**Curso git y github**

1*.* ***Git init*** = inicias el repositorio

2. ***Git add******+ nombre del archivo*** = Iniciaras el archivo en un estpacio de memoria ram llamado staging

3. ***Git commit -m "Mensaje de los cambios realizados"*** = Con este comando enviarás el archivo a tu repositorio local y guardará los cambios realizados al archivo

Regresar a un archivo

Si quieres regresar en el tiempo y volver a una versión anterior de tu archivo borrando los cambios realizados

Previamente de tu repositorio local se usa el comando Git reset.

Hay dos maneras --hard y --soft

***Git reset (ID referente al commit) --hard***

**Borrar archivos**

Si quieres volver a una versión anterior sin borrar el historial del archivo el comando que debes emplear es:

***git checkout +ID commit***

En este estado puedes realizar cambios pero si realizas nuevamente un commit ese será tu archivo actual.

**Creación de ramas**

Para crear una nueva rama se utiliza el comando git branch + nombre de la rama (git branch contenido), para entrar a la rama utilizas el comando:

***git checkout + nombre rama (git checkout contenido)***

Al cambiar de rama cambiarán los archivos en tu repositorio local pero seguirán estando guardados en cada rama solo si haces un commit a los cambios

**Fusión de ramas**

Debe situarse en la rama master para poder hacer el marge (acción que fusiona los commits)

Con git branch puedes saber cuántas ramas hay y sus nombres

Hacer la fusión con el siguiente código:

***git merge + nombre de cabecera (git marge cabecera)***

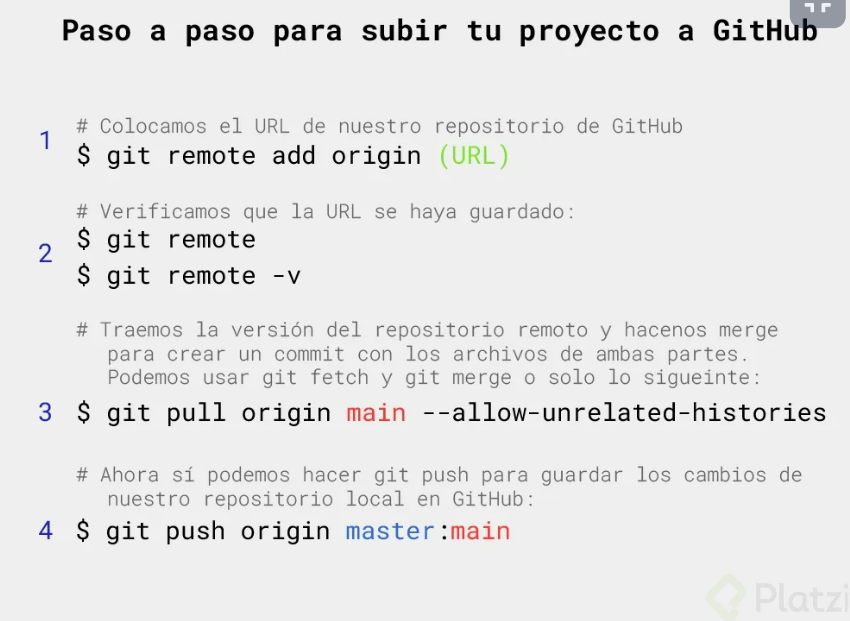
Aparecerá vim pidiendo un nombre del branch, shift + zz (salir del vim) y guardamos los cambios con un commit

**Solucionar conflictos al realizar merge**

Al realizar un merge de la rama #2 a la rama master y hay una edición en la misma línea en ambas ramas, se presentará un conflicto.

Para solucionarlo visual estudio code, te mostrará las diferencias y tú decidirás con cual quedarte, asegúrate bien la decisión.

**Subir proyecto a GitHub**

****

Códigos principales; para iniciar el repositorio remoto, primero copiar la url desde el nuevo repositorio creado en GitHub (<https://github.com/Jhondavid23/Hyperblog.git>)

***git remote add origin*** [***https://github.com/Jhondavid23/Hyperblog.git***](https://github.com/Jhondavid23/Hyperblog.git)

Luego de tener iniciado nuestro reposito remoto procedemos a hacer el push de nuestros archivos locales

***git push origin master***

Para traer los archivos del repositorio remoto se hace un git pull (en caso de que haya algún cambio)

***git pull origin master***

**Cambiar la configuración de git**

(Actualizar el correo electrónico)

Git config –global user.email [juandaviddiazorozco@gmail.com](mailto:juandaviddiazorozco@gmail.com)

**Hacer una llave pública y privada**

Para cifrar los datos en una encriptación asimétrica se utilizan las llaves públicas y privadas. Con esto no tendrás que colocar nuevamente tu cuenta de github cada que inicies un proyecto.

Debes estar en el home ~

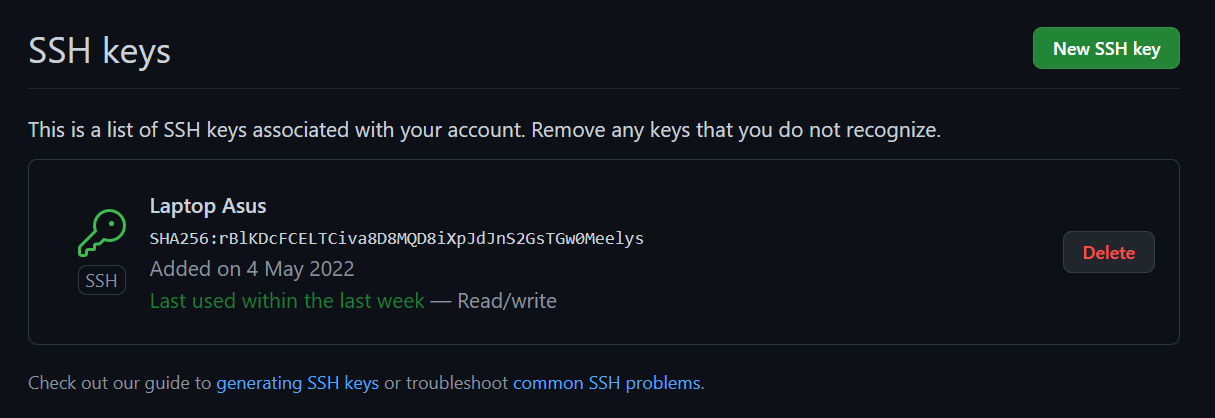
El código es: ssh-keygen –t rsa -b 4096 -C [juandaviddiazorozco@gmail.com](mailto:juandaviddiazorozco@gmail.com)

Luego de crear la llave hay que verificar que el servidor y las llaves estén corriendo, por lo cual se ejecuta el siguiente comando: eval $(ssh-agent -s)

Luego de verificar que está todo ok, se añade la llave privada con el comando *ssh-add ~/.ssh /id\_rsa*

Debe aparecer una respuesta así: Identity added: /c/Users/juand/.ssh/id\_rsa (juandaviddiazorozco@gmail.com)

Para enlazar la llave pública con github se va a configuración, ssh and gpg keys y se agrega una nueva.



Luego de eso, se cambia la configuración en local de la conección con github ty en vez de utilizar https, se copial la url ssh y se configura de la siguiente manera:

git remote set-url origin [git@github.com:Jhondavid23/Hyperblog.git](mailto:git@github.com:Jhondavid23/Hyperblog.git)

Luego se verifica con git remote –v que la url de origin haya cambiado.

Se procede a realizar el git pull, y pide autorización, se coloca *yes*

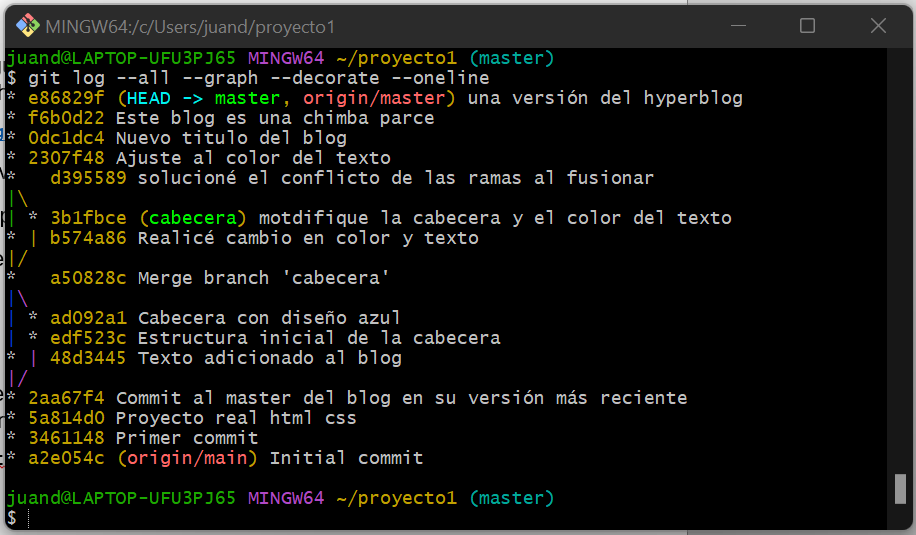
Se procede a realizar nuevamente el pull, git pull origin master.

**Etiquetas**

Con este comando puedes obtener de una manera detallada todo el historial de tu proyecto desde que arrancó, de una manera simple, compacta y visual.

git log --all --graph --decorate --oneline

Ahora llamado **arbolito**

****

**Crear etiquetas**

Copias el hash de tu commit y lo pegas en el siguiente código

git tag -a v0.1 -m "Resultado de las primeras clases del curso" 5a814d0

El nombre de la etiqueta en este ejemplo es (v0.1)

Con ***git tag*** puedo ver los tags que existen

Para borrar un tag escribo el código git tag –d + nombre del tag

En el código la -d es de delete.

Para borrar el tag de github se usa el siguiente código: ***git push origin :refs /tags/dormido***