

# Introduction to GitHub

Python-do-ECARES

Fabrizio Leone

<b>Introduction</b>	First steps	This Course	Standards	Browse History	Homework	More Functionalities	Exercises	Conclusion
●○○○○○	○○○○	○○○○○○○○○○○○○○○○	○○○○	○○○○○	○○○	○○	○○	○○

# Introduction

# Why GitHub, Why Here

- ▶ GitHub helps to achieve reproducibility.
- ▶ Reproducibility is key for credible empirical research.

## Some Background

- ▶ GitHub has long been the largest open source platforms for versioning. From 2018, it belongs to Microsoft.
- ▶ It has over 37M users.
- ▶ Most of the software and packages that you use are on GitHub. Python is among them.
- ▶ We use GitHub to organize our work throughout the course.

# 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. It is like a [social network, but for developers](#).
- ▶ 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 with unlimited co-authors.

## 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, .jl, .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).

# Realistic Workflow

Suppose you are beginning a new project. If you use GitHub, you can

1. Create a GitHub project with your favourite folders (e.g. code, slides, paper,...).
2. Work on your files locally and then save different versions of them on GitHub. (No more: paper\_v1, paper\_v1.A, paper\_v2.89.x7%,...).
3. Scroll through different versions of your files when you need. Collaborate with other people. (No more: paper\_v1\_myedition, paper\_v1\_youredition,...).



Introduction	First steps	This Course	Standards	Browse History	Homework	More Functionalities	Exercises	Conclusion
oooooo	●oooo	oooooooooooooooooooo	ooooo	oooooo	ooo	oo	oo	oo

# First steps

## Essential Vocabulary

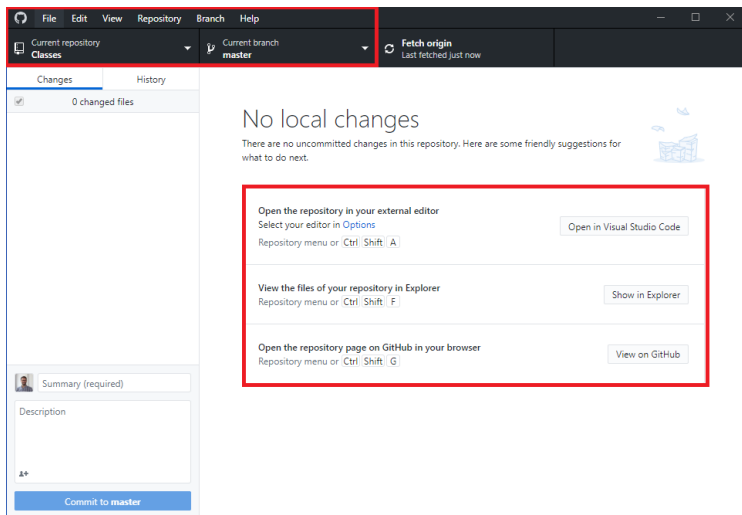
- ▶ **Repository** are the containers of your project. They can include folders, files, images, etc. Each repository is made of one or more branches.
- ▶ The default **branch** of your repository is called *master*. One can copy the master into other branches. In this way, you can work on multiple versions of the same file in parallel.
- ▶ **Commits** are changes you make to files contained in branches.
- ▶ Commits are only displayed locally until you **push** them to a branch online.
- ▶ You can **merge** branches by opening **pull** requests.

You can read more about these concepts [here](#).

# First Steps With GitHub Desktop

- ▶ You should already have (1) opened a GitHub account and (2) downloaded GitHub Desktop.
- ▶ If not, please do it now [here](#) and [here](#).

# Desktop Interface



## Desktop Interface

Please take a few moments to familiarise with GitHub Desktop. It allows you to

- ▶ Manage existing repositories, create versions, see history of changes, revert changes (this class).
- ▶ Create and manage new repositories (homework).
- ▶ Collaborate with other people (try it yourself).

Introduction	First steps	<b>This Course</b>	Standards	Browse History	Homework	More Functionalities	Exercises	Conclusion
oooooo	ooooo	●oooooooooooooooooooo	ooooo	oooooo	ooo	oo	oo	oo

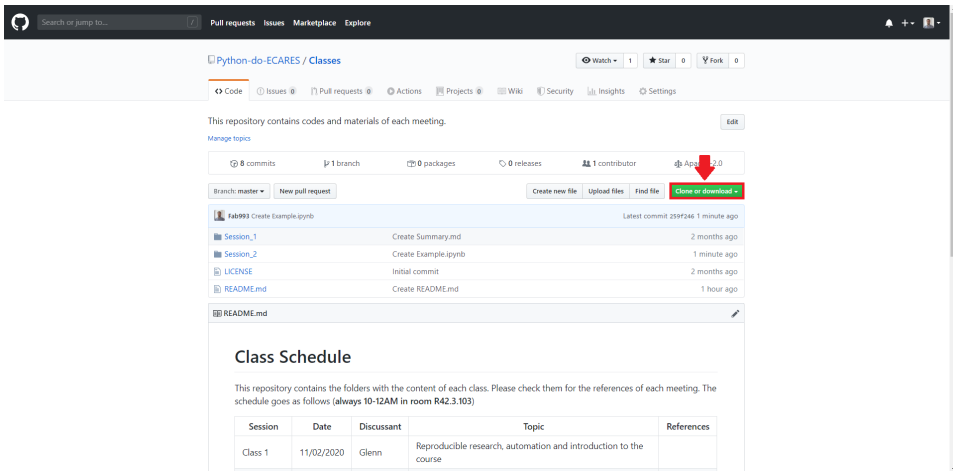
## This Course

# Use GitHub Desktop For This Course

- ▶ This section shows how to clone a repository, create your own branch and how to commit and push changes to it.
  
- ▶ Make sure to be familiar with each step. You will do this many times in the future.

# Step 1. Clone Repository

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





# Step 1.A. Open Repository

Choose "Open in Desktop".

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

Fab993 Create Example.ipynb

Session\_1 Create Summary.md

Session\_2 Create Example.ipynb

LICENSE Initial commit

README.md Create README.md

README.md

Clone with SSH

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

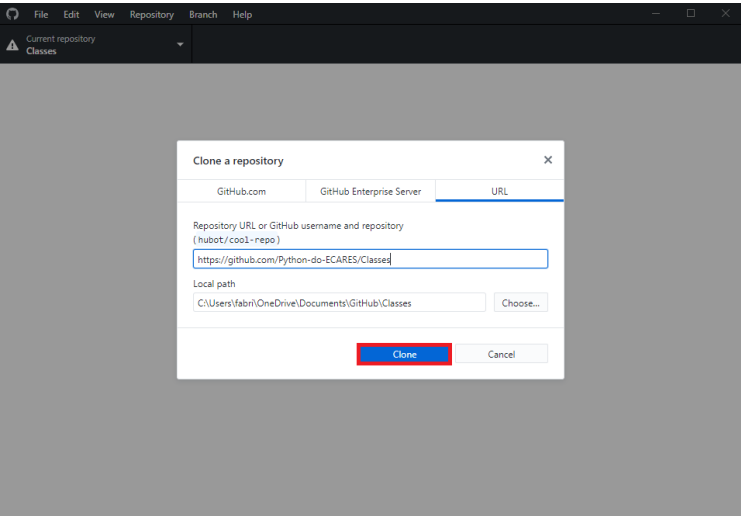
## Class Schedule

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	

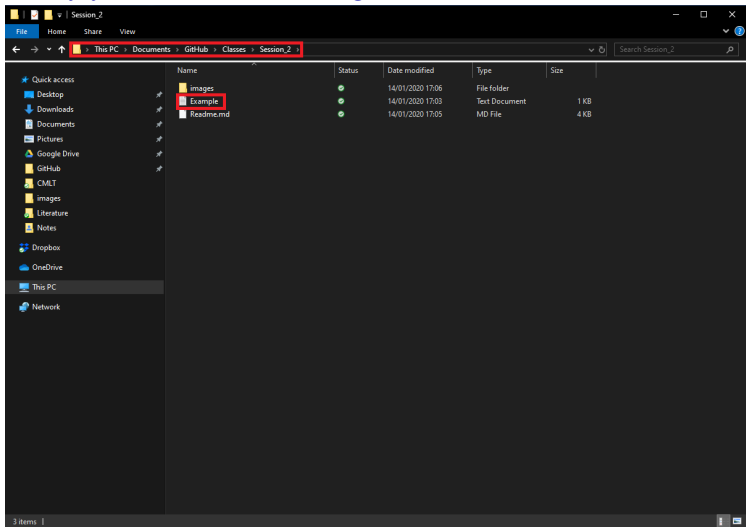
# Step 1.B. Clone Repository

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



# Step 1.C. Open Local Directory

Under the chosen directory, you will see the following files.



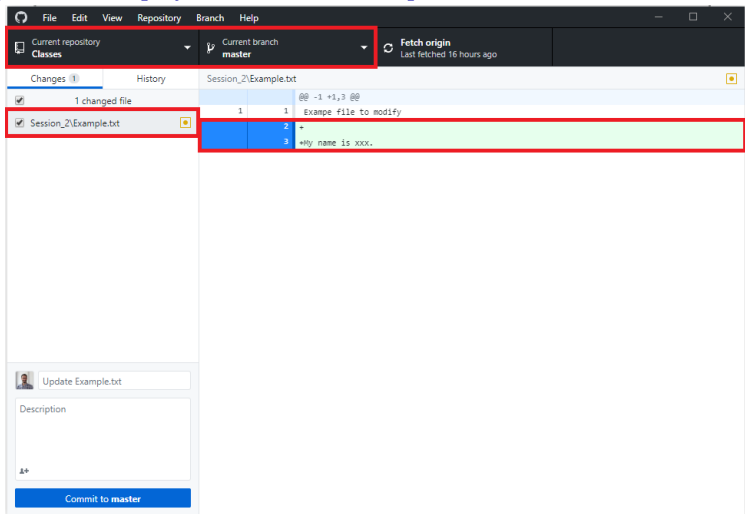
## Step 2. Make Changes

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



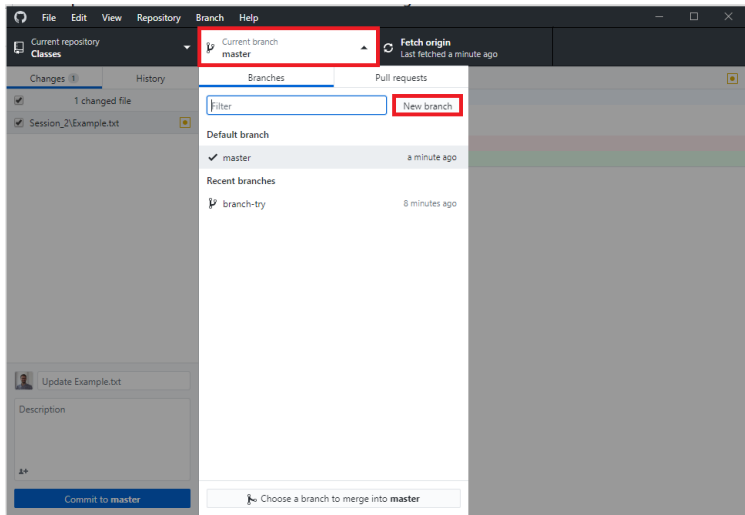
## Step 2.A. Save Changes

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



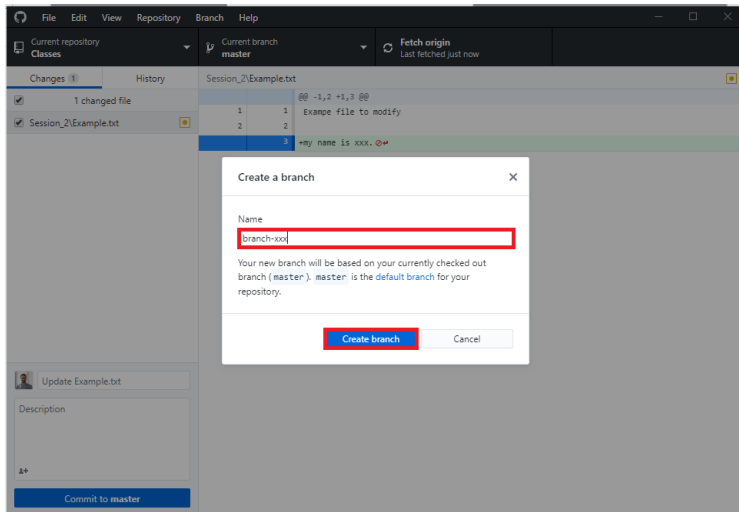
## Step 2.B. Create Your Own Branch And Commit Changes

Select "Master" and hit "New Branch".



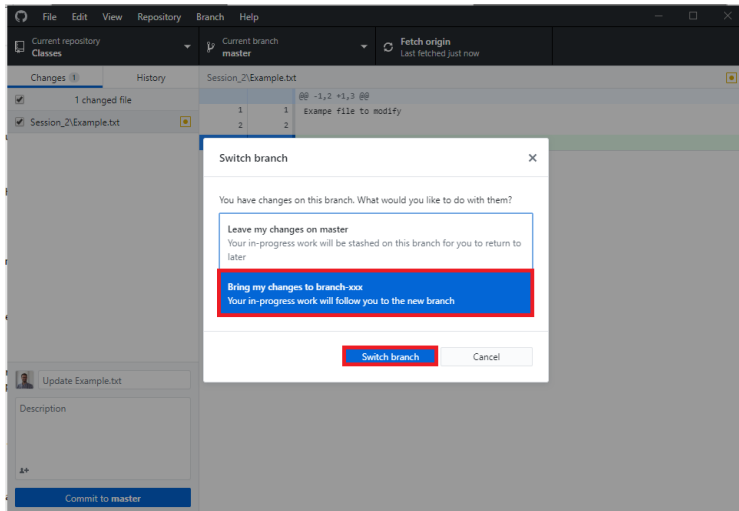
## Step 2.C. Name Branch

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



## Step 2.D. Switch Branch

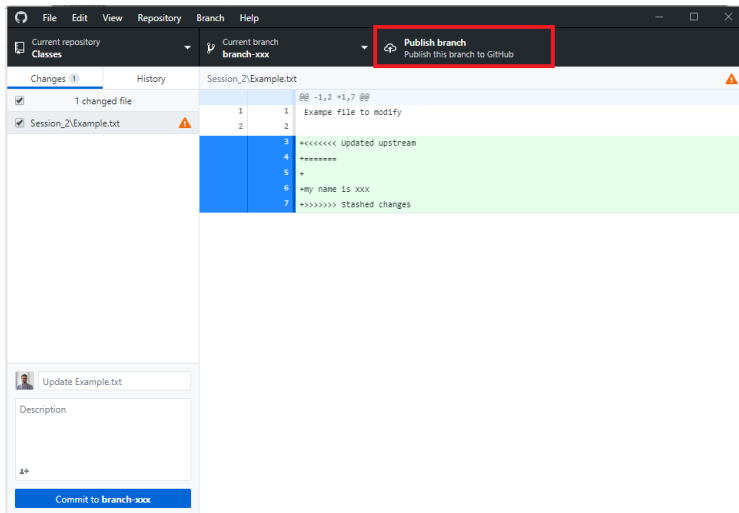
Switch changes to your own branch.





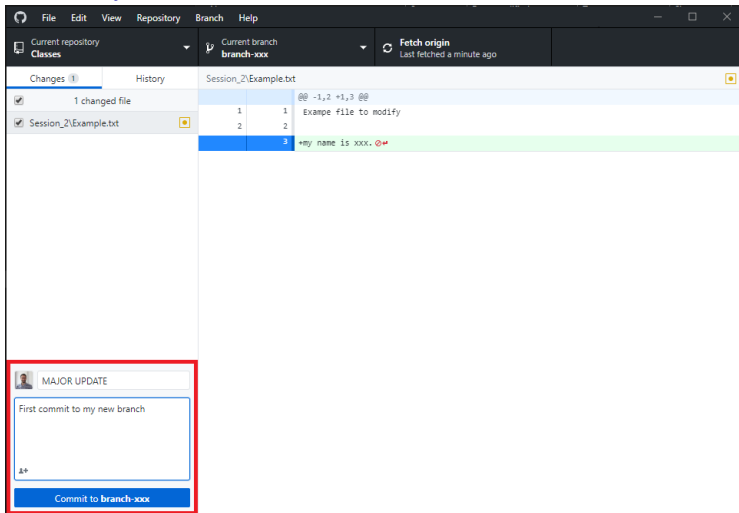
## Step 2.E. Publish Branch

Publish your branch online.



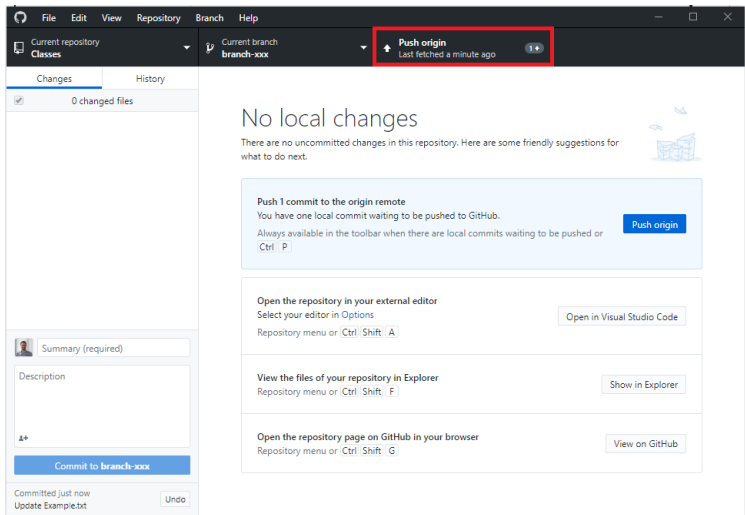
## Step 2.F. Commit to Branch

Give Description and summary. Then commit.



# Step 2.G. Push to Branch

Push changes to your own branch.



## Step 3. Next Steps

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

The screenshot shows the GitHub interface for the repository `Python-do-ECARES / Classes`. At the top, there are navigation links: `Code`, `Issues`, `Pull requests`, `Actions`, `Projects`, `Wiki`, `Security`, `Insights`, and `Settings`. Below these, a summary bar indicates `49 commits`, `2 branches` (highlighted with a red box), `0 packages`, `0 releases`, `1 contributor`, and `Apache-2.0` license. A section titled "Your recently pushed branches:" shows a branch named `branch-xxx` pushed 1 minute ago, with a `Compare & pull request` button. Below this is a table of recent commits:

Commit	Message	Time
<code>Feb993</code>	Update README.md	Latest commit Sec9f91 27 seconds ago
<code>Session_1</code>	Create Summary.md	2 months ago
<code>Session_2</code>	Update README.md	27 seconds ago
<code>LICENSE</code>	Initial commit	2 months ago
<code>README.md</code>	Create README.md	18 hours ago

Below the commit table is the `README.md` file content, which includes a section titled `Class Schedule`. The text states: "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 shows the GitHub interface for the repository `Python-do-ECARES / Classes`. The top navigation bar includes links for Pull requests, Issues, Marketplace, and Explore. The repository header shows 1 Watch, 0 Stars, and 0 Forks. The main content area has tabs for Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below these are filters for Overview (selected), Yours, Active, Stale, and All branches, along with a search bar. The 'Default branch' section shows the `master` branch, updated 1 minute ago by `Fab993`, with a 'Default' label and a 'Change default branch' button. The 'Your branches' section is highlighted with a red box and shows the `branch-xxx` branch, updated 5 minutes ago by `Fab993`, with a progress bar at 17/1 and a 'New pull request' button. The 'Active branches' section also shows the `branch-xxx` branch with the same details and a 'New pull request' button. The footer contains copyright information for GitHub, Inc. and links to Terms, Privacy, Security, Status, Help, Contact GitHub, Pricing, API, Training, Blog, and About.

## Taking Stocks

- ▶ You are now able to create and maintain **your own branch** for each session of this course.
- ▶ **You are not allowed to push to the master. If you try, you will get an error.** This is a useful feature if you manage a project and do not want collaborators to modify the master directly.
- ▶ Please make sure to **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	This Course	Standards	Browse History	Homework	More Functionalities	Exercises	Conclusion
ooooooo	ooooo	oooooooooooooooooooo	●oooo	oooooo	ooo	oo	oo	oo

# Standards

# Commit and Push

- ▶ **Committing** and **pushing** are two important actions that you have to familiarize with.
- ▶ *Committing changes* to a branch, means that you are creating a new version of your file.
- ▶ *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.
- ▶ Only push changes online if you have made a stable change.



# Commit and Local Saving

- ▶ Notice that **committing to GitHub** and **saving your file locally** are very different things.
- ▶ If you save locally, you only change the file on your device. If you commit, you add a “node” to the chain of your versions.
- ▶ With this respect, we recommend to commit changes regularly, but also to use **standards**.

# Commit Standards

- ▶ We encourage you to adopt the following standards to commit tidily.
- ▶ “Summary” should be either **Minor Change**, **Major Change** or **Bug Fixes**.  
The first should indicate small changes in syntax 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.

# Commit Standards

- ▶ 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 commit 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	This Course	Standards	Browse History	Homework	More Functionalities	Exercises	Conclusion
ooooooo	ooooo	oooooooooooooooooooo	ooooo	●ooooo	ooo	oo	oo	oo

## Browse History

## Browse Through History Of Changes

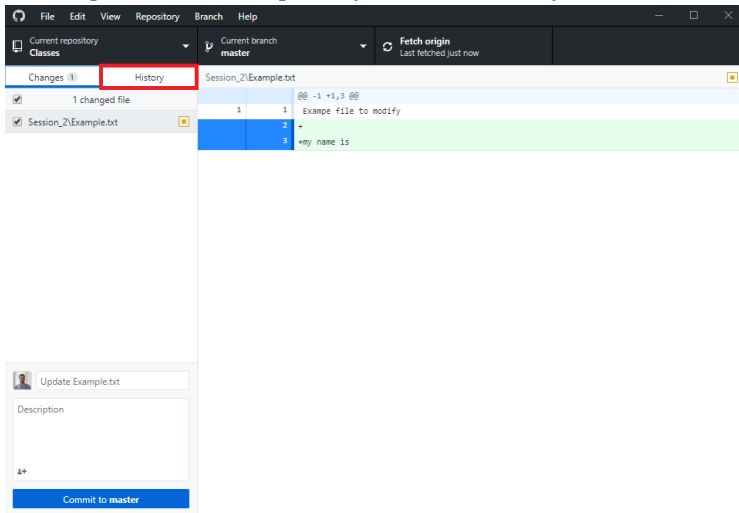
With GitHub Desktop, you can also browse through the entire history of your commits. This is very helpful to

- ▶ **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 meaningful 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.

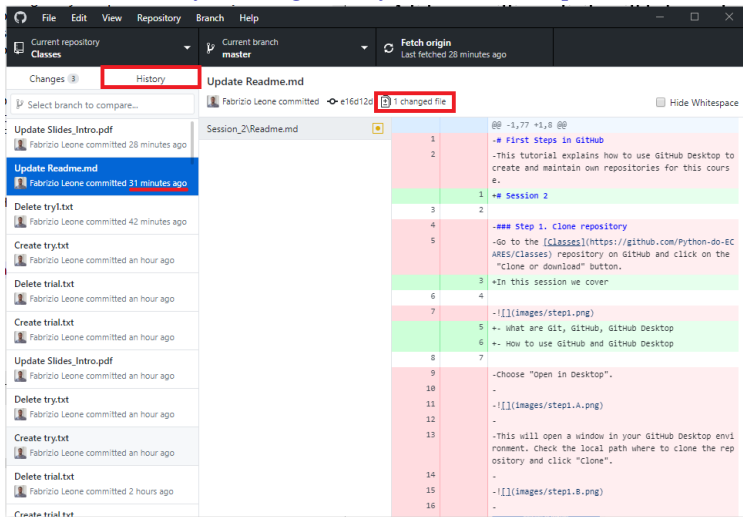
# Revert Back Changes

Step1. Open GitHub Desktop at the current repository. Then hit "History".



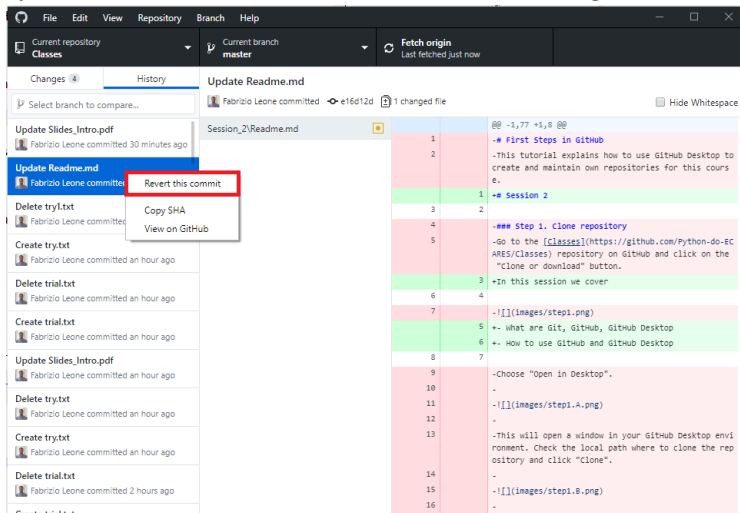
# Revert Back Changes

Step2. Scroll through the whole history of changes that you have made up to that moment.



# Revert Back Changes

Step3. Find the node you want to restore. Click on "Revert this commit" to go back to that version.





## More Details

- ▶ You can switch back and forth your versions as many times as you want.  
**Your local directories will change accordingly.**
- ▶ If instead of "Revert this commit" you hit "View on GitHub", you can see online all the differences between two versions of your file.

Introduction	First steps	This Course	Standards	Browse History	<b>Homework</b>	More Functionalities	Exercises	Conclusion
ooooooo	ooooo	oooooooooooooooooooo	ooooo	oooooo	●ooo	oo	oo	oo

# Homework

# Homework

- ▶ You now know the basics to manage your homework (and your research) with GitHub.
- ▶ If you have to **present**, please send your material to Glenn, Federico or myself after the meeting. We will upload it on the master.
- ▶ Do your homework before each meeting, and keep track of what you do using GitHub.

## GitHub For Homework

- ▶ We will upload the exercises for each session on GitHub. You can find them [here](#).
- ▶ After installing GitHub Desktop has created a folder called “GitHub” on your computer. Make sure to locate it conveniently for you.
- ▶ For each session, create a corresponding sub-folder inside it (e.g. Session\_1, Session\_2,...).
- ▶ Create and save there your solutions.
- ▶ While working on them, keep GitHub Desktop opened and get acquainted with commit, push and description actions.
- ▶ **Remember to commit to your own branch only.**

Introduction	First steps	This Course	Standards	Browse History	Homework	More Functionalities	Exercises	Conclusion
ooooooo	ooooo	oooooooooooooooooooo	ooooo	oooooo	ooo	●o	oo	oo

## More Functionalities

## More Functionalities

Other useful GitHub functionalities you may want to check

- ▶ Communicate with other people (your coauthors, other developers, etc.) by creating [issues](#).
- ▶ [Pull and merge](#) different branches. This is particularly useful for collaboration.
- ▶ [Automatic workflow](#). No need to manage your work manually.
- ▶ Create a [website](#).
- ▶ Create a [Project Board](#).
- ▶ GitHub integrations for [R Studio](#), [VS code](#) and [Atom](#) editors.
- ▶ [Gitignore](#) files.

Introduction	First steps	This Course	Standards	Browse History	Homework	More Functionalities	Exercises	Conclusion
ooooooo	ooooo	oooooooooooooooooooo	ooooo	oooooo	ooo	oo	●o	oo

# Exercises

# Exercises

Try to do the following at home.

- ▶ Familiarise with GitHub online. Search for people, projects and browse directories.
- ▶ Use GitHub Desktop to create a new directory on your own GitHub profile.
- ▶ Create a new branch and call it “branch-try”. Open an issue and assign it to yourself. Familiarize with mentions and comments. Then close the issue.
- ▶ Create a new file (e.g. .txt) on your branch. Then, use GitHub Desktop to open a pull request. After merging the two branches, delete “branch-try”.



Introduction	First steps	This Course	Standards	Browse History	Homework	More Functionalities	Exercises	Conclusion
ooooooo	ooooo	oooooooooooooooooooo	ooooo	oooooo	ooo	oo	oo	●o

# Conclusion

## Conclusion

- ▶ GitHub and GitHub Desktop are powerful tools that help you to organize and make your research reproducible.
- ▶ **GitHub features a bit of a learning curve.** First commits will be quite messy: you will likely commit too often, you will create too many branches and you will be tempted to modify things from the web interface rather than from GitHub Desktop (or the terminal).
- ▶ Before adopting your own standards, **see what more experienced people do.** There are plenty of web articles that suggest best practices and standards for GitHub.
- ▶ Use GitHub to organize your research from the very initial step until the end. Sooner or later you will master it.