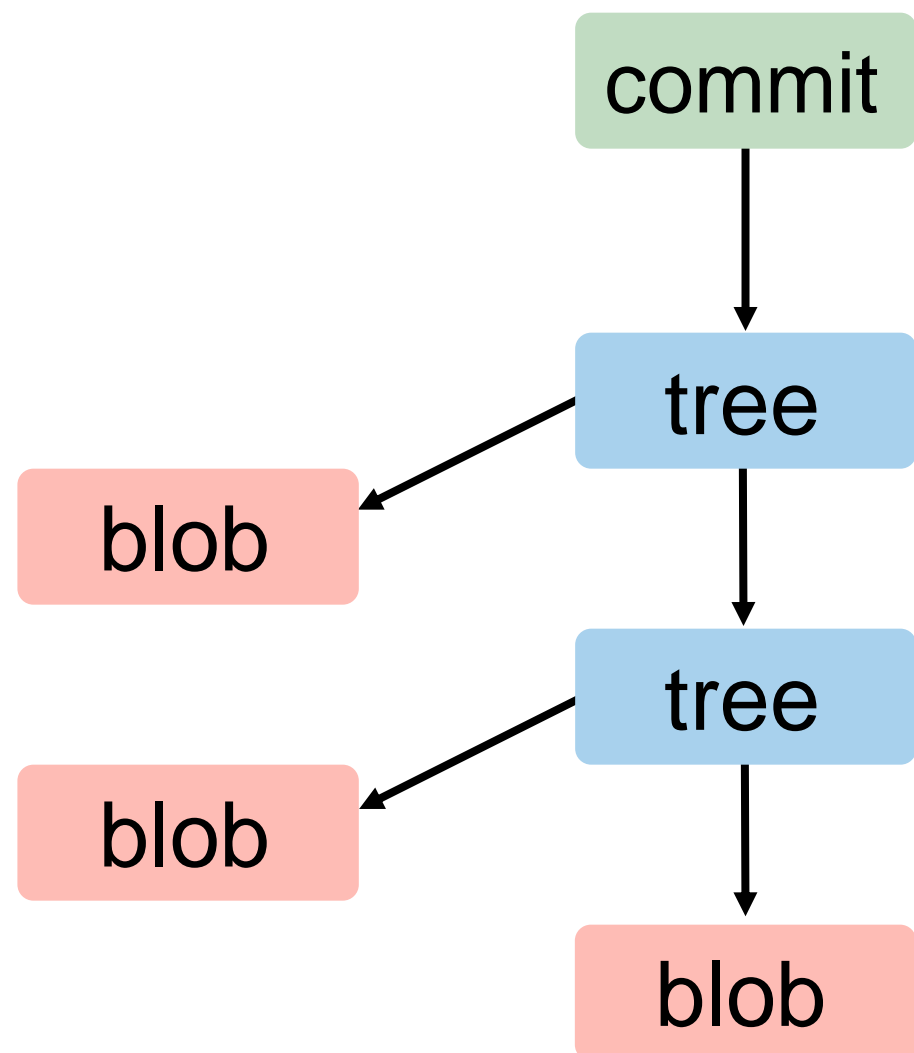
A “blob” is *content* under  
version control (a file)

blob

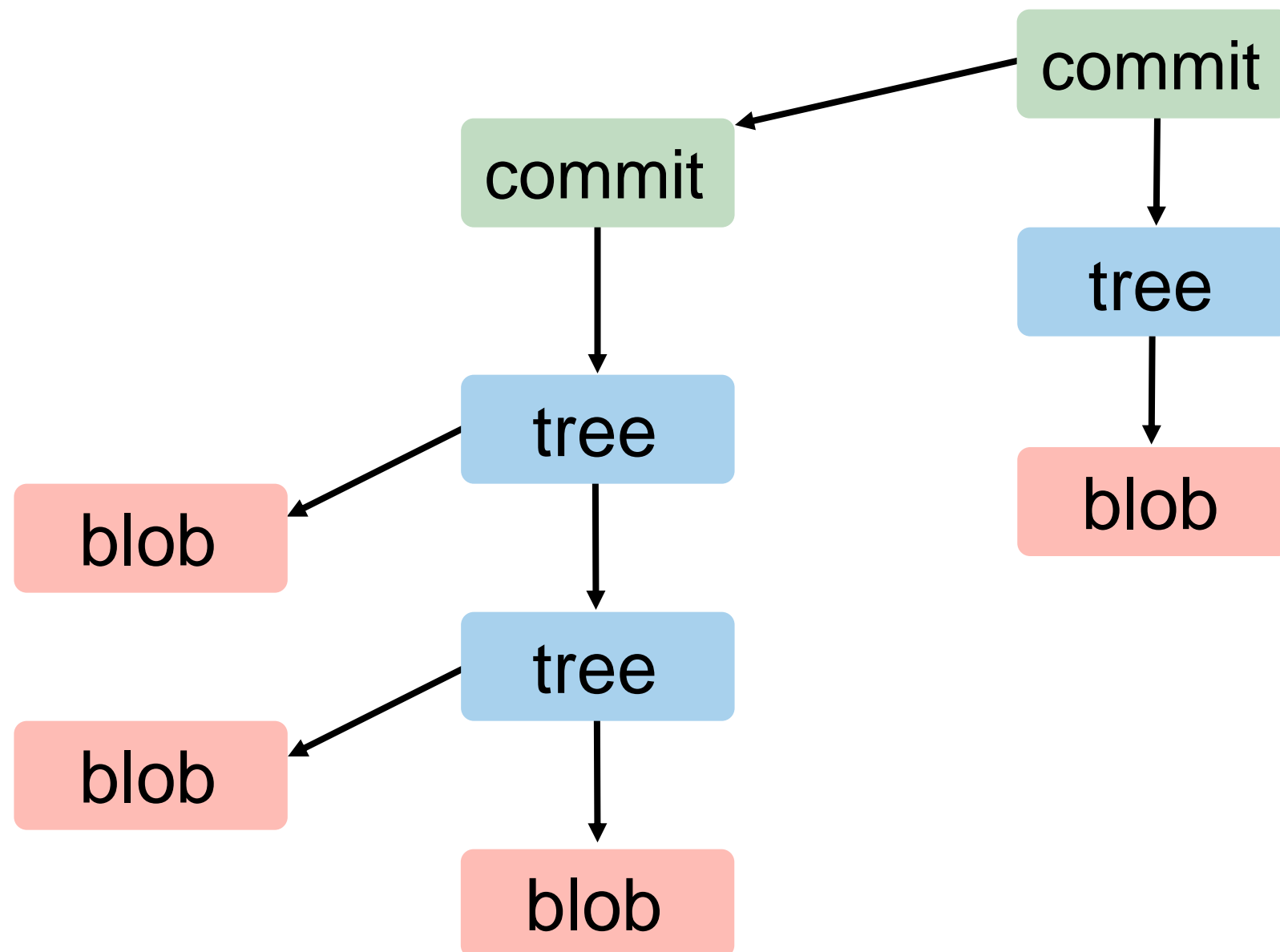
You can have *trees* of blobs  
(directories of files)



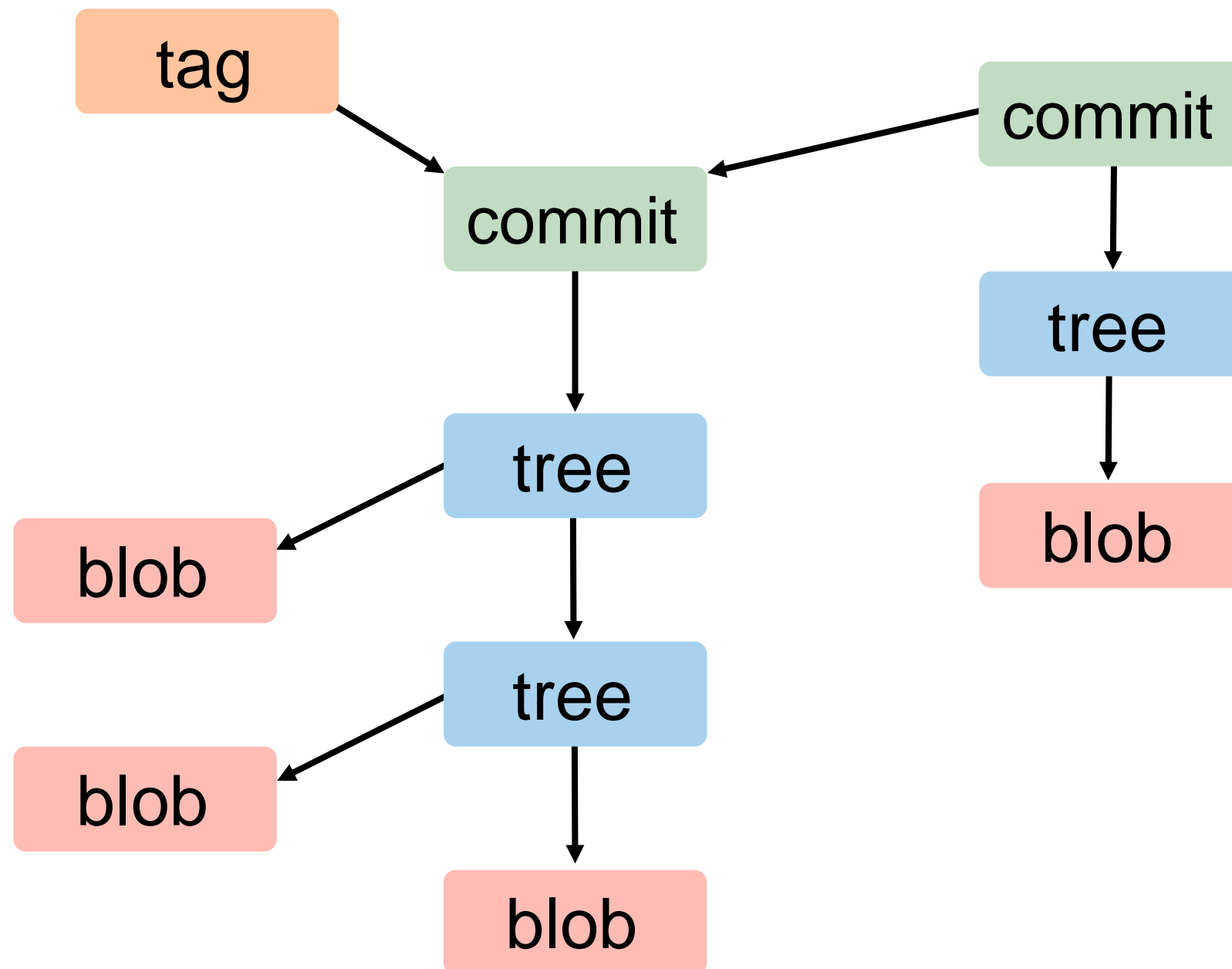
A “commit” is a tree of blobs  
(a set of changes)



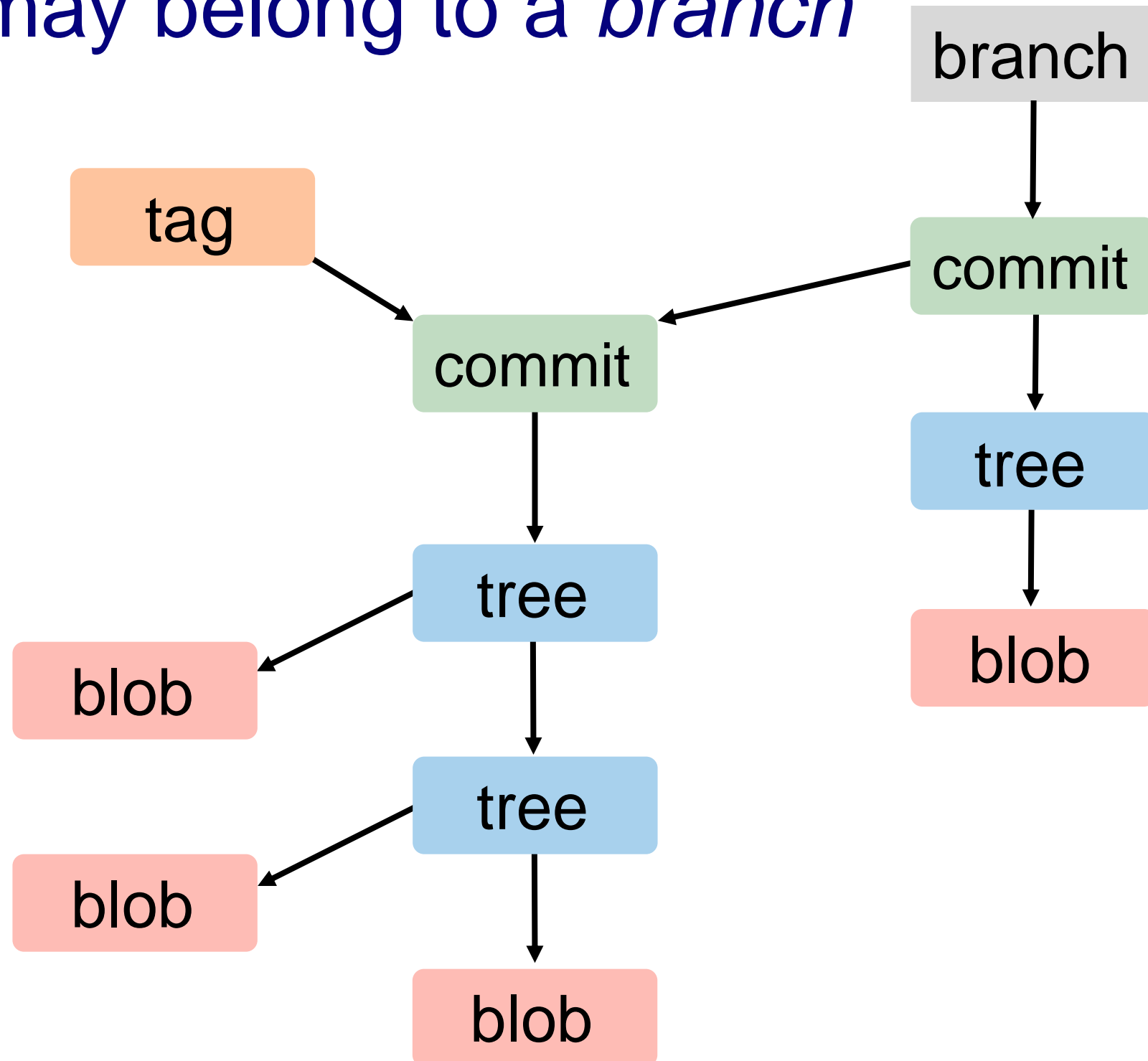
Most commits modify  
(or merge) earlier  
commits



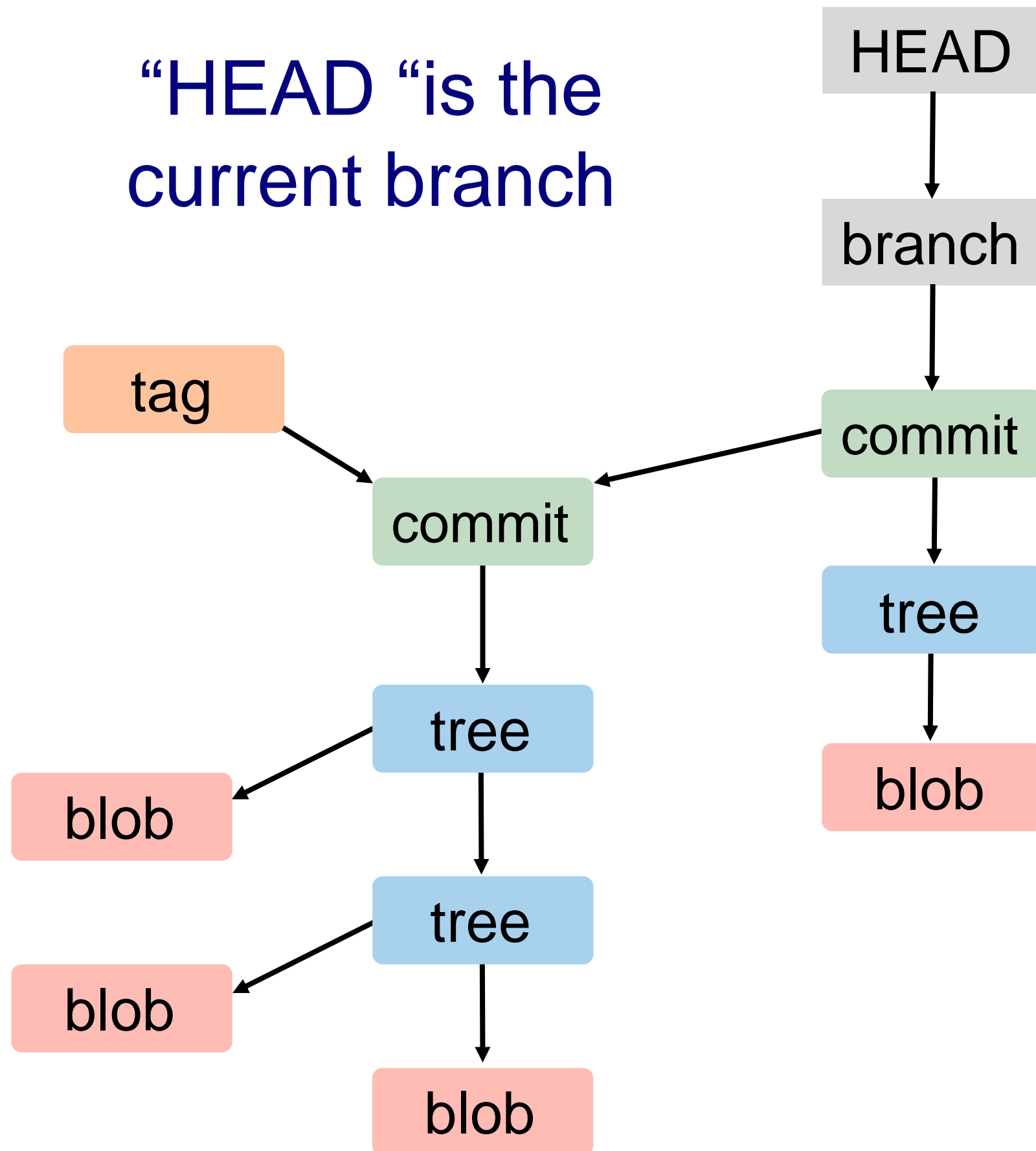
You can “tag” an interesting commit



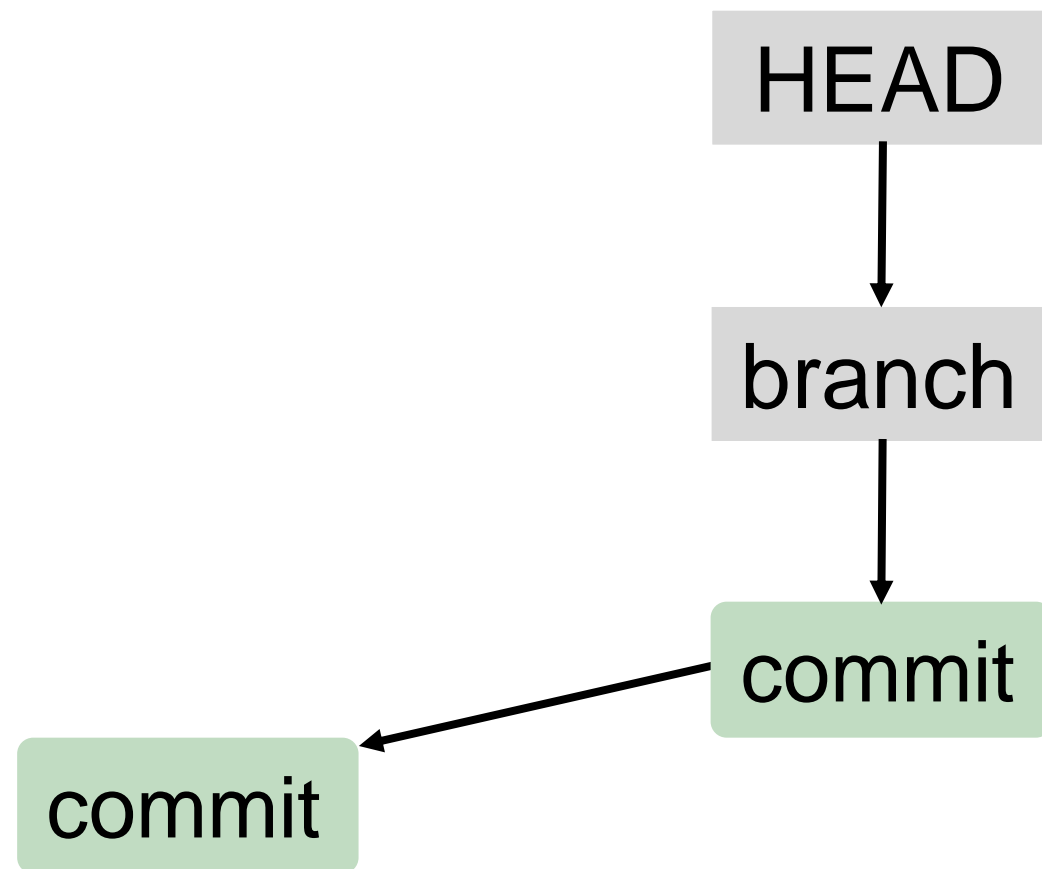
A graph of commits  
may belong to a *branch*



“HEAD “is the  
current branch





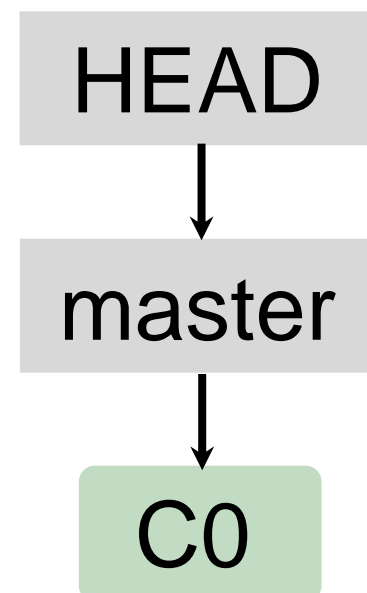


Let's focus on *commits*  
and *branches*

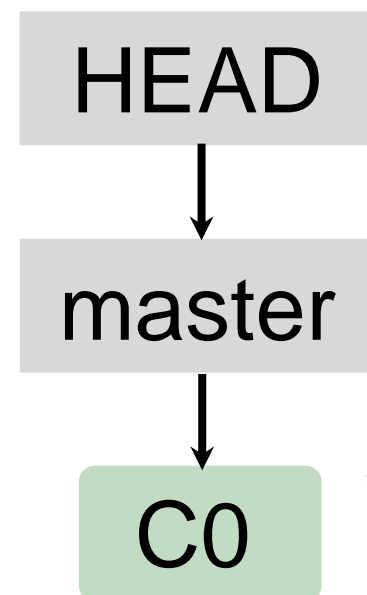
# Basic git

# Create a git repo

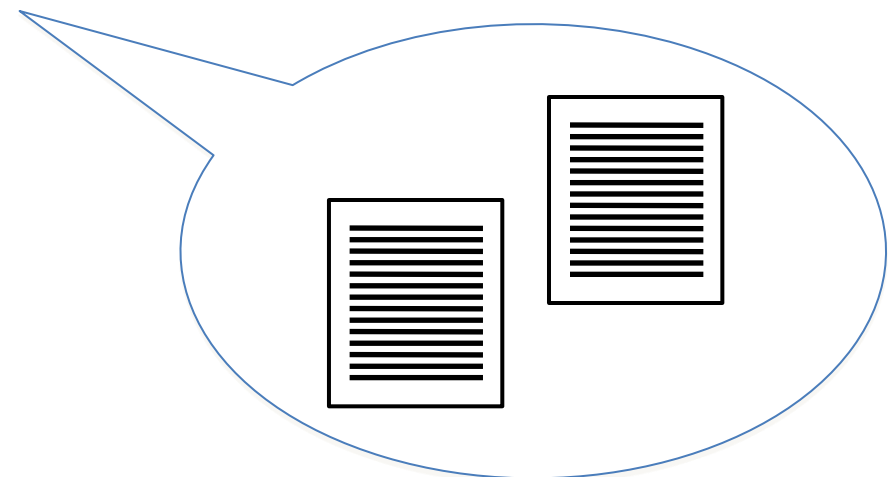
```
mkdir repo  
cd repo  
git init
```

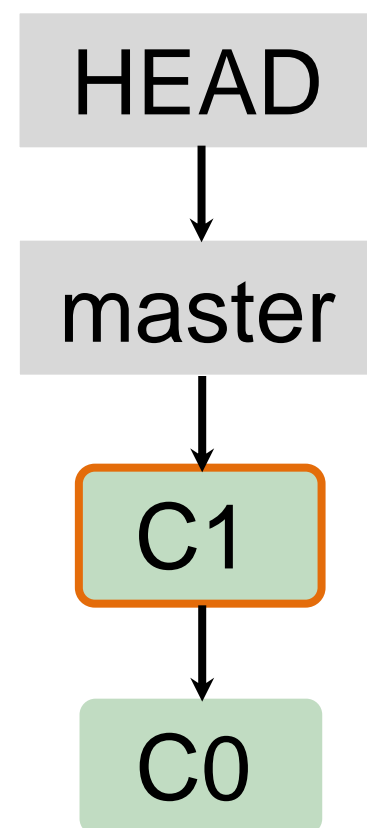


Tell git to “stage”  
changes



**git add ...**

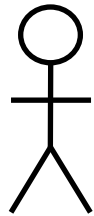




Commit your  
changes

**`git commit ...`**

# Collaborating

 **John**

**Jane** 

Local repo

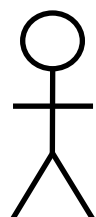
Public repo

Local repo

master

C1

C0

 **John**

**Jane** 

Local repo

Public repo

Local repo

**git clone ...**

master

C1

C0

master

C1

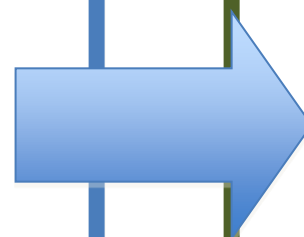
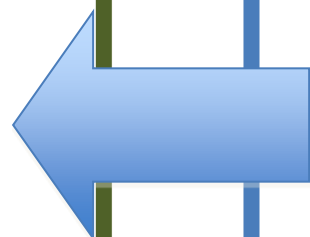
C0

master

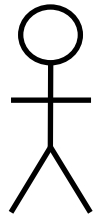
C1

C0

**git clone ...**

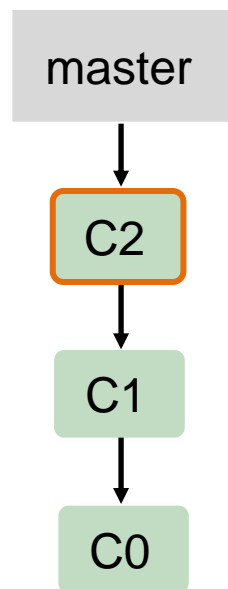




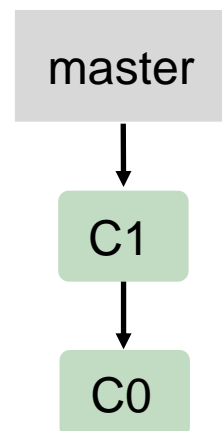
 **John**

**Jane** 

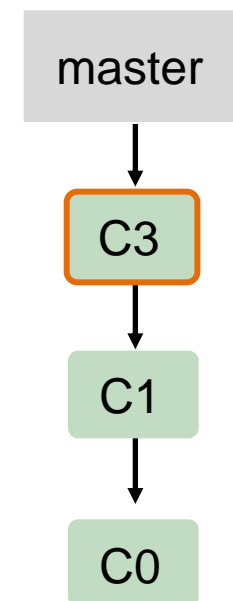
Local repo



Public repo

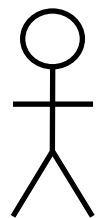


Local repo



**git add ...**  
**git commit ...**

**git add ...**  
**git commit ...**

 **John**

**Jane** 

Local repo

Public repo

Local repo

**git pull**

master

C2

C1

C0

master

C1

C0

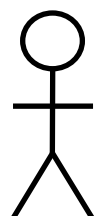
master

C3

C1

C0

(nothing new to pull)

 **John**

**Jane** 

Local repo

**git push**

master

C2

C1

C0

Public repo

master

C2

C1

C0

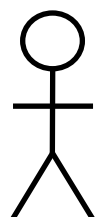
Local repo

master

C3

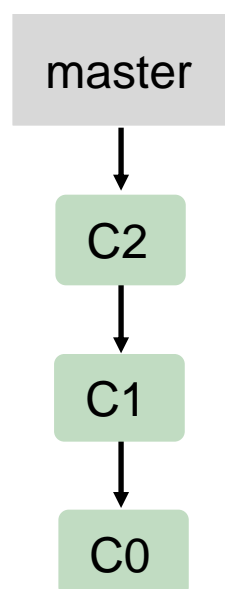
C1

C0

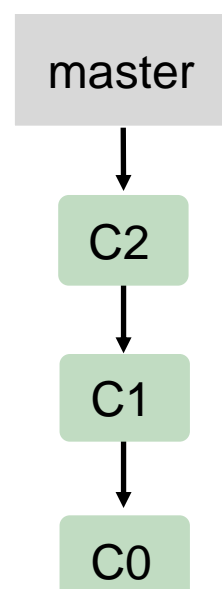
 **John**

**Jane** 

Local repo

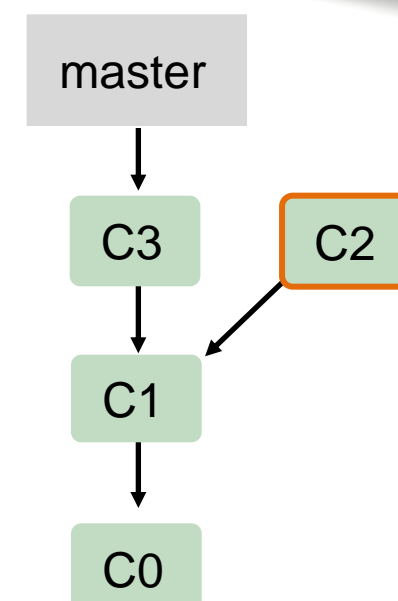
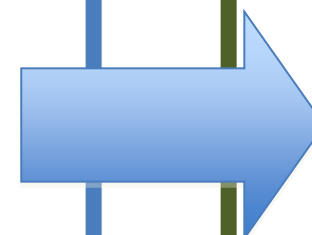


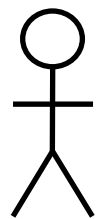
Public repo



Local repo

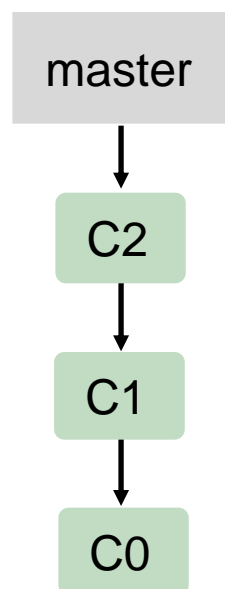
**git fetch**



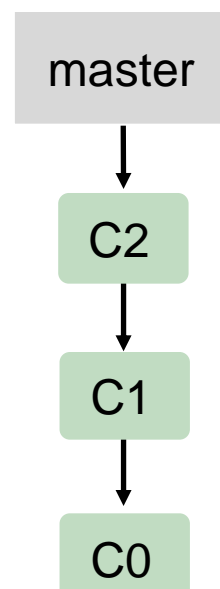
 **John**

**Jane** 

Local repo

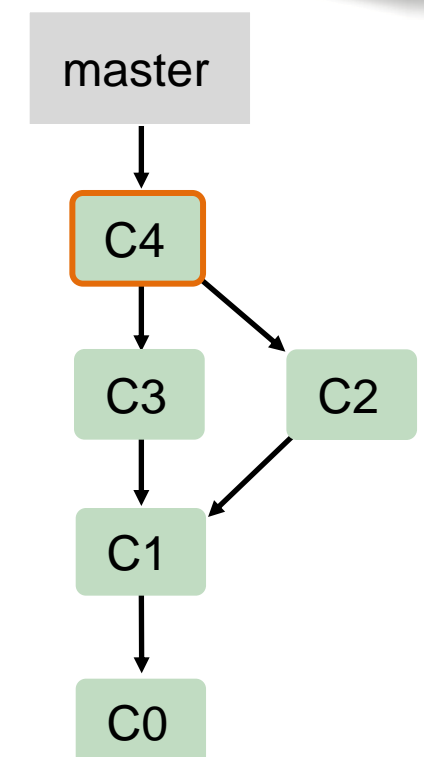


Public repo

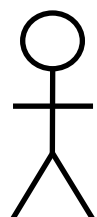


Local repo

**git merge**

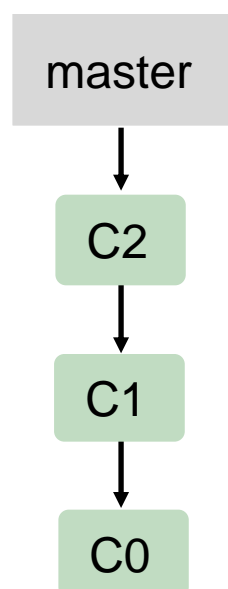


**NB: git pull = fetch + merge**

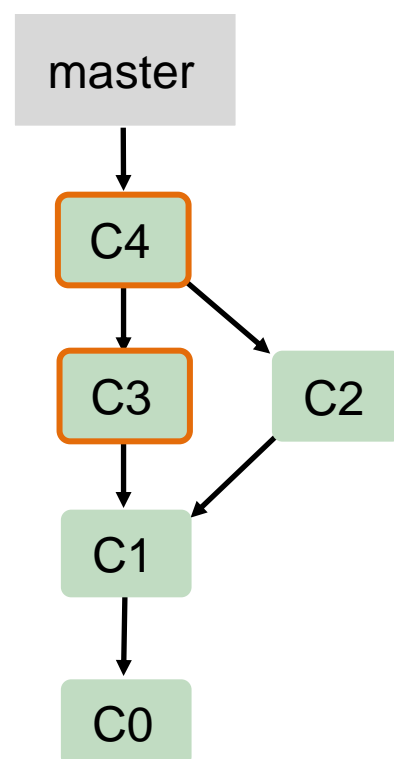
 **John**

**Jane** 

Local repo

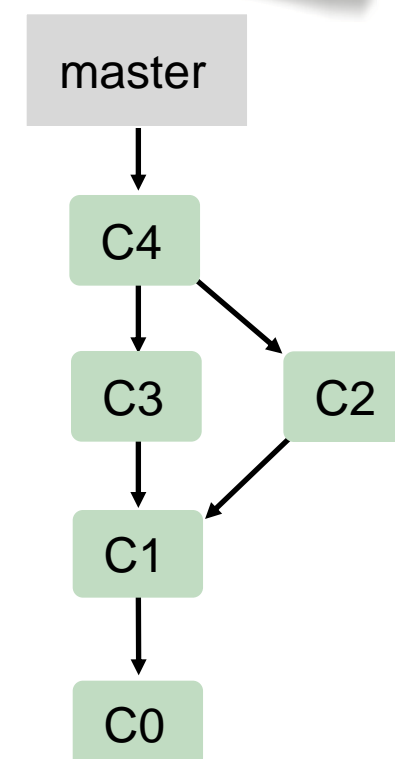
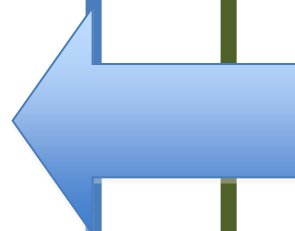


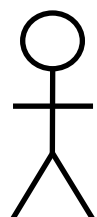
Public repo



Local repo

**git push**

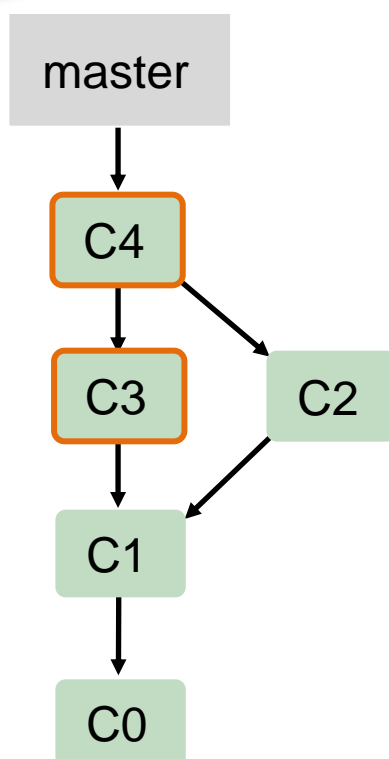


 **John**

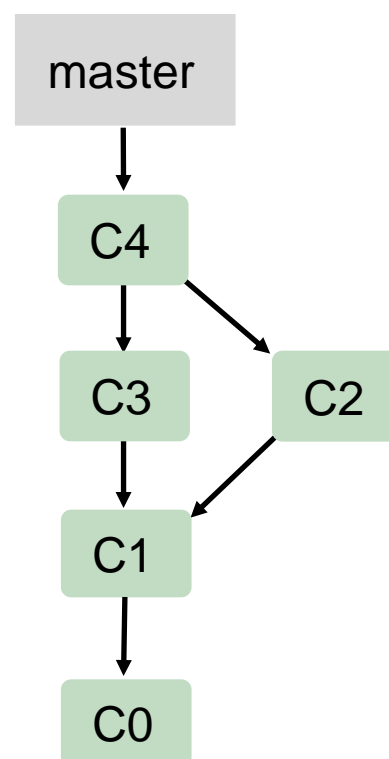
**Jane** 

Local repo

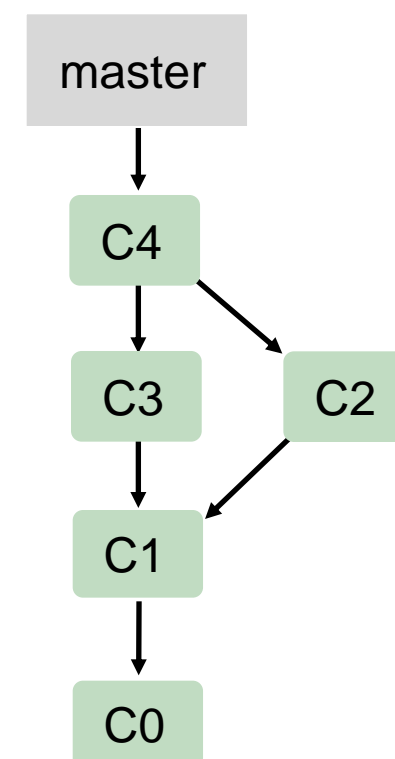
**git pull**



Public repo



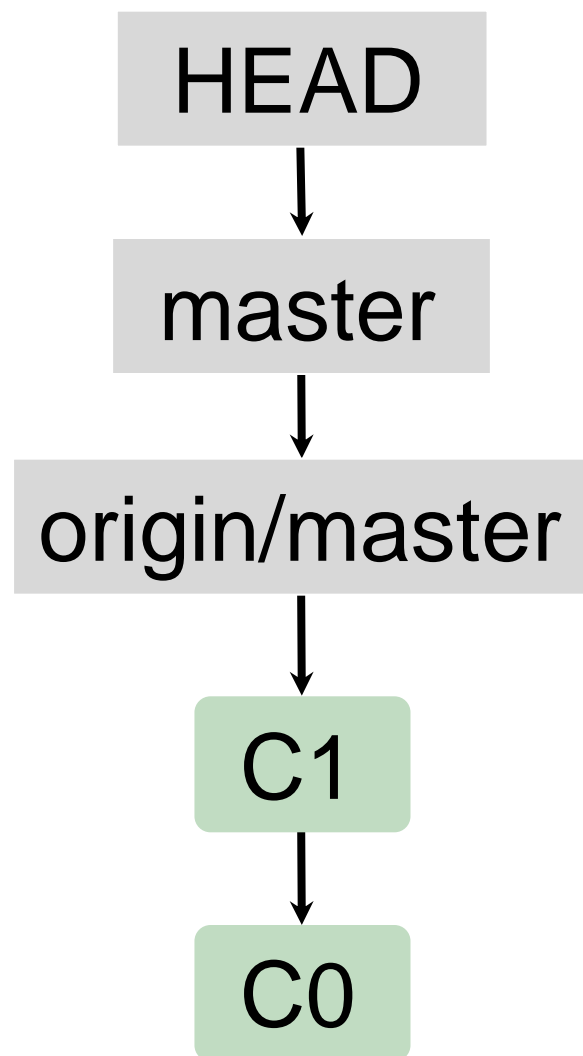
Local repo

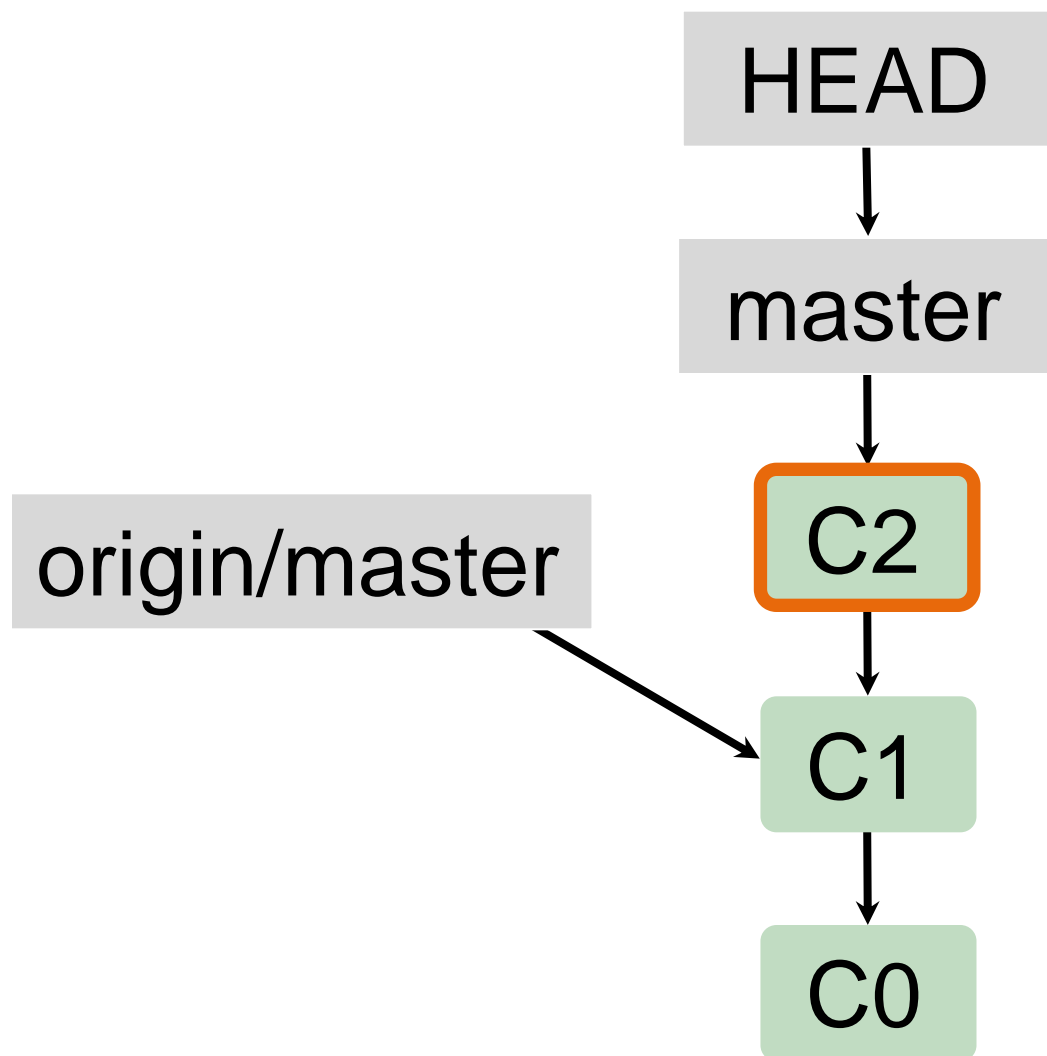


# **Branching and merging**



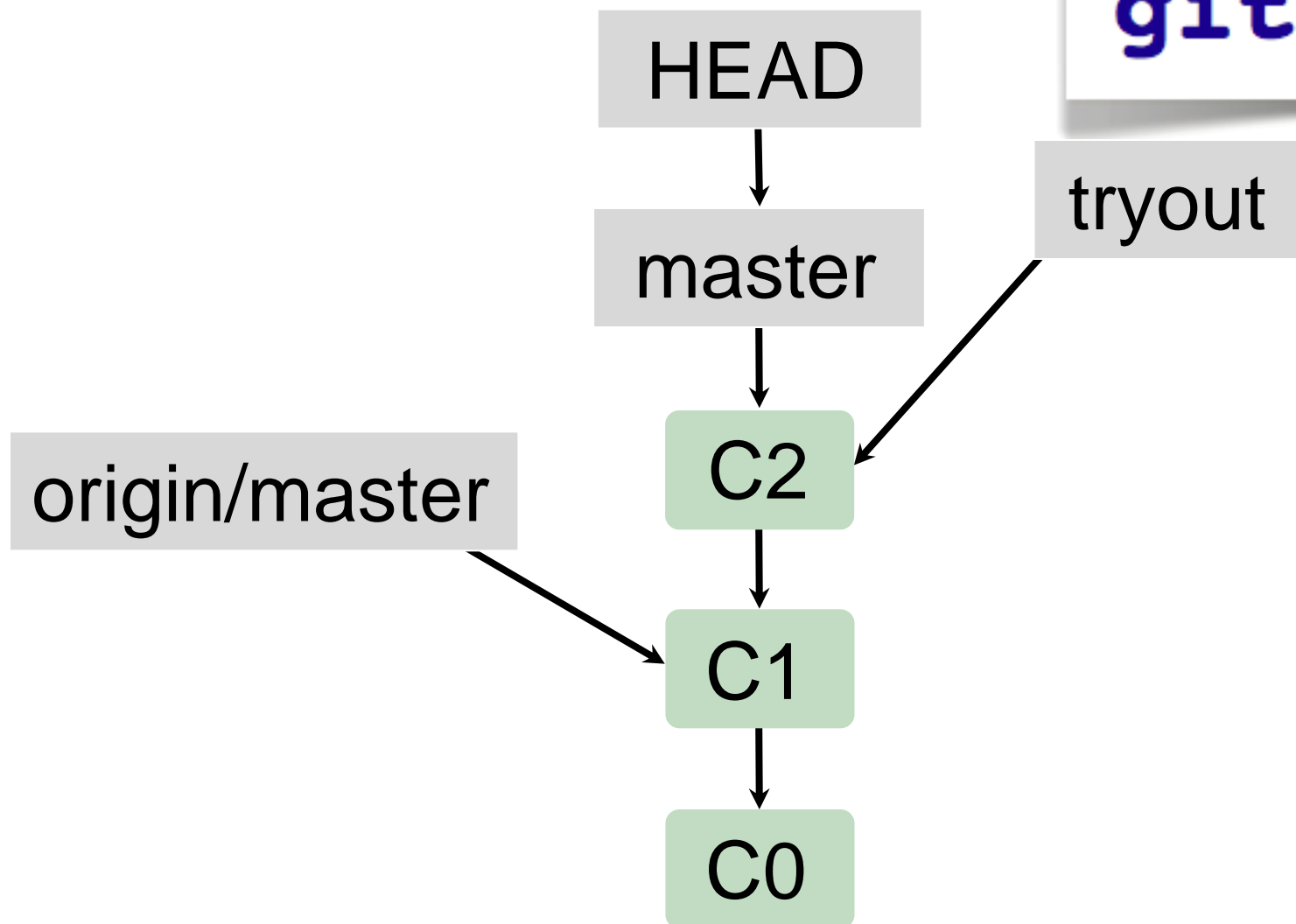
“origin” refers to the  
remote repo



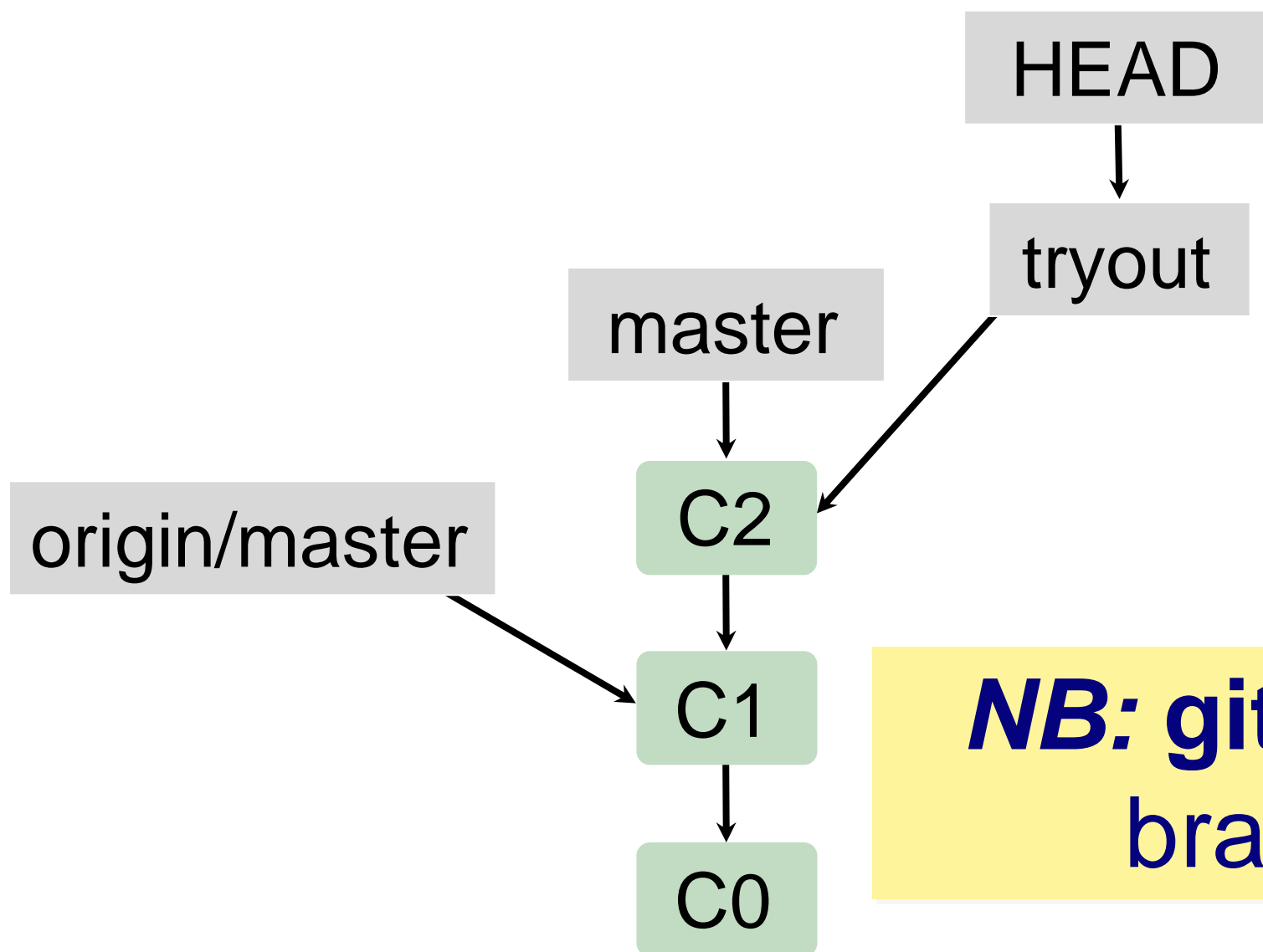


...  
**git commit ...**

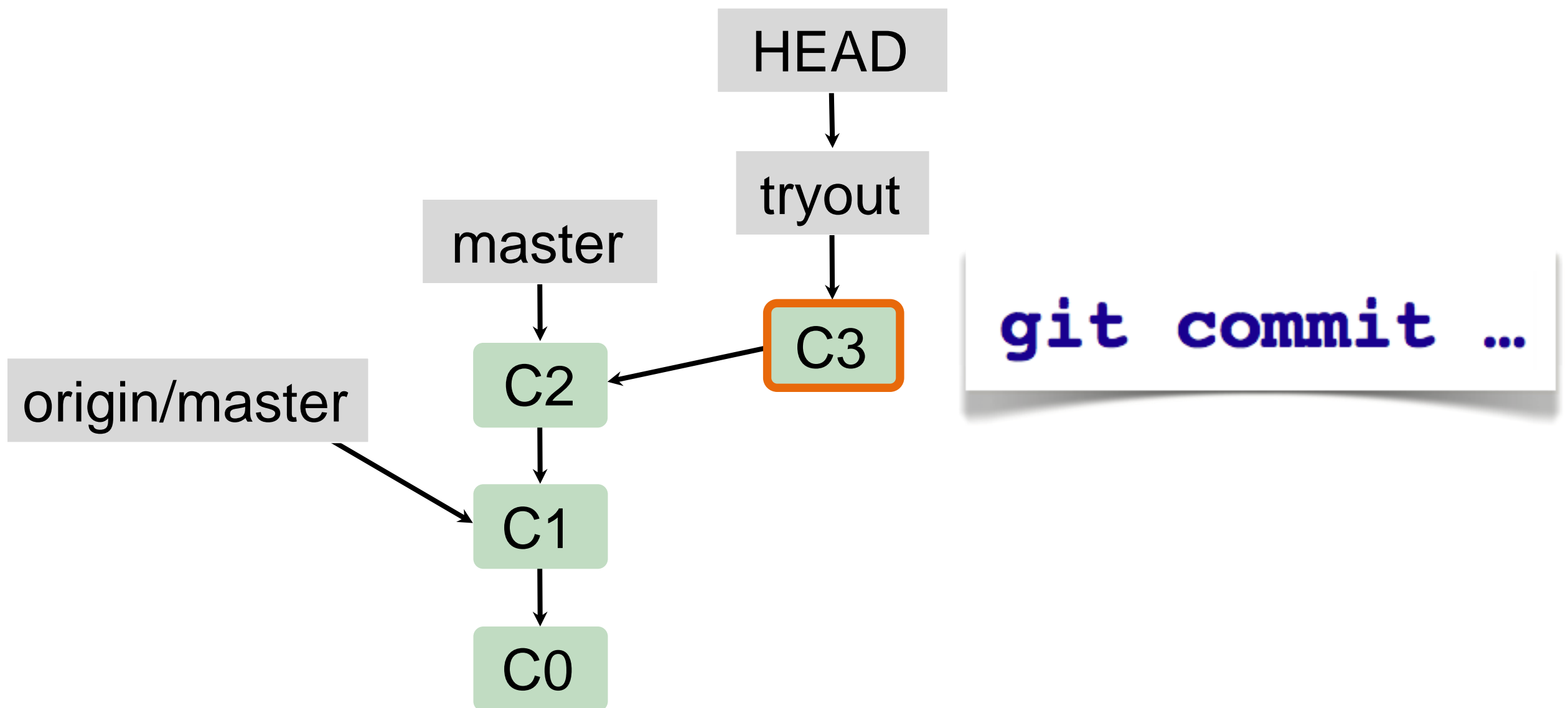
# git branch tryout

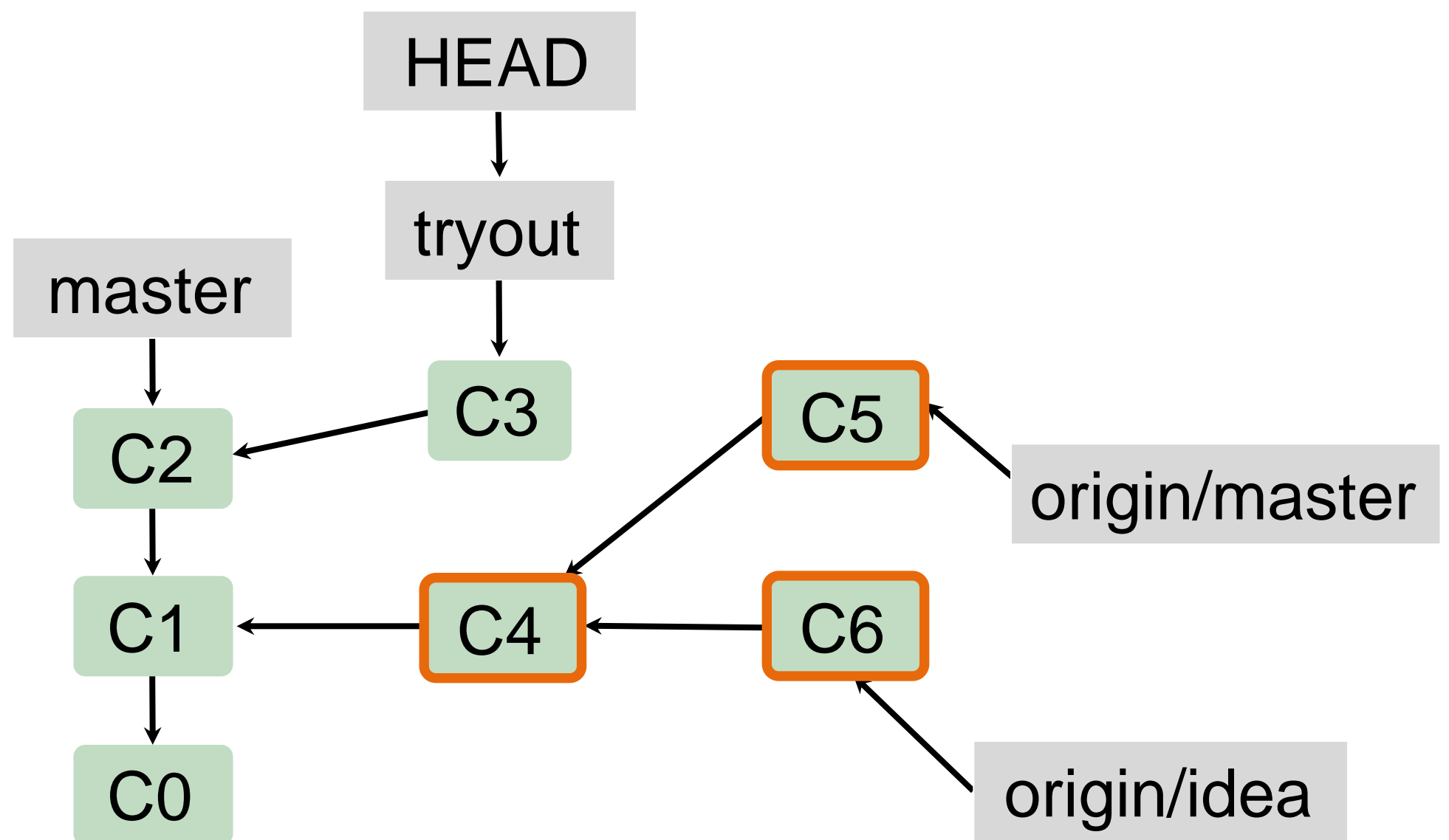


# git checkout tryout



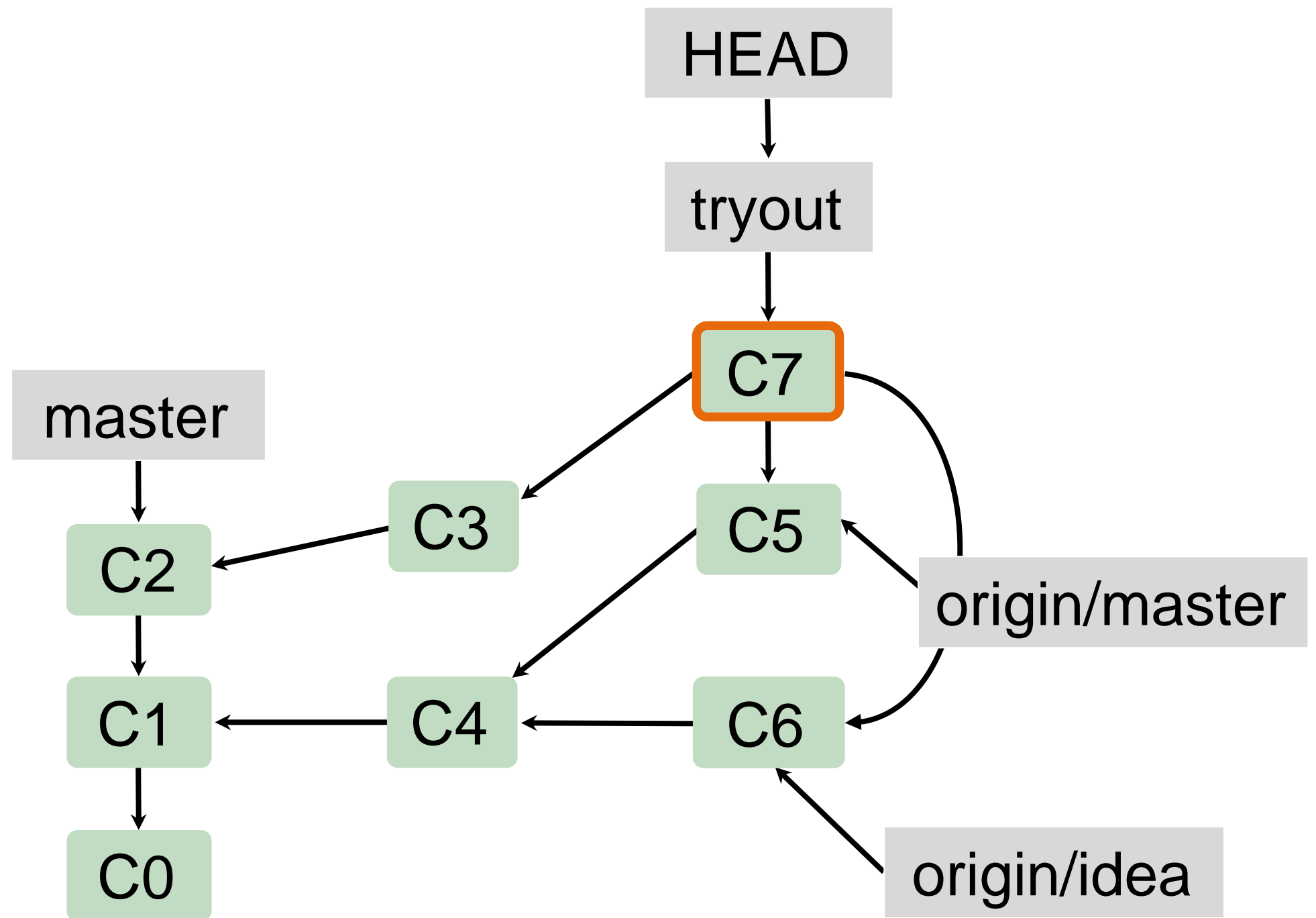
***NB:*** `git checkout -b ...` =  
branch + checkout



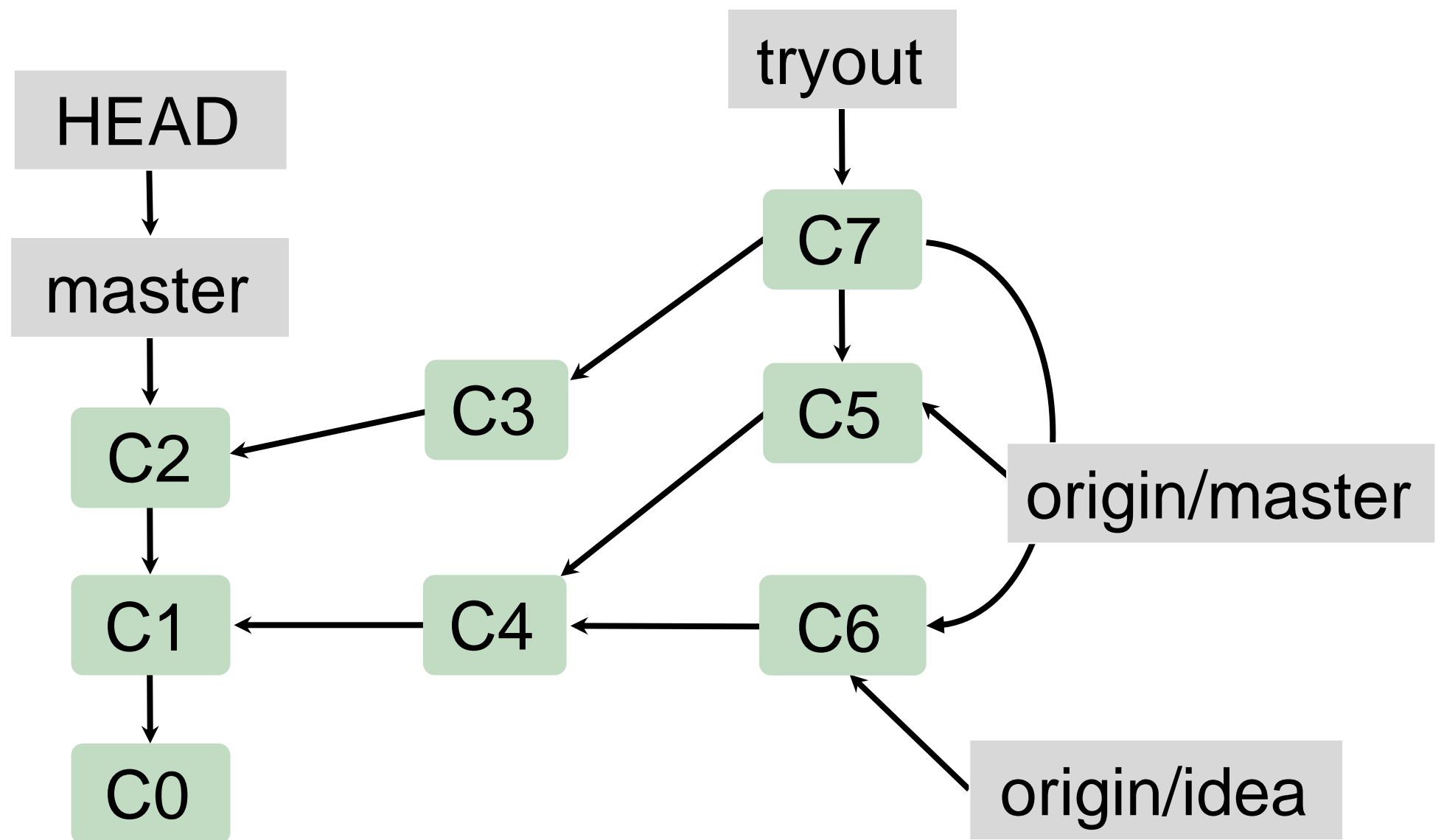


**git fetch origin**

**git merge origin/master origin/idea**

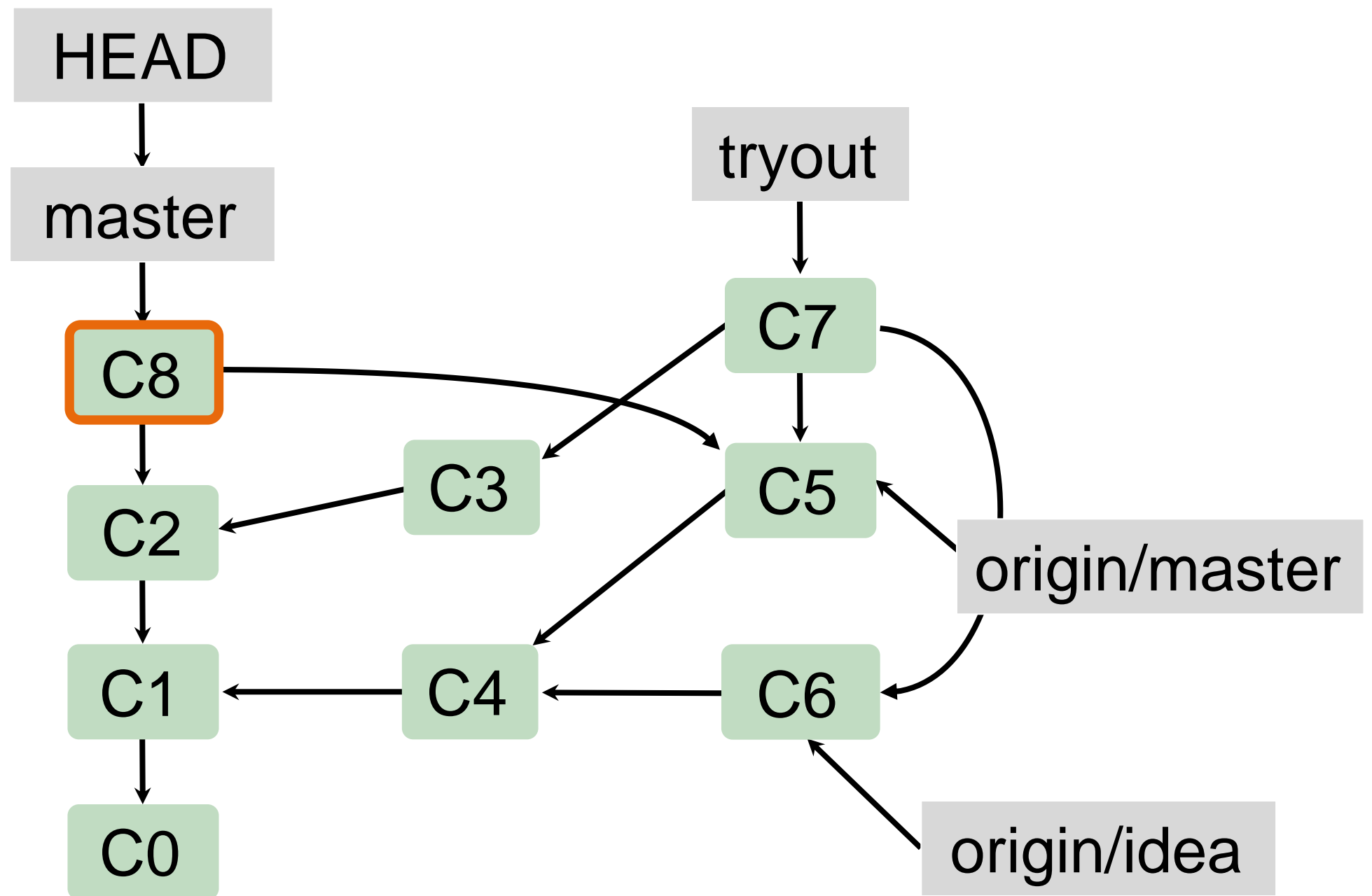


# git checkout master

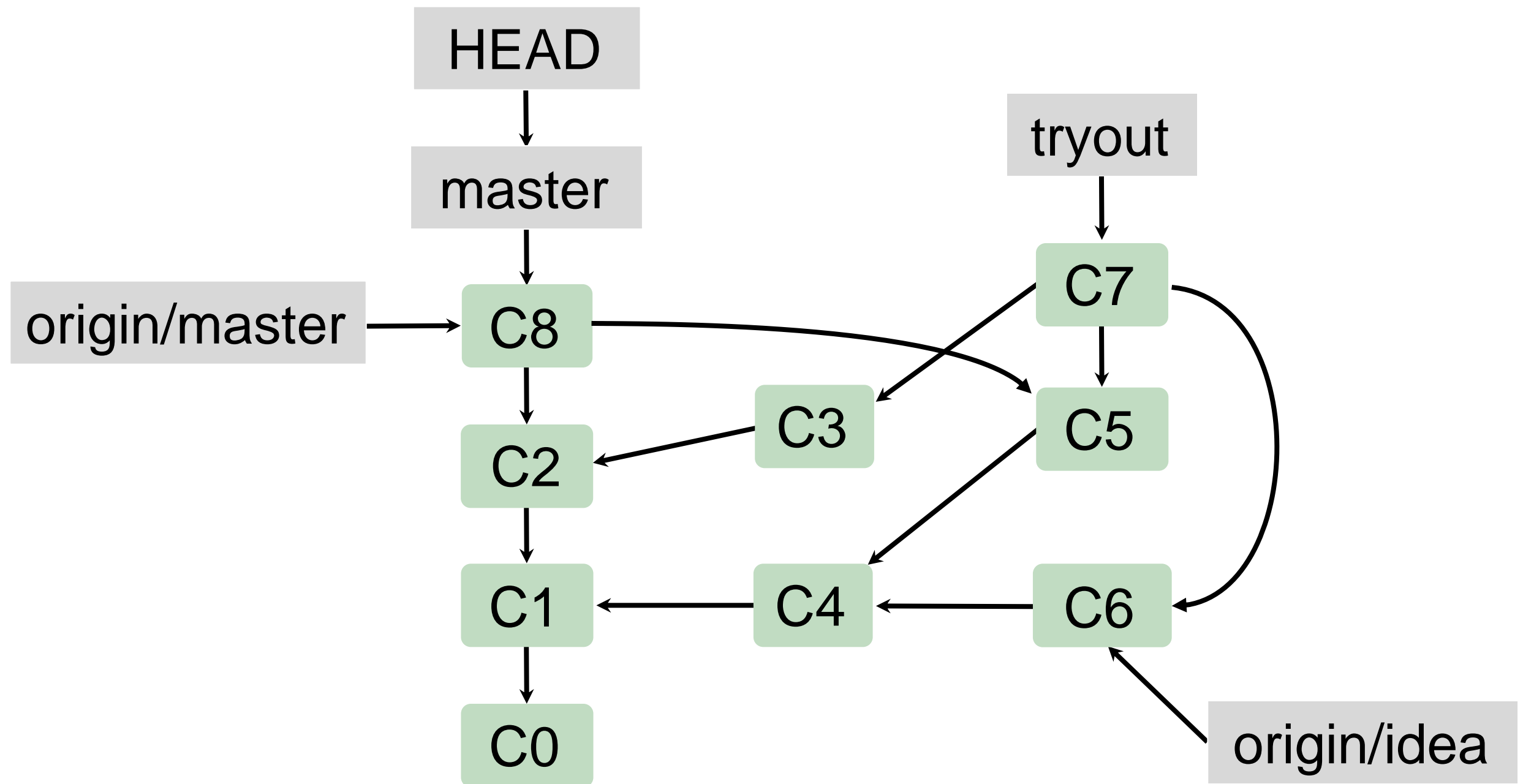




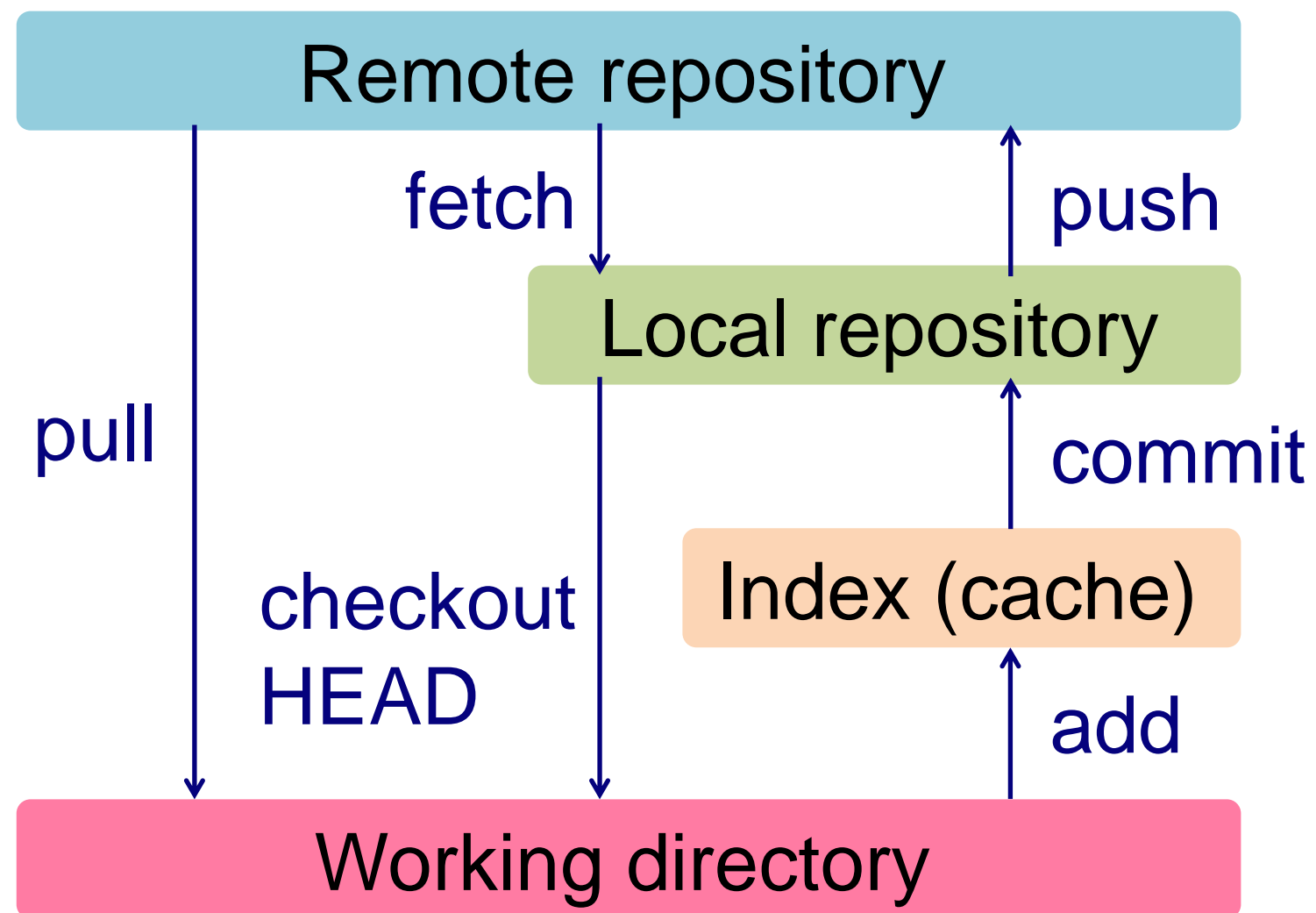
# git merge



**git push**



# Overview git workflow



# Getting started

# First steps

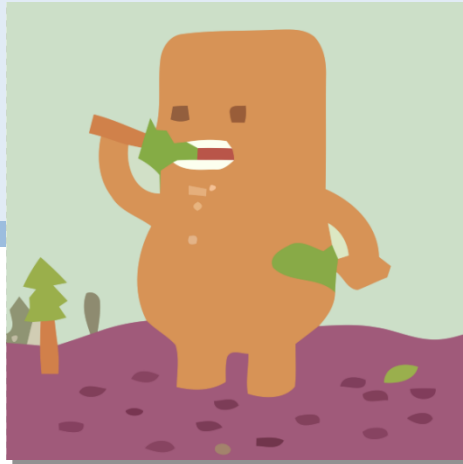
- > Follow instructions on P2 Blog for installation
- > Send your ssh public key to [joel.krebs@students.unibe.ch](mailto:joel.krebs@students.unibe.ch)
- > Create meaningful commits with according messages
- > Hints that make your life easier:
  - Create a .gitignore file
  - Always pull before you push
  - Don't panic when merge conflicts occur

**More to git**

# More to git ...

- > Merging and mergetool
- > Squashing commits when merging
- > Resolving conflicts
- > User authentication with ssh
- > gitx and other graphical tools
- > git configure — remembering your name
- > git remote — multiple remote repos
- > github — an open source public repo
- > ...

# Resources



<http://git-scm.com/>



<http://book.git-scm.com/index.html>

git ready

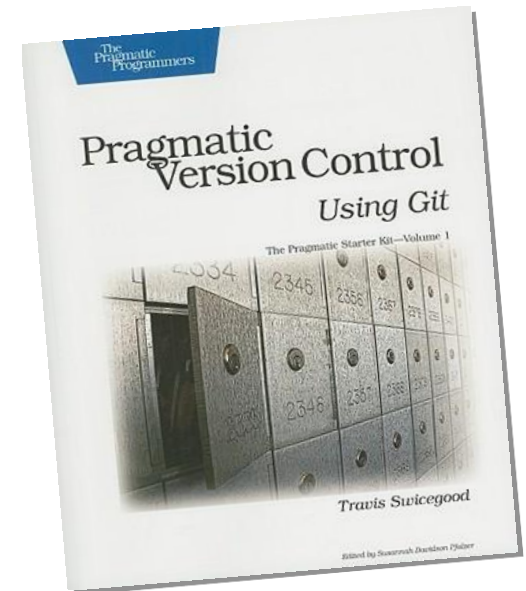
<http://gitready.com/>



<https://github.com/>

Getting Git  
Scott Chacon

<http://www.slideshare.net/chacon/getting-git>



<http://oreilly.com/>





## **Attribution-ShareAlike 3.0**

### **You are free:**

- to copy, distribute, display, and perform the work
- to make derivative works
- to make commercial use of the work

### **Under the following conditions:**



**Attribution.** You must attribute the work in the manner specified by the author or licensor.



**Share Alike.** If you alter, transform, or build upon this work, you may distribute the resulting work only under a license identical to this one.

- For any reuse or distribution, you must make clear to others the license terms of this work.
- Any of these conditions can be waived if you get permission from the copyright holder.

**Your fair use and other rights are in no way affected by the above.**

<http://creativecommons.org/licenses/by-sa/3.0/>