Git and Github seminar

Daniel de Andrés Hernández, dpto: Física Experimental de Altas Energías
Tomás Sánchez Sánchez-Pastor, dpto: Computación Científica

GOBIERNO DE ESPAÑA
MINISTERIO DE ECONOMÍA, INDUSTRIA Y COMPETITIVIDAD
Ciemat
Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas

Git and GitHub

in a nutshell

- 1. Basics
  - Git locations
  - Commit model
  - Branching
  - Merging
- 2. Advanced functions:
  - Rebase
  - Stash
- 3. Workflows

BASICS

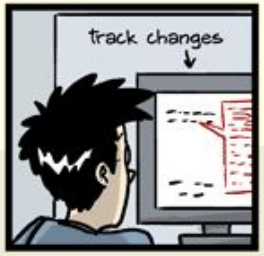
"FINAL".doc

FINAL.doc!

FINAL_rev.2.doc

FINAL_rev.6.COMMENTS.doc

FINAL_rev.8.comments5.
CORRECTIONS.doc

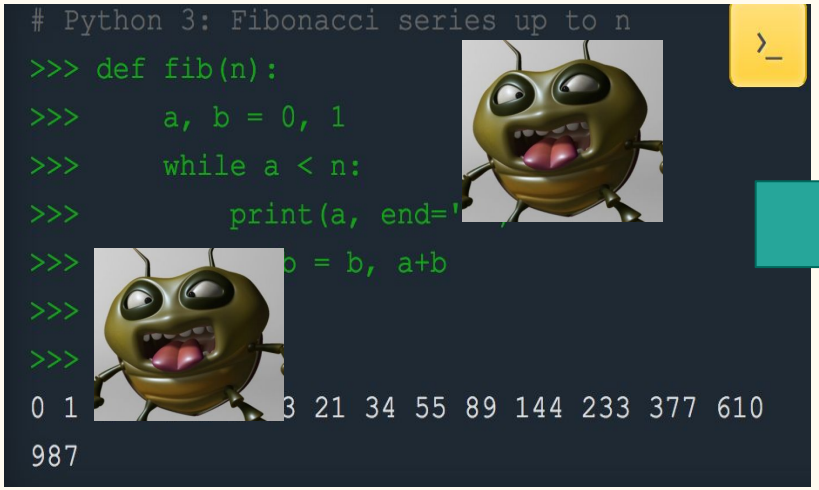


FINAL_rev.18.comments7.
corrections9.MORE.30.doc

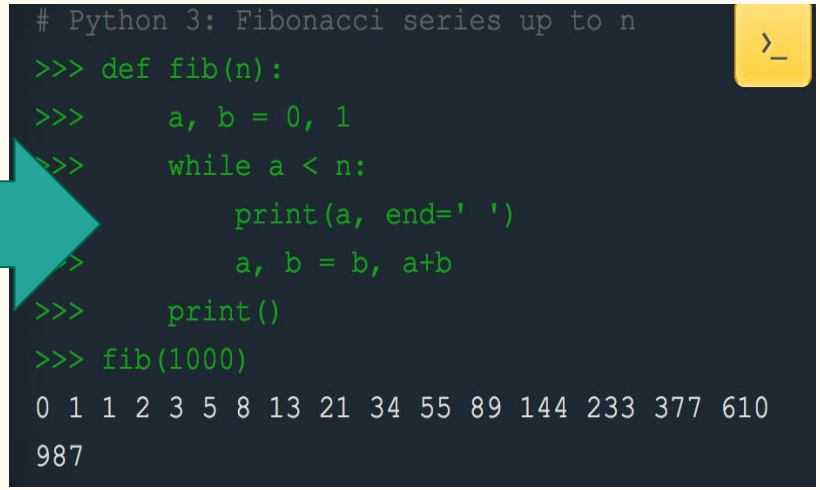
FINAL_rev.22.comments49.
corrections.10.#@$%WHYDID
ICOMETOGRADSCHOOL????.doc

Git and GitHub in a nutshell

- Git allows us to continuously improve our project.
- Each **commit** is a snapshot of the project at a given time.
- We can review the project history (the commits) and undo the changes.
- Git is very efficient.



20 Files



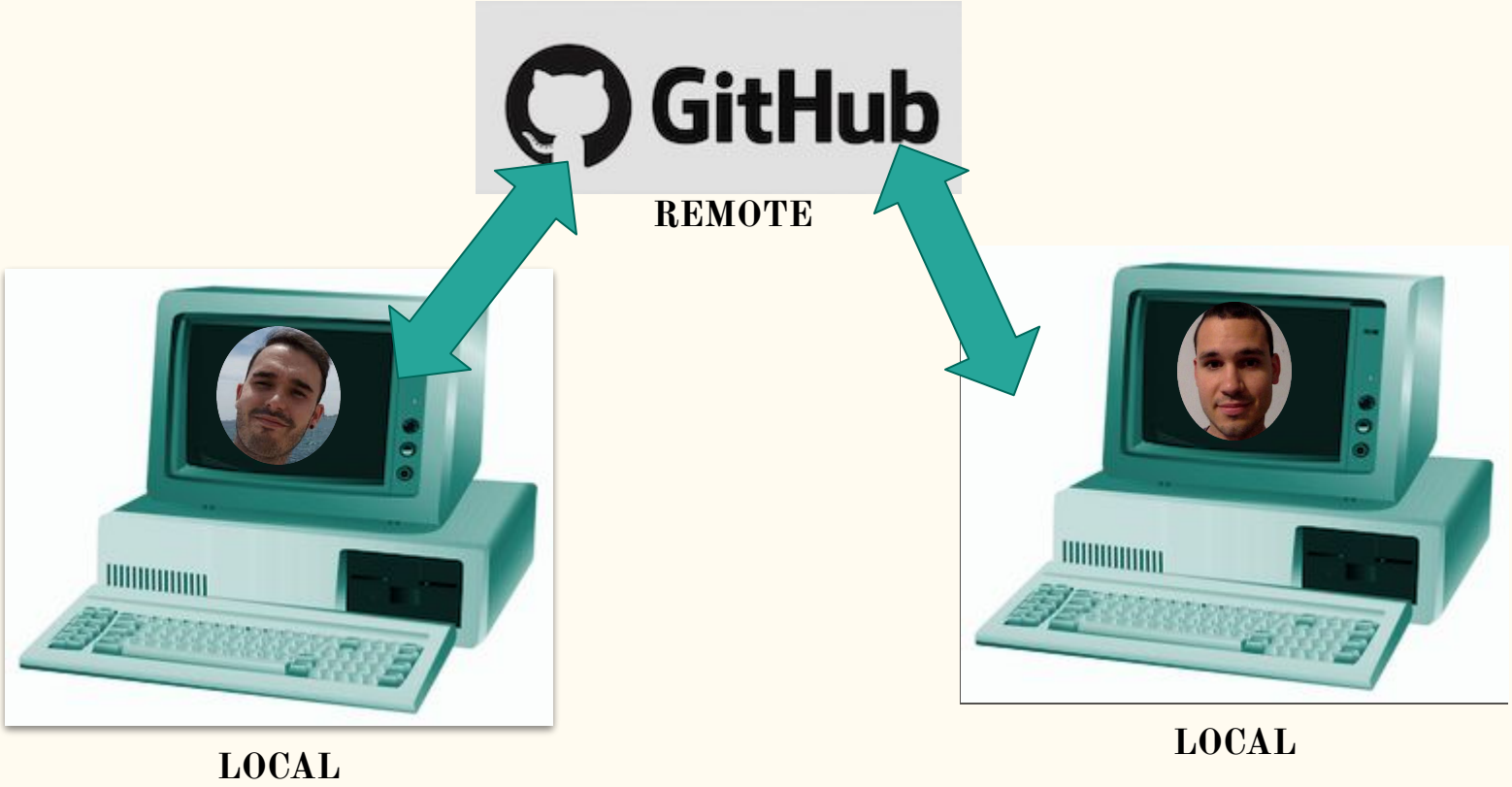
21 files

COMMITS

1 new file is stored

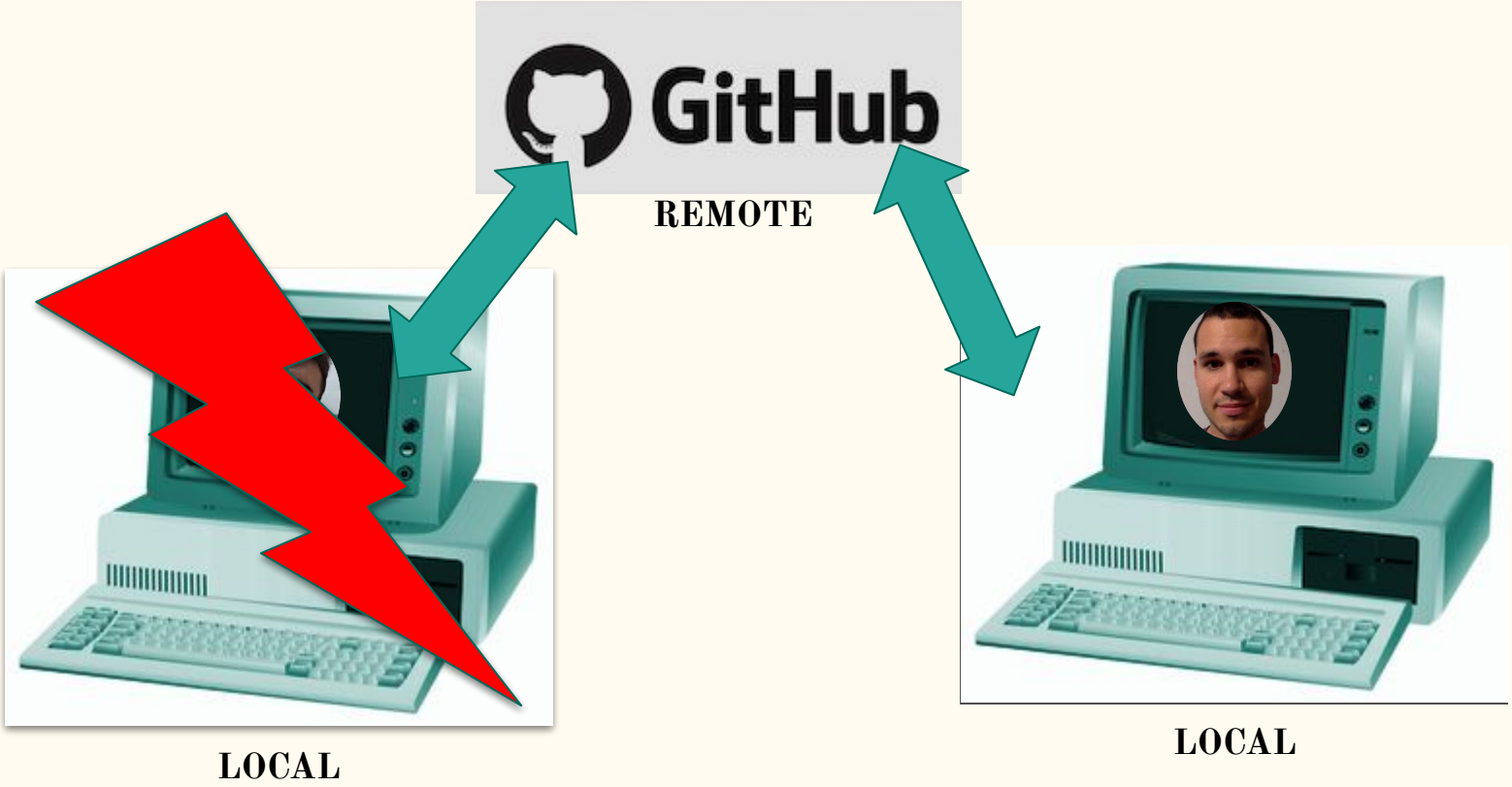
Git is a distributed version control system

- Each user has a local project history (repository).
- There is a single remote repository that is consider the source of truth.
- Easily synchronise: pull and push commands.



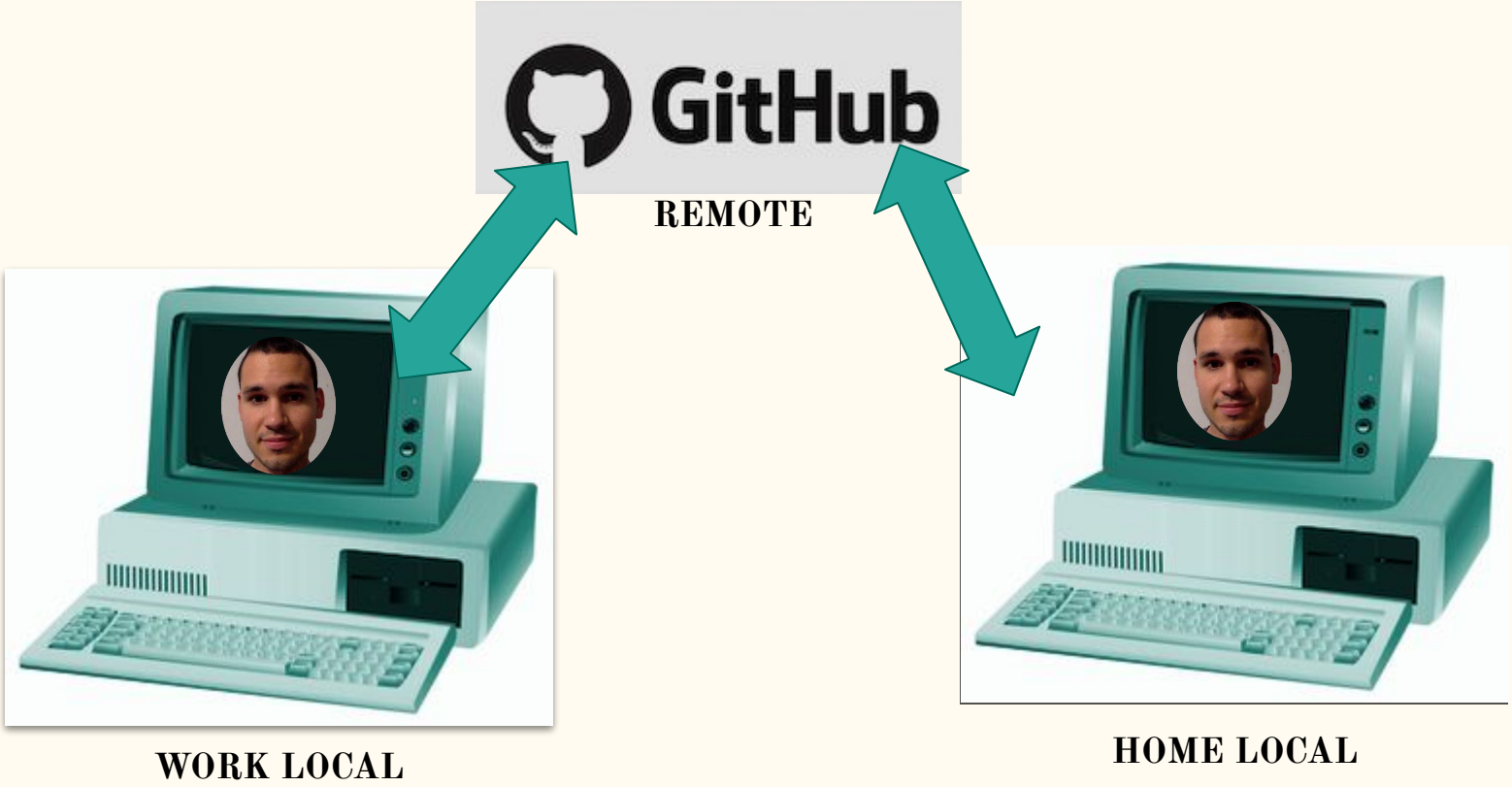
Git is a distributed version control system

- Each user has a local project history (repository).
- There is a single remote repository that is consider the source of truth.
- Easily synchronise: pull and push commands.



Git is a distributed version control system

- Each user has a local project history (repository).
- There is a single remote repository that is consider the source of truth.
- Easily synchronise: pull and push commands.

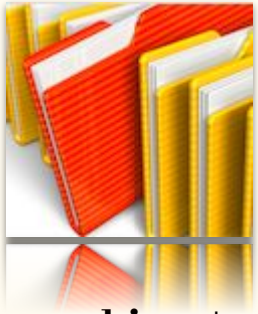


Git locations

- **Working Tree**
- **Staging area / index**
- **Local repository**
- **Remote repository**

Git locations

- **Working Tree :** A directory in your computer that contains the files of a single commit.

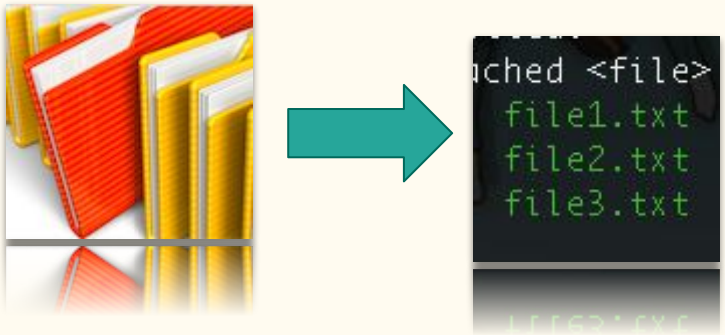


working tree

You can access to the working tree (i.e. the files, code, etc) of every commit.

Git locations

- **Working Tree :** A directory in your computer that contains the files of a single commit.
- **Staging area / index :** Files that will be in the next commit.



working tree **staging area**

use the command **git add [file]** to select the files which are going to be added in the next commit.

Git locations

- **Working Tree :** A directory in your computer that contains the files of a single commit.
- **Staging area / index :** Files that will be in the next commit.
- **Local repository :** Contains all the commits of the project.

working tree

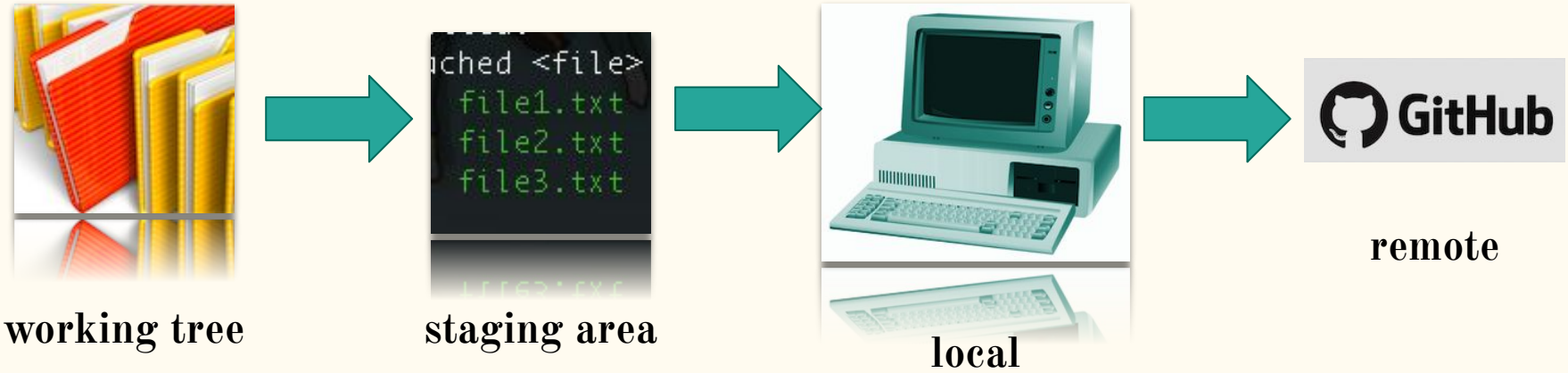
staging area

local

commit the files in the local repository using **git commit -m" message"**

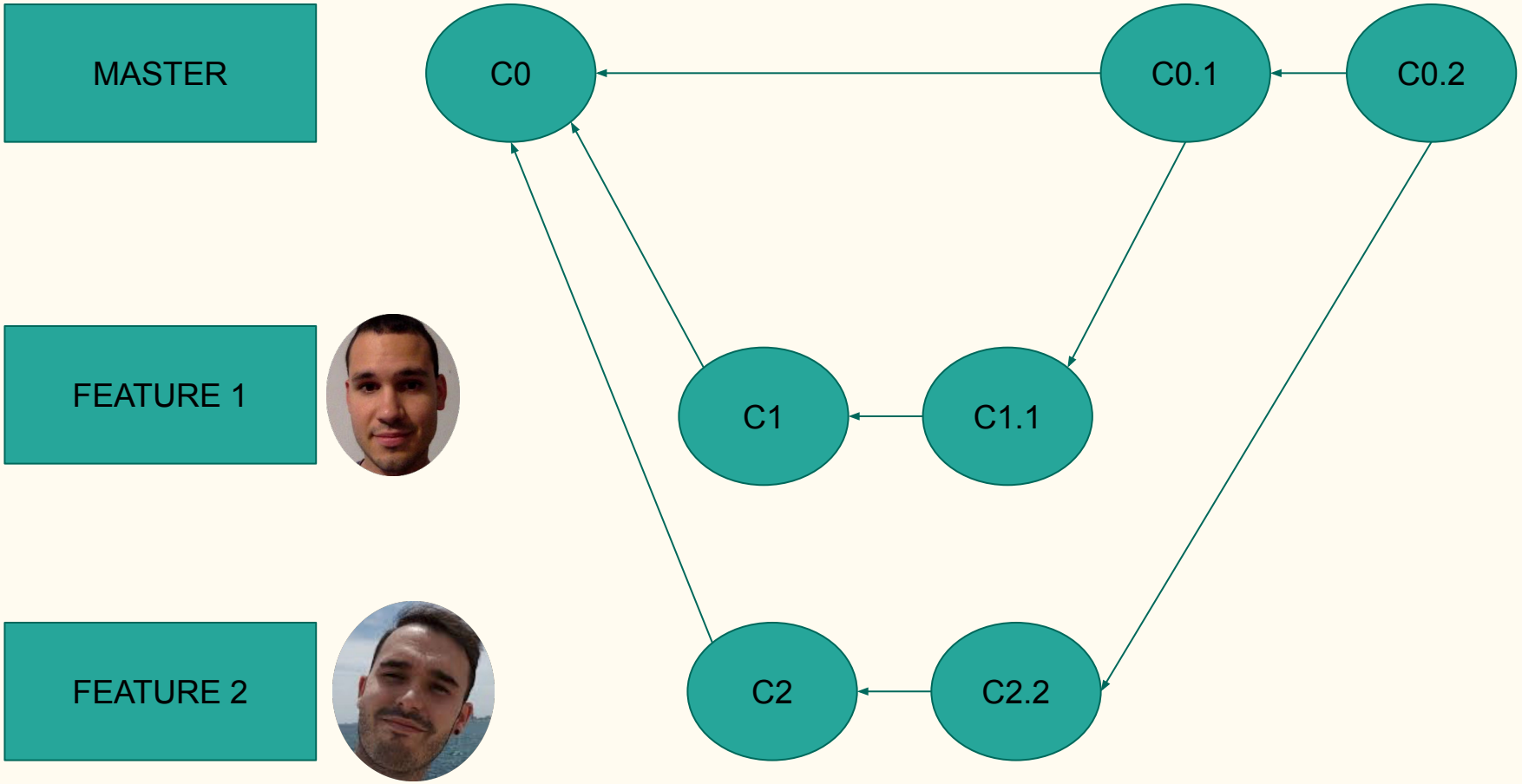
Git locations

- **Working Tree :** A directory in your computer that contains the files of a single commit.
- **Staging area / index :** Files that will be in the next commit.
- **Local repository :** Contains project's local commits.
- **Remote repository :** Contains project's truth-commits.



upload and download with **git push** and **git pull** respectively.

Git commit model



Hands on : git basics, repositories and commits

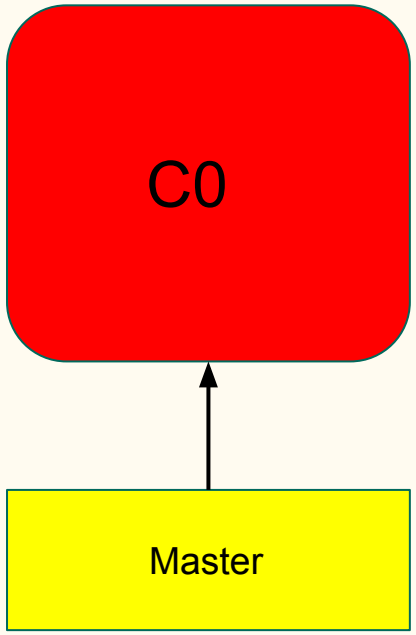
1. REPOSITORIES

- Create a remote repository using GitHub
- Clone a remote repository
- Create a local repository
- Upload your local content to a remote repository

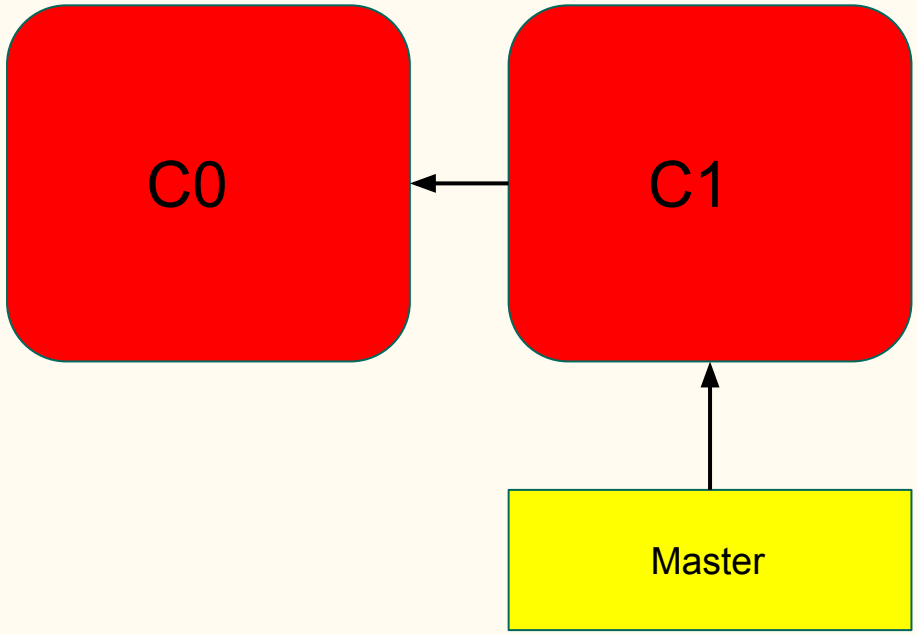
2. COMMIT AND LOCATIONS

- Familiarize with commit model
- References & SHA-1s in Git
- Commit to a local repository
- Push to a remote repository
- Retrieve an older commit

Branching: branches as movable pointers to commits



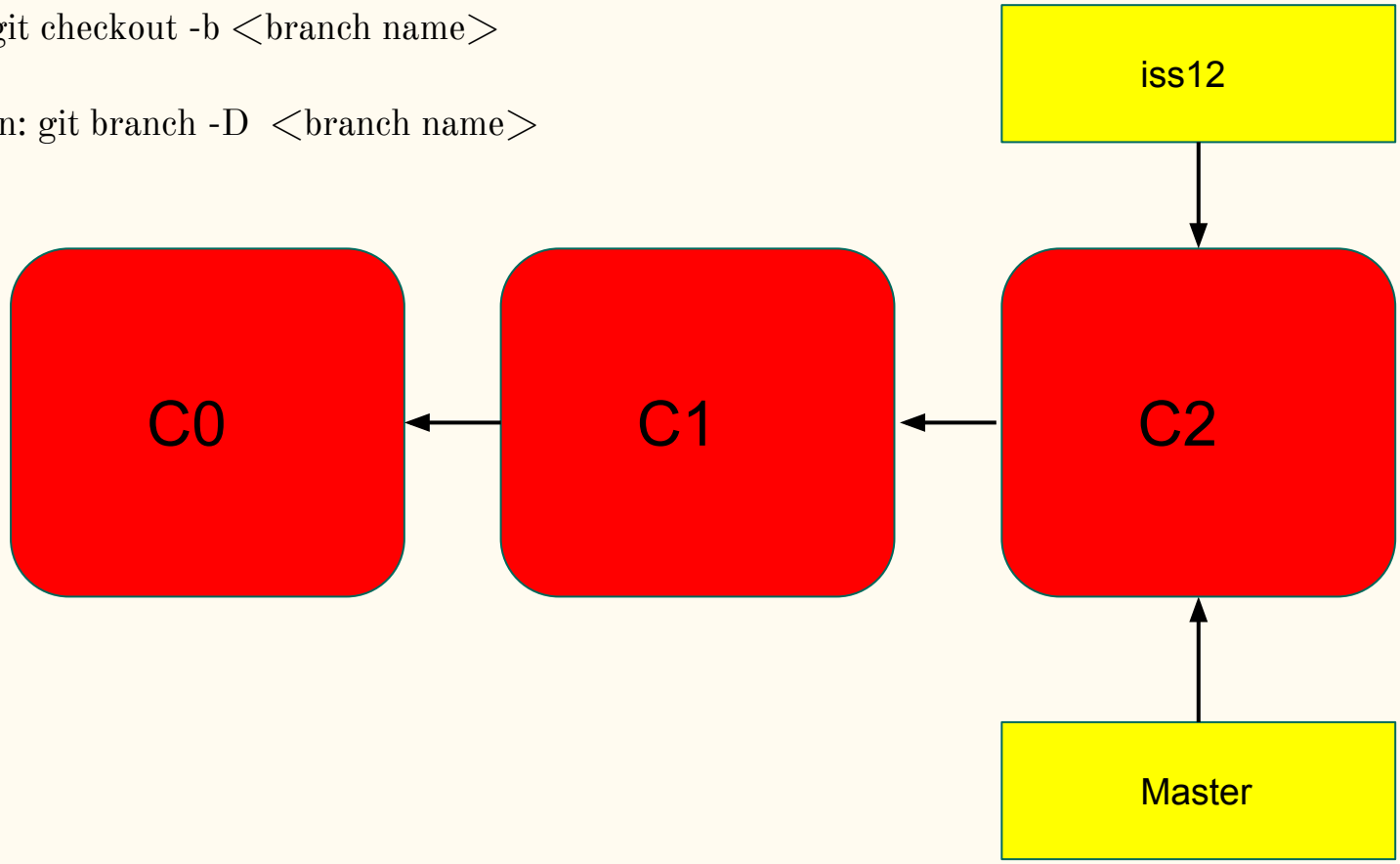
Branching: branches as movable pointers to commits



Branching: branches as movable pointers to commits

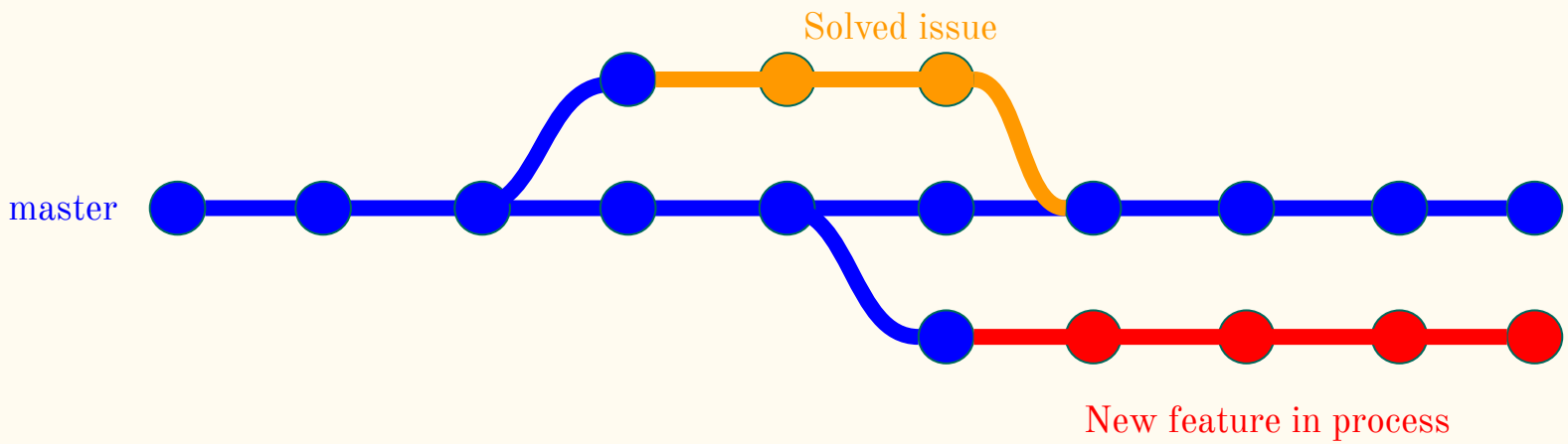
Creation: git checkout -b <branch name>

Destruction: git branch -D <branch name>



Branching

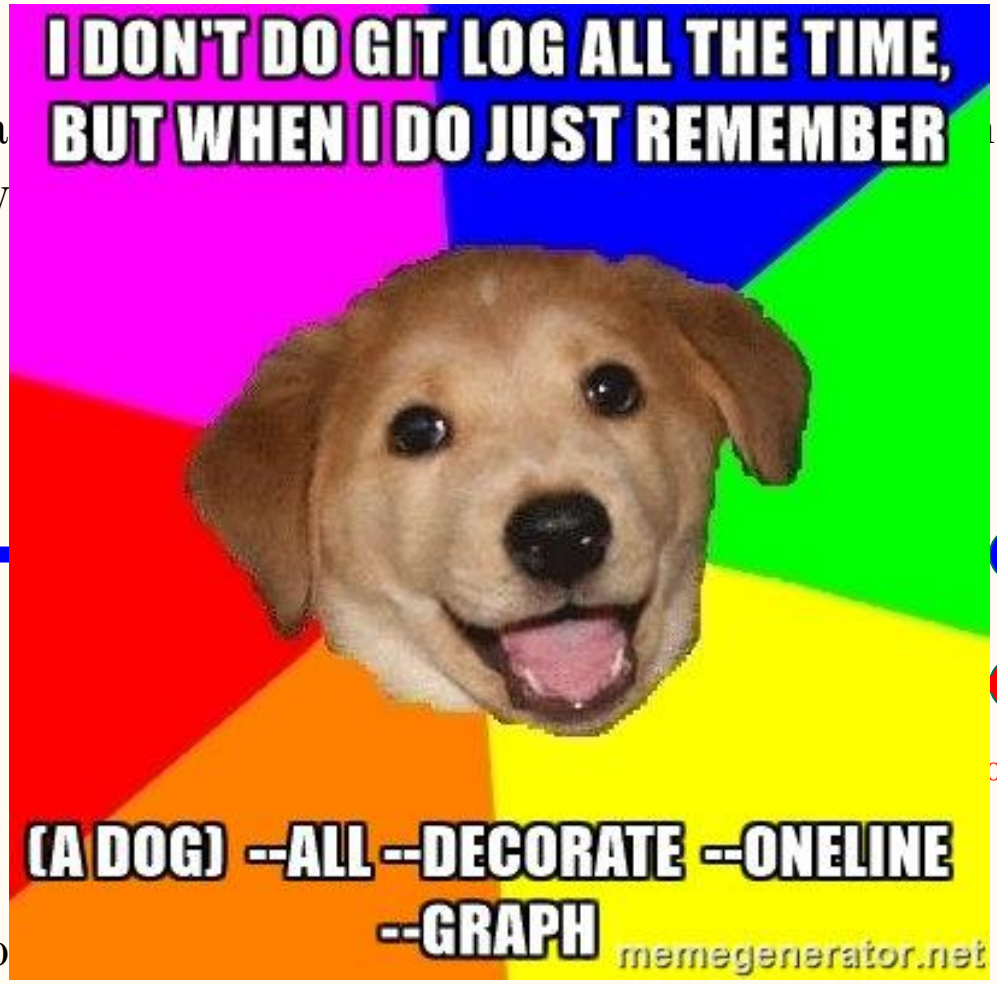
- The master branch is not special, is just the default name when you **git init** a repository. Nevertheless, for convenience, master must be your reference.



- Try **git adog** to show a graph of the branches history.

Branching

- The master bra you **git init** a repository. Nev reference.

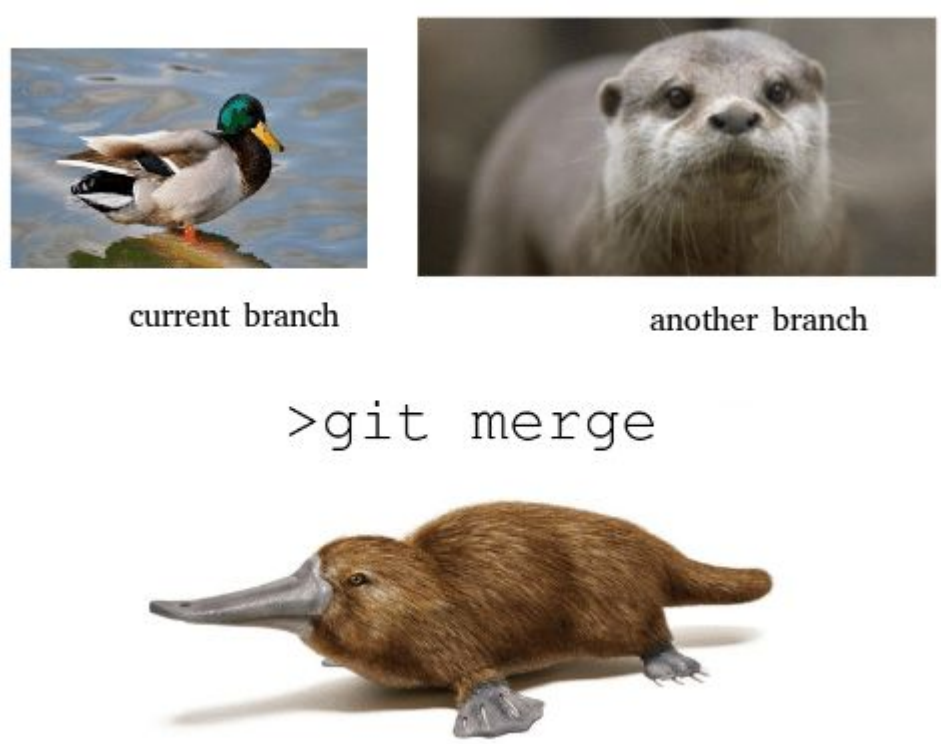


- Try **git adog** to

Merging

- **git merge** is a fundamental operation applied to two branches that put together every change that has been made into a single branch.
- Warning! Merge is a source of conflicts!

Expectation



Reality

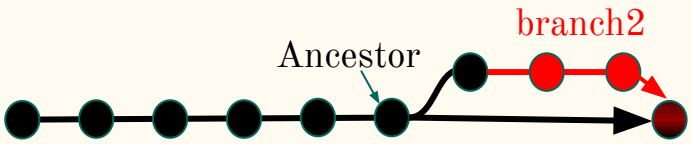
Merging

- **git merge** is a fundamental operation applied to two branches that put together every change that has been made into a single branch.
- Warning! Merge is a source of conflicts!

Expectation

Reality

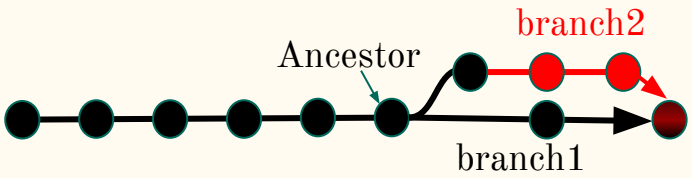
Merging: fast forward



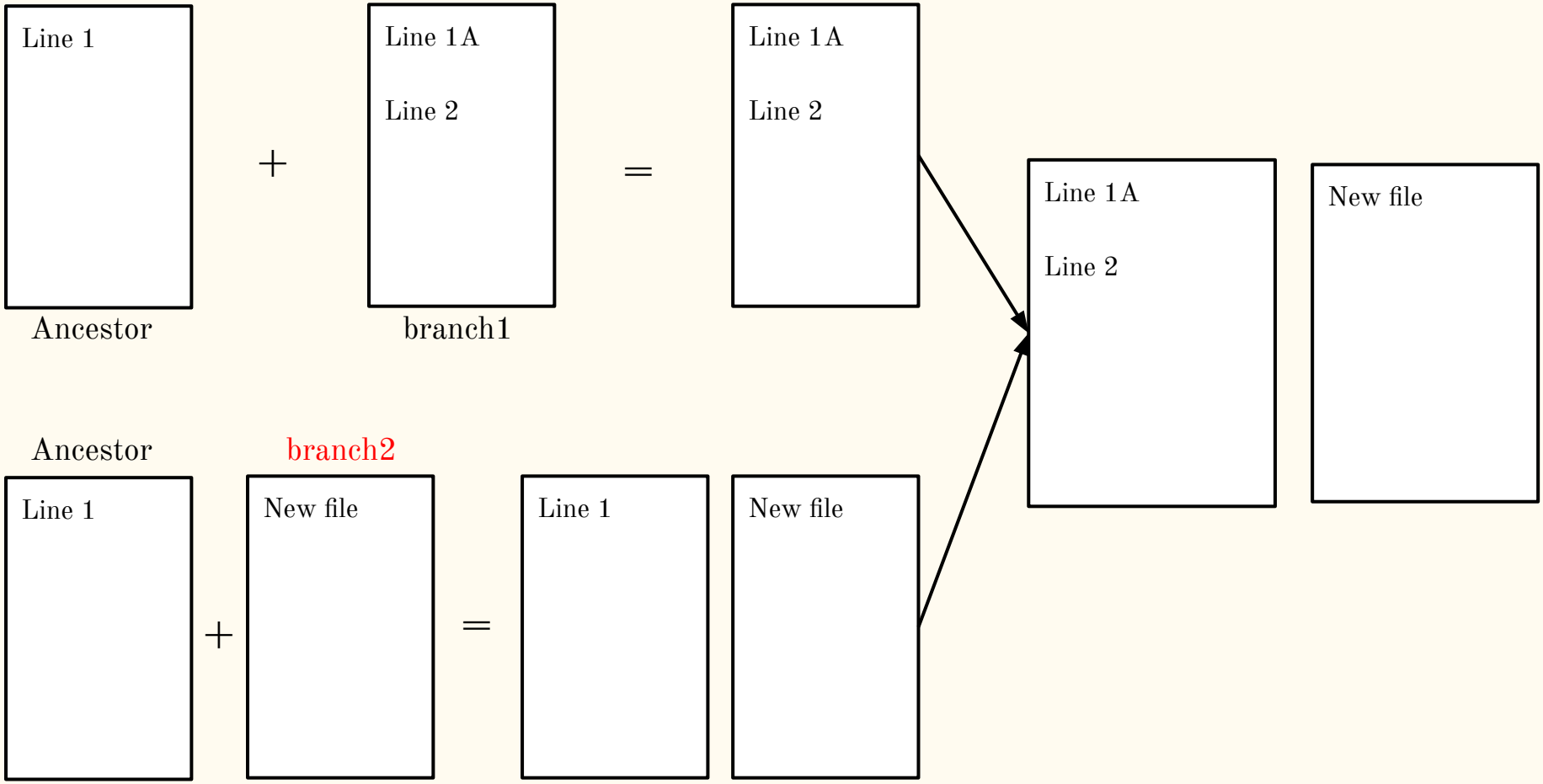
- When Git merges, firstly tries a fast forward.

Ancestor		branch2		
• Line 1 • Line 2	+	• Line 1 • Line 2 • Line 3	=	• Line 1 • Line 2 } common • Line 3 difference

Merging: 3-way merge



- If not possible, tries a 3-way merge.



Merging: tips

- Merge conflicts happen when you merge branches that have competing commits.
- Before merging, always check that you have pulled the remote repository.
- Delete one of the merged branches.
- NEVER merge with master.

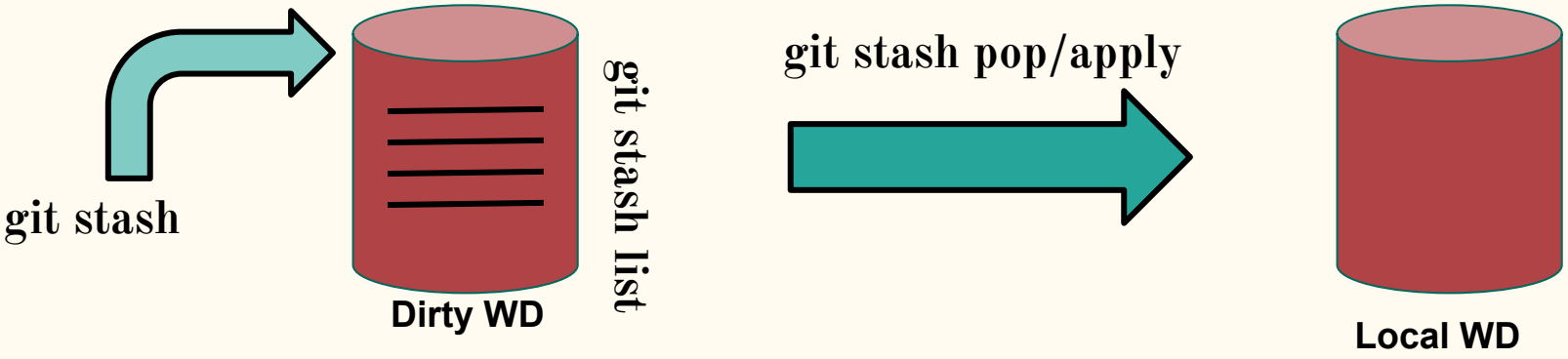
ADVANCED

Advanced functions: Stash

- git stash saves the uncommitted changes in a dirty working directory and let you recover them whenever you want.

- Useful for solving quick bugs and preventing pull errors.

- Main commands:



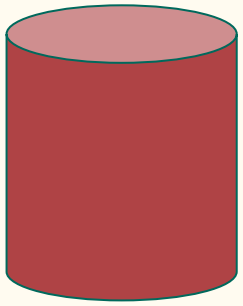
Advanced functions: Stash

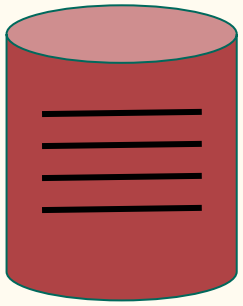
- git stash saves the uncommitted changes in a dirty working directory and let you recover them whenever you want.

- Useful for solving quick bugs and preventing pull errors.

- Main commands:

git stash clear

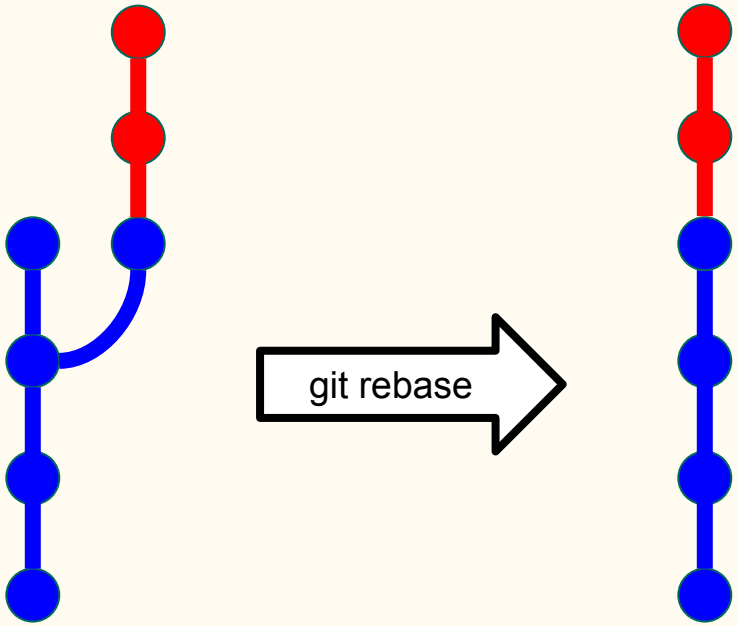




Advanced functions: Rebase

- git rebase <target branch> reapply commits on top of another branch.

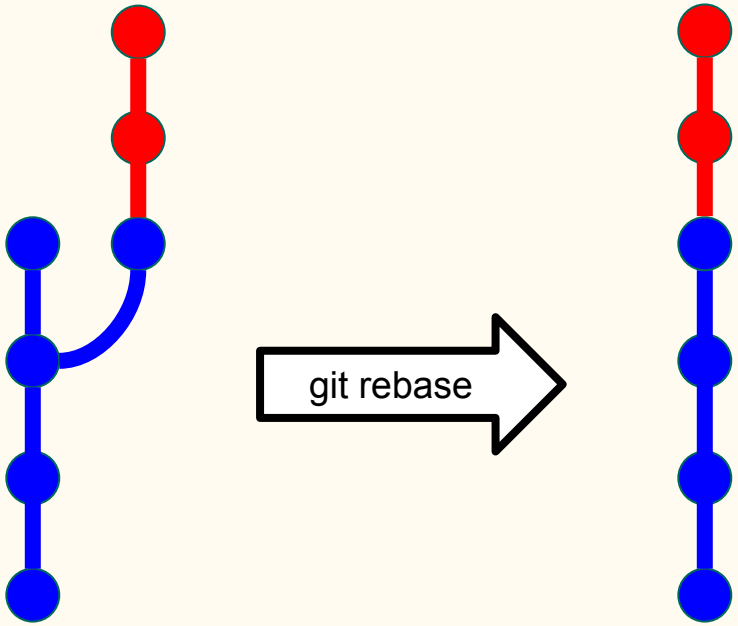
- Change the commit history and clean the graph.



Advanced functions: Rebase

- git rebase <target branch> reapply commits on top of another branch.

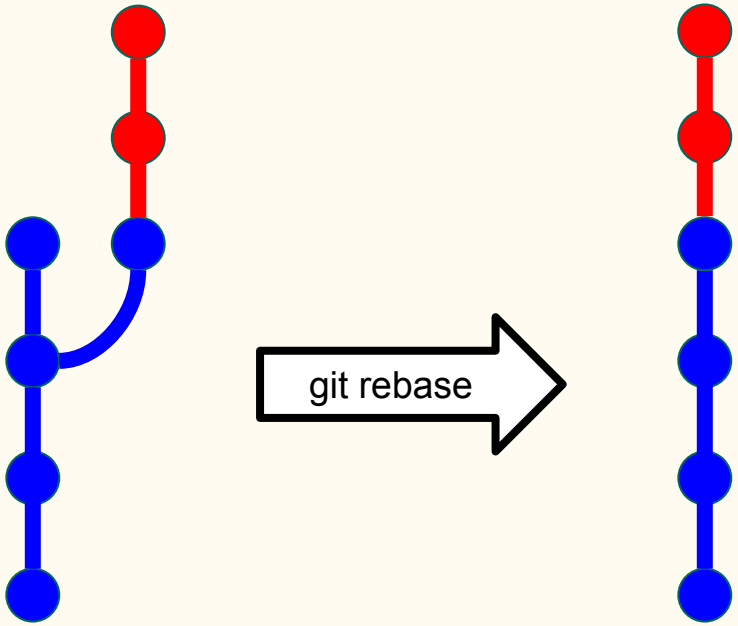
- Change the commit history and clean the graph.



Advanced functions: Rebase

- git rebase <target branch> reapply commits on top of another branch.

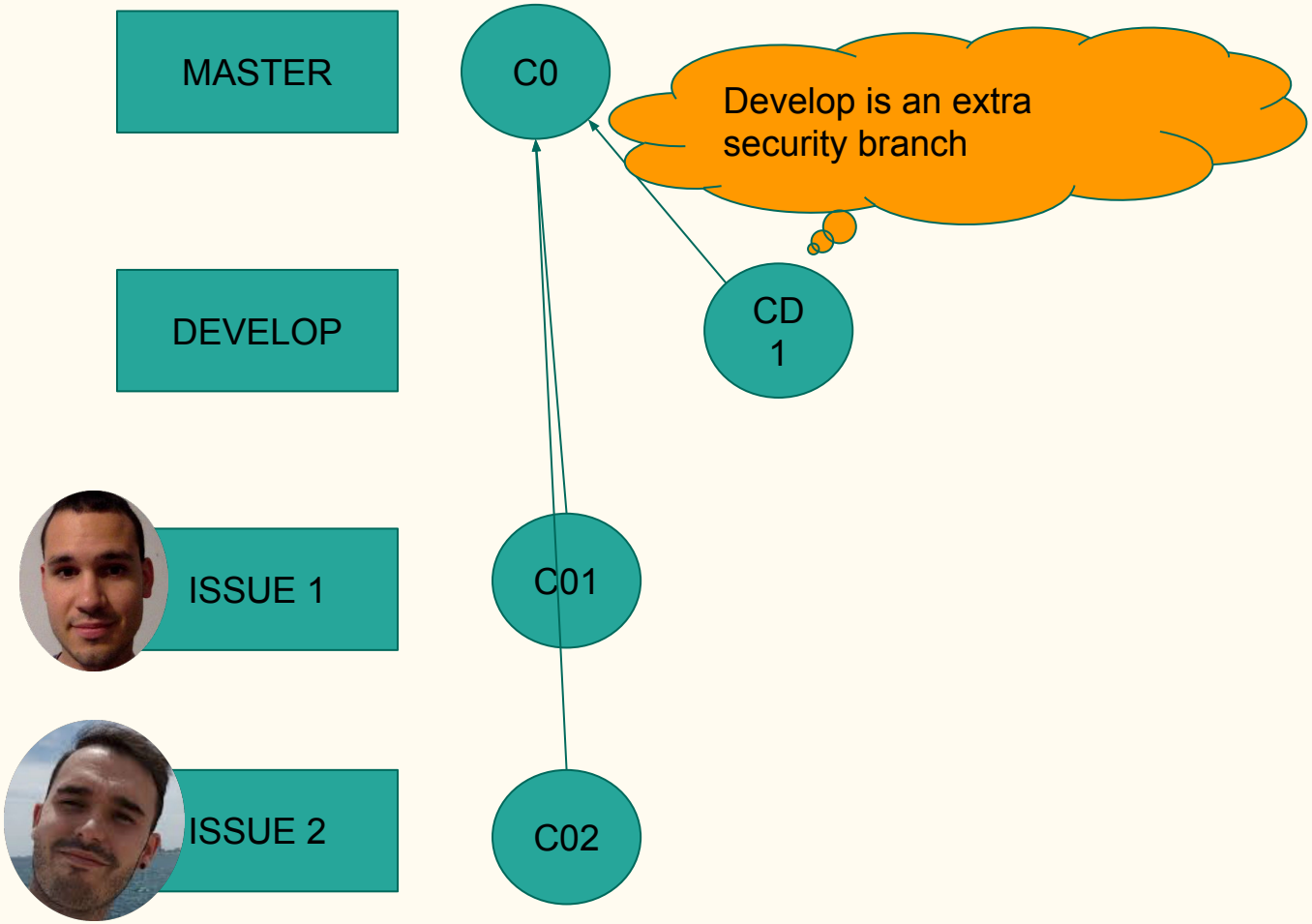
- Change the commit history and clean the graph.



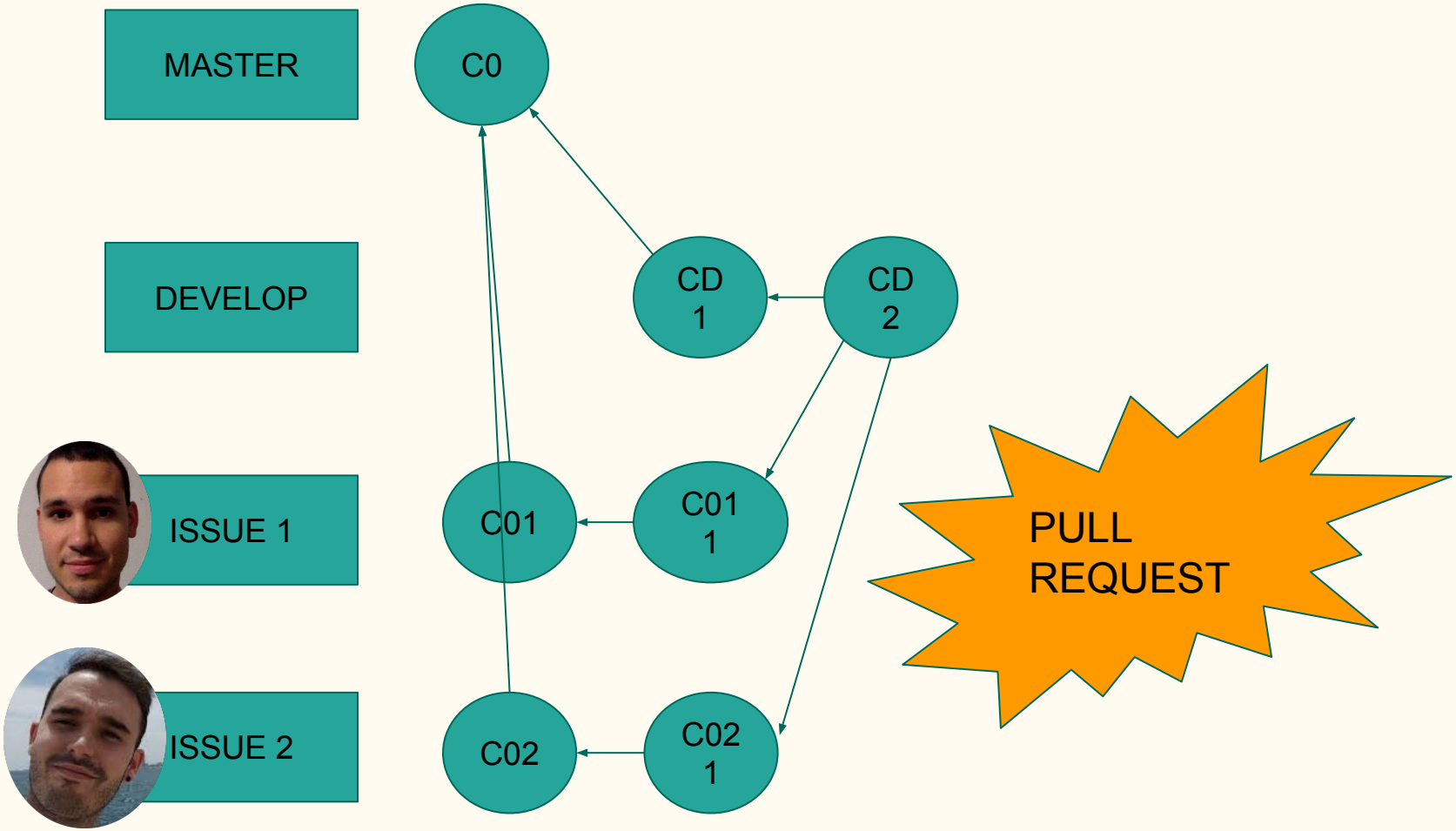
WORKFLOWS

Pull Requests and Issues

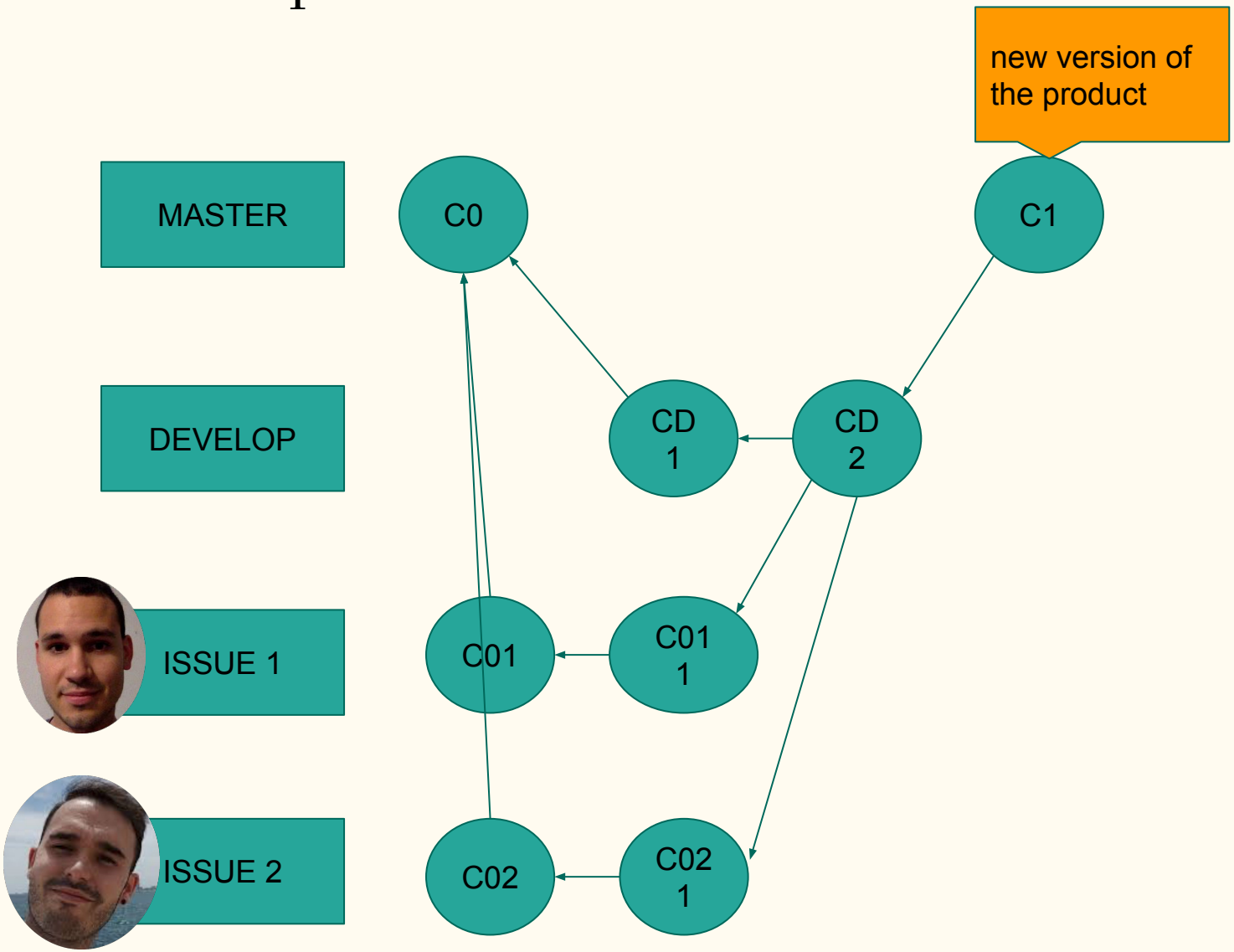
Pull Requests and Issues



Pull Requests and Issues



Pull Requests and Issues



Pull Requests and Issues