

Introducción a Git & Github

Conceptos básicos

Daniel Jiménez M.

UNAL

13 -11 -2021

Contenido

- ① ¿Qué es git?
- ② ¿Cómo crear un proyecto en git?
- ③ Explorando el estado del repo
- ④ Creando las credenciales de usuario
- ⑤ Evaluando algunos cambios
- ⑥ Trabajando con staging
- ⑦ Recuperando versiones en git
- ⑧ Manejo de Ramas y Merge

¿Qué es git?

Git en el mundo del software (vertiente donde ahora se mueven), es un sistema de versionamiento y trabajo colaborativo que permite desarrollar distintas herramientas de manera optima.

¿Qué es git?

Independiente al lenguaje o tipo de proyecto que maneje (Python,R,Java,Octave¹), o en el sistema operativo en que este, git permite integrar por distintas fases de proyectos todos los requerimientos que se le soliciten, por lo tanto en git hay roles y ramas, de lo cual hablaremos más adelante.

¹Octave es la versión libre de matlab, muy recomendado

¿Qué es git?

Los parámetros de seguridad git es muy bueno y sostenible, pero tiene ciertas particularidades, en especial para el trabajo colaborativo.

¿Cómo crear un proyecto en git?

Empecemos por crear una carpeta, esto se puede hacer de dos maneras:

- Vaya a la terminal o CMD de su pc y busque el directorio **Desktop**
 - Esto se logra con el comando cd : change directory, ejemplo: cd ~/Desktop
 - Ahora escriba el siguiente comando: mkdir ejericico1
- Vaya al escritorio y con click izquierdo cree una nueva carpeta

¿Cómo crear un proyecto en git?

Una vez creada la carpeta, vaya a la terminal o cmd de su computador y escriba el siguiente comando:

- cd ejercicio1

Una vez dentro de la carpeta que acaba de crear coloque el siguiente comando:

- git init

Este comando permitirá inicializar git dentro de su máquina y con base a ello, podra empezar a trabajar en temas de versionamientos

¿Cómo crear un proyecto en git?

```
remote: Total 114 (delta 57), reused 82 (delta 28), pack-reused 0
Receiving objects: 100% (114/114), 8.05 MiB | 8.06 MiB/s, done.
Resolving deltas: 100% (57/57), done.
[→ Desktop cd EDA_Course
[→ EDA_Course git:(master) × ls
Github_tools  Notebooks      README.md      presentaciones
[→ EDA_Course git:(master) × cd Github_tools
[→ Github_tools git:(master) × mkdir ejercicio1
[→ Github_tools git:(master) × cd ejercicio1
[→ ejercicio1 git:(master) × git init
hint: Using 'master' as the name for the initial branch. This default branch name
      is subject to change. To configure the initial branch name to use in all
      of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /Users/danjimenez/Desktop/EDA_Course/Github_
tools/ejercicio1/.git/
[→ ejercicio1 git:(master) ]
```

¿Cómo crear un proyecto en git?

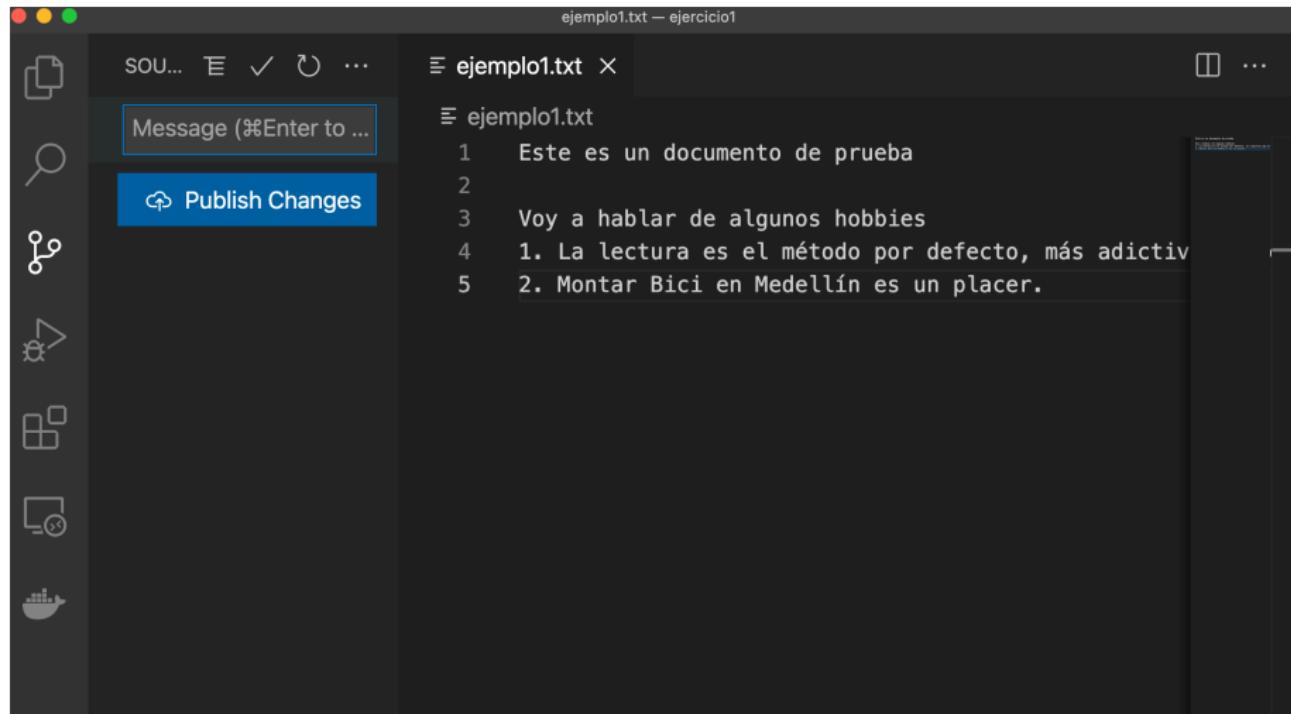
Ahora revisemos que existe dentro del repo que acabamos de crear:

```
Initialized empty Git repository in /Users/danjimenez/Desktop/EDA_Course/Github_tools/ejercicio1/.git/
[→ ejercicio1 git:(master) ls -la
total 0
drwxr-xr-x  3 danjimenez  1010544492   96 Nov 11  08:31 .
drwxr-xr-x  4 danjimenez  1010544492  128 Nov 11  08:25 ..
drwxr-xr-x  9 danjimenez  1010544492  288 Nov 11  08:31 .git
[→ ejercicio1 git:(master) ]
```

¿Cómo crear un proyecto en git?

Empecemos a crear un archivo con CODE

Abra CODE y grabe un documento .txt en la carpeta de ejercicio1



Creando las credenciales de usuario

Ahora enfrentamos un problema y es la autentificación

```
$ git commit -m "Este es el primer commit de este archivo"  
*** Please tell me who you are.
```

Run

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

```
to set your account's default identity.  
Omit --global to set the identity only in this repository.
```

```
fatal: unable to auto-detect email address (got 'Education-Platzi@LAPTOP-E8  
0MR8RH.(none)')
```

Creando las credenciales de usuario

Por defecto debe crear las credenciales que les pide el siguiente comando
git config --list

ejercicio1 — git config --list — git — less ↵ git config --list — 80x24

```
credential.helper=osxkeychain
user.email=danieljimenez@mercadolibre.com.co
user.name=Daniel Jiménez
user.mail=daniel.jimenez@mercadolibre.com.co
pull.ff=only
core.repositoryformatversion=0
core.filemode=true
core.bare=false
core.logallrefupdates=true
core.ignorecase=true
core.precomposeunicode=true
(END)
```

Creando las credenciales de usuario

Para autenticarnos debemos escribir los siguientes comandos:

- `git config --global user.name + su nombre:`
 - Ejemplo: `git config --global user.name "Pepito Perez, PhD."`
- `git config --global user.email + su mail:`
 - Ejemplo: `git config --global user.email "pepito.perez@algunavaina.com"`

Evaluando algunos cambios

The screenshot shows a GitHub desktop application window. The left sidebar contains various icons for repository management, with a blue badge indicating one unread notification. The main area displays a pull request titled "ejemplo1.txt" against the "ejercicio1" branch. The message field contains the text: "Este es un documento de prueba". Below it, the file content is shown as a numbered list:

```
1 Este es un documento de prueba
2
3 Voy a hablar de algunos hobbies
4 1. La lectura es el método por defecto, más adictivo que existe
5 2. Montar Bici en Medellín es un placer.
6 3. Cocino con mi hija, ya que ella le gusta la comida de papá.
7 4. Mi esposa no me deja conducir porque siente que anda con un niño
8
```

A blue button labeled "Publish Changes" is visible in the center-left of the main pane. The bottom status bar shows the branch is "master", there are 68 changes, 0 additions, and 0 deletions, and the file is in "Plain Text" mode.

Evaluando algunos cambios

Ahora evaluenmos los cambios con el comando git log

```
● ● ● ejercicio1 — git log ejemplo1.txt — git — less ↵ git log ejemplo1.txt — 80x24

commit 2024b0a9471039c36f654c4af89e3f7ef8dbfecb (HEAD -> master)
Author: Daniel Jiménez <danieljimenez@mercadolibre.com.co>
Date:   Thu Nov 11 09:04:07 2021 -0500

    coloque una historia de la vida real

commit 8b4c9e06ef13d7482efc5ee47790192bb068fe44
Author: Daniel Jiménez <danieljimenez@mercadolibre.com.co>
Date:   Thu Nov 11 08:47:03 2021 -0500

    No se me ocurrio nada asi que dejo esto la la la la

commit 569c0ddc7602f3a60a6a34f9f8a65b68bc9cf37
Author: Daniel Jiménez <danieljimenez@mercadolibre.com.co>
Date:   Thu Nov 11 08:38:16 2021 -0500

    Add text
(END)
```

Evaluando algunos cambios

Para ver los cambios sobre los archivos aplicamos el comando `git show + nombre del documento`

```
ejercicio1 — git show ejemplo1.txt — git — less + git show ejemplo1.txt — 102x28
commit 2024b0a9471039c36f654c4af89e3f7ef8dbfecb (HEAD -> master)
Author: Daniel Jiménez <danieljimenez@mercadolibre.com.co>
Date:   Thu Nov 11 09:04:07 2021 -0500

    coloque una historia de la vida real

diff --git a/ejemplo1.txt b/ejemplo1.txt
index e840956..b6bb591 100644
--- a/ejemplo1.txt
+++ b/ejemplo1.txt
@@ -4,4 +4,4 @@ Voy a hablar de algunos hobbies
 1. La lectura es el método por defecto, más adictivo que existe
 2. Montar Bici en Medellín es un placer.
 3. Cocino con mi hija, ya que ella le gusta la comida de papá.
-
+4. Mi esposa no me deja conducir porque siente que anda con un niño!!
(END)
```

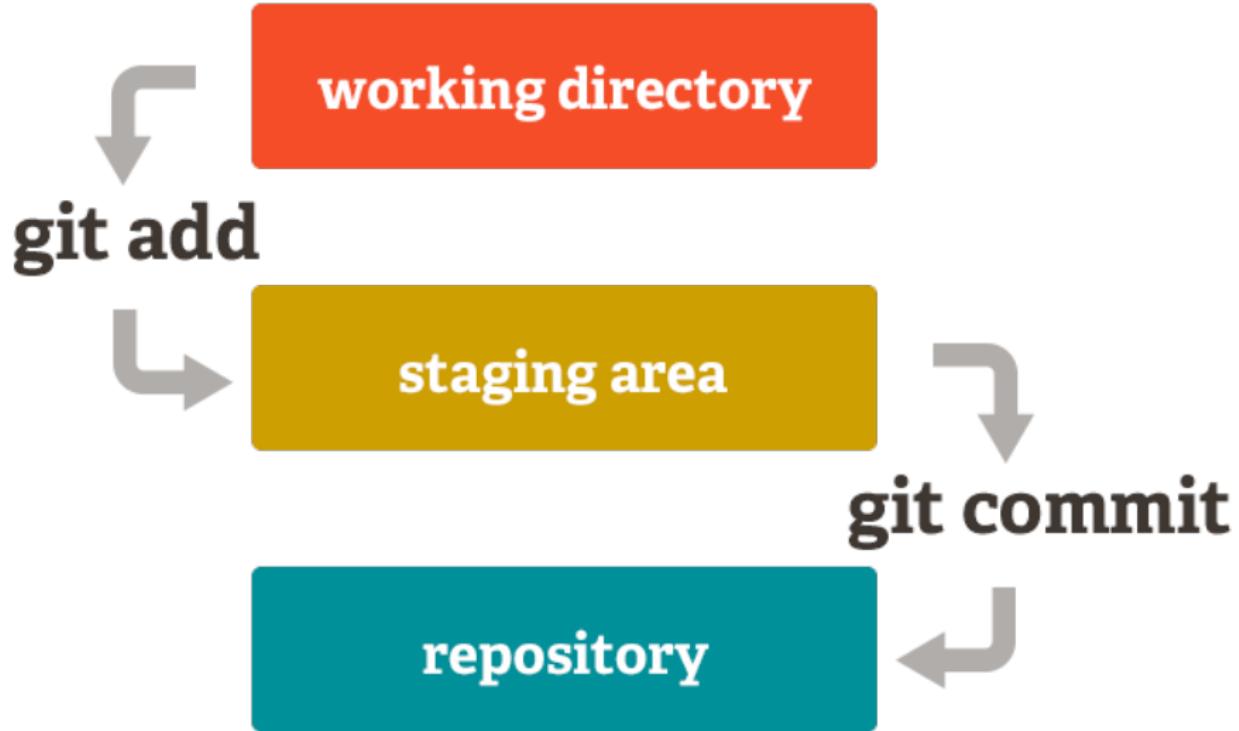
Evaluando algunos cambios

Para evaluar diferencias entre commits podemos hacer lo siguiente : `git`

Trabajando con staging

El **staging** es una zona de preparación en donde las versiones de los proyectos quedan grabadas en la ram, esto ocurre cuando se emplea el comando `git add`

Trabajando con staging



Trabajando con staging

Hay que tener presente que se puede tener cambios que no esten en staging: Cuando no se agregan los cambios.

Para trabajar lienzos de desarrollo es necesario trabajar con branch

Recuperando versiones en git

Para volver a una versión del código que estamos trabajando se emplea el comando `git reset + commit + --hard`

A continuación se desarrollará un ejemplo, note las siguientes fases: i) Se creará un cambio en un documento de ejemplo, ii) se agregará el cambio con los protocolos ya mencionados en clase, iii) se hará el commit, iv) Evaluaremos dos formas de rescatar una versión anterior.

Recuperando versiones en git

Este es un ejemplo sobre como manejar versionamiento de repositorios
Ahora quiero agregar unas líneas de nada a este documento a ver que pasa.

Pero ojo es importante anotar algo

1. Este ejercicio es útil solo si lo practican solos!!!
2. La practica hace al maestro, nunca lo olviden
3. En una diapositiva que viene a continuación viene una tarea sobre esta fase.

~
~
~
~
~
~
~
~
~
~
~
~
~
~
~

Recuperando versiones en git

```
→ Github_tools git:(master) ✘ git add Example.txt
→ Github_tools git:(master) ✘ git commit -m 'Agregue unas notas para la clase'
[master f5aa365] Agregue unas notas para la clase
 1 file changed, 6 insertions(+)
→ Github_tools git:(master) ✘ git log
→ Github_tools git:(master) ✘ █
```

Recuperando versiones en git

```
commit f5aa365e613bb71511c2ddb8883accebb6ac10b2 (HEAD -> master)
Author: Daniel Jiménez <danieljimenez@mercadolibre.com.co>
Date:   Sat Nov 13 11:48:06 2021 -0500

    Agregue unas notas para la clase

commit d8df69e04e3417c5dd70f92d539b554396cb9042 (origin/master, origin/HEAD)
Author: Daniel Jiménez <danieljimenez@mercadolibre.com.co>
Date:   Thu Nov 11 17:09:13 2021 -0500

    Add class 2

commit 42003a9ea2355ec0e72ee71f321f499a29b108ff
Author: Daniel Jiménez <danieljimenez@mercadolibre.com.co>
Date:   Thu Nov 11 17:07:05 2021 -0500

    Add class

commit 4dc6370ed2fc91e4cbb8b689005107016765cafc (tag: eda, origin/esteban)
Author: Daniel Jiménez <danieljimenez@mercadolibre.com.co>
Date:   Wed Nov 10 17:41:59 2021 -0500

    Quite eso

commit e32de2a41cb4f042a9026a19fabdc0906a38754e
Author: Daniel Jiménez <danieljimenez@mercadolibre.com.co>
Date:   Wed Nov 10 17:40:07 2021 -0500
:|
```

Recuperando versiones en git

```
Last login: Sat Nov 13 11:06:34 on ttys002
→ Github_tools git:(master) ✘ git reset aedbbe4e495c8bb1562e01353d8349bd90959805 --hard
warning: unable to rmdir 'Github_tools/ejercicio1': Directory not empty
HEAD is now at aedbbe4 First Class
→ Github_tools git:(master) ✘ vi Example.txt
→ Github_tools git:(master) ✘ █
```

Recuperando versiones en git

Recuperando versiones en git

Recuperando versiones en git

```
→ Github_tools git:(master) ✘ git status
On branch master
Your branch is behind 'origin/master' by 29 commits, and can be fast-forwarded
  (use "git pull" to update your local branch)

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:   Example.txt
```

Recuperando versiones en git

```
→ Github_tools git:(master) ✘ git commit -m 'Voy a mostrar que pasa cuando agrego algo y lo devuelvo con soft'  
[master 6be351c] Voy a mostrar que pasa cuando agrego algo y lo devuelvo con soft  
 1 file changed, 6 insertions(+)  
→ Github_tools git:(master) ✘
```

Recuperando versiones en git

```
commit 6de351c91dc5/catsbe/00811/50095d2/zoooo (HEAD -> master)
Author: Daniel Jiménez <danieljimenez@mercadolibre.com.co>
Date:   Sat Nov 13 12:02:52 2021 -0500

    Voy a mostrar que pasa cuando agrego algo y lo devuelvo con soft
```

```
Youtube_tools/Example.txt | 6 ++++++
1 file changed, 6 insertions(+)
```

```
commit aedbbe4e495c8bb1562e01353d8349bd90959805
Author: Daniel Jiménez <daniel.jimenez@mercadolibre.com.co>
Date:   Fri Oct 29 10:39:43 2021 -0500
```

```
First Class
```

```
Youtube_tools/Example.txt |  2 ++
README.md               | 18 ++++++++++++++-
presentaciones/clase0.Rmd | 49 ++++++++++++++++++++++++++++++++
presentaciones/clase0.pdf | Bin 0 -> 18290 bytes
4 files changed, 67 insertions(+), 2 deletions(-)
```

```
commit 9c96169002de401f05fac85c84ead3026c3d9fbf
Author: Daniel Jiménez Martínez <cjmenez187@aol.com>
Date:   Fri Oct 29 09:42:59 2021 -0500
```

Recuperando versiones en git

```
Este es un ejemplo sobre como manejar versionamiento de repositorios
```

```
Como borre todo de manera --hard, es casi imposible guardar los cambios que habia escrito  
y es por ello que vamos a trabajar ahora con --soft
```

```
En esto podremos tener una ventaja y es la forma de ejecutar un back to the past donde todo queda grabado en staging
```

Recuperando versiones en git

```
→ Github_tools git:(master) ✘ git log --stat
→ Github_tools git:(master) ✘ vi Example.txt
→ Github_tools git:(master) ✘ git add Example.txt
→ Github_tools git:(master) ✘ git commit -m "Vamos creando maestria en el manejo de git"
[master a24b8ad] Vamos creando maestria en el manejo de git
 1 file changed, 3 insertions(+)
```

Recuperando versiones en git

Ahora vamos a ver como era el archivo antes con la función `checkout`

```
Este es un ejemplo sobre como manejar versionamiento de repositorios

Como borre todo de manera --hard, es casi imposible guardar los cambios que habia escrito
y es por ello que vamos a trabajar ahora con --soft

En esto podremos tener una ventaja y es la forma de ejecutar un back to the past donde todo queda grabado en staging

~
~
```

Recuperando versiones en git

Notese que para guardar esta versión de los cambios debería hacer un commit

```
→ Github_tools git:(master) ✘ git status
On branch master
Your branch and 'origin/master' have diverged,
and have 2 and 29 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
      modified:   Example.txt
```

Recuperando versiones en git

Pero como no se ha hecho ningun cambio oficial se puede ver la versión de la rama master

```
→ Github_tools git:(master) ✘ git checkout master Example.txt
Updated 1 path from 84c0edd
→ Github_tools git:(master) ✘ vi Example.txt
```

Recuperando versiones en git

```
Este es un ejemplo sobre como manejar versionamiento de repositorios
```

```
Como borre todo de manera --hard, es casi imposible guardar los cambios que habia escrito  
y es por ello que vamos a trabajar ahora con --soft
```

```
En esto podremos tener una ventaja y es la forma de ejecutar un back to the past donde todo queda grabado en staging
```

```
Ahora aprenderemos a ver las distintas versiones que tenemos de este documentos y anticipo que ser viene la tarea....
```

```
La tarea seran con 10 ejemplos desde su terminal o máquina local,..... ahora les dire más.
```

```
~  
~  
~
```

Tarea

Va a escribir un documento de extensión .txt en donde escribirá su perfil, en dicho perfil hará unos cambios y esos cambios los va a comparar y volver en el tiempo con git reset y git checkout, me va a enviar evidencias de ello (captura de pantalla).

Recuerde que el plazo es Domingo 11:59 a.m

Otra vez es una tarea que solo tiene dos notas : 0 o 5

Tarea

Excusas no validas: las generales

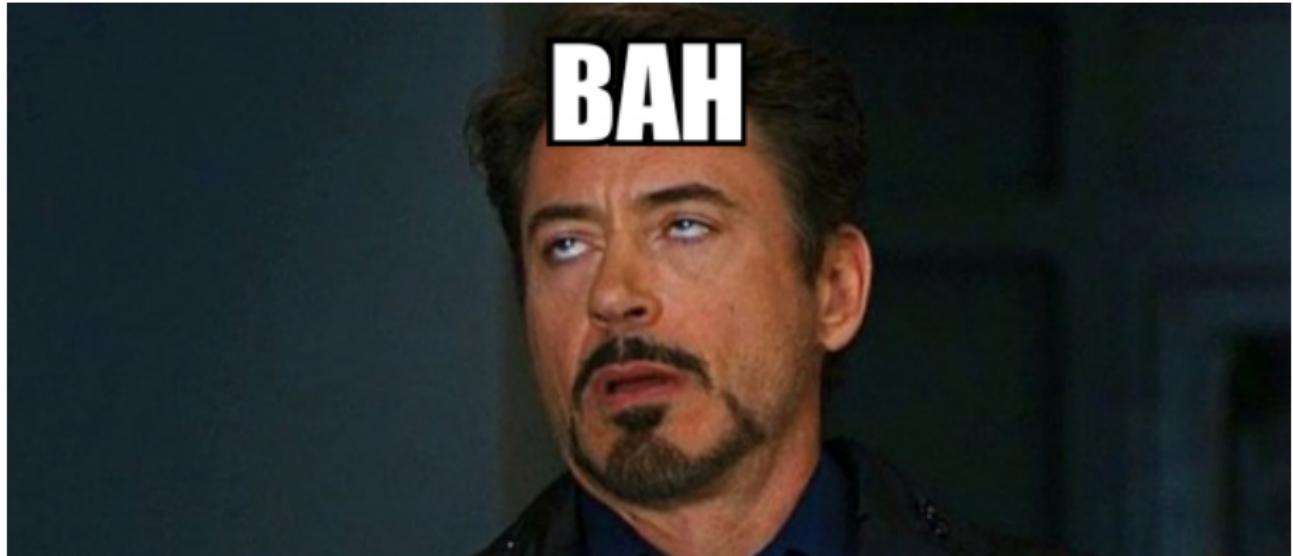
- ① Profe no supe hacerlo: Lo hemos hecho hasta la saciedad en clase, está el recurso de la monitoria y estan las grabaciones de las clases



Tarea

Excusas no validas: las generales

- ② Profe se me daño el computador : Lo siento no puedo hacer nada, solucione, seguro alguien en su entorno puede prestarle uno para la tarea, sino el de la oficina, sino en algún lugar de la unidad geográfica consigue uno.

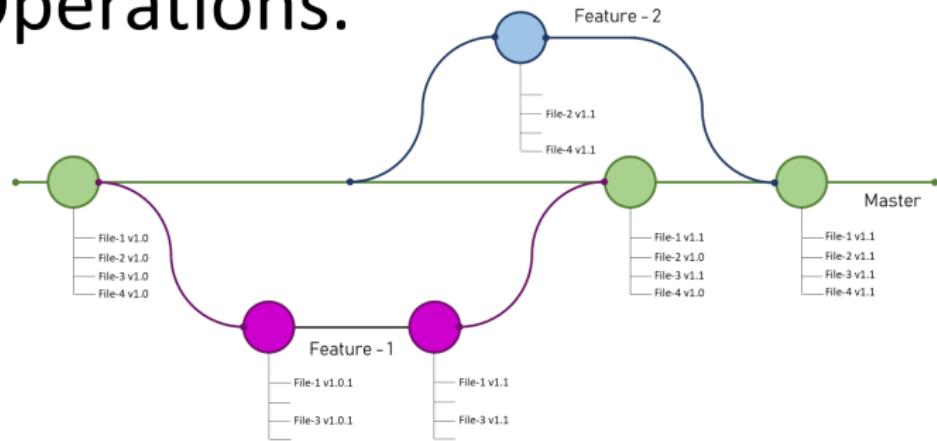


Tarea

- ③ Profe no me alcanzo el tiempo: se aviso con tiempo, y por ello todo lo envio después de las 11:59 am no será tomado en cuenta



GIT Branch and its Operations.



Manejo de Ramas y Merge

El manejo de ramas se da desde el repositorio virtual (recuerde que debe tener una cuenta de github), y desde acá se trabajará con un nuevo comando y es `git push`, del cual hablaremos más adelante.

Manejo de Ramas y Merge

Manejo de ramas

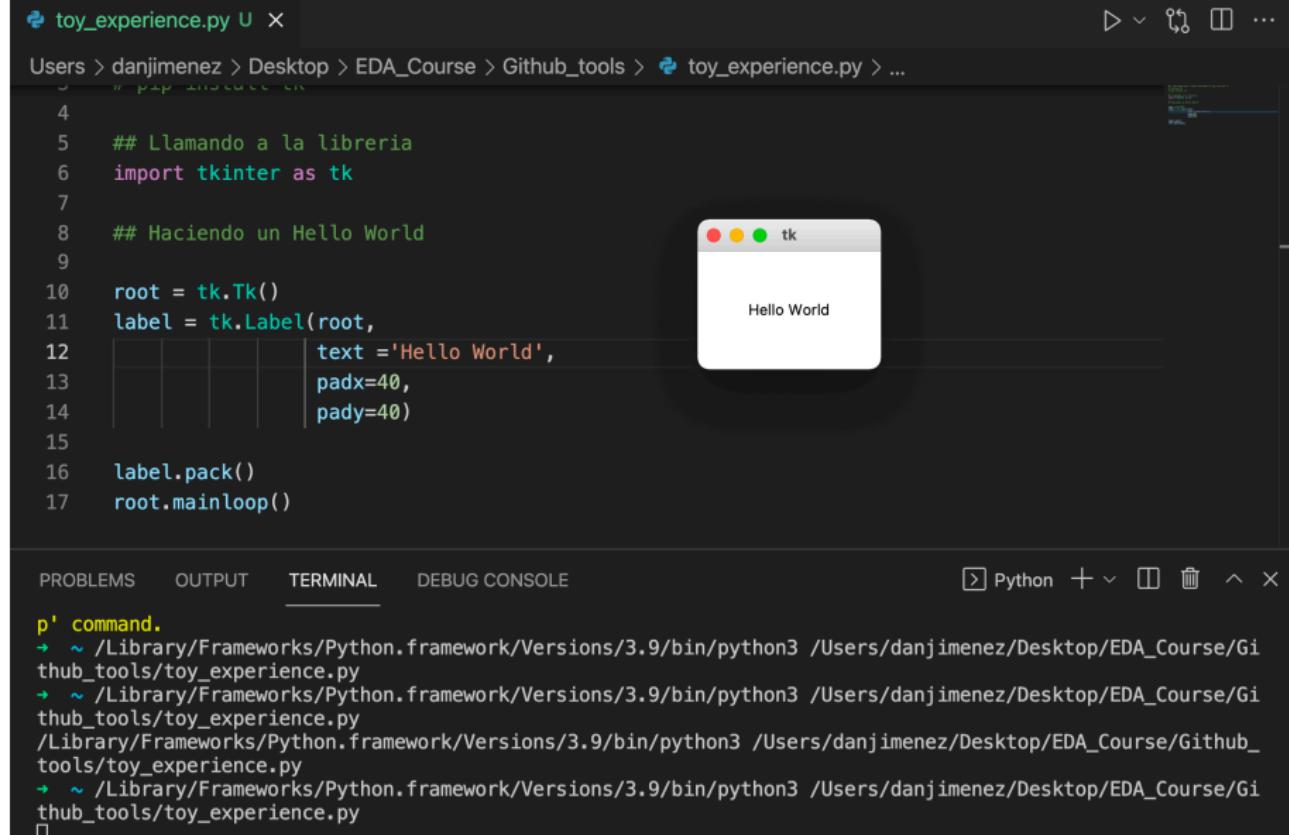
- ① Las ramas son formas en la cual se pueden trabajar de manera colaborativa por secciones del proyecto
- ② El ideal es que una vez tengamos verificados los cambios se pueden fusionar con la rama específica.
- ③ Por lo tanto las ramas son copias basadas en el último commit

Manejo de Ramas y Merge

El comando para crear las ramas es git branch + 'Nombre de la rama' y para movernos a la nueva rama se hace a través git checkout + 'Nombre de la rama'

```
→ Github_tools git:(master) ✘ git checkout python_example
Switched to branch 'python_example'
→ Github_tools git:(python_example) ✘ ls -lh
total 8
-rw-r--r-- 1 danjimenez 1010544492 545B Nov 13 12:20 Example.txt
drwxr-xr-x 4 danjimenez 1010544492 128B Nov 11 08:35 ejercicio1
→ Github_tools git:(python_example) ✘
```

Manejo de Ramas y Merge



The screenshot shows a code editor interface with a dark theme. On the left, a file named `toy_experience.py` is open, containing the following Python code:

```
4
5     ## Llamando a la libreria
6     import tkinter as tk
7
8     ## Haciendo un Hello World
9
10    root = tk.Tk()
11    label = tk.Label(root,
12                      text ='Hello World',
13                      padx=40,
14                      pady=40)
15
16    label.pack()
17    root.mainloop()
```

To the right of the code, a small window titled "tk" displays the text "Hello World". Below the code editor, there are tabs for PROBLEMS, OUTPUT, TERMINAL, and DEBUG CONSOLE. The TERMINAL tab is active, showing the command history:

```
p' command.
→ ~ /Library/Frameworks/Python.framework/Versions/3.9/bin/python3 /Users/danjimenez/Desktop/EDA_Course/Github_tools/toy_experience.py
→ ~ /Library/Frameworks/Python.framework/Versions/3.9/bin/python3 /Users/danjimenez/Desktop/EDA_Course/Github_tools/toy_experience.py
/Library/Frameworks/Python.framework/Versions/3.9/bin/python3 /Users/danjimenez/Desktop/EDA_Course/Github_tools/toy_experience.py
→ ~ /Library/Frameworks/Python.framework/Versions/3.9/bin/python3 /Users/danjimenez/Desktop/EDA_Course/Github_tools/toy_experience.py
```

Manejo de Ramas y Merge

```
|→ Github_tools git:(python_example) ✘ git status  
On branch python_example  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
    ejercicio1/  
    toy_experience.py
```

Manejo de Ramas y Merge

```
|→ Github_tools git:(python_example) ✘ ls -la
total 16
drwxr-xr-x  5 danjimenez  1010544492  160 Nov 13 16:19 .
drwxr-xr-x  6 danjimenez  1010544492  192 Nov 13 11:50 ..
-rw-r--r--  1 danjimenez  1010544492  545 Nov 13 12:20 Example.txt
drwxr-xr-x  4 danjimenez  1010544492  128 Nov 11 08:35 ejercicio1
-rw-r--r--  1 danjimenez  1010544492  317 Nov 13 16:28 toy_experience.py
```

Manejo de Ramas y Merge

```
git
→ Github_tools git:(python_example) ✘ git checkout master
Switched to branch 'master'
Your branch and 'origin/master' have diverged,
and have 2 and 29 different commits each, respectively.
(use "git pull" to merge the remote branch into yours)
→ Github_tools git:(master) ✘ ls -la
total 8
drwxr-xr-x  4 danjimenez 1010544492 128 Nov 13 16:32 .
drwxr-xr-x  6 danjimenez 1010544492 192 Nov 13 11:50 ..
-rw-r--r--  1 danjimenez 1010544492 545 Nov 13 12:20 Example.txt
drwxr-xr-x  4 danjimenez 1010544492 128 Nov 11 08:35 ejercicio1
```

Manejo de Ramas y Merge

```
git
→ Github_tools git:(python_example) ✘ git add toy.py
→ Github_tools git:(python_example) ✘ git commit -m 'Agregue un dibujo curioso'
[python_example e475b28] Agregue un dibujo curioso
 1 file changed, 19 insertions(+)
 create mode 100644 Github_tools/toy.py
→ Github_tools git:(python_example) ✘
```

Manejo de Ramas y Merge

Ahora que queremos unir las dos ramas, usaremos el comando `merge` , primero hay que ubicarnos en la rama principal, después colocar `git merge + 'Nombre de la rama a fusionar'`

Manejo de Ramas y Merge

```
→ Github_tools git:(python_example) ✘ git checkout master
Switched to branch 'master'
Your branch and 'origin/master' have diverged,
and have 2 and 29 different commits each, respectively.
  (use "git pull" to merge the remote branch into yours)
→ Github_tools git:(master) ✘ ls -la
total 24
drwxr-xr-x  5 danjimenez 1010544492   160 Nov 13 17:04 .
drwxr-xr-x  7 danjimenez 1010544492   224 Nov 13 16:47 ..
-rw-r--r--@ 1 danjimenez 1010544492  6148 Nov 13 16:48 .DS_Store
-rw-r--r--  1 danjimenez 1010544492   545 Nov 13 12:20 Example.txt
drwxr-xr-x  4 danjimenez 1010544492   128 Nov 11 08:35 ejercicio1
→ Github_tools git:(master) ✘ git merge python_example
Updating a24b8ad..e475b28
Fast-forward
  Github_tools/toy.py          | 19 ++++++=====
  Github_tools/toy_experience.py | 17 ++++++=====
  2 files changed, 36 insertions(+)
  create mode 100644 Github_tools/toy.py
  create mode 100644 Github_tools/toy_experience.py
→ Github_tools git:(master) ✘ █
```