**Git**

Sistema de control de versiones

**Git Tiene 3 Zonas**

**Working Area :** es la zona donde se encuentras los archivos que tienen cambios y que no han sido agregados al steging area. (los archivos de las carpetas en broto en ese momento sin ser agregados por el git add)

**Staging Area :** es la zona donde se indica que archivos están preparados para hacer commit, es decir archivos agregados en ese momento con git add

Git directory : es la zona donde se guardan los datos de los commits, archivos guardados con git commit –m “nombre del commit”



**Configuraciones iniciales**:

git config - - global user.name “jesus dario” 🡪 configura el nombre en git.

git config - - global user.name 🡪 muestra el nombre configurado en git.

git config - - global user.name “Jesus Dario” 🡪 cambiamos el nombre con el mismo comando de creación.

git config - - global user.email “jesusmarencodev@gmail.com” 🡪 configurar el correo en git.

git config - - global color.ui true 🡪 configurar los colores de los resultados.

git config - - global - -list 🡪 muestra todas las configuraciones globales.

Empezando el proyecto:

git help 🡪 documentación de los comandos de git

git init 🡪 marca el inicio del proyecto.

git status 🡪 muestra el estado de los archivos del proyecto.

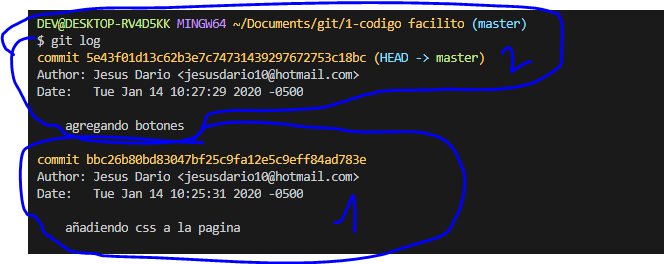
git add archivo completo (eje: git add index.html)🡪 agrega un archivo para ser añadido al commit.

git add . (al final espacio punto)🡪 agrega todos los archivos para ser añadidos al commit.

git commit –m “mensaje para guardar” 🡪 guarda un punto del proyecto a esto se le llama commit.

**git log**

Este comando lo tratare especialmente porque los muestra una lista de todos los commits registrados.

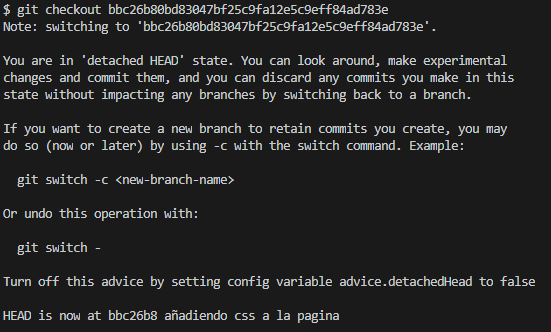


Cada commit genera un código SHA, este código es el que se ve en amarillo al lado de la palabra commit, este código generado es único para cada commit, mas adelante los usaremos.

**git checkout código SHA del commit al que queremos viajar**

**(ejemplo: git checkout bbc26b80bd83047bf25c9fa12e5c9eff84ad783e)**

Permite movernos entre commits sin perder los cambios, es como viajar en el tiempo en un proyecto



**git checkout master**

Nos mueve nuevamente al ultimo commit existente.

**git reset --soft**

borra el commit elegido pero no toca el código o “Working Area”

ejemplo: git reset--soft bbc26b80bd83047bf25c9fa12e5c9eff84ad783e

**git reset --mixed**

borra el “Staging Area”, sin tocar el “Working Area”. No se usa mucho

**git reset --hard**

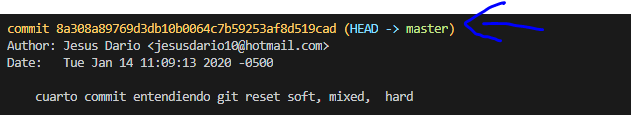
borra el commit commit elegido y el código también

ejemplo: git reset--hard bbc26b80bd83047bf25c9fa12e5c9eff84ad783e

**ramas y funciones**

**terminología de la sección:**

**Head**: es el commit donde nos encontramos en ese momento.



**Ramas**: una rama es una línea de tiempo en nuestro proyecto, que nos sirve para arreglar errores, experimentar, hacer grandes cambios etc.

**Rama master:** cuando hacemos git init se crea por default y es la rama principal.

* **git branch** 🡪 este comando muestra el listado de ramas existentes actualmente.

 el asterisco muestra en que rama estamos ubicados

* **git branch nombre\_rama** 🡪 crea una rama con el nombre proporcionado.

 cuando se crea una rama se crea una copia de master o de la rama actual y se empieza desde allí.

* **git chetckout nombre\_rama** 🡪comando para moverse entre ramas

 todos los commits que se hagan desde el cambio se iran a la rama donde estemos, en este caso test.

* A
* A
* a