**git help**

Muestra una lista con los comandos más utilizados en GIT.

**git init**

Podemos ejecutar ese comando para crear localmente un repositorio con GIT y así utilizar todo el funcionamiento que GIT ofrece.  Basta con estar ubicados dentro de la carpeta donde tenemos nuestro proyecto y ejecutar el comando.  Cuando agreguemos archivos y un commit, se va a crear el branch master por defecto.

**git add + path**

Agrega al repositorio los archivos que indiquemos.

**git add -A**

Agrega al repositorio TODOS los archivos y carpetas que estén en nuestro proyecto, los cuales GIT no está siguiendo.

**git commit -m "mensaje" + archivos**

Hace commit a los archivos que indiquemos, de esta manera quedan guardados nuestras modificaciones.

**git commit -am "mensaje"**

Hace commit de los archivos que han sido modificados y GIT los está siguiendo.

**git checkout -b NombreDeBranch**

Crea un nuevo branch y automaticamente GIT se cambia al branch creado, clonando el branch desde donde ejecutamos el comando.

**git branch**

Nos muestra una lista de los branches que existen en nuestro repositorio.

**git checkout NombreDeBranch**

Sirve para moverse entre branches, en este caso vamos al branch que indicamos en el comando.

**git merge NombreDeBranch**

Hace un merge entre dos branches, en este caso la dirección del merge sería entre el branch que indiquemos en el comando, y el branch donde estémos ubicados.

**git status**

Nos indica el estado del repositorio, por ejemplo cuales están modificados, cuales no están siendo seguidos por GIT, entre otras características.

**git clone URL/name.git NombreProyecto**

Clona un proyecto de git en la carpeta NombreProyecto.

**git push -u origin NombreDeBranch**

Luego de que hicimos un git commit, si estamos trabajando remotamente, este comando va a subir los archivos al repositorio remoto, específicamente al branch que indiquemos.

**git pull origin NombreDeBranch**

Hace una actualización en nuestro branch local, desde un branch remoto que indicamos en el comando.

El primer paso consiste en pasar a la rama "master" para traer los últimos cambios (*commits*) disponibles:

$ git checkout master

Ejecutar git pull para actualizar nuestra copia de trabajo con "master":

$ git pull

Luego volver a nuestra rama "featurex":

$ git checkout featurex

Ahora es posible hacer el *merge* entre "master" y "featurex", para que ésta última se sincronice con los cambios realizados "master" luego del *branch*:

$ git merge master featurex

En caso de no haber conflictos, el *merge* es automático y nuestra rama "featurex" queda actualizada con lo último en "master".

Opcionalmente se puede luego enviar este cambio al *remote*:

$ git push origin featurex

En caso de conflictos, se deberán resolver manualmente. Pero esto es material para otro artículo. Para mayor información acerca de la resolución de conflictos ver la sección "HOW CONFLICTS ARE PRESENTED" de la página de manual git-merge(1).