

TUTORIAL SOBRE GIT

Lo primero que se debe hacer es enlazar el repositorio a nuestra maquina, para lo cual tenemos que crearnos una clave ssh la cual la leerá el repositorio remoto y habilitará la conexión.

Configuraremos nuestro git, es importante colocar nuestro nombre y no el de otro usuario ya que lo que hagamos quedara registrado con ese nombre y correo (debe ser el correo que utilizamos siempre). Colocamos:

```
$ git config --global user.name "MiNombre"
```

```
$ git config --global user.email "tuemail@dominio.com"
```

Luego creamos la clave ssh colocando (en email tenemos que poner el correo que ponemos siempre):

```
$ ssh-keygen -t rsa -C "tuemail@dominio.com"
```

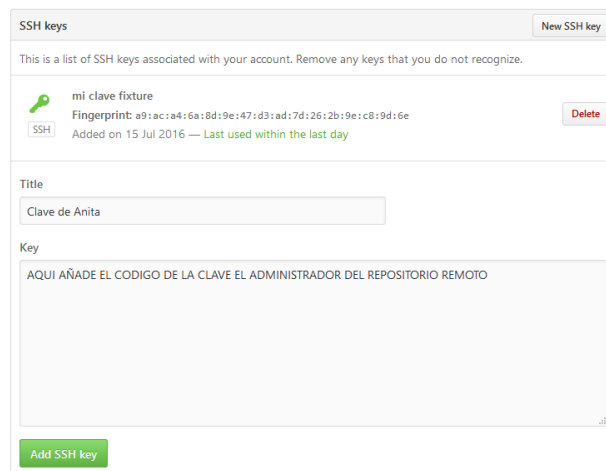
Y nos sale un código similar a este:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDAQVSDxCOhIoaw2tEveeL30BKykJZKVcmZq6Q/U6  
CUTpJWxw+aVIZIsWvkfgUHRjP9GrZ8NoN1PpbxziuzPN4fHQJgZu2Ekd2kJRrxR3gGFVvZZ7a+H  
hMdoK7dy/KehC0aJuZvfldz+KTzhVCHyn7YakGi ih1arFBMY8BvrtjiGJaV8z l4hPmGBCe9dPD  
KDIRvUd9Bp6GmFar2yRLJbHrWzLa8hMJN838DhpEaVh4T+HEuEXMluIFiNQIKMBT l4z7M0W4xFt2  
l4hPmGBCe9dPDKDIRvUd9Bp6GmFar2yRLJbHrWzLa8hMJN838DhpEaVh4T+HEuEXMluIFiNQIKMBT l4z7M0W4xFt2
```

Lo copiamos colocando:

```
$ clip < ~/.ssh/id_rsa.pub
```

Luego la pegamos en un worpad y se lo mandamos al dueño del repositorio remoto, el colocará esa clave en la interfaz de claves ssh:



Luego de que el administrador haya añadido nuestra clave deberíamos poder conectarnos. Creamos un repositorio local solo en caso de que no hayamos creado ningún repositorio. Debemos estar en la carpeta de nuestro proyecto:

```
$ git init
```

Añadimos un archivo que tengamos en la carpeta de nuestro proyecto al stayingIndex:

```
$ git add LEEME.txt
```

Confirmamos que lo llevaremos a la rama (en este caso master) de nuestro repositorio local:

```
$ git commit -m "Colocando el archivo léeme a mi repo"
```

Añadimos nuestro "origen" que es la dirección del repositorio remoto:

```
$ git remote add origin git@github.com:JuanCobaLopez/clarabella.git
```

Nos conectamos al repositorio remoto:

```
$ git remote set-url origin git@github.com:JuanCobaLopez/clarabella.git
```

También debemos actualizar nuestro repositorio local en base al remoto:

```
git pull --rebase origin master
```

Añadimos a lo que acabamos de colocar en nuestro repositorio local a la rama master del repositorio remoto con la instrucción PUSH.

```
$ git push origin master
```

USO DE RAMAS

Hasta el momento ya deberíamos saber como mandar archivos al repositorio remoto pero lo estaríamos mandando a lo bruto (directamente a la rama master), una manera mas organizada es hacerlo mediante otras ramas según sea el rol de cada miembro del equipo.

La rama master jamas debe ser usada para almacenar archivos sin terminar ya que esta rama es la que se encarga de guardar versiones de archivos ya listos para producción.

Para eso tenemos otras ramas las cuales son:

- Desarrollo
- Características
- Lanzamiento
- Revisiones

Desarrollo (a partir de Master): En esta rama debemos colocar nuestras versiones listas para una última revisión la cual lo hará un miembro del equipo designado para ello.

Características (a partir de desarrollo): Esta rama nace y muere cuando se planea implementar una característica o función a la parte que estamos trabajando de esta manera tenemos aparte la característica y aparte la anterior versión de lo que estamos haciendo, cuando la característica este ya bien probada, se fusiona con la rama desarrollo y listo. La rama características se elimina y se crea otra para implementar otra característica.

Lanzamiento (a partir de desarrollo): El nombre de esta rama lleva el numero de la versión de lo que estemos haciendo. Esta rama sirve para definir algunos detalles antes del lanzamiento de la versión. Es muy importante aquí se deben corregir errores también. Cuando la versión sea validada, se manda directamente a la rama master y los cambios hechos en esta versión a la rama desarrollo para mantener un histórico. Luego se elimina esta rama de lanzamiento y se crea otra para futuros lanzamientos.

Revisiones (a partir de master): Cuando el encargado de detectar errores revise la versión que se ha hecho en la rama master y detecte errores deberá crearse una rama auxiliar de "revisión" de manera que lo que haga no afecta al desarrollo. La solución del error debe

mandarse a la rama master y también a la rama desarrollo para mantener un histórico de este parche. Luego eliminamos la rama de revisión y luego crearemos otra cuando tengamos que reparar algo que este en master.

Nota: Esta manera de ramificar no es la única también existen otras maneras por ejemplo la ramificación por tarea, pero esta es la mas usual y serviría aprenderla ya que se usa en la mayoría de empresas, luego se pueden aprender otras maneras.

COMO CREAR RAMAS

Cada miembro según sea su rol debe crear su respectiva rama. Para crear una rama debemos colocar lo siguiente:

```
git branch miRamaLogin
```

Para ver todas las ramas de mi repo:

```
git branch
```

Si quiero cambiarme a otra rama entonces hago:

```
git checkout otraRamaPanelMensajes
```

Para borrar una rama:

```
git branch -d ramaExperimento
```

Para fusionar ramas debemos primero estar en la rama de destino:

```
git checkout ramaDeDesarrollo
```

Y luego fusionarla con la rama de origen:

```
git merge miRamaLogin
```

NOTA: No deben mezclarse archivos con el mismo nombre puede causar conflictos si se hace un cambio en la misma parte de código.

COMMITTS

Como los commits guardan estados de lo que estemos haciendo podemos verlos colocando:

```
git log --oneline
```

Si queremos cambiarnos a alguno colocamos:

```
git checkout códigoDelCommit
```

Si listamos los archivos que habían cuando se hizo el commit lo podemos hacer colocando:

```
$ ls
```

Osea que podemos ir y ver archivos antiguos.

Si el nombre de código lo queremos modificar por un nombre mas adecuado como el nombre de una versión hacemos:

```
git tag v1.0
```

Esto lo hace sobre el commit actual

Si queremos hacerlo sobre un commit específico hacemos:

```
git tag v0.0 codigodelCommit
```

El tag funciona para otras etiquetas como el checkout:

```
git checkout tagDelCommit
```

NOTA: A estos tags es bueno también ponerles nombres de fechas.

OTRA NOTA: Si no deja cambiar de rama por algún motivo, resetear el commit:

```
$ git reset --hard <commit-id>
```