

# Instalación Git Bash

## • ¿Qué es?

Es una aplicación de terminal que se utiliza como interfaz con un S.O. mediante comandos escritos

## • Compatibilidad

• Mac Os    • Windows    • Linux/Unix

La Terminal → Interfaz de línea de comandos (CLI) ←

↳ Damos instrucciones a la computadora a través de comandos que son órdenes

## • ¿Cuáles son las ventajas?

- Velocidad (Tree)
- Compatibilidad
- Sin distracciones
- Más Productivo

Entrada [stdin]

Salida [stdout]

Error [stderr]

Navegación e interacciones con ficheros

## • De tipo informativo

pwd: Mostrar el nombre de la carpeta en el que se encuentra situado

cd: Cambiar la carpeta de trabajo, nos podemos mover entre diferentes directorios

ls: Listar el contenido de directorios. Este comando lista los ficheros y carpetas.

## • Crear o eliminar Carpetas

mkdir: Crear una carpeta → Make directory

rmdir: Borrar una carpeta → Solo borra carpetas vacías

rm -r → Borra carpeta y todo lo que tiene adentro

## • Crear o eliminar Archivos

edit: Crear o editar archivos de texto

rm: Borrar archivos → remove

## • Copiar Archivos y Directorios

cp: Copiar un archivo o carpeta en el directorio específico

cp -r: Copiar Carpetas

mv: Mover un archivo ó carpeta a un archivo o carpeta.

# Git Hub

A modo de resumen, repasamos las palabras y conceptos clave del video sobre GitHub:

- GitHub es un lugar en la nube
- Repositorio es el lugar donde se irán almacenando los archivos de nuestro proyecto y a través del cual podremos hacer seguimiento de los mismos.
- Repositorios Remotos: viven en la nube, es decir, en GitHub
- Repositorios Locales: viven en nuestra computadora.

## Comandos !!!

git add . // agrega todos los archivos

git commit -m "mensaje" // comitea los cambios hechos

git push origin master // envia los cambios al repositorio remoto

git status // seguimiento del estado de los archivos

git init // crea el repositorio local

git config user.name "nombreUsuario" // agrega nuestra identidad

git config user.email "CorreoElectronico@gmail.com" // agrega nuestro email

git remote -v

## Bajando Archivos en GitHub

Git clone // clonar el repositorio en cualquier computadora

Ej: \$ git clone https://github.com/usuario/repositorio.git

↓  
Debo estar en la carpeta  
donde voy a clonar los  
archivos.

Git Pull // Descarga la actualización de los archivos cambios que he realizado

Ej: \$ git pull origin master

Para cambiar de Master a Main → \$ git branch -M main  
ó master

`rm -rf .git` → Desaparece la inicialización en una carpeta.

`git log --online` Muestra el mismo log pero en una linea → **Ramas**

Para volver un commit atrás →  
sin borrar lo anterior  
Copiando el código viejo

↑  
copio el código del commit  
git checkout ae17377

Para volver a la Rama Principal → Git checkout main

Para Borrar rm -rf \*  
lo de la carpeta

Al crear ramas lo que generamos son versiones del repositorio donde están los archivos del mismo más los cambios que le iremos sumando

## Git Branch

Para crear una nueva rama utilizamos el comando `git branch`, este nos permite crear, enumerar cambios el nombre y eliminar ramas. No permite cambiar entre ramas o volver a unir un historial bifurcado.

- `git branch`

Enumera todas las ramas de tu repositorio, es similar al `git branch --list`

- `git branch <branch>`

Crea una nueva rama llamada `<branch>`

- `git -d <branch>`

Elimina la rama llamada `<branch>`. Git evita que eliminemos la rama si tiene cambios que aún no se han fusionado con la rama Main.

- `git -D <branch>`

Fuerza la eliminación de la rama especificada, incluso si tiene cambios sin fusionar.

## Git checkout

Para moverse de una rama a otro, se ejecuta el comando

- `git checkout nombre_rama`

Para movernos a master `git checkout master`

Generalmente, Git solo permitirá que nos movamos a otra rama sino tenemos cambios. Si tenemos cambios, para cambiarnos de rama, debemos:

1. Eliminarlos (deshaciendo los cambios)
2. Confirmarlos (haciendo un `git commit`)

## Guardar cambios y subirlos al repositorio remoto

Una vez que terminamos de realizar los cambios que queremos en nuestra branch, ejecutamos los mismos comandos que vimos hasta ahora: git add, git commit, git status y git log.

Pero cuando queremos subir esos cambios, debemos utilizar git push con el nombre de la rama en que estamos posicionados:

- git push origin <branch>

Así también, para traer los cambios de esa rama utilizaremos el git pull agregando desde donde queremos traer los cambios

- git pull origin <branch>

## Ramas o branch

Comandos ??

// Crear una rama nueva

git branch nombreRama

// Nos muestra en que rama estamos

git branch

Podemos unir la rama master con la nueva para eso debemos estar en la master para ejecutar el sig comando:

// nos movemos a la nueva rama

git merge nombreRama

// Eliminar una rama

git branch -d nombreRama

\*\* Atajo \*\*

Podemos utilizar git checkout -b nuevaRama para crear la nuevaRama y viajar a ella.

## Tags

Podemos hacer versiones de nuestro proyecto

// Crear un tags

git tag versionAlpha -m "versión Alpha"

// listar Tags

git tag

// Borrar Tags

git tag -d nombreTags

// Hacer una versión en un commit anterior ej: f52f3da

git tag -a nombreTag f52f3da -m "Version alpha"

// Mostrar información del tag

git show nombreTag

---

---

// Para realizar la unión de una rama a la principal

- Debemos estar como master (git checkout master)
- git merge nombreRama

## Pasos para Rama

1) Realizamos los cambios en el código

En caso de querer saber donde estoy

↳ git log --oneline → Este permite ver si estoy en master

2) Creamos la rama git branch nombreRama

3) Nos movemos para esa rama git checkout nombreRama

Si en este momento quiero ver en que rama estoy

↳ git branch

4) Realizamos el add . para añadir los cambios  
git add .

5) Despues realizamos el git commit -m "mensaje"

6) Verifiquemos en que rama estamos

↳ git log --oneline

7) Debemos posicionarnos en la rama master  
↳ git checkout master

8) Para realizar la union de esta rama con la Principal

↳ git merge nombreRama → Para saber el nombre de la rama  
git branch

9) Para eliminar una rama

git branch -d nombreRama

## Problemas con las ramas

Si se modifica el mismo archivo tanto Master como Rama en el mismo sitio al momento de hacer merge habrá un conflicto

1) Debemos guardar los cambios tanto del Master como la Rama git add.  
Para ver en donde estoy ubicado git branch

git commit -m "mensaje"

2) Estando en Master → git checkout master realizamos la unión de este con la rama  
↳ git merge nombre Rama

3) En este momento habrá un conflicto donde podemos solucionar así:

- Aceptar cambios Actual → Master
- Aceptar cambios entrantes → Rama
- Aceptar ambos cambios → La edición de ambos se unirá

4) git status -s el nombre del archivo acompañado de uu  
↳ git add.  
↳ git commit -m "arreglado el conflicto"

git remote add origin url del Repository

Vehico

git remote -v

git log --oneline

git push origin main  
                  o  
                  master → Subir los archivos

Para cambiarme de rama

{ git checkout nombreRama  
      o  
      git switch nombre de la rama

Para subir una rama  
o hacer push

{ git push origin nombreRama