

## Pasos Básico Git

Creado por Mario Larenas



### ¿Qué es Git?

Git es, con diferencia, el sistema de control de versiones moderno más utilizado del mundo. Git es un proyecto de código abierto maduro y con un mantenimiento activo que desarrolló originalmente Linus Torvalds, el famoso creador del kernel del sistema operativo Linux, en 2005. Un asombroso número de proyectos de software dependen de Git para el control de versiones, incluidos proyectos comerciales y de código abierto.

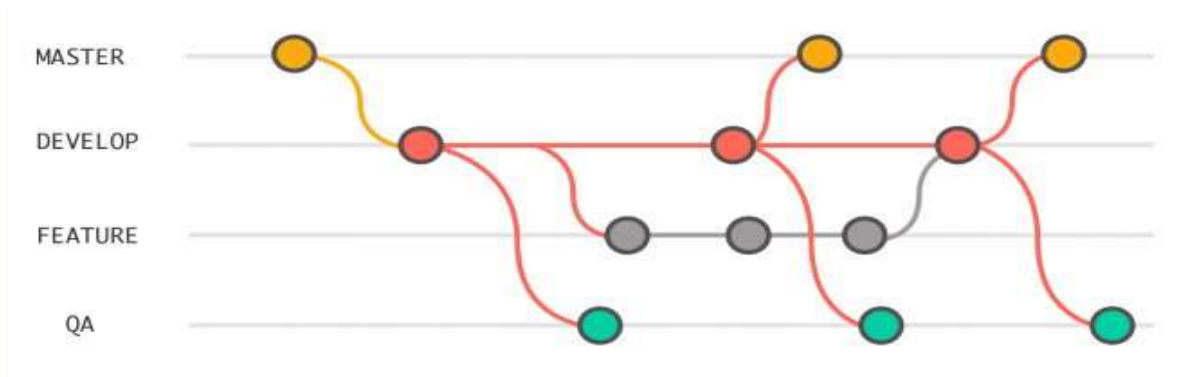
### Ramificación y fusión

La característica de Git que realmente lo distingue de casi todos los demás SCM es su modelo de ramificación.

Git te permite y te anima a tener múltiples sucursales locales que pueden ser completamente independientes entre sí. La creación, fusión y eliminación de esas líneas de desarrollo lleva unos segundos.

Esto significa que puede hacer cosas como:

- Cambio de contexto sin fricciones . Cree una rama para probar una idea, comprometa varias veces, vuelva al lugar desde donde se ramificó, aplique un parche, vuelva al lugar donde está experimentando y combínelo.
- Líneas de código basadas en roles . Tenga una rama que siempre contenga solo lo que va a producción, otra en la que combine el trabajo para realizar pruebas y varias más pequeñas para el trabajo diario.
- Flujo de trabajo basado en funciones . Cree nuevas ramas para cada nueva característica en la que esté trabajando para que pueda alternar sin problemas entre ellas, luego elimine cada rama cuando esa característica se fusione en su línea principal.
- Experimentación desechable . Crea una rama para experimentar, date cuenta de que no va a funcionar y simplemente bórrala, abandonando el trabajo, sin que nadie más la vea (incluso si has empujado otras ramas mientras tanto).



En particular, cuando empuja a un repositorio remoto, no tiene que empujar todas sus ramas. Puede optar por compartir solo una de sus ramas, algunas de ellas o todas. Esto tiende a liberar a las personas para que prueben nuevas ideas sin preocuparse por tener que planificar cómo y cuándo van a fusionarlas o compartirlas con otros.

#### Pasos Básico Git

1. Instalar git en nuestro equipo. Existe para la mayoría de los S.O.

<https://git-scm.com/>

<http://git-scm.com/downloads>

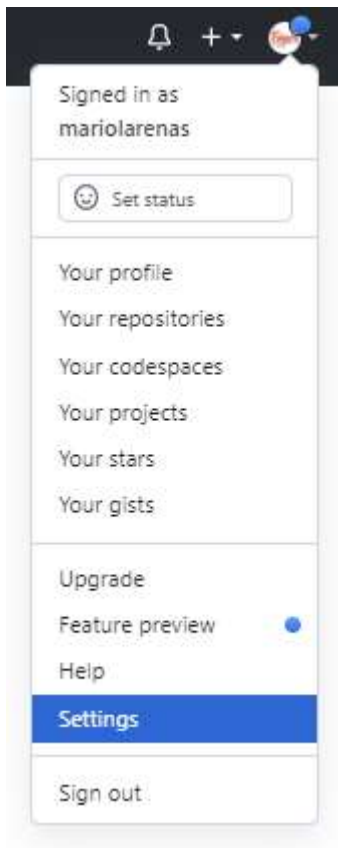
Hoy en día existen diferentes plataformas para la administración de código Git:

- a. <https://github.com/>
- b. <https://about.gitlab.com/>
- c. <https://bitbucket.org/>
- d. <https://sourceforge.net/>
- e. <https://help.launchpad.net/Code/Git>
- f. <https://help.launchpad.net/Code/Git>

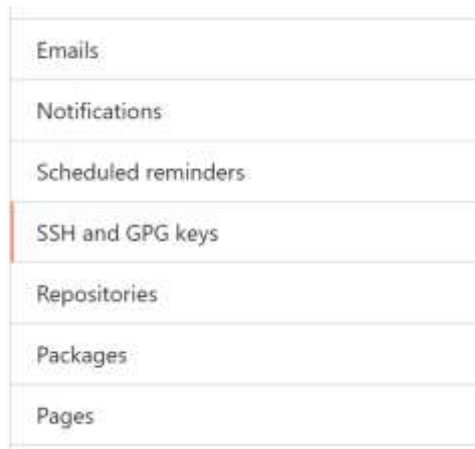
2. Ingresar a <http://gitlab.com/> crear cuenta y/o ingresar a su cuenta



Deben crear una llave SSH para ello van a configuraciones (settings)



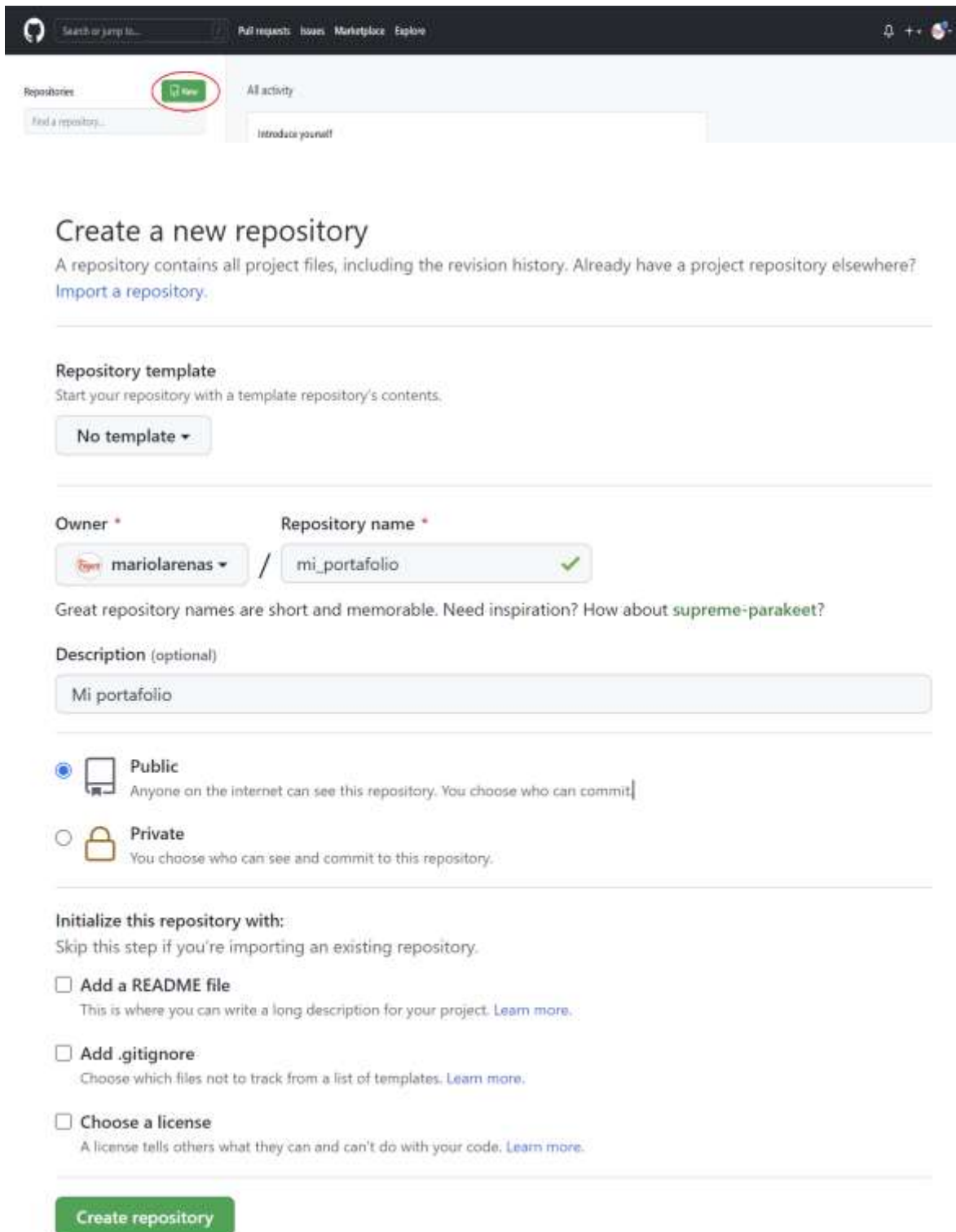
Seleccionan SSH and GPG keys



Y crean una nueva SSH key:



3. En sus repositorios crean un nuevo repositorio ejemplo: "mi\_portafolio"



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

Repositories **New** All activity

Find a repository... Introduce yourself

## Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

### Repository template

Start your repository with a template repository's contents.

No template ▾


Owner \* Repository name \*


 mariolarenas ▾ / mi\_portafolio ✓

Great repository names are short and memorable. Need inspiration? How about [supreme-parakeet?](#)

Description (optional)

Mi portafolio

☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with:**  
Skip this step if you're importing an existing repository.

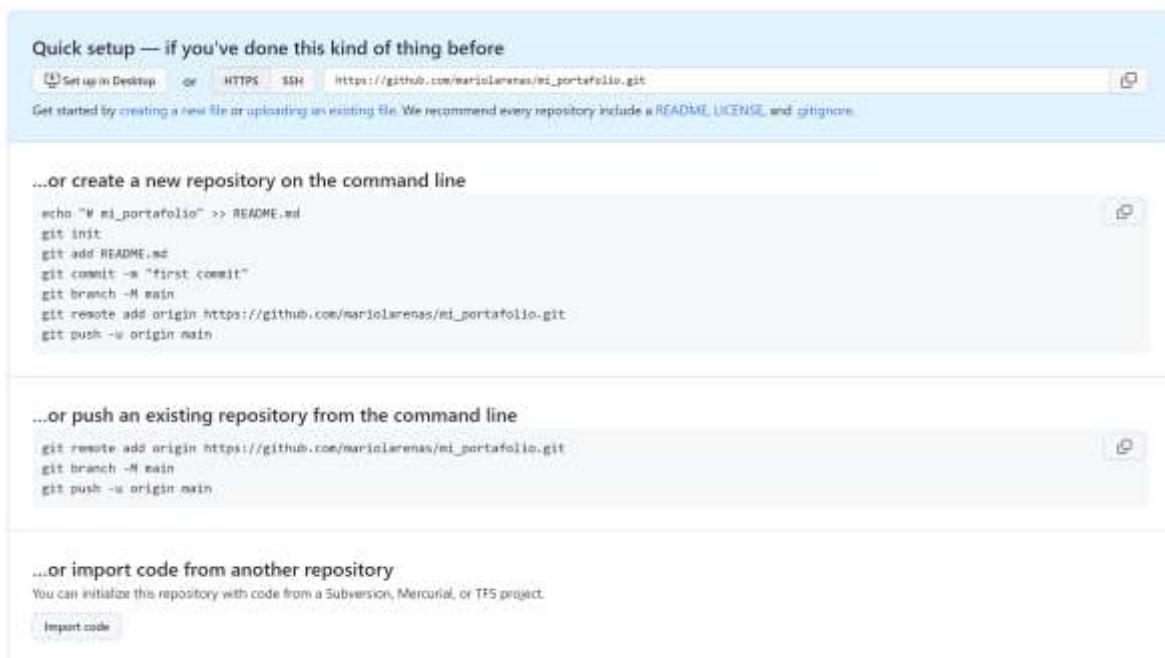
☐ **Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

☐ **Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

**Create repository**

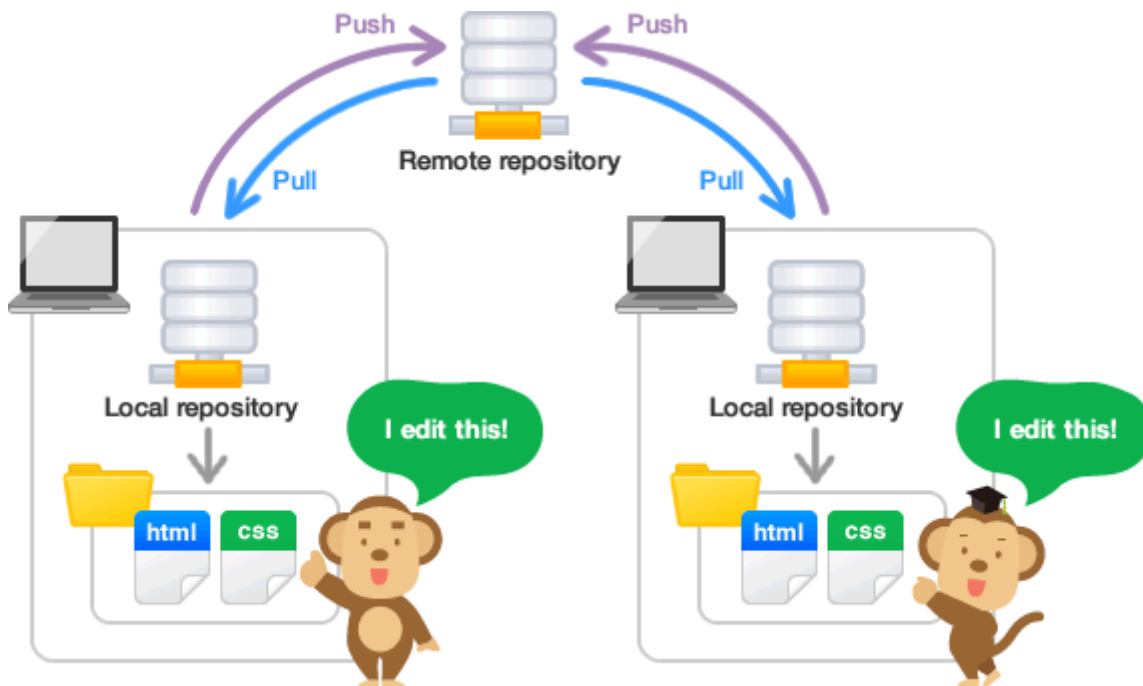
4. Al crear su repositorio Github le muestra los comandos para crear la copia en tu computador



Esta es la dirección que se debe configurar para tener acceso a ese repositorio desde tu repositorio local PC a él repositorio externo GitHub



**Nota:** Tener claro que nosotros trabajaremos en el repositorio Local y subiremos nuestros cambios al repositorio remoto o clonaremos la información de este para luego trabajar localmente, como se muestra en la imagen siguiente:



#### 4.1. Tienes 2 formas:

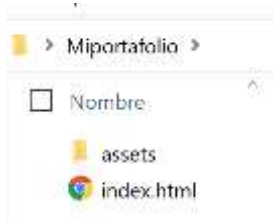
a) ...o crea un nuevo repositorio en la línea de comando

- `echo "# mi_portafolio" >> README.md`
- `git init`
- `git add README.md`
- `git commit -m "first commit"`
- `git branch -M main`
- `git remote add origin https://github.com/mariolarenas/mi_portafolio.git`
- `git push -u origin main`

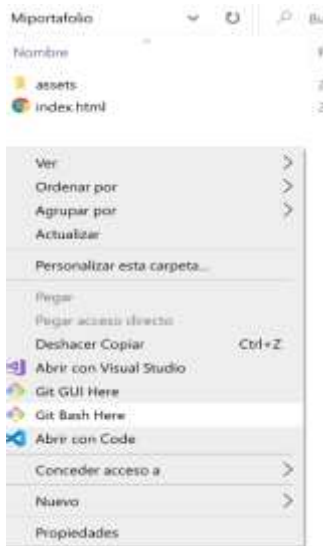
b) ...o enviar un repositorio existente desde la línea de comando

- `git remote add origin https://github.com/mariolarenas/mi_portafolio.git`
- `git branch -M main`
- `git push -u origin main`

5. Creamos o nos vamos a la carpeta que contenga localmente los datos. En este ejemplo git status



6. Botón derecho del mouse dentro de la carpeta contenedora y seleccionar “Git Bash here”



7. Lo primero es iniciar Git con el comando:

\$ git init (Nota: En caso de que sea primera vez que usen git pedirá usuario y clave de github y si aparece en celeste como en la imagen Master significa que ya se inicio git y eso quedo en la configuración global)

```
MINGW64/c:/Users/Sebastian/Desktop/Repositorio Miportafolio/miportafolio
Sebastian@LAPTOP-D7K8882T MINGW64 ~/Desktop/Repositorio Miportafolio/miportafolio
$ git init
Reinitialized existing Git repository in C:/Users/Sebastian/Desktop/Repositorio Miportafolio/miportafolio/.git/
Sebastian@LAPTOP-D7K8882T MINGW64 ~/Desktop/Repositorio Miportafolio/miportafolio (master)
$
```

realizamos los pasos del item 4.1 b)

- git remote add origin [https://github.com/mariolarenas/mi\\_portafolio.git](https://github.com/mariolarenas/mi_portafolio.git)
- git branch -M main

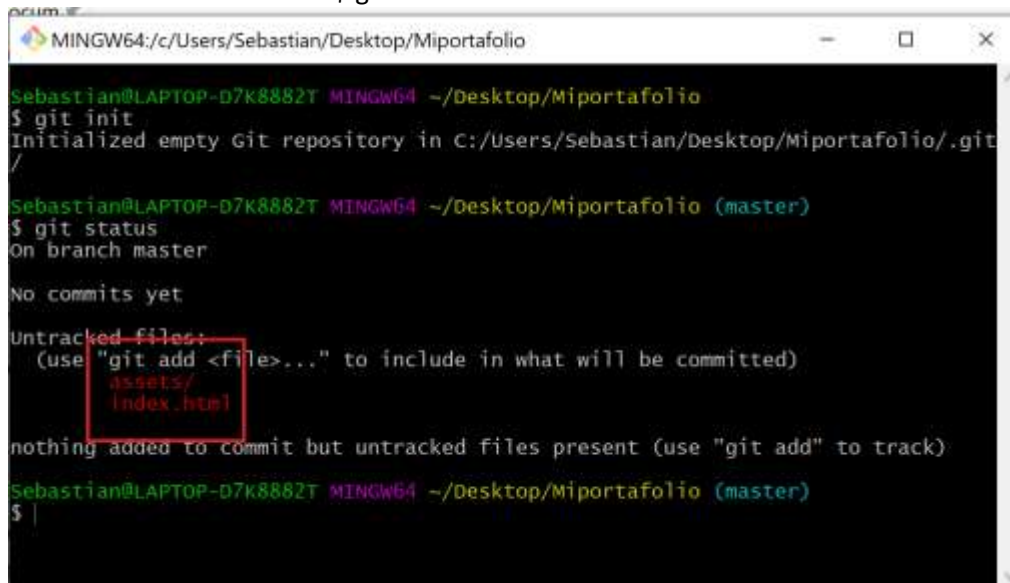
solo 1 vez para comprender verificamos el archivo config que esta en la carpeta local .git

```
[core]
  repositoryformatversion = 0
  filemode = false
  bare = false
  logallrefupdates = true
  symlinks = false
  ignorecase = true
[remote "origin"]
  url = https://github.com/mariolarenas/mi_portafolio.git
  fetch = +refs/heads/*:refs/remotes/origin/*
```

Git stat

8. Como el repositorio no tiene nada y queremos subir los archivos de Mi portafolio al repositorio creado anteriormente en GitHub "https://github.com/mariolarenas/mi\_portafolio.git"

Verificamos el estatus con: \$ git status



```
Sebastian@LAPTOP-D7K8882T MINGW64 ~/Desktop/Miportafolio
$ git init
Initialized empty Git repository in C:/Users/Sebastian/Desktop/Miportafolio/.git/

Sebastian@LAPTOP-D7K8882T MINGW64 ~/Desktop/Miportafolio (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  assets/
  index.html

nothing added to commit but untracked files present (use "git add" to track)

Sebastian@LAPTOP-D7K8882T MINGW64 ~/Desktop/Miportafolio (master)
$
```

Nos aparecen los archivos nuevos en rojo significa que no han sido adicionados!!! En nuestro repositorio local

\$ git add . //Esto añade todos los archivos y si quiero añadir un solo archivo  
git add nombrearchivo.extensión ejemplo: \$ git add index.html



```
Sebastian@LAPTOP-D7K8882T MINGW64 ~/Desktop/Miportafolio (master)
$ git add .
```

```
Sebastian@LAPTOP-D7K8882T MINGW64 ~/Desktop/Miportafolio (master)
$ git status
On branch master

No commits yet
```

```
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   assets/css/style.css
    new file:   assets/img/bg_hero.png
    new file:   assets/img/cuppon.png
    new file:   assets/img/dmark.png
    new file:   assets/img/iguana_page.png
    new file:   assets/img/meet_coffee.png
    new file:   assets/img/ricomida.png
    new file:   assets/img/suricata.png
    new file:   assets/js/script.js
    new file:   index.html
```

Y si se hace un git status se verifica que los archivos quedan en color verde lo que indica que fueron adicionados  
Luego se debe realizar un commit para aceptar los cambios realizados

```
$ git commit -m "Un comentario referente a los cambios realizados"
recordar que todo esto en el repositorio local
```

```
Sebastian@LAPTOP-D7K8882T MINGW64 ~/Desktop/miportafolio (main)
$ git commit -m "Index de prueba para mi portafolio"
[main (root-commit) 471a35b] Index de prueba para mi portafolio
1 file changed, 12 insertions(+)
create mode 100644 index.html
```

Verificamos que los archivos se actualizaron al repositorio local

```
Sebastian@LAPTOP-D7K8882T MINGW64 ~/Desktop/miportafolio (main)
$ git status
On branch main
nothing to commit, working tree clean
```

git bran

Renombrando la Rama:

Con una opción -m o -M, <rama antigua> cambiará de nombre a <nueva rama>. Si <Rama Antigua> tenía un reflog correspondiente, se le cambia el nombre para que coincida con <nueva rama> y se crea una entrada de reflog para recordar el cambio de nombre de la rama. Si existe <Nuevarama>, se debe usar -M para forzar el cambio de nombre.

```
git branch -M main
```

```
$ git branch -M main
Sebastian@LAPTOP-D7K8882T MINGW64 ~/Desktop/Mi Portafolio 211220/mi_portafolio (main)
$
```

\*\*\* Si queremos subir desde nuestro repositorio local al repositorio de github \*\*\*\*\*

Cuando especificas `$ git push origin` estás enviando explícitamente tus cambios al origin repositorio remoto.

\*\*\* Si queremos bajar el repositorio de github a a nuestro repositorio local COPIA siempre que no lo tengamos añadido sino nos saldrá un error\*\*\*\*

- `git remote add origin https://github.com/mariolarenas/mi_portafolio.git`

Para actualizar la información a la rama remota:

Al especificar `$ git push` sin parámetro de repositorio, de forma predeterminada enviará la rama actual a la rama remota de seguimiento.

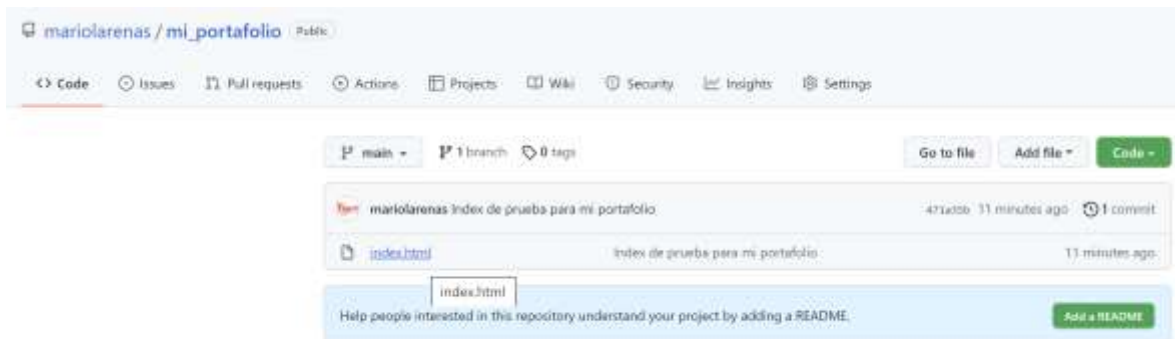
Cuando especificas `$ git push origin` estás enviando explícitamente tus cambios al origin repositorio remoto.

Para que cada rama este actualizada

```
$ git push -u origin main
$ git push origin main
```

```
Sebastian@LAPTOP-D7K8882T MINGW64 ~/Desktop/miportafolio (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 428 bytes | 428.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/mariolarenas/mi_portafolio.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

8.1 Ahora nos dirigimos al repositorio externo github para verificar que el contenido creado en mi repositorio local fue añadido a nuestro repositorio externo github:



8.2\*\*\* Si queremos actualizar nuestro repositorio local con el repositorio de github\*\*\*\*\*

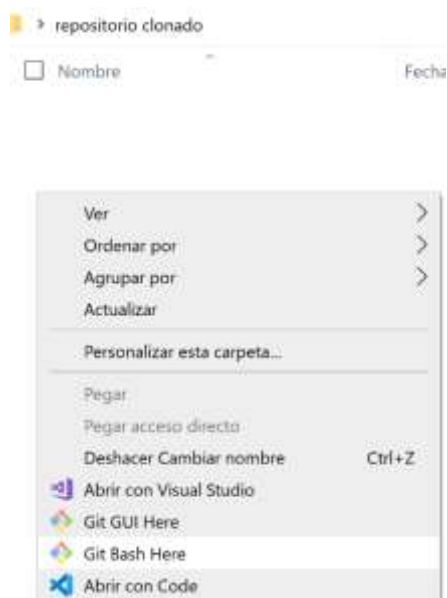
Esto es para actualizar nuestros archivos locales con los archivos externos

Ya iniciado git escribimos  
git pull

8.3 \*\*\* Si nuestro repositorio ya tiene archivos y estos queremos modificar, debemos clonar el repositorio remoto.

Elegimos una carpeta donde clonar nuestro repositorio remoto

Dentro de ella "Git Bash here"

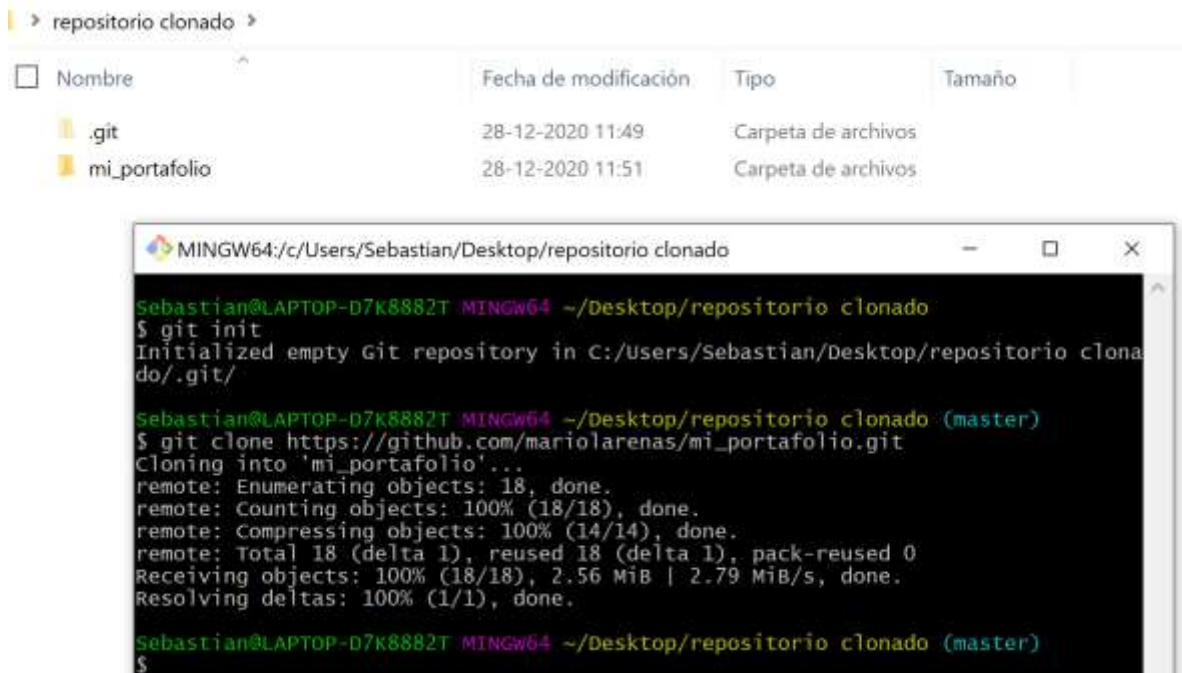


1ro En la consola Iniciar git:

```
$ git init
```

luego escribir “git clone” más la ruta del repositorio a clonar de github

```
$ git clone https://github.com/mariolarenas/mi_portafolio.git
```



Acá te darás cuenta que te dejara una carpeta con el mismo nombre que asignaste a tu repositorio de GitHub más el contenido en su interior.

9. Si terminaste tus modificaciones, actualizaciones de tus archivos los puedes subir

Recordar los pasos serian : add – commit – push

Add : adicionas archivos

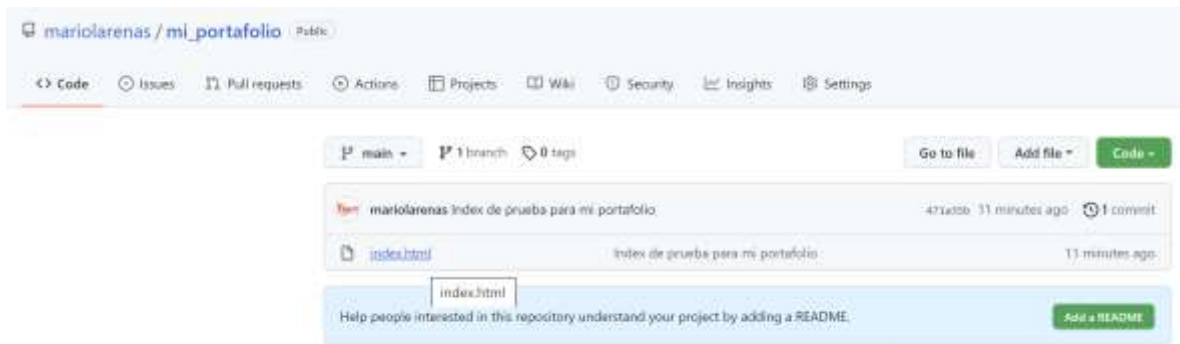
Commit: aceptas los cambios

Push: copias archivos de tu repositorio local a repositorio externo u otras ramas

Git log me muestra lo que se ha realizado

```
$ git log
```

10. Verificamos en el github nuestros archivos



## 11. RAMAS (Branchs):

# git branch (me muestra todas las ramas)

```
Awaker@Legion_Y520 MINGW64 ~/Desktop/Web en Git/developed (master)
$ git branch
* master
```

Git Branch -a muestra las ramas locales más las remotas

```
Awaker@Legion_Y520 MINGW64 ~/Desktop/Web en Git/developed (master)
$ git branch -a
* master
remotes/origin/master
```

## 12. Crear una rama nueva y direccionar a ella

Git checkout -b sebastian

```
Awaker@Legion_Y520 MINGW64 ~/Desktop/Web en Git/developed (master)
$ git checkout -b sebastian
Switched to a new branch 'sebastian'
```

## 13. Mostrar los cambios realizados

git log

```
Awaker@Legion_Y520 MINGW64 ~/Desktop/Web en Git/developed (sebastian)
$ git log
commit 1cf7847599075bbfa739dcf05ce4b67062825fb9 (HEAD -> sebastian, origin/master, prueba, master)
Author: mariolarenas <mario.larenas@gmail.com>
Date: Wed Jun 10 10:49:07 2020 -0400

Archivos originales formulario
```



14. Hacemos un cambio en el html por ejemplo (Recordar que estamos en la rama sebastian)

```
<body>
  <div class="contenedor">
    <form action="/" class="form">
      <div class="form-header">
        <h1 class="form-title"><span>Contacto</span></h1>
      </div>
      <label for="nombres" class="form-label">Nombres</label>
      <input type="text" id="nombre" class="form-input" placeholder="Escriba su nombre">
      <label for="apellidos" class="form-label">Apellidos</label>
      <input type="text" id="apellido" class="form-input" placeholder="Escriba sus apellidos">
      <label for="direccion" class="form-label">Dirección</label>
      <input type="text" id="direccion" class="form-input" placeholder="Ingrese dirección">
      <label for="email" class="form-label">Email</label>
      <input type="text" id="email" class="form-input" placeholder="Ingrese email">
      <label for="mensaje" class="form-label">Mensaje</label>
      <textarea id="mensaje" class="form-textarea" placeholder="Aqui va su mensaje"></textarea>
      <input type="submit" class="btn-submit" value="Enviar Mensaje">
    </form>
  </div>
</body>
```

Luego hacemos un git add formulario.html

```
Awaker@Legion_Y520 MINGW64 ~/Desktop/Web en Git/developed (sebastian)
$ git add formulario.html

Awaker@Legion_Y520 MINGW64 ~/Desktop/Web en Git/developed (sebastian)
$ git commit -m "Segundo comentario"
On branch sebastian
Untracked files:
  developed/

nothing added to commit but untracked files present
```

15. Git commit – m "segundo comentario"

```
Awaker@Legion_Y520 MINGW64 ~/Desktop/Web en Git/developed (sebastian)
$ git commit "Segundo Comentario"
error: pathspec 'Segundo Comentario' did not match any file(s) known to git

Awaker@Legion_Y520 MINGW64 ~/Desktop/Web en Git/developed (sebastian)
$ git commit -m "Segundo Comentario"
[sebastian 3cd9a06] Segundo Comentario
1 file changed, 3 insertions(+)
```

Con esto tenemos los cambios hechos en la rama sebastian pero no en la master este cambio no se pierde solo esta en el formulario de la rama sebastian

16. Para actualizar de la rama sebastian a la master de git. Debo ir a la rama master  
git checkout master git checkout master

```
Awaker@Legion_Y520 MINGW64 ~/Desktop/Web en Git/developed (sebastian)
$ git checkout master
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

Awaker@Legion_Y520 MINGW64 ~/Desktop/Web en Git/developed (master)
$ |
```

Con eso me cambio de rama

17. Juntamos lo de la rama sebastian con master

Git merge sebastian

Trae lo de sebastian a la master local

```
Awaker@Legion_Y520 MINGW64 ~/Desktop/Web en Git/developed (master)
$ git merge sebastian
Updating 1cf7847..3cd9a06
Fast-forward
 formulario.html | 3 +++
 1 file changed, 3 insertions(+)

Awaker@Legion_Y520 MINGW64 ~/Desktop/Web en Git/developed (master)
$ |
```

18. Git Status

```
Awaker@Legion_Y520 MINGW64 ~/Desktop/Web en Git/developed (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

19. adicionamos todo y aceptamos

Add y commit

```
Awaker@Legion_Y520 MINGW64 ~/Desktop/Web en Git/developed (master)
$ git add .

Awaker@Legion_Y520 MINGW64 ~/Desktop/Web en Git/developed (master)
$ git commit -m "se subio el archivo html de sebastian a master"
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

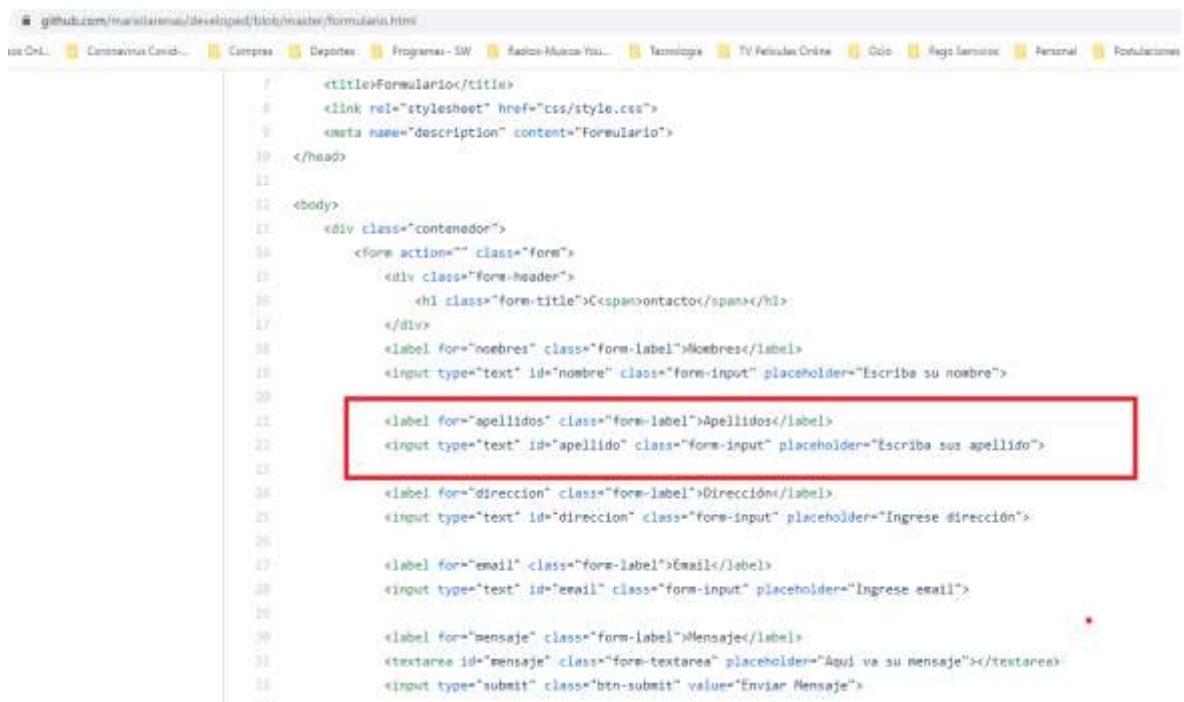
## 19. Subimos a git web

Git push origin master

```
Awaker@Legion_Y520 MINGW64 ~/Desktop/web en Git/developed (master)
$ git push origin master
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), 344 bytes | 344.00 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/mariolarenas/developed.git
1cf7847..3cd9a06 master -> master
```

Ojo. Para hacer un push debo estar situado en la rama master local para subir a la rama master web sino no te dejara subir desde ejemplo la rama sebastian

## 20. chequeamos en el git web github

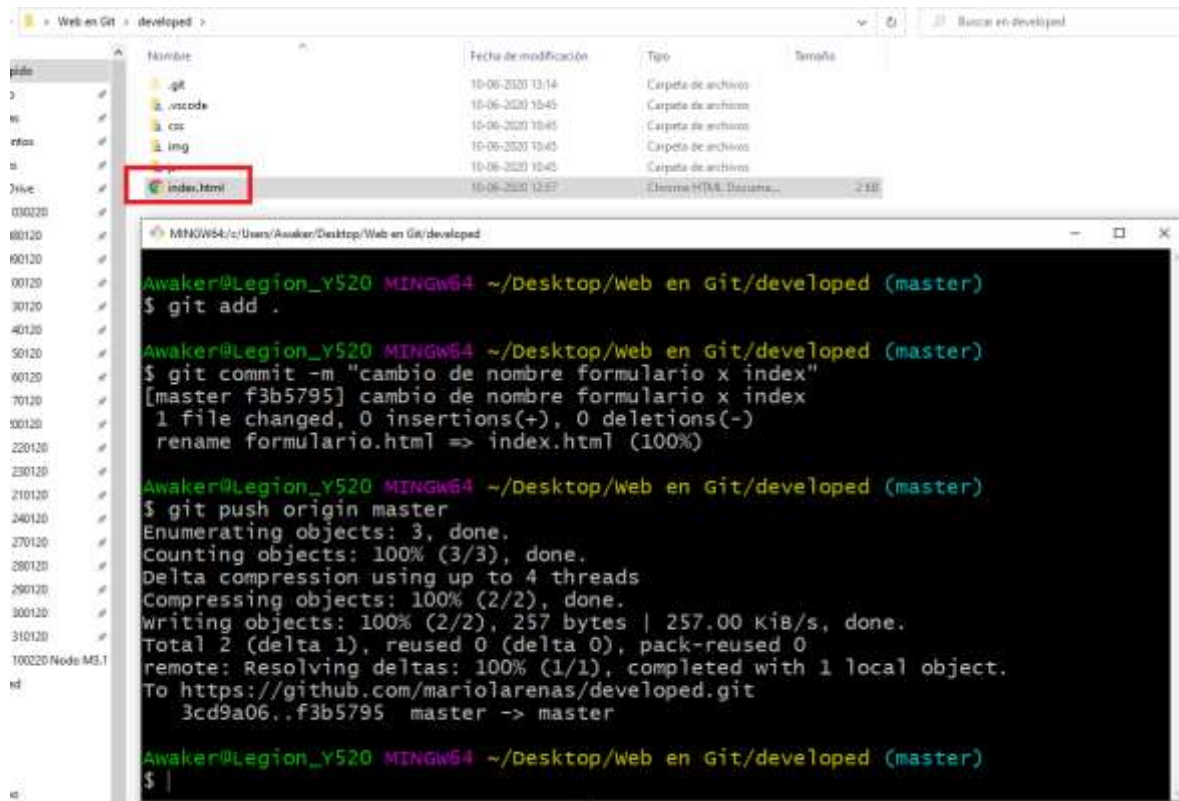


## 21. Por ultimo los tres pasos básicos.

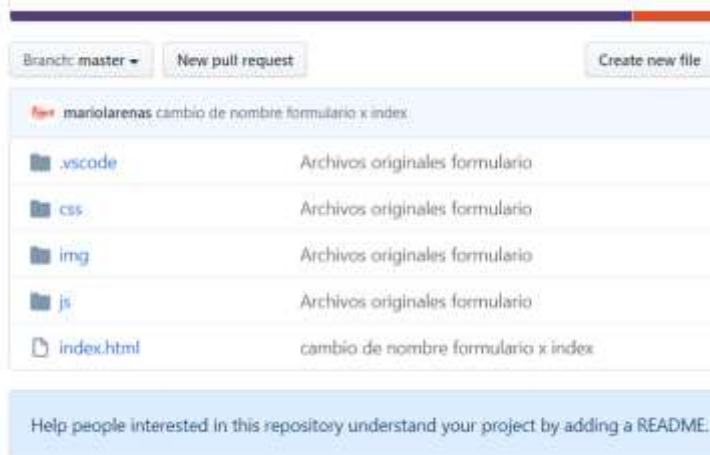
git add, git commit y git push

para cambiar el nombre del archivo formulario por index





Chequeando en github



22. Parte grafica de git : gitk





23. Si otro usuario subió archivos al repositorio que van a compartir

Cada uno debe actualizar desde el repositorio web a sus repositorios locales

Con el siguiente comando:

Git fetch origin

```
Sebastian@LAPTOP-D7K8882T MINGW64 ~/Desktop/miportafolio (main)
$ git fetch origin
```

Este copia los cambios que se subieron al GitHub y los copia al repositorio LOCAL de su pc

Luego hacen un merge origin/main

```
Sebastian@LAPTOP-D7K8882T MINGW64 ~/Desktop/miportafolio (main)
$ git merge origin/main
Already up to date.
```

Verifican con git status

Para subir o eliminar archivos:

Recordar los pasos básicos para subir archivos:

seguir pasos → add → commit → push

Para eliminar una rama de nuestro repositorio local ejecutaremos el siguiente comando:

```
$ git branch -d nombre_rama
```

En el caso de que esa rama contenga trabajos sin fusionar, el comando anterior nos devolverá el siguiente error:

error: The branch 'nombre-rama' is not an ancestor of your current HEAD.  
If you are sure you want to delete it, run 'git branch -D nombre-rama'.

Si aún así queremos eliminar esa rama, se puede forzar el borrado de la siguiente manera:

```
$ git branch -D nombre-rama
```

En el caso de querer eliminar una rama del repositorio remoto, la sintaxis será la siguiente:

```
$ git push origin :nombre-rama
```

Y de esta forma, desaparecerá la rama nombre-rama del servidor.

Resumen de lo básico:

Resumen Git visto en clases. En github crear un repositorio

2. copiar repositorio: <https://github.com/usuario/repositorio.git>

3. crear carpeta contenedora eje en el escritorio

4. boton derecho del mouse git bash here

5. \$ git init

6. \$ git remote add origin <https://github.com/usuario/repositorio.git>

7. cambiamos la rama a Main

\$ git branch -M main Si queremos subir archivos de nuestro repo local al repo externo github1. \$ git add archivo

git add . (si queremos adicionar todos los archivos) 2. \$ git commit -m "pequeño comentario de la transacción"

3. \$ git push -u origin main si queremos actualizar la información desde repo externo al repo local1. \$ git pull si queremos clonar un repositorio externo a un repo local1. \$ git clone <https://github.com/usuario/repositorio.git>

EXTRA:

Algunos Comandos básicos de Linux que se podemos utilizar en Git:

Linux es una familia completa de sistemas operativos Unix de código abierto, que se basan en el kernel de Linux.

#### 1. comando pwd

Usa el comando **pwd** para encontrar la ruta del directorio (carpeta) de trabajo actual en el que te encuentras. El comando devolverá una ruta absoluta (completa), que es básicamente una ruta de todos los directorios que comienzan con una barra diagonal (/) Un ejemplo de una ruta absoluta es **/home/nombredeusuario**.

#### 2. comando cd

Para navegar por los archivos y directorios de Linux, usa el comando **cd**. Te pedirá la ruta completa o el nombre del directorio, dependiendo del directorio de trabajo actual en el que te encuentres.

Supongamos que estás en **/home/nombredeusuario/Documentos** y deseas ir a **Fotos**, un subdirectorio de **Documentos**. Para hacerlo, simplemente escribe el siguiente comando: **cd Fotos**.

Otro escenario es si deseas ir a un directorio completamente nuevo, por ejemplo, **/home/nombredeusuario/Peliculas**. En este caso, debes escribir **cd** seguido de la ruta absoluta del directorio: **cd /home/ nombredeusuario/Peliculas**.

Hay algunos atajos para ayudarte a navegar rápidamente:

- **cd ..** (con dos puntos) para ir un directorio hacia arriba
- **cd** para ir directamente a la carpeta de inicio
- **cd-** (con un guión) para ir al directorio anterior

Como nota al margen, el shell de Linux distingue entre mayúsculas y minúsculas. Por lo tanto, debes escribir el nombre del directorio de forma exacta.

#### 3. comando ls

El comando **ls** se usa para ver el contenido de un directorio. Por defecto, este comando mostrará el contenido de tu directorio de trabajo actual.

Si deseas ver el contenido de otros directorios, escribe **ls** y luego la ruta del directorio. Por ejemplo, ingresa **ls/home/nombredeusuario/Documentos** para ver el contenido de **Documentos**.

Hay variaciones que puedes usar con el comando **ls**:

- **ls -R** también listará todos los archivos en los subdirectorios
- **ls -a** mostrará los archivos ocultos

- **ls -al** listará los archivos y directorios con información detallada como los permisos, el tamaño, el propietario, etc.

#### 4. comando cat

**cat** (abreviatura de concatenate, en inglés) es uno de los comandos más utilizados en Linux. Se utiliza para listar el contenido de un archivo en la salida estándar (stdout). Para ejecutar este comando, escribe **cat** seguido del nombre del archivo y su extensión. Por ejemplo: **cat archivo.txt**.

Aquí hay otras formas de usar el comando **cat**:

- **cat > nombreadarchivo** crea un nuevo archivo.



```
Sebastian@LAPTOP-D7K8882T MINGW64 ~/Desktop/miportafolio (main)
$ cat > nombreadarchivo.txt
```

- **cat nombreadarchivo1 nombreadarchivo2>nombreadarchivo3** une dos archivos (1 y 2) y almacena la salida de ellos en un nuevo archivo (3)
- convertir un archivo a mayúsculas o minúsculas, **cat nombreadarchivo | tr a-z A-Z> salida.txt**

Pueden ver mas aca:

<https://www.hostinger.es/tutoriales/linux-comandos>