Git and GitHub

Dr David Glance

Notices

- Some content for this presentation was obtained from:
  - https://try.github.io
  - https://en.wikipedia.org/wiki/Version_control
  - Adrien Thebo http://adrienthebo.github.io/vcs_presentation/ (CC-BY-NC-SA-3.0)

What is Source Code Control?

- Provides a number of solutions to various problems:
  - Allows for the storage of source code
    - Problem: source code being on a specific developer’s machine
  - Allows for the distribution of source code
    - Problem: source code needs to be shared with development team and also public
  - Allows versioning and branches of source code
    - Problem: a bug needs to be fixed in existing version of product whilst development continues on next version
  - Allows for multiple people to collaborate on code
    - Problem: One developer’s changes overwriting another
  - Allows for formal review of code and merging
    - Problem: Bad code accepted that “breaks the build” or “breaks the product”

What does Source Code Control need to do?

- Manage a collection of files as a repository
- Keep metadata and history of who did what to what files
  - Important for auditing and *blame*
- Be able to detect differences in files and potentially merge different changes
  - In practice this is difficult
- Allow developer to switch easily between various versions (commits) on the same or other branches

Source code control system

- SCCS: First code version control system
  - Bell Labs in 1972
- Revision Control System (RCS)
  - 1982
  - Changes of files stored as “diffs”
  - Used exclusive locks on single files
- Centralised systems
  - CVS Concurrent Version System
    - Check out entire project
    - Support for multiple users
  - SVN Subversion

Then there was Git…

- Created by Linus Torvalds in 2005 for Linux Kernel development
- They were using BitKeeper but this was proprietary
- The name came from Linus Torvalds:
  “I’m an egotistical bastard, and I name all my projects after myself. First ‘Linux’, now ‘git’.”
- Really gained popularity through GitHub, web-based hosting service for Git repositories
- GitHub has recently been bought by Microsoft – nobody is sure if this is a good thing or not

Git Basics

- Source is kept in a “repository” or “repo”
- Copying repo from GitHub or other remote server to local disk is called “cloning”
- Updating local files from repo involves a “pull” from remote repo
- Updating remote repo from working directory involves a “push”
- Main branch of repo is called “master” branch
- When a branch is merged into master the process is called a “merge” or this is done using a “Pull Request”
- PR is a way of getting code reviewed before it is merged

Getting started

- You can use a GUI (GitHub app or IDE like PyCharm, XCode, Visual Studio) but easier if you learn how to use command line
- Creating a git repository:

```
$ cd ~/DevProjects
$ mkdir myproject
$ cd myproject
$ git init
```

What did that do?

- Created a directory called .git

```
$ ls
total 24
-rw-r--r--   1 dglance  staff   23 30 Jul 11:15 HEAD
drwxr-xr-x   2 dglance  staff   64 30 Jul 11:15 branches
-rw-r--r--   1 dglance  staff  137 30 Jul 11:15 config
-rw-r--r--   1 dglance  staff   73 30 Jul 11:15 description
drwxr-xr-x  12 dglance  staff  384 30 Jul 11:15 hooks
drwxr-xr-x   3 dglance  staff   96 30 Jul 11:15 info
drwxr-xr-x   4 dglance  staff  128 30 Jul 11:15 objects
drwxr-xr-x   4 dglance  staff  128 30 Jul 11:15 refs
```

Key file is exclude

- info/exclude allows you to specify which files you want git to ignore (also done with a file called .gitignore)

```
$ cat info/exclude
# git ls-files --others --exclude-from=.git/info/exclude
# Lines that start with '#' are comments.
# For a project mostly in C, the following would be a good set of
# exclude patterns (uncomment them if you want to use them):
# *.[oa]
# *~
```

Add a file

- Create a file and check status

```
$ touch afile.txt
$ ls
afile.txt
$ git status
On branch master

No commits yet


Untracked files:
  (use "git add <file>..." to include in what will be committed)

afile.txt


nothing added to commit but untracked files present (use "git add" to track)
```

Add for commit

- Do as it suggests – add the file

```
$ git add afile.txt
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

new file:   afile.txt
```

And commit…

- Commit the file

```
$ git commit –m “Committing initial file”
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 afile.txt
```

Meanwhile, we switch to GitHub

- We will now create a repository in GitHub
- And then push the files to the "master" branch
- And check they end up on GitHub

```
$ git remote add origin https://github.com/uwacsp/gitlecture1.git
$ git push –u origin master
Counting objects: 3, done.
Writing objects: 100% (3/3), 231 bytes | 231.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/uwacsp/gitlecture1.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

Making changes

- Once the file is changed you can repeat the last process of commit and push, or add any new files
- Demo
- You can delete files using git rm
  - *DANGER* Be very careful with this – you can recover from a previous version but not changes you made since the last commit

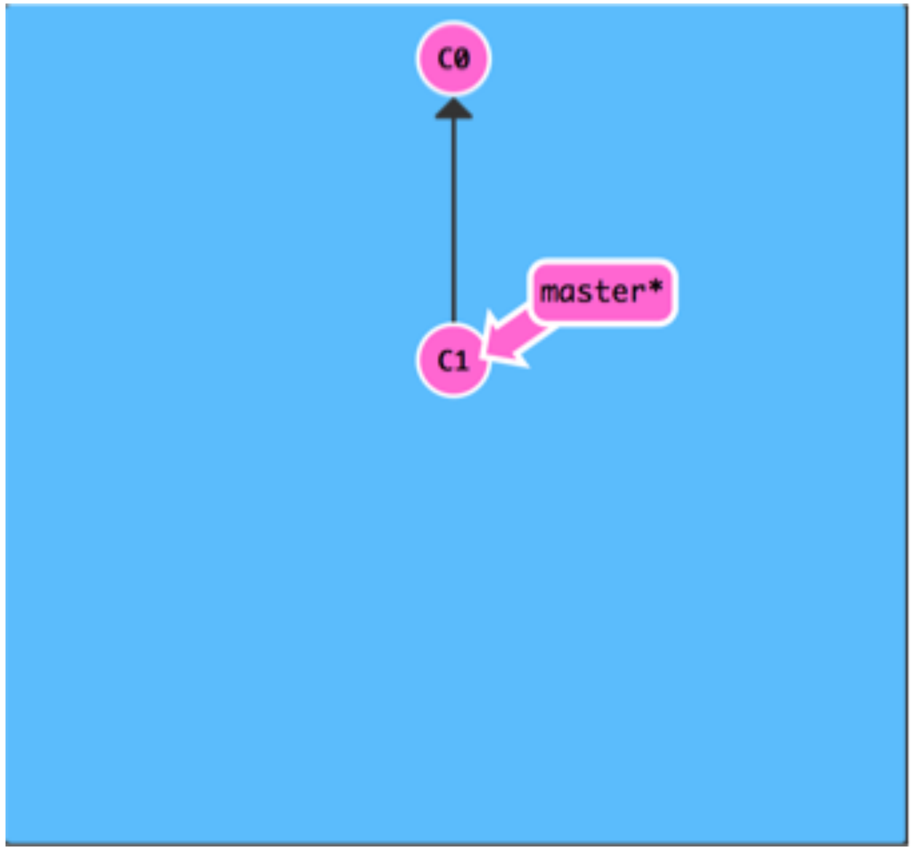
Getting other people’s updates

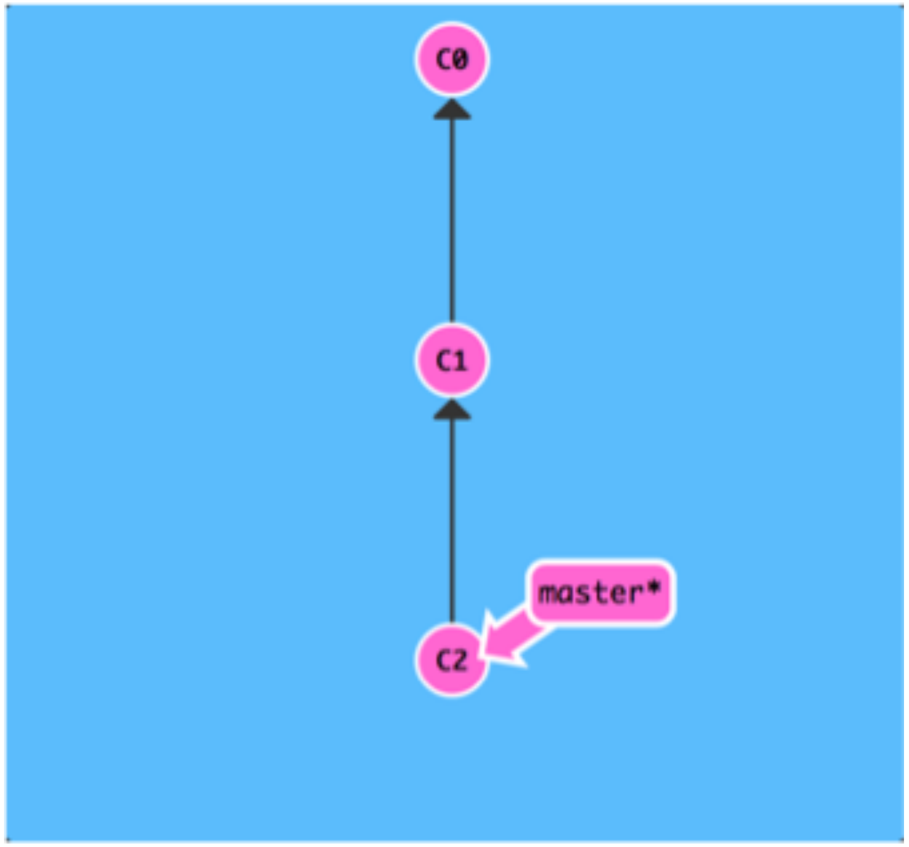
- If the repo changes because another developer has changed a file, you have to get that update before changing it yourself
- People still shouldn’t be generally working on the same files
- Can use branches

Using branches

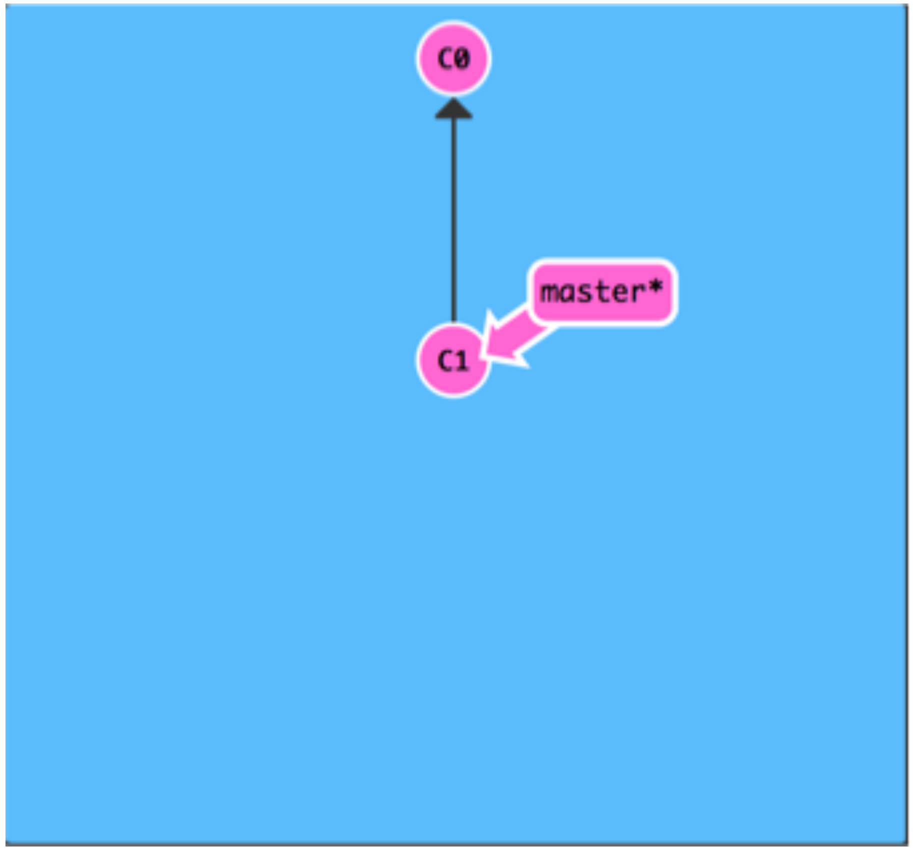
- Create a new branch with:
  - git checkout –b anewbranch
- Confirm that you are on that branch
  - git branch
- Now, any changes made on the branch will not be seen by anyone working on master and vice-versa
- Once the branch is ready to merge back into master, we can merge using:
  - first checkout master
  - git merge <branch to merge>

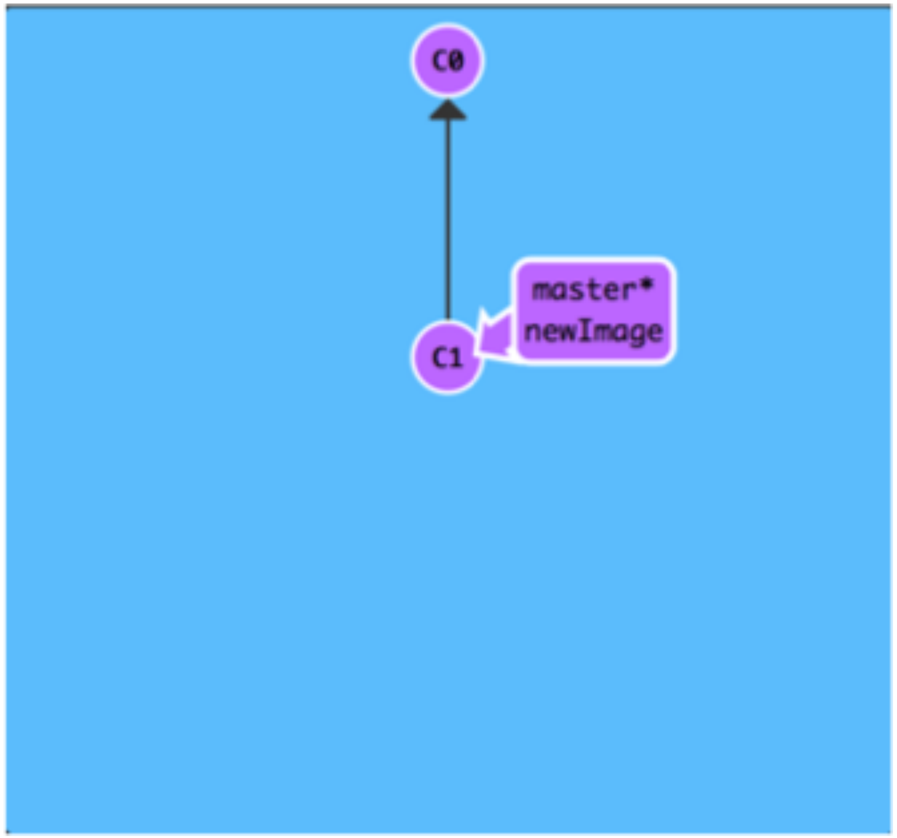
git commit



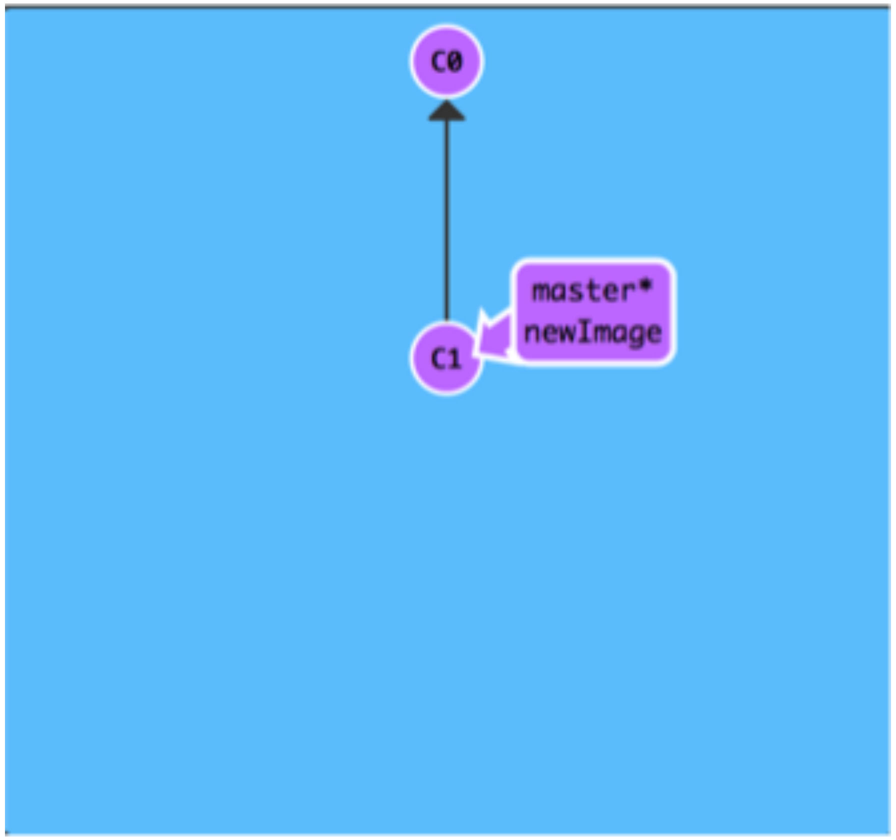


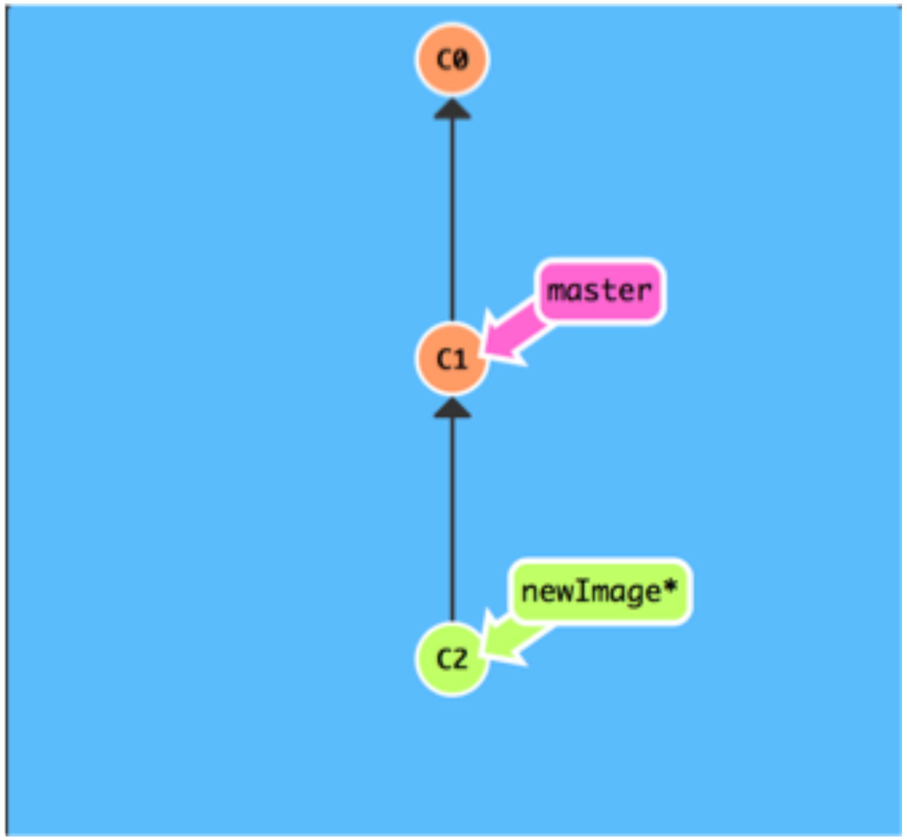
git branch newImage





git checkout newImage; git commit

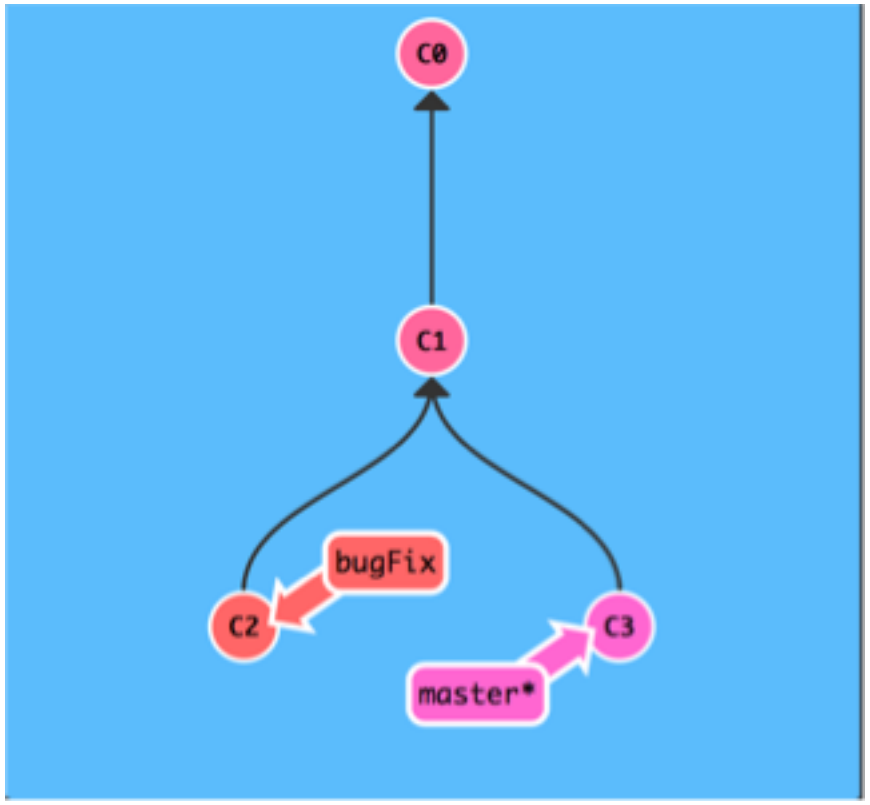


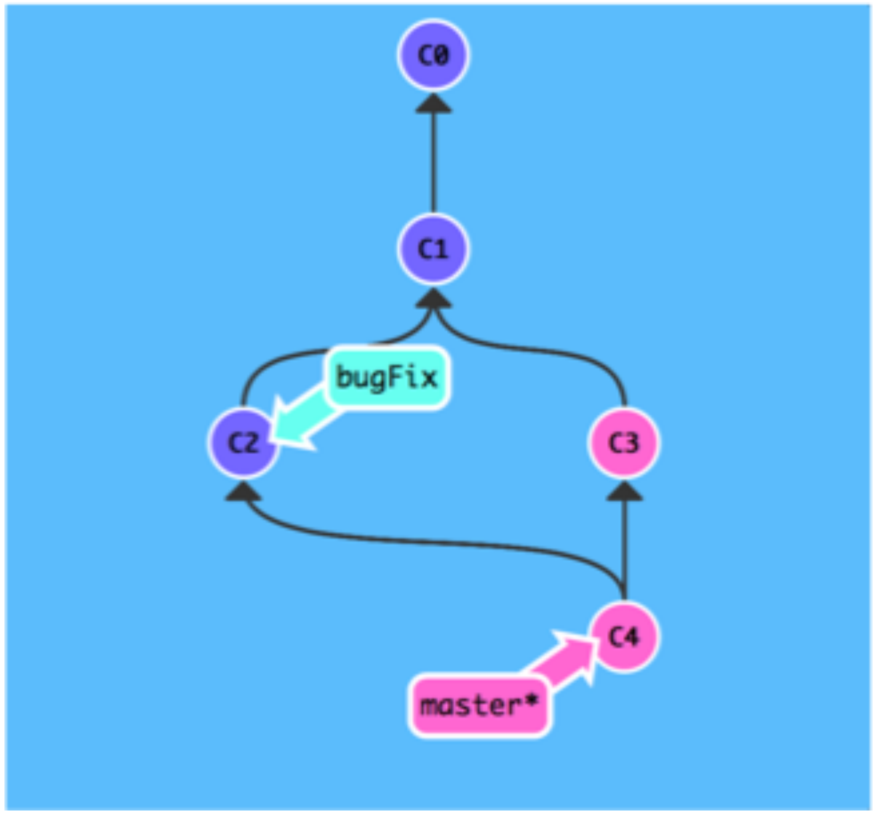


git checkout –b newImage

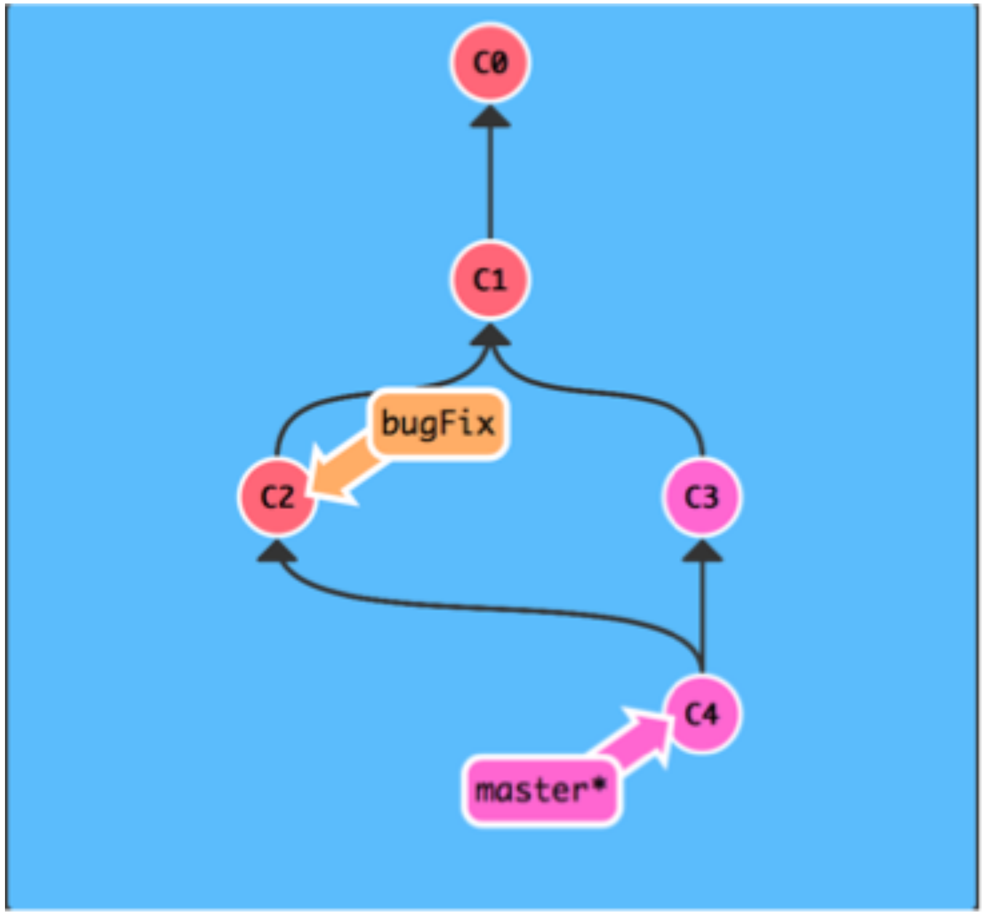
- same as 2 commands:
  - git branch newImage
  - git checkout newImage

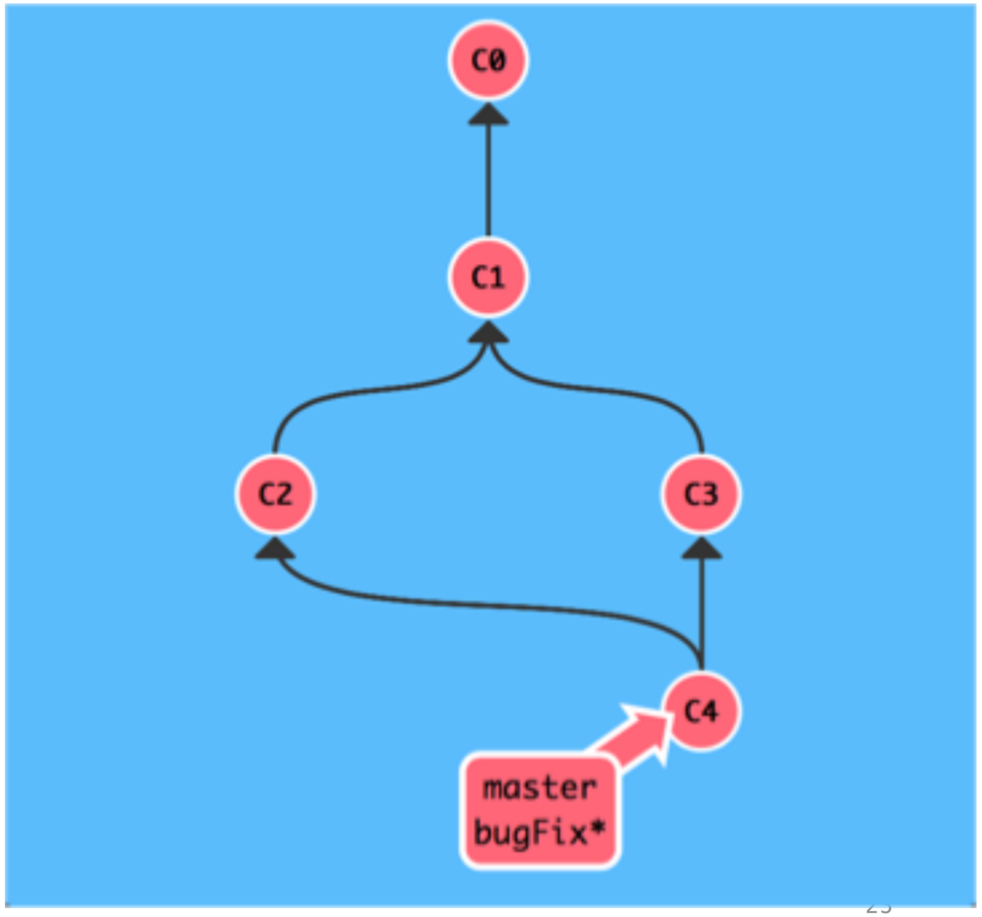
git merge bugFix





git checkout bugFix; git merge master





Pull Request

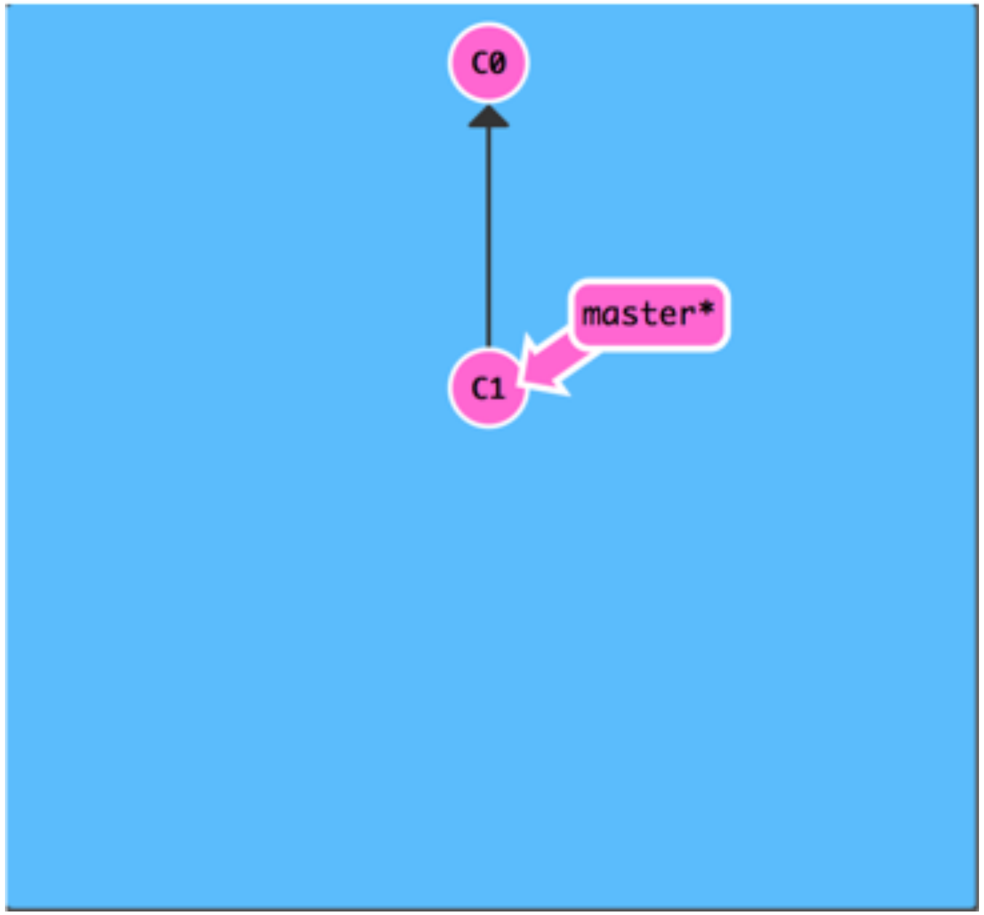
A pull request allows Git to merge code changes from a branch into the master (or other branch)

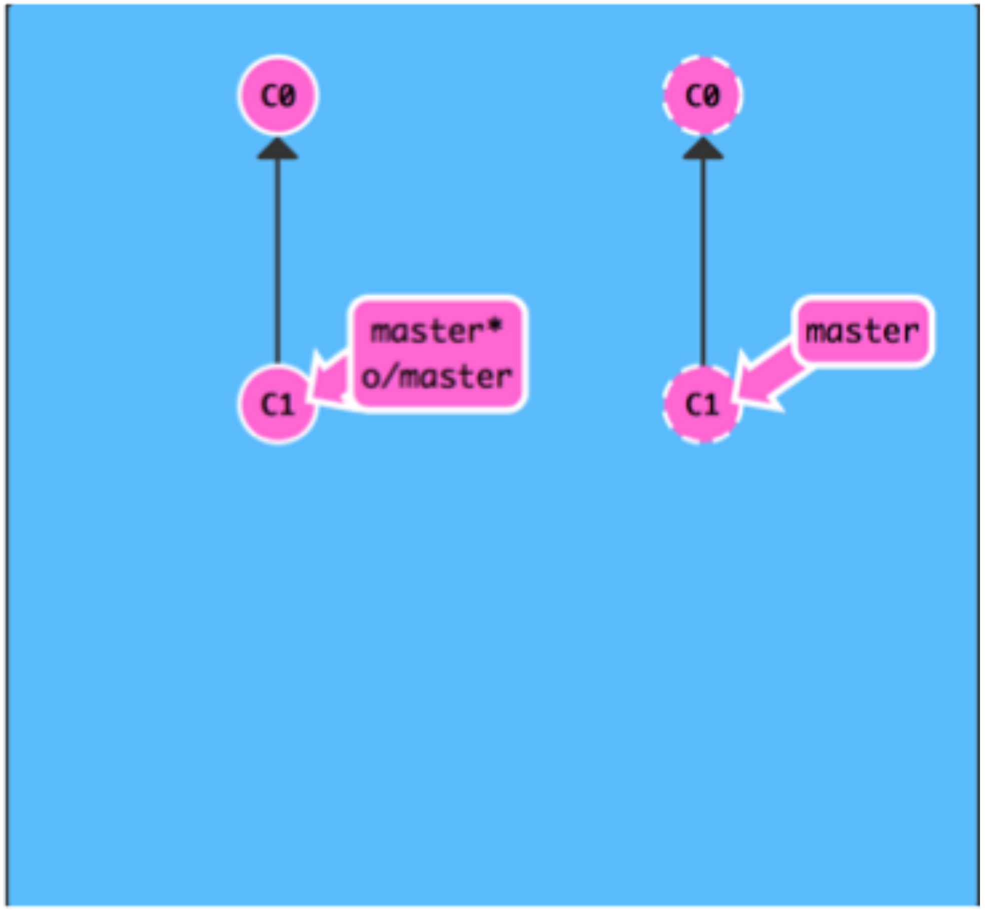
Once code has been pushed to a branch, the option to create a Pull Request becomes available

Demo…

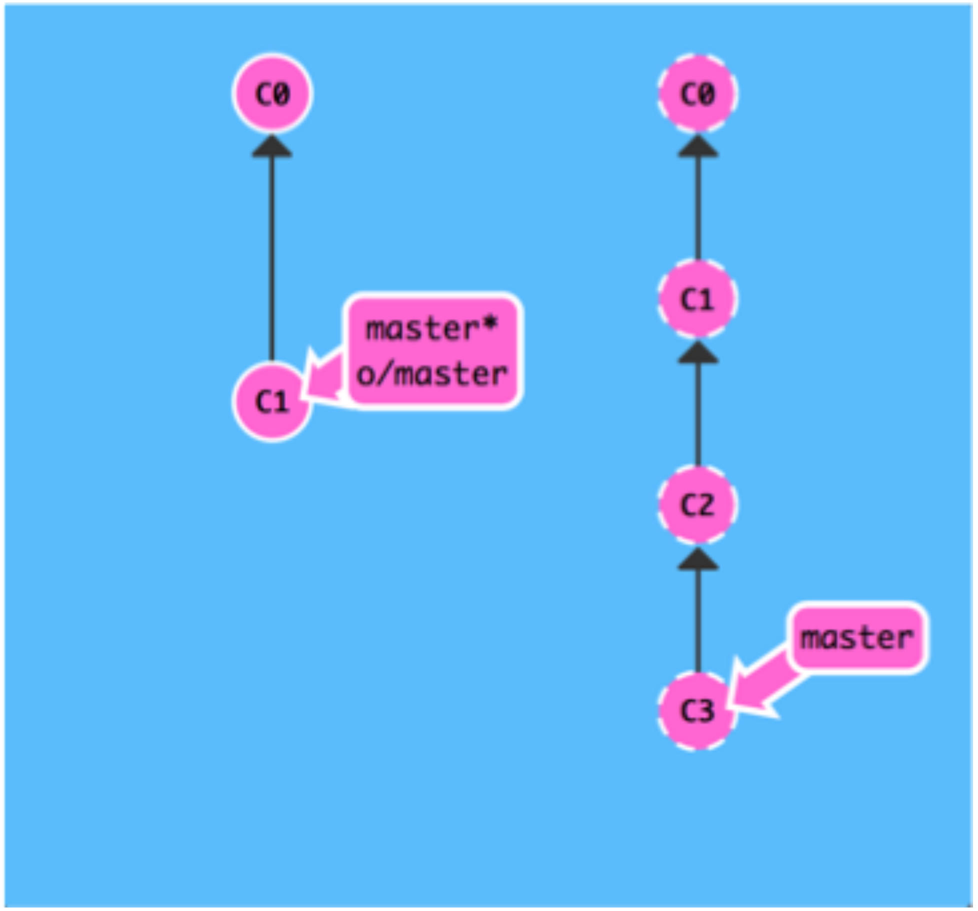
Pull Requests can be created by collaborators who don’t have the rights to merge the request. This way, someone can review the request and changes can be made before it is accepted and pulled in

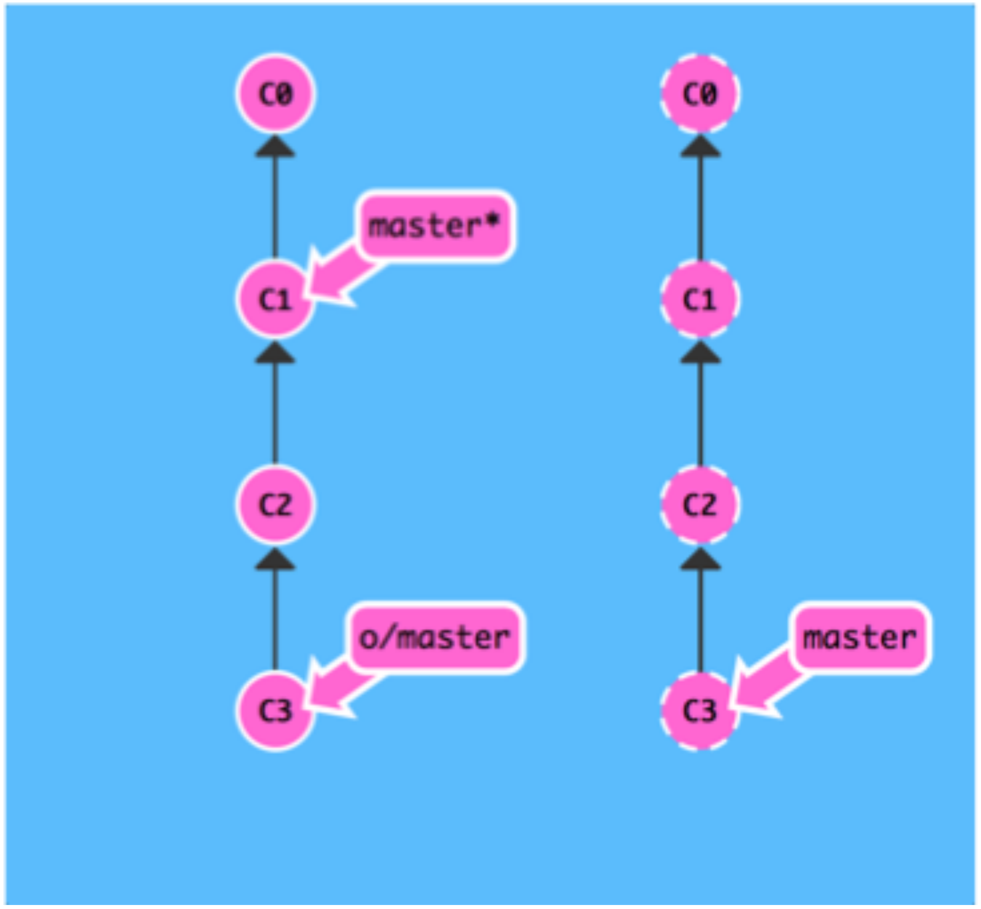
git clone



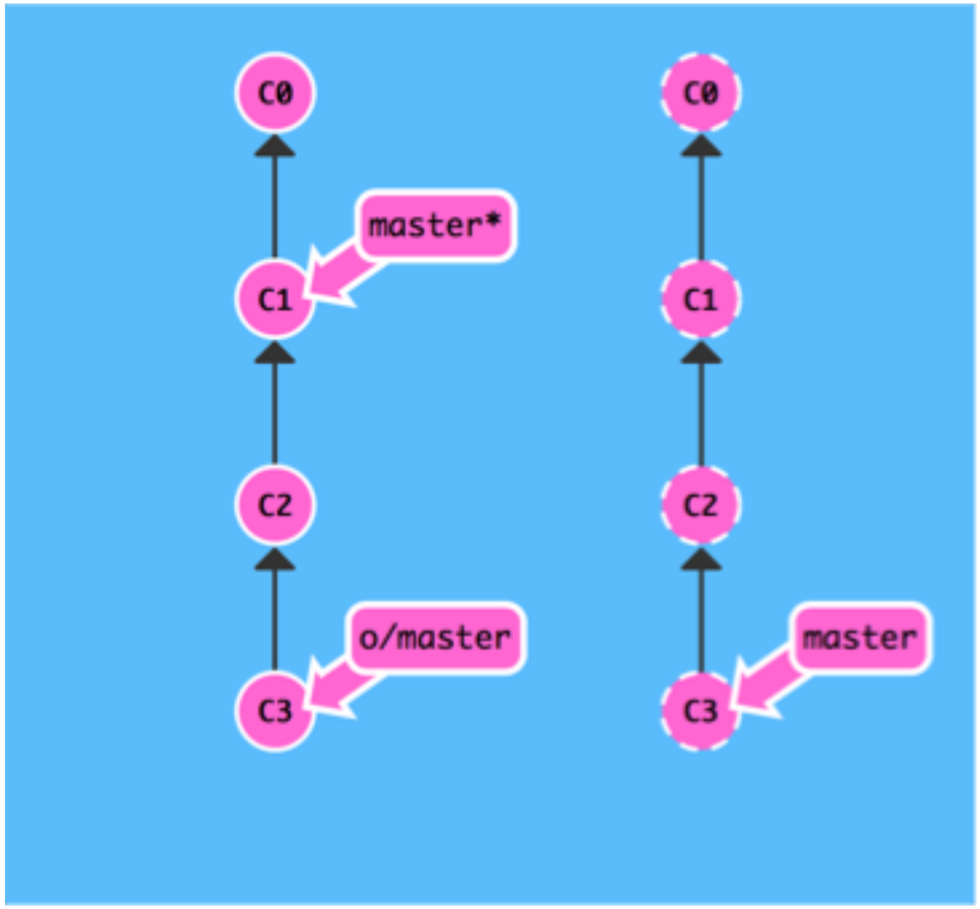


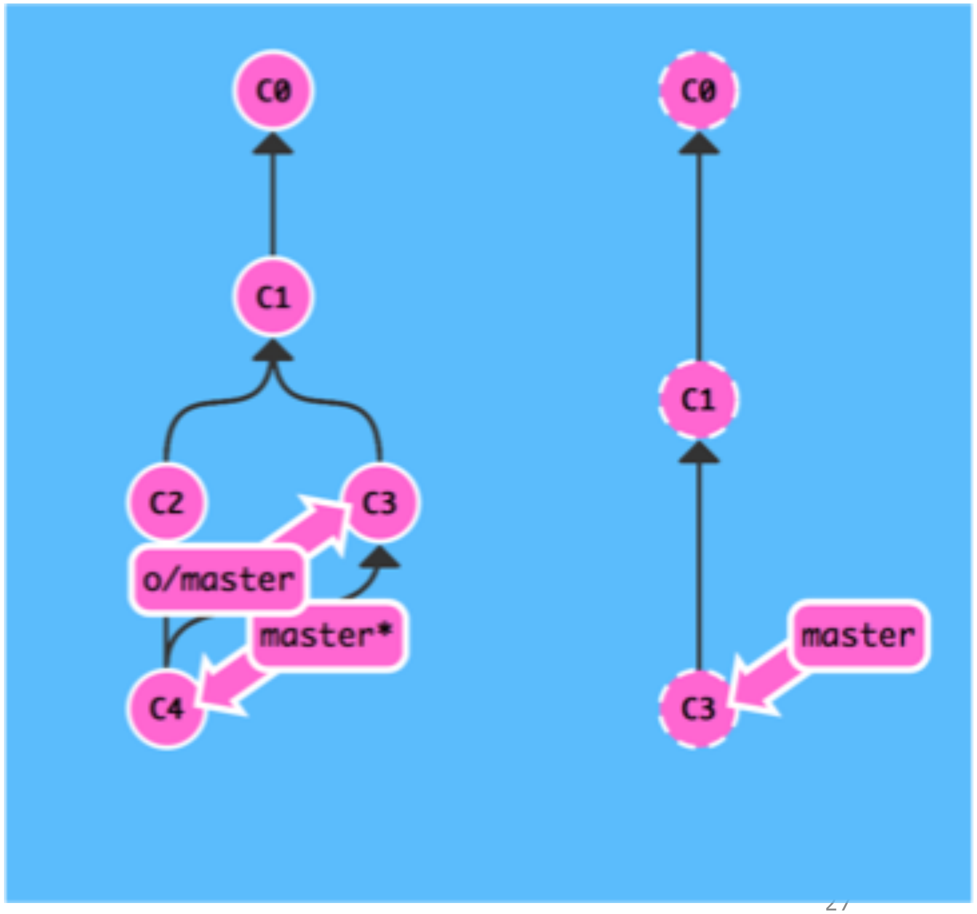
git fetch


C0
C1
master*
o/master
C0
C1
C2
C3
master

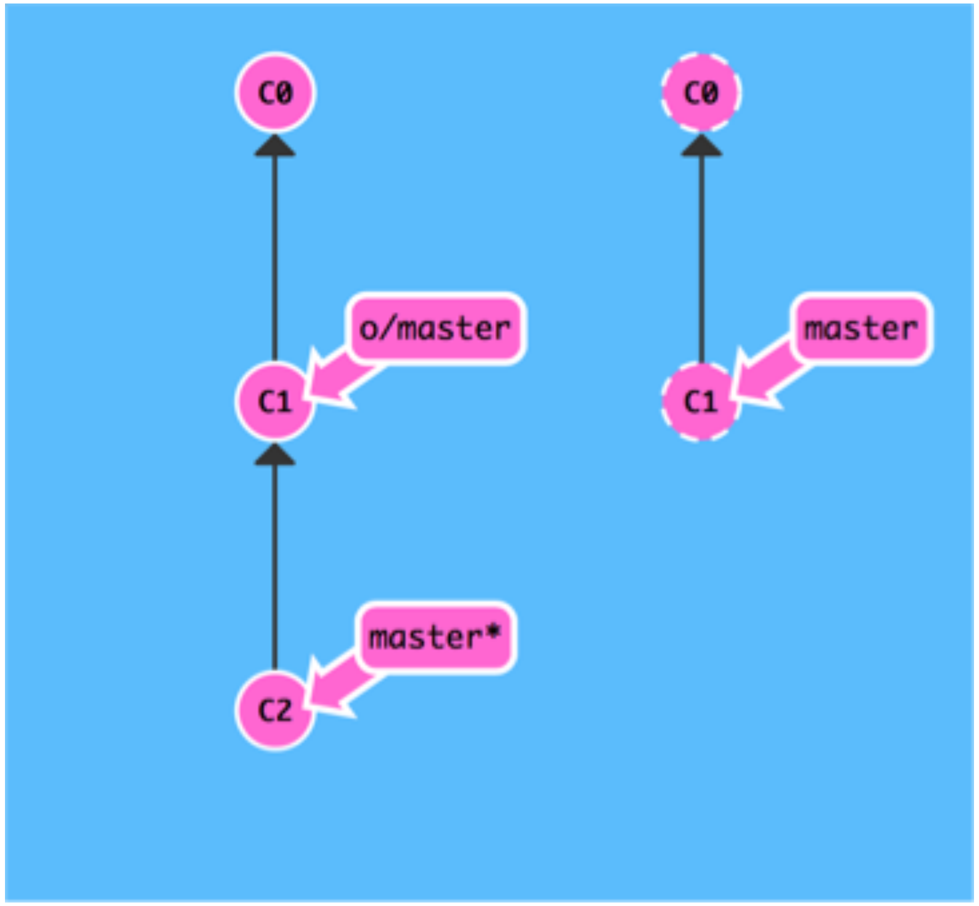

C0
master*
C1
C2
o/master
C3
C0
C1
C2
master
C3

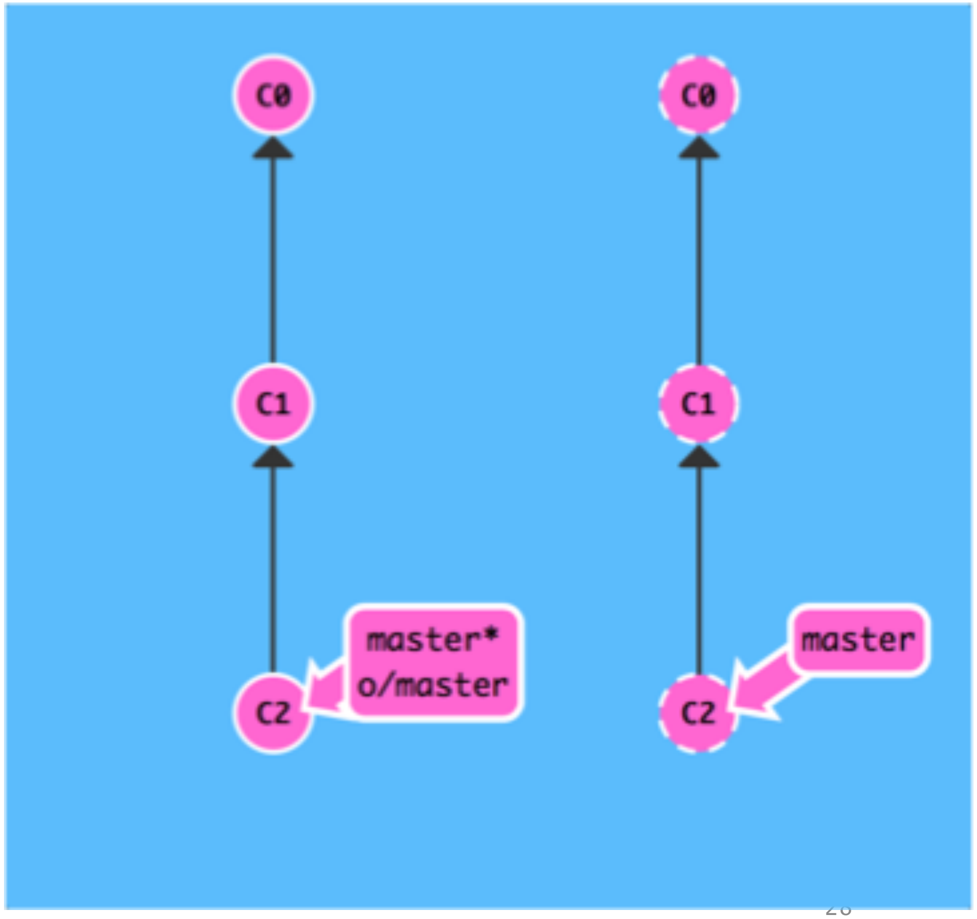
git merge o/master (both steps with git pull)





git push


C0
C1
C2
o/master
master*
C0
C1
master


C0
C1
C2
master*
o/master
C0
C1
C2
master

Continuous Integration

Git can be used to provide code for continuous integration

On commit, code can be built and tests run automatically with the results provided back to the developer

Can optionally update a Pull Request

Systems that allow this type of integration include (next slide)

DevOps

Git is also fundamental to deployments of code into test and production environments

This is now done in an automated fashion and Git provides an easy and reliable way of doing this

We cover this in Cloud Computing

Git and Employment

Many companies are now asking to see evidence of development skills

They use sites like:

GitHub	Stack Overflow (and related)

It is worth hosting projects publicly so that you can use the site as a type of electronic portfolio

When shouldn’t you use Git?

Git is optimised for developers working on code – it *is not* a collaborative document management system like Google Docs, Apple iCloud, Microsoft Office 360, etc.

If you are working on documents collaboratively, Google Docs is a better option

- People can edit and see changes from multiple people in real time
- Changes are on a character by character level – not just lines
- Git does not work well with binary files like word documents