**En github siempre se va a ver la rama maestro principal.**

Si queremos que se vean las ramas:

git checkout *nombreRama* para movernos a la rama.

git show-branch muestra las ramas que existen y sus historias.

git show-branch –all lo mismo pero con mas datos

gitk abre en un software la historia en una interfaz mas visual.

Hacemos git pull origin master para traer la ultima versión subida, nos movemos a la rama a subir y :

git push origin *nombreRama*

**Flujo de trabajo en git:**

Supongamos que tenemos 2 ramas, una footer y un header. Lo van a trabajar 2 programadores por separado. Veremos que pasa cuando queremos fusionar el trabajo del footer con el trabajo del header y cuando los 2 estén listos para enviar a master.

Primero creamos localmente una carpeta, se recomienda que tenga el nombre del proyecto a clonar.

Cuando te incorporas a un proyecto ya iniciado no se va a usar git init, xq no hay que iniciar un nuevo repo sino traerlo de algún lado.

Vas a GitHub, copias el url /https o SSH), y usas:

git clone URL como el repo es público no va a pedir ni user y pass.

(si en vez de clonar el repo. solo lo descargamos en un archivo .zip –github lo permite- no vamos a tener un repo local nuestro y no vamos a pdoer subir modificaciones al repo remoto).

Habitualmente los desarrollos de Git tienen ignoradas las dependencias mediante el archivo .gitignore, por ello es importante que las instalemos de nuevo en el repositorio clon, en local.

Si modificamos algún archivo del proyecto, hacemos el add y el commit, traemos de nuevo el repo git pull origin master, si damos git log veremos también nuestro commit que va a estar en nuestro master local, pero no en el origin master ni en las origin ramas queestan subidas.

Ahora para subir nuestro cambio hay que usar git push origin master, nos pide usuario y contraseña. Pero si no tenemos permiso no nos va a dejar.

Agregar un usuario al repo:

Ir al settings del repositorio, colaborators, y se agrega el mail. Pero para agregar un mail tiene que ser un mail público. Sino con el usuario alcanza.

Una vez que el usuario acepte la invitación podrá hacer push a ese proyecto.

Supongamos que en nuestro repo local creamos ua carpeta y le ponemos una img, que es un archivo binario. Las buenas practicas indican que éstos deben estar a parte, que no se agreguen al repo, que sean ignorados. A modo de ejemplo lo añadimos al repo local.

Nos posicionamos en la rama header con git checkout.

git add carpetaImg/img.png

git commit –am “logo del header”

git pull origni header para traer la ultima versión de rama header.

git push origin header veremos que ya aparece en github.

El problema es que las imágenes son pesadas, si agregamos varias imágenes el repo va a ser pesado. Si editamos la img el cambio va a ser notado por git, va a pesar mas el archivo. Si la subimos a veces tarde en hacerse la actualización de la img por un tema de cache. Si queremos actualizar la actualización ctrl+f5.

Del otro lado para hacer el footer:

Si ya clonamos el repo de hithub a nuestro local, hay que ver donde estamos con git status, probablemente estemos en la master. **(git clone clona solo el master o también todas las ramas?)** Nos traemos al rama footer con git pull origin footer, nos movemos ahí con checkout.

Una vez que agregamos el footer el en archivo html del repo local:

Hago el add y el commit de las modificaciones.

git pull origin footer para traer los posibles cambios,

git push origin footer

y ya tenemos el footer actualizado en github.

Ahora para fusionar todo y suponiendo que ya nos trajimos todos los cambios de todas las ramas:

Ir a la rama maestra git checkout master

Git merge header

Git push origin master para subir el header modificado.

Y lo mismo con el footer: git checkout master, git merge footer y git push origin master.

En un entorno profesional suele blokearse la rama master para evitar que cualquiera haga merges en ella.

Se suele enviar la rama donde trabajamos a la rama staging, que es la rama de testing. Normalmente esto sería un merge con ésta rama. Pero la forma correcta de hacerlo en un entorno profesional es otra, que se hace de esta manera para revisar el código previamente: un pull request. Es un estado intermedio antes de hacer el merge. Permite que otros miembros del equipo vean los cambios que hiciste y si lo aprueban se ejecuta el merge con staging. Una vez que se pureba en staggin y esta todo OK, se hace el pull request pero a la rama master.

Pull request es algo de github no de git. De hecho si vemos el árbol de git log veremos que el pull request es como una pausa antes del merge.

El pol request también permite a personas que no son colaboradores trabajar y apoyar en una rama. Usualmente esto lo hace un especailista en DevOps.

Si creamos una rama copia de la master para solo arreglar errores, es decir sobreescribir líneas de código. Vamos a tener que hacer un pull request entre la master y la rama y luego el merge. Se puede hacer desde la consola o desde github. Igualmente esto es mas tema de DevOps.