**Git**

**Introducción**

Tiene tres estados: Working directory, staging área, repositorio

**Flujo de trabajo**

**1.-** Modificar, crear, editar una serie de archivos.

**2.-** Escoger los archivos que están listos

**3.-** Registro de nuestro proyecto.

**Primeros Pasos**

* Instalación<https://git-scm.com/>
* Configuraciones básicas: Poner el sig comando para poner nuestro nombre: **git config --global user.name “cesar”**
* Ver nuestro nombre: **git config --global user.name**
* Editar el nombre poniendo el mismo comando **git config --global user.name “juan”**
* Configurar nuestro correo: **git config --global user.email “**[**celuis20@hotmail.com**](mailto:celuis20@hotmail.com)**”**
* Consultar nuestro mail:  **git config --global user.email**
* Cambiar de colores las indicaciones que nos hace git: **git config --global color.ui true**
* muestra todas nuestras configuraciones que hemos realizado: **git config --global –list**

**Nuestro Primer proyecto**

**Comandos:**

**git init:** marca el inicio de nuestro proyecto, aquí le decimos a git que empiece a monitorear todos nuestros cambios, este comando se usa una vez.

**git status:** para ver el estatus de archivos agregados

**git add:** agregar archivos y luego poder hacer un coming ej: **git add index.html**

**git add –A:** agregar todos los archivos

**git commit –m “Mensaje”:** guardamos los cambios con un mensaje para identificarlo

**Todos los comandos**

git init // inicia proycto seuusa solo una vez   
git status // muestra lo que ya subimos o lo que nos falta por subir   
git add // adiciona carpetas   
git commit -m "mensaje" /// asigna cmentarios de lo que cambiamos   
git log /// muestra todos los commit que hicimos   
git checkout “Pegamos el commit”  // viajamos a través de nuestros commit o nuestras ramas. Ej: **git checkout f06a8d4252cd185024a8b8ddd6bb32514dce1ac5**

git checkout master //me mueve al ultimo comming que tenemos  
git reset –soft f06a8d4252cd185024a8b8ddd6bb32514dce1ac5  //no borra código, se usa como el checkout  
git reset --mixed // avanzado   
git reset --hard f06a8d4252cd185024a8b8ddd6bb32514dce1ac5  //borra todo

**Recuperar un archivo borrado o volver a una versión posterior a un commint**

*# Vemos el log de los commit:*

git log --pretty=oneline

*# Se muestra algo como ésto*

5542cd7844e3a035542cd7848c19482029963813 cambios 3

0bbfc3d4b41c8cb572ad780a346ddfbb0f5cfed5 2

f03af74db09f6f1802024af5c1371756b5b9a557 1

f64da54ea58649f863cbddfdeadf67dbd046d7d8 :boom::camel: Added .gitattributes & .gitignore files

*# Si queremos volver a como estaba todo cuando hicimos el commit 1 (perdiendo lo posterior al commit 1)*

git reset --hard f03af74db09f6f1802024af5c1371756b5b9a557

**4.- Ramas y fusiones**

**Head:** Es el commit donde nos encontramos ahora**. Ej: commit f06a8d4252cd185024a8b8ddd6bb32514dce1ac5**

**Ramas:** es una línea de tiempo en nuestro proyecto y nos sirve para arreglar errores, experimentar, hacer grandes cambios

**Rama master:** es la rama donde comenzamos a trabajar, es la rama principal y estable de nuestro proyecto

**$ Git branch //nos muestra las ramas que tenemos creadas**

git branch “nombre de la rama” //crear una nueva rama ej: git branch test

git checkout test // nos sirve para movernos entre ramas

git branch –D //forza el borrado de una rama

git branch –d // borra un rama

**Fusiones**

La rama test es solo para pruebas podemos fusionar la rama test con la master pasando todo a master

**Fusión** es la creación de un nuevo commit juntando una rama con otra

**PASOS PARA HACER LA FUSION ENTRE RAMAS**

**Primero nos situamos en la rama que va a absorber o donde se va insertar los commit de la otra rama.**

**1.- git checkout master**

**2.- git merge “rama” ejm: git merge test //comando para realizar la fusión entre ramas**

**Master es la rama principal o de producción, la rama test es una rama de prueba**

**Procesos para la fusión:**

**Existen 2:**

**Fast-forward //simple y automático, no pregunta nada**

**Manual merge //largo y manual; la fusión tiene que pasar por nosotros y nosotros elegimos los archivos correctos para crear el nuevo commit**

**Recuperar Rama borrada**

**Verificamos el último commit**

cesar@TECNICO1 MINGW64 ~/Desktop/Cesar Hurtado/Cesar Hurtado/Git (master)

$ git fsck --lost-found

Checking object directories: 100% (256/256), done.

dangling blob 89c84a6366ba188bca452b73d08d99dd7349626c

dangling commit 46e81ac3de16b610be35d96fbba040669a432a6a

**Vemos que hay dentro de ese commit**

$ git show 46e81ac3de16b610be35d96fbba040669a432a6a

commit 46e81ac3de16b610be35d96fbba040669a432a6a

Author: cesar <celuis20@hotmail.com>

Date: Tue Mar 7 12:22:18 2017 -0500

cambio fondo y letra

**diff --git a/index.html b/index.html**

**index 60c4e81..cb919a4 100644**

**--- a/index.html**

**+++ b/index.html**

@@ -6,4 +6,14 @@

<body>

<h1>Prueba nueva</h1>

</body>

+<style>

+body{

+ background-color: black;

+

+}

+h1{

+

+ color: #FFFFFF;

+}

:

**Recuperamos el commit ya verificado**

$ git branch recuperado 46e81ac3de16b610be35d96fbba040669a432a6a

**Ver los cambios, nos movemos a la rama recuperado**

$ git checkout recuperado

**Eliminar ramas que no nos sirven**

Git branch –D “rama a eliminar”

**Crea una Rama y muévete a la vez con el comando**

$ git checkout -b texto

**Problemas:**

**Cuando se bloquea un directorio en git; borrar directorio index.lock**

rm -rf .git/index.lock

**Referencias:**

**https://manuais.iessanclemente.net/index.php/Control\_de\_versiones\_con\_Git\_y\_GitHub**