CCMASTERS EDUCATION®

Curso de Desarrollo de proyectos geofísicos con Python y GIT

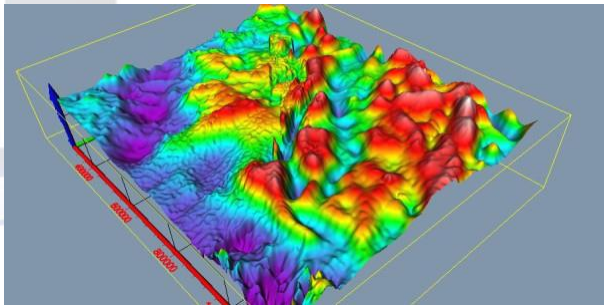
Juan Camilo Mejía Fragoso, Geólogo
Esp. en Geología Minera
Candidato a Magister en Geofísica

CCMASTERS EDUCATION®

Módulo 1

Fundamentos y conceptos básicos

GitHub



CCMASTERS EDUCATION®

Fundamentos de la Terminal, Git y Github

git

CCMASTERS EDUCATION®

¿Qué es la Terminal?

Una interfaz de línea de comandos (CLI) que permite a los usuarios interactuar con el sistema operativo a través de comandos de texto.

Proporciona un método directo para ejecutar comandos y administrar el sistema.

Se puede acceder a través de Command Prompt, PowerShell o Git Bash.

Búsqueda en el menú de inicio o usando el comando o en la caja de ejecución (Win + R).

CCMASTERS EDUCATION®

Importancia de la Terminal

- Permite realizar tareas complejas de manera eficiente.
- Ofrece un control detallado sobre las configuraciones y sistemas del SO.
- Esencial para tareas de administración, resolución de problemas y automatización.
- Facilita tareas desde operaciones básicas de archivos hasta scripting avanzado y administración de sistemas.

Descriptores de rutas

1.Rutas en Windows

- En Windows, las rutas pueden ser absolutas o relativas.
- Las rutas absolutas comienzan con una letra de unidad (como **C:**) seguida del camino al directorio o archivo.
- Las rutas relativas hacen referencia a una ubicación en relación al directorio actual.

2.Sintaxis de la Ruta

- Utiliza barras invertidas (****) en lugar de barras normales (**/**) para separar directorios.
- Ejemplo de ruta absoluta: **C:\Usuarios\ejemplo\Documentos**
- Ejemplo de ruta relativa: **Documentos\NuevoDocumento.txt**

3.Directorio Actual y Cambio de Directorio

- El comando **cd** se utiliza para cambiar el directorio actual.
- **.** representa el directorio actual y **..** el directorio padre.

4.Caracteres Especiales en Rutas

- Los caracteres como ***** y **?** se pueden utilizar como comodines en rutas.
- Ejemplo: **C:\Usuarios*\Documentos** para referirse a los documentos de todos los usuarios.

output 02/01/2021 12:40 Text Document 1 KB

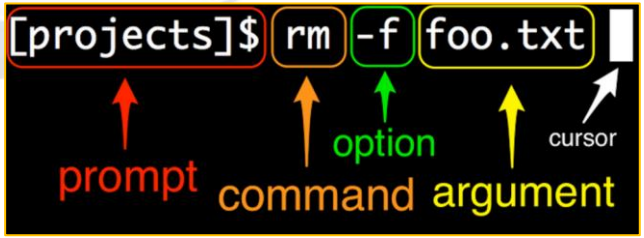
C:\Windows\system32\cmd.exe

```
C:\Users\Wietze\Documents>certutil -"f" /u r"c a ch"e -Asep"lei"t h"t"tp"s:"/"/wi"etze.github.io/robots.txt output.txt

****  Online  ****
  0000  ...
  0068
CertUtil: -URLCache command completed successfully.

C:\Users\Wietze\Documents>
```

- **Prompt**: Es el texto que precede al lugar donde el usuario escribe el comando, generalmente proporciona información como el directorio actual o el usuario que está logueado.
- **Command**: Es el comando que se va a ejecutar, en este caso rm, que en un sistema Linux se utiliza para eliminar archivos o directorios. En caso de Windows se usa “del”.
- **Option**: Son modificadores del comando que cambian su comportamiento, -f es una opción que fuerza la eliminación sin pedir confirmación.
- **Argument**: Es la entrada sobre la cual el comando actúa, aquí foo.txt sería el archivo que se está intentando eliminar.



Atajos de teclado básicos

- **Ctrl + C** Cancela el comando actual o detiene la ejecución de un programa.
- **Ctrl + Z** Marca el final de la entrada de un archivo o comando.
- **Ctrl + A** Selecciona todo el texto en la línea actual de la entrada.
- **Ctrl + V o Shift + Insert** Pega el texto del portapapeles en la terminal.
- **Arriba y Abajo (Flechas)** Navega a través del historial de comandos que has introducido.
- **Tab** Autocompleta los nombres de archivos y carpetas. Si hay múltiples coincidencias, seguir presionando repetirá la autocompletación.
- **Ctrl + Flecha Izquierda/Derecha** Mueve el cursor una palabra a la izquierda o a la derecha.
- **Ctrl + R** Busca un comando anterior escribiendo parte de él (solo en PowerShell).

Operaciones básicas con directorios

1. **Ver Directorio Actual: cd**
   - Muestra el directorio en el que te encuentras actualmente.
2. **Cambiar de Directorio: cd [ruta]**
   - Permite navegar a otro directorio. Ejemplo: **cd Documents** cambia al directorio Documents.
3. **Listar Contenidos de un Directorio: dir**
   - Muestra los archivos y subdirectorios dentro de un directorio. Puede combinarse con opciones para personalizar la salida.
4. **Crear un Nuevo Directorio: mkdir [nombreDirectorio]**
   - Crea un nuevo directorio en la ubicación actual. Ejemplo: **mkdir NewFolder** crea una carpeta llamada NewFolder.
5. **Eliminar un Directorio: rmdir [nombreDirectorio] / del [nombreArchivo]**
   - **rmdir** elimina un directorio vacío, mientras que **del** elimina archivos. Opciones adicionales permiten eliminar directorios con contenido.

Manipulación de archivos

1. **Crear un Archivo: copy con [nombreArchivo]**
   - Crea un nuevo archivo en blanco. Finaliza con Ctrl + Z.
2. **Copiar Archivos: copy [origen] [destino]**
   - Duplica archivos de una ubicación a otra. Ejemplo: **copy example.txt D:\Backup**
3. **Mover o Renombrar Archivos: move [origen] [destino]**
   - Mueve archivos de un directorio a otro o los renombra. Ejemplo: **move example.txt ..\Documents**
4. **Eliminar Archivos: del [archivo]**
   - Elimina uno o más archivos de manera permanente. Ejemplo: **del example.txt**
5. **Buscar Archivos: dir [nombreArchivo] /s**
   - Busca archivos dentro del directorio actual y subdirectorios. Ejemplo: **dir example.txt /s**

Operadores de control

1. **Redirección de Salida: > y >>**
   - **>** redirige la salida de un comando a un archivo, sobrescribiéndolo.
   - **>>** redirige la salida a un archivo, añadiéndola al final. Ejemplo: **dir > lista.txt** guarda la lista de archivos en **lista.txt**.
2. **Redirección de Entrada: <**
   - **<** toma el contenido de un archivo como entrada de un comando. Ejemplo: **sort < desordenado.txt > ordenado.txt**
3. **Piping: |**
   - Envía la salida de un comando como entrada a otro. Ejemplo: **dir | find "example"** busca la palabra "example" en la lista del directorio.
4. **Operadores de Control: && y ||**
   - **&&** ejecuta el segundo comando solo si el primero es exitoso.
   - **||** ejecuta el segundo comando solo si el primero falla. Ejemplo: **del archivo.txt && echo Archivo eliminado**

Comandos avanzados y utilidades

1. **PowerShell Cmdlets**
   - Utiliza cmdlets avanzados en PowerShell para tareas de administración y automatización.
2. **Tasklist y Taskkill**
   - **tasklist** muestra los procesos en ejecución.
   - **taskkill** termina procesos por ID o nombre.
3. **SFC (System File Checker)**
   - Ejecuta **sfc /scannow** para escanear y reparar archivos del sistema.
4. **DISM (Deployment Image Service and Management Tool)**
   - Herramienta para reparar y preparar imágenes de Windows.
5. **Schtasks**
   - Programa tareas para que se ejecuten automáticamente a una hora específica.
6. **Net Use**
   - Conecta, desconecta y muestra recursos de red (como unidades de red).
7. **Robocopy**
   - Herramienta robusta para copiar archivos, con capacidades de reanudación y manejo de errores.

CCMASTERS EDUCATION®

¿Qué es Git?

- Git es un sistema de control de versiones que originalmente fue diseñado para operar en un entorno Linux. Actualmente Git es multiplataforma, es decir, es compatible con Linux, MacOS y Windows.

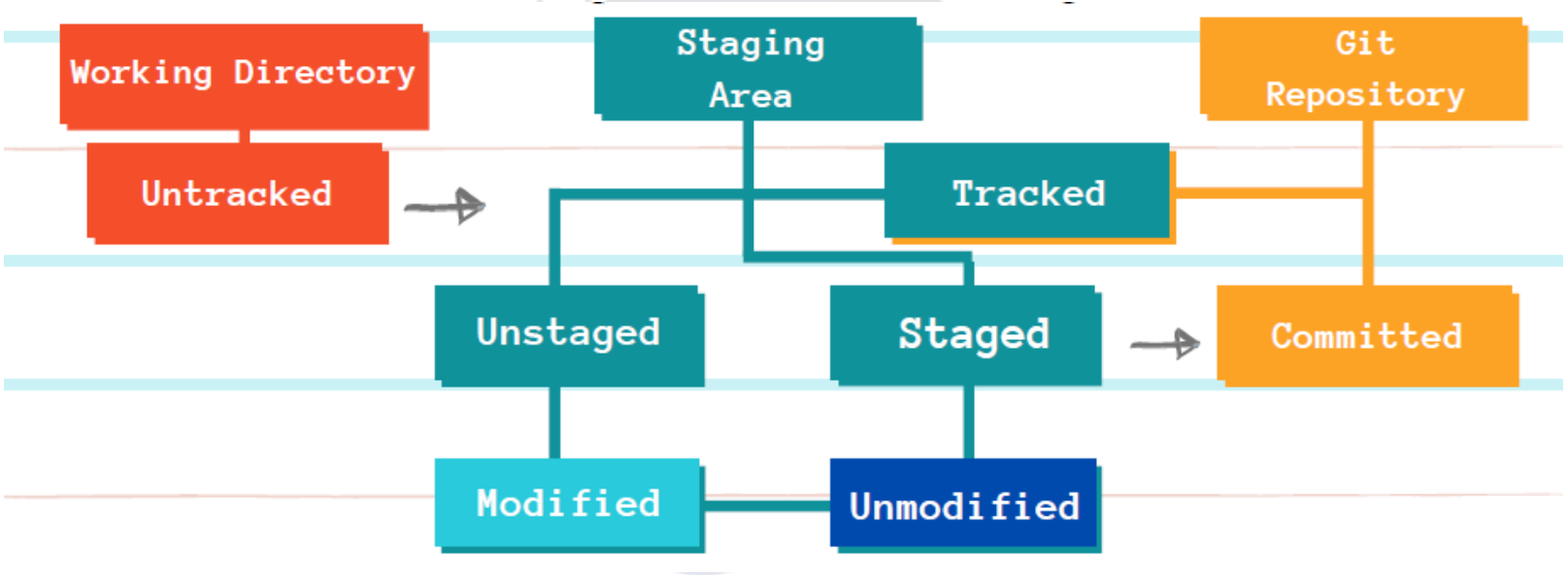
Almacena la información como un conjunto de archivos

No existen cambios sin que Git lo sepa

Casi todo en Git es mediante recursos locales

Git cuenta con tres estados: Staged, Modified y Committed

Flujo de trabajo en Git



Comandos básicos

- Instalación: Descarga e instalación de Git desde su sitio web oficial.
- Configuración: Establecimiento del nombre de usuario y correo electrónico con **git config**.
- Comandos Básicos: **git init** crea un nuevo repositorio; **git add [archivo]** añade archivos al área de preparación; **git commit -m "[mensaje]"** guarda los cambios con un mensaje descriptivo. **git status** permite conocer el estado del repositorio.

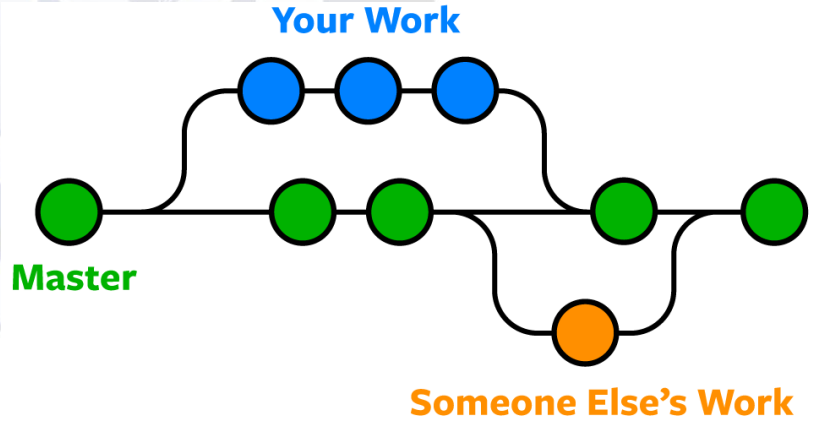
Trabajar con dispositivos remotos

- Conexión con Repositorio Remoto: Uso de **git remote add origin [URL]** para vincular el repositorio local con uno remoto.
- Push y Pull: **git push origin master** sube cambios al repositorio remoto; **git pull origin master** actualiza el repositorio local con los cambios remotos.
- Resolución de Conflictos: Estrategias para resolver conflictos cuando ocurren tras un **git pull**.

Branching y Merging

•Creación de Ramas: **git branch [nombreRama]** para crear una nueva rama; **git checkout [nombreRama]** para cambiar de rama.

•Merging: Uso de **git merge [nombreRama]** para combinar cambios de una rama a otra, y cómo resolver conflictos de merge manualmente si ocurren.



¿Y entonces qué es Github?

- GitHub es un servicio de alojamiento que ofrece a los desarrolladores repositorios de software usando el sistema de control de versiones, Git.

Permite alojar proyectos de forma gratuita (freemium)

Puedes compartir tus proyectos de forma sencilla

Permite colaborar para mejorar los proyectos de otros

Es la opción ideal para trabajar en equipo en un mismo proyecto