Data Science Apprenticeship   
Block Release Day November 2021  
Source Code Management with Git

# Introduction

The aim of this session is to give a practical introduction to version control and source code management. There is a preliminary activity—making sure that you have access to the GitHub system, and the appropriate software on your machine. Then, there are three tasks for you to attempt.

# Preliminary

In order to work with the GitHub system, you need to do two things.

Firstly, you need to register to use the GitHub system. Go to:

<https://education.github.com>

Graphical user interface, text, application, email

Description automatically generatedand sign up (by clicking the “join global campus” button) for an account using your Nottingham email address. Once you have created your account, go to [*https://github.com/settings/tokens/new*](https://github.com/settings/tokens/new). It might ask you to log in again. Type “testing” in the note box, tick “repo” and press “generate token”.

make a copy of the token, you will need it later. Once you have done this, **let us know your username so that we can add you to the remainder of the projects for this session.**

Secondly, you need to make sure that the Git system is installed on your computer. Go to

<https://git-scm.com/downloads>

and download the appropriate version for your machine (Windows, Mac or Linux).

We will walk you through the basics in a step-by-step way in this worksheet, but if you need more information, have a look at the documents here:

<https://git-scm.com/book/en/v2>

and the videos here:

<https://git-scm.com/videos>

# Task 1: Check in and Check out

Let’s start by connecting to an existing code repository, checking out the current copy of the code, making a modification to one of the files, then checking it in again. The most common way of using git is from the *command line*, that is, by typing in commands into a terminal window. There are also a number of graphical interfaces to git, which you can download and explore here:

<https://git-scm.com/downloads/guis>

But, for now, let’s concentrate on the command line. Open a command line window:

* On Windows, press Windows-X then select Command Prompt *or* PowerShell (varies between versions of Windows)
* On Mac, open Applications → Utilities → Terminal

this should bring up a window into which you can type commands. You will probably have already used this to launch Jupyter Notebooks earlier in the course. Change your working folder to the Desktop, or wherever you store your work for the course:

* On Windows: type cd C:\Users\(username)\Desktop
* On Mac/Linux: type cd ~/Desktop/

Then you should type in (or copy/paste) the following command:

git clone https://github.com/colingjNottingham/CheckInAndOut.git

This will make a local copy of a project called *CheckInAndOut*. Now have a look in the current folder. You should be able to see a list of files, mostly *.ipynb* ones, that form this project. Note that one of the files has your student number in the filename. Open this one in Jupyter notebooks. There is a line there asking you to insert your name. Insert your name there, then save the file in Jupyter notebooks.

Now, go back to the command prompt. We are going to upload the modified file back to the repository. Firstly, enter the following command at the command line:

git add 999999999.ipynb

but with your student number where it says 999999999. This tells the git system which files you want to upload in the new version.

Now we are going to “commit” the files into the repository. That is, you are create a date-stamped version of the project on your computer. This is what we mean by calling this *version control*. If we wanted to look back to an earlier version of the program, then we could do so—future commits don’t over-write the old version, they just create an additional version. So, enter the following command:

git commit -m “changed my name to #######”

where ####### is your name. This final part of the command, after the -m, is a comment to you, and potentially to other users of the same project, to explain what your latest update did.

Now, finally, we are going to put the changed file back into the main version of the project, the one stored “on the cloud” in Github’s servers. Type:

git push origin main

It will ask you for a username (this should be your Nottingham email address) and a password (this is the “Personal Access Token” that you generated earlier in the worksheet.

Go to

<https://github.com/colingjNottingham/CheckInAndOut/tree/main>

and click on the filename with your student number. You should be able to see the modified file, uploaded to the repository. Once other people in the class have done this, then you can have a look at their files and confirm that their names are there too. You can download the latest version of the project by doing:

git pull

If you get stuck at some point, try:

git status

which will give you a summary of the current state of the repository.

There are more details about what is going on here, which you can read in your own time later.

<https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository>

# Task 2: Branches and Merge

So far, we have worked on one file in a project, modified it, then uploaded it. This is fine for small projects where each person has their own well-defined area of responsibility. But, often, a team is working on a complex collection of code, and might be making a number of modifications across a number of files. Git uses a facility called *branching* in order to manage this. The main idea is that we create a new version of the code, called the branch, make changes to that branch, test if it works, and then merge our code back into the main project. If we have worked on files that are otherwise unchanged, then our updated files replace those in the main project. More cleverly, though, git tries to merge different versions of the same file (basically, by using some text-matching process). This doesn’t always work, but it is rather magical when it does.

Let’s have a go at this. Start by taking a copy of the project called *alphabet*:

git clone https://github.com/colingjNottingham/alphabet

have a look at the project folder, there should be a single file in it called *alphabet.txt*.

Let’s create a new branch to work on this document:

git checkout -b mynameNewBranch

where instead of “my name” you put your name (without spaces). This creates the new branch, and switches you so that that is the branch that you are currently working in. You can see the current list of branches by:

git branch

and the one that is currently active (in git parlance “checked out”) has an asterisk (\*) next to it. Try this command, and the new branch should have that asterisk.

At the moment, this new branch is just on your computer. Push it up to the GitHub cloud repository using the following command:

git push origin mynameNewBranch

note the similarity with the earlier use of the push command. If you look at the project on github, you can now see the change. Go to

<https://github.com/colingjNottingham/alphabet>

in a web browser, and where there is the “main” button, press it to see the list of branches:

Graphical user interface, text, application, chat or text message

Description automatically generated

Now, we are going to change the code on this branch. Edit the *alphabet.txt* file (if you don’t know how to do this, ask us) and insert your name after the letter that it begins with, e.g.:

C  
Colin Johnson  
D  
E

Now, we want to add this to the repository on that branch. So, as before, do:

git add .

Actually, this is slightly different from before. The “.” means “the current folder”, so everything in the current folder will be uploaded to the repository. This is a common shorthand. Then:

git commit -m “added my name”

git push origin mynameNewBranch

Go back on the web to have a look at:

<https://github.com/colingjNottingham/alphabet/>

and compare the version in the main branch of the repository with your version. Now, switch back to the main branch:

git switch main

and then let’s request for our changes to be merged into the main branch:

git merge mynameNewBranch

check on the web that your changes have been included. As the class goes on, gradually all of your names should get added to the main branch.

Finally, we can delete the branch that we were using:

git branch -d mynameNewBranch

git push origin --delete mynameNewBranch

this will delete first the branch on your machine, then on the GitHub servers. Again, if you check back on the project webpage, you should now see that your branch has gone!

# Task 3: Create your own projects

So far, we have edited projects created by someone else. How do you create your own project?

Find a file or set of files (e.g. your assessment) and make a copy of these into that folder. Open up the command line, and navigate to that folder by using the cd command that we saw earlier. Now do:

git init

where “init” stands for “initialise”. This turns the folder into a git repository, locally to your computer. There isn’t that much that is special about this, really—it just adds some additional files in a hidden folder called *.git* that contains a history of the file versions, and how that file links to the cloud copy on GitHub.

Now, as usual, add then commit the files:

git add .

git commit -m “first commit”

So, now we have a *local* (on your computer) version of this. Let’s connect it to the cloud version on GitHub. In a web browser, go to:

<https://github.com>

and click on the new button next to where it says “Repositories” near the top left-hand corner:

Graphical user interface, text, application, chat or text message

Description automatically generated

Give it a name (e.g. “assessment”), and then click the “Create repository” button.

git remote add origin git@github.com:username/assessment

where username is our username on GitHub, then

git push -u origin main

to connect the repository on your computer to the GitHub servers. Now, you should be able to see your project on the GitHub site. Experiment with changing files on your machine and uploading them to the project.

Alternatively, you can start by creating it on the GitHub site, and then clone it to your machine.

# Additional Topics

If you finish these tasks, investigate one of the following topics, e.g. by looking at the references mentioned earlier:

<https://git-scm.com/book/en/v2>

<https://git-scm.com/videos>

or by googling, or by experimentation.

* How the merging works, and what its limitations were. If you and a colleague have both finished the tasks, create a new project (e.g. one like the alphabet one) and see how complex you can make changes before the merge command gives errors.
* How you can rewind a branch to an earlier version.
* The current example uses a public repository, that anyone can access. Have a look at how private repositories work.
* Find out about the GitHub command line interface here: <https://cli.github.com/manual/>, install it on your machine, and see how it allows you to simplify some of the tasks above.
* Find out about README, LICENSE, and .gitignore files.