Class Notes: Intro to remote code servers like GitHub

Pranav Minasandra

June 12, 2024

Note 1: Throughout this document, commands are written in `typewriter-text`, and must usually be entered as they are in your command-line. Within the typewriter text, some things might need to be modified, and are indicated using `<>`, like so: `command1 option1 <you need to change this>`. Sometimes, commands are long and need to be written in separate lines. When I write these, commands will begin with a `$`, like so:

```
$ meaningof argument1 "Life" argument2 "The Universe" argument3 "Everything"
```

In such cases, you only enter the command after the `$`, but don't have to type the `$` itself.

Note 2: While this document might seem full of command line arguments, in reality, there are just the following four git commands:

Command	Use
`remote`	Add and manage remote locations
`push`	Upload local changes
`pull`	Download remote changes
`clone`	Download a repository

Contents

1 Why should we upload code online?

This is a deeply philosophical question — what exactly does research code accomplish by being shared online? Clearly the sharing of scientific code is, in 99% of cases, not achieving the same goal as code put up by Ubuntu or Tesla (or any other software company). Keeping the discussion of the purpose of sharing scientific code aside for another day, let's look at the more pragmatic reasons.

1.1 Journals need us to put up code

With the rise of open science, more and more journals insist upon usable, replicable code and data availability.

1.2 Back up is necessary

Even a privately visible online repository of your code can be a lifesaver in those surprisingly frequent computer crashes and data losses.

1.3 Collaboration

If you are working on a project together with someone else, writing code together is the best idea to use an online repository on something like GitHub, so you can work on your own machines and push to the same repository.

1.4 Ease of working

Often, we write code on small personal laptops and run them on bigger, better computers with more capabilities. A remote repository makes synchronising code on both systems really easy.

2 Set-up

For this workshop, we will be using GitHub, a popular online Git server owned by Microsoft. Alternatives to GitHub include GitLab and BitBucket. The Max Planck Society seems at first glance to have their own GitLab subsciption, but I have never used this personally.

With routine use of Git, it eventually becomes easy to set up one's own Git servers on a local network, e.g. in your office.

3 Getting help

If you need GitHub-related help after the session on July 10, 2023, you can use the following resources.

- GitHub docs
- Chacon & Straub, *Pro Git* (Apress Publications, 2014) (freely available online as a PDF)

4 Preparing for the session

Make sure to create a GitHub account by the time of the workshop.

Everything that follows from this point will be covered in the workshop.

---

5 Linking your computer to GitHub

GitHub has moved towards mildly inconveniencing users in favour of added security. For this, a public-private key based authentication system has been implemented, which needs us to put in some time before we freely start using their service. For more about public-private key based authentication, click here.

First, we need to get a pair of keys, called the 'private' and 'public' keys. The public key can be shared as much as you like, and uploaded on any software. The private key is your own, and must be shared as little as possible. Run the command

```
$ ssh-keygen -t ed25519 -C "<your-email-ID-here>"
```

and follow the instructions that happen at this point.

Now, we need to enable your computer to use these keys for remote login into GitHub. Run the following commands

```
$ eval "$(ssh-agent -s)"
$ ssh-add ~/.ssh/id_ed25519 # Replace that filename in case
                            # you named the file
                            # differently above
```

Your computer now has access to your keys. Importantly, your computer now holds the 'private' key. Now we need to give GitHub your 'public' key, so a secure channel can be set up between them.

Type `cat ~/.ssh/id_ed25519.pub` to print out the contents of your public key, and copy this content. Then, go to GitHub, click on your profile photo to access a drop-down menu, and then click on 'Settings' In the 'Access' section, you can add a new SSH key (paste your public key in the 'Key' field).

6 Changing the name of the default branch

This is a bit of a controversy. In 2020, in the wake of the intense racial tension in the US after the inhumane and unforgivable murder of George Floyd, GitHub changed the name of the default branch from the historical `master` to a simpler `main`. However, the change wasn't retroactive, and a lot of software, such as the local `git` command, use `master` as the name of the default branch. Growing up in India, I was surrounded by schoolmasters and headmasters and postmasters. I even happen to have a Master of Science degree. The association of the pretty common word 'master' to slavery seems a bit far-fetched to me. However, different cultures use words differently, and the desire to move away from a painful word is completely understandable. So here are two options, and you need to choose one:

6.1 Change the GitHub default branch name to 'master'

Go to https://github.com/settings/repositories, and change the name of the default branch to 'master'.

6.2 Or, change Git's default setting to 'main'

Alternatively, you can change the local Git software on your computer to work with 'main' as the default branch.

```
$ git config --global init.defaultBranch main
```

Note: Only do one of the above two options.

7 Creating a repository on GitHub

These days GitHub is the de-facto place where code is put up online. One change-in-mindset that really helps is to think of the code you put up on GitHub as the 'original' code, and that the code you have locally on your computer is the hacky garbage version. You play around with the garbage version until it achieves some milestone, after which you push changes from your machine onto the online server to make it available publicly (or privately, to your collaborators).

To create a repository on GitHub, sign in on github.com and then, in the left-hand column, click on the button that says 'New'. Alternatively, go to your GitHub profile, click on the tab 'Repositories', and then click on the blue button that says 'New'. You will now be taken to a web-page where you can choose the name of your repository, whether it is public or private, and whether you want to add some files already. Let us talk already about some of these decisions.

7.1 The name of your repository

This is a very subjective, personal choice. Definitely stay away from long names for repos like 'code_for_2020_paper_on_
Also avoid uninformative names like 'unicorn_box'. Good names are short and sweet, and together inform a lot about what the repo contains. For me, good repo name examples are 'python-docs-theme' (the theme for the python docs website), 'indicators-financial-meltdowns' (academic code to predict recessions), and 'motionmapperpy' (code to map motions to a higher dimensional space).

7.2 Public vs Private

A public repository is available to everyone on the internet. If you are publishing a paper, it is always a good idea to put up all your code (and its entire version control history) as a public GitHub repository. Then you show all your work, and no questions can be raised about your open-science practices. More sensitive code (if you are working on a top secret topic, for example) can be made private, and shared

only with the GitHub accounts of specific collaborators after the repository has been created. Discuss with your supervisors, and make the decision about whether your code should be public or private.

7.3 README file

In the root folder of your repository, you can create a folder called README.md. This file is supposed to contain a brief (sometimes not so brief) description of the code, who wrote it, for what they wrote it, how to set up and run the code, and whom to contact in case of problems. A good README can make life much easier for anyone else who uses your code.

7.4 .gitignore

In the root folder of your repository, you can create a file called '.gitignore'. The file should contain a '.' at the beginning. In this file, you can list line-by-line all the files on your computer that may be in the Git repository, but you prefer Git and GitHub should simply ignore these files. For instance, if your .gitignore has the following lines

```
$ cat .gitignore

*.pdf
comments.txt
```

then Git simply ignores the existence of all files ending with '.pdf' (which is all pdf files) and the specific file 'comments.txt'.

7.5 License

Simply putting up code online does not give people the right to use bits of your code in their own projects. Sure, most of the time, we in academia in fields like animal behaviour simply send our colleagues the URLs to our GitHub repos and ask them to run the code. But in the long-run, as open-science practices become more predominant and as work inevitably becomes more quantitative and code-centric, you want your code to be used by a wide range of people. For this, you need to choose an appropriate license.

It is very much beyond the scope of this workshop to discuss different open-source licenses in detail. In fact, it is my very strong opinion that scientific code needs its own license, separate from typical open-science code. However, standard practice in today's world is to add either a GNU GPL 3.0 License or an MIT License to your code. As an exercise, look up the difference between these two popular licenses.

8 Making an online repository available locally (git clone)

To download a ready repository from an online server, you can use the command

```
$ git clone <url>
```

.

For the sake of this tutorial, we will clone a repository that I have created specifically for this workshop, where all of you should already have been added as collaborators.

```
$ git clone git@github.com:pminasandra/github-tutorial-2024-jun.git
```

Note: The URL above does not start with 'https://'. This is because we are downloading this repository from SSH, and not normally like we would a random person's code. Always clone using the SSH URL if you are going to make edits on your own. To get this URL, navigate first to your repository on GitHub (in this case, https://www.github.com/pminasandra/github-tutorial-2024-jun, and then click the drop-down button that says 'Code'. In the drop-down, choose the tab 'SSH', and copy the URL from there.

In the README, there are some tasks that test your Git and GitHub skills. Try them out. (The parts about uploading changes obviously apply to someone attending my workshop only, and not to those reading this online).

9 Making a ready repository available online (git remote, push, pull)

Many times, you already have a long-running Git repository locally on your computer that doesn't yet have an associated GitHub repository. In technical terms, the GitHub repository on which the code will eventually go is called your local repository's 'remote'. You will hear sentences like 'add a new remote and push to it'. What this means is that you should add another repository (maybe on GitHub, maybe on BitBucket or some other alternative) to the local repository on your computer. Okay, don't let the terminologies confuse you, we'll talk about these in detail in the workshop.

We will take the poem repository you made in the morning, and put that up on GitHub with the full log of all the changes you made. Create a repository on GitHub with a nice, catchy, and descriptive name. Don't add anything to your repository already (so no README, .gitignore, or LICENSE), which are the thing GitHub prompts you for.

After this, GitHub immediately gives you the SSH link, something that starts with `git@github.com:`. Copy this to your clipboard.

Now open GitBash, and navigate to your repository of choice. Making sure that all recent changes are staged and committed, run the command

```
$ git remote add github <the-link-you-copied>
```

Now, to upload all your changes, you can type the command

```
$ git push github master
```

At any point, to retrieve changes made by you or someone else online, type `git pull github master`.

To push or pull some other branch, simply use

```
$ git <push or pull> github <branch>
```

Note than 'github' here is the name of the remote associated with the URL you copied. I like to call this thing 'github' for the sake of this example, but usually people call their one and only remote 'origin'. Now if you go on GitHub to the repository you just created, you can find not only your poem, but also the entire history of everything you did this morning.

10 More about pushing and pulling

The command `git push` when called appropriately uploads all commits since the most recent `git push` onto GitHub. Symmetrically, `git pull` downloads all changes made on GitHub (e.g., by you on a different computer, or by your collaborators) since the last `git pull` onto your local system. It is always a good practice to `pull` before you `push`.

11 Collaborative GitHub

As a class, we will all modify the repository you cloned earlier. Follow the instructions on the README file in the cloned repository. You can see this with

```
$ cat README.md
```

or directly on the GitHub repository. Follow the instructions to get a good refresher of everything we have done so far.