Cambiar nombre de usuario en git: $ git config –global user.name "Tiranoexe"
Cambiar mail de usuario en git: $ git config –global user.email .ªzulpablin@gmail.com"
Ver que el nombre y mail estan puestos $ git congif –list
Subir un archivo
git status git add "file.html"git commit -m .Ests"te es un commit"
Como editar un archivo al que ya hice un commit
abrir archivo en VScode para poder editarlo: $ code file.html
editarlo guardarlo git status deberia dar modified: file.html En caso de ser un solo archivo, hacemos un add: git add file.html si son mas de 1 archivo git add . git status muestra que ya quedaron añadidos los archivos modificados ahora si el commit git commit -m "mensaje nuevo con los cambios"
Ver historia del archivo $ git log Lo que aparezca mas arriba es la version mas reciente
Usando $ git log –stat se ven los cambios especificos que se hicieron en un archivo a partir del commit
Con la letra Q salgo en caso de que se muestren cambios mas largos de lo que se pueda ver en pantalla.
Ver los cambios del archivo (Solo muestra los cambios entre el archivo viejo y el nuevo, no todo el historial) $ git show file.html
Comparar cambios de una version con otra a eleccion $ git diff (numero del commit) (numero del commit con el que se quiere comparar) A estos numeros se los saca al hacer git -log file.html- La primera deberia ser la version original o la mas vieja, para que los cambios tengan sentido. La segunda es la version nueva o una mas reciente que la primera, asi se nota bien si algo fue agregado o eliminado.
Volver a un archivo anterior $ git reset (numero del commit) –hard **TODO** vuelve al estado del commit al que lo reseteamos, volviendo a ese commir la nueva **HEAD** $ git reset (numero del commit) –soft Hace que tambien se vuelva a la version anterior deseada, pero lo que esta en *Staging* sigue ahi, por ende, disponible para el proximo Commit

En un directorio en donde estan los archivos del proyecto, por ejemplo, el directorio Html, del proyecto Responsive Coffee shop", en donde estan los archivos: index.html, style.css y script.js junto con la carpeta images

Al entrar por consola en el archivo index.html y escribimos el comando **git init** pasan 2 cosas: Se crea un area en memoria RAM llamada *Staging*, un area completamente desconectada que es donde se van agregando los cambios Se crea un *Repositorio*, la carpeta /.git/, en donde se encuentran todos los cambios al final del proyecto.

Una vez realizados los cambios al archivo, se lo agrega al mismo a la *Staging Area* usando el comando Git Add index.html. En este momento el archivo queda a la espera de ser agregado en el *Repositorio*, mientras se pueden agregar otros archivos o se puede remover este archivo usando el comando Git rm. Usando el

comando Git commit -m "mensaje indicando los cambios en el commit" el archivo se mueve al *Repositorio*, llamado *Master*, en donde se encuentran todos los cambios realizados.

Estados del archivo Antes de utilizar el Git Add, el archivo esta sin rastrear, *Untracked*,luego del Add, el archivo entra al estado *Tracked* para luego irse al *Staging*. Al usar Git Commit -m los cambios pasan de estar trackeados en Staging a estar trackeados en el *Repositorio*, y aqui es donde se le da a cada cambio los indicadores del commit (la serie de numeros y letras que identifica a cada cambio en el archivo visto en Git Log)

Desde la rama *Master*, puedo usar "git checkout (numero de commit) file.html"para traer los cambios que desee hacia mi carpeta. En caso de hacer un commit luego de esto, se borraran todos los cambios posteriores a la version de commit actual. Se puede hacer "git checkout master file.html"se vuelve a la version **Master** del archivo, la ultima version de la que se habia hecho un commit.

¿Que pasa cuando estoy realizando cambios en un archivo pero no quiero mandarlo al repositorio principal? o cuando quiero crear una rama de desarrollo para luego unirla a la rama maste? O en caso de que 3 personas trabajen en el desarrollo de una pagina web, una de ellas se encargue del header, otra del footer y la ultima del contenido central. Aqui es cuando aparecen las **Ramas**: Romper en diferentes lineas de tiempo el codigo y luego volverlos a unir. En el ejemplo anterior, tanto el header, footer y contenido central son una rama diferente de la otra, para que cada una de las personas pueda trabajar independientemente en la que le corresponde.

Por defecto, uno siempre se encuentra en la rama **Master** en donde estan todas las versiones de los cambios realizados por los commit. Si por cualquier motivo, en alguna version quiero hacer algo mas experimental, para eso se crea la rama **Development** en la que se copia una version de la rama **Master** y le pongo un nombre, en este caso, "**Esperimentos**". Toda la rama Development es completamente diferente en codigo y contenido que la rama Master.

En caso de encontrar un bug en la version actual de la **master**, debes crear una rama especial, **Bugfixing** o mas conocida como **Hotfix**. Luego de eliminar los bugs y realizar cambios en la rama Hotfix, para volver a la rama Master, se realiza un **Merge**, creando una nueva version dentro de la Master. Tambien se puede hacer un Merge entre una version de la rama Development y la rama Hotfix para crear una nueva version de la Master. Se debe tener en cuenta en el momento de hacer un Merge, de que no haya un conflicto en el que algunos archivos se rompan, o que, siguiendo con el ejemplo, algun cambio realizando en **Bugfixing** rompa un cambio en **Development**