# **Comandos básicos de la terminal y sus aplicaciones:**

La terminal es una herramienta muy útil que nos permite interactuar con las distintas funcionalidades del sistema operativo que estemos usando a través de la línea de comandos (**CLI**

**Comando:** Un comando es una instrucción que el usuario proporciona a la línea de comandos o terminal para realizar una acción en específico o ejecutar un proceso.

**<comando> [--opciones . . . ] [ argumentos . . . ]**

Algunos comandos básicos e importantes que se deben tener en cuenta cuando se esta usando la terminal son:

**ls [--opciones . . . ] [ argumentos . . . ]:**

Este comando, nos permite listar los elementos presentes en la ruta que se especifique como argumento, desde archivos y directorios, hasta elementos ocultos y links de archivos.

Algunas opciones que pueden usarse sobre el comando **ls**, pueden ser:

* + **-a:** Lista los elementos incluso los que están ocultos.
  + **-l:** Lista los elementos de manera vertical en formato de lista.
  + **-h:** Lista los elementos con su información de manera mas legible.
  + **-S:** Lista los elementos por orden de fecha de modificación.
  + **-R:** Lista los elementos de manera recursiva (elementos dentro de elementos)

**cd [--opciones . . . ] [ argumentos . . . ]:**

Este comando, significa “change directory”, permite acceder a los directorios del sistema especificándolos como argumento, en caso de que no existan se genera un error.

**pwd [--opciones . . . ] [ argumentos . . . ]:**

Este comando significa “print working directory”, y muestra por pantalla el directorio o ruta actual en la que nos encontramos en el sistema.

**touch [--opciones . . . ] [ argumentos . . . ]:**

Este comando, nos permite crear un archivo vacio con el nombre que le pasemos como argumento, en la ubicación que le especifiquemos.

**rm [--opciones . . . ] [ argumentos . . . ]:**

Este comando permite eliminar elementos del sistema, desde archivos hasta carpetas, especificándolos como argumento.

**mkdir [--opciones . . . ] [ argumentos . . . ]:**

Este comando significa “make directory” y nos permite crear nuevos directorios en la ruta que le indiquemos, y el nombre que le pasemos como parámetro.

**rmdir**  **[--opciones . . . ] [ argumentos . . . ]:**

Este comando significa “remove directory” y nos permite eliminar directorios en la ruta que le indiquemos, y el nombre que le pasemos como parámetro, pero solo los elimina si están vacíos.

**cp [--opciones . . . ] [ argumentos . . . ]:**

Este comando nos permite copiar elementos a través del sistema, y hacia la ruta que le indiquemos como argumento

**mv [--opciones . . . ] [ argumentos . . . ]:**

Este comando nos permite mover elementos a través del sistema. También puede servir para renombrar elementos moviéndolos a la ubicación actual con un nombre diferente que se quiera.

**clear:**

Este comando nos permite limpiar el contendido que se encuentre en pantalla dejándola libre.

# **Comandos básicos usados en Git y GitHub:**

El sistema de control de versiones conocido como GIT, se presenta y se usa sobre la interfaz de línea de comandos (CLI) o terminal, lo que nos permite tener control sobre el código fuente que estemos manipulando o usando, para esto, se presentan varias funcionalidades en su mayoría comandos, y que van precedidas por el comando o palabra **git,** para hacer referencia al uso del software, algunos de los más usados son:

**git:** Este comando nos permite conocer todas las opciones que se pueden llegar a usar con el sistema de control de versiones de **git.**

**git help:** Este comando nos permite conocer las opciones que nos ofrece **git** y también la usabilidad que le podamos dar a cada una.

**git init:** Este comando es usado para inicializar un repositorio en la ruta en donde se ejecute y a partir de ese momento, se va a comenzar con el rastreo y cambios en archivos en tiempo real.

**git status:** Este comando nos permite saber el estado en que se encuentra nuestro repositorio actual, y las acciones correspondientes a realizar, puede mostrar los archivos nuevos, los cambios, las eliminaciones, adiciones, y más.

**git log:** Este comando nos permite conocer un historial de commits que han ocurrido en el sistema, y se puede filtrar en cómo es la información de salida correspondiente con opciones adicionales.

**git add:** Este comando nos permite agregar los cambios y nuevos archivos para que sean rastreados por **git** en el repositorio actual del sistema. Este comando puede agregar tanto elementos individuales como grupos de elementos, e incluso todo el contenido del directorio de manera recursiva si así lo amerita, esto se hace especificando con un punto ( . ) para hacer referencia al directorio actual.

**git commit:** Este comando, nos permite confirmar los cambios y es una especie de referencia a los cambios que se realizaron los cuales se identifican por un id único o también conocido como hash, el cual entiende **git**, de manera interna

**git config:** Este comando nos permite acceder a la configuración de git, mostrando las diferentes opciones que se pueden usar con el mismo, desde configuración de uso, hasta de personalización.

**git config -list:** Este comando nos permite listar la configuración que tenga **git** en el sistema, y puede ser accedida desde el archivo **.gitconfig** ubicado en el root del sistema que se este usando.

**git config user.name <nombre>:** Este comando nos permite definir el nombre de usuario asociado al repositorio que se use, si se quiere establecer de manera global para todo el sistema se debe usar la opción adicional **–global,** luego de **config**

**git config user.email <email >:** Este comando nos permite definir el email asociado al repositorio que se use, si se quiere establecer de manera global para todo el sistema se debe usar la opción adicional **–global,** luego de **config.**

**git remote add:** Este comando nos permite agregar un repositorio remoto a nuestro sistema local. Este por lo general especifica el nombre que tendrá el repositorio en el sistema para ser reconocido (generalmente es **origin**), también puede obtener los repositorios a través de varias formas y protocolos, usando tanto **HTTP,** como **ssh.**

**git remote -v:** Este comando nos permite listar los repositorios remotos que tengamos en nuestro sistema, y adicionalmente nos muestra la forma en que han sido agregados.

**git pull [repo-name] <rama>:** Este comando nos permite traer los cambios que se hayan hecho en el repositorio remoto hacia el sistema local.

**git pull [repo-name] <rama>:** Este comando nos permite mandar los cambios que se hayan hecho en el sistema local hacia el repositorio remoto.

**git clone <repo-route>:** Este comando nos permite clonar un repositorio existente en nuestro sistema, permitiendo hacer cambios sobre el mismo.