GIT



Cada punto es un desarrollo distinto, en cada uno o se agrego un nuevo modulo, una nueva funcionalidad, etc.

Cuando se separa de master, es porque algun programador “copia” del programa y comienza a trabajar en el proyecto en paralelo. Cuando termina con esa nueva funcionalidad, vuelve a fusionarse con el master.

CONCEPTOPS CLAVE

SOURCE: git-scm.com

**Snapshots:** crear un punto. Se puede volver a cualquier punto hacia atras.

HAY TRES ESTADOS

working directory: en donde estaré trabajando con mis proyectos.

staging area: a medida que voy terminando un proyecto, los voy pasando al staging area.

repository: cuando me decidi a guardar un cambio, lo paso al repository.

COMANDOS BASICOS

Con la consola del sistema operativo.

**git init:** para empezar a utilizar git en un proyecto.

**git add<file>:** para pasar los archivos del working directory al staging area.

**git status:** ver en donde estan los archivos (si en working directory, staging area, etc.).

**git commit:** pasa el archivo del staging area al repository (crea un primer snapshot).

**git push:** sube el archivo a un repositorio remoto. Se sube a un servidor para que pueda ser accedido por otros desarrolladores que tengan permiso para modificar el código.

**git pull:** en caso de estar trabajando con otros desarrolladores, te traes los cambios que hayan realizado en el archivo.

**git clone:** crear una copia del codigo que se encuentra en el servidor central al pc.

NOTA: A traves de Git CMD, los comandos que se utilizan en Linux tambien serviran en este lugar.

**ls:** comando para listar directorios y archivos del lugar en donde me encuentro.

**pwd:** muestra el directorio en el que estoy.

**cd nombreDeLaCarpeta:** entro en la carpeta que se ubique en el directorio en el que estoy.

PASOS

A traves de la terminal de Git CMD, que por defecto comienza en el ~~escritorio~~, voy entrando en carpetas hasta llegar a la carpeta en donde esta mi proyecto. Una vez accedida, en la terminal escribo git init.

Esto creará una carpeta oculta y en VSCode aparecera como .git.

Si le diera git status, veria que me avisa de Untracked files, y que use git add para agregar los documentos al stage area.

Ahora luego de agregar un archivo si escribo git status, me mostraría ese archivo en verde, y tambien de utilizar el comando git rm para quitarlo de ahi.

Crear un primer snapshot:

Una vez tengo los archivos en el stage area, cuando quiera pasarlos al repository (git commit), me preguntara: Please tell me who you are, esto para saber qué desarrollador está subiendo los cambios.

Para configurar esto, en la consola aparecerá el comando:

**git config --global user.mail “escriboElEmail”**

NOTA: De llegar a escribirlo mal, simplemente vuelvo a typearlo.

Ahora el mail quedará configurado para el usuario que va a realizar los cambios.

Configurar usuario:

**git config --global user.name “nombreDelUsuario”**

Ahora si, con git commit, me enviara al “bean”



NOTA: Puede que una vez dentro necesite presionar la tecla i para comenzar a escribir. (Debajo aparecera un mensaje que dice -- INSERTAR --)

**Escribo un mensaje, ej: mi primer commit, presiono escape y donde decia insertar escribo :wq**

Ahora vere en la consola que dira “x files changed.”

Y vere algo parecido a un hash con numeros (id) asociados a los archivos.

Yo **podria ahora escribir git log** y me saldra en pantalla informacion acerca de los commits.

Si yo ingresara por fuera de esa consola en algun archivo y le cambiara/agregara algo, escribiendo **git status** me avisaria en la consola que el archivo fue modificado.

Con **git checkout -- nombreDelArchivo**

Los cambios realizados en ese archivo luego del commit, se borrarian y el archivo volveria al estado que tenia en su ultimo snapshot.

Si realizo cambios en el archivo, con **git diff**

Veria los cambios realizados desde el ultimo snapshot en la consola. En verde aparecera lo que he agregado y en rojo lo que he quitado.

Realizar un nuevo snapshot del archivo:

**git add nombreDelArchivo ó git add .** (agrega todo de una)

**git commit**  ó **git commit –m “el comentario”** (evita el editor)

Aparecera el editor para que nuevamente escriba un comentario/mensaje (primero presionando la tecla i), presiono escape, debajo escribo **:wq**

Aparecera en la consola avisando que un nuevo cambio fue realizado.

Con git log, veria que aun esta la foto del codigo anterior

NOTA: Estoy parado actualmente en donde diga (HEAD => master).

Ignorar un archivo:

En la carpeta del proyecto, creo un archivo que se llamará **.gitignore**

Dentro de .gitignore escribo nombreDeLaCarpeta o nombreDelArchivo.TipoDeArchivo (ej. text.txt).

Este .gitignore deberia agregarlo al directory: git add .gitignore y git commit

Crear una version alternativa del proyecto (una rama):

**git branch unNombre**

Ahora si escribo git branch

apareceria en la consola unNombre

master

Para ingresar a esa version unNombre: **git checkout unNombre**

Para volver a la rama master: **git checkout master**

**IMPORTANTE:** Al hacer esto veria como en mi editor de codigo, las cosas que habia realizado en el branch unNombre no estarian mas. Habria ingresado nuevamente al archivo en la rama master.