

# ***Programación Avanzada***

## **IIC2233 2024-1**

Hernán Valdivieso - Daniela Concha - Francisca Ibarra - Dante Pinto - Francisca Cattán



# Anuncios

1. ¡Respondan el formulario!
2. Hoy tenemos la primera Actividad 0, que no lleva puntaje.
3. Mañana se publica la Tarea 1.

---

# Guía de estilo: PEP8

# Ejemplo de cambio de estilo ¿Qué cambia en el formato?

## Antes de PEP8

```
def mifuncion(nombre,apellido):
    print(nombre+" "+apellido)
    print(time.today()      )

listaDeProfes = [["Daniela", "Concha"],
["Francisca", "Cattan"], ["Hernan",
"Valdivieso"],["Francisca",
"Ibarra"],["Dante", "Pinto"]]

for lista in listaDeProfes:
    mifuncion(lista[0],

                lista[1])

import time
```

## Después de PEP8

```
import time

def mi_funcion(nombre, apellido):
    print(nombre + " " + apellido)
    print(time.today())

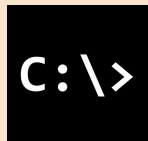
lista_de_profes = [
    ["Daniela", "Concha"],
    ["Francisca", "Cattan"],
    ["Hernan", "Valdivieso"],
    ["Francisca", "Ibarra"],
    ["Dante", "Pinto"]
]

for lista in lista_de_profes:
    mi_funcion(lista[0], lista[1])
```

# Uso de terminal

# Edición y ejecución de códigos

terminal



editor de código



gestión de cambios



git



GitHub



python<sup>TM</sup>

# Comandos de terminal

`comando argumentos* -opciones*`

## Windows:

- `cd` Ver y cambiar directorio actual
- `dir` Listar contenido del directorio
- `mkdir` Crear un directorio nuevo
- `echo` Mostrar mensaje en la terminal
- `>` Guardar contenido en un archivo



`echo "contenido" > nombre.extension`

## Linux y macos:

- `pwd` Ruta absoluta actual
- `cd` Ver y cambiar directorio actual
- `ls` Listar contenido del directorio
- `mkdir` Crear un directorio nuevo
- `touch` Crear un archivo nuevo



`touch nombre.extension`

# El curso en GitHub

<https://github.com/IICT2233>



**Llevando el curso  
a mi PC**

# Antes de partir... ¿tenemos git?

Escribir en la consola (Git Bash cuenta) lo siguiente...

git

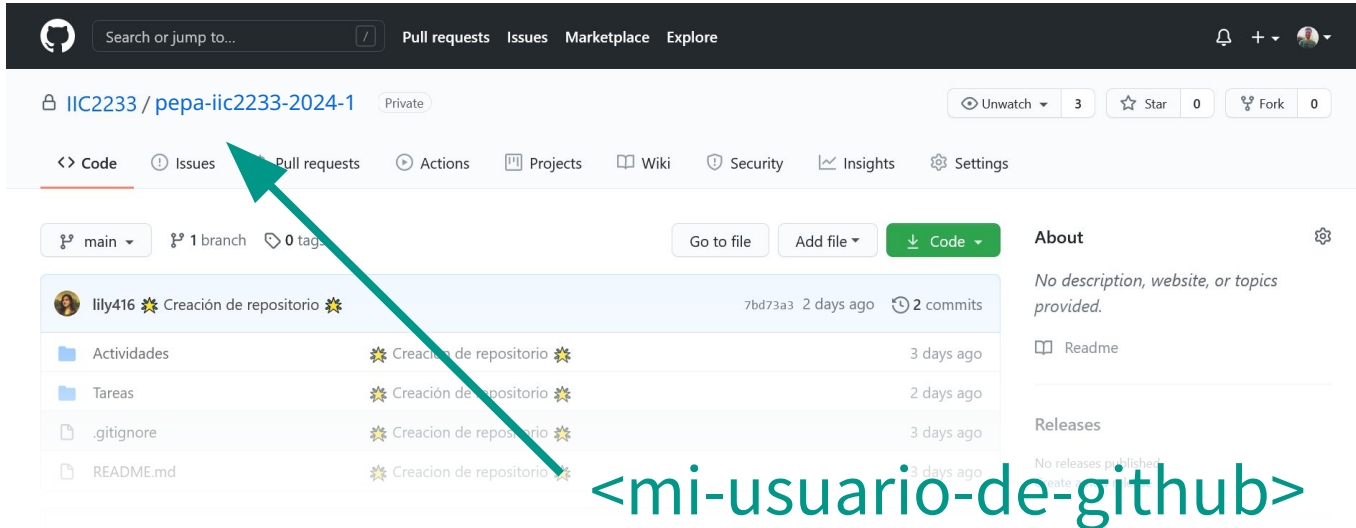
Si la consola les responde con la documentación del uso de git (algo como...  
usage: git [--version] ...), significa que tienen git instalado. Si no lo tienen  
instalado, las instrucciones están en [la wiki de la página oficial del curso](#).

# Antes de partir... ¿tenemos un repositorio?

[https://github.com/IIC2233/&ltmi-usuario-de-github>-iic2233-2023-2](https://github.com/IIC2233/<mi-usuario-de-github>-iic2233-2023-2)

Si completaron el form, debería haberles llegado un **correo de GitHub**, con una invitación a un repositorio en la organización del curso en GitHub.

Si no lo han hecho... ¡Completenlo!



The screenshot shows the GitHub interface for the repository `IIC2233 / pepa-iic2233-2024-1`. The repository is private and has 3 unwatched items, 0 stars, and 0 forks. The 'Pull requests' tab is highlighted with a green arrow pointing to it from the text `&ltmi-usuario-de-github>` at the bottom right. The repository has 1 branch (main) and 0 tags. The commit history shows a recent commit by `lily416` titled 'Creación de repositorio' with 2 commits. The file list includes `Actividades`, `Tareas`, `.gitignore`, and `README.md`, all with recent commit history. The 'About' section indicates no description, website, or topics are provided.

# Llevando el repositorio a nuestro PC

Los ayudantes ya nos dieron nuestro repositorio. Si queremos bajar ese repositorio, hacer cambios y subirlos, tenemos que crear una copia local. Es hora de **clonar**.



# Llevando el repositorio a nuestro PC

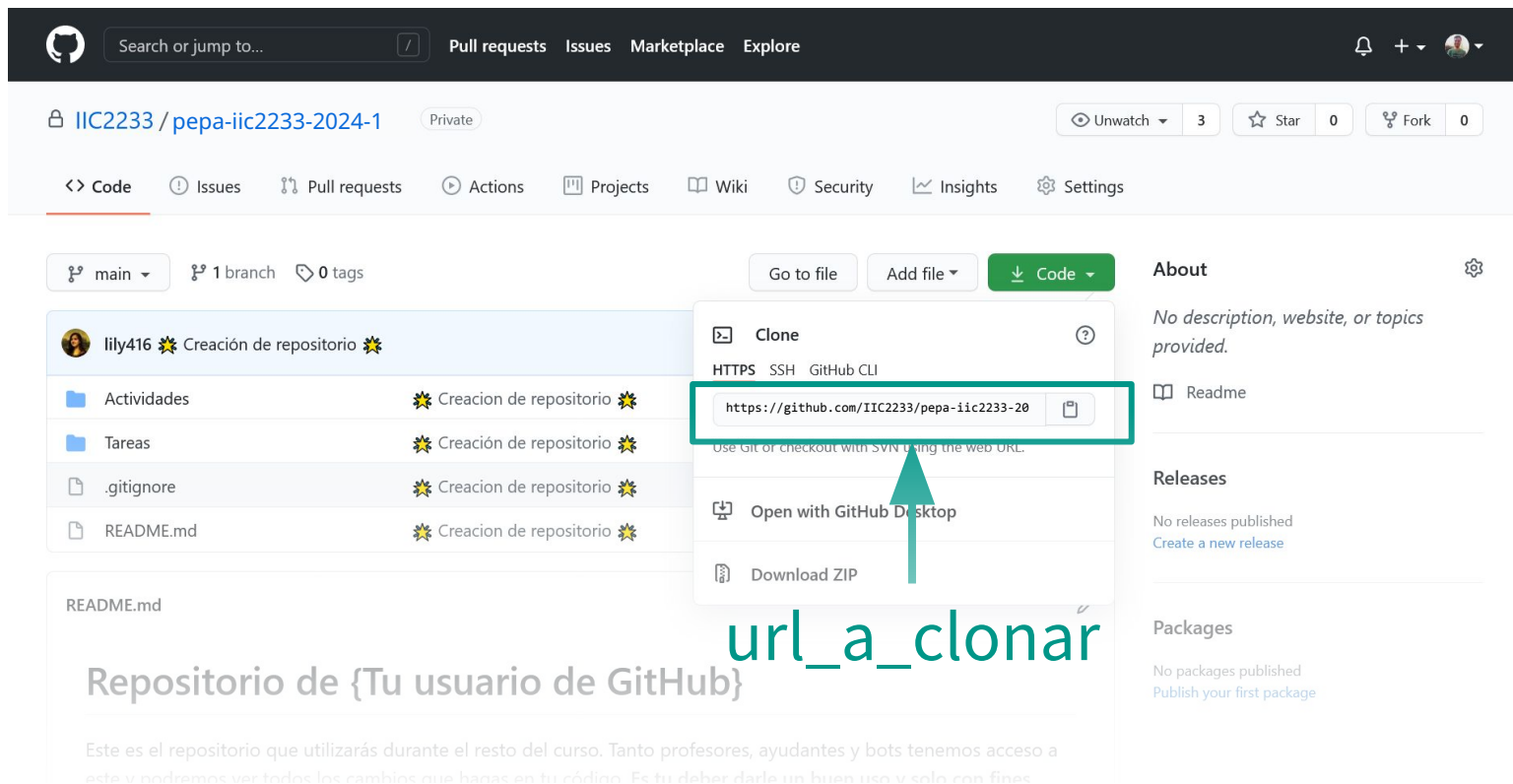
Los ayudantes ya nos dieron nuestro repositorio. Si queremos bajar ese repositorio, hacer cambios y subirlos, tenemos que crear una copia local. Es hora de **clonar**.

```
git clone url_a_clonar
```

¡NO HAGAN NADA AÚN! Nos falta algo...

¿De dónde obtengo la URL para clonar mi repositorio?

# Llevando el repositorio a nuestro PC



The screenshot shows the GitHub interface for a repository named `IIC2233 / pepa-iic2233-2024-1`. The repository is private and has 3 watchers, 0 stars, and 0 forks. The main branch is selected. A dropdown menu is open under the 'Code' button, showing options to clone the repository using HTTPS, SSH, or GitHub CLI. The HTTPS URL `https://github.com/IIC2233/pepa-iic2233-20` is highlighted with a green box. A green arrow points to this URL, and the text `url_a_clonar` is written in green below the arrow. The repository's README.md file is visible in the background, showing the title 'Repositorio de {Tu usuario de GitHub}' and a description.

Search or jump to... Pull requests Issues Marketplace Explore

IIC2233 / pepa-iic2233-2024-1 Private Unwatch 3 Star 0 Fork 0

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 1 branch 0 tags

lily416 Creación de repositorio

Actividades Creación de repositorio

Tareas Creación de repositorio

.gitignore Creación de repositorio

README.md Creación de repositorio

Clone ?

HTTPS SSH GitHub CLI

`https://github.com/IIC2233/pepa-iic2233-20`

Use Git or checkout with SVN using the web URL.

Open with GitHub Desktop

Download ZIP

About

No description, website, or topics provided.

Readme

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

README.md

Repositorio de {Tu usuario de GitHub}

Este es el repositorio que utilizarás durante el resto del curso. Tanto profesores, ayudantes y bots tenemos acceso a este y podremos ver todos los cambios que hagas en tu código. Es tu deber darle un buen uso y solo con fines

# Llevando el repositorio a nuestro PC

Recuerden estar en la carpeta en la que quieren mantener el repositorio, ya que se creará una carpeta con los contenidos.

```
git clone url_a_clonar
```

Repositorio  
en GitHub

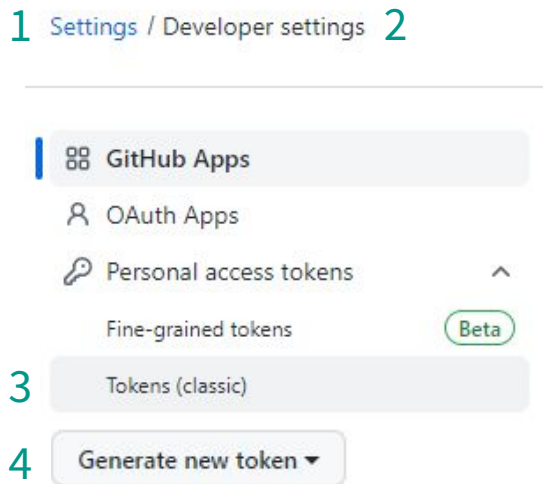


Repositorio  
local

# Personal Access Token

Para poder poder acceder a repositorios que no son públicos necesitamos un **token**. Este lo podemos conseguir desde nuestro perfil en la página de github o directamente desde:

<https://github.com/settings/tokens/new>





# Personal Access Token

5 **Note**

IIC2233

What's this token for?

6 **Expiration \***

No expiration ⌵ The token will never expire!

**Select scopes**

Scopes define the access for personal tokens. [Read more about OAuth scopes.](#)

7

<input checked="" type="checkbox"/> repo	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<hr/>	
<a href="#">Full control of public user GPG keys (Developer Preview)</a>	
<input checked="" type="checkbox"/> write:pgp_key	Write public user GPG keys
<input checked="" type="checkbox"/> read:pgp_key	Read public user GPG keys

8 **Generate token** Cancel

- Debe tener una descripción (Note)
- No debe expirar (Expiration)
- Todos los permisos (Scopes)

¡¡TENDRÁN UNA SOLA OPORTUNIDAD  
PARA VERLO Y COPIARLO!!

Si fallan pueden borrarlo y crear uno nuevo.

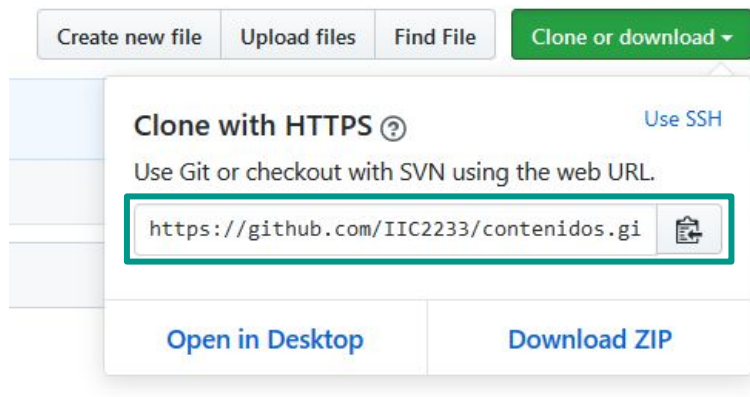
# Repositorios del curso

Ahora que saben clonar, aprovechemos de descargar los repositorios del curso: Syllabus (donde se suben enunciados) y contenidos (donde se sube material del curso).

```
git clone url_a_clonar
```

```
git clone https://github.com/IIC2233/syllabus.git
```

```
git clone https://github.com/IIC2233/contenidos.git
```



Syllabus



contenidos



pepa-iic2233-2024-1

# Movernos entre carpetas

Dado que estamos trabajando en la consola, es necesario conocer al menos este comando, que nos permite cambiar de carpeta o directorio (*change directory*).

```
cd directorio_destino
```

Para ingresar a una carpeta basta con que escribamos su nombre en `directorio_destino`. Si queremos salir de esa carpeta, subiendo un nivel, tenemos que usar *punto-punto*:

```
cd ..
```

# Trabajemos con nuestro repositorio

Para poder ocupar los comandos de `git`, debemos estar dentro de un repositorio clonado, por lo que debemos movernos a la carpeta correspondiente.

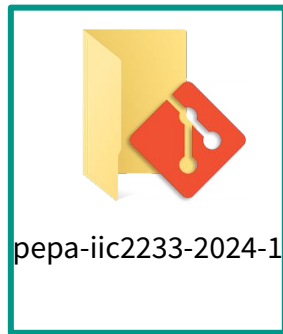
```
git clone url_a_clonar  
cd <usuario>-iic2233-2024-1
```



Syllabus



contenidos



pepa-iic2233-2024-1

# Desde ahora... `git status`

Usen `git status` muy seguido. Antes y después de hacer algo, ocupen `git status`. Esto les permitirá saber qué están haciendo, si les faltó un paso, y de hecho les sugerirá comandos si es que les falta algo por hacer.

`git status`

```
~/Desktop/pepa-iic2233-2024-1 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

# Editemos algo... README.md

Es un archivo para ser leído antes de la ejecución de cualquier código. En su repositorio hay un README para ustedes y **en cada tarea ustedes deben hacer otro README** para que los ayudantes puedan revisar de mejor forma su entrega.

## README.md

```
1  # Repositorio de {Tu usuario de GitHub}
2
3  Este es el repositorio que utilizarás dur
4
5  **Asegúrate de seguir la estructura de la
6
7  Todo el proceso de recolección de tareas
8
9  ## Datos del alumno
10
11  | Nombre      | Mail UC      |
12  | :-----:   | :-----:   |
13  | {Tu nombre} | {Tu correo UC} |
14
```

## README.md

```
1  # Repositorio de pepa
2
3  Este es el repositorio que utilizarás dur
4
5  **Asegúrate de seguir la estructura de la
6
7  Todo el proceso de recolección de tareas
8
9  ## Datos del alumno
10
11  | Nombre      | Mail UC      |
12  | :-----:   | :-----:   |
13  | Pepa        | pepa@uc.cl   |
14
```

# Hicimos un cambio... git status

Antes de hacer cualquier cosa... git status.

```
~/Desktop/pepa-iic2233-2024-1 (main)
```

```
$ git status
```

```
On branch main
```

```
Your branch is up to date with 'origin/main'.
```

```
Changes not staged for commit:
```

```
(use "git add <file>..." to update what will be committed)
```

```
(use "git restore --staged <file>..." discard changes in working directory)
```

```
modified: README.md
```

# Seleccionar cambios... git add (y status)

Hagámosle caso a `git status`, usemos `git add`. Con esto, definimos la lista de cambios que queremos declarar.

```
~/Desktop/pepa-iic2233-2024-1 (main)
```

```
$ git add README.md
```

```
~/Desktop/pepa-iic2233-2024-1 (main)
```

```
$ 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)
```

```
    modified:   README.md
```



# Declarar cambios... git commit (y status)

Para declarar los cambios que hemos realizado, y ponerles un mensaje describiendo los cambios, utilizamos `git commit -m "Mensaje"`.

```
~/Desktop/pepa-iic2233-2024-1 (main)
$ git commit -m ":sparkles: Agregué mis datos"
[main 87907ba] :sparkles: Agregué mis datos
1 file changed, 2 insertions(+), 2 deletions(-)

~/Desktop/pepa-iic2233-2024-1 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

nothing to commit working tree clean
```

Los mensajes son **muy importantes**. Son una ayuda a ustedes en el futuro.

# Subir cambios... git push (y status)

Para dejar en GitHub los cambios que hemos declarado, debemos hacer los que nos dice `git status` y hacer `git push`.

```
~/Desktop/pepa-iic2233-2024-1 (main)
```

```
$ git push
```

```
Enumerating objects: 5, done.
```

```
Counting objects: 100% (5/5), done.
```

```
Delta compression using up to 4 threads
```

```
Compressing objects: 100% (3/3), done.
```

```
Writing objects: 100% (3/3), 354 bytes | 354.00 KiB/s, done.
```

```
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
```

```
remote: Resolving deltas: 100% (2/2), completed with 2 local objects. To
```

```
https://github.com/IIC2233/pepa-iic2233-2024-1.git
```

```
7bd73a3.. 87907ba main -> main
```

# Subir cambios... `git push` (y `status`)

Ahora, si hacemos `git status` nuevamente, nos damos cuenta de que tenemos lo mismo que GitHub y no tenemos cambios pendientes por declarar. Recuerden usar `git status` muy seguido.

```
~/Desktop/pepa-iic2233-2024-1 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

# Flujo de git

Mi computador

Lista de cambios

Repositorio local

GitHub

`git add`



`git commit`



`git push`



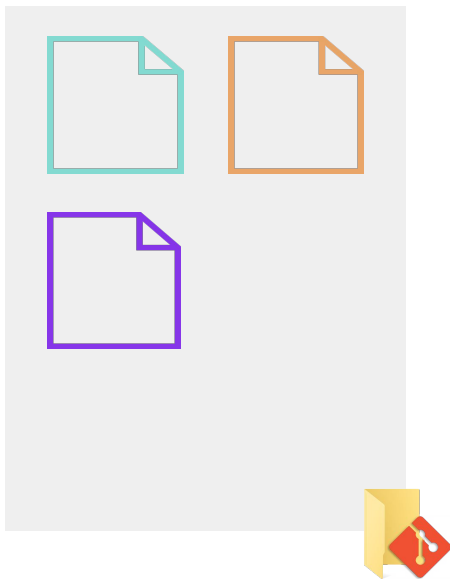
Todo esto ocurre localmente

Internet

# Entendiendo git

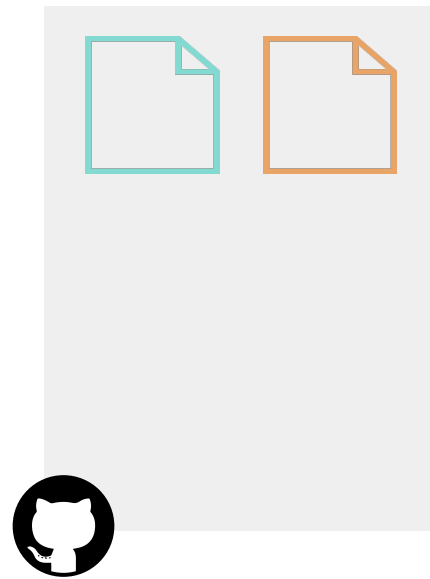
# ¿Qué acabamos de hacer?

Mi computador



```
git add README.md
```

GitHub

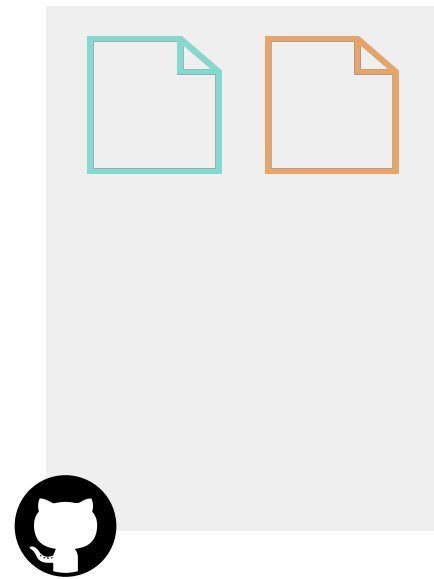
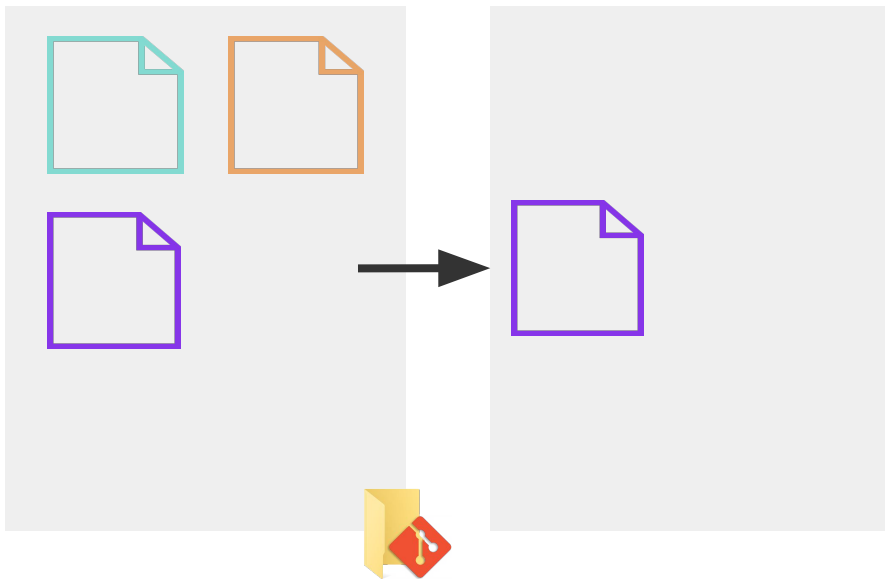


# ¿Qué acabamos de hacer?

Mi computador

Lista de cambios  
(*Staging area*)

GitHub





# ¿Qué acabamos de hacer?

Mi computador

Lista de cambios  
(*Staging area*)

GitHub

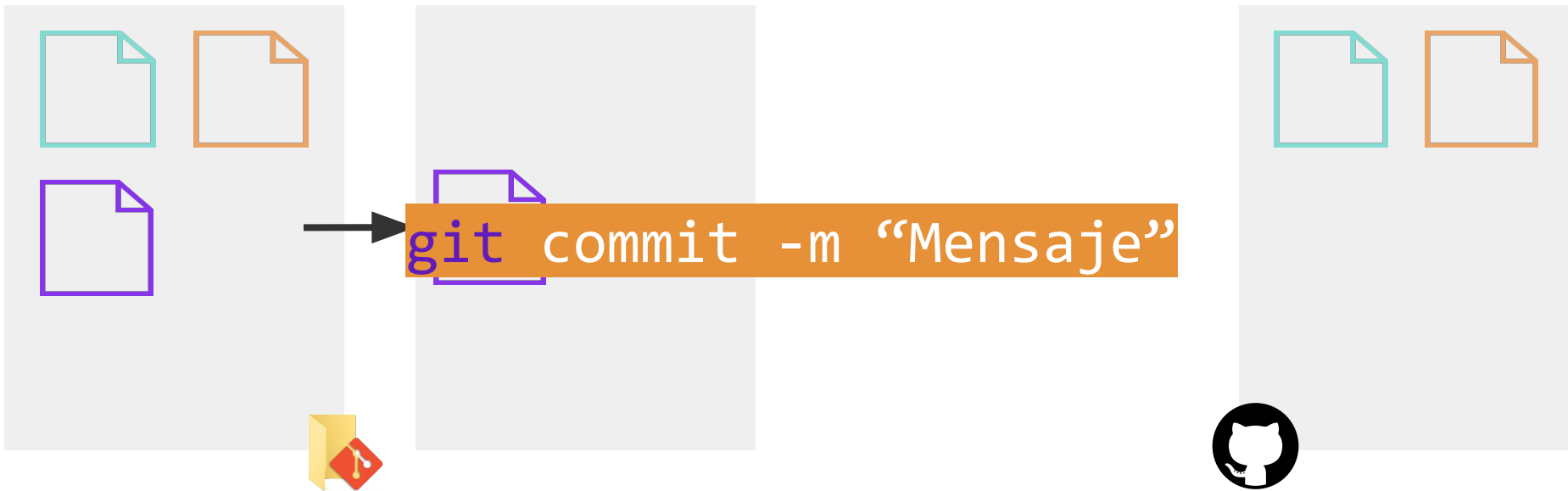


# ¿Qué acabamos de hacer?

Mi computador

Lista de cambios  
(*Staging area*)

GitHub



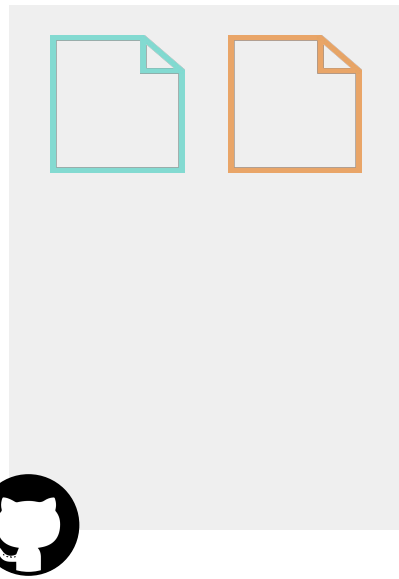
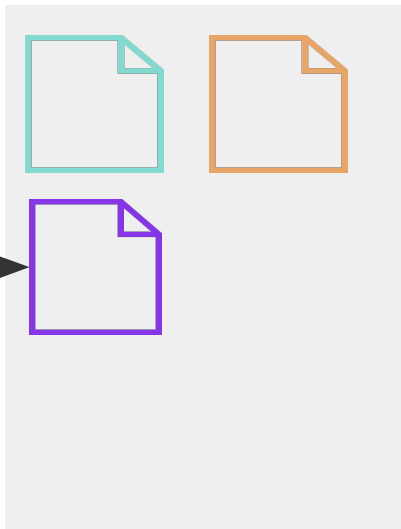
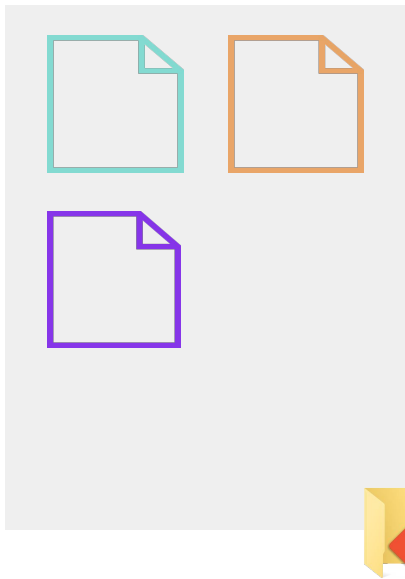
# ¿Qué acabamos de hacer?

Mi computador

Lista de cambios  
(*Staging area*)

Repositorio local

GitHub



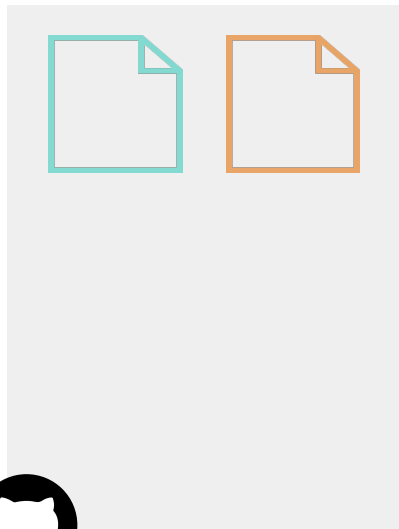
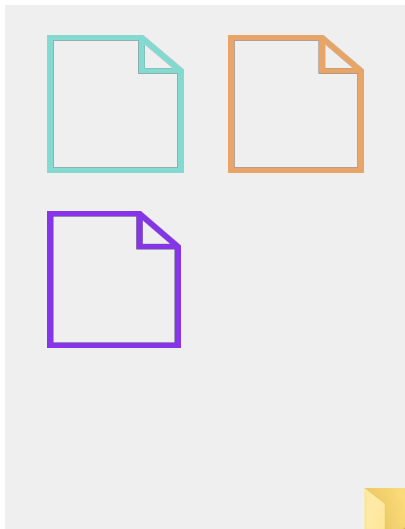
# ¿Qué acabamos de hacer?

Mi computador

Lista de cambios  
(*Staging area*)

Repositorio local

GitHub



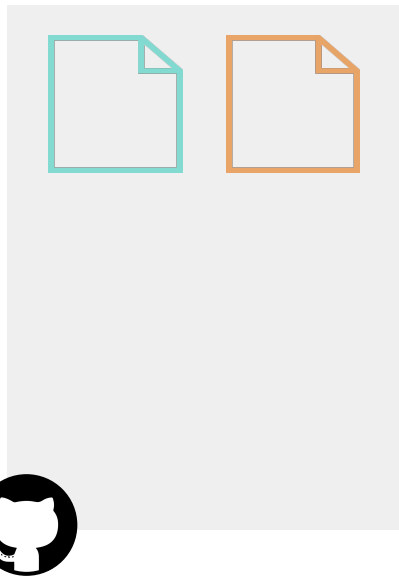
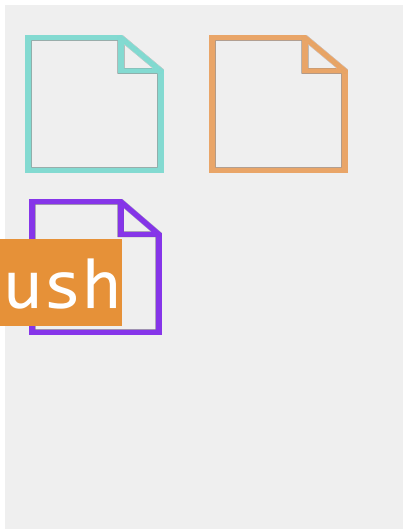
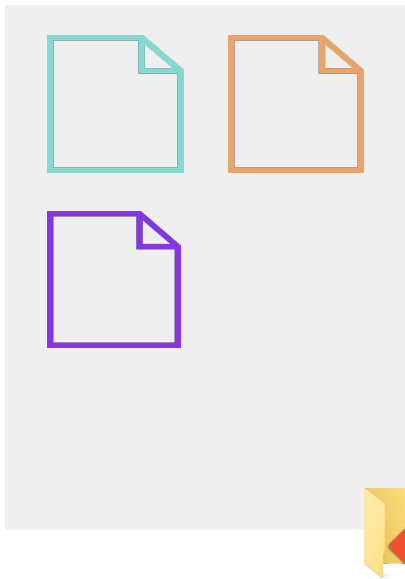
# ¿Qué acabamos de hacer?

Mi computador

Lista de cambios  
(*Staging area*)

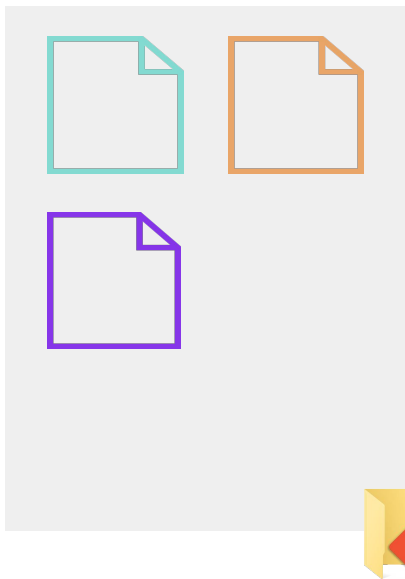
Repositorio local

GitHub

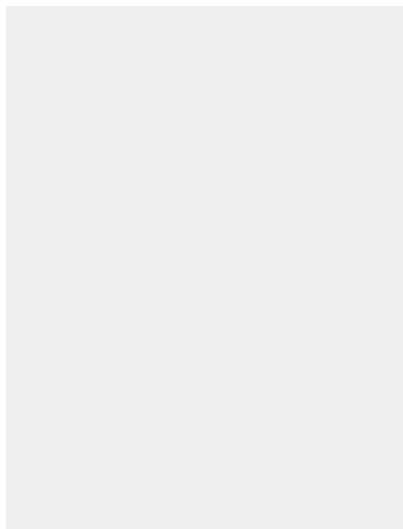


# ¿Qué acabamos de hacer?

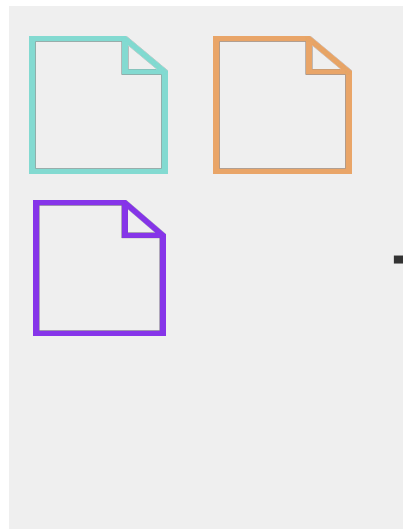
Mi computador



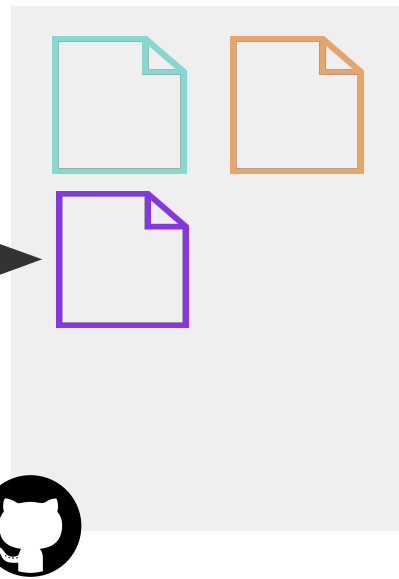
Lista de cambios  
(*Staging area*)



Repositorio local

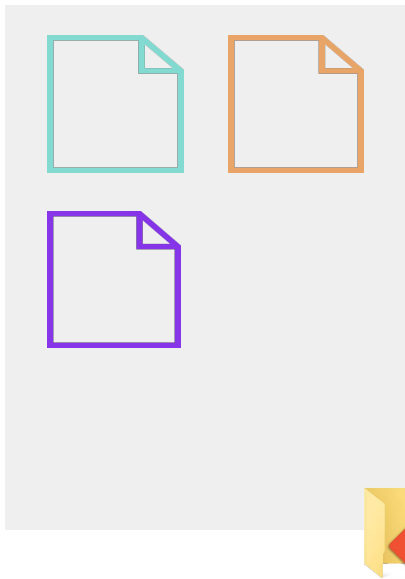


GitHub  
(Repositorio remoto)

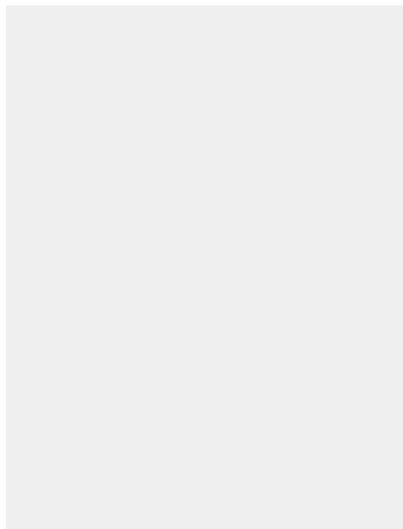


# ¿Qué acabamos de hacer?

Mi computador



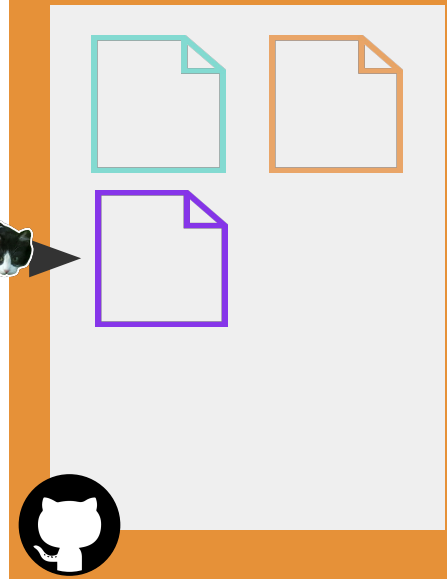
Lista de cambios  
(*Staging area*)



Repositorio local



GitHub  
(Repositorio remoto)



# El/la alumno/a que ocupa git status





**Siempre** hagan  
*commit y push*  
de su trabajo.

- Cada vez que avancen en algo importante de su actividad o tarea.
- Si llevan programando más de media hora.
- Cada vez que paren de programar para dedicarse a otra cosa.

---

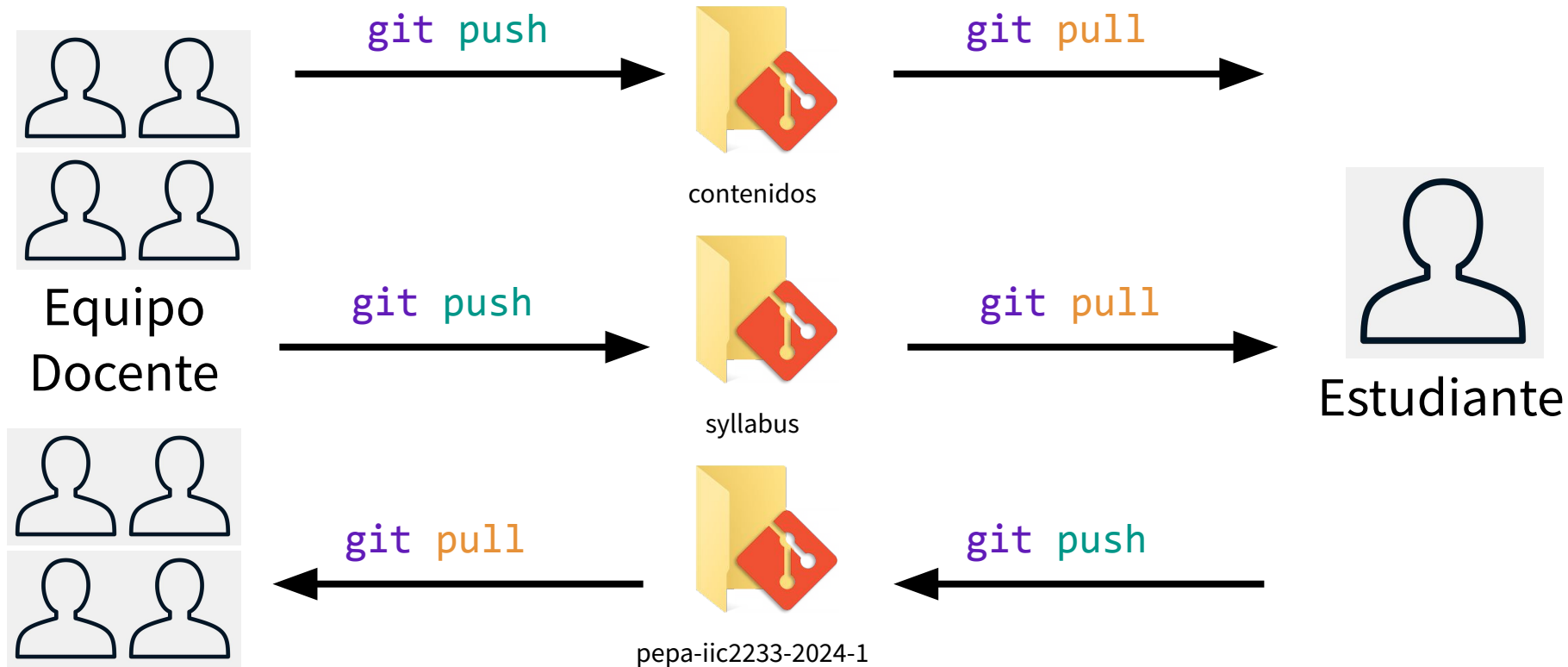
**De verdad:**  
**Siempre hagan**  
***commit y push***  
**de su trabajo.**

- Tener su trabajo en GitHub es una copia de seguridad.
  - *Shit happens:*
    - Accidentes con líquidos.
    - Robos en Deportes.
    - Fallas de *hardware* o *software*.
    - Cortes de internet.
    - Echar a perder la tarea.
    - Y muchas otras cosas.
-



**El alumno/a que aprendió  
cómo funciona git**

# El flujo durante el semestre



# ***Programación Avanzada***

## **IIC2233 2024-1**

Hernán Valdivieso - Daniela Concha - Francisca Ibarra - Dante Pinto - Francisca Cattán



# Actividad 0

# Primeros pasos de la Actividad

1. Actualizar repositorio.
  - 1.1. Si aún no clonan: `git clone https://github.com/IIC2233/syllabus.git`
  - 1.2. Si ya clonaron: `git pull`
2. Vayan a la carpeta “Actividades”, luego “AC0”, copien el contenido **a su repo personal**.
3. Desarrollen la Actividad **en su repo personal**.
4. Recuerden subir su commit al terminar:  
`git add Actividades/AC0/main.py`  
`git commit -m 'AC0 subida'`  
`git push`

# Comentarios ACo

-

NO GIT PUSH NO GAIN!