

# Git Essentials

30/06 y 07/07

Ing. Paola B. Torres y Esp. Ing. Daniela Armijo



### Esp. Ing. Daniela Armijo

- Ing. Química
- Especialista en Docencia Univ. de UNCuyo
- Docente en: Análisis Matemático I, Álgebra y Geometría Analítica, Física I, y Análisis Matemático II
- Estudiante en la Tecnicatura en Programación de FRSR - UTN.

### Ing. Paola B. Torres

- Ing. Química
- Estudiante de Doctorado en FRSR - UTN
- Grupo Vinculado Bionanotecnología y sistemas complejos FRSR UTN-INFAP
- Ayudante en Sistemas Dinámicos I



**UTN FRSR**  
Facultad Regional San Rafael  
Universidad Tecnológica Nacional



**CHARLES  
UNIVERSITY**

## Esp. Ing. Daniela Armijo

- Ing. Química
- Especialista en Docencia Univ. de UNCuyo
- Docente en: Análisis Matemático I, Álgebra y Geometría Analítica, Física I, y Análisis Matemático II
- Estudiante en la Tecnicatura en Programación de FRSR - UTN.

## Ing. Paola B. Torres

- Ing. Química
- Estudiante de Doctorado en FRSR - UTN
- Grupo Vinculado Bionanotecnología y sistemas complejos FRSR UTN-INFAP
- Ayudante en Sistemas Dinámicos I



**UTN FRSR**  
Facultad Regional San Rafael  
Universidad Tecnológica Nacional



**CHARLES  
UNIVERSITY**

## Esp. Ing. Daniela Armijo

- Ing. Química
- Especialista en Docencia Univ. de UNCuyo
- Docente en: Análisis Matemático I, Álgebra y Geometría Analítica, Física I, y Análisis Matemático II
- Estudiante en la Tecnicatura en Programación de FRSR - UTN.

## Ing. Paola B. Torres

- Ing. Química
- Estudiante de Doctorado en FRSR - UTN
- Grupo Vinculado Bionanotecnología y sistemas complejos FRSR UTN-INFAP
- Ayudante en Sistemas Dinámicos I



CHARLES  
UNIVERSITY

# 1. Introducción

- ¿Qué es Git?
- ¿Cómo funciona Git?
- ¿Cómo se relaciona Git con GitHub u otros servicios?

# 2. Comandos básicos

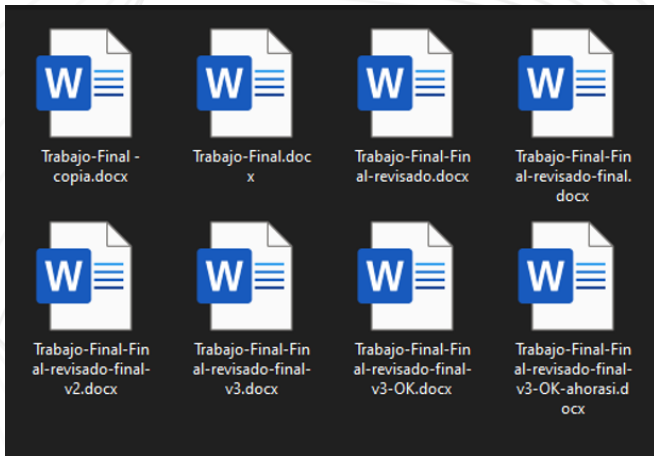
- Comandos útiles de terminal (bash)
- Configuración básica de Git
- Creación de un repositorio Git
- Confirmar cambios en el repositorio
- Gestión de archivos
- Historial de cambios y estado del repositorio
- Hash: identificador de commits

# 3. Colaboración

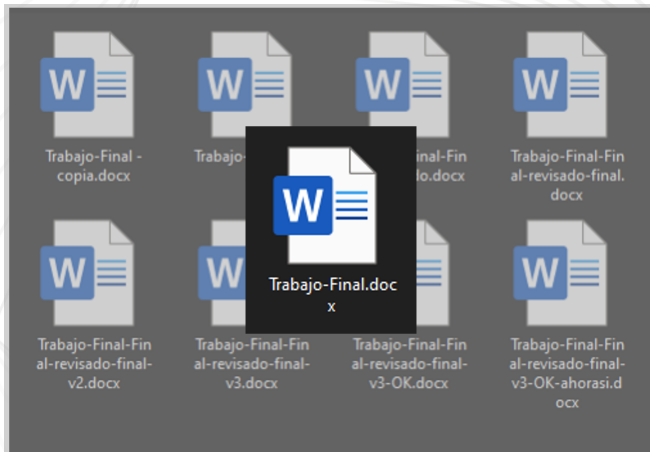
- Ramas
- Sincronización
- Resolución de conflictos
- GitHub funcionalidades
- Buenas practicas

# 4. Referencias

# ¿Qué es Git?



# ¿Qué es Git?



# ¿Qué es Git?



- Es una herramienta de control de versiones distribuido
- Permite trabajar en un proyecto sin necesidad de estar conectados a un servidor central.
- Un sistema de control de versiones es una herramienta que se utiliza para mantener un registro de los cambios que se hacen en un proyecto
- Documentación: [Web oficial git](#)



# ¿Qué es Git?



- Es una herramienta de control de versiones distribuido
- Permite trabajar en un proyecto sin necesidad de estar conectados a un servidor central.
- Un sistema de control de versiones es una herramienta que se utiliza para mantener un registro de los cambios que se hacen en un proyecto
- Documentación: [Web oficial git](#)

# ¿Qué es Git?



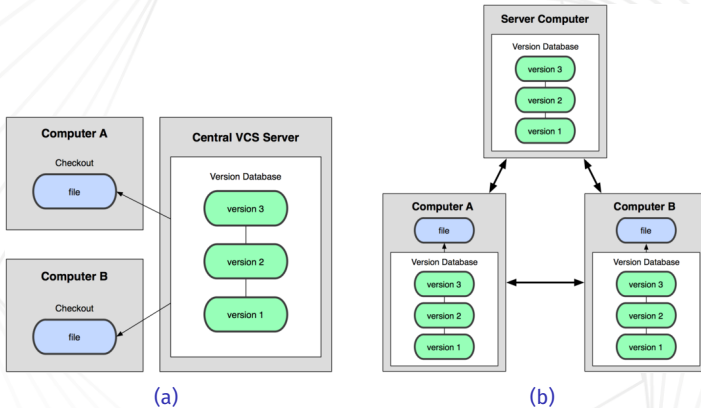
- Es una herramienta de control de versiones distribuido
- Permite trabajar en un proyecto sin necesidad de estar conectados a un servidor central.
- Un sistema de control de versiones es una herramienta que se utiliza para mantener un registro de los cambios que se hacen en un proyecto
- Documentación: [Web oficial git](#)

# ¿Qué es Git?

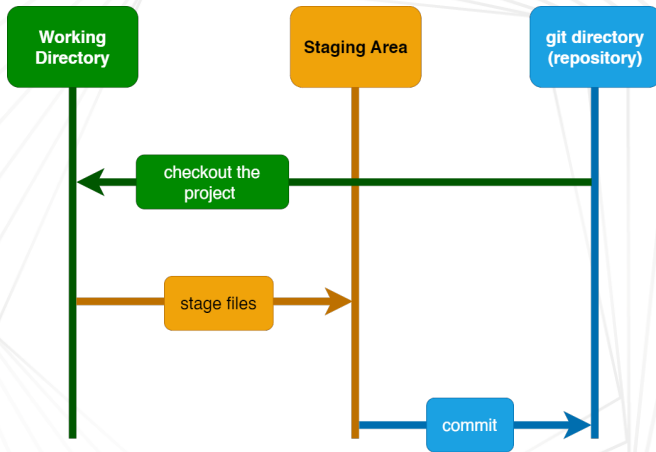


- Es una herramienta de control de versiones distribuido
- Permite trabajar en un proyecto sin necesidad de estar conectados a un servidor central.
- Un sistema de control de versiones es una herramienta que se utiliza para mantener un registro de los cambios que se hacen en un proyecto
- Documentación: [Web oficial git](https://git-scm.com/)

# Centralizado vs Distribuido



# ¿Cómo funciona Git?



# Plataformas



# ¿Cómo se relaciona Git con GitHub?

- Es una plataforma de alojamiento de código en la nube que utiliza el **sistema de control de versiones Git**.
- Colaboración en proyectos privados y abiertos.
- Herramientas adicionales: seguimiento de problemas, integración continua, revisión de código y colaboración en proyectos de código abierto.



# ¿Cómo se relaciona Git con GitHub?

- Es una plataforma de alojamiento de código en la nube que utiliza el **sistema de control de versiones Git**.
- Colaboración en proyectos privados y abiertos.
- Herramientas adicionales: seguimiento de problemas, integración continua, revisión de código y colaboración en proyectos de código abierto.





# ¿Cómo se relaciona Git con GitHub?

- Es una plataforma de alojamiento de código en la nube que utiliza el **sistema de control de versiones Git**.
- Colaboración en proyectos privados y abiertos.
- Herramientas adicionales: seguimiento de problemas, integración continua, revisión de código y colaboración en proyectos de código abierto.



# 1. Introducción

■ ¿Qué es Git?   ■ ¿Cómo funciona Git?   ■ ¿Cómo se relaciona Git con GitHub u otros servicios?

# 2. Comandos básicos

■ Comandos útiles de terminal (bash)   ■ Configuración básica de Git  
■ Creación de un repositorio Git   ■ Confirmar cambios en el repositorio  
■ Gestión de archivos   ■ Historial de cambios y estado del repositorio  
■ Hash: identificador de commits

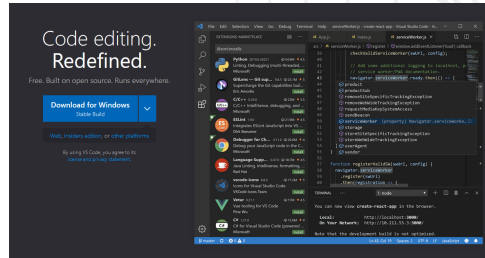
# 3. Colaboración

■ Ramas   ■ Sincronización   ■ Resolución de conflictos   ■ GitHub funcionalidades  
■ Buenas practicas

# 4. Referencias

# Editores de Código

- VS Code
- Notepad++
- VIM



# Comandos utiles de terminal

```
$ ls
$ cd <directorio>
$ cd ..
$ pwd
$ clear
$ mkdir <nombre_directorio>
$ touch <nombre_archivo>
$ rm <nombre_archivo>
$ mv <nombre_archivo> <directorio>
$ cp <nombre_archivo> .
$ cp <nombre_archivo> <directorio>
```

# Configuración básica de Git

```
git config  
git config --global user.name "nombre"  
git config --global user.email "email"
```

Ref: Documentación [git config](#)

# Creación de un repositorio desde GitHub

## Crear un repositorio en GitHub

Vamos a [GitHub](#)

## Clonar un repositorio desde GitHub: HTTPS

```
git clone https://github.com/user/repositorio.git .
```

## Clonar un repositorio desde GitHub: SSH

```
git clone git@github.com:/user/repositorio.git .
```

Ref: Documentación para [autenticación ssh](#)

# Creación de un repositorio Git

```
mkdir <nombre_carpeta>  
git init
```

## Conectar repositorio a la nube de GitHub

```
git remote add origin git@github.com:user/repositorio.git  
git branch -M main  
git push -u origin main
```

# Confirmar cambios en el repositorio

## Primero: Agregar archivos

```
$ git add [archivo]
```

## Segundo: git commit

```
$ git commit -m 'insertar mensaje simple y preciso'
```

## Tercero: git push

```
$ git push
```



# Confirmar cambios en el repositorio

## Primero: Agregar archivos

```
$ git add [archivo]
```

## Segundo: git commit

```
$ git commit -m 'insertar mensaje simple y preciso'
```

## Tercero: git push

```
$ git push
```

# Confirmar cambios en el repositorio

## Primero: Agregar archivos

```
$ git add [archivo]
```

## Segundo: git commit

```
$ git commit -m 'insertar mensaje simple y preciso'
```

## Tercero: git push

```
$ git push
```

# Gestión de archivos

## Agregar archivos

```
$ git add <nombre_archivo>
```

## Mover archivos

```
$ git mv <nombre_archivo>
```

## Eliminar archivos

```
$ git rm <nombre_archivo>
```

# Gestión de archivos

## Agregar archivos

```
$ git add <nombre_archivo>
```

## Mover archivos

```
$ git mv <nombre_archivo>
```

## Eliminar archivos

```
$ git rm <nombre_archivo>
```

# Gestión de archivos

## Agregar archivos

```
$ git add <nombre_archivo>
```

## Mover archivos

```
$ git mv <nombre_archivo>
```

## Eliminar archivos

```
$ git rm <nombre_archivo>
```

# Historial de cambios y estado del repositorio

## Estado de los archivos

```
$ git status
```

## Historial

```
$ git log
```

```
$ git log --pretty=oneline
```

```
$ git log --graph
```

# Historial de cambios y estado del repositorio

## Estado de los archivos

```
$ git status
```

## Historial

```
$ git log  
$ git log -pretty=oneline  
$ git log -graph
```

# Identificar commits

```
* commit ce12c98c14fc5effa0005992df8800c89824e4c3
| Author: Paola <paortorres89@gmail.com>
| Date: Tue May 16 11:23:27 2023 -0300
|
|     fixed bug in write_output_vtf
|
* commit 5c7d3a347fdd1b1122a6181a4b9158e2042a1e1f
| Author: blancoapa <blancoapa@natur.cuni.cz>
| Date: Mon May 15 21:21:24 2023 +0200
|
|     Add missing file
```

## ¿Qué es un Hash?

Una función criptográfica hash es un algoritmo matemático que transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con una longitud fija (40 caracteres)



# Identificar commits

```
* commit ce12c98c14fc5effa0005992df8800c89824e4c3
| Author: Paola <paortorres89@gmail.com>
| Date:   Tue May 16 11:23:27 2023 -0300
|
|     fixed bug in write_output_vtf
|
* commit 5c7d3a347fdd1b1122a6181a4b9158e2042a1e1f
| Author: blancoapa <blancoapa@natur.cuni.cz>
| Date:   Mon May 15 21:21:24 2023 +0200
|
|     Add missing file
```

## ¿Qué es un Hash?

Una función criptográfica hash es un algoritmo matemático que transforma cualquier bloque arbitrario de datos en una nueva serie de caracteres con una longitud fija (40 caracteres)

```
* commit ce12c98c14fc5effa0005992df8800c89824e4c3
Author: Paola <paotorres89@gmail.com>
Date: Tue May 16 11:23:27 2023 -0300

    fixed bug in write_output_vtf

* commit 5c7d3a347fdd1b1122a6181a4b9158e2042a1e1f
Author: blancoapa <blancoapa@natur.cuni.cz>
Date: Mon May 15 21:21:24 2023 +0200

    Add missing file
```

### ¿Para que nos sirve el Hash?

```
$ git diff <hash_a> <hash_b>
$ git diff --name-only <hash_a> <hash_b>
$ git checkout <hash_commit>
$ git reset -- hard <hash>
```

```
* commit ce12c98c14fc5effa0005992df8800c89824e4c3
Author: Paola <paotorres89@gmail.com>
Date: Tue May 16 11:23:27 2023 -0300

    fixed bug in write_output_vtf

* commit 5c7d3a347fdd1b1122a6181a4b9158e2042a1e1f
Author: blancoapa <blancoapa@natur.cuni.cz>
Date: Mon May 15 21:21:24 2023 +0200

    Add missing file
```

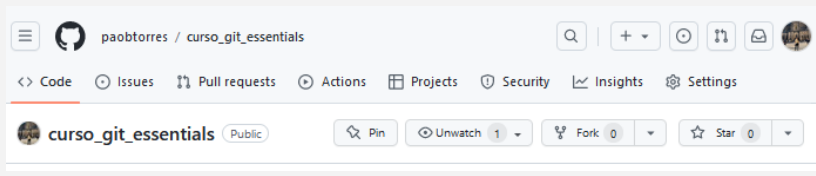
### ¿Para que nos sirve el Hash?

```
$ git diff <hash_a> <hash_b>
$ git diff --name-only <hash_a> <hash_b>
$ git checkout <hash_commit>
$ git reset -- hard <hash>
```

# Bifurcación

## Fork (bifurcación)

Permite copiar un repositorio en nuestro GitHub para poder realizar cambios en un repositorio publico para el cual no tenemos permisos.



# Pull request

The screenshot shows a GitHub repository named 'hello-git' which is public and forked from 'paolabeatriz/hello-git'. It has 1 branch (main) and 0 tags. The repository is currently 1 commit ahead and 1 commit behind the upstream main branch. The commit history shows three commits: 'Initial commit' for README.md, 'script to find the max value' for new-script.py, and 'Rename test-script-py to test-script.py' for test-script.py.

**hello-git** Public  
forked from [paolabeatriz/hello-git](#)

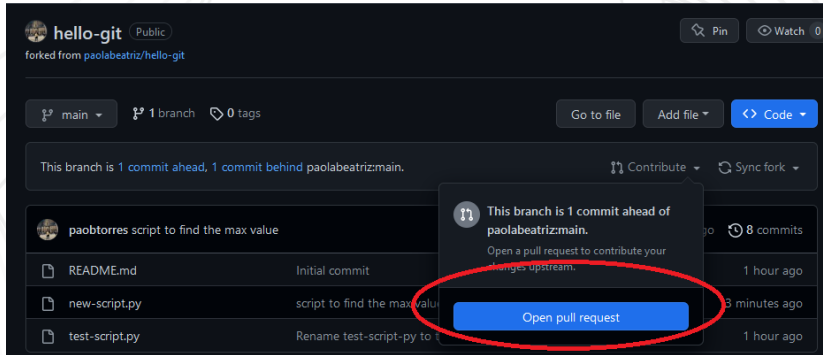
[main](#) [1 branch](#) [0 tags](#) [Go to file](#) [Add file](#) [Code](#)

This branch is [1 commit ahead](#), [1 commit behind](#) [paolabeatriz:main](#). [Contribute](#) [Sync fork](#)

**paobtorres** script to find the max value 7df97c3 now [8 commits](#)

<a href="#">README.md</a>	Initial commit	1 hour ago
<a href="#">new-script.py</a>	script to find the max value	now
<a href="#">test-script.py</a>	Rename test-script-py to test-script.py	1 hour ago


# Pull request



# Pull request

## Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

 base repository: paolabeatriz/hello-git


base: main

...

head repository: paobtorres/hello-git

compare: main

✓ Able to merge. These branches can be merged together.




Este script podría ser de utilidad

Write

Preview

H B I ≡ <> 🔗 ≡ ≡ ≡ @ 📎 ↶ 🗑

Leave a comment


Attach files by dragging & dropping, selecting or pasting them. 

☒ Allow edits by maintainers ⓘ 

Create pull request

ⓘ Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).

🔗 1 commit

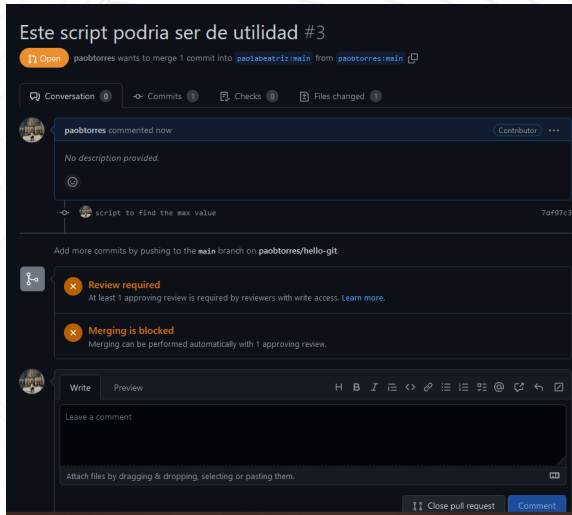
 1 file changed

Ing. Paola B. Torres y Esp. Daniela Armijo

Comandos básicos

39/71

# Pull request





## Práctica

Añadir su nombre de usuario de GitHub al final del archivo `welcome.md` que se encuentra en el repositorio original del curso, mediante un pull request.

### Para esto deberan:

Primero realizar un fork del repositorio del curso: [curso \\_git\\_essentials](#)

```
$ git clone (Clonar el repositorio al que hizo Fork)
```

Editar `welcome.md` y agregar su usuario de github

```
$ git add welcome.md
```

```
$ git commit "escriba un mensaje"
```

```
$ git push
```

Por ultimo un Pull request solicitando añadir los cambios al repositorio original del curso.

# 1. Introducción

■ ¿Qué es Git?   ■ ¿Cómo funciona Git?   ■ ¿Cómo se relaciona Git con GitHub u otros servicios?

# 2. Comandos básicos

■ Comandos útiles de terminal (bash)   ■ Configuración básica de Git  
■ Creación de un repositorio Git   ■ Confirmar cambios en el repositorio  
■ Gestión de archivos   ■ Historial de cambios y estado del repositorio  
■ Hash: identificador de commits

# 3. Colaboración

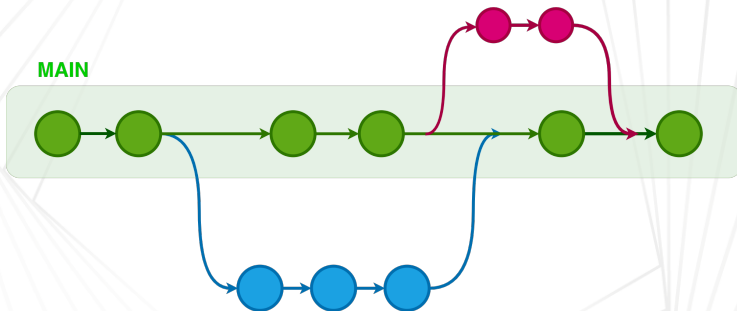
■ Ramas   ■ Sincronización   ■ Resolución de conflictos   ■ GitHub funcionalidades  
■ Buenas practicas

# 4. Referencias

# Ramas con Git branch

¿Para que sirven las ramas?

Permite trabajar de manera paralela sin realizar cambios en el código principal



Ejemplo en gitlab

## Gestión de ramas

```
$ git branch <nombre_rama> (Crear)
$ git branch <nombre_rama> -d (Eliminar)
$ git switch <nombre_rama> (Cambiar a una rama)
$ git checkout -b <nombre_rama> (Cambiar a una rama)
$ git merge (Integrar una rama)
```

### git switch vs git checkout

**git switch:** Comando específico para cambiar entre distintas ramas

**git checkout:** tiene varias funciones, incluyendo la capacidad de cambiar entre ramas, hash, tags y commits

Ref: Documentación [git branch](#)

## Gestión de ramas

```
$ git branch <nombre_rama> (Crear)
$ git branch <nombre_rama> -d (Eliminar)
$ git switch <nombre_rama> (Cambiar a una rama)
$ git checkout -b <nombre_rama> (Cambiar a una rama)
$ git merge (Integrar una rama)
```

### git switch vs git checkout

**git switch:** Comando específico para cambiar entre distintas ramas

**git checkout:** tiene varias funciones, incluyendo la capacidad de cambiar entre ramas, hash, tags y commits

Ref: Documentación [git branch](#)

# Cambios temporales

## Stash

Permite guardar temporalmente los cambios que hemos realizado en un archivo, o conjunto de archivos, sin tener que hacer commit

### Reservar cambios

```
$ git stash  
$ git stash list  
$ git stash pop  
$ git stash drop  
$ git stash apply  
$ git stash clear
```

Ref: Documentación [git stash](#)

# Cambios temporales

## Stash

Permite guardar temporalmente los cambios que hemos realizado en un archivo, o conjunto de archivos, sin tener que hacer commit

### Reservar cambios

```
$ git stash  
$ git stash list  
$ git stash pop  
$ git stash drop  
$ git stash apply  
$ git stash clear
```

Ref: Documentación [git stash](#)

# Ignorar Archivos

## .gitignore

Es un archivo de git que se crea en el repositorio en el cual se esta trabajando. Permite ignorar archivos o directorios de los que no deseamos hacer seguimiento.

## Creación del archivo .gitignore

```
$ touch .gitignore  
$ vim .gitignore
```

Documentación [.gitignore](#)



# Ignorar Archivos

## .gitignore

Es un archivo de git que se crea en el repositorio en el cual se esta trabajando. Permite ignorar archivos o directorios de los que no deseamos hacer seguimiento.

## Creación del archivo .gitignore

```
$ touch .gitignore  
$ vim .gitignore
```

Documentación [.gitignore](#)

# Sincronización en remoto

## Fetch

Se utiliza para descargar el historial de cambios del repositorio remoto al repositorio local, pero sin aplicar los cambios.

## Pull

Descarga los cambios del repositorio remoto y los fusiona con los cambios locales. Si hay conflictos en la fusión git intentara de combinar los cambios por defecto.

## Comandos de Sincronización

```
$ git fetch  
$ git config pull.rebase false  
$ git pull
```

# Sincronización en remoto

## Fetch

Se utiliza para descargar el historial de cambios del repositorio remoto al repositorio local, pero sin aplicar los cambios.

## Pull

Descarga los cambios del repositorio remoto y los fusiona con los cambios locales. Si hay conflictos en la fusión git intentara de combinar los cambios por defecto.

## Comandos de Sincronización

```
$ git fetch  
$ git config pull.rebase false  
$ git pull
```

# Sincronización en remoto

## Fetch

Se utiliza para descargar el historial de cambios del repositorio remoto al repositorio local, pero sin aplicar los cambios.

## Pull

Descarga los cambios del repositorio remoto y los fusiona con los cambios locales. Si hay conflictos en la fusión git intentara de combinar los cambios por defecto.

## Comandos de Sincronización

```
$ git fetch  
$ git config pull.rebase false  
$ git pull
```

# Resolución de conflictos

```
paoto@DESKTOP-OT7LS2T MINGW64 /d/Documentos/00-Research/projects/hello-git-fork (main)
$ git push
To https://github.com/paobtorres/hello-git.git
! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/paobtorres/hello-git.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

paoto@DESKTOP-OT7LS2T MINGW64 /d/Documentos/00-Research/projects/hello-git-fork (main)
$ git pull |
```

```
$ git diff
$ git reset <nombre_archivo>
$ git reset --hard
$ git reflog
```

# Resolución de conflictos

```
paoto@DESKTOP-OT7LS2T MINGW64 /d/Documentos/00-Research/projects/hello-git-fork (main)
$ git push
To https://github.com/paobtorres/hello-git.git
! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com/paobtorres/hello-git.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

paoto@DESKTOP-OT7LS2T MINGW64 /d/Documentos/00-Research/projects/hello-git-fork (main)
$ git pull |
```

```
$ git diff
$ git reset <nombre_archivo>
$ git reset --hard
$ git reflog
```

# GitHub funcionalidades

- **Markdown**
- Issues
- GitHub Pages (version Paga)
- Organizaciones

# GitHub funcionalidades

- Markdown
- Issues
- GitHub Pages (version Paga)
- Organizaciones



# GitHub funcionalidades

- Markdown
- Issues
- GitHub Pages (version Paga)
- Organizaciones

# GitHub funcionalidades

- Markdown
- Issues
- GitHub Pages (version Paga)
- Organizaciones

# Markdown: readme.md

Es un archivo que debe incluir una descripción general del proyecto, instrucciones de instalación y configuración. Si es un repositorio publico es recomendable agregar información para constribuir. Ejemplo de archivo [readme.md](#)

Ref: Documentación [markdown](#)



sistemas\_dinamicos\_I/

README.md

in main

Cancel changes

Commit changes...

Edit

Preview

Spaces ▾

2 ▾

Soft wrap ▾

```
1  # Sistemas Dinamicos I
2
3  En este repositorio encontraran los scripts utilizados en los practicos de Analisis Matematico y Algebra
4
5  ## Instrucciones
6
7  Para ejecutar los scripts realizados en [Python] existen dos opciones:
8
9  ### Si nunca han utilizado python
10 Accediendo a Google Colab en el siguiente link: [Google Collab](https://colab.research.google.com/)
11
12 - Una vez en la pagina van a: Archivo -> Nuevo Notebook
13 - El archivo se guardara en su carpeta de [Google Drive](https://drive.google.com/)
14 - Copian y pegan el script que quieren ejecutar en la celda y con Ctrl+Enter ejecutan su contenido. De esta forma
   pueden ir editando y creando nuevas celdas con cada ejercicio.
15 - Funciona desde todos los dispositivos (Computadora, Android, iPhone)
16
17 ### Si utilizan python o conocen las bases:
18 - Pueden utilizar [Spyder](https://www.spyder-ide.org/) o el editor de codigo de su preferencia. (Funciona en visual
   studio code)
19 - Copian el codigo desde el link que corresponde y ejecutan.
20
21 Deben tener en cuenta que tal vez necesiten tener algunas dependencias instaladas si no las han utilizado previamente.
   Las utilizadas en los scripts dados se pueden instalar de la siguiente manera:
22 - `pip3 install numpy`
23 - `pip3 install matplotlib`
24 - `pip3 install pandas`
25
26 Luego de instalarlas prueban ejecutar nuevamente el codigo.
27
```

# GitHub funcionalidades

- Markdown
- Issues
- GitHub Pages (version Paga)
- Organizaciones

# Issues

pyMBE > Issues > #15

 Closed

 Issue created 3 months ago by  Rita Dias Developer

Edit

Reopen issue



## bug: `get_radius_map` only returns the radius of `particle`'s with `acidity=Inert`

`get_radius_map` breaks for `particle`'s with `acidity=Inert`. The reason is that when the `pmb.df` is filtered in that function, it is also filtering the column with the `state_two`. This column is `NaN` for `Inert` particles, so they are discarded when the `NaN` are filtered. Ultimately, this causes `Inert` particles not to show up in the `radius_map`.

Edited 3 months ago by Pablo M. Blanco



0

0






Drag your designs here or [click to upload](#).

# Issues

## Activity

Sort or filter ▾

-  **Rita Dias** assigned to [@paobtorres](#) 3 months ago
-  **Pablo M. Blanco** changed the description 3 months ago
-  **Pao** created branch [bug-get\\_radius\\_map](#) to address this issue 3 months ago



**Pao** @paobtorres · 3 months ago

Developer



[@rsdias](#) Thanks Rita for all the details on the bug it was more easy to fix it.

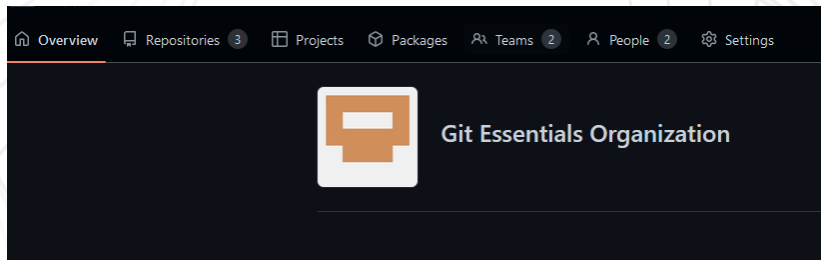


# GitHub funcionalidades

- Markdown
- Issues
- GitHub Pages (version Paga)
- Organizaciones



# Organizaciones



Ref: Documentación [organizaciones](#)

# Herramientas gráficas



## Desktop

GitHub desktop: [Pagina oficial](#)



## GitKraken

GitKraken: [Pagina oficial](#)



SourceTree: [Pagina oficial](#)

# Buenas practicas

- Nombrar los archivos/carpetas/repositorios sin espacios, ni letras como ñ o acentos.
- Realizar `commits` frecuentemente y con indicaciones claras
- Utilizar ramas para desarrollo de tareas especifica
- Continuar aprendiendo y manejar ingles.

# 1. Introducción

■ ¿Qué es Git?   ■ ¿Cómo funciona Git?   ■ ¿Cómo se relaciona Git con GitHub u otros servicios?

# 2. Comandos básicos

■ Comandos útiles de terminal (bash)   ■ Configuración básica de Git  
■ Creación de un repositorio Git   ■ Confirmar cambios en el repositorio  
■ Gestión de archivos   ■ Historial de cambios y estado del repositorio  
■ Hash: identificador de commits

# 3. Colaboración

■ Ramas   ■ Sincronización   ■ Resolución de conflictos   ■ GitHub funcionalidades  
■ Buenas practicas

# 4. Referencias

# Referencias

- Documentación oficial de [git](#)
- Documentación de [GitHub](#) (español)
- Atajos de git: [cheat-sheet](#)

# Agradecimientos

- Facultad Regional San Rafael
- AGENCIA I+D+i (FONCyT) PICT-2021-GRFTI-00090
- UTN (PIDs PATCASR0008459 and PATCASR0008463)
- Soft Matter group from Charles University



UTN FRSR  
Facultad Regional San Rafael  
Universidad Tecnológica Nacional



CHARLES  
UNIVERSITY

## IN CASE OF FIRE



1. git commit



2. git push



3. git out!

```
$ git add  
$ git commit  
$ git push
```