Cheatsheet

Comandos

Instrucciones codificadas para ser interpretadas por un sistema operativo.

$$Comandos = Argumentos + Opciones$$

Comandos básicos para la navegación e interacción con carpetas

De tipo informativo	Crear o eliminar carpetas	Crear o eliminar archivos	Copiar archivos y directorios
pwd/ Mostrar el nombre de la carpeta en el que uno se encuentra situado (print working directory). **cd /** Cambiar la carpeta de trabajo: Con este comando nos podemos mover entre diferentes directorios. Si queremos ir a un directorio en particular. **ls /** Listar el contenido de directorios (list): Este comando lista los ficheros y carpetas. **ls - t /** Ordena los archivos según la fecha de modificación donde el primer archivo será el que se modificó más recientemente.	**mkdir /** Crear un carpeta (make directory). **rmdir /** Borrar un carpeta (remove directory)	**gedit /** Crear y editar archivos de texto. **rm /** Borrar archivos.	**cp /** Copiar un archivo o carpeta en el directorio especificado (copy). **cp -r /** Copiar carpetas. **Lmv /** Mover un archivo o carpeta a un archivo o carpeta (move)

¿Cómo indicar una ruta, directorio o archivo?

	Ruta Absoluta	Ruta Relativa
Definición	Ruta exacta de donde se encuentra una carpeta, archivo o directorio.	Damos por sentado que estamos diciendo la ubicación desde donde estamos parados
Ejemplo	**pwd** Mostrar toda la ruta completa hasta el directorio donde nos encontramos.	**cd**
Como se diferencia	Se usa **/** al comienzo de la ruta	No se utiliza la barra

¿Qué es Git Bash?

Aplicación de terminal que se utiliza como interfaz con un sistema operativo mediante comandos escritos (CLI).

Comandos de la terminal Bash

Mostrar	
$ ls	Lista archivos en el directorio o carpeta
$ ls -a	Lista todos los archivos, incluyendo los archivos ocultos
$ ls -l	Muestra toda la información de una carpeta: usuario, grupo, permisos, tamaño, fecha y hora de creación.
$ ls -R	Muestra las carpetas y los archivos contenidos en ellos de manera recursiva
$ pwd	Muestra la carpeta en la que se está trabajando actualmente
more [Nombre del archivo]	Muestra el contenido de un archivo

Crear	
$ mkdir [Carpeta]	Crea una nueva directorio o carpeta
$ touch [Nombre del archivo]	Crea un nuevo archivo

Eliminar	
$ rm [Nombre del archivo]	Elimina un archivo
$ rmdir [Nombre de la carpeta]	Elimina una carpeta vacía
$ rm -r [Nombre de la carpeta]	Elimina una carpeta y su contenido

Copiar/Mover/Renombrar	
$ mv [ruta/archivo1] [ruta/archivo2]	Renombra archivos (archivo2 no debe existir o será sobreescrito)
$ mv [ruta/carpeta1] [ruta/carpeta2]	Renombra la carpeta1 como carpeta2 (carpeta 2 no debe existir)
$ mv [ruta/carpeta1] [ruta/carpeta2]	Mueve contenido de carpeta1 a carpeta2 (carpeta 2 debe existir)
$ cp [ruta/archivo1] [ruta/archivo2] $ cp [ruta/archivo1] [ruta/archivo2]	Copia un archivo o carpeta
opción: -r	Indica que copie recursivamente el contenido de las subcarpetas

Navegación entre carpetas	
$ cd	Sube un nivel de carpeta
$ cd	Cambia de carpeta
$ cd/chosen/ directory	Cambia a una carpeta específica

Otros comandos	
$ clear	Limpia la pantalla de la terminal
$ comando --help	Muestra ayuda del comando

Atajos de teclado	
$ ctrl + c	Finaliza un proceso vigente que está corriendo en la terminal
$ ctrl + l	Limpia la pantalla de la termina

Caracteres especiales	
“” (comillas)	Nos permiten utilizar términos que consistan en más de una palabra
. (el punto)	Permite hacer referencia al directorio donde estamos ubicados actualmente

Notas:

1. Para llamar una carpeta de un nivel superior al que me encuentro ../Nombre de la capeta
2. Para mover varios archivos a una carpeta se deben listar los archivos y al final poner la ruta o el nombre de la carpeta. Para que no salga error la carpeta ya debe existir.
3. Para modificar un archivo desde GIT Bash / Terminal Mac - Linux usar comando $nano. Para modificarlo desde VS Code solo utilizar la parte superior del portal.
4. Se puede usar la tecla Tab para autocompletar el nombre de las carpetas o archivos. Limitación: solo llama a las carpetas o documentos que se encuentran en la carpeta donde estoy ubicada.
5. Es posible mover documentos de otra carpeta a la carpeta en la que estoy ubicada actualmente:

   ```
   $mv ../nombrecarpetainicio/"nombre archivo.txt" .
   ```

6. Con el comando ls se pueden poner varias flags a la vez, es decir, es posible correr ls -aRs

Búsqueda en la terminal

Find y grep

	Función	-name	-size	-i	.(punto)	-n	-r	“” (comillas)
Find	Busca archivos	Para buscar	Buscar por	Para que no	Permite buscar a			Permite n

	o carpetas, podemos hacerlo por nombre completo o que coincidan con ciertas letras o frases, archivos de ciertos usuarios, creados en cierta fecha y un montón de opciones más.	por nombre	tamaño de archivos	se tenga en cuenta la sensibilidad a mayúsculas.	partir del directorio donde estamos ubicados actualmente.			buscar un término que consista en más de una palabra.
Grep	Nos permite buscar texto en los archivos, puede ser el nombre de una funcionalidad que hemos diseñado, el de una variable que hayamos definido o cualquier tipo de texto.			Para que no se tenga en cuenta la sensibilidad a mayúsculas.	Permite buscar a partir del directorio donde estamos ubicados actualmente.	Nos muestra la línea donde encuentra la coincidencia dentro de el/los archivo/s.	Buscar en todas las subcarpetas recursivamente.	Permiten buscar un término que consista en más de una palabra.

Git

1. Introducción a Git

Software de control de versiones

Mantiene eficientemente las actualizaciones sobre el código fuente.

Objetivo

Llevar registro de los cambios de los archivos y coordinar el trabajo que varias personas realizan sobre estos.

2. Creando nuestro primer repositorio local

Repositorio

Es un almacén de archivos donde se irán almacenando los documentos en pequeños paquetes llamados **commits.** Estos paquetes permiten hacer un seguimiento de los cambios que se van realizando en el proyecto ya que cada uno tiene un registro de la fecha de creación y del autor.

Commits

Historial de cambios que se fueron haciendo en el proyecto.

¿Cómo hacer un backup de una carpeta en git?

1. Crear un repositorio local en la carpeta en la que tenemos los archivos.

```
$ git init
```

Nota

Git no hace seguimiento en carpetas que están vacías.

2. Decirle al repositorio mi identidad

```
$ git config user.name "mi-usuario-de-GitHub"
$ git config user.email "micorreo@email.com"
```

Notas

- Para verificar el usuario que está identificado usar el comando

```
$ git config user.name
$ git config user.email
```

- Para configurar tu usuario como predeterminado para todos los repositorios

```
$ git config --global user.name "mi-usuario-de-GitHub"
$ git config --global user.email "micorreo@email.com"
```

3. Agregar archivos al repositorio
   Para agregar archivos al repositorio ya se debe haber inicializado un repositorio local. Para agregar archivos debemos usar:

   - Para subir algunos archivos
     ```
     $ git add nombre-de-los-archivos
     ```
   - Para subir todos los archivos
     ```
     $ git add .
     ```

   Para ver el status de los archivos y su estado de estos respecto al repositorio

```
$ git status
```

Untracked files (U)	Archivos sin seguimiento. Aparecen en rojo

Changes to be committed (A)	Cambios listos para el commit. Aparecen en verde

Nota

Si un archivo fue modificado git lo identifica como un archivo nuevo, por lo tanto, su status volverá a ser *untracked* (control de versiones). Por lo tanto, es importante que despues de modificar un archivo volvamos a ejecutar git add.

4. Crea el commit

Commit

Confirmación en la que le estamos diciendo al repositorio que los archivos que fuimos agregando los deseamos oficialmente como un paquete de modificaciones que tendrán una marca de tiempo y un autor.

> Generan puntos cronológicos en la línea de tiempo de un proyecto que nos permiten identificar el estado del mismo hasta ese momento especifico y volver sobre las versiones si es que así lo necesitamos.

Para crear un commit deben haberse agregado previamente los archivos (git add) y luego usar el comando:

```
$ git commit -m "Breve-descripción-de-las-modificaciones-al-proyecto"
```

Para ver el historial de los commits

```
$ git log
```

3. GitHub

Plataforma colaborativa que nos va a permitir llevar un control de versión sobre nuestro código. Es un lugar en la nube donde se hospedan proyectos de programación.

Repositorio

Lugar donde se irán almacenando los archivos de un proyecto y a través del cual podemos hacer seguimiento de los mismos. Los repositorios de GitHub se les denomina *repositorios remotos* porque se encuentran ubicados en la nube.

> ≠ *Repositorios locales:* copia de un repositorio almacenada en nuestra propia computadora.

4. ¿Cómo conectar el repositorio local a un repositorio en GitHub?

- Creo un repositorio en GitHub
- Copia el URL del repositorio
- Ejecuta el siguiente comando y pega en URL del repositorio

```
$ git remote add origin https://link-del-repositorio-de-GitHub.com/
```

Nota

Para verificar que todo salió bien y que los repositorios fueron vinculados se puede correr el comando

```
$git remote -v
```

5. Subir archivos al repositorio remoto

Para subir los archivos a GitHub todo debe estar commiteado. Ahora, para subirlos debemos

usar el comando:

```
$ git push origin main
```

Ramas

Una rama dentro de un repositorio hace referencia a una copia alternativa del mismo. En esta se pueden agregar nuevas funcionalidades sin tener que modificar la línea original del tiempo.

6. Bajando archivos al repositorio local

Para bajar archivos del repositorio remoto usamos el comando:

```
$ git clone http://URLexactadelrepositiriodondeestanlosarchivos.com/
```

Este comando permite hacer una copia exacta en la computadora de todos los archivos existentes en un repositorio remoto.

> Nota
> Debe ejecutarse solamente una vez. La primera vez en la que quieren descargarse los archivos de git y cuando estos no existan ya en la computadora.

Como mantener los archivos sincronizados

Para actualizar los archivos que ya existen en nuestra computadora con versiones modificadas y cargadas en GitHub debemos usar el comando:

```
$ git pull origin main
```

7. Resolviendo conflictos

Conflicto

Se generan cuando un archivo que quieres pushear a git ya fue modificado y pusheado por otro miembro del equipo.

¿Cómo resolver un conflicto?

1. Traer los cambios realizados por el otro usuario.

   $ git pull origin main

   El resultado tendrá un error que básicamente dice que git intento unificar los dos documentos pero no logro. Al abrirlo en nuestro editor de código veremos lo siguiente por cada conflicto que exista en el archivo:

   ```
   <<<<<<< HEAD
     Mis otros cambios
   =======
   Mis cambios
   >>>>>>> 3216f3fd5ca65cfd3252ae76808d8f659a715fa6
   ```

2. Por cada conflicto se debe tomar una decisión:
   i. Dejar mi parte.
   ii. Dejar la parte del otro usuario.
   iii. Combinar las dos partes y hacer una nueva versión del documento.

3. Realizar un nuevo commit como si fuera una modificación del archivo y pushearlo en GitHub

Nota

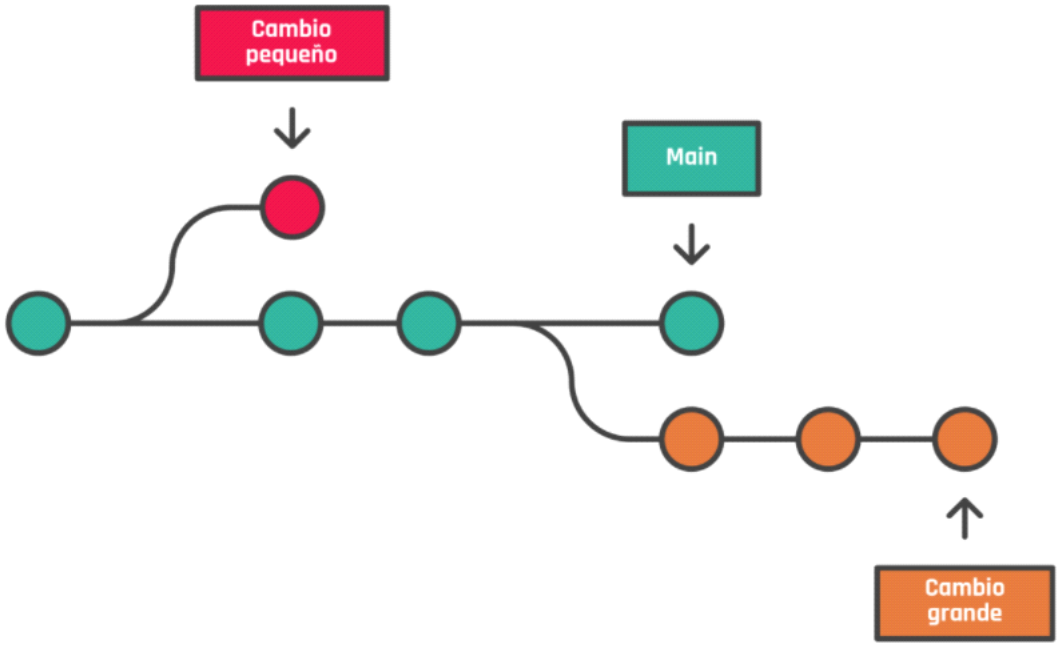
Una alternativa para minimizar los conflictos es usar ramas alternativas de forma que cada persona este modificando una rama diferente y no la rama original.

8. Ramas de Git

Rama

Una ***rama*** es solo una línea de desarrollo que tiene como base el repositorio, pero que no actúa directamente sobre él, o sea que no lo modifica. Al crear ramas lo que generamos son versiones del repositorio donde están los archivos del mismo más los cambios que le iremos sumando.

El repositorio que creamos con Git o clonamos de GitHub tiene por defecto una rama raíz llamada ***Main*** en la cual vamos guardando las versiones de nuestro trabajo cada vez que hacemos un commit.



Branch

Una ***branch*** funcionará como un espacio de trabajo solo para nosotros, para que guardemos nuestros cambios y hagamos todas las pruebas necesarias hasta que estemos seguros de socializar esos cambios.

Git branch

`git branch`	Enumera todas las ramas de tu repositorio, es similar a git branch --list.
`git branch <branch>`	Crea una nueva rama llamada <branch>.
`git branch -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 branch -D <branch>`	Fuerza la eliminación de la rama especificada, incluso si tiene cambios sin fusionar.

Git checkout

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

```
$ git checkout nombre_rama
```

Generalmente, Git solo permitirá que nos movamos a otra rama si no 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 comandos: git add, git commit, git status y git log. Y cuando queramos subir esos cambios usamos el comando:

```
$ git push origin <branch>
```

> Nota
> Para traer los cambios de una rama utilizamos el comando git pull agregando desde donde queremos traer los cambios:
>
> ```
> $ git pull origin <branch>
> ```