


INICIAR LA APLICACIÓN DE GIT EN LA CARPETA PRINCIPAL DEL PROYECTO A TRABAJAR

LOS COMANDO BASICOS QUE SE USARAN DE FORMA RECURRENTES SON:

- GIT INIT (SE USA SOLO UNA VEZ, NORMALMENTE ES SOLO CUANDO SE QUIERE SUBIR DOCUMENTOS A UN REPOSITORIO REMOTO; LUEGO DE ESO SE PUEDE OMITIR)
- GIT CLONE (PARA PODER DESCARGAR EL REPOSITORIO REMOTO DE MANERA RAPIDA Y FACIL)

bash


 Copy code

```
git clone <url_del_repositorio> [<nombre_de_la_carpeta_destino>]
```

- `<url_del_repositorio>` es la URL del repositorio remoto que quieres clonar.
- `<nombre_de_la_carpeta_destino>` es opcional y especifica el nombre de la carpeta en la que se clonará el repositorio. Si no se especifica, se creará una carpeta con el mismo nombre que el repositorio.

Por ejemplo, si quieres clonar el repositorio "mi-proyecto" desde GitHub y quieres que la carpeta resultante se llame "mi-proyecto", puedes hacerlo así:


bash

 Copy code

```
git clone https://github.com/usuario/mi-proyecto.git
```

Si deseas que la carpeta se llame de manera diferente, puedes especificarlo después de la URL del repositorio:

bash

 Copy code


```
git clone https://github.com/usuario/mi-proyecto.git otra-carpeta
```

- GIT BASH (SIRVE PARA VER LA RAMA EN LA QUE SE ESTA TRABAJANDO). CABE RECALCAR QUE SE PUEDE CREAR NUEVAS RAMAS PARA TRABAJAR Y REALIZAR CAMBIOS SIN PREOCUPARSE DE LA RAMA PRINCIPAL (MAIN) DE LA SIGUIENTE MANERA

1. Crear una nueva rama:

Para crear una nueva rama, puedes usar el comando `git branch` seguido del nombre de la nueva rama. Por ejemplo, si quieres llamar a tu nueva rama "nueva-funcionalidad", ejecutarías:


bash

 Copy code

```
git branch nueva-funcionalidad
```

O puedes hacerlo y cambiar inmediatamente a esa rama utilizando `-b`:

bash

 Copy code

```
git checkout -b nueva-funcionalidad
```

Y PARA PODER JUNTAR O COMBINAR LOS CAMBIOS REALIZADOS DE LA RAMA NUEVA A LA PRINCIPAL SE REALIZA DE LA SIGUEINTE MANERA.

Una vez que hayas terminado tu trabajo en la nueva rama y quieras agregar esos cambios a la rama principal, primero asegúrate de estar en la rama principal:

```
bash
```

[Copy code](#)

```
git checkout master
```

Luego, fusiona los cambios desde tu nueva rama:

```
bash
```

[Copy code](#)

```
git merge nueva-funcionalidad
```

- GIT ADD (SE USA PARA AGREGAR LOS ARCHIVOS QUE QUERAMOS CARGAR AL REPOSITORIO REMOTO, SI TENEMOS VARIOS ARCHIVOS CON CAMBIOS QUE QUEREMOS SUBIR PODEMOS HACERLO DE LA SIGUIENTE MANERA)

Por lo tanto, si deseas agregar todos los archivos ubicados en la carpeta "carpetal" para que sean incluidos en el próximo commit, puedes hacer lo siguiente:

```
bash
```

[Copy code](#)

```
git add carpetal/.
```

- GIT COMMIT (SE UTILIZA PARA CONFIRMAR LOS CAMBIOS REALIZADOS A LOS ARCHIVOS QUE HEMOS AGREGADO CON 'GIT ADD')

```
bash
```


[Copy code](#)

```
git commit -m "Agregando archivos de la carpetal"
```

- GIT PUSH (SE USA PARA SUBIR LOS ARCHIVOS DEL REPOSITORIO LOCAL AL REPOSITORIO REMOTO)

- Si estás en la rama principal de tu repositorio local y la rama principal en tu repositorio remoto también se llama "main" (o "master"), puedes simplemente ejecutar:

bash


 Copy code

```
git push
```

Esto enviará los cambios de tu rama local "main" al repositorio remoto en la rama "main".

- Si tu rama principal en el repositorio remoto tiene un nombre diferente, digamos "master", entonces deberías especificar la rama remota y local explícitamente:

bash


 Copy code

```
git push origin main:master
```

Esto enviará los cambios de tu rama local "main" al repositorio remoto, pero los colocará en la rama "master" del repositorio remoto.

- Si estás trabajando en una rama diferente a la principal y deseas enviar tus cambios a una rama específica en el repositorio remoto, puedes hacerlo así:

bash

 Copy code

```
git push origin nombre_de_tu_rama_local:nombre_de_tu_rama_remota
```

Donde `nombre_de_tu_rama_local` es el nombre de tu rama local y

`nombre_de_tu_rama_remota` es el nombre de la rama correspondiente en el repositorio remoto.

- GIT PULL (SE USA PARA PODER ACTUALIZAR Y DESCARGAR LOS CAMBIOS SUBIDOS POR LOS DEMAS INTEGRANTES DE UN REPOSITORIO)

CADA USUARIO O INTEGRANTE DEL REPOSITORIO ESTARA TRABAJANDO EL REPOSITORIO REMOTO DE FORMA LOCAL, POR LO QUE LOS COMMIT QUE SE HAGAN EN CADA MAQUINA SERAN PERSONALES HASTA QUE SE SUBAN AL REPOSITORIO REMOTO CON PUSH. SI SE DESEA REGRESAR A UN COMMIT ANTERIOR PARA SEGUIR TRABAJANDO DESDE UNA VERSION ANTERIOR A LOS CAMBIOS QUE SE HAN HECHOS SE PUEDE REALIZAR DE LA SIGUIENTE MANERA:

- GIT LOG (PARA VER EL HISTORIAL DE COMMIT'S QUE SE HAN REALIZADO EN EL REPOSITORIO REMOTO Y LOCAL)

```
sql Copy code

commit 5f3a1e2d6e50a4e4a95b7b83161f0b29c1eef3e3
Author: Autor <email@ejemplo.com>
Date:   Mié Abr 6 18:02:21 2024 +0200

    Mensaje del commit anterior


commit 2fd67b4e9d9911d3b53a4f5b80ee5a52f2b4fb10 (HEAD -> rama_actual)
Author: Autor <email@ejemplo.com>
Date:   Mié Abr 6 17:58:11 2024 +0200

    Mensaje del commit actual
```

- GIT CHECKOUT (PARA VOLVER O RECUPERAR UN COMMIT ANTERIOR)

Por ejemplo, si el hash del commit es `5f3a1e2d6e50a4e4a95b7b83161f0b29c1eef3e3`, puedes usar solo los primeros caracteres para identificarlo:


bash

 Copy code

```
git checkout 5f3a1e2
```

Sin embargo, para mayor comodidad, Git permite que uses cualquier prefijo del hash del commit que sea lo suficientemente único para identificarlo. Entonces, en realidad, podrías escribir:

bash

 Copy code


```
git checkout 5f3a1e2d
```

O incluso menos caracteres si ese prefijo es lo suficientemente único para identificar el commit.

Si prefieres no usar el hash del commit, también puedes usar referencias relativas, como `HEAD~n`, donde `n` es el número de commits atrás desde el commit actual, o referencias simbólicas como `HEAD`, `HEAD^`, `HEAD~1`, etc.

Por ejemplo:

bash

 Copy code

```
git checkout HEAD~1
```

Esto te llevará al commit anterior al commit actual.