

Lecture 05 – Tools of the Trade

Prof. Brendan Kochunas

NERS/ENGR 570 - Methods and Practice of Scientific Computing (F20)



COLLEGE OF ENGINEERING
NUCLEAR ENGINEERING & RADIOLOGICAL SCIENCES
UNIVERSITY OF MICHIGAN

Outline

- Configure Tools & Infrastructure
- Version Control w/ Git
- Class exercise with Git

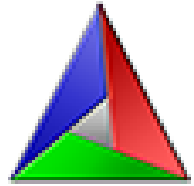
Learning Objectives: By the end of Today's Lecture you should be able to

- *(Skill)* Configure and build a CMake based project
- *(Value)* Understand why we should do version control
- *(Knowledge)* Basic concepts in git
- *(Skill)* Working on a distributed software project using git and Make.



Infrastructure Tools

Now you know how to track changes
What's next?



CMake <https://cmake.org/>

What is it?

- Open source
- Cross platform (truly)
 - Supports multiple build tools, toolchains, & environments.
- Includes
 - CMake – configuration tool
 - CTest – testing tool
 - automates testing and collection and publishing of test results
 - CPack – generates installers
 - CDash – Web interface for viewing test results

Software Projects Using CMake

- CGAL
- Geant4
- GROMACS
- Trilinos
- VTK and Paraview
- zlib
- LAPACK
- HDF5
- Netflix

(GNU) Make

- Likely the most popular tool for defining executables and compiling software in Linux
 - Alternatives: Ninja, Tup, Gulp
- Standard in Linux/Unix
- CMake generates makefiles for you
 - It can also generate inputs for other build tools
 - Gives you fancy features
 - parallel, percentage complete, default targets, help, coloring.

- Makefiles must define targets using rules
 - Can also define variables, use include

```
target ... : prerequisites ...  
           recipe  
           ...
```

```
edit : main.o kbd.o command.o display.o \  
      insert.o search.o files.o utils.o  
      cc -o edit main.o kbd.o command.o display.o \  
      insert.o search.o files.o utils.o  
  
main.o : main.c defs.h  
      cc -c main.c  
  
...  
  
clean :  
      rm edit main.o kbd.o command.o display.o \  
      insert.o search.o files.o utils.o
```

Examples from GNU Make manual: <https://www.gnu.org/software/make/manual/make.html>



CMake Example/Demo



Version Control

What is Version Control?

- A way of tracking detailed changes in source code over time.
 - e.g. what changed? when did it change? who changed it?
- Version control is an **essential** component to software development.
 - Has been used by software developers for decades
- Version control is generally performed using an external program.
- Version control is applied to a “repository”.
 - Source code lives in one (or more) *repositories* (e.g. repos) available to team members/contributors.
- A repo is a computational scientist’s laboratory notebook.
 - Like a laboratory notebook, it is only useful if it is used properly.
- Also known as: “source code control”, “revision control”, “source code versioning”

Slide source: Best Practices for HPC Software Developers – Session 3: Distributed Version Control and Continuous Integration Testing

What version control does

- Establishes a comment context for code contributions and the exchange of ideas
- Establishes a chronological sequence of events
 - A single change is commonly referred to as a *commit*
- Serves as “truth” for a software project
- If you don’t have a “tangible” common reference for your source code, *there is nothing for your team to discuss.*
- Results from uncontrolled code are (generally) *not reproducible.*
- Recall your most frustrating document-sharing experience...
 - ...and imagine it continuing for months... or years with the people involved changing and the document getting larger and larger.
 - (it doesn’t work)

Utility of Version control

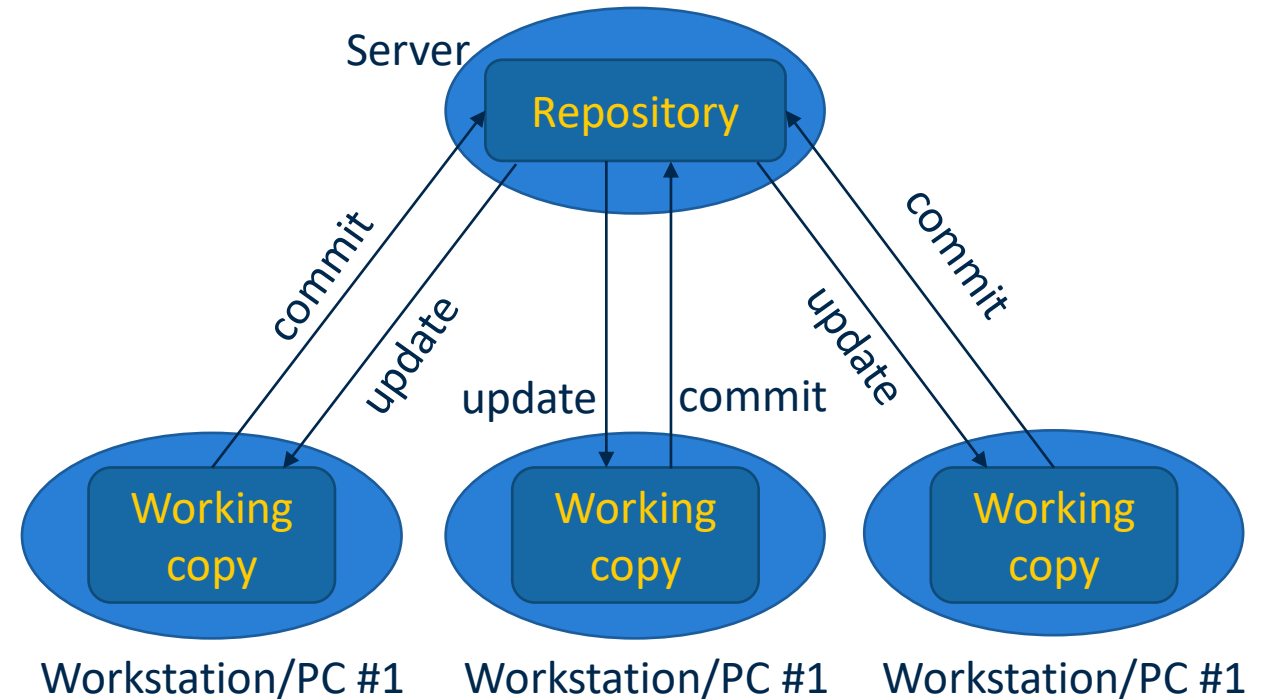
- What if I program by myself?
 - Still offers same advantages.
 - Try to remember the steps you took to complete a homework assignment in high school.
 - Working on different machines is no problem.
 - If you eventually want to collaborate with someone then there's no extra work!
- Can I use version control for other things?
 - YES!
 - Most version control programs are excellent It's great for text files. Working with binary files, its not so good.
 - You may not think its worth the extra effort now, but your future self will thank you.
 - Commonly used for LaTeX documents.

Version Control and Versioning

- Version control allows you to define and publish a specific version of the code. (e.g. one specific commit)
 - A popular description for how to describe a version and what it means is defined here:
 - <http://semver.org/>
 - Short for “Semantic Versioning”
- Short Definition from**
<http://semver.org/>
- Given a version number MAJOR.MINOR.PATCH, increment the:
 - MAJOR version when you make incompatible API changes,
 - MINOR version when you add functionality in a backwards-compatible manner, and
 - PATCH version when you make backwards-compatible bug fixes.
 - Additional labels for pre-release and build metadata are available as extensions to the MAJOR.MINOR.PATCH format.

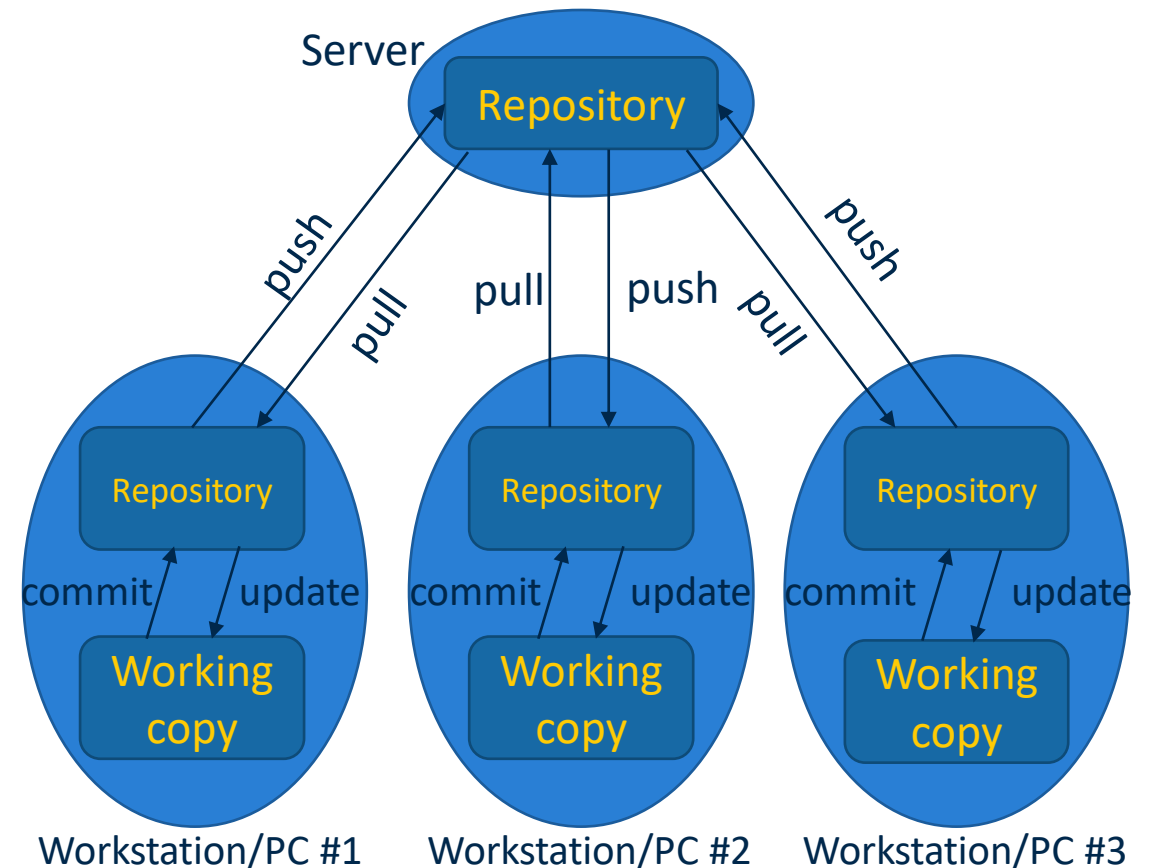
Models for Version Control Programs (1)

- Simple: “Centralized” Version Control
 - There is one repository containing the “master” version (or “trunk”) of the source code
 - Everyone syncs with this repository
 - e.g. *checks out* files, *changes* them, and *commits* these changes.
 - Requires that people must cooperate/coordinate to make sure their changes don’t conflict with each other.
 - Limited in capability to create development branches.



Models for Version Control Programs (2)

- Modern: “Distributed” Version Control
 - Everyone has an entire copy of the repository (and history!)
 - There is a “main” repo agreed upon by convention
 - People typically work in development branches with isolated changes until ready to merge
 - Allows for more flexibility for design and development procedure



Open Source Tools for Version Control

Centralized Version Control

- Concurrent Version System (CVS)
 - One of the first. Don't recommend you start here.
 - Should not need to work with it
 - There are many tools to migrate a CVS repository to a newer one.
- Subversion (SVN)
 - Modern successor to CVS
 - Supports branching

Distributed Version Control

- git
 - Probably the most popular version control program
 - Written by same person who wrote linux.
 - Several programs built around git offering different interfaces
 - gitlab, github, gitorious
- Mercurial (hg)
 - favors ease of use
 - syntax similar to subversion

Version Control Disclaimer

- It's a tool, and its only as good as its user.
- It does not define a *development process*
- Up to you to choose an approach that best works for you and your team
- Identifying and using good software development processes is hard.
 - That's why we'll talk about it in a future lecture

<http://xkcd.com/1296/>



	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

A little about git

- Initially released in 2005
 - original author was Linus Torvalds (the linux guy)
- Difficult to learn without putting in some time
 - That's what the demo is for!
- It is difficult to learn without understanding the underlying concepts.
- Teams that use git well often have an expert.

<https://xkcd.com/1597/>



Common Nomenclature in Git

- *Repos* – repositories. the full history of the project
- *Clones/cloning* of repos – making a copy of
- *Commits/committing* within repos – making code changes
- *Branches/branching* within repos – isolated development
- *Remotes* – references to other repos on other machines
- *Pulls/pulling* – incorporating changes from another repo to your local repo
- *Pushes/pushing* – publishing changes from your local repo to another repo
- *Revisions* – also known as commits, can be referred to by a the SHA-1 (e.g. 1e95a651f4aeea1aa00173347f254dfb93ae350a)
- *Workspace* – local state
- *History* – the graph, specifically a directed-acyclic-graph (DAG)

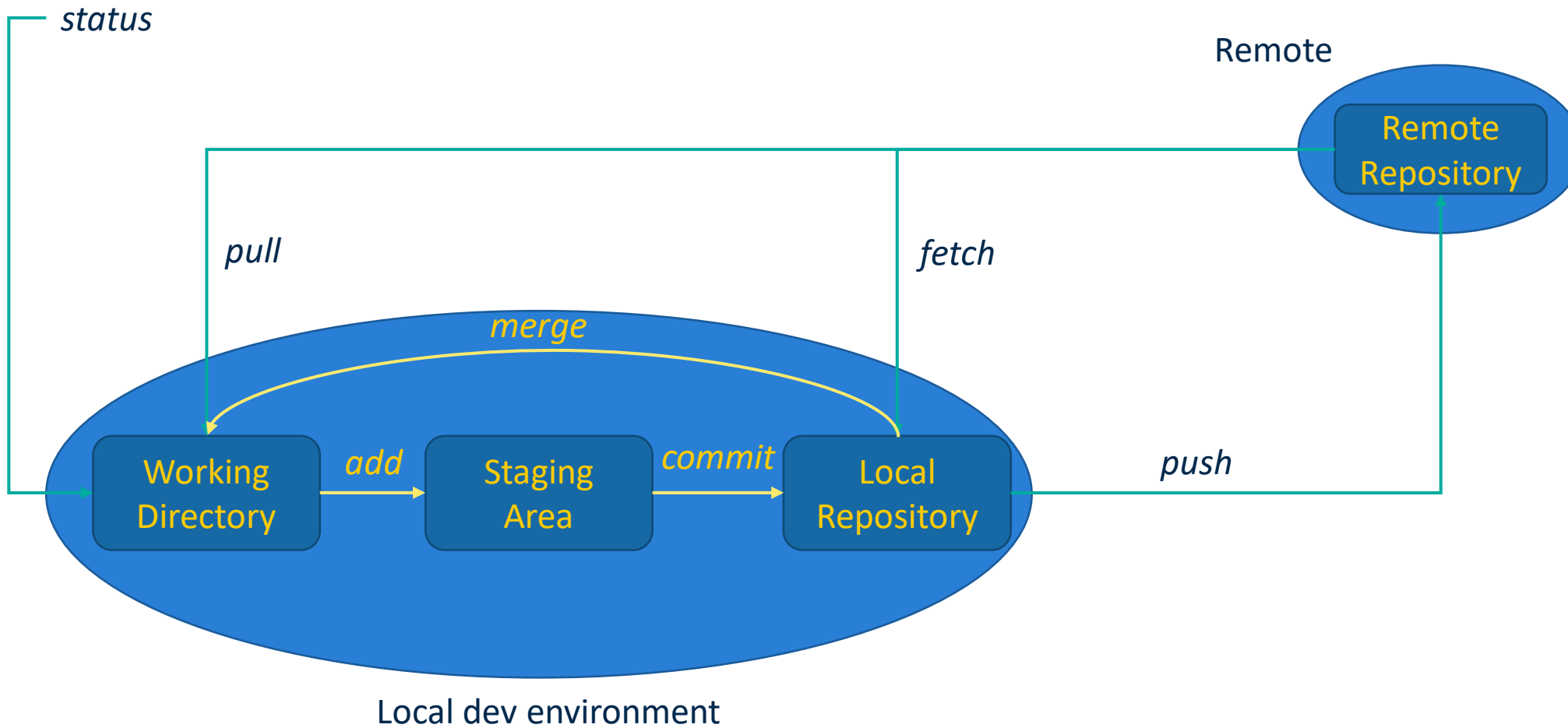


Git Concepts

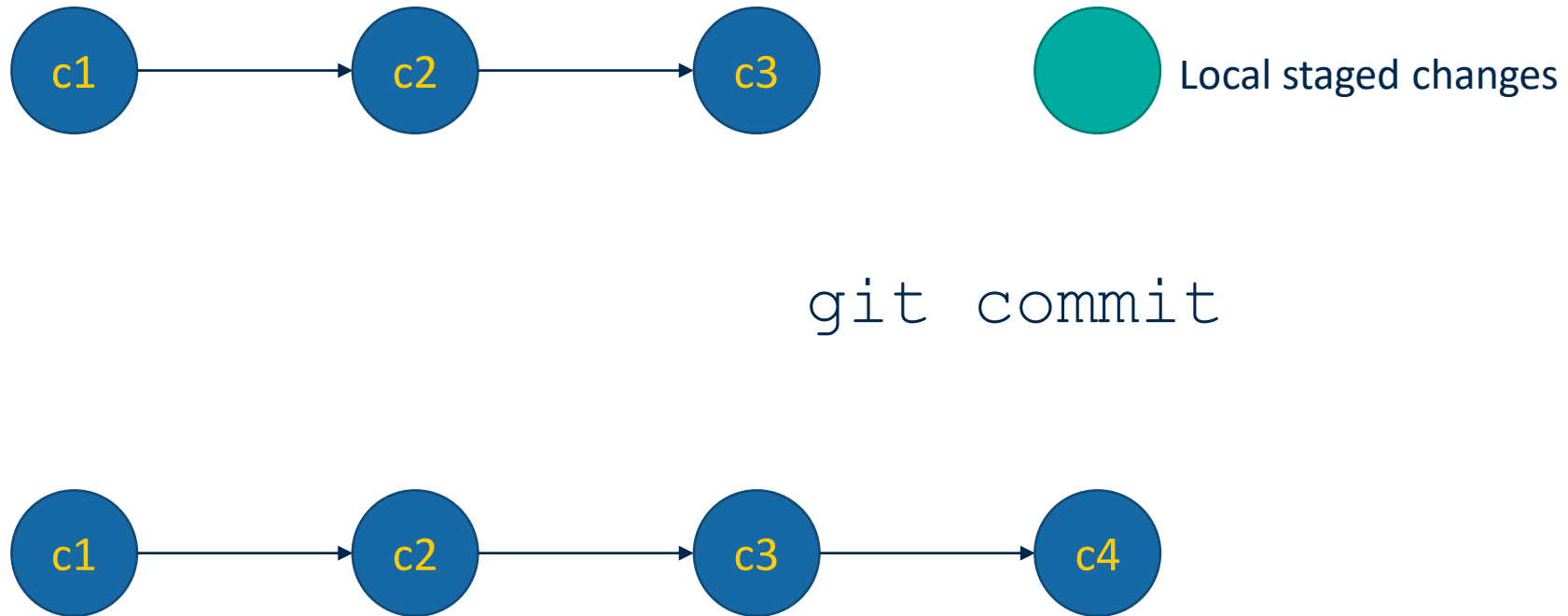
<https://learngitbranching.js.org/>

<https://ideas-productivity.org/resources/howtos/git-tutorial-and-reference-collection/>

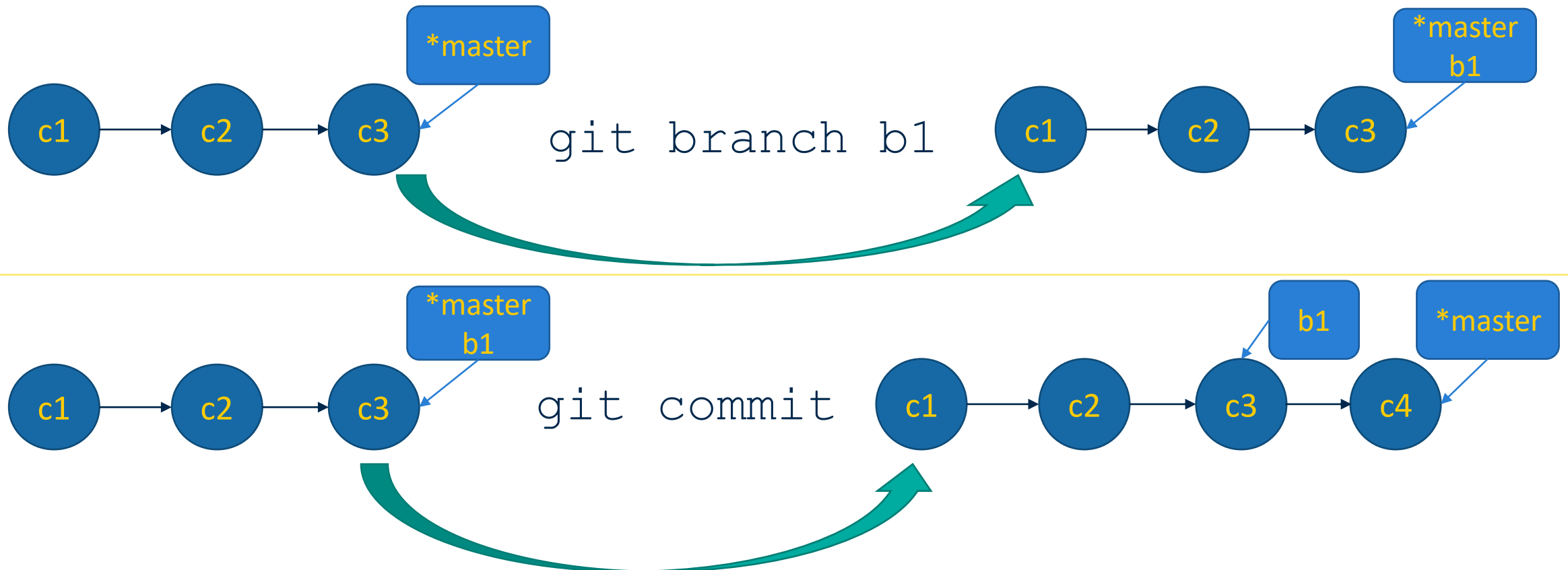
Overview of Git Concepts



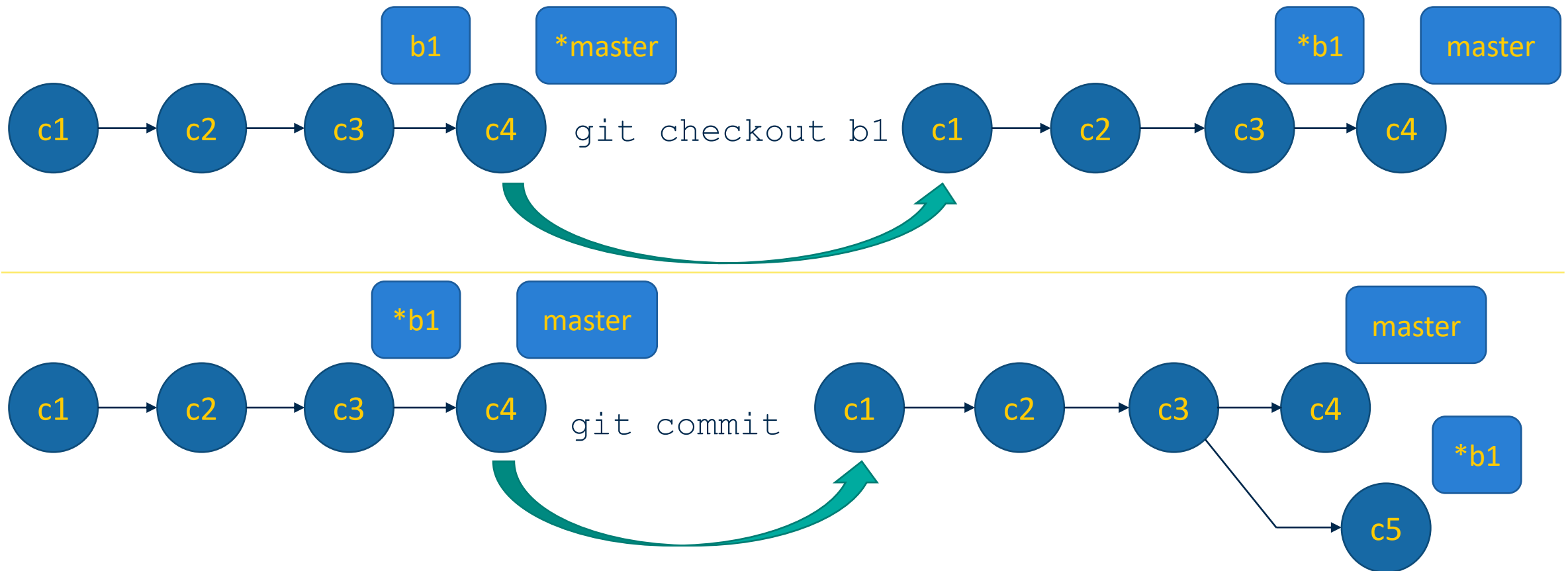
git concepts: commits



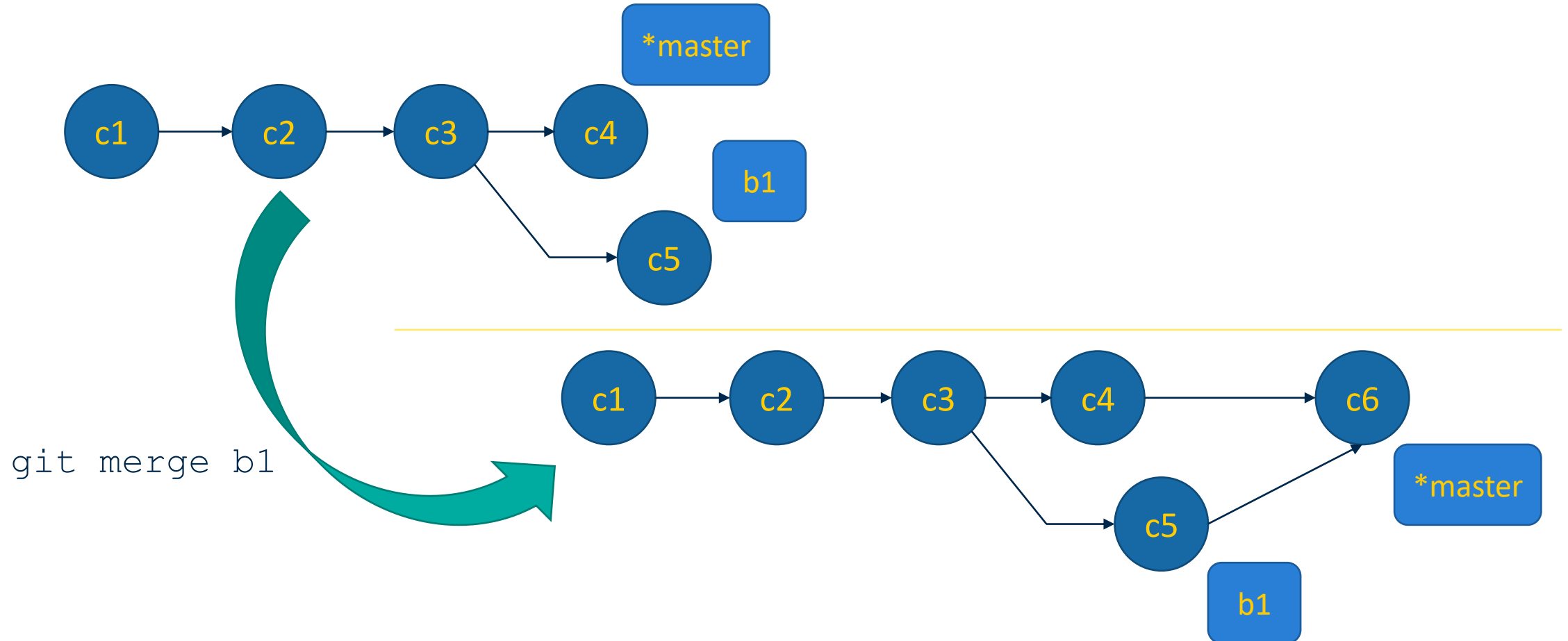
git concepts: branches (1) – creating a branch



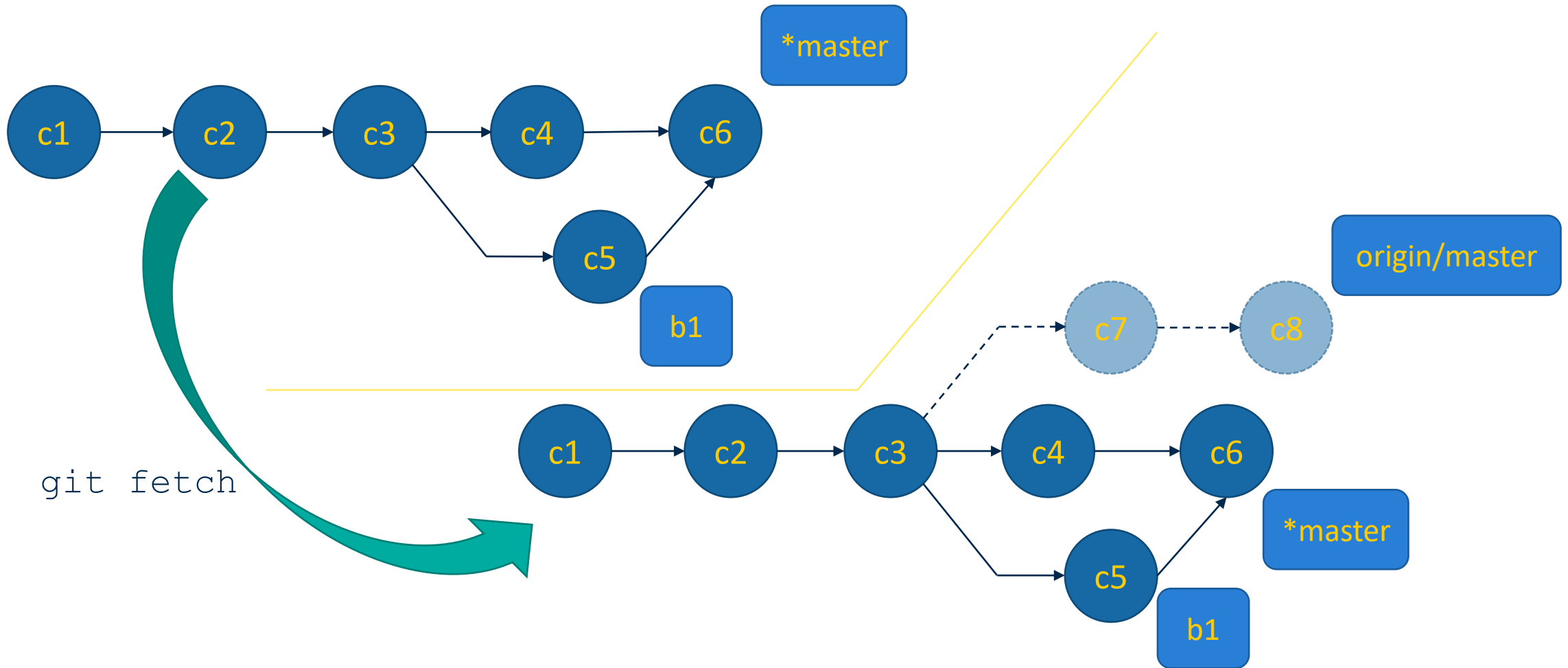
git concepts: branches (2) – working on a branch



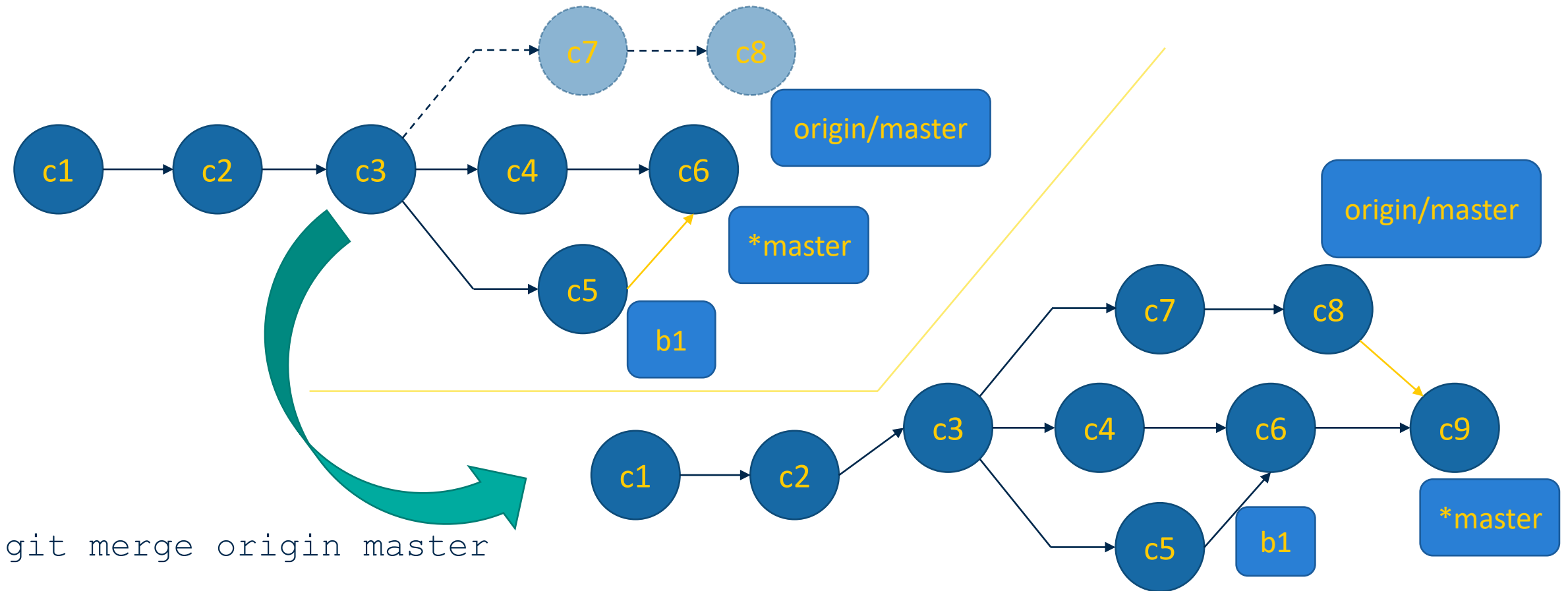
git concepts: merge



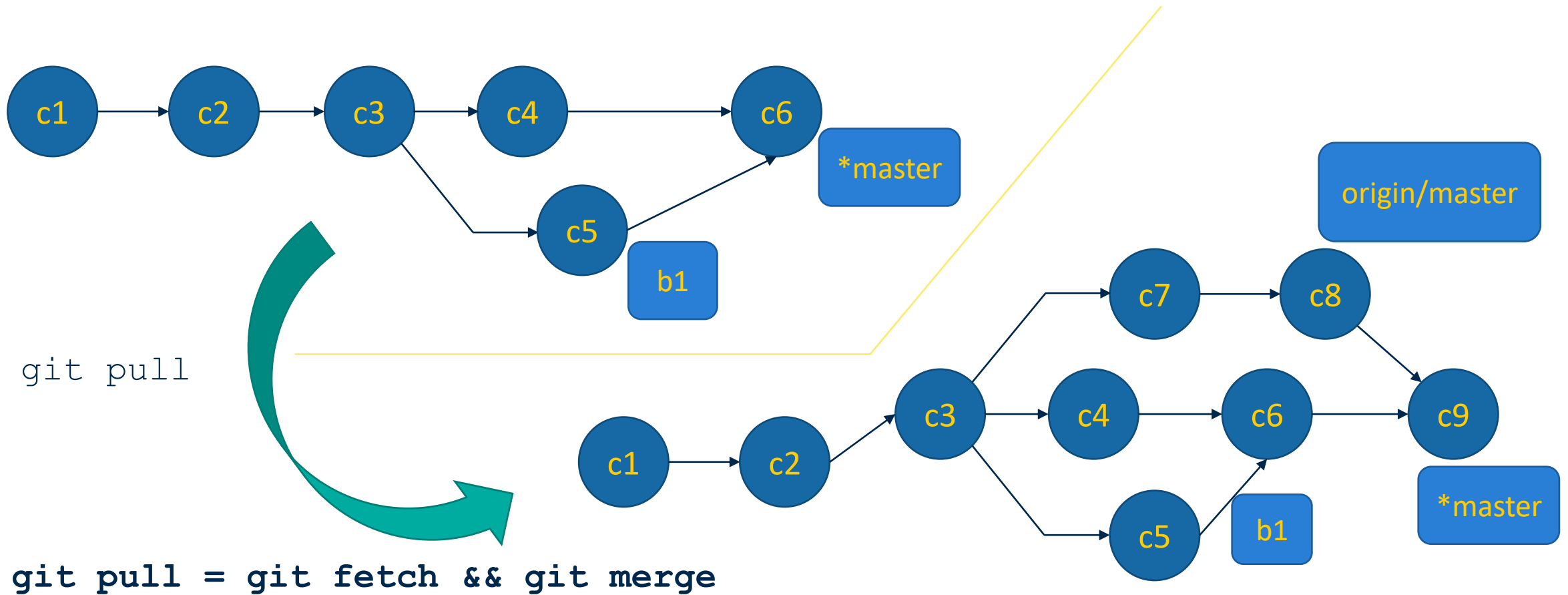
git concepts: fetch



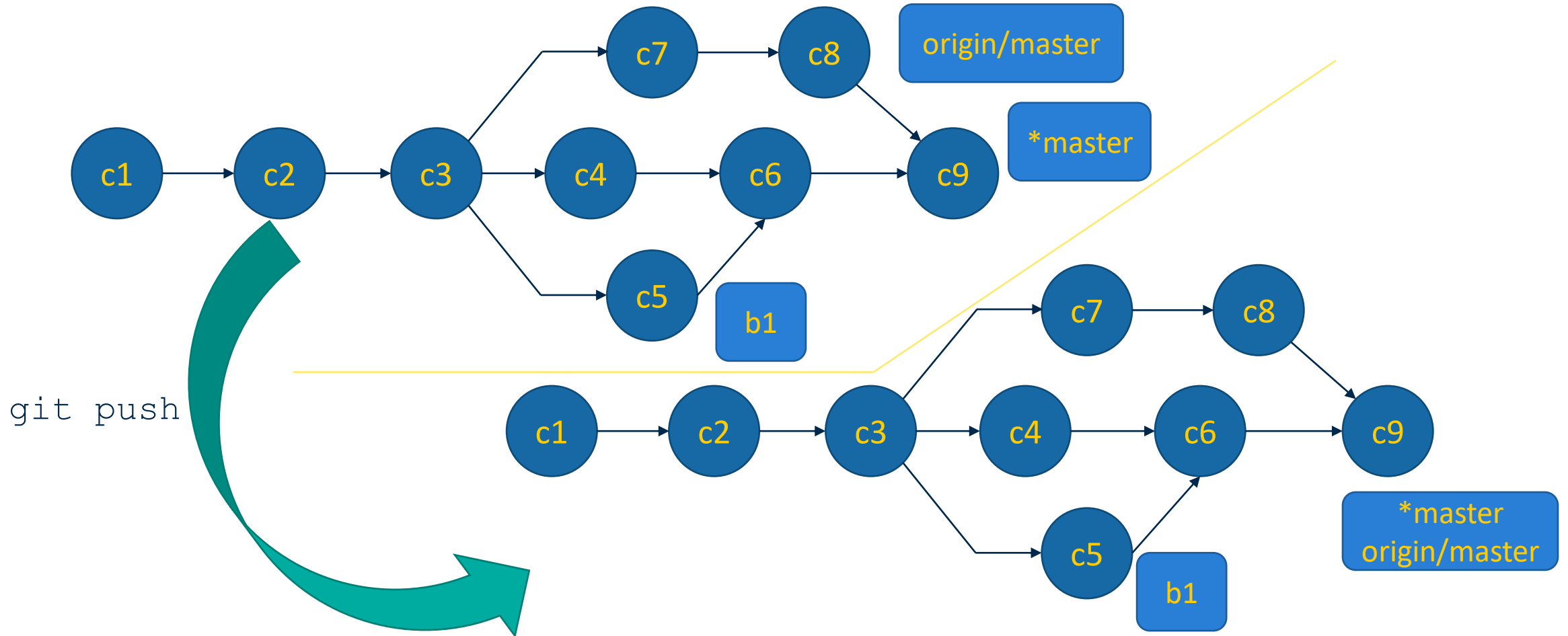
git concepts: fetch (merge w/ remote)



git concepts: pull

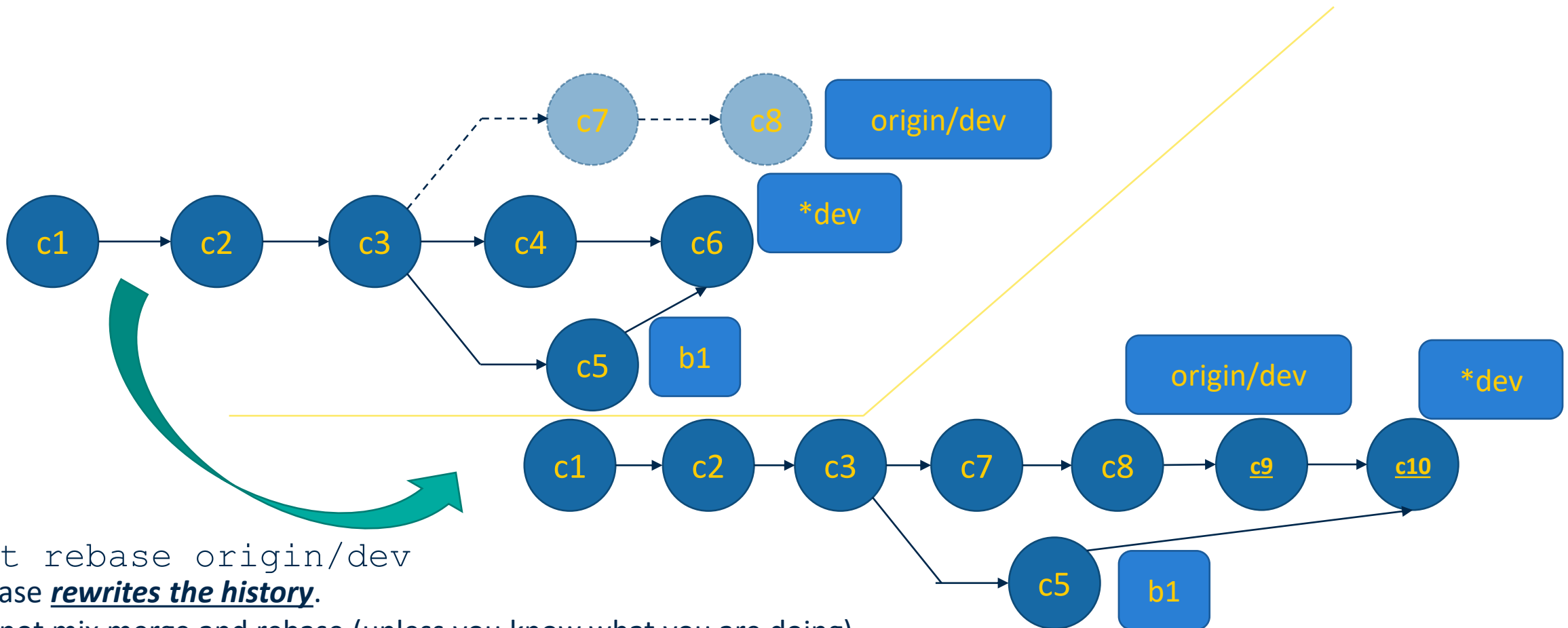


git concepts: push



git concepts: rebase

(a dangerous, but sometimes useful alternative to merge)



git rebase origin/dev
rebase **rewrites the history**.

Do not mix merge and rebase (unless you know what you are doing)