Curso de Git

#Que es Git?

Sistema de control de versiones

Nos ayuda a ir formando una linea de tiempo de nuestro proyecto

#Donde puedo usar un sistema de control de versoines

Nos ayuda a hacer trabajos colaborativos

## #GIT EN EL MUNDO

Hola, después de ver algunos vídeos presenciales sobre control de versiones y Git sería bueno que leyeras unas cuantas lineas acerca de lo que representa Git en el mundo del software, Git es usado por las grandes empresas, basicamente, es la herramienta que permite que muchas de las aplicaciones que usas hoy en día sean tal y como las conoces de esta forma, a continuación algunos ejemplos que podrás corroborar cuando visitemos la página de Git para descargarlo, pero de mientras, me gustaría mencionarte esta lista de grandes empresas emplean Git para su realización, seguramente conoces a casi todas.

* Facebook
* Twitter
* Netflix
* Google
* LikedIn

Esas son por supuesto empresas que venden productos de Internet, usarlas es el pan de todos los días en estos tiempos, pero... y si te digo que Git influye en la creación no solo de aplicaciones, si no que también tiene presencia en los equipos de trabajo que desarrollan sistemas operativos para los dispositivos, tales como:

* Microsoft
* Android
* LINUX

Incluso, el software que nos ayuda a crear más software, tales como sistemas de bases de datos, frameworks, IDE's también hacen uso de GIT! Algunos ejemplos:

* Rails (Framework de Ruby para desarrollo web)
* Postgres SQL (Sistema de base de datos)
* Eclipse (Antiguo IDE para crear aplicaciones Android)áá

No cabe duda de que hay muchos ejemplos de equipos de trabajo que usan Git para trabajar, mejor no perdamos el tiempo y empecemos a aprender a usar Git para mejorar nuestra formación profesional como desarrolladores, no solo de software, si no de contenido y productos profesionales para hacer de este mundo un lugar mejor.

#GitHub/ Gitlab

Son redes sociales para los programadores

Puedes compartir con otras personas

TRABAJO A NIVEL LOCAL CON GIT

#Tres áreas en Git

Tiene 3 secciones llamados arboles

1. Directorio de trabajo: Lugar donde esta alojado los archivos del proyecto
2. Index: Es donde se agregan los archivos para que git sepa si hay cambios
3. Git Area: Se alberga el ultimo commit hecho

#Agregando archivos – Tracking files

Git init -> Iniciar Git para crear un repositorio local

Git status -> Ver estado de nuestro repositorio

Git add -> añadir cambios al index

Git add -a para agregar todos los archivos de un directorio al index área

Git rm –-cached ‘nombre archivo’ para eliminar un archivo del index

#Primeros commits y viajes en el tiempo

Un commit es un punto de grabado del proyecto

Git commit -m “comentario”

Git log -> para consultar la lista de commits que se hicieron en mi repositorio

Git Checkout -> para viajar a un commit o una rama en mi repositorio

Git checkout master -> para volver al ultimo commit que hicimos

#Reset

Head -> status actual de mi proyecto, es decir a donde hicimos el ultimo salto o checkout

Master-> el ultimo commit elaborado

Reset -> son saltos como los checkout pero borran todo a su paso

3 niveles de borrado:

* 1. Reset soft -> borra los commits posteriores en el camino. Mantiene intacto mi directorio de trabajo. No toca el código ni el trabajo.
  2. Reset midset -> borra commits a su paso y borra cosas que estén agregadas al index área, pero no las borra de mi directorio de trabajo en forma física. No se usa muy a menudo.
  3. Reset hard -> este borra todo lo que ha sucedido después del commit al que se dirige y convierte al commit al que esta viajando en el nuevo master.

Git reset –soft “clave del commit”

Git reset –hard “clave del commit”

#Que son las ramas

Un mismo Proyecto puede tener diferentes ramas de trabajo

Las ramas son espacios paralelos de trabajo dentro de un mismo repositorio

#Trabajo en ramas

Git branch -> para ver las ramas de mi proyecto

Git branch “nombre de la rama” -> para crear una nueva rama

Git checkout “nombre de la rama” -> para cambiar de rama

Se crea la rama con todos los commits que ya tenia en master

Git merge “nombre de la rama” -> para fusionar las ramas

Git branch “nombre de la rama” – d -> para borrar una rama

#Trabajando con Github

Git clone “dirección del repositorio” -> clonar un repositorio de github

Git remote add origin “link del repositorio” -> para agregar un repositorio local