TUTORIAL DE GITHUB

https://guides.github.com/activities/hello-world/

¿Que es GitHub?

GitHub es una plataforma de desarrollo colaborativo para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de computadora.

Este tutorial te enseñará lo esencial de github tales como:

Crear repositorios, ramificaciones (*branches), confirmar (commits), etc.*

Para completar este tutorial, necesitas de una cuenta en github GitHub.com account y acceso a internet.

Para comenzar a usar GIT

Es indispensable tener la aplicación git instalada en la computadora y una cuenta creada en github.com para poder comenzar a trabajar con esta herramienta, una vez ya creada iremos a la aplicación llamada git bush y allí colocaremos el primer comando que sirve para introducir el nombre de usuario en la aplicación y el email.

Comando nombre de usuario: *git config --global user.name "usuario"*
Comando E-mail*: git config --global user.email email@ejemplo.com*

Al concluir estos pasos se deberá crear una carpeta que será la que va a contener todos los repositorios que nosotros coloquemos en este caso la carpeta llevará el nombre de "Repositorios" que estará en el escritorio.

Comando para ubicarnos en el escritorio: *cd desktop*
Comando para ubicarnos dentro de la carpeta: *cd Repositorios*

Una vez que estemos dentro de la misma debemos usar el comando *git init* para crear un repositorio en la misma, con este comando se creara

una carpeta .git con un repositorio completamente vacio.
Ya en este paso podremos comenzar a colocar nuestros proyectos dentro de la carpeta

Comando para añadir el proyecto a git: *git add nombreproyecto.formato*

Al concluir esto deberemos corroborar el estado del proyecto agregado utilizando el comando *git status* en el cual nos indica en cual rama se encuentra el repositorio y el estado del proyecto
Una vez finalizado esto se deberá hacer un commit, que se refiere al mensaje de confirmación para este proyecto con el comando:
git commit -m "commit para el proyecto"

Al estar con el commit colocado el trabajo ya estará disponible para subirlo a la página github con los siguientes comandos

*git remote add *nombreproyecto* *link obtenido de las instrucción de github**
*git push -u *nombreproyecto* master*

Al hacer esto nos pedirá nombre de usuario y contraseña para poder habilitar al git a subir ese archivo al repositorio de github
Por otra parte git también ofrece la posibilidad de clonar un repositorio existente por ejemplo para tener un respaldo si deseas editar un proyecto o si deseas contribuir en el mismo, además de que también se utiliza para clonar librerías o proyectos desde la página web github

Comando git local: *git clone*
Comando git remoto: *git clone [URL]*

Repasando los conceptos básicos de git local

git init: crear nuevo repositorio
git add: añadir nuevo proyecto a git
git commit -m: escribir mensaje de confirmación
git clone: clonar proyecto para poder editarlo o respaldarlo
git status: conocer en qué estado se encuentra el proyecto y en que rama(branch) se encuentra
git log: proporciona una lista cronológica de cambios realizados sobre el proyecto

GIT LOCAL

Es un software de control de versiones diseñado con el propósito de llevar un registro de los cambios en archivos de computadora y coordinar el trabajo que varias personas realizan sobre archivos compartidos.

Cada vez que confirmas un cambio, o guardas el estado de tu proyecto en Git, él básicamente toma una foto del aspecto de todos tus archivos en ese momento, y guarda una referencia a esa copia instantánea. Git maneja sus datos como una secuencia de copias instantáneas.

Casi todas las operaciones son locales

Tienes toda la historia del proyecto ahí mismo, en tu disco local, la mayoría de las operaciones parecen prácticamente inmediatas.

Para navegar por la historia del proyecto, Git lee directamente de tu base de datos local, NO NECESITA CONECTARSE A UN SERVIDOR.

Si quieres ver los cambios introducidos en un archivo entre la versión actual y la de hace un mes, Git puede buscar el archivo hace un mes y *hacer un cálculo de diferencias localmente*, en lugar de tener que pedirle a un servidor remoto que lo haga u obtener una versión antigua desde la red y hacerlo de manera local.

Puedes trabajar tranquilamente en tu repositorio local y luego subir los cambios.

Poner como hacer git pull

Los Tres Estados

CONFIRMADO (committed) Confirmado significa que los datos están almacenados de manera segura en tu base de datos local.

MODIFICADO (modified) Modificado significa que has modificado el archivo pero todavía no lo has confirmado a tu base de datos.

PREPARADO (staged). Preparado significa que has marcado un archivo modificado en su versión actual para que vaya en tu próxima confirmación.

Fundamentos básicos de Git - Obteniendo un repositorio Git

Obteniendo un repositorio Git

Puedes obtener un proyecto Git de dos maneras:

A. tomar un proyecto o directorio existente e importarlo en Git.

B. clonar un repositorio existente en Git desde otro servidor.

Inicializando un repositorio en un directorio existente

Si estás empezando a seguir un proyecto existente en Git, debes ir al directorio del proyecto y usar el siguiente comando: *git init*

Esto crea un subdirectorio nuevo llamado .git, el cual contiene todos los archivos necesarios del repositorio – un esqueleto de un repositorio de Git. Todavía no hay nada en tu proyecto que esté bajo seguimiento.

Luego ponemos: *git add*

Para especificar qué archivos quieres controlar, seguidos de un *git commit* para confirmar los cambios:

al agregar el archivo sintaxis.md, se convierte en verde new file: sintaxis.md (preparado para ser confirmado)

En este momento, tienes un repositorio de Git con archivos bajo seguimiento y una confirmación inicial.

Al confirmar los cambios puedes agregar mensaje, de la cual es una breve descripción de cuáles fueron los cambios hechos, para que otros colaboradores puedan saber ¿qué hiciste? Y ¿para qué?

Ej: $ git commit -m “agregado del archivo sintaxis.md”

Clonando un repositorio existente

Si deseas obtener una copia de un repositorio Git existente — por ejemplo, un proyecto en el que te gustaría contribuir — el comando que necesitas es *git clone*

git clone [url]

Ej: $ git clone https://github.com/terehsieh/TP-2.git

Guardando cambios en el Repositorio

Cada archivo de tu repositorio puede tener dos estados:

- Rastreados (tracked files): Todos aquellos archivos que estaban en la última instantánea del proyecto; pueden ser archivos sin modificar, modificados o preparados
- Sin rastrear: cualquier otro archivo en tu directorio de trabajo que no estaba en tu última instantánea y que no están en el área de preparación (*staging area*).

Cuando CLONAS por primera vez un repositorio TODOS ESTAS RASTREADOS.

Para saber en qué estados están tus archivos puedes usar el comando *git status*

Si agrego un archivo aparece -----"Untracked files" ("Archivos no rastreados" en inglés), son archivos que no tenías en el commit anterior.

Rastrear Archivos Nuevo

Para agregarlos tengo q poner *git add* NOMBRE DEL ARCHIVO

El comando *git add* puede recibir tanto una ruta de archivo como de un directorio; si es de un directorio, el comando añade recursivamente los archivos que están dentro de él.

Preparar Archivos Modificados

git add. git add es un comando que cumple varios propósitos - lo usas para empezar a RASTREAR archivos nuevos, PREPARAR archivos, y hacer otras cosas como MARCAR COMO RESUELTO ARCHIVOS en conflicto por combinación.

Ramificaciones en Git

Una rama Git es simplemente un apuntador móvil apuntando a una de esas confirmaciones. La rama por defecto de Git es la rama master. Con la primera confirmación de cambios que realicemos, se creará esta rama principal master apuntando a dicha confirmación. En cada confirmación de cambios que realicemos, la rama irá avanzando automáticamente.

Crear una Rama Nueva

¿Qué sucede cuando creas una nueva rama? Bueno…, simplemente se crea un nuevo apuntador para que lo puedas mover libremente.

Para crear una rama: *git branch* ***nombre de la rama***

Ej: *git branch rama1*

Agregado de rama1 y rama2