**GIT**

Para que git monitoree los archivos tienes que poner:

Las iniciales **cd** luego el disco donde tienes la carpeta **e** con barra **/** luego en **nombre de la carpeta** nuevamente barra **/** **nombre de la carpeta que quieres monitorear**, al finalizar cierras con **barra /**

Ejemplo:

cd e/cursos-ORACLE/prueba-git1/

para ver si lo monitorea escribir **ls**

te va a aparecer el nombre del archivo

luego escribes **git init**

y te va a salir que se inició un repositorio o sea que está monitoreando el archivo.

Para saber y estar seguros de que git esta viendo los archivos y modificaciones escribimos: **git status**

**Te va a salir esto:**

$ git status

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)

git documents.docx

index.html

~$t documents.docx

~WRL2244.tmp

Ahora para que empieze a monitorear el archivo tienes que escribir **"git add <file>..."**

**Git add index.html**

Te va a aparecer el archivo monitoreado en verde

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: index.html

Untracked files:

(use "git add <file>..." to include in what will be committed)

git documents.docx

~$t documents.docx

~WRL2244.tmp

Y lo que no monitorea en rojo.

Para eso escribimos: **git add .**

Y en la pantalla no te va a salir nada pero si digitas: **git status**

Puedes ver que te sale todos los archivos con verde, eso significa que está monitoreando todas los archivos de la carpeta.

$ git status

On branch master

No commits yet

Changes to be committed:

(use "git rm --cached <file>..." to unstage)

new file: git documents.docx

new file: index.html

new file: ~$t documents.docx

new file: ~WRL2244.tmp

para dejar de monitorear algún archivo tenemos que escribir: "git rm --cached <file>..."

para guardar alguna modificación del archivo digitalizamos: **git commit –m “le ponemos un mensaje referente al archivo y modificaciones, no tan largo, con comillas”**

te va a salir:

$ git commit -m "Creando archivo index.html con listas de cursos"

[master (root-commit) 048ab02] Creando archivo index.html con listas de cursos

4 files changed, 15 insertions(+)

create mode 100644 git documents.docx

create mode 100644 index.html

create mode 100644 ~$t documents.docx

create mode 100644 ~WRL2244.tmp

por si git no esta monitoreando el archivo escribir:

**git config –-global user.email “correo”** enter

luego

git config –-global user.name “bertha” enter

Antes de cualquier interacción con git, es necesario informar quién eres para que almacene correctamente los datos del autor de cada uno de los cambios en el código.

git config --local user.name "Tu nombre aquí"

git config --local user.email "Tu@email.aqui”

Al ejecutar el comando git status, recibimos información que puede no ser tan clara, especialmente cuando nos encontramos con términos como HEAD, working tree, index, etc.

Solo para aclarar un poco, ya que entenderemos mejor cómo funciona Git durante el curso, aquí hay algunas definiciones interesantes:

* HEAD: Estado actual de nuestro código, es decir, donde nos colocó Git
* Working tree: Lugar donde los archivos realmente están siendo almacenados
* index: Lugar donde Git almacena lo que será *commiteado*, es decir, la ubicación entre el *working tree* y el repositorio de Git en sí.

Además de eso, los posibles estados de nuestros archivos son explicados con detalle en este link: <https://git-scm.com/book/es/v2/Fundamentos-de-Git-Guardando-cambios-en-el-Repositorio>.

Créame, aunque parezca confuso ahora, ¡durante el curso todo tendrá mucho más sentido! :-D

Cunado usamos local es para un archivo y cuando usamos global es para todos los archivos

Para limpiar la pantalla escribimos

Clear y enter, aparecerá vacio y podemos seguir escribiendo cosas.

Cuando escribimos: git log --oneline

$ git log --oneline

4f97f0e (HEAD -> master) arreglado documents git

048ab02 Creando archivo index.html con listas de cursos

Nos aparece un detalle más corto de nuestros archivos

Para ver a detalle las modificaciones de todos los archivos escribimos: **git log -p**

Y nos va a aparcer:

$ git log --p

fatal: unrecognized argument: --p

Pc@DESKTOP-PUE89RR MINGW64 /e/cursos-ORACLE/prueba-git1 (master)

$ git log -p

commit 4f97f0e73e19579cb6578e36888a79be69794123 (HEAD -> master)

Author: bertha <berthahuanca889@gmail.comgit config --global user.email berthahuanca8890@gmail.comgit status>

Date: Sat Sep 10 11:14:15 2022 -0400

arreglado documents git

**diff --git a/git documents.docx b/git documents.docx**

**index 7c20167..7246521 100644**

**--- a/git documents.docx**

**+++ b/git documents.docx**

@@ -23,4 +23,49 @@ Untracked files:

~WRL2244.tmp

Ahora para que empieze a monitorear el archivo tienes que escribir "git add <file>..."

Git add index.html

-fds

+Te va a aparecer el archivo monitoreado en verde

+$ git status

+On branch master

+

+No commits yet

+

+Changes to be committed:

+ (use "git rm --cached <file>..." to unstage)

+ new file: index.html

+

+Untracked files:

+ (use "git add <file>..." to include in what will be committed)

+ git documents.docx

Al dar enter te muestra todas las modicicaciones en los archivos y para salir de eso tienes que poner;

**:q** y enter y finamente podrás salir de eso.

git log --pretty="format:%h %s"

Como vimos en el último video, podemos ver el historial de cambios de nuestro proyecto de manera muy sencilla, usando el comando git log.

Aunque es fácil, este comando es muy poderoso. Ejecuta git log --help y ve algunas de las posibles opciones. Para obtener algunos ejemplos más fáciles de entender, puedes buscar sobre git log o echar un vistazo a este link: <https://devhints.io/git-log>.