Git = Es un sistema de control de versiones, la cual nos permite guardar con todo el detalles solo los cambios ocurridos en un determinado archivo.  
Los repositorios son como una base de datos que tendrá en cuenta todos los cambios que realizamos en nuestro código

COMANDO DE GIT:  
ls = para listar directorios y archivos

pwd = para ver en qué ruta de nuestro sistema está posicionado el terminal

git init = para iniciar un proyecto nuevo (una sóla vez por repositorio)

git status = para ver el estado de mis archivos

git status -s = solo muestra los archivos modificados

git show: Mostrara todos los cambios que hemos hecho, esto incluye las líneas que hemos cambiado, cuando y quien hizo dicho cambios.

git add Nombre del Archivo = para agregar un archivo al staging area

git add. = Agrega todos los archivos que hayan cambiado en la carpeta donde estás parado.  
(Después de $ git add . se suele hacer un commit para actualizar la base de datos)

git commit = para crear un primer punto de control de nuestro código

git commit -m “mensaje a escribir” = si quieres enviar algún mensaje

git commit -am = es un “git add” automático de los cambios  
(funciona con los archivos a los cuáles ya les hiciste “git add” previamente)

git config --global user.email ["you@example.com](mailto:"you@example.com" \t "/home/aldemar/Documentos\\x/_blank)" = para configurar el email del usuario

git config --global [user.name](http://user.name/" \t "/home/aldemar/Documentos\\x/_blank) “Your Name” = para configurar el nombre del usuario

git config --global -l = para visualizar el usuario y el e-mail

git log Nombre del Archivo = para ver todos los commit que se han creado

git log --stat = ver todos los cambios específicos que se hicieron en los archivos a partir del commit  
(con la “q” salgo de ahí)

git log --all --graph --decorate --oneline = para ver todos los commit de manera más comprimida  
(alias unaliascualquiera=“git log --all --graph --decorate --oneline” = Para ponerle un alias y que sea más fácil acordarse, en mi caso, es “todosloscommits”)

git checkout = para cambiar rápidamente de rama

git checkout – “Nombre del archivo” = para revertir los cambios de los archivos

git checkout -b Nombre de la nueva rama = Crea una rama a partir de la que te encuentres parado con el nombre “nombre\_rama\_nueva”, y luego salta sobre la rama nueva, por lo que quedas parado en esta última.

git checkout -t origin/Nombre de la rama = Si existe una rama remota de nombre “nombre\_rama”, al ejecutar este comando se crea una rama local con el nombre “nombre\_rama” para hacer un seguimiento de la rama remota con el mismo nombre.

git checkout Nombre de la rama a la cual quiero cambiar = Auto descriptivo

git checkout Nombre del commit Nombre del archivo = para ver cómo era un archivo antes  
(también se puede con “git checkout master Nombre del archivo” aunque no es lo mismo, ojo)

git diff Nombre del archivo = para ver las diferencias hechas en los archivos  
(se puede hacer también con “git diff Número del commit 1 Número del commit 2”)

git show = todos los cambios históricos realizados  
(la misma función se puede realizar con el comando git log -nombre de archivo-)

git show-ref --tags = para ver a qué commit está conectado un Tag

git show-branch = para ver todas las ramas que existen y su historia  
(“git show-branch --all” = lo mismo pero más detallado)

git pull = Para traer los archivos o cambios a un servidor remoto.

git log --oneline = Muestra en una línea los commit realizados

git reset --hard “Número del commit” = Viajamos al commit en específico “Número del commit” y eliminamos los cambios futuros

git push = subir todos los cambios locales al servidor remoto de github

git push origin Nombre de la rama = Commitea los cambios desde el branch local origin al branch “nombre\_rama”.

git push origin --tags = para enviar todos los tags al repositorio

git pull = Cuando realizamos cambios directamente en github pero no de forma local para descargar los cambios realizados

git pull origin master = para hacer un pull desde el origin remoto (en rama master) y luego el merge a la rama

git fetch = compara los archivos locales con los del servidor  
(si hay alguna diferencia nos pediría realizar un get pull para hacer un match de los archivos locales)

git tag Nombre de la version -m “Descripción de la versión” = para hacer versiones de nuestro proyecto.

git tag -d Nombre del tag = para borrar un tag

git push --tags = subir los tags

git branch = para ver todas las ramas

git branch Nombre de la rama = para crear una rama

git branch -a = Lista todas las ramas locales y remotas.

git branch -d Nombre de la rama = Elimina la rama local con el nombre “nombre\_rama”.

git branch -d Nombre de la rama = para borrar una rama

git merge Nombre de la rama a unir = para unir una rama a la rama “master”  
(para eso hay que ubicarse en la rama “master”, no desde la rama a unir)

git remote prune origin = Actualiza tu repositorio remoto en caso que algún otro desarrollador haya eliminado alguna rama remota.

git reset --hard HEAD = Elimina los cambios realizados que aún no se hayan hecho commit.

git tag -a Nombre del Tag -m “Mensaje del Tag” Número del Tag = para los Tags

gitk = ver todo de manera más agradable a la vista