# Guía de Comandos

## *Comandos de Configuración*

### Mkdir

***Crea un directorio nuevo tomando en cuenta la ubicación actual. Crear el repositorio Git***

### Git Config --local User.name "Nombre y Apellido"

### Git Config --local user.email"ejemplo@mail.com"

***Para configurar mi repositorio con mi nombre de usuario y correo,la configuración puede variar de Local a Global.***

## *Comandos de Creación*

### Git init

***Para iniciar un repositorio***

## *Comandos de Estado*

### Git Status

***Nos muestra el estado de los archivos que han cambiado***

### Git Log

***Nos muestra el historial de nuestros commit***

### *Git log --oneline*

***Nos muestra una lista ordenada de los commit. Sólo nos maostrará los commit de la rama en la que estemos ubicados.***

## *Git log --oneline --all*

*Me muestra una lista de todos los commit y todas las ramas*

## *Git log --oneline --all --graph --decorate*

*Nos muestra una lista de los commit pero con los dibujos de los commit como asteriscos y las líneas de nuestras ramas*

## Git Diff

*Nos muestra las modificaciones realizadas*

## Git Diff --Stat

*Nos muestra un resumende las modificaciones realizadas*

## Git Diff --numstat

*Nos muestra sólo de forma numérica las modificaciones en nuestro proyecto*

# *Comandos de Cambios:*

## Git Add

*Añade los cambio y modificacioneas a nuestra área de preparación (staging área)para luegoser confirmados.*

## *Git Add "nombre del archivo"*

*Para añadir archivos específicos por el nombre.*

## *Git Add .*

***El punto después del comando Add hace referencia a todo, osea que añadirá todos los nuevos cambios o modificaciones***

## Git Commit

***Es el comando para "confirmar" los cambios añadidos y preparados con el comando add en el staging area, al ejecutar el comando se guarda la versión en el repositorio y se pide que se guarde un mensaje que describa los cambios guardados en el archivo confirmado. Después de hacer el comando git commit, se debe escribir un mensaje descriptivo del commit. Se puede acelerar el proceso escribiendo el mensaje en el comando del commit de lasiguiente manera:***

## Git commit (Se abre el editor que trae por defecto Git) Vim

***En la primera línea escribiremos el mensaje de nuestro commit, luego saldremos con Esc : wq! O con ctrol S Ctrol X***

## Git commit –m "mensaje"

***Se guarda en el repositorio el archivo y se le añade un mensaje***

## Git commit –am "mensaje"

***Se guardan todos los archivos preparados y se añade un comentario al commit***

## Git Checkout "n° hash"

***Nos permite viajar en el tiempo en los diferente commit de nuestro proyecto***

## Git show

*Muestra información del commit específico solo cuando utilizo la etiqueta con comentario.*

## Git stash

*Me permite generar un campo temporal en el que se va a guardar el progreso de nuestro proyecto. Hacemos la modificación que el cliente pide y ya luego podemos continuar en donde lo habíamos dejado.*

## Git stash pop

*Recuperaremos las modificaciones en las que estábamos trabajando y podremos seguir trabajando en ellas.*

# *Comandos de Ramas:*

## Git Branch

*Nos permirte ver las ramas del repositorio*

## Git Branch "nombre rama"

*Nos permite crear una nueva rama*

## Git branch -d [nombre de la rama]

*Borramos la rama indicada, esto es necesario despúes de que ya fusionamos y terminamos de usar la rama auxiliar*

## Git branch –D "nombre rama"

*Elimina la rama que estamos indicando y que no vamos a utilizar. Siempre debemos estar ubiados en la rama Master*

## Git branch -m (nombre actual de la rama "Master") (nuevo nombre "Main")

*Cambia nombre de la rama Master a Main*

## Git checkout

***Este comando te permite crea ramas y te ayuda a navegar entre ellas. Por ejemplo, el siguiente comando crea una nueva y automáticamente se cambia a ella. Tambien nos permitia viajar en los diferentes commit.***

### Git Checkout -b "nombre de la rama"

***Nos permite crear una nueva rama***

### Git Checkout "nombre rama"

***Nos permite movernos de una rama a otra***

## Git merge

***Nos permite la fusión de nuestras ramas***

## Git merge --abort

***Permite eliminar la fusión de ramas en caso de no poder solucionar un conflicto o arreoentirnos de la fusión***

## *Comandos de Control o Seguimiento:*

### Git Tag

***Nos muestra la lista de las versiones del proyecto***

### Git Tag "Versión"

*Me permite crear etiquetas para versionar un proyecto*

**Git tag -a <verison> - m "esta es la versión x"**

**Me permite agregar una etiqueta con comentario**

**Git tag –d (nombre de la versión)**

*Me permite eliminar una etiqueta*

## *Banderas de Estados de Git:*

*Cuando vemos el estado de los archivos en git, podemos encontrar las siguientes banderas . Podemos visualizarlas en Git Diff.*

**M: Modified**

**C: Copy edit**

**R: Rename edit**

**A: Added**

**D: Deleted**

**U: Unmerged**

## *Sintáxis de Git:*

**pwd**

*nos muestra la carpeta actual en la que nos encontramos.*

**cat**

*nos permite ver el contenido de un archivo.*

**cd**

*nos permite cambiarnos de carpeta.*

**cd ..**

*nos permite regresar al directorio o carpeta anterior.*

**cd -**

*nos lleva directamente al ultimo directorio visitado*

## ls

*nos permite ver los archivos de la carpeta donde estamos actualmente.*

## ls –l

*Ver todos los archivos como una lista en donde incluye el usuario, grupo, permisos sobre el archivo, tamaño, fecha y hora de creación.*

## clear

*nos permite limpiar la pantalla.*

## Head >>

*Es el puntero o posición donde estamos ubicados*

## Tres Estados de Git: