# Code Repositories: What they are, why you might care, and How to Use Them

**Erik Tollerud**

**@eteq**

Space Telescope Science Institute

Giacconi Fellow
Astropy Coordination Committee

astropy-powered
astropy.org

# 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

Atlassian
Bitbucket

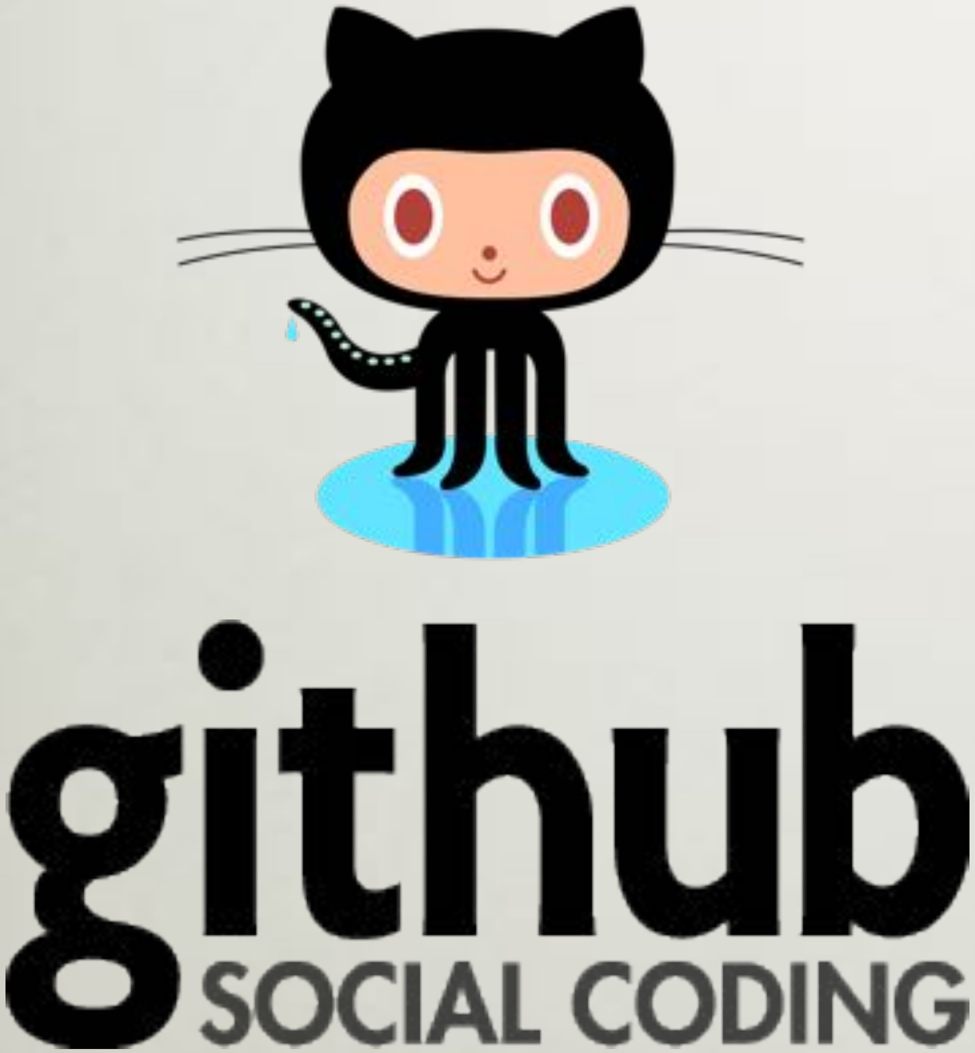
source
forge

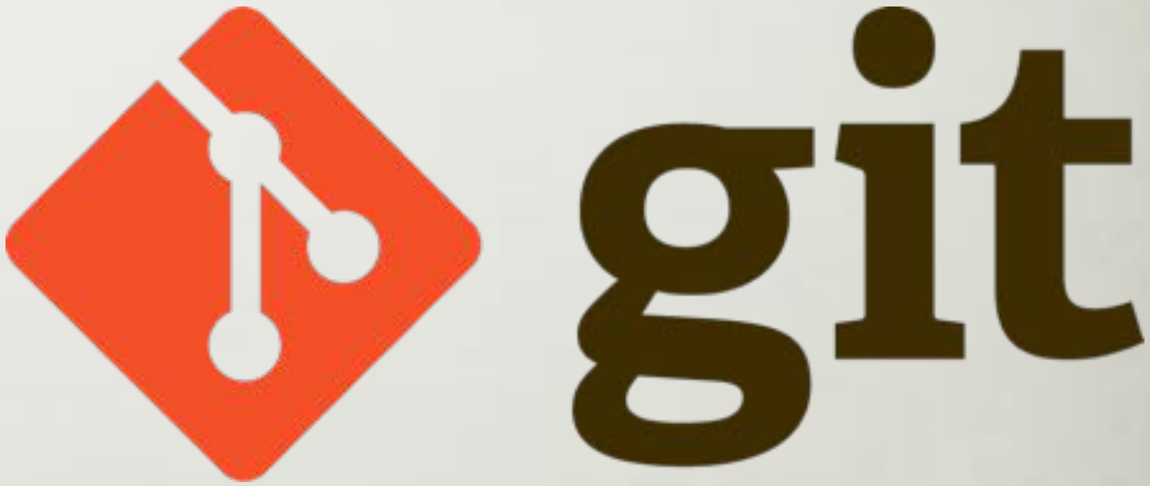
{P}

GitLab

PERFORCE

# Where can you do this?

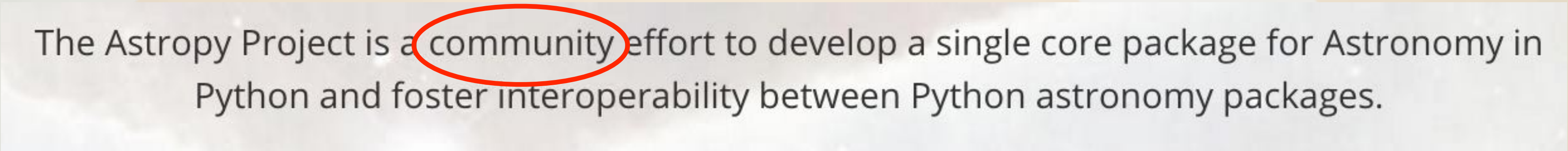




# As a "case study"...

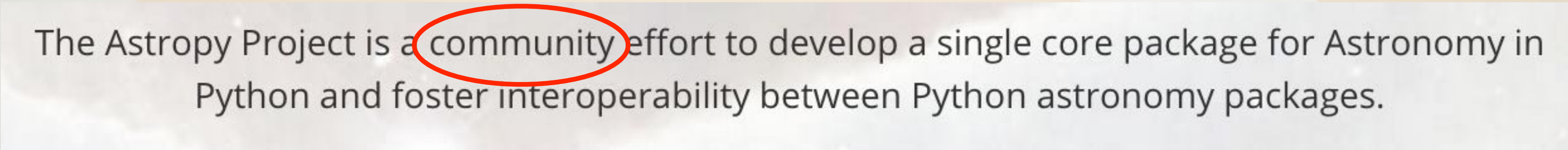
astropy
A Community Python Library for Astronomy

# 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?

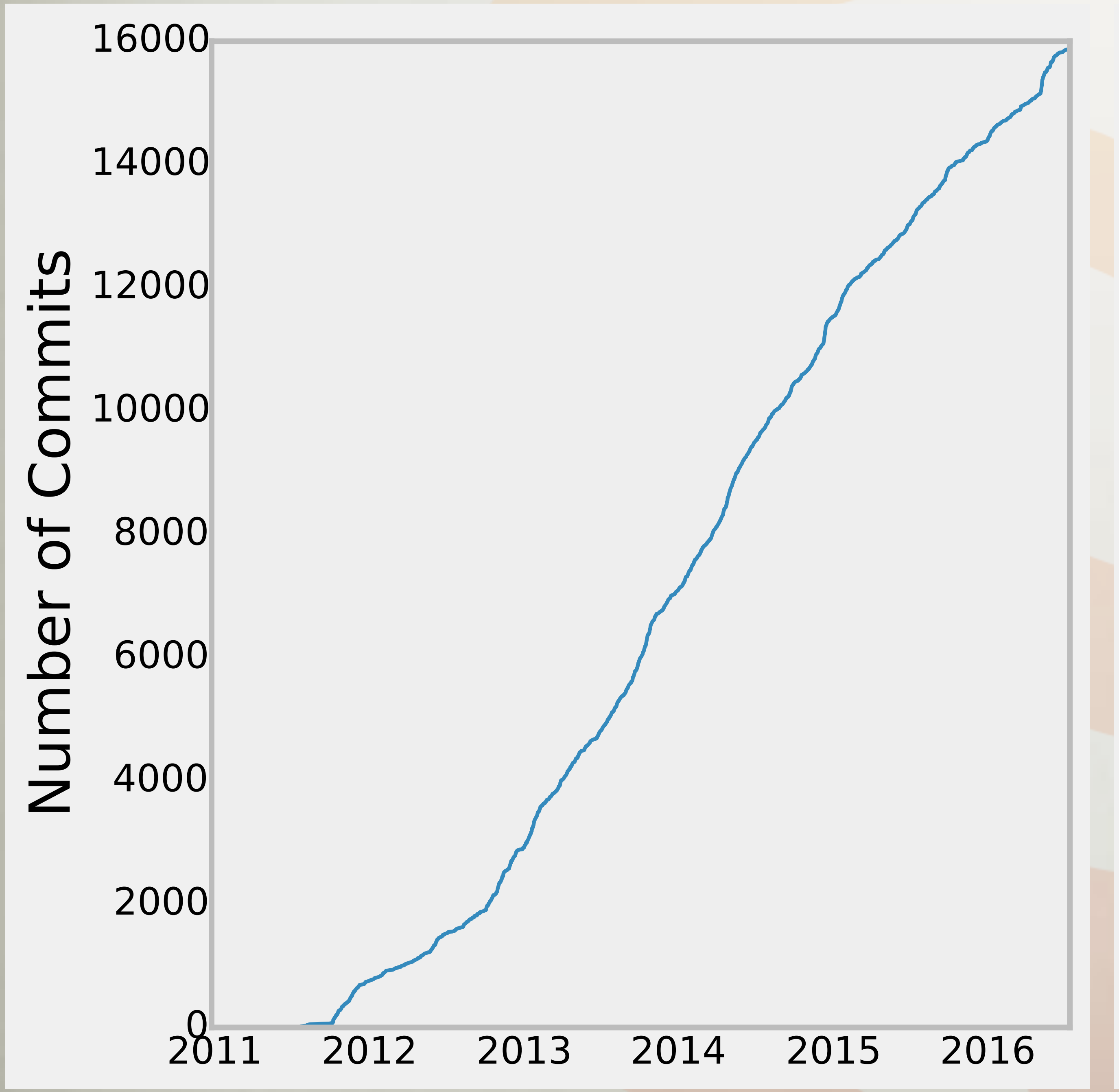


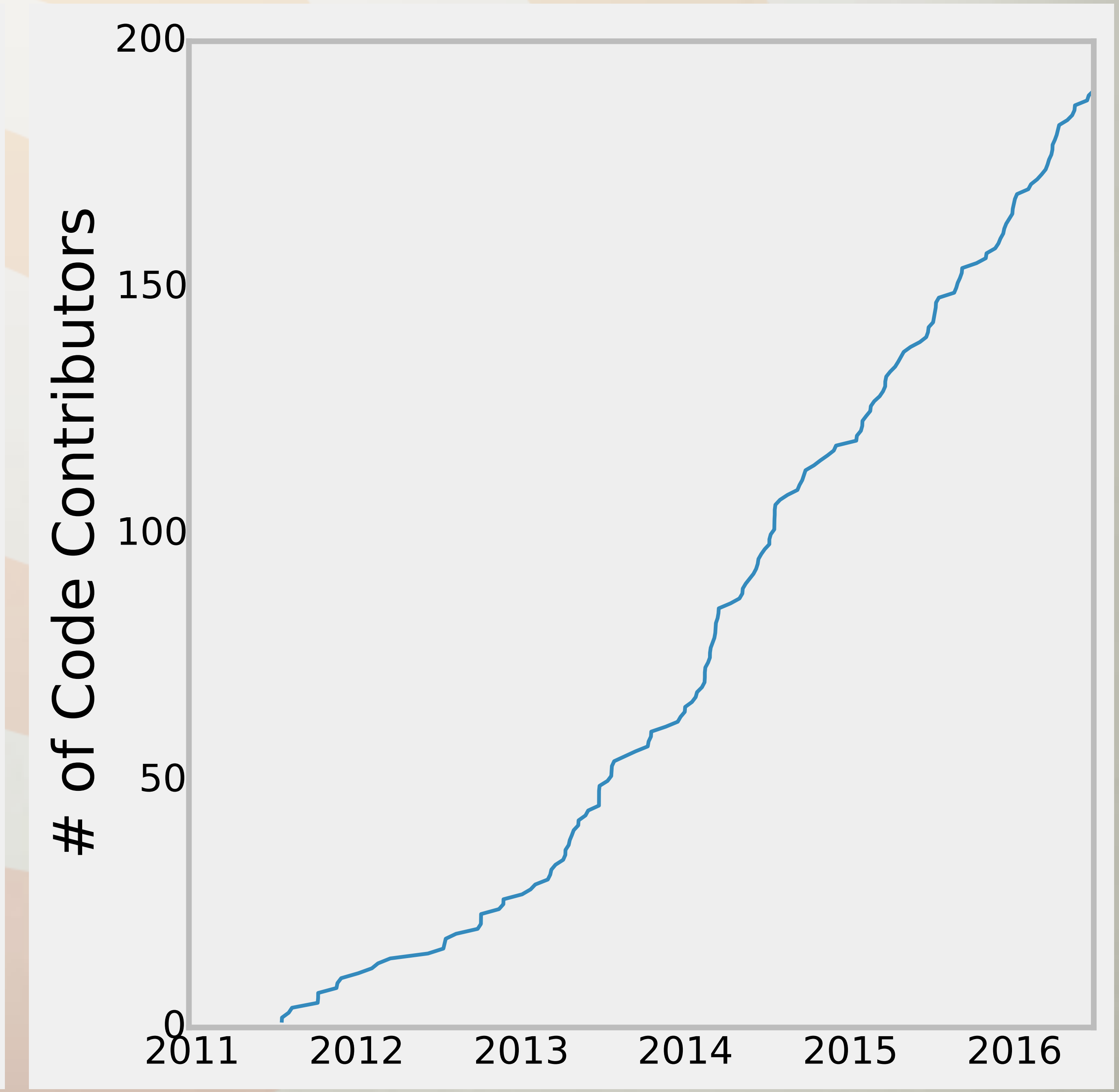
The Astropy Project is a community effort to develop a single core package for Astronomy in Python and foster interoperability between Python astronomy packages.

This means both ***by*** and ***for*** the community

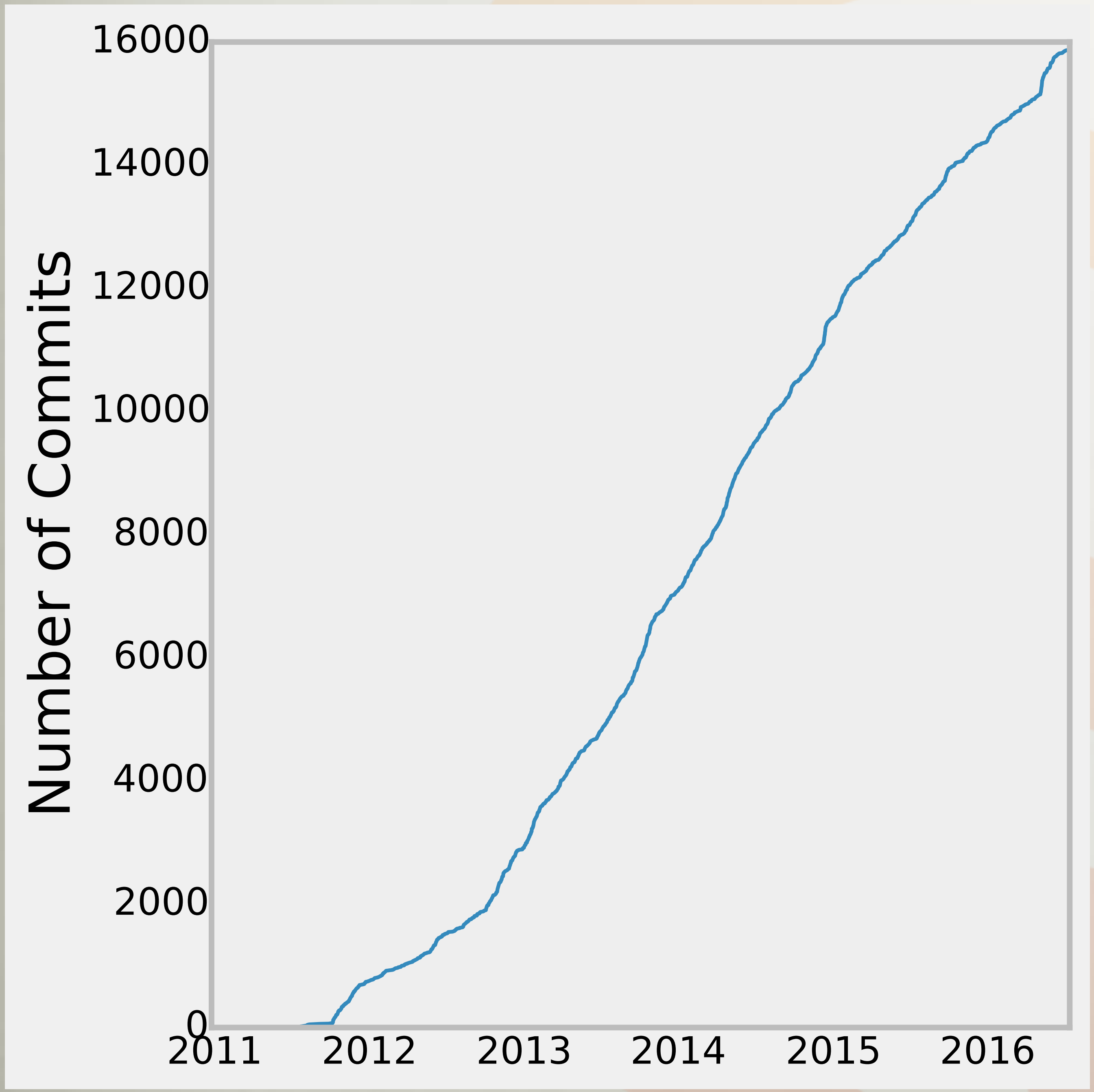
(Professional) Astronomers help write it

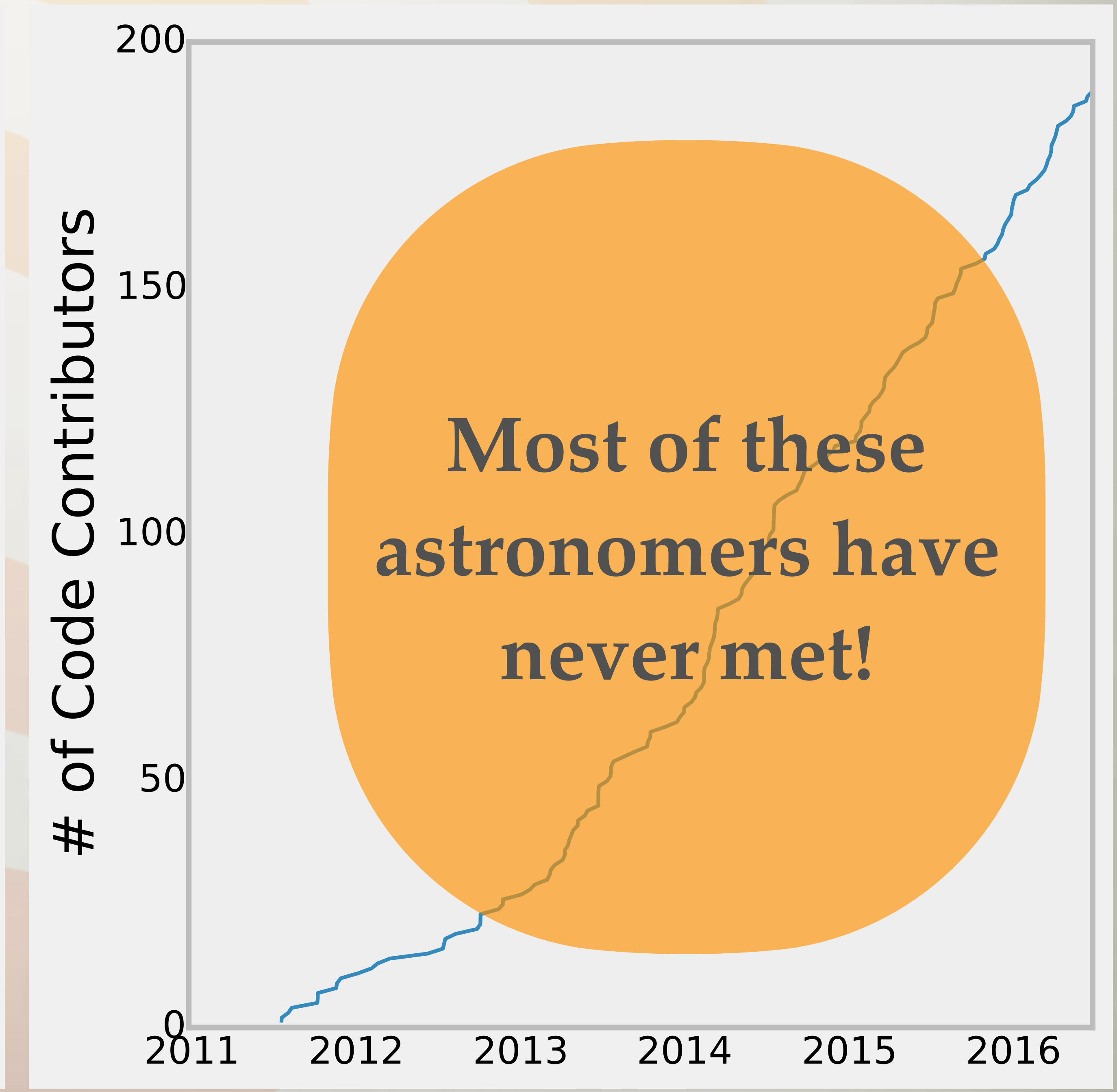
# Contributions are Growing





# Contributions are Growing



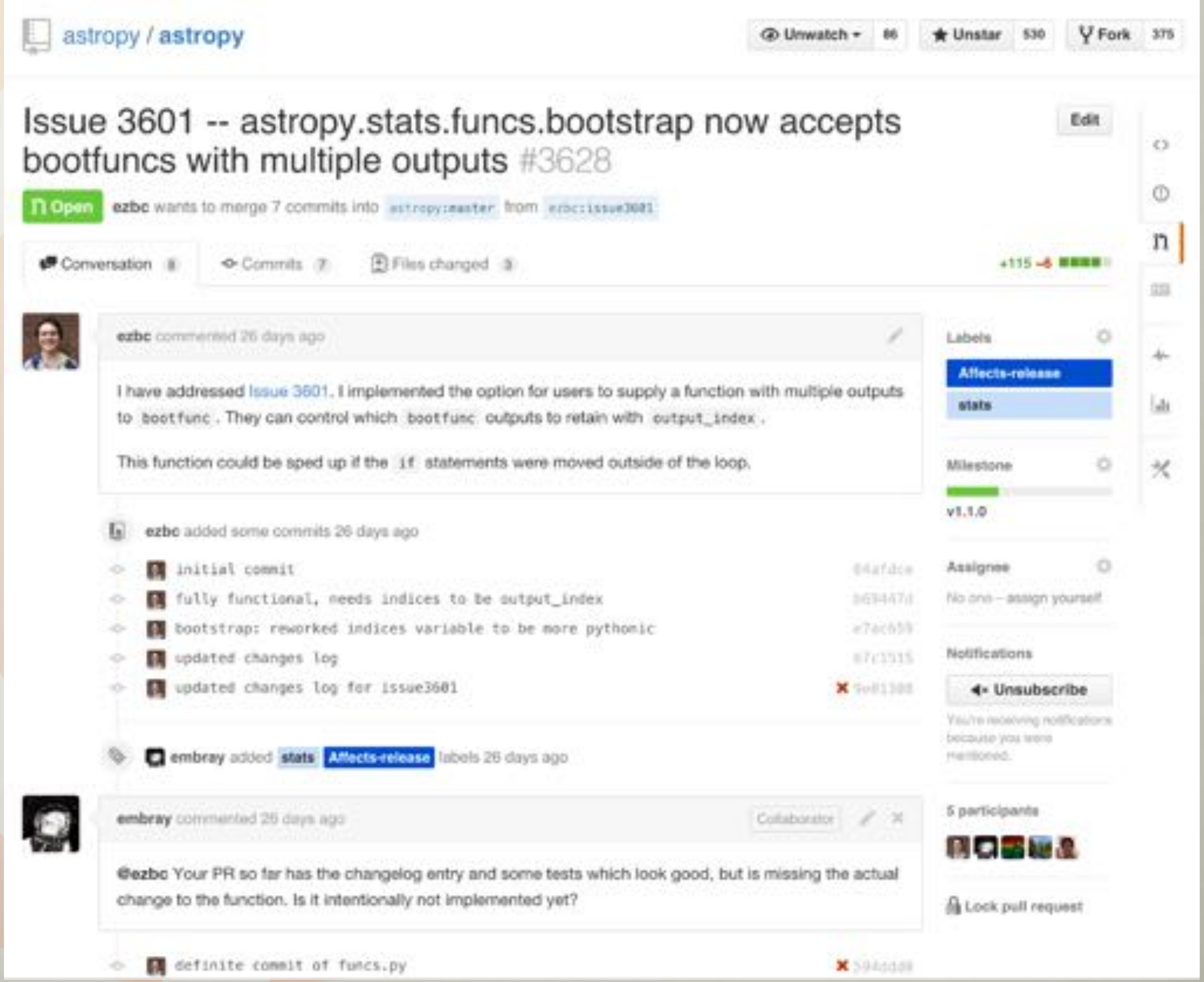


Most of these astronomers have never met!

# Keys To Distributed Development

# Keys To Distributed Development: Github

github
SOCIAL CODING



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`

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                                        87c1515
updated changes log for issue3601                        ✗ 9e01308
```

embray added stats Affects-release labels 26 days ago

embray commented 26 days ago

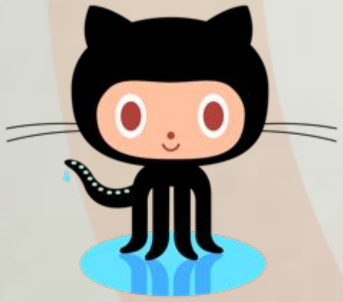
@ezbc Your PR so far has the changelog entry and some tests which look good, but is missing the actual change to the function. Is it intentionally not implemented yet?

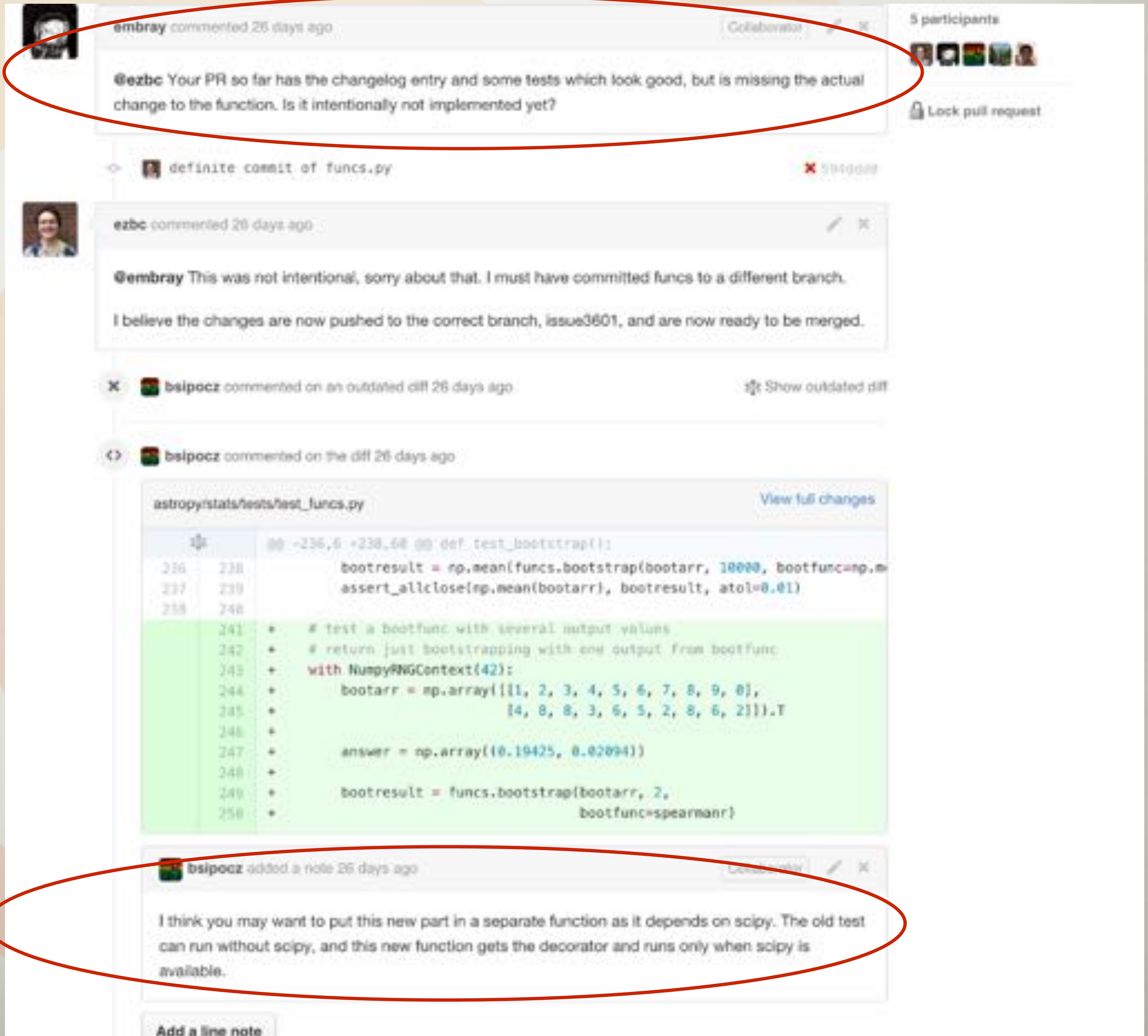
```
definite commit of funcs.py                              ✗ 594dd48
```

You're receiving notifications because you were mentioned.

# Keys To Distributed Development: Github

github
SOCIAL CODING





embray commented 26 days ago

@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` ✗ 594dddd

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

bsipocz commented on the diff 26 days ago

astropy/stats/tests/test_funcs.py

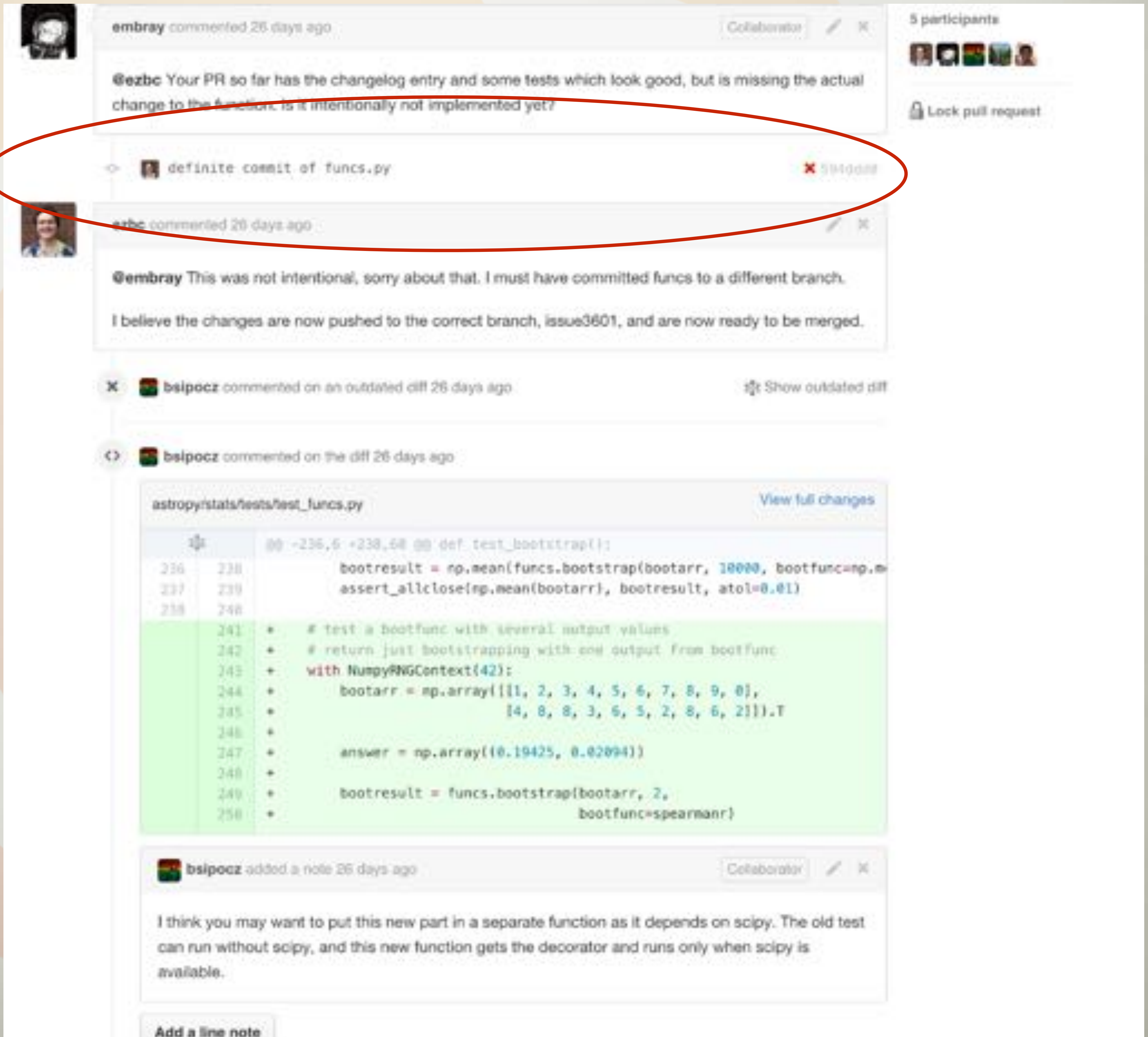
```
@@ -236,6 +238,68 @@ def test_bootstrap():
          bootresult = np.mean(funcs.bootstrap(bootarr, 10000, bootfunc=np.m
          assert_allclose(np.mean(bootarr), bootresult, atol=0.01)

+     # test a bootfunc with several output values
+     # return just bootstrapping with one output from bootfunc
+     with NumpyRNGContext(42):
+         bootarr = np.array([[1, 2, 3, 4, 5, 6, 7, 8, 9, 0],
+                             [4, 8, 8, 3, 6, 5, 2, 8, 6, 2]]).T
+
+         answer = np.array((0.19425, 0.02094))
+
+         bootresult = funcs.bootstrap(bootarr, 2,
+                                      bootfunc=spearmanr)
```

bsipocz added a note 26 days ago

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.

# Keys To Distributed Development: Github

github
SOCIAL CODING



**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` ✖ [illegible]

**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

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

astropy/stats/tests/test_funcs.py

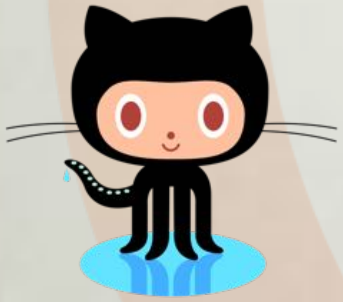
```
@@ -236,6 +238,68 @@ def test_bootstrap():
         bootresult = np.mean(funcs.bootstrap(bootarr, 10000, bootfunc=np.m
         assert_allclose(np.mean(bootarr), bootresult, atol=0.01)

+    # test a bootfunc with several output values
+    # return just bootstrapping with one output from bootfunc
+    with NumpyRNGContext(42):
+        bootarr = np.array([[1, 2, 3, 4, 5, 6, 7, 8, 9, 0],
+                            [4, 8, 8, 3, 6, 5, 2, 8, 6, 2]]).T
+
+        answer = np.array((0.19425, 0.02094))
+
+        bootresult = funcs.bootstrap(bootarr, 2,
+                                     bootfunc=spearmanr)
```

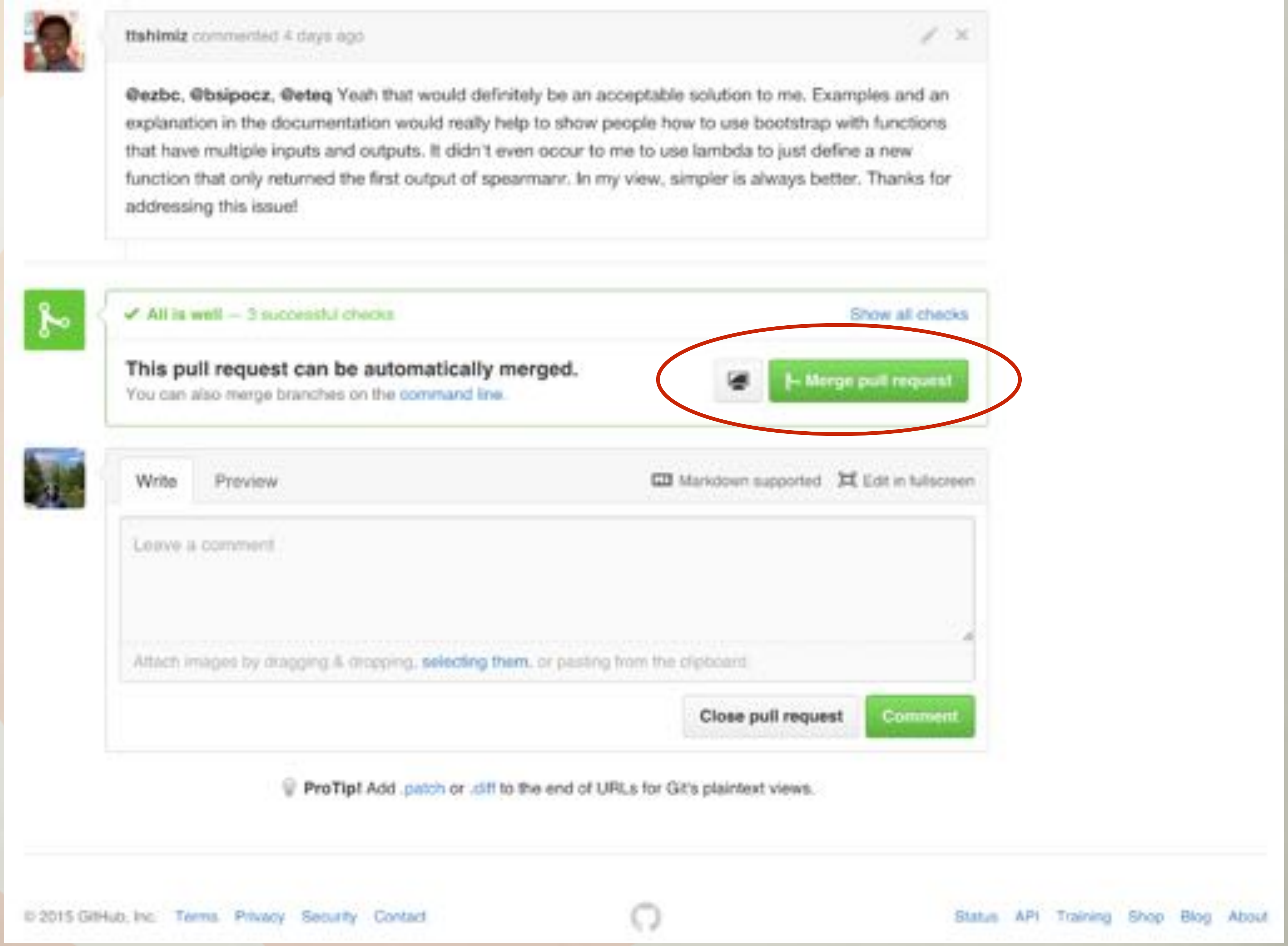
**bsipocz** added a note 26 days ago Collaborator

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.

# Keys To Distributed Development: Github

github
SOCIAL CODING





**@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

**This pull request can be automatically merged.**
You can also merge branches on the command line.

Attach images by dragging & dropping, selecting them, or pasting from the clipboard

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

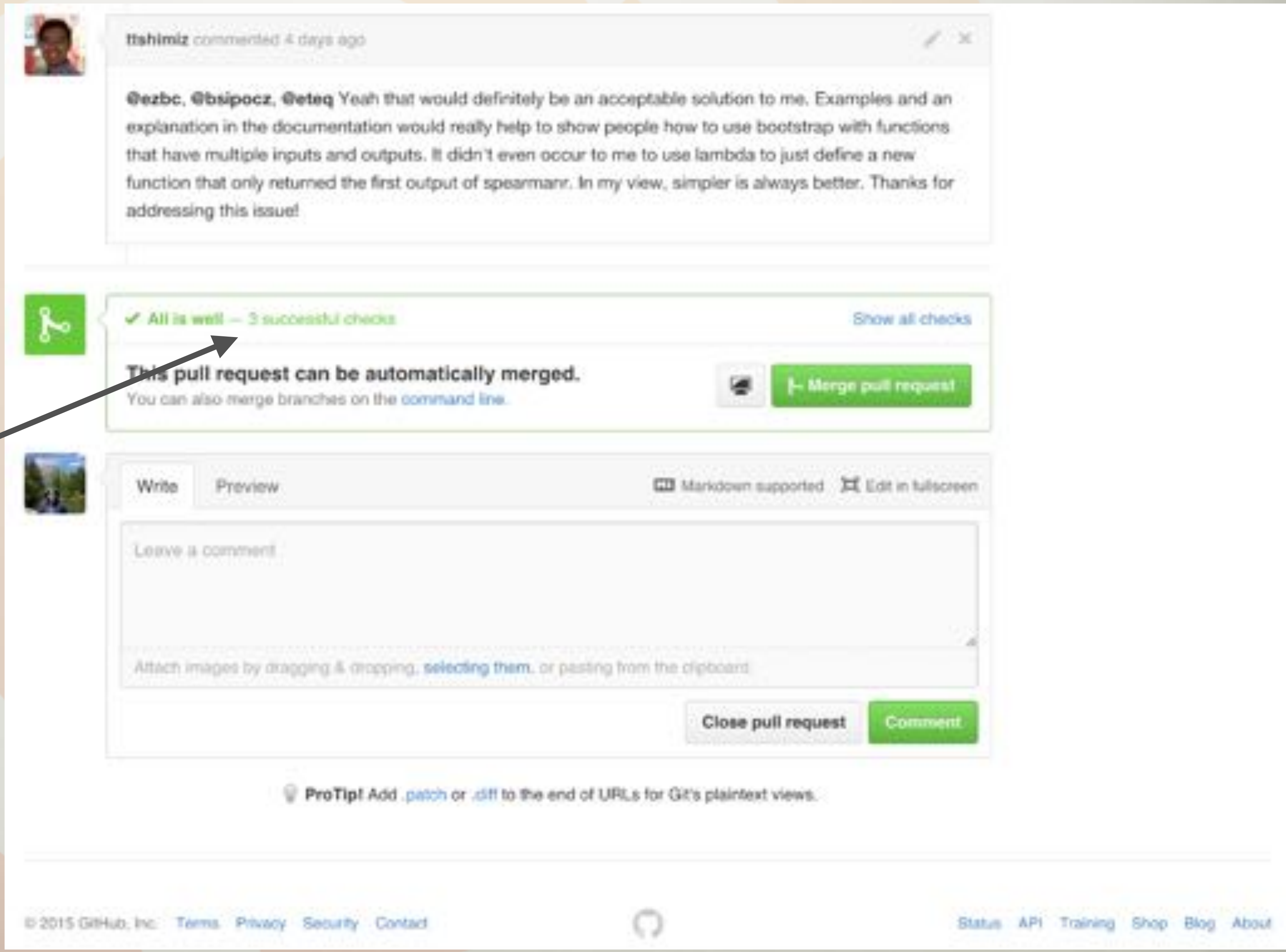
© 2015 GitHub, Inc.

# Keys To Distributed Development: Continuous Testing

py.test

Travis

AppVeyor



**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

**This pull request can be automatically merged.**

You can also merge branches on the command line.

Leave a comment

Attach images by dragging & dropping, selecting them, or pasting from the clipboard

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

© 2015 GitHub, Inc.

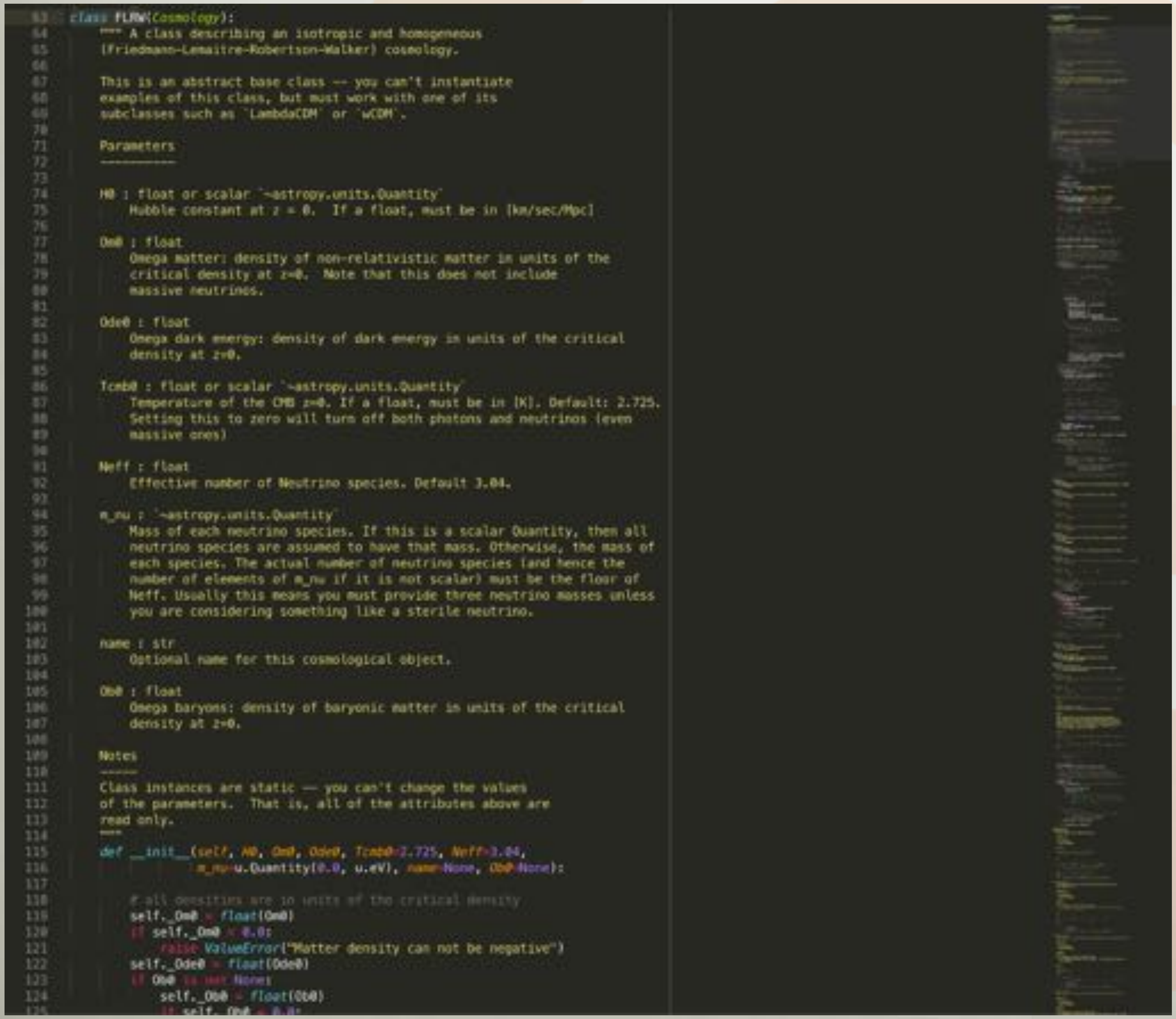
# Keys To Distributed Development: Docs

Sphinx

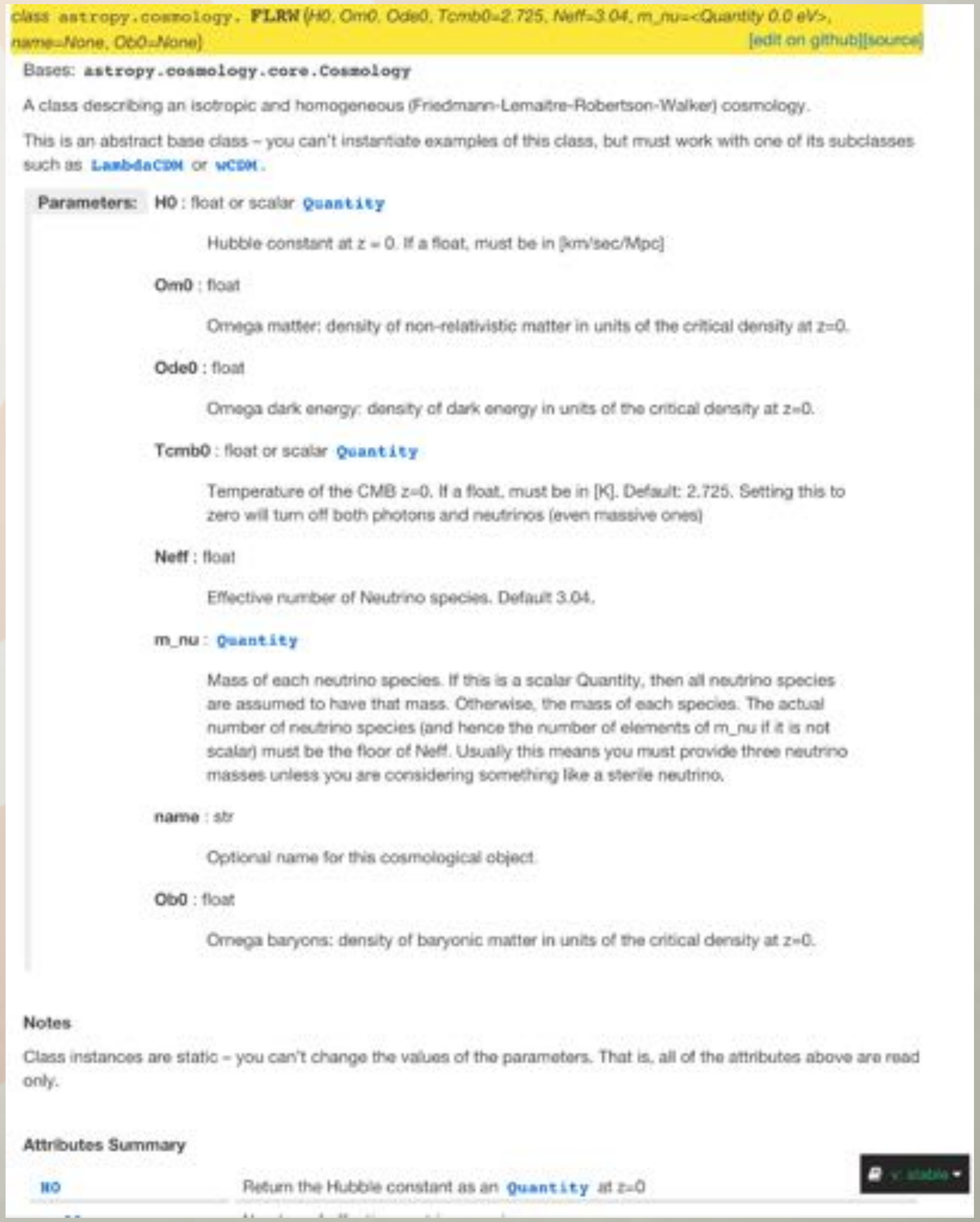
Python documentat on generator

Read the Docs

Create, host, and browse documentation.



```
class FLRW(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 `~astropy.units.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.  Note that this does not include
        massive neutrinos.

    Ode0 : float
        Omega dark energy: density of dark energy in units of the critical
        density at z=0.

    Tcmb0 : float or scalar `~astropy.units.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 : `~astropy.units.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.
    """
    def __init__(self, H0, Om0, Ode0, Tcmb0=2.725, Neff=3.04,
                 m_nu=u.Quantity(0.0, u.eV), name=None, Ob0=None):

        # all densities are in units of the critical density
        self._Om0 = float(Om0)
        if self._Om0 < 0.0:
            raise ValueError("Matter density can not be negative")
        self._Ode0 = float(Ode0)
        if Ob0 is not None:
            self._Ob0 = float(Ob0)
            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 |

# Keys To Distributed Development In a Shared Repo

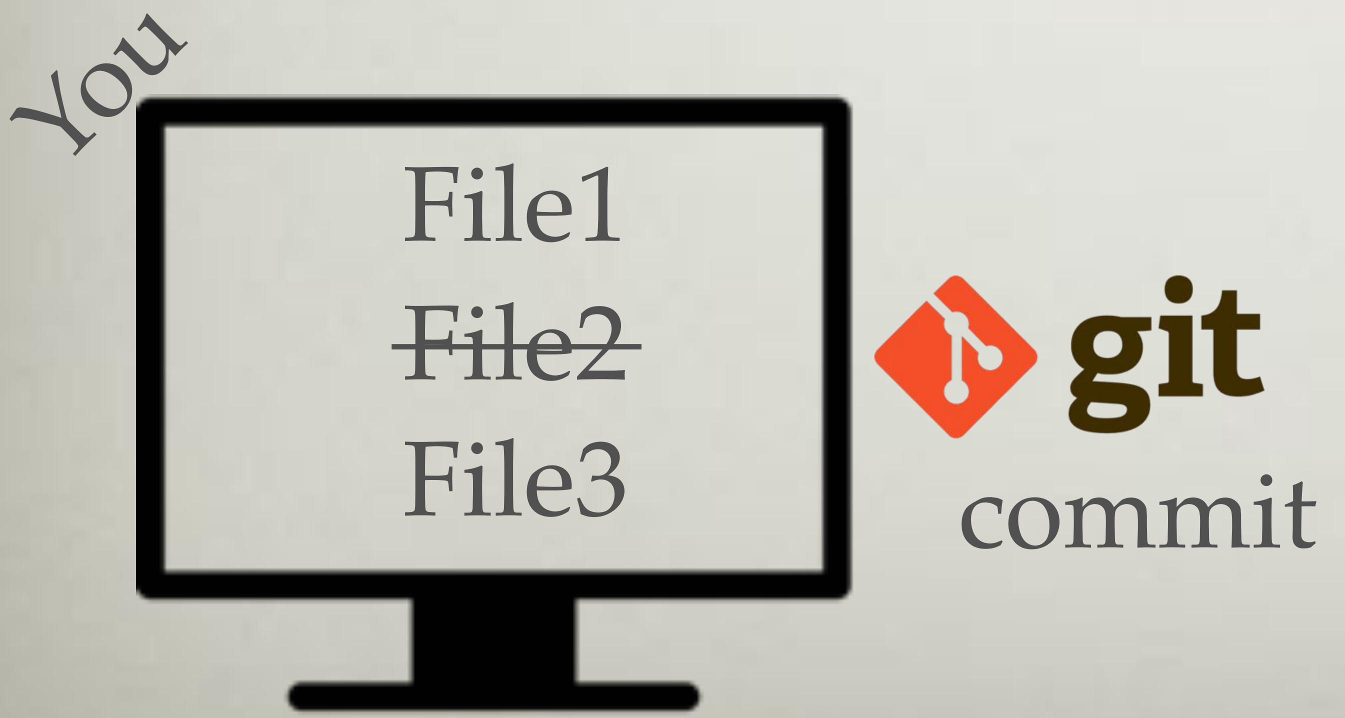
- *git* + Github for sharing
- Test *everything* (automatically)
- *Easy* documentation (⇒ thorough)

# Lets dig down on how you do Shared Development with Public Repos

# Lets dig down...

You
File1
File2

# Lets dig down...



You

File1

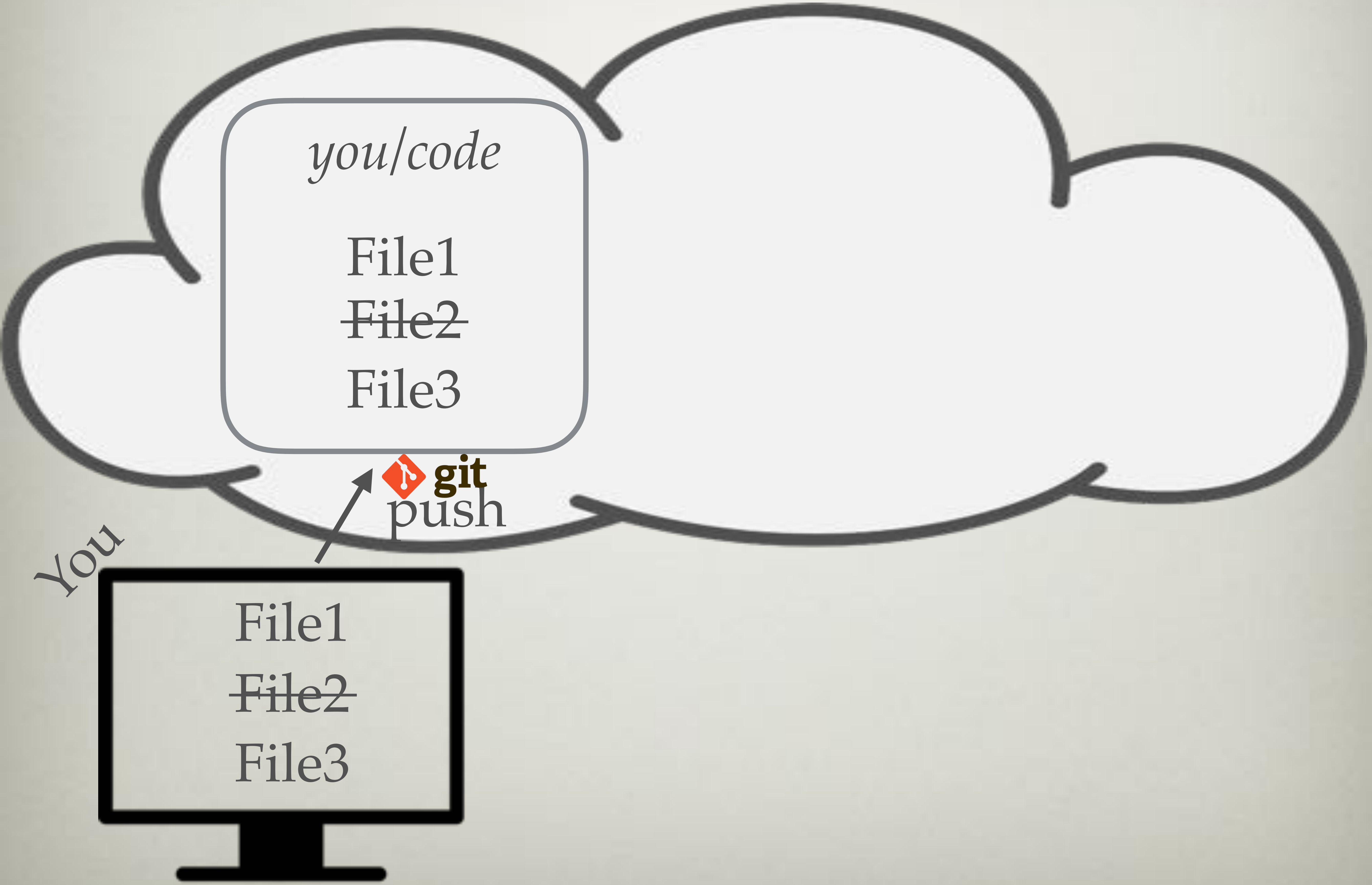
~~File2~~

File3

git

commit

# Lets dig down...



*you/code*

File1

~~File2~~

File3

git push

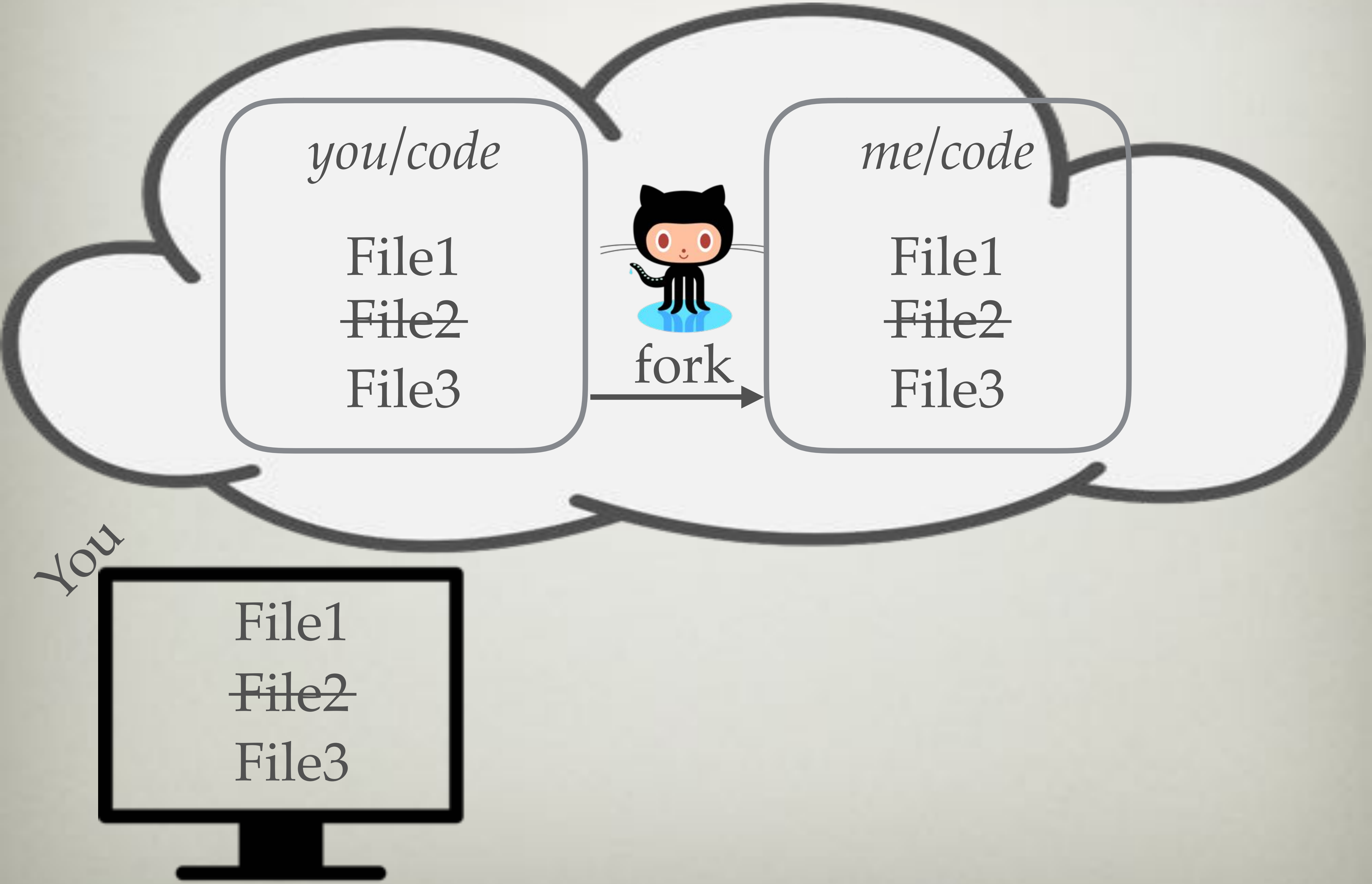
You

File1

~~File2~~

File3

# Lets dig down...



*you/code*

File1
~~File2~~
File3

fork

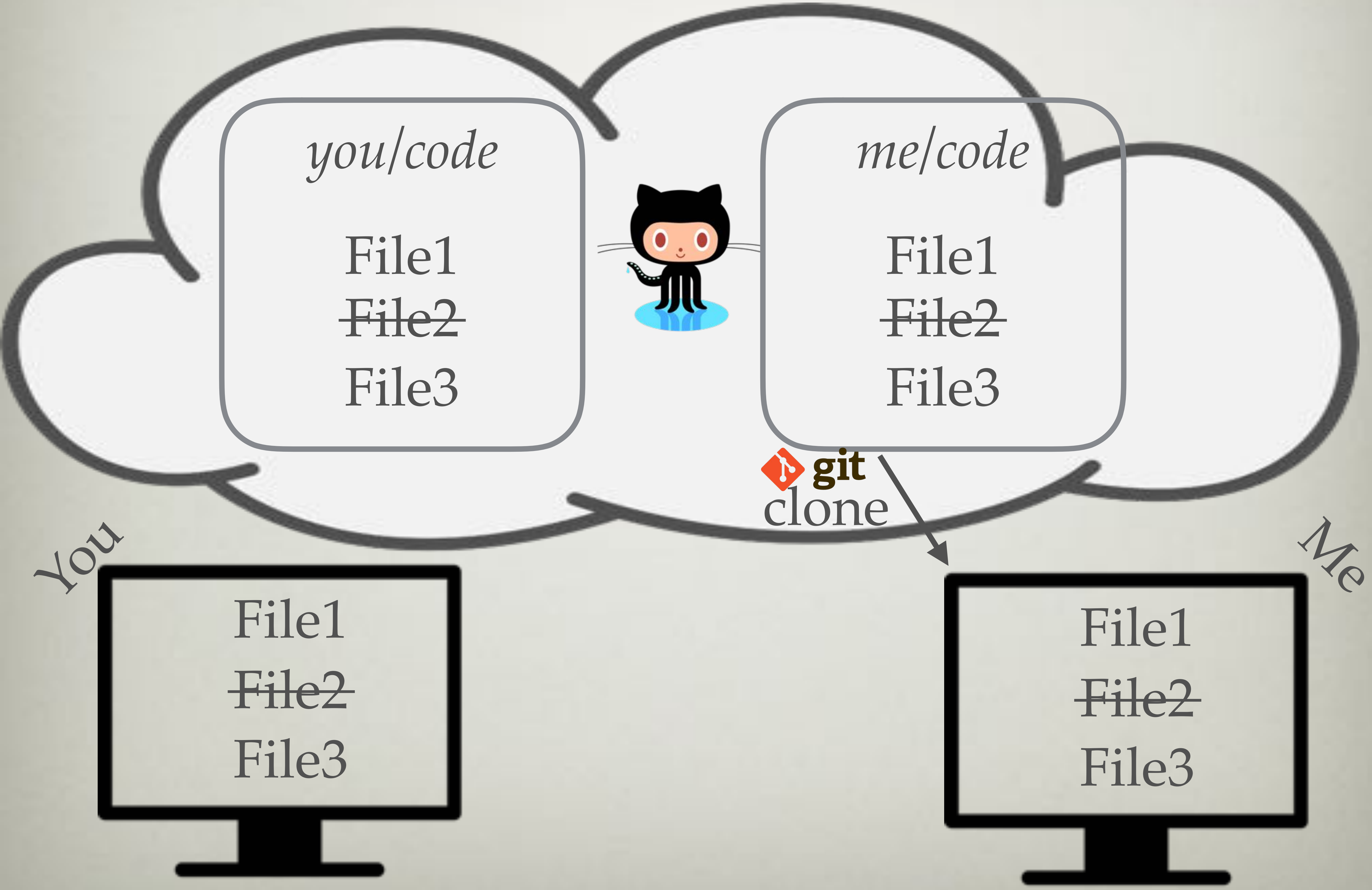
*me/code*

File1
~~File2~~
File3

You

File1
~~File2~~
File3

# Lets dig down...



*you/code*

File1

~~File2~~

File3

*me/code*

File1

~~File2~~

File3

git clone

You

File1

~~File2~~

File3

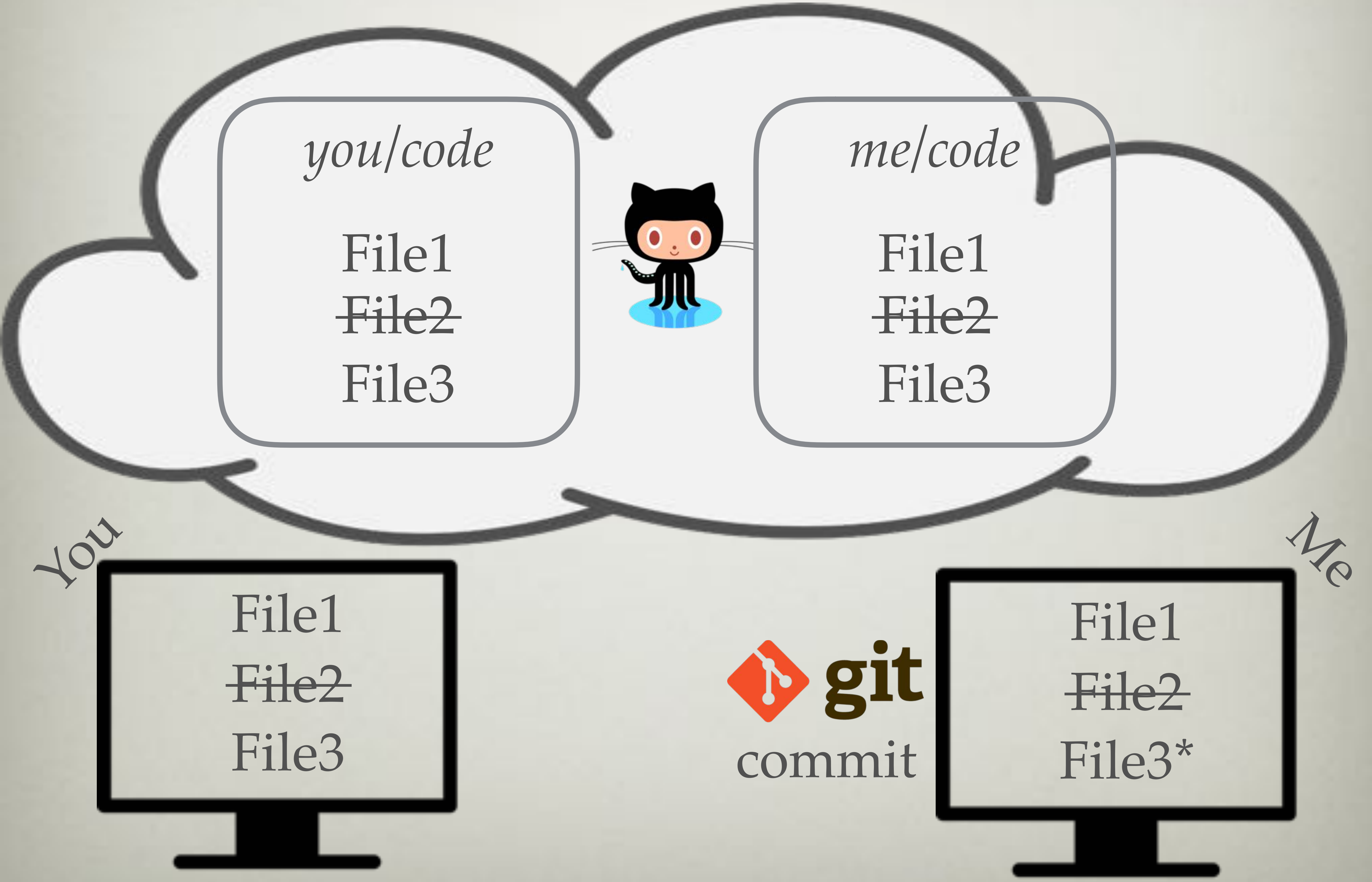
Me

File1

~~File2~~

File3

# Lets dig down...



*you/code*

File1
~~File2~~
File3

*me/code*

File1
~~File2~~
File3

You

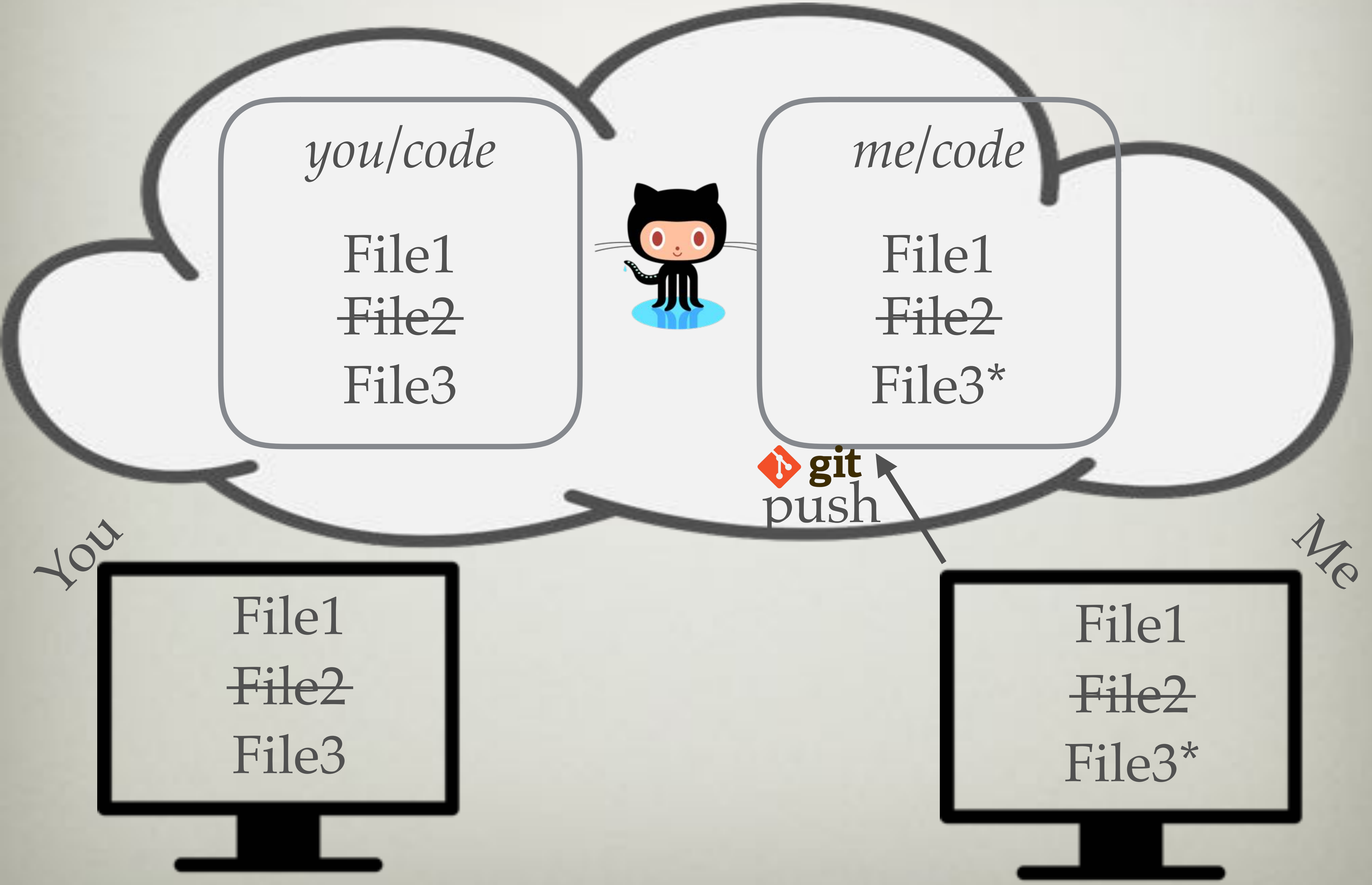
File1
~~File2~~
File3

git
commit

Me

File1
~~File2~~
File3*

# Lets dig down...



*you/code*

File1

~~File2~~

File3

*me/code*

File1

~~File2~~

File3*

git

push

You

File1

~~File2~~

File3

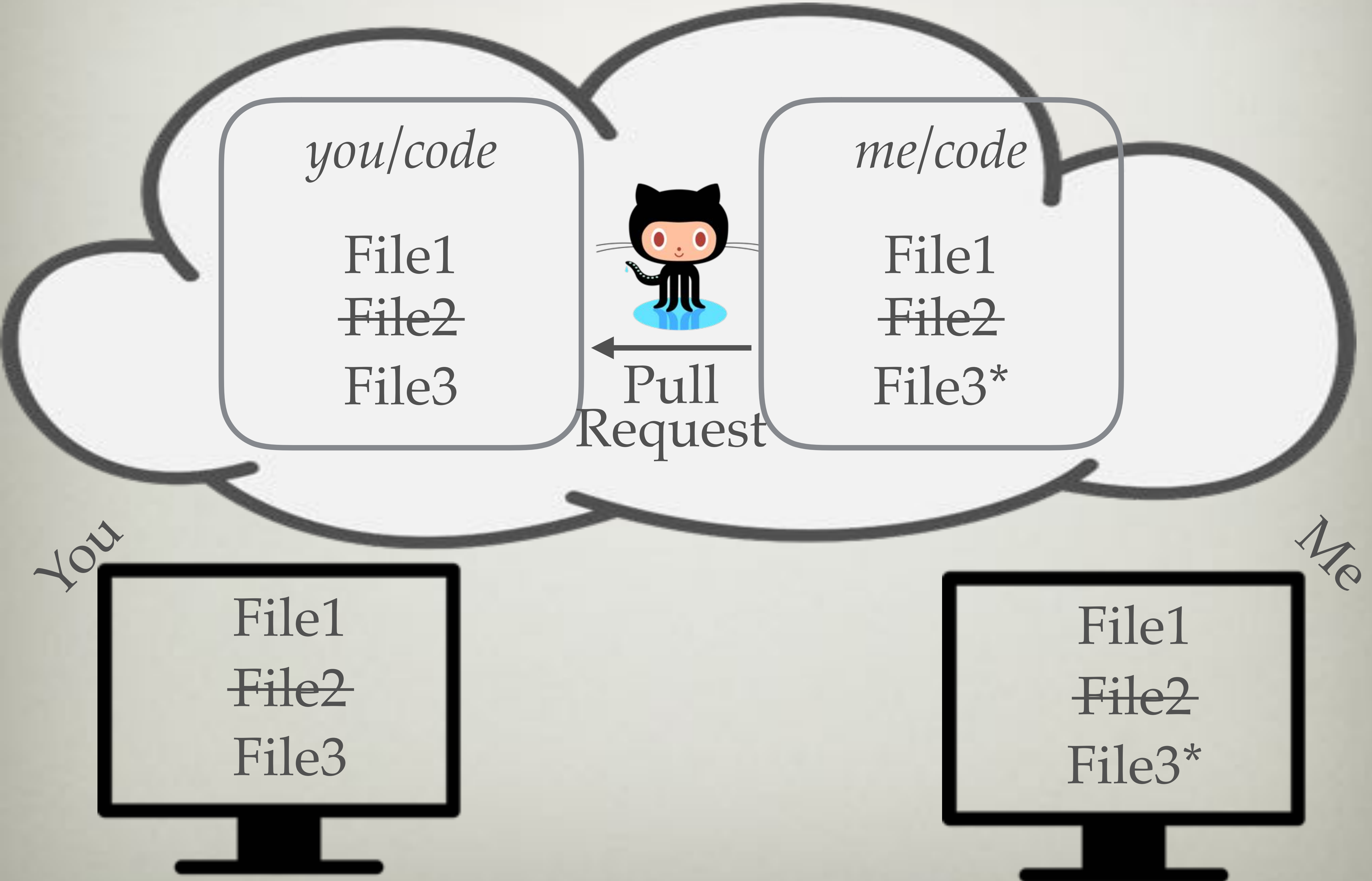
Me

File1

~~File2~~

File3*

# Lets dig down...



*you/code*

File1
~~File2~~
File3

Pull Request

*me/code*

File1
~~File2~~
File3*

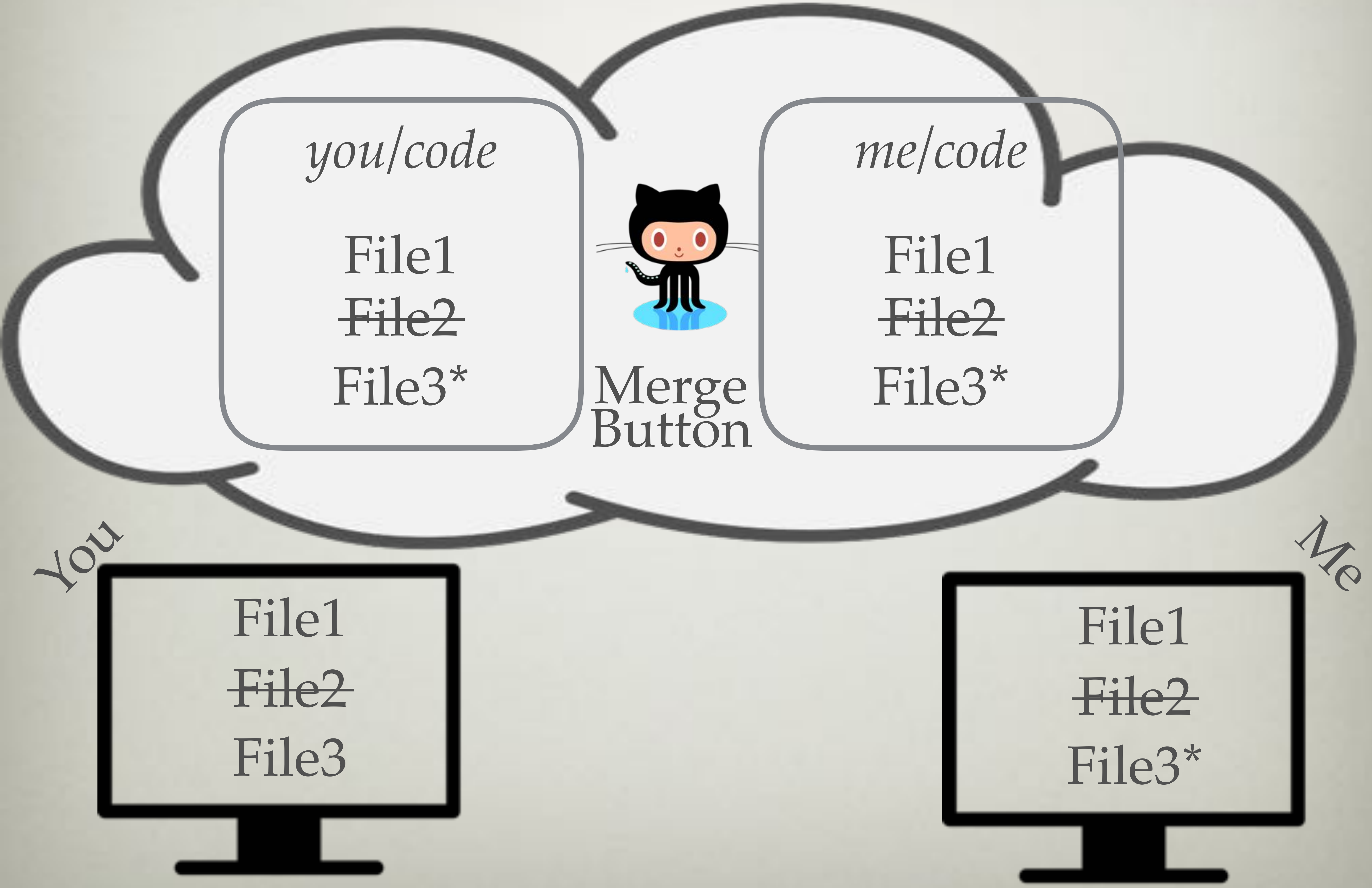
You

File1
~~File2~~
File3

Me

File1
~~File2~~
File3*

# Lets dig down...



*you/code*

File1
~~File2~~
File3*

Merge
Button

*me/code*

File1
~~File2~~
File3*

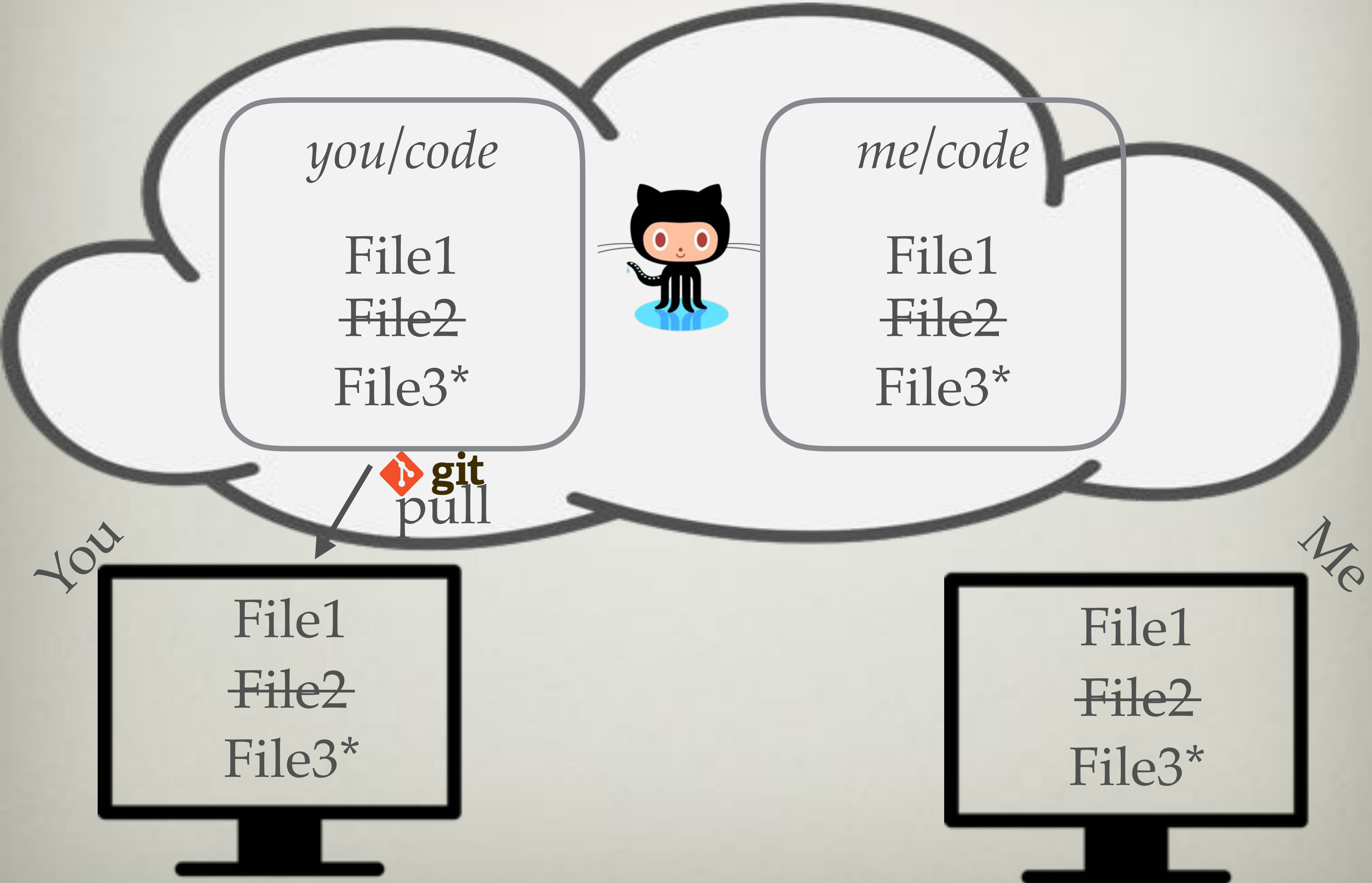
You

File1
~~File2~~
File3

Me

File1
~~File2~~
File3*

# Lets dig down...



*you/code*

File1

~~File2~~

File3*

*me/code*

File1

~~File2~~

File3*

git pull

You

File1

~~File2~~

File3*

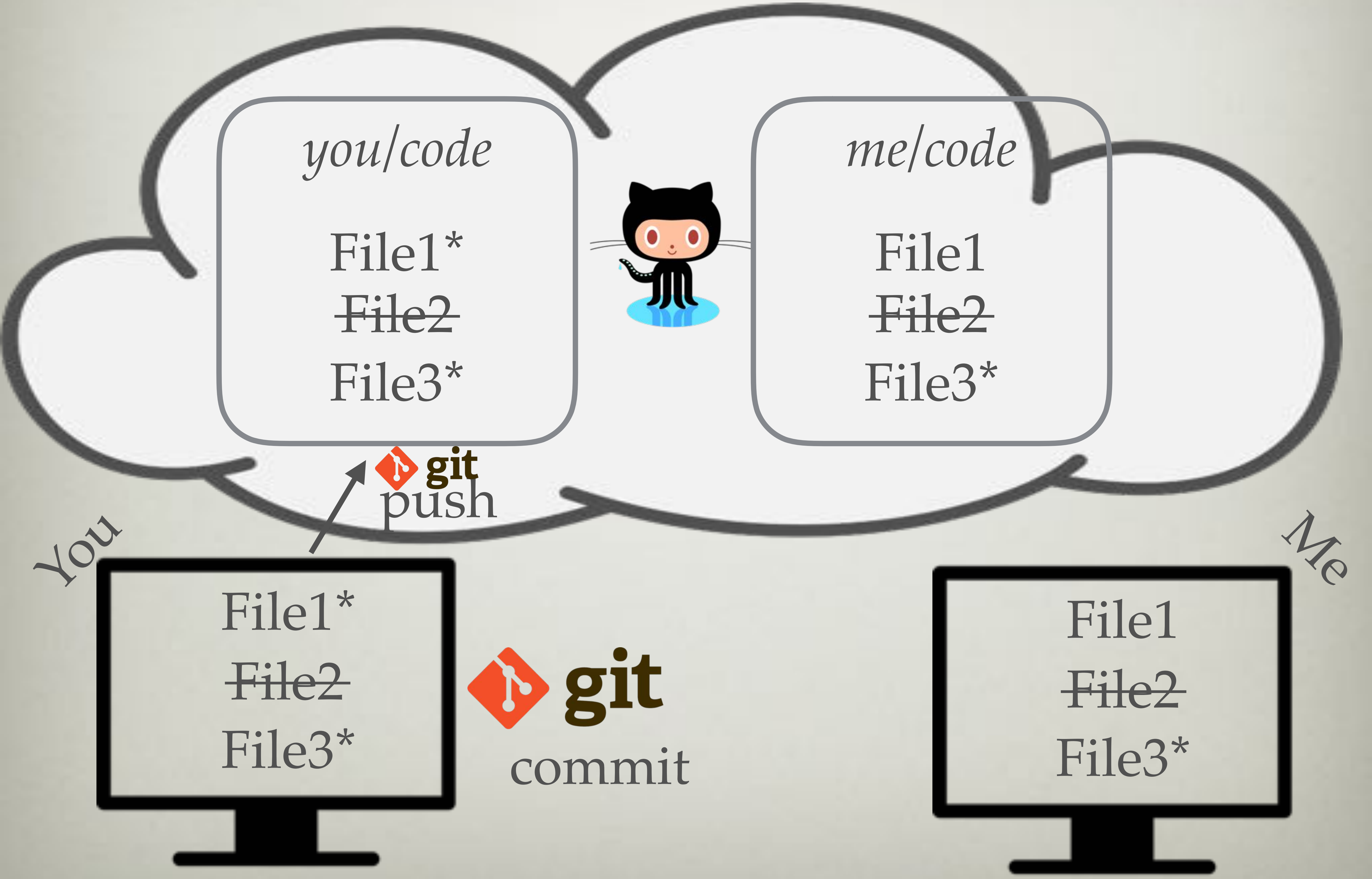
Me

File1

~~File2~~

File3*

# Lets dig down...



*you/code*

File1*
~~File2~~
File3*

*me/code*

File1
~~File2~~
File3*

git
push

You

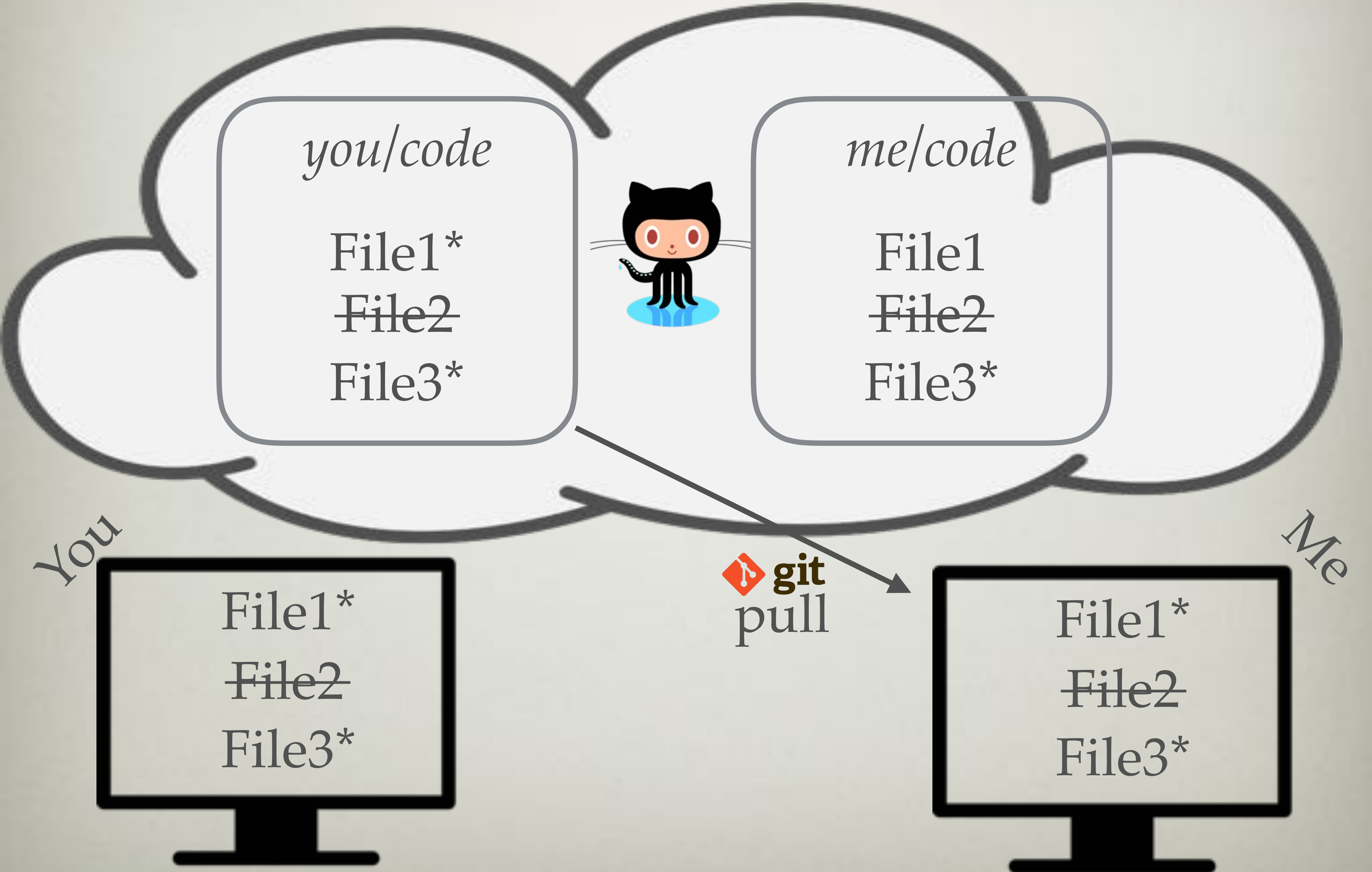
File1*
~~File2~~
File3*

git
commit

Me

File1
~~File2~~
File3*

# Lets dig down...



*you/code*

File1*
~~File2~~
File3*

*me/code*

File1
~~File2~~
File3*

You

File1*
~~File2~~
File3*

git
pull

Me

File1*
~~File2~~
File3*

# 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

# 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

# Versioning

# Versioning

- In vogue: “semantic versioning”

# 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

# 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

# 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)

# Now go do it yourself!

# Astropy's Development

# Astropy's Development