

# Principles of Software Construction: Objects, Design, and Concurrency

Software engineering in practice

Git, software development workflows, and monorepos

Josh Bloch

**Charlie Garrod**

Darya Melicher

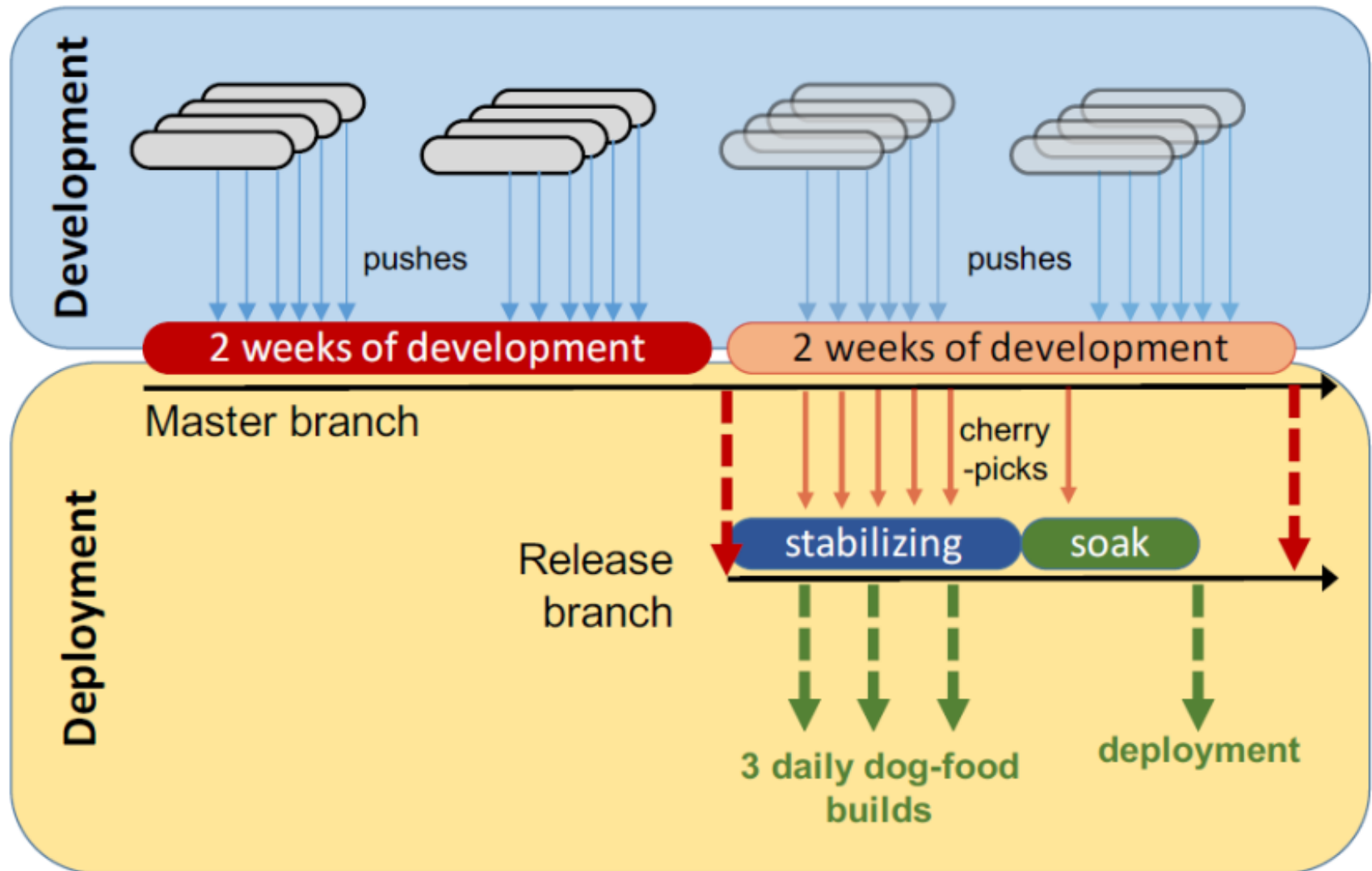


# Administrivia

- Homework 6 due next Wednesday
  - Checkpoint deadline Monday night

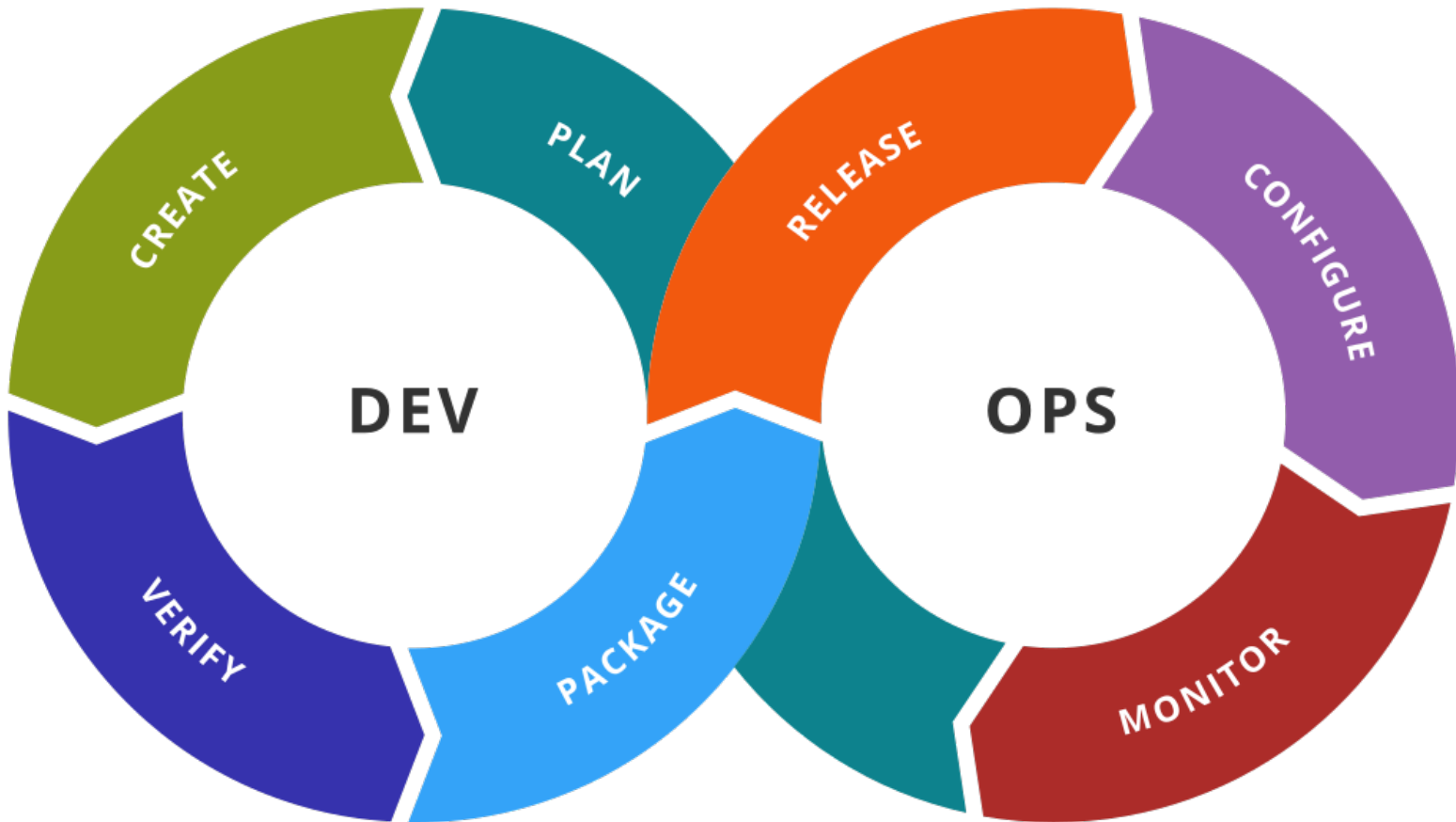
# Key concepts from Tuesday

# Compare to the Facebook release cycle





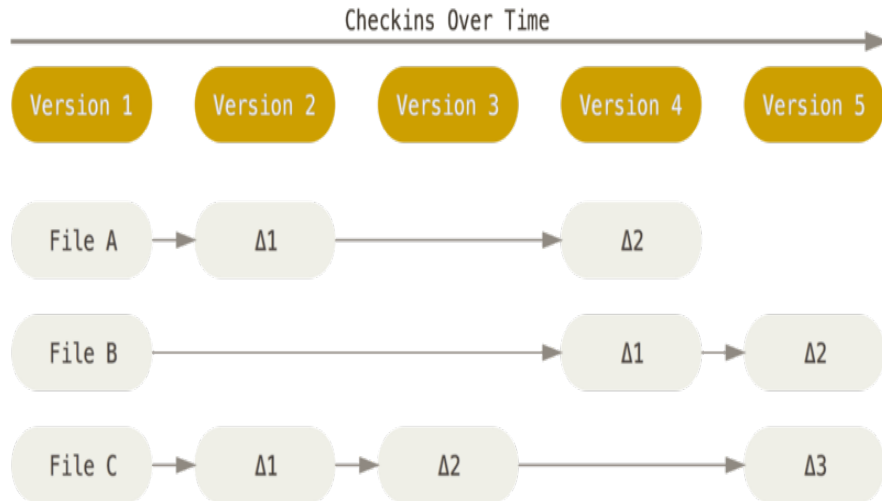
# DevOps: Development / Operations



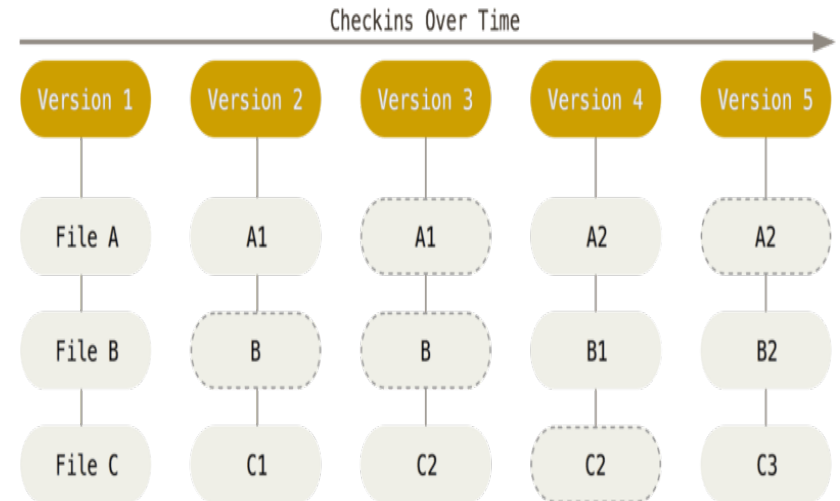
# Configuration management (CM)

- Definition (Pressman): *Configuration management “is a set of tracking and control activities that are initiated when a software engineering projects begins and terminates when software is taken out of operation.”*

# SVN (left) vs. Git (right)




- SVN stores changes to a base version of each file
- Version numbers (1, 2, 3, ...) are increased by one after each commit



- Git stores each version as a snapshot
- If files have not changed, only a link to the previous file is stored
- Each version is referred by the SHA-1 hash of the contents

<https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control>

# A brief Git history...

 [torvalds](#) / [linux](#)

Watch 6,587

Star 66,366

Fork 24,078

<> Code

Pull requests 238

Projects 0

Insights

Linux kernel source tree

🕒 797,570 commits

🌿 1 branch

🏷️ 582 releases

👤 ∞ contributors

📄 View license

Branch: master


New pull request

Create new file

Upload files

Find file

Clone or download

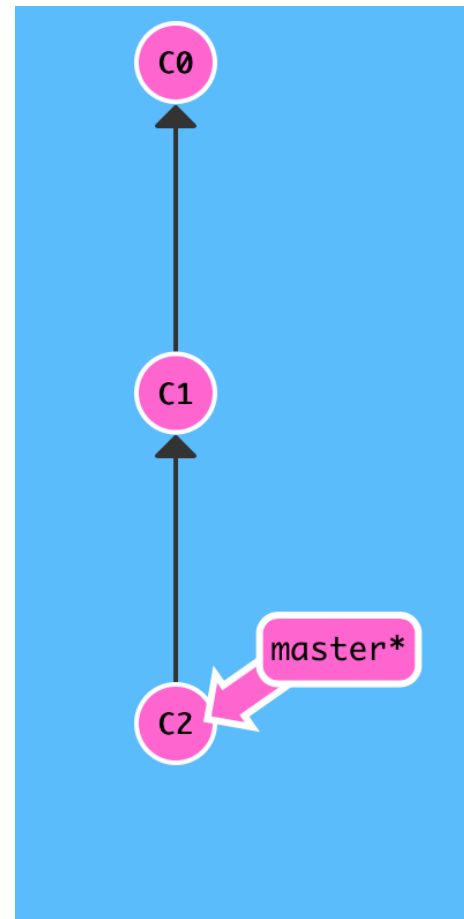
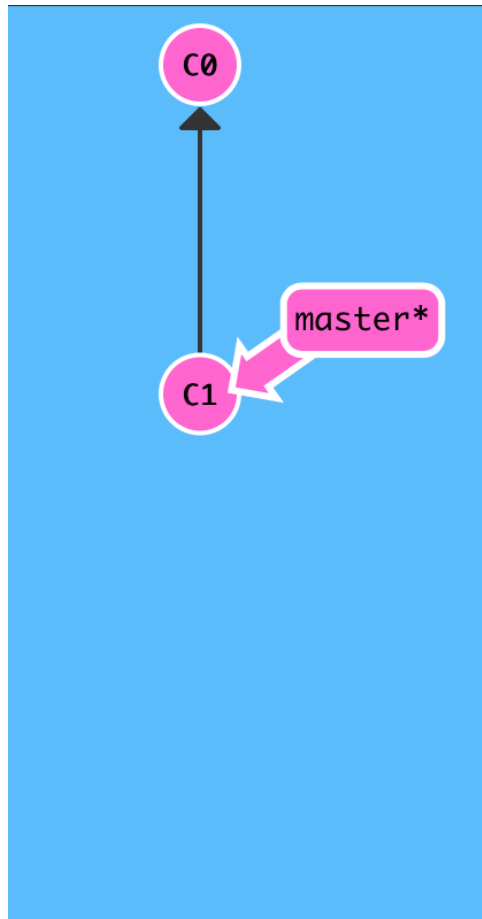
 **torvalds** Merge git://git.kernel.org/pub/scm/linux/kernel/git/davem/net ... Latest commit 60b5482 17 hours ago

<a href="#">Documentation</a>	Merge tag 'spi-fix-v4.20-rc4' of git://git.kernel.org/pub/scm/linux/k...	21 hours ago
<a href="#">LICENSES</a>	Merge tag 'docs-4.20' of git://git.lwn.net/linux	a month ago
<a href="#">arch</a>	Merge git://git.kernel.org/pub/scm/linux/kernel/git/davem/net	17 hours ago
<a href="#">block</a>	SCSI: fix queue cleanup race before queue initialization is done	15 days ago
...		

# Today

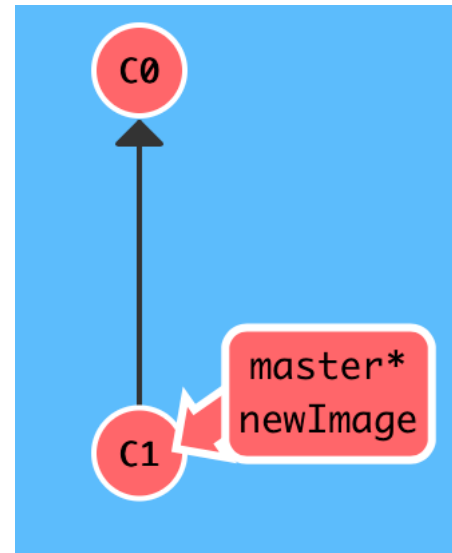
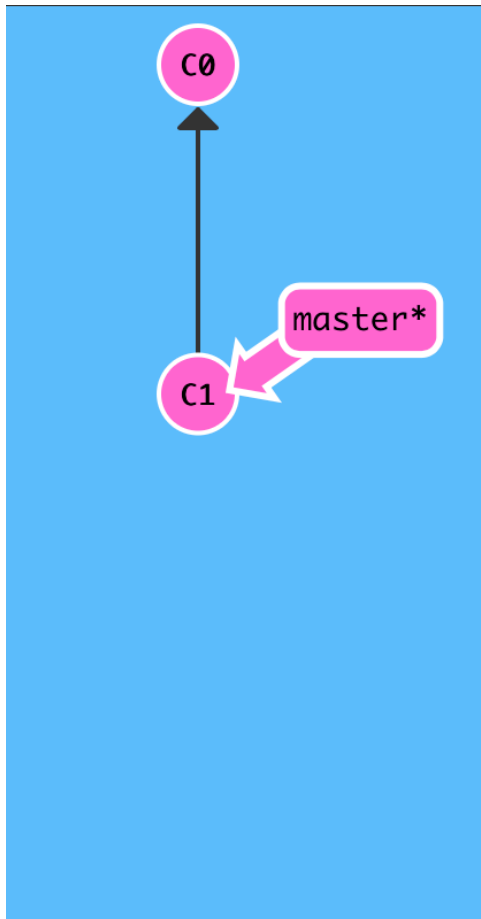
- Practical Git
- Common workflows using Git
- Developing at scale

# git commit

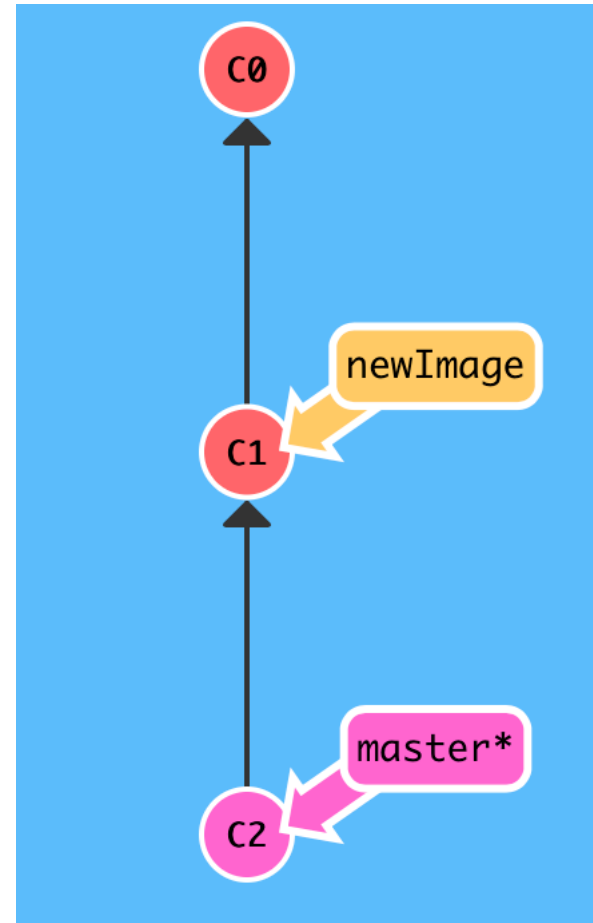
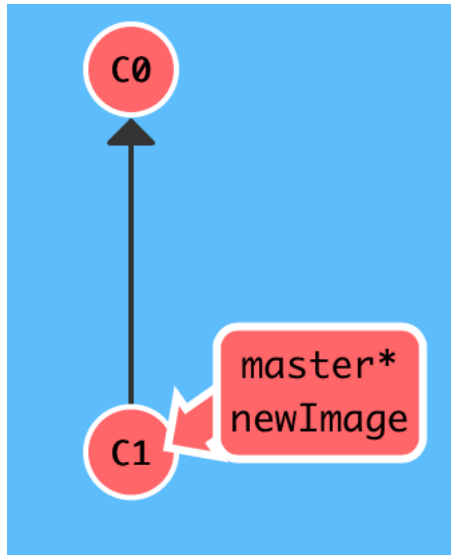


Graphics by <https://learngitbranching.js.org>

# git branch newImage

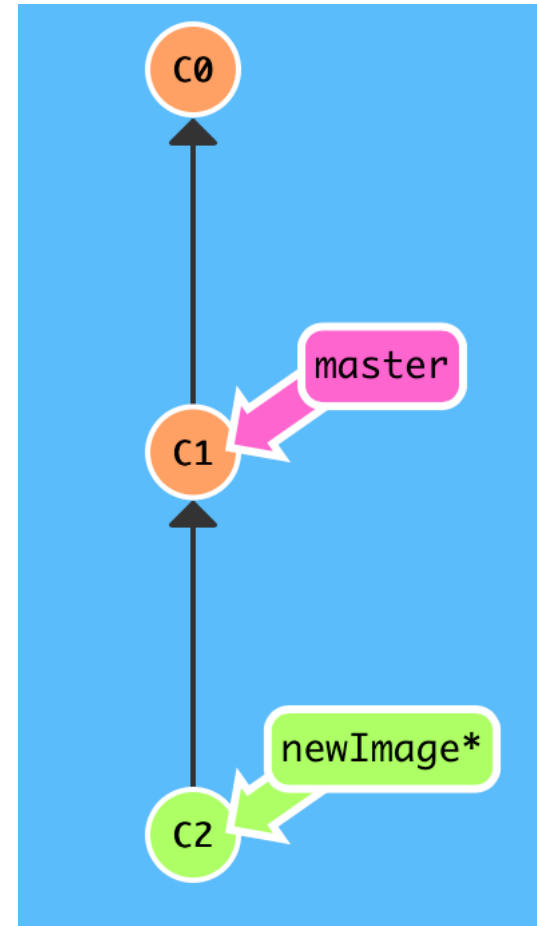
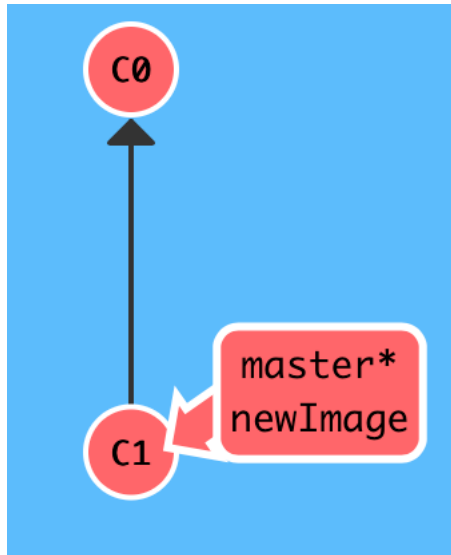


# git commit

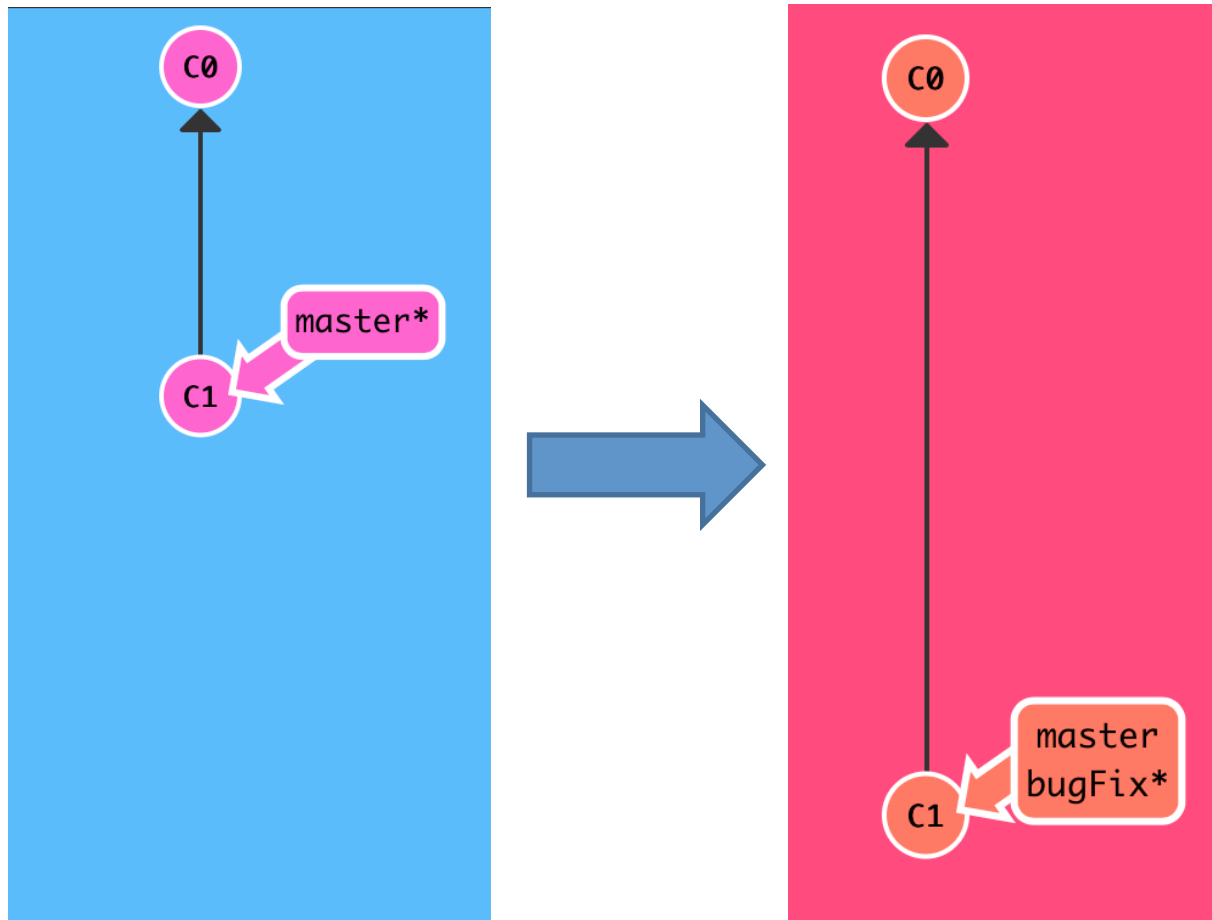




git checkout newImage; git commit

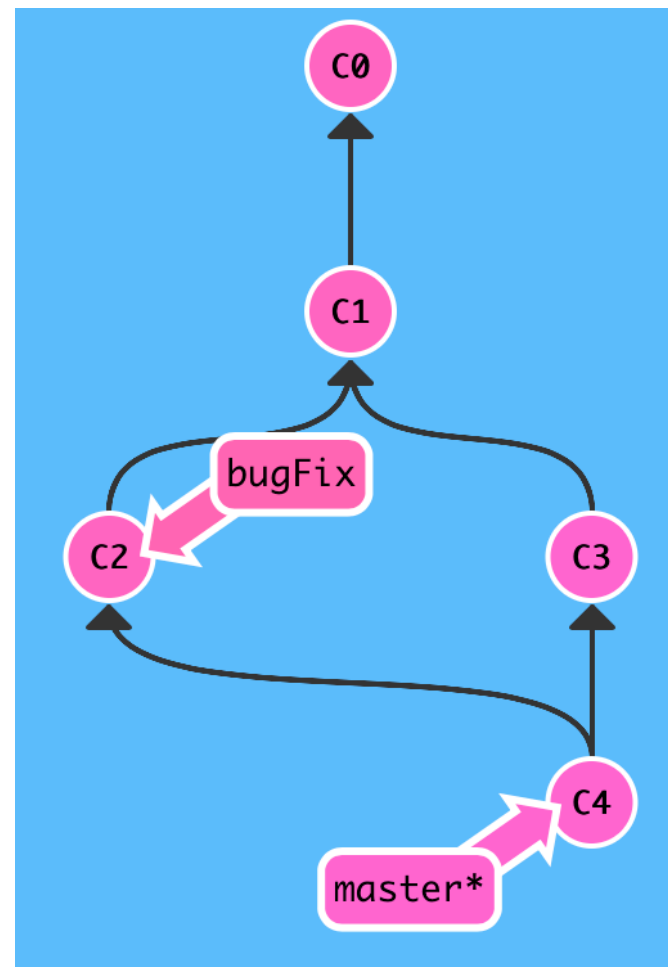
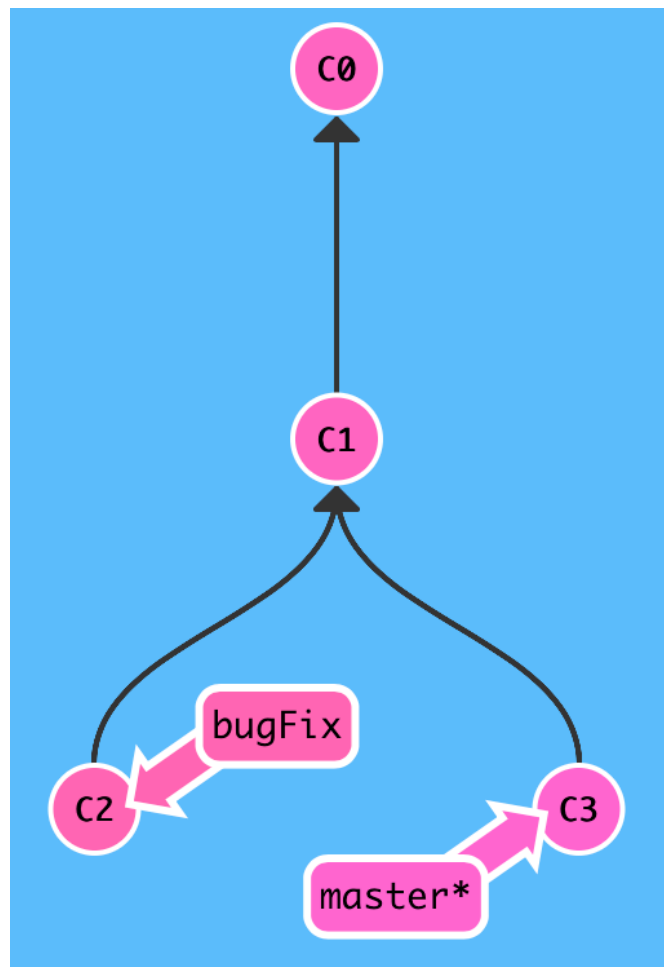


Activity: Make a new branch named bugFix and switch to that branch

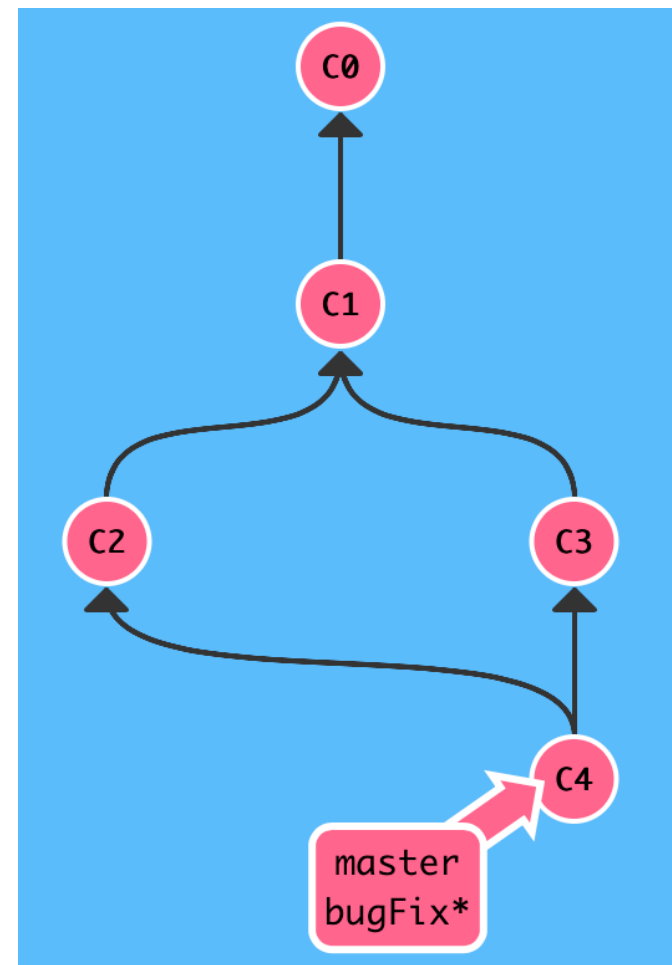
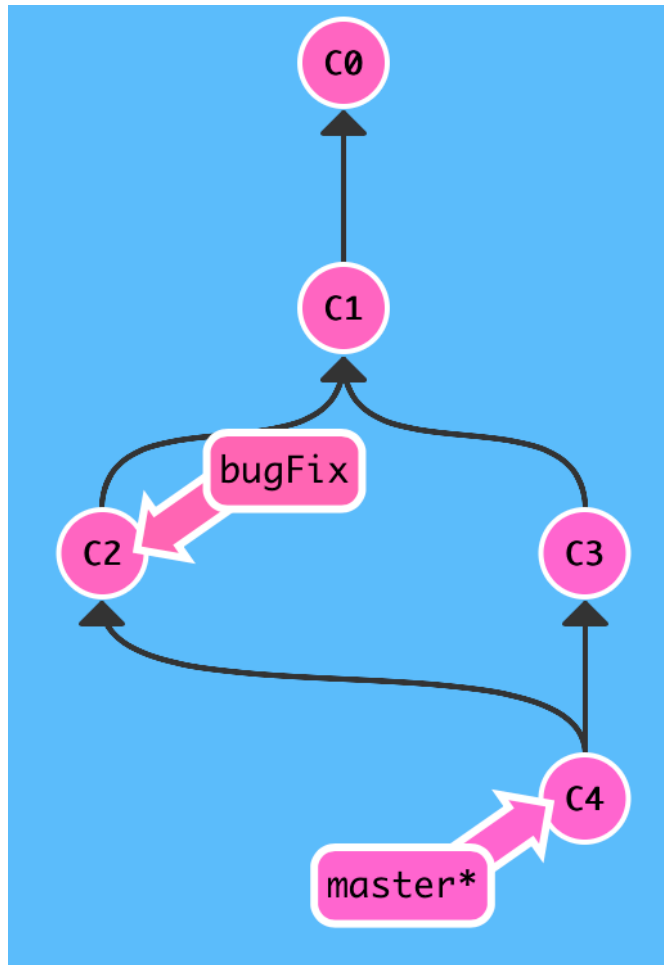


# Three ways to move work around between branches

## 1) git merge bugFix (into master)

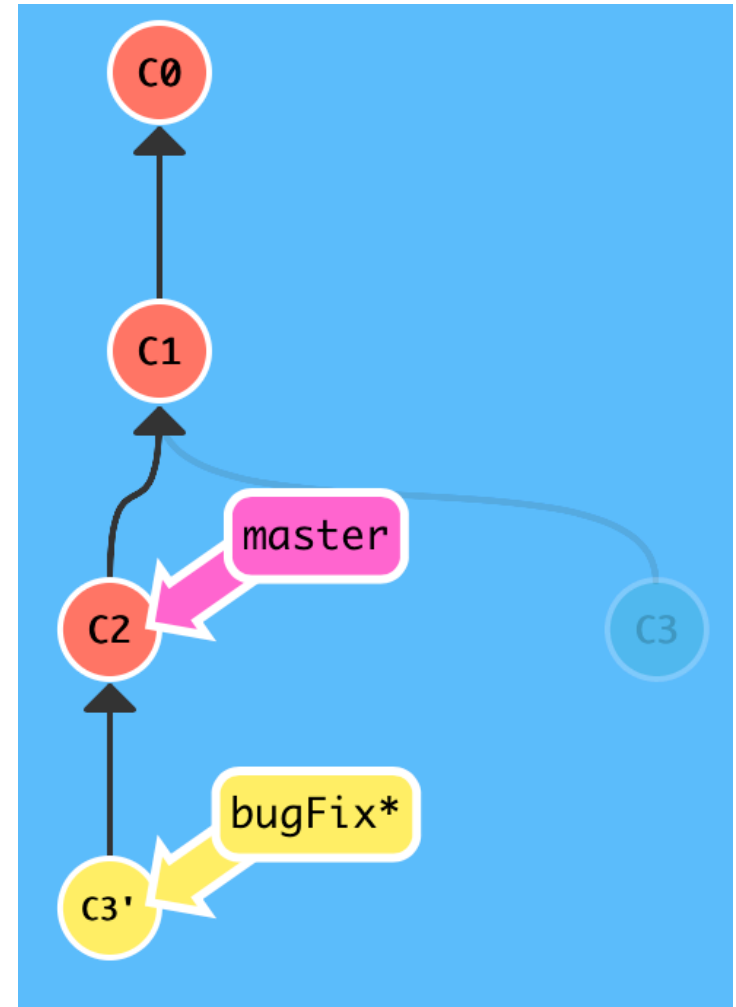
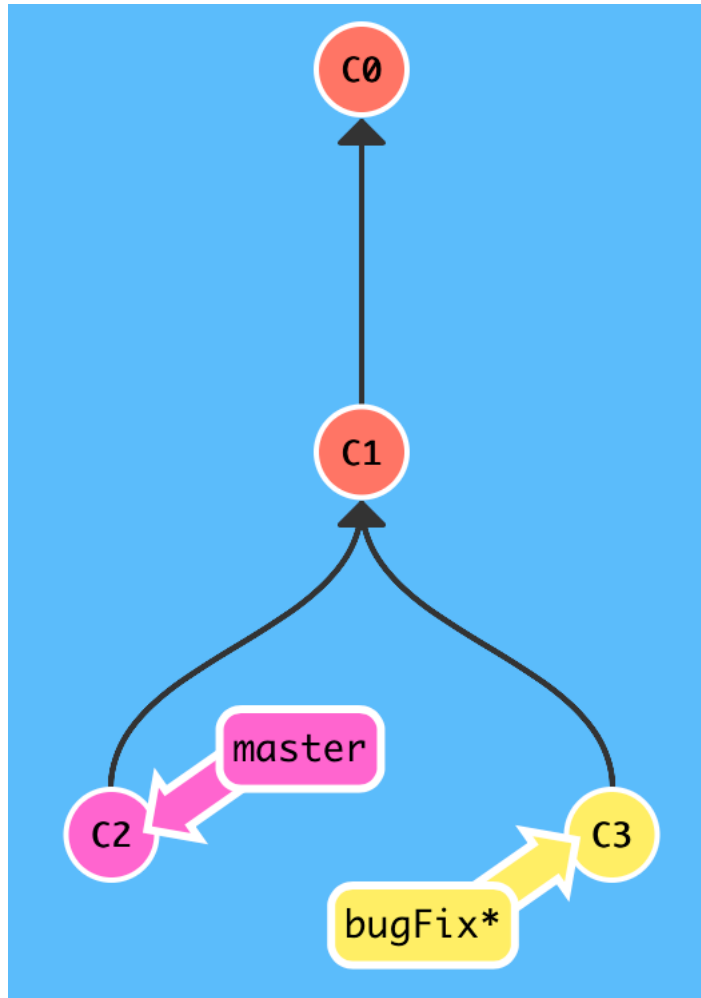


`git checkout bugfix; git merge master (into bugFix)`



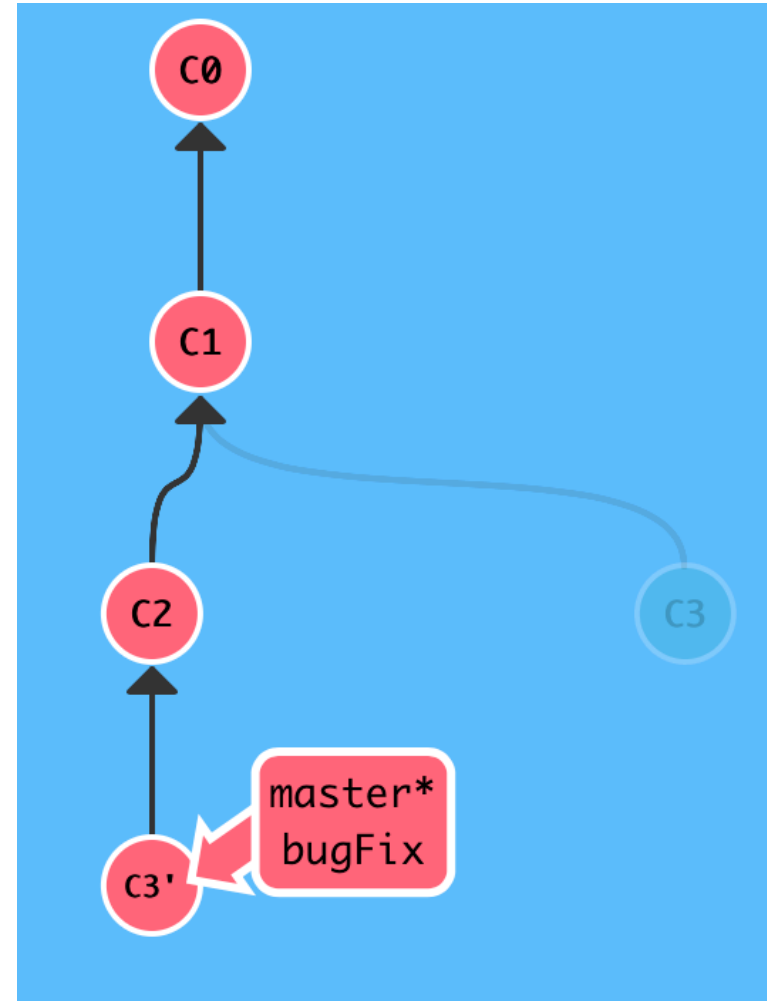
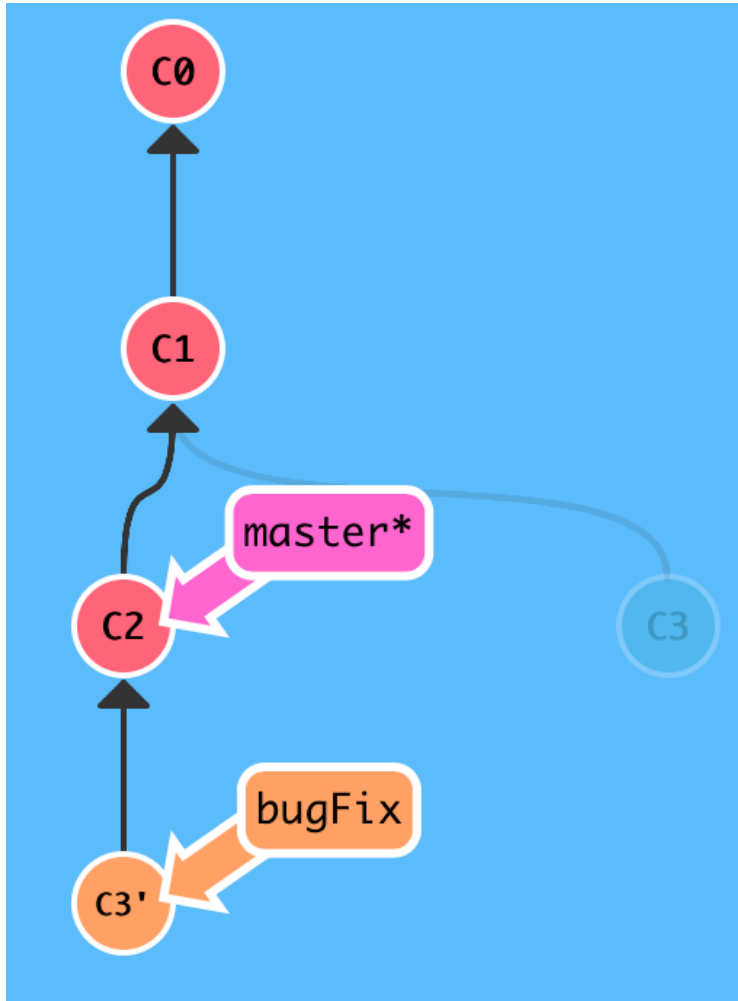
Move work from bugFix directly onto master

## 2) git rebase master



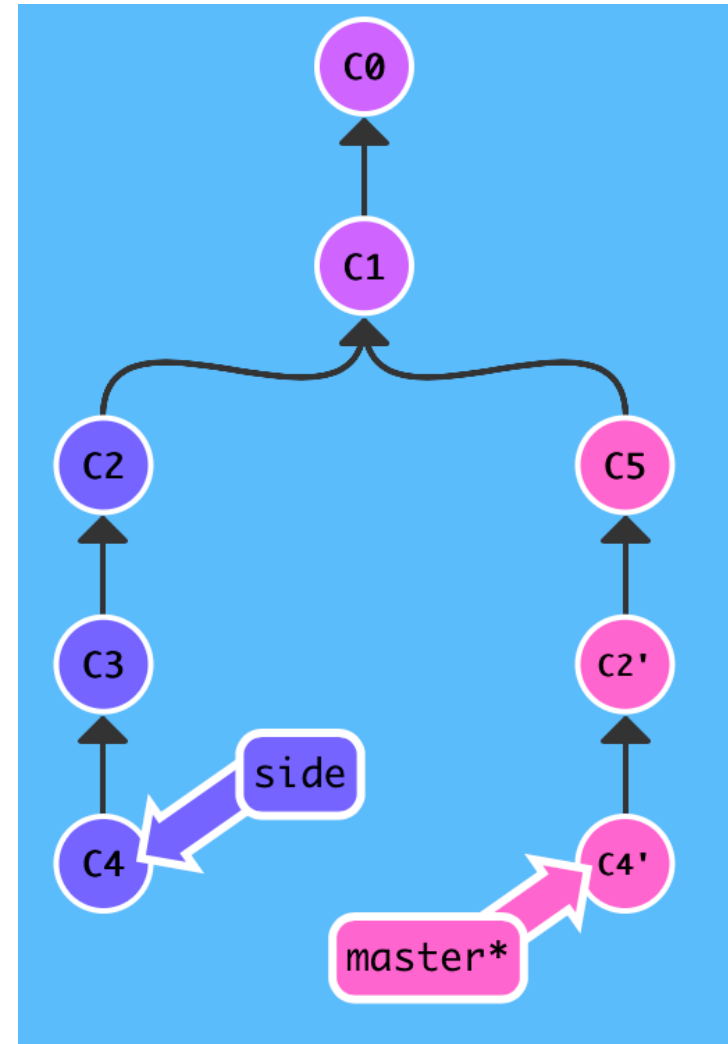
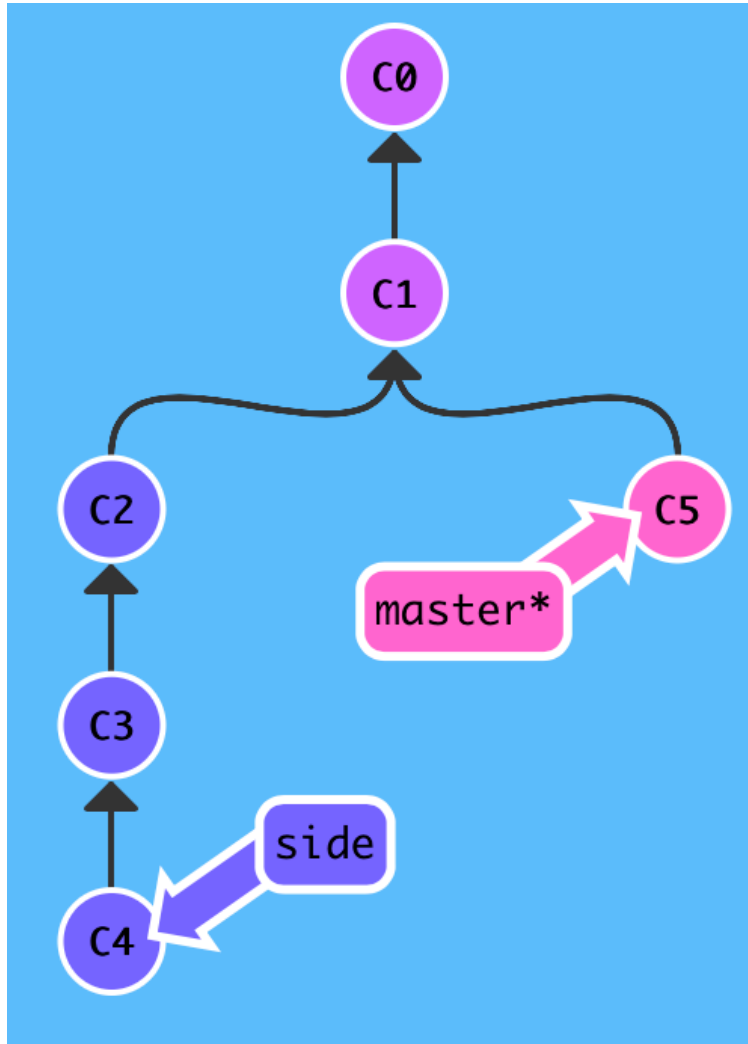
But master hasn't been updated, so:

`git checkout master; git rebase bugFix`

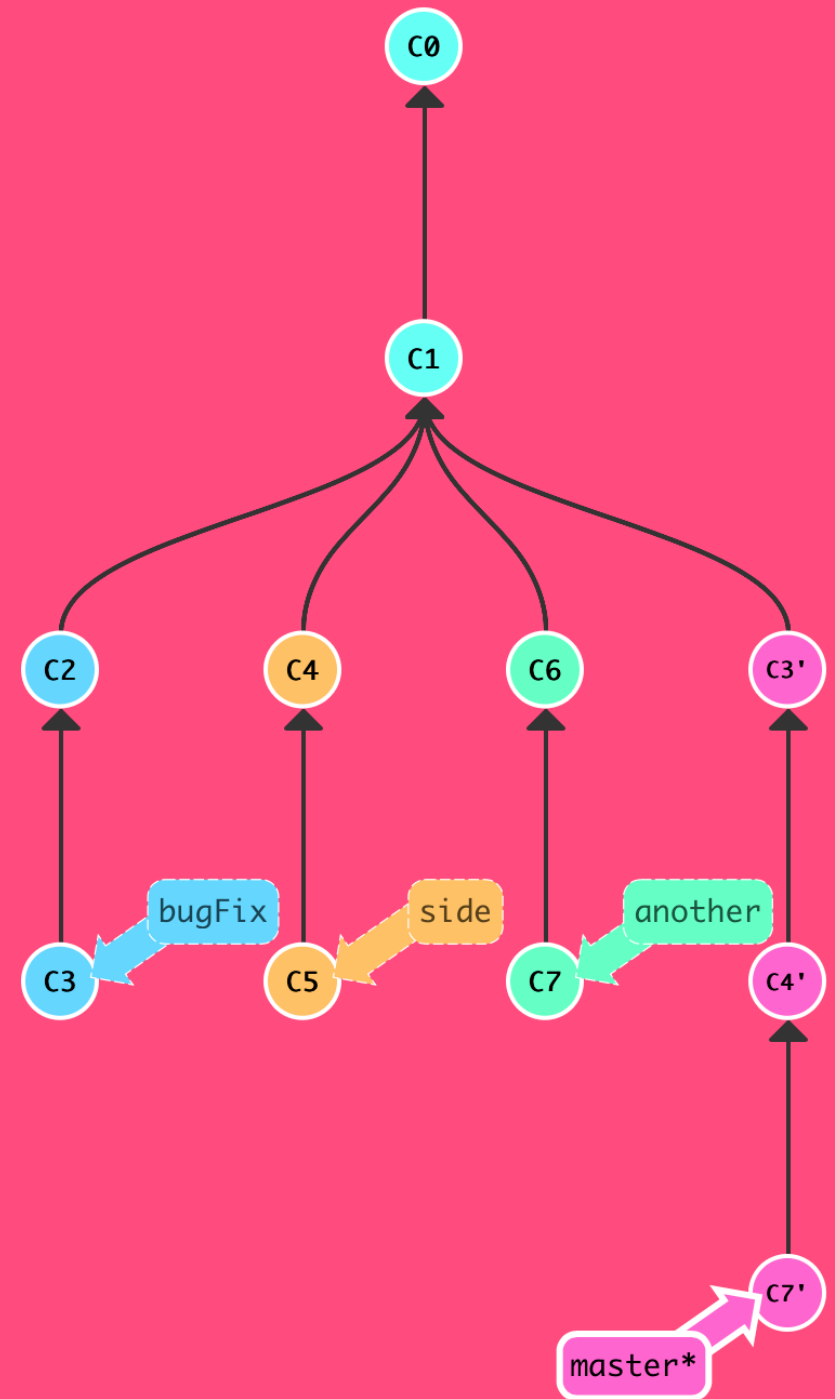
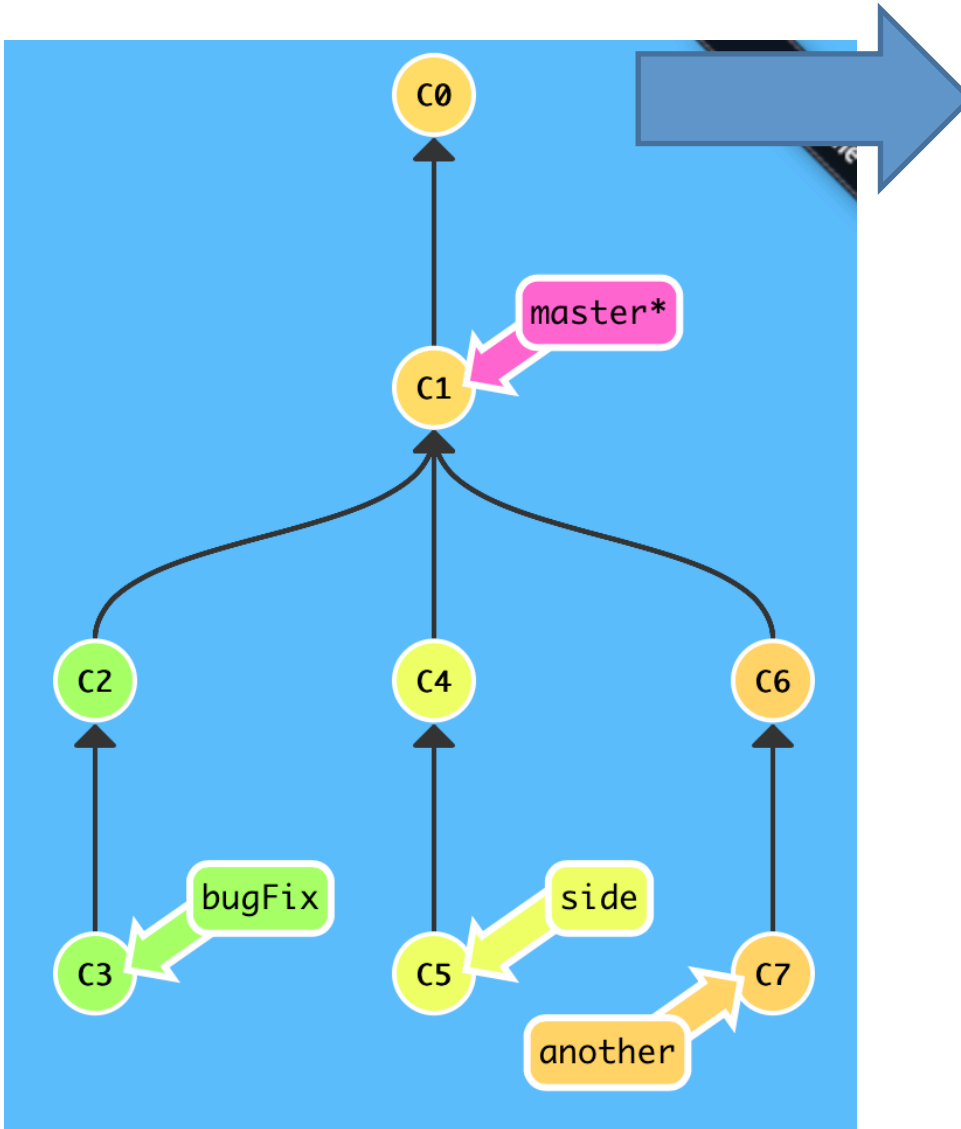


Copy a series of commits below current location

3) `git cherry-pick C2 C4`



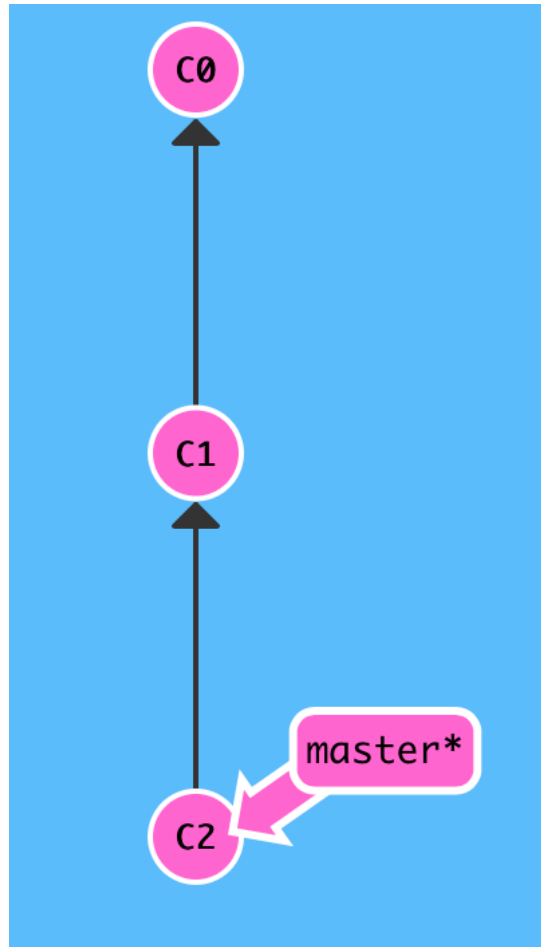
# Activity:



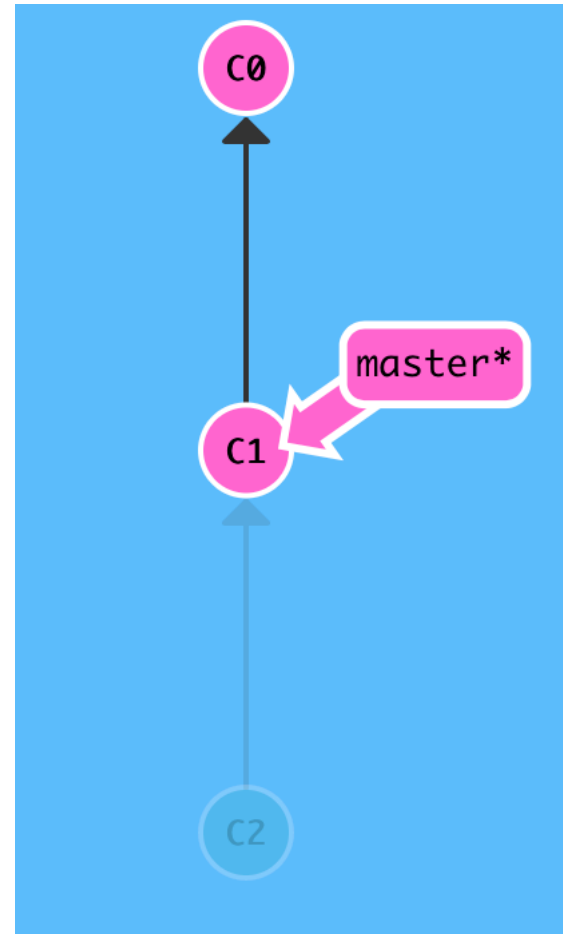


## Ways to undo work (1)

`git reset HEAD~1`

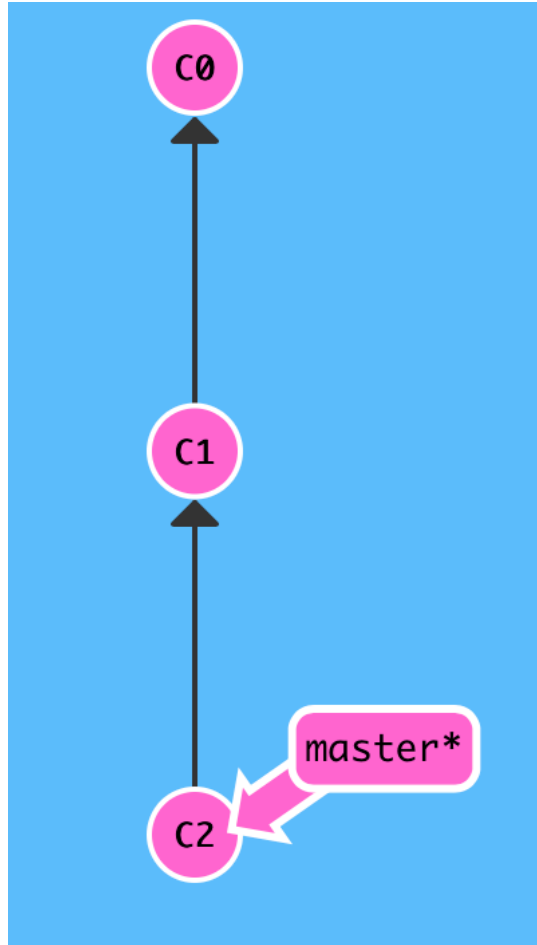


HEAD is the symbolic name for the currently checked out commit

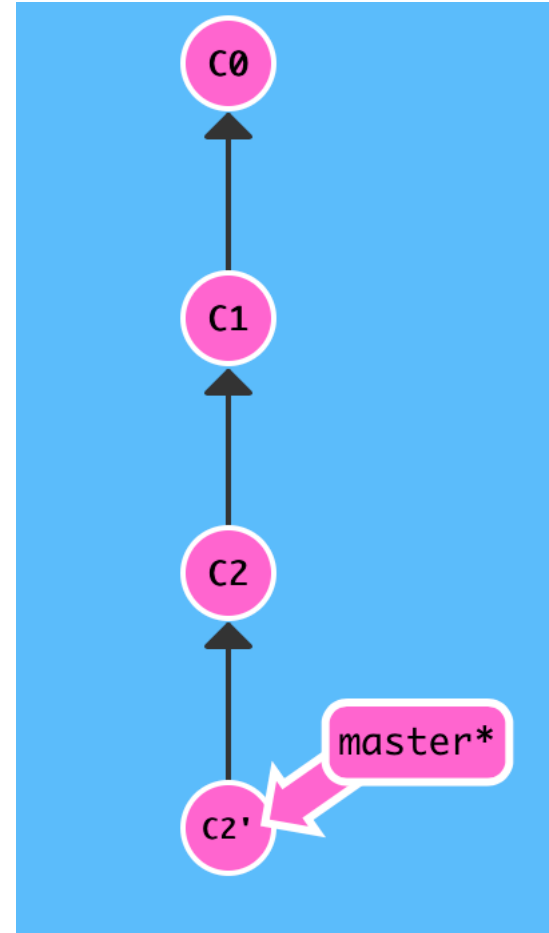


## Ways to undo work (2)

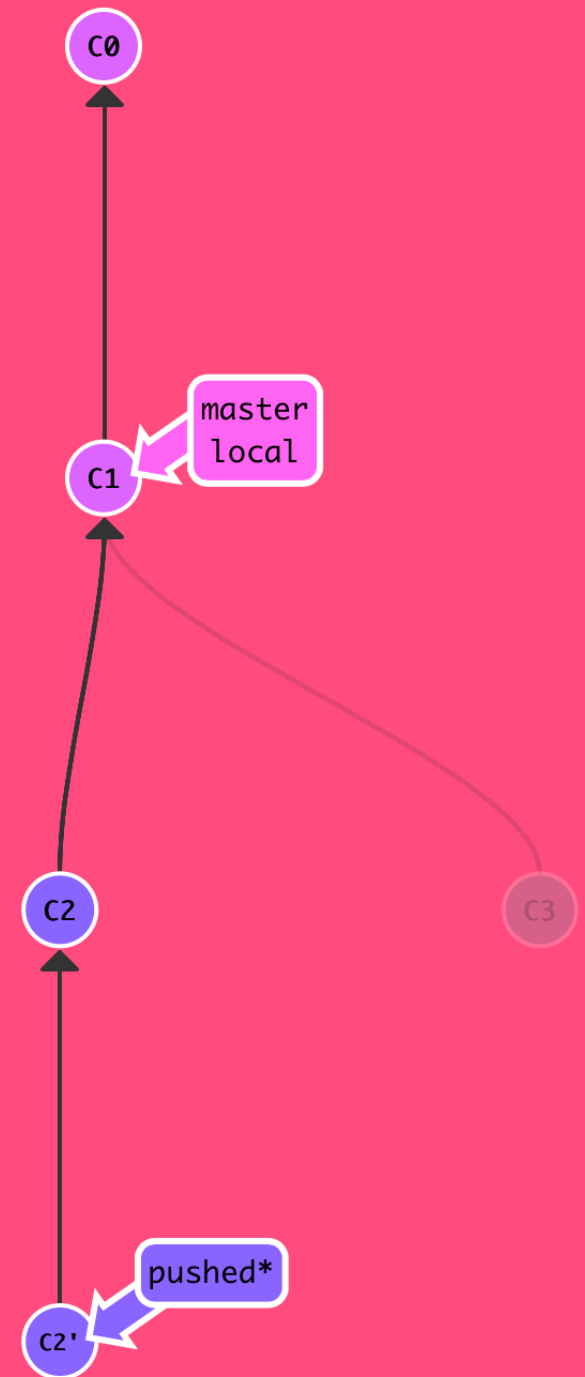
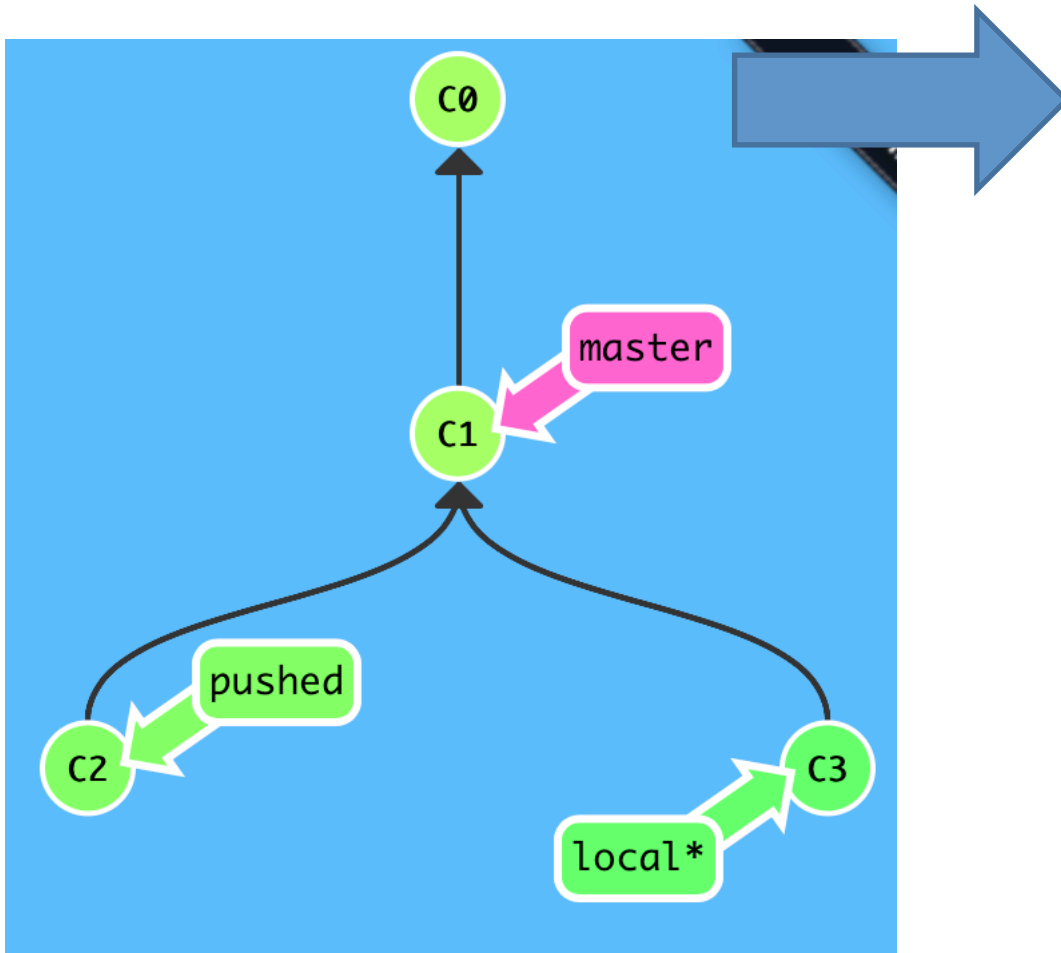
`git revert HEAD`



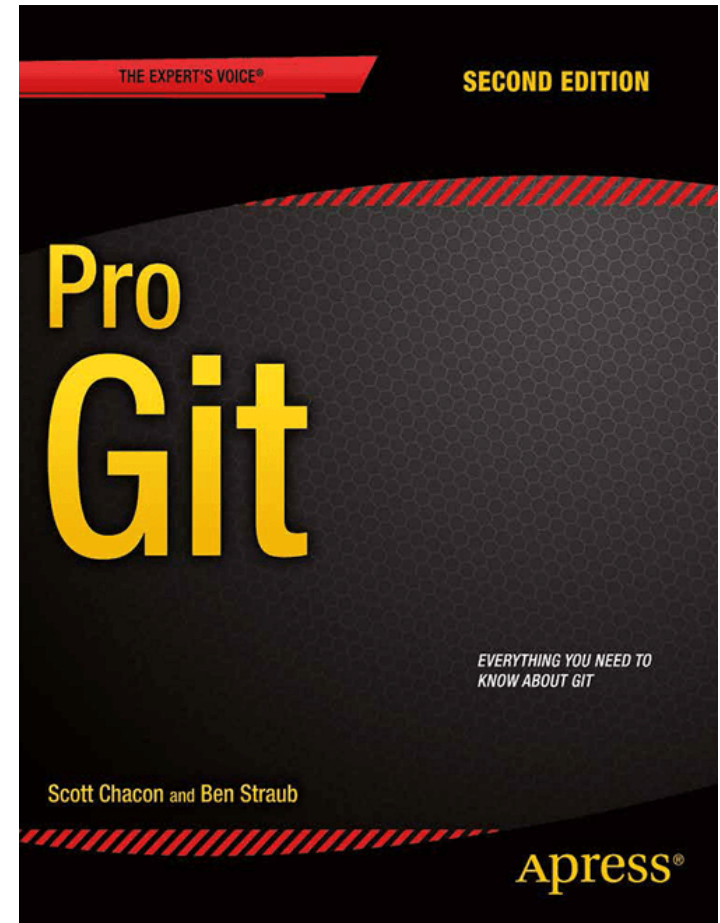
git reset does not work  
for remote branches



# Activity:



Highly recommended



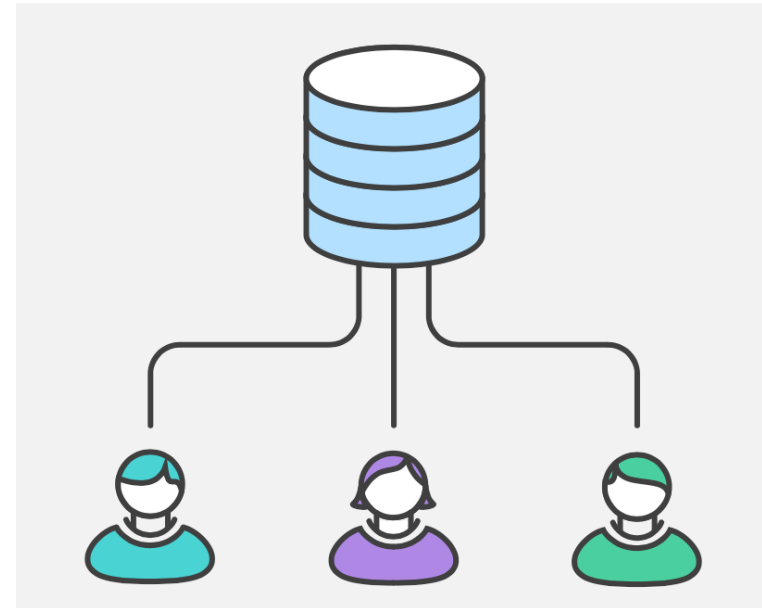
<https://git-scm.com/book/en/v2>

# Today

- Practical Git
- Common workflows using Git
- Developing at scale

# 1. Centralized workflow

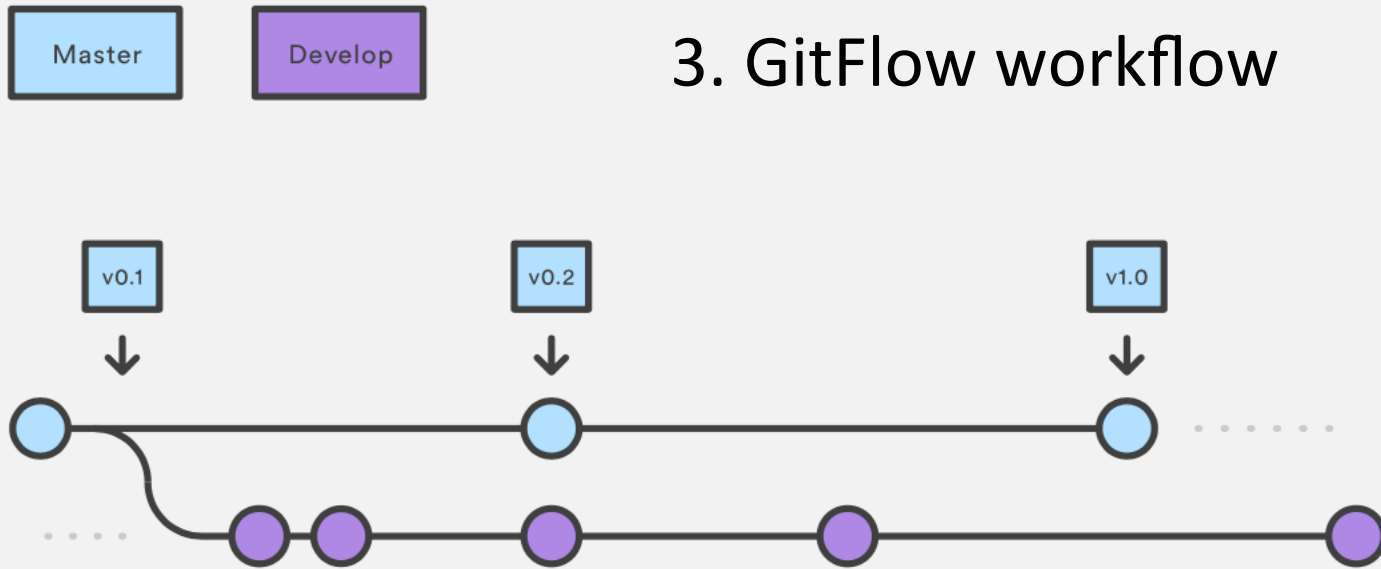
- Central repository to serve as the single point-of-entry for all changes to the project
- Default development branch is called master
  - all changes are committed into master
  - doesn't require any other branches



## 2. Git feature branch workflow

- *All* feature development should take place in a dedicated branch instead of the master branch
- Multiple developers can work on a particular feature without disturbing the main codebase
  - master branch will never contain broken code (enables CI)
  - Enables pull requests (code review)

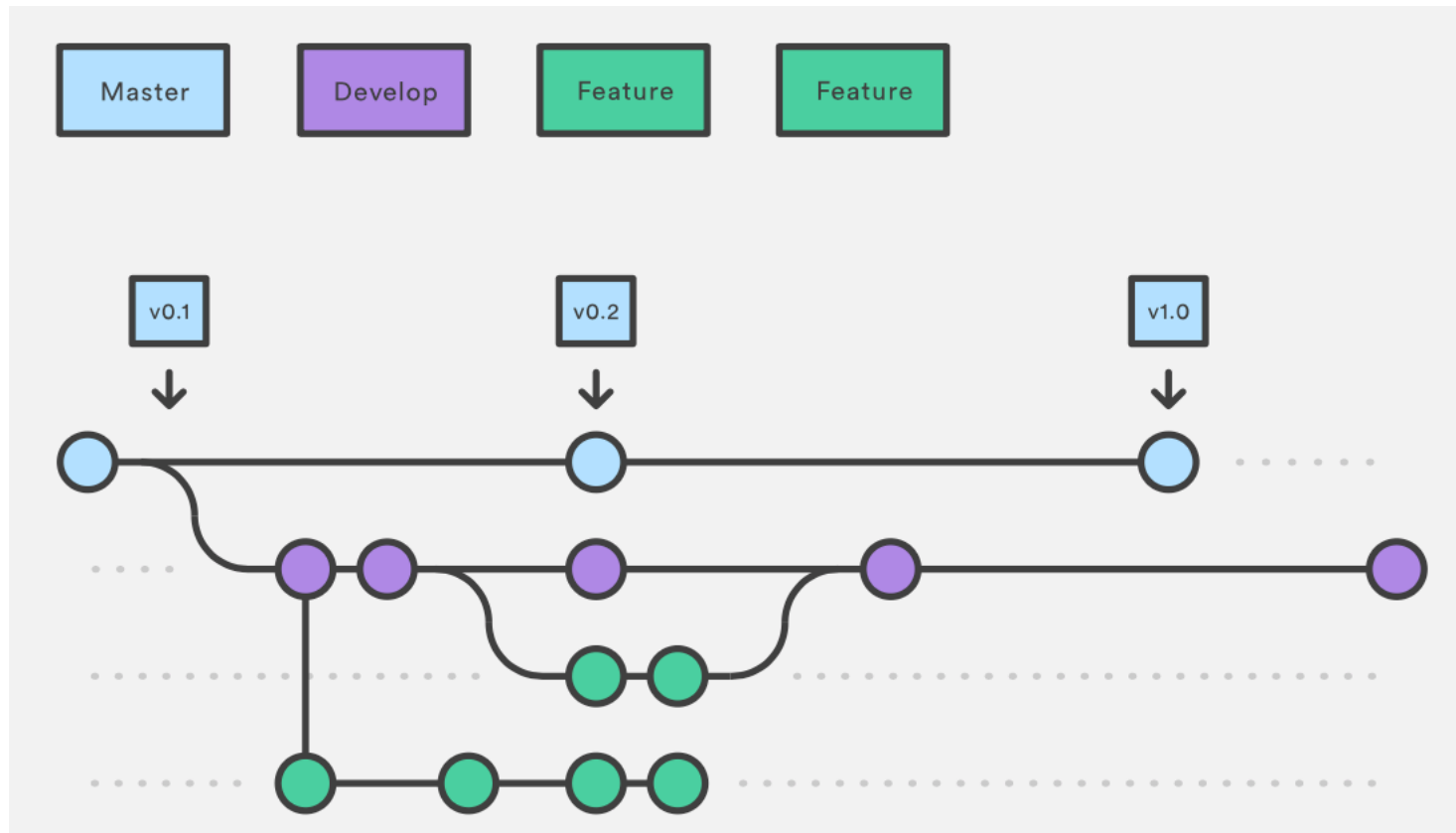
### 3. GitFlow workflow



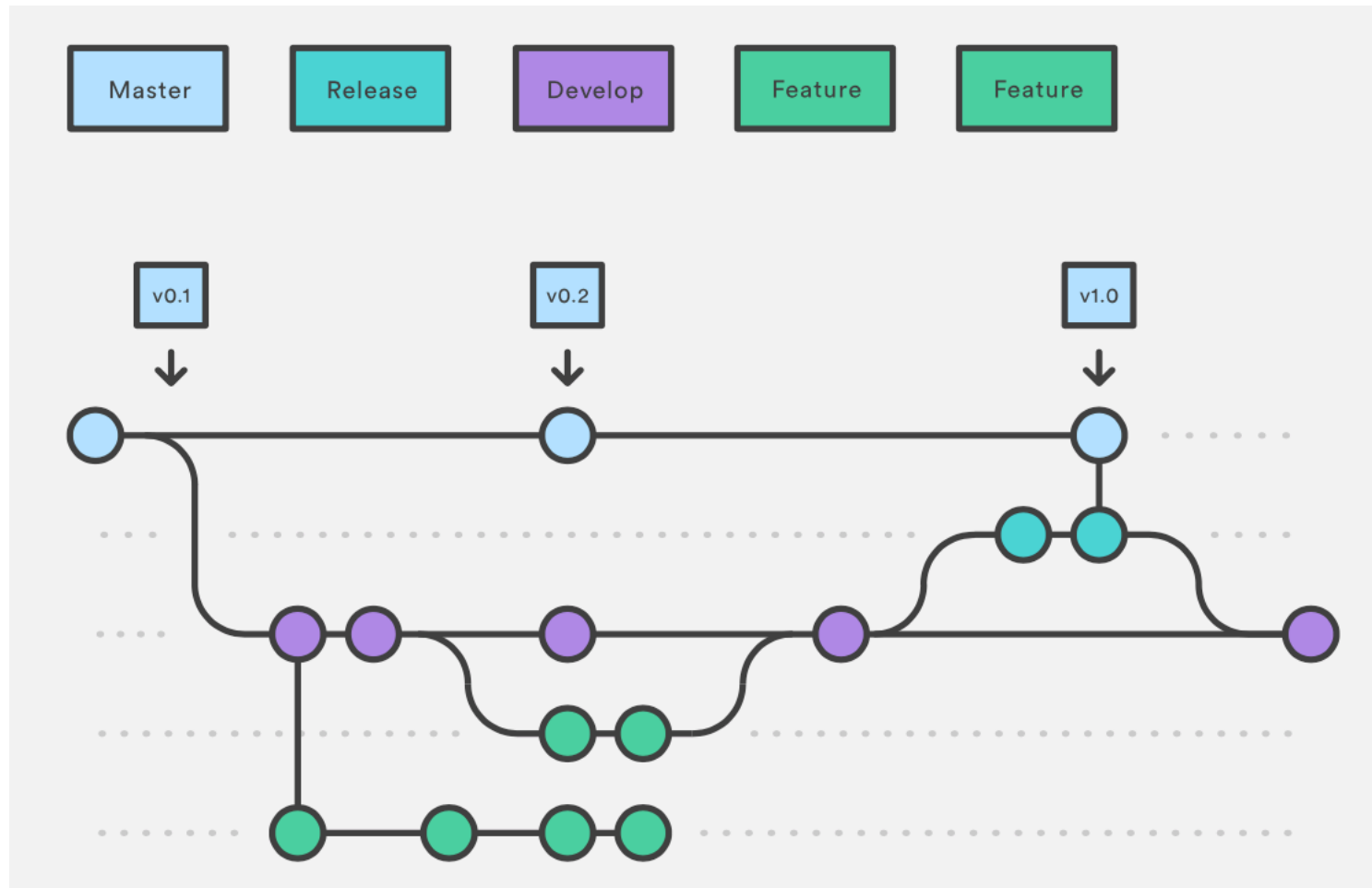
- Strict branching model designed around the project release
- Uses two+ branches
  - master stores the official release history; tag all commits in the master branch with a version number
  - develop serves as an integration branch for features



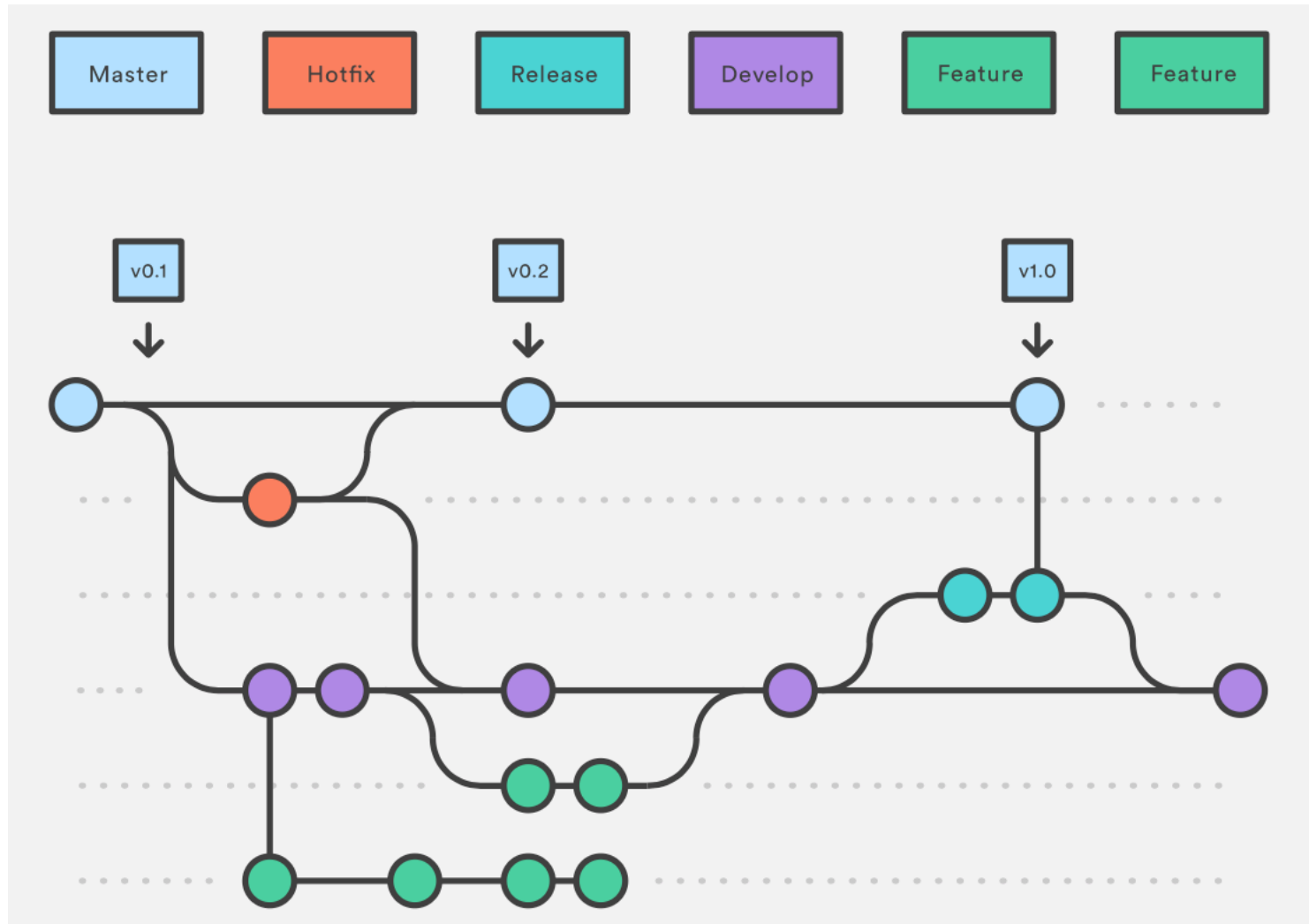
# GitFlow feature branches (from develop)



# GitFlow release branches (eventually into master)



# GitFlow hotfix branches

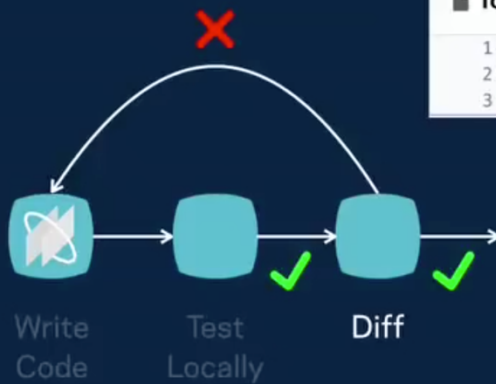


# Today

- Practical Git
- Common workflows using Git
- Developing at scale

# Pre-2017 release management model at Facebook

# Diff lifecycle: local testing

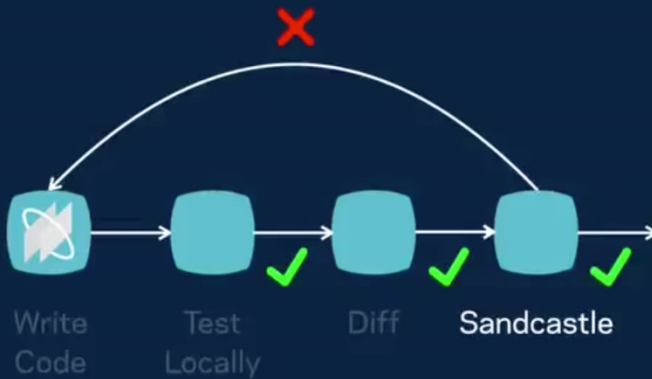


```
Tools/xctool/xctool/xctool/Version.m View Options ▼  
1 #import "Version.h"  
2  
3 NSString * const XCToolVersionString = @"0.2.1";  
1 #import "Version.h"  
2  
3 NSString * const XCToolVersionString = @"0.2.2";
```

```
PASS ExampleTest (0.050s)  
.  
OK (1 test, 4 assertions)  
OK  
(1 tests, 4 assertions, 0 incomplete, 0 failures)
```

Test and lint locally

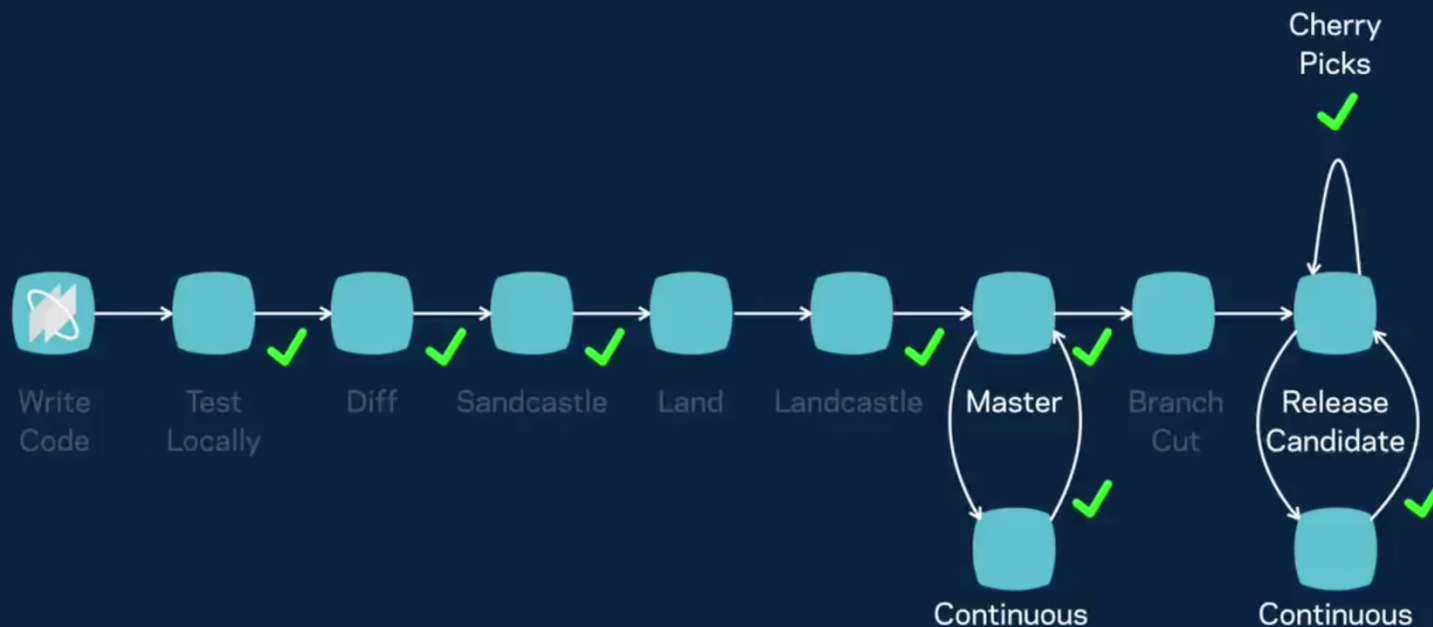
# Diff lifecycle: CI testing (data center)



	Facebook	Messenger	Groups	...
arm	✓	✓	✓	✓
x86	✓	✓	✓	✓
...	✓	✓	✓	✓

App and Build  
Configuration Matrix

# Diff lifecycle: diff ends up on master

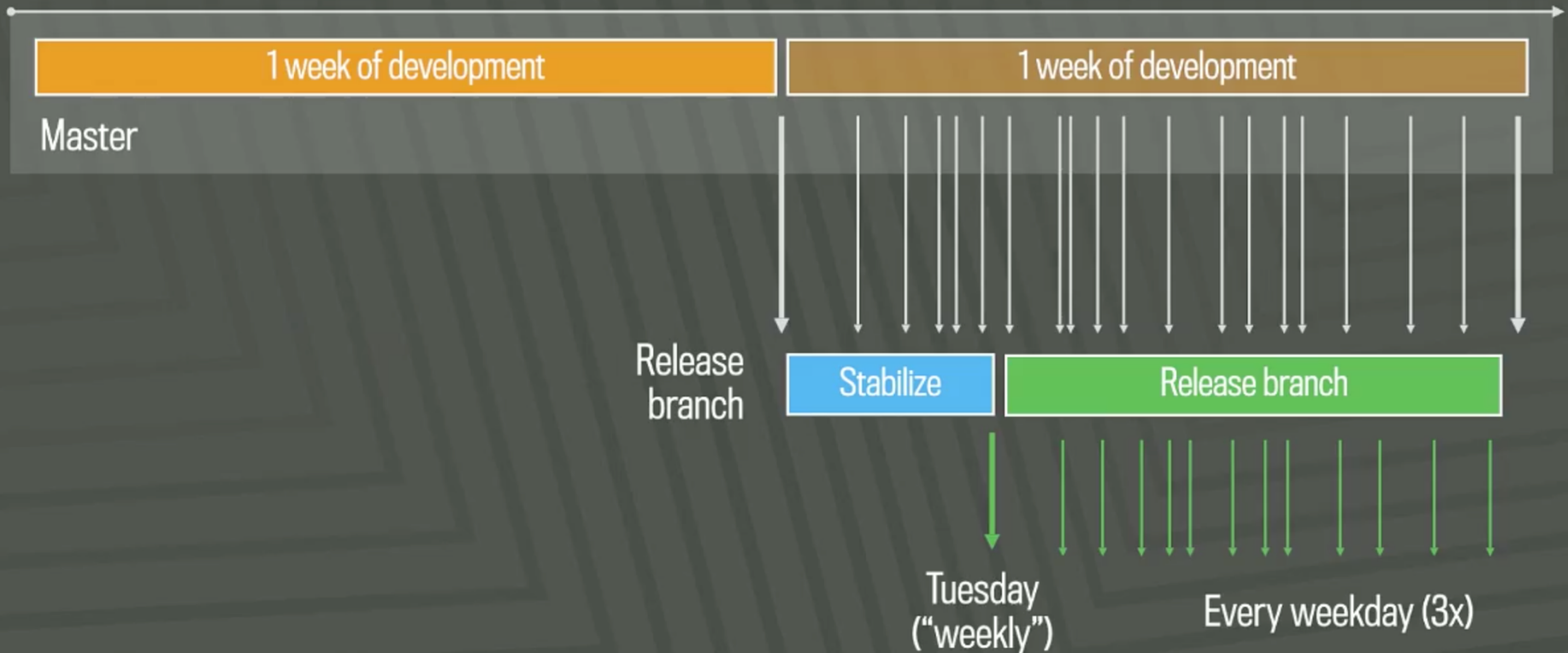


Dogfooding

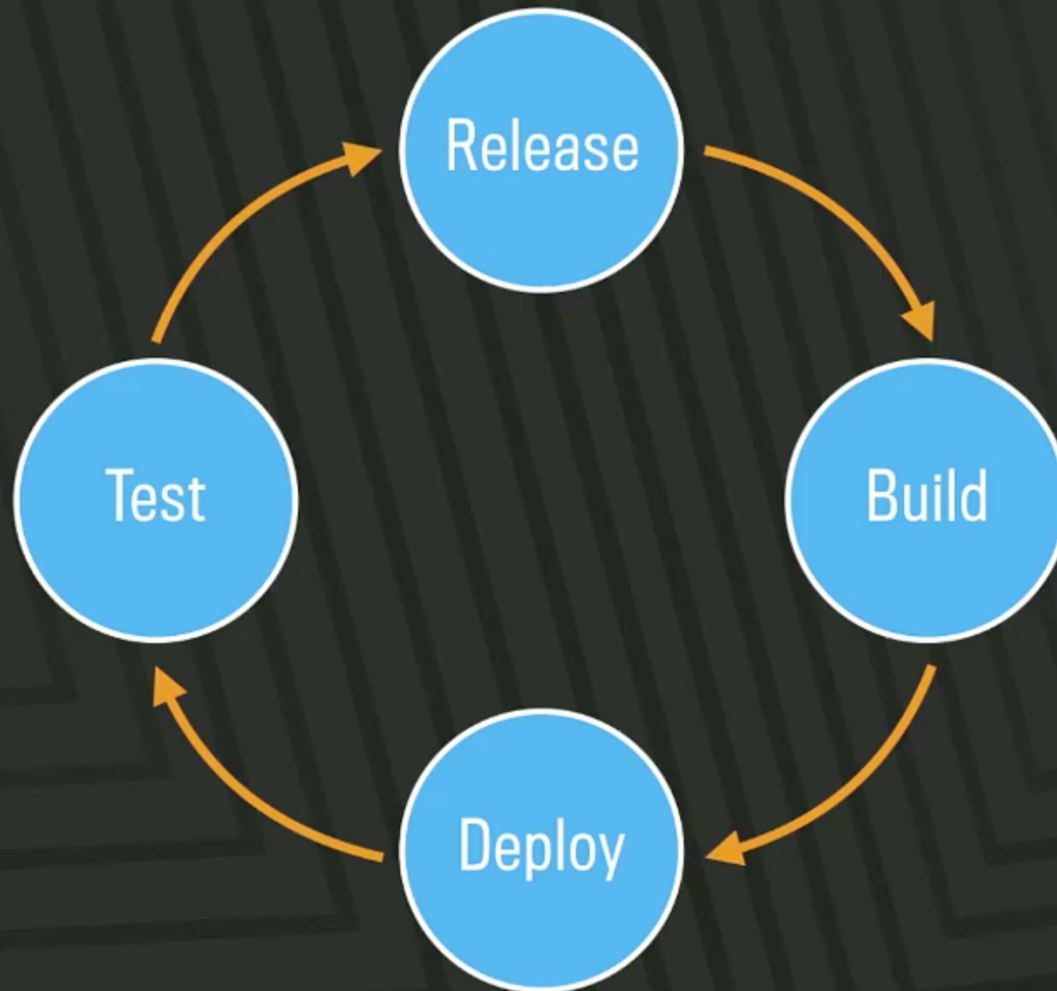


# Release every two weeks

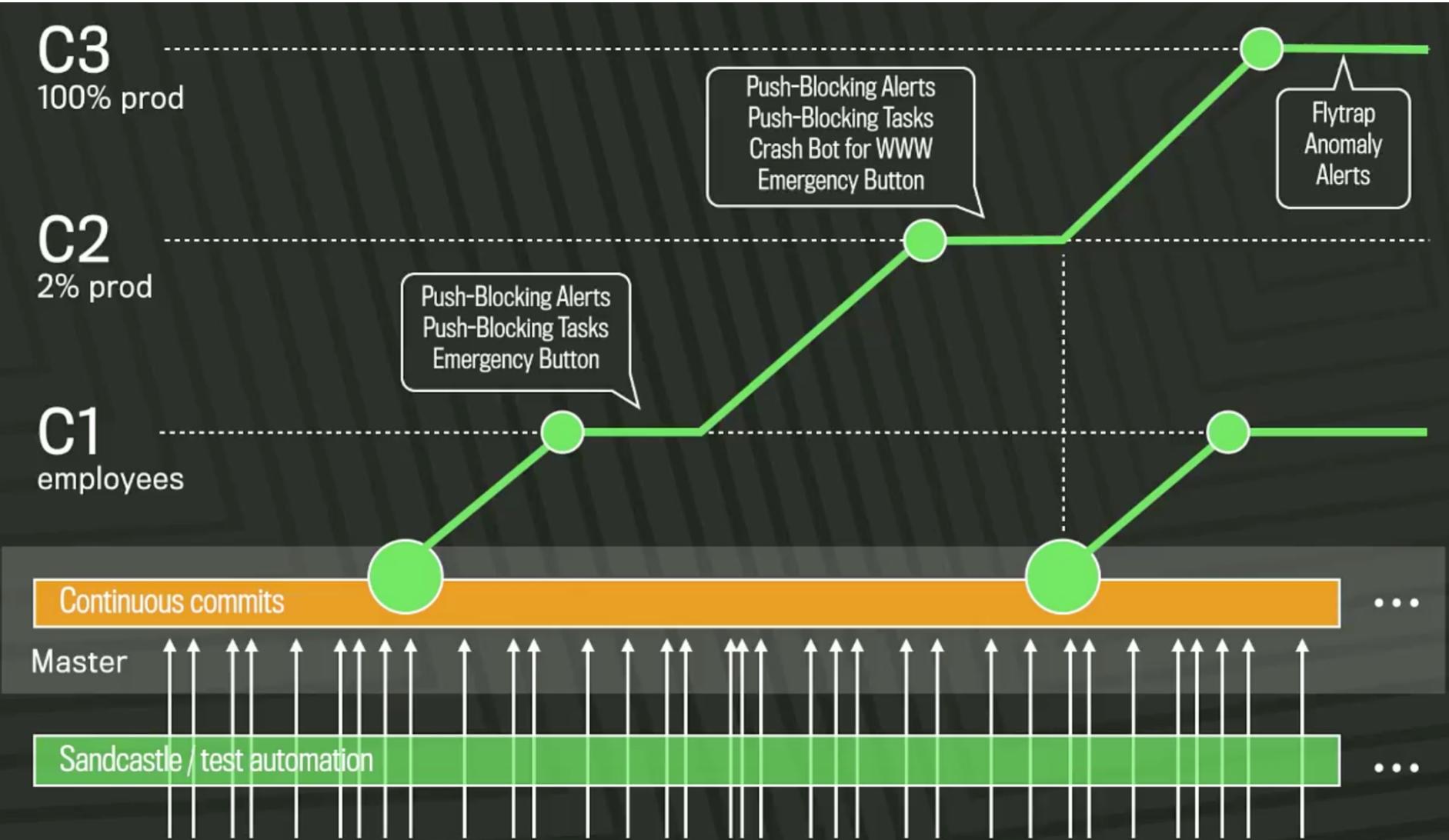
**www.facebook.com**



## Quasi-continuous web release



# Quasi-continuous push from master (1,000+ devs, 1,000 diffs/day); 10 pushes/day



# Google: similar story, huge code base

## Google repository statistics

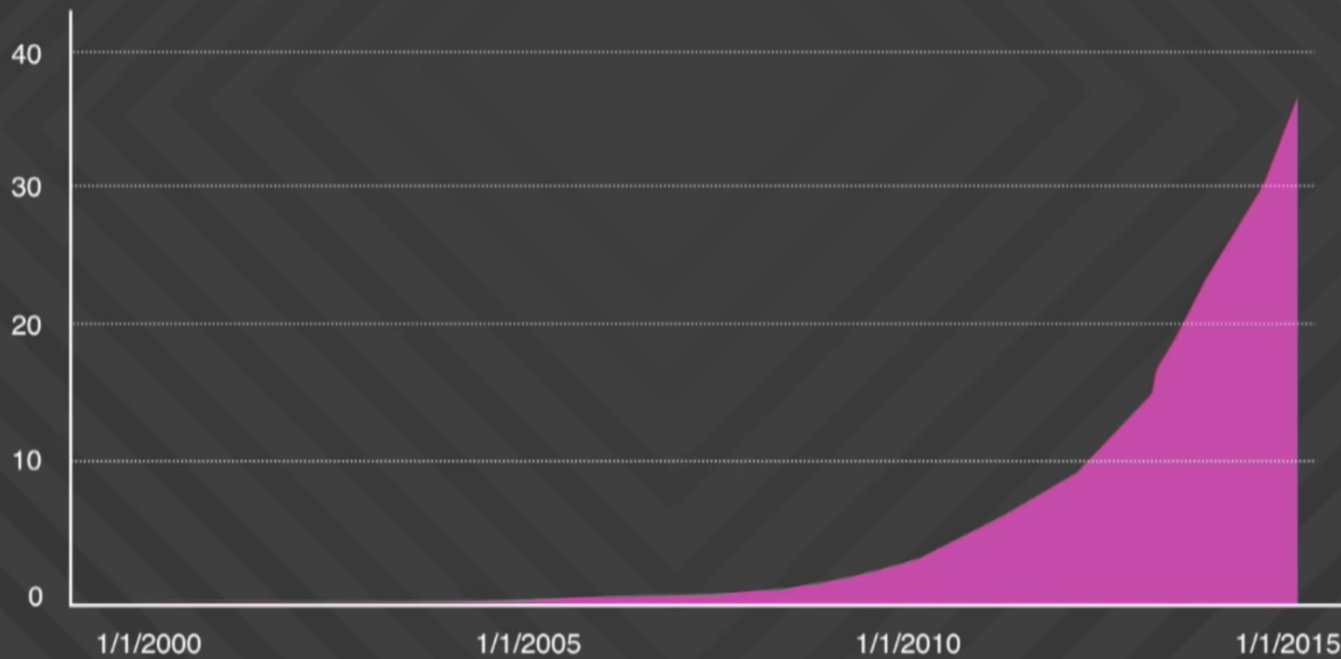
As of Jan 2015

Total number of files*	1 billion
Number of source files	9 million
Lines of code	2 billion
Depth of history	35 million commits
Size of content	86 terabytes
Commits per workday	45 thousand

\*The total number of files includes source files copied into release branches, files that are deleted at the latest revision, configuration files, documentation, and supporting data files.

# Exponential growth?

## Millions of changes committed (cumulative)



# Google Speed and Scale

- >30,000 developers in 40+ offices
  - 13,000+ projects under active development
  - 30k submissions per day (1 every 3 seconds)
- 
- All builds from source
  - 30+ sustained code changes per minute with 90+ peaks
  - 50% of code changes monthly
  - 150+ million test cases / day, > 150 years of test / day
  - Supports continuous deployment for all Google teams!



# Google code base vs. Linux kernel code base

## Some perspective

### Linux kernel

- 15 million lines of code in 40 thousand files (total)

### Google repository

- 15 million lines of code in 250 thousand files *changed per week, by humans*
- 2 billion lines of code, in 9 million source files (total)

# Managing a huge monorepo

- Automated testing...
- Lots of automation...
- Smart tooling...

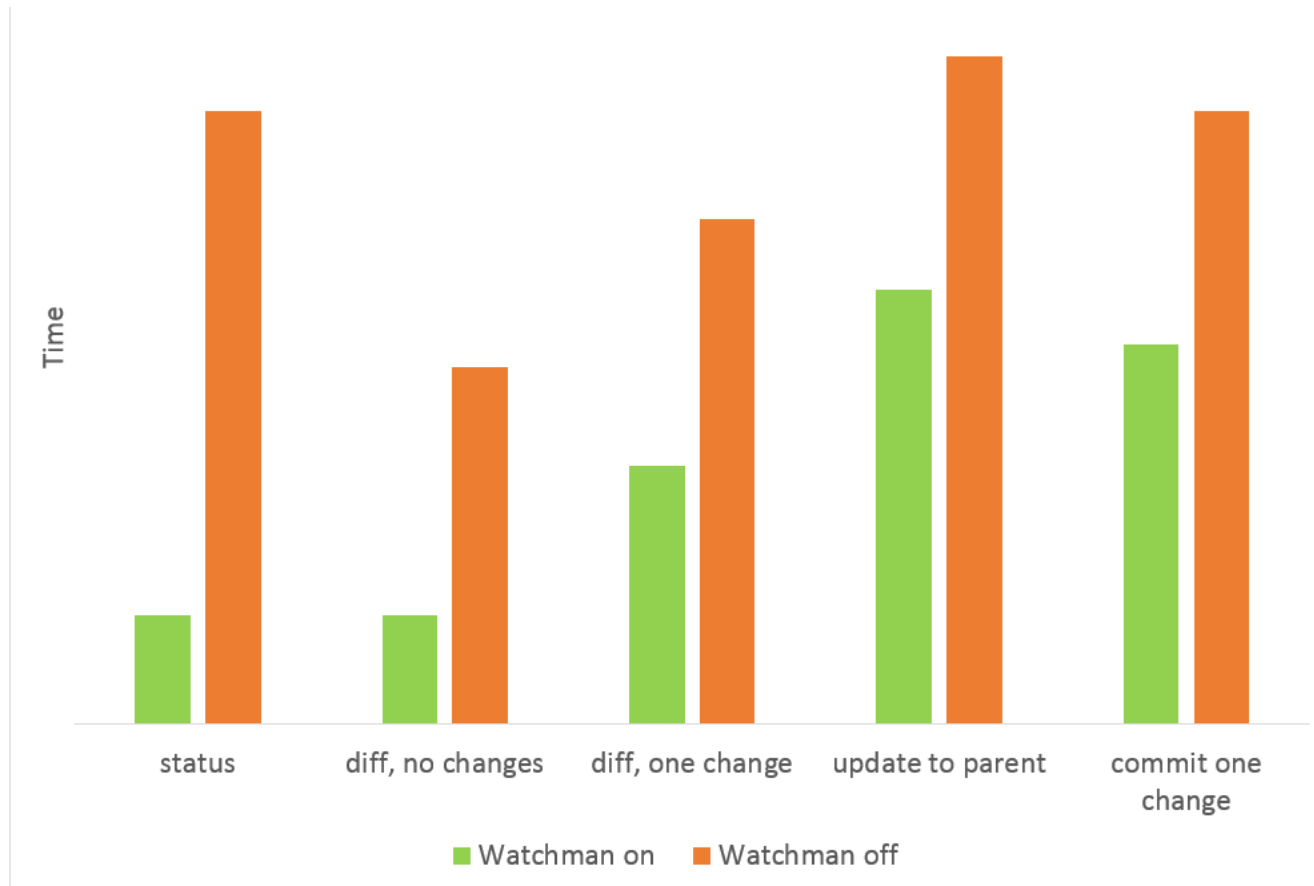


# Version control for a monorepo

- Problem: even git is slow at Facebook scale
  - 1M+ source control commands run per day
  - 100K+ commits per week

# Version control for a monorepo

- Use build system's file monitor, Watchman, to see which files have changed → **5x faster “status” command**



# Version control for a monorepo

- Sparse checkouts → **10x faster clones and pulls**
  - `clone` and `pull` download only the commit metadata, omit the files
  - When a user performs an operation that needs the contents of files (such as `checkout`), download the file contents on demand

