**Edgar Baizabal Montañez**

**Git/GitHub**

**Git**

Es un sistema de control de versiones. Sirve para manejar cambios en el código fuente:

* Save code snapshots (commits)
* Work with alternative code versions (“branches”)
* Move between branches & commits (“checkout”)

Con git se puede hacere fácilmente rollback a snapshots de código anterior o desarrollar nuevas características sin romper el codigo

**GitHub**

Es una nube de repositorios Git y servicios para el código que se maneja en la nube (repositorio solo es un espacio donde se almacena código). Cada proyecto de Git se puede conectar con Github (de la nube a local)

* Descargar o subir a la nube (push and pull)
* Desarrollo colaborativo
* Automatizacion CI/CD

**Git basics**

**Crear repositorio de Git**

Repositorio: Es un espacio donde se almacena código (folder) usado por Git para hacer el tracking de todos los cambios de un proyecto.

Se crean los repositorios de git con **git init.** Solo se requiere correr este comando una vez por folder.

**Commits**

Snapshot: copia puntual de archivos o datos en un momento concreto

**git add <file(s)>** se añaden todos los archivos que se agregaran a ese snapshot (cuando se añade un feature o se arregla un bug etc, se crea un nuevo snapshot). Al crear un nuevo snapshot se puede crear un nuevo commit.

git add . : git add mas el punto añade todos los archivos de el proyecto. Tambien se le puede hacer git add a una carpeta escribiendo el nombre de la carpeta.

**Este proceso se llama staging (staging area)**. En ese paso tu escoges los cambios que se harán para el siguiente commit

**git restore --staged .** :elimina todos los archivos que esten en el staging area

**git commit:** crea un commit que incluye todos los staged changes.

git add <files(s)> + git commit se ocupa para que tu puedas escoger que archivos van a ser de un snapshot y que archivos de otro.

git commit -m 'se añade un comentario para saber de que es el commit'

**Git status:** sirve para ver cuales son los archivos a los que se les va a hacer commit y están en el staging área.

**Git log:**

Texto

Descripción generada automáticamentesirve para ver darnos información de los commit (el mensaje, el autor, etc)

git log -1: nos muestra el ultimo commit, mientras que con -2 nos muestra los ultimos dos commits.

git log -p: nos da mas información del commit, es decir muestra con un + las lineas agregadas y con un - las lineas elimindas. Shows the asociated patches

**Apuntador Head**



Apunta al commit que es actualmente cargado. Una vez que tienes mas que tienes mas de un commit puede empezar a moverte entre los commits, por eso es importante el HEAD.

**git branch**: te despliega todas las Branch que hay en tu proyecto

**git checkout <id> (se pone sin los <>)**

Hace que te muevas temporalmente a otro commit. Si estas en el commit 6, te puedes mover al commit 4 y cargar todo el contenido del commit 4, pero los otros commits no se borran, solo no son visibles al momento

Importante: una vez que cargues el nuevo commit se van a ver ahora los cambios de ese commit en tu carpeta

Al ser un comand pormpt (git bash) usar control copy y control paste con clic derecho, no usar los atajos de ctrl+u y ctrl + v

**Importante: para regresar al commit principal se ocupa “git checkout master”**

**git revert <id>:** revierte los cambios de un commit . Haciendo un nuevo commit. Eso quiere decir que siempre puedes regresar a cambios desechos

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

**git reset --hard <id>** : aquí borras todos los commits de forma permanente que se hicieron después del commit del que se inserte el id. No se recomienda mucho porque borra para siempre

**Resumen de los key commands**

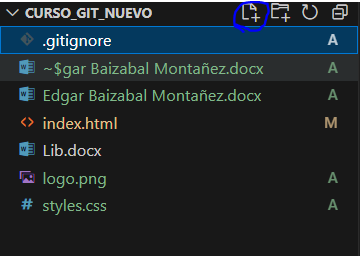
Texto

Descripción generada automáticamente con confianza media

**git ignore**

para añadir el archive .gitignore. Puede hacerse desde VS Code.

Solo dandole clilc en el icono de añadir nuevo archive y escribiendo .gitignore



git ignore es un archive de texto donde se va a escribir el nombre de todos los archivos del

Proyecto a los que no se les quiere hacer git add. Es decir que no los tome en cuenta para el git add

**Git branches**

Un branch es un contenedor que contiene muchos commits.

**git branch <name>** : toma el ultimo commit (el apuntado por el apuntador HEAD). Este commit es el punto de inicio de la nueva branch

Todos los commits y cambios que se hagan en una branch no afectaran a otra y no se veran en la otra

**Gir merge <name>:** junta los commits de dos branches. SI hay conflictos al unir el codigo eso ya se tiene que arreglar de forma manual

para crear nuevo archivo en la carpeta, desde git bash -> touch <nombre del archivo>

copia puntual de archivos o datos en un momento concreto

Un repositorio git tiene:

-working tree

-directorio git

-staging area

Etapas de los archivos al manejarse con git

Modifying -> staged -> Commit -> save in gut directory

Modifiying: agregar,eliminar, modificar al arhcivo

Staged: aqui se marcan todos los cambios para el rastreo y el archivo esta listo para que pueda ser rastreado en modificaciones. Entonces de aqui se hace el commit

Commit: se guardan los cambios en el directorio de .git

--------------------------Comando Git-----------------

git init -> iniciualiza un repositorio en la direccion dada por cd

carpeta de .git -> guardas los cambios de lso proyectos y el historial de cambios

cuando se clona una carpeta , se clona este repositorio.

El area afuera de .git es el el arbol de trabajo. El arbol de trabajo es la version

actual del proyecto

git add <nombre del file> -> git add will add a file to the staging area. Esta es el area donde se reserva todo lo que se queire gaurdar en el siguiente commit. Desde aqui se les va a dar le seguimeintoas los archivos.

git add -p -> nos muestra lso cambios hechos antes de agreegarlos al staged area y nso pregunta si los queremos agregar al statged area. Esto es porque muchas veces no se esta seguro de todo lo que se hizo antes del hacer un commit

git status -> da informacion del arbol actual y ver cuales archivso estan en el staging area

git commit -> con este comando le decimos a git que queremos cgaurdar los cambios. commit es lo equivalentee a save

commit se refiere a que se guarda un estado de lso arhcivos en ese momento

git commit -m 'se añade comentario desde la terminal'

git commit -a -> A shortcut to stage any changes top tracked files and commit them in one step. It skip the staged step of the files that are already benn tracked, not the new files

para nuevos archivos se tiene que si usar gity add primero para el track

git commit -a -m "se puede añadir el mensaje aqui"

git commit --amend -> esto es para añadir mas arhcivos al commit, una vez que ya se relaizo el commit, por si es que se olvido uno. A lso arhcivos pirmero se les debe de poner add para añadirlos al staging area

es importante no usar este comundo en repositorios publicos ya que que te modifica el ID delcommit a uno nuevo

git revert <ID del commit> -> permite revertir un commit hecho y vuelve a genrar uno nuevo con un nuevo mensaje y se le asigna un nuevo ID

git log -> git log will give us information about the author of each commit, its timestamp, and each commit message. Para salirte del historial se presiona "q"

git log -1 -> nos muestra el ultimo commit, mientras que con -2 nos muestra los ultimos dos commits.

git log -p -> nos da mas información del commit, es decir muestra con un + las lineas agregadas y con un - las lineas elimindas. Shows the asociated patches

git show <se pone el ID del commit que aparece asociado a cada uno cuando se muestran con git log>->Taking the commit ID, git show will show information about the commit and its associated patch.

git diff -> it shows only unstaged changes by default

git diff --staged -> it is used to see thee changes that are staged but not commited.

git rm -m 'añadir mensaje' -> rmeove files, lo borra de la carpeta

git mv -> <nombre del archivo> <nombre del arhcivo al que se quiere cambiar el nombre>

git checkout <nombre del arhcivo> -> nos permite regresar a una version anterior del archivo. Eso es para los archivos que no estan en el staging area

git reset <nombre del archivo> -> quita el archivo especificaod del staged area

git reset -p -> con esto git te pregunta a q que cambios se les quiere hacer el reset

git revert HEAD -> esto borra el ultimo commit que se hizo o el commit donde este el apuntador HEAD. it creates a new commit with inverse changes

git revert <ID del commit> -> esto es para que seal cualquier commit y no el de HEAD

git add\* -> añade todos los archivos del working tree al staged area

ls -l -> da la lista de todos los archivos contenidos

ls -la -> esto muestra absolutamente todos los archivso. EN linux los archivos con prefijo . no se van a mostrar con ls -l

.gitinore -> es un archivo en el que se le dice a git que archivos ignorar para el repositorio acutal en el que se este trabajando. file specifies intentionally untracked files that Git should ignore. Files already tracked by Git are not affected; see the NOTES below for details.

Tambien se tiene que hacer el commit el add y commit del git ignore

HEAD -> Git uses the HEAD alias to represent the currently checked-out snapshot of your project

-----------------------------------------------------BRANCHES----------------------------------------------

Bracnh -> pointer to a particular commit, represent an independent linea of a project.

The default bracng that git creates when a new repository is initilizez, is called master.

git branch -> da todas las branch que tenemos

git branch <Nombre de la nueva branch>

git checkout -> we use git checkout to check out the latest snapshots for both files and for branches

git checkout new-feature -> cambia a la rama new-feature

git checkout -b <nombre del branch> -> se crea y se cambia a ese branch

Si cambiamso de rama, se podra ver con ls -1 los archivos que estan ubicados sólo en esa rama.

git branch -D <nombre del branch> -> elimina el branch seleccionado

Merging : The termn that git uses for combining branched data and hsitory together

git merge <nombre del brach al que se quiere unir> esto es apartir del branch en el que s est ahorita, que seri ale master

git log --graph --online -> despliega a modo de ramificacion toods los commits, incluyendo slos merge de las ramificaciones

git merge --abort -> elimina el merge y regresa lso archivos al commit anterior anmtes de que el merge ocurriera

------------------------------------------------------GitHub-----------------------------------------

git clone <url del repositorio> -> nos permite crear una copio del reposotiro de GitHub en la computadora.

git push -> envio los commit que se hayan echo en el repositorio local, hacia el repositorio remoto en GitHub.

The git push command gathers all the snapshots we've taken and sends them to the remote repositor

git pull -> se jalan los nuevos cambios hechos desde GitHub

git remote show origin -> nos enseña toda la informacion que se tiene del repositorio remoto

git branch -r > con esto se ven las branches de nuestro repositorio remoto. Cuando se traaja con un repositorio remoto, los cambios y modoficaciones a nuestro trabajo se van guardando en las branches.

git fecth -> This command copies the commits done in thge rmeote repository to the remote branches. So we can see what other people have commited