Science & Technology Facilities Council | UK Research and Innovation | NERC SCIENCE OF THE ENVIRONMENT

# The Unix Shell

## Managing your code: quietly introducing *Git* - a friend for life - Part 2

Thanks to all contributors:

Alison Pamment, Sam Pepler, Ag Stephens, Stephen Pascoe, Kevin Marsh, Anabelle Guillory, Graham Parton, Esther Conway, Eduardo Damasio Da Costa, Wendy Garland, Alan Iwi, Matt Pritchard and Tommy Godfrey.

# Managing code in the olden days

- Create "*working_dir*"...add some code
- Write some outputs...change the code
- Publish a paper...change the code
- Copy "*working_dir*" to "*working_dir2*"
- Change the code
- Copy a version to a CD

*...now which version is current? Is it "working_dir" or "working_dir2"? And which one relates to that paper?*

# But those days are gone!

- Scientists are typically **required to publish data and code** (by their funders/institutions).
- Collaboration between scientists requires data-sharing; this implicitly relies on **code-sharing**.
- There are **tools that make it easy** to record our changes, document our workflow and "fix" releases of our code at important steps along the way.

# Introducing Git

There are many different Version Control tools:

- **SVN** (Subversion) is very popular and (relatively) easy to grasp; eclipsed by…
- **Git,** which is also:
  - More useful for collaboration
  - Distributed and *fast*
  - Very well supported in terms of tooling
  - Has free repository hosts  on the web (GitHub, BitBucket etc.,).

# More about Git

Git is a *distributed* Version Control System (VCS):

- you store a complete copy of a repository within your working copy.
- this means you can work offline:
  - there is no default 'central' server - if you want one, you (and your team) just nominate where it is - typically GitHub!

# What is a Git repository?

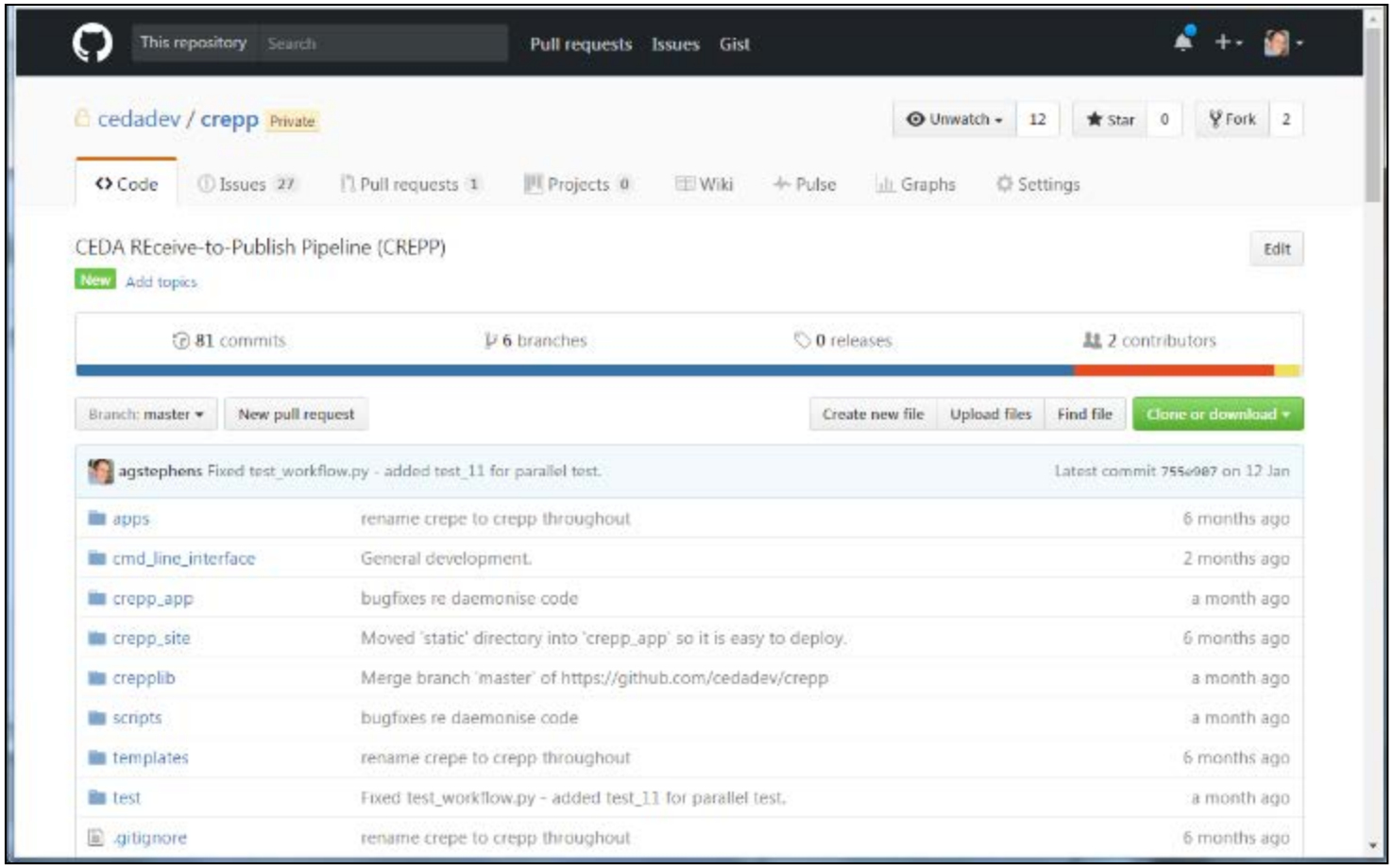
- A directory tree containing files and subdirectories.
- Old and different versions of those files and subdirectories.
- A set of information to enable you to navigate across versions.

# Not Introducing GitHub

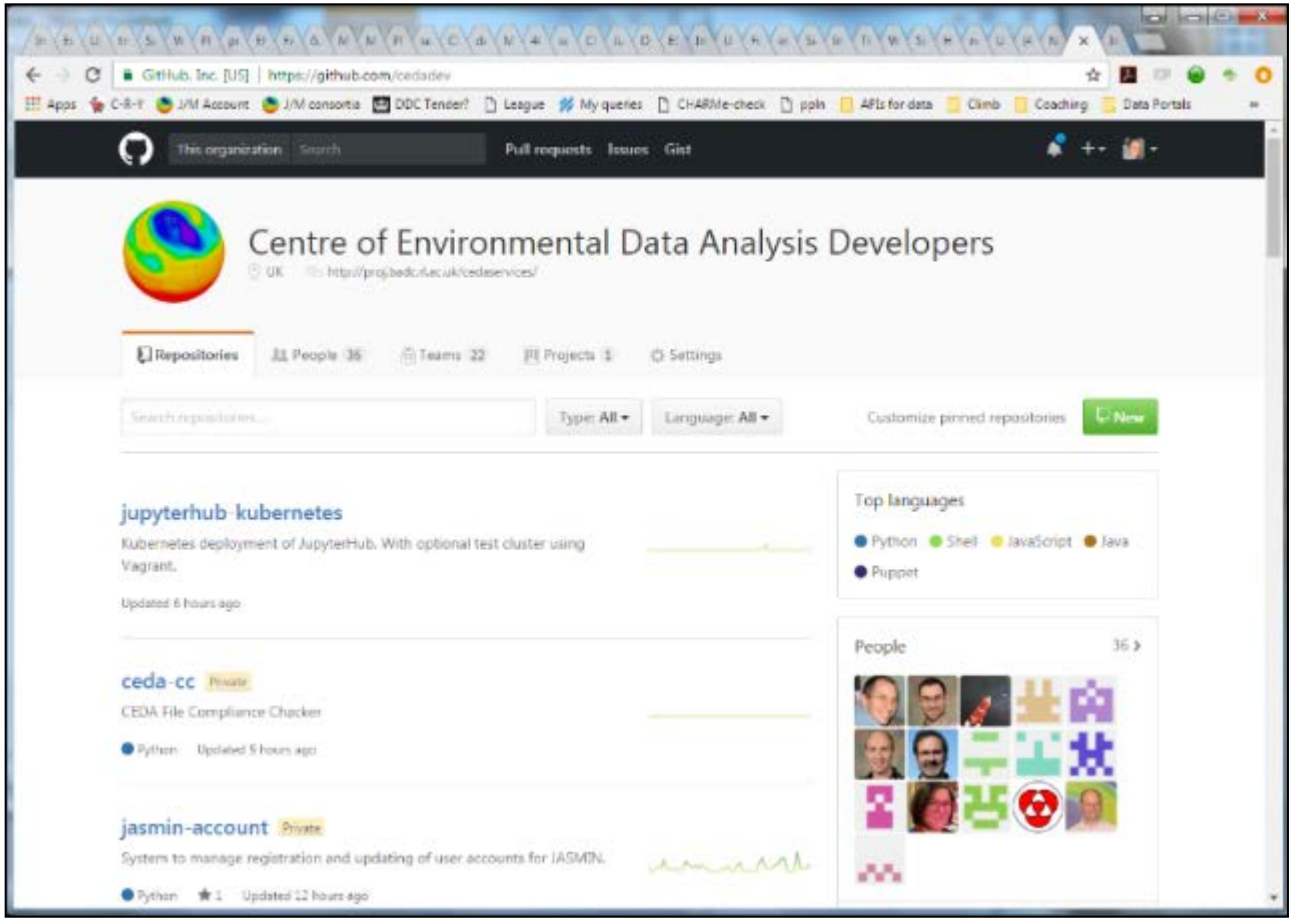
https://github.com

A service for hosting git repositories.

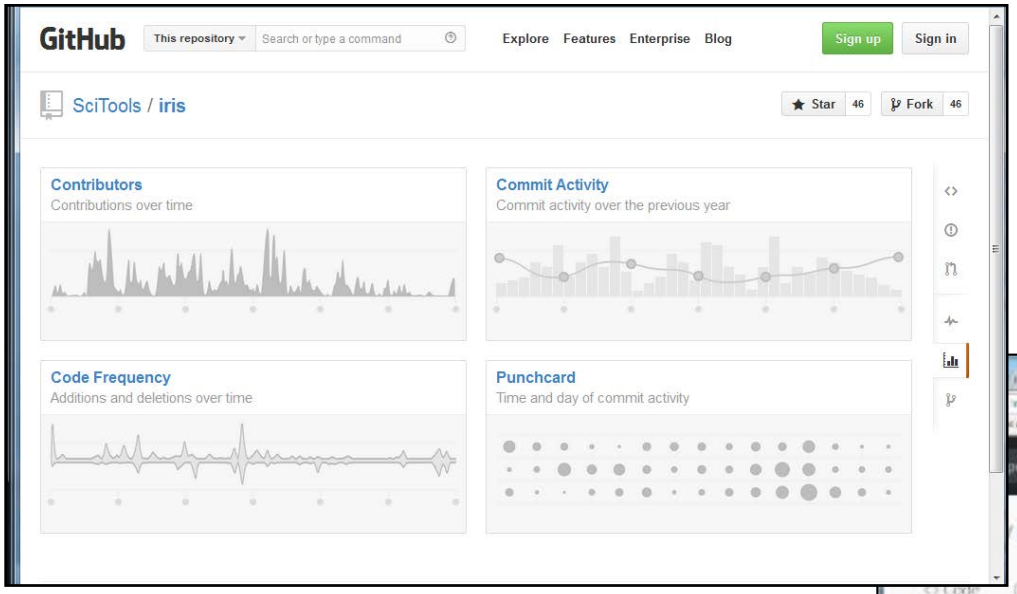
# GitHub: repositories (public *or* **private**)

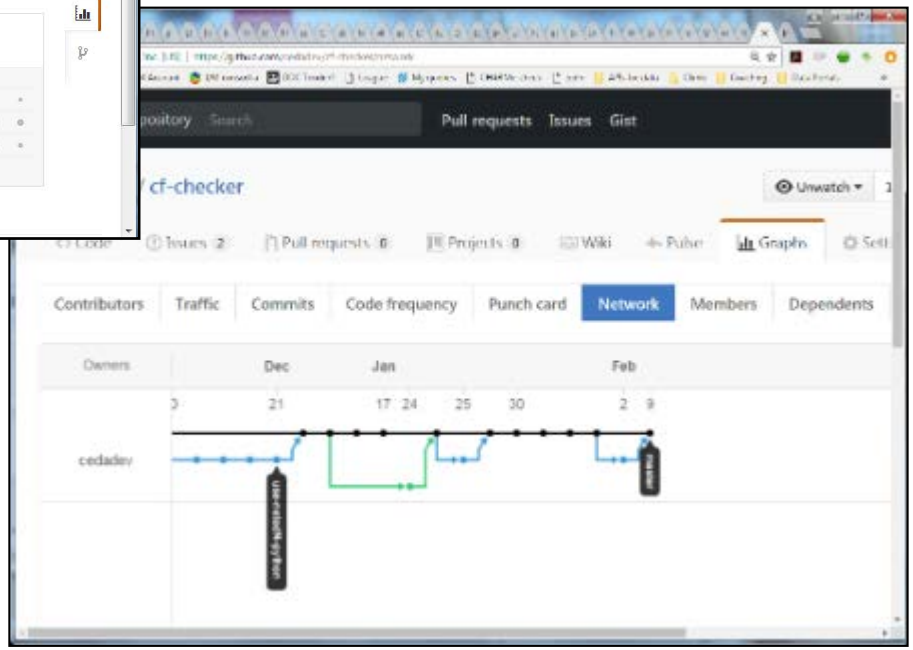


# GitHub: organisations

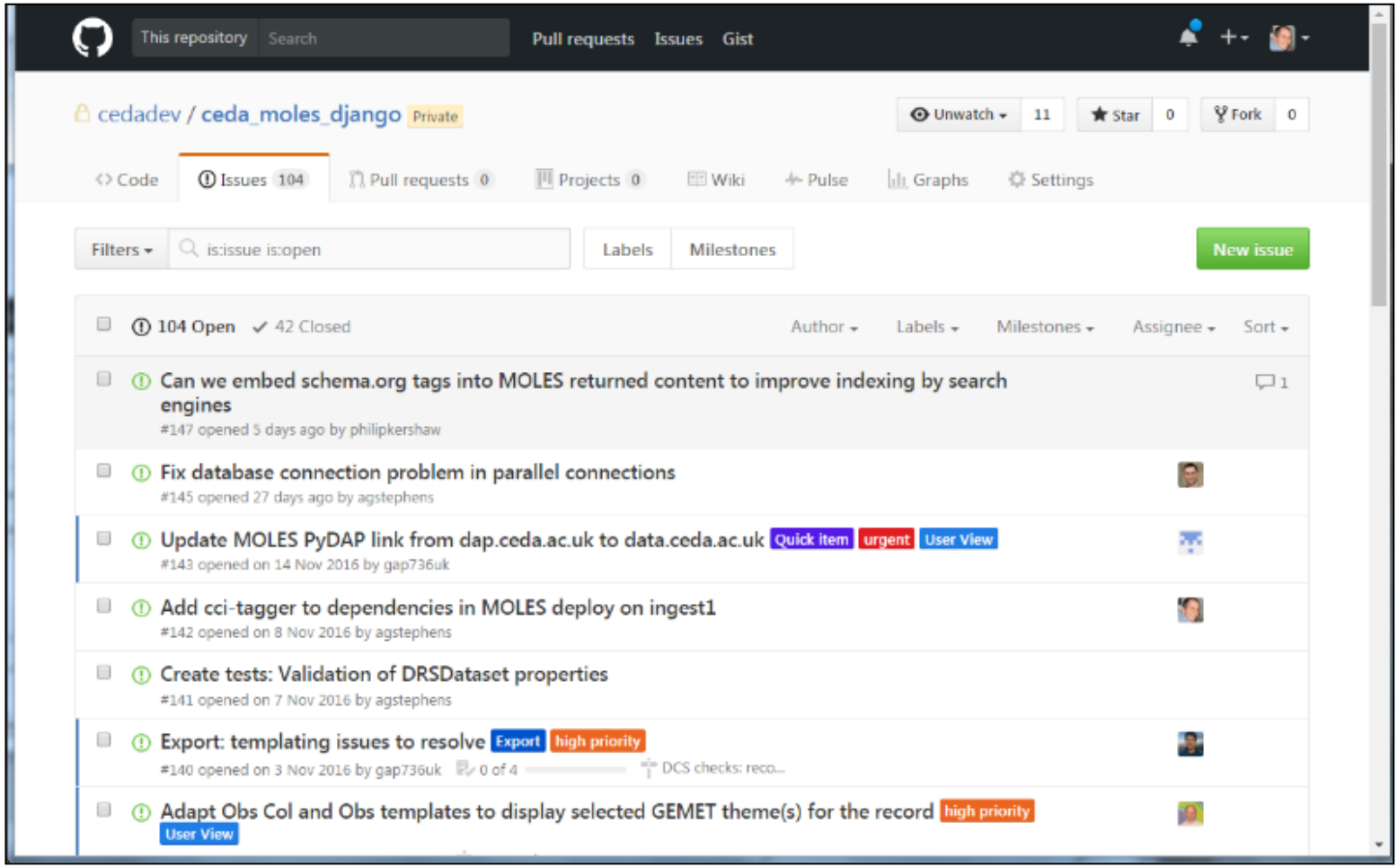


# GitHub: collaboration (branch/fork)



# GitHub: Issue tracking



# GitHub: history and change

6 cedamoles_app/admin_tools/integrity/routine_checks.py

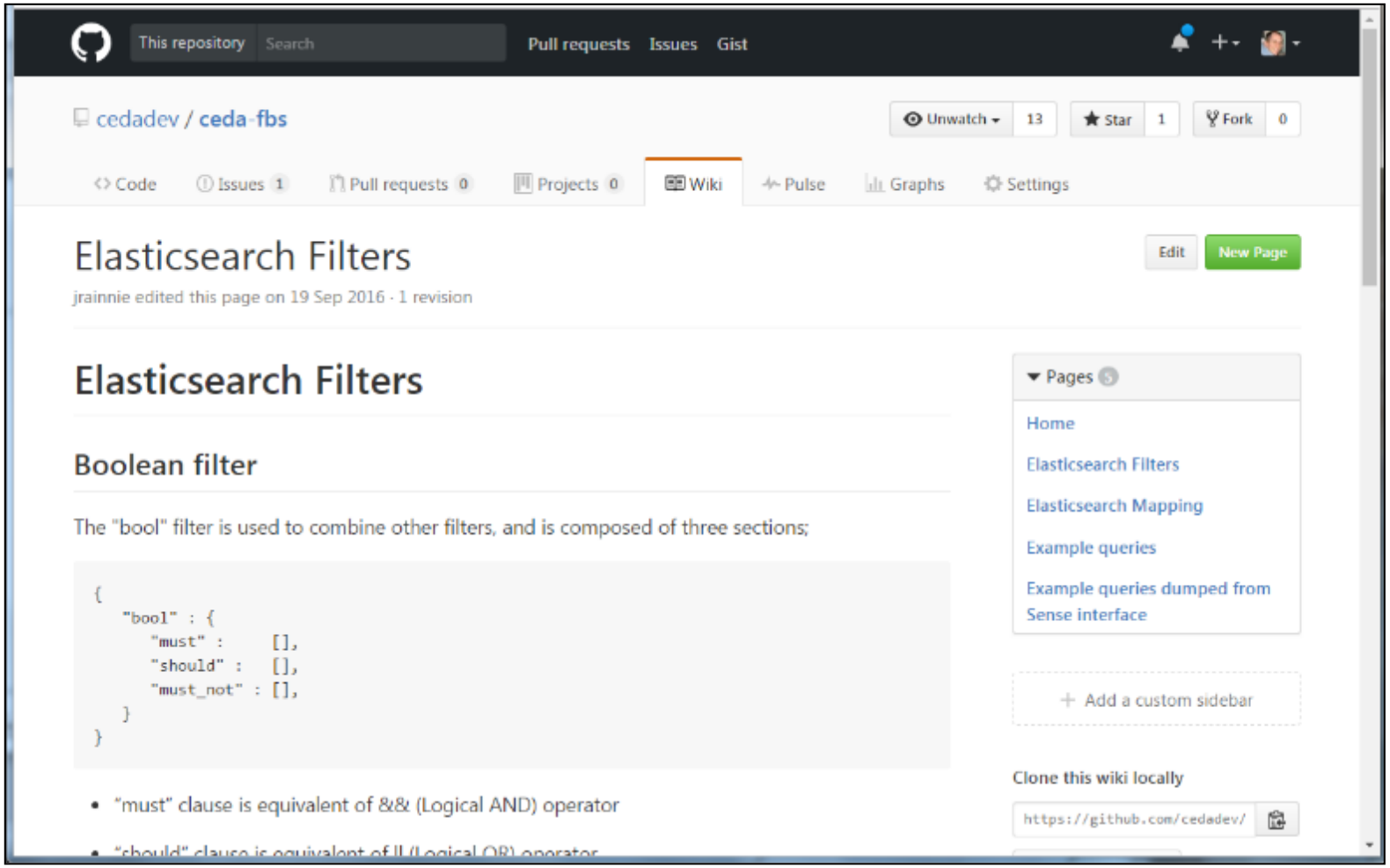
```
@@ -41,10 +41,10 @@ def run_checks(self):

 class ResultChecks(ChecksBase):

-    def check_internalPath(self):
+    def check_dataPath(self):
         found = Counter()
         for result in Result.objects.all():
-            path = result.internalPath
+            path = result.dataPath
             found.update([path])

         dupes = [(path, count) for (path, count) in found.items() if count > 1]
@@ -53,7 +53,7 @@ def check_internalPath(self):
         for path, count in dupes:
             print path, count
```

# GitHub: wikis

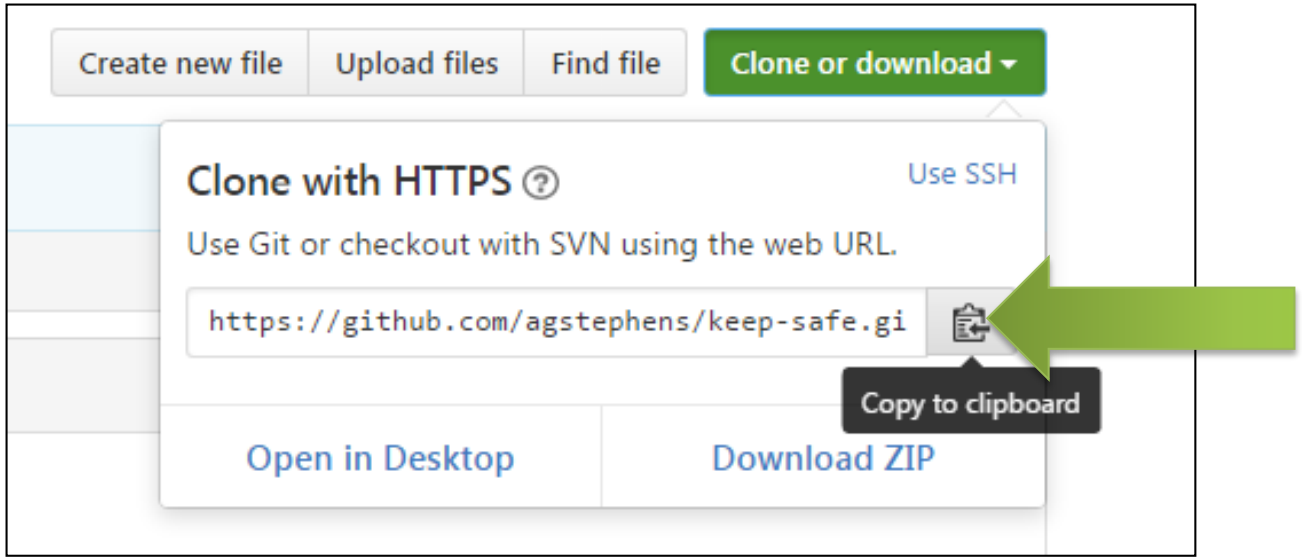


# GitHub does lots of funky things, but…

- On this course we are going only using it as a remote repository.
- We are going to concentrate on simply using git.

# Where to start 1: `git clone`

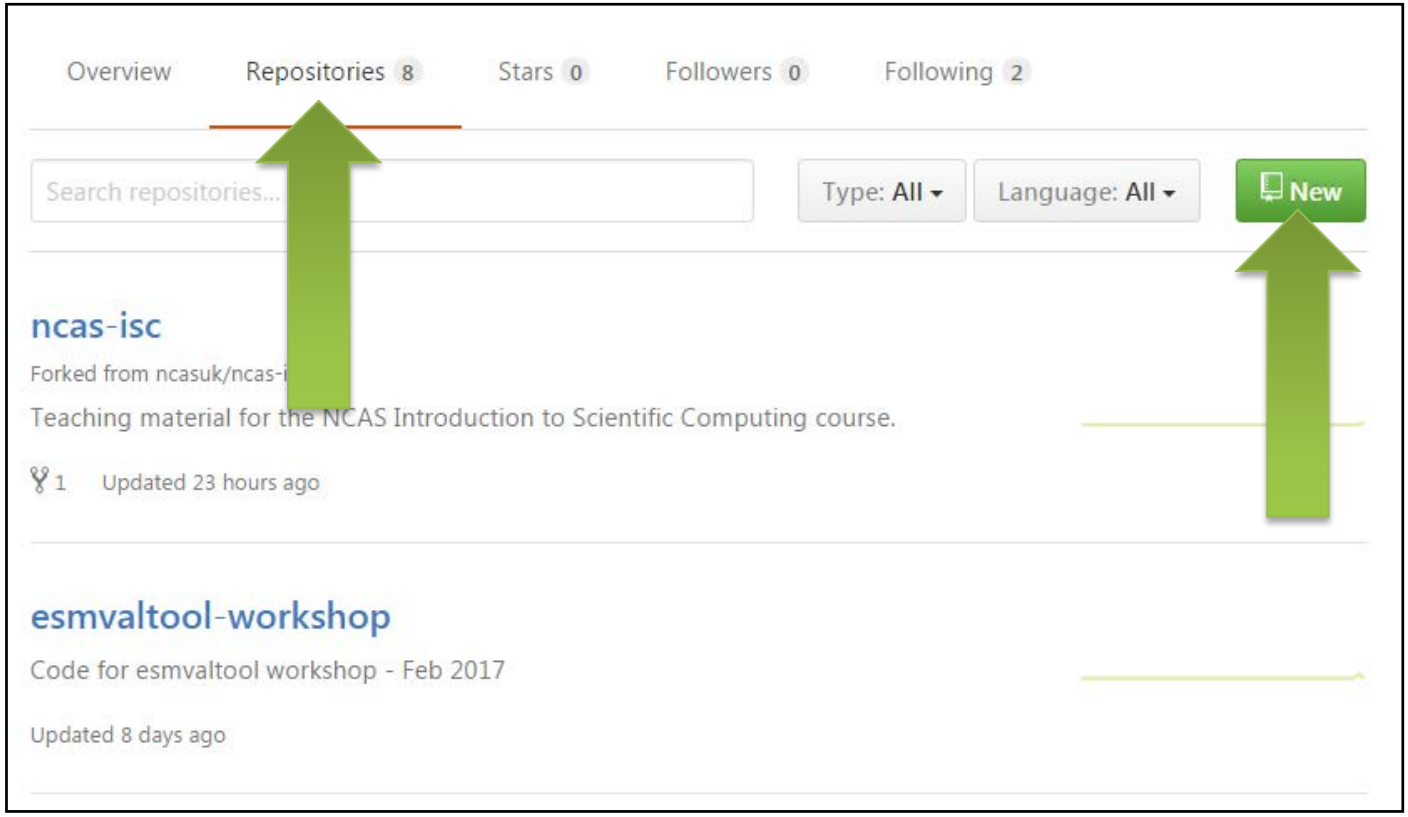
This makes a copy of a repository locally. We did this at the start of the course.




```
$ git clone
  https://github.com/agstephens/keep-safe
```

# Where to start 2: Create a repository on GitHub

Navigate to "Repositories" and click "New".



# Where to start 3: start a new repository from existing files

```
$ ls
x       y       z
$ git init
Initialized empty Git repository in
/Users/sjp23/play/york_workshop_shell/test-pakage/.git/
$ git add .
$ git commit –m'Initial commit from existing files'
[master (root-commit) 71ecfcf] Initial commit from
existing files
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 x
 create mode 100644 y
 create mode 100644 z
```

# Add a file to your local repo

1. Enter the repository directory:

```
$ cd ncas-isc
```

2. Create a new file:

```
$ echo "hello world" > hello.txt
```

3. Tell Git about the file:

```
$ git add hello.txt
```

4. Commit the file to the **local** Git repository:

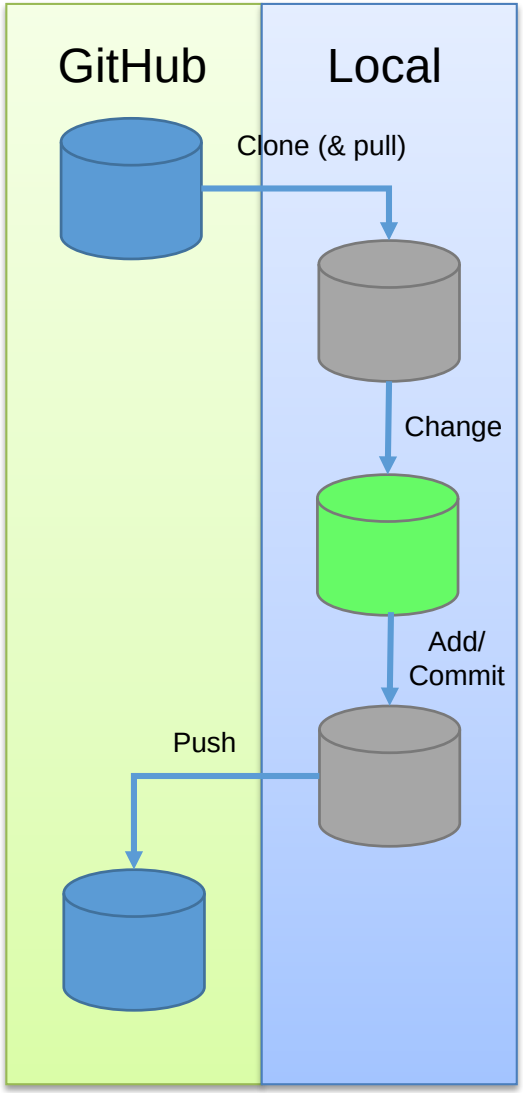
```
$ git commit -m "added hello"
```

5. Push any updates to the **remote** GitHub repo:

```
$ git push
```

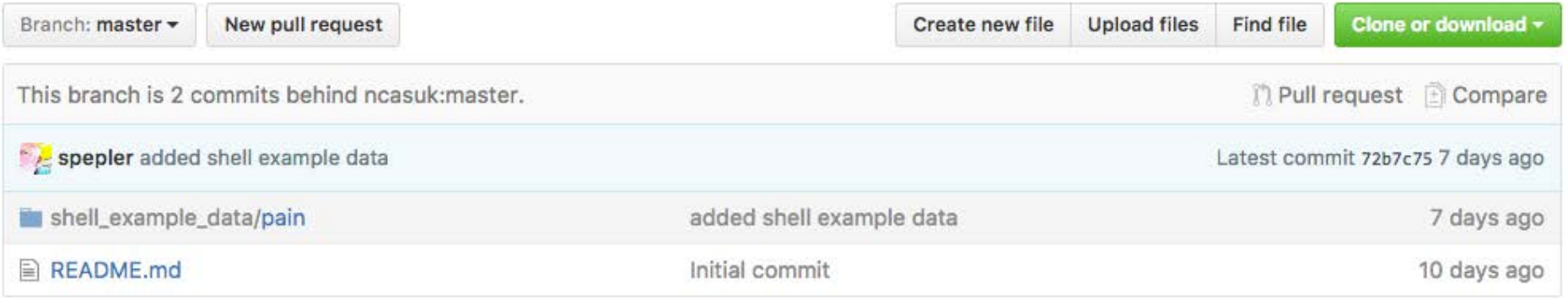
# So, what just happened?

- We *cloned* the remote repository to our file system.
  - Now there are two identical copies of one repo.
- We *created* a new text file.
- We *added* and *committed* that new file to the local version of the repo.
- We used *push* to update the remote repo.

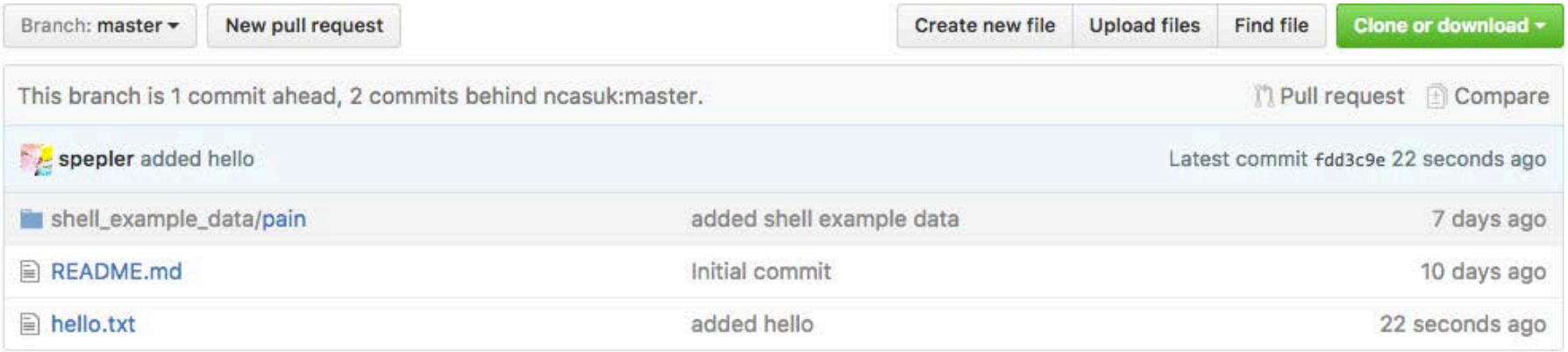


# Let's look on GitHub

## Before…




## After…



# The Plan: Use git / GitHub all week

- This stuff is hard to learn - we know that from experience.
- A presentation is quickly forgotten.
- So, we propose that you use Git/GitHub for every exercise.
- You are going to create and update your own Github repository with files from exercises throughout the course.

# Lets make some test files

```
$ mkdir mydir
$ echo "hi" > hi.txt
$ echo "testing..." > mydir/t1.txt
$ ls
hi.txt     hello.txt   mydir
```

# git Status

Use to see what stage files are at

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
  (use "git add <file>..." to include in what will be
committed)

        hi.txt
        mydir/

nothing added to commit but untracked files present (use
"git add" to track)
```

# git add

Adding files tells git to start looking after them or add a new version if it already knows about it.

```
$ git add .
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   hi.txt
        new file:   mydir/t1.txt
```

# Add another file

These files are all staged to go into the repository, but are not committed yet.

```
$ echo "testing..." > mydir/t2.txt
$ git add mydir/t2.txt
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   hi.txt
        new file:   mydir/t1.txt
        new file:   mydir/t2.txt
```

# Lets commit

Now the files are in the local repository. The working tree is the same as repository.

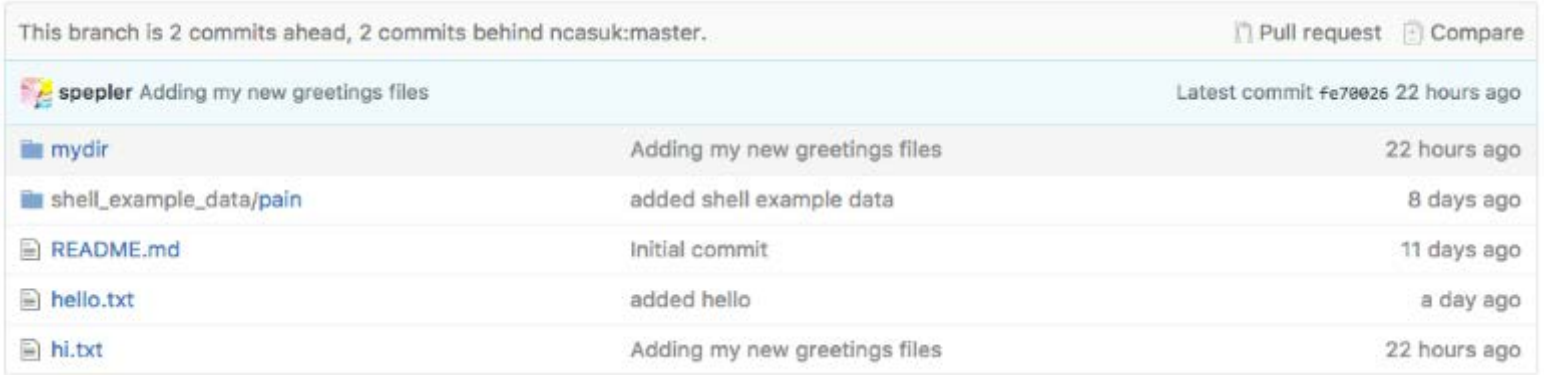
```
$ git commit -m 'Adding my new greetings files'
[master fe70026] Adding my new greetings files
 3 files changed, 3 insertions(+)
 create mode 100644 hi.txt
 create mode 100644 mydir/t1.txt
 create mode 100644 mydir/t2.txt

$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)
nothing to commit, working tree clean
```

# Push the new version back to GitHub

Make the repo on GitHub match the local repo.

```
$ git push
Counting objects: 5, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 465 bytes | 0 bytes/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To github.com:spepler/ncas-isc.git
   fdd3c9e..fe70026  master -> master
```



# Enough?

- If you are working on your own then that is all you need to know.
- You can keep track of changes in your code, you know its safe and you can share it with people.

# Working with other people

```
$ git clone git@github.com:spepler/ncas-isc.git ncas-isc2
Cloning into 'ncas-isc2'...
remote: Counting objects: 17, done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 17 (delta 1), reused 16 (delta 0), pack-reused
0
Receiving objects: 100% (17/17), done.
Resolving deltas: 100% (1/1), done.
$ cd ncas-isc2
$ ls
hello.txt     hi.txt mydir
$ emacs hello.txt
$ cat hello.txt
hello world
New line
```

Red Fred clones a copy of the repository and changes a file

# They commit their changes and push back to GitHub

```
$ git add hello.txt
$ git commit -m 'added new line'
[master d274491] added new line
 1 file changed, 1 insertion(+)
$ git push
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 283 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local
objects.
To github.com:spepler/ncas-isc.git
   fe70026..d274491  master -> master
```

# Black Bob downloads changes using git pull

```
$ git pull
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (1/1), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (3/3), done.
From github.com:spepler/ncas-isc
   fe70026..d274491  master     -> origin/master
Updating fe70026..d274491
Fast-forward
 hello.txt | 1 +
 1 file changed, 1 insertion(+)
```

# Black Bob looks at change log

```
$ git log hello.txt
commit d274491d34d96aa92eb110e472006070e537dda0
Author: Sam Pepler <sam.pepler@stfc.ac.uk>
Date:   Fri Feb 24 12:26:47 2017 +0000

    added new line

commit fdd3c9eb7cbea69cce46ea22326ed5c801bb75f8
Author: Sam Pepler <sam.pepler@stfc.ac.uk>
Date:   Thu Feb 23 11:13:13 2017 +0000

    added hello
```

# Exercise

- Clone the repository we made yesterday:

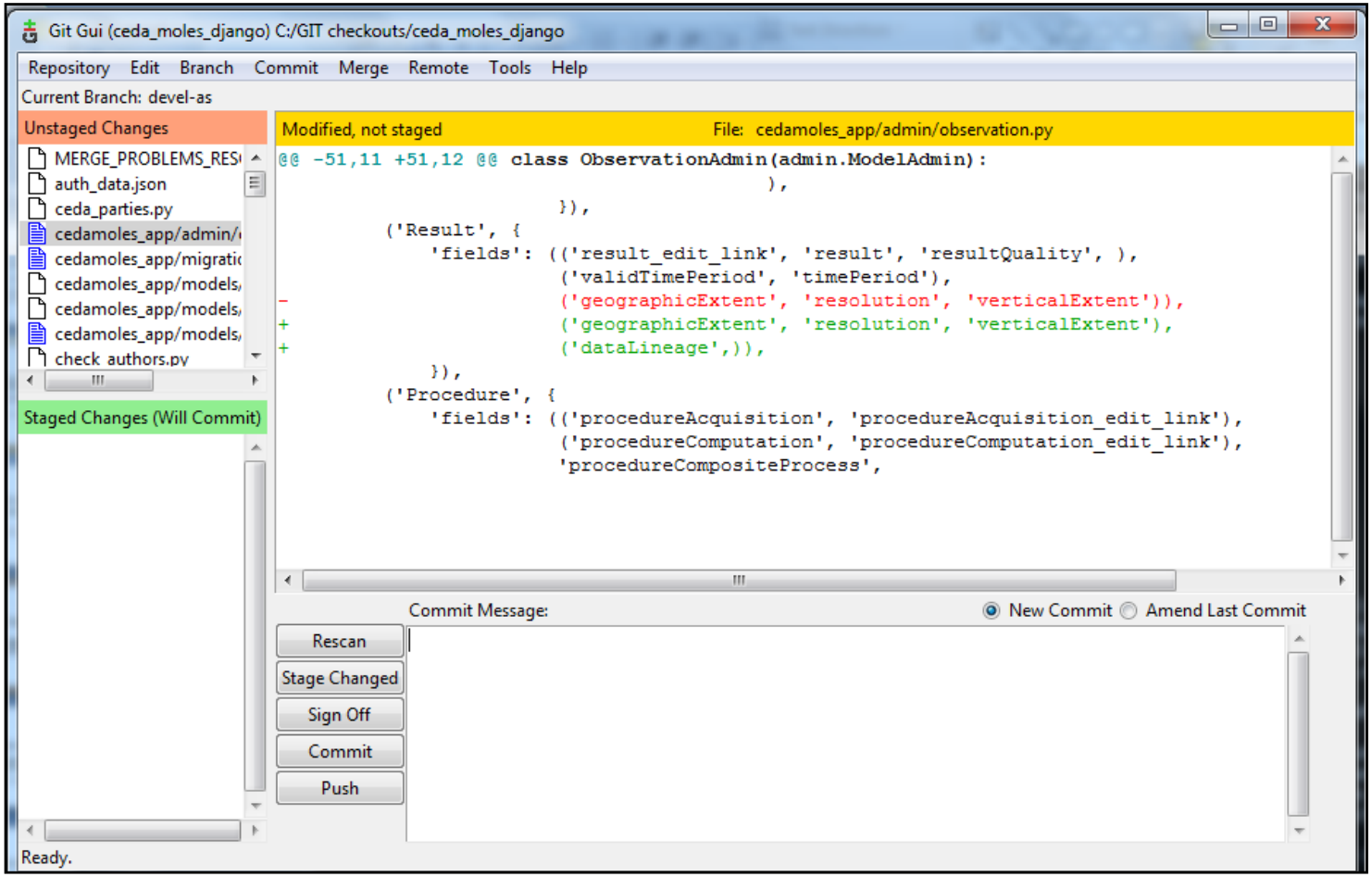
```
$ cd
$ git clone
  https://github.com/<username>/my-isc-work
```

- Make a new directory in your cloned repo and a new file containing a few lines of text.
- Add the file and directory to your local repo: **`$ git add <newfile> <newdir>`**
- Commit the changes: **`$ git commit -m 'Add some test files'`**
- Update Github repo: **`$ git push`**

# Other tools in the Git ecosystem

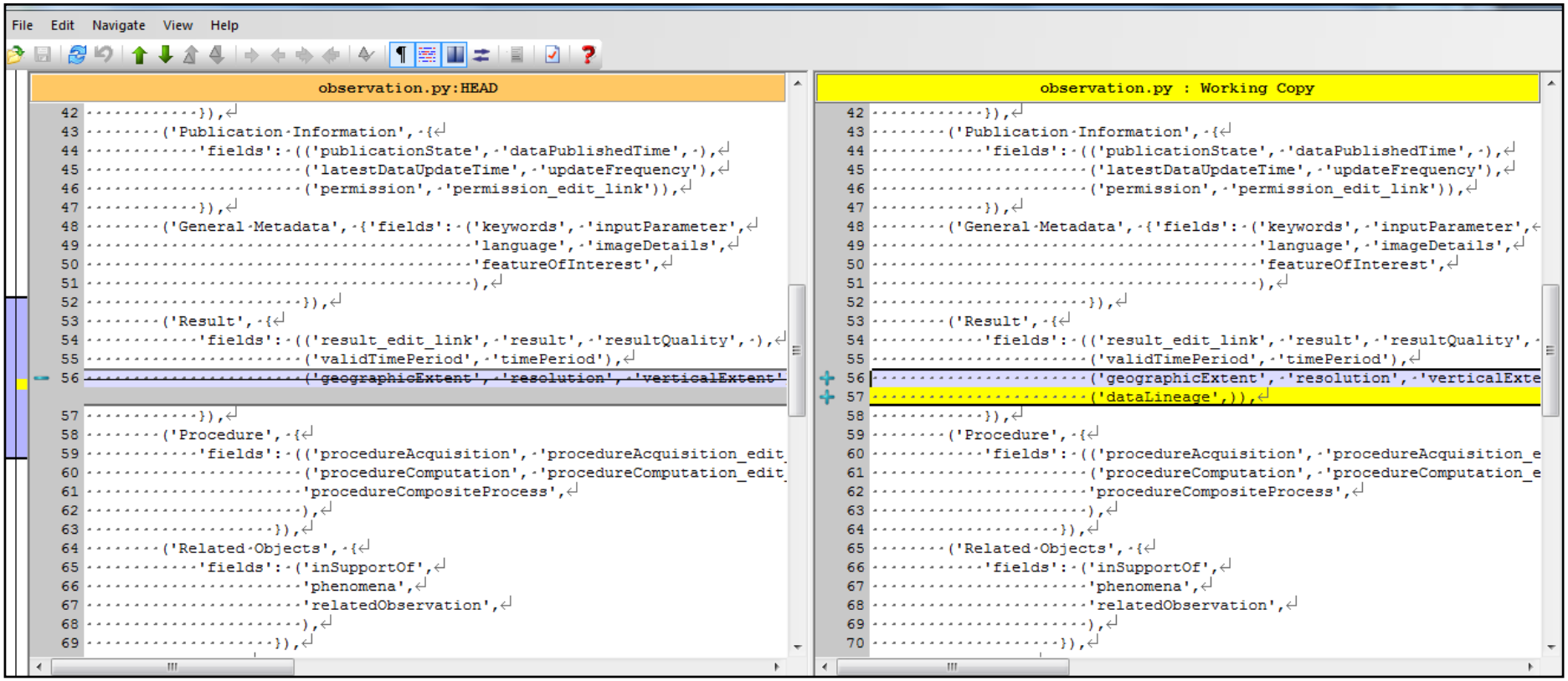
# git gui

Full set of GUIs for interacting with local and remote repos.



# TortoiseGIT (for Windows)

Provides GUIs for adding/committing/changing - including a side-by-side diff…



# The *why* - one more time

- Using version control will **save you time** – *No more accidentally deleting your workspace, or working on the wrong version of a file.*
- It will **make you a better programmer** – *It encourages good working practices: such as documenting change.*
- It will help you **collaborate more effectively** – *Others can access tagged releases of your code.*
- It will **boost your scientific integrity** – *Helping you document your work; aiding reproducibility.*
- It will **make you feel safe** – *No more waking up at 3 a.m. wondering if you backed up your work!*

# The NCAS GitHub organisation

An NCAS GitHub organisation has been set up.

This allows repositories to be set up that where users could share code when it has become a more formal collaboration.

If you want to become part of the NCAS GitHub please contact Ag, James or Dan and send them your GitHub account ID.

# Further information

Git documentation:

http://git-scm.com/documentation

Nice Git reference:

http://gitref.org/

GitHub:

http://github.com/

# Acknowledgements

We would like to Acknowledge the following authors for some of the content presented here:

*"Introduction to GIT".* Lukas Fittl (http://fittl.com).

"*Git and GitHub*". Darren Oakley.