

# Introduction to Git



## As a PhD student, do I really need a version control system?

- Are you in a team of developers or developing a project you have to share? ✓
- Do you need to track changes in your project? ✓
- Are you working in a project that needs to be maintained in the medium term? ✓
- Are you working on an analysis that you'll never share and you'll never look at it again after you'll finish



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

# Outline

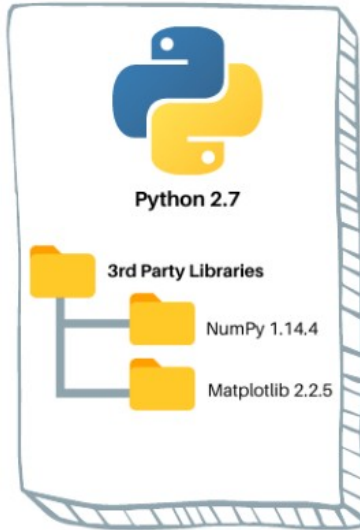


- ✗ Environment management with Conda/Mamba
- ✗ Installation and configuration of Git
- ✗ Introduction to GitHub repository platform: accounts, local and remote repositories
- ✗ SSH protocol and key generation, authentication

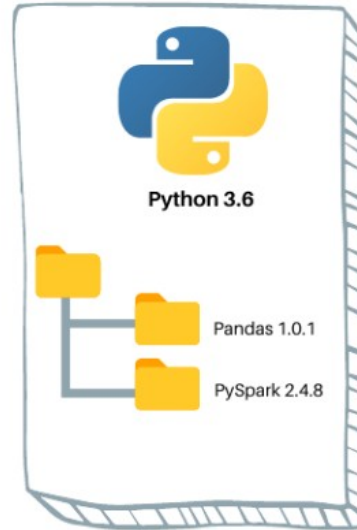
# Environment management



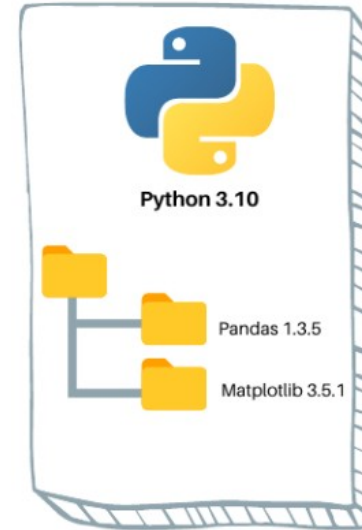
Virtual Environment 1



Virtual Environment 2



Virtual Environment 3



# Environment management



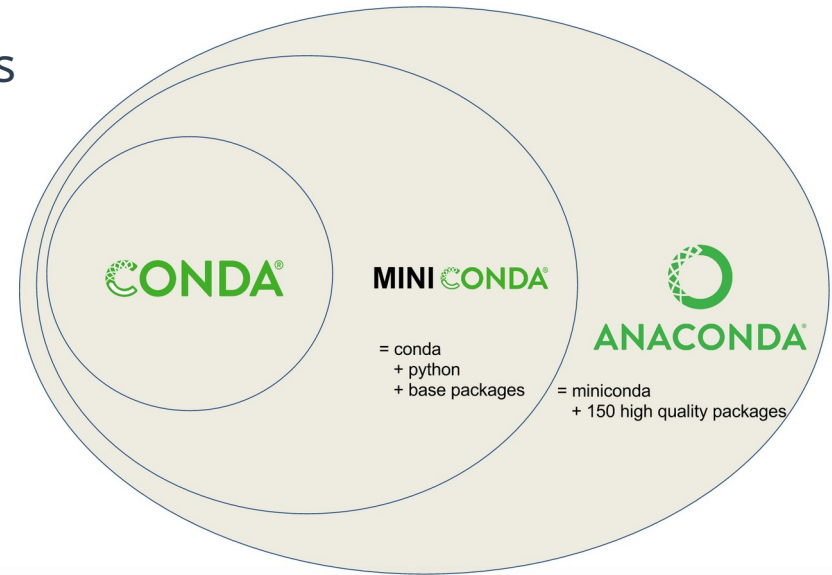
## Package and environment manager use cases

- **Software compatibility:** you'll **prevent obsolescence** due to future updates of your system libraries
- **Collaborative projects:** same environment for all team members will **avoid compatibility problems**
- **Developing incompatibilities:** **prevent conflicts** between modules/libraries developed by you and those of your system

# CONDA Family



- **Conda:** the package and environment manager
  - It will create and manage the environments
  - It will install packages and update them when you need it
- **Miniconda:** conda + python + few dependencies (~400MB)
- **Anaconda:** conda + lots and lots of packages (~3GB)



# MINICONDA



- Let's install MINICONDA

- `wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh` (Linux)
- `wget https://repo.anaconda.com/miniconda/Miniconda3-latest-MacOSX-x86_64.sh` (macOS)
- `chmod +x Miniconda3-latest-Linux-x86_64.sh` (Linux)
- `./Miniconda3-latest-Linux-x86_64.sh` (Linux)
- You'll have to accept the license
- You'll have to accept the location to be installed
- You can/can't accept base conda environment being activated
  - If you want to deactivate it: `conda config --set auto_activate_base false`
- Type on a term:
  - `conda version`
  - `conda -list`

You should have conda now

- `conda create -n "myenv" python=3.3.0`
- `conda activate myenv`
- `conda deactivate`

```
# >>> conda initialize >>>
# !! Contents within this block are managed by 'conda init' !!
__conda_setup="$(('/home/juan/miniforge3/bin/conda' 'shell.bash' 'hook' 2> /dev/null)"
if [ $? -eq 0 ]; then
    eval "$__conda_setup"
else
    if [ -f "/home/juan/miniforge3/etc/profile.d/conda.sh" ]; then
        . "/home/juan/miniforge3/etc/profile.d/conda.sh"
    else
        export PATH="/home/juan/miniforge3/bin:$PATH"
    fi
fi
unset __conda_setup
```

# CONDA problems... ??



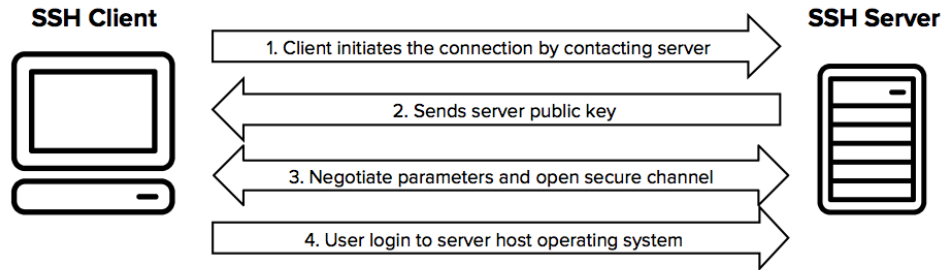
- CONDA had/have performance issues, which led to the development of Mamba as a faster alternative.
- Supposedly, after CONDA version 23.11 is expected to offer similar performance levels.



# SSH protocol



SSH stands for Secure Shell, and enables secure system administration and file transfers over insecure networks.





# SSH protocol



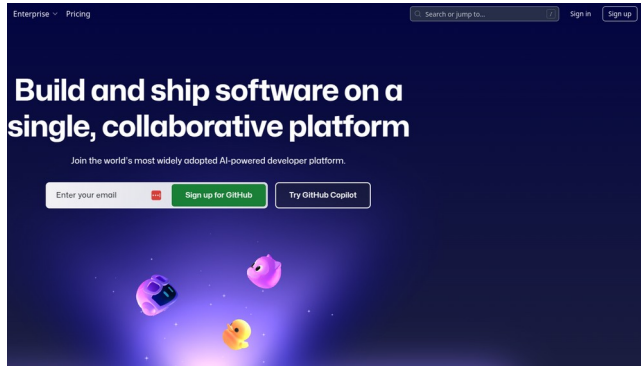
- To create ssh keys: `ssh-keygen -t rsa`

```
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//L2EUBqTJ9s0JNzBI juan@hp
The key's randomart image is:
+---[RSA 3072]-----+
|      .O+.++      |
|      E..+o.o     |
|      . = +.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 will have to copy your public key to your github account: `cat id_rsa.pub`

```
juan@hp: ~/.ssh$ cat id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQGDHcqSfDKf+8i8
cB9/Kp+dlGicI3AJScLw9Ze503q1Ip90lTYpJ+7jL+aJt13MR4
J3rvHVAATBdVTTbo5yf9N0j5u0MefCmj+dzxMH581VSehN8Q03S
Qgka35u06pZb0zFMkm7/PC81yGJNG8WM0QMU/nQboX8USiug15H
N69LwEax+b/L08yXzYfUrE6FvtA8gvQ4Nm2tATHLF/YuC3zrrUZ
iYhNdR7fycSQsZCDJDPjTl+Q+2b/PyDBsJ69/tHpBjuFt5FerDx
D9EZhh1qIW6XUXwReHE7e7fswF+RsToLPE9C8Lffjy8XyNj0K7X
5pfxwzVDHZHYcPg3ZsKoTJBHNa/7Z6kvzf/ZCTJo1MPcejhuaXt
0Giz1ohhjcrsMJPR9NRc2WsPpmeH9d05A3ogzbBqUXkJoCakHZZ
bPk2TqvFlevEjdCrziVSyZH32d4GXBmo/qNSwaxNLW8+ZFTkbY+
05z/Im7ZTmT20NaGB+RfraceeZHuwS6LxvnguCZ8= juan@hp
```

# Create a github account



## Sign up to GitHub

Email\*

▲ Email cannot be blank

Password\*

Password should be at least 15 characters OR at least 8 characters including a number and a lowercase letter.

Username\*

Username may only contain alphanumeric characters or single hyphens, and cannot begin or end with a hyphen.

Continue >

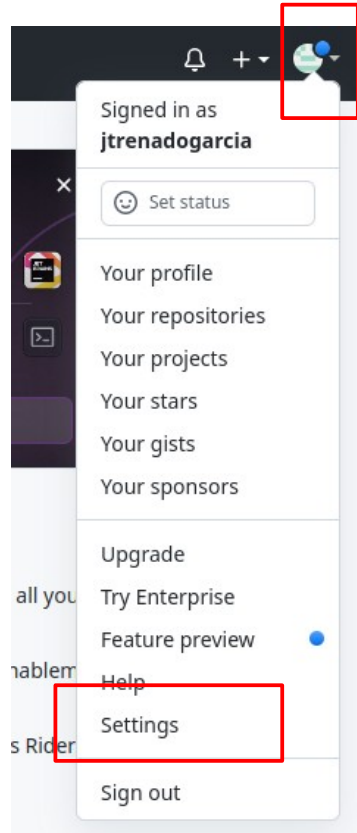
By creating an account, you agree to the [Terms of Service](#). For more information about GitHub's privacy practices, see the [GitHub Privacy Statement](#). We'll occasionally send you account-related emails.


You're almost done!

We sent a launch code to `jtregarcia@gmail.com`

→ Enter code

# Add your pub key to you github account



 **jtrenadogarcia**  
Your personal account

- Public profile
- Account
- Appearance
- Accessibility
- Notifications
- Access
  - Billing and plans
  - Emails
  - Password and authentication
  - Sessions
  - SSH and GPG keys**
  - Organizations
  - Moderation
- Code, planning, and automation
  - Repositories
  - Codespaces
  - Packages
  - Copilot
  - Pages
  - Saved replies
- Security
  - Code security and analysis

## 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'

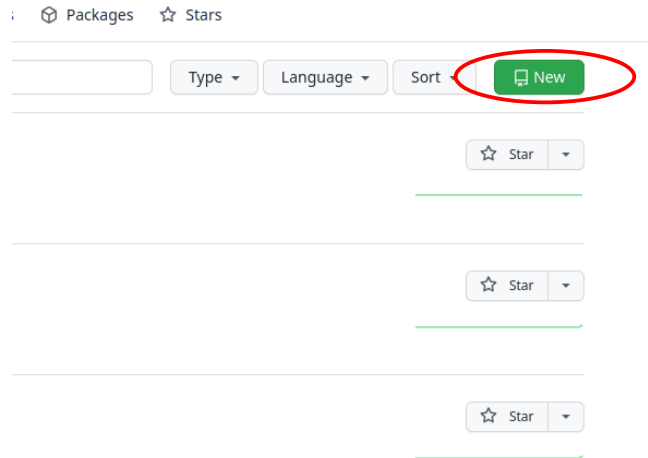
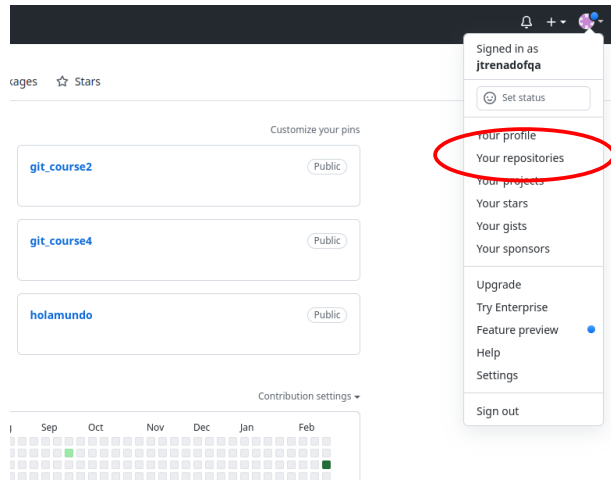
Paste you pub key here

Add SSH key

# 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 \*  /

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](#).

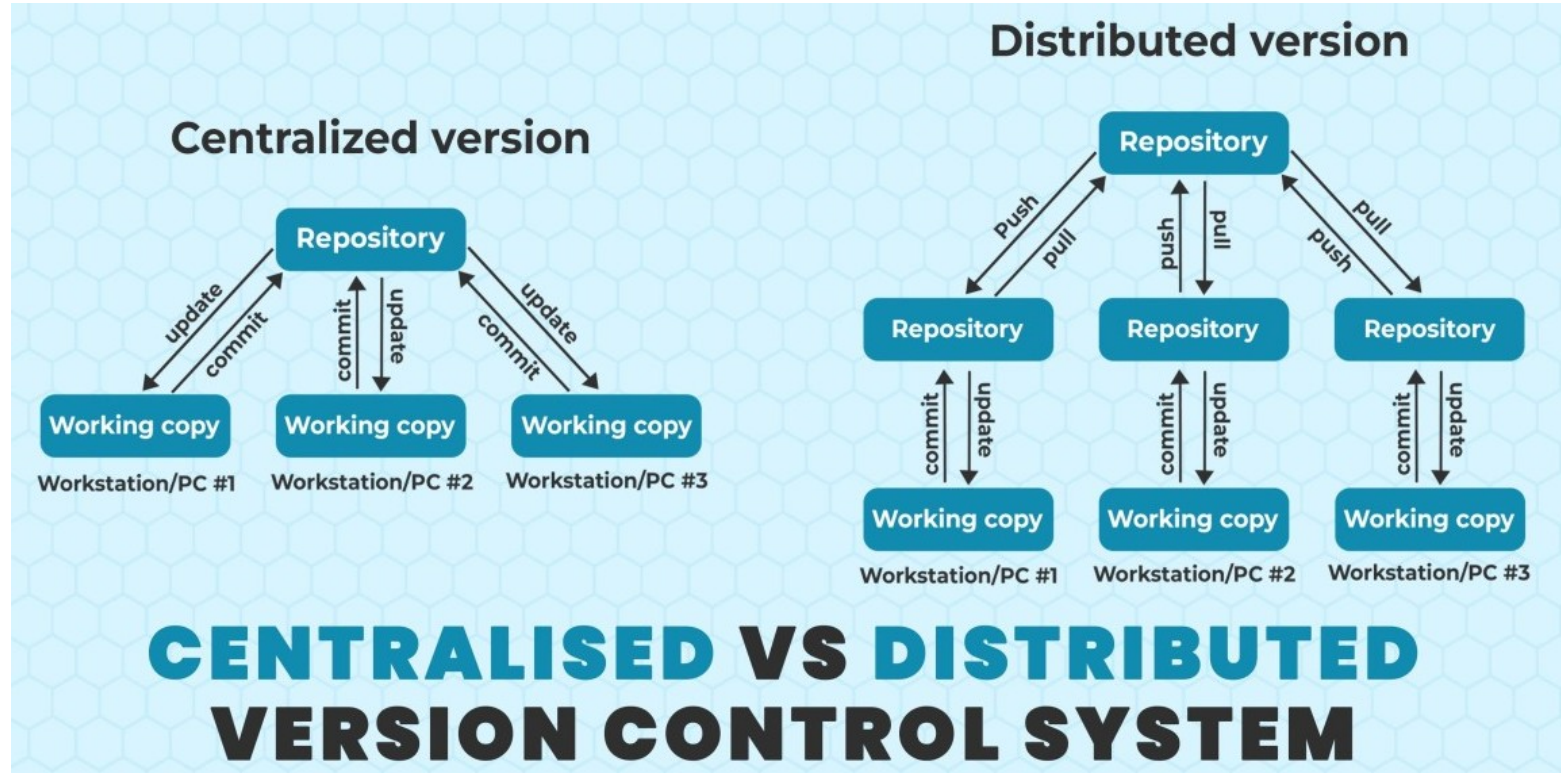
**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more](#).  
:gitignore template:

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more](#).  
License:

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

Create repository

# CVCS vs DVCS



# CVCS vs DVCS



## Distributed Version Control System (DVCS - GIT)

- Every developer has a copy of the entire repository locally, so you can work offline
- Working on branches is easy because every developer has a entire history of the code
- If the remote repo goes down or it crashes you can back it up from local
- Projects with long history or large binary files will need more space locally and they will be slower to download and push changes.
- Less merger conflicts, only when pushing/pulling changes.

## Centralized Version Control Systems (CVCS – Subversion)

- There is only one copy of the repository in a central server, so you need to be connected to the server to make changes.
- Working with branches is more complicated because it requires continuous communications with the server
- Suitable for projects with large binary files because they don't have to be upload/download continuously and they don't need a entire copy locally.
- More merger conflicts because we have to commit to remote continuously any change.

# GIT – Installation & Configuration



- For Ubuntu-like systems
  - `sudo apt-get install git`
- For other OS
  - <https://git-scm.com/book/en/v2/Getting-Started-Installing-Git>
- Configure your credentials 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"`**