



Git - Github

Preguntas a resolver:

- Git
- Github
- Repositorio
- Ramas
- Local & Remote
- Comandos
- Versiones

¿Qué son?

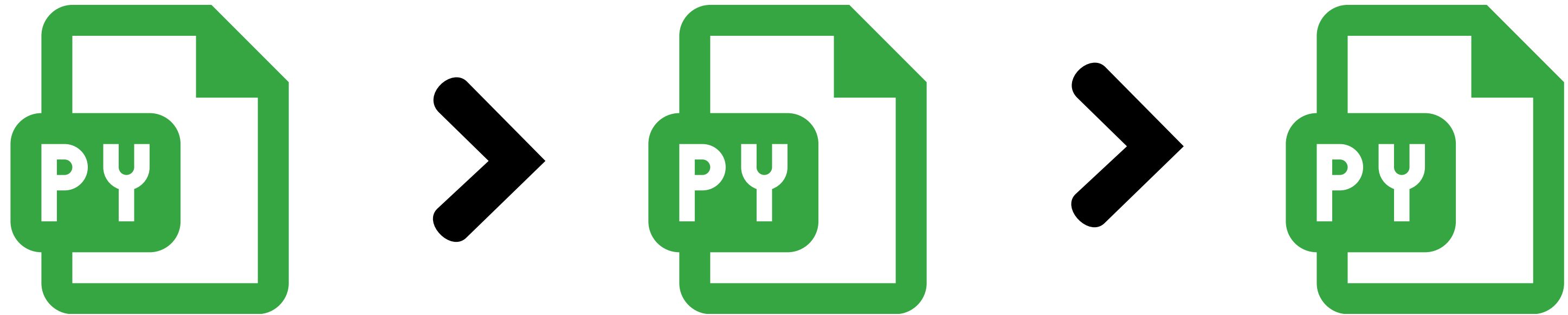
¿Para que sirven?

¿Cómo funcionan?

Imaginemos que tenemos un archivo de python



“Linea del Tiempo”



```
main.py
1
```

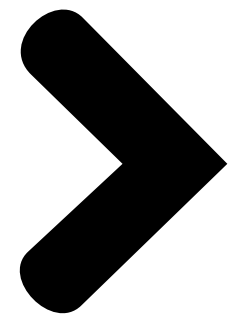
```
main.py
1 print("hola mundo")
```

```
main.py
1 print("hola mundo")
2 print("adios")
```

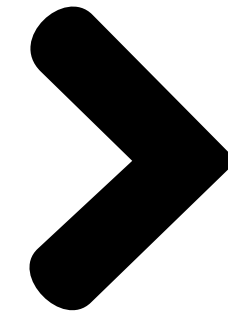
Los archivos sufren cambios a lo largo del tiempo
podríamos decir que pasan a través de estados, cada vez que
nosotros guardamos dichos cambios

“Linea del Tiempo”

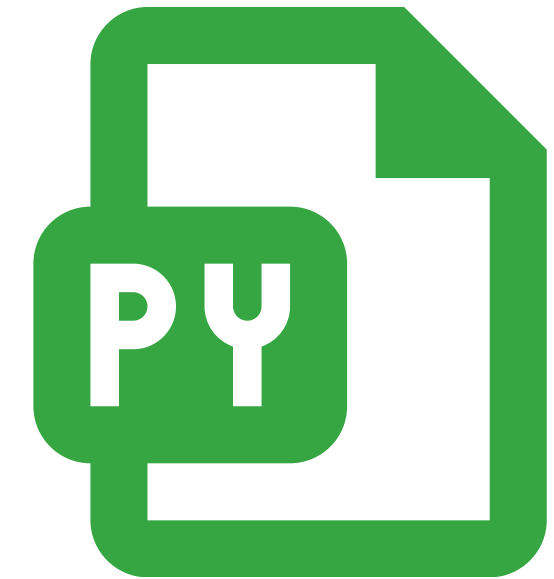
Estado 1



Estado 2



Estado 3



```
main.py
1
```

```
main.py
1 print("hola mundo")
```

```
main.py
1 print("hola mundo")
2 print("adios")
```

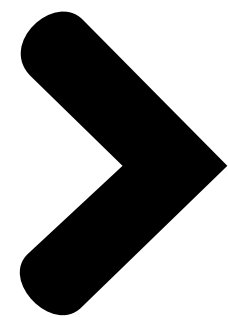
Actualmente si queremos tener varias copias de un archivo .con cambios que queramos probar, hacemos una copia y lo modificamos...

Git

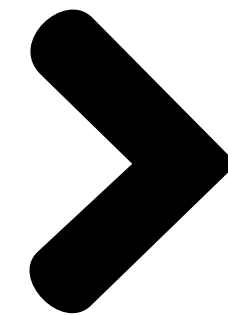
Es un sistema de control de versiones

“Linea del Tiempo”

Versión 0.0.1



Versión 0.0.2



Versión 0.0.3



```
main.py
1
```

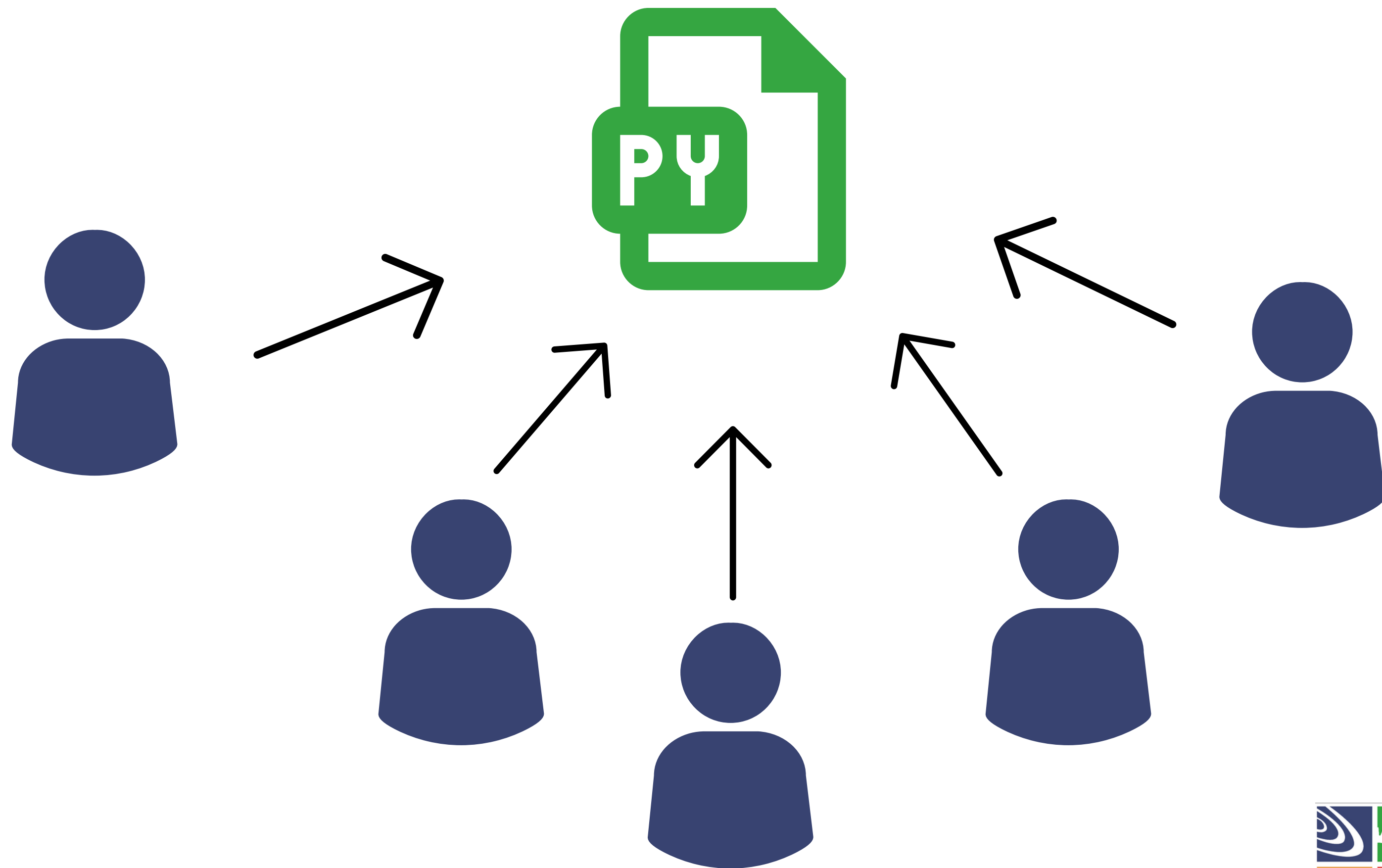
```
main.py
1 print("hola mundo")
```

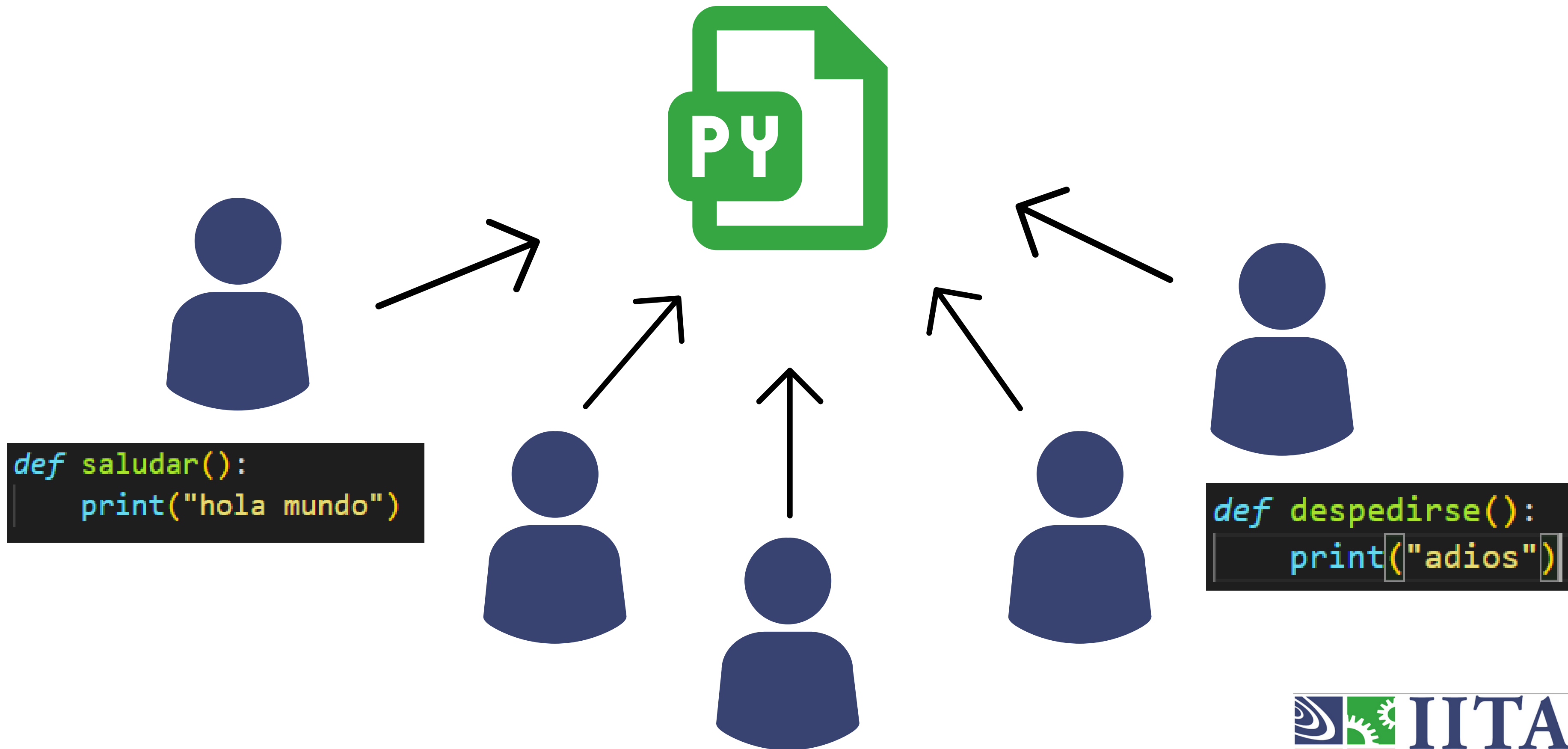
```
main.py
1 print("hola mundo")
2 print("adios")
```

Ahora tenemos una forma de llevar un “historial de cambios” para cada uno de nuestros archivos sin necesidad de tener varias copias del mismo.

**Imaginemos que quiero COMPARTIR
un archivo de python**





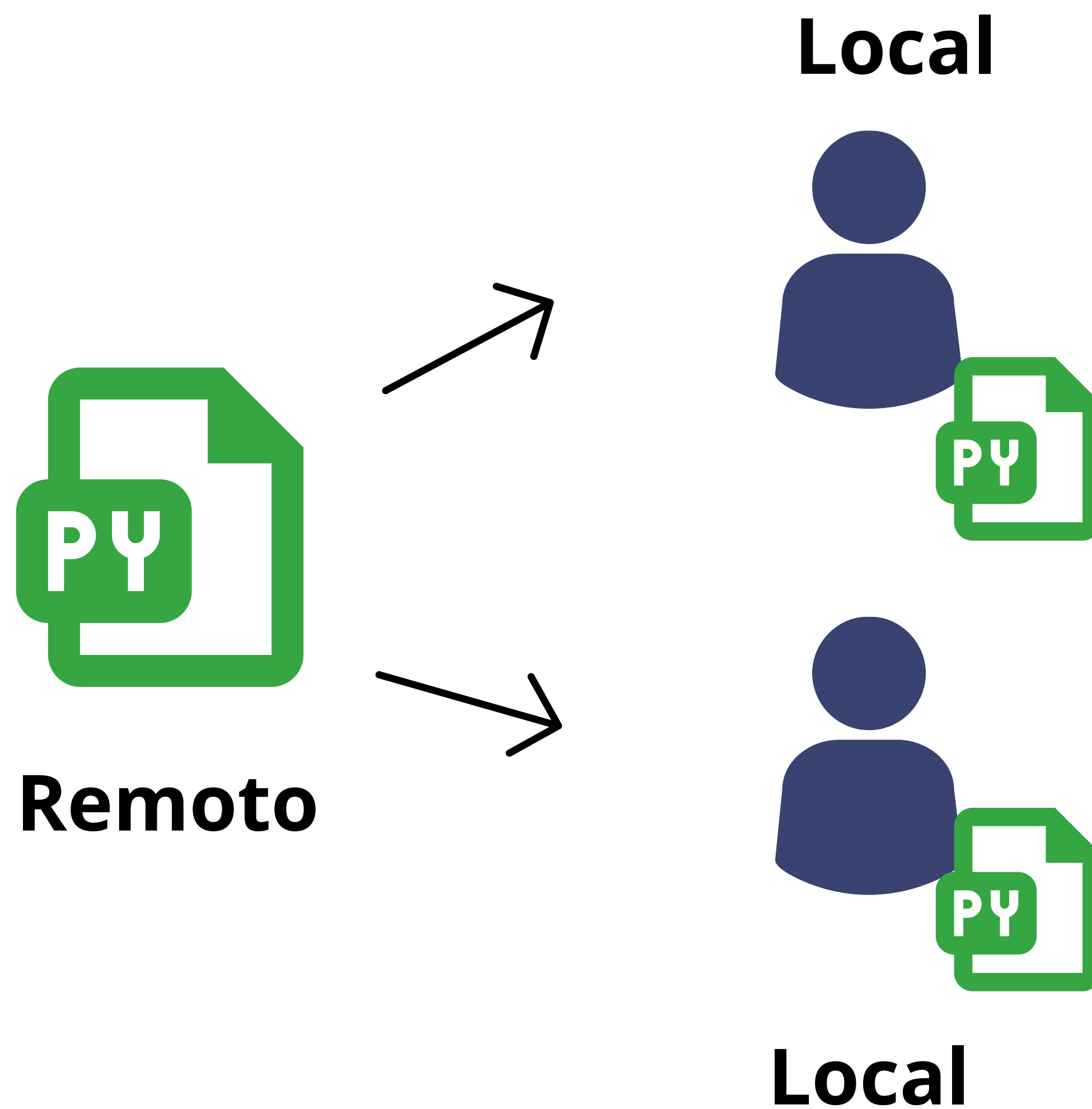


```
def saludar():  
    print("hola mundo")
```

```
def despedirse():  
    print("adios")
```

Repositorio

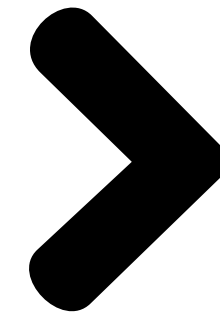
Es un espacio que sirve para compartir código u que otras personas puedan colaborar en el mismo.



Desarrollador 1



```
main.py
1 print("hola mundo")
2 print("adios")
```



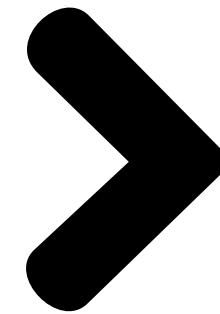
```
def saludar():
    print("hola mundo")

saludar()
print('adios')
```

Desarrollador 2



```
main.py
1 print("hola mundo")
2 print("adios")
```



```
def despedirse():
    print("adios")

print('hola mundo')
despedirse()
```



```
def saludar():  
    print("hola mundo")  
  
saludar()  
print('adios')
```

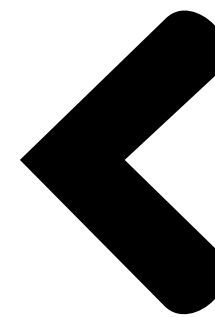


```
def despedirse():  
    print("adios")  
  
print('hola mundo')  
despedirse()
```

```
def saludar():  
    print("hola mundo")  
  
def despedirse():  
    print("adios")  
  
saludar()  
despedirse()
```

Pero ¿Qué pasa si quiero volver a mi versión?

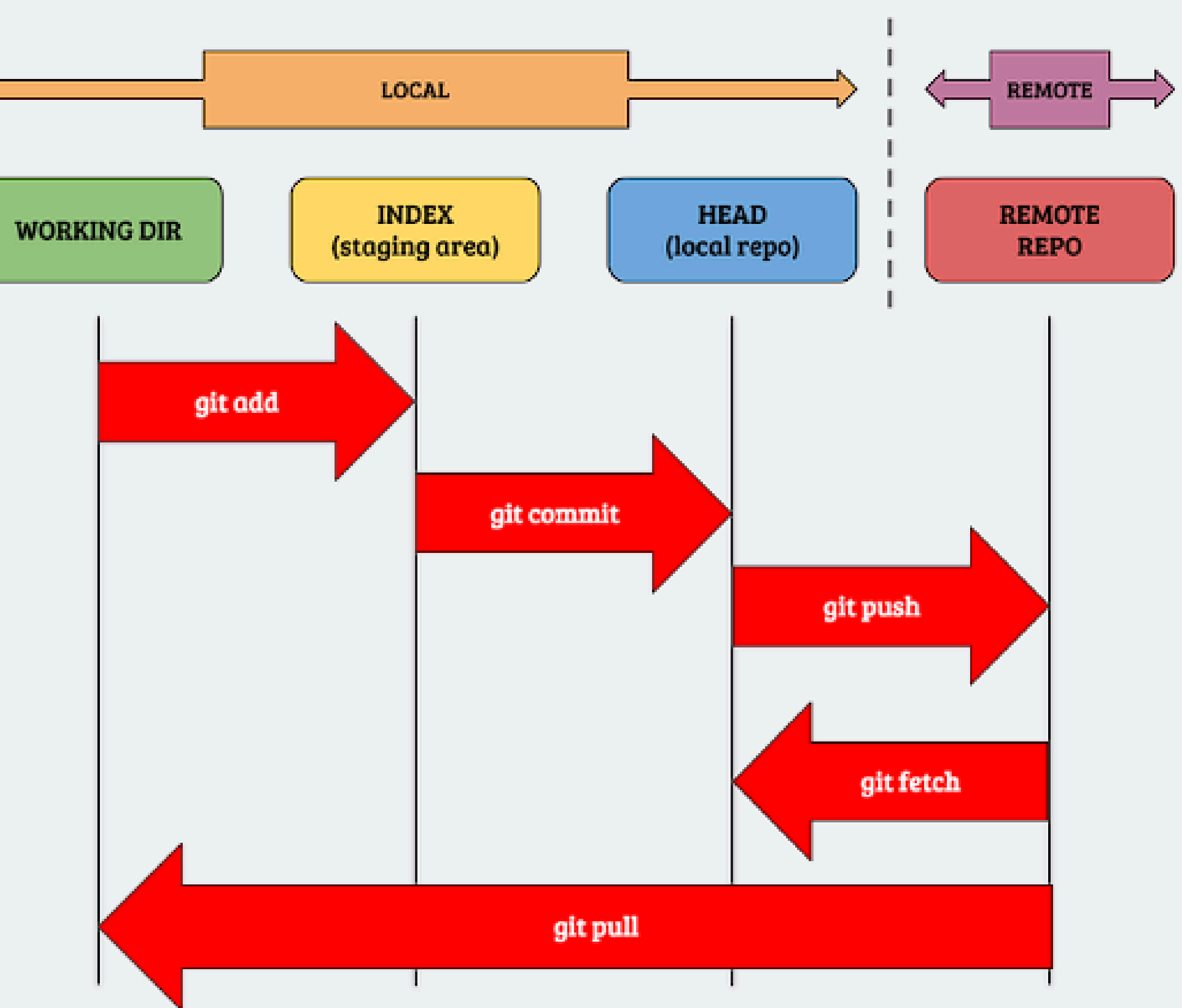
```
main.py
1 print("hola mundo")
2 print("adios")
```



```
def saludar():
    print("hola mundo")

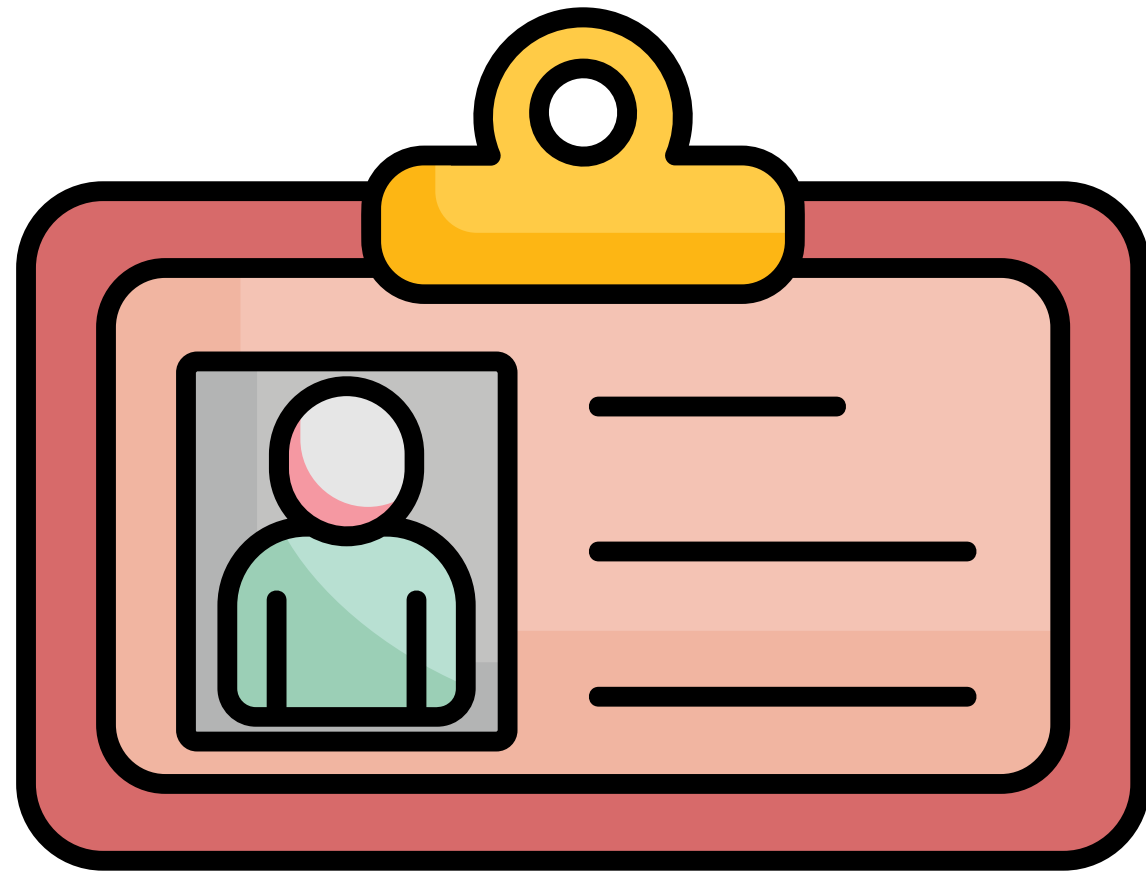
def despedirse():
    print("adios")

saludar()
despedirse()
```

Comandos Básicos

Configuración

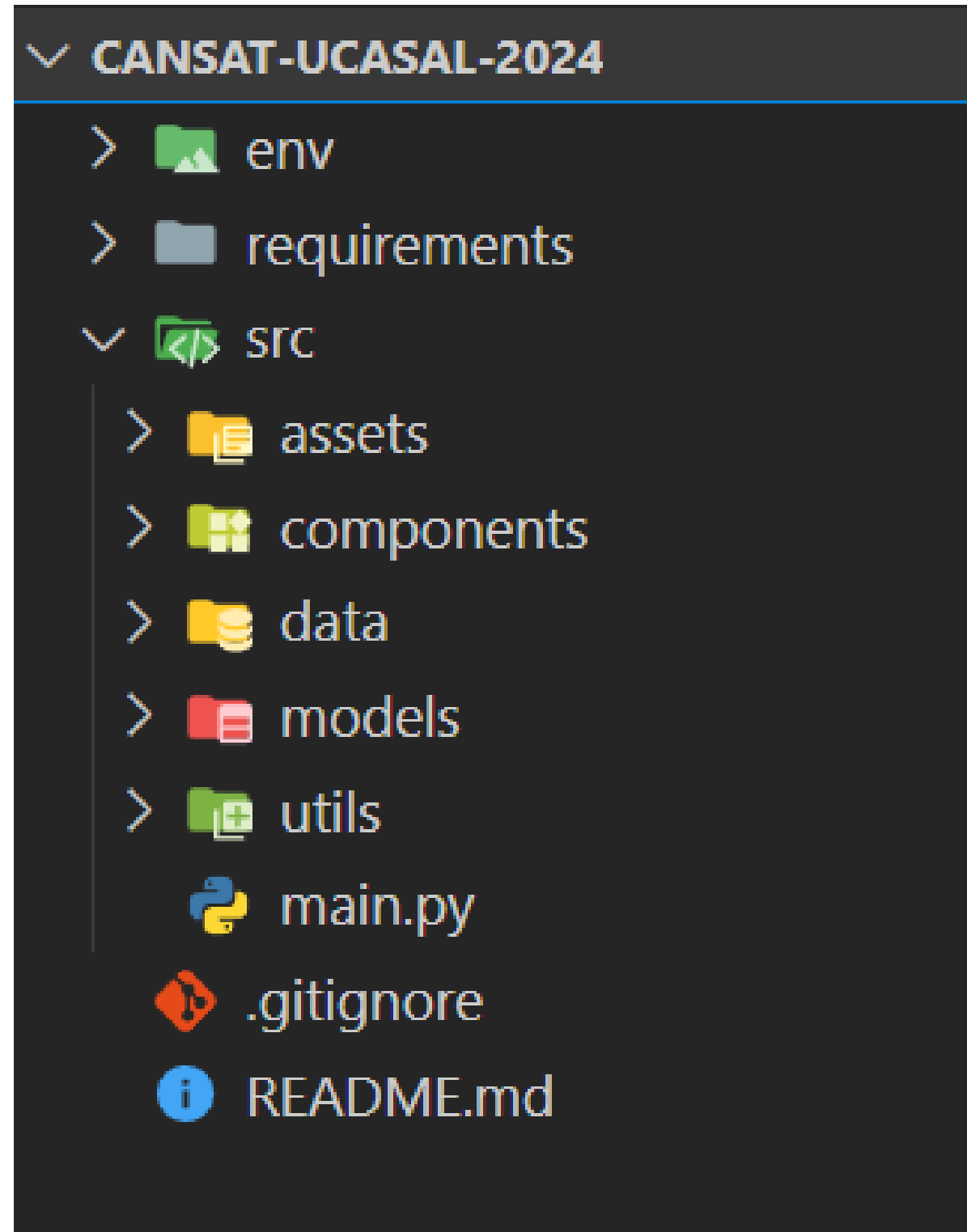


Antes que nada, cuando ya se tiene instalado **git**, se tiene que ejecutar los siguientes comandos para “**vincular**” una cuenta de nuestro espacio en remoto y nuestro dispositivo local:

```
git config --global user.name  
"github_username"
```

```
git config --global user.email "email_address"
```

Workspace



Para inicializar un repositorio en local se usa el comando **git init**

Iniciamos un repositorio vacío, es decir, que tenemos un “historial en blanco” de nuestros cambios en nuestro proyecto, a partir de eso podremos ir trabajando...

Staging Area



Con **git add** guardamos nuestros cambios de manera **provisional**, es el equivalente a ir juntando que cambios potencialmente serán guardados de manera definitiva

Git add

Añadir todos los archivos

```
git add .
```

Añadir un archivo concreto

```
git add [filename]
```

Añadir todos los archivos omitiendo los nuevos

```
git add --all
```

Añadir todos los archivos con una extensión específica

```
git add *.txt
```

Añadir todos los archivos dentro de un directorio

```
git add docs/
```

Añadir todos los archivos dentro de un directorio y con una extensión específica

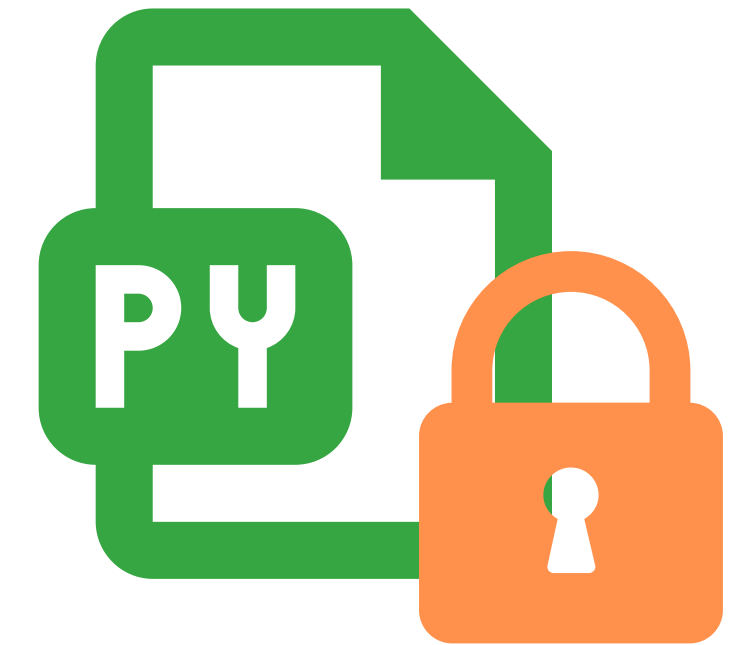
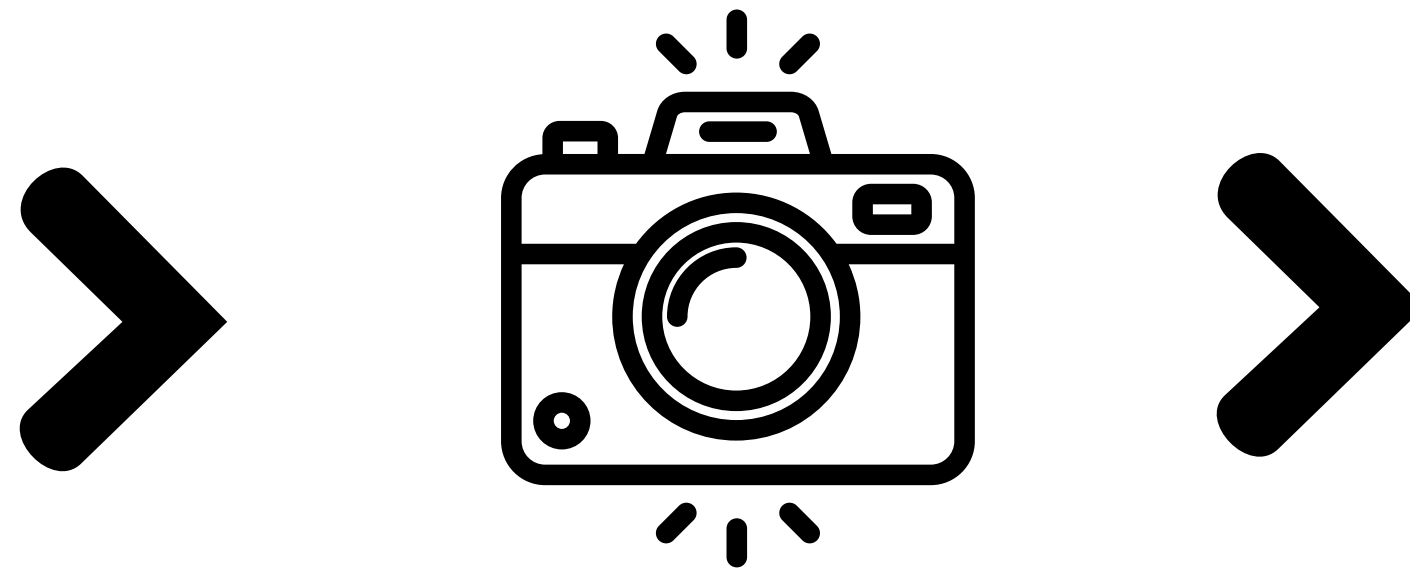
```
git add docs/*.txt
```

Local Repository (Head)

Cambio 1

Cambio 2

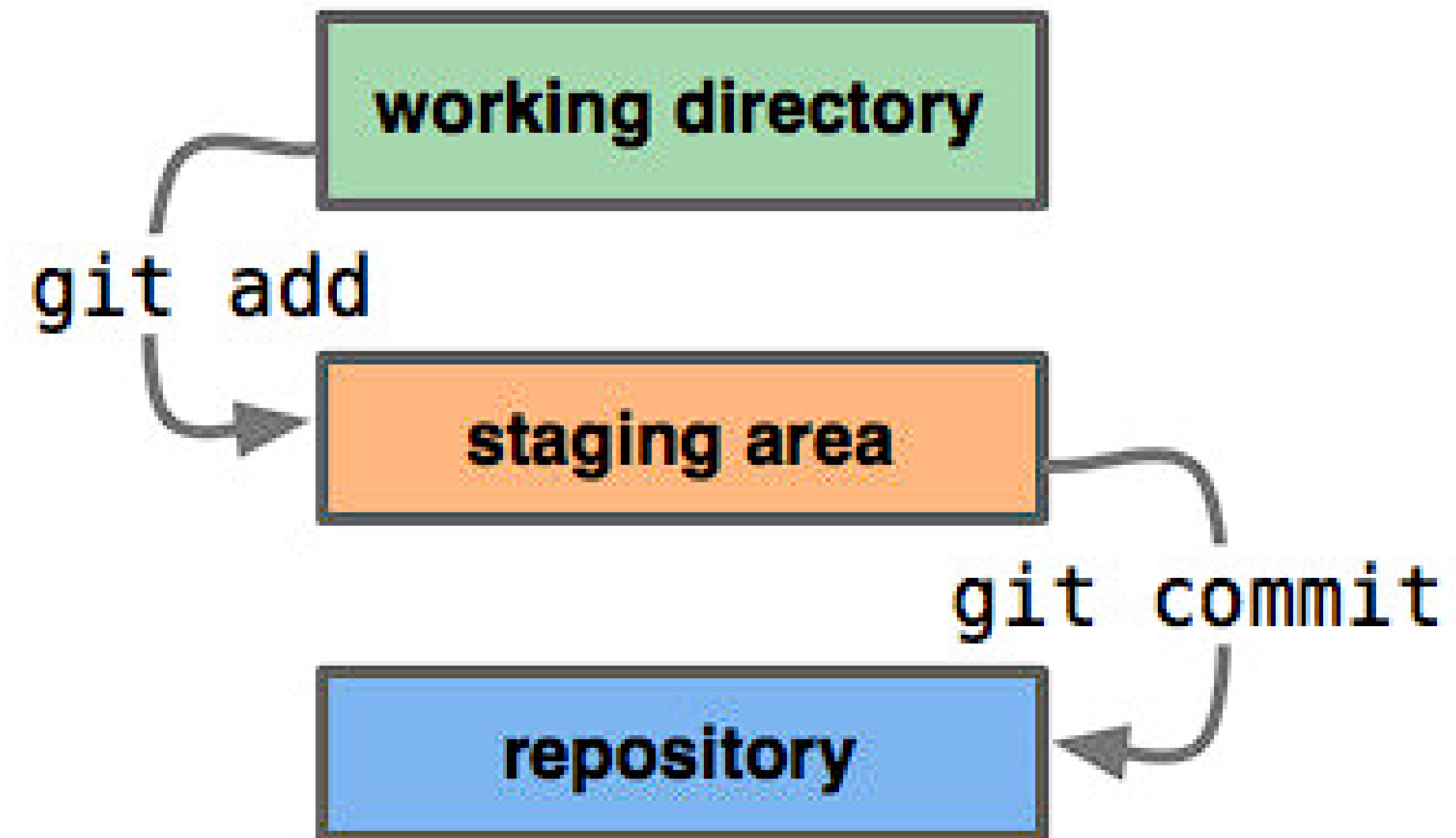
Cambio 3



git commit -m 'cambios x'

Con **git commit** guardamos nuestros cambios de manera **definitiva**. Ahora tenemos una nueva versión de nuestro proyecto identificada con el mensaje que hayamos puesto

En resumen



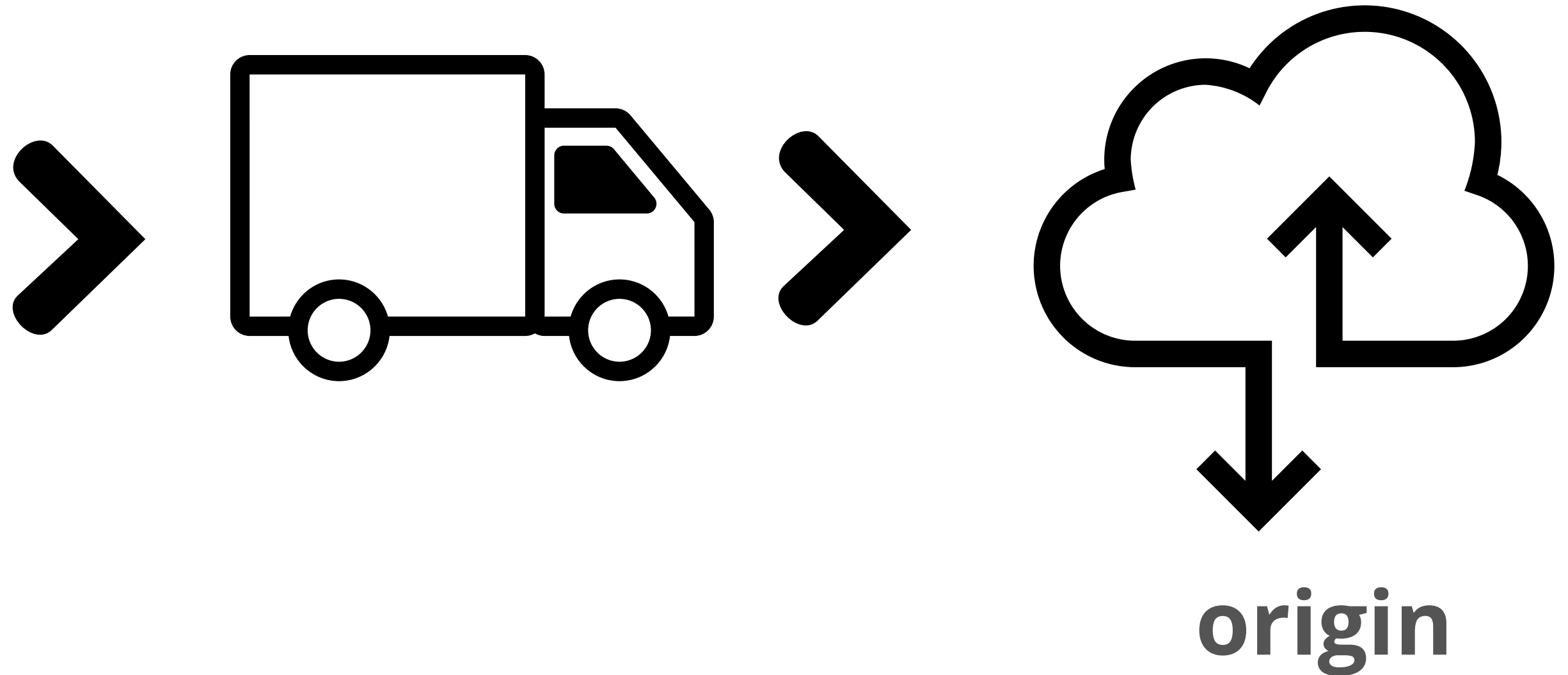
Github

Es una plataforma de desarrollo colaborativo para alojar proyectos (en la nube) utilizando un sistema de control de versiones Git.

```
git remote add origin URLRepo
```


git push

Commit
Commit
Commit



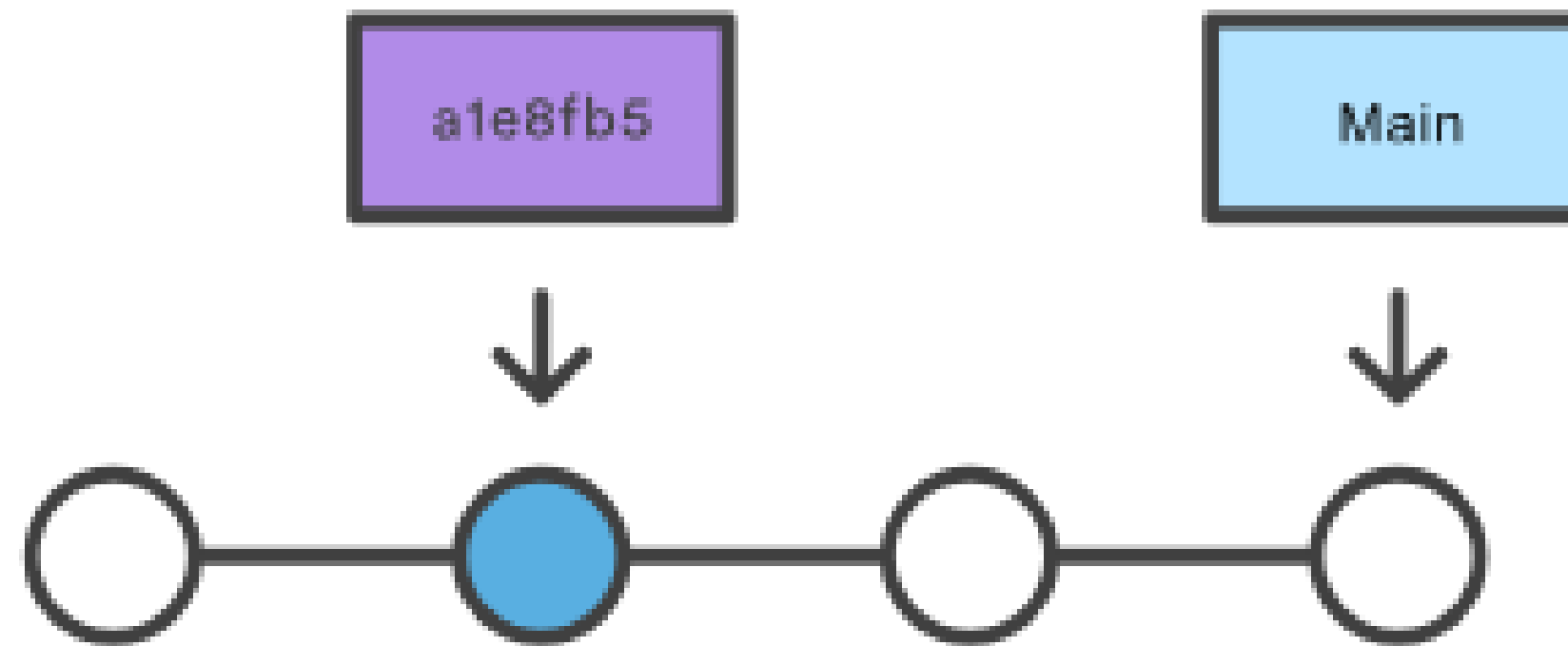
Con **git push** enviamos nuestro cambios a la nube o espacio remoto (llamado origin). SOLO toma cambios guardados de manera definitiva (commits)

git pull /fetch



Con **git pull** traemos y sumamos los cambios que se hayan hecho en el proyecto por parte de terceros a nuestra versión del proyecto en local. **git fetch**, en cambio, solo trae los cambios pero no modifica nuestro proyecto en local.

Volver hacia atras



para volver hacia atras, primero tenemos que decidir que tan atras queremos volver (usando nuestro historial de commits o cambios definitivos). Despues usamos el comando **git revert [el id del commit]** y por ultimo commiteamos este revert (para que quede registrado como hubo “rollback”)


Volver hacia atras

```
Ⓢ sandokan@pop-os:/mnt/sandokan_home/workshop/prueba-git-octubre-2024$ git log
commit c557d08e0bc78d2b16b9674c8d26e3cc69b54947 (HEAD -> master, origin/master)
Author: SPablo2191 <pablosandoval2191@gmail.com>
Date: Sat Nov 9 11:28:41 2024 -0300

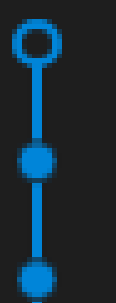

    feat: prueba de checkout

commit 57c46a99cc7812696eb728f7b6813dbb990d2844
Author: SPablo2191 <pablosandoval2191@gmail.com>
Date: Tue Oct 22 21:23:27 2024 -0300

    commit de prueba
```



```
Ⓢ sandokan@pop-os:/mnt/sandokan_home/workshop/prueba-git-octubre-2024$ git revert c557d08e0bc78d2b16b9674c8d26e3cc69b54947
[master c20c53b] Revert "feat: prueba de checkout"
1 file changed, 2 insertions(+), 15 deletions(-)
rewrite main.py (100%)
Ⓢ sandokan@pop-os:/mnt/sandokan_home/workshop/prueba-git-octubre-2024$
```



```
○ master origin Revert "feat: prueba de checkout"
feat: prueba de checkout
commit de prueba
```

Comandos

- **Clonar un repositorio:**
 - `git clone URLRepo`
- **Actualizar repositorio LOCAL**
 - `git pull origin master / git fetch origin master`
- **Actualizar repositorio REMOTO**
 - `git push origin master`

**A instalar y
Trabajar!**

Instalacion:

Git:

- <https://git-scm.com/download/win> . Pueden seguir esta guía:
<https://phoenixnap.com/kb/how-to-install-git-windows>

Fuentes:

- [Te lo explico con gatitos](#)
- [freecodecamp](#)
- [comandos basicos](#)