What is a Code Repository?

- A place to store your code
  - Possibly on your computer, possibly not… but definitely versioned
- A place to *show* your code, and work with others.

what About Actually Laying out the Code?

- There's not an easy answer for science code - it tends to develop "organically".
- Often it's best just to split files when they get too big.
- Always keep the novice user (or future you) in mind… Use descriptive names.
- ***Think modular!***

What About Packaging Code?

- Deliver your code in some form that others can install without thinking too hard about where anything goes.
  - Makefiles, ruby gems, python packages, etc.
  - (Includes sensible versioning!)

What About Packaging Python Code?

Python Packaging Terminology

- "**package**": the biggest thing. E.g., *astropy*, *numpy*, *sunpy*. A directory with an "__init__.py"
- "**module**": a single "something.py" file - the module is "something"
- "**subpackage**": a package within a package
- "**source directory/folder**": the directory/folder with all of a codes "stuff"
- "**repository**"/"**repo**": the source directory *in version control*
- "**submodule**": a git repo embedded in *another* git repo
    - "**astropy-helpers**": an example seen in Astropy packages

Sample Package Layout

```
README
LICENSE
setup.py

mypackage/__init__.py
mypackage/mymodule.py
mypackage/secondmodule.py
mypackage/subpackage/__init__.py
mypackage/subpackage/anothermodule.py
```

```
import mypackage
from mypackage import my module
from mypackage import secondmodule
from mypackage import subpackage
from mypackage.subpackage import anothermodule
```

The goal of packaging and installing is basically to make that work anywhere

Versioning

- In vogue: “semantic versioning”
- x.y.z (E.g., 0.2.3, 2.7.12, 3.6)
  - change *x* for breaking changes
  - change *y* for non-breaking changes
  - change *z* for bug-fixes
- Anything x.y.z<something else> is a pre-release
  - E.g., 1.2.3beta, 2.1.6rc2
- But don’t get too worked up. 0.1 -> 0.2 -> 0.3 is better than nothing.

Licensing Your Code

- Rule #1: Have a license!
- Rule #2: There is no rule #2.

(see problem sets for more)

Go to:
https://github.com/eteq/python-packaging

Get:
PackagingAndRepos.ipynb

(May want to skip to Problem 3)

More Git/Github Background Material

Lets dig down on how you do shared development with public repos using Github.

More Git/Github Background Material

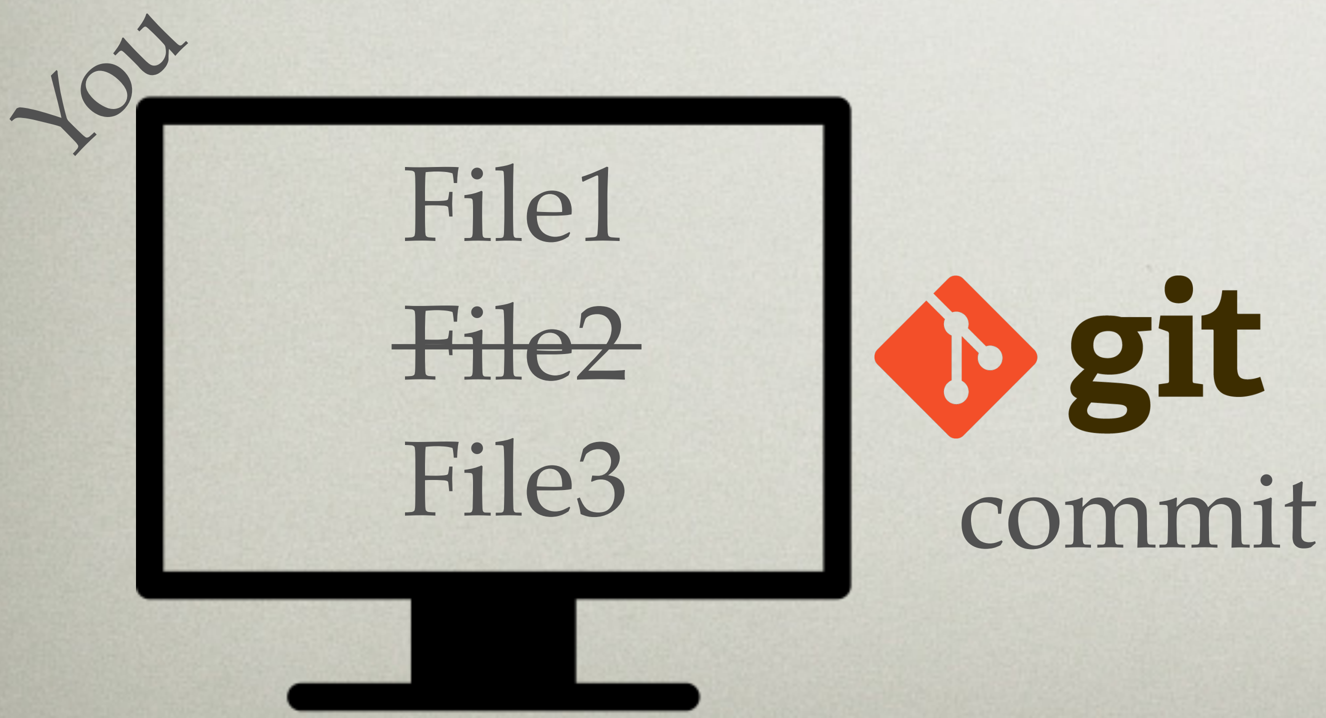
Point to note: git is **not** the same thing as Github. *git* is the software, Github is a web site/service for storing code that you interact with via *git*.

This results in most of the complexity - you need to manage your local copy *and* a Github copy, usually using *git* + the web site itself.

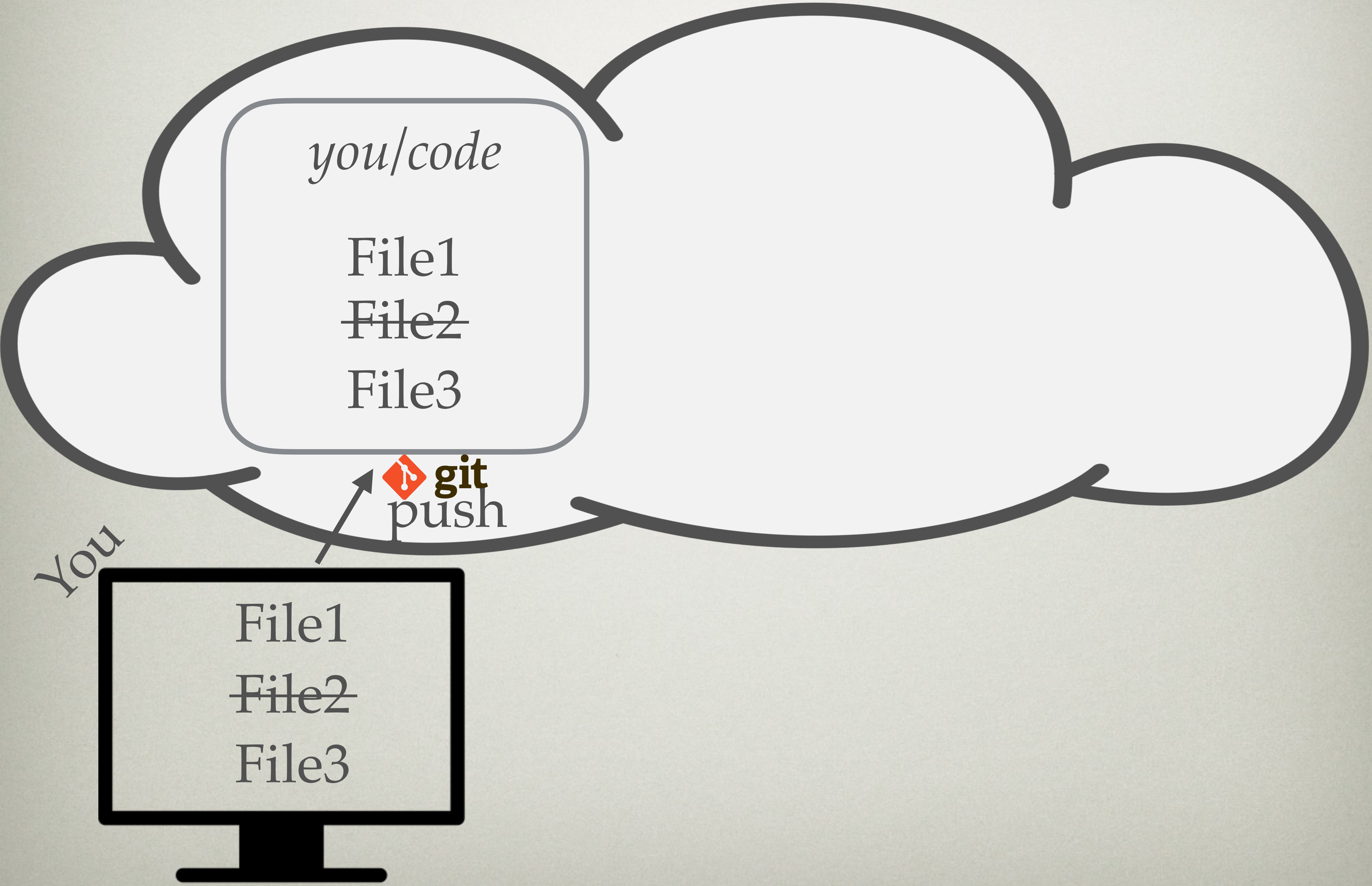
Lets dig down...

You
File1
File2

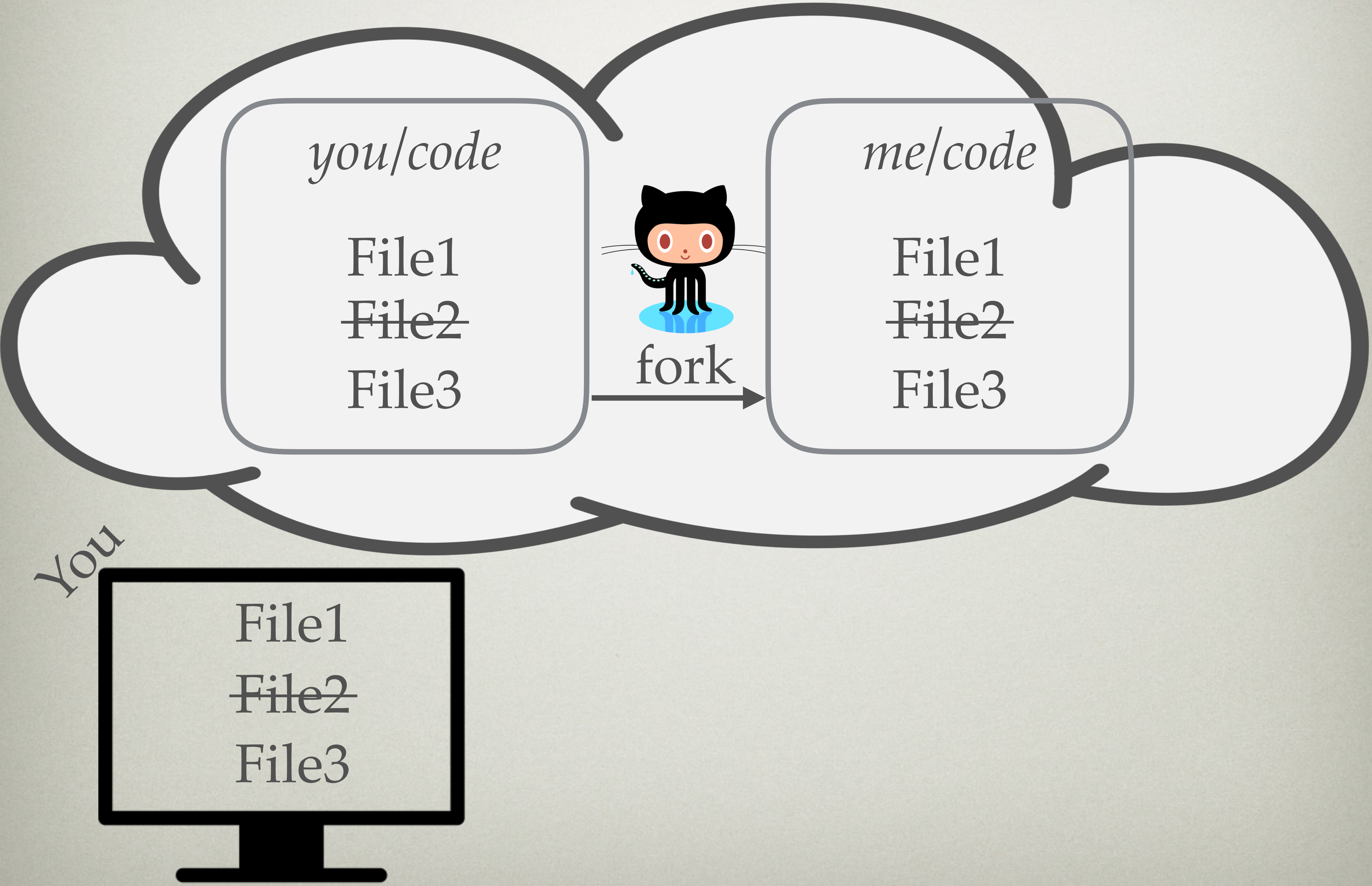
Lets dig down...



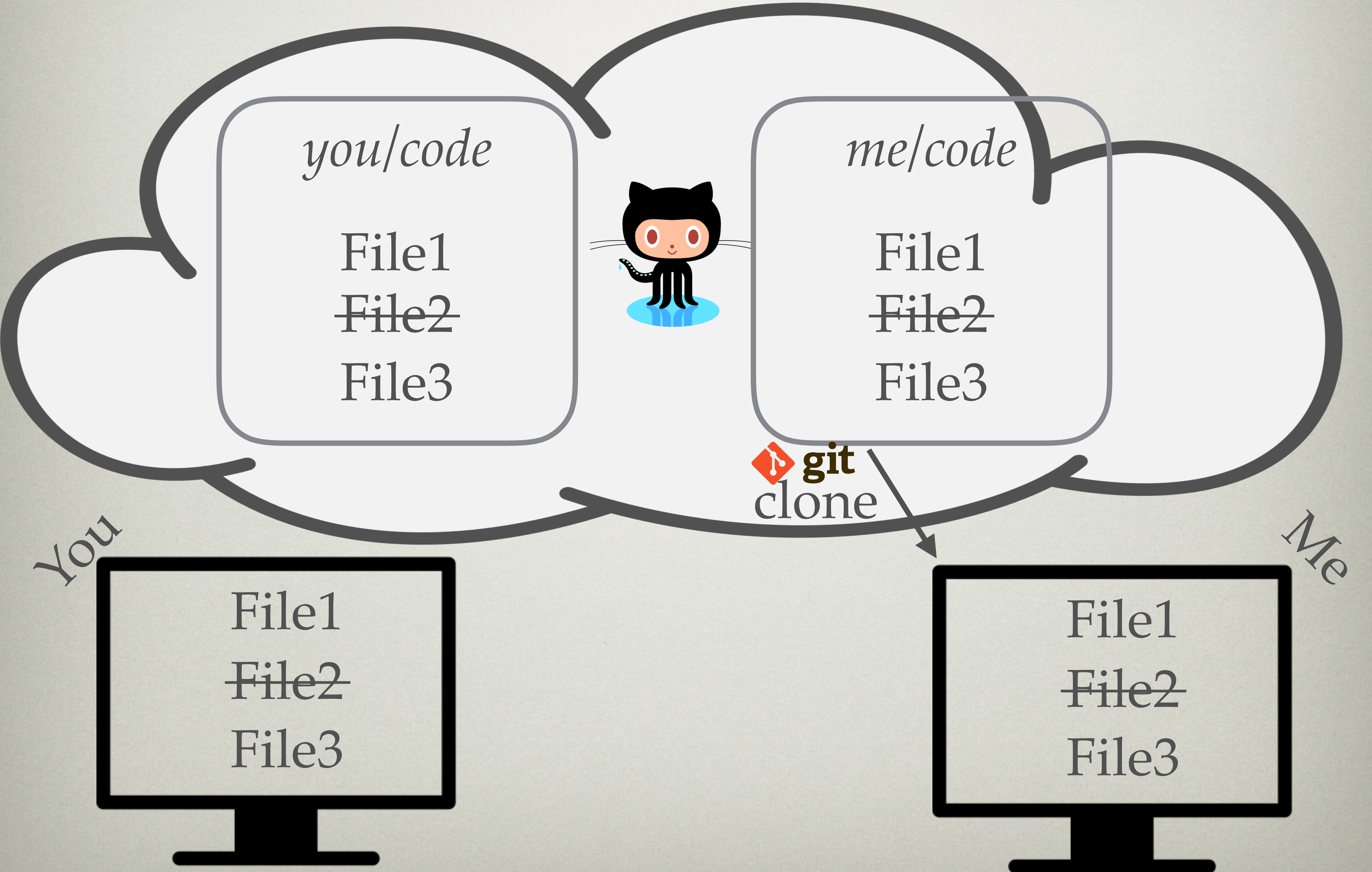
Lets dig down...



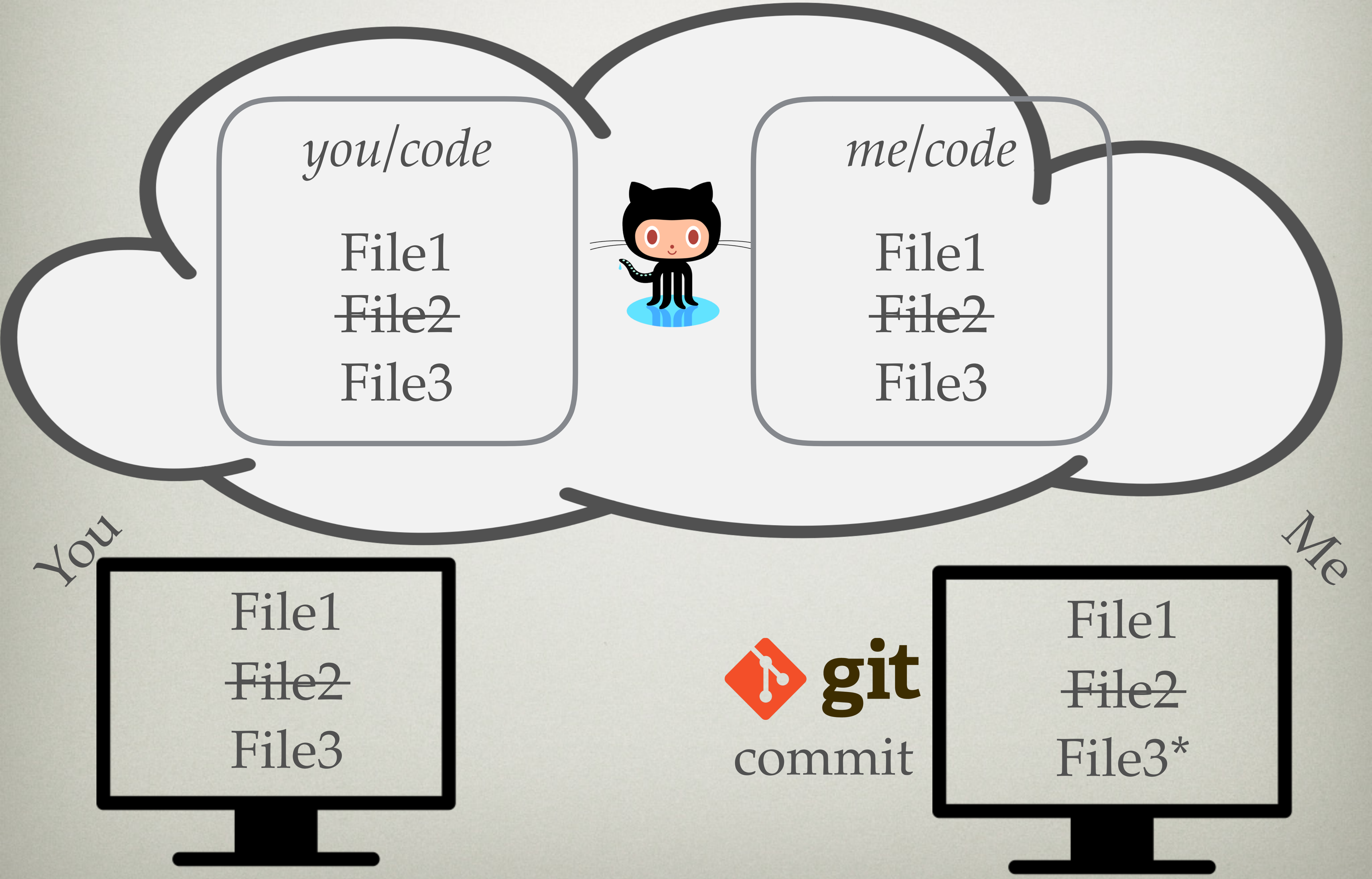
Lets dig down...



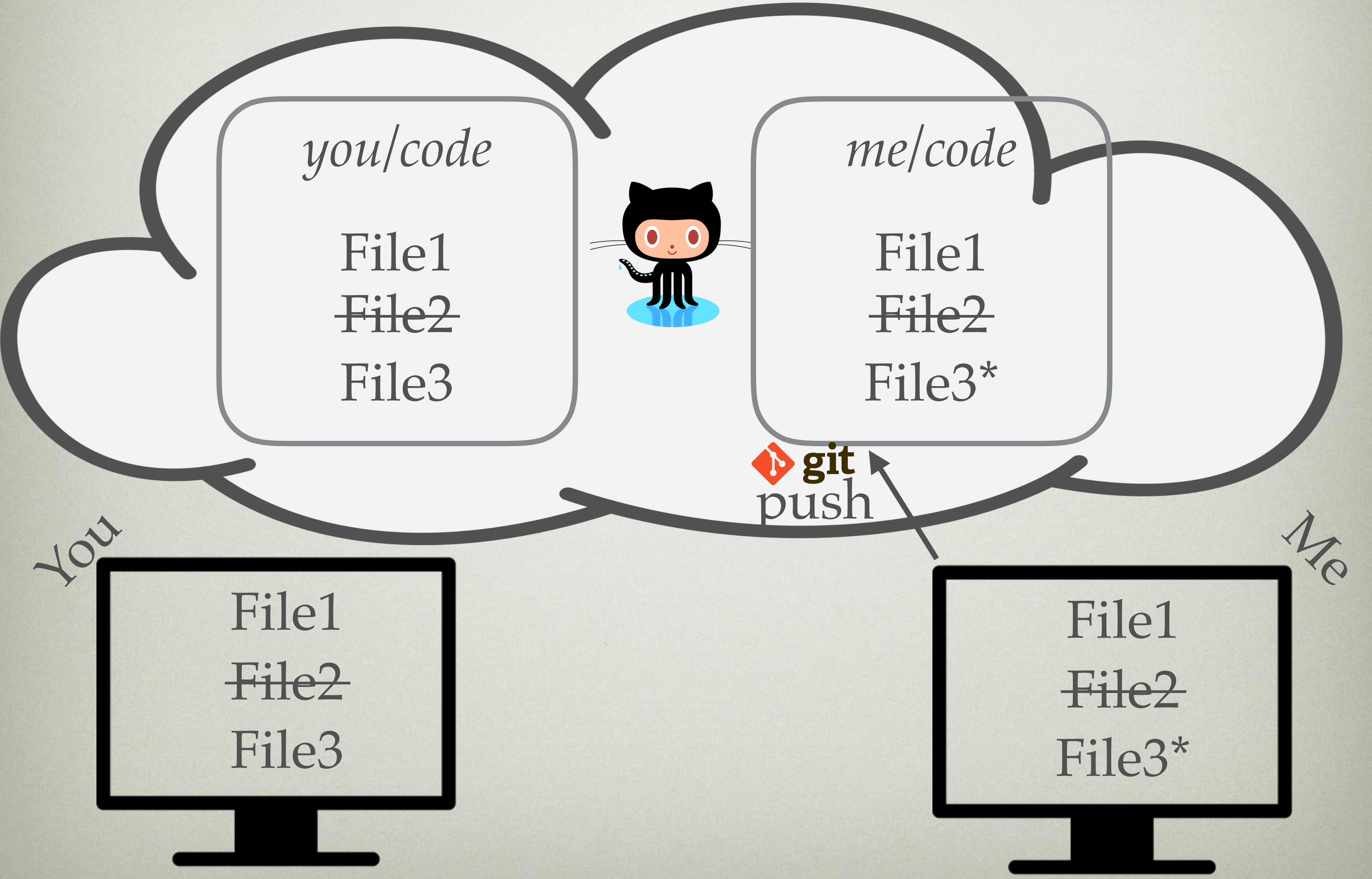
Lets dig down...



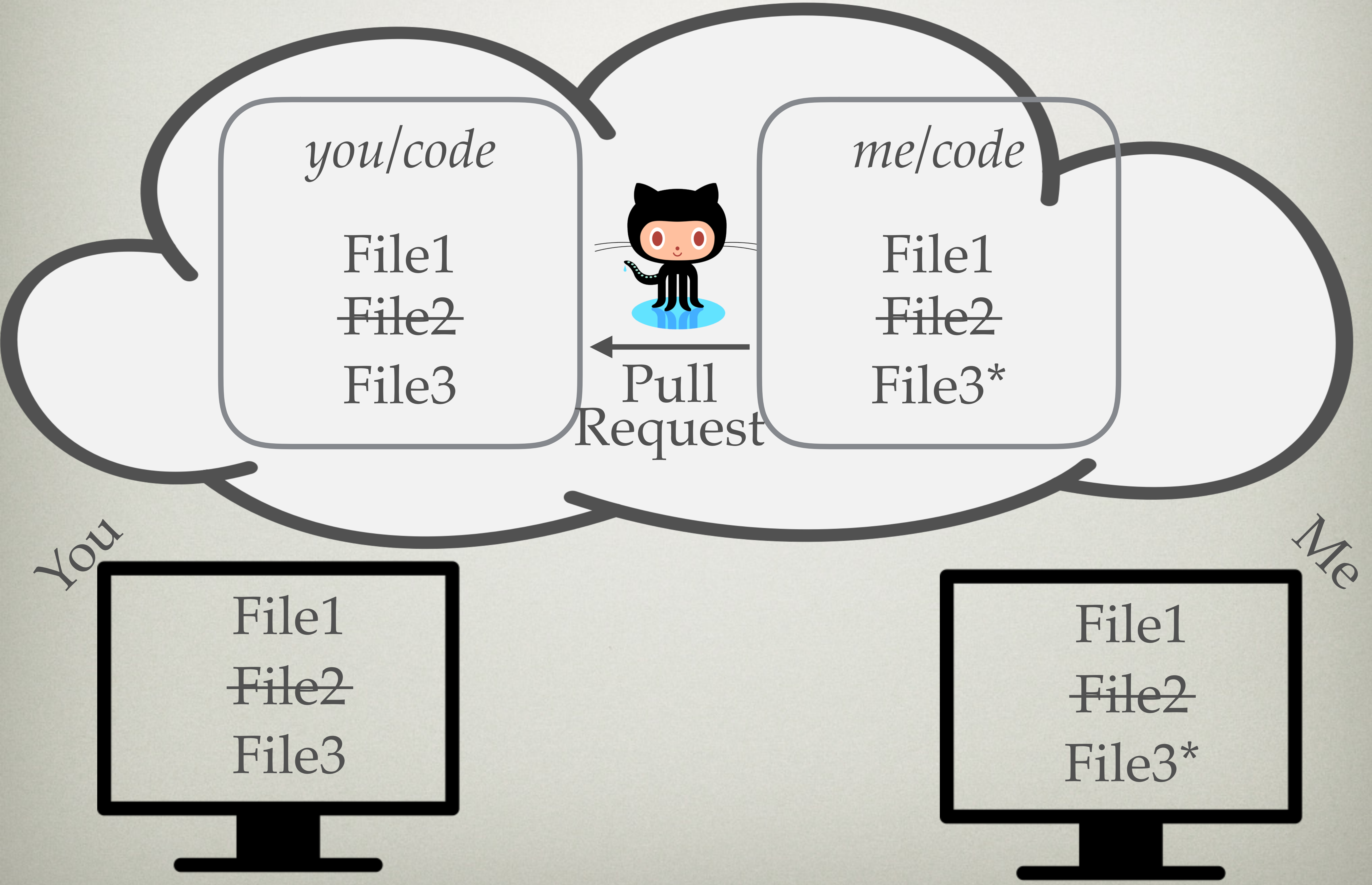
Lets dig down...



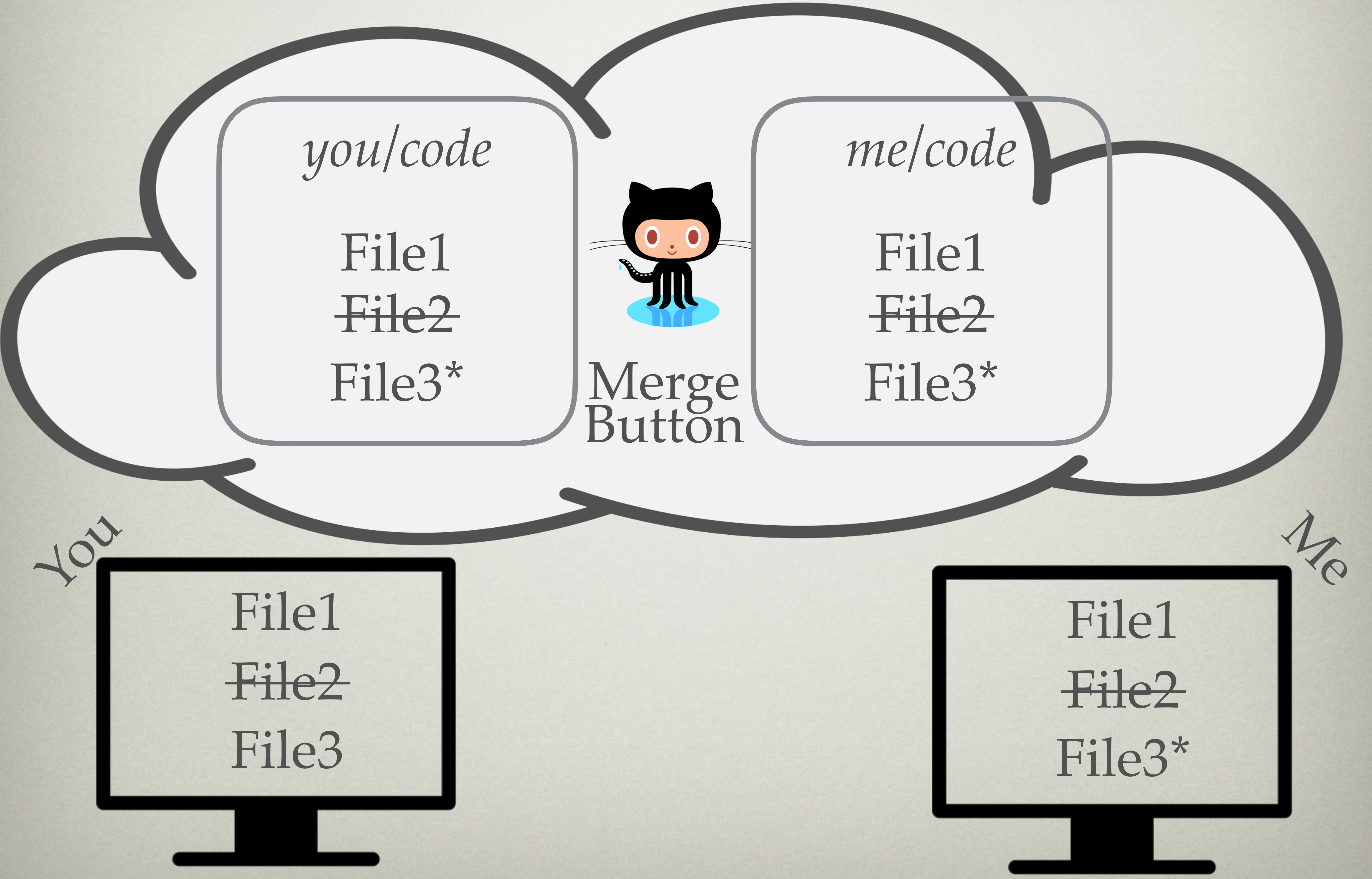
Lets dig down...



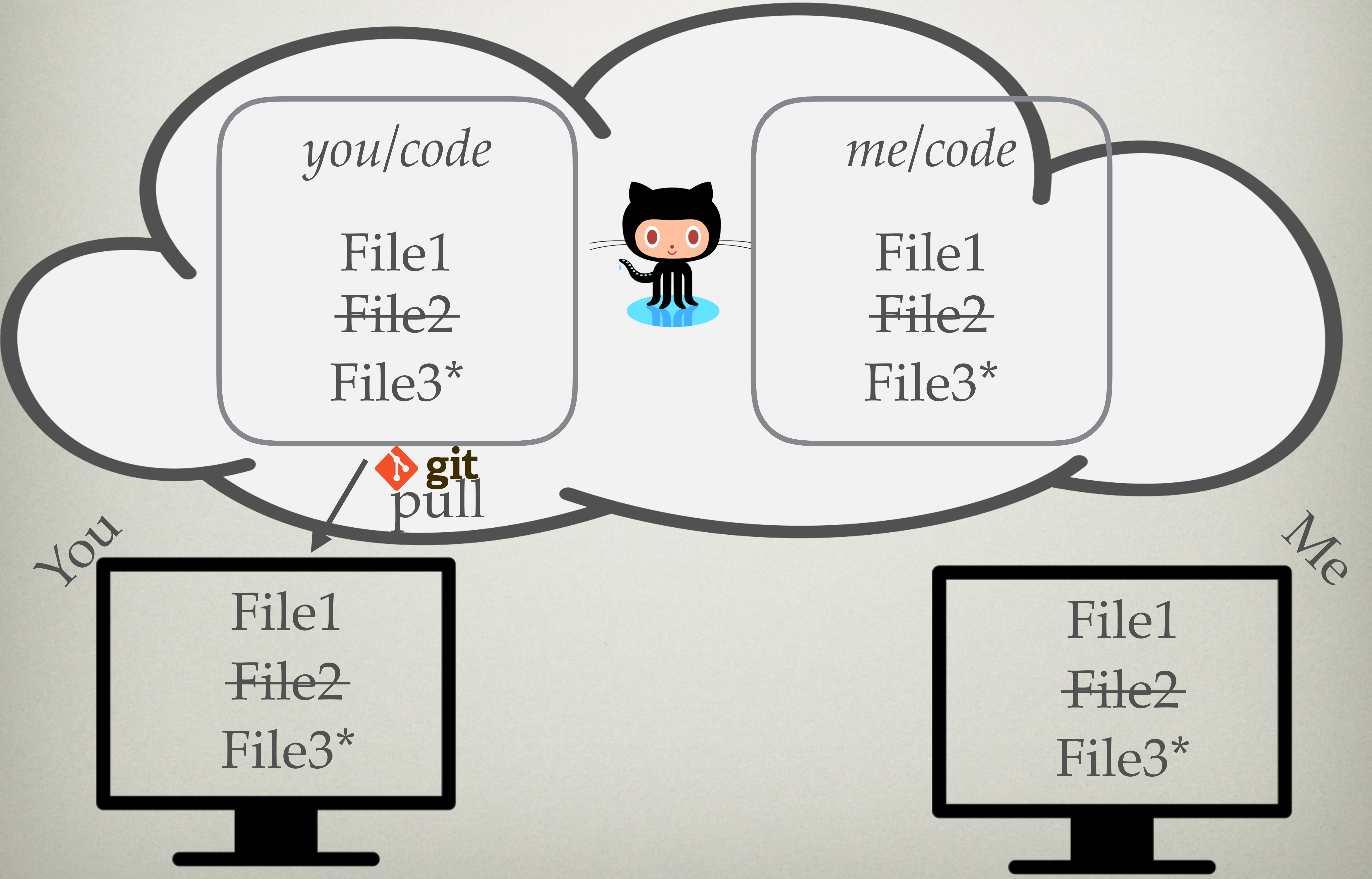
Lets dig down...



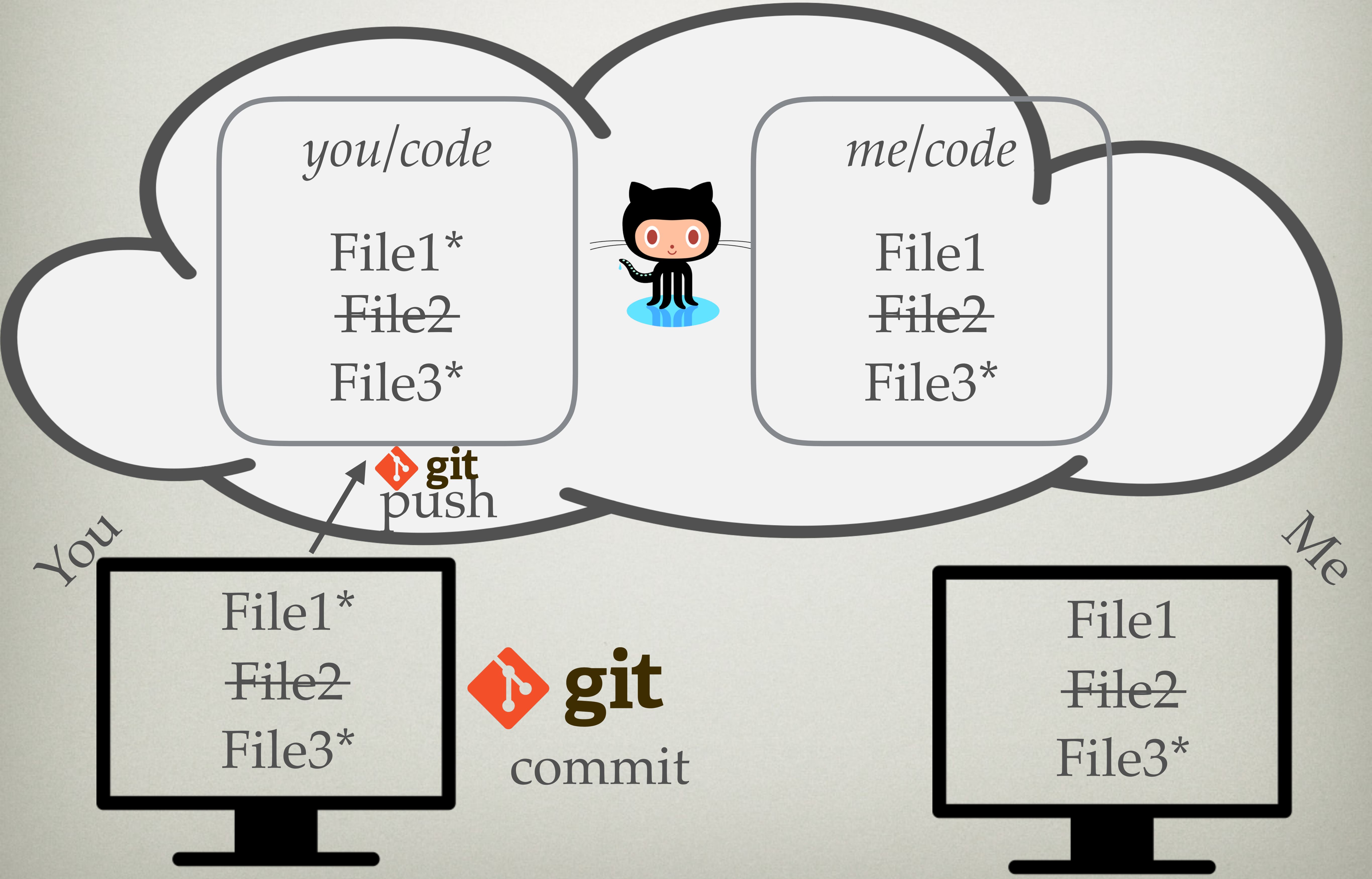
Lets dig down...



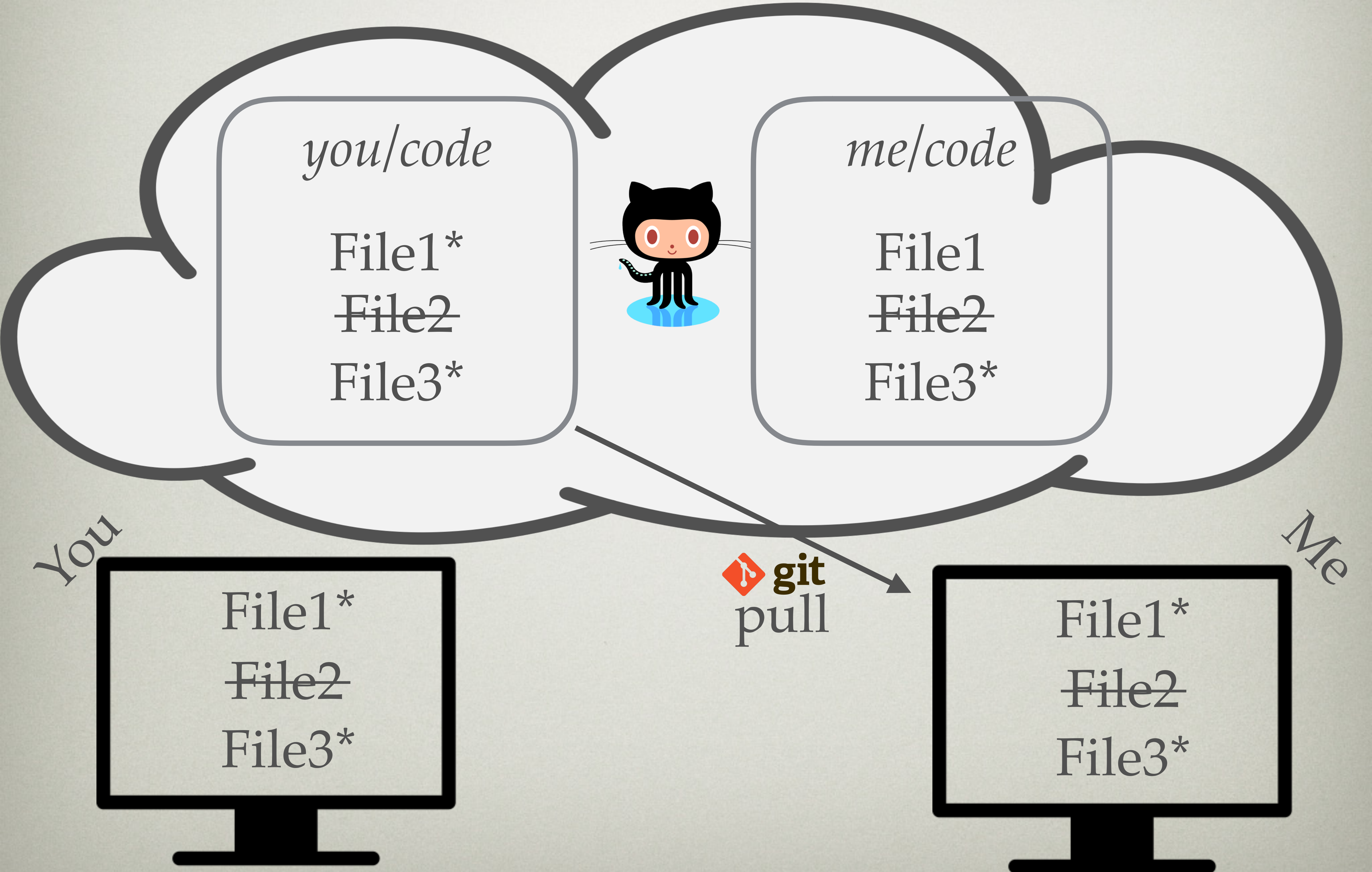
Lets dig down...



Lets dig down...



Lets dig down...


you/code
File1*
~~File2~~
File3*
me/code
File1
~~File2~~
File3*
You
File1*
~~File2~~
File3*
git
pull
Me
File1*
~~File2~~
File3*