

Configuraciones iniciales y comandos básicos

Conceptos

Header: Se refiere a un apuntador, nos indica en donde nos encontramos (versión de commit de la rama actual).

Master: Se refiere a la rama principal de desarrollo.

Branch: Se refiere a una rama de desarrollo diferente de la rama Master. Una rama siempre se creará a partir del header actual.

Comandos de inicialización y configuración básica

Estos comandos son configuraciones iniciales que debemos usar en Git, es importante memorizarlos.

```
Git init = Iniciar el repositorio en la carpeta seleccionada
Git configure user.name: permite configurar el nombre del usuario.
Git configure user.email: permite configurar el correo del usuario.
```

Estos comandos son los básicos para empezar a trabajar nuestro git.

```
Git add . : Nos permite añadir todos los cambios a la sección de staging.
Git commit: nos permite crear un nuevo commit con los últimos cambios.
Git commit -am: nos permite hacer un add y un commit al mismo tiempo.
```

Comandos para crear ramas de desarrollo

```
Git branch (nombre deseado para branch): nos permite crear una nueva rama de desarrollo.
Git branch -d (nombre del branch): nos permite eliminar una rama creada anteriormente.
Git merge (nombre de la rama para el merge): Nos permite unir los cambios de una rama con otra.
```

Comandos para gestionar Git

Gitk: abre un software gráfico que nos da un log e igualmente una representación gráfica en árbol de nuestro git.

Git status: Nos permite saber el status actual de nuestro repositorio.

Git log: nos tener un listado de todos nuestros commits.

Git log --oneline: nos permite tener un listado de nuestros commits pero resumidos, ideal si ya tenemos muchos commits.

Git checkout (hash del commit/branch): nos permite mover el Header de la branch actual hacia el último commit realizado de dicha rama.

Alias: Ésto nos permite crear variables reutilizables con códigos de git, así no tendremos que escribir códigos super largos cada vez, veamos un ejemplo...

```
alias arbol ="nuestro código"
```

Ahora cuando escribamos la palabra "arbol" se ejecutará el código que hayamos puesto entre las comillas, sin tantas complicaciones.

Eliminación de archivos

Git reset y git rm son comandos con utilidades muy diferentes, pero se pueden confundir muy fácilmente.

```
git rm --cached` : Elimina los archivos de nuestro repositorio local y del área de staging, pero los mantiene en nuestro disco duro. Básicamente le dice a Git que deje de trackear el historial de cambios de estos archivos, por lo que pasaran a un estado `untracked`.
```

```
git rm --force` : Elimina los archivos de Git y del disco duro. Git siempre guarda todo, por lo que podemos acceder al registro de la existencia de los archivos, de modo que podremos recuperarlos si es necesario (pero debemos usar comandos más avanzados).
```

Creación de repositorios remotos

Llaves publicas y privadas





Crear llaves localmente (localmente)

El código para crear una llave ssh es con git es:

```
ssh-keygen -t rsa -b 4096 -C "Corre electronico vinculado a GitHub"
```

Una vez hecho eso, Git nos recomendará la carpeta donde guardar, así que solamente damos enter. Le podemos poner una contraseña a esa llave ssh, ésto es seguridad adicional, por practicidad no la pondremos, pero es recomendable sí ponerla, por buenas

prácticas más que nada. La carpeta designada contendrá archivos como los que se muestran a continuación.

<input type="checkbox"/> Nombre ^	Fecha de modificación	Tipo	Tamaño
 id_rsa	18/02/2023 07:31 p. m.	Archivo	4 KB
 id_rsa.pub	18/02/2023 07:31 p. m.	Microsoft Publisher Document	1 KB
 known_hosts	18/02/2023 07:45 p. m.	Archivo	1 KB
 known_hosts.old	18/02/2023 07:45 p. m.	Archivo OLD	1 KB

id_rsa: Llave privada > La llave que no debemos compartir y debemos cuidar demasiado.
id_rsa.pub: llave pública > la que usaremos publicamente y sí podemos compartir, se usará para nuestras cuentas de GitHub u otras.

Necesitamos comprobar el proceso y agregarlo a Windows:

```
eval $(ssh-agent -s)
ssh-add ~/.ssh/id_rsa
```

El simbolo "~" es un shortcut absoluto, que buscará dentro de la carpeta home de nuestro sistema.

El comando ssh-add añadirá nuestra llava privada, si tiene otro nombre hay que sustituirlo.

Para añadir el proyecto remoto usamos el siguiente código:

```
git remote add "nombre" git@github.com:"nuestro proyecto"
```

Ejemplo (con el proyecto actual):

```
git remote add teamProject git@github.com:Baalgarthem/proyectoCompilador.git
```

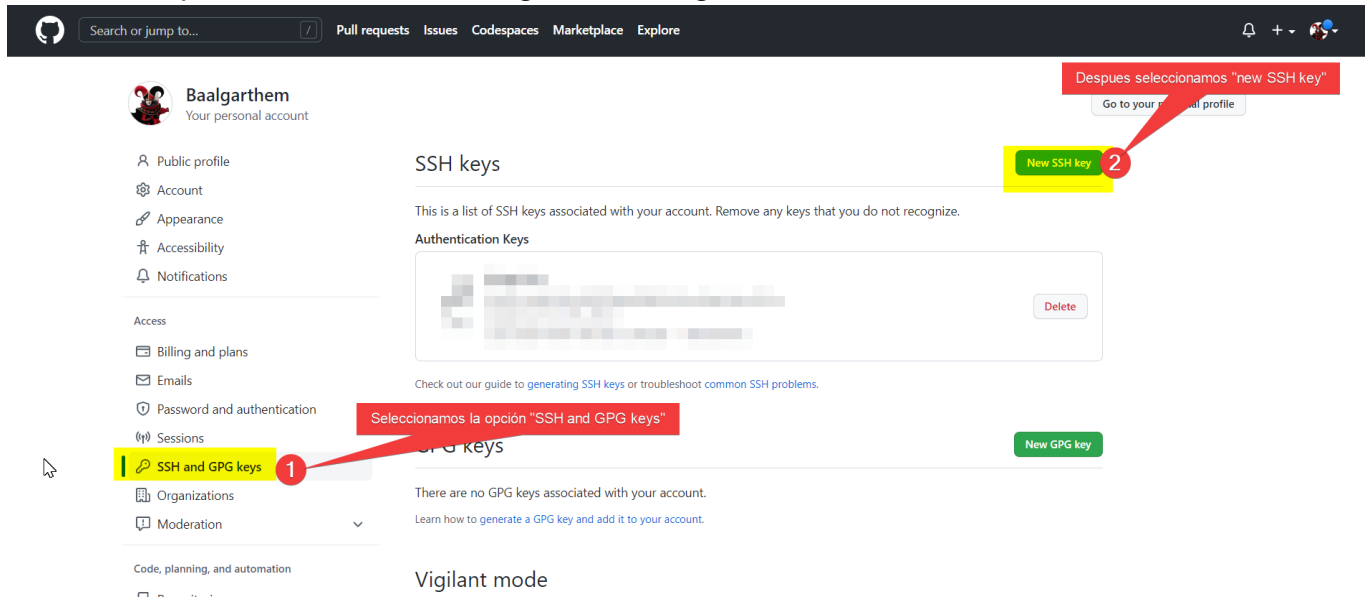
La evaluación (para iniciar el proceso ssh) y adición (para añadir ese proceso al repo) las debemos ejecutar en la carpeta donde se creó la llave ssh.

Conectar repositorio

Un paso importante para configurar nuestro repositorio con GitHub es ir a la página oficial:

<https://github.com/settings/keys>

Y hacer lo que se muestra en la siguiente imagen:



Cuando seleccionemos "New SSH key" metemos el contenido de nuestra llave publica.

Para comprobar que todo está correcto usamos el siguiente comando
git remote -v

Si hicimos todo correctamente, ya deberíamos poder hacer comandos remotos, tenemos dos direcciones remotas, una para hacer fetch y otra para hacer pull. Si nos aparecen como en la siguiente imagen, significa que ya terminamos de configurar el ámbito remoto.

```
6de27be se añade un documento de proyecto compilador, para estructurar y documentar el proyecto
Alush@Reign MINGW64 /d/Academico/Lenguajes y automatasm1/Proyecto Compilador (ejercicios)
$ git remote add teamProject git@github.com:Baalgarthem/proyectoCompilador.git

Alush@Reign MINGW64 /d/Academico/Lenguajes y automatasm1/Proyecto Compilador (ejercicios)
$ git remote -v
teamProject      git@github.com:Baalgarthem/proyectoCompilador.git (fetch)
teamProject      git@github.com:Baalgarthem/proyectoCompilador.git (push)

Alush@Reign MINGW64 /d/Academico/Lenguajes y automatasm1/Proyecto Compilador (ejercicios)
$
```

Si ejecutamos

git remote

veremos solamente la conexión remota (hacia nuestro GitHub) que hayamos configurado, tal cual se muestra a continuación:

```
teamProject      git@github.com:Baalgarthem/proyectoCompilador.git (push)

Alush@Reign MINGW64 /d/Academico/Lenguajes y automatasm1/Proyecto Compilador (ejercicios)
$ git remote
teamProject
```

Comandos necesarios para trabajar con repositorios remotos.

Para recapitular, entendamos que los comandos add, commit, trabajan sobre nuestros archivos locales. O sea que no afectarán al repositorio remoto de ninguna forma.

El comando para enviar archivos al repositorio remoto es "push".
Veamos su uso.

```
git push teamProject master
```

El comando dice "Haz un push de la rama master perteneciente a teamProject". Ésto enviará solamente la rama Master a nuestro repositorio online en GitHub. Si hay cambios que no hemos guardado entonces no se enviarán, es importante saber ese dato para decidir cuando hacer nuestros commits y push's.