



# Introduction to Git

ikalantzis@ceid.upatras.gr

Patras Linux User Group 2010

(Crappy PDF Edition)

[plug-git.heroku.com](http://plug-git.heroku.com)

# What is Git?

Git is an  
open-source,  
distributed  
version control system



OK, but...

**What is a version  
control system?**

And what makes it distributed?

Revision history of Git (software) – Wikipedia, the free encyclopedia

Article Discussion Read Edit View history Search

## Revision history of Git (software)

From Wikipedia, the free encyclopedia  
View logs for this page

Browse history

From year (and earlier):  From month (and earlier):  Tag filter:   Deleted only

For any version listed below, click on its date to view it. For more help, see [Help:Page history](#) and [Help:Edit summary](#).  
 External tools: [Revision history statistics](#) · [Revision history search](#) · [Number of watchers](#) · [Page view statistics](#)

(cur) = difference from current version, (prev) = difference from preceding version, m = minor edit, → = section edit, ← = automatic edit summary  
 (latest | earliest) View (newer 50 | older 50) (20 | 50 | 100 | 250 | 500)

- (cur | prev)  00:03, 14 November 2010 H3llBot (talk | contribs) **m** (44,852 bytes) *(BOT: Checking dead links; Marked 1 link with {{Dead link}} (Further info: WP:DEADLINK))* (undo)
- (cur | prev)  03:34, 12 November 2010 120.159.129.104 (talk) (44,807 bytes) *(Undid revision 396230259 by XLinkBot - Not vandalism. Earlier linked article contains dated opinion piece, new link contains rebuttal to balance views and address issues.)* (undo)
- (cur | prev)  01:37, 12 November 2010 Jpatokal (talk | contribs) (44,628 bytes) *(←Name: remove visible {{sic}}, there's nothing wrong with it)* (undo)
- (cur | prev)  00:51, 12 November 2010 XLinkBot (talk | contribs) (44,636 bytes) *(BOT--Reverting link addition(s) by 120.159.129.104 to revision 395584748 (http://unspecified.wordpress.com/2010/03/26/why-git-aint-better-than-x/))* (undo)
- (cur | prev)  00:47, 12 November 2010 120.159.129.104 (talk) (44,815 bytes) *(Add rebuttal for 'git better than x')* (undo)
- (cur | prev)  18:29, 8 November 2010 142.103.111.195 (talk) (44,636 bytes) *(←Characteristics)* (undo)
- (cur | prev)  19:22, 7 November 2010 129.67.45.223 (talk) (44,634 bytes) *(←External links)* (undo)
- (cur | prev)  15:59, 7 November 2010 24.6.158.106 (talk) (44,544 bytes) *(Tridge also wrote 15,000 lines of C intending to create a free BitKeeper clone (http://sourceforge.net/projects/sourcepuller). The whole "I just teinnetted and typed help" story is just not true.)* (undo)
- (cur | prev)  23:45, 5 November 2010 76.98.33.248 (talk) (44,990 bytes) *(←Projects using Git)* (undo)
- (cur | prev)  09:29, 5 November 2010 147.102.232.249 (talk) (44,929 bytes) *(← See also: fixing wikilink)* (undo)
- (cur | prev)  19:06, 29 October 2010 137.112.104.92 (talk) (44,929 bytes) *(←Projects using Git)* (undo)
- (cur | prev)  15:11, 29 October 2010 Ebrambot (talk | contribs) **m** (44,851 bytes) *(robot Adding: he:וּא)* (undo)
- (cur | prev)  15:49, 27 October 2010 Tedyu002 (talk | contribs) **m** (44,837 bytes) *(←Projects using Git: Add PostgreSQL)* (undo)
- (cur | prev)  15:03, 27 October 2010 Alisha.4m (talk | contribs) (44,746 bytes) (undo)

Revision history

**It's all about how you  
manage changes.**

Git is a lot more than that.

# Why?

Keep track of everything,  
revert back if necessary

Don't be afraid to mess with code

Seamless collaboration

# Fully distributed

(Almost) everything is local.

Every clone is a backup.

Work offline.

(In a way, it's like having a local copy of  
Wikipedia)

# First steps

# Installation

```
$ sudo aptitude install git-core git-doc
```

```
$ yum install git-core
```

# Initial Configuration

```
$ git config --global user.name "John Doe"  
$ git config --global user.email "john@exc  
$ git config --global color.ui true
```

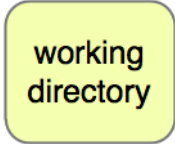
/etc/gitconfig

~/.gitconfig

.git/config

```
$ cd path/to/repos  
$ mkdir hello  
$ cd hello
```

# Working directory



working  
directory

# git init

```
$ git init # creates .git dir  
$ ls -la
```

Write some code

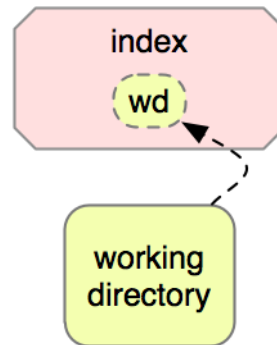
```
$ vim hello.c  
$ vim world.c
```

# git add

Add files to index

```
$ git add hello.c  
$ git add world.c
```

# The staging area



# git status

Status of working dir and index

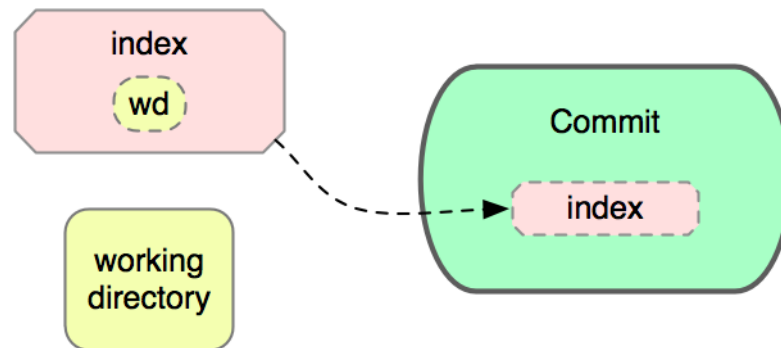
# .gitignore

Files you don't want in your repo

```
*.o  
*.pyc  
*.log  
local_settings.py  
SECRET.txt  
testing/
```

git commit

A commit is a snapshot of the index  
**NOT THE WORKING DIRECTORY**

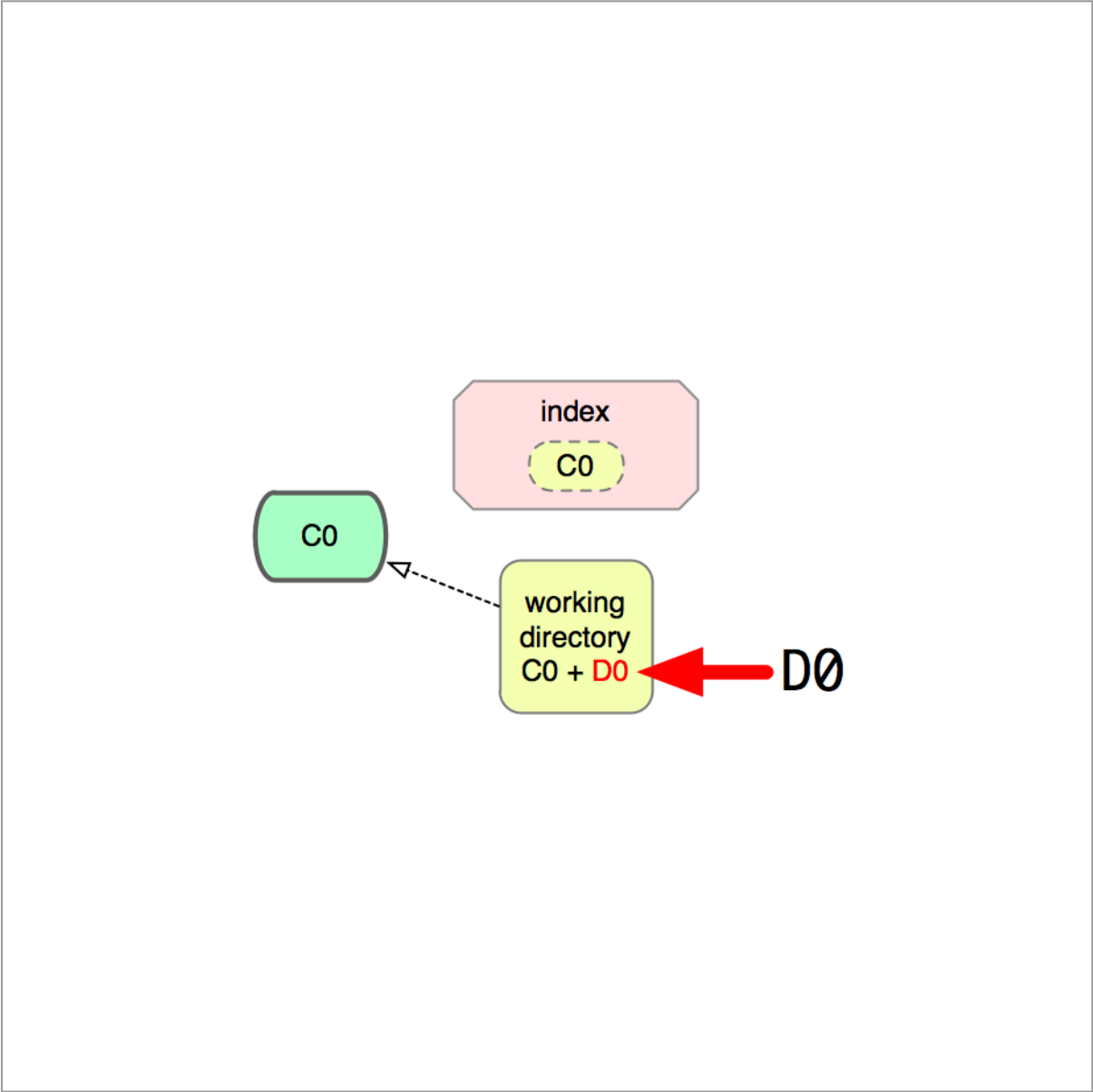


# git log

Show a list of commits

Now, make some changes

```
$ vim hello.c
```



Review the changed files

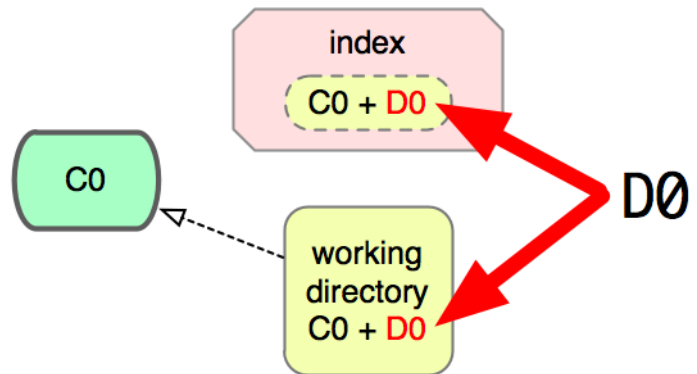
```
$ git status
```

# git diff

Differences between index and working dir

`git add -p`

Interactively add changed chunks



git diff

shows nothing!

# git diff --staged

Diff between commit and index

# git commit -m

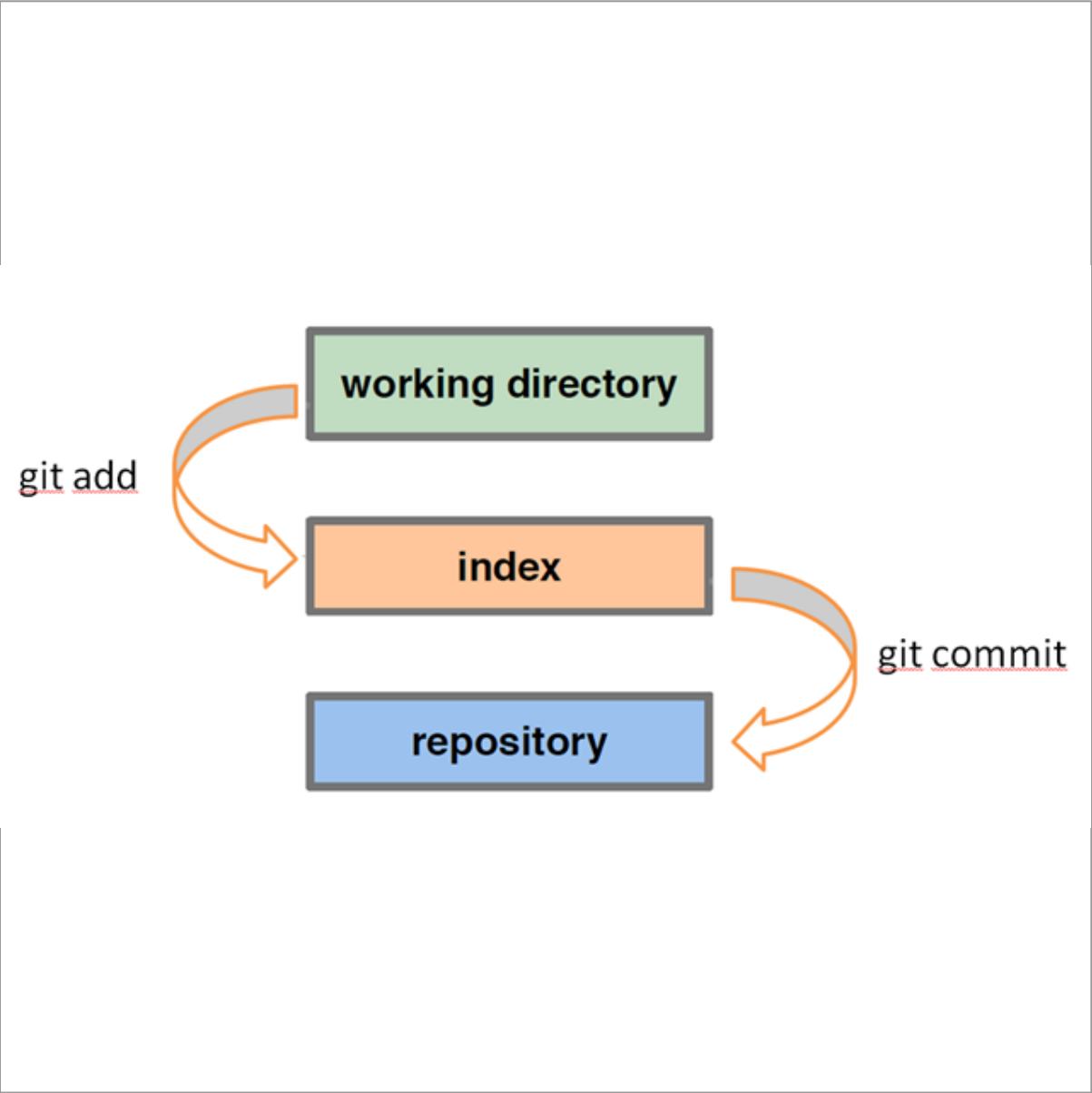
Create a commit with a message

```
$ git commit -m "2nd commit."
```

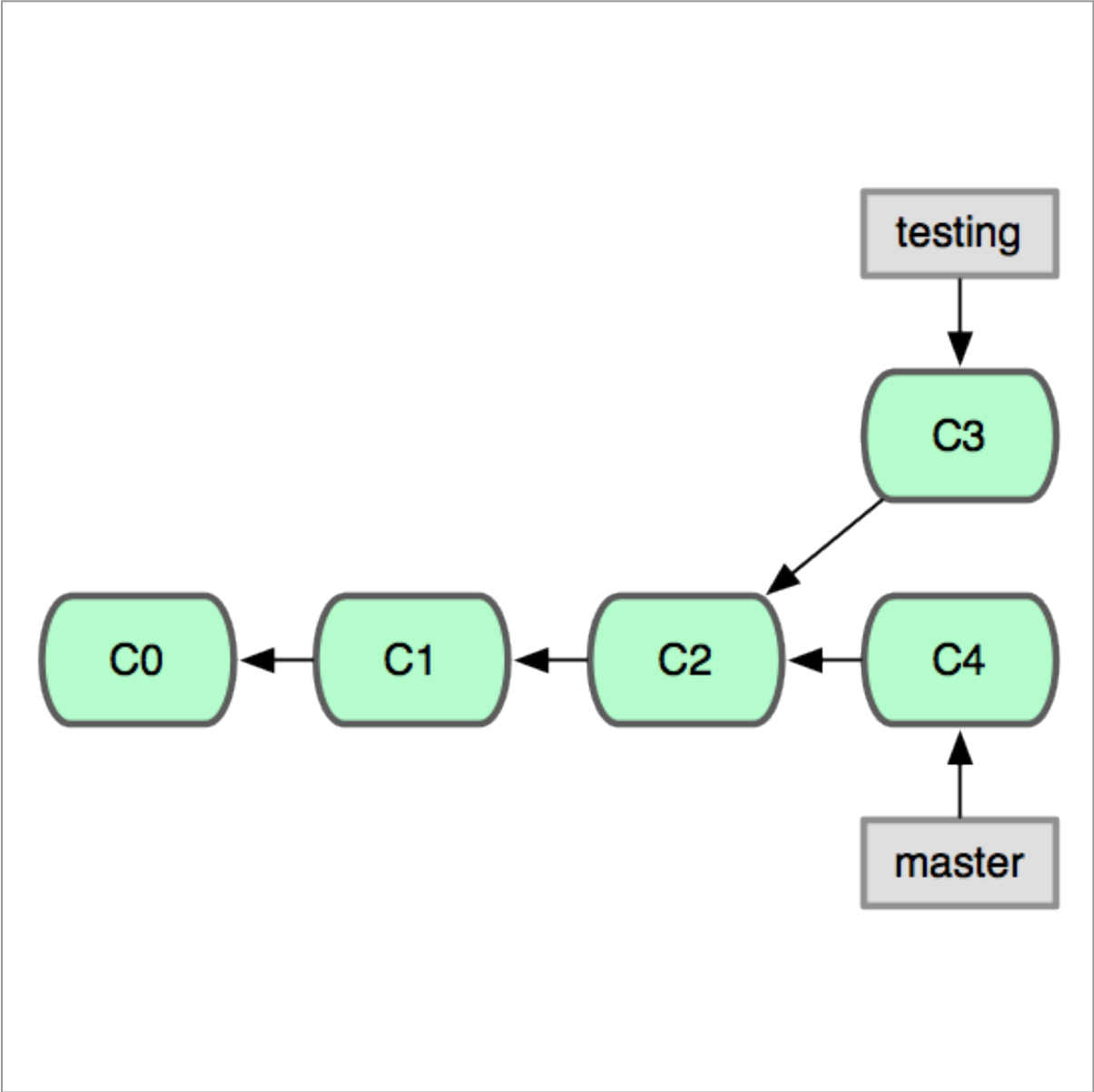
Commit history is  
**immutable!**

# Basic workflow

1. Make changes to working dir
2. Stage those changes to the index
3. Commit the current state of the index



# Branching and Merging



# Branching is cool

Try out an idea, see if it works, then  
decide

Isolate work units

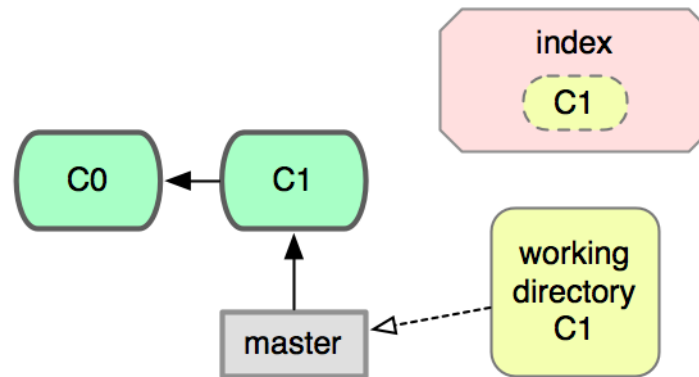
# git branch

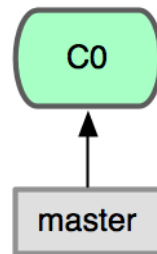
List all local branches

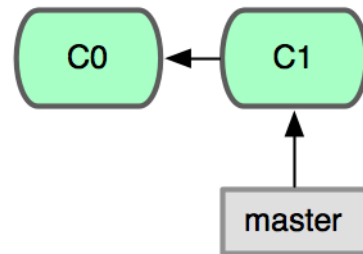
The default branch is named

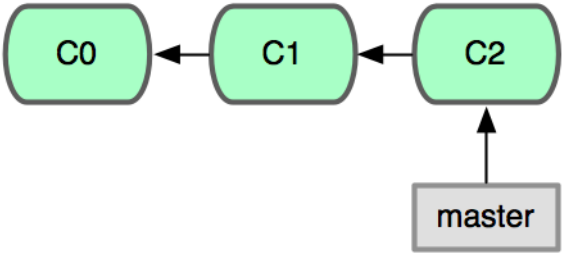
**master**

Branches are movable pointers to commits







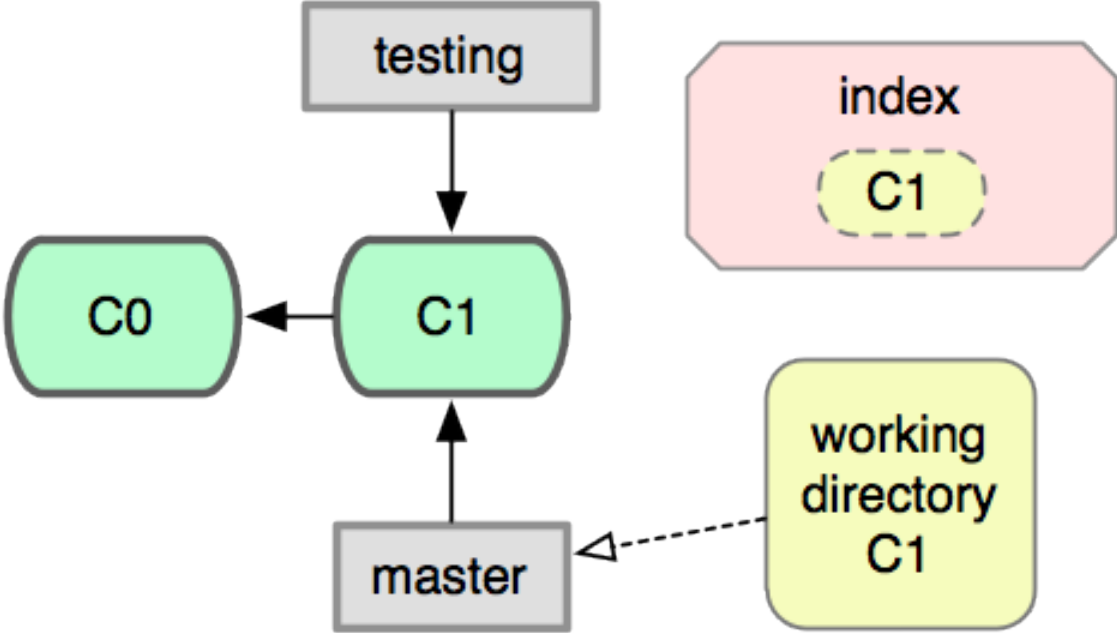


# git branch

Create a new branch from  
the current commit

```
$ git branch testing
```

Working dir does **not** change

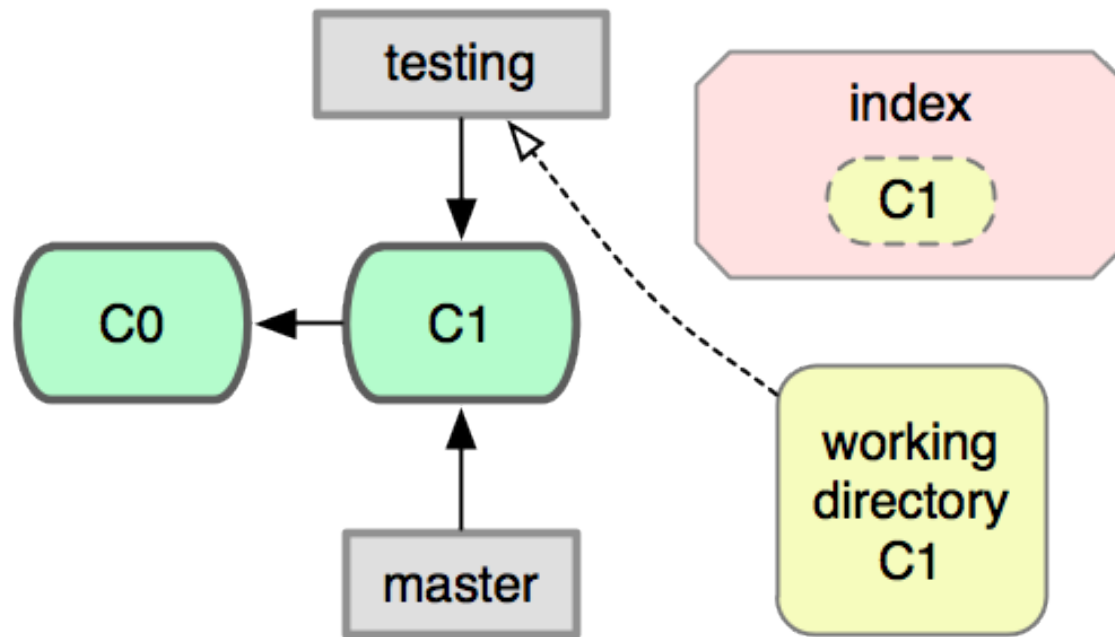


# git checkout

Switch working dir to the given branch

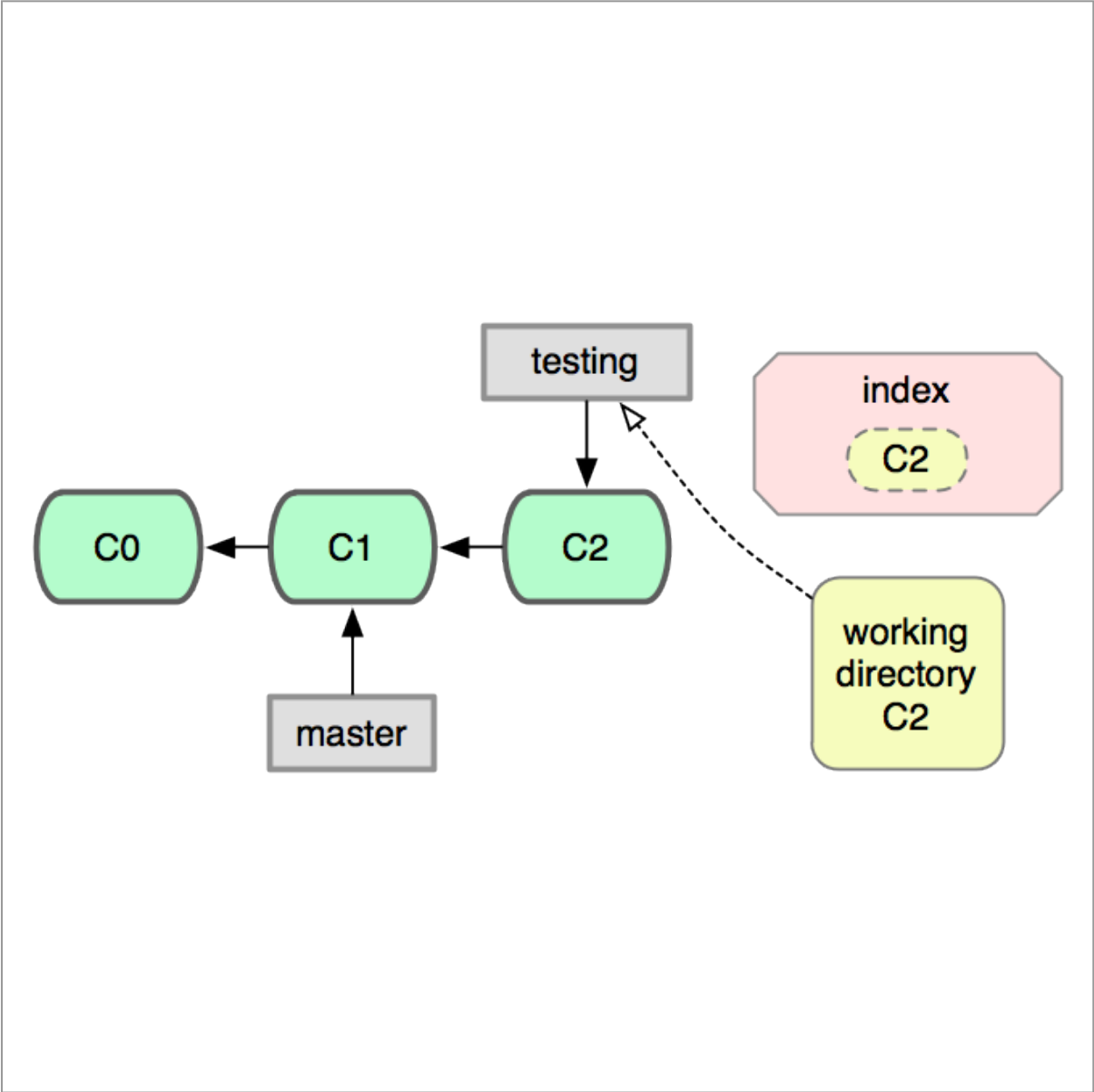
```
$ git checkout testing
```

## Current status of working dir



While on this branch,  
make a change and commit it.

```
$ vim hello.c  
$ git add hello.c  
$ git commit -m 'Just a test.'
```



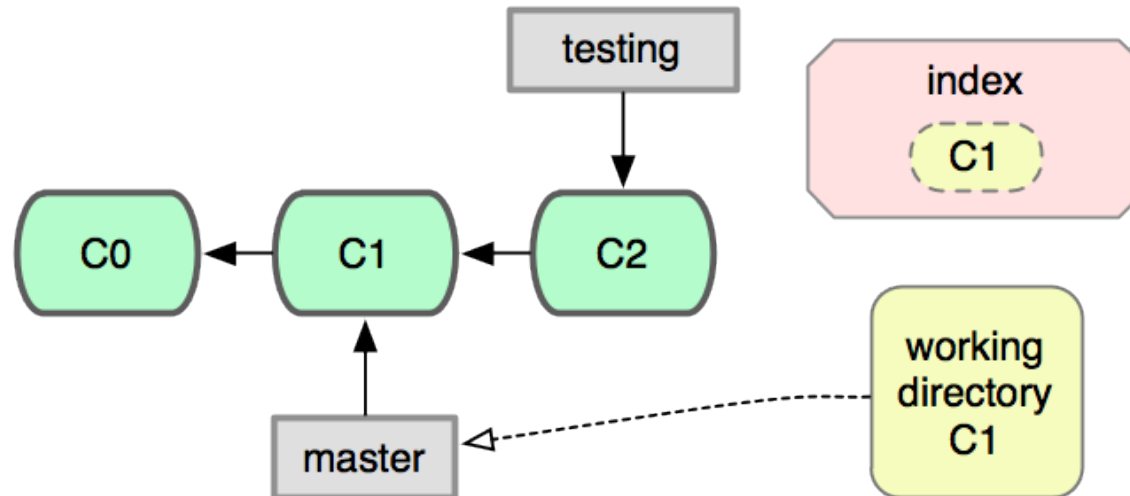
# git branch -v

List branches and the commits they  
point to

Now back to the **master** branch.

```
$ git checkout master
```

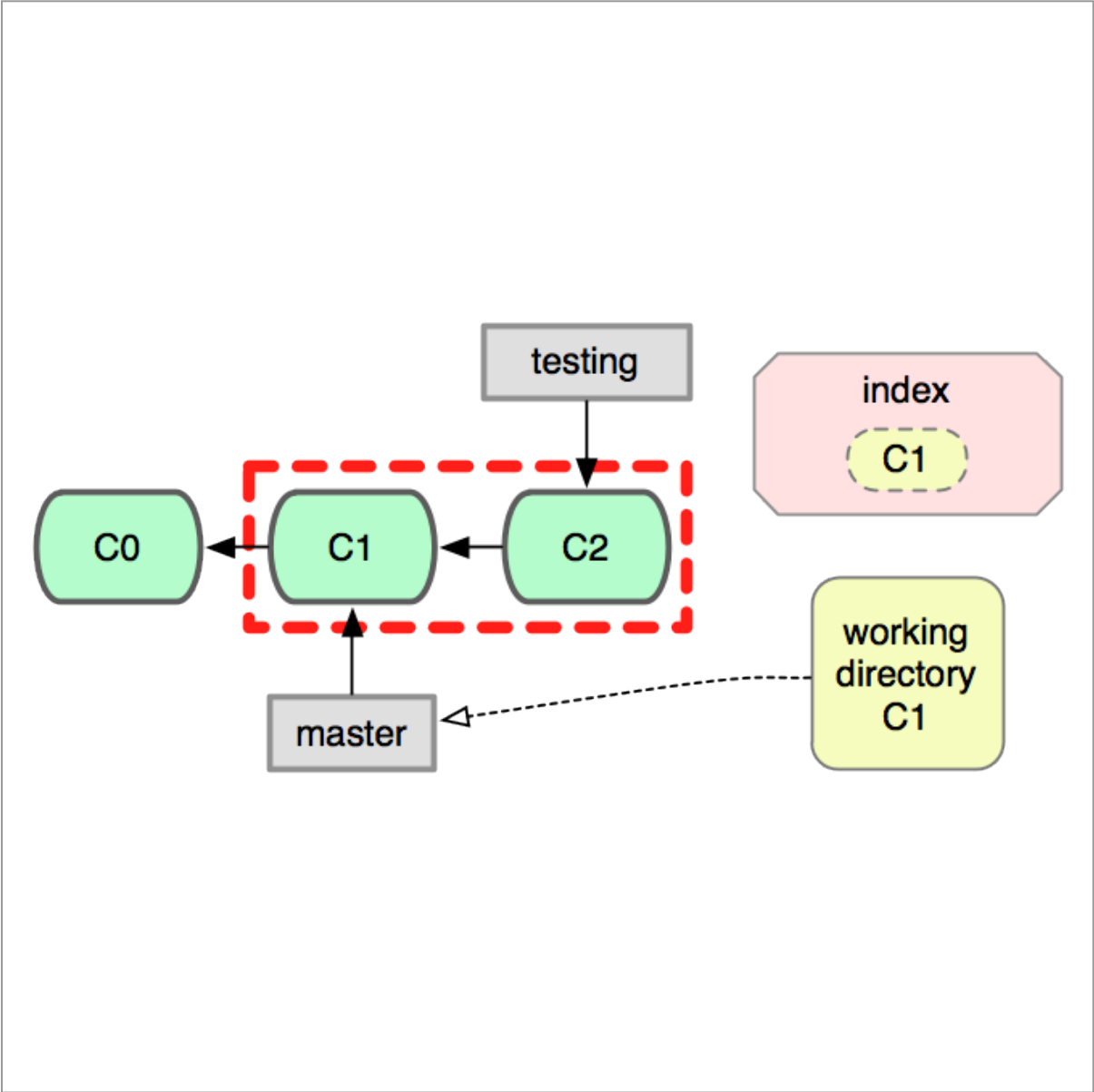
Working dir has been changed to **master**.



# git diff

Diff between two commits

```
$ git diff master testing
```

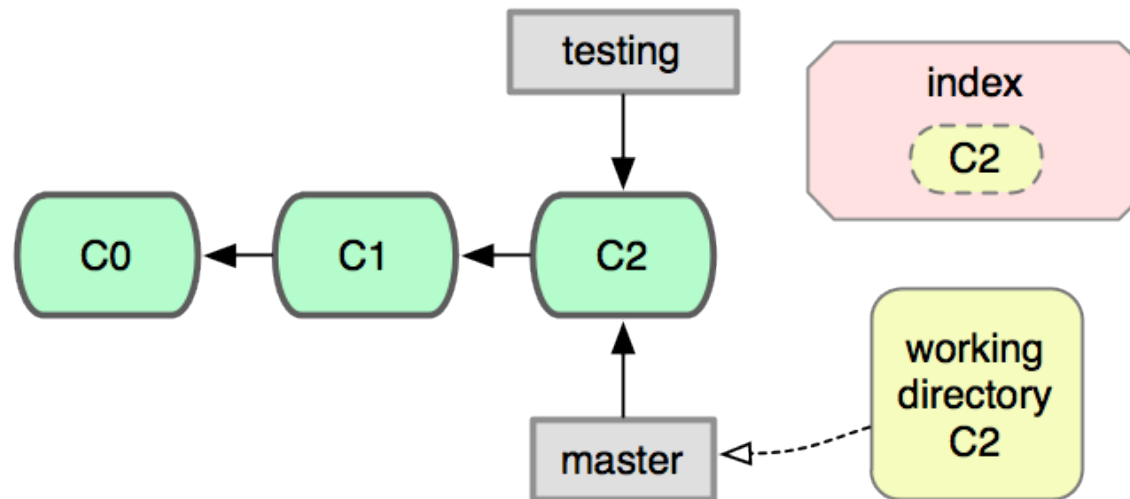


# git merge

Merge given commit into the current  
branch

```
$ git merge testing
```

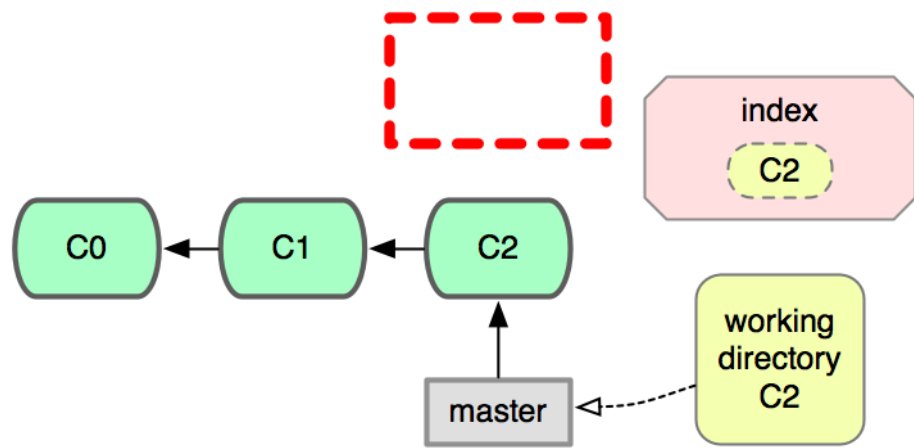
Now both branches point at the same commit



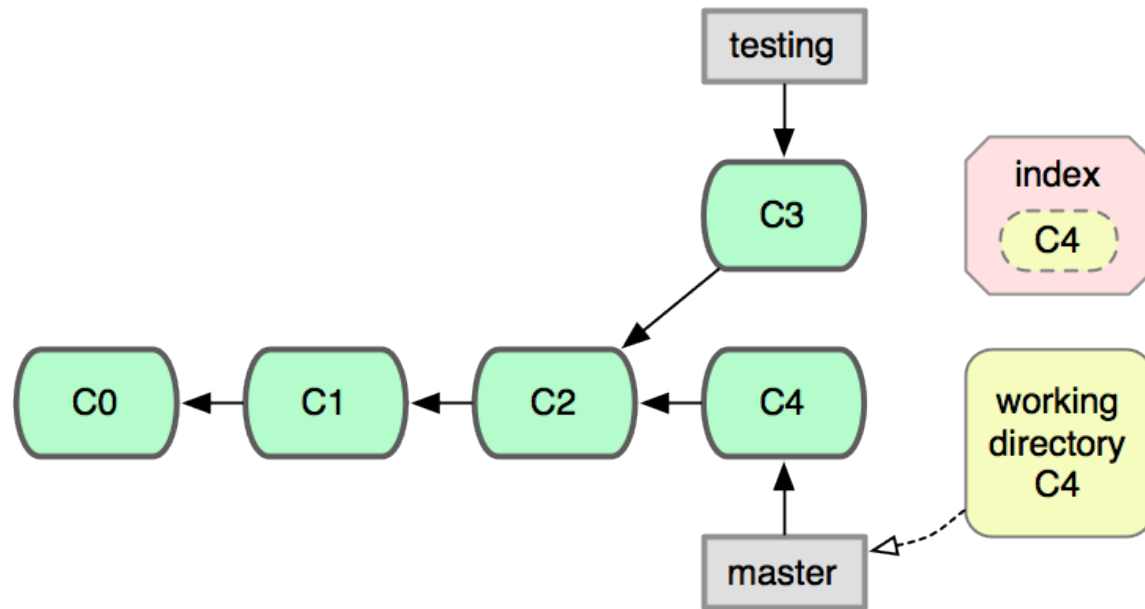
# git branch -d

Delete given branch

```
$ git branch -d testing
```



What if both branches have forward commits?



Merge `testing` into `master`

```
$ git merge testing
```



This is known as a  
**recursive merge**  
and uses a 3-way merge strategy

merge strategies: man git-merge

## References

b4f2e412d087bae0a523fe6ea40fcad30f

b4f2e41

master

HEAD

HEAD^

**HEAD** is a reference to the latest commit  
of your current checkout.

HEAD^, HEAD^^, HEAD~3 ...

Merge conflicts !

Modify the same line in two branches  
and attempt to merge.

```
$ git merge testing
```

Clean files are staged.  
Files with conflicts are left unstaged.

# Resolving conflicts

Edit files

`git add` to mark as properly merged

Commit

# Distributed workflow

Back to your repos

```
$ cd ..
```

# git clone

Make a copy

```
$ git clone hello helloclone  
$ cd helloclone
```

Works with remote repos too

```
$ git clone https://github.com/ask/celery.g
```

SSH

http://

git://

# git remote

List all remotes

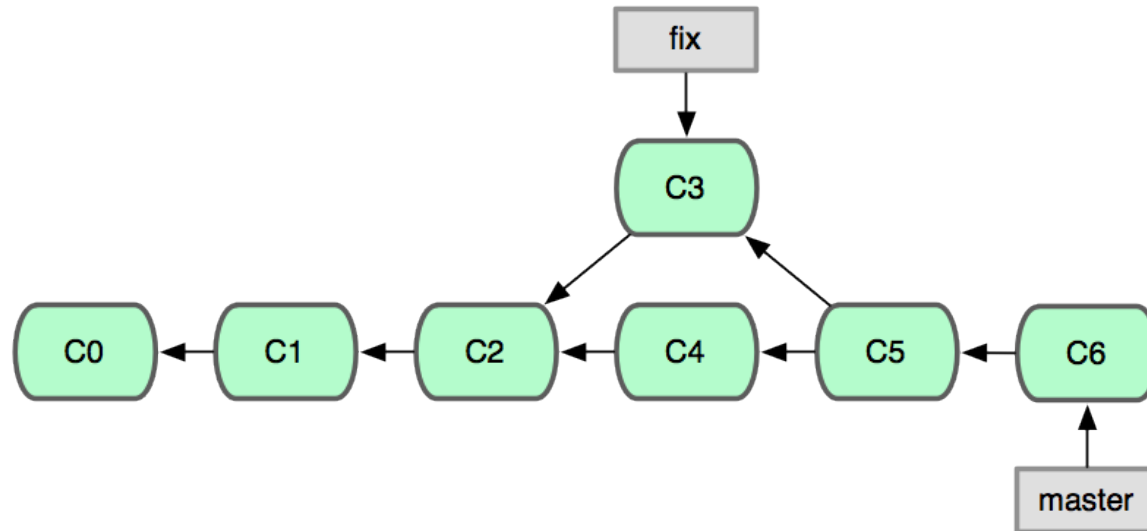
```
$ git remote -v
```

# git branch -a

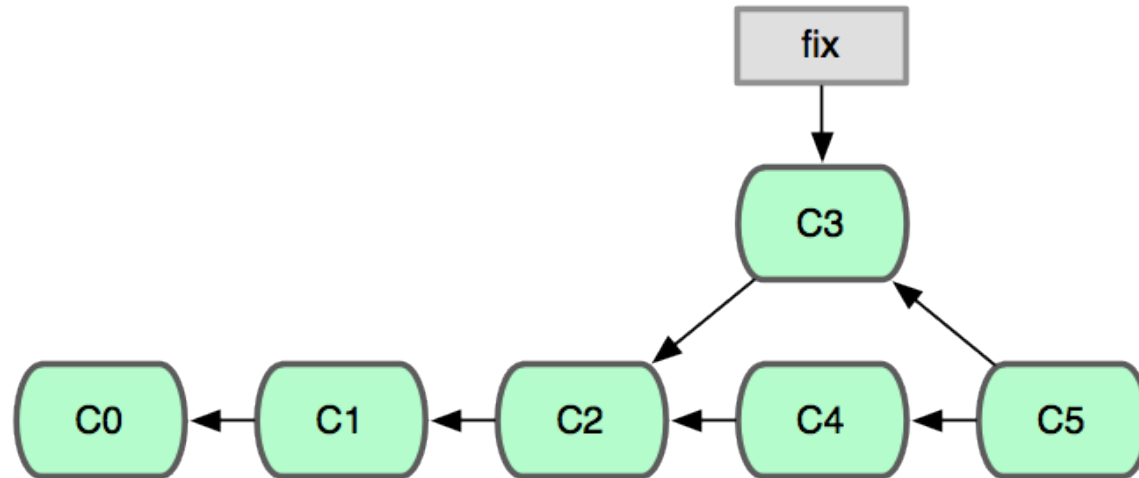
List all branches, local and remote

Assume that upstream makes a commit  
on the **master** branch

Current state of **upstream**



Current state of **local repo**

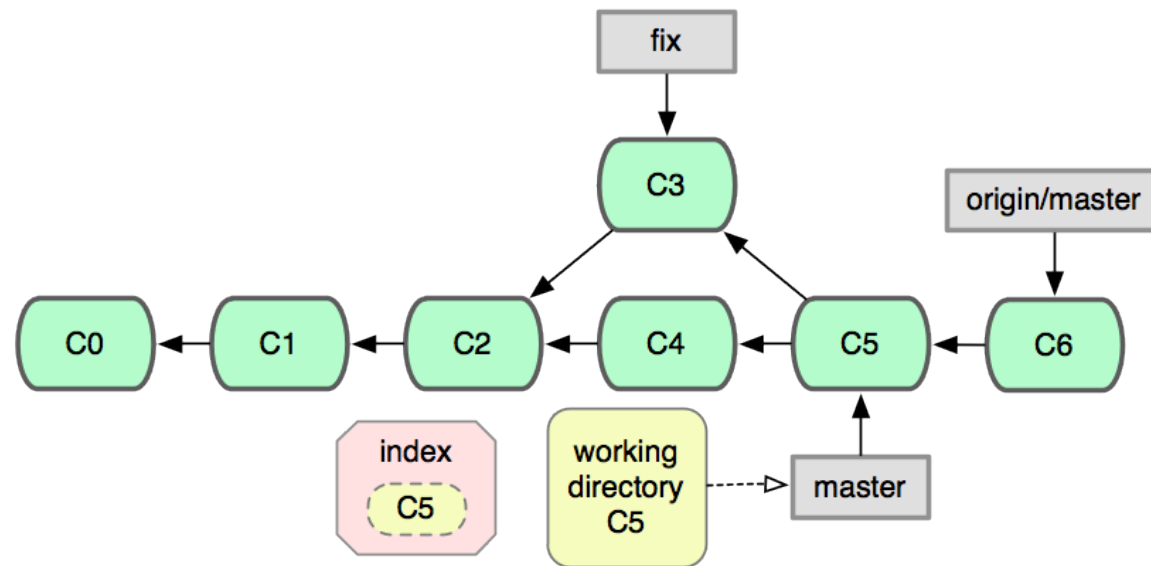


# git fetch

Fetch commits from the given remote

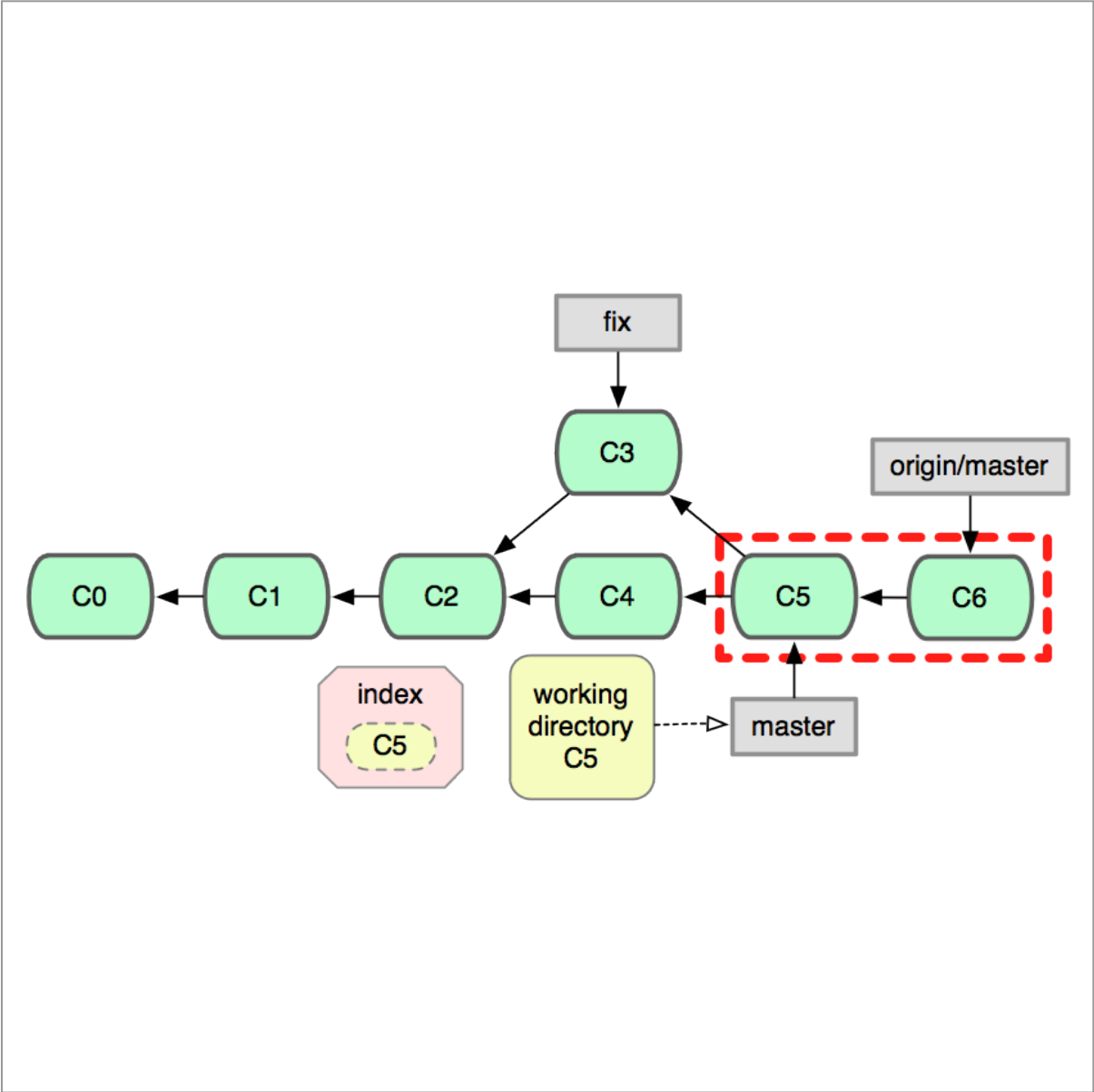
```
$ git fetch origin
```

## Local repo after git fetch



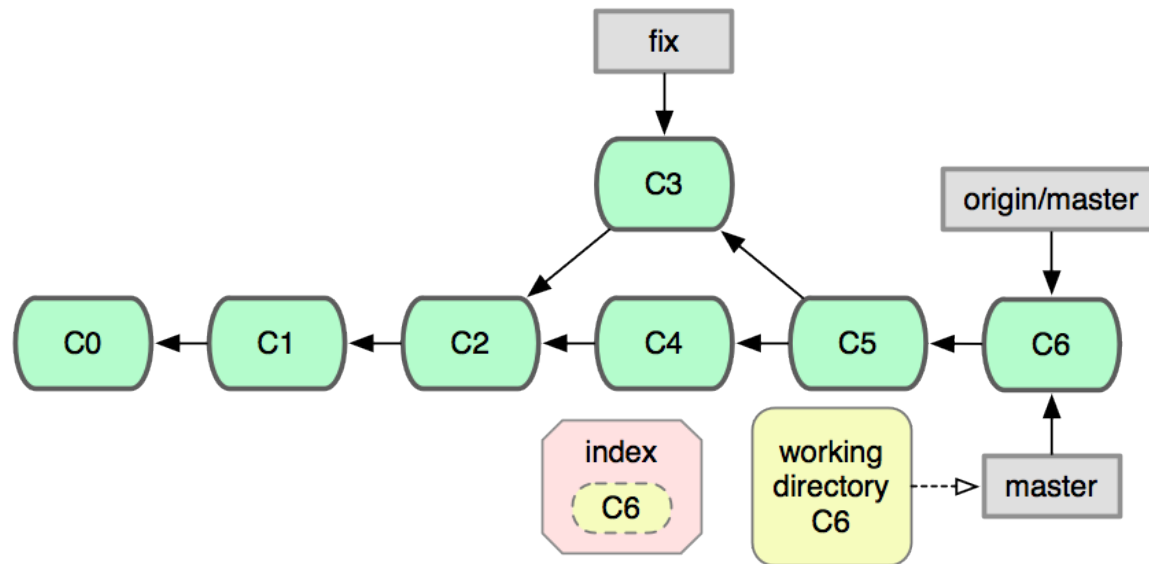
Review the changes

```
$ git diff head origin/master
```



Merge the upstream changes  
into your local **master** branch  
`$ git merge origin/master`

## Status after merging



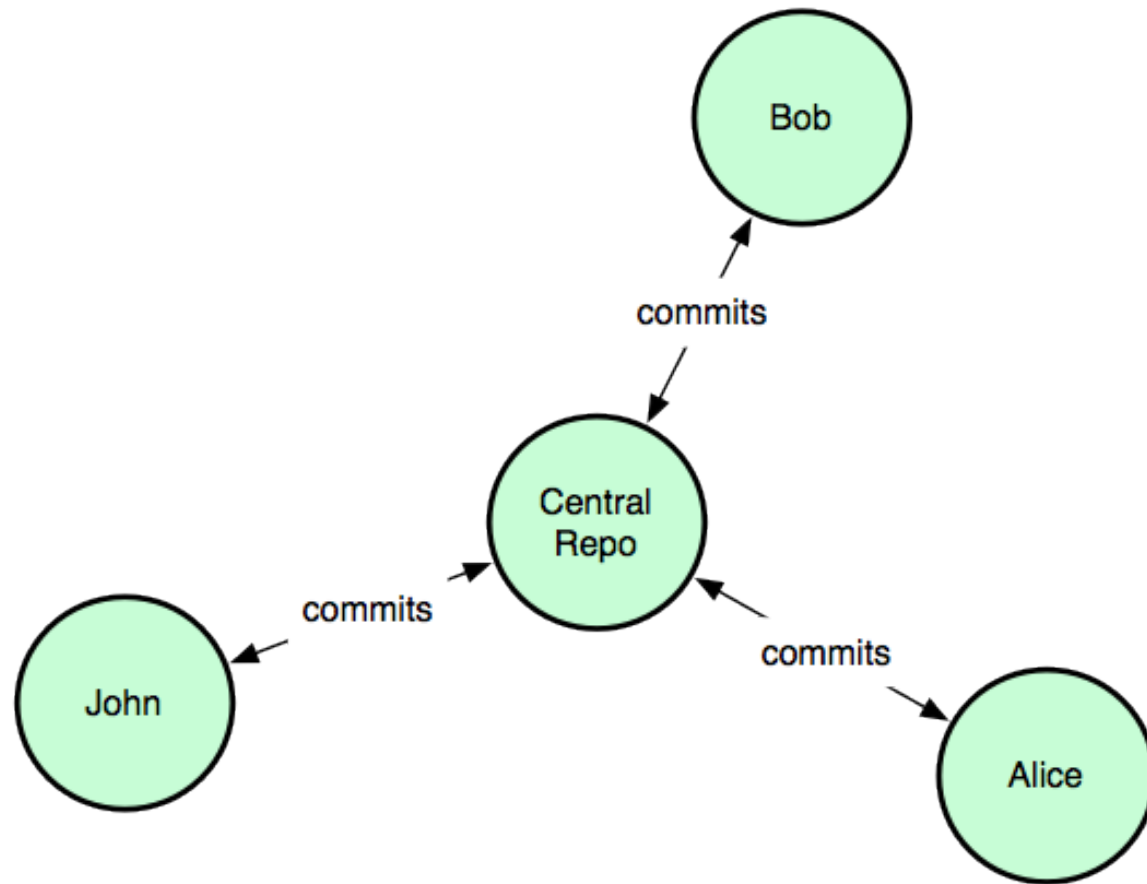
What if you want to share **your** changes?

# git push

Push **your** commits to a remote

```
$ git push origin master
```

## Moving commits around between repositories



# Github



[git.ceid.upatras.gr](https://git.ceid.upatras.gr)

Where do I go from  
here?

[learn.github.com](https://learn.github.com)

[gitref.com](https://gitref.com)

[gitready.com](https://gitready.com)

[progit.com](https://progit.com)

# Thanks!

That was a lot of slides!

# Questions ?

[plug-git.heroku.com](https://plug-git.heroku.com)