



https://



G-Mail



LinkedIn



Linktree



MI Web

UTILIDADES GIT

COMANDOS GIT

GIT CLONE, ADD, COMMIT, PUSH, PULL, ETC ...

clone	git clone	Clonar un Repositorio
Por Ssh	<code>git clone git@github.com:Gerardo-Mtnz-Dev/my_linux.git</code>	
Por Https	<code>git clone https://github.com/Gerardo-Mtnz-Dev/my_linux.git</code>	
Así los clona en directorio <code>/my_linux</code> Para clonar el Repo con otro nombre de Dir, añadir <code>"FolderName"</code>		
	<code>git clone git@github.com:Gerardo-Mtnz-Dev/my_linux.git gerar</code>	<code>~/gerar/my_lynux</code>

COMANDO	ACCION
<code>git add .</code>	Preparar para añadir Todo
<code>Git commit -m "xxxx"</code>	

IGNORAR ARCHIVOS (.gitignore)

REGLAS / PATRONES DEL ARCHIVO

#	al principio	Ignorar las líneas en blanco y aquellas que comiencen con #.
*		Todo lo que..... (* corresponde a cero o más caracteres)
*.txt		cualquier archivo de tipo txt
*.[oa]		cualquier archivo que termine en ".o" o ".a"



<https://>[G-Mail](#)[LinkedIn](#)[Linktree](#)[Mi Web](#)

*~	todos los archivos que terminen con una tilde
/	<i>al principio</i> para evitar recursividad.
/	<i>al final</i> para especificar un directorio.
!	<i>al principio</i> El patrón se niega
[0-9]	corresponde a cualquier caracter entre ellos (en este caso del 0 al 9).
[abc]	corresponde a cualquier caracter dentro de los corchetes (en este caso a, b o c)
?	corresponde a un caracter cualquiera;
a/**/z	directorios anidados; a/**/z coincide con a/z, a/b/z, a/b/c/z, etc.



EJEMPLO DE UN ARCHIVO .GITIGNORE:

```
# ignora los archivos que genera Bash
*.SWO

# ignora los archivos que genera Vim
*.swp

# ignora los archivos que genera VS Code
*.vscode

# ignora los archivos temporales que genera Files de Linux:
*.DS_store

# Ignorar los ejecutables de compilación de C:
*.out

# Ignorar los archivos de compilación de C:
*.o

#fin

#Ejemplos

# ignora los archivos terminados en .a
*.a
# pero no lib.a, aun cuando había ignorado los archivos terminados en .a en la línea anterior
!lib.a
# ignora unicamente el archivo TODO de la raiz, no subdir/TODO
/TODO
```



<https://>[G-Mail](#)[LinkedIn](#)[Linktree](#)[MI Web](#)

```
# ignora todos los archivos del directorio build/
build/
# ignora doc/notes.txt, pero no este: doc/server/arch.txt
doc/*.txt
# ignora todos los archivos .txt del directorio doc/
d oc/**/*.txt
```

#fin



RAMAS



CREAR / ELIMINAR RAMAS

```
git branch Branch-Name
```

Crear una Rama

```
git checkout -b Branch-Name
```

Atajo para crear y cambiar de rama

```
git branch
```

ó

```
git branch --list = git branch -l
```

comprobar las ramas que tenemos creadas

```
git branch -d Branch-Name
```

Para eliminar una rama en local

```
git push origin -delete Branch-Name
```

ó

```
git push origin :Branch-Name
```

Para eliminar una rama en remoto

```
git checkout Branch-Name
```

cambiar de rama

```
git push -u origin Branch-Name
```

Subir una rama al repositorio

```
git pull Branch-Name
```

Bajarse a local una rama

```
git diff [first-branch]...[second-branch]
```

Muestra diferencias de contenido entre dos ramas

```
git pull -rebase
```

ó las 3 órdenes

```
git config pull.rebase true
```

Forzar pull en ramas

```
git pull
```

```
git config pull.rebase false
```





https://



G-Mail



LinkedIn



Linktree



Mi Web



FUSIONAR RAMAS

1º `git status`

Asegurarse de que no haya cambios pendientes en repositorio local o en área preparación.

2º `git checkout main`

Asegurarse de estar en la rama Main (ó master). Solo se puede fusionar desde esta rama

3º `git merge Branch-Name`

Fusionar la rama creada con la rama main

4º `git branch -d Branch-Name`

Eliminar una rama en local

`git push origin -- delete Branch-Name`5º **ó**

Eliminar una rama en remoto

`git push origin :Branch-Name`

CONFLICTOS AL FUSIONAR RAMAS

CONFLICTOS

Cuando fusionamos dos ramas (o fusiona una rama local y una remota), a veces puede surgir un conflicto. Por ejemplo, dos desarrolladores, sin saberlo, trabajan en la misma parte de un archivo. Uno de ellos envía sus cambios al repositorio remoto de Github. Cuando los lleve a su repositorio local, obtendrá un conflicto de fusión.

Git tiene una forma de manejar los conflictos, por lo que puede ver ambos conjuntos de cambios y decidir cuál desea conservar.

- 1· Cuando tenemos un conflicto de fusión, toma nota de los archivos que tienen un conflicto.
- 2· En tu editor de código, abra un archivo en conflicto y busque estos marcadores de conflicto:

<<<<<< HEAD

Marca el inicio de los cambios.

=====

Divide sus cambios de los cambios en la otra rama.

>>>>>> branch-name

Marca el final de los cambios.

3· Después de editar el archivo, podemos usar el comando `git add` para preparar el nuevo contenido fusionado

4· El paso final es crear una nueva confirmación con la ayuda del comando `git commit`. Veamos ahora los comandos de Git que pueden desempeñar un papel importante en la resolución de conflictos.

COMANDO

ACCION

`git log --merge`

Producir la lista de confirmaciones que están causando el conflicto

`git diff`

Identificar las diferencias entre los repositorios o archivos



<https://>

G-Mail



LinkedIn



Linktree



Mi Web

`git reset --mixed`

Deshacer los cambios en el directorio de trabajo y el área de preparación

`git merge --abort`

Salir del proceso de fusión y volver al estado anterior a que comenzara la fusión

`git reset`

Restablecer los archivos en conflicto a su estado original



CLAVE SSH



GENERAR UNA NUEVA CLAVE SSH

ssh-keygen

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/usuario/.ssh/id_rsa):
```

Presionamos ENTER y nos guardara las claves en el directorio `.ssh/` dentro de la carpeta home de nuestro usuario.

Si previamente ya teniamos generadas unas claves veremos lo siguiente:

```
/home/usuario/.ssh/id_rsa already exists.
Overwrite (y/n)?
```

Si sobrescribimos las claves no podremos usar las que ya teniamos generadas. Despues deberiamos ver el siguiente mensaje:

```
Enter passphrase (enter for no passphrase):
```

Si tenemos una passphrase añadiremos una capa mas de seguridad para evitar accesos no autorizados. Despues de añadir la passphrase veremos algo similar a esto:

```
Your public key has been saved in /home/usuario/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:qW4y5xagI+gZ50zFNCWnptGr+ETSmcoxfafEslyJVwk {user}@{user}
The key's randomart image is:
+---[RSA 3072]-----+
|  . E                                     |
|  . = . .                               |
| o.*   o                               |
|=oB+= o .                               |
|*BBB=B .S                               |
|*=O*Boo.                               |
|o*.oo..                               |
| o .                                   |
| .                                     |
+----[SHA256]-----+
```

Ahora si vamos al directorio



