

GIT Tricks

1

GET OUT OF JAIL FREE - WITH GIT!

Cloning a repository with MPLAB and GIT

The image displays three overlapping screenshots of the 'Clone Repository' dialog box in the MPLAB IDE, illustrating the steps for cloning a Git repository.

Top Screenshot (Remote Repository Step):

- Steps:** 1. Remote Repository, 2. Remote Branches, 3. Destination Directory.
- Remote Repository:** Specify Git Repository Location. Repository URL: . User: . Password: . Proxy Configuration... button.
- Specify Destination Folder:** Clone into: (Leave empty to specify a new directory).

Middle Screenshot (Remote Branches Step):

- Steps:** 1. Remote Repository, 2. Remote Branches, 3. Destination Directory.
- Remote Branches:** Select Remote Branches. ☒ Simple1-dev, ☒ master*. Select All, Select None buttons.
- Message:** Clone name can't be empty

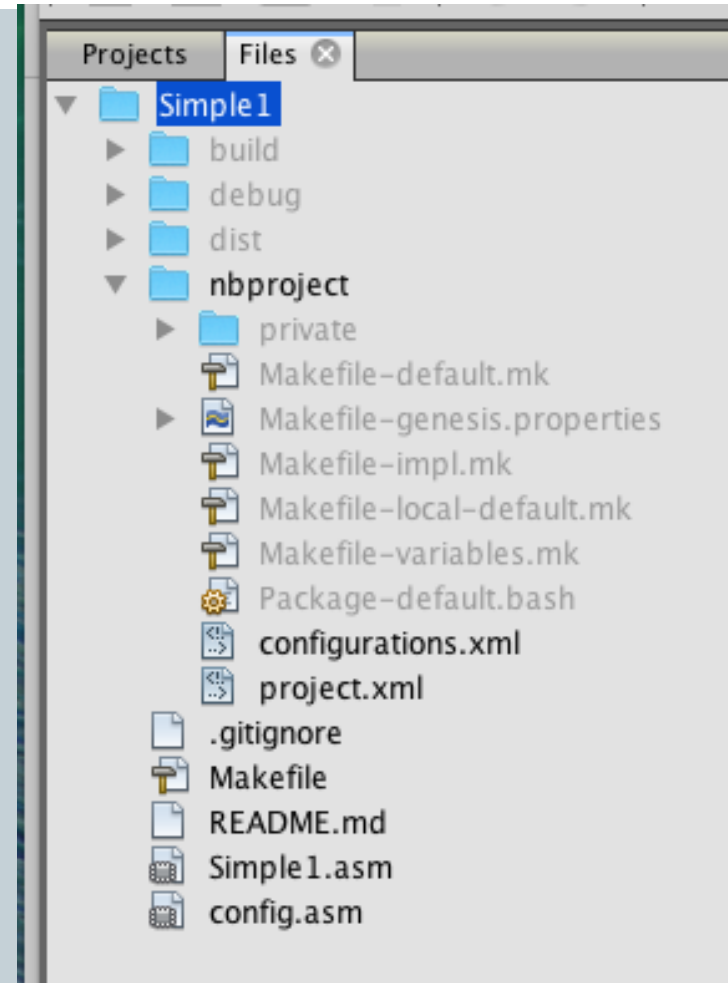
Bottom Screenshot (Destination Directory Step):

- Steps:** 1. Remote Repository, 2. Remote Branches, 3. Destination Directory.
- Destination Directory:** Specify the Parent Directory and Name for this Clone. Parent Directory: Browse... button. Clone Name: . Checkout Branch: . Remote Name: . ☒ Scan for NetBeans Projects after Clone.
- Buttons:** < Back, Next >, Finish, Cancel, Help.

What files are in your GIT repository and in MPLAB

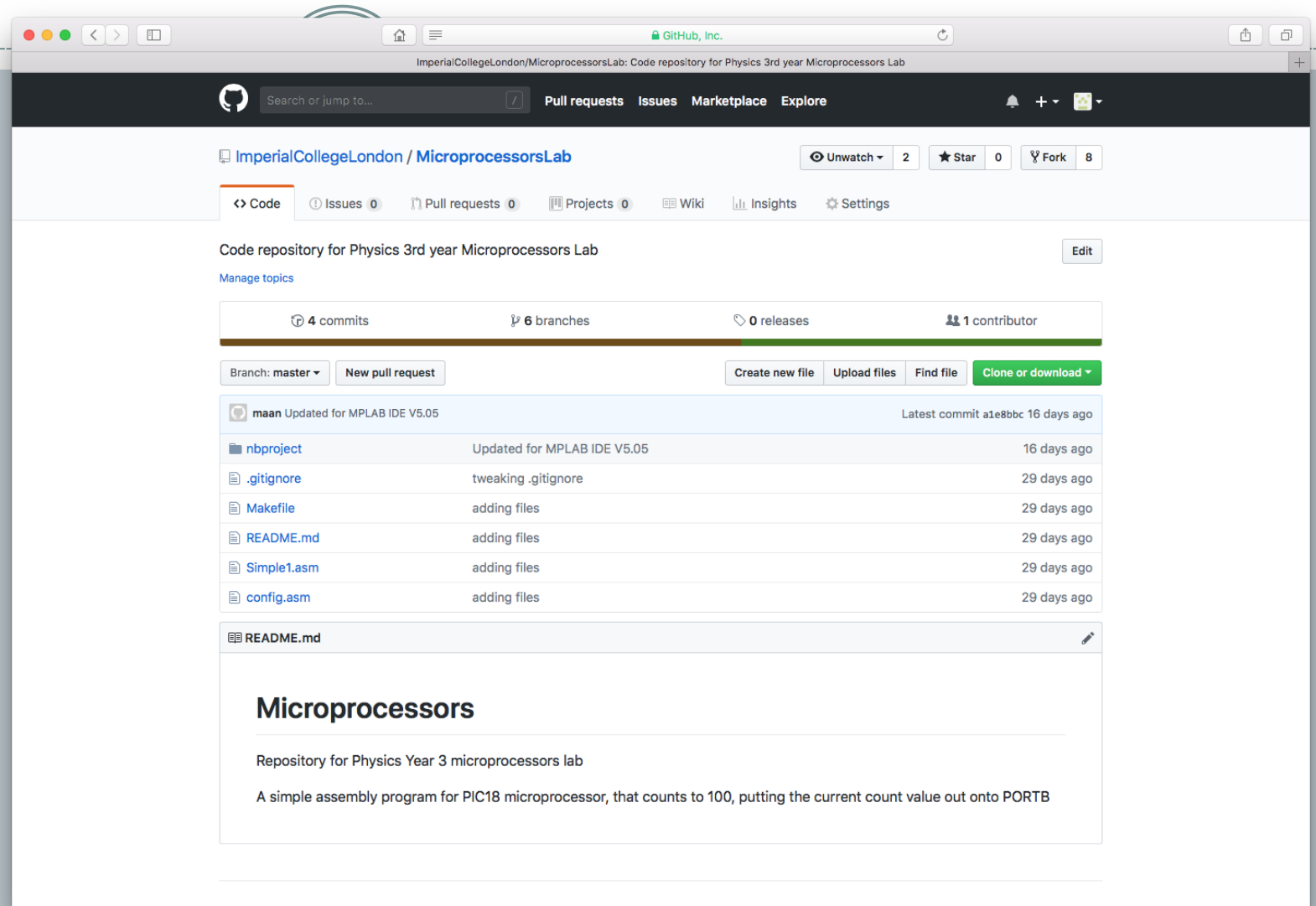
3

- Open the files tab in MPLAB to see what's there
- Only the files not greyed out are essential and stored by GIT
 - .gitignore tells GIT which files not to store
 - ✦ this is a very useful file and you should copy it to any MPLAB repository you make from scratch
 - README.md is a text file describing the project
 - Simple.asm and config.asm are your source files
 - ✦ Config.asm is another useful file that you should copy to your own projects
 - Makefile, configurations.xml and project.xml are all used internally by MPLAB
- All other files and directories in this directory are temporary and can be regenerated by MPLAB as and when they are needed
- On disk there is a hidden directory called .git where your local repository is actually stored
 - It is a database of all changes that have been made to the files in the repository over time at each "commit"



Viewing your repository in GITHUB

- You can view the repository on-line (the origin)
- The stored files are visible and you can see what is in the separate branches
- You can do a few other things here too...



Commit changes from MPLAB

5

Add a useful message to say what the changes have been made

Files with changes are listed here, choose the ones you want to include in this commit

Commit - Simple1.asm

Commit Message:

Changed output port to PORT C

Author: pptman <mark.neil@imperial.ac.uk> Committer: pptman <mark.neil@imperial.ac.uk>

☐ Amend Last Commit

☒ Files to Commit:

...	File	Status	Commit Action	Repository Path ▲
<input checked="" type="checkbox"/>	Simple1.asm	-/Modified	Commit	Simple1.asm

By right-clicking on a row you may specify some additional Actions.

☒ Update Task

Commit Cancel Help

Hit
commit

Do a commit whenever you want to save things in your history for later, or before switching to a different branch. Note in your lab book what you have done!

Push, Pull and Fetch

6

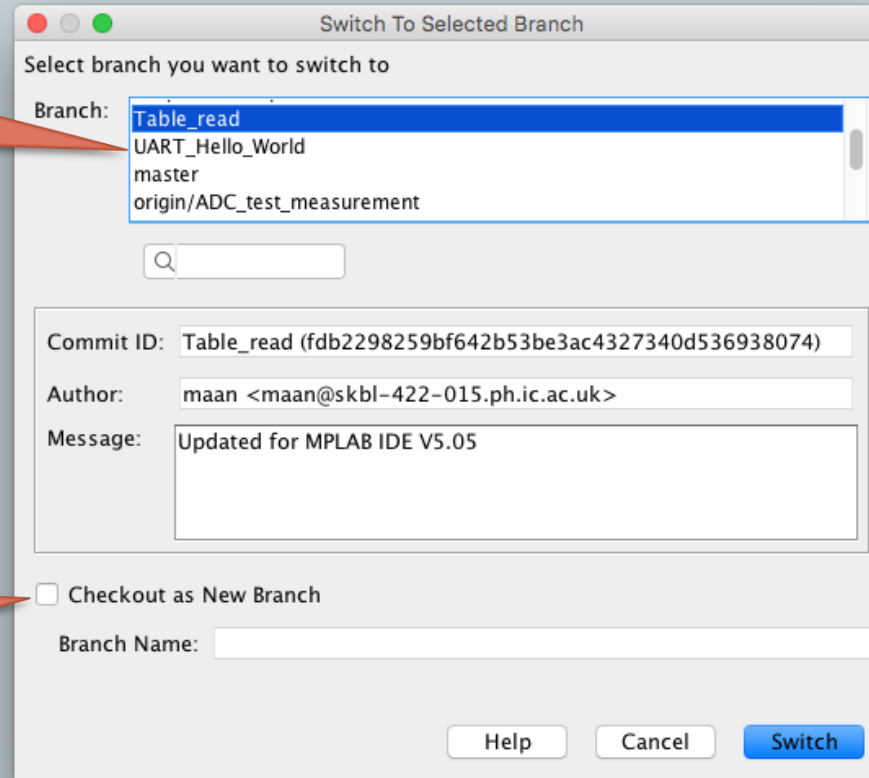
- A Push will send any commits in your local repository up to the origin on GitHub
 - Only if you have write privileges to the origin
- A Fetch will download any commits on the origin that anyone else might have made, to your local repository, but you won't see them yet if you have local modifications
- A Pull, does the same as a fetch but then also merges the changes (if it can) with any changes you have made to your local copy

Switching branches from MPLAB

7

Choose the branch you want to switch to
If it is not local then select the one on origin

Make sure that you tick this box if you are switching to a branch that is only on origin

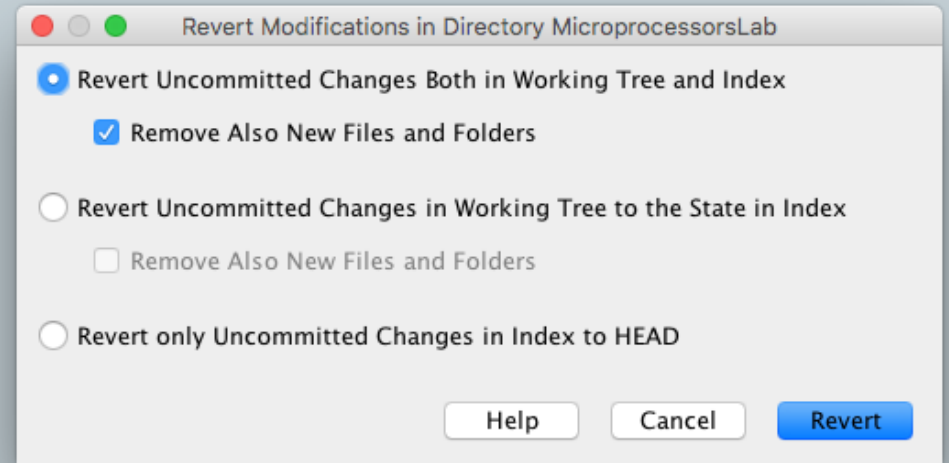


This is what it looks like on a mac, by the way!

Fixing things with a revert

8

- Sometimes things can go wrong with a switch (usually because MPLAB can have configurations.xml open when you switch)
- Or you may just want to bin what you have been doing and revert to your last committed state
- Choose Git->Revert Modifications to undo everything
- Close your project and reopen it to make sure things are all correct



What's a Fork?

A Fork is your own copy of a repository that is linked back to the original

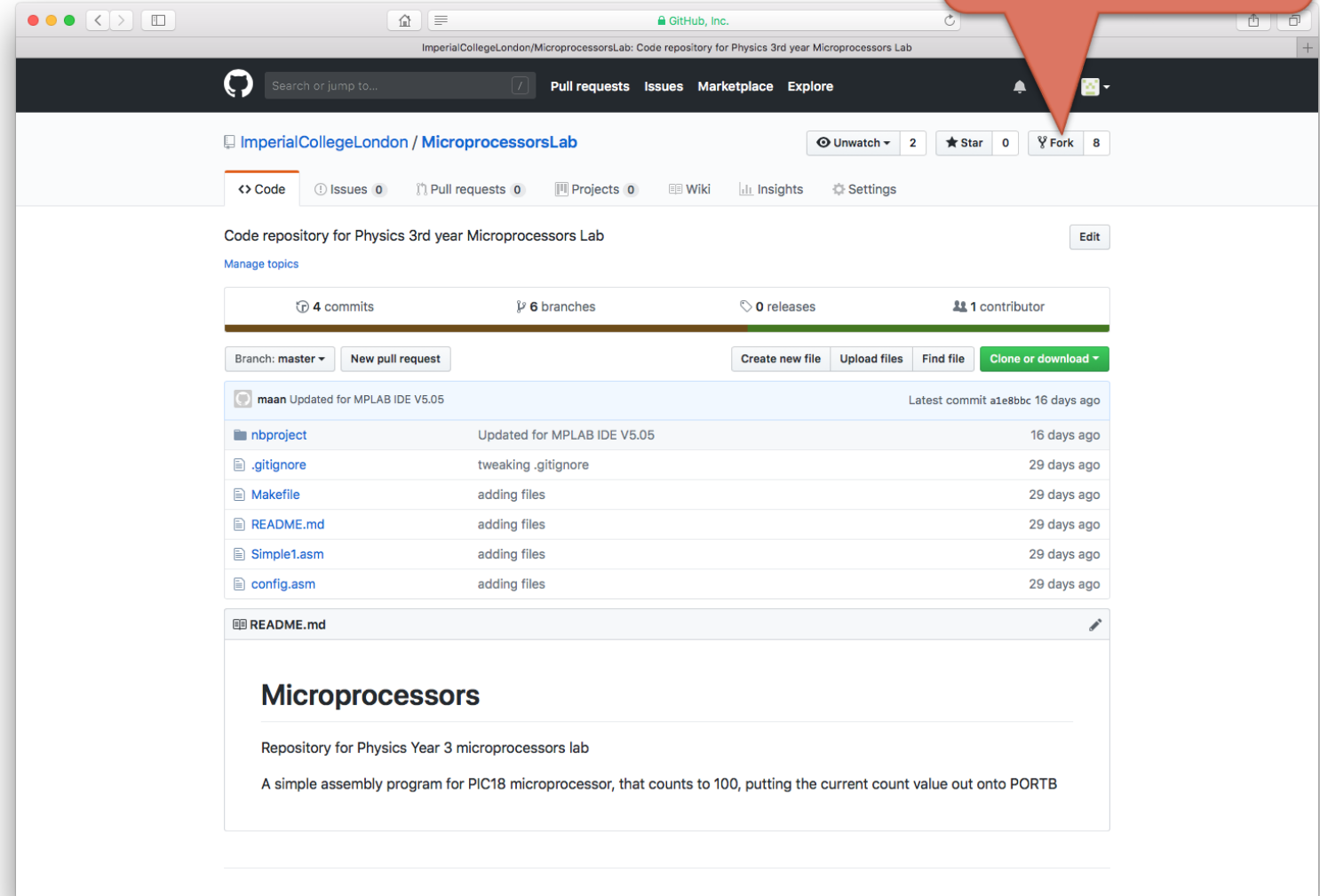
You can store your own modifications to the code in a fork

You can also let the owner of the original code about any changes that you have made that they might find useful (eg bug fixes)

- To do this you create a “Pull request”

If the original is public then your copy will be public too

Create one by clicking here



Make your own Private copy of a repository

10

This works and can all be done via the web, but there are probably many other ways of doing it “properly” ...

Choose owner
ImperialCollegeLondon
if you want to make it
private (without
paying!)

Click + to create a
new repository (you
need to be signed in
to GitHub)

Choose a name for
your new copy

Finally, hit create

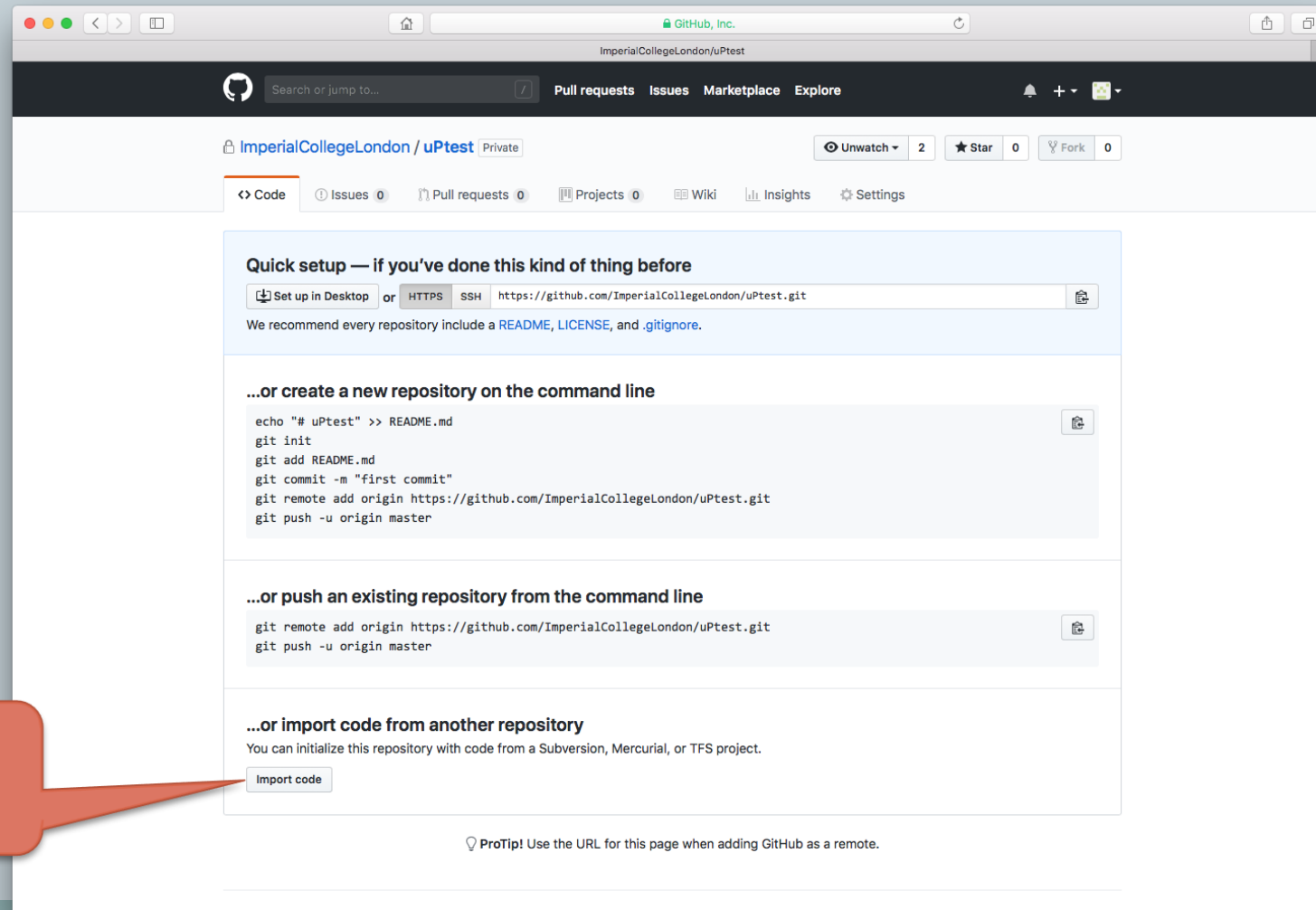
The screenshot shows the GitHub 'Create a new repository' page. The browser address bar shows 'GitHub, Inc.' and the page title is 'Create a New Repository'. The navigation bar includes 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The main form has the following fields and options:

- Owner:** A dropdown menu showing 'ImperialCollegeLondon'.
- Repository name:** A text input field containing 'uPtest' with a green checkmark icon to its right.
- Description (optional):** A text input field.
- Visibility:** Two radio buttons: 'Public' (unselected) and 'Private' (selected).
- Initialize this repository with a README:** An unchecked checkbox.
- Add .gitignore:** A dropdown menu showing 'None'.
- Add a license:** A dropdown menu showing 'None'.
- Create repository:** A green button at the bottom.

Callouts from the surrounding text blocks point to specific elements: the 'ImperialCollegeLondon' owner dropdown, the '+' icon in the top right navigation bar, the 'uPtest' repository name input, and the 'Create repository' button.

Next page...

11



Hit the “Import
Code” button

Doing the copy

12

Add the url of the repository you are copying

Import your project to GitHub

Import all the files, including the revision history, from another version control system.

Your old repository's clone URL

[Learn more about the types of supported VCS.](#)

Your existing repository

ImperialCollegeLondon/uPtest [Change repository](#)

[Cancel](#) [Begin import](#)

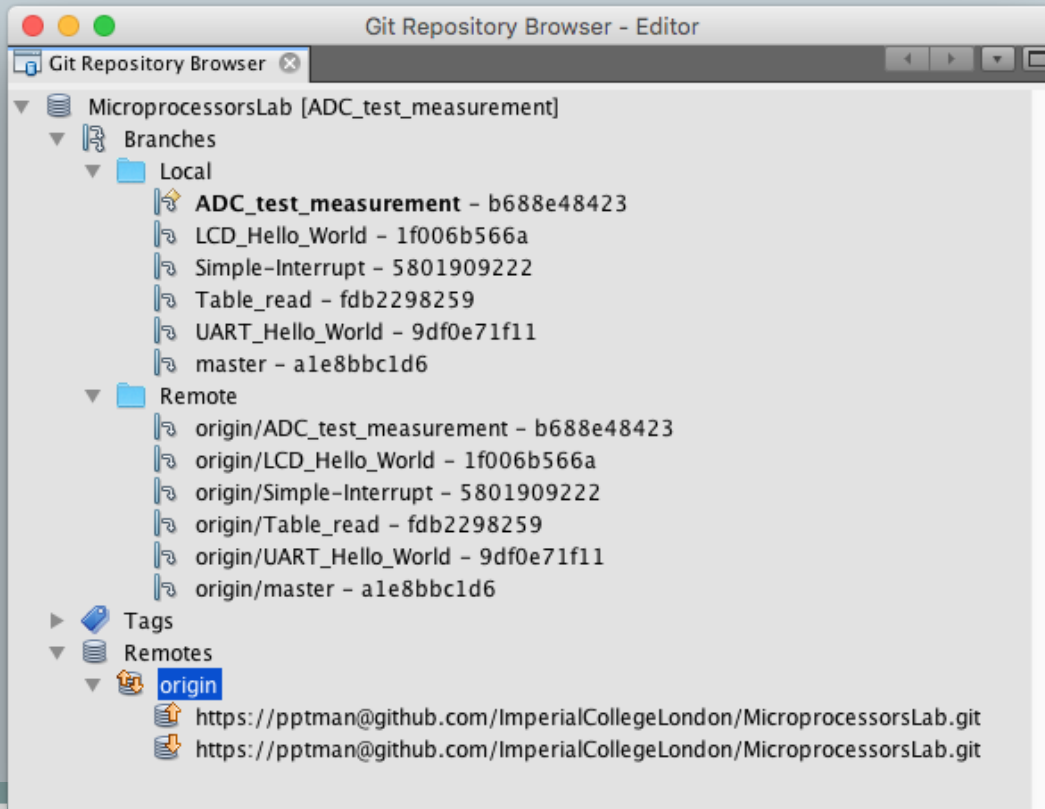
Click begin import to start the copy – you'll get an email when its all done

- You will now have your own private copy that you can do what you want with.
- You can give access to your private copy to other users, such as your partner and demonstrator

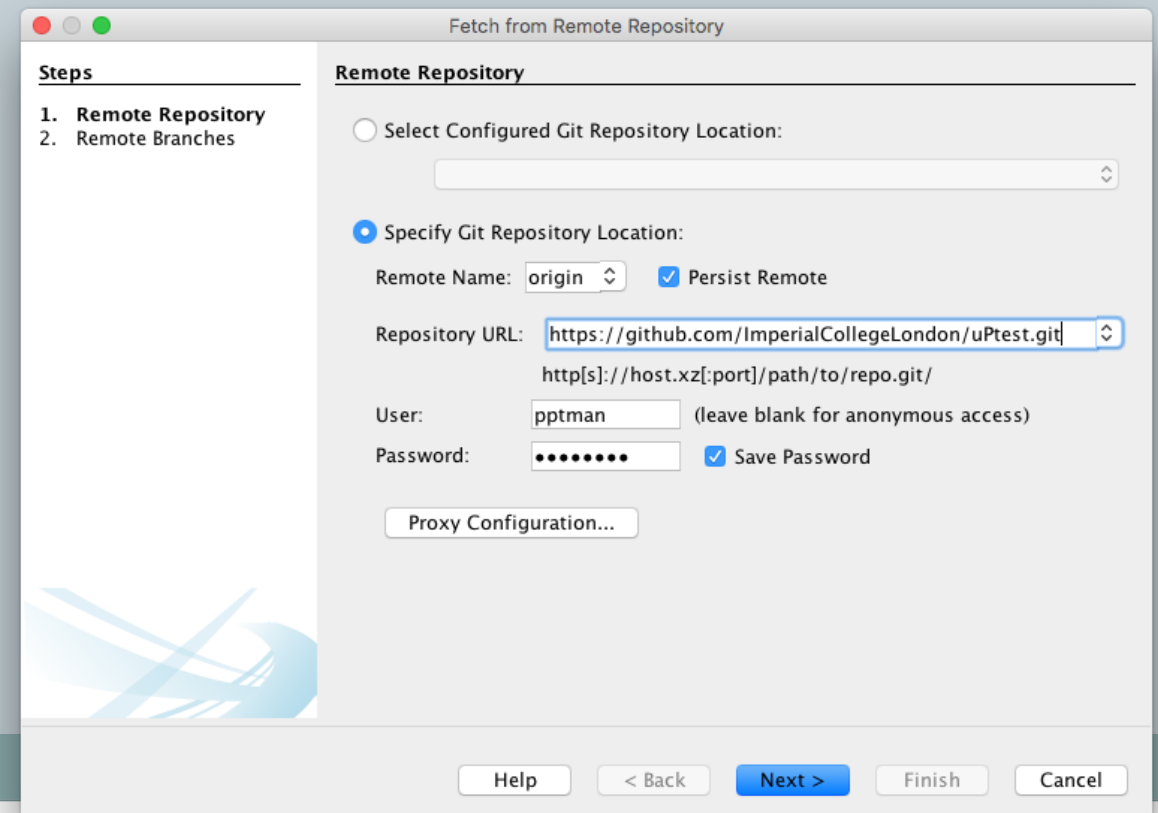
Changing your origin to a new remote (eg fork or copy)

13

- In the repository browser window
 - Select origin
 - Right click and select “Remove”



- “Fetch” the new remote
 - Git>Remote>Fetch...
 - Specify new repository location and branches



Other GIT tools

GIT is originally a command line tool written for linux

Is installed in MacOSX by default

You can install on your college windows 10 account using the new “magic” cloud software system

Ultimately the command line version lets you do the most with GIT

But it gets complicated...

Check out <https://git-scm.com/docs> for details

Real programmers only use command-line GIT!

```
ph-maandisplay:MicroprocessorsLab maan$ ls -l
total 40
-rwxr-xr-x@ 1 maan  staff  3381 23 Jul 01:05 Makefile
-rwxr--r-- 1 maan  staff   188  9 Oct 13:51 README.md
-rwxr--r-- 1 maan  staff   442  9 Oct 13:51 Simple1.asm
drwxr-xr-x 3 maan  staff    96 11 Sep 18:58 build
-rwxr-xr-x@ 1 maan  staff  4438 23 Jul 01:05 config.asm
drwxr-xr-x@ 3 maan  staff    96 23 Jul 01:05 debug
drwxr-xr-x@ 3 maan  staff    96 23 Jul 01:05 dist
drwxr-xr-x@ 11 maan  staff   352  9 Oct 13:53 nbproject
ph-maandisplay:MicroprocessorsLab maan$ git log
commit a1e8bbc1d6b3ff8b581570e498f08b37ceffc5e8 (HEAD -> master, origin/master)
Author: maan <maan@skbl-422-015.ph.ic.ac.uk>
Date:   Mon Sep 24 14:40:06 2018 +0100

    Updated for MPLAB IDE V5.05

commit d1bdc7acddf2f21662afb3b3adf27f60778bf0dd
Author: Mark Neil <mark.neil@imperial.ac.uk>
Date:   Wed Sep 19 14:02:00 2018 +0100

    Updated README.md

commit 1f983ba5c21b1d5476f16eeedec88967f7de3dd5
Author: Mark Neil <mark.neil@imperial.ac.uk>
Date:   Tue Sep 11 19:11:04 2018 +0100

    tweaking .gitignore

commit c169f07794dd01235bd097da4d73cef031c73afc
Author: Mark Neil <mark.neil@imperial.ac.uk>
Date:   Tue Sep 11 19:01:53 2018 +0100

    adding files
ph-maandisplay:MicroprocessorsLab maan$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
ph-maandisplay:MicroprocessorsLab maan$ man git
```

A Mac again, but would look the same on Linux and similar on a windows machine!

A GUI for GIT?

Installing GIT may well install GitHub desktop for you too that can do some stuff in a GUI

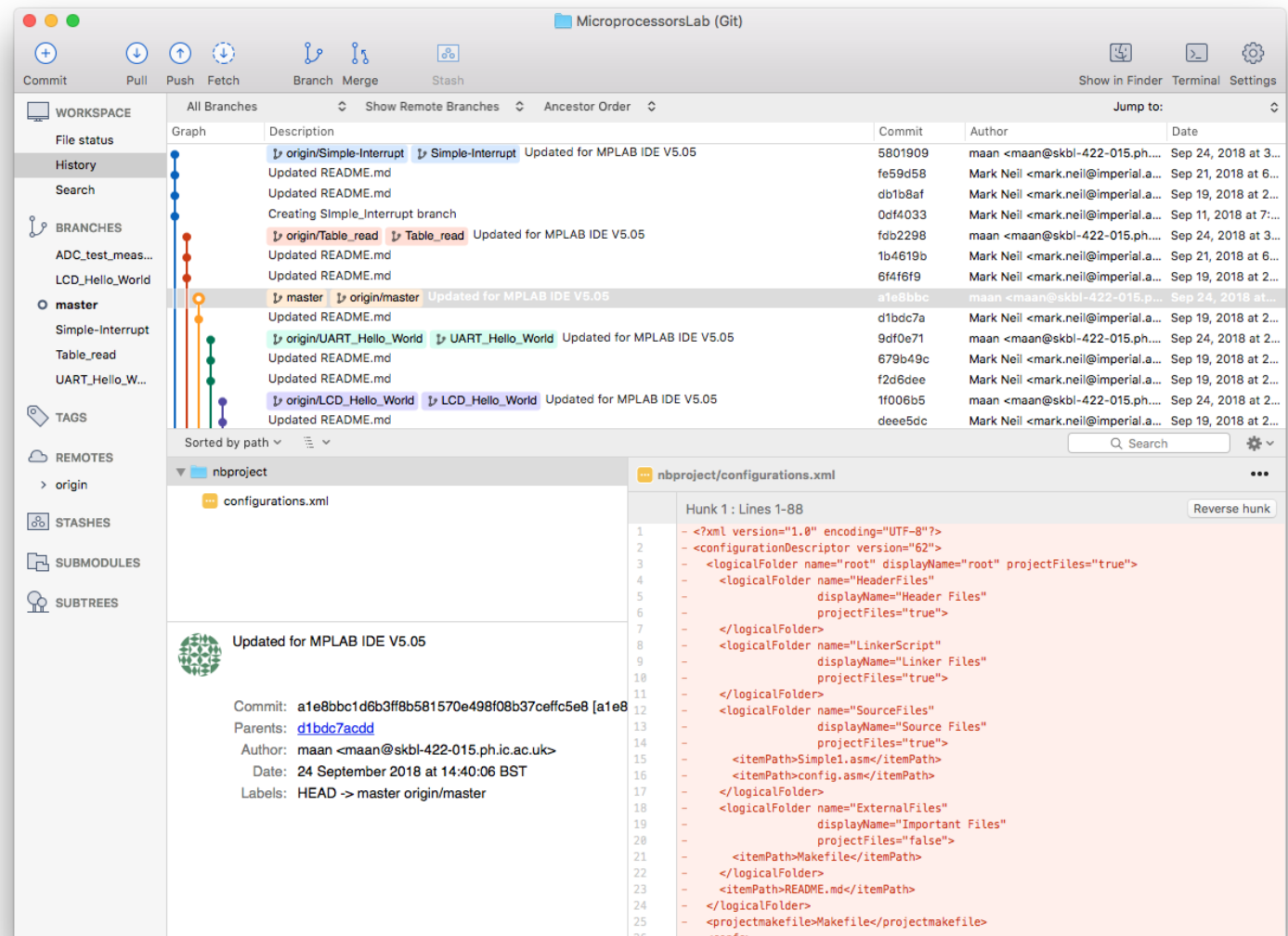
Other GUI programs are available

My favourite is “SourceTree”

Good because it gives you a nice graphical view of your repository structure

And what happened in each commit

Switching branches or to different commits is as simple as double-clicking with the mouse



Doing a merge

16

- This is where GIT gets really useful
- You (or different people) can work on different aspects of the code – usually in different branches
- A “merge” is when you combine changes from one branch into another
- Most of the time this just works as it is relatively easy for GIT to spot how the code has developed in the 2 branches as it tracks the changes between them
- Sometimes you have to help it along the way by telling it which modifications to keep and which to reject
 - But there are tools to help you do this
- You still need to test your code after the merge to make sure it all works!