Code Versioning

Olivier Mattelaer (CISM/CP3)

based on slides from

Damien Francois (CISM)
Juan Cabrera (UNAMUR)
Jonathan Lambrechts (IMMC)
Scott Chalcon (git)

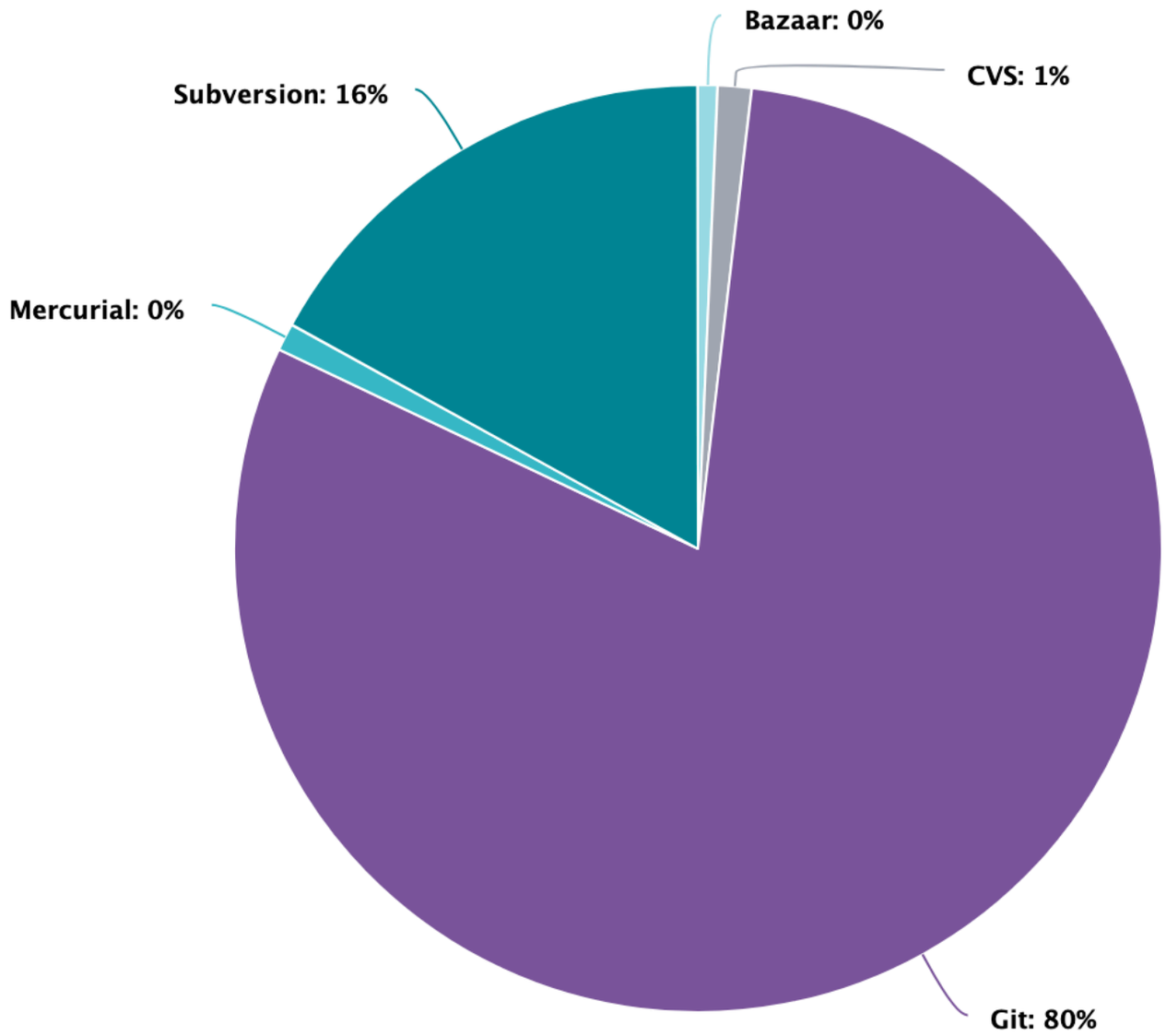
Game based on gameshell from

Pierre Hyvernat (SMB)
Rodolphe Lepigre (SMB)

What is code versioning



Why git?



https://www.openhub.net/repositories/compare

Number for 11/2026

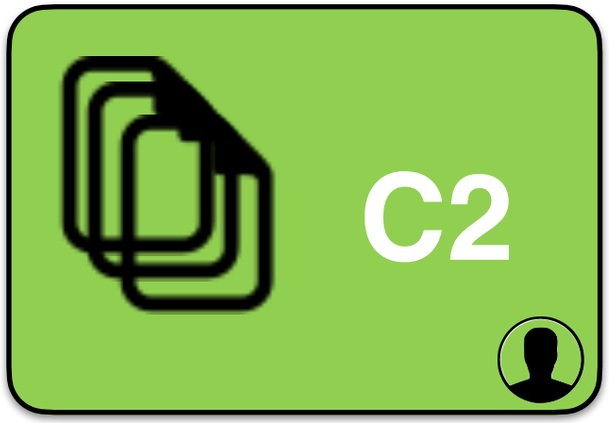
TODAY

Mix of "Theory" and game...

1. History of modification
    - Mission 1->6
2. Workflow
    - Mission 7->11
3. Team Work
    - Mission 12-> 16

1. Commit

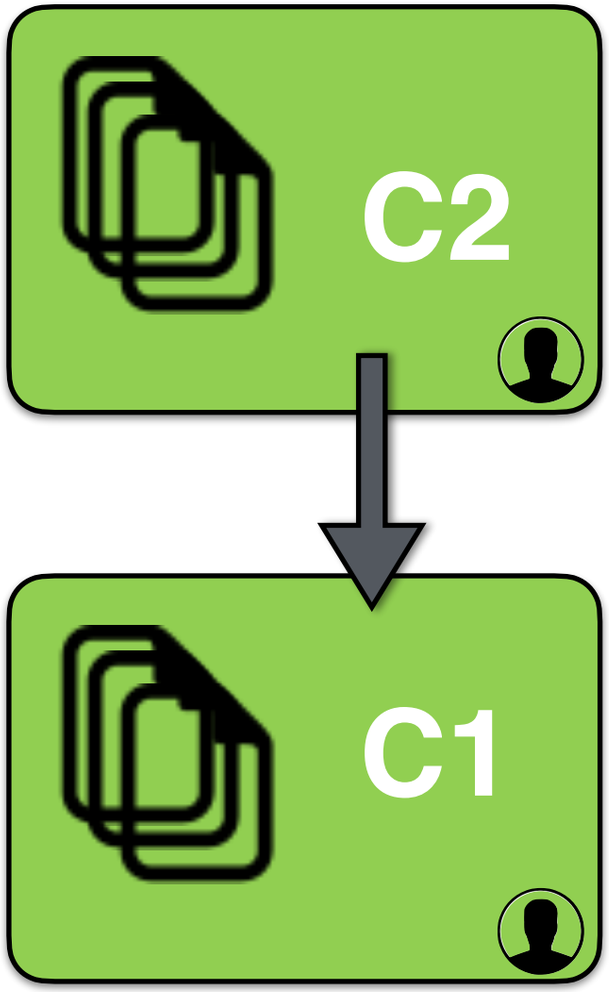
- An history: Is a **succession** of **snapshot** of your files at key time of their development
  - Each **snapshot** is called **COMMIT**



- Commit is
  - All the files at a given time
  - A unique name (SLHA1)
  - MetaData (who created/when/info)

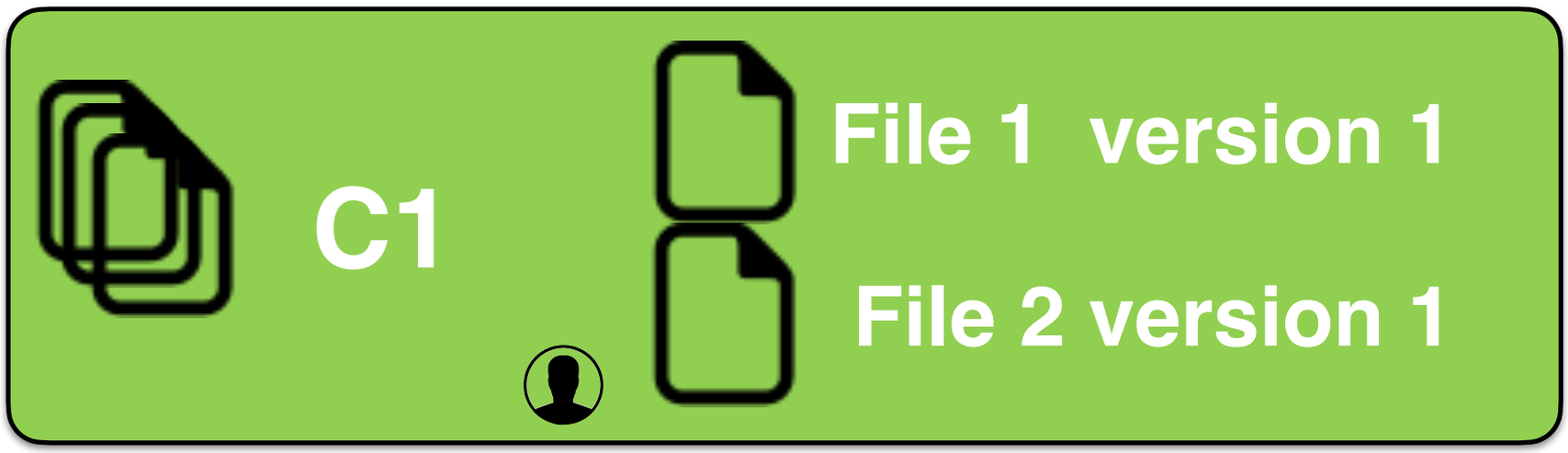
1. Commit

- An history: Is a **succession** of **snapshot** of your files at key time of their development
  - Each **snapshot** is called **COMMIT**

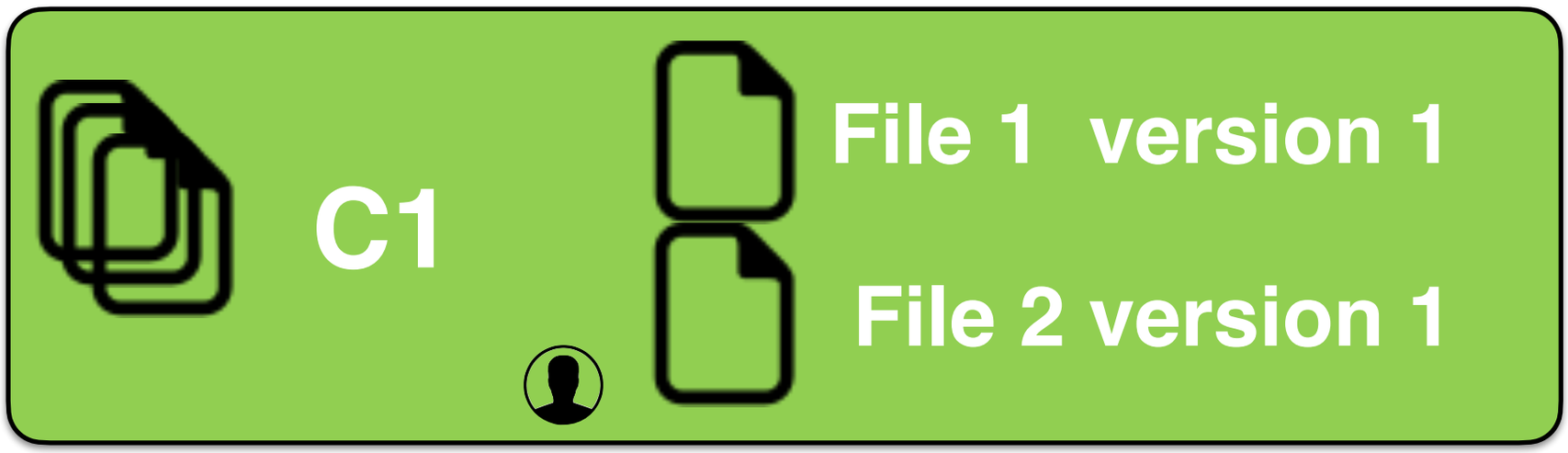


- Commit is
  - All the files at a given time
  - A unique name (SLHA1)
  - MetaData (who created/when/info)
  - Pointer to previous(es) commit

1. Commit

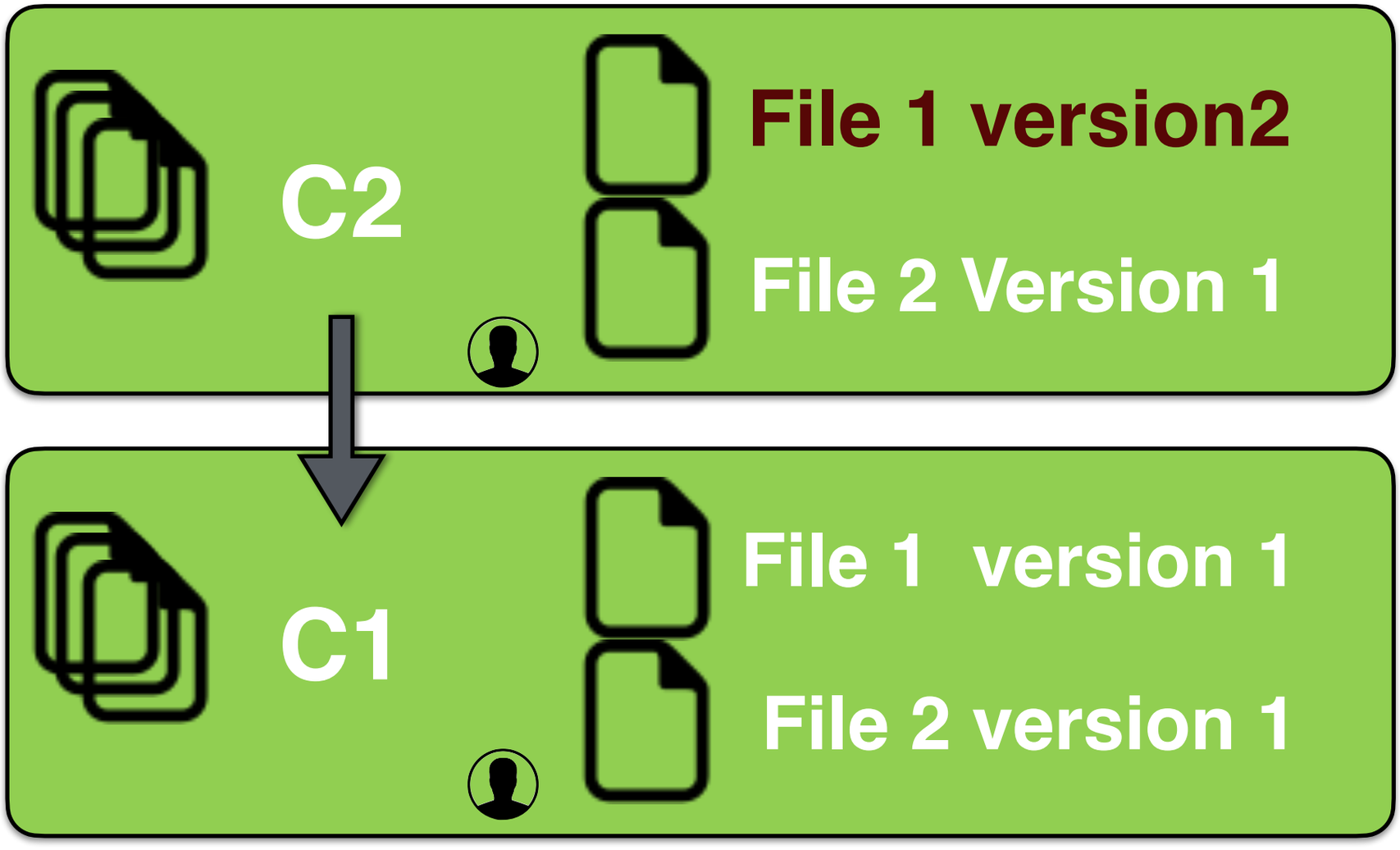


1. Commit



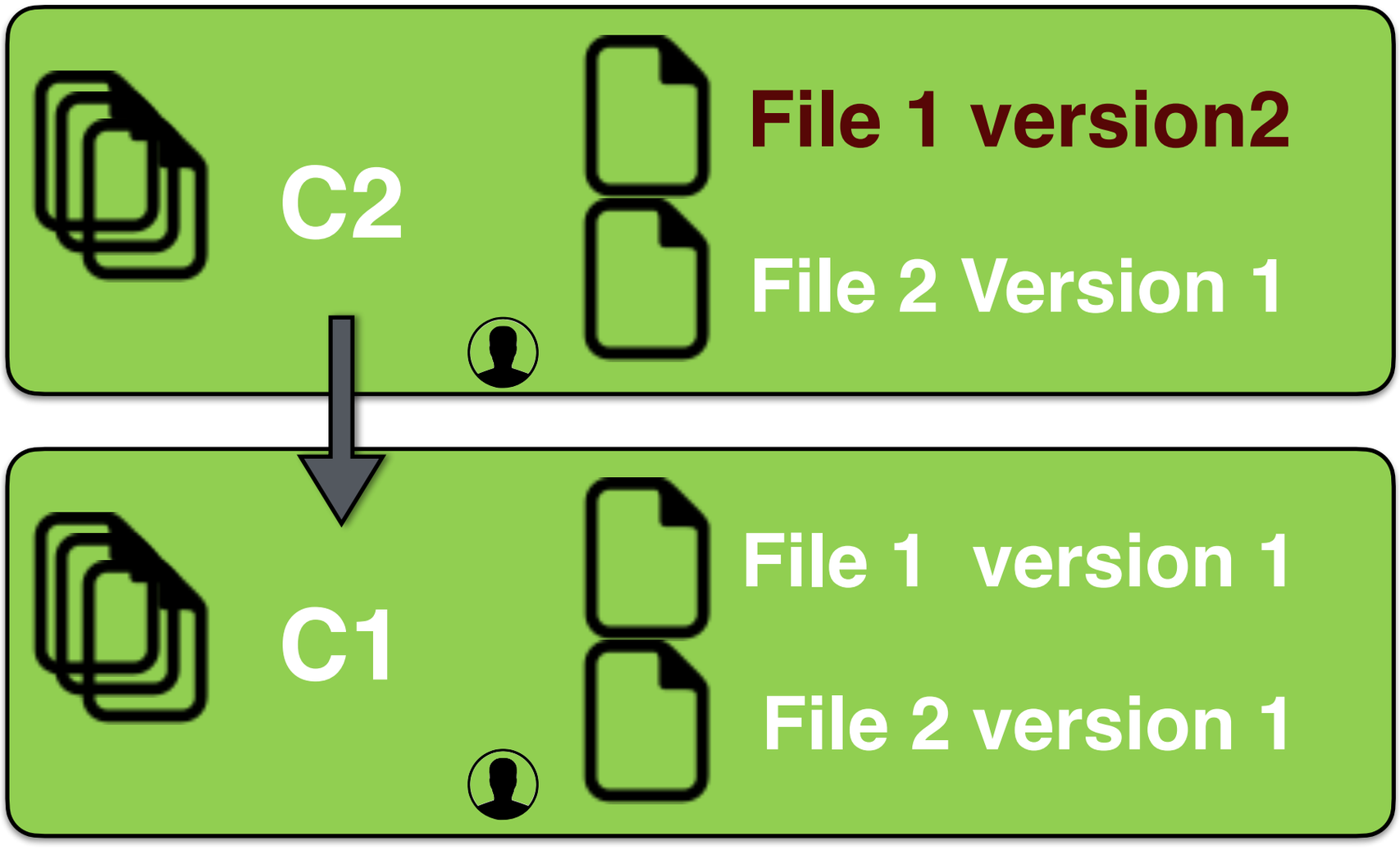
Edit file 1

1. Commit



Edit file 1

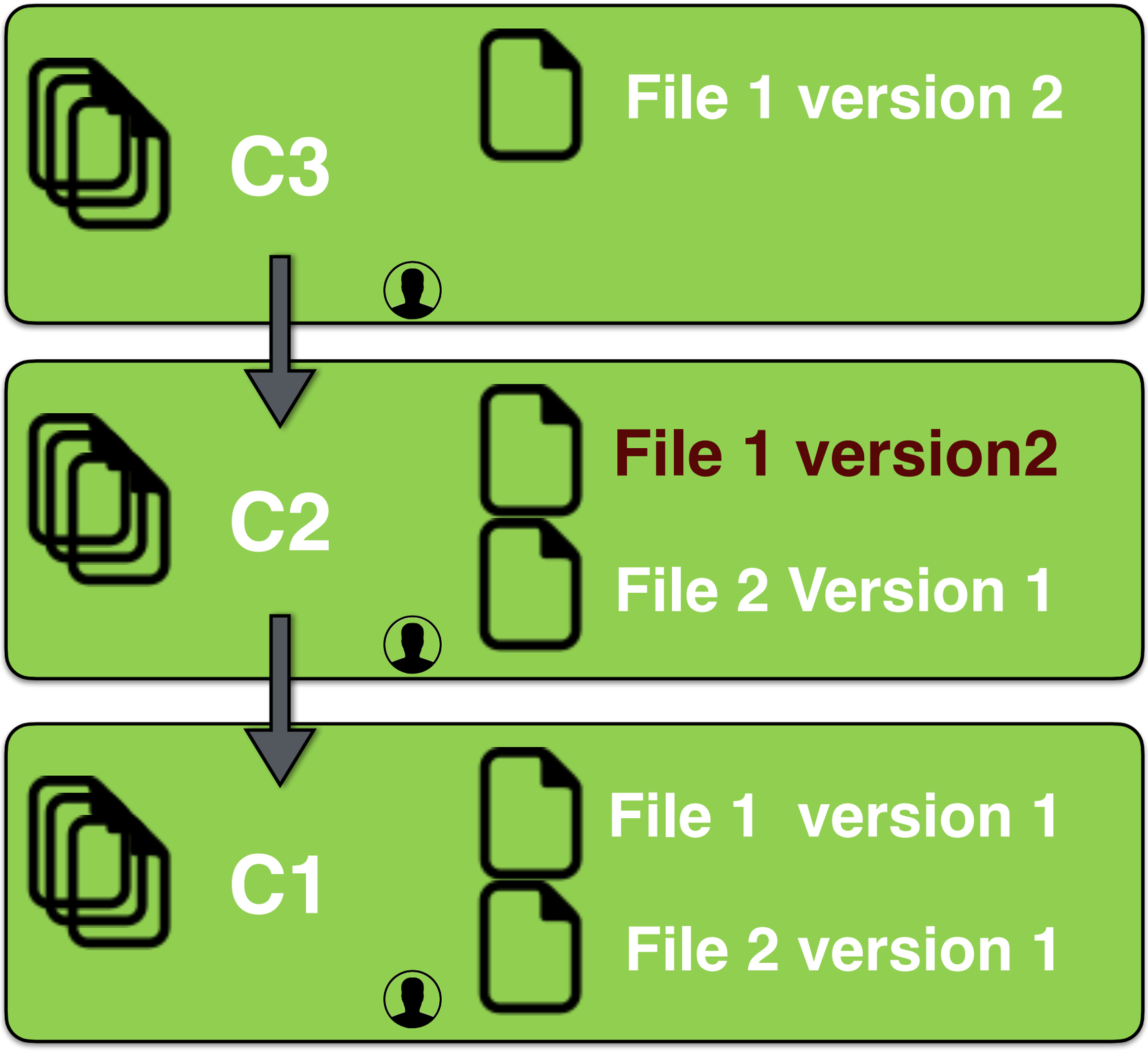
1. Commit



Remove file 2

Edit file 1

1. Commit


C3
File 1 version 2
Remove file 2
C2
File 1 version2
File 2 Version 1
Edit file 1
C1
File 1 version 1
File 2 version 1

1. Commit



1. Simplify representation of commit/history

Git Three area

Workspace

Index

Repository

Git Three area

Workspace

Index

Repository



./WORKDIR

Git Three area

Workspace

Index

Repository





./WORKDIR

Git Three area

Workspace

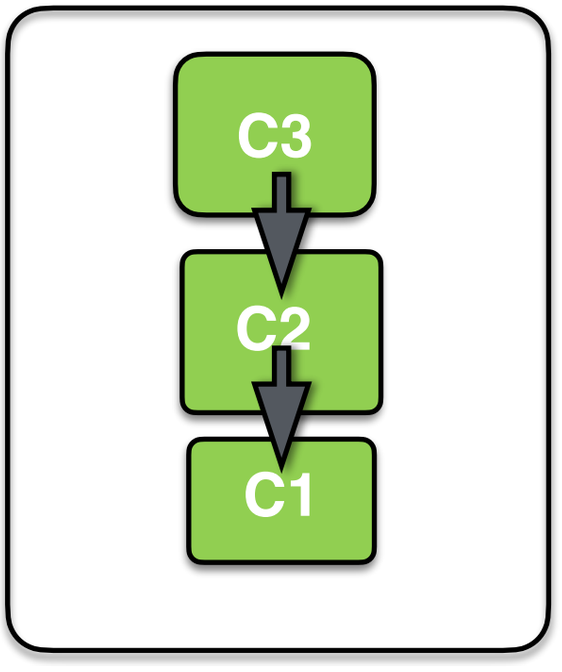
Index

Repository





./WORKDIR



Git Three area

Workspace

Index

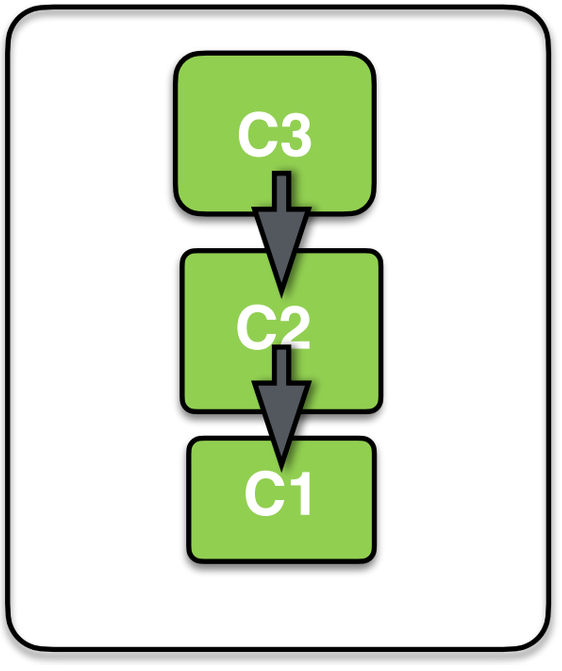
Repository





./WORKDIR

.git/



Git Three area

Workspace

Index

Repository



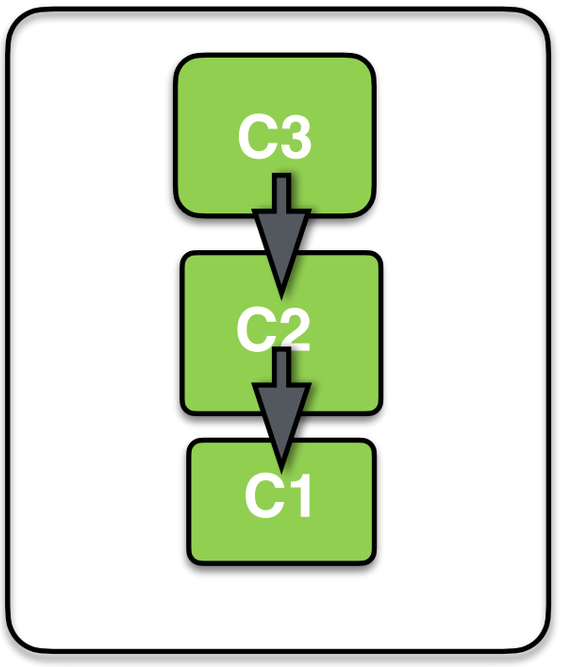




./WORKDIR

Staging area

.git/



Git Three area

Workspace

Index

Repository





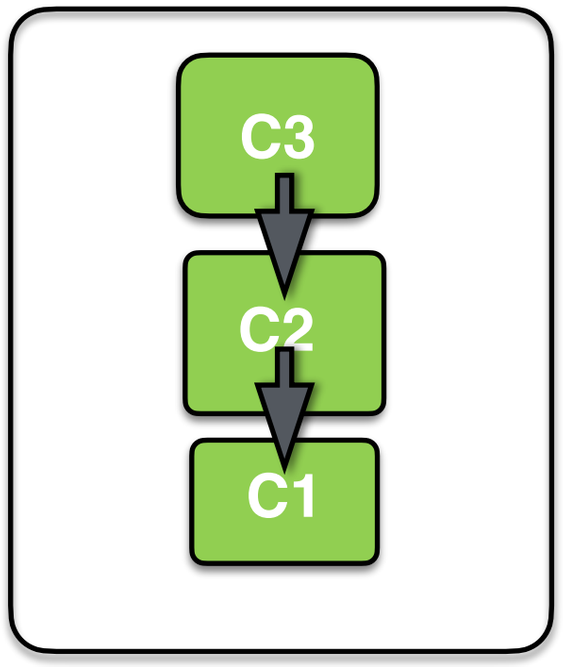


./WORKDIR

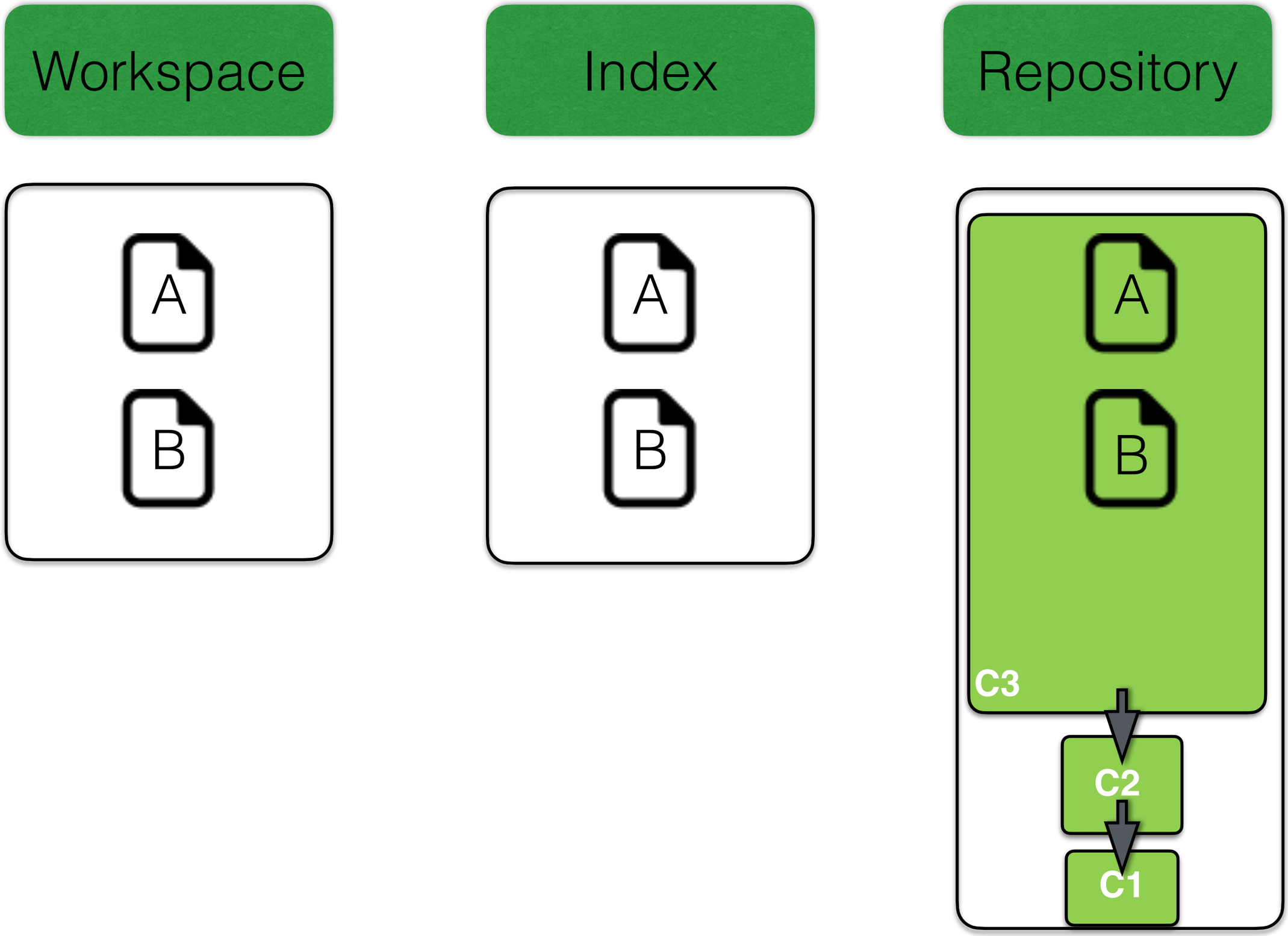
.git/index

Staging area

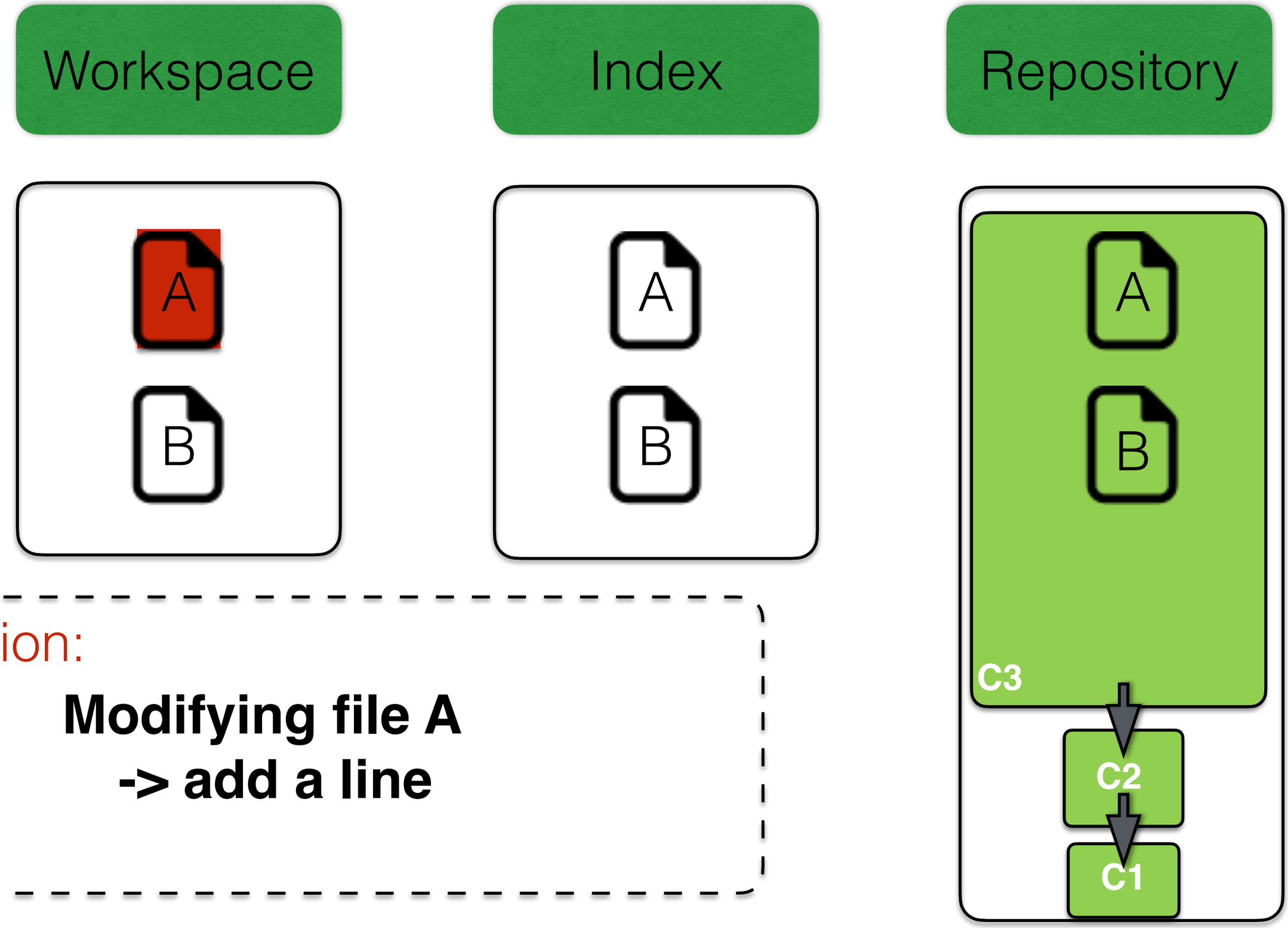
.git/



Git Three area



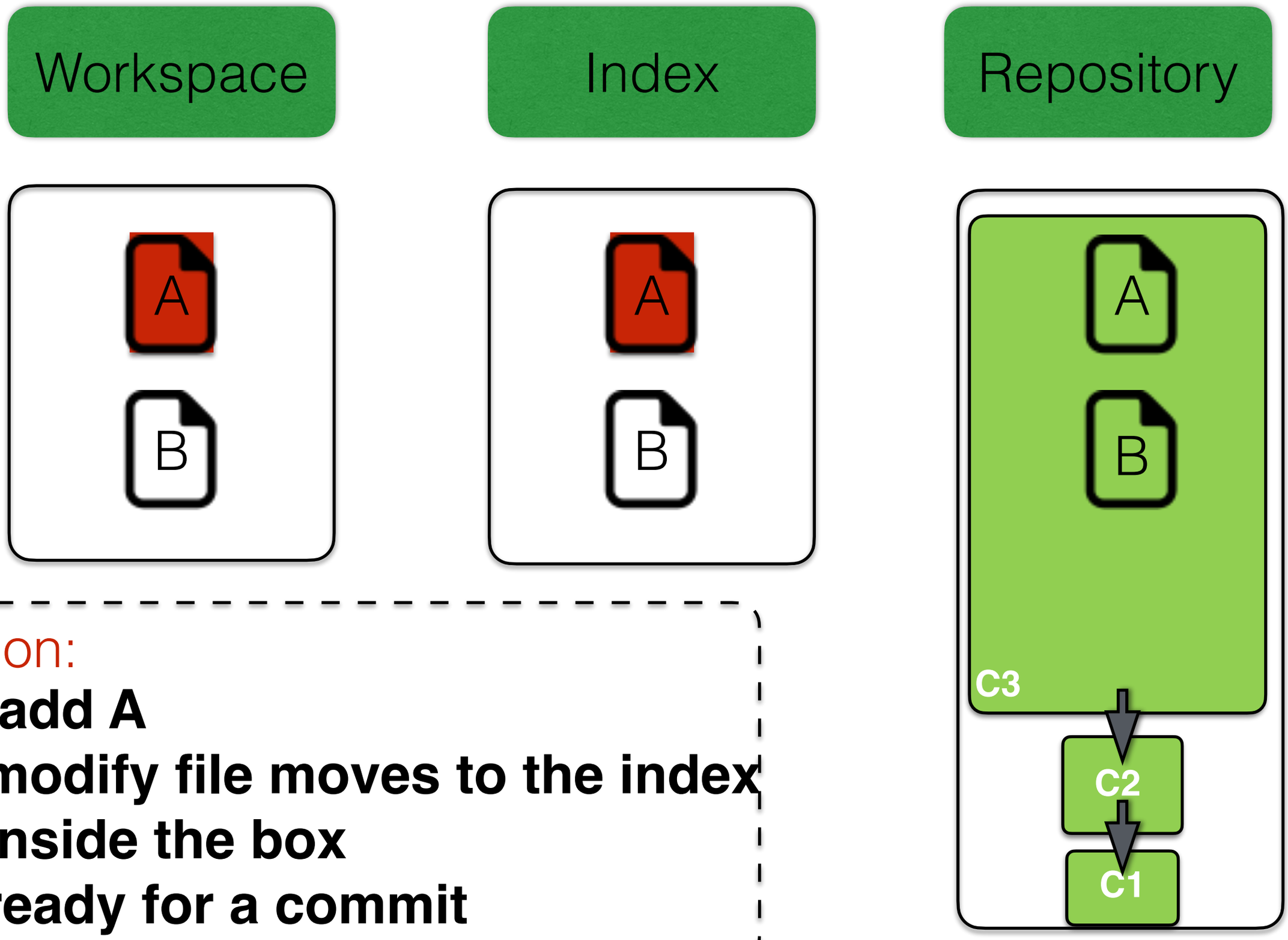
Git Three area



Action:

Modifying file A
-> add a line

Git Three area



Action:

git add A

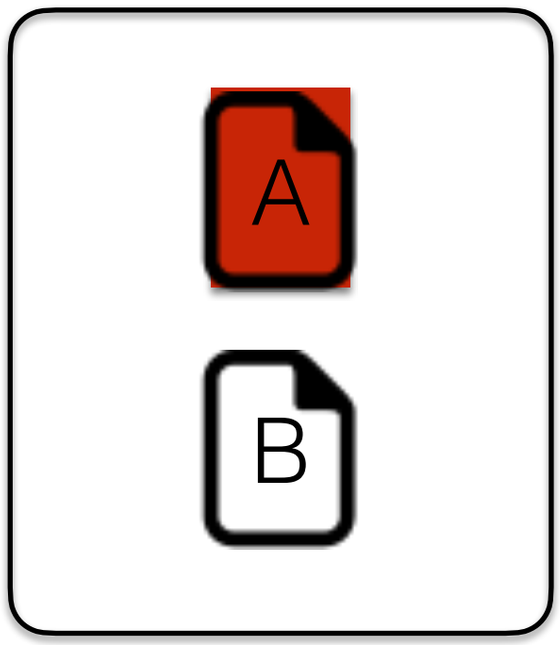
-> modify file moves to the index

-> inside the box

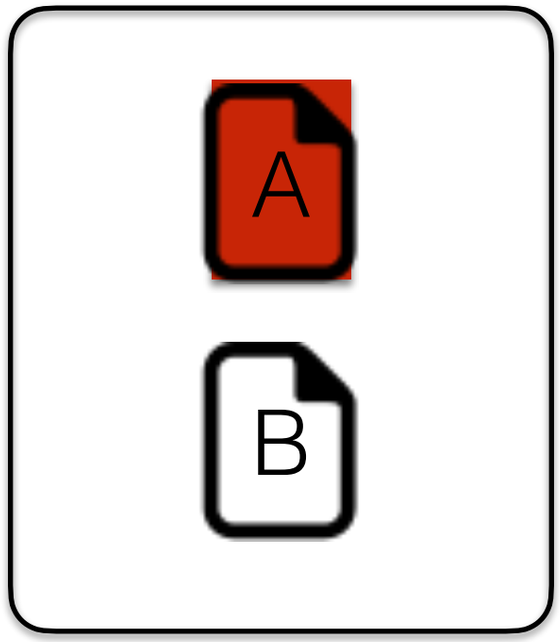
-> ready for a commit

Git Three area

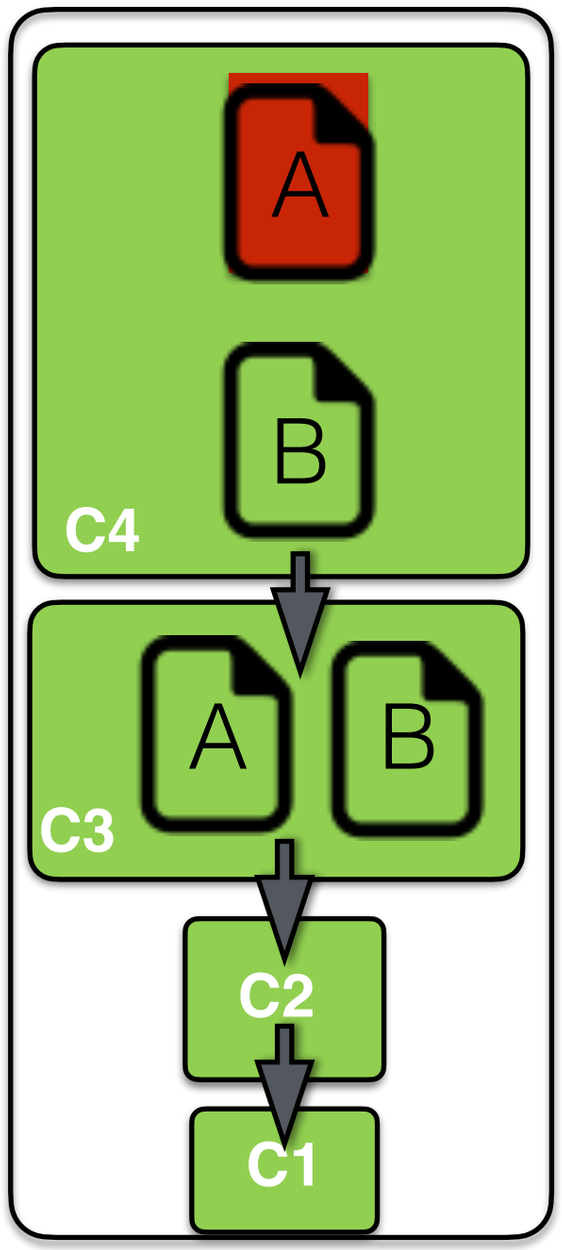
Workspace



Index



Repository



Action:

git commit -m "change color"

-> save the index current status

Into a new commit inside the Repository

1. Commit

Head: place where the new commit will be attach



Head: latest committed version

Head~1

Head~2

Useful trick

Git status does indicates which restore command to use for most common situation

```
[test_git]$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   hello

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello
```

Let's play

Game installed on lemaitre4 and manneback
module load gameshell/git

Basic command of the game
gsh goal
gsh check

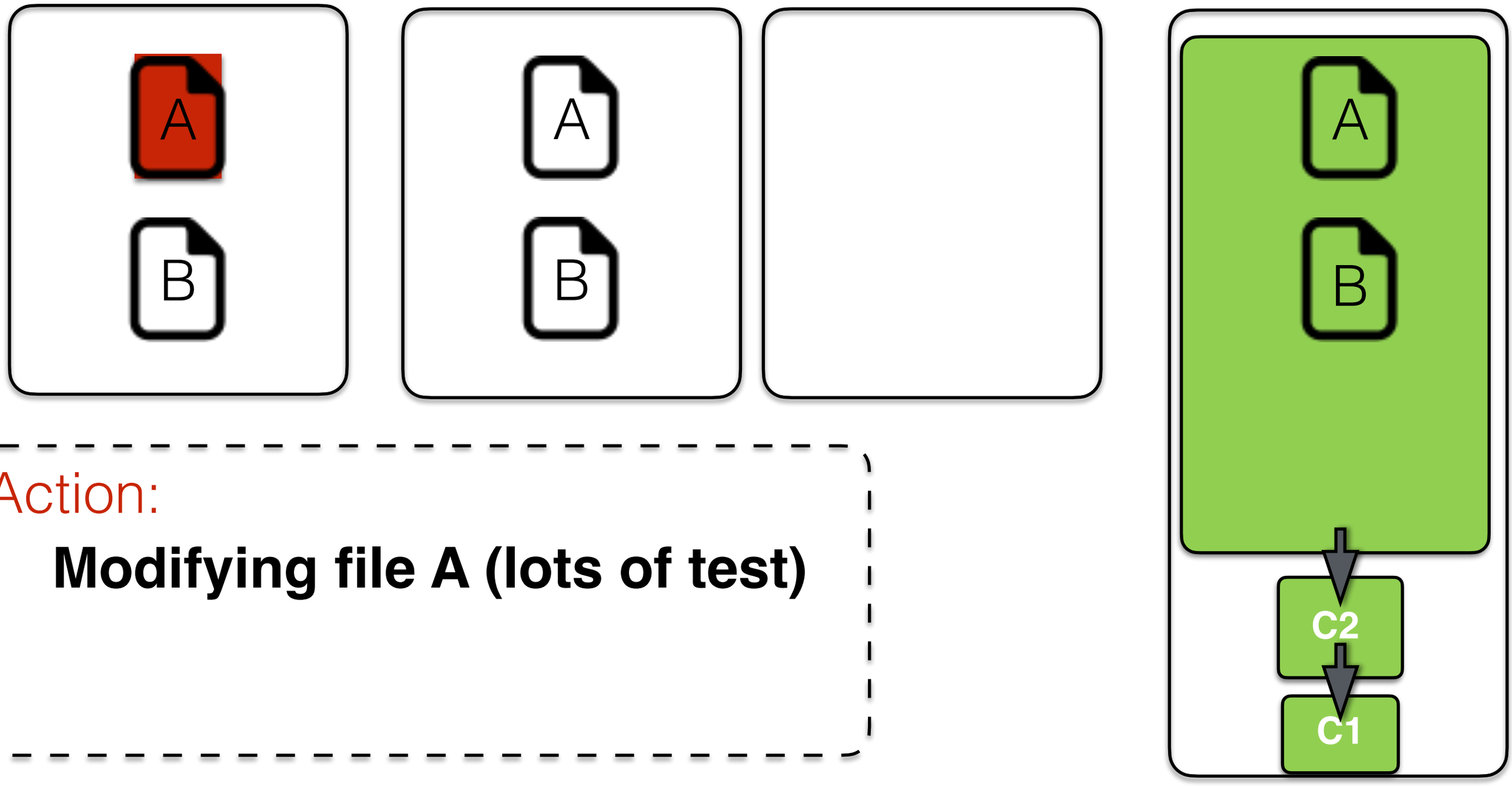
Issue with a level: restart it with
gsh reset

Exercise #1
Stop at mission #8
You have ~45 min

Workflow

My local workflow

Workspace | Index | Stash | Repository

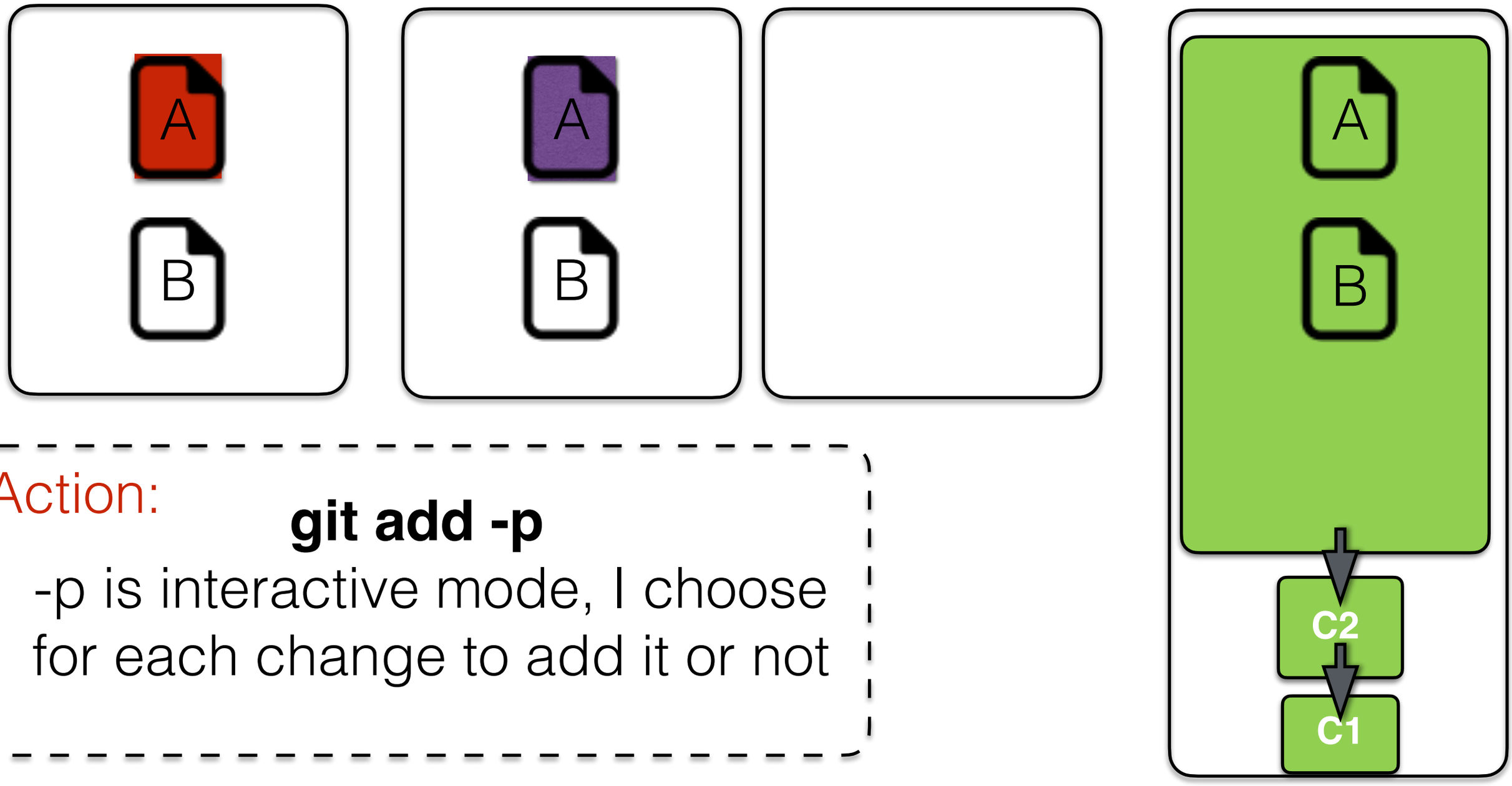


Action:

Modifying file A (lots of test)

My local workflow

Workspace | Index | Stash | Repository



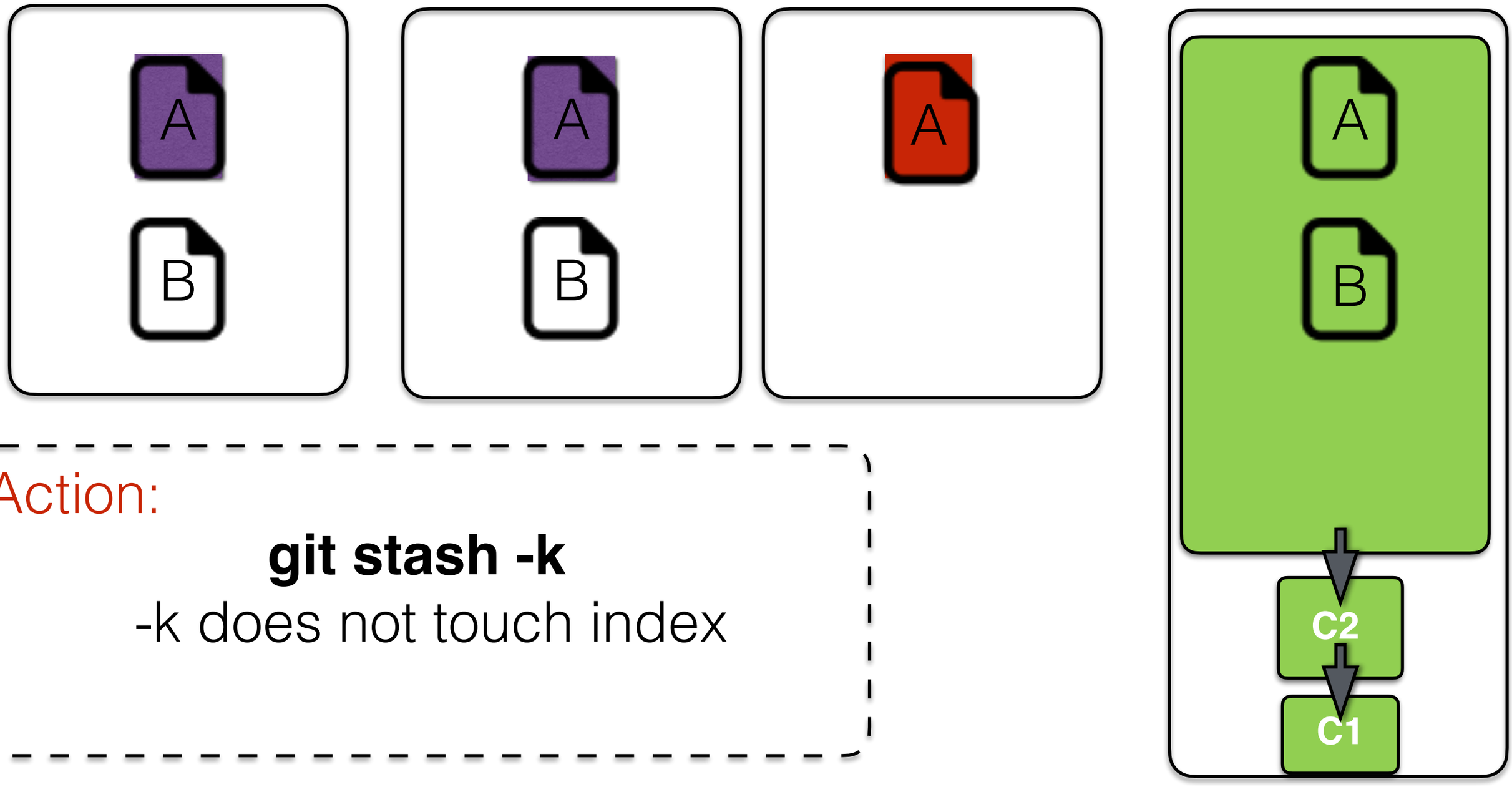
Action:

git add -p

-p is interactive mode, I choose for each change to add it or not

My local workflow

Workspace | Index | Stash | Repository



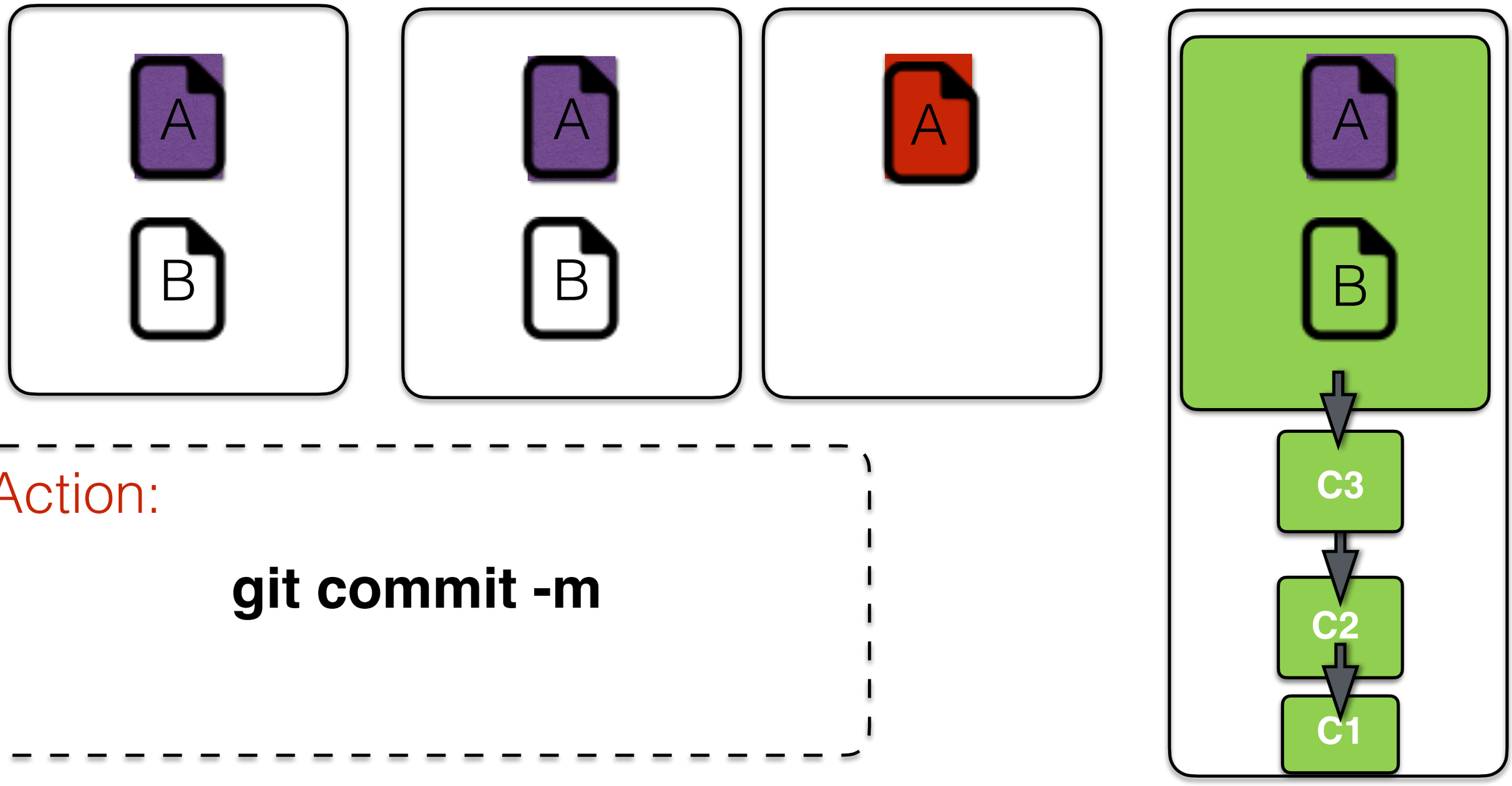
Action:

git stash -k

-k does not touch index

My local workflow

Workspace | Index | Stash | Repository



Action:

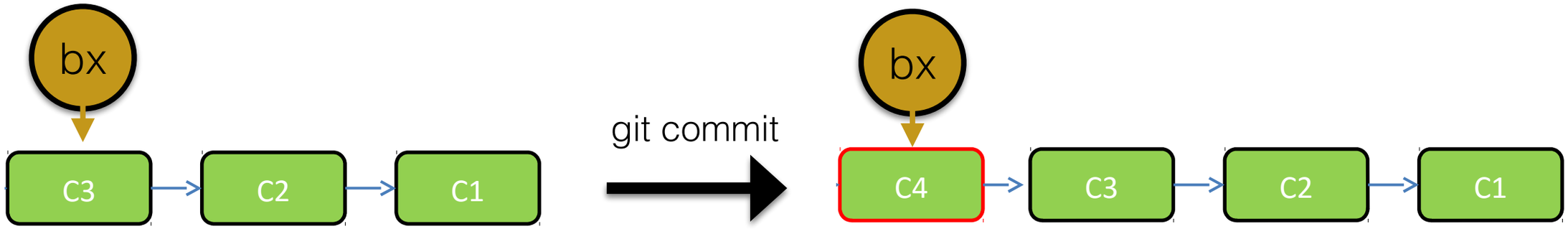
git commit -m

branch in git

- Branch is **pointer** to a commit (represent an history) which represent a **line of development**
- A branch can point at other commit, it **can move**!
- A branch is a way to organise your work and working histories

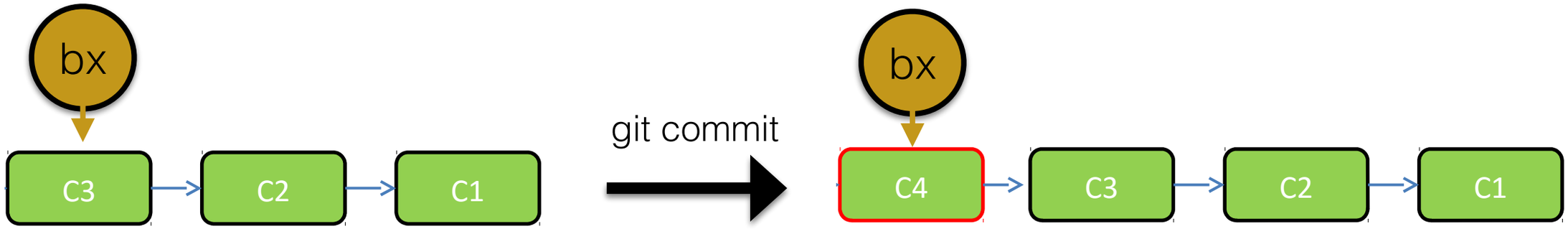
branches

- default branch: master (or main)
- When doing a commit, the branch moves to the new commit



branches

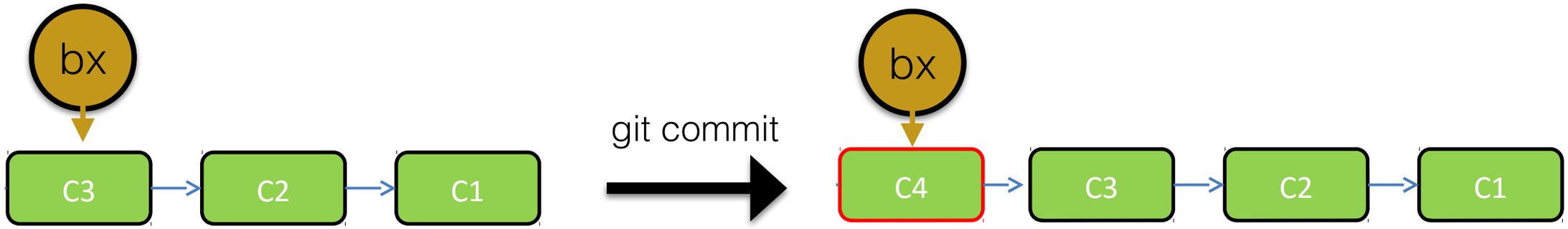
- default branch: master (or main)
- When doing a commit, the branch moves to the new commit



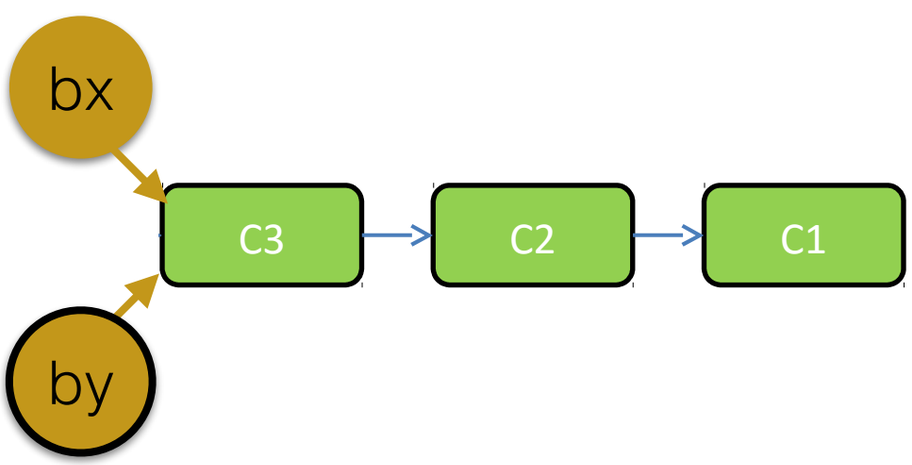
- creating a new branch: add a pointer (git switch -c by)

branches

- default branch: master (or main)
- When doing a commit, the branch moves to the new commit

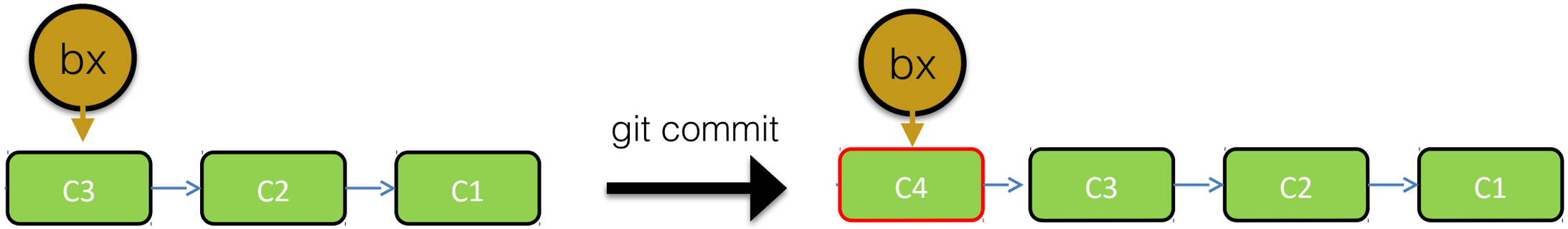


- creating a new branch: add a pointer (git switch -c by)

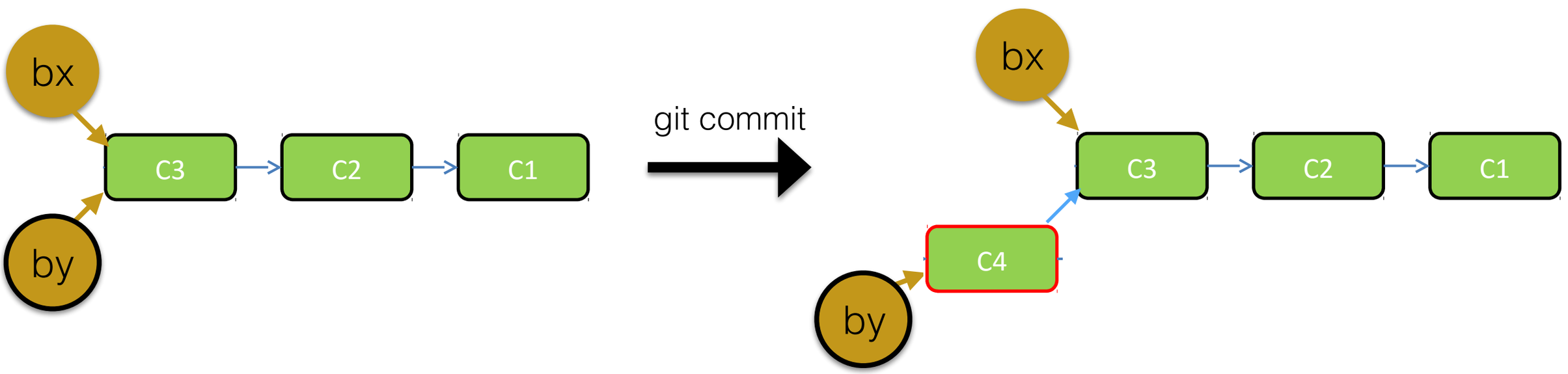


branches

- default branch: master (or main)
- When doing a commit, the branch moves to the new commit

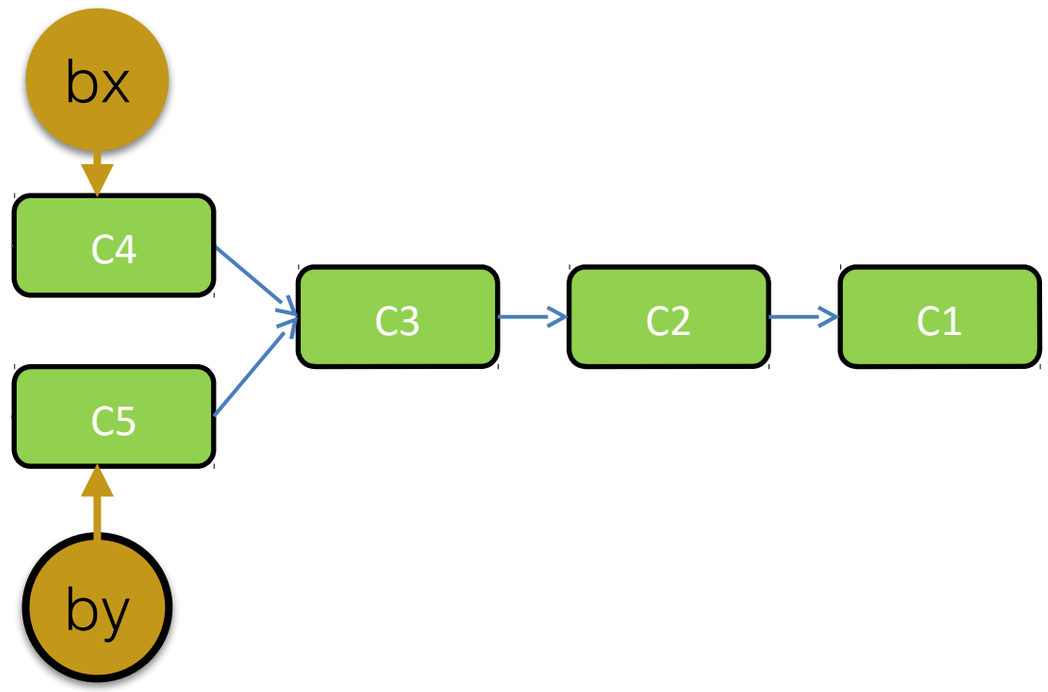


- creating a new branch: add a pointer (git switch -c by)
  - only selected branch affected by commit!



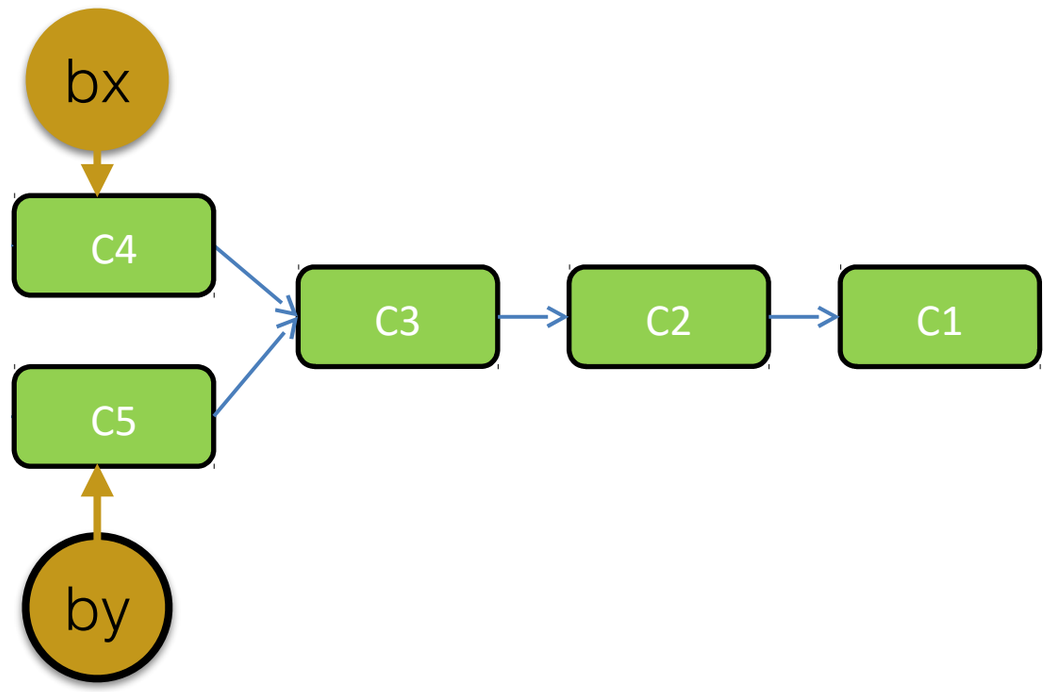
merging

- The interest of branch is that you can **merge** them
  - Include in one (branch) file the modification done somewhere else



merging

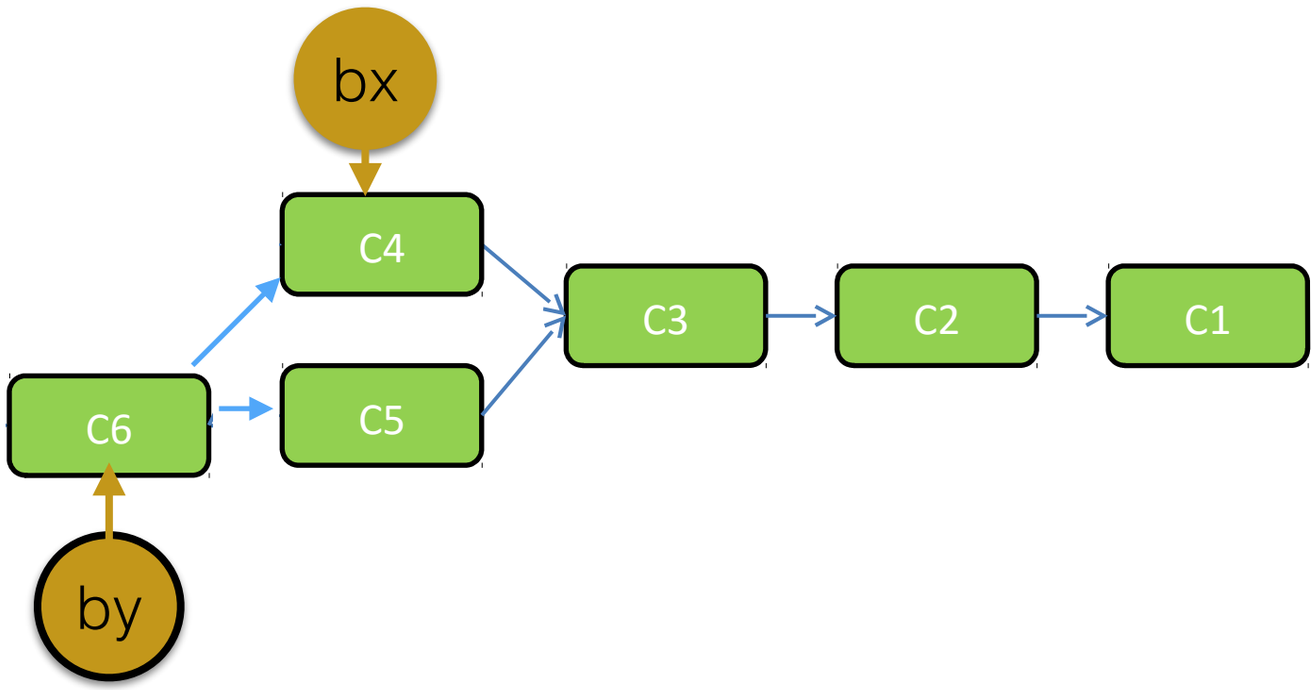
- The interest of branch is that you can **merge** them
  - Include in one (branch) file the modification done somewhere else



git merge bx

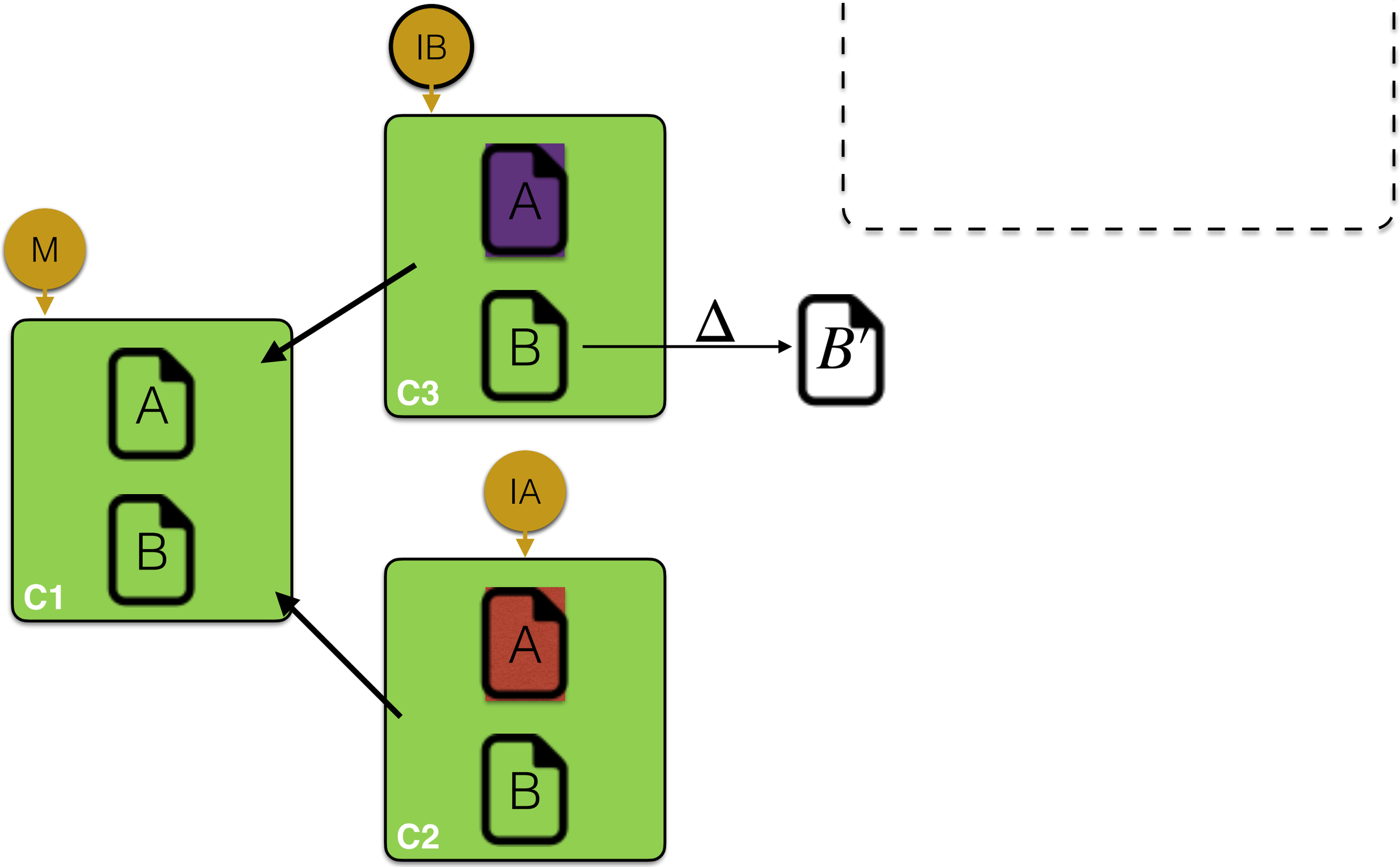
merging

- merging two different modifications



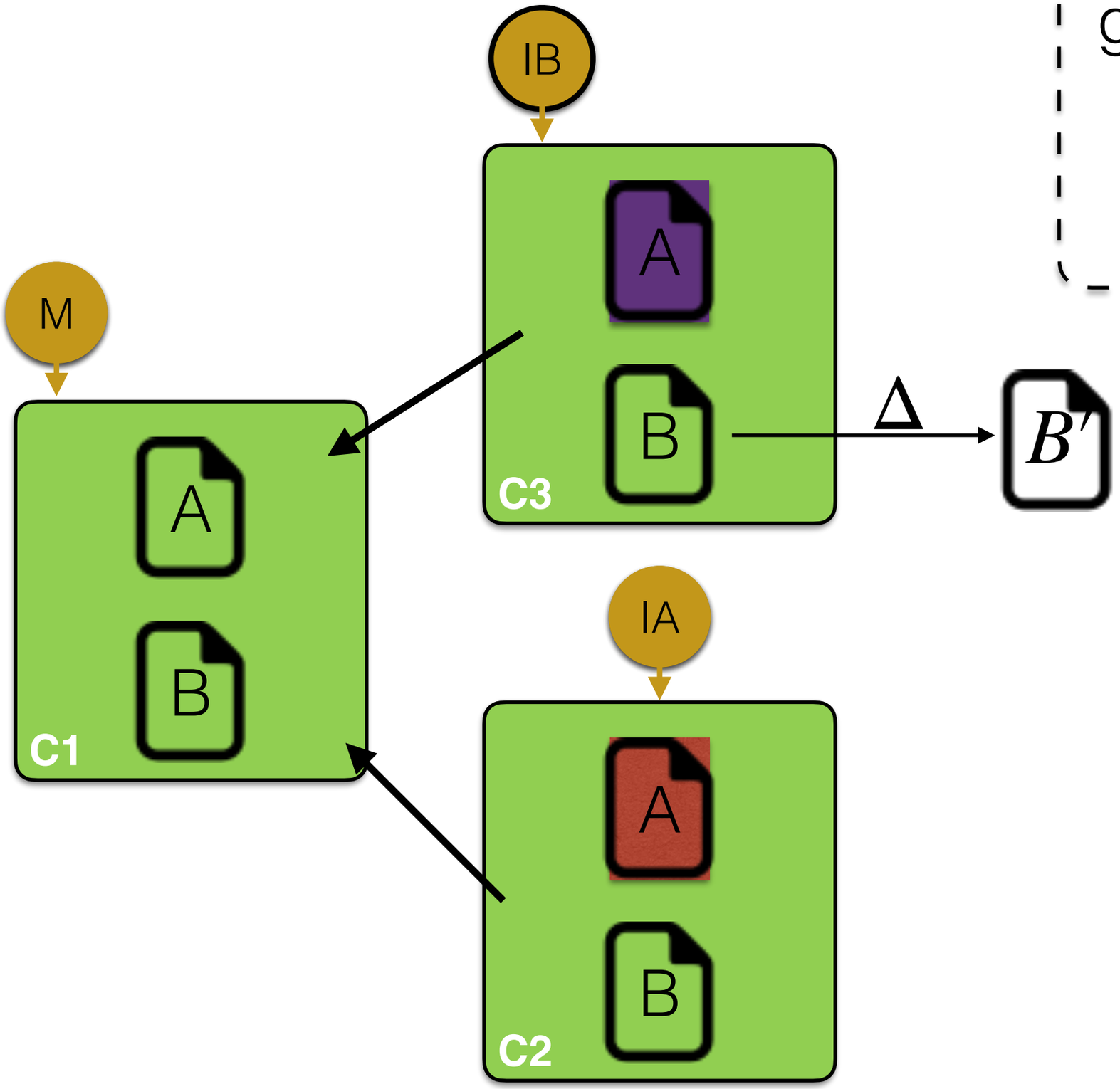
git merge bx

Working on the wrong branch

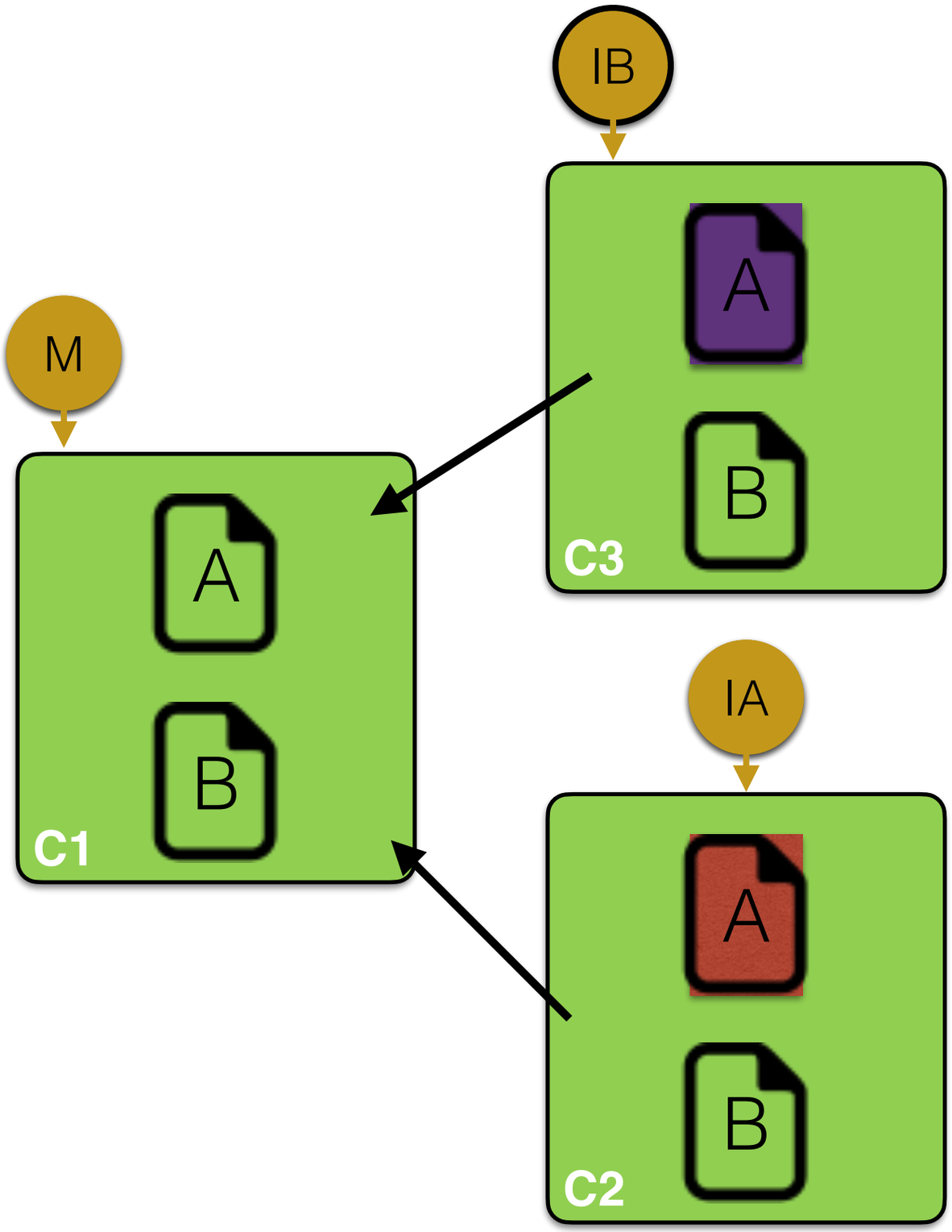


Working on the wrong branch

git stash save “INFO”



Working on the wrong branch



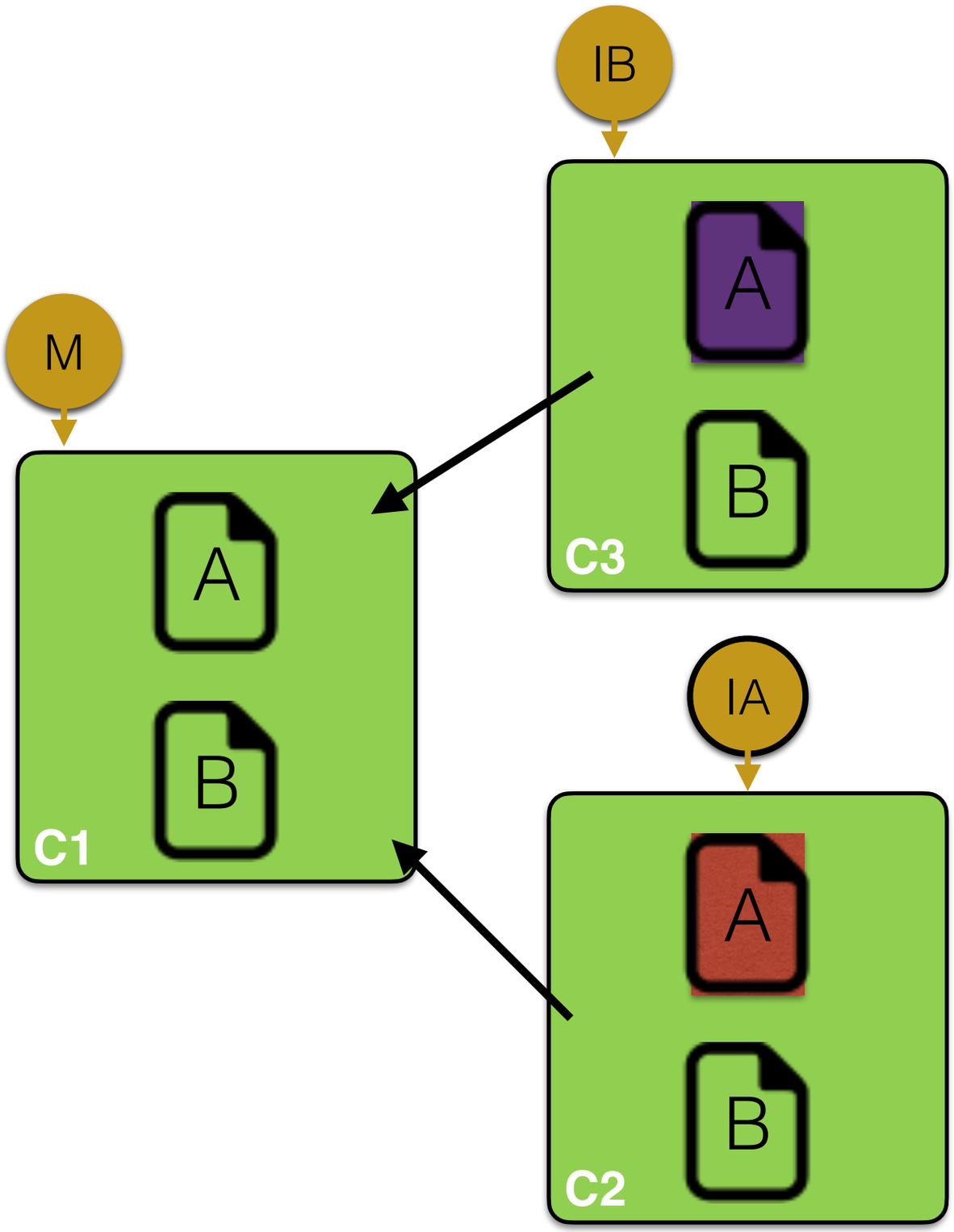
git stash save “INFO”

Stash storage

git stash list

stash@{0} Δ

Working on the wrong branch



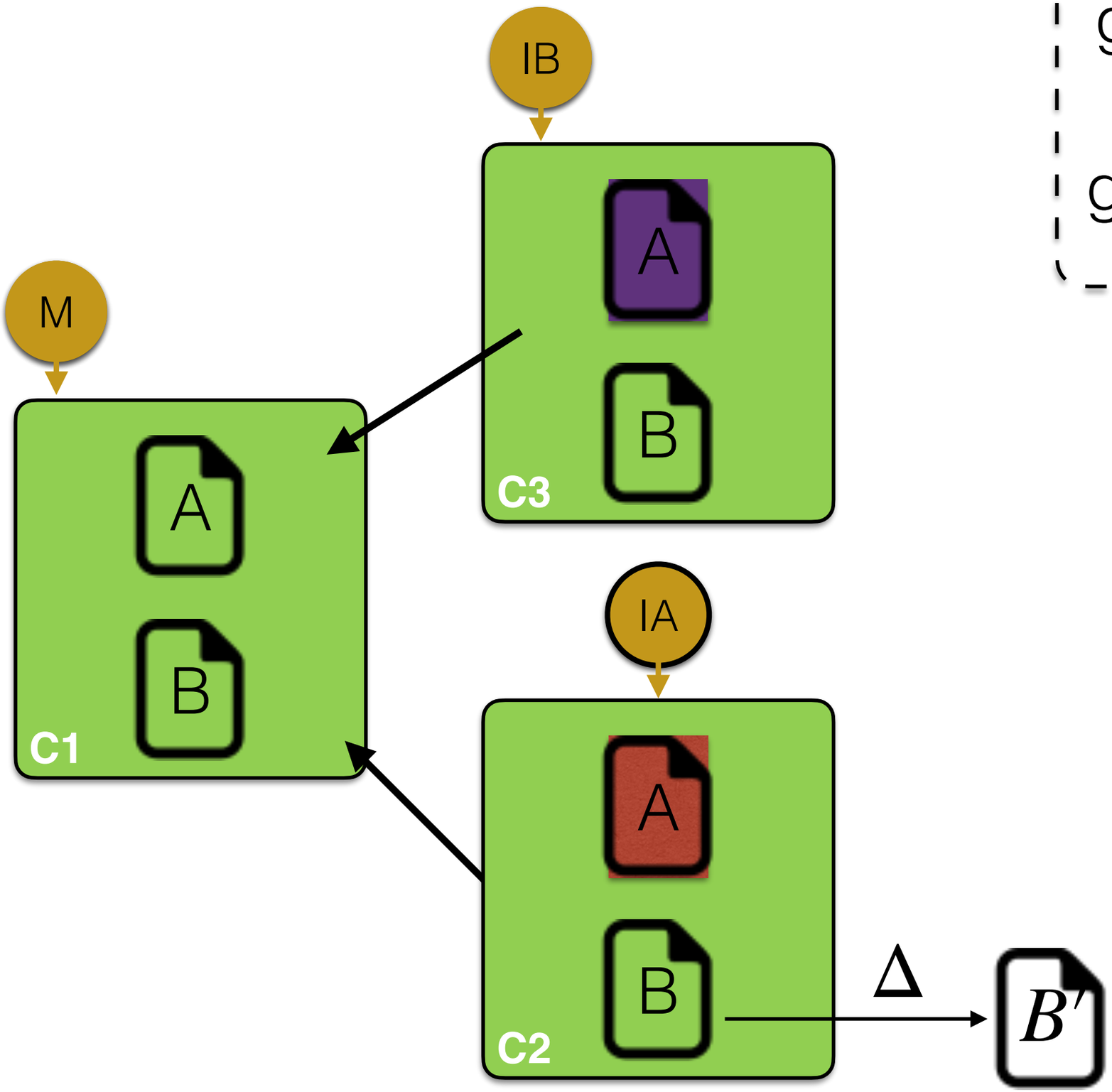
git stash save “INFO”

git switch IA

Git stash list

stash@{0} Δ

Working on the wrong branch




```
git stash save “INFO”
  git switch IA
git stash pop
```

Git stash list



Do it yourself

- Mission #8-14

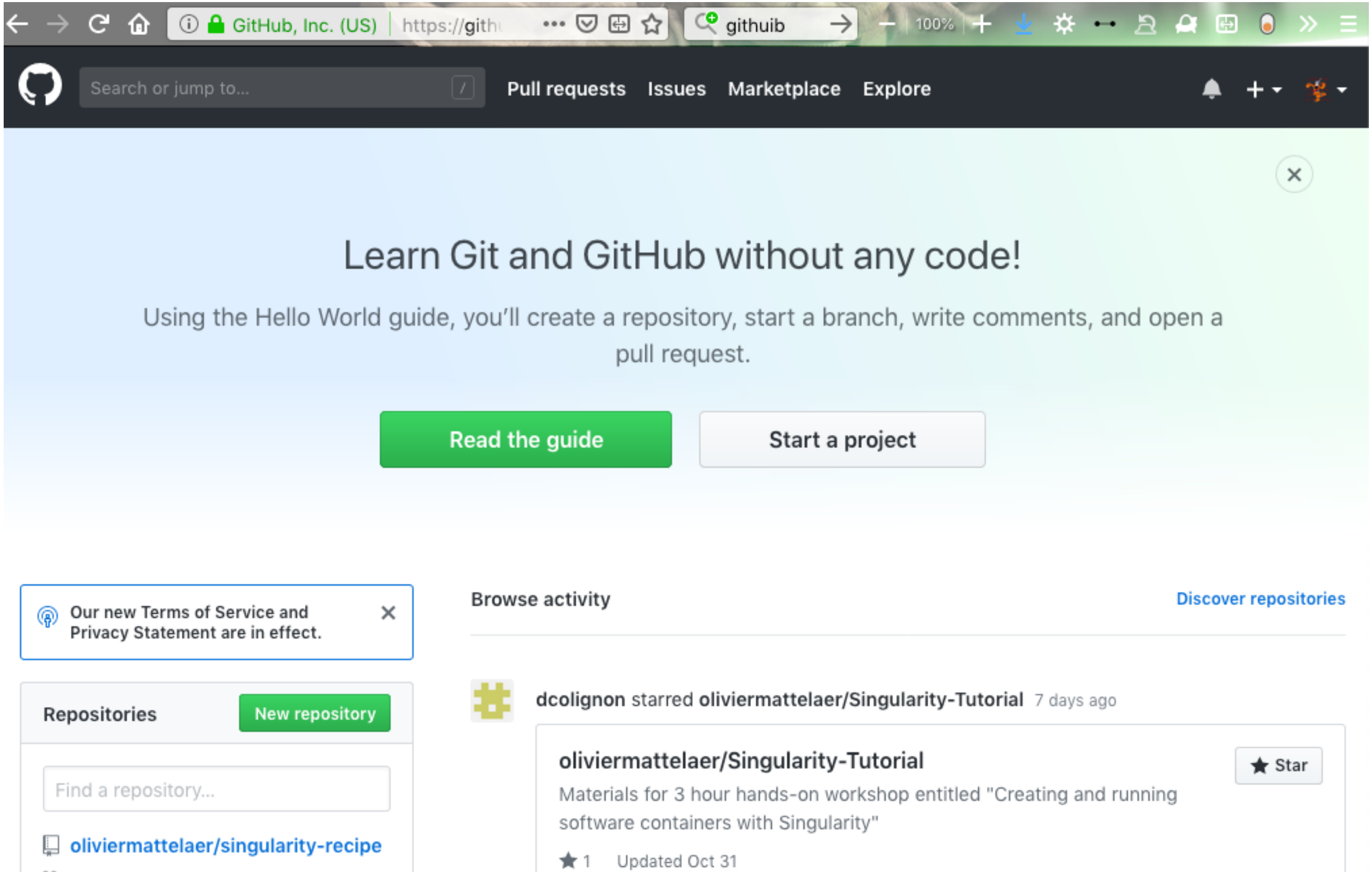
Conflict

- **Multiple version of files** are great
  - Not always easy to know how to merge them
  - Conflict will happen (same line modify by both user)

- Conflict need to be resolved manually!
  - Boring task
  - need to understand why a conflict is present!
- **Do not be afraid of conflict!** Do not try to avoid them at all cost!
- stay in sync as most as possible and keep line short

Team Work

GitHub/Gitlab



Remote Branches

This is a remote branch

My Machine

The Server

origin/master

master

C1

C0

clone

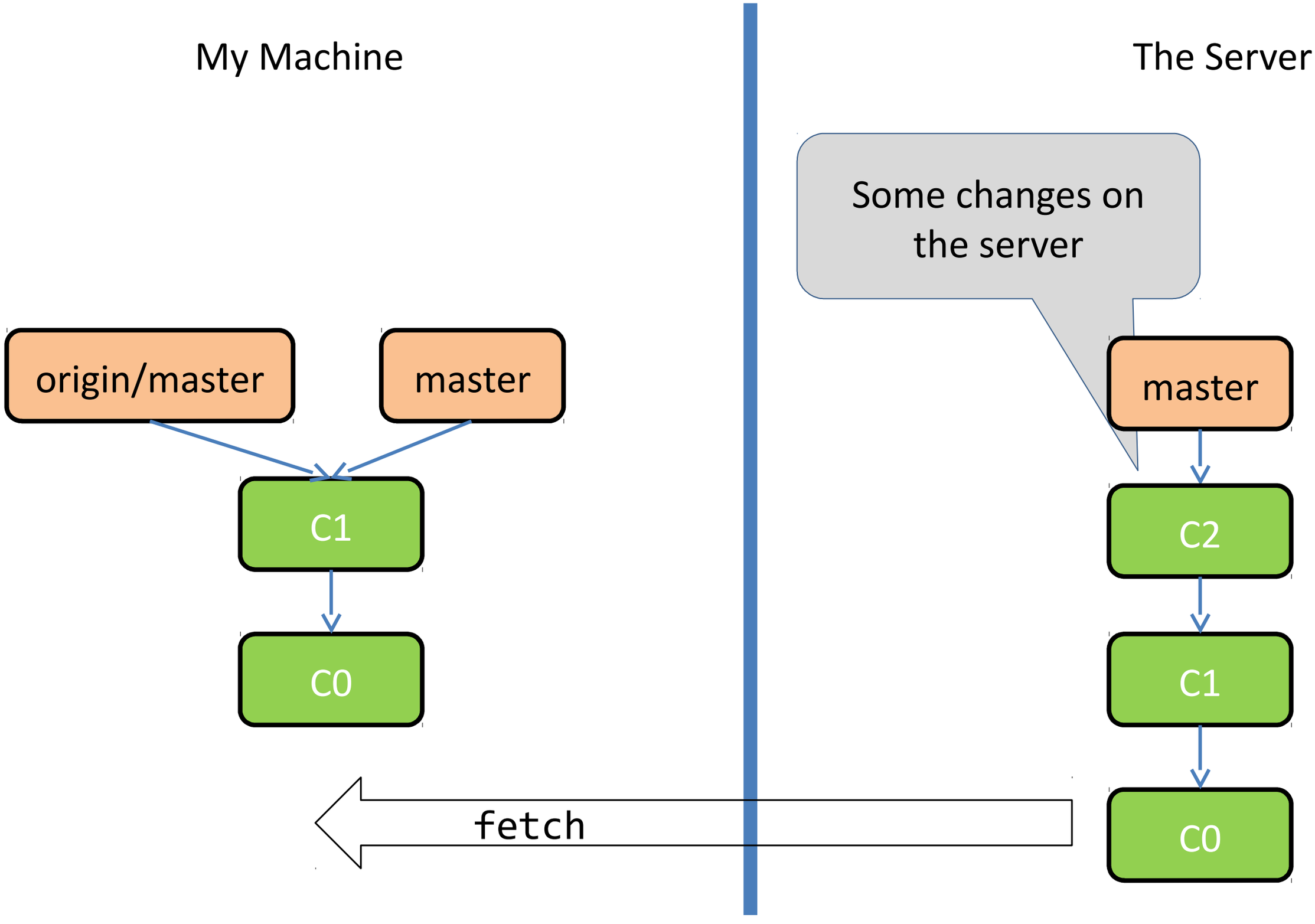
master

C1

C0

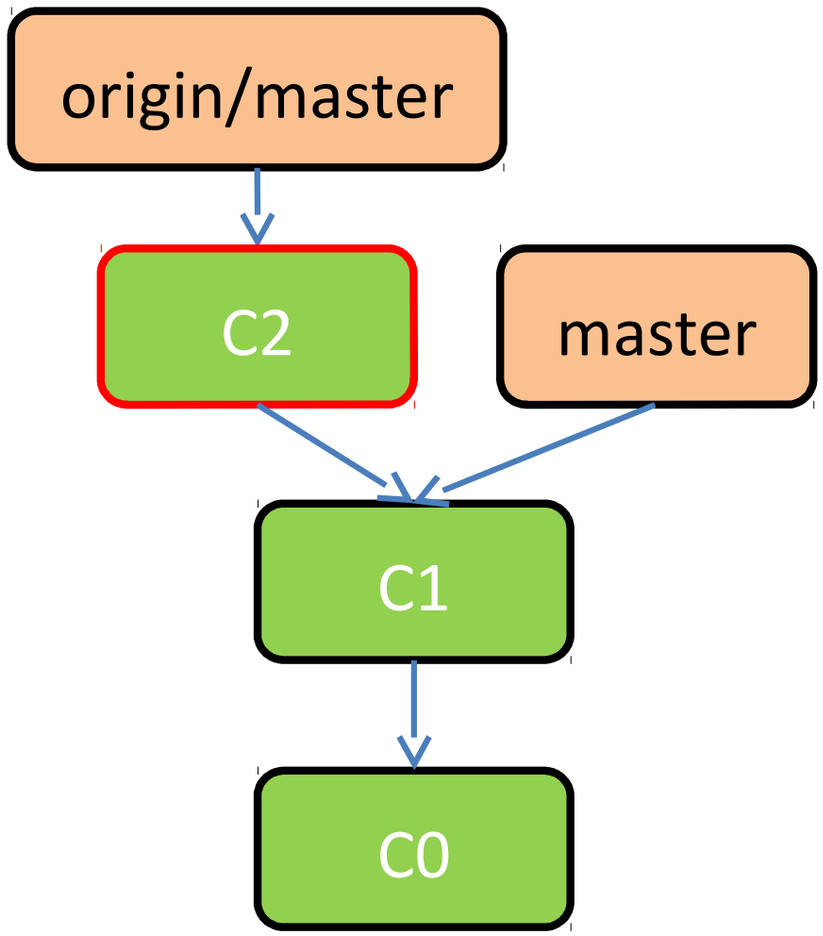
origin is just a name for a “remote”

Remote Branches - fetch

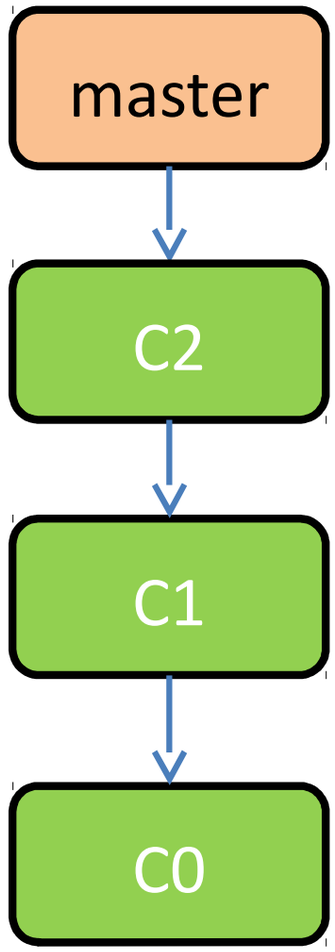


Remote Branches - fetch

My Machine

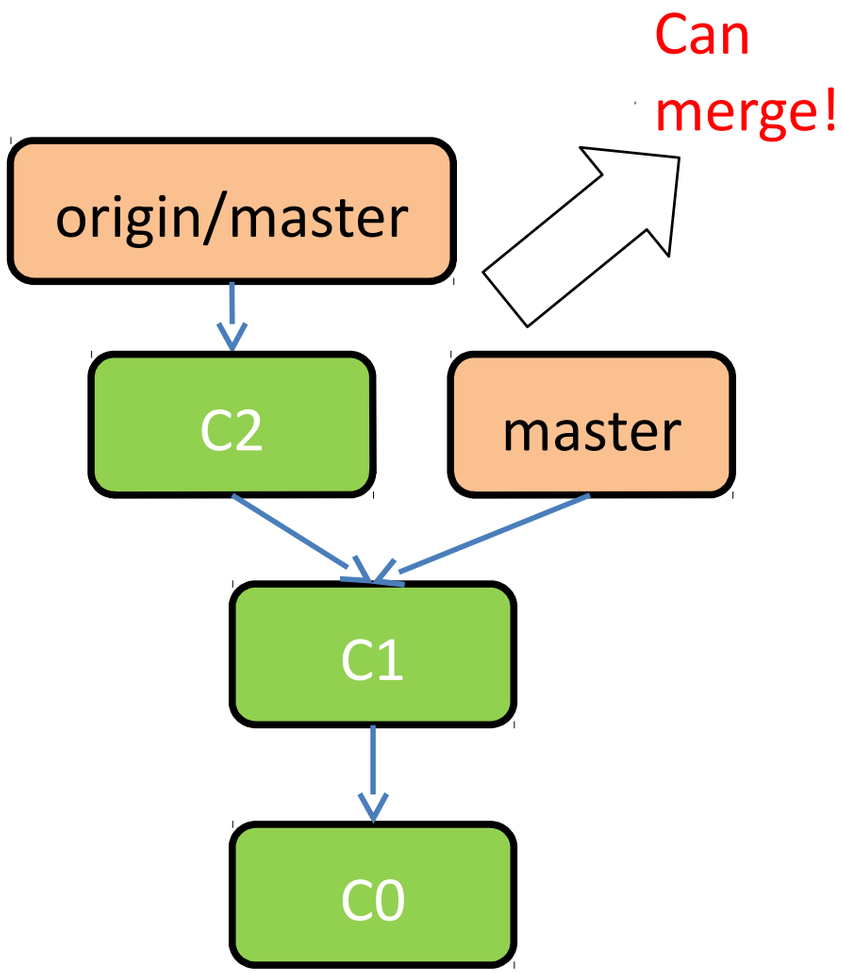


The Server



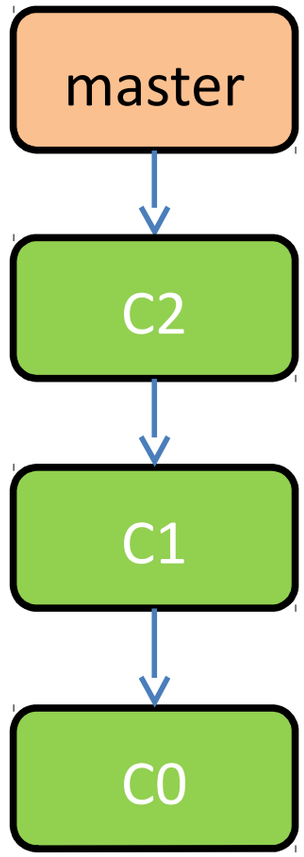
Remote Branches - fetch

My Machine



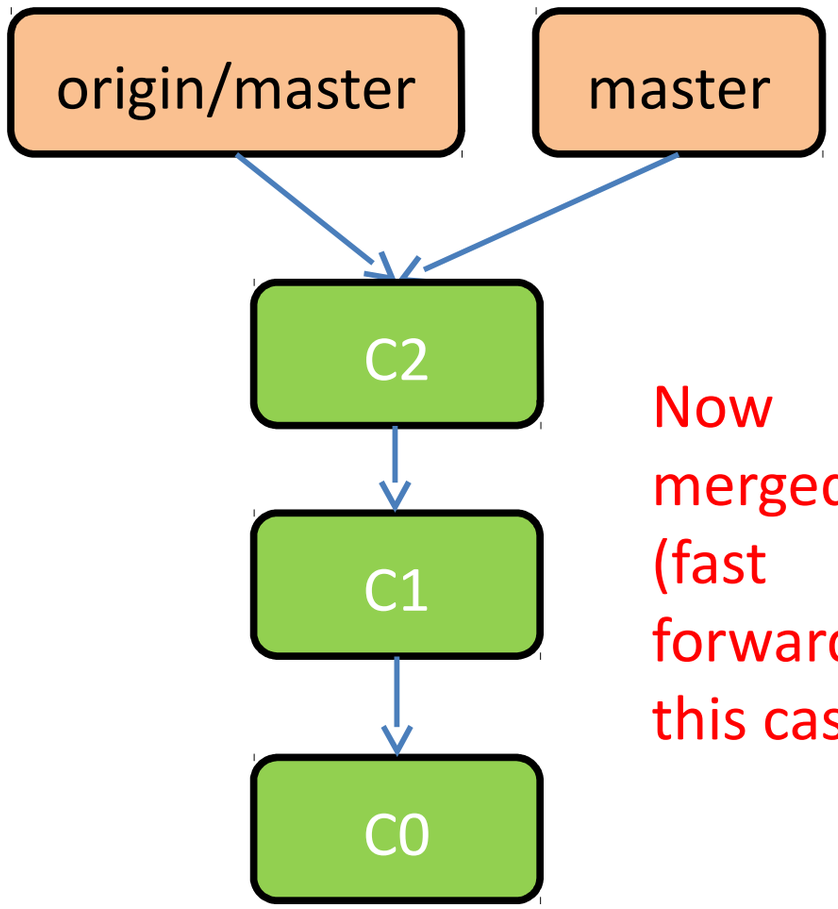
git merge

The Server



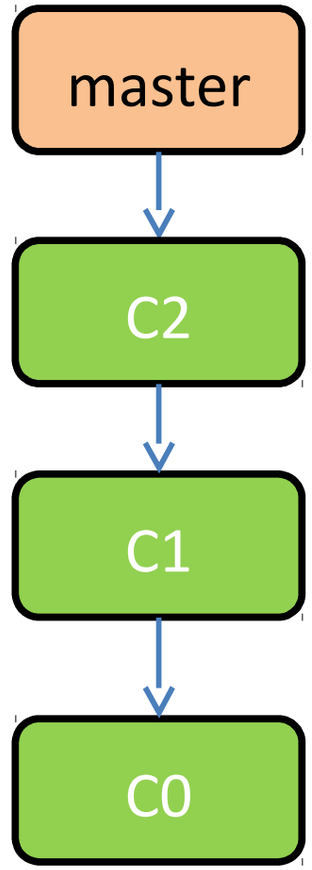
Remote Branches - fetch

My Machine



Now merged (fast forward in this case)

The Server

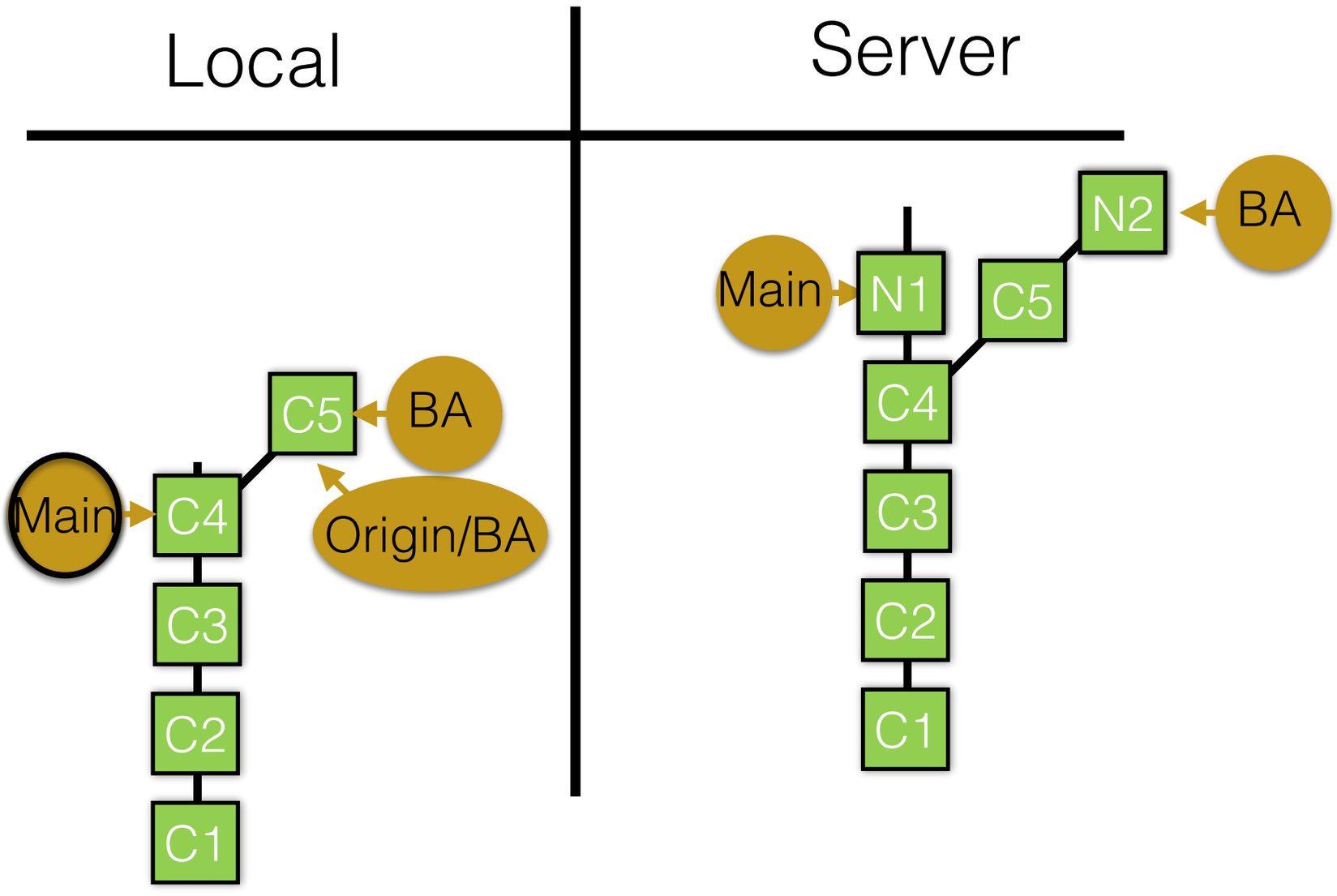


Remote Branches

- 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”

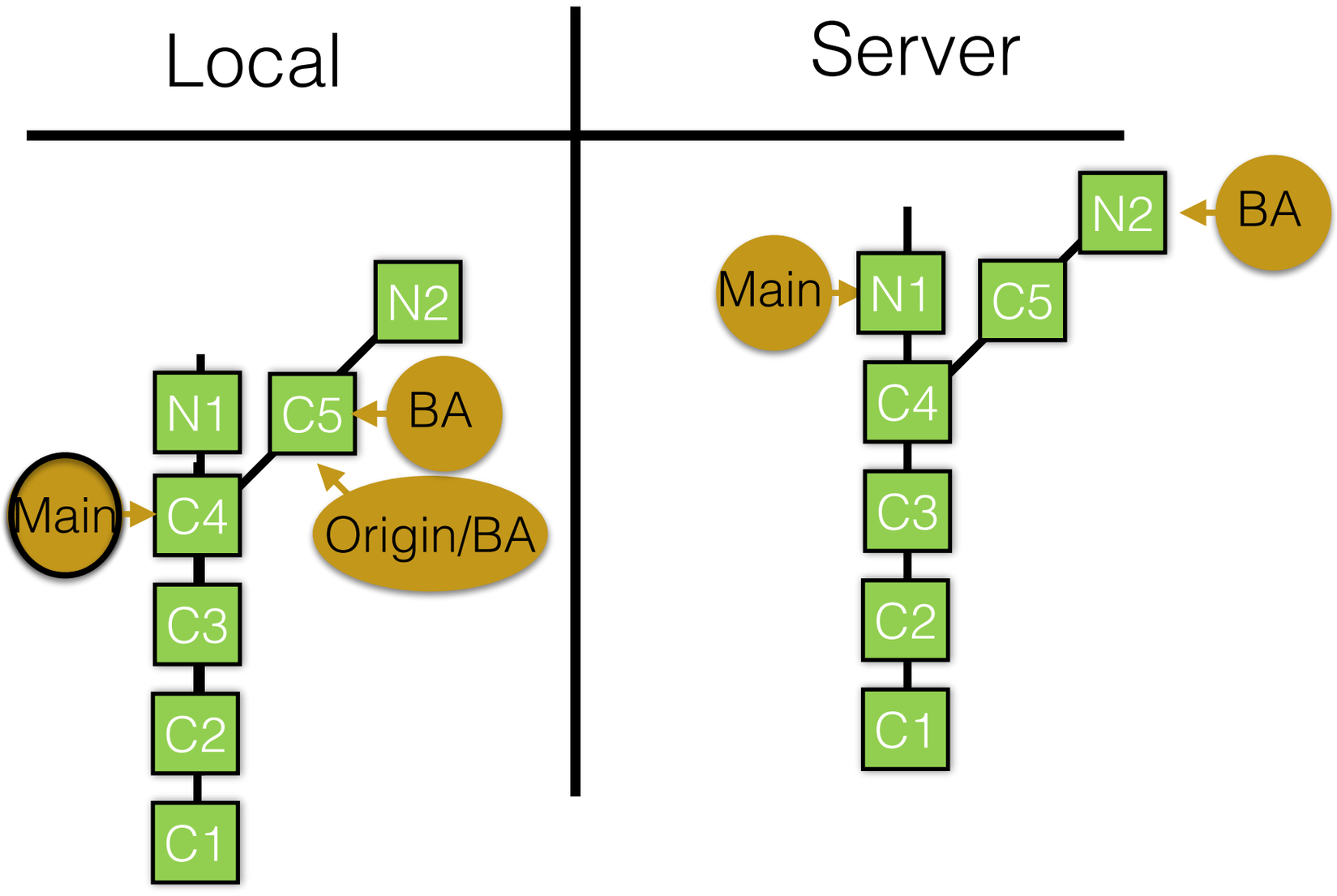
Be careful !!

Git pull



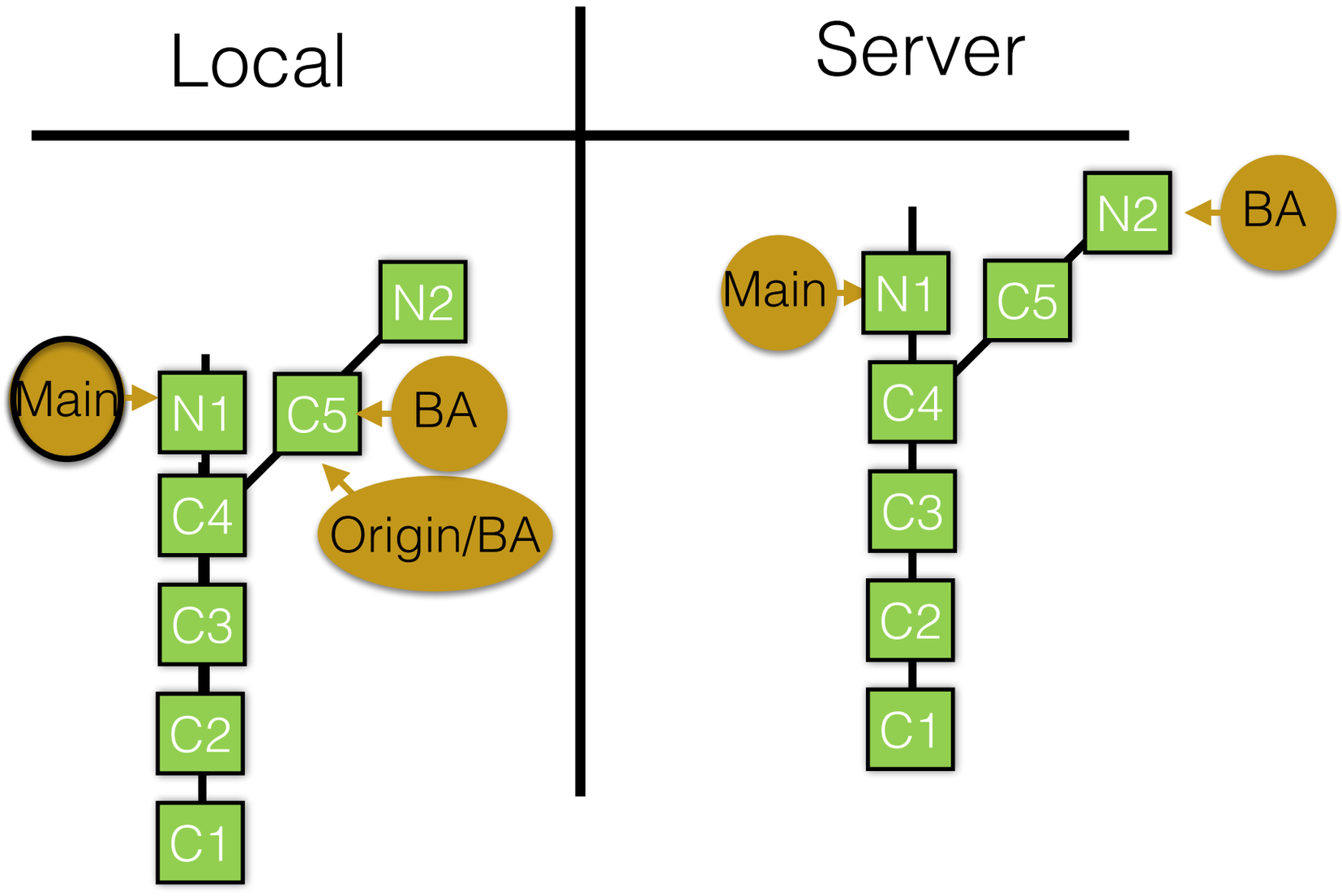
Be careful !!

Git pull



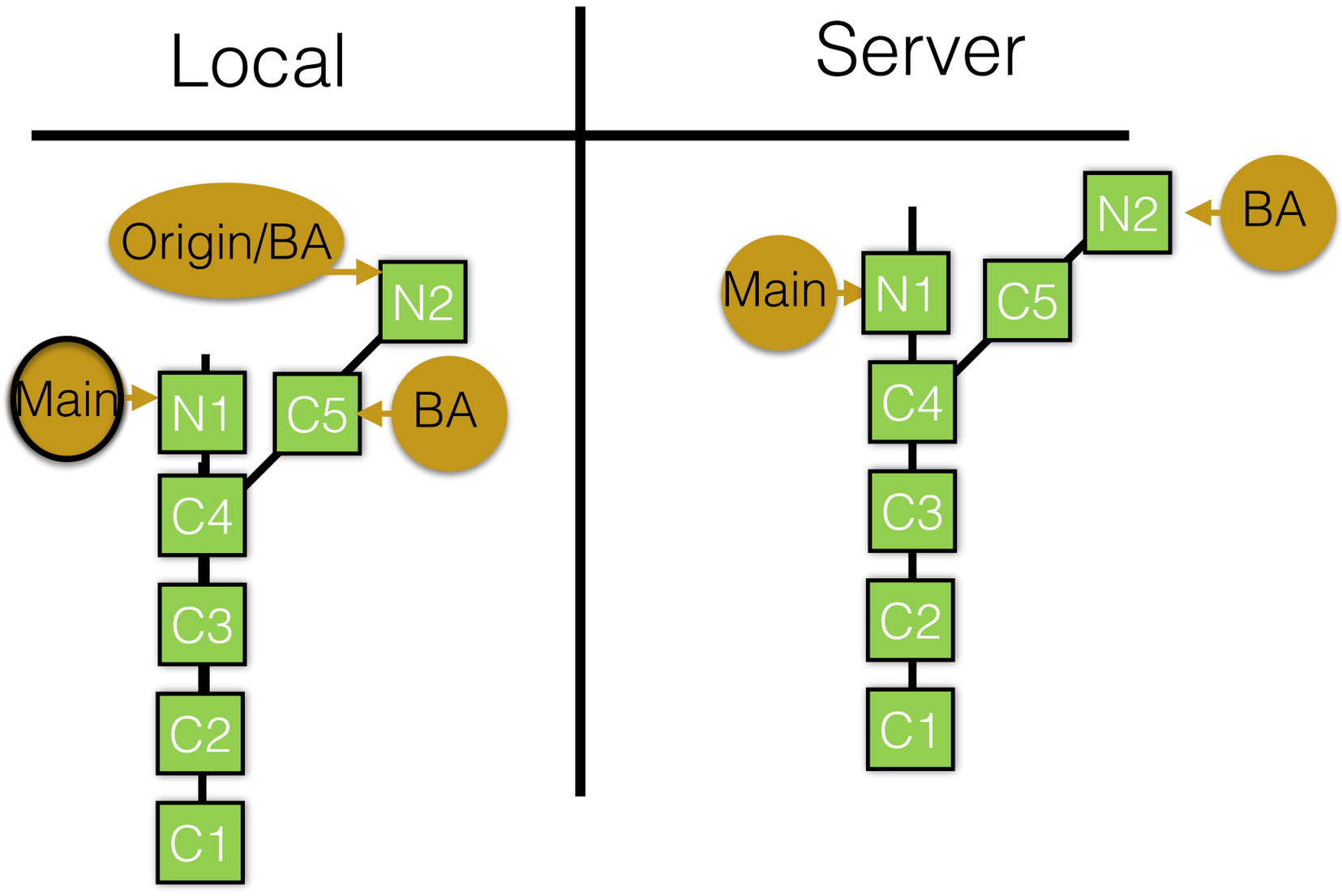
Be careful !!

Git pull



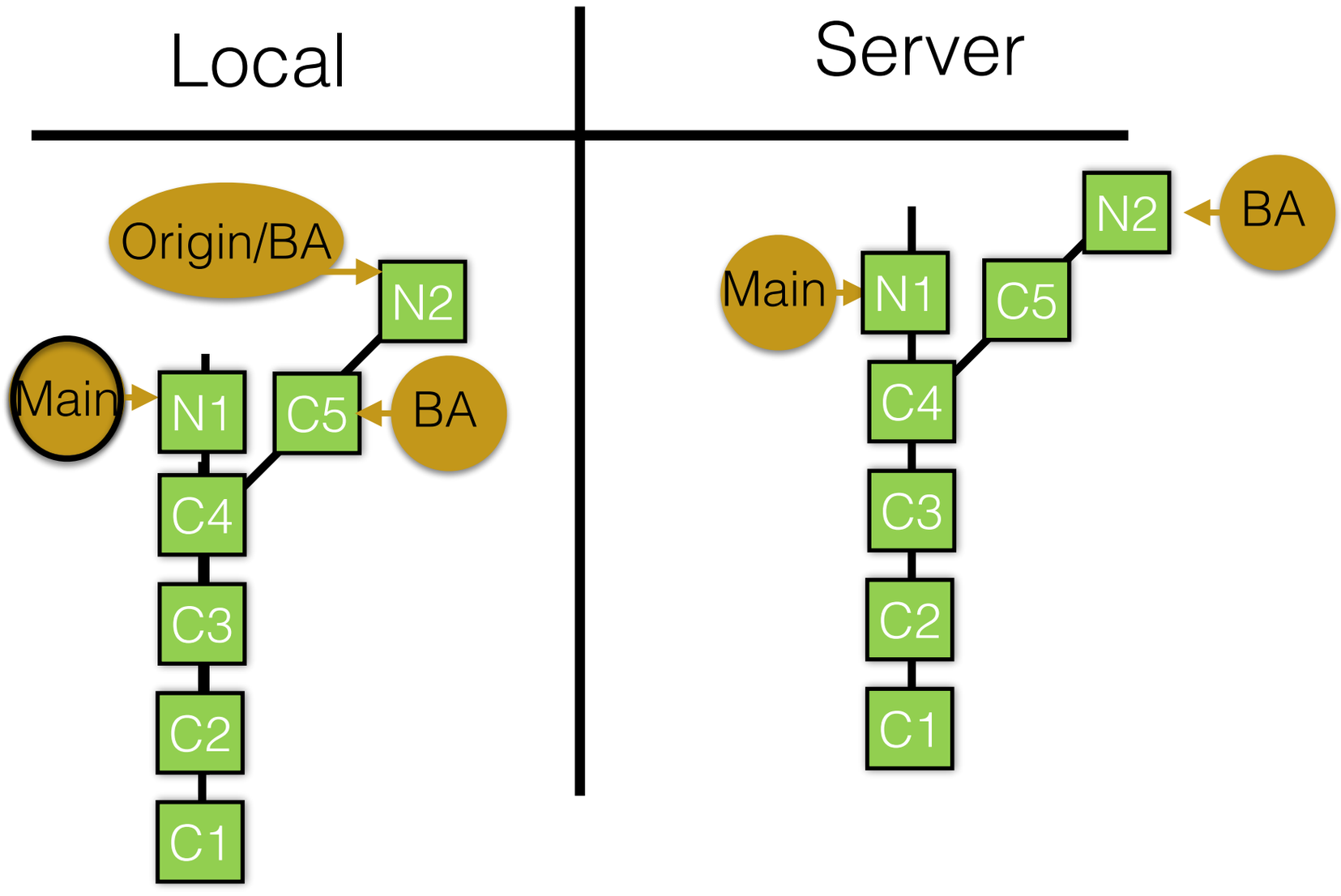
Be careful !!

Git pull



Be careful !!

Git pull

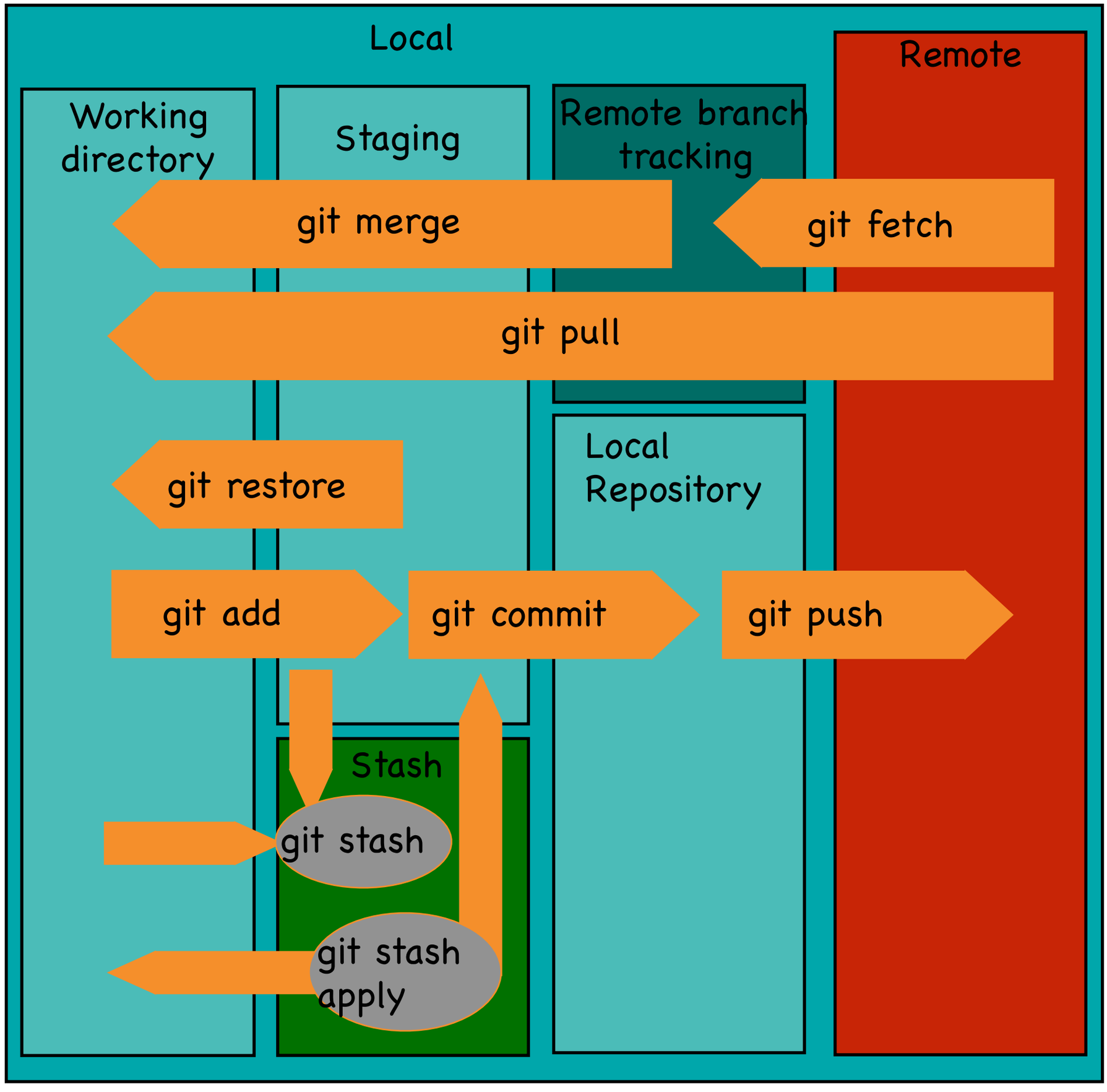


Do not do “merge BA” but merge original/BA



Do it yourself

- Mission #13-17


Local
Remote
Working directory
Staging
Remote branch tracking
git merge
git fetch
git pull
Local Repository
git restore
git add
git commit
git push
Stash
git stash
git stash apply

Information

- On cluster, use "module load git"
  - The command you learned need version 2.22
- Be careful with "restore" (no safety net)
- Restrict yourself to edit the history of commit (especially when shared)
  - Git reset # especially with --hard
  - Git rebase (only useful case is git pull --rebase)
- Merging with work of others:
  - Git merge **origin**/branch_name

Conclusion

- Versioning is crucial both for small/large project
  - Avoid dropbox for paper / project
- make meaningful commit
  - logical block
  - meaningful message

More information

- Why an index: http://gitolite.com/uses-of-index.html
- technical tutorial on git (details on storage structure): https://www.youtube.com/watch?v=xbLVvrb2-fY

- https://git-scm.com/doc