**¿Qué es Git?**

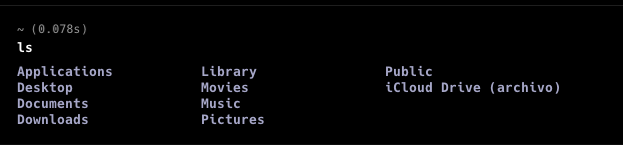
git es un software de **control de versiones** y **distribuido**

**¿Quién lo creo?**

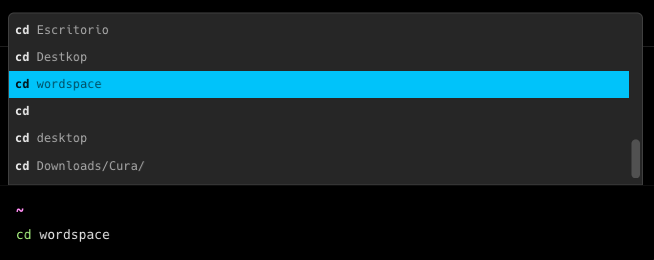
lo creo Linus Benedict Torvalds en 2005 y también es el creador de linux

**Comando Básico de GIT**

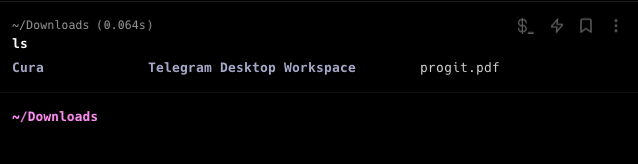
ls: Es para saber que **carpetas** hay en en **directorio** que estamos



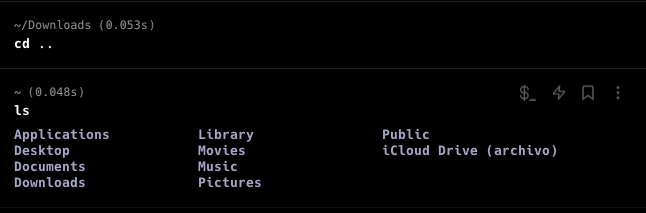
cd : es para ir a la carpeta o directorio



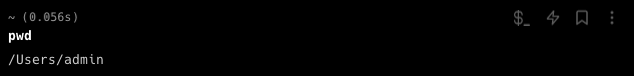




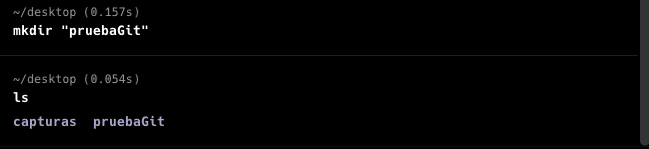
cd .. o cd ../ : es para ir una carpeta anterior de la actual



pwd: es para saber la url en la que estoy actualmente



mkdir “ ” : es para crear una carpeta o directorio



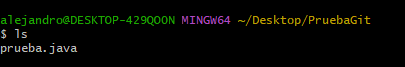
clear : para limpiar la terminal

touch : nos sirve para crear un archivo y el formato que va a tener

**touch “archivo.formato”**

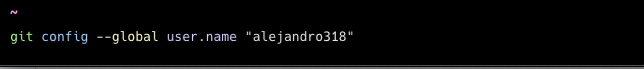


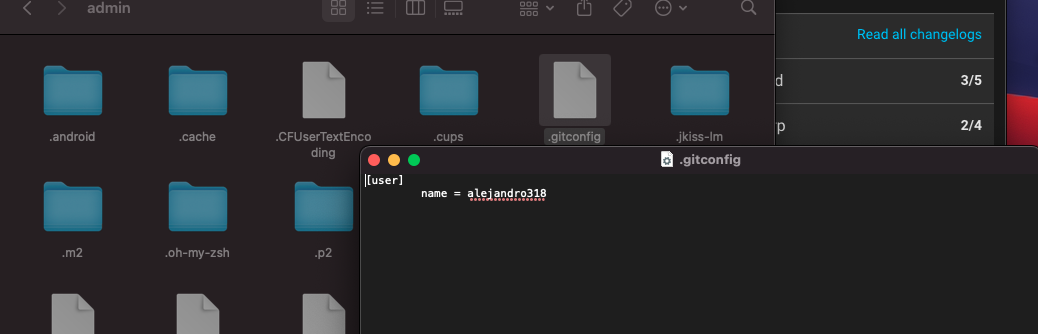
ahí se nos crea el fichero o archivo



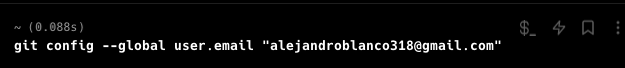
**Comando de Configuración(IMPORTANTE!!!!!)**

el comando git config name para configurar de manera global el usuario de mi github solo para el equipo que estoy usando



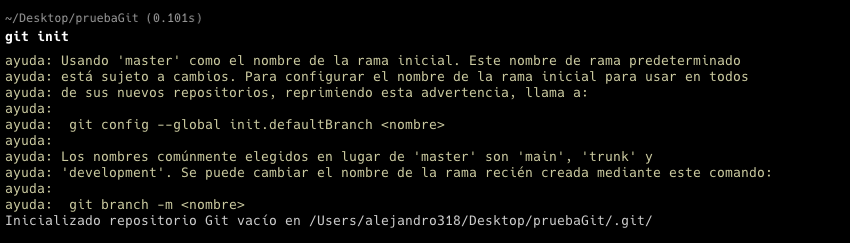


**git config -–global user.email “ ”**: es para configurar de manera global el email que se usará en el equipo que estoy usando para subir nuestro proyecto a github



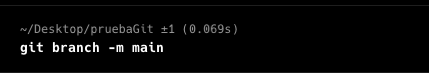
**inicializar el uso de git**

**git init**: es para empezar usar git como un sistema de control de versiones en un directorio

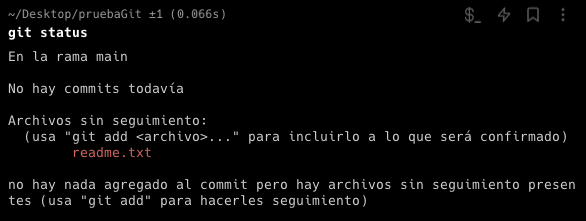


**Cambio de nombre a rama actual**

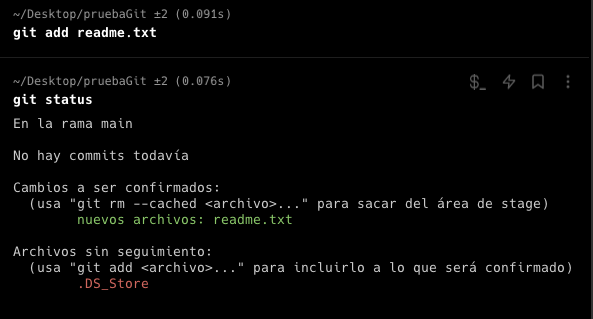
**git branch -m** (nombre con el cual se va a remplazar)



**git status** : saber en qué estados se encuentran mis fichero y en que rama estoy

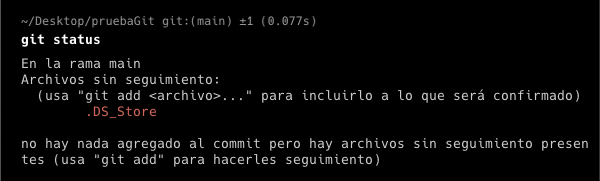


**git add (fichero):** es para decir que este fichero o estos ficheros son seleccionados para subirlo a la area **stage**



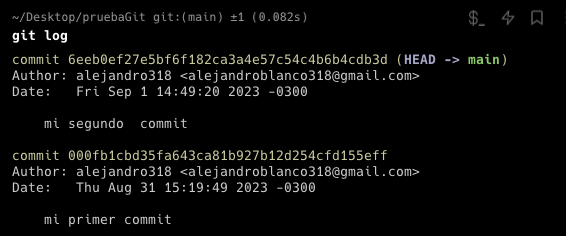
**git commit -m “mensaje”:** es una acción que registran uno o varios cambios en nuestro historial de git. Es decir que todos los ficheros que estén en la area stage les voy a tomar como una foto como un punto de guardado para nuestro historial en git. Este commit se asociará a un **hash**(esto nos va a servir, si queremos volver a un punto que guardamos anterior para revertir un cambio que hicimos)





**git log :** es para ver el seguimiento de los commits que tiene nuestro proyecto donde está apuntando el **HEAD**(manera local). Este seguimiento contiene:

1. su hash + rama que se subió
2. quien hizo el commit(por esto es importante ingresar nuestro nombre y email)
3. que dia y a que hora
4. el comentario que hizo.



**git checkout <archivo o commit o rama>:** es para decirle al HEAD a donde queremos que apunte,deshacer un cambio de un archivo,se desplace a un commit anterior, un commit en específico o a una rama específica(no es lo recomendable usarlo para cambia de rama usar mejor **switch**)

**Dato**

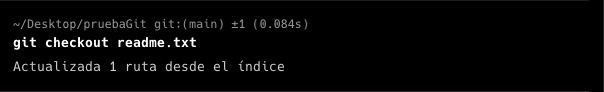
si queremos ir un commit en específico tenemos que tener en cuenta que no tiene que haber un archivo modificado en el head que estamos ya que no nos dejar ir al commit que queremos y también si el commit es anterior a nuestro commit tenemos que tener en cuenta que si habíamos creado un archivo o carpeta se eliminará ya que en el commit que fuimos esa carpeta o archivo no existía(es como volver a pasado)

**archivo no modificado**



**archivo modificado**

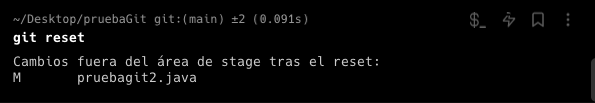




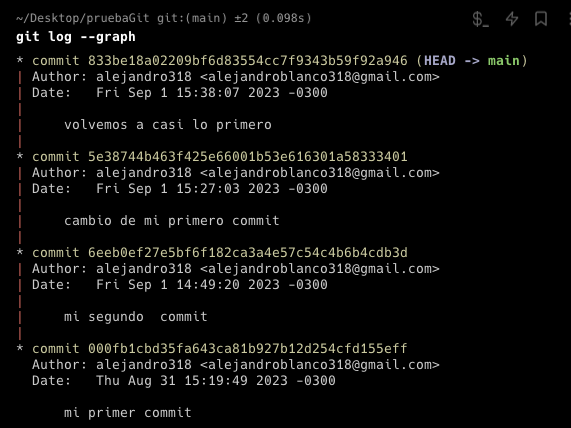
archivo restaurado al último commit que tenemos



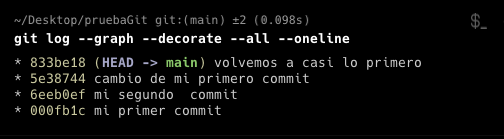
git reset: nos muestra una lista de todos los archivos modificados que tenemos a partir de nuestro último commit



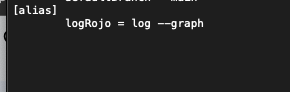
git log –graph : es para ver el listado de commits con una linea roja

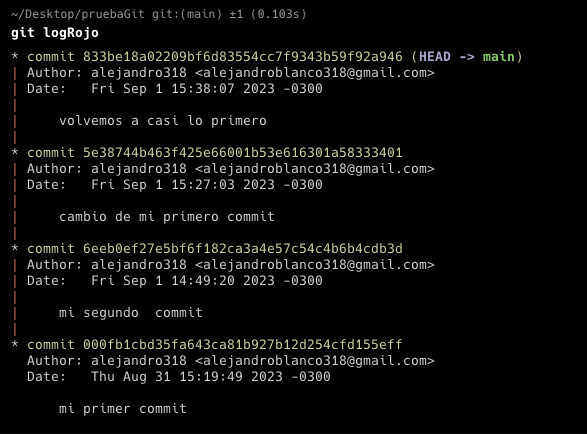


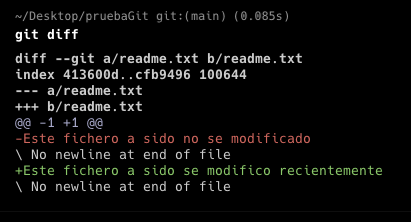
**git log —graph —decorarte —all —oneline:** es para ver el listado de commits pero con el hash abreviado y solo con el comentario del commit. También nos muestra en donde estamos situados en nuestros commits y rama.

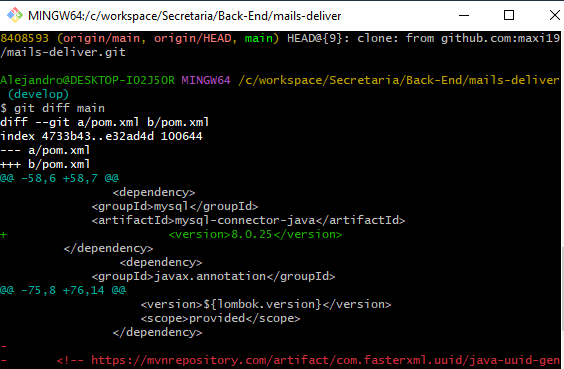


**git config —global alias.<nombre> = “contenido”**: nos crea una variable global en nuestro .gitconfig con el contenido que queramos. le podemos poner un comando que usamos mucho para que nos facilite a la hora de usarlo



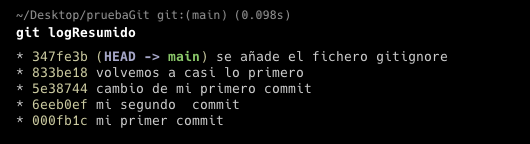


**git diff :** es para ver cambios que hicimos a un fichero o varios.o También a una rama visualizados en la consola. Nos muestra lo que se modificó y lo que lo se agregó

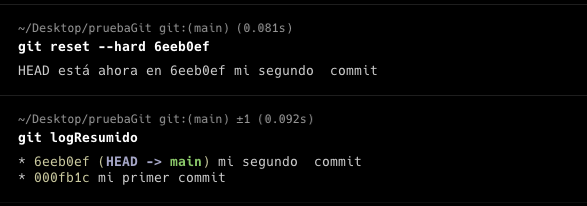


**git resert —hard <identificadorCommit> :** es para volver y eliminar uno o varios commit o restaurar un commit que borramos(siempre cuando volvamos a un commit, fijarse que nos apunte a la rama no a mi HEAD)

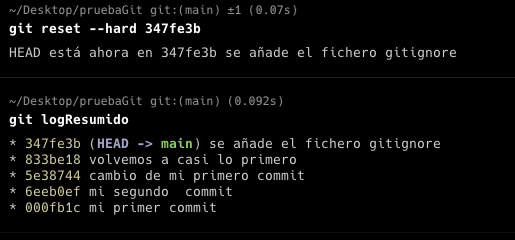
MI historial de commits:



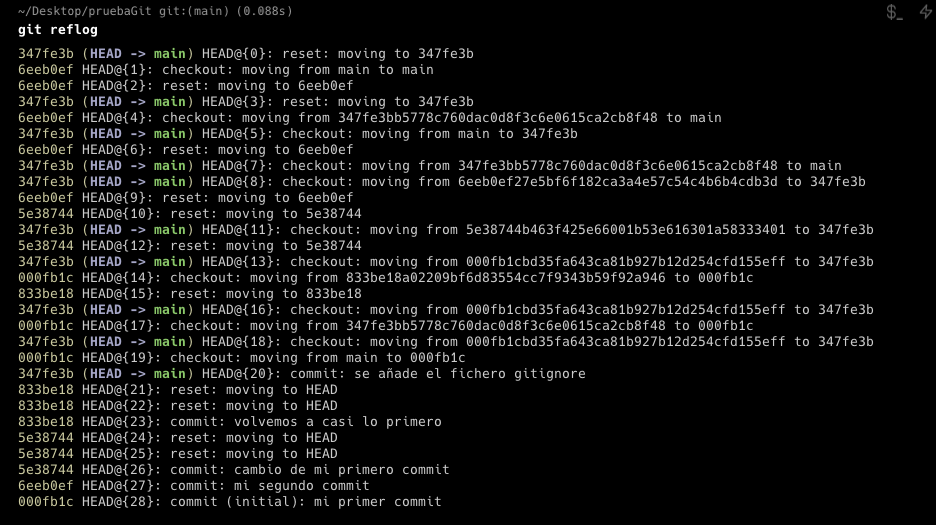
Eliminamos 3 commits:



restauramos 3 commits:



**git reflog :** es lo mismo que git log pero nos deja ver el historial completo de todos los que hicimos en nuestro repositorio local



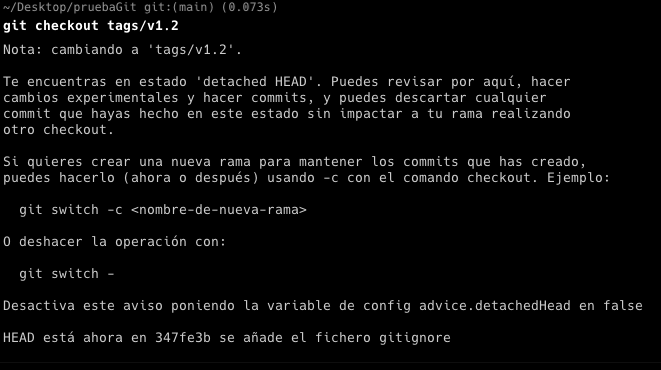
**git tag:** es para ver el listado de tags



**git tag “etiqueta”:** es para agregarle una descripción al commit que estamos actualmente como por ejemplo: la versión de la aplicación que estamos desarrollando. También con este tag podemos usarlo para ir a un commit es específico sin usar el identificador

****

**Nos desplazamos a otro commit anterior a través del tag:**

****

**Ramas**

**git branch <nombre>:** crea una rama a partir de commit que esté situado nuestro **HEAD**



**git switch <nombreDeLaRama>:** es para cambiar de rama



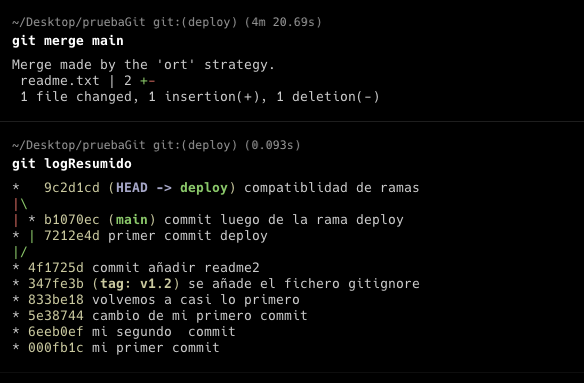


**git merge -m “comentario”:** es para fusionar o juntar dos ramas con la funcionalidad que estamos trabajando o para ver si los cambios que hicimos en la rama que estamos trabajando son compatibles con la rama que lo estamos fusionando ya que en un entorno de trabajo no vamos a ser los únicos en trabajar en una funcionalidad.

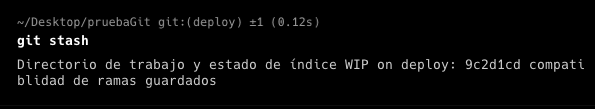
**Dato**

si cuando vamos a hacer un merge tenemos que tener en cuenta que si dos equipos modificaron el mismo bloque de código ya que esto nos causará un problema en que no nos dejara hacer el merge sería un conflicto de merge. Tratar de hablar con el otro equipo para repartirse los funcionalidades sin que entre ustedes se afecte si funcionalidad. Además, cuando hacemos un merge nos genera un commit luego

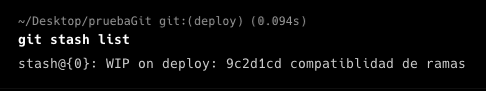




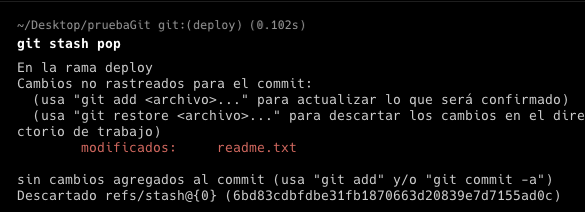
**git stash** : es para hacer un commit temporal de manera local sin tener que hacer commit normal. Ejemplo : nos llaman para solucionar un error en rama main, estamos desarrollando en la rama deploy pero tenemos el desarrollo a medias lo podríamos commitear pero no sería una buena práctica ya que cuando commitemos solo lo hacemos porque hicimos un cambio que funciona, por eso hacemos un stash.



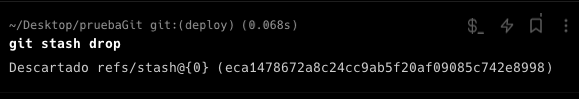
**git stash list:** para ver el listados de commits temporales que tengo



**git stash pop:** es para traer el commit temporal



**git stash drop:** es para eliminar el commit temporal que hicimos



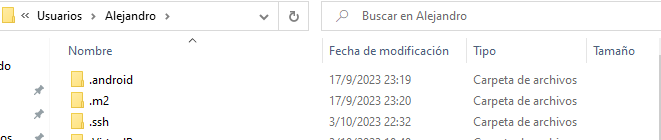
**git branch -d <rama>** : es para eliminar una rama(la rama en términos generales no se elimina del todo nos quedo la referencia es como los commit cuando lo eliminamos)



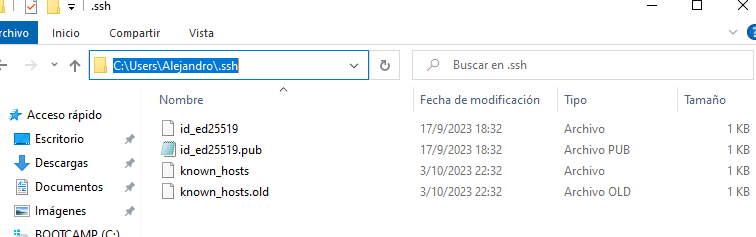
**Configuración Autenticacion SSH**

**buscamos la carpeta .shh(dependiendo del Sistema operativo)(aca es donde guardaras tu clave .ssh)**

**Windows:**

****

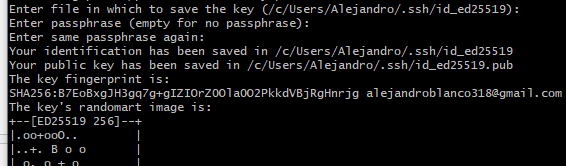
**en mi caso ya tengo mi Clave SHH**



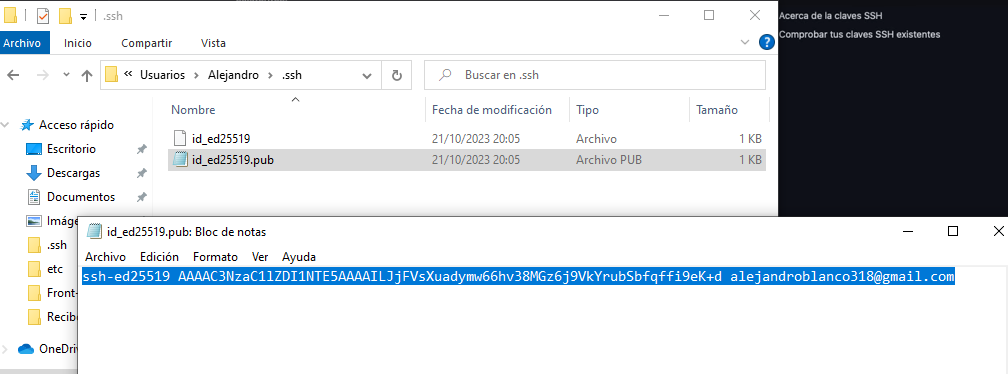
ponemos el siguiente comando en git



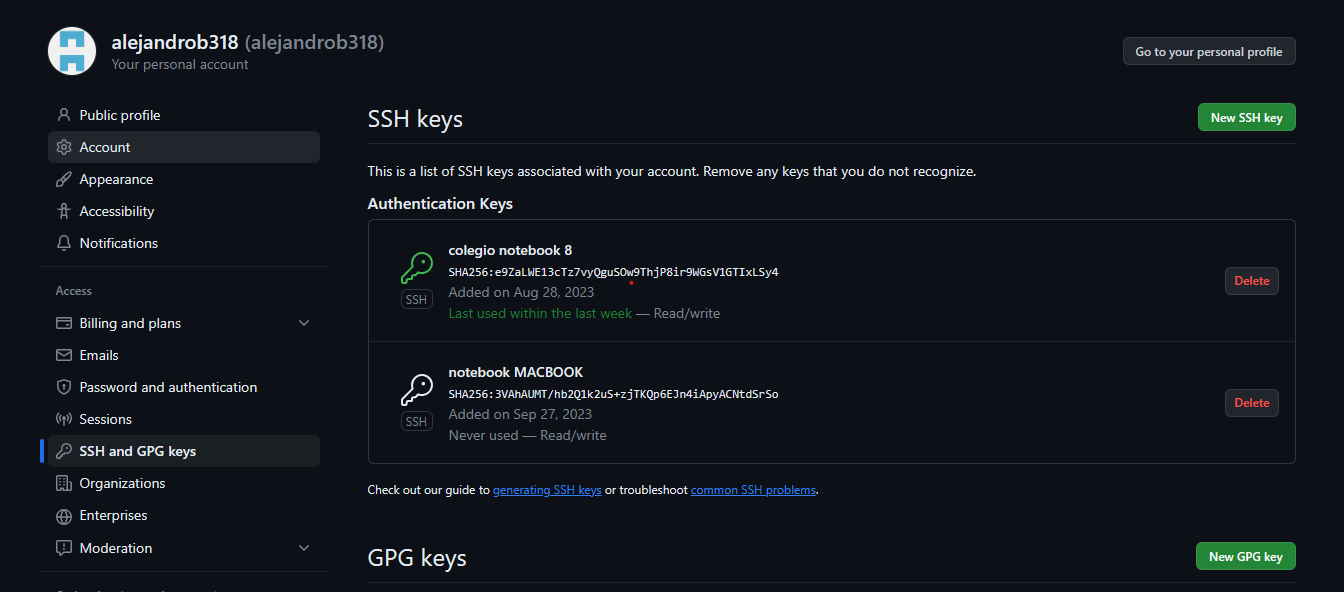
luego todo enter (en caso que queramos configurar lo primero que te pide es el nombre que queres llamarle a tu clave ssh)



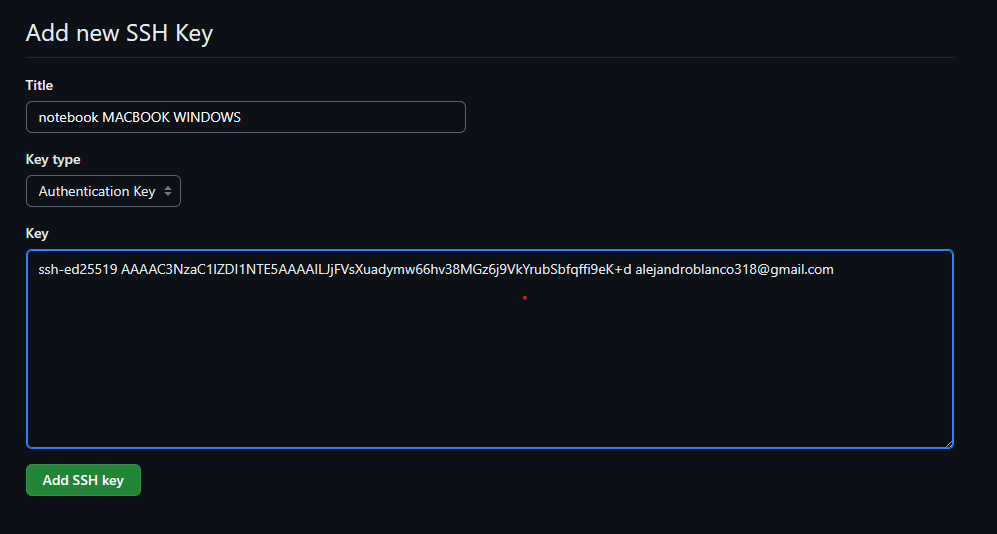
luego vamos a la carpeta .shh y copiamos el contenido de id\_ed25519



luego lo pegamos en git > settings > shh and gpg key > new SSH



y guardamos



y listo

Git Remote: nos sirve para indicar en que repositorio remoto vamos a subir nuestro proyecto

git remote add origin {url-de-repositorio}

Git push origin {rama} : nos para subir el proyecto especificando en que rama lo vamos a subir

**Glosario:**

**open source:** es un término que se usa para decir que el código de un software es libre de ver,modificar o distribuir sin ninguna delito legal

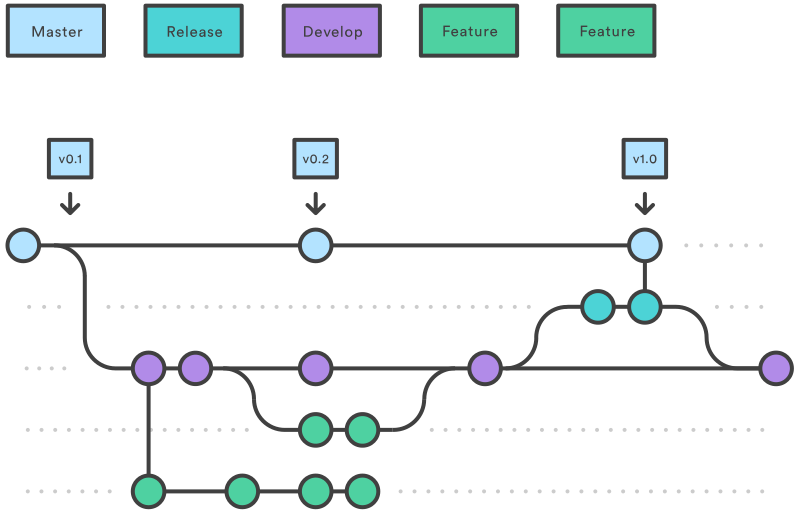
**distribuido**: que tenemos el proyecto de manera local y remota(github)

**control de versión:** podemos ver un historial de todos los cambios hechos en un proyecto y en el estado que se encuentra. la documentación que se hizo, volver a una versión anterior y etc.

**directorio** : es una carpeta que puede contener archivos o carpetas

**-m “mensaje”** : va junto a un comando que va a tener un mensaje

**rama:** se lo define como una línea independiente de desarrollo que contiene un proyecto similar a otra rama pero con un diferente cambio sin tomarlo en cuenta.Esto lo hacemos para que no se mezclen el desarrollo que se está por implementar. También podemos unir todas las ramas en la principal para agregar esos cambios



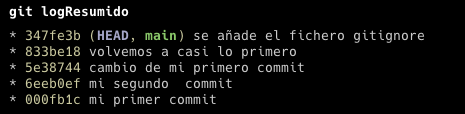
**.DS\_STORE:** es un fichero que crea mac cuando creamos una carpeta

**hash :** es un identificador único que no se puede repetir. Con el hast lo asociamos con un commit y con un comentario describiendo lo que se subió

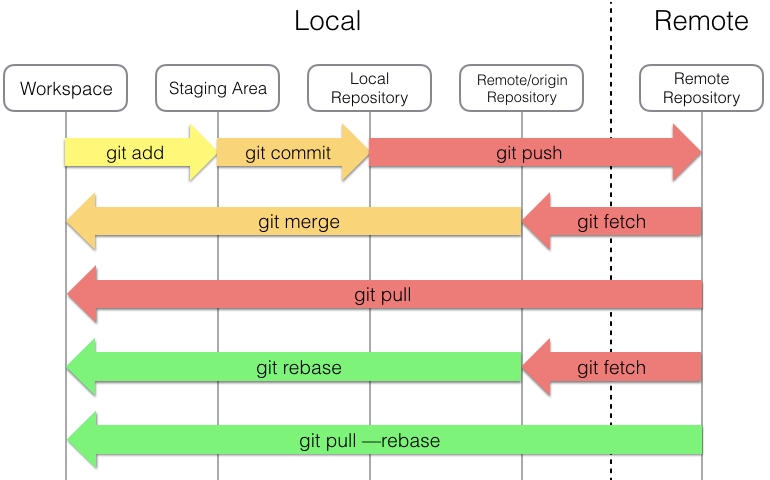
**.gitignore:** es una fichero que nos ayuda a ignorar a que se añadan en la etapa stage un archivo en específico o todos los ficheros de un mismo formato

**HEAD:** indicadonde estoy actualmente en mis commits

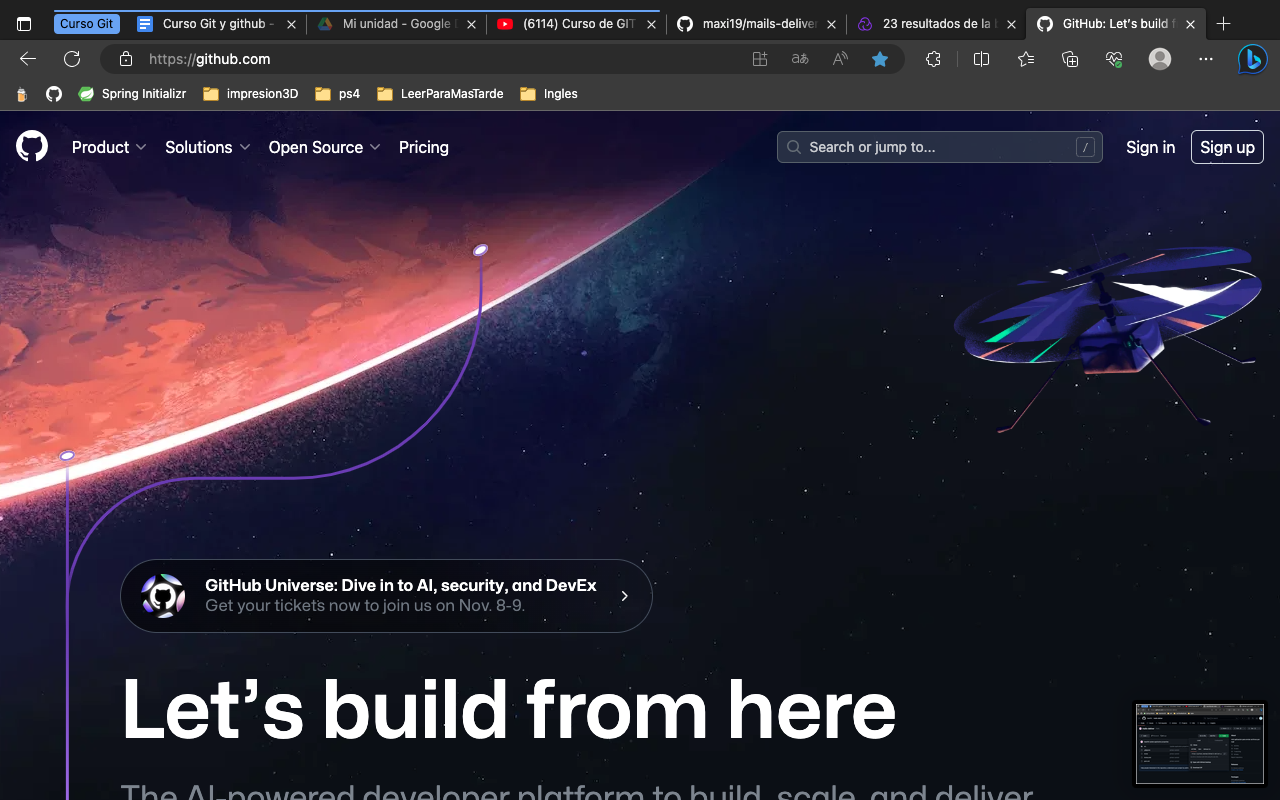
**main**(rama): indica en donde finaliza el commit de esa rama y en que rama me encuentro



**stage:** es una etapa o área en donde solo se suben los fichero o archivos que fueron seleccionamos con git add



**GITHUB:** es una plataforma que usa git de forma oculta ya que se usa como repositorio remoto en donde muchas personas podrán interactuar con nuestro proyecto verlo, descargarlo o que otras personas desarrollen el mismo proyecto con mayor facilidad de acceso y todas nuestro historial general de git(commit,ramas,etc)



**Repositorio:** es el lugar donde se guardó un proyecto. Repositorio puede ser local o remoto.Local se dice cuando se está guardando en tu ordenador interno y remoto en cuando se sube a un servidor o puede llamarse nube

**Readme:** es un archivo donde guardamos la documentación del proyecto

**Issues**; que tengo incidencias en nuestro proyecto

**Pull requests:**