Guía git clone

1. Crea un repositorio en GitHub:
    - Inicia sesión en tu cuenta de GitHub.
    - Haz clic en el botón "+", en la esquina superior derecha de la página, y selecciona "New repository" (Nuevo repositorio).
    - Ingresa un nombre para tu repositorio, opcionalmente añade una descripción y elige las configuraciones deseadas.
    - Asegúrate de dejar desmarcada la opción "Initialize this repository with a README" (Inicializar este repositorio con un README) y no agregues ningún archivo al repositorio en este momento.
    - Haz clic en "Create repository" (Crear repositorio) para finalizar la creación.
2. Abre una terminal en tu sistema operativo. En clase utilizamos bash (terminal integrada de linux), pueden utilizar la terminal integrada de windows en su defecto.
3. En tu repositorio local, abre una terminal y navega hasta el directorio raíz del proyecto.
4. Navega al directorio donde deseas clonar el repositorio utilizando el comando `cd ruta_del_directorio`.
5. Obtén la URL del repositorio que deseas clonar. Por lo general, puedes encontrarla en la página web del repositorio o en la interfaz de usuario de Git.

Ejemplo: <https://github.com/usuario/repositorio.git>

1. En la terminal, ejecuta el comando `git clone <URL_del_repositorio>`. Reemplaza `<URL_del_repositorio>` con la URL que copiaste en el paso anterior. Por ejemplo:

```
git clone https://github.com/usuario/repositorio.git
```

2. Presiona Enter y Git comenzará a descargar todos los archivos del repositorio en el directorio actual.
3. Una vez completada la clonación, tendrás una copia local completa del repositorio en tu máquina.
4. Para ver la lista de ramas remotas en Git, puedes utilizar el comando `git remote -v`. Este comando mostrará todas las ramas remotas disponibles en el repositorio.
5. Para ver los elementos ocultos en la terminal y determinar dónde se encuentra el directorio `.git` de tu repositorio Git, puedes utilizar el comando `ls` con la opción `-a` para mostrar todos los archivos, incluyendo los ocultos:

   ls -a
6. Inicializa Git en tu repositorio local ejecutando el siguiente comando:

   ```
   git init
   ```
7. Agrega los archivos de tu repositorio local al área de preparación utilizando el comando:

   ```
   git add .
   ```

   Esto agregará todos los archivos y cambios en el directorio actual al área de preparación de Git. Si deseas agregar archivos específicos, reemplaza `.` con los nombres de archivo correspondientes.
8. Realiza un commit de los cambios utilizando el comando:

   ```
   git commit -m "Mensaje del commit"
   ```

   Proporciona un mensaje descriptivo para el commit que refleje los cambios realizados.

9. Agrega el repositorio remoto de GitHub como destino de tu repositorio local ejecutando el siguiente comando:

```
git remote add origin <URL_del_repositorio>
```

   Reemplaza `<URL_del_repositorio>` con la URL del repositorio remoto en GitHub. Por ejemplo:

```
git remote add origin https://github.com/usuario/repositorio.git
```

10. Empuja los cambios a la rama principal en el repositorio remoto utilizando el siguiente comando:

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

    Reemplaza `main` con el nombre de la rama principal en tu repositorio de GitHub, que puede ser `main` o `master` según la configuración.

    Si es la primera vez que empujas a esta rama, utiliza `-u` para establecerla como la rama de seguimiento por defecto.

11. Proporciona tus credenciales de GitHub si se te solicitan.

Configurar las variables globales de git en tu local:

1. Siguiendo las siguientes instrucciones:

a. Abre una terminal o línea de comandos en tu sistema.

b. Ejecuta los siguientes comandos, reemplazando `<nombre>` con tu nombre y `<correo>` con tu dirección de correo electrónico:

```
git config --global user.name "<nombre>"
git config --global user.email "<correo>"
```

Estos comandos establecerán tus variables globales de nombre de usuario y dirección de correo electrónico respectivamente.

c. Puedes verificar que las variables globales se hayan configurado correctamente ejecutando los siguientes comandos:

```
git config --global user.name
git config --global user.email
```

Estos comandos mostrarán los valores configurados para el nombre de usuario y la dirección de correo electrónico.

Bonus: Git bash

Instalación Git Bash en windows:

1. Ve al sitio web oficial de Git Bash en **https://gitforwindows.org/**.
2. Haz clic en el botón "Download" (Descargar) en la página principal del sitio.
3. Selecciona la versión de instalación que coincida con la arquitectura de tu sistema operativo (32 bits o 64 bits). Si no estás seguro de qué versión elegir, puedes verificar la arquitectura de tu sistema operativo siguiendo estos pasos:
    - En Windows: Abre el "Panel de control" y busca "Sistema". En la sección "Tipo de sistema" o "System type", encontrarás la información sobre la arquitectura de tu sistema operativo.
4. Una vez descargado el instalador, ejecútalo y sigue las instrucciones del asistente de instalación.
5. Durante el proceso de instalación, se te presentarán varias opciones. Asegúrate de seleccionar las siguientes opciones:
    - "Use Git from Git Bash only" (Usar Git solo desde Git Bash)
    - "Checkout Windows-style, commit Unix-style line endings" (Usar estilo de Windows para la línea de comandos y estilo Unix para los commits)
    - "Use Windows' default console window" (Usar la ventana de consola predeterminada de Windows)

- "Default (fast-forward or merge)" (Opción de mezcla rápida o fusión por defecto)

6. Continúa siguiendo las instrucciones del asistente de instalación hasta que la instalación se complete.
7. Una vez finalizada la instalación, puedes abrir Git Bash desde el menú de inicio o buscándolo en tu sistema.

cd: cambiar de directorio.

ls: ver lo que hay en el directorio.

ls -a: ver todo aunque esté oculto.

Manejo de git/github en equipo, en un mismo repositorio en GitHub:

1. Crea un repositorio en GitHub: Inicia sesión en tu cuenta de GitHub y crea un nuevo repositorio. Puedes darle un nombre descriptivo, agregar una descripción opcional y seleccionar otras configuraciones según tus necesidades.
2. Agrega a los colaboradores: En la página de configuración del repositorio en GitHub, ve a la pestaña "Settings" (Configuración) y selecciona "Manage access" (Gestionar acceso). Desde allí, puedes invitar a otros usuarios como colaboradores ingresando sus nombres de usuario de GitHub o sus direcciones de correo electrónico. Los colaboradores recibirán una invitación por correo electrónico y, una vez que la acepten, podrán acceder al repositorio.
3. Clona el repositorio: Cada colaborador debe clonar el repositorio en su computadora local utilizando el siguiente comando en la línea de comandos:

   ```
   git clone <URL_del_repositorio>
   ```

   Reemplaza `<URL_del_repositorio>` con la URL del repositorio que se muestra en la página principal del repositorio en GitHub. Esto creará una copia local del repositorio en la computadora del colaborador.
4. Realiza cambios y commits: Cada colaborador puede realizar cambios en los archivos del repositorio clonado en su computadora utilizando un editor de código o

cualquier otra herramienta. Luego, deben agregar los cambios y realizar commits utilizando los comandos `git add` y `git commit`.

5. Comparte los cambios: Una vez que los colaboradores han realizado sus commits locales, pueden compartir sus cambios con los demás colaboradores utilizando el comando `git push`. Esto enviará los commits al repositorio en GitHub.
6. Mantén el repositorio actualizado: Para mantenerse al día con los cambios realizados por otros colaboradores, cada colaborador puede ejecutar el comando `git pull` para obtener las últimas actualizaciones del repositorio remoto en su rama local.

Es importante tener en cuenta que cuando varios colaboradores trabajan en el mismo repositorio, pueden surgir conflictos si dos o más colaboradores realizan cambios en el mismo archivo al mismo tiempo. Git proporciona herramientas para resolver estos conflictos mediante la fusión (merge) de cambios o la selección de versiones específicas.

git init → Da comienzo al control de versiones.

git status → muestra el status de nuestro repositorio en concordancia con el remote

git add . → añadimos el archivo a nuestra fase de staging, que esten ubicados en el directorio que apuntamos.

git add <file> → agregamos a staging unicamente el archivo que le indiquemos.

git commit -m “mensaje” → deja listo el commit para pushearlo

git push origin master → enviamos el codigo

git checkout -b juan/etl_test

git branch → para visualizar todas las branch que tenemos en nuestro entorno local.

git push origin juan/etl_test

*branch = rama

Como crear un .gitignore:

Para crear un archivo `.gitignore` que ignore el archivo `config` en tu repositorio de Git, puedes seguir estos pasos:

1. Abre un editor de texto en tu computadora.

2. Crea un nuevo archivo y guarda el archivo con el nombre `.gitignore` (incluyendo el punto al principio).
3. Dentro del archivo `.gitignore`, escribe la siguiente línea:

   ```
   config
   ```

   Esto indicará a Git que ignore cualquier archivo o carpeta con el nombre "config" en cualquier directorio del repositorio.
4. Guarda y cierra el archivo `.gitignore`.
5. Coloca el archivo `.gitignore` en la raíz de tu repositorio de Git.
6. Realiza un commit para agregar el archivo `.gitignore` a tu repositorio:

   ```
   git add .gitignore
   git commit -m "Agregado archivo .gitignore para ignorar el archivo 'config'"
   ```

   Esto agregará el archivo `.gitignore` a tu historial de commits.

A partir de este momento, Git ignorará el archivo `config` y no lo incluirá en los cambios, ni lo rastreará cuando realices operaciones como `git add` o `git commit`.

Recuerda que los archivos y carpetas que ya hayan sido agregados y comiteados en el repositorio antes de crear el archivo `.gitignore` seguirán siendo rastreados por Git. Para dejar de rastrearlos, deberás eliminarlos del historial de commits utilizando herramientas como `git rm --cached`.

Instalación de Requirements:

1. Abre un editor de texto en tu computadora.
2. Crea un nuevo archivo y guarda el archivo con el nombre `requirements.txt`.
3. Dentro del archivo `requirements.txt`, agrega la siguiente línea:

   Comando para escribir automanticamente las dependencias necesarias para desplegar su proyecto: pip freeze > requirements.txt

   ```
   psycopg2
   ```

```
```

Esto indica que deseas incluir el paquete `psycopg2` en tu proyecto.

4. Si tienes otras dependencias o requisitos en tu proyecto, puedes agregarlas en líneas separadas dentro del archivo `requirements.txt`.
5. Guarda y cierra el archivo `requirements.txt`.

A partir de este momento, puedes utilizar el archivo `requirements.txt` para instalar todas las dependencias de tu proyecto, incluyendo `psycopg2`. Para hacerlo, puedes ejecutar el siguiente comando:

```
pip install -r requirements.txt
```

Esto instalará todas las dependencias listadas en el archivo `requirements.txt`, incluyendo `psycopg2`, utilizando el gestor de paquetes `pip`. Asegúrate de estar en el entorno virtual o en el entorno adecuado para tu proyecto antes de ejecutar este comando.

Readme.md

1. Inicia sesión en tu cuenta de GitHub y navega hasta el repositorio donde deseas agregar el archivo `README.md`.
2. Haz clic en el botón "Add file" y selecciona la opción "Create new file".
3. En el campo de nombre del archivo, escribe `README.md`.
4. Dentro del archivo `README.md`, puedes escribir el contenido en formato Markdown para proporcionar información sobre tu script. Aquí tienes un ejemplo básico de cómo estructurar el contenido:

```
# Nombre del Proyecto

Descripción breve del proyecto o del script.

## Requisitos

- Dependencia 1
- Dependencia 2
```

```
## Instalación

1. Paso 1 para instalar el proyecto.
2. Paso 2 para instalar el proyecto.

## Uso

Explica cómo utilizar el script y proporciona ejemplos.

## Contribución

Si deseas contribuir a este proyecto, sigue los pasos a continuación:

1. Haz un fork de este repositorio.
2. Crea una nueva rama para tu contribución.
3. Realiza los cambios y haz commits en tu rama.
4. Envía una solicitud de pull (pull request) a la rama principal.

## Licencia

Indica la licencia bajo la cual se distribuye el script (por ejemplo, MIT, GNU, et
c.).

## Contacto

- [Nombre del autor](mailto:correo@ejemplo.com)
- [Sitio web del proyecto](https://www.ejemplo.com)
```

Puedes personalizar y ampliar el contenido del archivo `README.md` según tus necesidades.

5. A medida que escribas en el archivo `README.md`, verás una vista previa de cómo se verá el contenido en la parte derecha de la pantalla.
6. Una vez que hayas terminado de escribir el contenido, desplázate hacia abajo y verás la sección "Commit new file".
7. Proporciona un título y una descripción para el commit y haz clic en el botón "Commit new file" para guardar el archivo `README.md` en tu repositorio.

El archivo `README.md` ahora estará disponible en la página principal de tu repositorio en GitHub y proporcionará información sobre tu script a los visitantes del repositorio.