

# OPEN SOURCE SOFTWARE: SHARING USING SOURCE CODE REPOSITORIES

Erik Tollerud

@eteq

Brigitta Sipocz

@AstroBrigi / bsipocz



# **WHAT IS OPEN SOURCE SOFTWARE(FOR OUR PURPOSES)?**

Software where the source code is open  
and available to the public.



# WHAT IS OPEN SOURCE SOFTWARE(FOR OUR PURPOSES)?

Software where the source code is open  
and available to the public.

AND

Software that follows practices to be useful to  
others *and* can be contributed to by others  
(this generally requires a certain amount of  
“software engineering”)



# WHY IS OPEN SOURCE SOFTWARE USEFUL (IN ASTRO)?



# WHY IS OPEN SOURCE SOFTWARE USEFUL (IN ASTRO)?

- For you
  - Many eyes = credit, fewer bugs
  - Future you is your most-likely user



# WHY IS OPEN SOURCE SOFTWARE USEFUL (IN ASTRO)?

- For you
  - Many eyes = credit, fewer bugs
  - Future you is your most-likely user
- For the community
  - Lets people share effort within Astro
  - Lets Astro share effort with other fields



# WHY IS OPEN SOURCE SOFTWARE USEFUL (IN ASTRO)?

- For you
  - Many eyes = credit, fewer bugs
  - Future you is your most-likely user
- For the community
  - Lets people share effort within Astro
  - Lets Astro share effort with other fields
- For Science
  - It makes your work Reproducible!



# **WHAT IS A CODE REPOSITORY?**



# WHAT IS A CODE REPOSITORY?

- A place to store your code
- Possibly on your computer, possibly not... but definitely versioned



# 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.

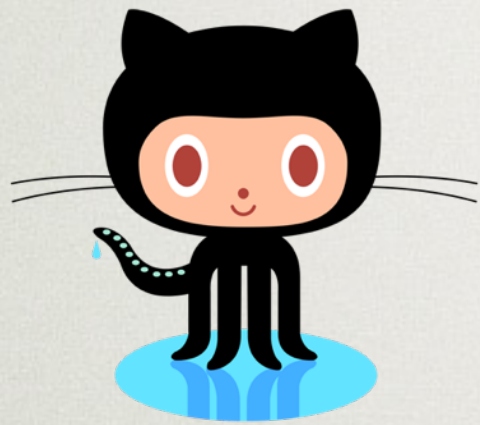


# WHAT IS A CODE REPOSITORY?

- 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.



# WHERE CAN YOU DO THIS?



**github**  
SOCIAL CODING



GitLab

Atlassian  
**Bitbucket**

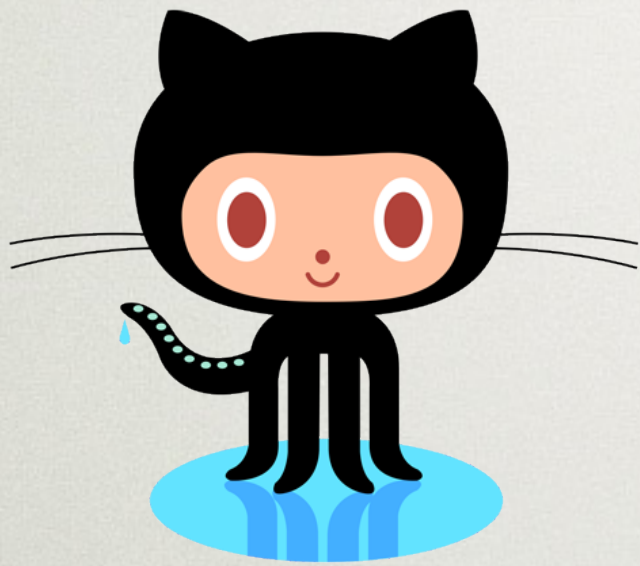
**SOURCE**  
**forge**



**PERFORCE**



# WHERE CAN YOU DO THIS?



**github**  
SOCIAL CODING



**git**



**AS A “CASE STUDY”...**






# WHAT IS THE PHILOSOPHY BEHIND THE CODE?

The Astropy Project is a community effort to develop a single core package for Astronomy in Python and foster interoperability between Python astronomy packages.



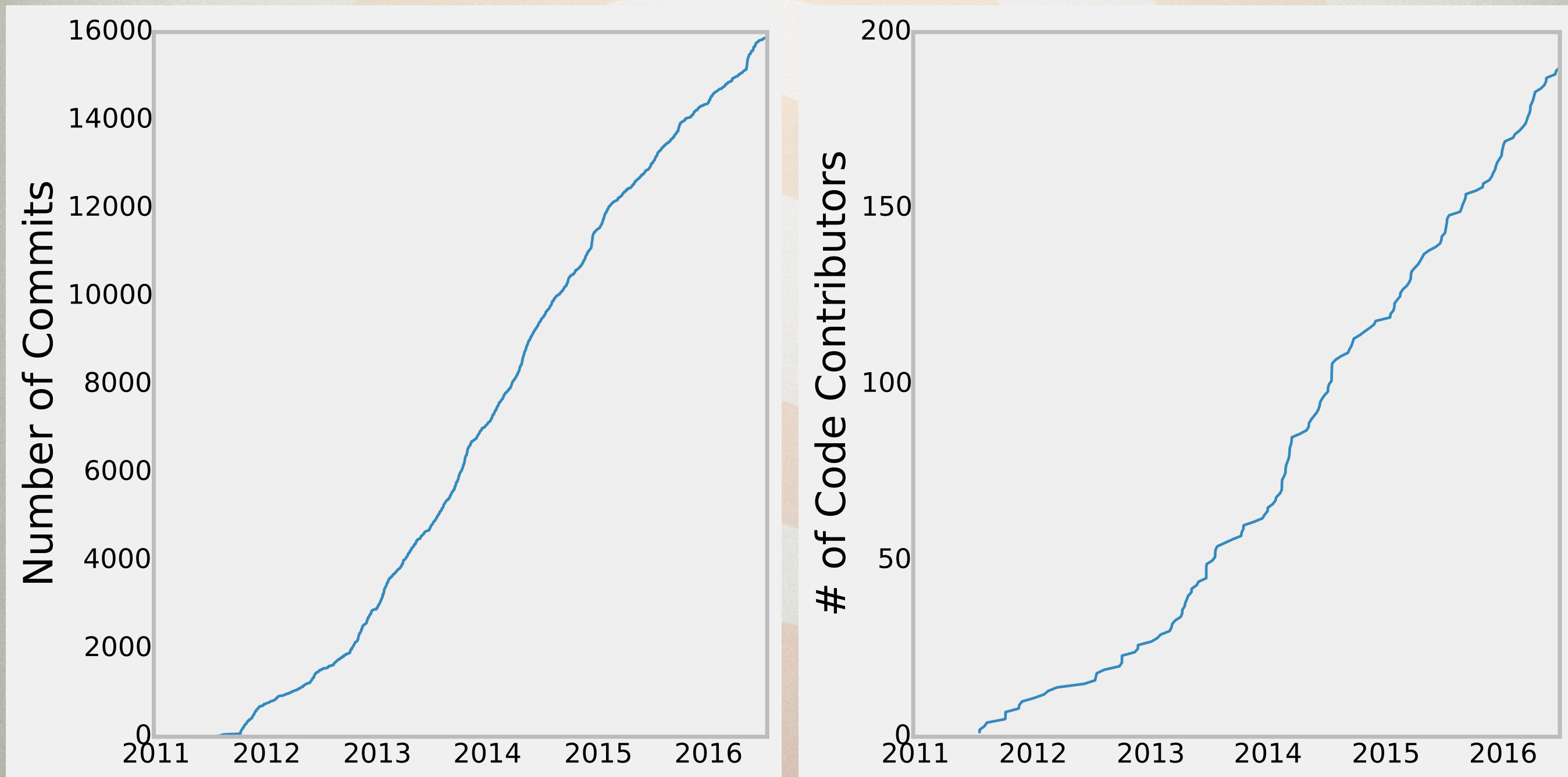
# WHAT IS THE PHILOSOPHY BEHIND THE CODE?

The Astropy Project is a community effort to develop a single core package for Astronomy in Python and foster interoperability between Python astronomy packages.

This means both *by*  (Professional) Astronomers help write it  
and *for* the  
community

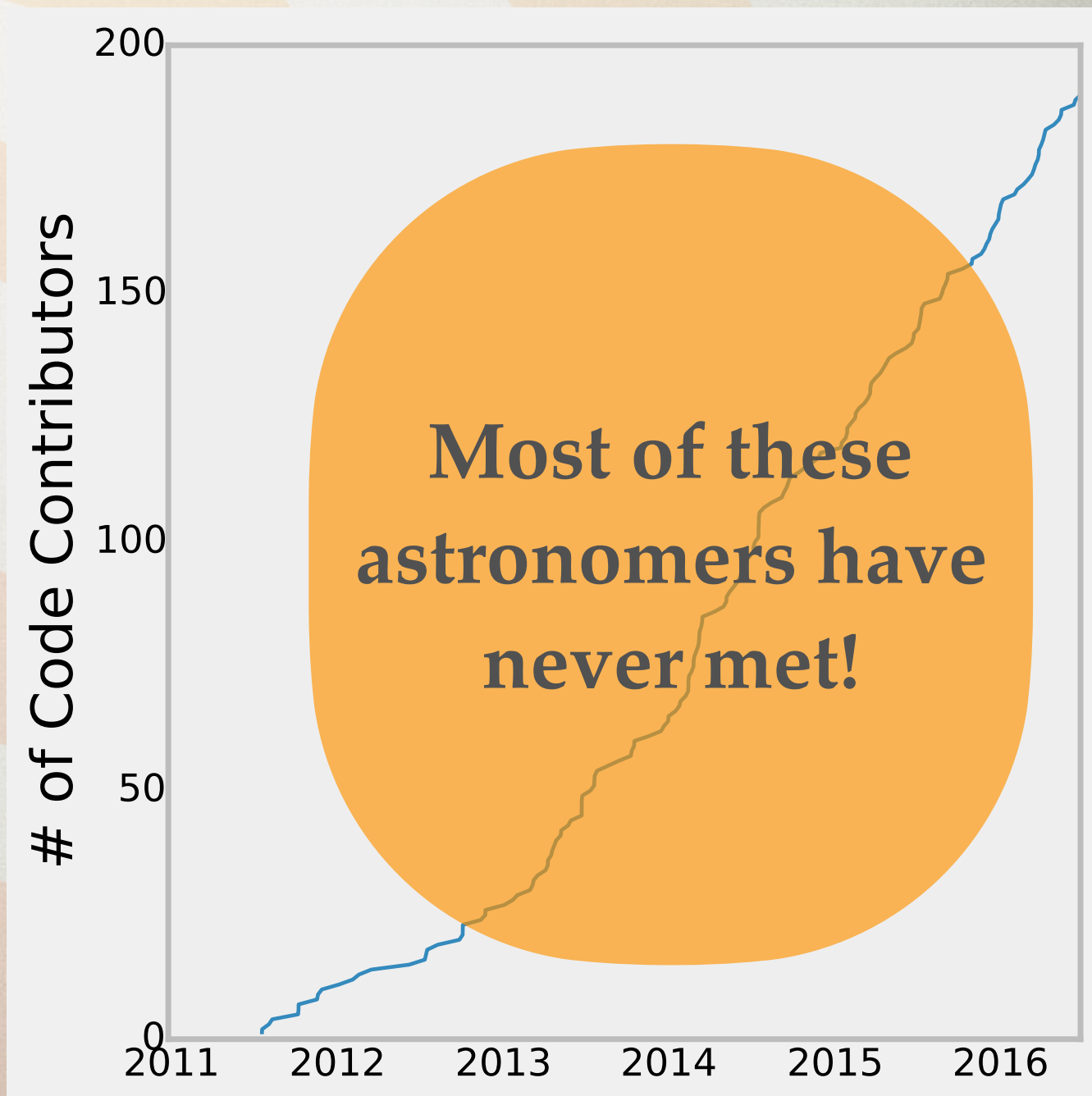
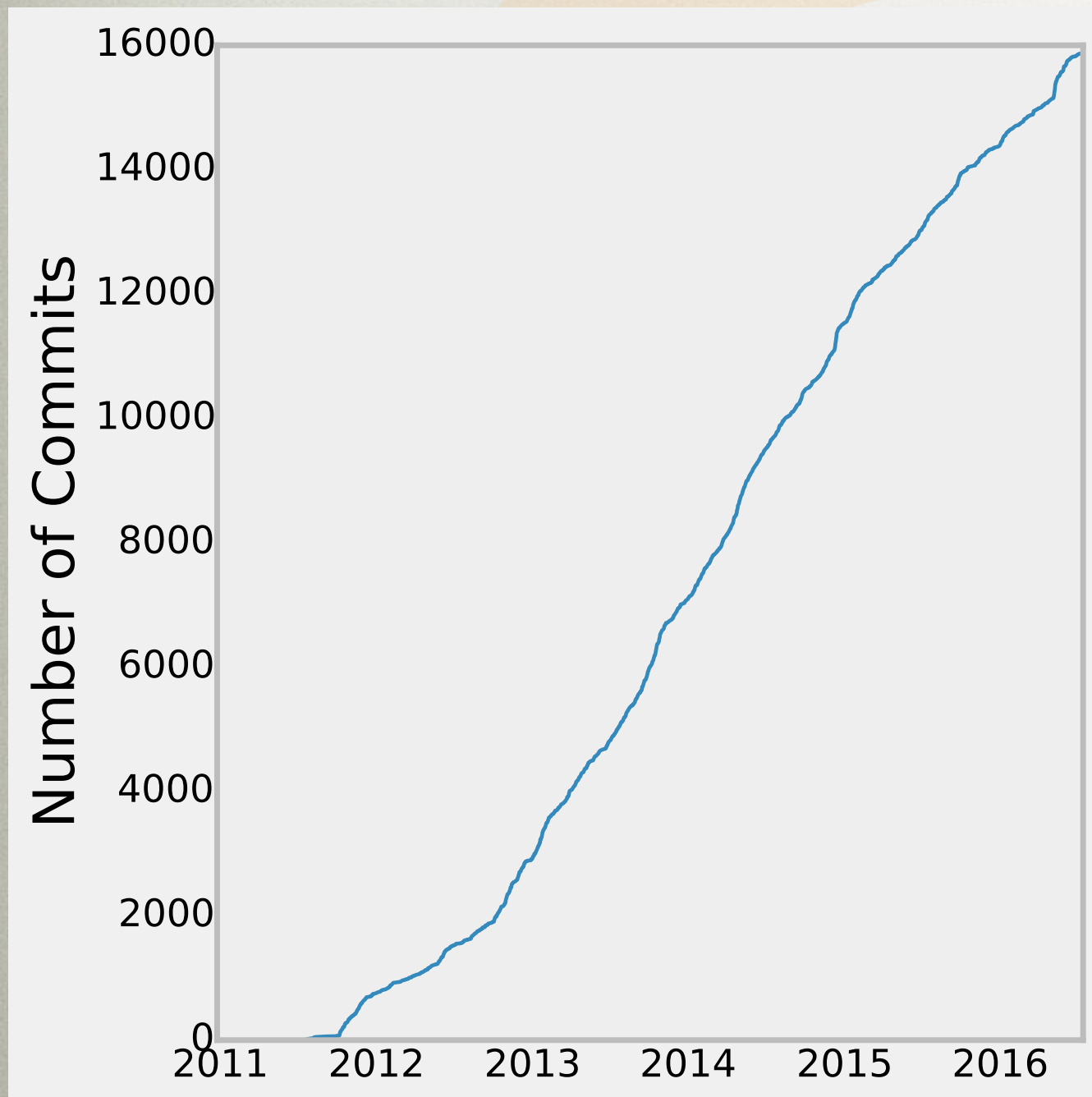


# CONTRIBUTIONS ARE GROWING





# CONTRIBUTIONS ARE GROWING





# KEYS TO DISTRIBUTED DEVELOPMENT










# KEYS TO DISTRIBUTED DEVELOPMENT: GITHUB


github  
SOCIAL CODING







 [astropy](#) / [astropy](#)

 Unwatch  86  Unstar 530  Fork 375

## Issue 3601 -- astropy.stats.funcs.bootstrap now accepts bootfuncs with multiple outputs #3628

 Open **ezbc** wants to merge 7 commits into `astropy:master` from `ezbc:issue3601`


 Conversation 8  Commits 7  Files changed 3 +115 -6














**ezbc** commented 26 days ago


I have addressed [Issue 3601](#). I implemented the option for users to supply a function with multiple outputs to `bootfunc`. They can control which `bootfunc` outputs to retain with `output_index`.

This function could be sped up if the `if` statements were moved outside of the loop.




**ezbc** added some commits 26 days ago

-   initial commit 84afdce
-   fully functional, needs indices to be `output_index` b69447d
-   bootstrap: reworked indices variable to be more pythonic e7ac659
-   updated changes log b7c1515
-   updated changes log for issue3601  9e01308






**embray** added `stats` `Affects-release` labels 26 days ago



**embray** commented 26 days ago Collaborator

@ezbc Your PR so far has the changelog entry and some tests which look good, but is missing the actual change to the function. Is it intentionally not implemented yet?

  definite commit of `funcs.py`  594ddd0

### Labels

- `Affects-release`
- `stats`


### Milestone

v1.1.0

### Assignee


No one—assign yourself

### Notifications

 Unsubscribe

You're receiving notifications because you were mentioned.

### 5 participants



### Lock pull request



# KEYS TO DISTRIBUTED DEVELOPMENT: GITHUB

github  
SOCIAL CODING



**embray** commented 26 days ago Collaborator 5 participants

@ezbc Your PR so far has the changelog entry and some tests which look good, but is missing the actual change to the function. Is it intentionally not implemented yet?

**definite** commit of funcs.py 594ddd0

**ezbc** commented 26 days ago

@embray This was not intentional, sorry about that. I must have committed funcs to a different branch. I believe the changes are now pushed to the correct branch, issue3601, and are now ready to be merged.

**bsipocz** commented on an outdated diff 26 days ago Show outdated diff

**bsipocz** commented on the diff 26 days ago

astropy/stats/tests/test\_funcs.py View full changes

```
@@ -236,6 +238,60 @@ def test_bootstrap():
    236     238         bootresult = np.mean(funcs.bootstrap(bootarr, 10000, bootfunc=np.m
    237     239         assert_allclose(np.mean(bootarr), bootresult, atol=0.01)
    238     240
    241     +     # test a bootfunc with several output values
    242     +     # return just bootstrapping with one output from bootfunc
    243     +     with NumpyRNGContext(42):
    244     +         bootarr = np.array([[1, 2, 3, 4, 5, 6, 7, 8, 9, 0],
    245     +                             [4, 8, 8, 3, 6, 5, 2, 8, 6, 2]]).T
    246     +
    247     +         answer = np.array((0.19425, 0.02094))
    248     +
    249     +         bootresult = funcs.bootstrap(bootarr, 2,
    250     +                                     bootfunc=spearmanr)
```

**bsipocz** added a note 26 days ago Collaborator

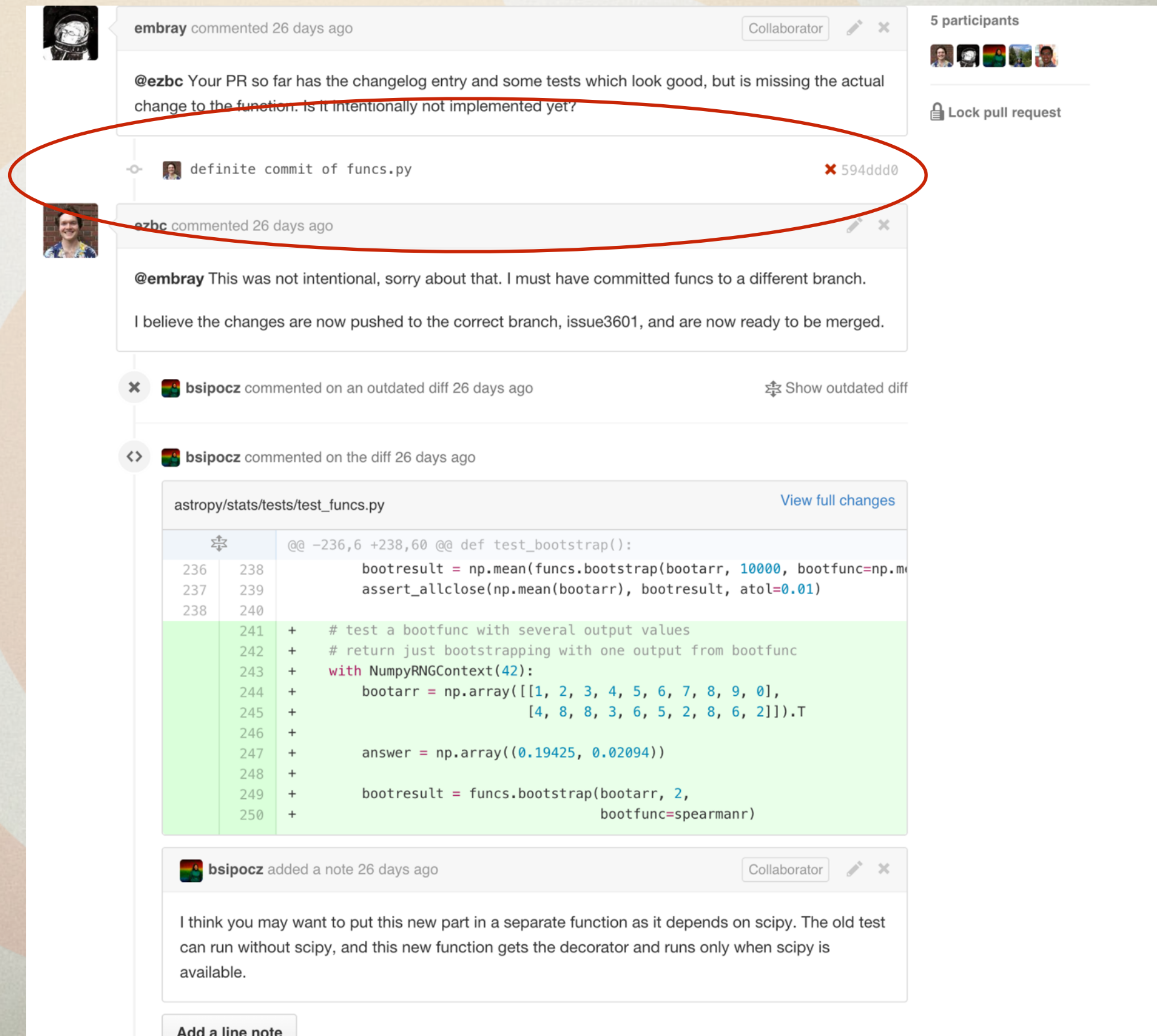
I think you may want to put this new part in a separate function as it depends on scipy. The old test can run without scipy, and this new function gets the decorator and runs only when scipy is available.

Add a line note



# KEYS TO DISTRIBUTED DEVELOPMENT: GITHUB

github  
SOCIAL CODING

A screenshot of a GitHub pull request discussion. The interface shows a pull request for 'definite commit of funcs.py' with a commit hash of '594ddd0'. The discussion includes comments from users 'embray', 'ezbc', and 'bsipocz'. A red circle highlights the commit hash and the comment from 'ezbc' which explains a branch mismatch. The bottom of the screenshot shows a code diff for 'astropy/stats/tests/test\_funcs.py' with a new test function 'test\_bootstrap' and a note from 'bsipocz' suggesting a separate function for scipy-dependent code.

embray commented 26 days ago Collaborator ✎ ✕

@ezbc Your PR so far has the changelog entry and some tests which look good, but is missing the actual change to the function. Is it intentionally not implemented yet?

definite commit of funcs.py ✖ 594ddd0

ezbc commented 26 days ago ✎ ✕

@embray This was not intentional, sorry about that. I must have committed funcs to a different branch. I believe the changes are now pushed to the correct branch, issue3601, and are now ready to be merged.

✕ 🇧🇷 bsipocz commented on an outdated diff 26 days ago ⚙ Show outdated diff

<> 🇧🇷 bsipocz commented on the diff 26 days ago

astropy/stats/tests/test\_funcs.py View full changes

@@ -236,6	+238,60 @@	def test_bootstrap():
236	238	bootresult = np.mean(funcs.bootstrap(bootarr, 10000, bootfunc=np.m
237	239	assert_allclose(np.mean(bootarr), bootresult, atol=0.01)
238	240	
	241	+ # test a bootfunc with several output values
	242	+ # return just bootstrapping with one output from bootfunc
	243	+ with NumpyRNGContext(42):
	244	+ bootarr = np.array([[1, 2, 3, 4, 5, 6, 7, 8, 9, 0],
	245	+ [4, 8, 8, 3, 6, 5, 2, 8, 6, 2]]).T
	246	+
	247	+ answer = np.array((0.19425, 0.02094))
	248	+
	249	+ bootresult = funcs.bootstrap(bootarr, 2,
	250	+ bootfunc=spearmanr)

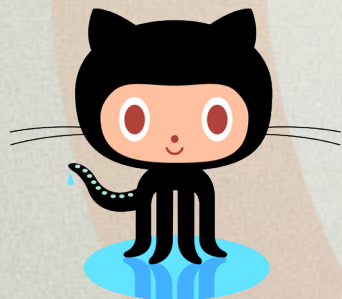
🇧🇷 bsipocz added a note 26 days ago Collaborator ✎ ✕


I think you may want to put this new part in a separate function as it depends on scipy. The old test can run without scipy, and this new function gets the decorator and runs only when scipy is available.

Add a line note




# KEYS TO DISTRIBUTED DEVELOPMENT: GITHUB






ttshimiz commented 4 days ago


@ezbc, @bsipocz, @eteq Yeah that would definitely be an acceptable solution to me. Examples and an explanation in the documentation would really help to show people how to use bootstrap with functions that have multiple inputs and outputs. It didn't even occur to me to use lambda to just define a new function that only returned the first output of spearmanr. In my view, simpler is always better. Thanks for addressing this issue!



✓ All is well — 3 successful checks [Show all checks](#)

**This pull request can be automatically merged.**  
You can also merge branches on the [command line](#).

 **Merge pull request**



Write Preview Markdown supported Edit in fullscreen


Leave a comment

Attach images by dragging & dropping, [selecting them](#), or pasting from the clipboard.

Close pull request Comment

💡 **ProTip!** Add `.patch` or `.diff` to the end of URLs for Git's plaintext views.

© 2015 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Contact](#)

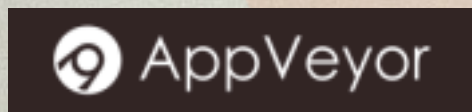


[Status](#) [API](#) [Training](#) [Shop](#) [Blog](#) [About](#)



# KEYS TO DISTRIBUTED DEVELOPMENT: CONTINUOUS TESTING

py.test



A screenshot of a GitHub pull request page. At the top, a comment from user 'ttshimiz' is visible, dated '4 days ago'. Below the comment, a green status bar indicates 'All is well — 3 successful checks' with a 'Show all checks' link. A message states 'This pull request can be automatically merged.' with a 'Merge pull request' button. Below this is a 'Write' tab for comments, with a 'Preview' tab and links for 'Markdown supported' and 'Edit in fullscreen'. A text area for 'Leave a comment' is present, along with instructions to 'Attach images by dragging &amp; dropping, selecting them, or pasting from the clipboard.' At the bottom of the comment section are 'Close pull request' and 'Comment' buttons. A 'ProTip!' at the bottom suggests adding '.patch' or '.diff' to the end of URLs for Git's plaintext views. The footer contains copyright information for GitHub, Inc. and links to Terms, Privacy, Security, and Contact, along with a GitHub logo and links to Status, API, Training, Shop, Blog, and About.



# KEYS TO DISTRIBUTED DEVELOPMENT: DOCS



## SPHINX

### PYTHON DOCUMENTATION GENERATOR



## Read the Docs

Create, host, and browse documentation.

```
63 class FLRW(Cosmology):
64     """ A class describing an isotropic and homogeneous
65     (Friedmann-Lemaitre-Robertson-Walker) cosmology.
66
67     This is an abstract base class -- you can't instantiate
68     examples of this class, but must work with one of its
69     subclasses such as `LambdaCDM` or `wCDM`.
70
71     Parameters
72     -----
73
74     H0 : float or scalar `~astropy.units.Quantity`
75         Hubble constant at z = 0. If a float, must be in [km/sec/Mpc]
76
77     Om0 : float
78         Omega matter: density of non-relativistic matter in units of the
79         critical density at z=0. Note that this does not include
80         massive neutrinos.
81
82     Ode0 : float
83         Omega dark energy: density of dark energy in units of the critical
84         density at z=0.
85
86     Tcmb0 : float or scalar `~astropy.units.Quantity`
87         Temperature of the CMB z=0. If a float, must be in [K]. Default: 2.725.
88         Setting this to zero will turn off both photons and neutrinos (even
89         massive ones)
90
91     Neff : float
92         Effective number of Neutrino species. Default 3.04.
93
94     m_nu : `~astropy.units.Quantity`
95         Mass of each neutrino species. If this is a scalar Quantity, then all
96         neutrino species are assumed to have that mass. Otherwise, the mass of
97         each species. The actual number of neutrino species (and hence the
98         number of elements of m_nu if it is not scalar) must be the floor of
99         Neff. Usually this means you must provide three neutrino masses unless
100         you are considering something like a sterile neutrino.
101
102     name : str
103         Optional name for this cosmological object.
104
105     Ob0 : float
106         Omega baryons: density of baryonic matter in units of the critical
107         density at z=0.
108
109     Notes
110     -----
111     Class instances are static -- you can't change the values
112     of the parameters. That is, all of the attributes above are
113     read only.
114
115     def __init__(self, H0, Om0, Ode0, Tcmb0=2.725, Neff=3.04,
116                  m_nu=u.Quantity(0.0, u.eV), name=None, Ob0=None):
117
118         # all densities are in units of the critical density
119         self._Om0 = float(Om0)
120         if self._Om0 < 0.0:
121             raise ValueError("Matter density can not be negative")
122         self._Ode0 = float(Ode0)
123         if Ob0 is not None:
124             self._Ob0 = float(Ob0)
125             if self._Ob0 < 0.0:
```

```
class astropy.cosmology.FLRW(H0, Om0, Ode0, Tcmb0=2.725, Neff=3.04, m_nu=<Quantity 0.0 eV>,
name=None, Ob0=None) [edit on github][source]
```

Bases: `astropy.cosmology.core.Cosmology`

A class describing an isotropic and homogeneous (Friedmann-Lemaitre-Robertson-Walker) cosmology.

This is an abstract base class – you can't instantiate examples of this class, but must work with one of its subclasses such as `LambdaCDM` or `wCDM`.

Parameters: H0 : float or scalar `Quantity`

Hubble constant at z = 0. If a float, must be in [km/sec/Mpc]

Om0 : float

Omega matter: density of non-relativistic matter in units of the critical density at z=0.

Ode0 : float

Omega dark energy: density of dark energy in units of the critical density at z=0.

Tcmb0 : float or scalar `Quantity`

Temperature of the CMB z=0. If a float, must be in [K]. Default: 2.725. Setting this to zero will turn off both photons and neutrinos (even massive ones)

Neff : float

Effective number of Neutrino species. Default 3.04.

m\_nu : `Quantity`

Mass of each neutrino species. If this is a scalar Quantity, then all neutrino species are assumed to have that mass. Otherwise, the mass of each species. The actual number of neutrino species (and hence the number of elements of m\_nu if it is not scalar) must be the floor of Neff. Usually this means you must provide three neutrino masses unless you are considering something like a sterile neutrino.

name : str

Optional name for this cosmological object.

Ob0 : float

Omega baryons: density of baryonic matter in units of the critical density at z=0.

Notes

Class instances are static – you can't change the values of the parameters. That is, all of the attributes above are read only.

Attributes Summary

H0 Return the Hubble constant as an `Quantity` at z=0

v: stable



# KEYS TO DISTRIBUTED DEVELOPMENT IN A SHARED REPO

- *git* + Github for sharing
- Test \*everything\* (automatically)
- *Easy* documentation ( $\Rightarrow$  thorough)



**LETS DIG DOWN ON HOW  
YOU DO SHARED  
DEVELOPMENT WITH  
PUBLIC REPOS**



# LETS DIG DOWN...

You





# LETS DIG DOWN...

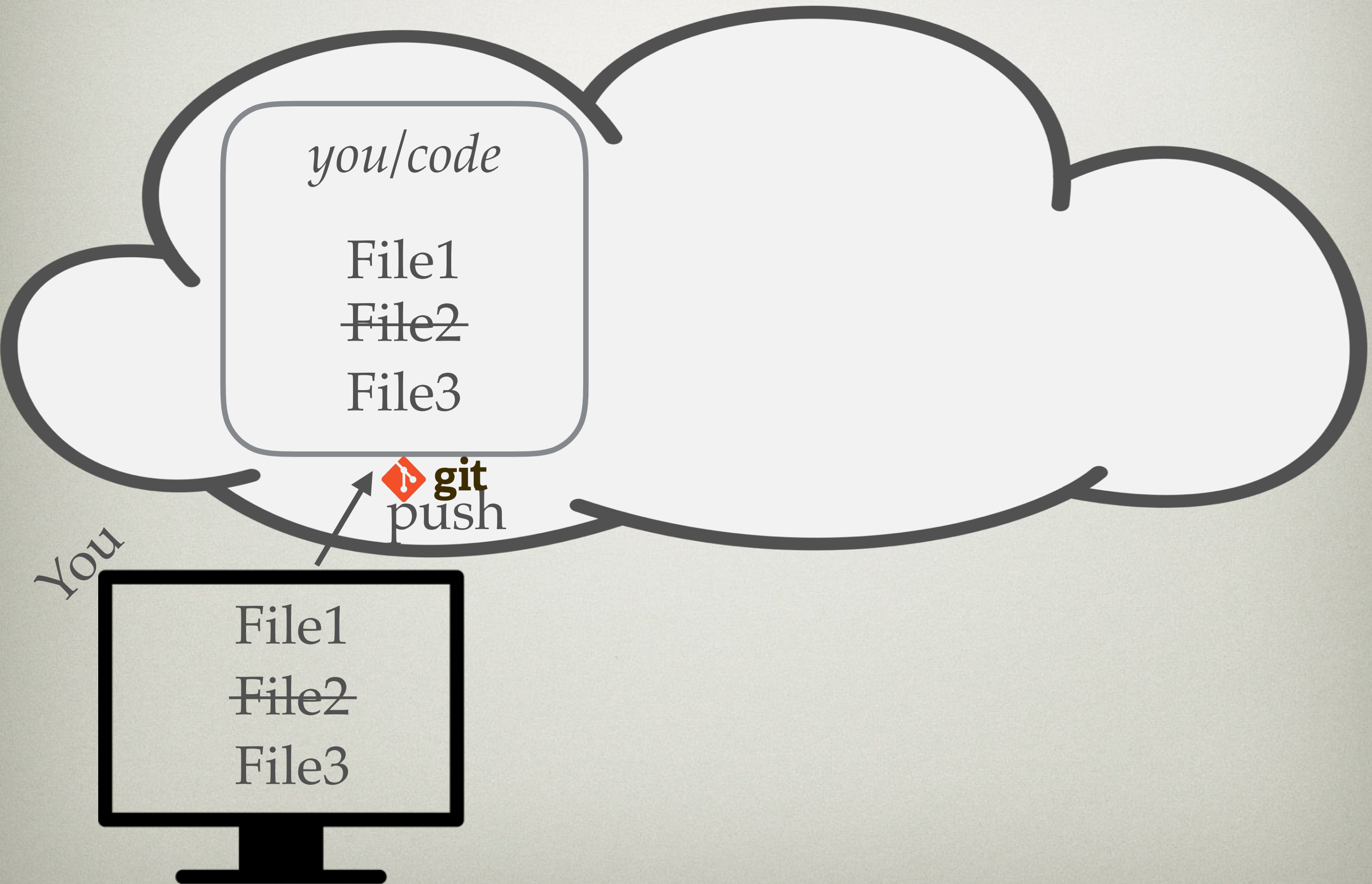
You



**git**  
commit

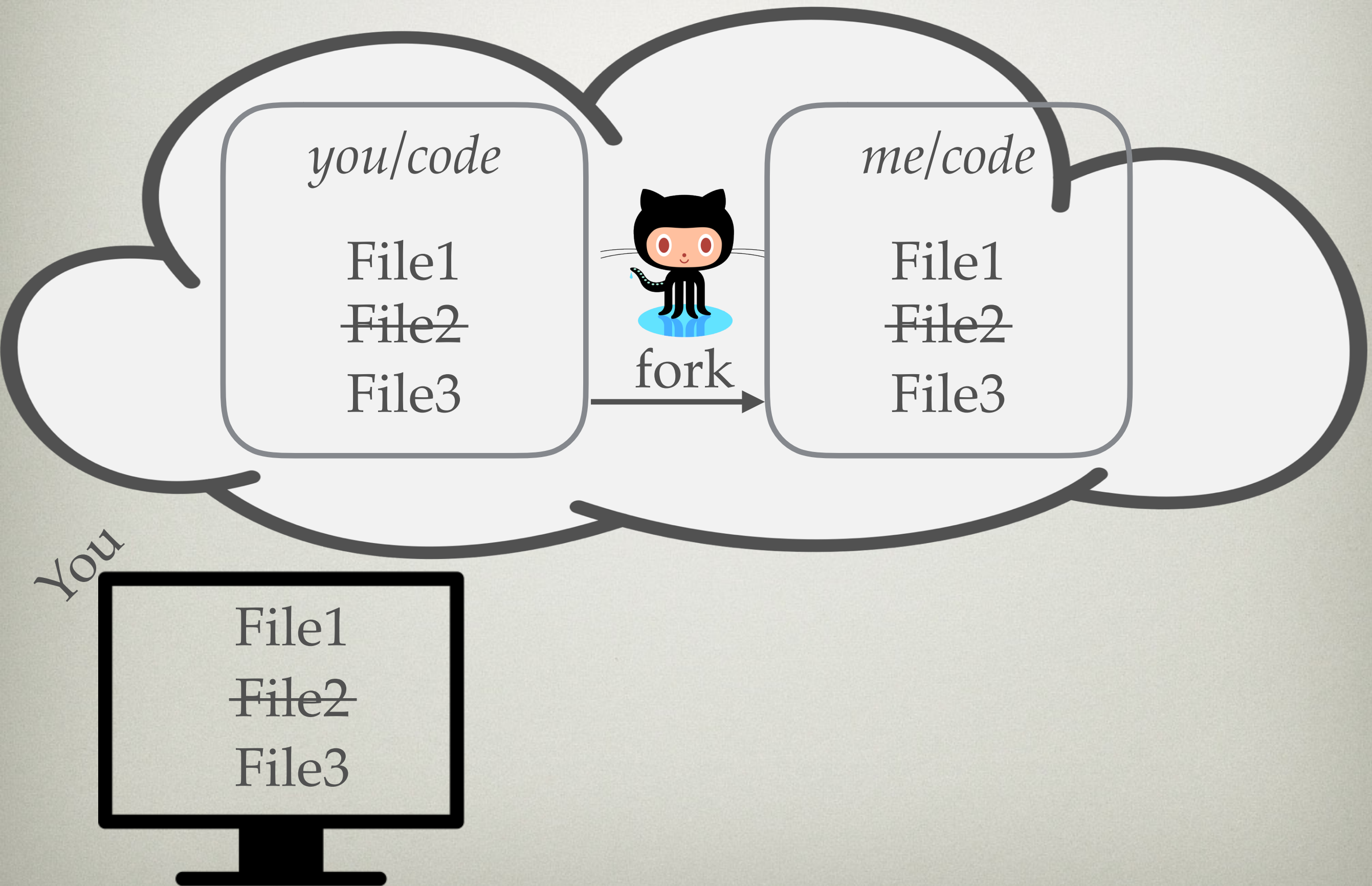


# LETS DIG DOWN...



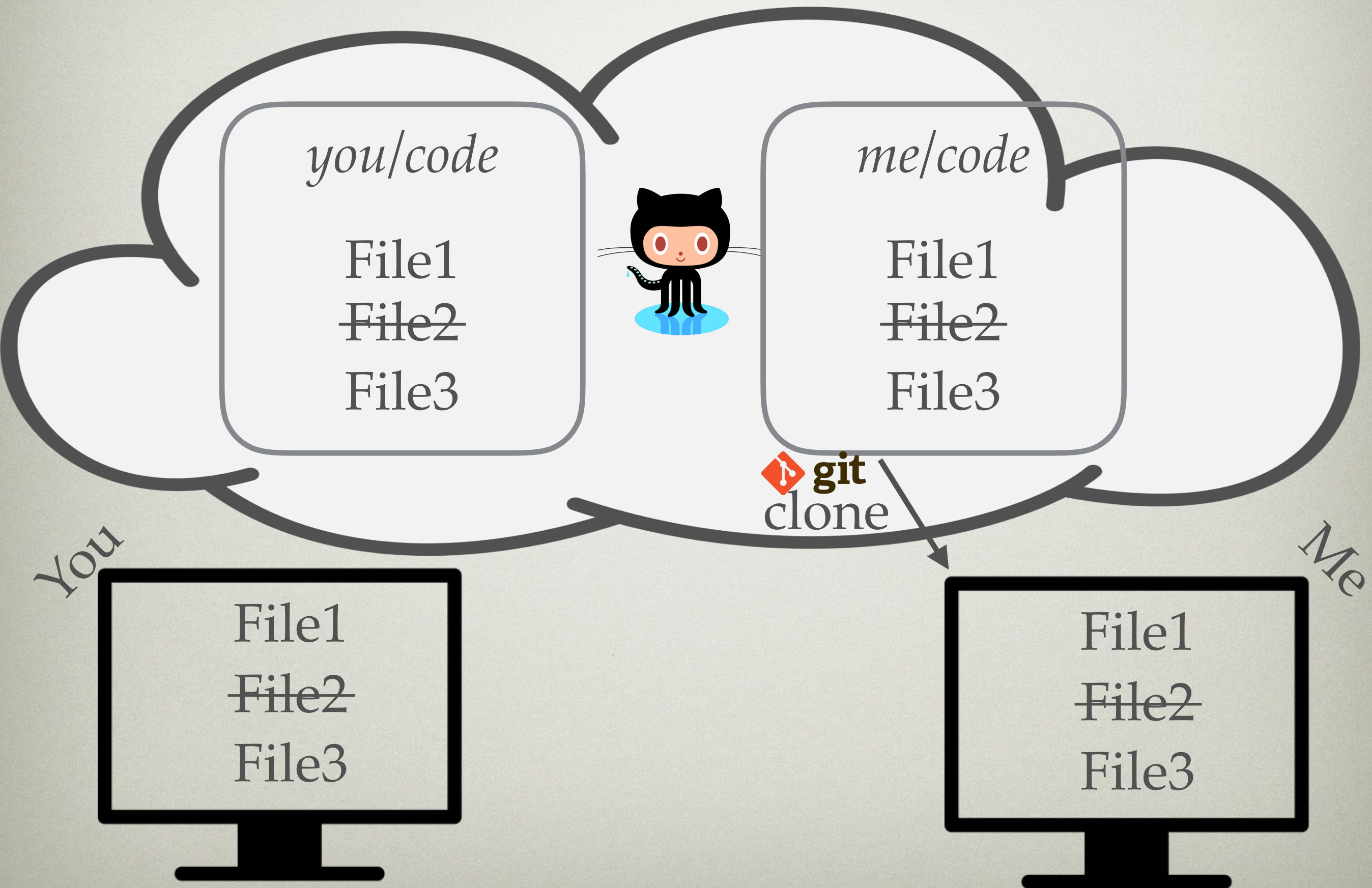


# LETS DIG DOWN...



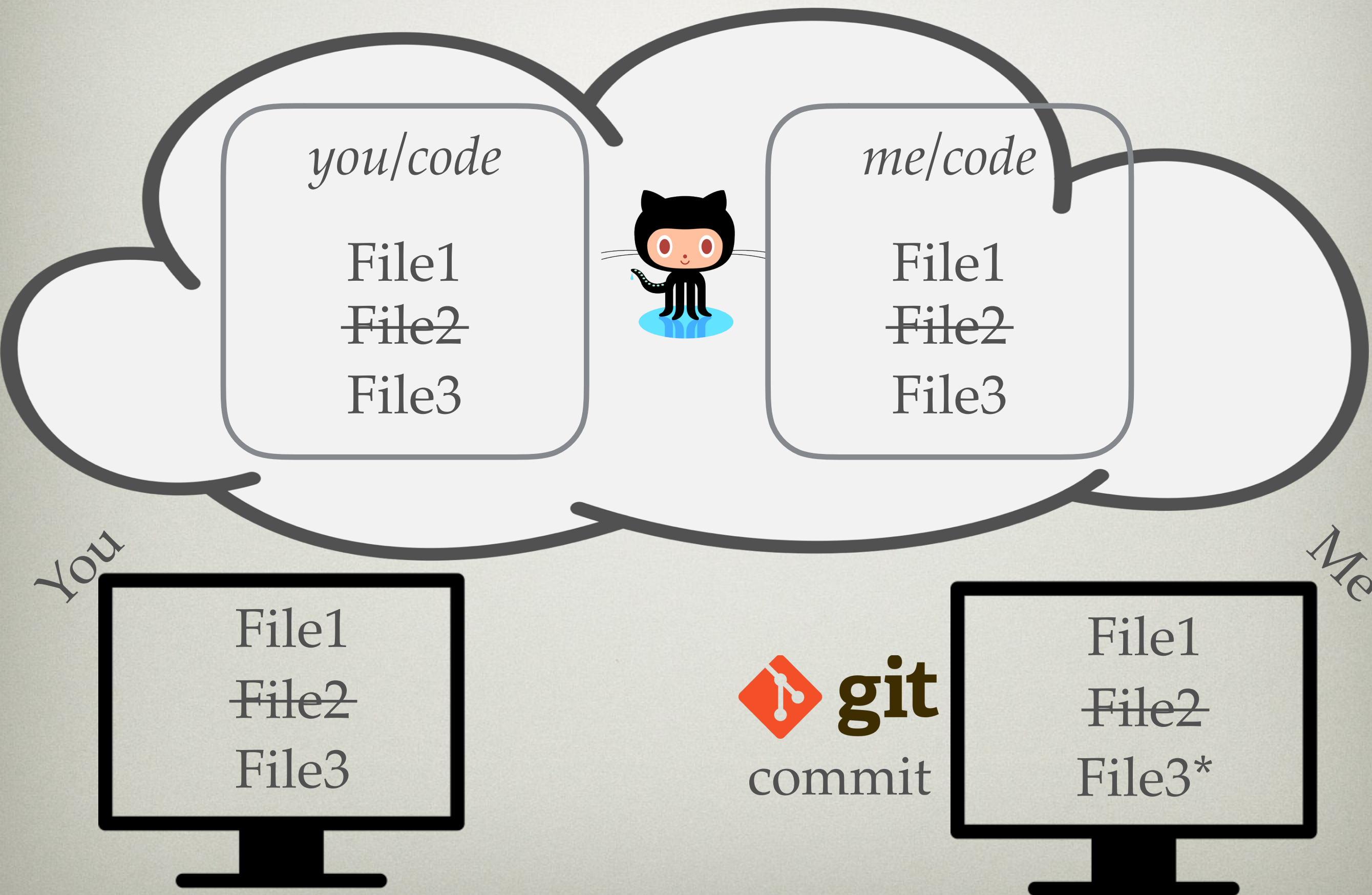


# LETS DIG DOWN...



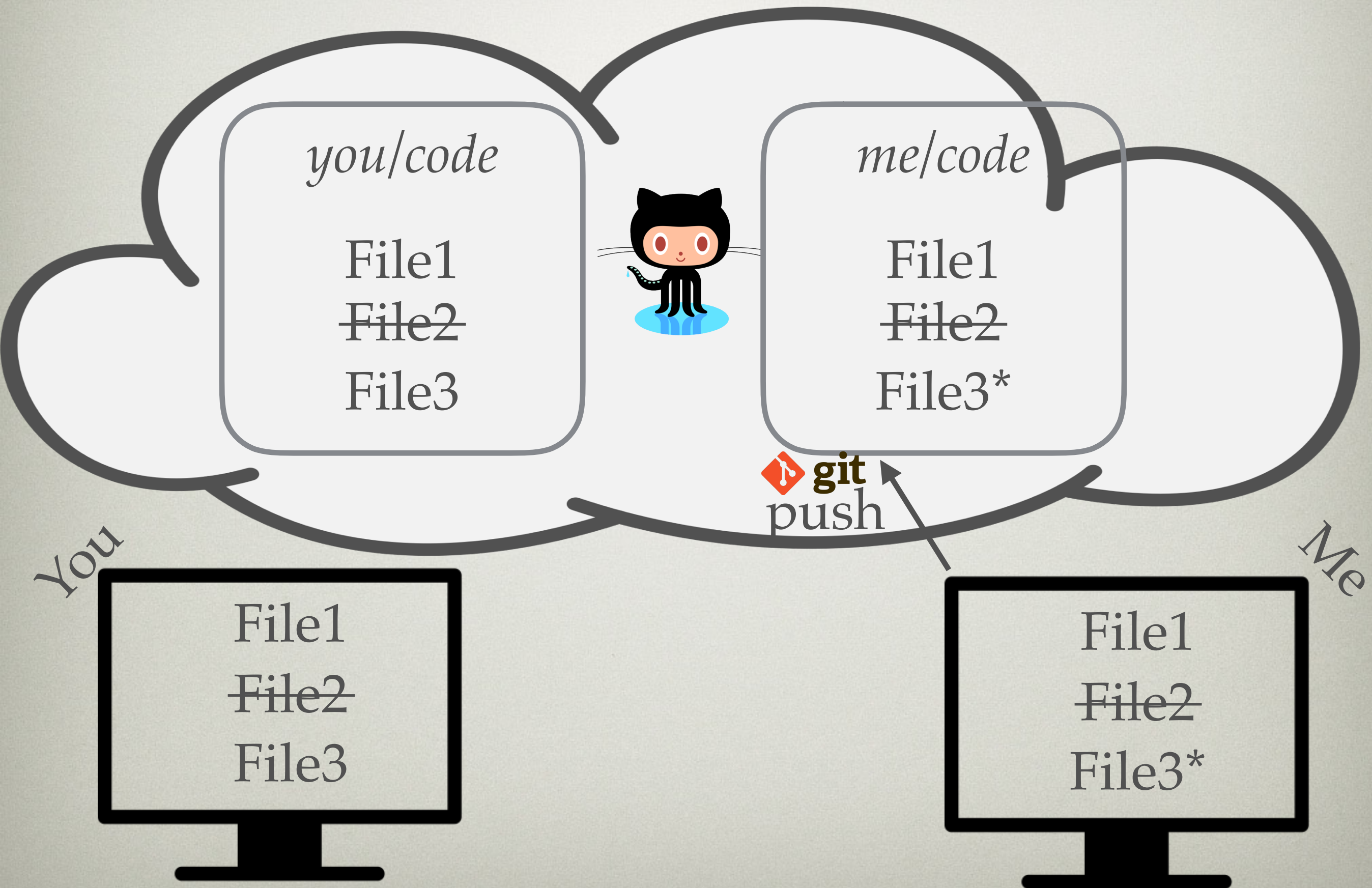


# LETS DIG DOWN...



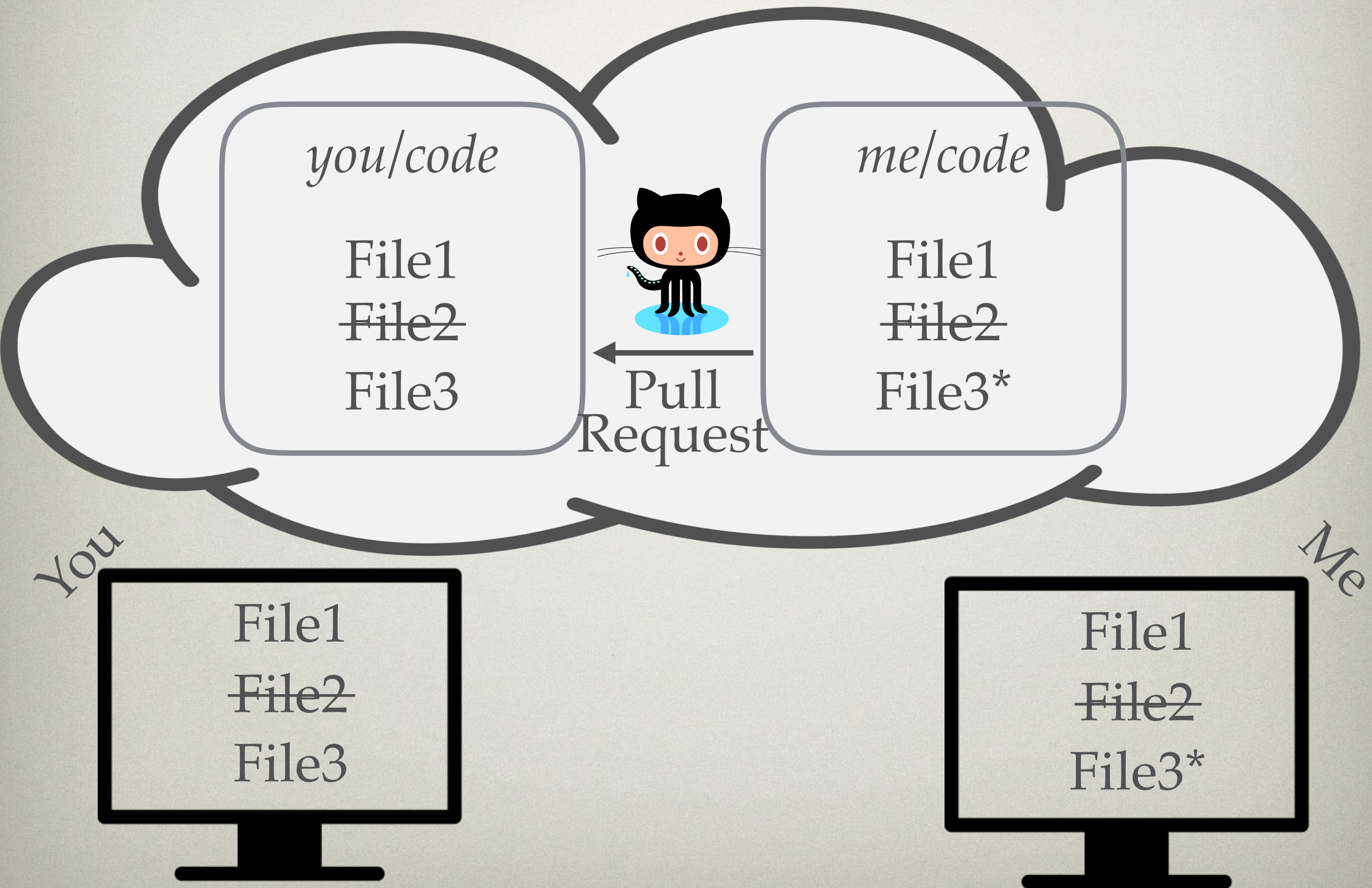


# LETS DIG DOWN...



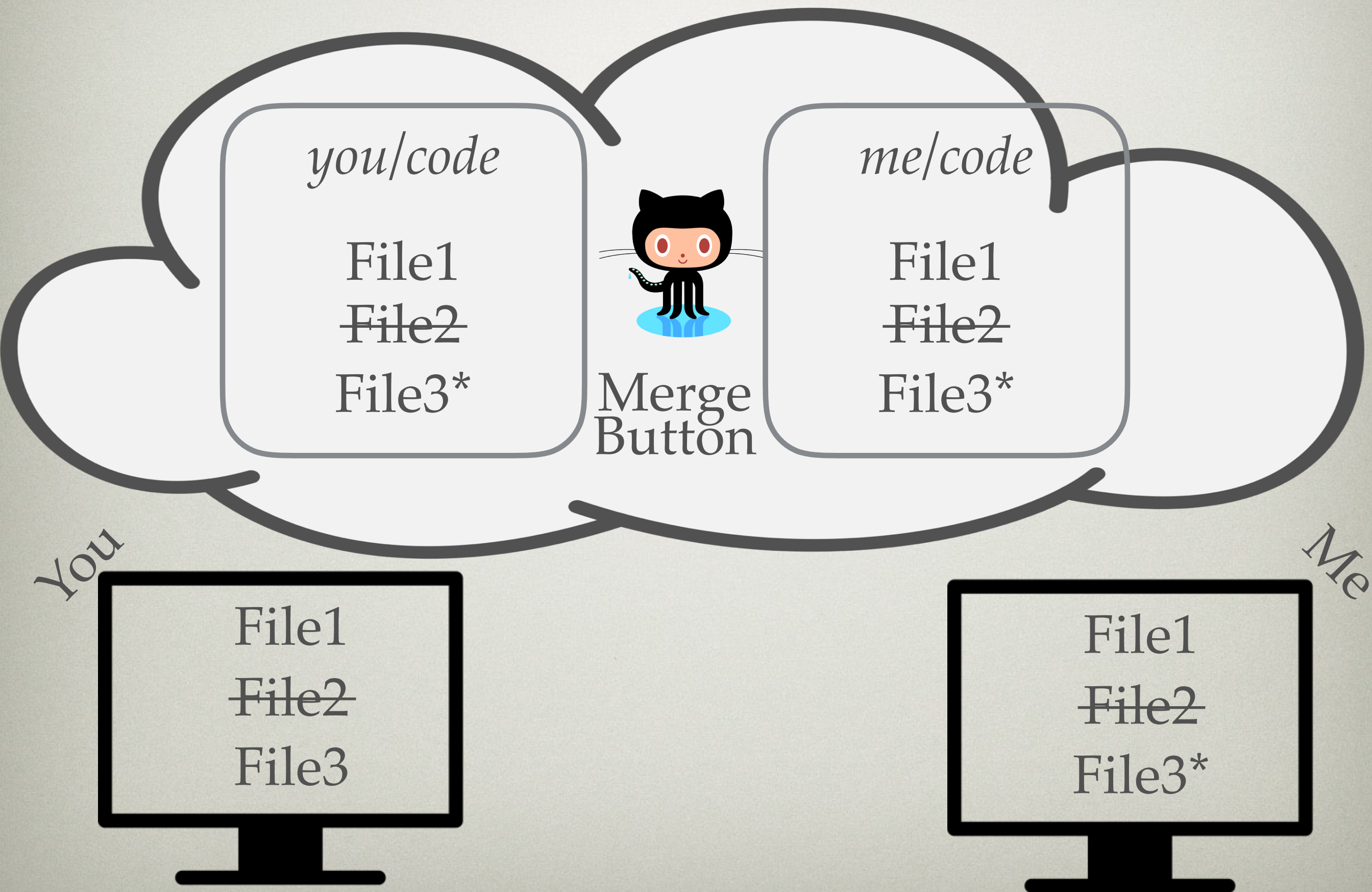


# LETS DIG DOWN...



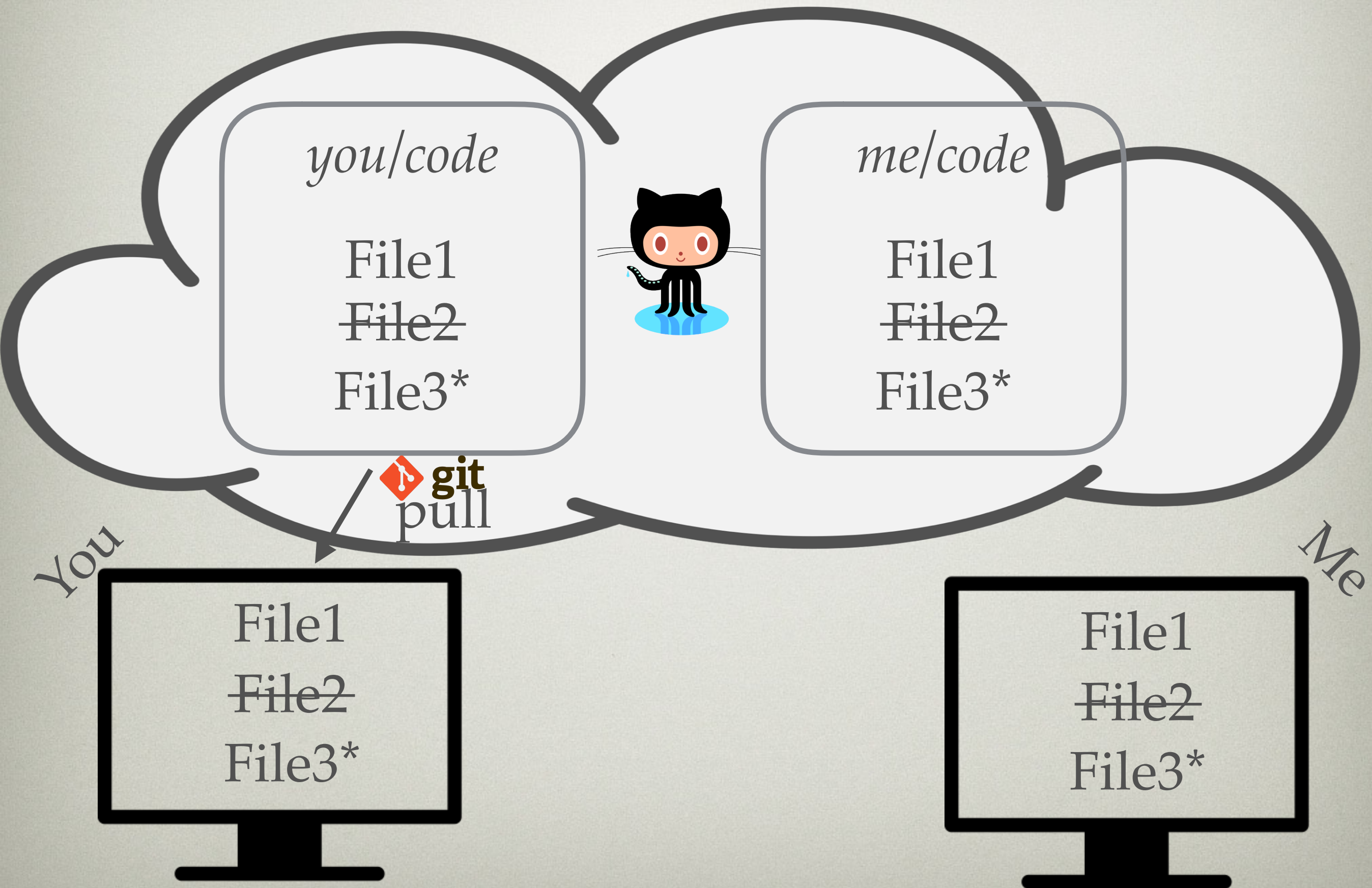


# LETS DIG DOWN...



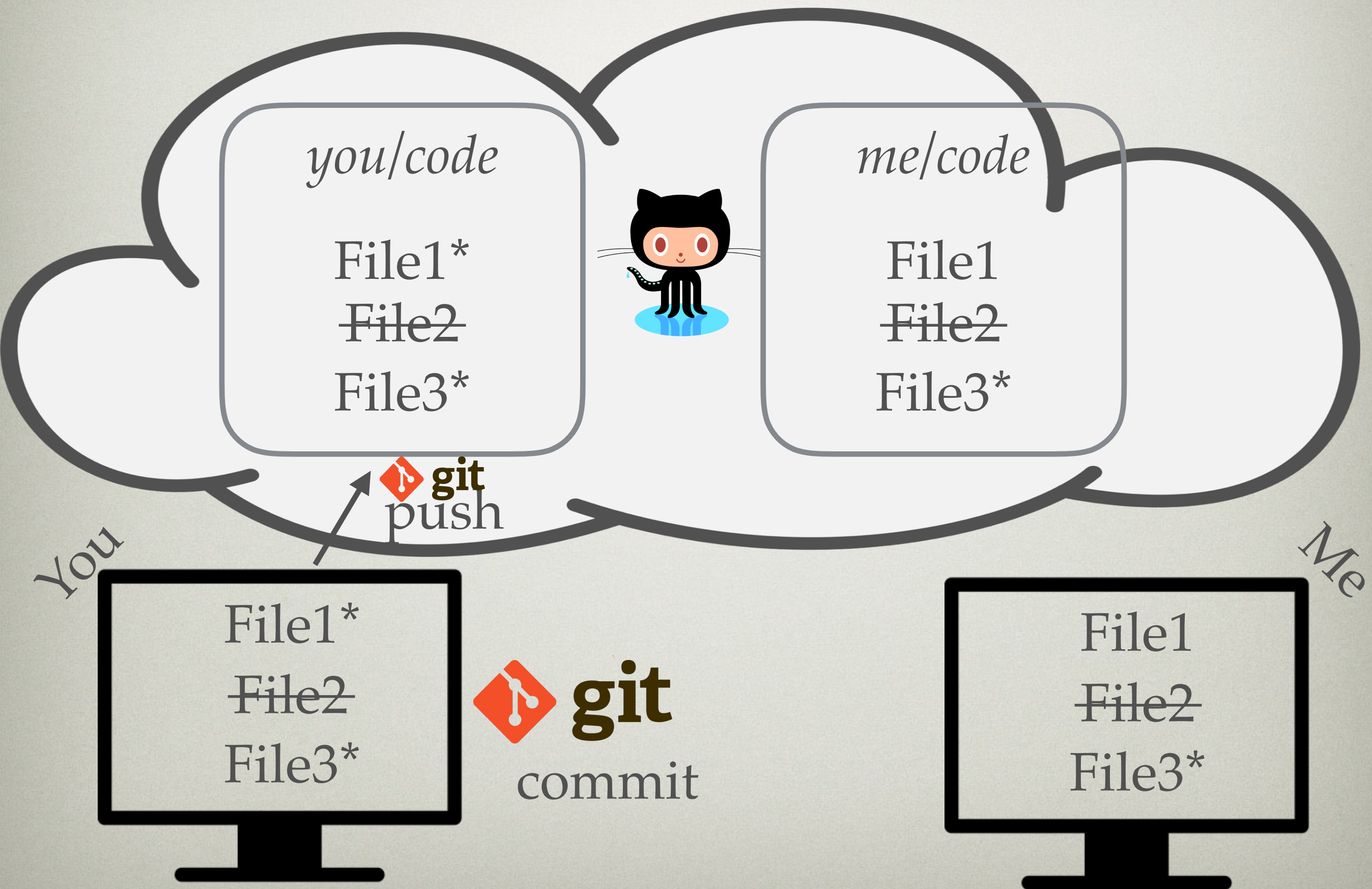


# LETS DIG DOWN...



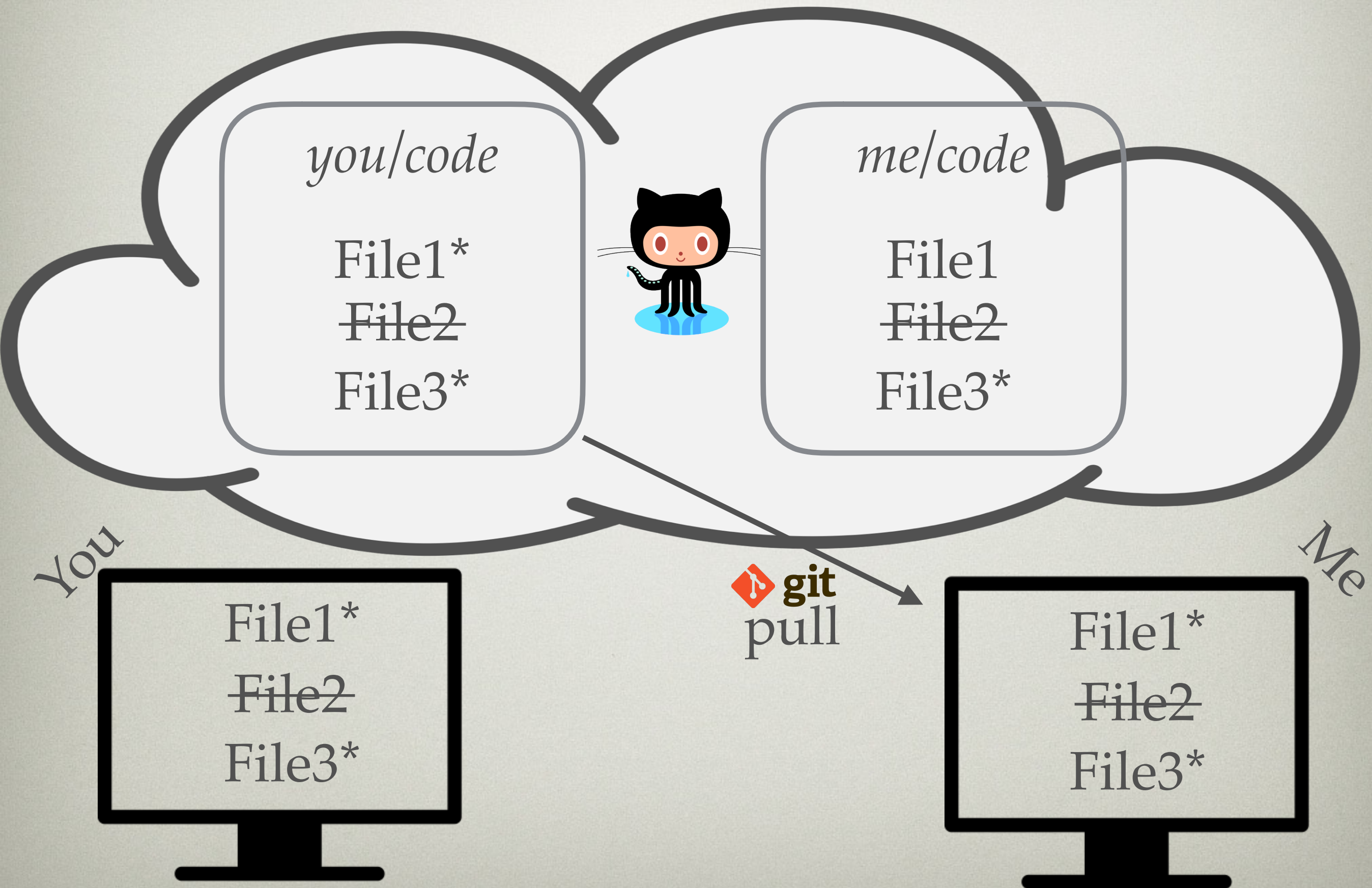


# LETS DIG DOWN...





# LETS DIG DOWN...





# NOW START ON THE NOTEBOOKS!

You can get to the notebooks at:

<http://bit.ly/ahw17-gh-day3>

(or `git pull` in previous days' repo)

Start with the “start here” notebook, work up through prob 2. Problems 3+ are better after the break.

If you think you know that already, you can try the notebooks on documentation and testing (although you might still need to do parts of “start here” to have something to work with). Or: help those who have questions!



# 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**



# PYTHON PACKAGING TERMINOLOGY

- “package”: the biggest thing. E.g., *astropy*, *numpy*, *sunpy*. A directory with an “\_\_init\_\_.py”



# 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”



# 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



# 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”



# 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\*



# 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**



# 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!*



# LICENSING YOUR CODE

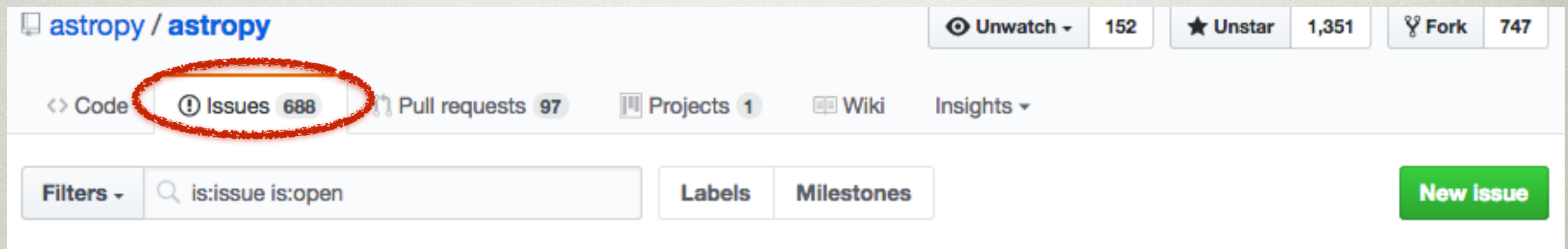
- Rule #1: Have a license!
- Rule #2: There is no rule #2.

(see problem sets for more)



# HOW TO START CONTRIBUTING TO EXISTING LIBRARIES

- Look at the contributing guidelines, e.g. “CONTRIBUTING.md” files on GitHub repos, documentation pages.
- Look at the issues page

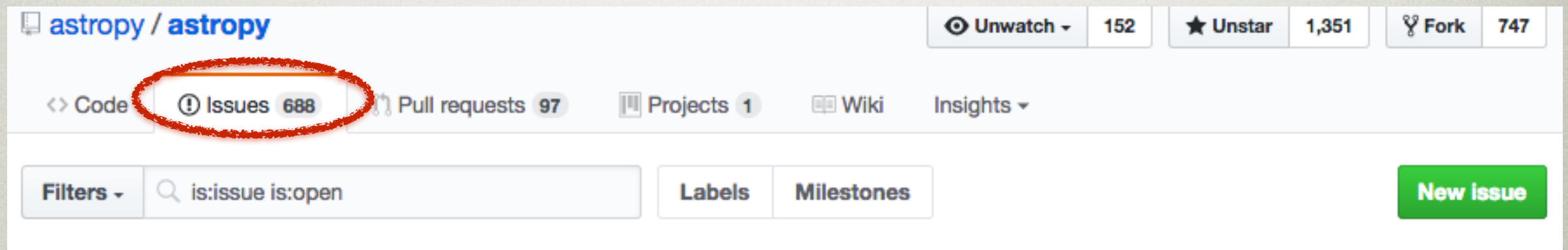


- Use the labels to navigate and find something to work on, e.g. **Package-novice** are supposed to be newcomer friendly.
- Open a PR with the fix. Expect to receive a code review and comments before your fix is merged.



# HOW TO START CONTRIBUTING TO EXISTING LIBRARIES

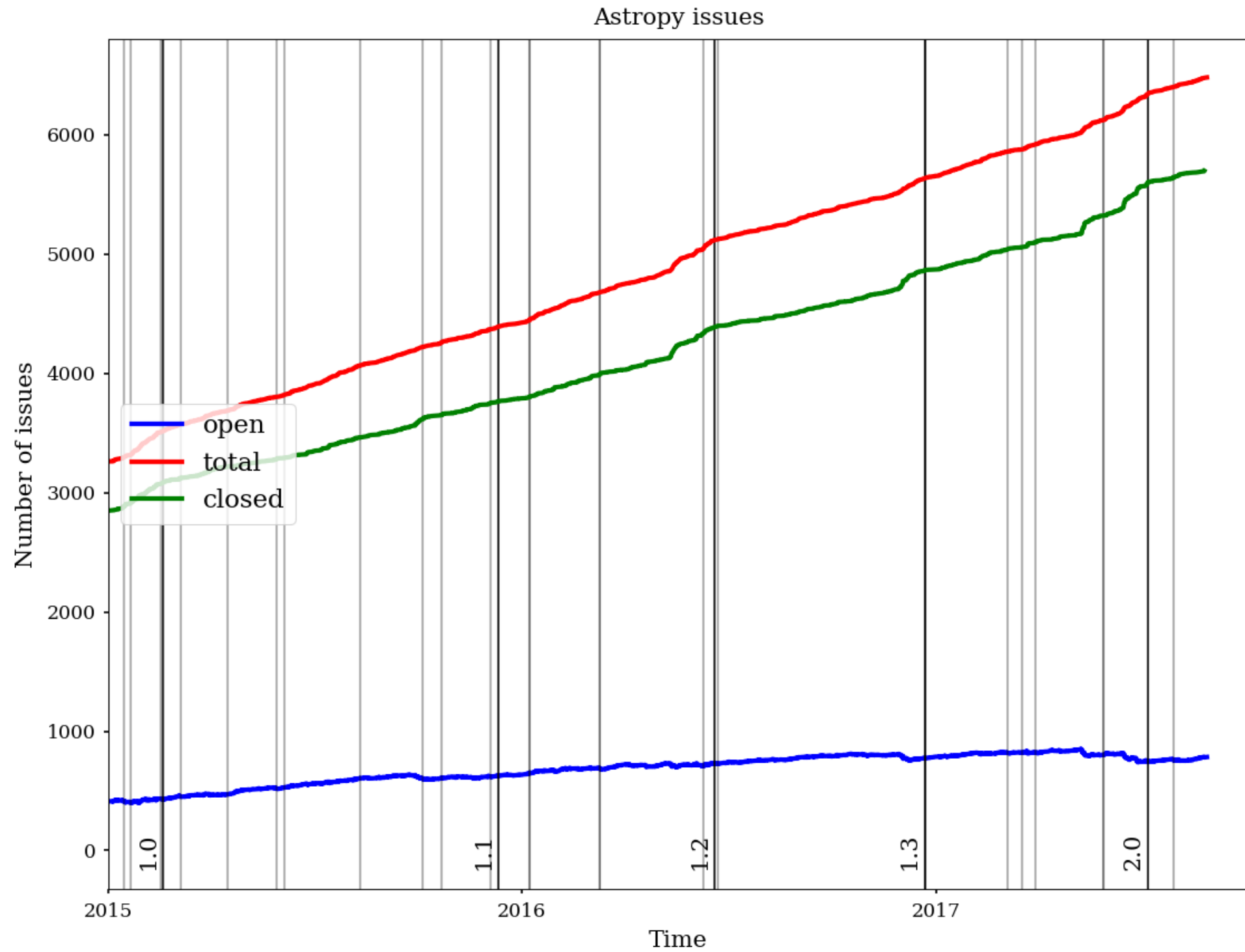
- Look at the guidelines, e.g. “CONTRIBUTING.md” files on GitHub repos.
- Look at the issues page



- Use the labels to navigate and find something to work on, e.g. **Package-novice** are supposed to be newcomer friendly.
- Open a PR with the fix. Expect to receive a code review and comments before your fix is merged.

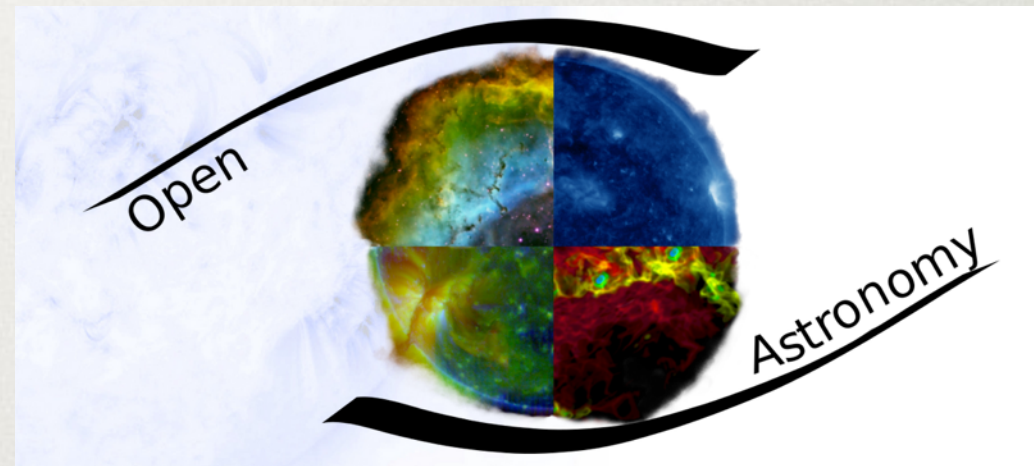


# HOW TO START CONTRIBUTING





# GOOGLE SUMMER OF CODE WITH OPENASTRONOMY



- 10 week long program to work on open source astronomy software



# NOW GO DO IT YOURSELF!

You can get to the notebooks at:

<http://bit.ly/ahw17-gh-day3>

(or `git pull` in previous days' repo)

Start with the “start here” notebook, work up through prob 2. Problems 3+ are better after the break.

If you think you know that already, you can try the notebooks on documentation and testing (although you might still need to do parts of “start here” to have something to work with). Or: help those who have questions!