

Getting Started with Git and Github

Many classes in the CCI will either require or encourage you to use git and github in your work. These can be complex and confusing at first, and how you use them depends a bit on what you need to do.

git vs github

The first distinction that's normally quite important to make is that git and github are related but different things.

git is a piece of software that lives on your computer. It is used for saving versions of files that you are working on -- this is called 'version control'. Git is used on one folder at a time -- a folder that is being versioned with git is called a **git repository**. There are many different ways to use this software -- one of these is the command line.



github is a website online. It's used to share git repositories, and has a bunch of tools to allow people to collaborate on code. It's not the only website people use to do this, but it's the main one and it's very popular.



what about github desktop, git.arts.ac.uk?

github desktop is a piece of software you can use to manage git repositories on your computer. It's by no means the only way to do this -- Visual Studio Code also has tools for managing git repositories, and some people use git from the command line. It's a way of using **git** that is graphical rather than text-based.



git.arts.ac.uk is a version of the Github website that is managed by the CCI. It is often used for classes, as it allows CCI students (but not anyone else) to access the files. If you have a github

account, it will not work with `git.arts.ac.uk` (you need to use your UAL account instead), but otherwise repositories will work the same (e.g. github desktop and the command line will still work).

and github pages?

Github Pages is a web hosting service run by `github.com`, that allows you to turn a repository containing web-compatible code into a website. The website will update when you update the files in the git repository. This doesn't happen automatically -- if you want a Github Pages site you need to approve this in settings.

SUMMARY:

- **github desktop** does the same job as using git from the command line -- it runs locally on your computer and allows you to both version files, and pull repositories from online.
- **github** and **git.arts.ac.uk** are both websites that people use to share and collaborate on git repositories.

what is a repository?

A **git repository** is a folder full of files that are being versioned using **git**. Tools like **Github** are used to share these repositories! Each git repository you have should have its own folder.

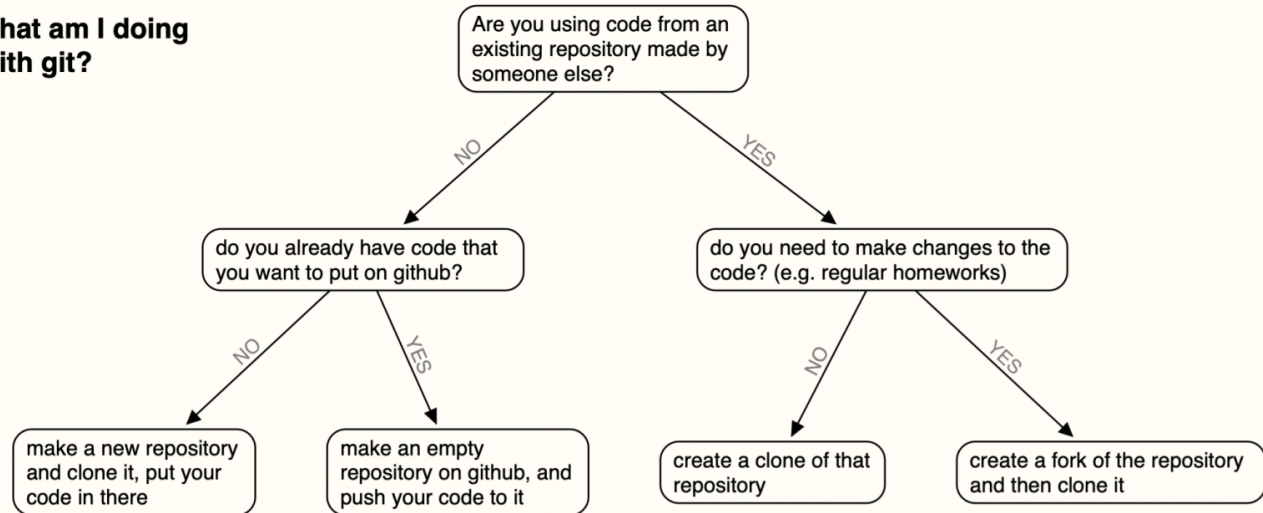
There are lots of ways of checking whether a folder is a git repository -- git stores the

how should I use git?

The core thing that git and github are used for is keeping track of a changing set of files. How you use these tools can depend on whose files they are! If you are making changes to your teacher's files, that's going to work differently to working on files that just belong to you.

Below are some common use cases.

what am I doing with git?



At each of these points, you should end up with 2 things -- a local repository on your computer, that you can edit and run, and a remote repository on github (or git.arts). These things will be related, and if you own the remote repository, you should be able to push to it.

git setup

If you want to use git from the command line, there are a couple of extra setup steps, including making an SSH key. If you plan to use git for many projects, or learn how to use git professionally, this is a good idea. There are instructions for this process here.

If you are just getting started, or only want to use git for a one-off, you can follow these instructions to set up and configure Github Desktop.

Make a new git repository online

Follow these instructions to create a new git repository and clone it locally.

Push existing code to an empty git repository

If you have some code that you wish to publish on github, the steps are as follows:

- make an empty git repository online (this is the remote!) -> do this by following the steps above, but make sure you *do not* create a README or any other files.
- initialise your code as a git repository (this is local)
- link the remote repository to your local repository
- push your local files to the online repository

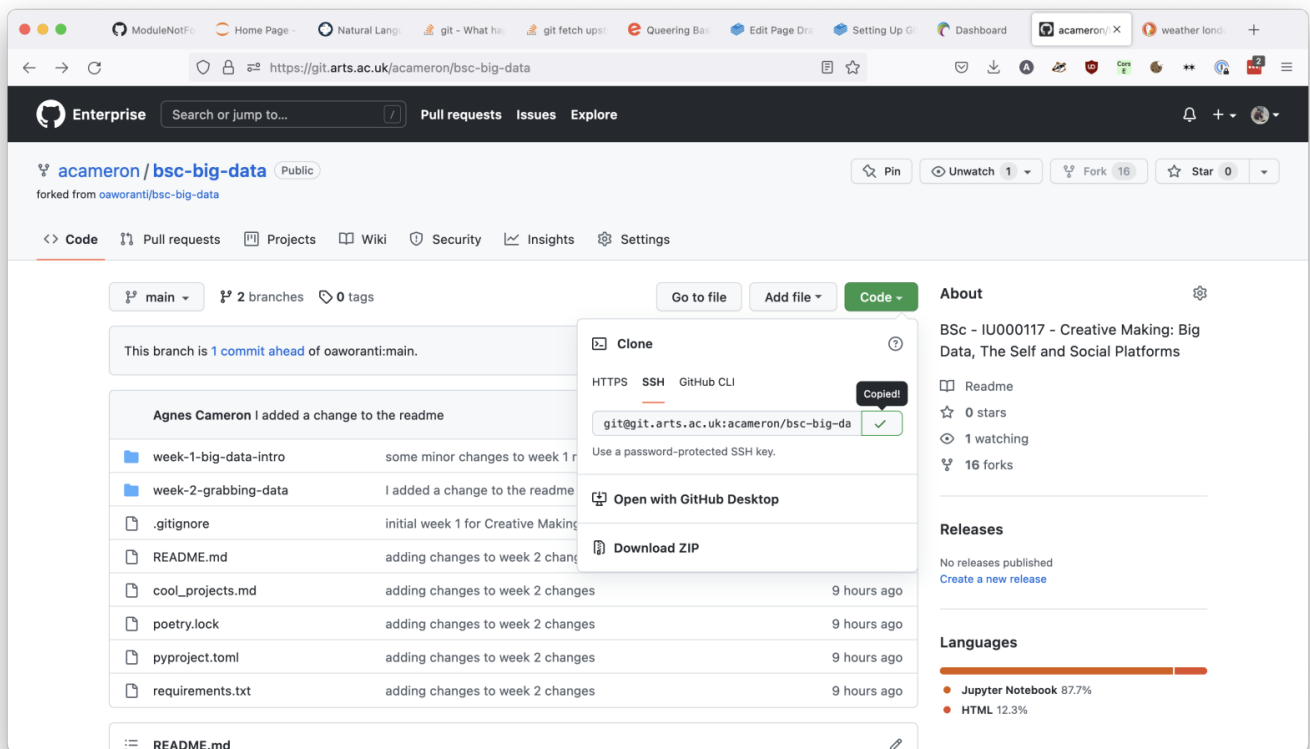
There is a full run-down of these instructions here.

Cloning a git repository

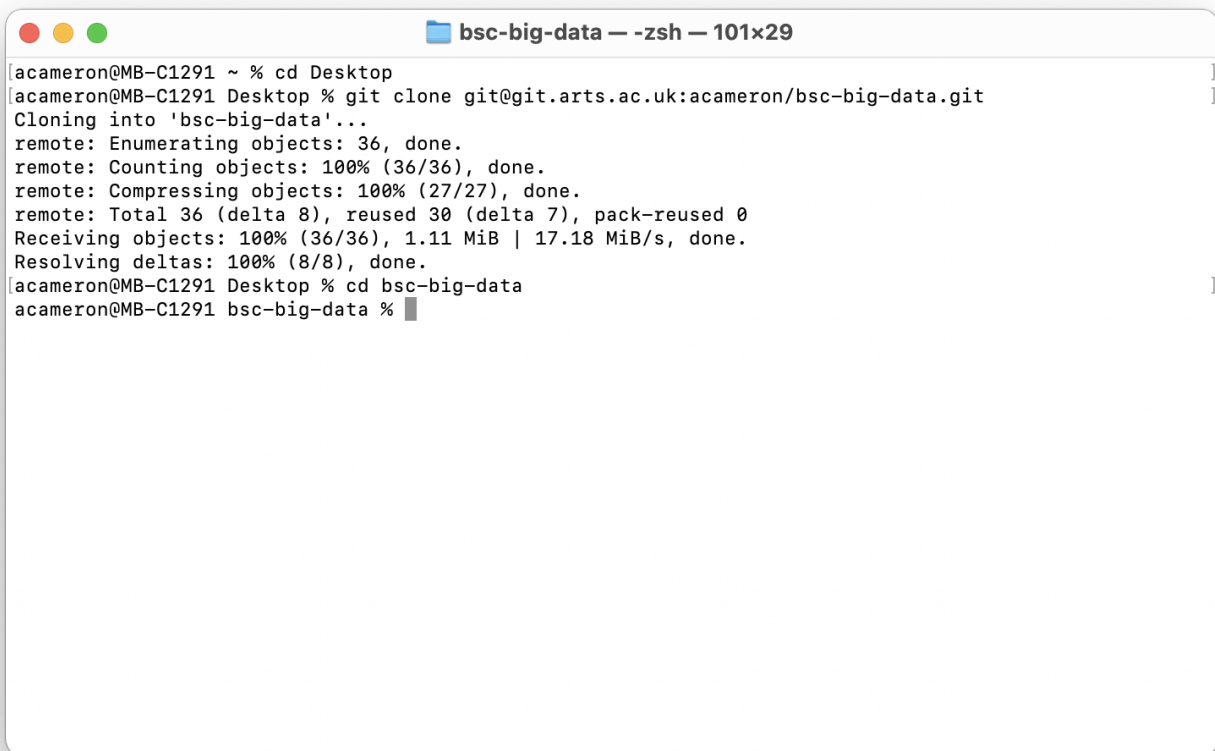
This is for when you want a copy of someone's code, and you might want to edit it, but you don't need to publish the changes that you have made. This is quite a common way to test out other peoples' scripts and tools. It's simpler than forking as you are just making a copy.

You can clone repositories either using the command line:

Get the url of the repository from the green 'code' button to the right of the repository's page.



Navigate to the folder where you want to have the repository on your computer, then use the `git clone` command to clone it.

A terminal window titled "bsc-big-data — -zsh — 101x29" showing the process of cloning a git repository. The user is at the Desktop directory and runs "git clone git@git.arts.ac.uk:acameron/bsc-big-data.git". The output shows the cloning progress, including enumerating, counting, and compressing objects, and receiving the repository data. The user then changes the directory to "bsc-big-data".

```
acameron@MB-C1291 ~ % cd Desktop
acameron@MB-C1291 Desktop % git clone git@git.arts.ac.uk:acameron/bsc-big-data.git
Cloning into 'bsc-big-data'...
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 36 (delta 8), reused 30 (delta 7), pack-reused 0
Receiving objects: 100% (36/36), 1.11 MiB | 17.18 MiB/s, done.
Resolving deltas: 100% (8/8), done.
acameron@MB-C1291 Desktop % cd bsc-big-data
acameron@MB-C1291 bsc-big-data %
```

```
cd where/you/want/repo
git clone git@git.arts.ac.uk:yourname/repo-name.git
cd repo-name
```

Making a fork of a git repository

This is something you will want to do if you need to make and version changes to someone else's code. It's a really common thing to do for classes that require a code-based homework each week. I wrote a longer guide to doing this [here](#) that uses the command line -- there's also a tutorial [here](#) for doing the same thing with Github Desktop.

Revision #11

Created 21 February 2024 10:26:37 by agnes cameron

Updated 7 October 2024 11:32:35 by agnes cameron