Introduction to GitHub

Python-do-ECARES
Fabrizio Leone

Introduction

Git and GitHub

▶ Git is a version control system, i.e. a way to keep track of the whole history of things you do on a file. It is useful to save, manage and edit all the different versions of your project.

▶ GitHub is a web service that allows to conveniently work with Git. It allows you to create your own directories, see projects of other people and collaborate with them.

▶ GitHub offers four different subscription plans. We will work with a free one, which means that *everybody* can see what we do. However, with an academic account, you can also create private repositories.

GitHub: What It Does And Does Not Do

▶ No matter which subscription plan you choose, GitHub offers very limited storage space (you cannot upload files > 100MB). Therefore, it is **not** suitable for storing large files (e.g. datasets). **GitHub is not a substitute for a cloud**.

▶ GitHub is a platform where to upload mostly **source files** (e.g. .tex, .txt, .m, .R, .do, .py, .doc,...) and light pdf.

▶ You can read more about Git and GitHub here and here.

GitHub Desktop

▶ In this course, we interact with GitHub mostly through the GitHub Desktop application.

▶ GitHub Desktop provides a simple yet powerful desktop interface to GitHub.

▶ You can download GitHub desktop here.

▶ You can also interact with GitHub using the Terminal (not covered in this class).

Introduction
0000
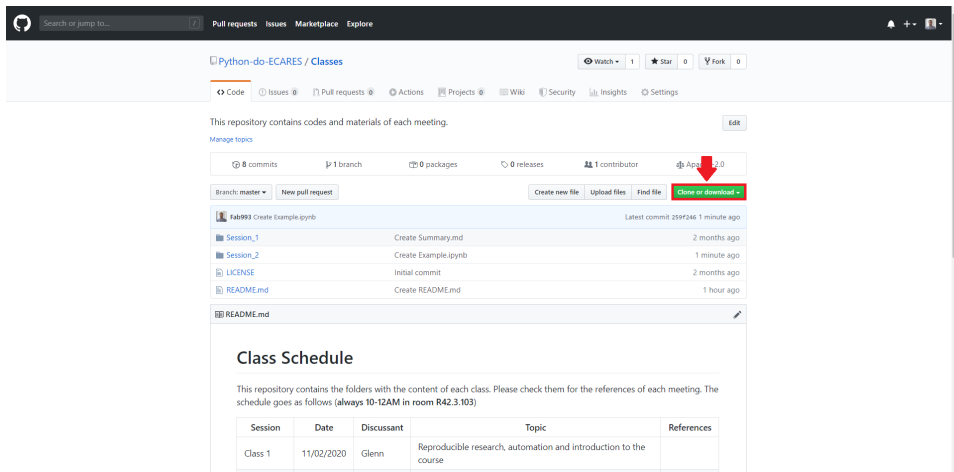
First Steps With GitHub Desktop
●000000000000000

Commit and Push
0000

Browse Through History
000

More Functionalities
00

First Steps With GitHub Desktop

Introduction
○○○○

First Steps With GitHub Desktop
○●○○○○○○○○○○○○○

Commit and Push
○○○○

Browse Through History
○○○

More Functionalities
○○

Step 1. Clone Repository

Go to the Classes repository on GitHub and click on the "Clone or download" button.

Introduction
○○○○

First Steps With GitHub Desktop
○○●○○○○○○○○○○○○

Commit and Push
○○○○

Browse Through History
○○○

More Functionalities
○○

Step 1.A Open Repository

Choose "Open in Desktop".

Introduction
0000

First Steps With GitHub Desktop
0000●000000000000

Commit and Push
0000

Browse Through History
000

More Functionalities
00

Step 1.B Clone Repository

Check the local path where to clone the repository and click "Clone".

Introduction
o o o o

First Steps With GitHub Desktop
o o o o o ● o o o o o o o o o

Commit and Push
o o o o

Browse Through History
o o o

More Functionalities
o o
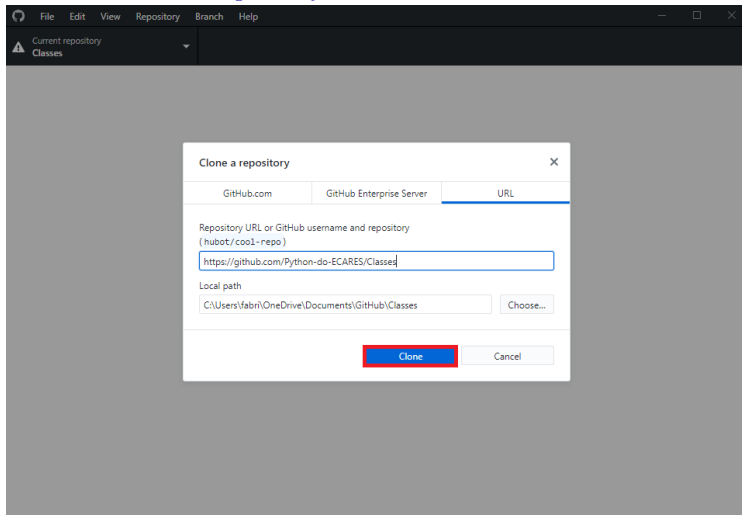
Step 1.C Open Local Directory

Under GitHub Desktop/Classes/Session_2 you will see the following files.
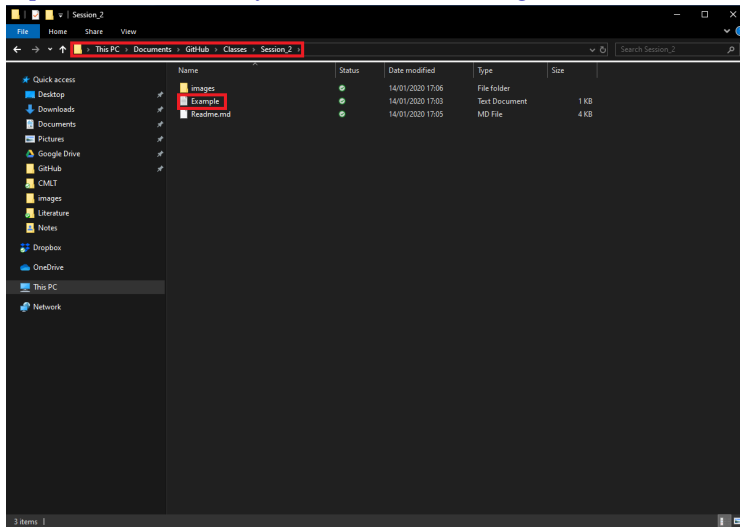
Step 2 Make Changes

Open *Example.txt*. Add a comment line with your name.

Introduction
0000

First Steps With GitHub Desktop
0000000●000000000

Commit and Push
0000

Browse Through History
000

More Functionalities
00

Step 2.A Save Changes

Save the file. Changes will be displayed in GitHub Desktop as follows.

Introduction
○○○○

First Steps With GitHub Desktop
○○○○○○○●○○○○○○○○

Commit and Push
○○○○

Browse Through History
○○○

More Functionalities
○○
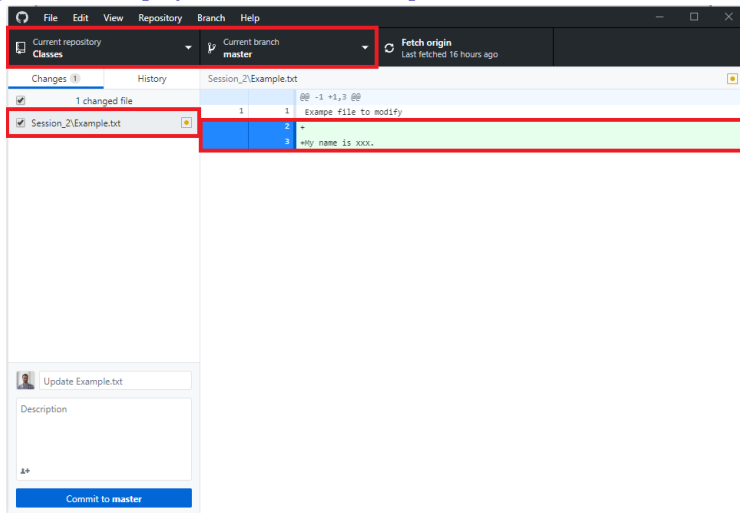
Step 2.B Create Your Own Branch And Commit Changes

Select "Master" and "New Branch". Never directly commit your changes to the Master during this course.

Introduction
○○○○

First Steps With GitHub Desktop
○○○○○○○○○●○○○○○○○○

Commit and Push
○○○○

Browse Through History
○○○

More Functionalities
○○
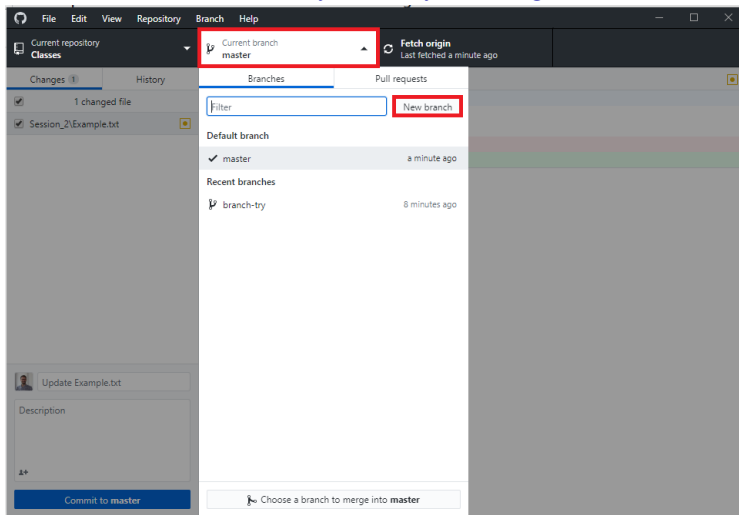
Step 2.C Name Branch

Give the new branch your name. Then click "Create Branch".

Introduction
○○○○

First Steps With GitHub Desktop
○○○○○○○○○●○○○○○○

Commit and Push
○○○○

Browse Through History
○○○

More Functionalities
○○
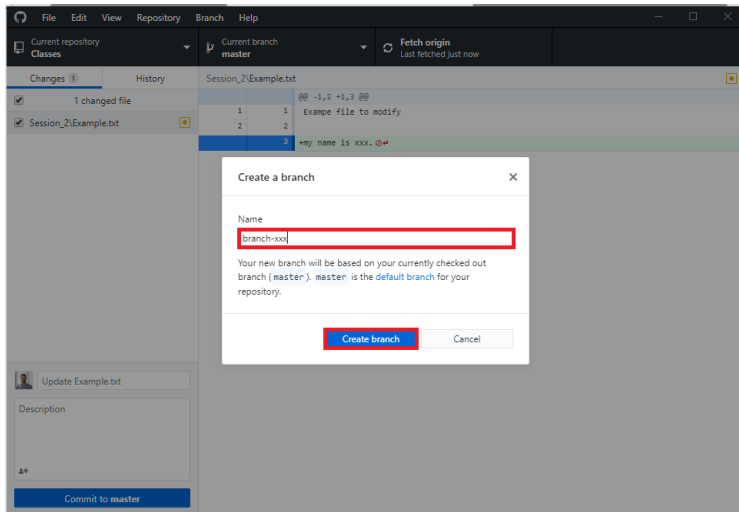
Step 2.D Switch Branch

Switch changes to your own branch.

Introduction
0000

First Steps With GitHub Desktop
00000000000●00000

Commit and Push
0000

Browse Through History
000

More Functionalities
00
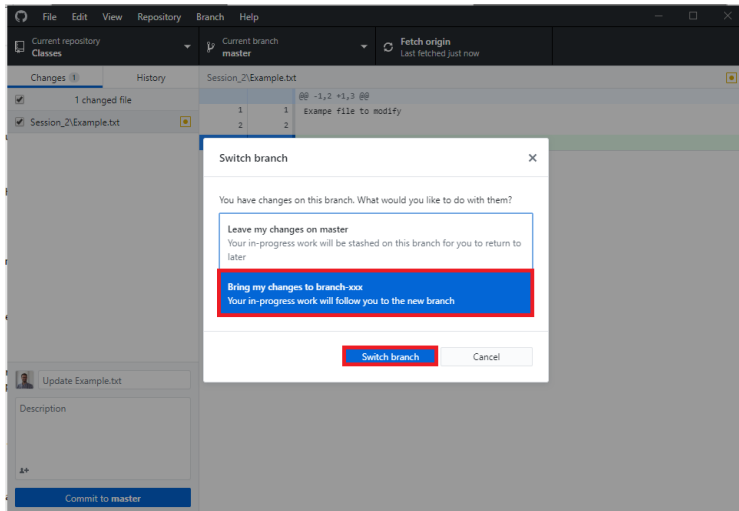
Step 2.E Publish Branch

Publish your branch online.

Introduction
oooo

First Steps With GitHub Desktop
ooooooooooo●oooo

Commit and Push
oooo

Browse Through History
ooo

More Functionalities
oo
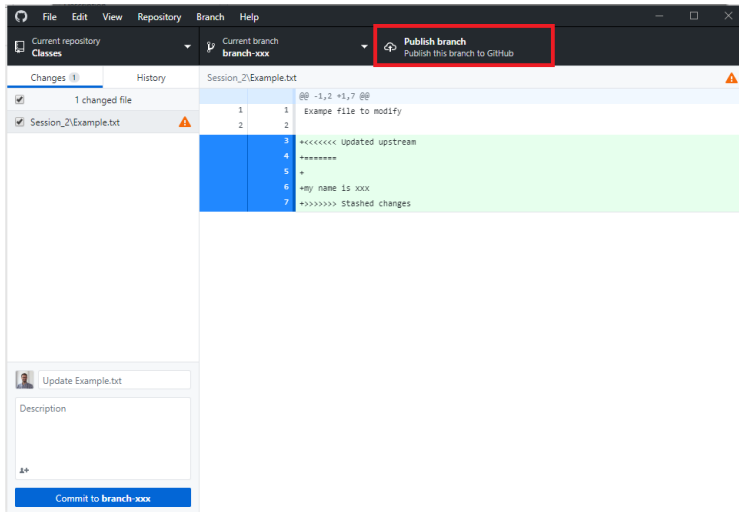
Step 2.E Commit to Branch

Give Description and summary. Then commit.

Introduction
○○○○

First Steps With GitHub Desktop
○○○○○○○○○○○○○●○○

Commit and Push
○○○○

Browse Through History
○○○

More Functionalities
○○
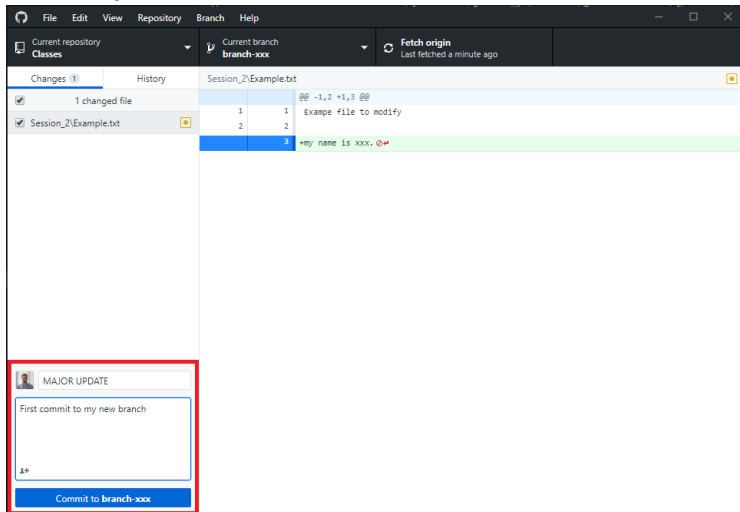
Step 2.E Push to Branch

Push changes to your branch.

Introduction
0000

First Steps With GitHub Desktop
0000000000000●00

Commit and Push
0000

Browse Through History
000

More Functionalities
00
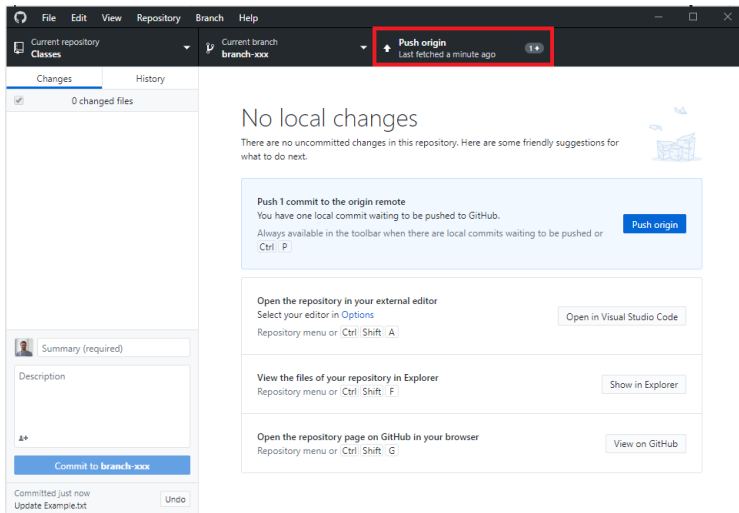
Step 3 Next Steps

Go to "Classes" page. Notice that now there are 2 branches. Click on "branches".

Introduction
○ ○ ○ ○

First Steps With GitHub Desktop
○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ● ○

Commit and Push
○ ○ ○ ○

Browse Through History
○ ○ ○

More Functionalities
○ ○

Step 3.A Done!

Your directory will now appear below "branches" as follows.

Introduction
0000

First Steps With GitHub Desktop
00000000000000●

Commit and Push
0000

Browse Through History
000

More Functionalities
00

Taking Stocks

▶ You are now able to create and maintain **your own repositories**.

▶ Please **make sure your only push changes to your own branch during this course**. You are encouraged to collaborate and see what other people is up to, but **never** commit changes directly to other people's branches.

▶ Good news is that you can always revert changes back if you do so by mistake. This is why GitHub is so useful!

Introduction
0000

First Steps With GitHub Desktop
00000000000000000

Commit and Push
●000

Browse Through History
000

More Functionalities
00

Commit and Push

Commit and Push

► **Committing** and **pushing** are the main two words you have to familiarize with.

► *Committing changes* to a branch, means that you are "saving" your changes.

► *Pushing changes* means, instead, that you are publishing them online on GitHub.Think of the pushing action as a way of creating different stable releases of your code.

► With this respect, we recomment to commit changes regurlarly (you can always revert them back), but to only push them online if you have made a stable change.

Introduction
0000

First Steps With GitHub Desktop
0000000000000000

Commit and Push
0000

Browse Through History
000

More Functionalities
00

Social Norms

▶ We encourage you to adopt the following standards to commit and push tidely.

▶ Summary should be either **Minor Change**, **Major Change** or **Bug Fixes**. The first should indicate small changes in syntaxis or general improvements. The second to major modifications (e.g. add new section or function), while the third is to notify that you have fixed some bug.

▶ **Description** should briefly explain what the summary refers to.

Introduction
0000

First Steps With GitHub Desktop
0000000000000000

Commit and Push
000●

Browse Through History
000

More Functionalities
00

Social Norms

▶ Suppose you **create a new function for data cleaning in your code**. When pushing this change to GitHub, you want to give **Major Change** as summary and "added function for data cleaning" as description.

▶ A tidy pushing activities will create a full history of changes in GitHub that you can scroll through to check different versions of your code.

▶ Finally, it will also help other people to understand your work.

Introduction
0000

First Steps With GitHub Desktop
0000000000000000

Commit and Push
0000

Browse Through History
●○○

More Functionalities
○○

Browse Through History

Browse Through History

With GitHub Desktop, you can also browse through the entire history of your commits. This is very helpful to

▶ **Revert back changes**. Imagine you update a code but, at some point, you realise that one of the previous releases was better.

▶ **Compare different versions**. From time to time, you may want to look back at previous versions of your work (slide, paper, code,...).

Browse Through History

▶ **A tidy committing activity will help you to easily browse through stable versions of your work**. If you commit to often, you have to search a lot (versions only differ marginally from one another). If you commit too infrequently, you may lose information.

▶ You find a nice tutorial on how to do both here.

Introduction
0000

First Steps With GitHub Desktop
000000000000000000

Commit and Push
0000

Browse Through History
000

More Functionalities
●○

More Functionalities

More Functionalities

► Communicate with other people (your coauthors, other developers, etc.) by creating **issues**. Look at here for how to do it.

► You find a nice tutorial on how to do both here.