Getting Started at SciPy Sprints

Corran Webster

About Me

- Co-chair of SciPy sprints
- Using Python since ~1995
- Developer and trainer at Enthought
- Sporadic contributor to many projects

The Tools

Your Python environment

Version Control system

Project code repo

Python Environment

- If you can run the code, you've got most of what you need to develop.
- May need more tools:
  - *sphinx* for documentation
  - *nosetests* for testing
  - compilers (*gcc*, *Xcode*, *VisualStudio 2008*)
- Project docs should tell you more

Version Control

Version control software allows multiple developers to work on a codebase at the same time without breaking each other's code.

- Most projects use git
- Some use mercurial, including Python itself
- Older projects may use SVN or even CVS

Code Repo

All projects need a central repo that holds the reference copy of the codebase

- Many projects use Github
- Bitbucket, Google Code, Gitorious are alternatives
- Older projects may use SourceForge or self-hosted repositories

Most sites let you create a free user account and host open source repositories for free.

Setting up Git and Github

Create an account on Github if you don't already have one.

`github.com`

Then install git on your computer.

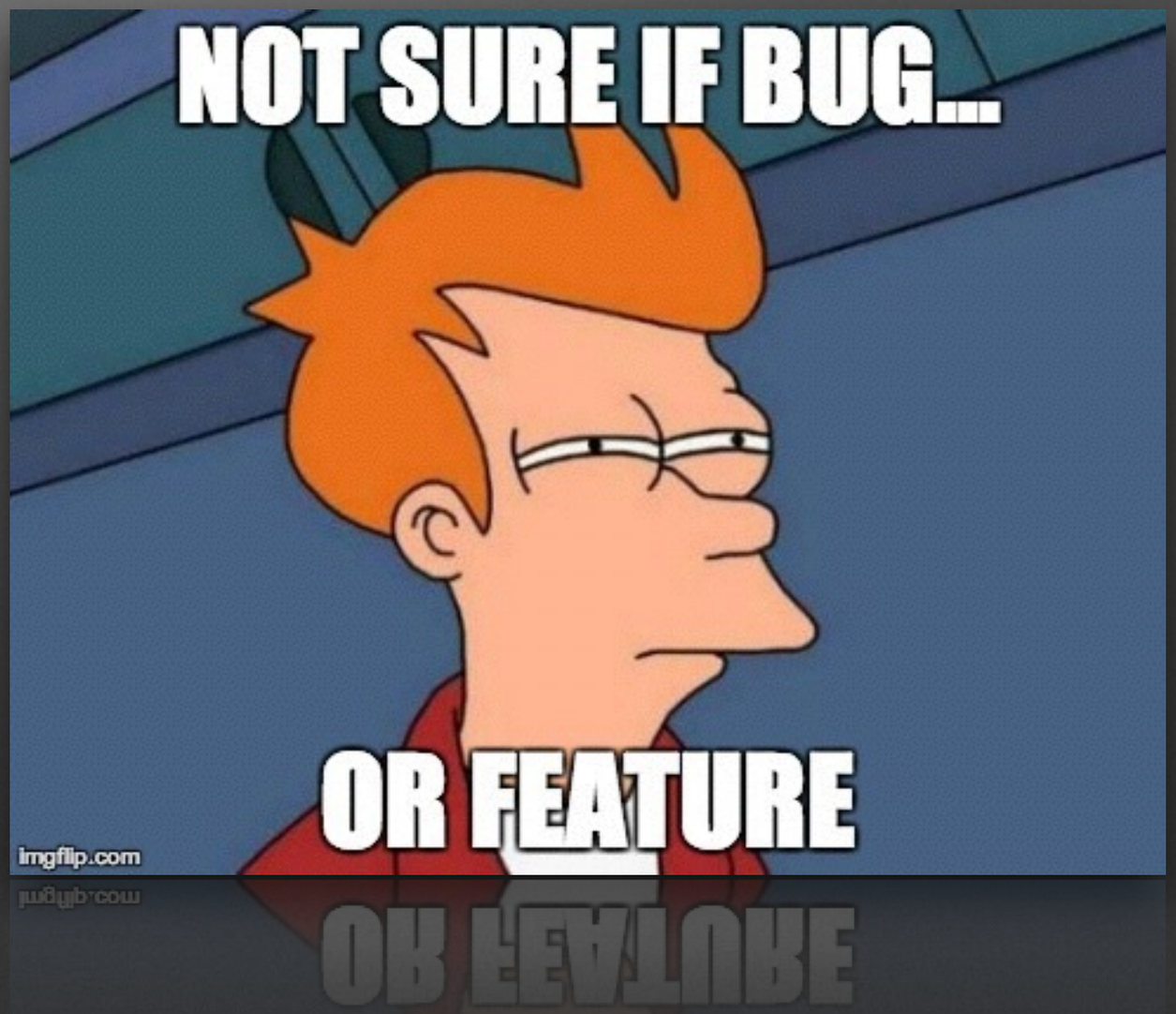
`help.github.com/articles/set-up-git`

Github also provides an app for Windows and OS X if you prefer to use that.

Set these up now.

Giving Feedback

- All software has bugs
- Letting developers know about issues that you are facing is valuable to them
- Avoid repeated notifications - look through existing issues first
- Reporting a bug doesn't mean it will be fixed!

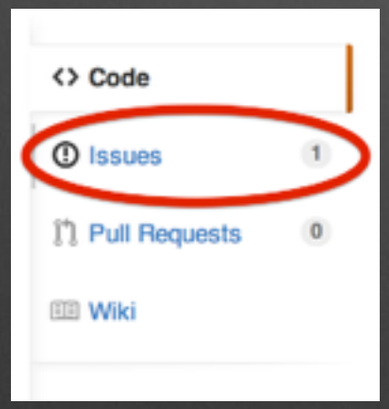


Reporting an Issue

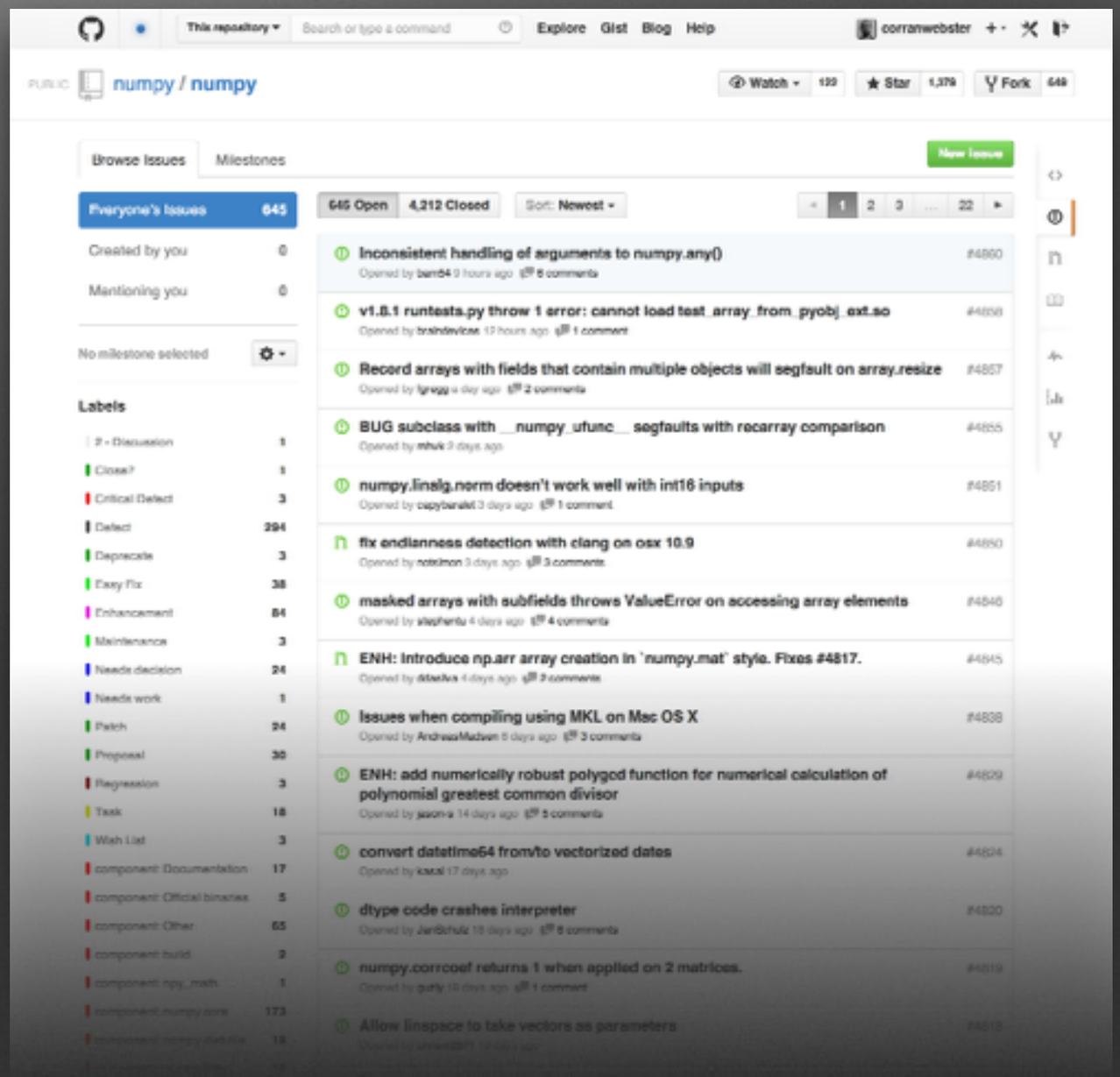
- Go to:

github.com/corranwebster/scipy-sprints-2014

- Click on the Issues button

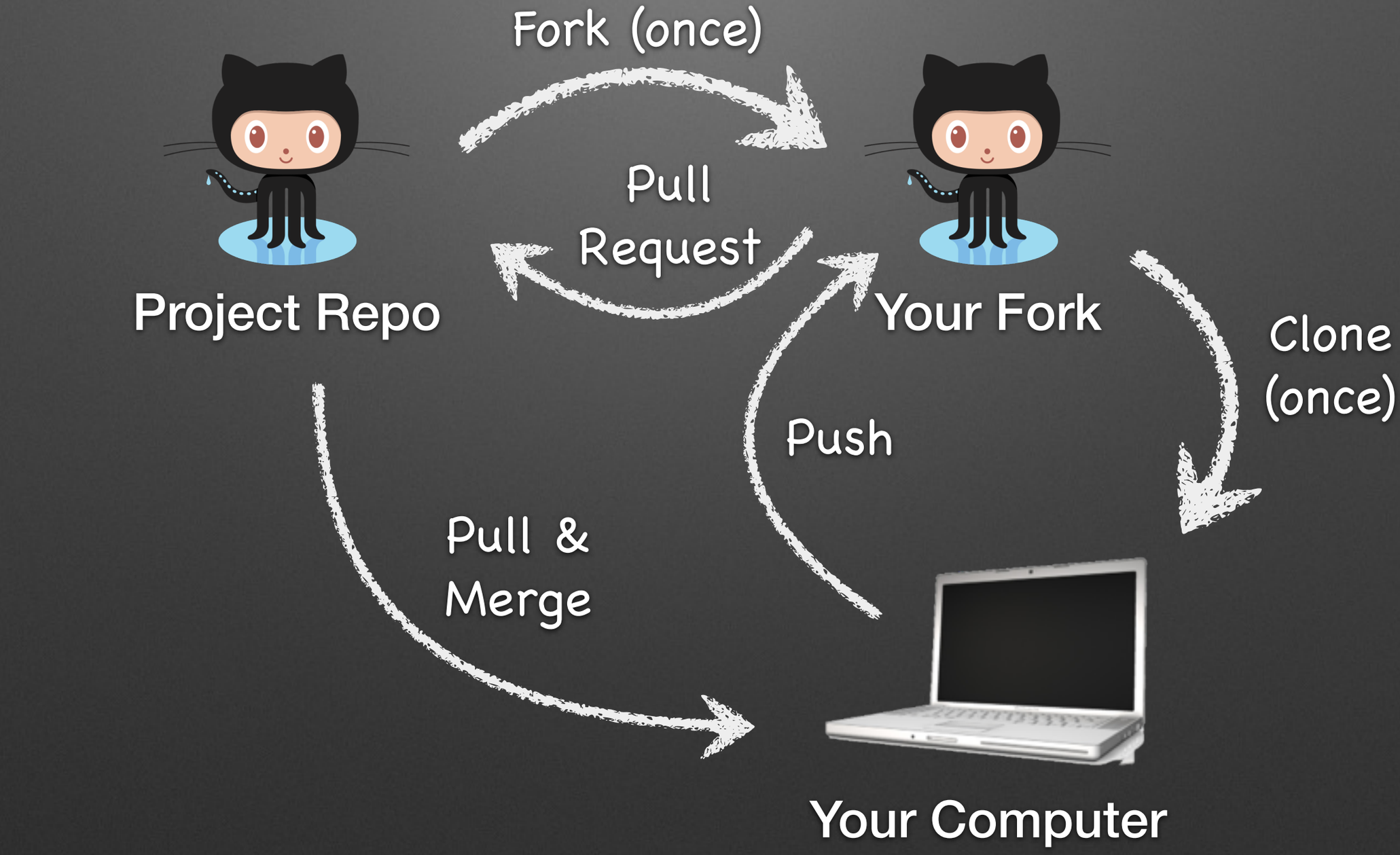


- Click “New Issue” button and suggest something



Contributing

Basic Workflow



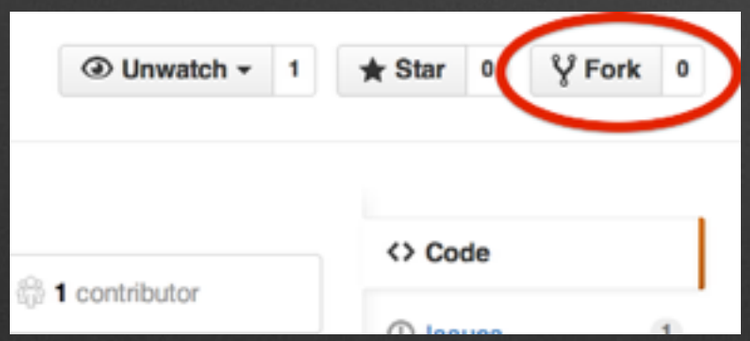
Forking a Repo

A fork of a code repo is your own personal copy of the code that you can edit freely.

- Go to:

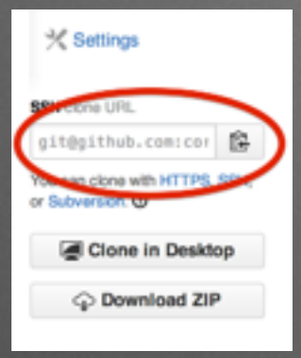
github.com/corranwebster/scipy-sprints-2014

- Click the Fork button



Clone and Develop

Clone the repo:



- copy the URL
- at the command line:

```
git clone [URL]
```

you may need to enter a password for github

- `cd` into the new directory and add the remote project repo

```
git remote add upstream [PROJECT URL]
```

Use setup.py to work with the code:

```
python setup.py develop
```

This adds the packages to your sys.path so you can import from your cloned repo.

Changes you make to python code are live when you next run the code.

Building Documentation

- If you are lucky, docs are built by setup.py
- Otherwise, Sphinx documentation can be built by finding the docs directory and doing either

```
make html
```

or

```
make.bat html
```

Running Tests

- Tests can usually be run as stand-alone modules
- it is often more convenient to use the nose project's tools to "sniff out" all the tests and run them at once
- the command to run nose is

```
nosetests
```

- nose has many command-line options. Do `nosetests -h` to see them all.

Branch and Edit

You usually want to work on a *branch* within your repo. This lets you keep the original code around.

- to create a branch

```
git branch [branch]
```

- to switch to a branch

```
git checkout [branch]
```

- to do both at once

```
git checkout -b [branch]
```

Now you can edit and run the code until you are happy with the result.

When you have reached a good checkpoint, for each changed file do:

```
git add [file]
```

and then after you have added all the files:

```
git commit
```

Push and Pull

- When you are ready to share your new code, do

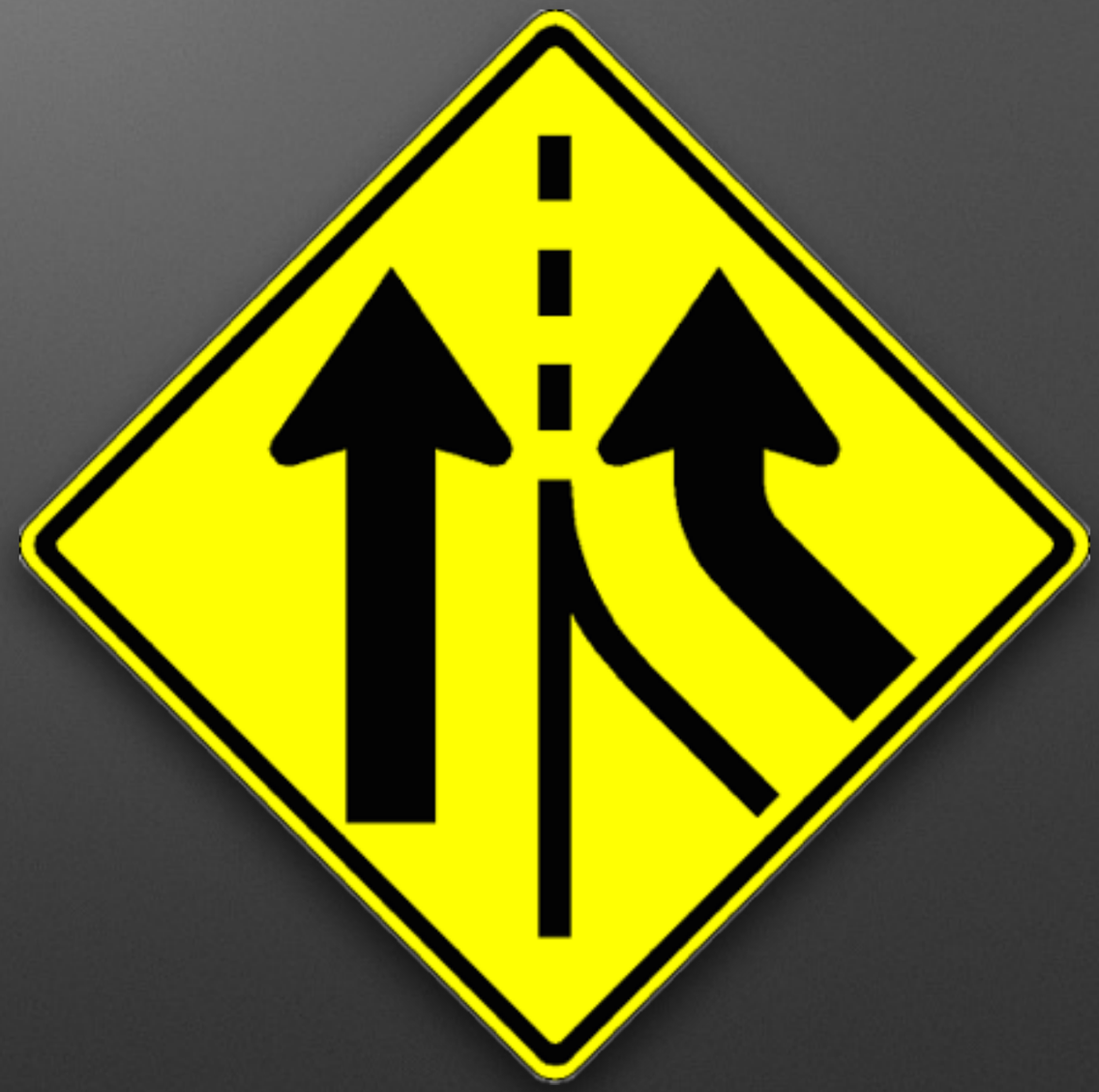
```
git push origin [branch]
```

- to get updates from the project repo, checkout the branch you want to update and do

```
git pull upstream [branch]
```

Merging

When you pull some files may have incompatible changes. You may need to manually merge them.



`git status`

shows which files need work.

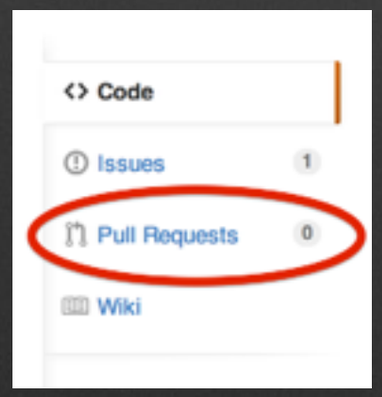
Edit those files, then `git add` and `git commit` the changes.

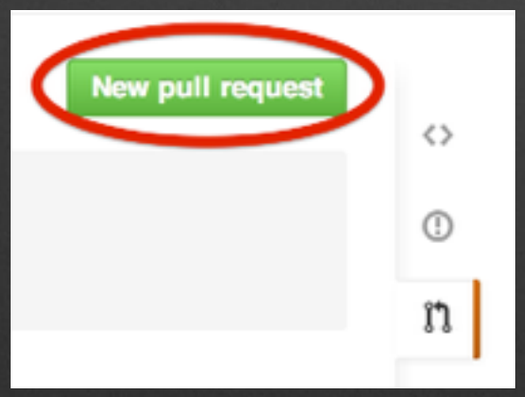
Pull Requests

You have an awesome bugfix or feature for your favorite project. It's documented, tested, and the tests all run and pass.

Now its time to make a pull request.

Go to your branch or your repo on Github and create one.





Peer Review

Hopefully someone on the project dev team will review your pull request.

If it is simple and good, they may just merge it.

They may comment on it, suggest fixes, improvements or changes.

You can edit, add, commit and push updates to your branch, and they will be included in the pull request.

And they may reject your pull request or ignore it.

Thank you!