**Curso de Git - Apuntes y Teoría**

### Comandos Básicos de Navegación y Creación de Directorios

Git se utiliza desde la terminal, por lo que conocer los comandos básicos de navegación es fundamental antes de trabajar con repositorios. A continuación, se presentan algunos comandos comunes:

* clear: Limpia la terminal, pero no cambia el directorio de trabajo actual.
* reset: Borra el historial de la consola.
* ls: Muestra el listado de carpetas y archivos en el directorio actual.
* pwd: Muestra la ruta completa del directorio en el que estamos ubicados.
* cd [directorio]: Permite ingresar a un directorio específico.
* cd ..: Permite salir al directorio anterior.
* mkdir miweb: Crea una nueva carpeta llamada “miweb” en el directorio actual.

Para iniciar un repositorio de Git:

* git init: Inicializa un repositorio vacío en la carpeta actual. Esto crea una carpeta oculta .git que contiene todos los archivos necesarios para el control de versiones.
* ls -a: Muestra archivos ocultos, incluyendo .git.
* cd .git: Permite ingresar a esa carpeta y visualizar sus contenidos, que representan los archivos internos que Git usa para rastrear el repositorio. A esta estructura se le conoce como el “detalle de implementación”.

**Nota:** La carpeta .git es ignorada por defecto al compartir el proyecto y no se sube a repositorios remotos.

### Flujo de Trabajo con Git

El flujo general de trabajo con Git consiste en tres etapas principales:

**1. Working Directory (Computadora)** → **2. Staging Area (Stage)** → **3. Repository (Commit)** → **4. Remote (Servidor)**

La etapa de *Staging* sirve para preparar los archivos antes de confirmar (commit) los cambios.

### Agregando Cambios al Staging

* code .: Abre la carpeta actual en Visual Studio Code.
* git status: Muestra el estado del repositorio, incluyendo archivos modificados y no rastreados.

Para agregar archivos al área de staging:

* git add archivo.txt: Agrega un archivo específico.
* git add .: Agrega todos los archivos (no es recomendable si no se tiene control del contenido).
* git add \*.txt: Agrega todos los archivos con extensión .txt.
* git add archivo3.txt archivo4.txt: Agrega varios archivos al mismo tiempo.

**Nota:** Cuando se modifican archivos existentes, se deben volver a agregar al staging con git add, incluso si ya estaban en el repositorio.

Para confirmar los cambios:

* git commit -m "Mensaje": Realiza el commit con un mensaje explicativo.
* git commit: Si no se agrega -m, se abrirá el editor predeterminado para escribir el mensaje de commit.

**Nota:** En Git, hacer un “commit” significa guardar los cambios de forma permanente en el historial del proyecto.

### Eliminando Archivos del Repositorio

* ls: Visualiza los archivos disponibles.
* rm archivo2.txt: Elimina el archivo.
* git status: Verifica el cambio.
* git add archivo2.txt: Añade la eliminación al staging.
* git commit -m "Eliminando archivo 2": Confirma la eliminación.

Alternativa resumida:

* git rm archivo2.txt: Elimina el archivo y lo agrega al staging en un solo paso.

Para restaurar un archivo en el staging antes de confirmar:

* git restore --staged archivo1.txt: Quita el archivo del staging.

### Volver a Staging y Descartar Cambios

* git restore archivo1.txt: Restaura el archivo al último commit.
* git status: Verifica el estado tras la restauración.

### Renombrar o Mover Archivos

* mv archivo1.txt nuevo\_nombre.txt: Cambia el nombre o mueve el archivo.
* git add archivo1.txt nuevo\_nombre.txt: Agrega ambos cambios al staging.
* git commit -m "Renombrando archivo": Confirma los cambios.

Comando simplificado:

* git mv archivo.txt archivo1.txt: Renombra el archivo y lo agrega al staging.

### Ignorar Archivos y Directorios

**Nota:** En proyectos reales se desea evitar subir archivos sensibles o específicos del entorno, como contraseñas o configuraciones locales.

Para ignorar archivos:

1. Crear un archivo .gitignore en el directorio raíz.
2. Añadir los nombres de los archivos y carpetas que se deben excluir (por ejemplo: .env).

Después de ignorarlos:

* git status: El archivo ignorado ya no aparecerá.
* git status -s: Muestra el estado de los archivos con una notación más concisa.

### Visualización de Cambios

* git diff: Muestra los cambios realizados (líneas agregadas en verde).
* git diff --staged: Muestra los cambios ya preparados para commit.

### Revisión del Historial

* git log: Muestra el historial completo de commits con autor, fecha y mensaje.
* git log --oneline: Muestra los commits en una sola línea con el hash abreviado.

### Ramas en Git (Branches)

Git permite desarrollar nuevas funcionalidades en paralelo sin afectar la rama principal.

* git branch: Muestra todas las ramas existentes.
* git checkout -b ramab: Crea y cambia a una nueva rama llamada ramab.
* git checkout ramab: Cambia entre ramas existentes.
* git merge ramab: Fusiona los cambios de la rama ramab a la rama actual.

**Nota:** Las ramas se usan para mantener organizado el flujo de trabajo, especialmente en equipos.

### Conexión con GitHub

Para conectar un repositorio local con uno remoto en GitHub:

* git remote add origin https://github.com/Alejandro40K/miweb.git
* git push -u origin main

### Crear un Nuevo Repositorio desde la Línea de Comandos

echo "# miweb" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git branch -M main  
git remote add origin https://github.com/Alejandro40K/miweb.git  
git push -u origin main

En caso de que ya exista el repositorio en GitHub:

git remote add origin https://github.com/Alejandro40K/miweb.git  
git branch -M main  
git push -u origin main

Este resumen incluye comandos esenciales, comentarios personales y explicaciones para ayudarte a comprender cada parte del flujo de trabajo con Git y GitHub.