# A very short tutorial on how to use Git and GitHub

By Jorge Agramont

This guide was elaborated to be use in a computer with Unix operative system (OS).

Git is a version control system, that helps us track the history of a project and also to collaborate with others.

## Check Git version and installation

Maybe you already have Git installed. Check the version of Git

```
git --version
```

Check where is Git installed

```
which git
```

It should be installed in the global environment of your PC, if that is the case then you will get something like this: `/usr/local/bin/git`. If git is installed in a conda environment then you can update it using `conda update git`, but don't use that for the global installation.

### Install Git

If it is the case that Git is not installed then install it using Homebrew

Once you have isntalled Homebrew install Git:

```
brew install git
```

Another option to install git:

```
sudo apt update
```

```
sudo apt install git -y
```

### Update Git

If you have installed git using Brew then first update Homebrew

```
brew update
```

Then update Git

```
brew upgrade git
```

Verify the Git version

```
git --version
```

## Configuration

Check your current configurations:

```
git config --list
```

Alternatively, to make sure that you are checking the global configurations of Git you can run:

```
git config --global --list
```

And if you want to check the configurations of a specific git repository you can use: `git config --local --list`

Create a Global Git Configuration File The use of `""` is important because there is a space between your name and last name.

```
git config --global user.name "Your Name"
git config --global user.email "your-email@example.com"
```

Now you can verify your configuration running again: `git config --global --list`.

### SSH key

This is important to connect securely your Git to GitHub.

First of all you will need to create a GitHub account.

Once you have created a GitHub account or if you already have one you can check if you have an existing SSH key configuration:

```
ls -al ~/.ssh
```

If you see files like `id_ed25519.pub` or `id_rsa.pub`, these are your public SSH keys.

Check if the SSH Key is Linked to GitHub

```
ssh -T git@github.com
```

If your SSH key is set up correctly and added to GitHub, you'll see a message like: `Hi <username>! You've successfully authenticated, but GitHub does not provide shell access.`

If you don't have an SSH key, create one:

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

Use your GitHub email address. Save the key in the default location (~/.ssh/id_ed25519) just pressing Enter. Optionally, set a passphrase for added security.

Now add the SSK key to the SSH agent

```
eval "$(ssh-agent -s)"
ssh-add ~/.ssh/id_ed25519
```

### Add the SSH key to GitHub

Copy the SSH public key

```
# If you are using Mac:
pbcopy < ~/.ssh/id_ed25519.pub
```

```
# If you are using Ubuntu:
cat ~/.ssh/id_ed25519.pub | clip.exe
```

Log in to your GitHub account.

Go to Settings > SSH and GPG Keys > New SSH Key.

Paste your SSH key into the key field and give it a title (e.g., "My MacBook SSH Key").

Save the key.

Now test the connection once again

```
ssh -T git@github.com
```

## Create a new Git repository

In your PC navigate to your Project foulder or create a new folder

```
mkdir my_new_repo
cd my_new_repo
```

## Initialize a new repository

Once inside the foulder write

```
git init
```

Create a new file, for example a .txt file called hello.txt

```
echo "Hello Git" > hello.txt
```

Now add the file to Git, this will save only one single file.

```
git add hello.txt
```

You can also add all the changes at the same time to be saved

```
git add .
```

Commit the file. To commit you need to write a message that indicated the type of change you are commiting (confirming). Write something short but explenatory of what you did.

```
git commit -m "here it goes your message"
```

To verify the changes you can type:

```
git log
```

This will show you all prevous saved changes

## Connect your repository to a remote repository

Go to your GitHub account and create a new repository.

Now you can connect your local repository to a remote repository like GitHub. For doing this you will need to use your SSH URL (You can get this in your GitHub account, go to your repository and look for a green button that says code, click there and then click in SSH and copy the URL):

```
git remote add origin git@github.com:yourusername/your-repository.git
```

Go to your GitHub account and create a new repository. Get the URL of the new repository and then run:

```
git remote add origin https://github.com/yourusername/your-repository.git
```

Push your local commit to the remote repository:

```
git push -u origin master
```

This will link your local and remote repository and push your changes to the remote repository in GitHub. The flag `-u` stands for upstream. After running the command once, you can use `git push` and `git pull` without specifying the remote or branch name every time.

If you made changes in the remote you can pull those changes to your local repository.

```
git pull origin master
```

If you are not sure which branch you are tracking you can use the following to check:

```
git branch -vv
```

## Additional funtions

Check the status of your repository using

```
git status
```

See the current branch:

```
git branch
```

## Create a .gitignore File

Open a terminal and navigate to your repository

```
cd /path/to/your/repository
```

Create the gitignore File

```
touch .gitignore
```

Open the .gitignore file in your text editor and specify the files or directories you want Git to ignore. Each entry should be on a new line.

If you want to ignore a specific file then write the name of the file

```
secret.txt
```

If you want to ignore all log files:

```
*.log
```

You can also ignore a whole directory

```
/my_dir/
```

You can also ingnore everithing in a directory except one File

```
/data/*
!/data/important.csv
```

Finally save the .gitignore file

If you've already added a file to the repository before creating the .gitignore, Git is already tracking it. To make Git ignore it, you'll need to remove the file from tracking:

```
git rm --cached filename
```

Then commit

```
git commit -m "Update .gitignore"
```

You can check which files are being ignored using:

```
git status --ignored
```

## Clone a GitHub repository

Lets say we want to clone the following repository: https://github.com/kipkurui/Python4Bioinformatics.git

In your PC (in the Ubuntu terminal) navigate to the directory where you want to clone the repository:

```
cd /path/to/your/project/directory
```

Follow the link to the github repository and copy the URL of the repository after selecting HTTPS.

Then run the following:

```
git clone git@github.com:username/repository.git
```

That in our case will be:

```
git clone https://github.com/kipkurui/Python4Bioinformatics.git
```

Once cloned the repository you can check it navigating to the directory and then check the files:

```
cd Python4Bioinformatics
```

```
ls
```

You should be able to see the files in your terminal and also navigating in your computer.

If you know that are changes in the repository and you want to get them in your local copy of the repository, after navigating to the repository directory use:

```
git pull
```

Finally you can start a project using the current directory (the local copy of the repository)

```
code .
```