THE UNDERGRADUATE
ARTIFICIAL INTELLIGENCE SOCIETY

UAIS

INTRO TO GIT & GITHUB

git | GitHub

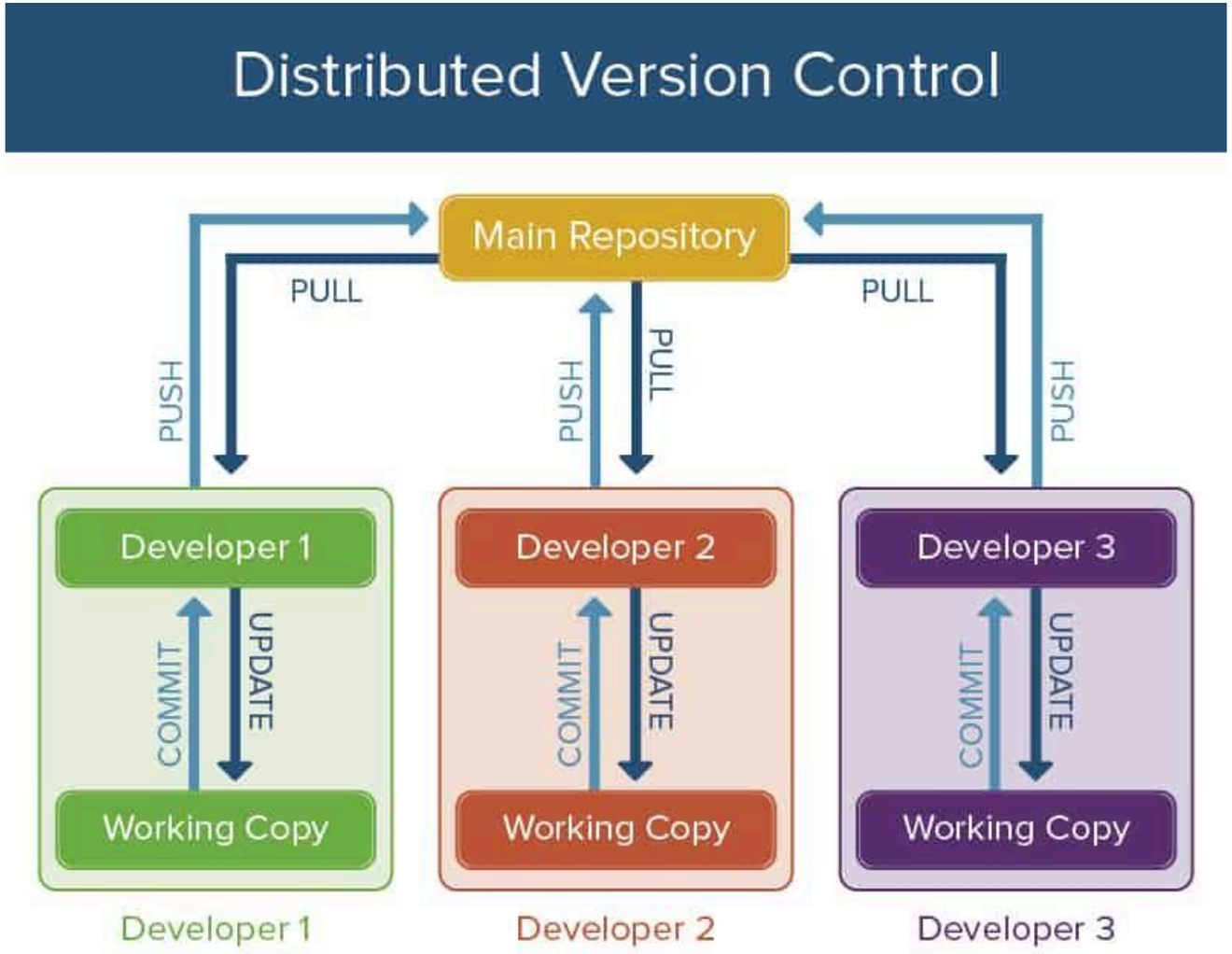
THE UNDERGRADUATE
ARTIFICIAL INTELLIGENCE SOCIETY

WHAT TO EXPECT

1. A conceptual introduction to:
    a. Version control
    b. Git's mental model (workspace → staging → repo)
2. A hands on workshop to:
    a. Practice core commands: clone, pull, push, branch, commit, merge, stash,
    b. Experience merge conflicts (and fix them!)
    c. Use GitHub for pull requests
    d. Use GitHub for issues

What is Version Control?

- Version control = a system that tracks changes to files over time.
- Lets you:
- Revert to previous versions
- See who changed what and when
- Like “Google Docs history” — but for code.



Git = Distributed Version Control

- Every user has a full copy of the repo.sitory

THE UNDERGRADUATE
ARTIFICIAL INTELLIGENCE SOCIETY

WHAT IS GITHUB?

- **GitHub is a cloud platform built on top of Git that makes collaboration easy.**

- Think of it as Google Drive for Git repositories, with:
  - Remote hosting of your repos
  - Issue tracking & project boards
  - Pull Requests (PRs) for proposing changes
  - Code reviews and discussions

THE UNDERGRADUATE
ARTIFICIAL INTELLIGENCE SOCIETY

GIT VS GITHUB

	
1. It is a software	1. It is a service
2. It is installed locally on the system	2. It is hosted on Web
3. It is a command line tool	3. It provides a graphical interface
4. It is a tool to manage different versions of edits, made to files in a git repository	4. It is a space to upload a copy of the **Git** repository
5. It provides functionalities like Version Control System Source Code Management	5. It provides functionalities of Git like VCS, Source Code Management as well as adding few of its own features

Source: https://www.abdulrahmanuk.com/day-8-task-basics-of-git-github/

KEY TERMS

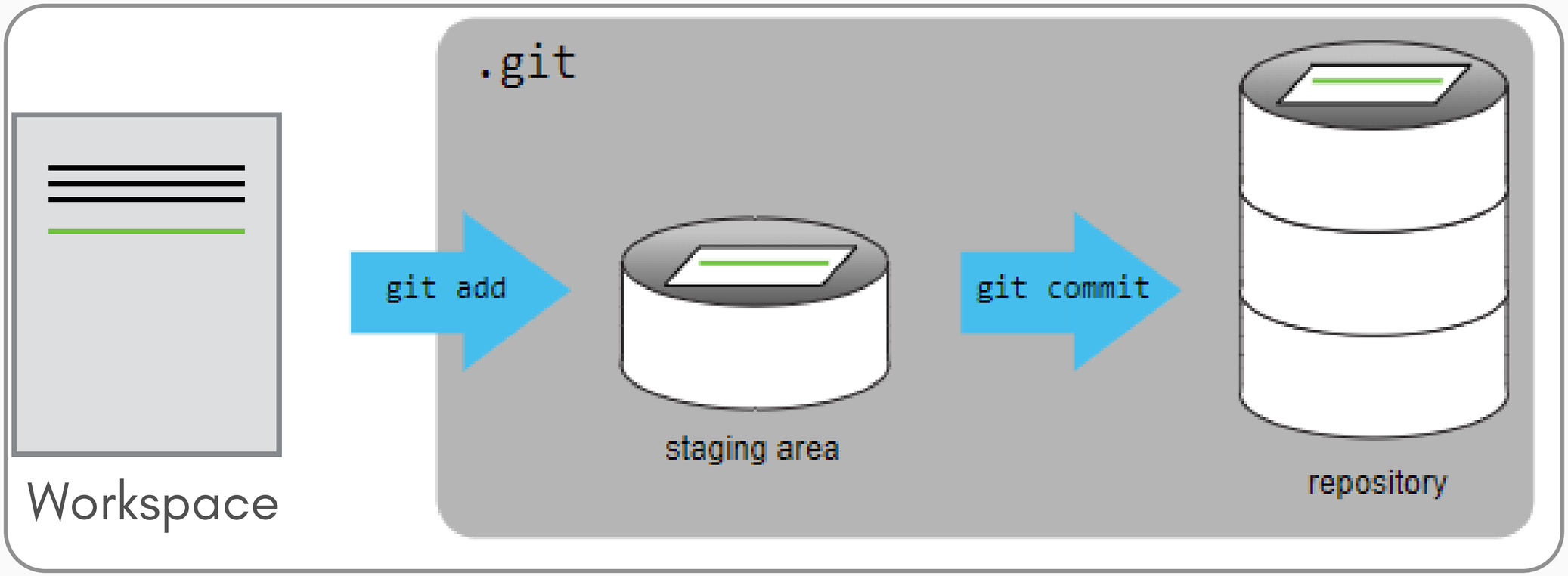
Repository (Repo)
- Collection of files and their history organized in folders, branches, tags
- Is stored in a normal file system somewhere (local or remote machine)

Commit
- A snapshot of your project at a moment in time
- Records what changed, when, and by whom
- Each commit has a unique ID and message

Workspace
- The folder on your computer where you make changes to files
- Cchanges here aren’t saved to Git yet

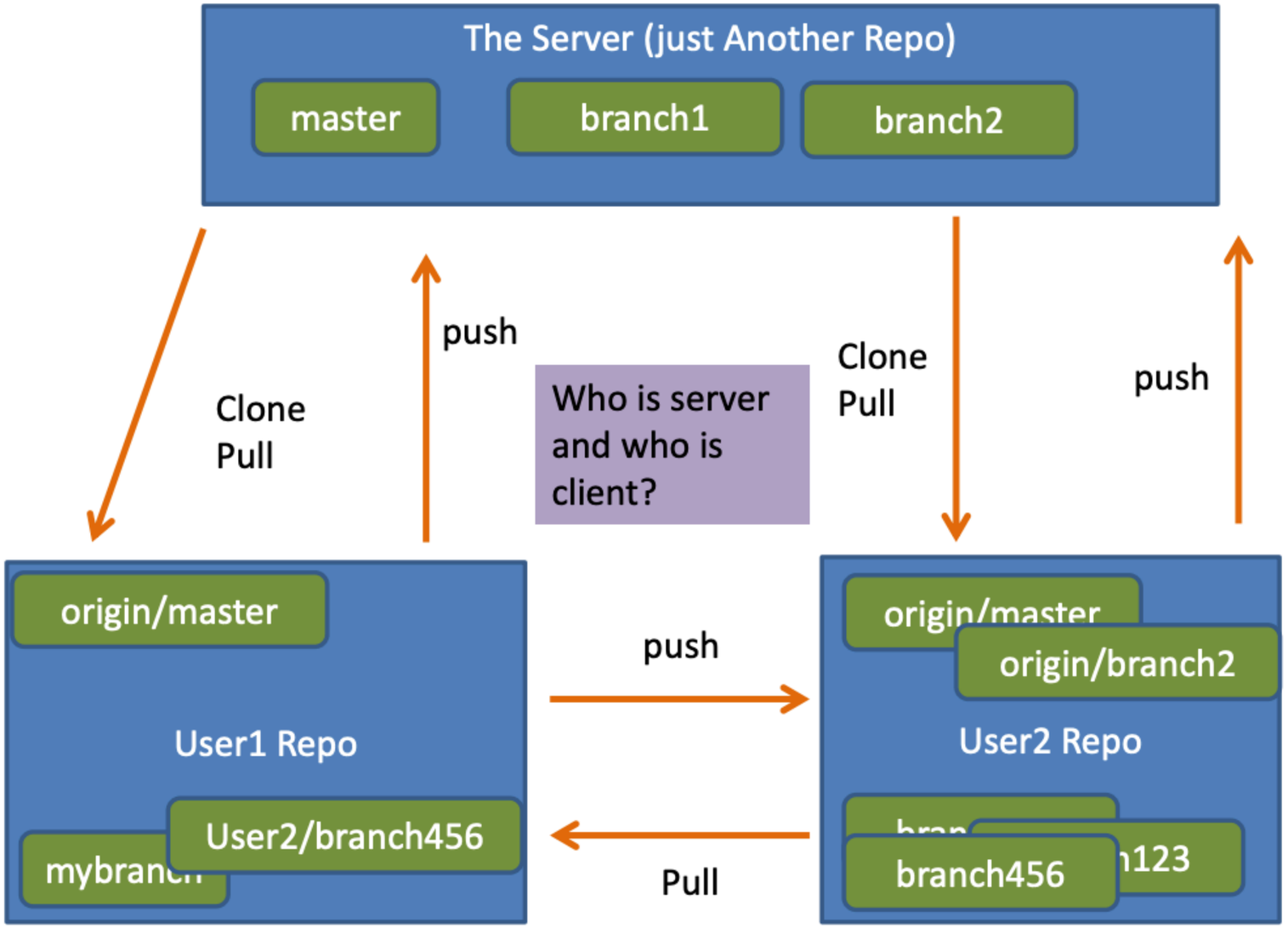
Index (Staging Area)
- A temporary snapshot of selected files in your workspace

GIT'S THREE AREAS



Source: https://github.com/swcarpentry/git-novice/issues/522

GIT Model



Source: JivyGroup 2014
yoad@jivygroup.com

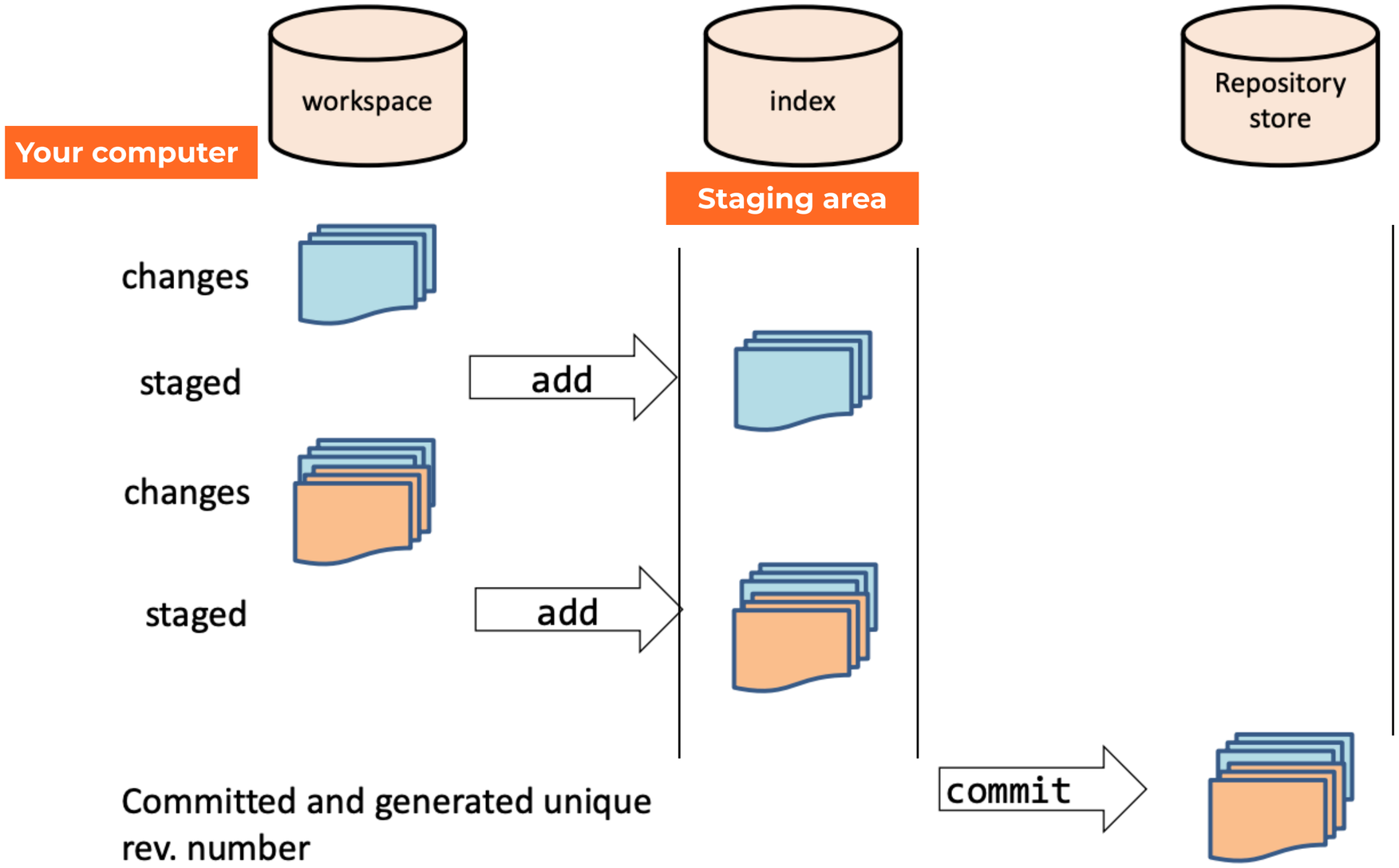
Local Git - Brief Intro

Single repository concepts

LOCAL

GIT Model

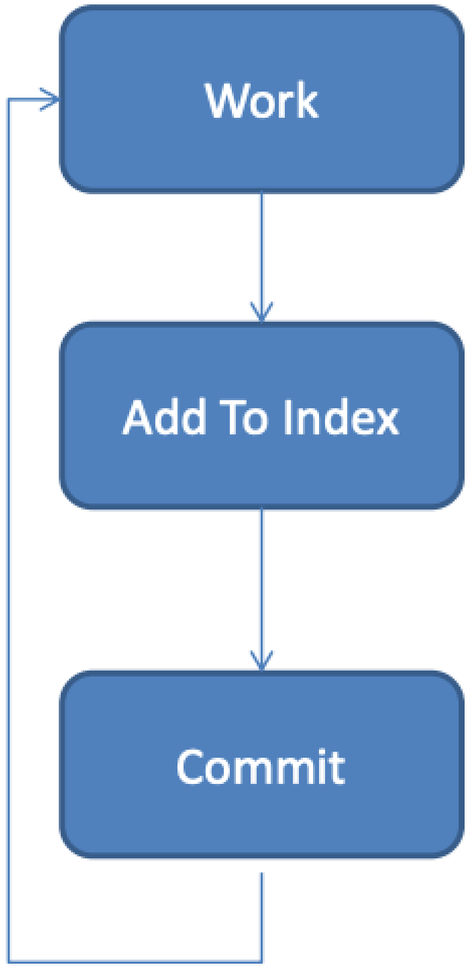
LOCAL



GIT Model

LOCAL

Simple Local Git



- Add / remove / edit files

- Add (same as “stage”) the changes
- Not all files must be staged
- Can stage changes to same file several times
- Think “snapshot” of the work

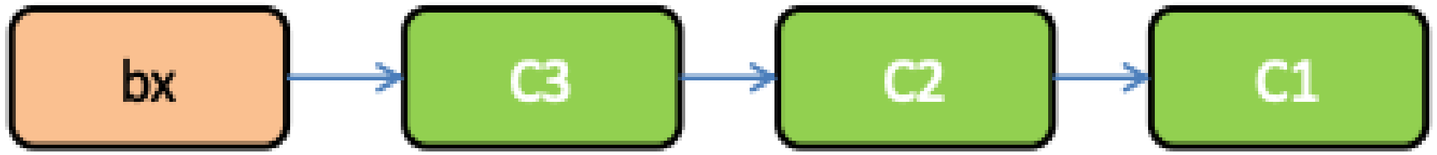
- Changes from the index stored
- Get a unique rev. number
- Index emptied

GIT Model

LOCAL

Branches

- Branch is a pointer to a commit (and thus those before it)
- A branch is a way to organize working histories



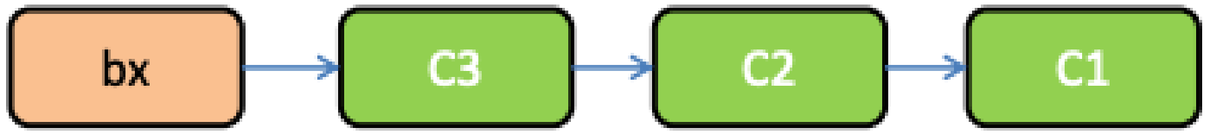
On branch bx -> Commit of C4

GIT Model

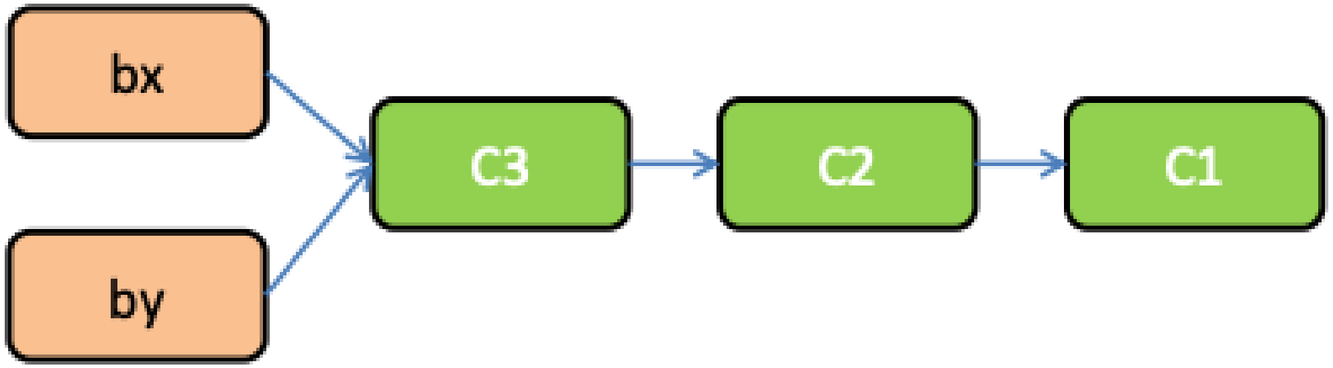
LOCAL

Branches

Here's a branch



Creating a new branch just adds another "pointer" to the same commit

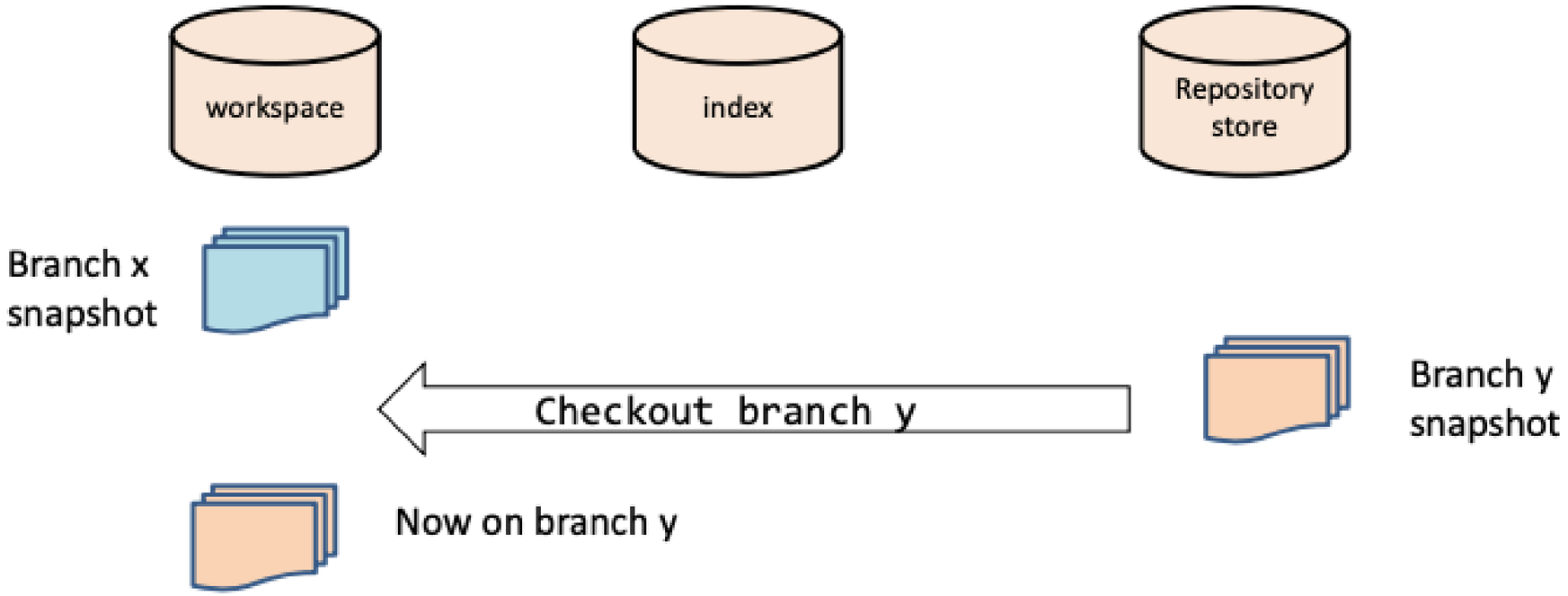


GIT Model

LOCAL

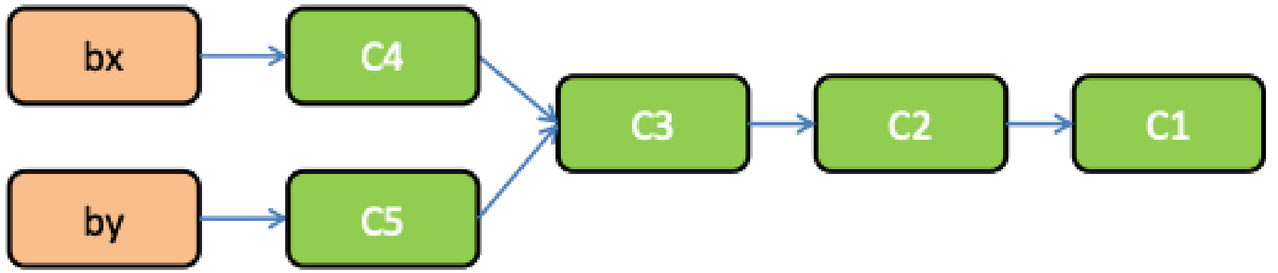
Branches

Checking out a branch puts the snapshot (the commit) it points to into the workspace

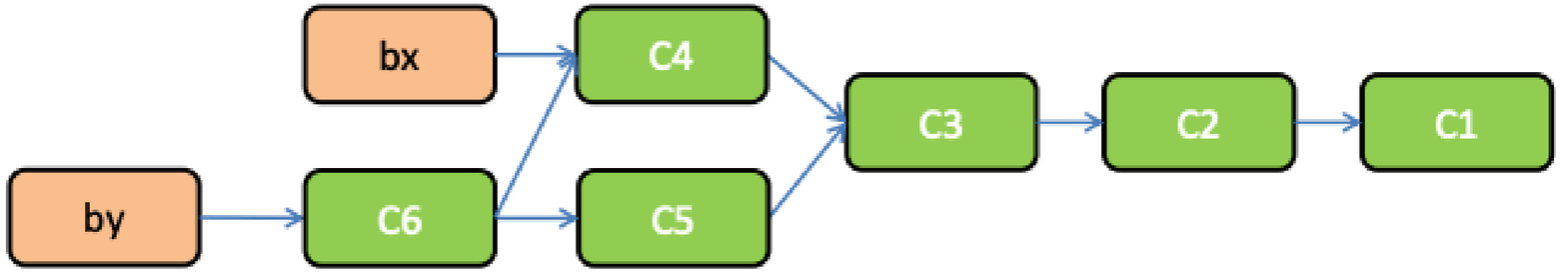


Merging

Combining one ore more branches into the current branch

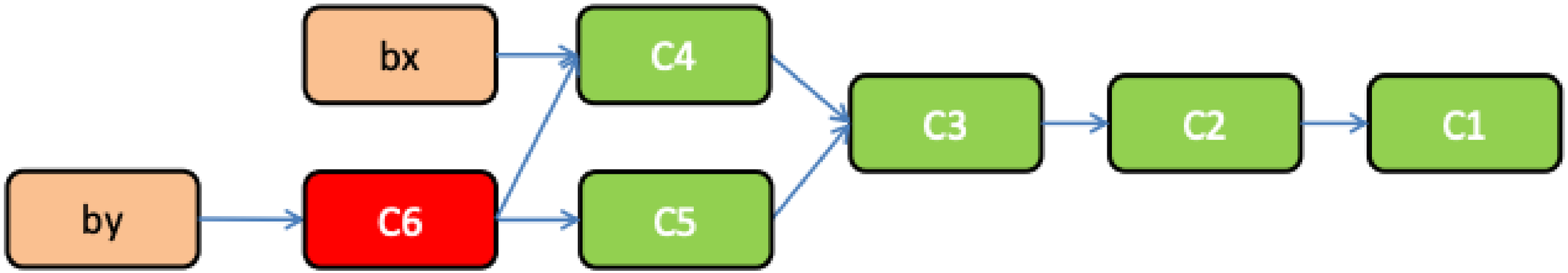



```
Merging bx into by
```



Merging - Conflicts!

- When merging, if there are conflicts - need to solve them.
- After solving, need to “add” the changes and commit the merged workspace



Remote Git - Brief Intro

Single repository concepts

REMOTE

Collaborating

- **Clone** – creates a local repository by copying another (remote) repository
- Repositories which were cloned can exchange changes between them

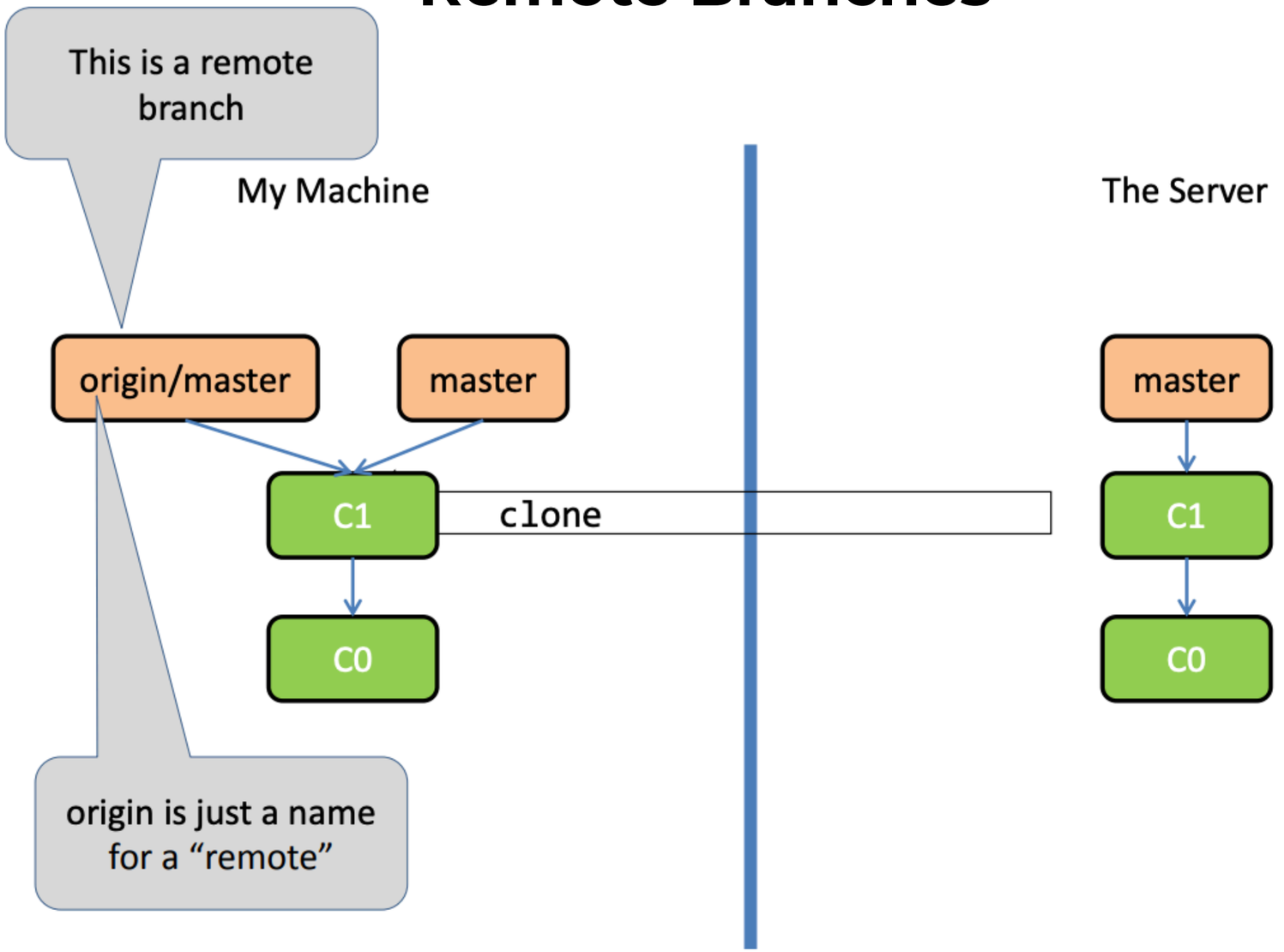
GIT Model

REMOTE

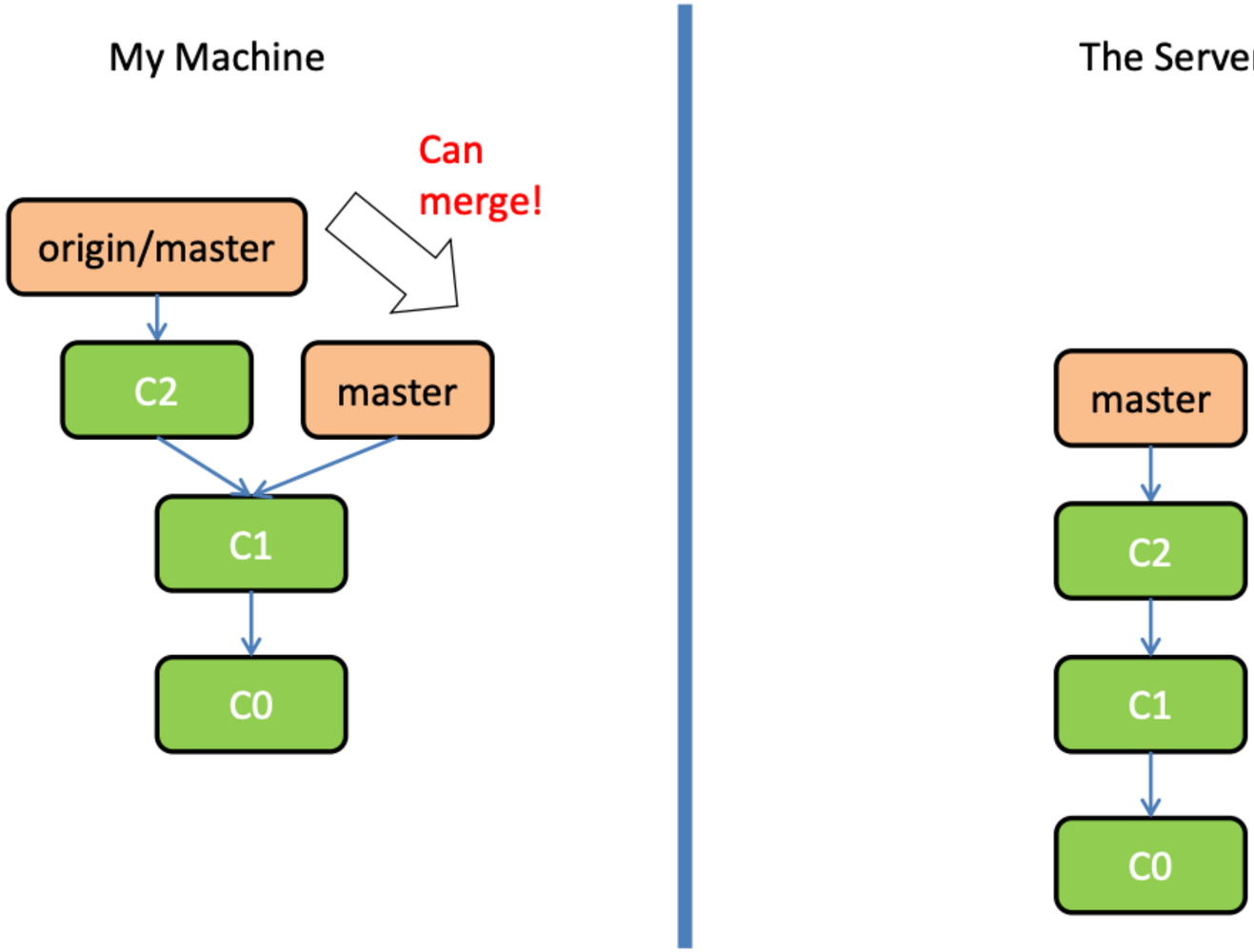
“git fetch”

- **Fetch** – gets changes from a remote repository that you don’t already have.
- Fetch gets the changes to the local repository but does not touch the index or workspace.

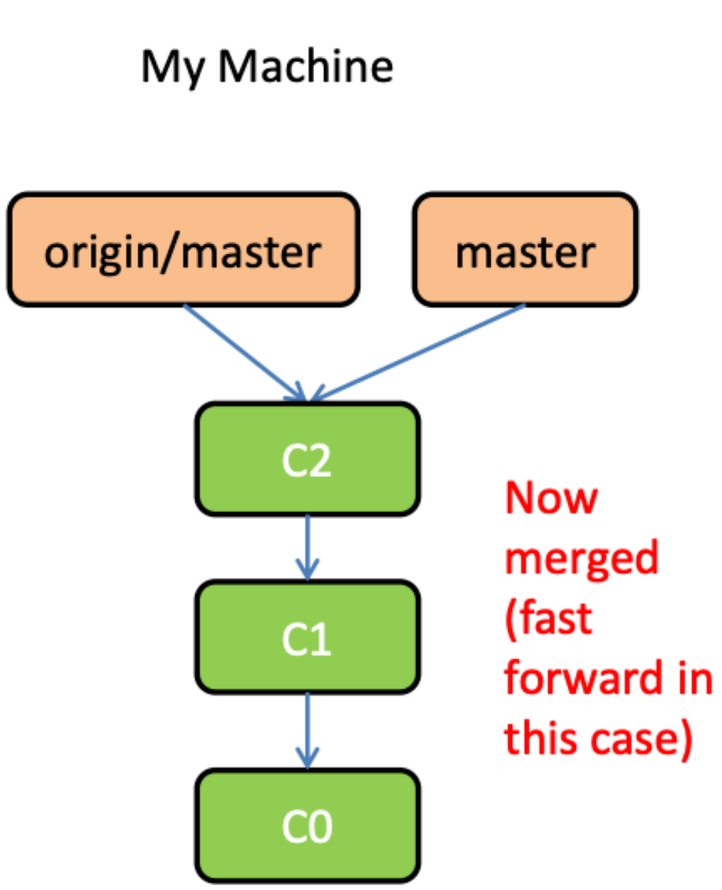
Remote Branches

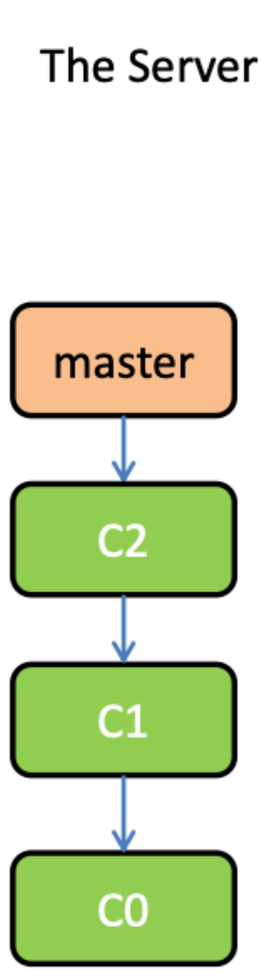


Remote Branches



Remote Branches

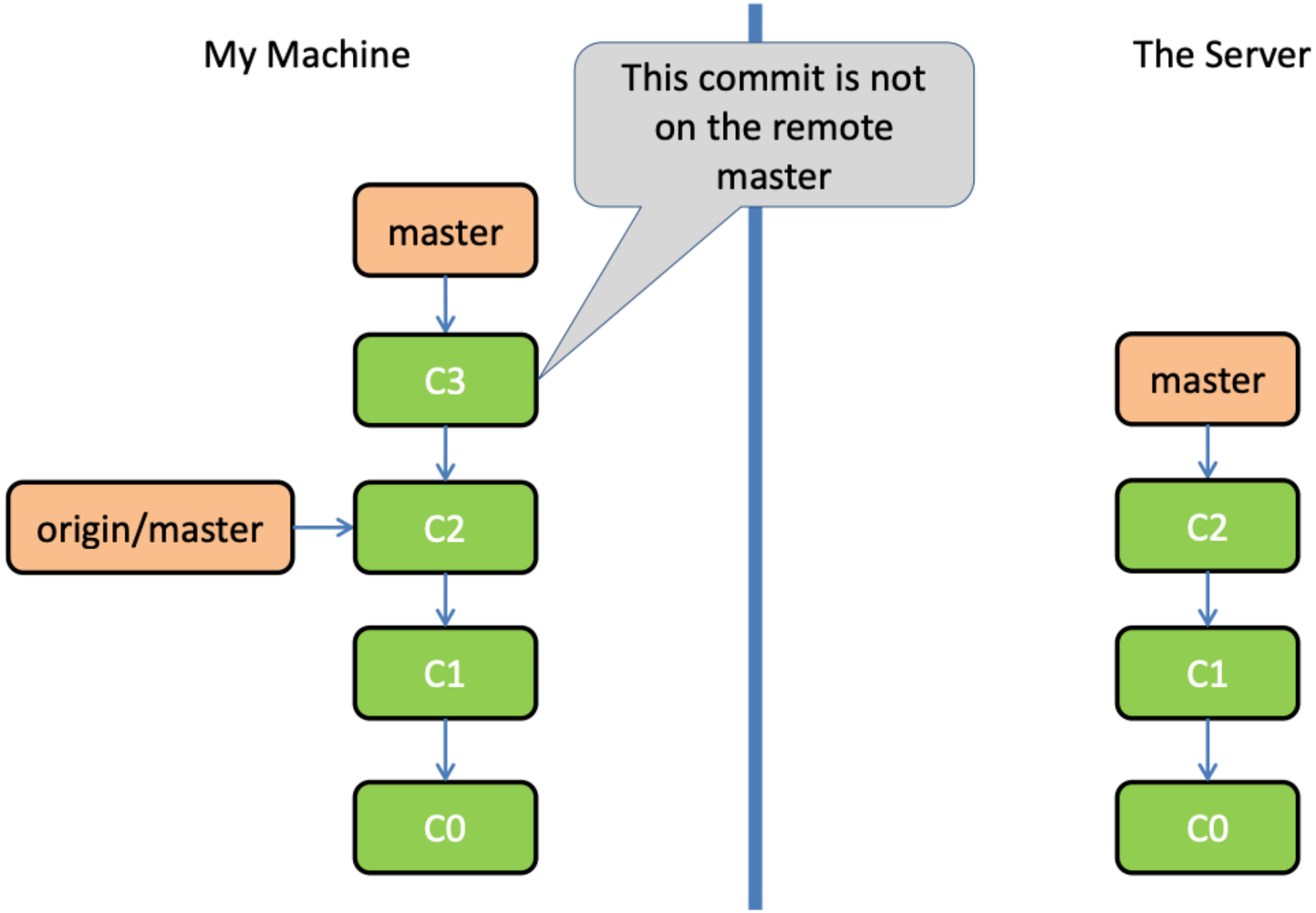




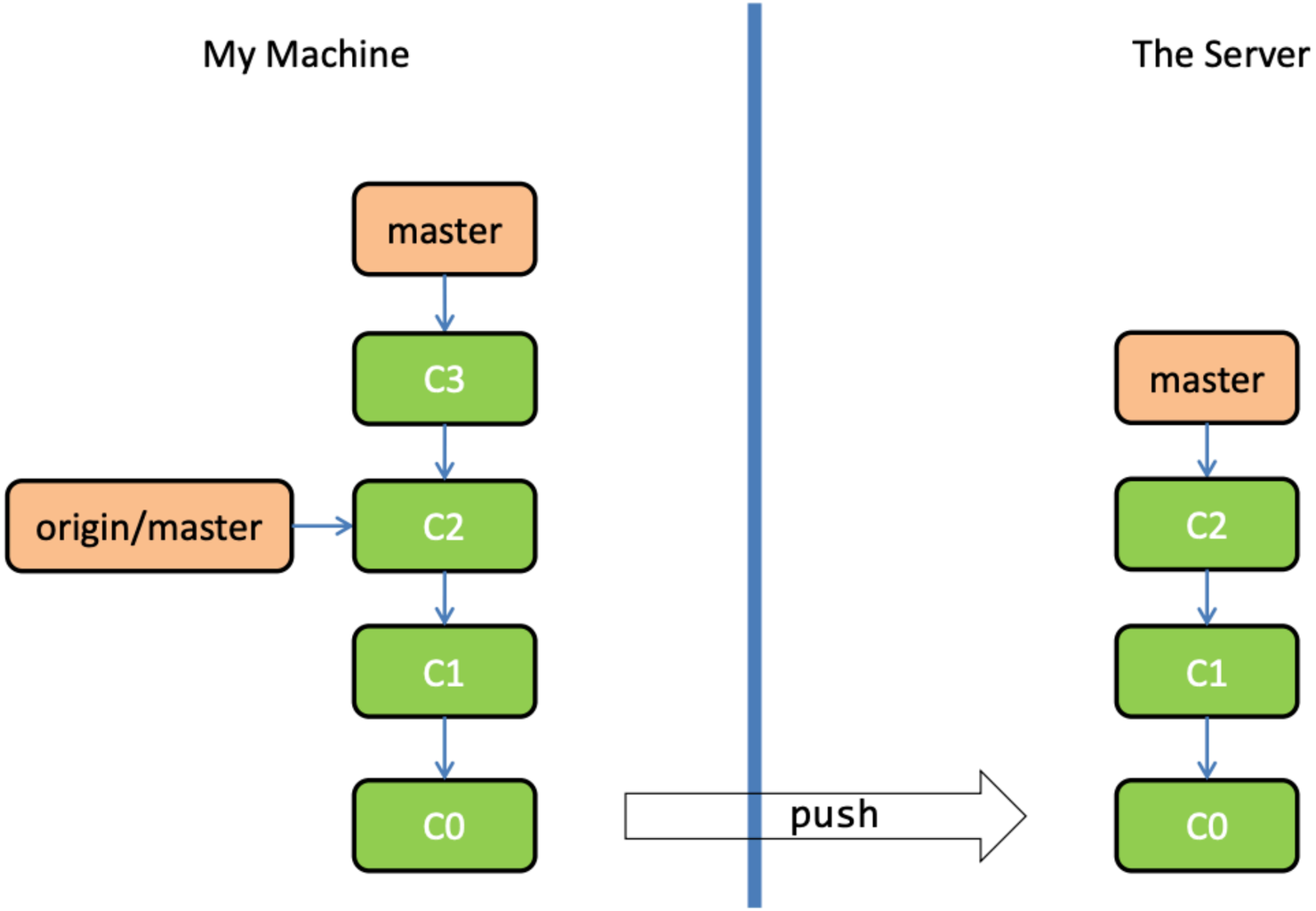
Remote Branches - “git push”

- Will take local objects (commits, tags) which are required to make a remote branch complete – and send them.
- Will merge those local changes into the remote branch
- Will only do a “fast-forward” merge (other merge type, if required, will fail the push)

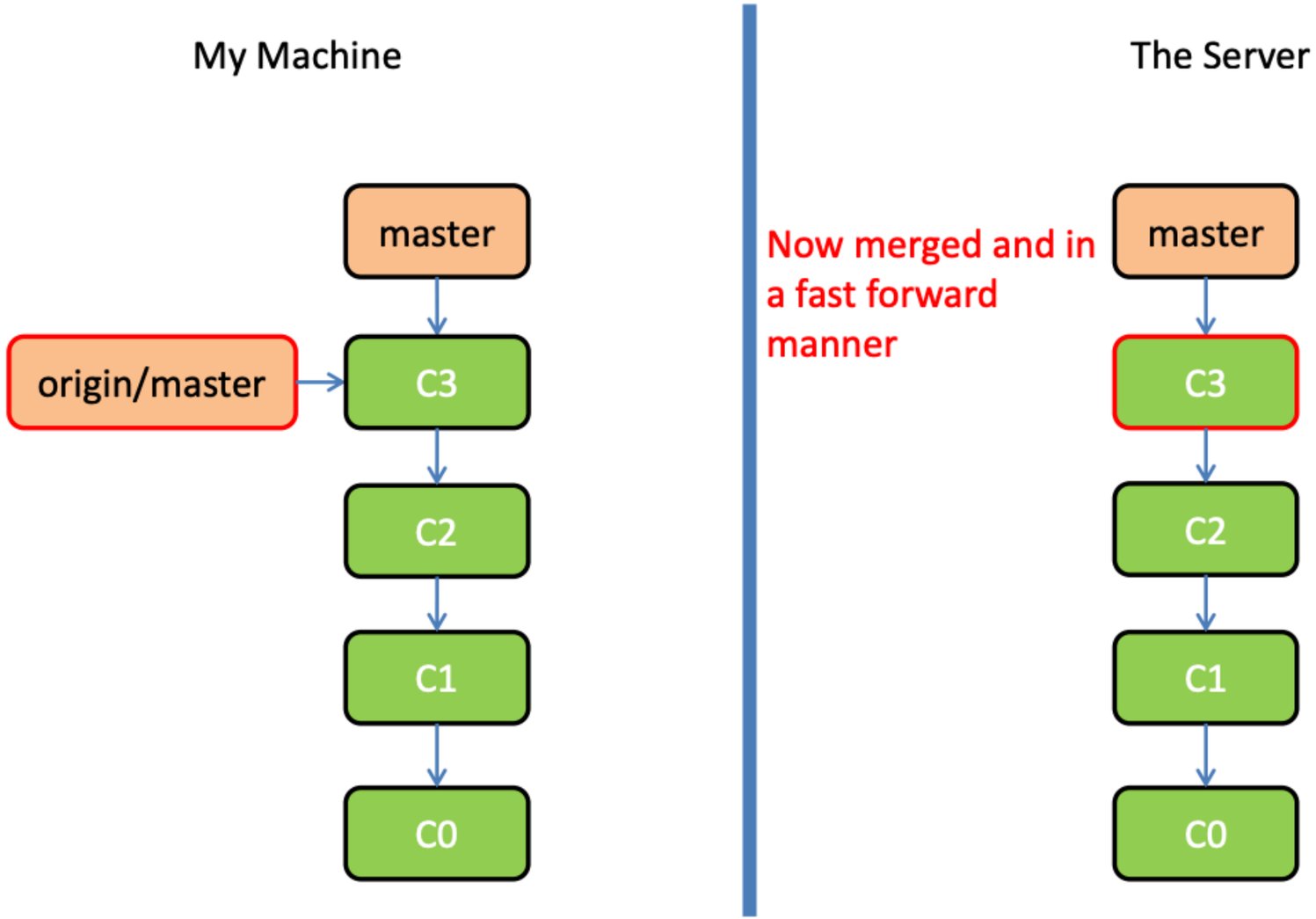
Remote Branches - “git push”



Remote Branches - “git push”



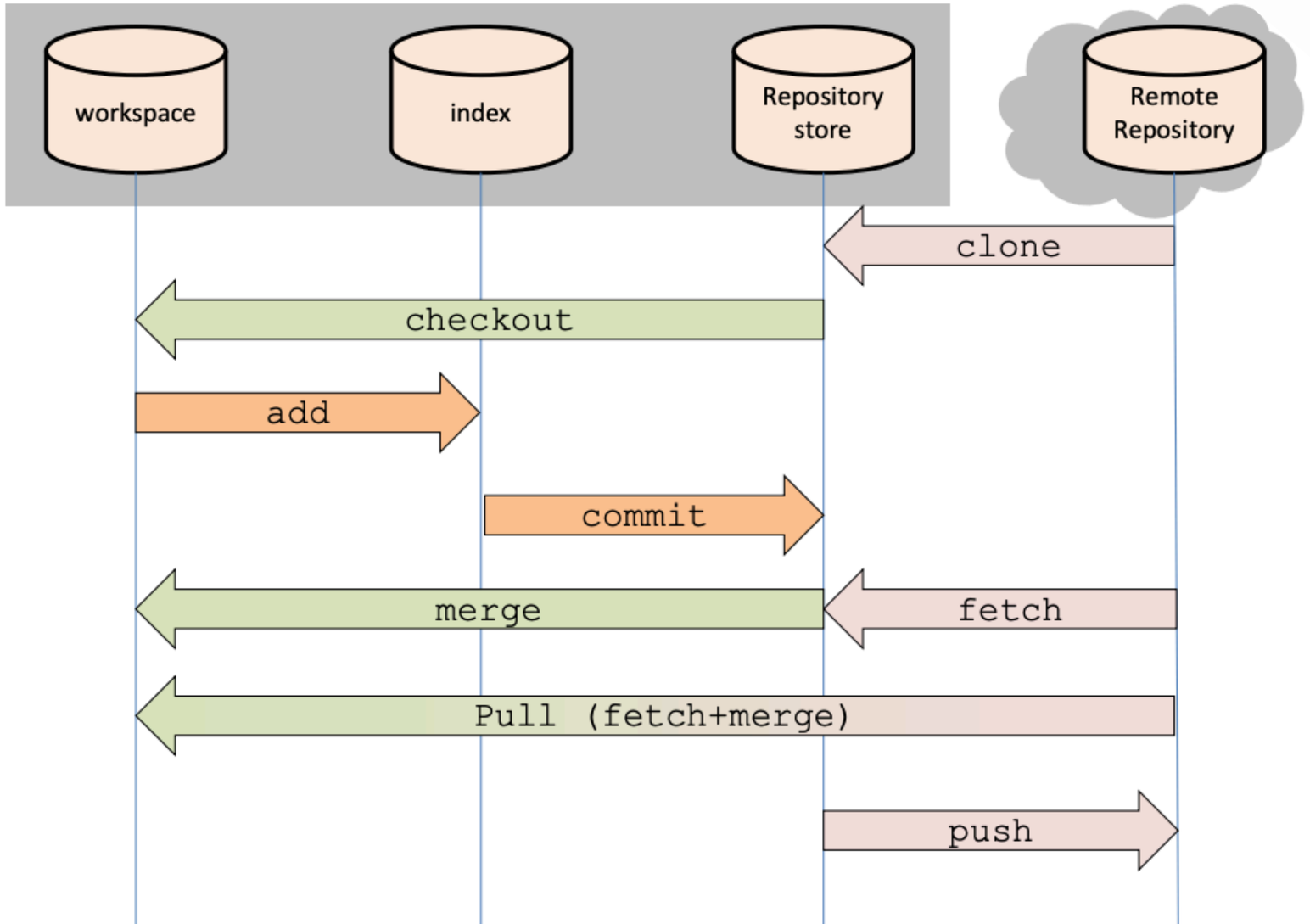
Remote Branches - “git push”



Remote Branches - “git pull”

1. Fetch from a remote
2. Merge changes from “remote branch” into the “remote tracking branch

Remote Branches - Summary



Remote Branches

- Reminder - Remote branches represent a branch on a remote repository
- The branch origin/master for example is a local pointer to the “master” on “origin”
- It reflects what the local repository currently knows about the state of “master” on “origin”
- You can “checkout” to get a “remote tracking branch”

Time to GIT yourself up and running...

Code Along

Get Familiar with Git!

git

THE UNDERGRADUATE
ARTIFICIAL INTELLIGENCE SOCIETY

THANK YOU

Feel free to stick around and chat!

- Website:uais.dev
- Instagram: uais.ualberta
- Email: uais@ualberta.ca