

Current set-up



- Conda installed ✓
- Environment activated ✓
- Git installed ✓
- GitHub account set up ✓
- SSH keys generated and added to GitHub ✓

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

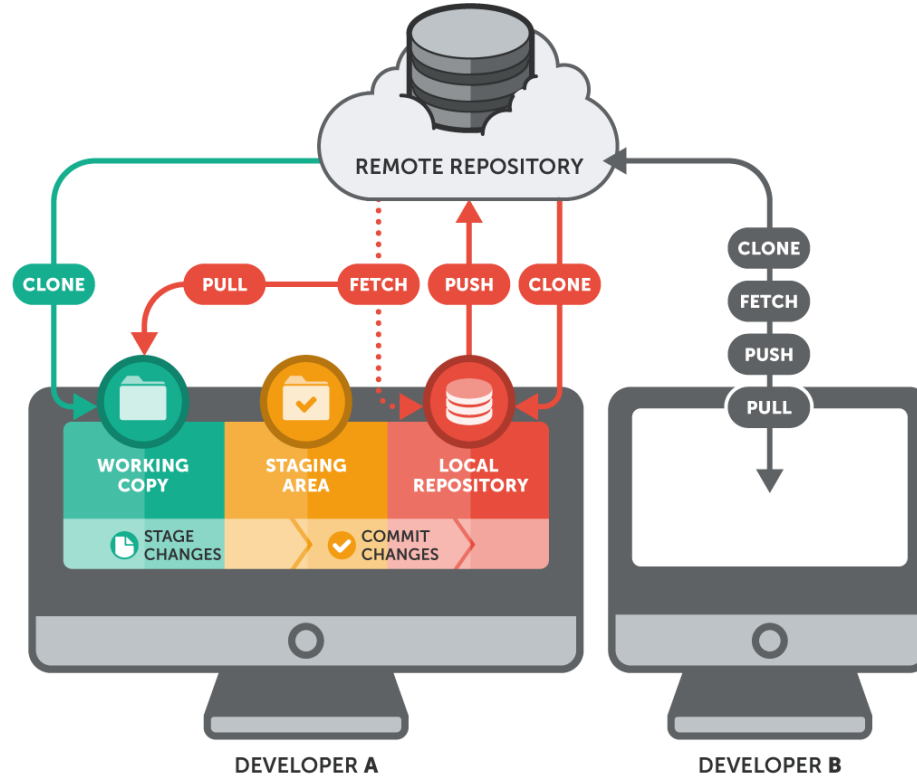
Slides: <https://github.com/jtrenadofqa/gitcourseUB>

Outline



- ✗ Git Basics
- ✗ Commits and commit history; undoing things
- ✗ Remote
- ✗ History

Git Structure



Git Basics



Now, let's create one locally.

- Create a directory: **mkdir test**
- Go inside: **cd test**
- Type: **git init**

```
(base) juan@dell:~/prueba/.git$ ls -a
.  ..  branches  config  description  HEAD  hooks  info  objects  refs
(base) juan@dell:~/prueba/.git$
```

- List all the files inside the folder: **ls -a**
- Add some dummy files to your folder: **touch filename**
- Add those to your local repo: **git status, git add filename, git commit -m "comment"**

You have created a local repo, but it is not connected to any remote server yet.

Git Basics



Now we have to connect our local repo with a remote one in GitHub



Once you have it created

- Type on your terminal inside your work directory:
git remote add origin git@github.com:username/repo_name.git
- Type: **git remote -v**

If you have any mistake in the url you can edit the config file in `.git/config`

- To connect both: **git push origin master**

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

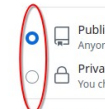
Owner *

jtrenadofga

Repository name *

Great repository names are short and memorable. Need inspiration? How about [silver-octo-palm-tree](#)?

Description (optional)



Public
Anyone on the internet can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with:

☐ Add a README file
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

☐ You are creating a public repository in your personal account.

Create repository

Lifecycle of your files v0.1



Untrack

Track

Unmodified

Modified

Unstaged

Staged

Commit



Lifecycle of your files v0.1



- Go to your working directory, it should already be synched with your remote repo
- Get the status of your project: **git status**

```
juan@hp:~/Escritorio/Universidad/Investigación/Presentaciones/Curso GIT 2025$ git status
On branch master
nothing to commit, working tree clean
juan@hp:~/Escritorio/Universidad/Investigación/Presentaciones/Curso GIT 2025$
```

- Create some dummy files: **touch dummyfile1, touch dummyfile2** (untracked files)
- Get the status of your project again: **git status**

```
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    dummyfile1
    dummyfile2

nothing added to commit but untracked files present (use "git add" to track)
```

Lifecycle of your files v0.1



- Add those files to your staging area: **git add dummyfile1** (dummyfile1 begins to be tracked)

```
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   dummyfile1

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        dummyfile2
```

- Write some dummy code (or some text) to one of your dummy files
- Status again: **git status** (now you have the modified files under the tracking section)

```
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   dummyfile1

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   dummyfile1

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        dummyfile2
```

**Your last
changes are
not staged**

Lifecycle of your files v0.1



- Again add the modified file to your staging area: **git add dummyfile1**
- Take a snapshot: **git commit -m "really good comment"**

```
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    dummyfile2

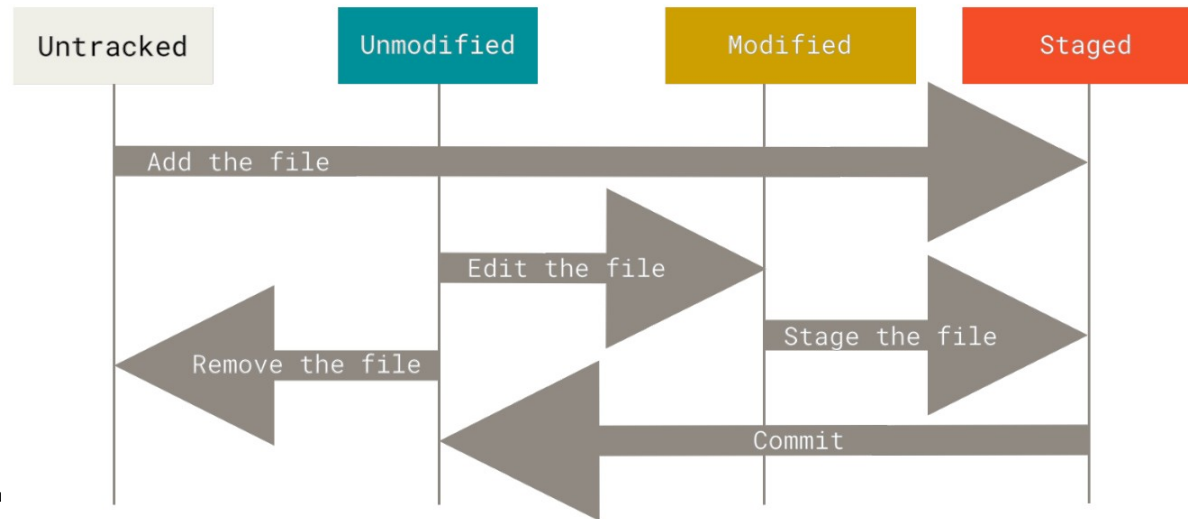
nothing added to commit but untracked files present (use "git add" to track)
```

Basics: file states



A file in Git can be one of three states:

- **Modified:** The file has changes, but they haven't been committed yet.
- **Staged:** The file is marked to be included in the next commit.
- **Committed:** The changes have been saved in your **local** repository.



Lifecycle of your files v0.2



- Go to your working directory
- Stage dummyfile2

```
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
       new file:   dummyfile2
```

- If you want to unstage a file: **git restore --staged dummyfile2**
- Get the status of your project again: **git status**

```
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
       dummyfile2

nothing added to commit but untracked files present (use "git add" to track)
```

Lifecycle of your files v0.3: .gitignore



- Create file called **not_in_git.txt**: **touch not_in_git.txt**
- Get the status of your project: **git status**

```
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    dummyfile2
    not_in_git.txt

nothing added to commit but untracked files present (use "git add" to track)
```

```
1 .*
2 *.odp
3 *.odt
4 Curso git.pdf
5 Git course - session 3.pdf
6 not_in_git.txt
```

If we don't want to track **not_in_git.txt** in our repo we can include it in .gitignore file.

- Get the status after including **not_in_git.txt** inside .gitignore: **git status**

```
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    dummyfile2

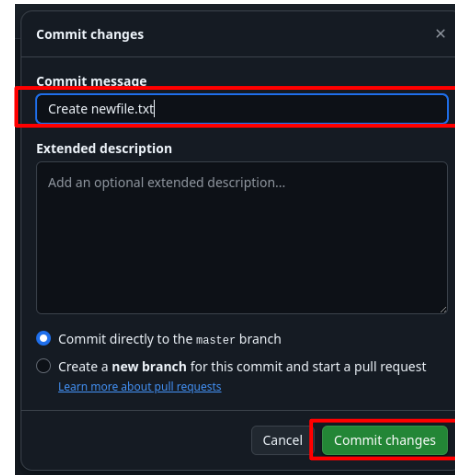
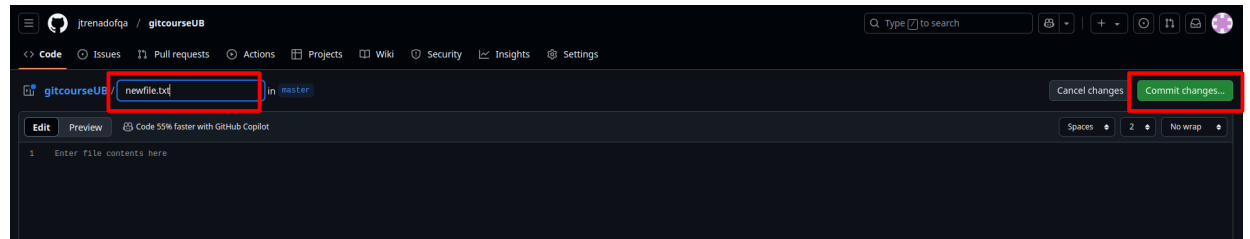
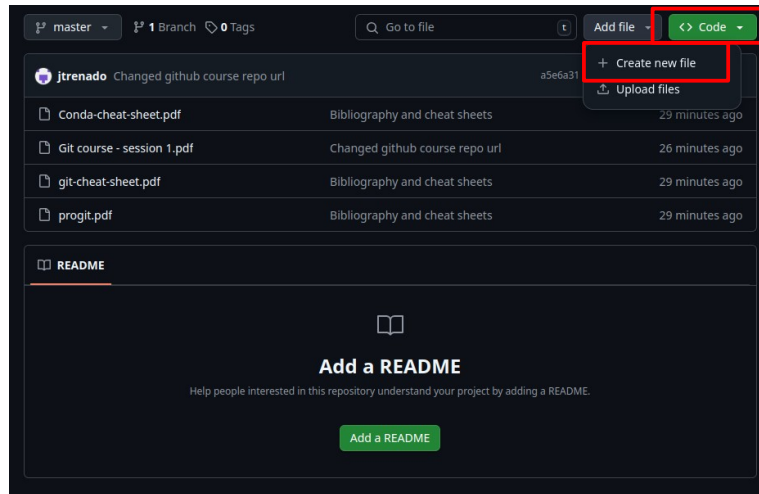
nothing added to commit but untracked files present (use "git add" to track)
```

Github has a list of templates for a huge variety of project types

Basics: create file in origin



- Go to your remote repository and create a new file in there.



Default commit message:
Create "newfile"... but you can
add an extended description.

Basics: fetch



- Check the status of your repository: **git status**

Changes in remote are not communicated in real time to local repos, to retrieve metadata for any change you have to fetch the repo.

- Fetch your repo: **git fetch**
- If you don't see any difference: **git branch --set-upstream-to=origin/master master**
- Check the status of your repository: **git status**

```
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each, respectively.
  (use "git pull" if you want to integrate the remote branch with yours)

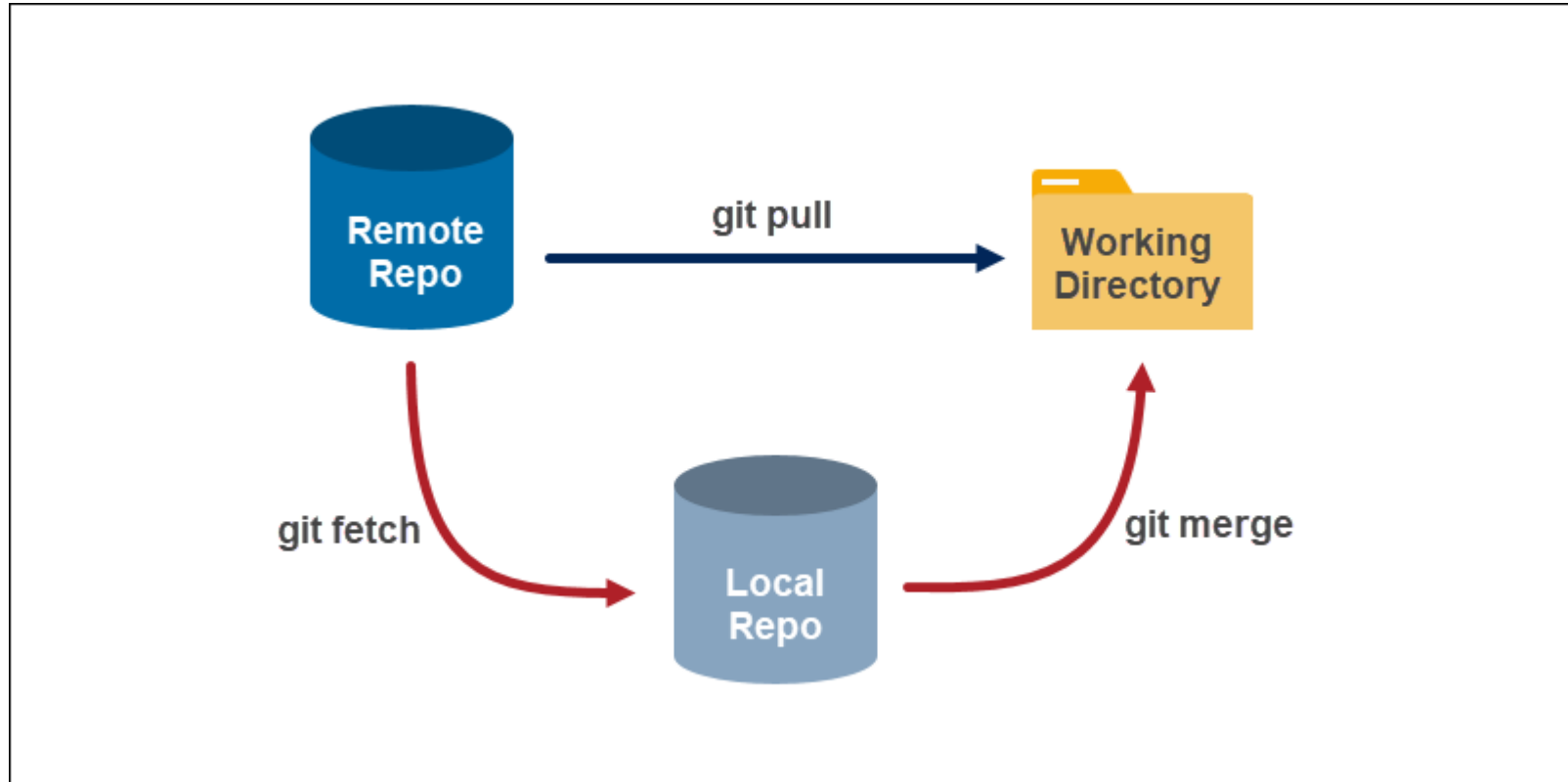
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  dummyfile2

nothing added to commit but untracked files present (use "git add" to track)
```



If you list your files you'll see that the remote file is not yet in the folder

Basics: pull=fetch+merge (or rebase)



Basics: pull=fetch+merge (or rebase)



- Get changes from origin: **git pull**
- If you have merging conflicts lets use merge by default: **git config pull.rebase false**

```
hint: You have divergent branches and need to specify how to reconcile them.
hint: You can do so by running one of the following commands sometime before
hint: your next pull:
hint:
hint:   git config pull.rebase false # merge
hint:   git config pull.rebase true  # rebase
hint:   git config pull.ff only      # fast-forward only
hint:
hint: You can replace "git config" with "git config --global" to set a default
hint: preference for all repositories. You can also pass --rebase, --no-rebase,
hint: or --ff-only on the command line to override the configured default per
hint: invocation.
```

- You should have now in your local repo the file created in remote.
- Check the status of your repository: **git status**

```
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  dummyfile2

nothing added to commit but untracked files present (use "git add" to track)
```


Basics: delete files



- Remove files from your project
 - 1st method:
 - Remove first from your working directory: **rm "dummyfile1"**
 - Stage the file: **git rm filename**
 - Commit the deleted file: **git commit -m "dummyfile1"**
 - Push changes: **git push**
 - 2nd method, git removes and stages at the same time.
 - Remove and stage using git: **git rm "dummyfile1"**
 - Commit the deleted file: **git commit -m "dummyfile1"**
 - Push changes: **git push**

Basics: Let's check history



- Check the history of your repo: **git log**

Last commit and
current position
(HEAD)

```
commit cf6ead4164d8ddb238f0e1ecfee1c0cef12be75 (HEAD -> master, origin/master)
Author: Juan <jtrenado@gmail.com>
Date: Tue Mar 11 15:17:41 2025 +0100    Author and email

    newfile removed

commit b49c6591caccf2bf50e6dadd8e70bbcecd4c2c42
Author: Juan <jtrenado@gmail.com>
Date: Tue Mar 11 15:17:15 2025 +0100

    dummyfile removed

commit 7a33fa812b469893f0635a63e7a1dd027b58757f Checksum or unique ID
Merge: adf062d b76add8                  for the DB
Author: Juan <jtrenado@gmail.com>
Date: Tue Mar 11 15:08:01 2025 +0100

    Merge branch 'master' of github.com:jtrenadofqa/gitcourseUB

commit b76add84953a1a9f83f0081e8de82c8cf35f01de
Author: jtrenadofqa <86599774+jtrenadofqa@users.noreply.github.com>
Date: Tue Mar 11 13:45:27 2025 +0100

    Create newfile.txt    Date and time of the commit

commit adf062d22d3b58351e8eba7805a7f15090d7a391
Author: Juan <jtrenado@gmail.com>
Date: Tue Mar 11 13:29:25 2025 +0100

    Committing dummy files
```

- Compact and useful version
of your log: **git log --oneline**

```
cf6ead4 (HEAD -> master, origin/master) newfile removed
b49c659 dummyfile removed
7a33fa8 Merge branch 'master' of github.com:jtrenadofqa/gitcourseUB
b76add8 Create newfile.txt
adf062d Committing dummy files
a5e6a31 Changed github course repo url
0ddb319 Changed github course repo url
5aa9d85 Bibliography and cheat sheets
4ebf57c Day 1, slides
```

Basics: log + diff



- Variations of git log: **git log -p -"Number"** (show only last "Number" entries)
log patch shows differences between commits

```
juan@Dell:~/test/refactoring/nr_eob_ub$ git log -p -2
commit f40e946960c429801fd09632cd81299934a02042 (HEAD -> refactoring_ecc, origin/refactoring_ecc)
Author: Juan <jtrenado@fga.ub.edu>
Date:   Wed Feb 8 14:50:53 2023 +0100

    EOBsim to simulate EOB without NR paths, EOBsim_from_NR inherits from EOBsim to generate EOB simulations from NR

diff --git a/nr_eob_ub/sim/EOBsim.py b/nr_eob_ub/sim/EOBsim.py
index 89614dc..41b8e40 100644
--- a/nr_eob_ub/sim/EOBsim.py
+++ b/nr_eob_ub/sim/EOBsim.py
@@ -1,42 +1,29 @@
import numpy as np
-from nr_eob_ub.nr_post import nr_reader
+from nr_eob_ub.nr_post import nr_reader
 from nr_eob_ub.eob_post import eob_reader
 from nr_eob_ub.eob_post.generator import eob_generator
 from nr_eob_ub.nr_post import gw_utils
 from nr_eob_ub.nr_post import gw_signals
-from nr_eob_ub.db import nrfilter

"""
+Class to generate EOB simulations from previous NR simulations
+EOB parent class to generate EOB simulations
"""

class EOBsim:
-    def __init__( self, path, list_modes = "all", leading_mode = (2, 2),
-                  dE = 0, dJ = 0, **kwargs ):
+    def __init__( self, path, list_modes = "all", leading_mode = (2, 2), **kwargs ):
         self.path = path
-    def __init__( self, list_modes = "all", leading_mode = (2, 2), **kwargs ):
+    def __init__( self, list_modes = "all", leading_mode = (2, 2), **kwargs ):
         self.leading_mode = leading_mode
         self.indices_to_compute = list_modes
         self.adm_param = nrfilter.get_ADM_param( self.path )
```

a: source file
b: destination file
-: identification for source file
+: identification for destination file
@@ -1,42 +1,29 @@ data below (chunk)
represents source file from line 1 and includes 42 lines, AND destination file from line 1 and includes 29 lines.
White lines: lines from a/ and b/.
Red lines: lines from a/.
Green lines: lines from b/.

- Check each version file using index hash: **git show "index hash"**