**Comandos básicos para Git**

*(Es un software de control de versiones diseñado por Linux)*

**$ git init** : comando que permite iniciar git. Esto crea un nuevo subdirectorio llamado .git que contiene todos los archivos necesarios del repositorio. Crea un repositorio

**$ git status** : comando que permite saber el estado de nuestro repositorio. Determina que estado tienen los archivos es el comando git status.

**$ git add** : este comando actualiza el índice con el contenido actual que se encuentra en el árbol de trabajo, para preparar el contenido organizado para el próximo commit. Este se realiza antes de hacer un commit.

***$ git add .*** : colocar todos los archivos modificados al índice.

***$ git add <archivo>*** : coloca solo el archivo mencionado al índice.

**$ git commit** : permite guardar los cambios realizados en el registro del repositorio.

***$ git commit –amend*** : permite guardar el cambio en el utimo registro guardado sin necesidad de crear uno nuevo registro en el repositorio.

**$ git show <código del commit> :** muestra la infomación del commit

**$ git log** : muestrar los commit registrados o historial de commit log.

***$ git log –oneline*** : muestra los commit en una línea y su código de 7 digitos de identificación

***$ git log –oneline | cat*** : para sistemas operativos unix

***$ git log --oneline -–decorate*** : sirve para ver el nombre de los punteros en el repositorio o commit. Ejemplo: (HEAD -> master)

***$ git log --oneline –graph*** : muestra la relación de un commit y los que lo acompañan ejemplo

\* 62c566c Estamos en construcción:

\* 05c64fc Inicia el repositorio el git de mi proyecto

El commit 62c566c es el commit sucesor de 05c64fc y el commit 62c566c es el commit raíz ya que no tiene sucesor

***$ git log --oneline --decorate –all*** : permite ver todos los commit

***$ git log --oneline --decorate --all --graph***

**$ git diff** : muestra una serie de cambios realizados a los archivos del proyecto, cambios que no estén en el staging o índice. Muestra los cambios entre el árbol de trabajo y el índice o un árbol, los cambios entre el índice y un árbol, los cambios entre dos árboles, los cambios entre dos objetos blob o cambios entre dos archivos en el disco.

***$ git diff HEAD~<numero>*** : sintaxis para referir a los commit anteriores al HEAD ejemplo: $ git diff HEAD~1 HEAD ( ~ significa anterior )

**$ git checkout** : comando que permite realizar varias tareas

***$ git checkout -- <archvio>*** : permite deshacer los cambios en el directorio de trabajo. Deshace la última modificación del archivo esto en el caso que el status este en rojo.

*Ejemplo: $ git checkout -- index.html*

**$ git reset HEAD <archvio>** : permite deshacer los cambios, esto a nivel que se haya ejecutado $ git add. Al hacer este cambio ya se puede utilizar el $ git checkout -- <archvio>

**$ git reset <código del commit>** : permite borrar un determinado commit, en este caso debe de exitir el commit, hay que tener cuidado porque este comando es dañino o destructivo y borra en caso de trabajar local no hay problema pero en caso de trabajar con un grupo en la red puede ocasionar problemas al sincronizar commit de otras personas para ello se pide utilizar otro comando.

***$ git reset --hard <código del commit>*** : elimina el commit tanto del staging o índice como del cambio local

***$ git reset --soft <código del commit>*** : elimina el commit y deja la información en el índice o staging

**$ git revert HEAD** : revierte los cambios de un commit especifico

***$ git revert --no-commit HEAD*** : revierte el cambio y se muestra en el índice o staging dejando el mismo commit

***$ git revert –-continue*** : comando que permite teminar la reversión realizada con el comando anterior.

**$ git branch** : una de las cosas que podemos es ver una lista de ramas

***$ git branch <nombre de la rama>*** : crea una rama con determinado nombre

***$ git branch -m <nombre de la rama> <nombre de la nueva rama>*** : permite cambiar el nombres de las ramas

*Nota: para cambiar nombre de la rama y eliminar rama debo estar en la rama master*

***$ git branch -d <nombre de la rama>*** : elimina una rama

***$ git checkout <nombre de la rama>*** : permite cambiarse de rama

***$ git checkout -b <nombre de la rama>*** : permite crear y cambiar de rama de manera inmediata. –b es el parámetro que indica crear u cambiar de rama

**$ git merge <nombre de la rama>** : comando que permite fusionar las ramas (merge)

**$ git merge –abort :** permite deshacer la fusión el merge

**$ git config --global alias.<nombre del alias>** **'<comando original>'** : permite escribir un alias del comando original, esto es útil reducir el teclear el comando original. Ejemplo

$ git config --global alias.lodag 'log --oneline --decorate --all --graph'

**$ git config --global --get-regexp alias :** permite consultar el todos los alias creados en git. Tomando en cuenta que hay que colocar la palabra alias como se encunetra en el comando

**$ git config --glogal --unset alias.<nombre del alias> :** elimina un alias creado. Ejemplo:

$ git config --global --unset alias.lodag

**Tag ligeros**

**$ git tag :** permite crear una etiqueta, esto es uno de los uso del comando tag hay muchos más ejemplo:

**$ git tag :** permite ver todos los tag

**$ git tag -l :** permite ver todos los tag

**$ git tag -l "v0.1.\*" :** permite ver de manera filtrada los tag determinados

**$ git tag <nombre de la etiqueta> :** crea un etiqueta en el último commit realizado en el más actualizado

**$ git tag <nombre de la etiqueta> <código del commit> :** crea una etiqueta en un determinado commit

**$ git tag –d <nombre del etiqueta> :** elimina, borra una etiqueta

**Tag anotadas compuestos**

**$ git tag –a <nombre de la etiqueta> :** crea una etiqueta anotada con el subcomando –a

**$ git show <nombre de la etiqueta> :** muestra toda la información del tag o etiqueta

**$ git stash** : permite guardar los archivos del índex antes de hacer un commit, esto en caso que para el trabajo para hacer otro sin perder ni modificar los archivos de índex

**$ git stash list :** muestra todos los stash realizados, todo lo guardado

**$ git stash apply :** aplicar o recupera los cambios anterior guardados

**$ git stash pop :** aplicar o recupera los cambios anterior guardados y borrar el stash

**$ git stash drop :** elimina los stash guardados del ultimo al primer stash guardado

**$ git stash save “<mensaje>” :** guarda un stash con un mensaje

**$ git stash –h :** muestra todos los subcomando y atributos del comando stash