

# 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  $> 100\text{MB}$ ). 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  
○○○○

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

Commit and Push  
○○○○

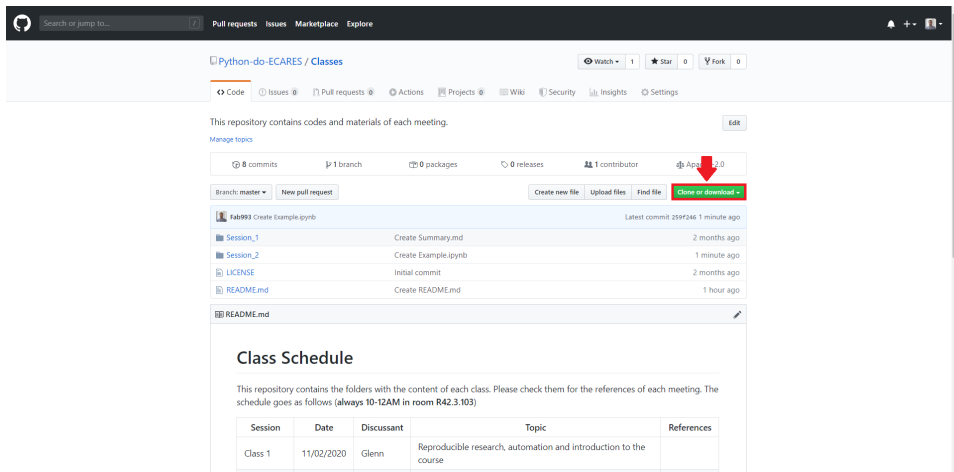
Browse Through History  
○○○

More Functionalities  
○○

# First Steps With GitHub Desktop

# Step 1. Clone Repository

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



Python-do-ECARES / Classes

Watch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

This repository contains codes and materials of each meeting. [Edit](#)

[Manage topics](#)

8 commits 1 branch 0 packages 0 releases 1 contributor 0 forks 2.0

Branch: master New pull request Create new file Upload files Find file **Clone or download**

Fab993 Create Example.ipynb Latest commit 259f246 1 minute ago

Session\_1 Create Summary.md 2 months ago

Session\_2 Create Example.ipynb 1 minute ago

LICENSE Initial commit 2 months ago

README.md Create README.md 1 hour ago

README.md

## Class Schedule

This repository contains the folders with the content of each class. Please check them for the references of each meeting. The schedule goes as follows (always 10-12AM in room R42.3.103)

Session	Date	Discussant	Topic	References
Class 1	11/02/2020	Glenn	Reproducible research, automation and introduction to the course	

# Step 1.A Open Repository

Choose "Open in Desktop".

The screenshot shows the GitHub web interface for the repository 'Python-do-ECARES / Classes'. The repository is currently on the 'master' branch. A modal window is open for cloning the repository, showing the 'Clone with SSH' option. The 'Open in Desktop' button is highlighted with a red box. Below the repository description, there is a 'Class Schedule' section with a table of class sessions.

Python-do-ECARES / Classes

Watch 1 Star 0 Fork 0

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

This repository contains codes and materials of each meeting. [Manage topics](#)

8 commits 1 branch 0 packages 0 releases 1 contributor Apache-2.0

Branch: master New pull request

Create new file Upload files Find file Clone or download

Clone with SSH Use HTTPS

You don't have any public SSH keys in your GitHub account. You can [add a new public key](#), or try cloning this repository via HTTPS.

Use a password protected SSH key.

git@github.com:Python-do-ECARES/Classes

Open in Desktop Download ZIP

### Class Schedule

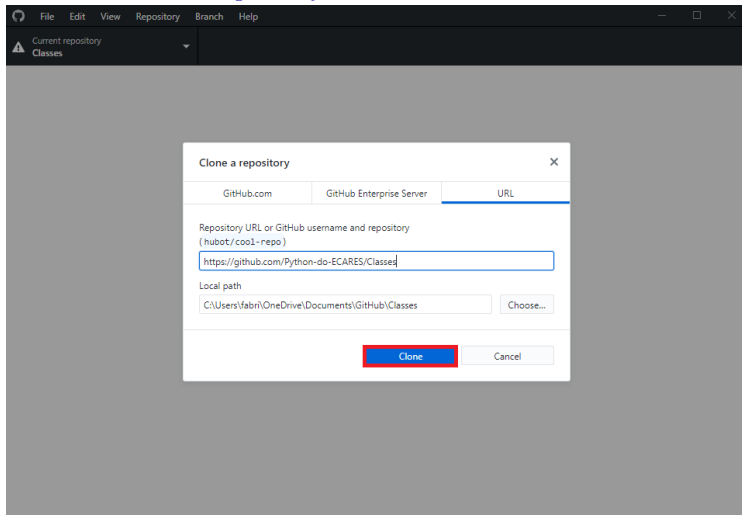
This repository contains the folders with the content of each class. Please check them for the references of each meeting. The schedule goes as follows (always 10-12AM in room R42.3.103)

Session	Date	Discussant	Topic	References
Class 1	11/02/2020	Glenn	Reproducible research, automation and introduction to the course	



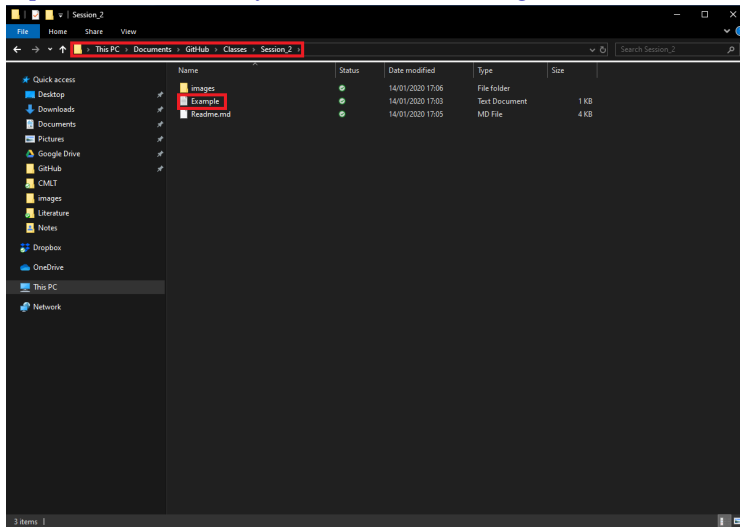
## Step 1.B Clone Repository

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



## Step 1.C Open Local Directory

Under GitHub Desktop/Classes/Session\_2 you will see the following files.



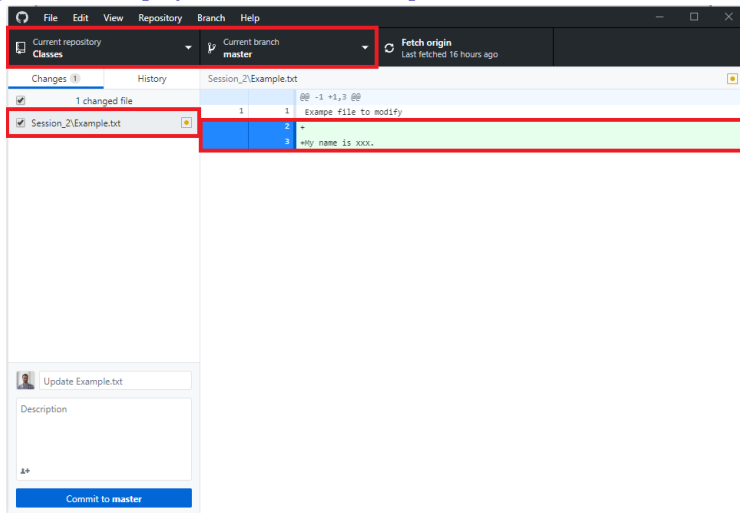
## Step 2 Make Changes

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



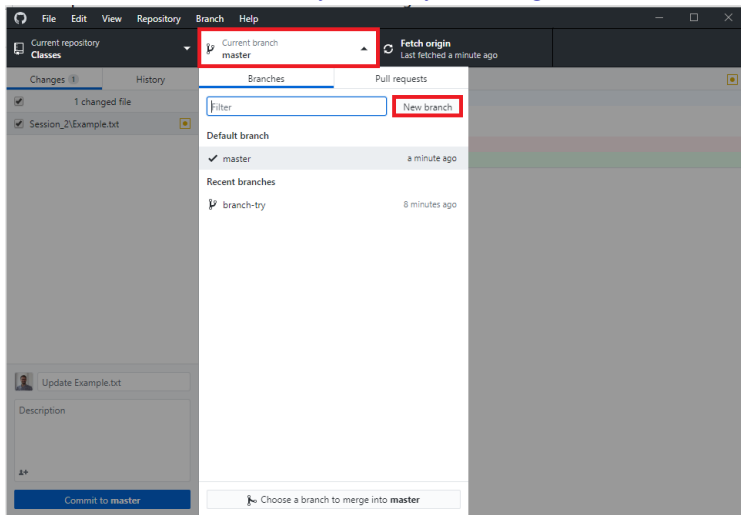
## Step 2.A Save Changes

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



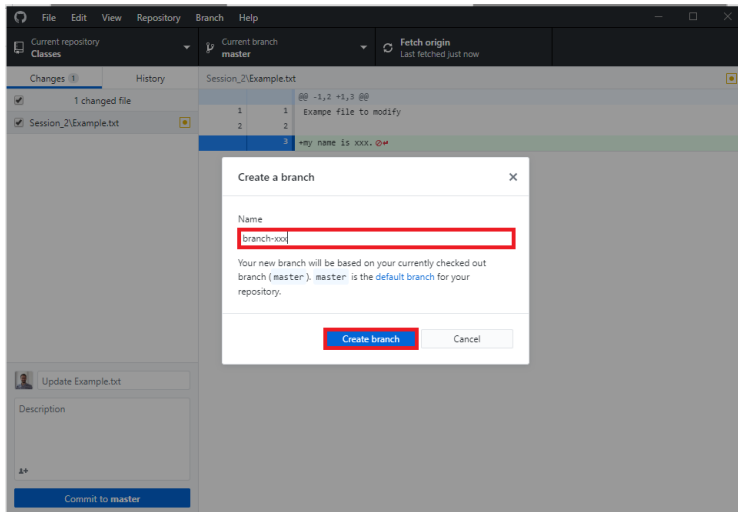
## Step 2.B Create Your Own Branch And Commit Changes

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



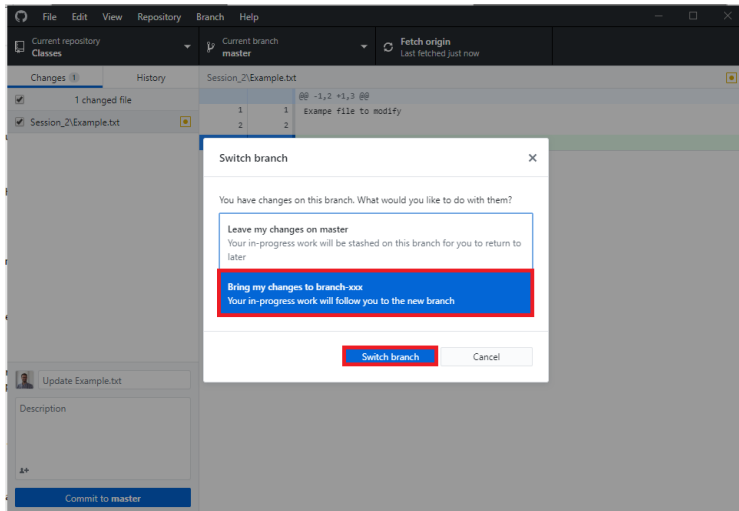
## Step 2.C Name Branch

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

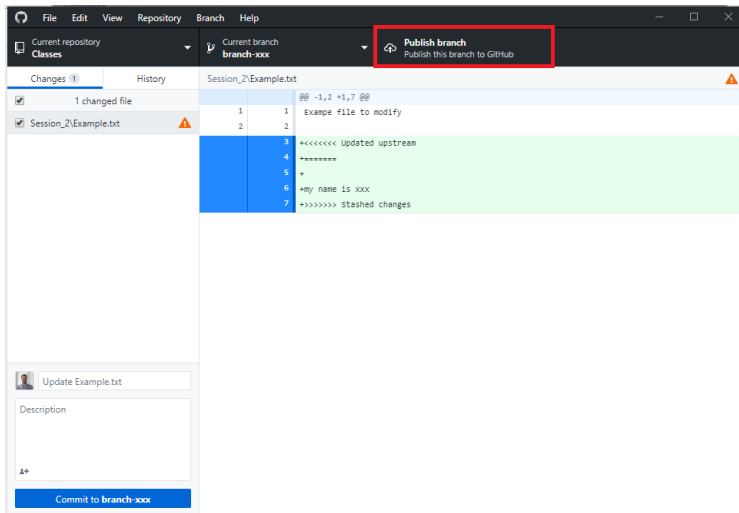


## Step 2.D Switch Branch

Switch changes to your own branch.



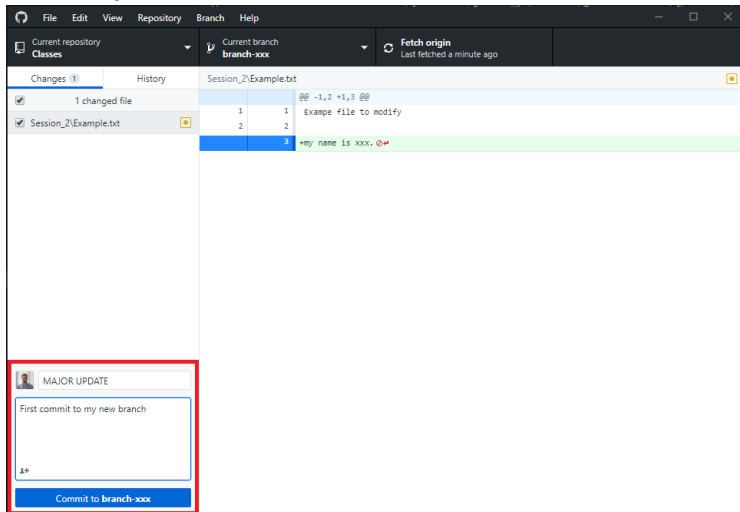
Publish your branch online.





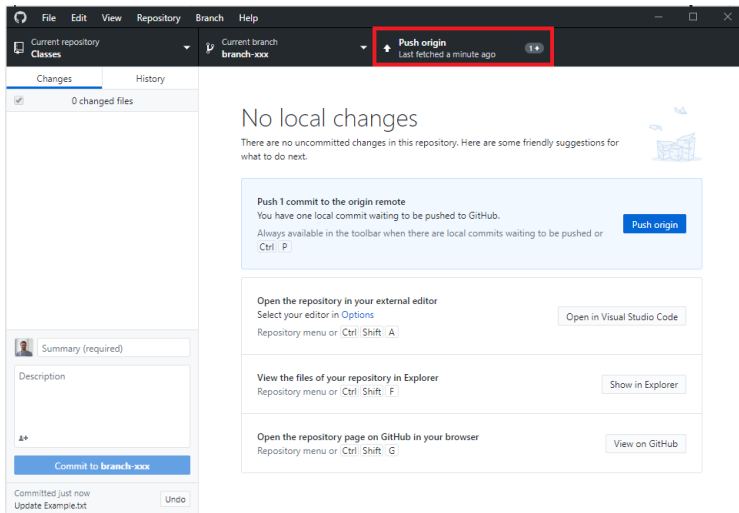
## Step 2.E Commit to Branch

Give Description and summary. Then commit.



## Step 2.E Push to Branch

Push changes to your branch.



## Step 3 Next Steps

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

Python-do-ECARES / Classes

49 commits **2 branches** 0 packages 0 releases 1 contributor Apache-2.0

Your recently pushed branches:

- branch-xxx (1 minute ago) [Compare & pull request](#)

Branch: master [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download](#)

File	Commit	Time
Session_1	Create Summary.md	2 months ago
Session_2	Update Readme.md	27 seconds ago
LICENSE	Initial commit	2 months ago
README.md	Create README.md	18 hours ago

### Class Schedule

This repository contains the folders with the content of each class. Please check them for the references of each meeting. The schedule goes as follows (always 10-12AM in room R42.3.103)

Session	Date	Discussant	Topic	References
Class 1	11/02/2020	Glenn	Reproducible research, automation and introduction to the course	

# Step 3.A Done!

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

The screenshot displays the GitHub web interface for the repository 'Python-do-ECARES / Classes'. The top navigation bar includes links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The repository's main page shows the 'Code' tab selected, with options for 'Issues', 'Pull requests', 'Actions', 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. Below the repository name, there are tabs for 'Overview', 'Yours', 'Active', 'Stale', and 'All branches'. The 'Overview' tab is active, showing the 'Default branch' as 'master' and 'Your branches' as 'branch-xxx'. The 'branch-xxx' branch is highlighted with a red box, indicating it is the current branch. The 'Active branches' section also shows 'branch-xxx' as the active branch. The bottom of the page features a footer with copyright information and links to 'Contact GitHub', 'Pricing', 'API', 'Training', 'Blog', and 'About'.

# 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  
○○○○

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

Commit and Push  
●○○○

Browse Through History  
○○○

More Functionalities  
○○

# 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 recommend to commit changes regularly (you can always revert them back), but to only push them online if you have made a stable change.

## 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.



## 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  
○○○

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

Commit and Push  
○○○○

**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,...).

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

# Browse Through History

- ▶ You find a nice tutorial on how to do both [here](#).

Introduction  
○○○○

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

Commit and Push  
○○○○

Browse Through History  
○○○

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](#).