Introduction to Git

Questions you should answer before using Git

- What is a VCS.
- Differences between CVCS (SVN) and DVCS (git).
- When we choose to use CVCS or DVCS.

As a physicist, do you really need a version control system?

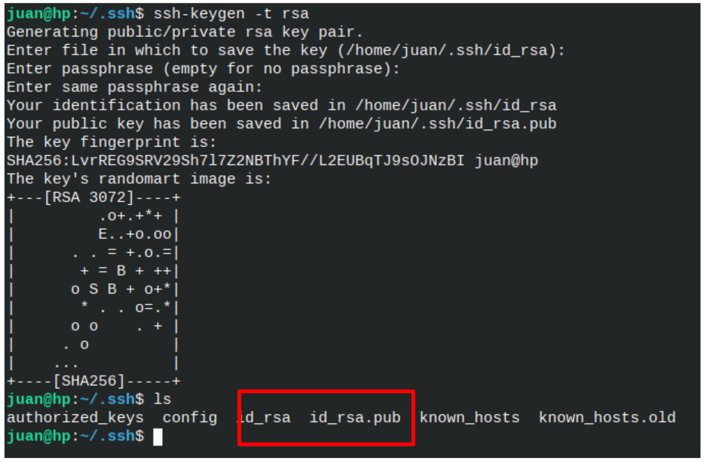
Introduction to Git

For the rest of this workshop you need to have git installed, a github account already created and ssh-keys activated to communicate with your repos.

- What is ssh protocol, ssh-keys and how to generate them.
- How to add a ssh-key to your github account

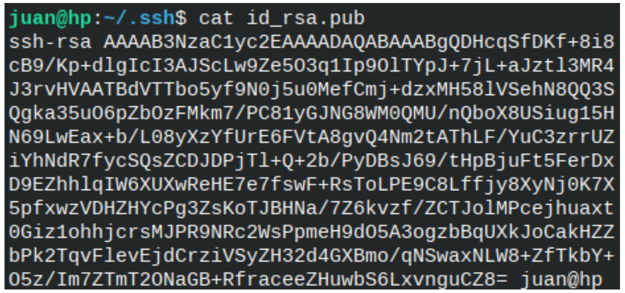
ssh-keys

- To create ssh keys: ssh-keygen -t rsa



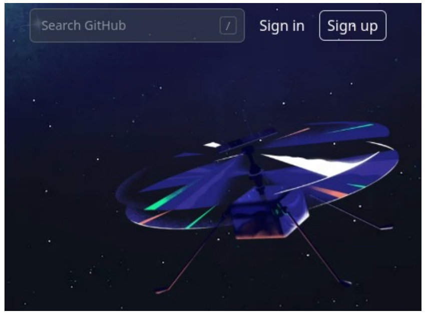
```
juan@hp:~/.ssh$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/juan/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/juan/.ssh/id_rsa
Your public key has been saved in /home/juan/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:LvrREG9SRV29Sh7l7Z2NBThYF//L2EUBqTJ9sOJNzBI juan@hp
The key's randomart image is:
+---[RSA 3072]----+
|         .o+.+*+ |
|         E..+o.oo|
|      . . = +.o.=|
|       + = B + ++|
|      o S B + o+*|
|       * . . o=.*|
|      o o    . + |
|     . o         |
|    ...          |
+----[SHA256]-----+
juan@hp:~/.ssh$ ls
authorized_keys  config  id_rsa  id_rsa.pub  known_hosts  known_hosts.old
juan@hp:~/.ssh$ 
```

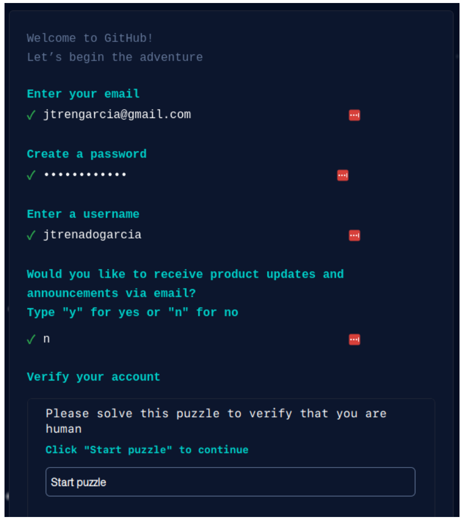
- You have to copy your public key to your github account: cat id_rsa.pub



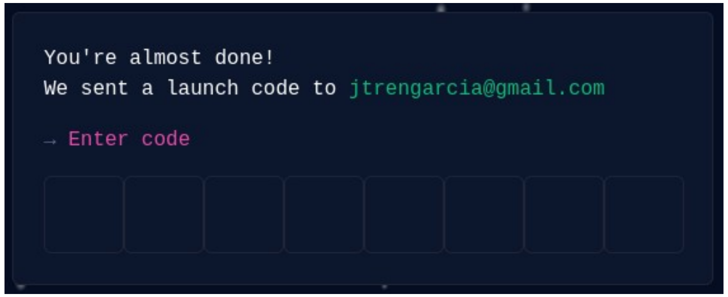
```
juan@hp:~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDHcqSfDKf+8i8
cB9/Kp+dlgIcI3AJScLw9Ze5O3q1Ip9OlTYpJ+7jL+aJztl3MR4
J3rvHVAATBdVTTbo5yf9N0j5u0MefCmj+dzxMH58lVSehN8QQ3S
Qgka35uO6pZbOzFMkm7/PC81yGJNG8WM0QMU/nQboX8USiug15H
N69LwEax+b/L08yXzYfUrE6FVtA8gvQ4Nm2tAThLF/YuC3zrrUZ
iYhNdR7fycSQsZCDJDPjTl+Q+2b/PyDBsJ69/tHpBjuFt5FerDx
D9EZhhlqIW6XUXwReHE7e7fswF+RsToLPE9C8Lffjy8XyNj0K7X
5pfxwzVDHZHYcPg3ZsKoTJBHNa/7Z6kvzf/ZCTJolMPcejhuaxt
0Giz1ohhjcrsMJPR9NRc2WsPpmeH9dO5A3ogzbBqUXkJoCakHZZ
bPk2TqvFlevEjdCrziVSyZH32d4GXBmo/qNSwaxNLW8+ZfTkbY+
O5z/Im7ZTmT2ONaGB+RfraceeZHuwbS6LxvnguCZ8= juan@hp
```

Create a github account

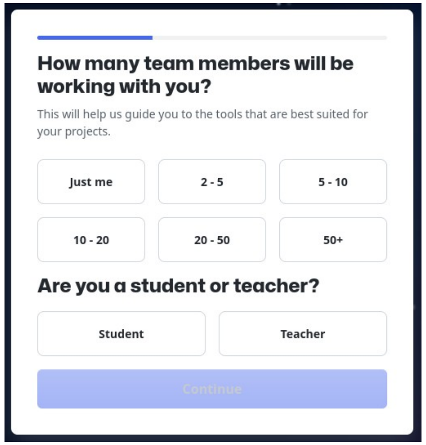




```
Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ jtrengarcia@gmail.com

Create a password
✓ ••••••••••••

Enter a username
✓ jtrenadogarcia

Would you like to receive product updates and
announcements via email?
Type "y" for yes or "n" for no
✓ n

Verify your account

  Please solve this puzzle to verify that you are
  human
  Click "Start puzzle" to continue
```



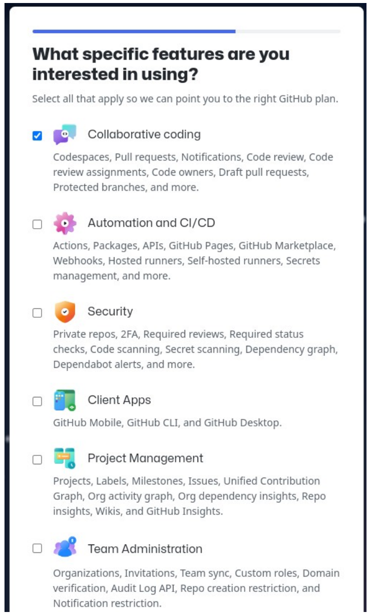
```
You're almost done!
We sent a launch code to jtrengarcia@gmail.com

→ Enter code
```



How many team members will be working with you?

This will help us guide you to the tools that are best suited for your projects.

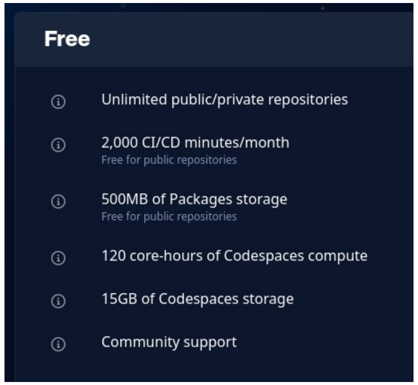
Are you a student or teacher?



What specific features are you interested in using?

Select all that apply so we can point you to the right GitHub plan.

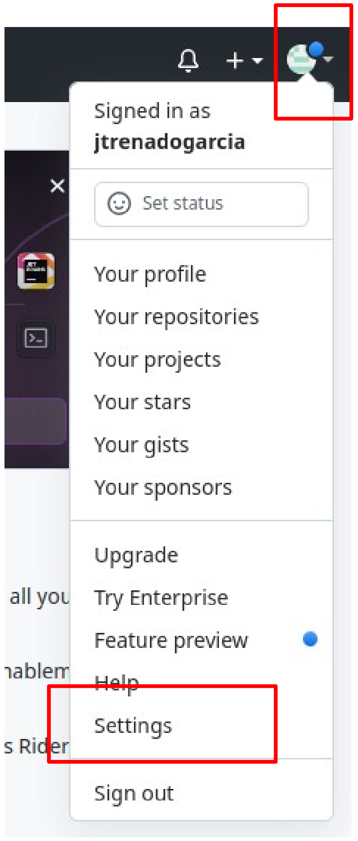
- [x] Collaborative coding
  Codespaces, Pull requests, Notifications, Code review, Code review assignments, Code owners, Draft pull requests, Protected branches, and more.
- [ ] Automation and CI/CD
  Actions, Packages, APIs, GitHub Pages, GitHub Marketplace, Webhooks, Hosted runners, Self-hosted runners, Secrets management, and more.
- [ ] Security
  Private repos, 2FA, Required reviews, Required status checks, Code scanning, Secret scanning, Dependency graph, Dependabot alerts, and more.
- [ ] Client Apps
  GitHub Mobile, GitHub CLI, and GitHub Desktop.
- [ ] Project Management
  Projects, Labels, Milestones, Issues, Unified Contribution Graph, Org activity graph, Org dependency insights, Repo insights, Wikis, and GitHub Insights.
- [ ] Team Administration
  Organizations, Invitations, Team sync, Custom roles, Domain verification, Audit Log API, Repo creation restriction, and Notification restriction.

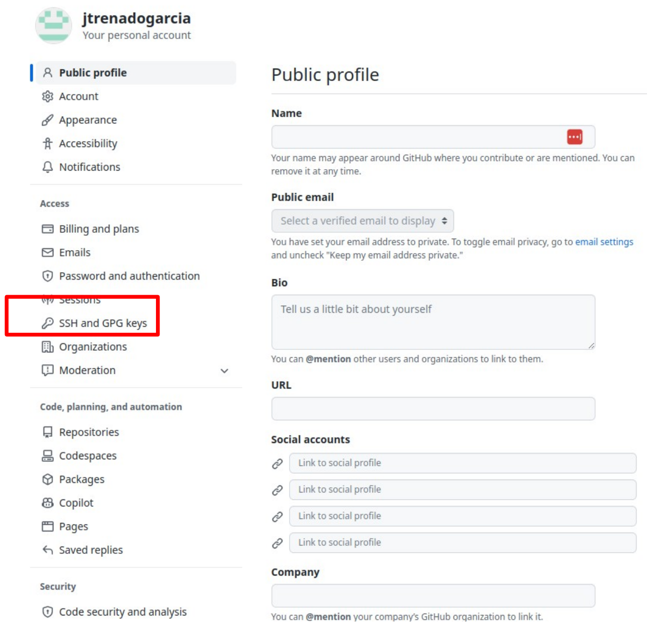


Free

- Unlimited public/private repositories
- 2,000 CI/CD minutes/month
  Free for public repositories
- 500MB of Packages storage
  Free for public repositories
- 120 core-hours of Codespaces compute
- 15GB of Codespaces storage
- Community support

Add your pub key to you github account





Public profile

Name

Your name may appear around GitHub where you contribute or are mentioned. You can remove it at any time.

Public email

Select a verified email to display

You have set your email address to private. To toggle email privacy, go to email settings and uncheck "Keep my email address private."

Bio

Tell us a little bit about yourself

You can @mention other users and organizations to link to them.

URL

Social accounts

Link to social profile

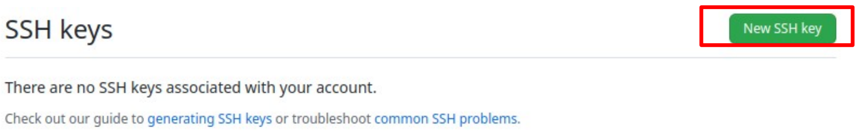
Link to social profile

Link to social profile

Link to social profile

Company

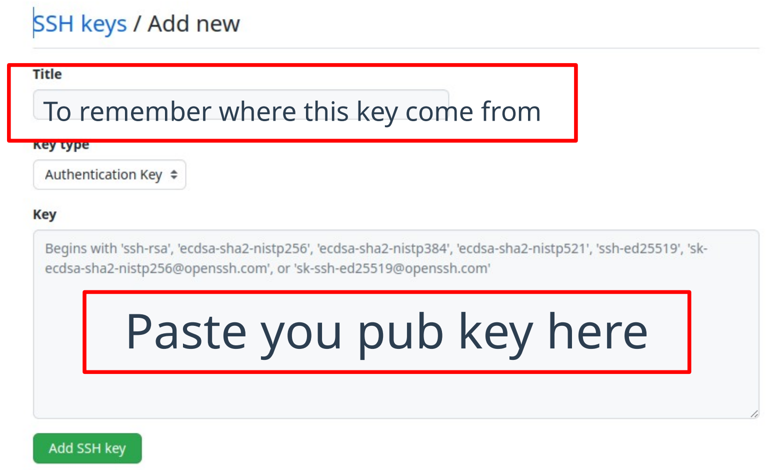
You can @mention your company's GitHub organization to link it.



SSH keys

There are no SSH keys associated with your account.

Check out our guide to generating SSH keys or troubleshoot common SSH problems.



SSH keys / Add new

Title

To remember where this key come from

Key type

Authentication Key

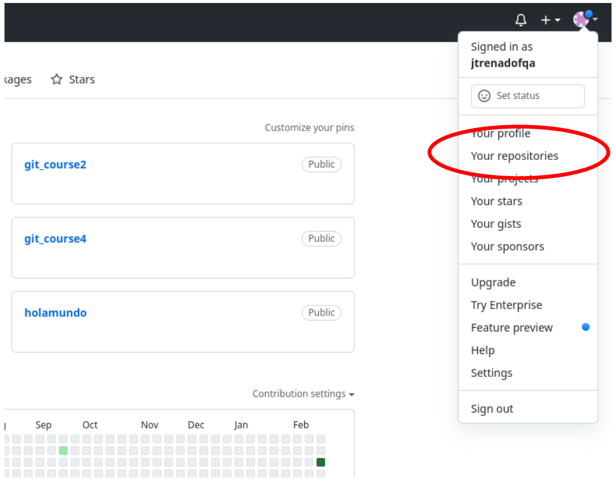
Key

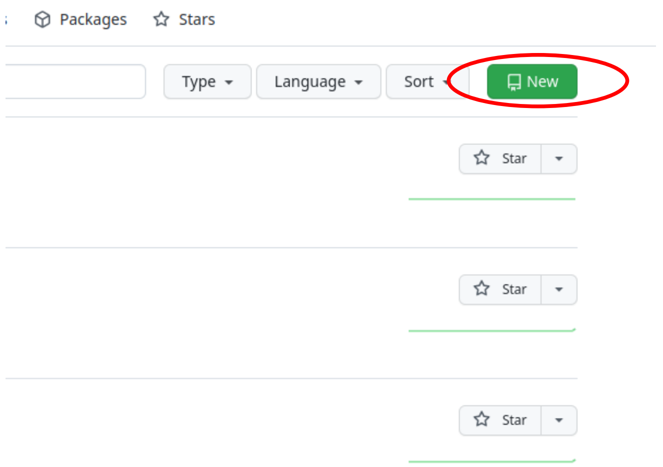
Begins with 'ssh-rsa', 'ecdsa-sha2-nistp256', 'ecdsa-sha2-nistp384', 'ecdsa-sha2-nistp521', 'ssh-ed25519', 'sk-ecdsa-sha2-nistp256@openssh.com', or 'sk-ssh-ed25519@openssh.com'

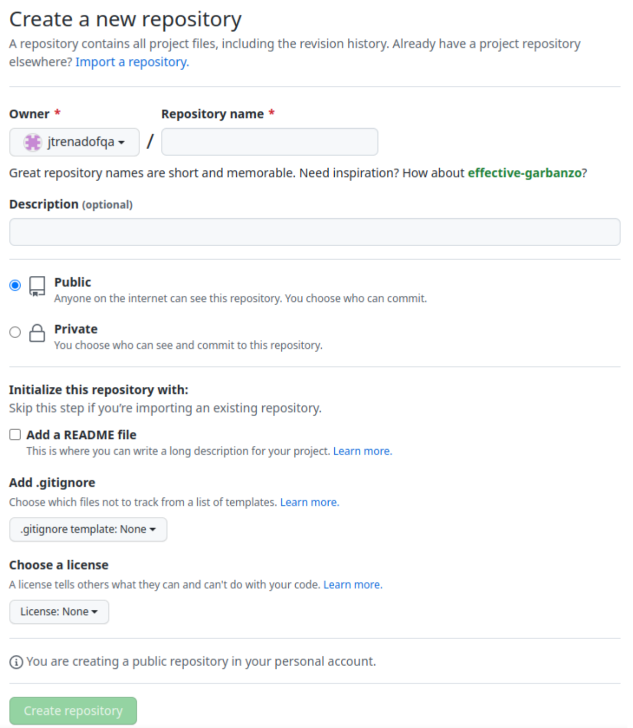
Paste you pub key here

Basics: create origin repo

- Create a remote repository called holamundo







Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner * jtrenadofqa / Repository name *

Great repository names are short and memorable. Need inspiration? How about effective-garbanzo?

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:
Skip this step if you're importing an existing repository.

☐ Add a README file
This is where you can write a long description for your project. Learn more.

Add .gitignore
Choose which files not to track from a list of templates. Learn more.
.gitignore template: None

Choose a license
A license tells others what they can and can't do with your code. Learn more.
License: None

You are creating a public repository in your personal account.

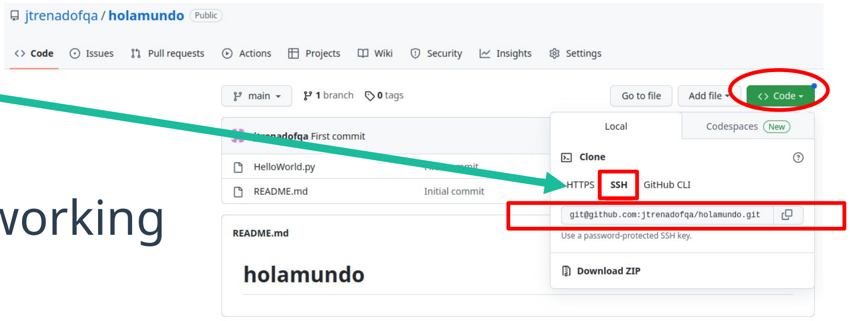
Create repository

Basics: clone origin and identity

- Clone locally your repository through ssh protocol: **git clone git@github.com:jtrenadofqa/holamundo.git**

  You'll need a personal access token for command line authentication through https

  If you can clone without password your keys are working



jtrenadofqa First commit

HelloWorld.py

README.md Initial commit

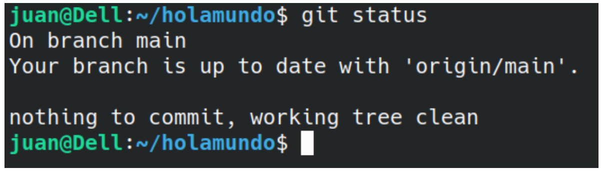
```
git@github.com:jtrenadofqa/holamundo.git
```

Use a password-protected SSH key.

- Configure identity and editor:
  - User: **git config --global user.name “Your name”**
  - Email: **git config --global user.email “Your email”**
  - Editor: **git config --global core.editor “Your editor”**

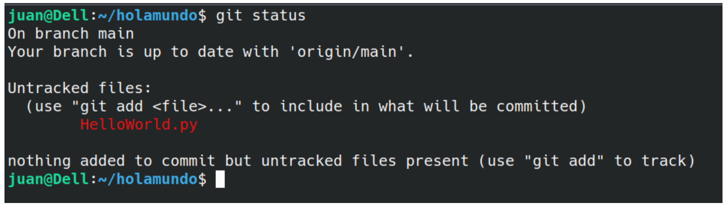
Basics: status and file states

- Go inside the directory and list all the files, included hidden files (ref. Page 418. Section *Git Internals* of *ProGit* book).
- There is a man page for any git command: man git "command"
- Check the status of your local repository: **git status**



```
juan@Dell:~/holamundo$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
juan@Dell:~/holamundo$ 
```

- Create an empty document called HelloWorld.py (linux/mac - **touch HelloWorld.py**)
- Check the status of your local repository: **git status**

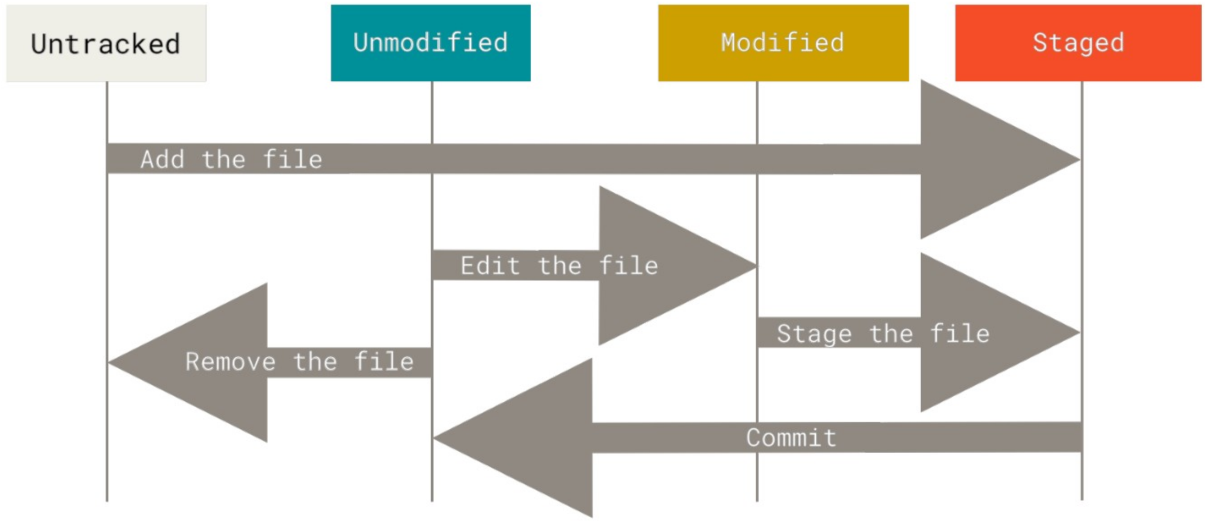


```
juan@Dell:~/holamundo$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

nothing added to commit but untracked files present (use "git add" to track)
juan@Dell:~/holamundo$ 
```

Basics: file states

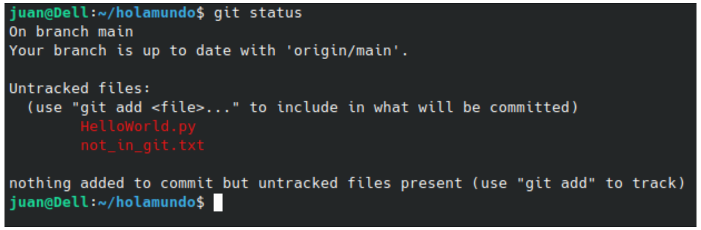
Files have three states **within** git database:

- Modified: file has changes but they are not committed.
- Staged: file is marked to be committed in the next commit.
- Committed: data is stored in your **local** database.



Basics: .gitignore

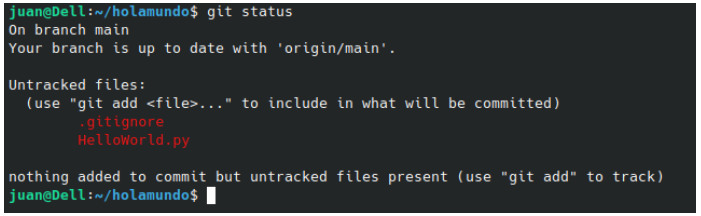
- Create file called not_in_git.txt: touch not_in_git.txt
- Get the status of your project: **git status**



```
juan@Dell:~/holamundo$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

nothing added to commit but untracked files present (use "git add" to track)
juan@Dell:~/holamundo$ 
```

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**

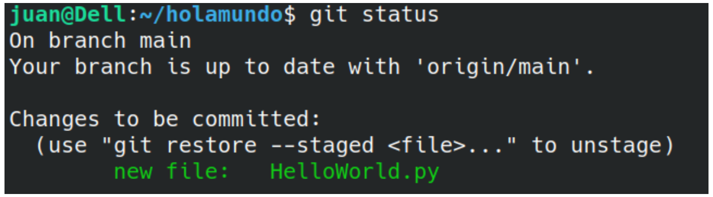


```
juan@Dell:~/holamundo$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

nothing added to commit but untracked files present (use "git add" to track)
juan@Dell:~/holamundo$ 
```

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

Basics: stage, commit and push

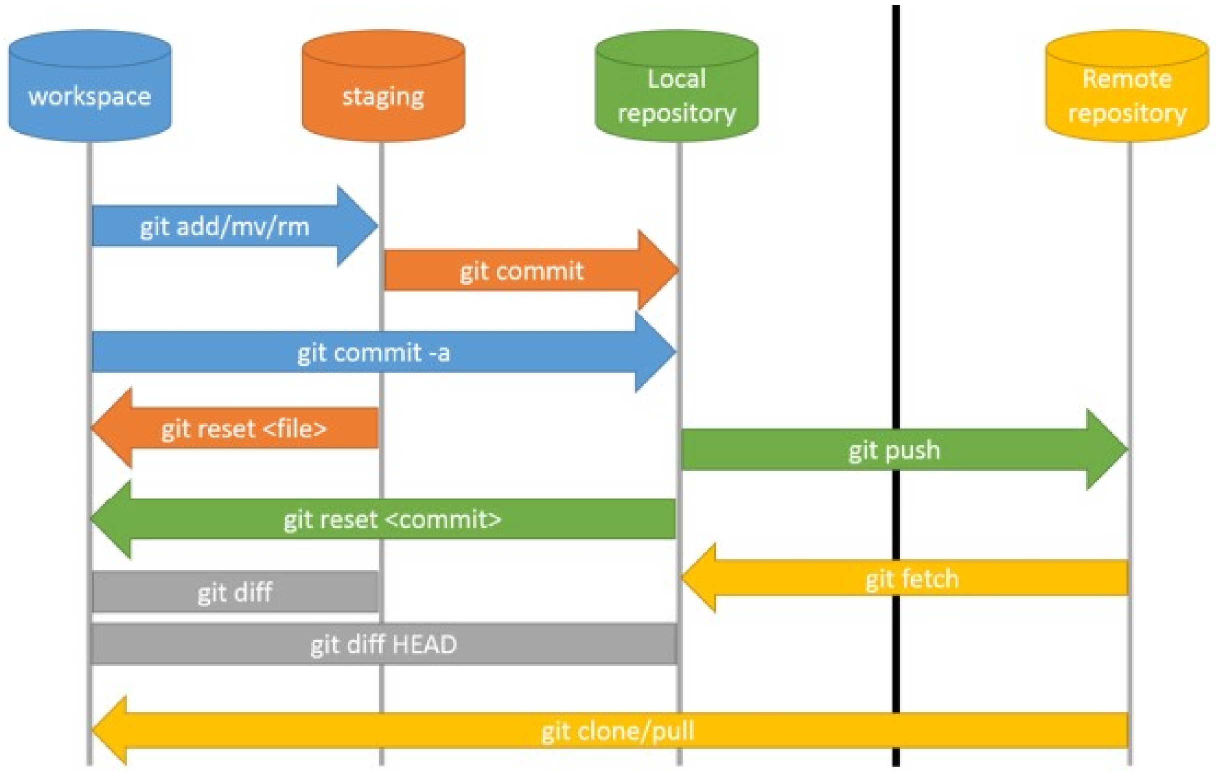
- Stage the file HelloWorld.py: **git add HelloWorld.py**
- Check the status of your local repository: **git status**



```
juan@Dell:~/holamundo$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   HelloWorld.py
```

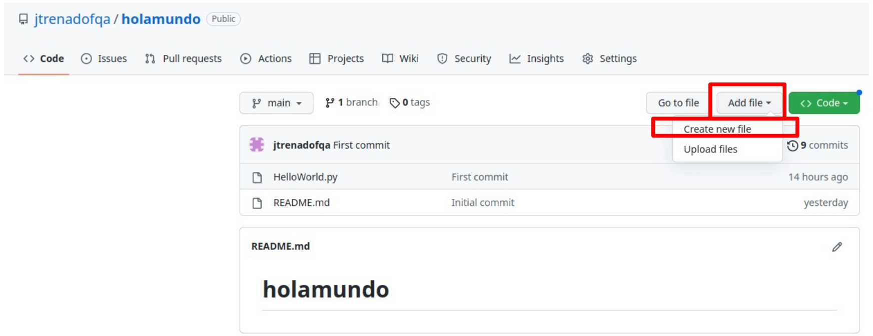
- I made a mistake!!!! I need to unstage!!: **git restore --staged HelloWorld.py**
- Check the status of your repository: **git status**
- Stage again HelloWorld.py: **git add HelloWorld.py**
- Time to commit: **git commit -m “”**
- Commit again: **git commit -m “First commit”**
- Check your remote repository
- Push commits to origin: **git push**

Basics: git workflow



Basics: create file in origin

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



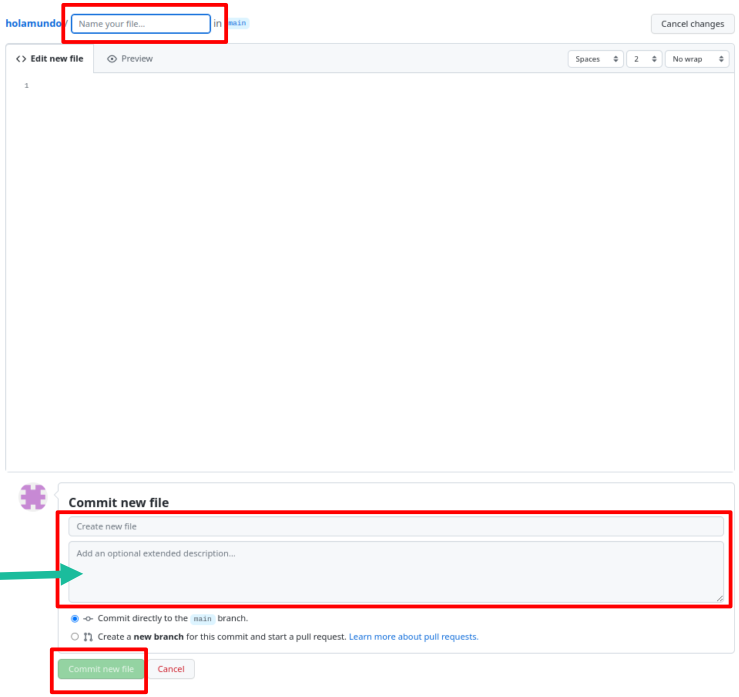
jtrenadofqa / holamundo Public

jtrenadofqa First commit — 9 commits

HelloWorld.py	First commit	14 hours ago
README.md	Initial commit	yesterday

README.md

holamundo



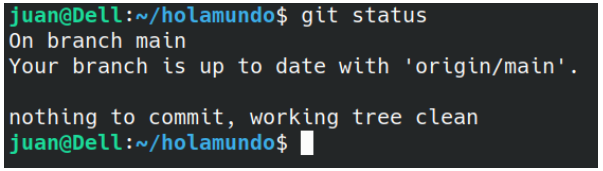
holamundo / Name your file... in main

Cancel changes
Edit new file
Preview
Spaces
2
No wrap
Commit new file
Create new file
Add an optional extended description...
Commit directly to the main branch.
Create a new branch for this commit and start a pull request. Learn more about pull requests.
Commit new file
Cancel

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

Basics: fetch

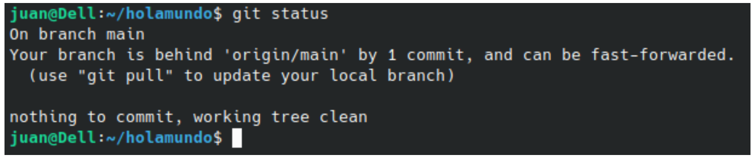
- Check the status of your repository: **git status**



```
juan@Dell:~/holamundo$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
juan@Dell:~/holamundo$ 
```

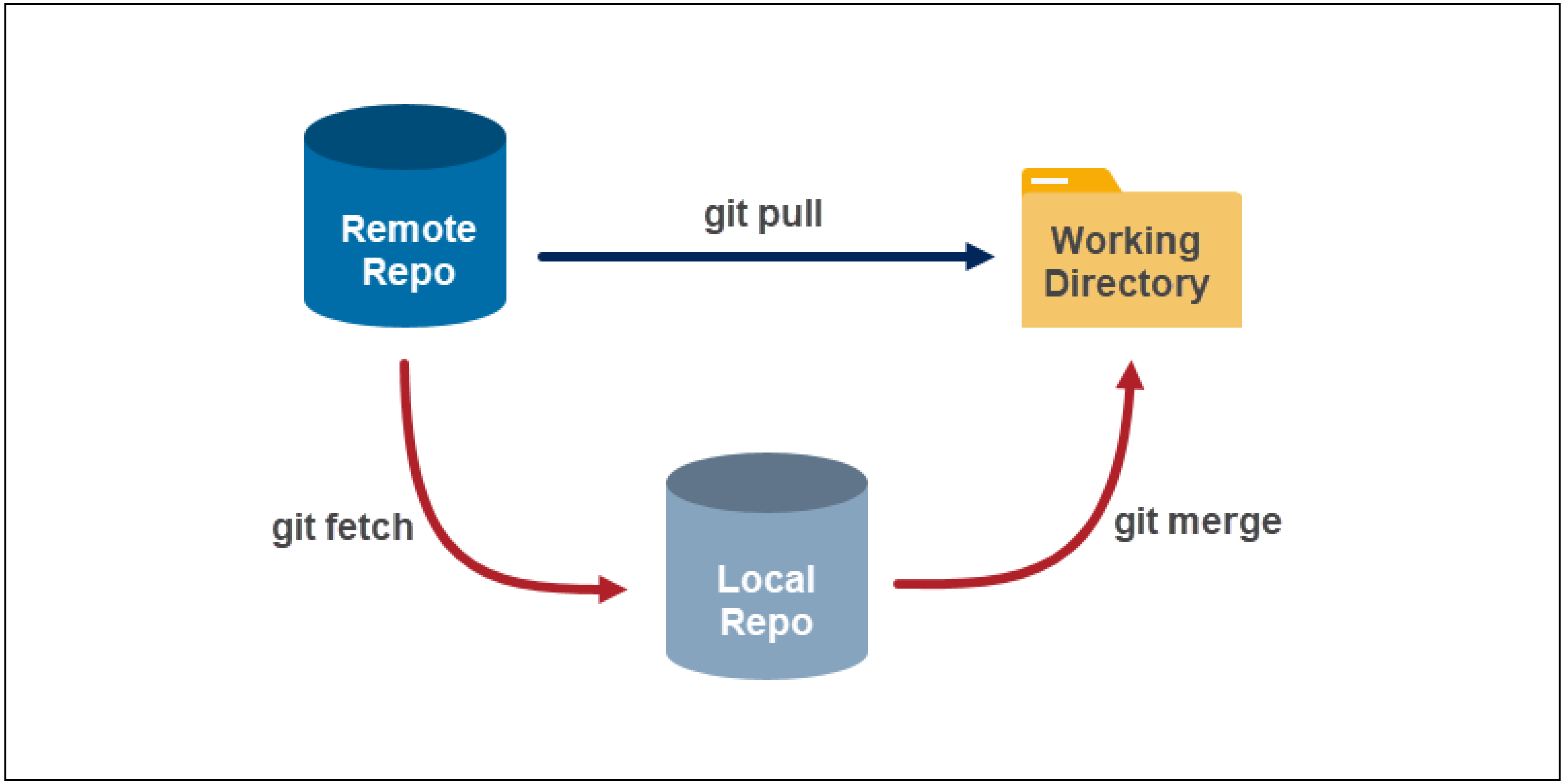
Changes in remote are not communicated in real time to local repos, to retrieve metadata for any change you have to fetch the repo.

- Fetch your repo: **git fetch**
- Check the status of your repository: **git status**



```
juan@Dell:~/holamundo$ git status
On branch main
Your branch is behind 'origin/main' by 1 commit, and can be fast-forwarded.
  (use "git pull" to update your local branch)

nothing to commit, working tree clean
juan@Dell:~/holamundo$ 
```

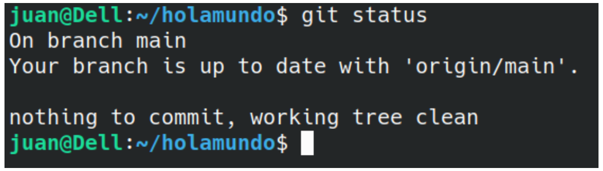
Basics: pull=fetch+merge (or rebase)



Basics: pull=fetch+merge (or rebase)

- Get changes from origin: **git pull**

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



```
juan@Dell:~/holamundo$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
juan@Dell:~/holamundo$ 
```

If you try to fetch now you'll see you don't get any message and your status is up to date with origin.

Basics: delete files

- Remove files from your project
  - 1st method:
    - Remove first from your working directory: **rm "filename"**
    - Stage the file: **git rm filename**
    - Commit the deleted file: **git commit -m "filename deleted"**
    - Push changes: **git push**

  - 2nd method, git removes and stages at the same time.
    - Remove and stage using git: **git rm filename**
    - Commit the deleted file: **git commit -m "filename deleted"**
    - Push changes: **git push**

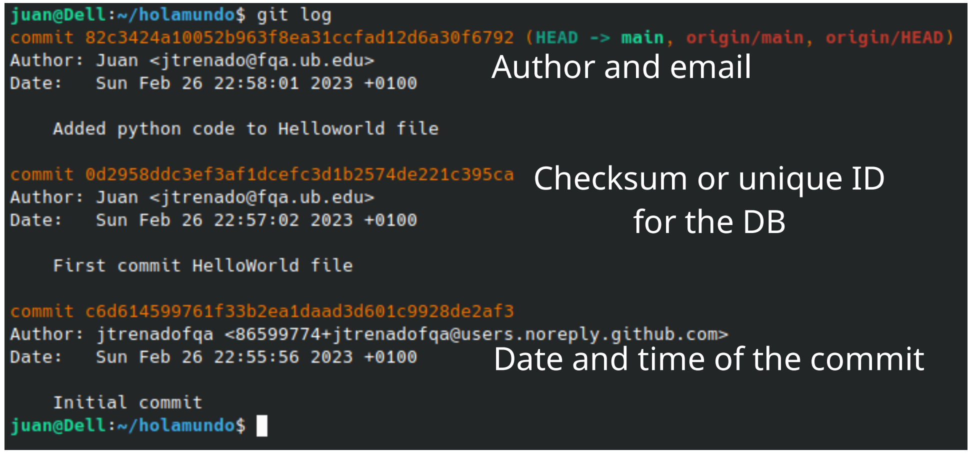
Basics: log

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

Last commit and current position (HEAD)

First commit

Commit message



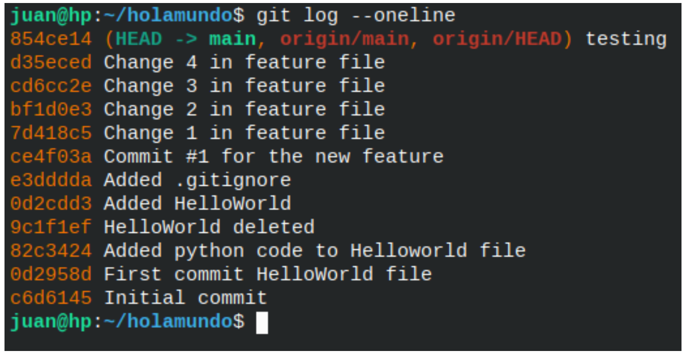
```
juan@Dell:~/holamundo$ git log
commit 82c3424a10052b963f8ea31ccfad12d6a30f6792 (HEAD -> main, origin/main, origin/HEAD)
Author: Juan <jtrenado@fqa.ub.edu>
Date:   Sun Feb 26 22:58:01 2023 +0100

    Added python code to Helloworld file

commit 0d2958ddc3ef3af1dcefc3d1b2574de221c395ca
Author: Juan <jtrenado@fqa.ub.edu>
Date:   Sun Feb 26 22:57:02 2023 +0100

    First commit HelloWorld file

commit c6d614599761f33b2ea1daad3d601c9928de2af3
Author: jtrenadofqa <86599774+jtrenadofqa@users.noreply.github.com>
Date:   Sun Feb 26 22:55:56 2023 +0100

    Initial commit
juan@Dell:~/holamundo$
```

Author and email

Checksum or unique ID for the DB

Date and time of the commit

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

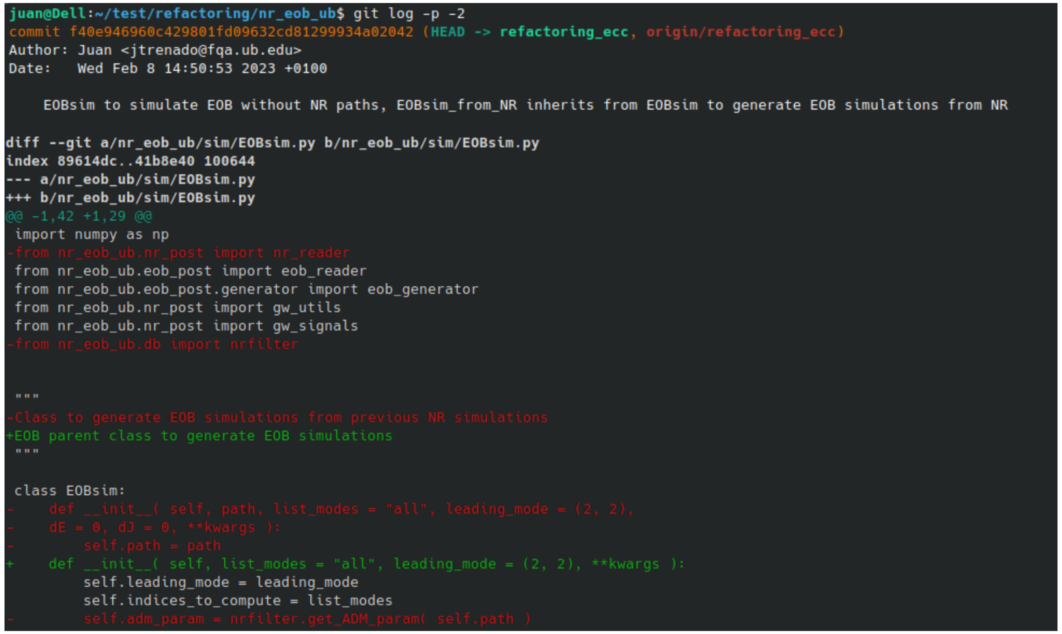


```
juan@hp:~/holamundo$ git log --oneline
854ce14 (HEAD -> main, origin/main, origin/HEAD) testing
d35eced Change 4 in feature file
cd6cc2e Change 3 in feature file
bf1d0e3 Change 2 in feature file
7d418c5 Change 1 in feature file
ce4f03a Commit #1 for the new feature
e3dddda Added .gitignore
0d2cdd3 Added HelloWorld
9c1f1ef HelloWorld deleted
82c3424 Added python code to Helloworld file
0d2958d First commit HelloWorld file
c6d6145 Initial commit
juan@hp:~/holamundo$
```

Basics: log + diff

- Variations of git log: **git log -p -"Number"**

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@fqa.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.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 ):
-        self.path = path
+    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"**