Scientific Software Development

Inga Ulusoy, Scientific Software Center, Interdisciplinary Center for Scientific Computing, Heidelberg University

March 2023

What you will learn...

- How to develop scientific software professionally:
  - No more broken code through version control/rigorous code review/testing/...
  - Track your progress and analyze your code
  - Manage your implementation in a way that facilitates further additions to the code
  - Generate reproducible and reliable scientific results
  - Make your contributions as a "scientific software developer" visible

Overview

Unit 1
Introduction to git: Using git and GitHub

Unit 2
Clean coding as a team: Adhere to guidelines and common coding principles

Unit 3
Think before you code: Planning your programming project

Unit 4
Documenting software

Unit 5
Software testing

Unit 6
Continuous integration (CI): GitHub Actions; also: publishing code

Unit 1: Introduction to git and GitHub

- What is git?
- What is GitHub?
- Creating branches
- Merging branches
- Jupyter notebooks
- Pre-commit hooks
- Useful git commands

What is git?

- A version control system (VCS) = software to track changes in files and work collaboratively, supports non-linear workflows
- Originally created in 2005 by Linus Torvalds (Linux kernel) and others
- "git" can mean anything, depending on your mood. Stupid. Contemptible and despicable. Simple. Take your pick from the dictionary of slang. "Global information tracker": you're in a good mood, and it actually works for you. Angels sing, and a light suddenly fills the room. "Goddamn idiotic truckload of sh*t": when it breaks. [source: https://github.com/git/git]

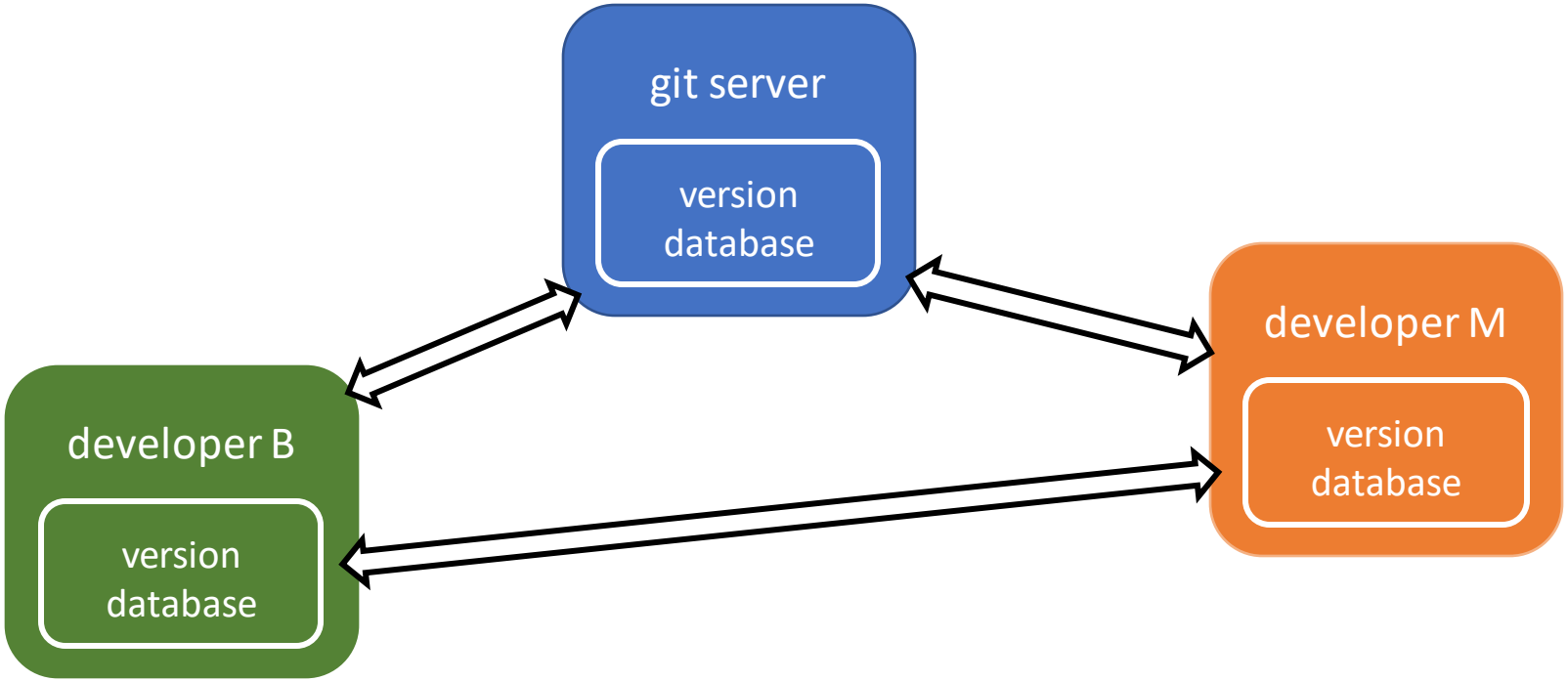
Characteristics of git

- Non-linear development
- Distributed development
- Compatibility (HTTP, ssh)
- Efficiency

Very useful to keep track of development among a team of developers

git version control system (VCS)

git is a distributed version control system that keeps track of your changes to the software. Distributed means that every developer has a complete copy of the project, including its history.



git version control system (VCS)

- Revert file to previous state
- Revert entire project to previous state
- Review changes over time
- Review and track issues, …

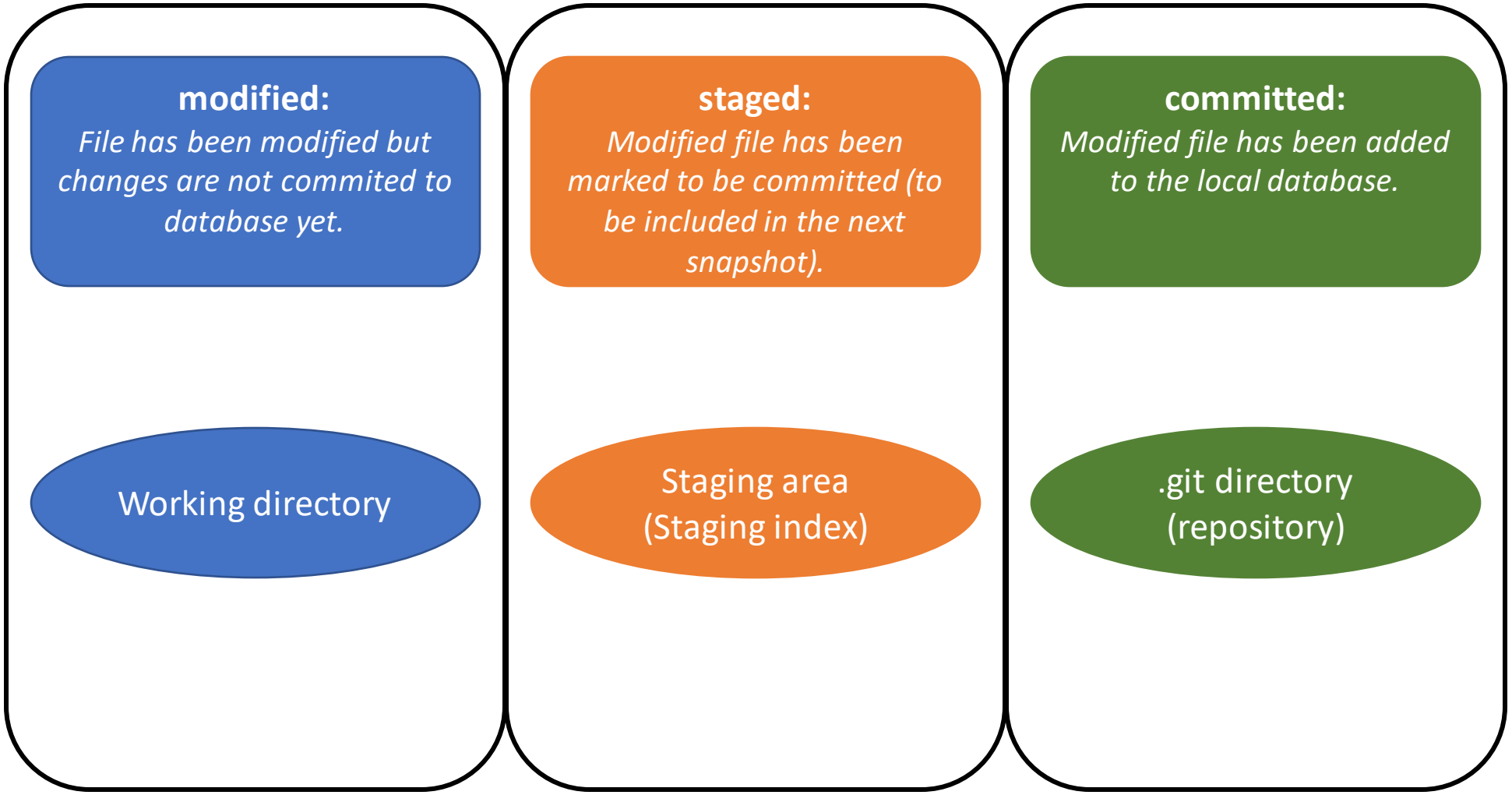
Commit:
A snapshot of your file system
A commit saves the state of your project at that time.

Repository:
A directory containing your project and additional files to communicate with git
This is not equivalent to your working directory.

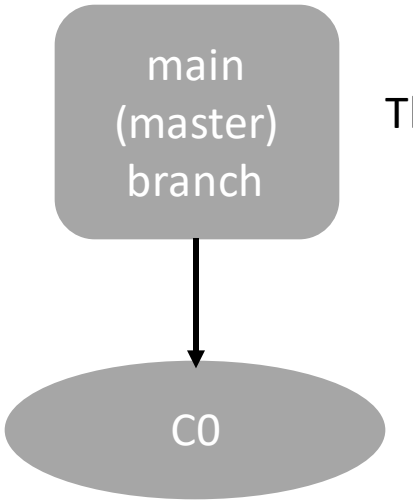
Checkout:
A copy of repository content in your working directory
This can be a specific file, commit, branch, …

Branch:
A new line of development in your project
The branch leaves the main status of the project unchanged.

The three stages of git



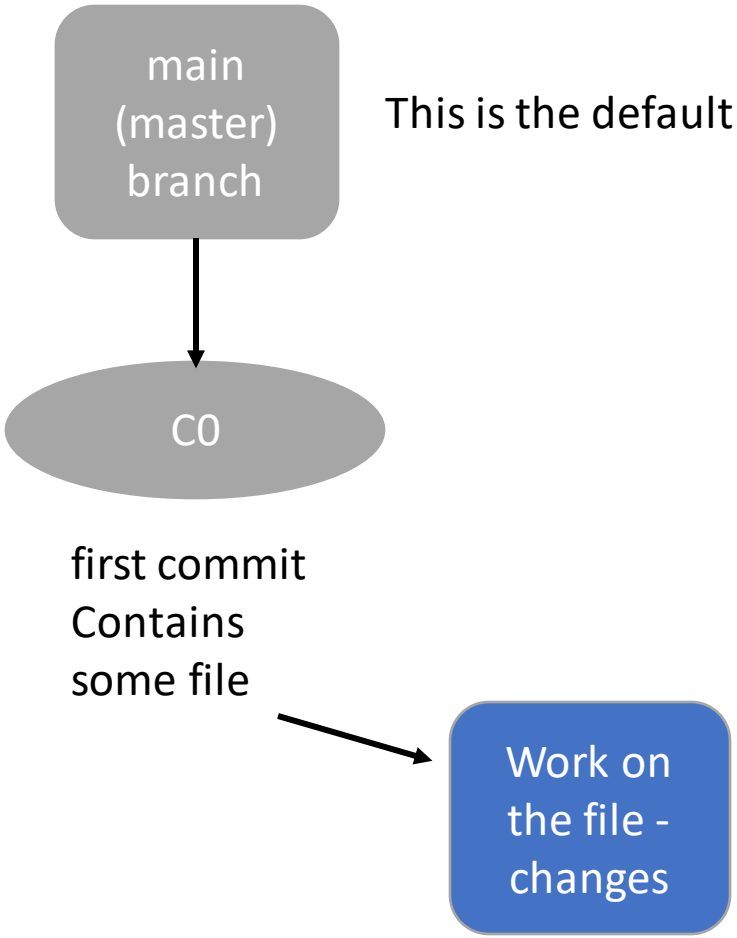
git commits



This is the default branch.

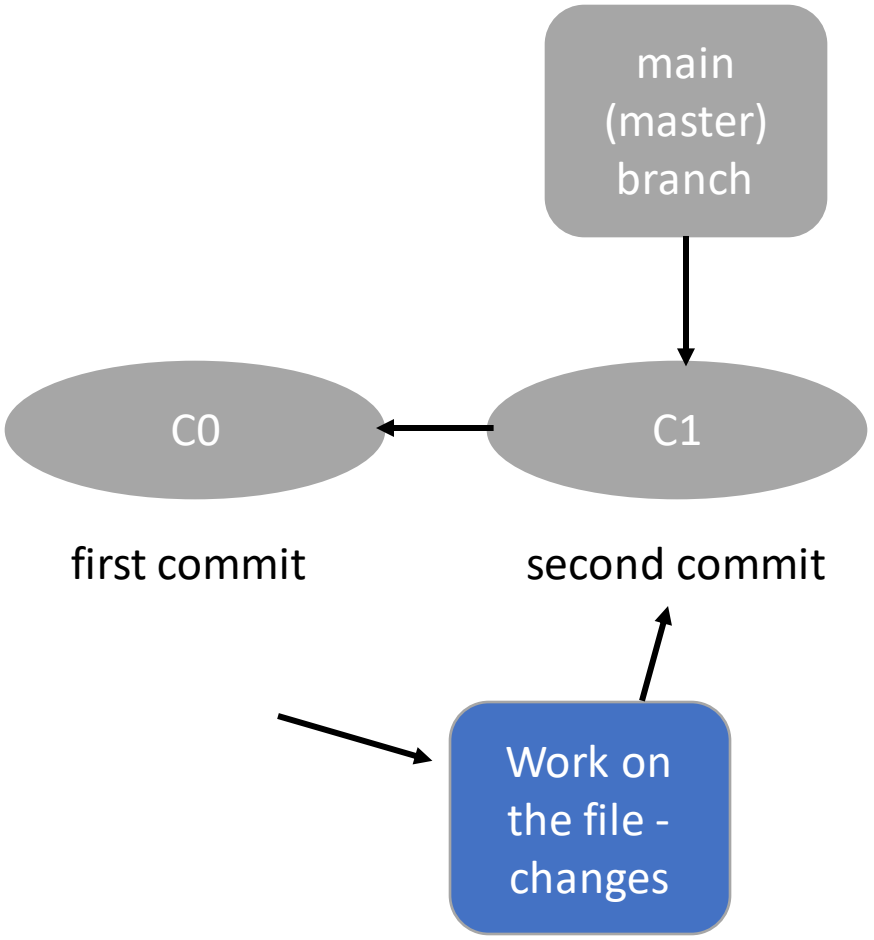
first commit
Contains
some file

git commits



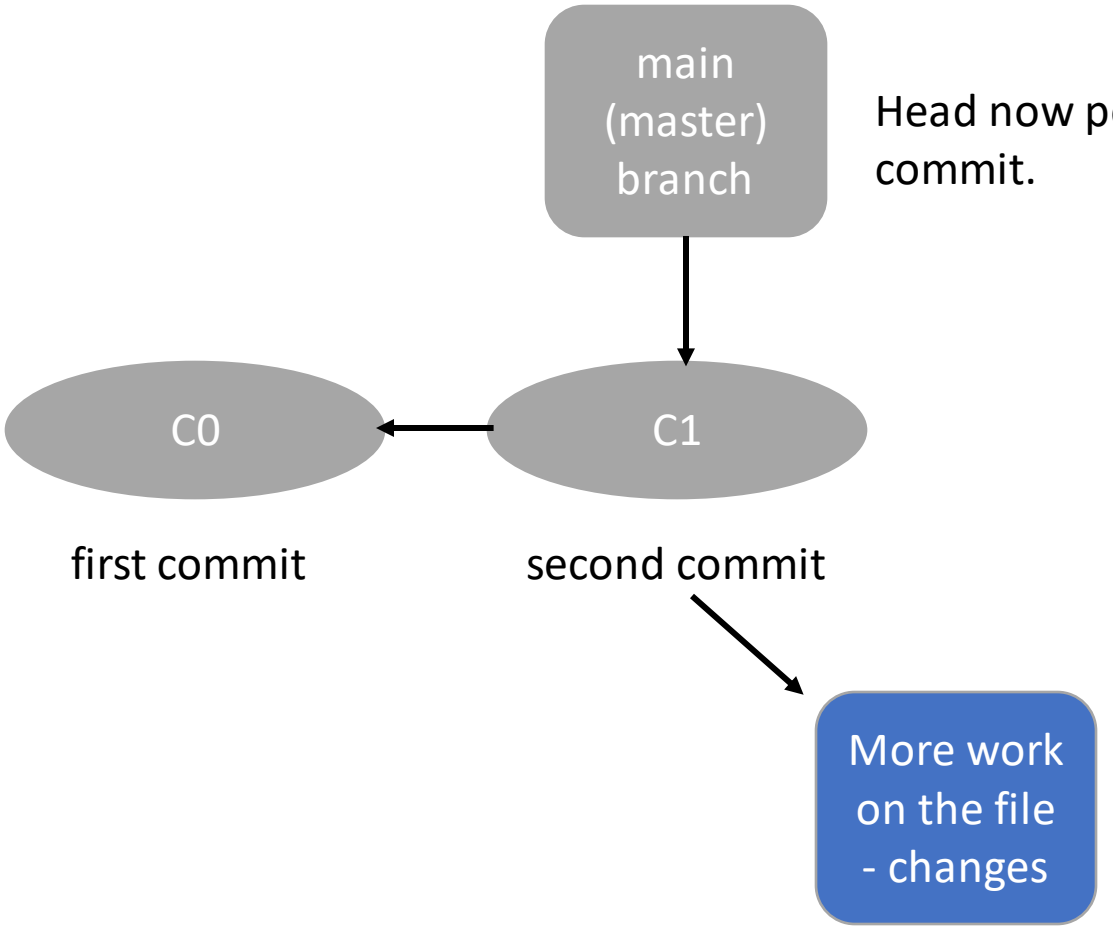
This is the default branch.

git commits

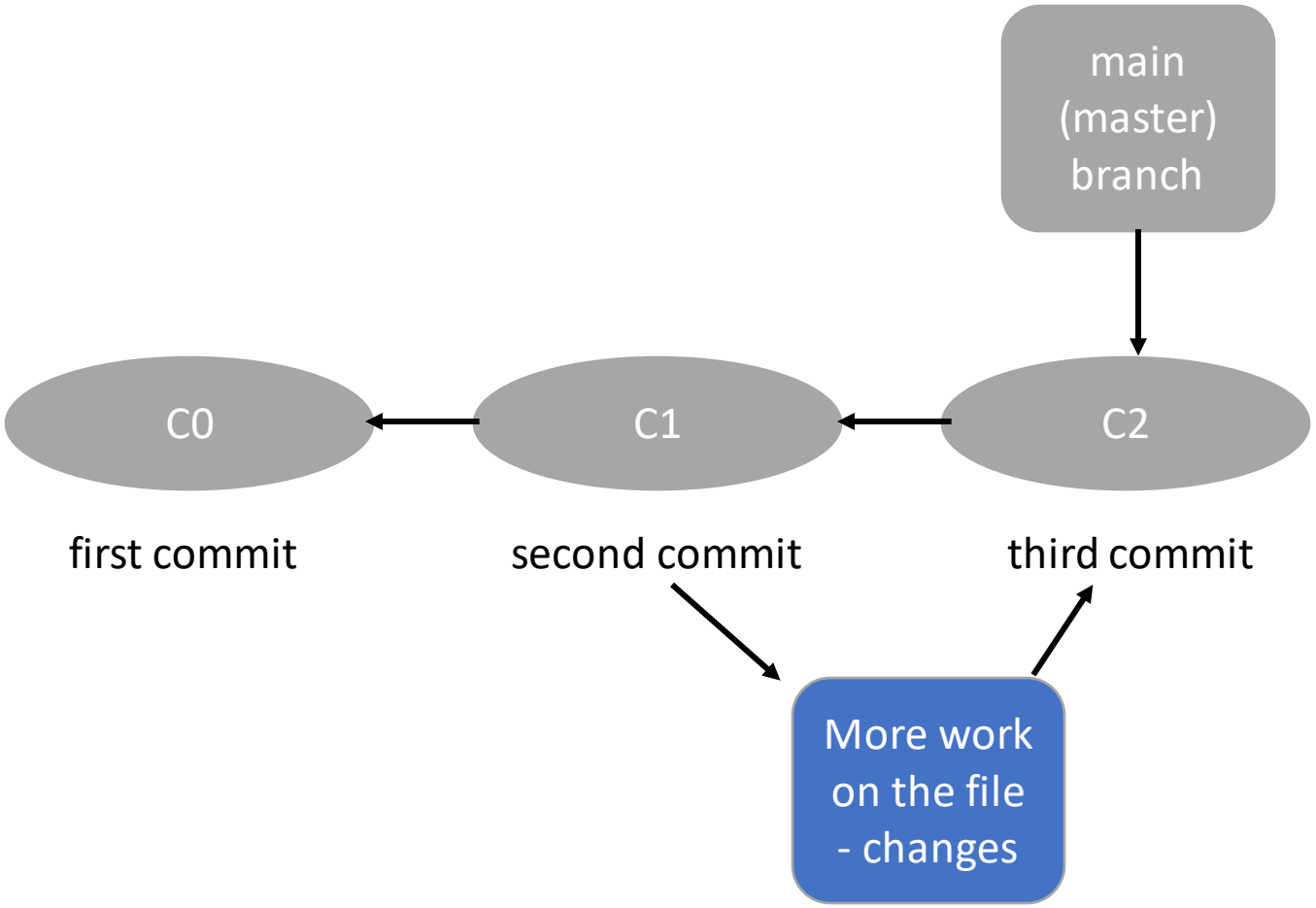


Head now points to second commit.

git commits


main
(master)
branch
Head now points to second commit.
C0
C1
first commit
second commit
More work on the file - changes

git commits



Head now points to third commit.

git

- Please download and complete the intro in the game Oh-My-Git!

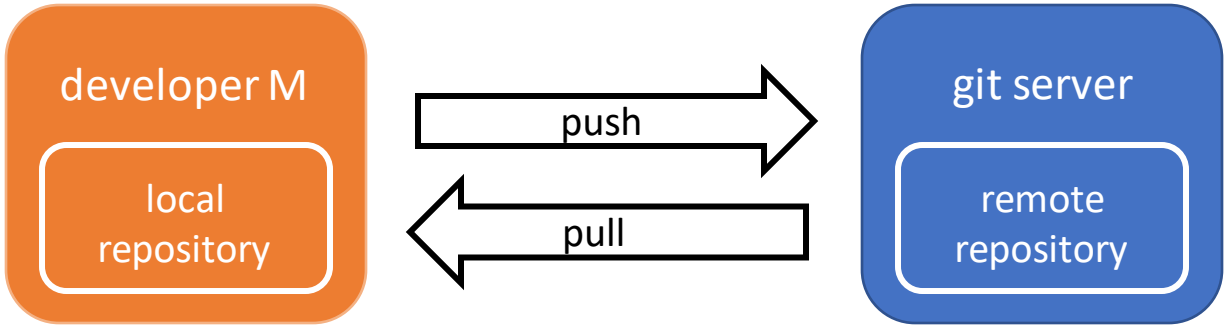
https://ohmygit.org/

Unit 1: Introduction to git and GitHub

- *What is git?*
- What is GitHub?
- Creating branches
- Merging branches
- Jupyter notebooks
- Pre-commit hooks
- Useful git commands

Remote repositories

Remote repositories are hosted on the internet or network. You can set up your own git server or use services such as GitHub, GitLab or Bitbucket.



Remote repositories: GitHub

- Cloud-based service helping developers store their development projects, offering git version control, website interface
- Currently 73 million developers use GitHub, 200 million repos: largest source code host
- For-profit: makes money hosting private repos + business plans
- Subsidiary of Microsoft since 2008

Additional benefits of GitHub

- Issue tracking: You can open issues, comment on them, close them in Pull Requests
- Pull Requests: When you finished a line of implementation, allows to comment, highlight lines of code, interact with collaborators (review)
- Documentation through GitHub pages, wiki, Readme
- GitHub actions: CI/CD
- Discussions
- Projects (agile)
- …

git cloud hosting services

- GitHub
- GitLab
- BitBucket
- …

Differences: GitHub mostly focused on collaboration while GitLab emphasizes features and offers platform for web developers. Also: more private repos on GitLab (same for BitBucket).

We will use GitHub. Please create a user account on www.github.com

GitHub

Using GitHub in your browser does not require you to know all the git commands. Please complete this GitHub learning lab:

https://lab.github.com/githubtraining/introduction-to-github

You will learn about commits, branches, merging and pull requests in this learning lab. Pull requests are very useful as they give you a chance to discuss your changes to the code with your collaborators.

Unit 1: Introduction to git and GitHub

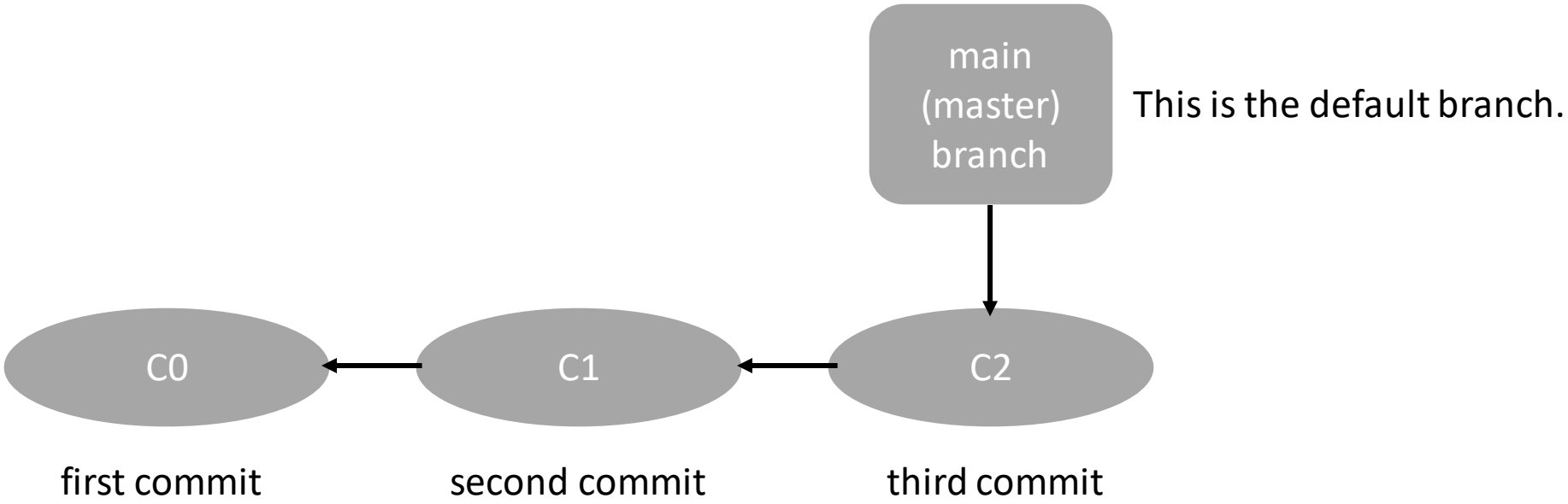
- *What is git?*
- *What is GitHub?*
- Creating branches
- Merging branches
- Jupyter notebooks
- Pre-commit hooks
- Useful git commands

Creating branches

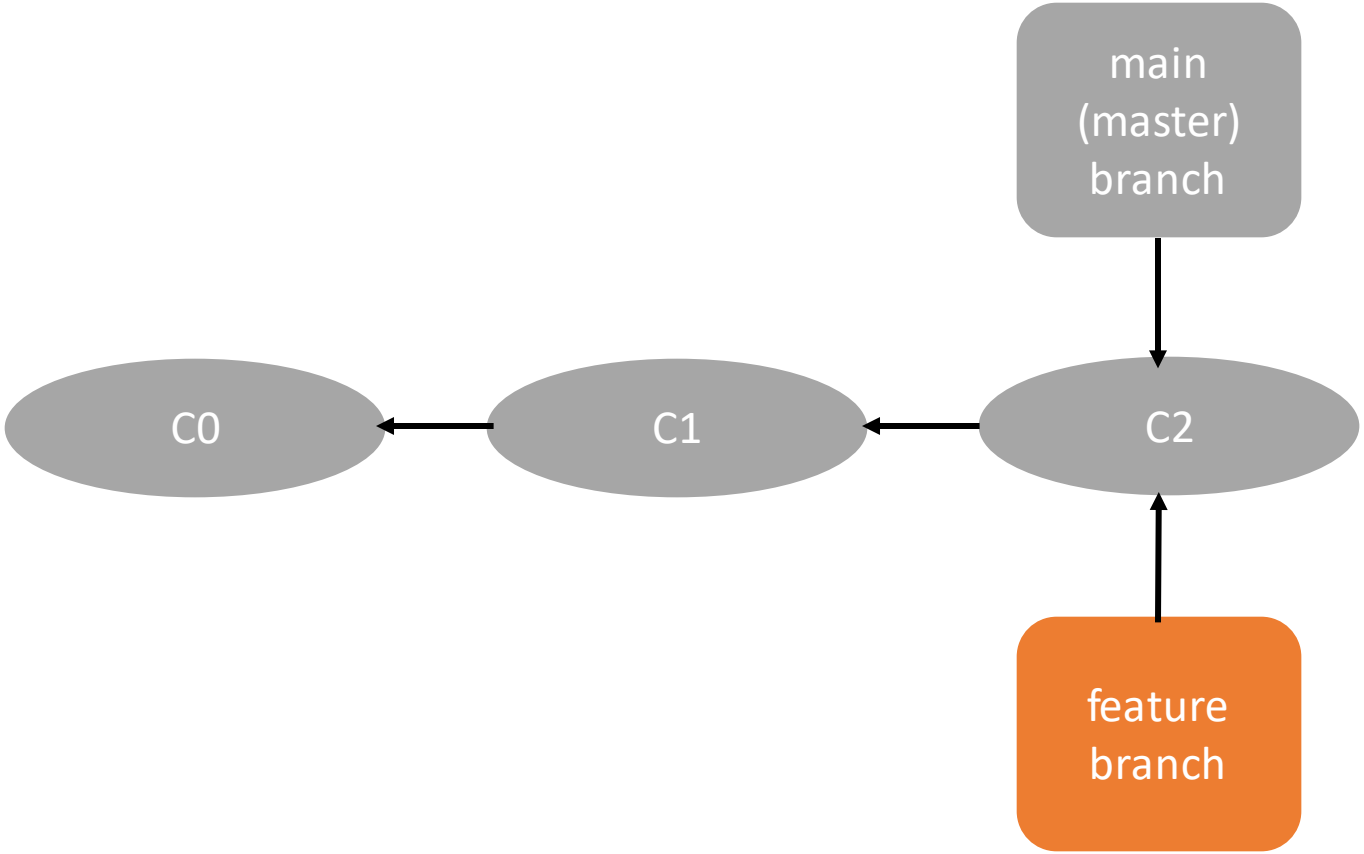
Branches represent a new line of development and keep your commit history clean.

```
git checkout –b "my-branch"
```

git commits and branches

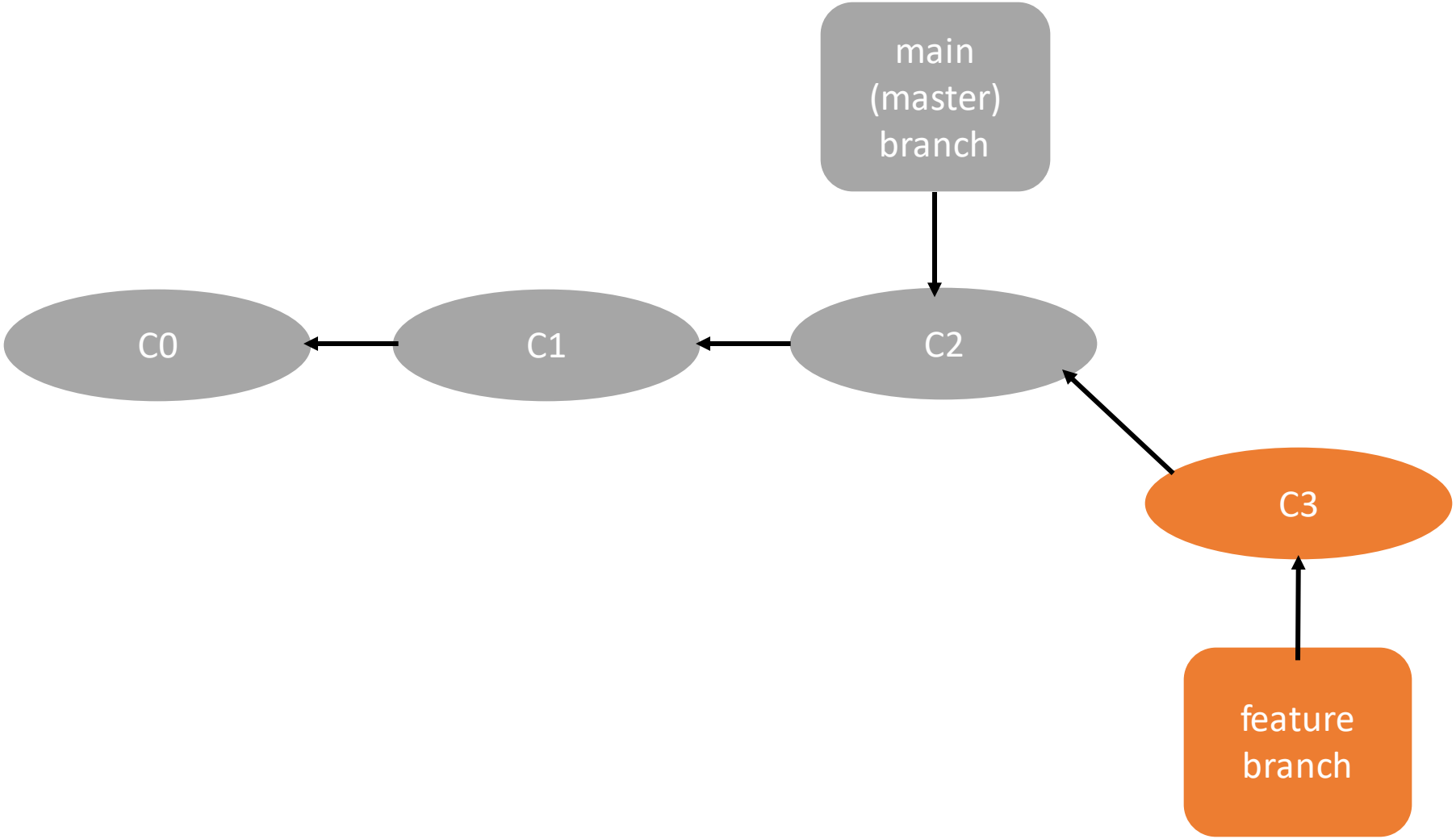


git commits and branches

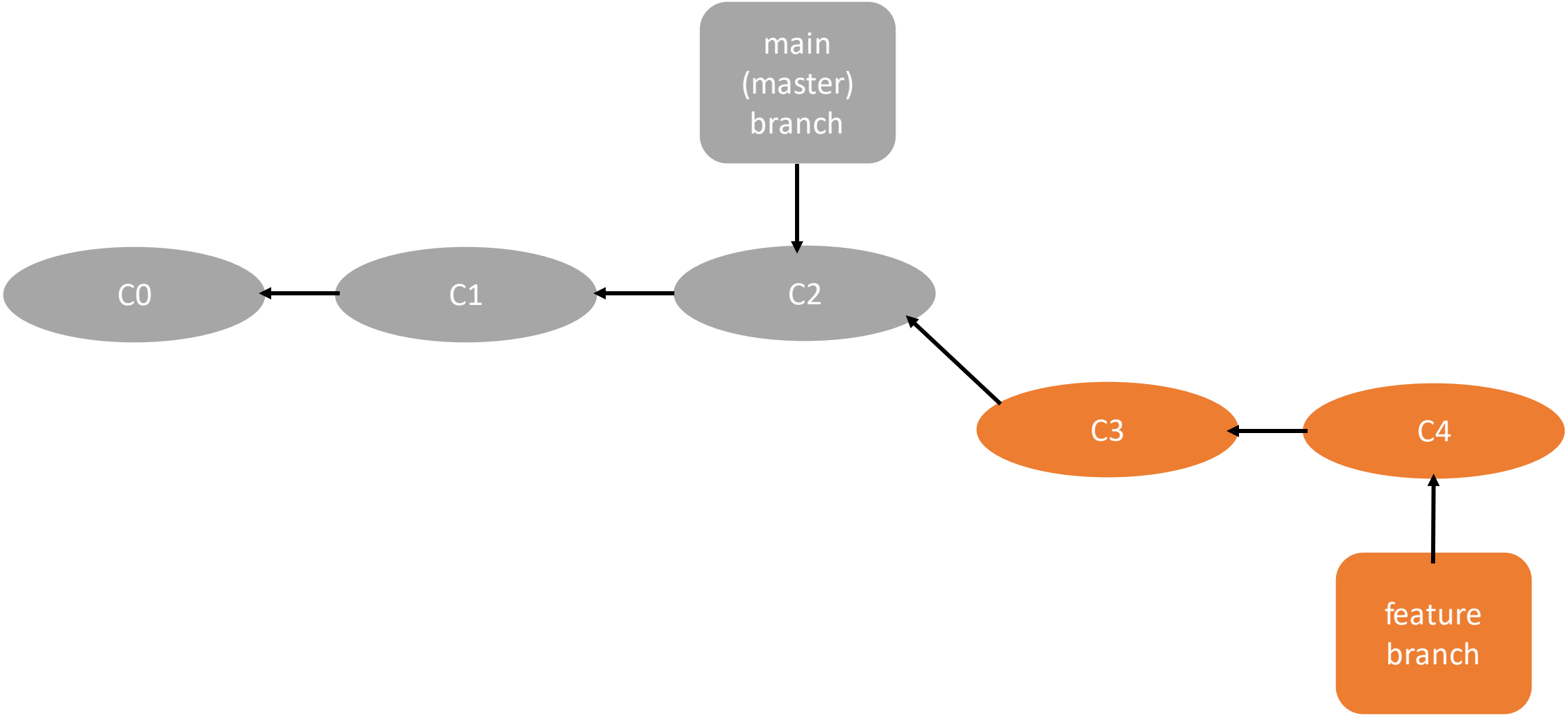


This is the new feature branch.

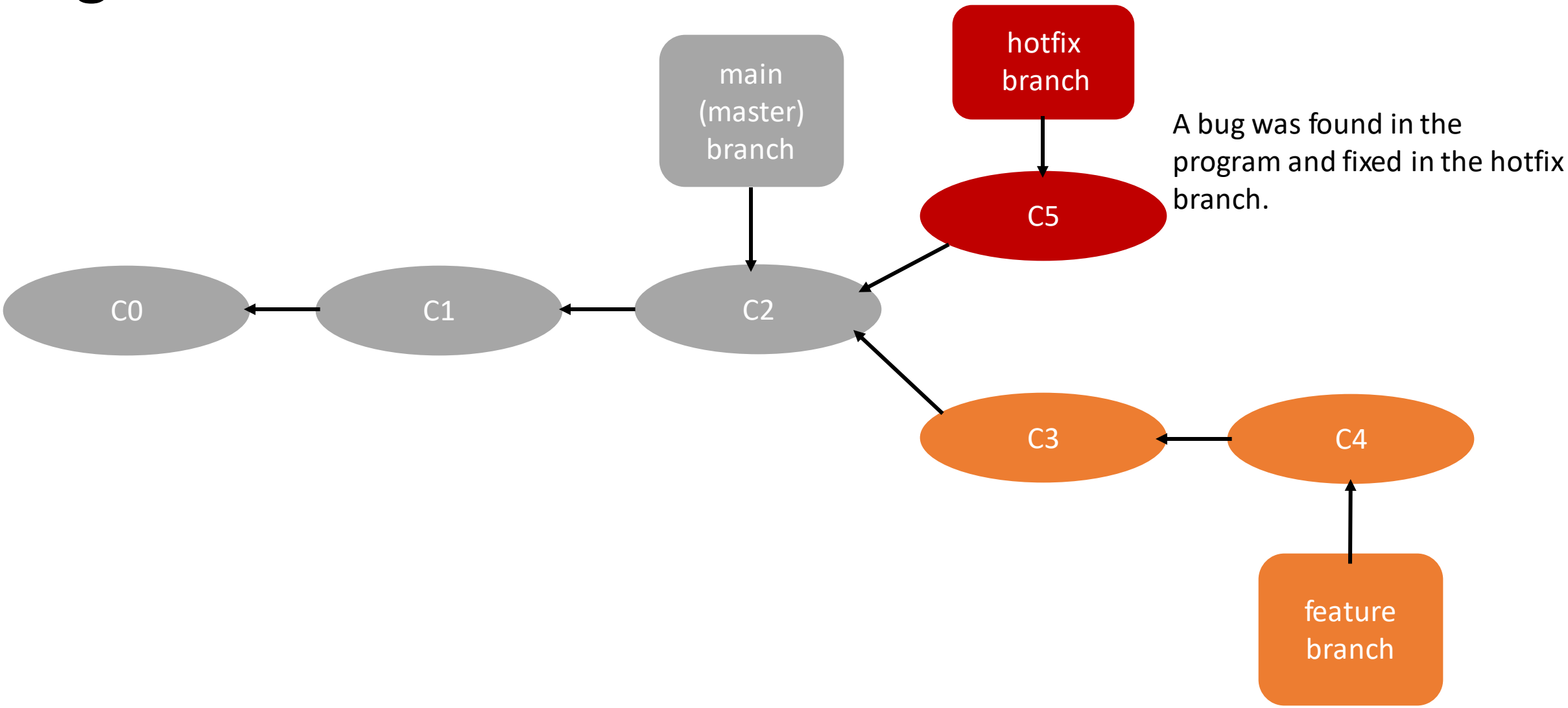
git commits and branches



git commits and branches



git commits and branches



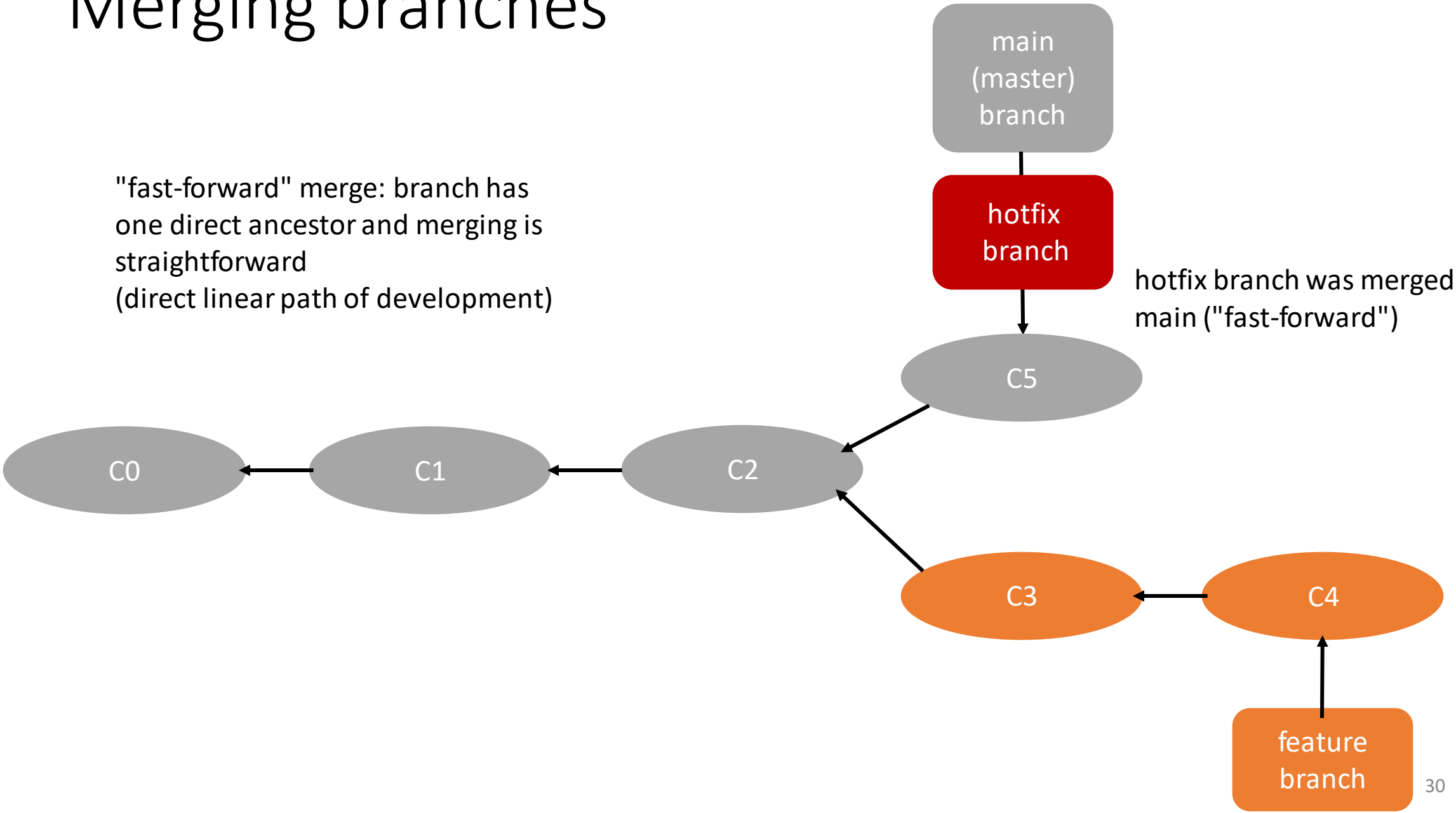
A bug was found in the program and fixed in the hotfix branch.

Unit 1: Introduction to git and GitHub

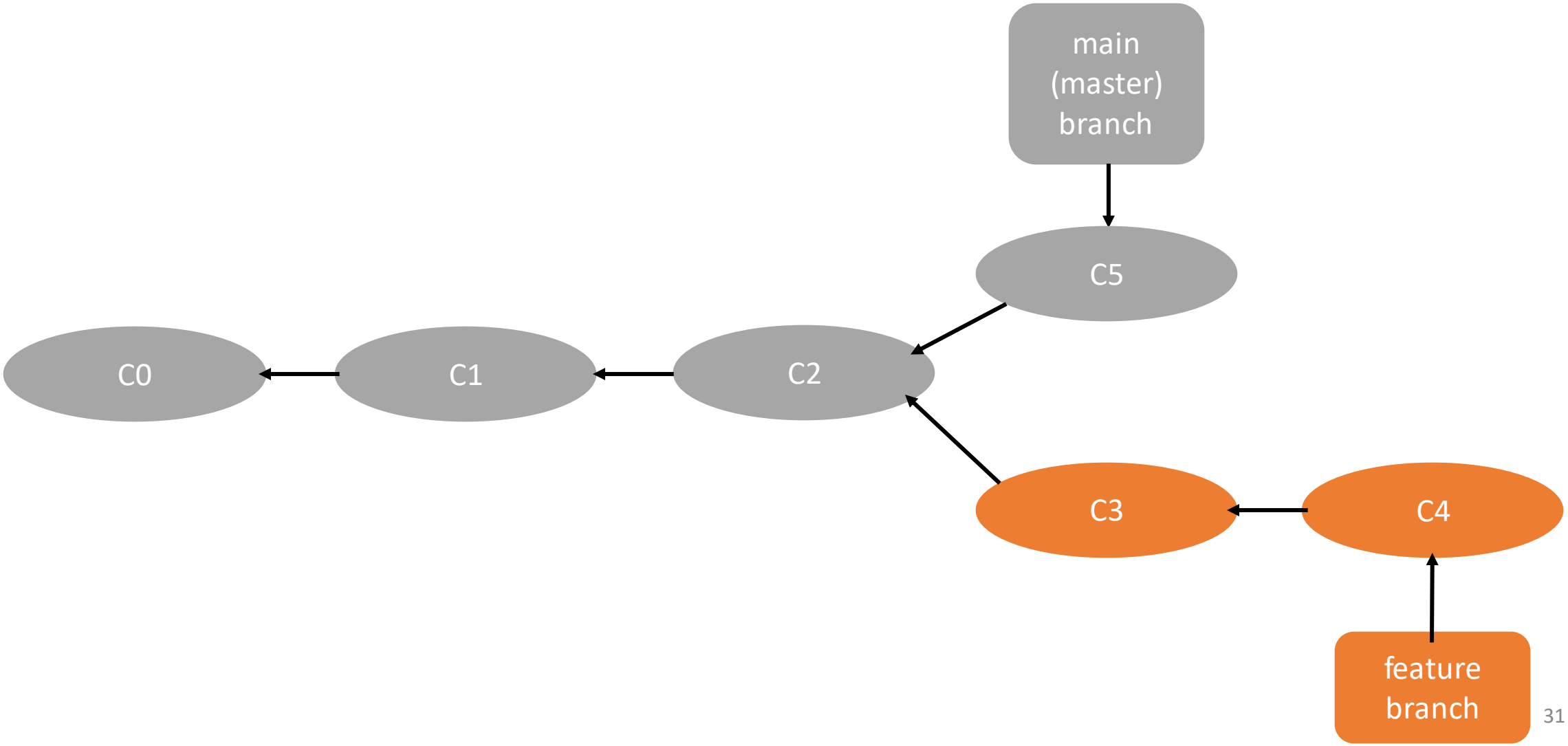
- *What is git?*
- *What is GitHub?*
- *Creating branches*
- Merging branches
- Jupyter notebooks
- Pre-commit hooks
- Useful git commands

Merging branches

"fast-forward" merge: branch has one direct ancestor and merging is straightforward
(direct linear path of development)

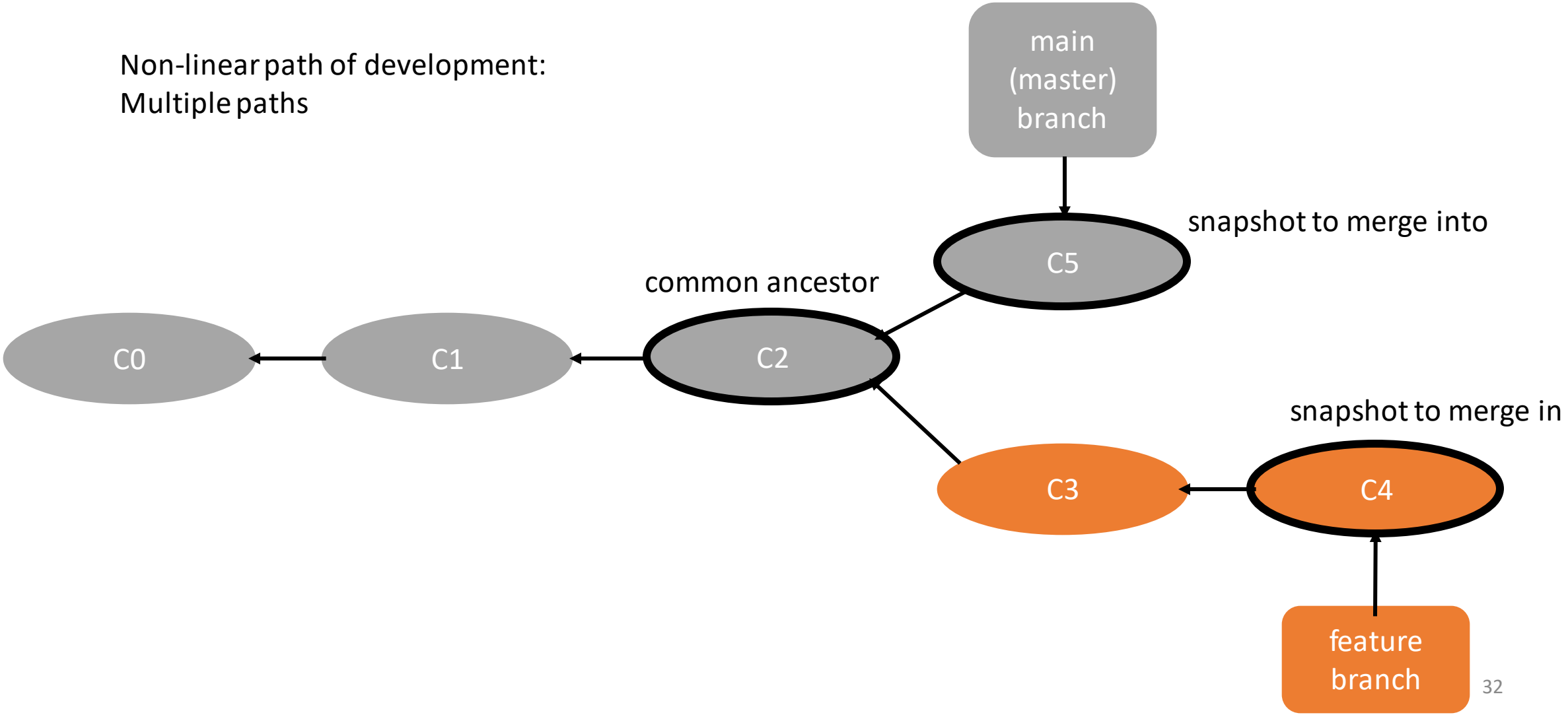


Merging branches

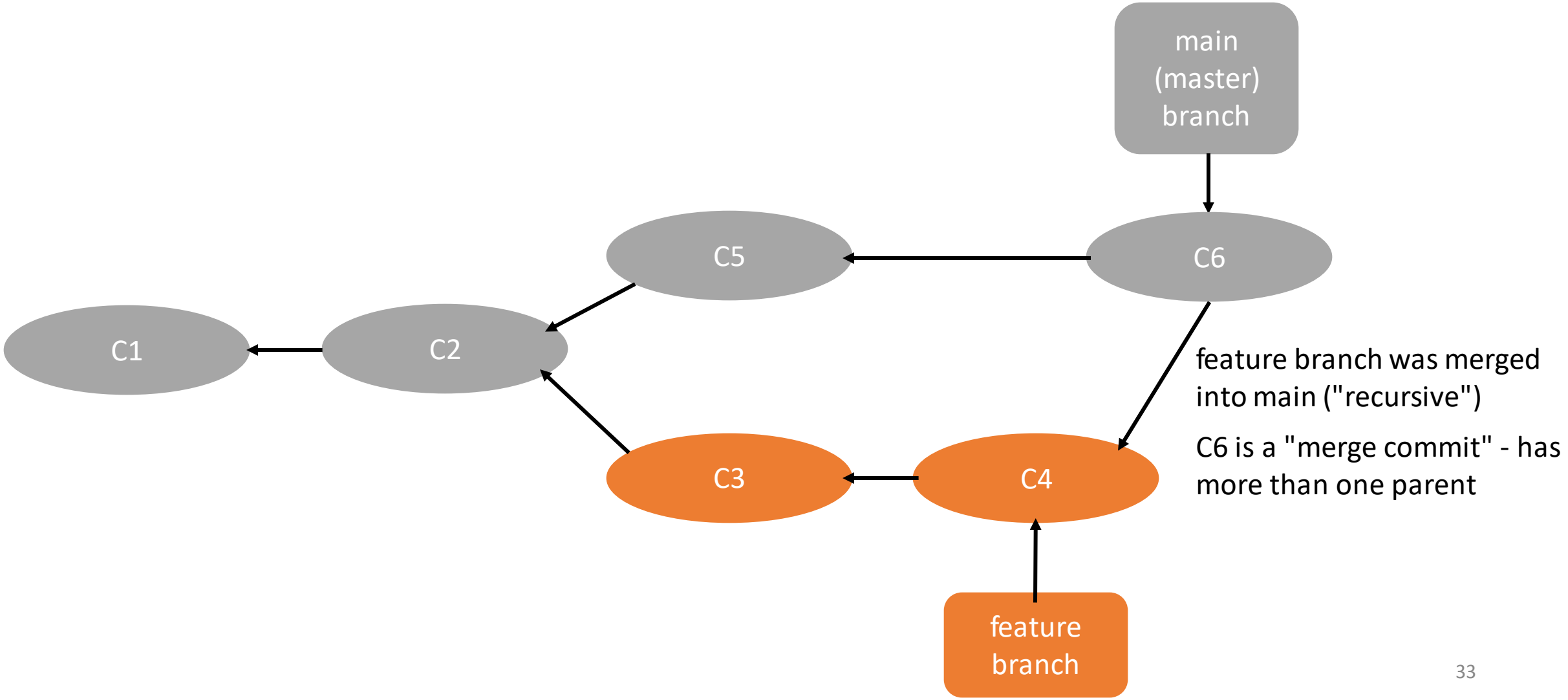


Merging branches

Non-linear path of development:
Multiple paths



Merging branches



feature branch was merged into main ("recursive")

C6 is a "merge commit" - has more than one parent

git commits, branches and merges

- Not every automatic merge happens without conflict – sometimes you have to manually resolve the conflicts and commit
- To merge a branch using the command line: switch to main using `git checkout main` and then merge `git merge my-branch-to-merge`
- To delete a branch using the command line: `git branch -d my-branch-to-delete`
- To list your branches: `git branch` (the * marks which branch you have currently checked out)
- Display branches on remote: `git branch -r`
- Delete branch on remote: `git push origin –delete my-branch-to-delete`
- Other useful git commands: `git diff` and `git status`

https://dangitgit.com/en

git workflow and git commands

Workflow will be demonstrated during the life session!!

1. `git clone the-repository-URL` to clone an exisiting repository from the remote server or turn a local directory into a git repository through `git init`
2. create a branch in your local repository where you make your changes to the code through `git branch your-branch-name` and then checking out the branch through `git checkout your-branch-name`
3. make your changes in your branch
4. stage your changes through `git add your-changed-files`
5. commit your changes to your local repository through `git commit –m "your-meaningful-commit-message"`
6. push your changes to the remote repository through `git push` or `git push –u origin your-branch-name` if your local branch is not yet initialized on the remote

Unit 1: Introduction to git and GitHub

- *What is git?*
- *What is GitHub?*
- *Creating branches*
- *Merging branches*
- Jupyter notebooks
- Pre-commit hooks
- Useful git commands

Jupyter Notebooks

- Jupyter Notebooks provide a web-based interactive computational environment
- An interesting perspective on interactive notebooks can be found here: https://www.theatlantic.com/science/archive/2018/04/the-scientific-paper-is-obsolete/556676/
- Jupyter Notebooks support different kernels, we will use the iPython kernel (the Python execution backend of Jupyter)
- Possible to write both code and markdown (explanatory text), generate visualizations,… - **very useful as the first step in the life cycle of scientific software ("the toddler")**

Jupyter Notebooks

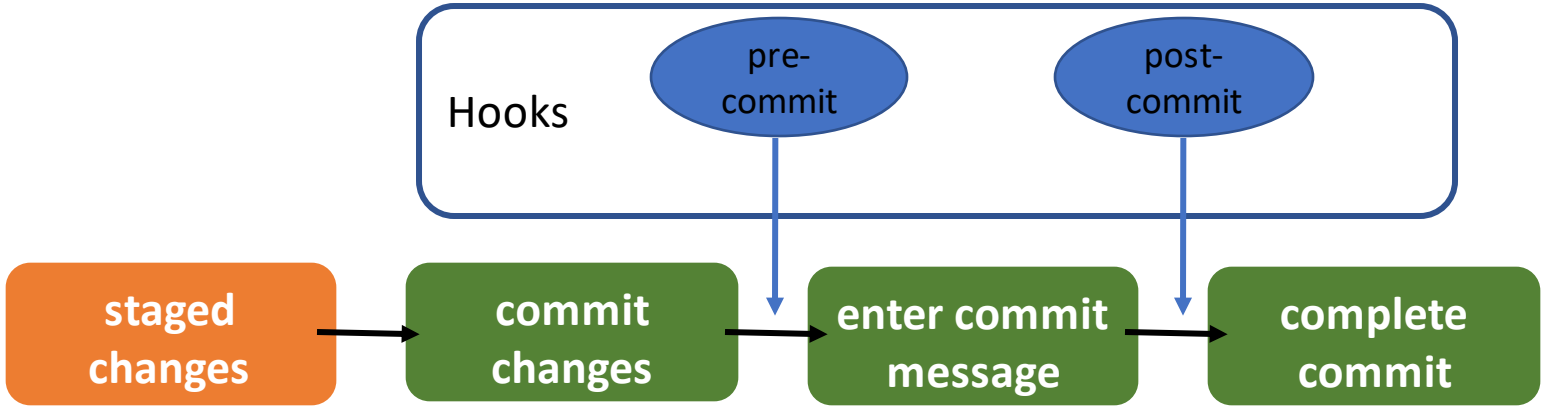
A quick demonstration …

Unit 1: Introduction to git and GitHub

- *What is git?*
- *What is GitHub?*
- *Creating branches*
- *Merging branches*
- *Jupyter notebooks*
- Pre-commit hooks
- Useful git commands

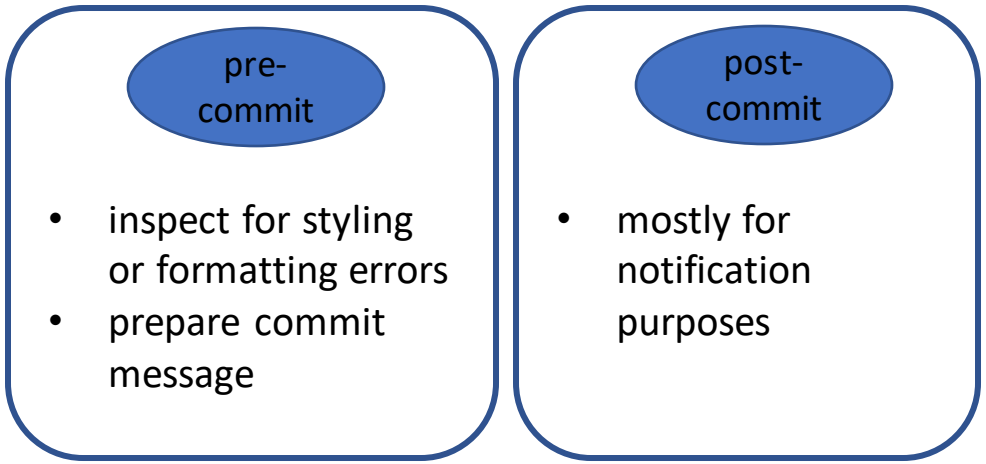
git hooks

- hooks are scripts that run automatically upon a particular event
- hooks live in the `.git/hooks` directory (take a look at your .git/hooks directory to see which sample hooks are there. If you want to activate one of them, remove the "`.sample`" extension and make the script executable).
- hooks are local: they are not copied upon `git clone`! Maintaining hooks and ensuring everyone is using the same hooks is tricky.
- *Usually hooks are used to ensure clean commits* (i.e. ensure the commit message contains a certain amount of information, check for stylistic errors, etc).



git hooks

- Use to encourage certain commit policies



Jupyter notebooks on github

- The thing about Jupyter notebooks and git: The diffs that usually show in a pull request, and generally any diffs between commits are rendered as JSON, which is somewhat readable, but the images (binary) lead to problems. There is a variety of tools that address this problem; we will be using pre-commit hooks (nbstripout) to clear the notebooks before committing to the remote repository.
- Diffs blow up git history

https://nextjournal.com/schmudde/how-to-version-control-jupyter

Use a package manager for your hooks

https://github.com/kynan/nbstripout

- We will use pre-commit for managing our hooks.

```
pip install pre-commit
```

```
pip install --upgrade nbstripout
```

- Add a .pre-commit-config.yaml file to your repository with the contents:

```
repos:
- repo: https://github.com/kynan/nbstripout
  rev: 0.5.0
  hooks:
    - id: nbstripout
```

- Run `pre-commit install` to install the hook.
- Feel free to add additional useful hooks.

  https://github.com/pre-commit/pre-commit-hooks

Unit 1: Introduction to git and GitHub

- *What is git?*
- *What is GitHub?*
- *Creating branches*
- *Merging branches*
- *Jupyter notebooks*
- *Pre-commit hooks*
- Useful git commands

Useful git commands

- If you don't know your git, this page may be helpful: https://dangitgit.com/
- There is a git cheat sheet here: https://education.github.com/git-cheat-sheet-education.pdf
- We will later use VSCode (or your preferred IDE), which does not require you to know all the commands by heart. But it is good to have a basic understanding of what these do – there are situations when you may need them still.

Unit 1: Introduction to git and GitHub

- *What is git?*
- *What is GitHub?*
- *Creating branches*
- *Merging branches*
- *Jupyter notebooks*
- *Pre-commit hooks*
- *Useful git commands*

Live lesson

- We will work on some code examples in git repositories during the live lessons. You will need git, Python and Jupyter Notebooks installed on your computer. You will need numpy, pandas and seaborn.
- Please check the "preparing for the class" document and section of the video.

Live lesson - Demonstrations

- The following demonstrations will take place in the beginning of the live session:
  - Set up a git repository
  - Interact with GitHub as remote server
  - Install and use pre-commit hooks