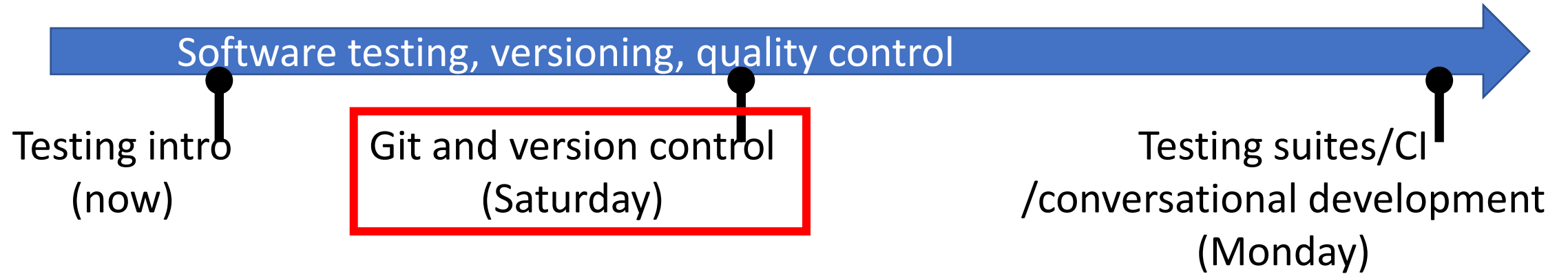


# Introduction to version control & Git

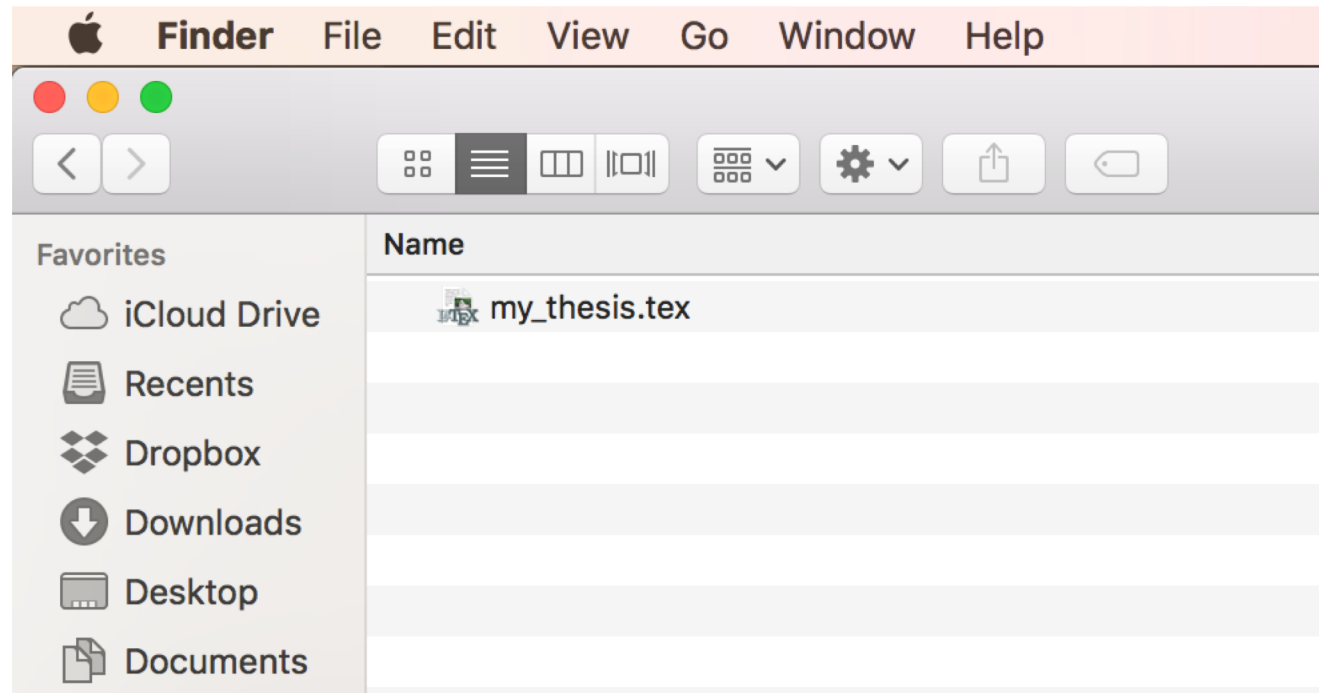
Alessandro Corbetta

# My Lectures



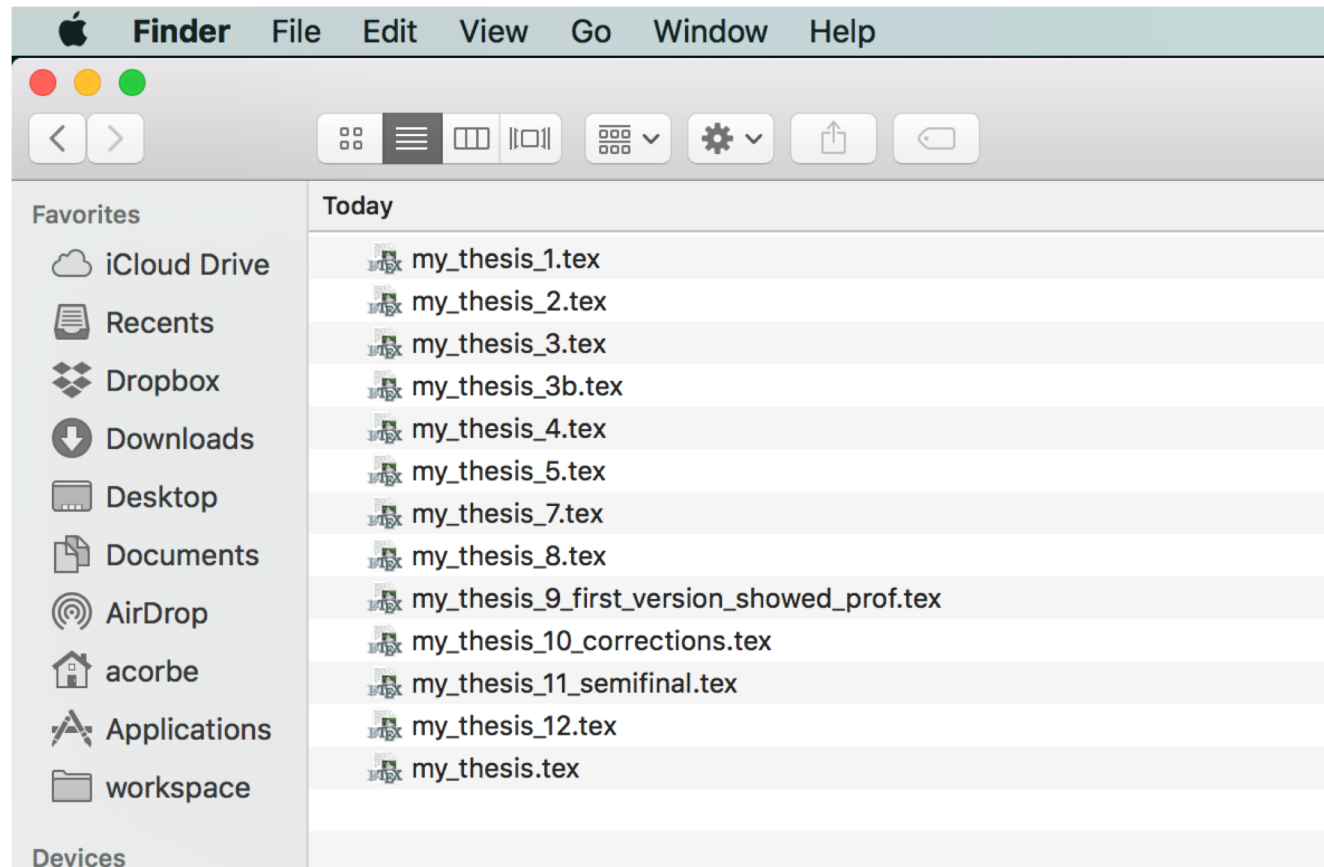
# Why version control is what everybody needs

- Soft start: latex alone



# Why version control is what everybody needs

- Soft start: latex alone

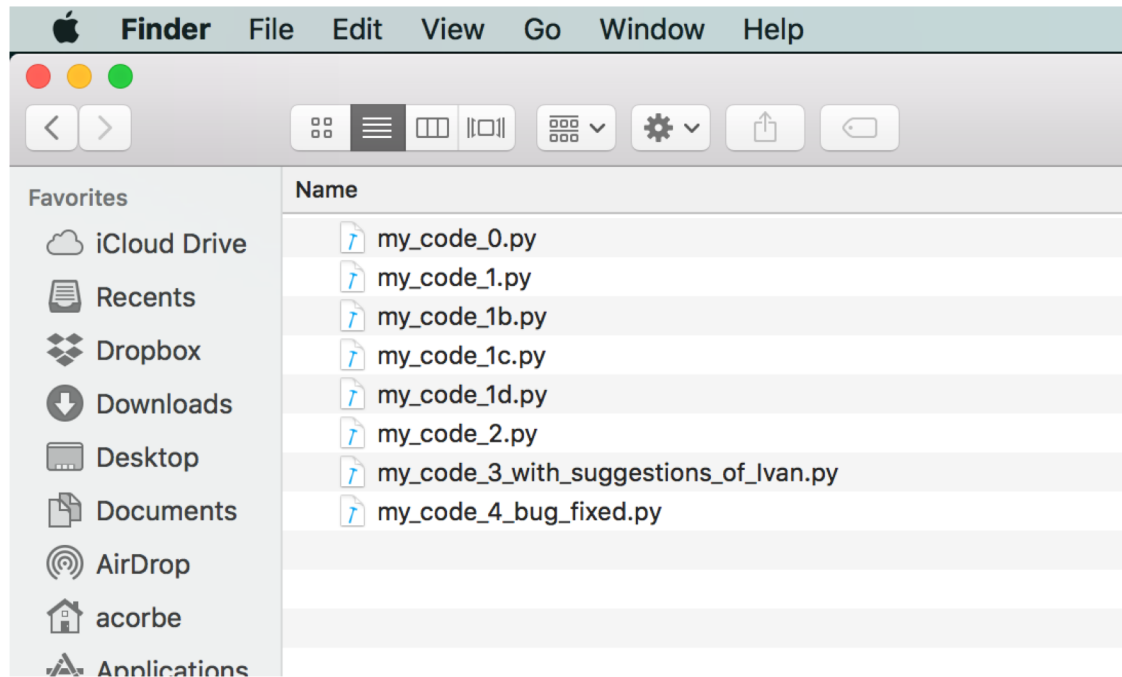


- What changed across the versions?
- A sentence was better written in some past version. Which one?

# Why version control is what everybody needs

- Case of an “ordinary code”

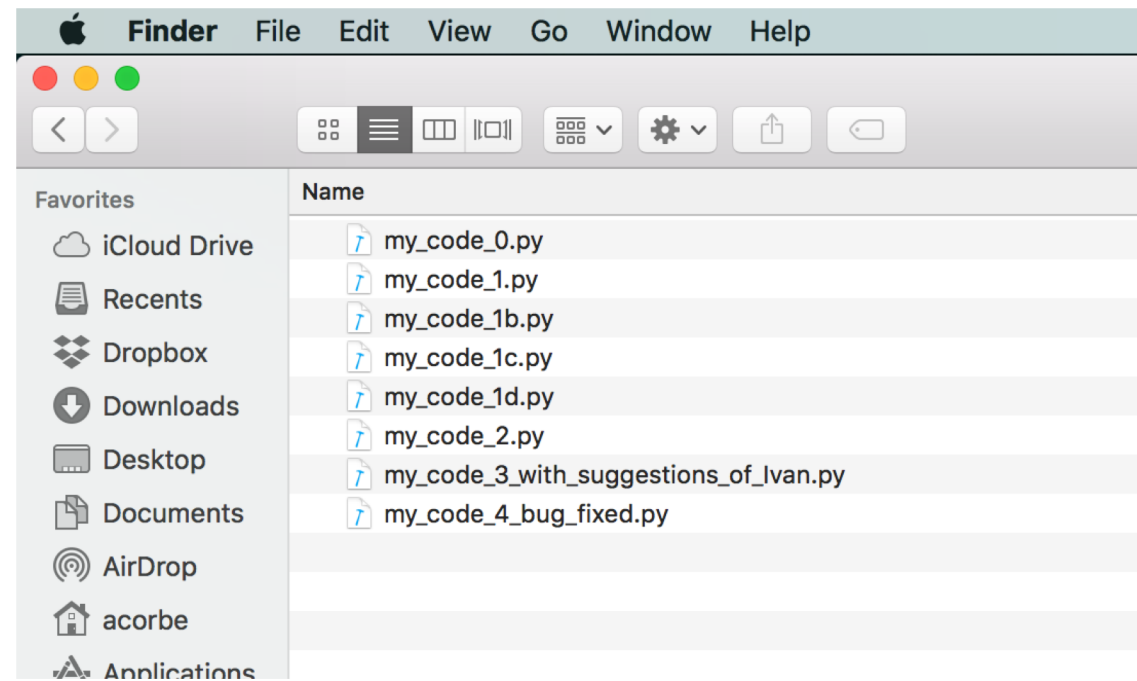
THIS BUG CERTAINLY APPEARED WHEN  
I DID **THAT** EDIT IN **SOME** VERSION...



- What changed across version?
- When was this bug introduced?
- What if I want to temporarily branch the development to try an edit?
  
- NOTE: still the vanilla version... we have the whole source in one file.  
What if we want to keep many versions of different files

# Why version control is what everybody needs

- Case of a “collaborative code” (Ivan & myself)
  - Ivan sent updates as email attachment. What did he change?
  - I changed simultaneously other things in the same file
  - What/how merge?



# Why version control is what everybody needs

- Case of a “collaborative code” with 100/1000 developers

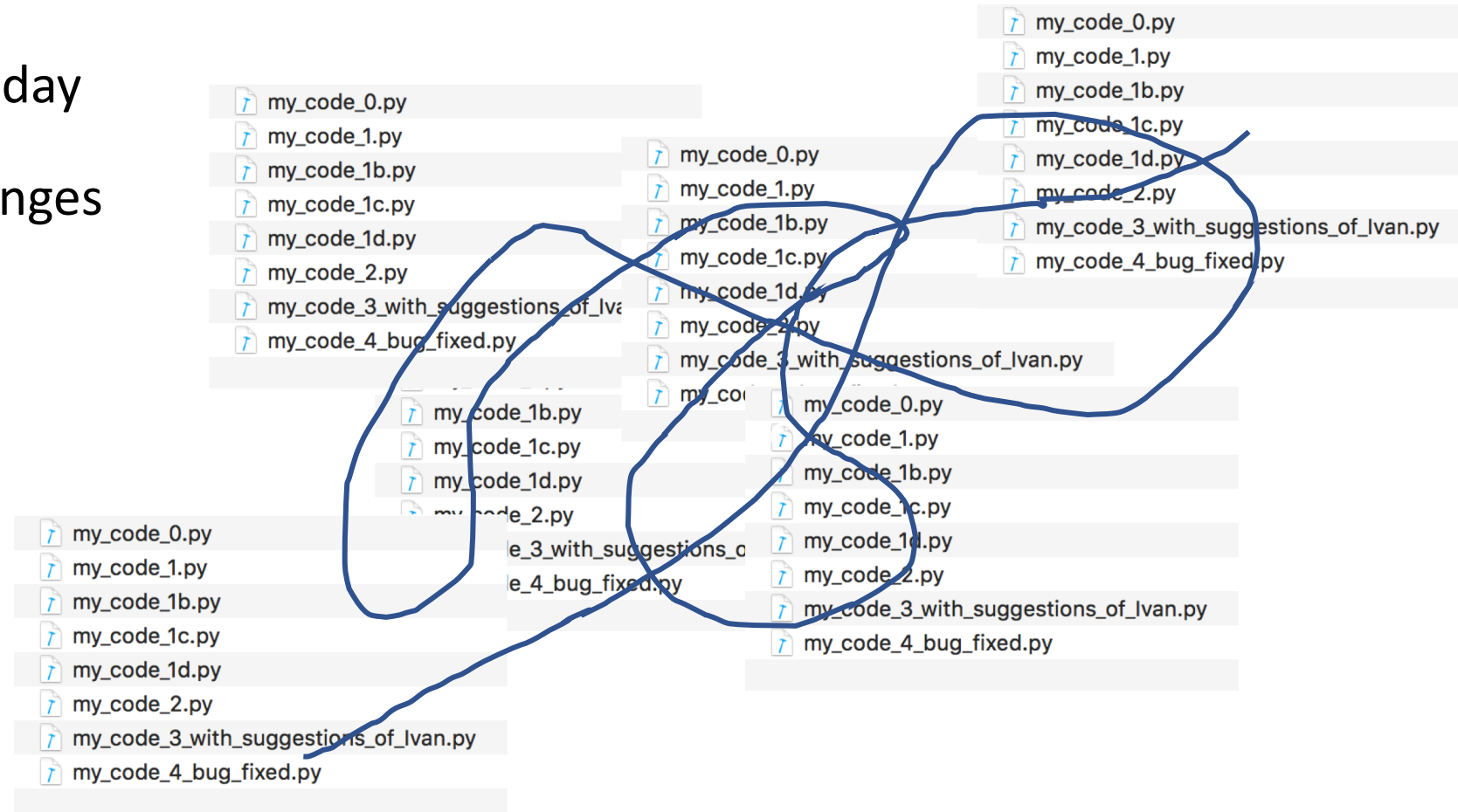
- 100/00 patches per day
- tons of email exchanges
- People going crazy
- Code NOT working



# Why version control is what everybody needs

- Case of a “collaborative code” with 100/1000 developers

- 100/00 patches per day
- tons of email exchanges
- People going crazy
- Code ain't working





# General solution for this general problem:

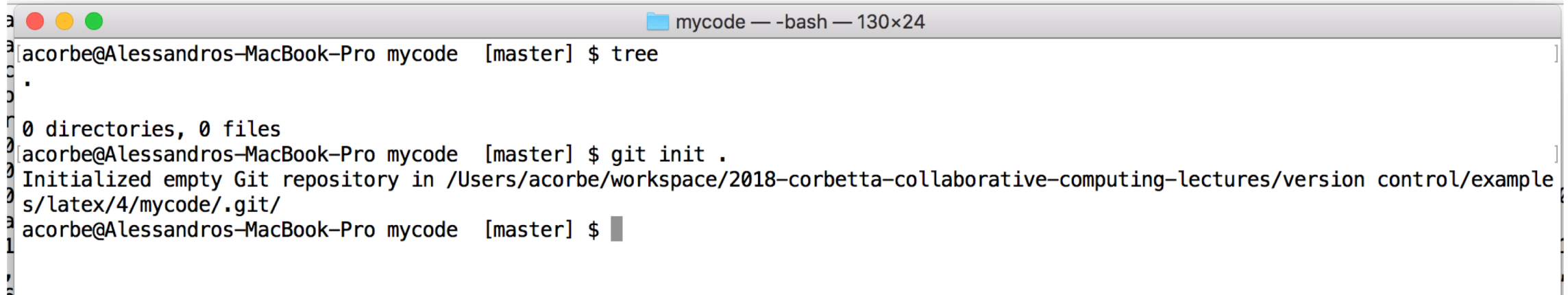
- We employ a (distributed) **Version Control** system
  - Most internationally used: **Git** (by far)
- What Git helps us with:

SOLO benefits	COLLABORATIVE <i>additional</i> benefits
Code in a “repository”	Common “remote repository”
Track all past versions + rollback	Merge contributions of different developers
Compare past versions	See who/when wrote any line
Branch development	

# Git commands for the SOLO version

```
git init .
```

- Clean start

A terminal window titled 'mycode -- -bash -- 130x24' showing the execution of 'git init .' and the resulting output. The terminal text is as follows:

```
acorbe@Alessandros-MacBook-Pro mycode [master] $ tree
.
0 directories, 0 files
acorbe@Alessandros-MacBook-Pro mycode [master] $ git init .
Initialized empty Git repository in /Users/acorbe/workspace/2018-corbetta-collaborative-computing-lectures/version control/examples/latex/4/mycode/.git/
acorbe@Alessandros-MacBook-Pro mycode [master] $
```

- Git knows that the mycode folder has content to be tracked.

# Git commands for the SOLO version

- Let's use the `converter.py` example. After copying the basic files

```
acorbe@Alessandros-MacBook-Pro mycode [master] $ tree
├── converter.py
└── tests
    └── tests_unbiased.py

1 directory, 2 files
acorbe@Alessandros-MacBook-Pro mycode [master] $ cat converter.py
import numpy as np
def binary_number_string_parser(inp
    , datatype = float):

    return 0

def main():

    return binary_number_string_parser(0)

if __name__ == '__main__':
    main()

acorbe@Alessandros-MacBook-Pro mycode [master] $
```

# Git commands for the SOLO version

## git add <> / status / commit

- Let's use the `converter.py` example. After copying the basic files

```
acorbe@Alessandros-MacBook-Pro mycode [master] $ tree
├── converter.py
└── tests
    └── tests_unbiased.py

1 directory, 2 files
acorbe@Alessandros-MacBook-Pro mycode [master] $ cat converter.py
import numpy as np
def binary_number_string_parser(inp
    , datatype = float):

    return 0

def main():

    return binary_number_string_parser(0)

if __name__ == '__main__':
    main()

acorbe@Alessandros-MacBook-Pro mycode [master] $
```

I want Git to start tracking these two files.

```
$ git add converter.py
$ git add tests/tests_unbiased.py

acorbe@Alessandros-MacBook-Pro mycode_1 [master] $ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   converter.py
    new file:   tests/tests_unbiased.py

acorbe@Alessandros-MacBook-Pro mycode_1 [master] $
```

```
$ git commit -m 'Initial commit'
# I took an initial snapshot.
```

# Git commands for the SOLO version

`git log`

- `git log` (1 snapshot + hash -- unique)

```
[acorbe@Alessandros-MacBook-Pro mycode_1 [master] $ git log
commit ac1408245b78a7a8b53f5040b8780c1733fdf6af (HEAD -> master)
Author: Alessandro Corbetta <corbisoft@gmail.com>
Date: Tue May 1 16:27:09 2018 +0430

    initial commit
acorbe@Alessandros-MacBook-Pro mycode_1 [master] $ █
```

---

# Git commands for the SOLO version

## git diff

- Starting with edits...
  - `git diff` shows the difference with the last commit
  - Standard patch format
- Works analogously (but with arguments to compare any version with any other version)

```
acorbe@Alessandros-MacBook-Pro my_code_2 [master] $ git diff
diff --git a/converter.py b/converter.py
index 4511134..58edb46 100644
--- a/converter.py
+++ b/converter.py
@@ -1,13 +1,16 @@
+from __future__ import print_function
import numpy as np
+import sys
+
def binary_number_string_parser(inp
    , datatype = float):
-
+    print (inp)
    return 0

def main():
-
-    return binary_number_string_parser(0)
+
+    return binary_number_string_parser(sys.argv[1])

if __name__ == '__main__':
acorbe@Alessandros-MacBook-Pro my_code_2 [master] $
```

# Git commands for the SOLO version

## git diff

- Starting with edits...
  - After staging (`git add converter.py`) and committing, the log shows 2 versions

```
my_code_2 — -bash — 130x34
[acorbe@Alessandros-MacBook-Pro my_code_2 [master] $ git add converter.py
[acorbe@Alessandros-MacBook-Pro my_code_2 [master] $ git commit -m 'added command line capturing'
[master 96bf846] added command line capturing
 1 file changed, 6 insertions(+), 3 deletions(-)
[acorbe@Alessandros-MacBook-Pro my_code_2 [master] $ git log
commit 96bf846d46db1177cd04613cbb9dbdd71dd1d6f7 (HEAD -> master)
Author: Alessandro Corbetta <corbisoft@gmail.com>
Date: Tue May 1 16:39:02 2018 +0430

    added command line capturing

commit ac1408245b78a7a8b53f5040b8780c1733fdf6af
Author: Alessandro Corbetta <corbisoft@gmail.com>
Date: Tue May 1 16:27:09 2018 +0430

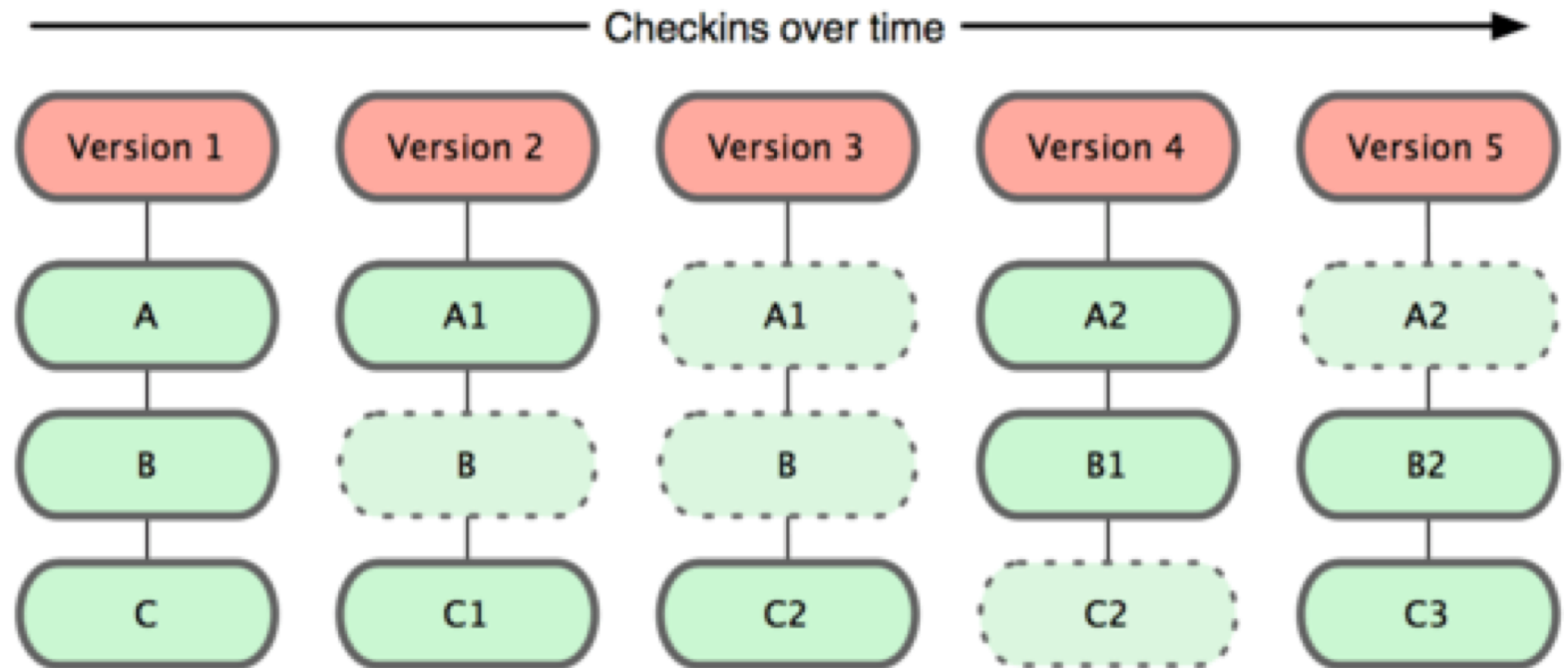
    initial commit
acorbe@Alessandros-MacBook-Pro my_code_2 [master] $
```

# Git commands for the SOLO version

- Each commit generates a complete snapshot of the repository

`git checkout <hash>`

To go back to the previous version (some caveat “we are in detached head”)





# Git commands for the SOLO version

- Each commit generates a complete snapshot of the repository

`git checkout <hash>`

To go back to the previous version (some caveat “we are in detached head”)

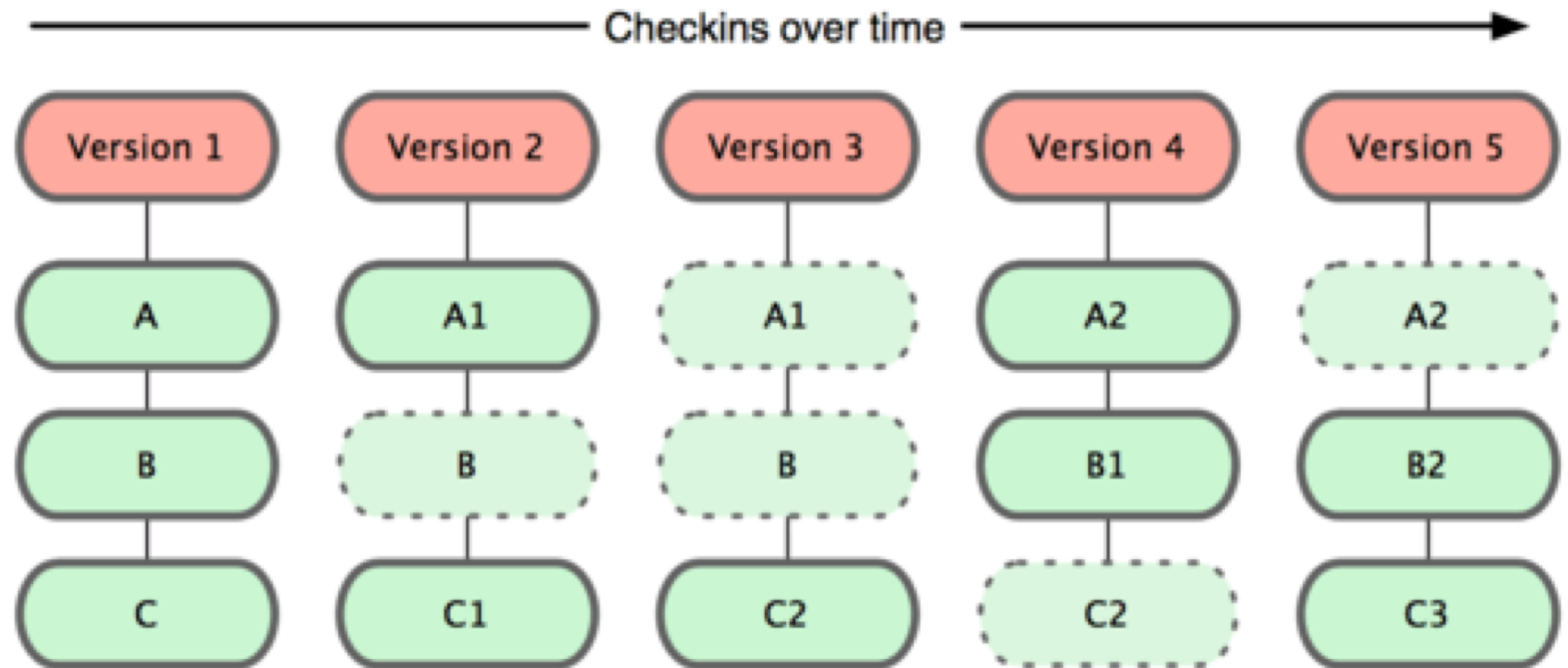
**NOTE:** need to use

```
git mv f1 f2
```

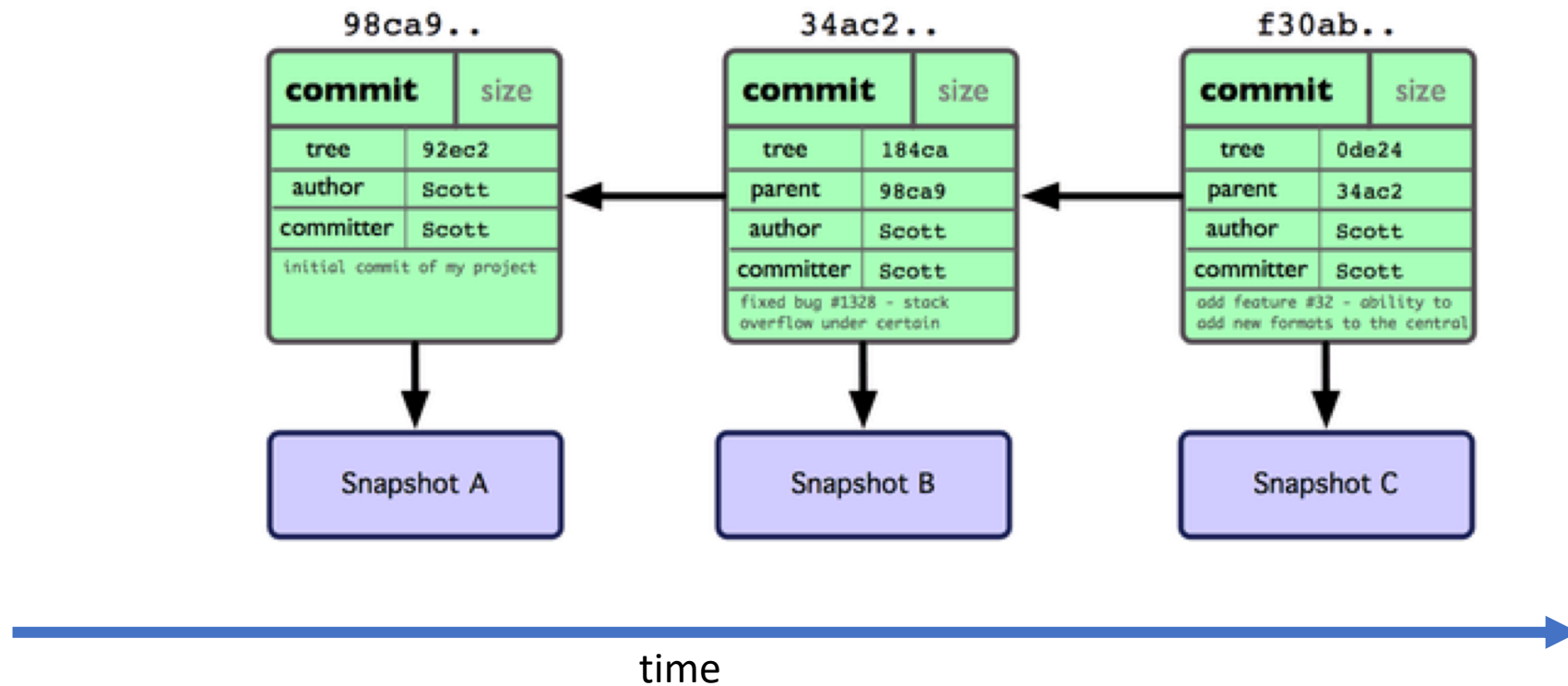
or

```
git rm f1
```

To move/remove and obtain the expected behavior

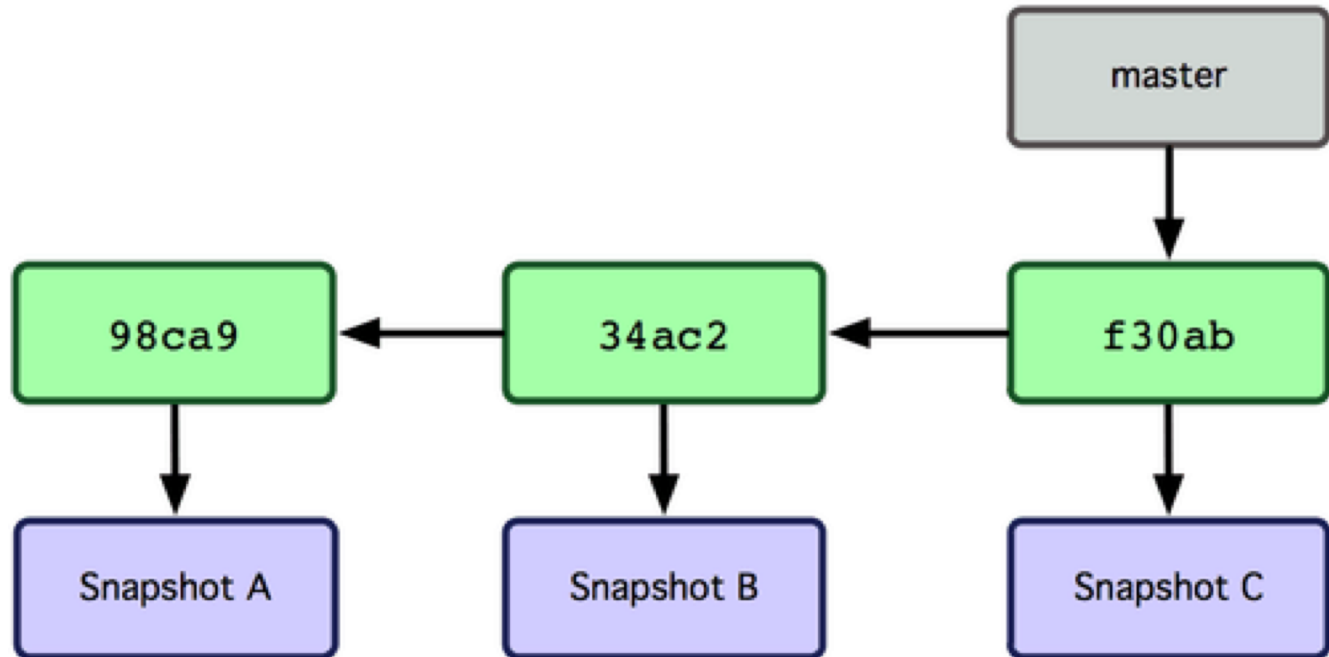


# Versions in git are a linked lists of hashes



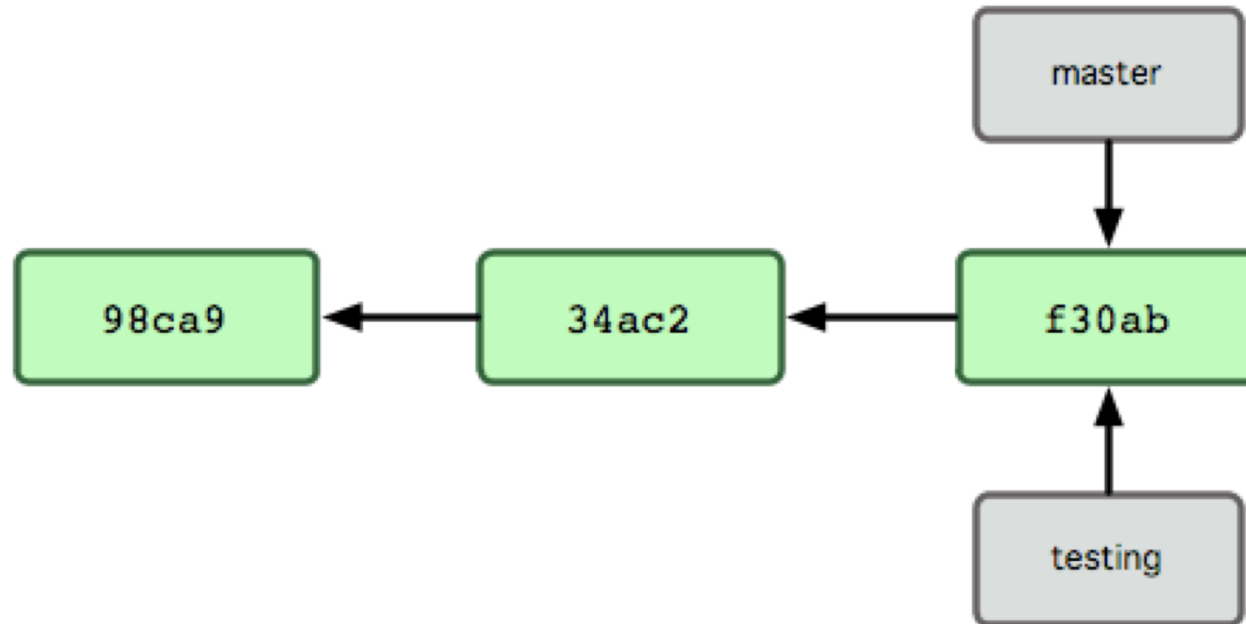
# Branches and branching

- By default we are in the `master` branch.
- Commit after commit the branch is advanced automatically

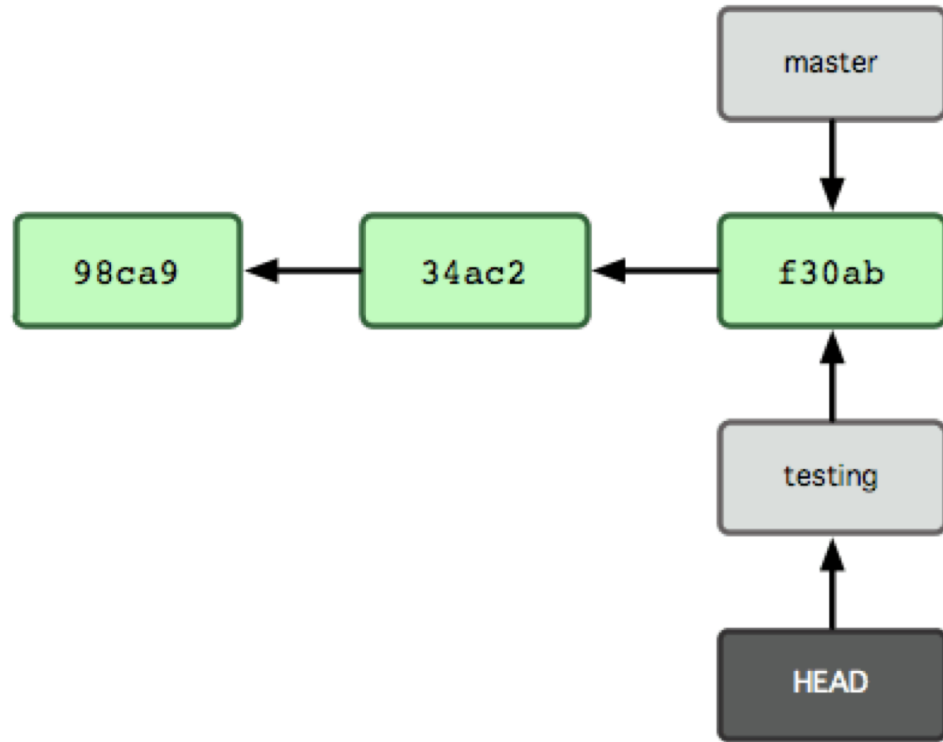


# Branches and branching – `git branch <>`

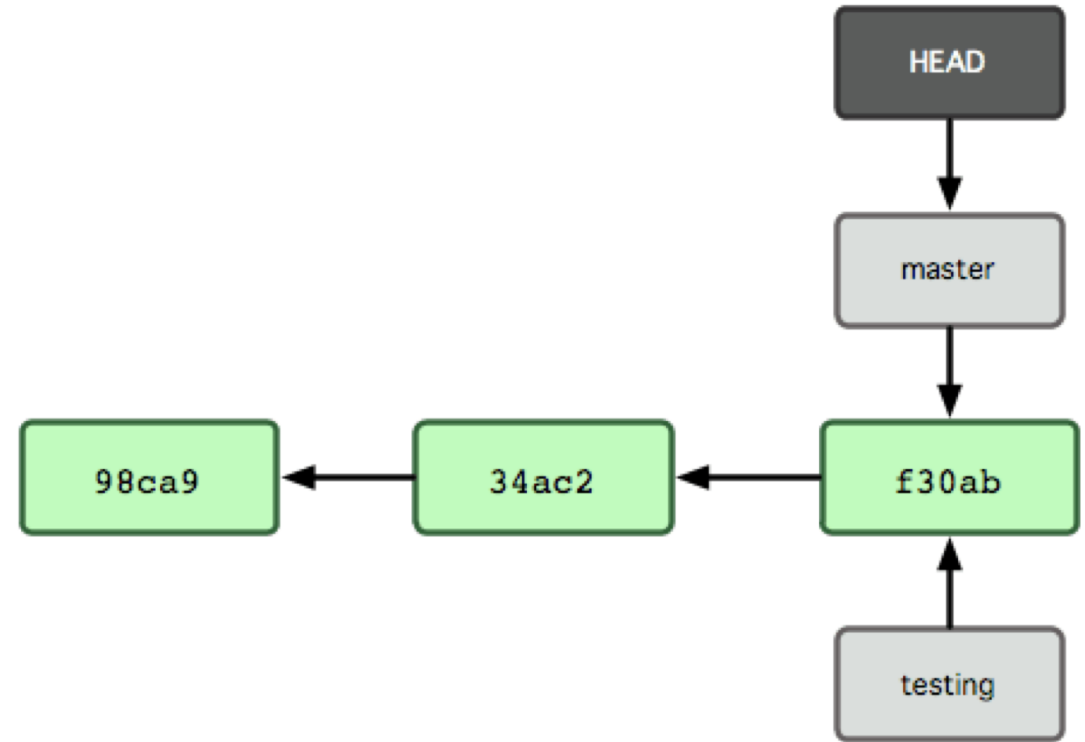
- We create a new branch via  
`git branch testing`



# HEAD: a special branch pointing to the current branch



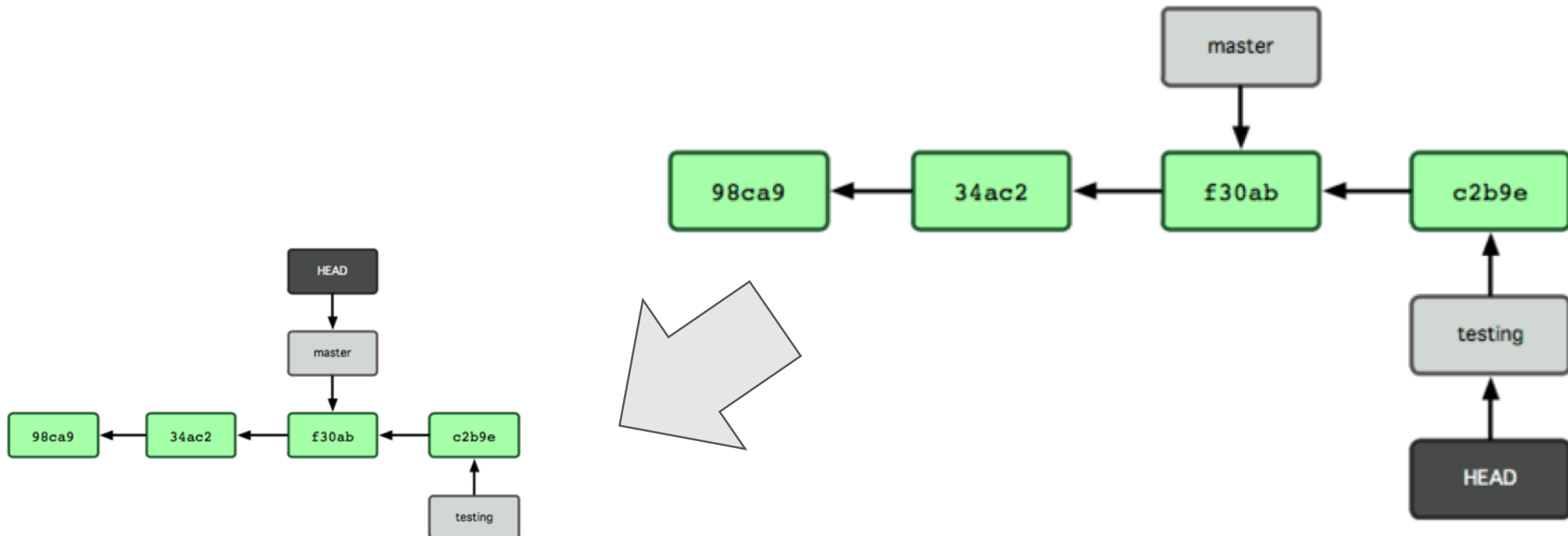
`git checkout testing`



`git checkout master`

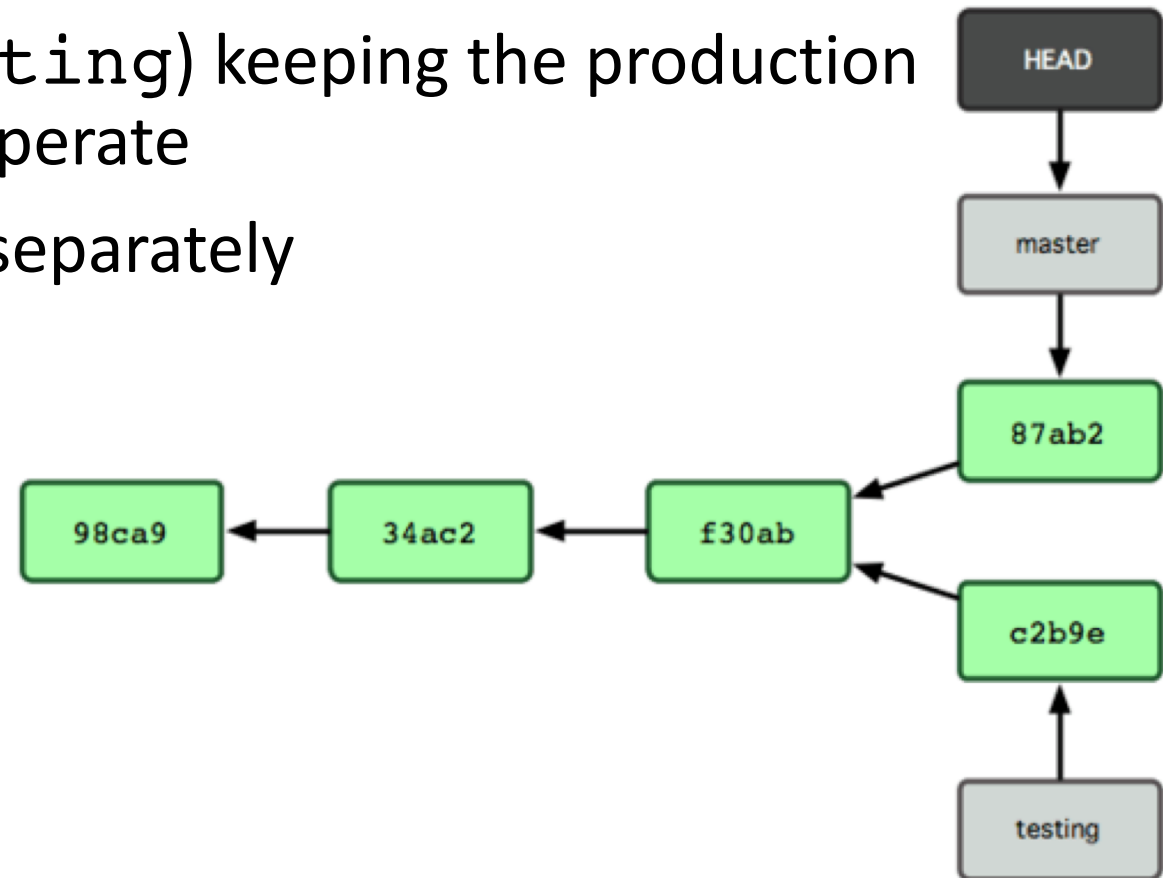
# Branches can advance independently!

- The wonder starts!!
- I can test new modifications (in `testing`) keeping the production code (in `master`) always ready to operate



# Branches can advance independently!

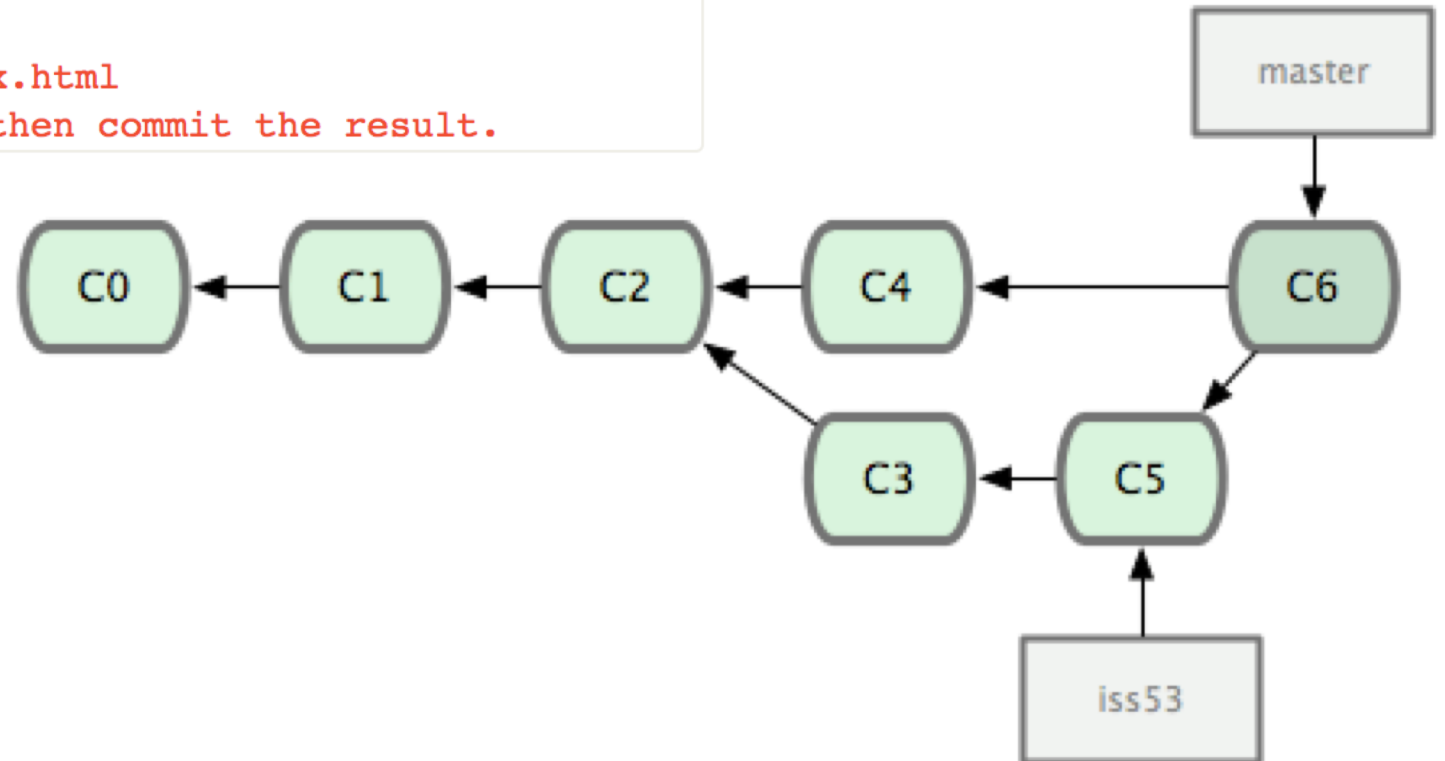
- The wonder starts!!
- I can test new modifications (in `testing`) keeping the production code (in `master`) always ready to operate
- `Master` and `testing` can evolve separately



# Merging

- `git checkout master`
- `git merge iss53`

```
$ git merge iss53  
Auto-merging index.html  
CONFLICT (content): Merge conflict in index.html  
Automatic merge failed; fix conflicts and then commit the result.
```

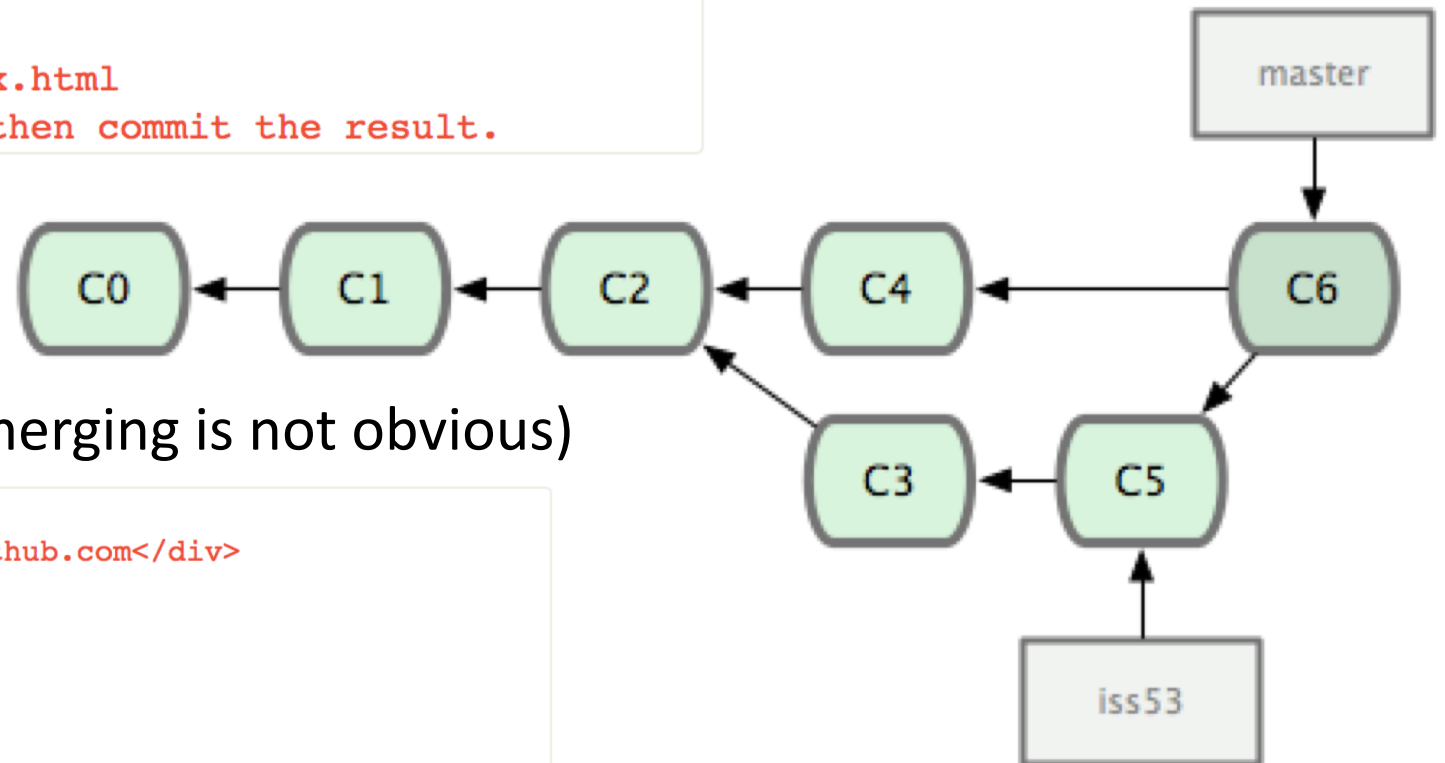




# Merging

- `git checkout master`
- `git merge iss53`

```
$ git merge iss53
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```

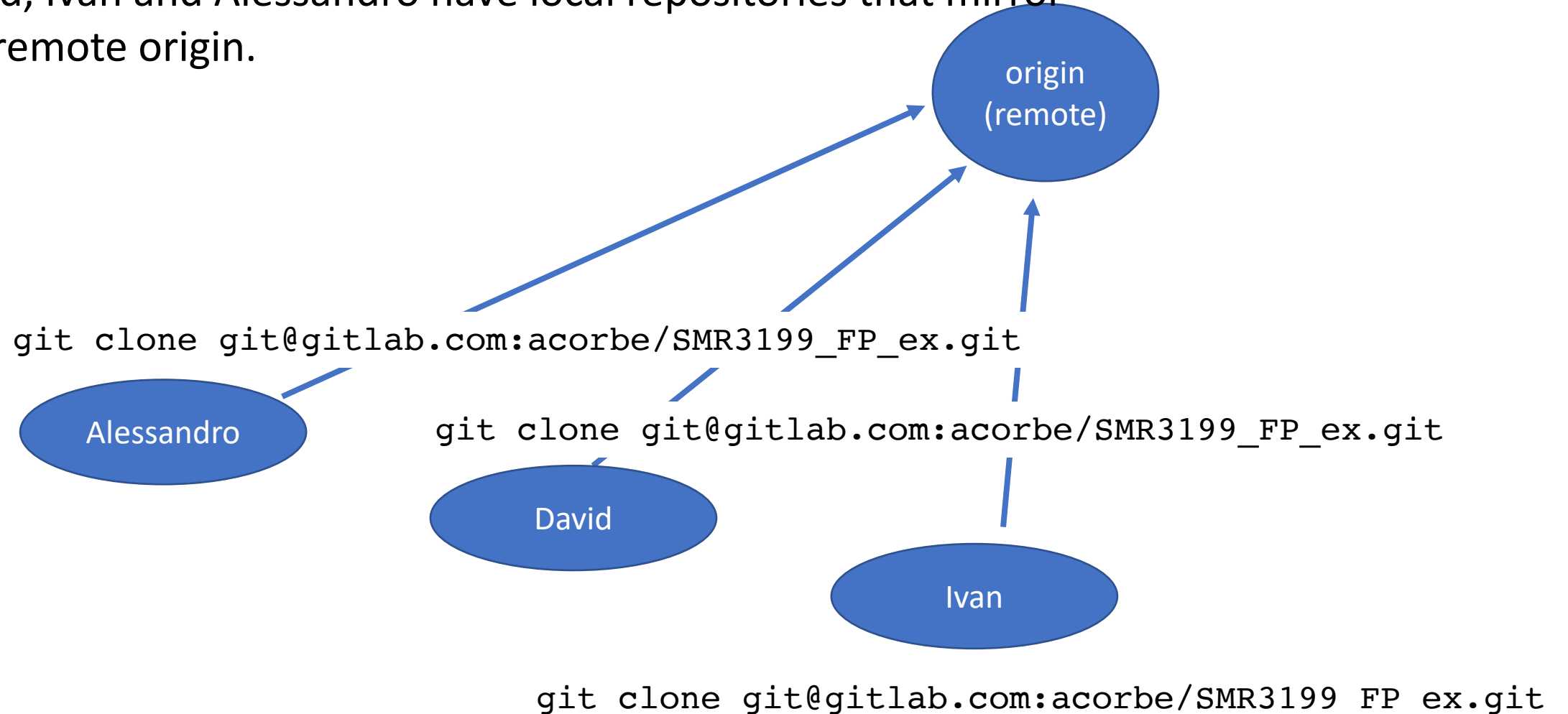


Manual merging might be needed (if merging is not obvious)

```
<<<<<< HEAD:index.html
<div id="footer">contact : email.support@github.com</div>
=====
<div id="footer">
  please contact us at support@github.com
</div>
>>>>>> iss53:index.html
```

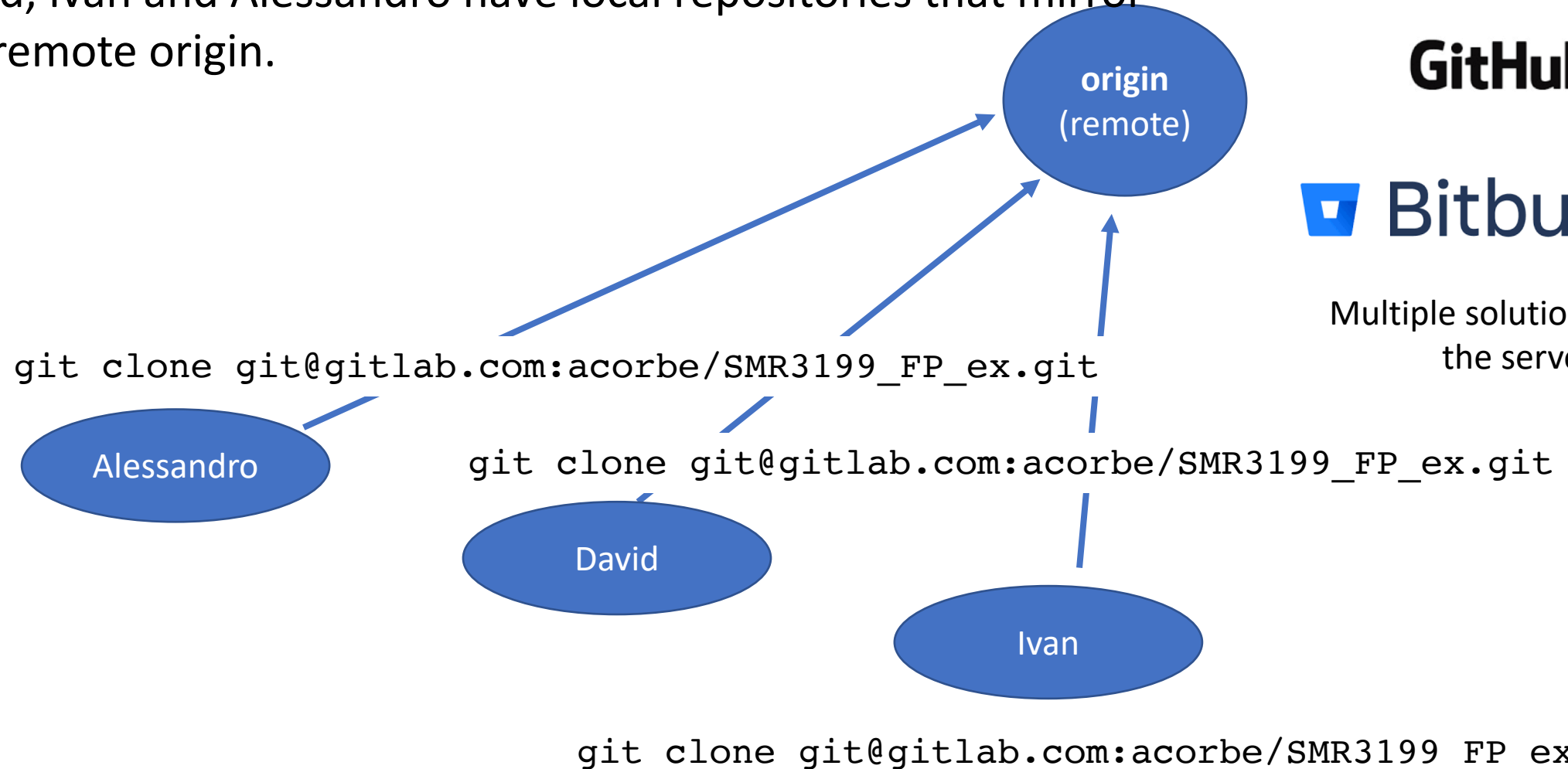
# Collaborative development

David, Ivan and Alessandro have local repositories that mirror the remote origin.



# Collaborative development

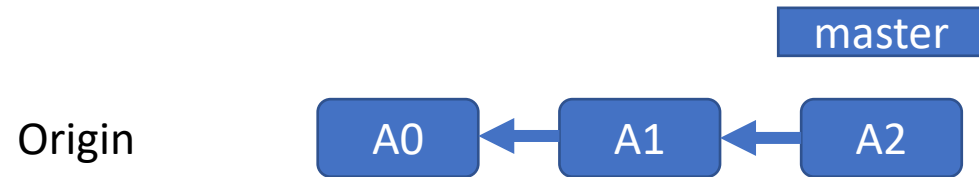
David, Ivan and Alessandro have local repositories that mirror the remote origin.



Multiple solutions to host the server yourself

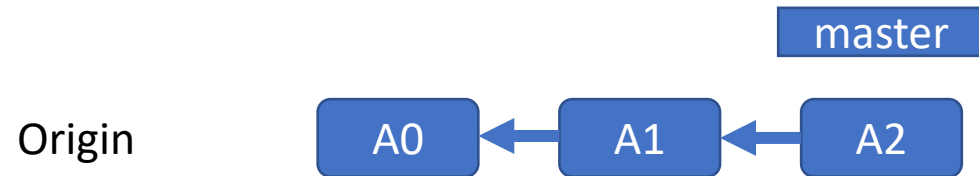
# Collaborative development

What happens after clone?



# Collaborative development

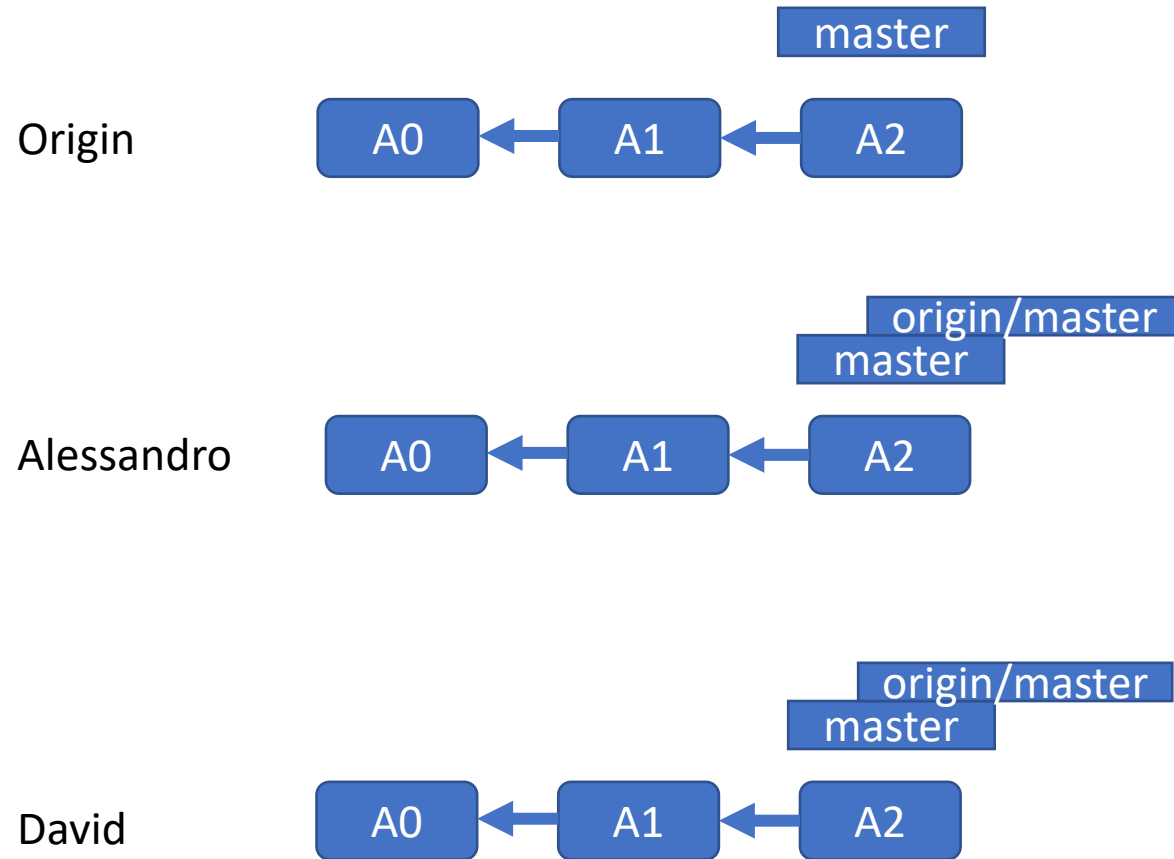
What happens after clone?



Alessandro & David both do `git clone`

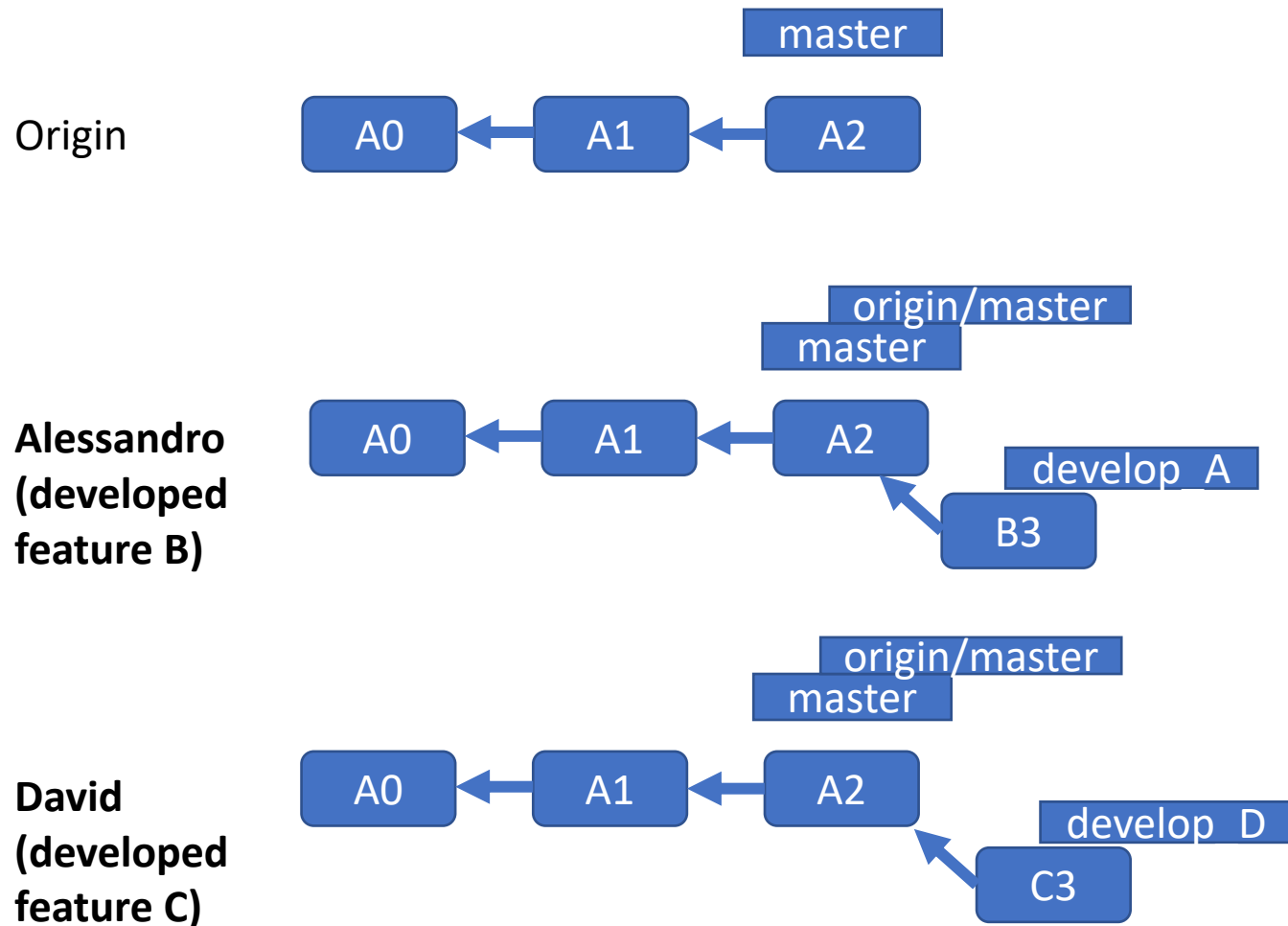
# Collaborative development

What happens after clone?



# Collaborative development

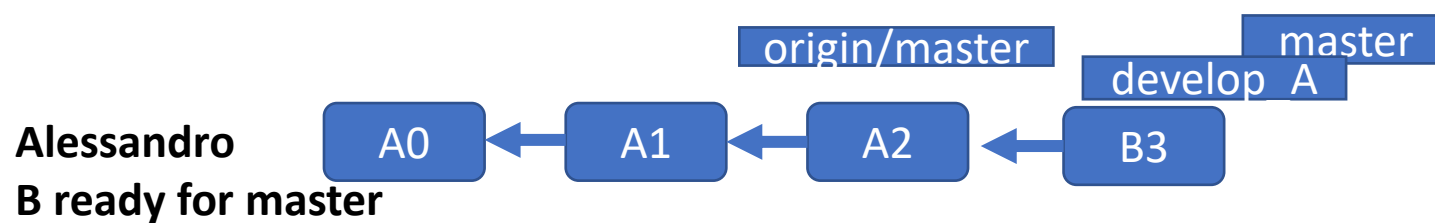
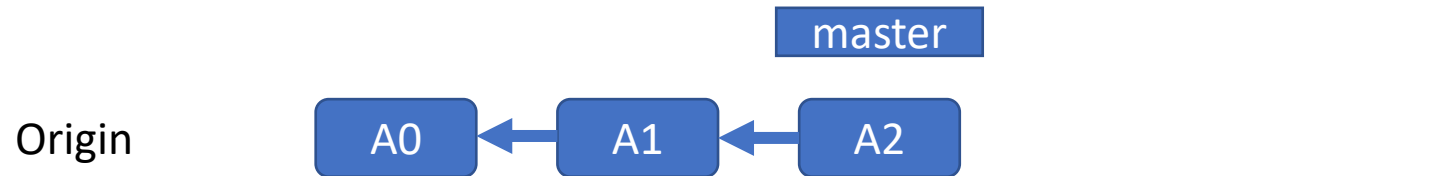
What happens after clone?



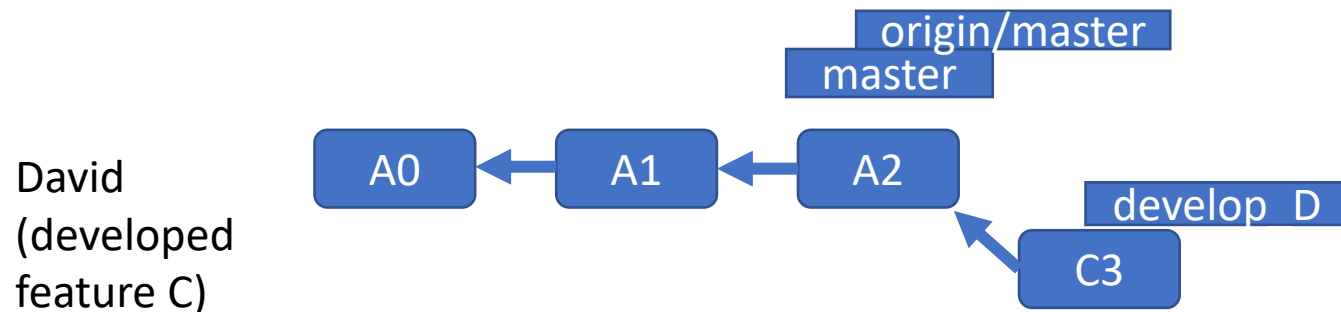
Both David and Alessandro committed

# Collaborative development

What happens after clone?



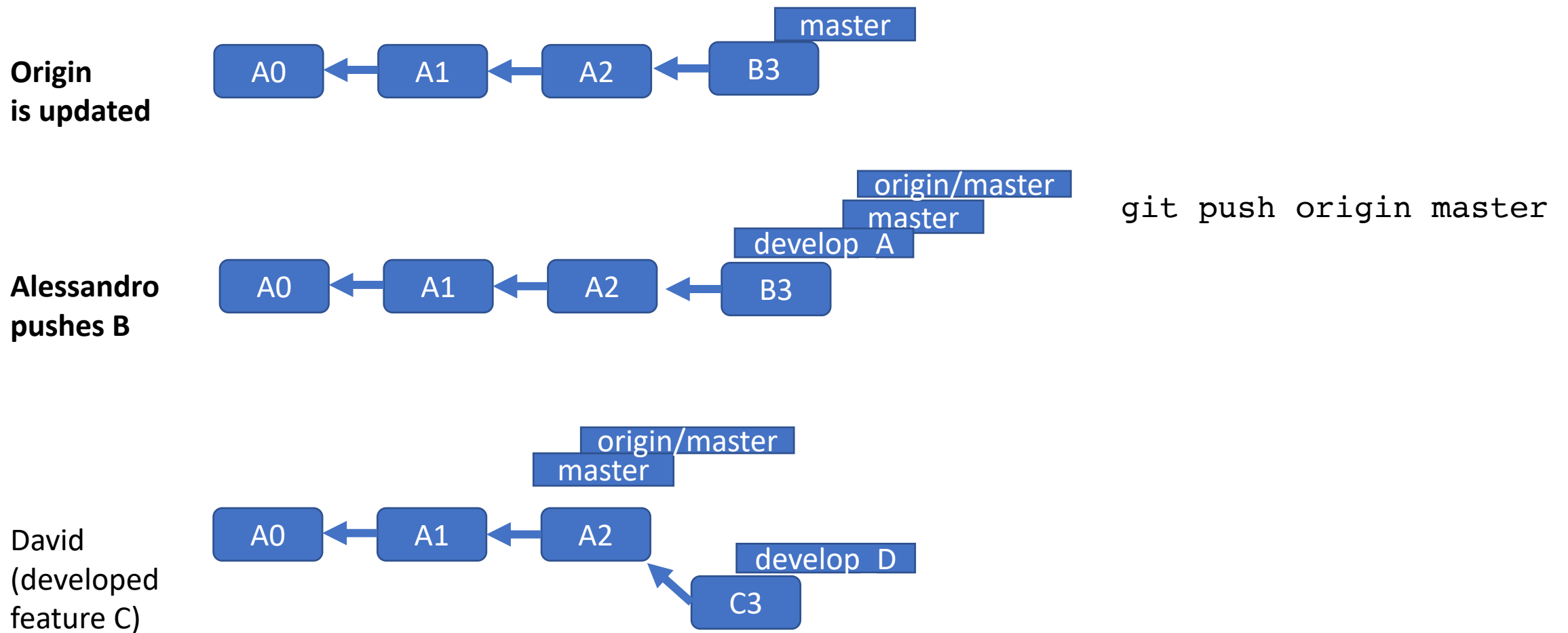
```
git checkout master  
git pull  
git merge develop_A
```





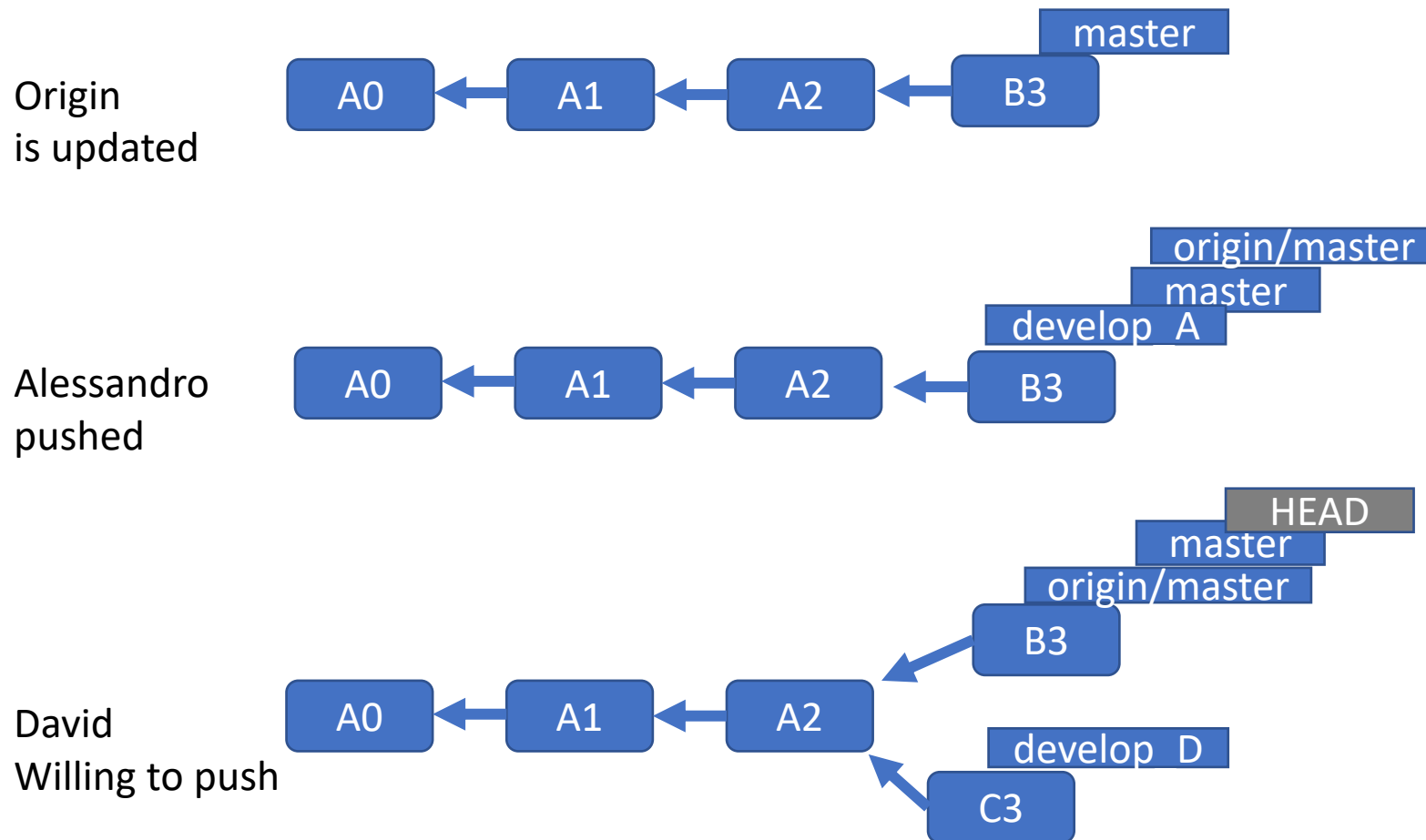
# Collaborative development

What happens after clone?



# Collaborative development

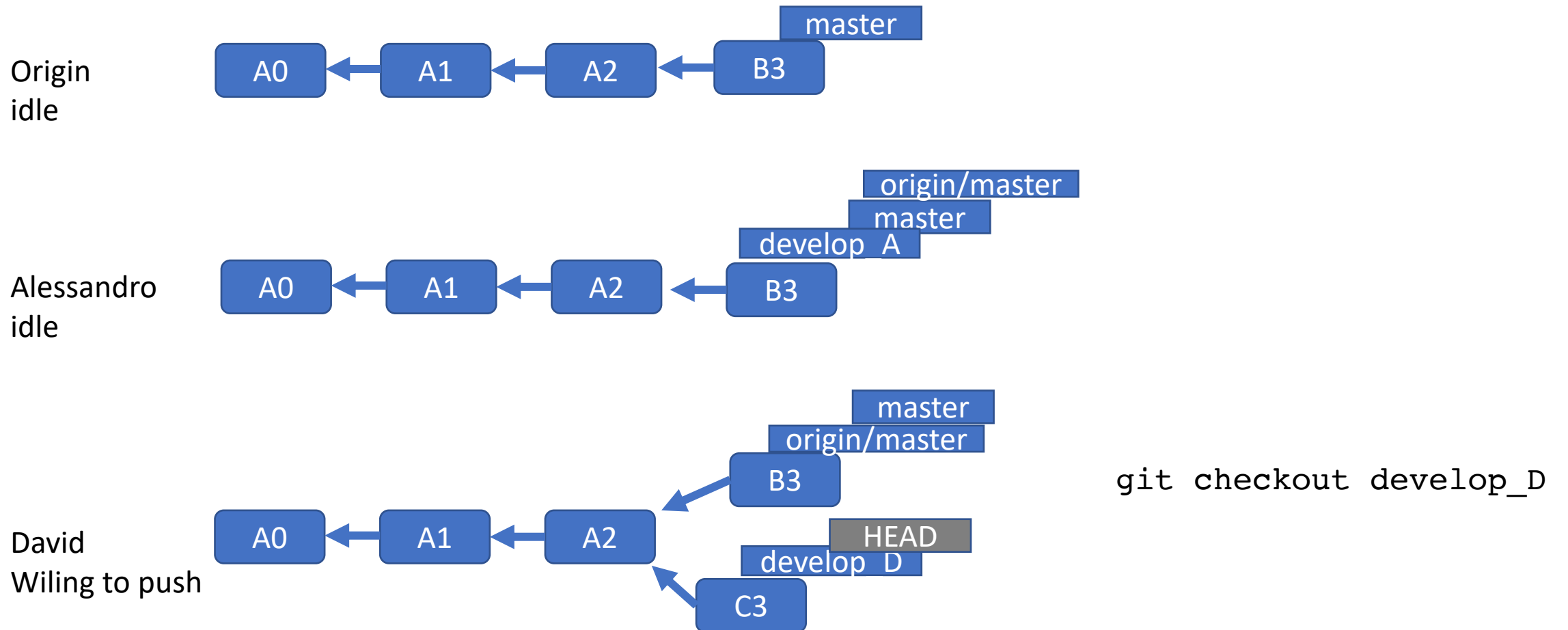
What happens after clone?



```
git checkout master  
git pull #UPDATES!!
```

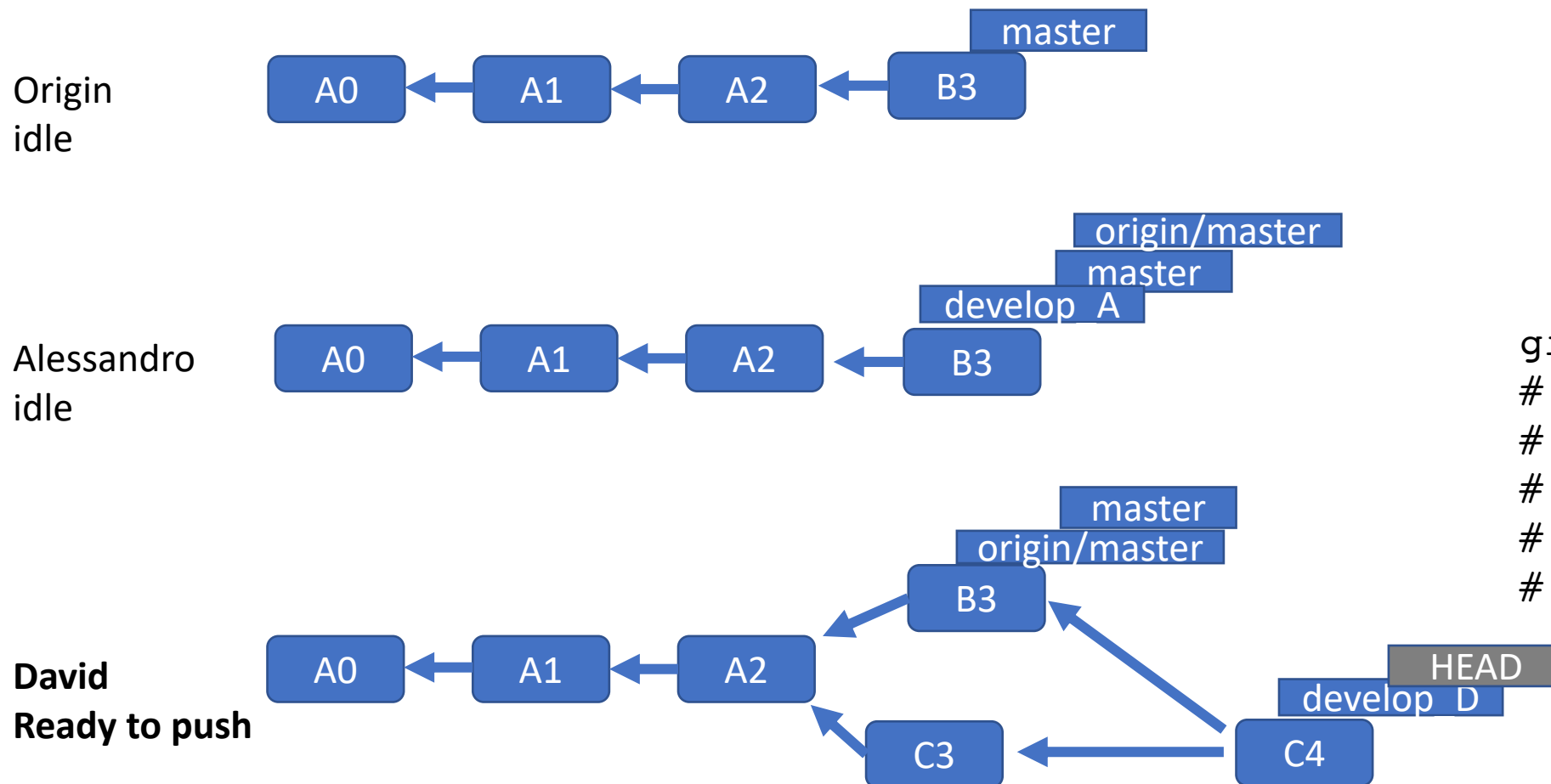
# Collaborative development

What happens after clone?



# Collaborative development

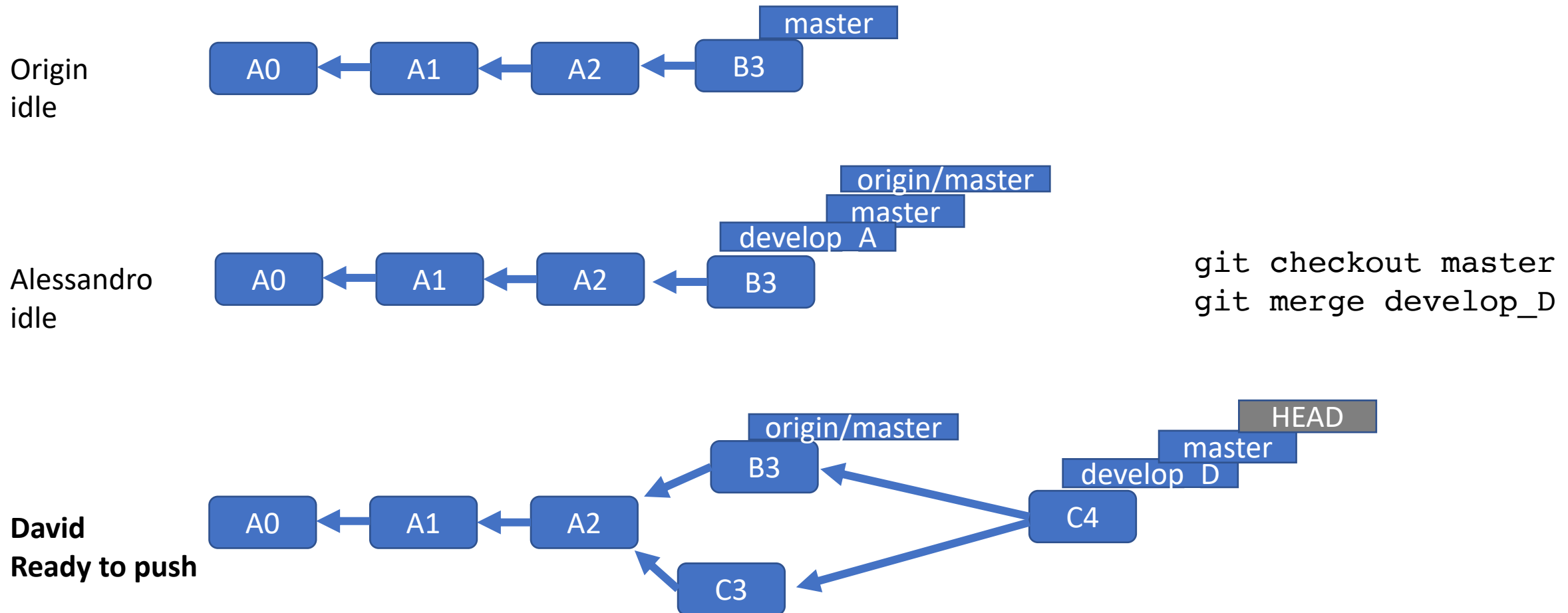
What happens after clone?



```
git merge master
# develop now is
# newer than master
# by definition can
# be safely merged
# into master
```

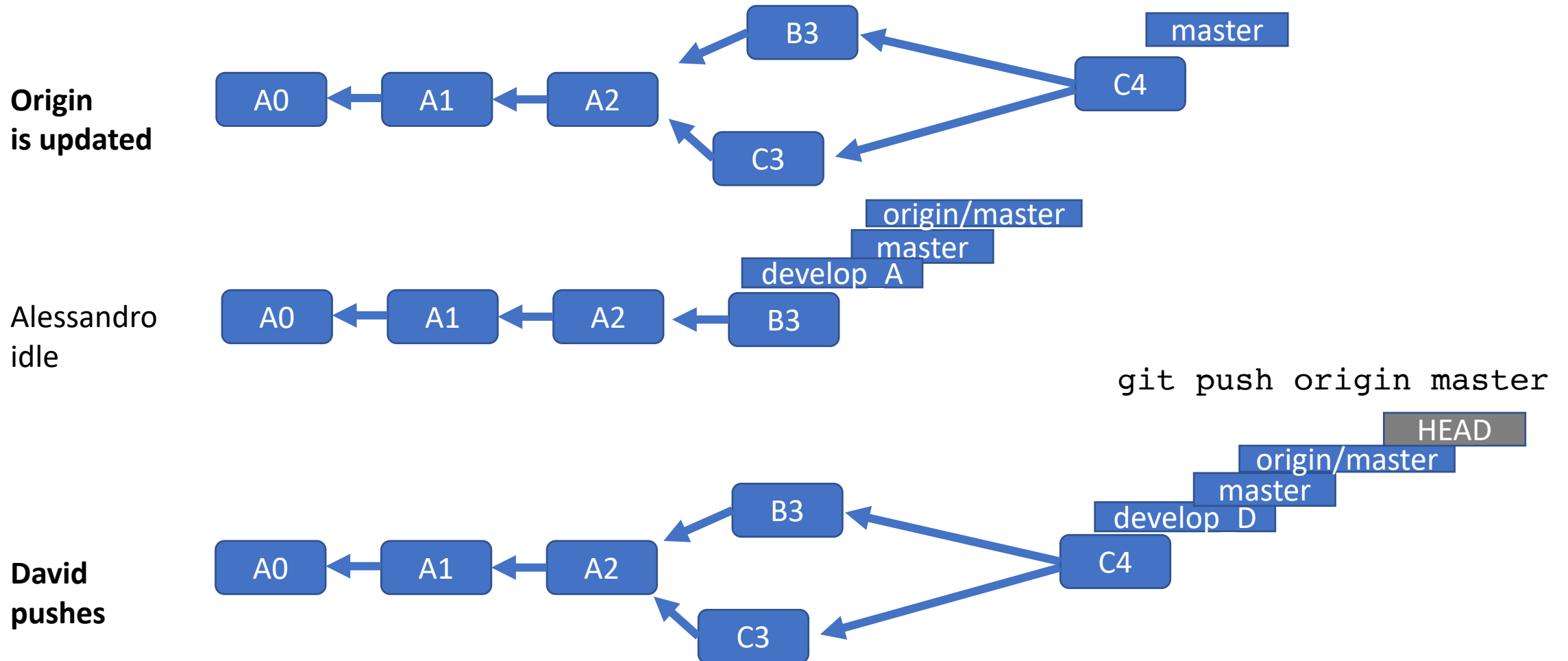
# Collaborative development

What happens after clone?



# Collaborative development

What happens after clone?



# Pulling/Pushing commits to the origin

- The origin remote can change OR we want to change the origin remote
- To update <branch> locally
  - `git checkout <branch>`
  - `git pull origin <branch>`
- To push the contribution back to the remote
  - `git checkout <branch>`
  - `git push origin <branch>`

# Pulling/Pushing commits to the origin

Alessandro C > kinectWebserver > Commits

hog kinectWebserver [Create merge request](#)

03 Apr, 2018 1 commit

**Integrated flow-based**  
Werner Kroneman authored 4 weeks ago e1459cb6

26 Feb, 2018 2 commits

**Merge branch 'hog' of git.phys.tue.nl:acorbe/kinectWebserver into hog**  
Werner Kroneman authored 2 months ago

**Using new cl5**  
Werner Kroneman authored 2 months ago

23 Feb, 2018 1 commit

**Added support for ROI with HA-HOG.**  
Werner Kroneman authored 2 months ago

21 Feb, 2018 5 commits

**Fixed bug where the heartbeat would fail if no people were present.**  
Werner Kroneman authored 2 months ago

**Update .gitmodules**  
Werner Kroneman authored 2 months ago

**Adjusted submodule path to relative.**  
Werner Kroneman authored 2 months ago

