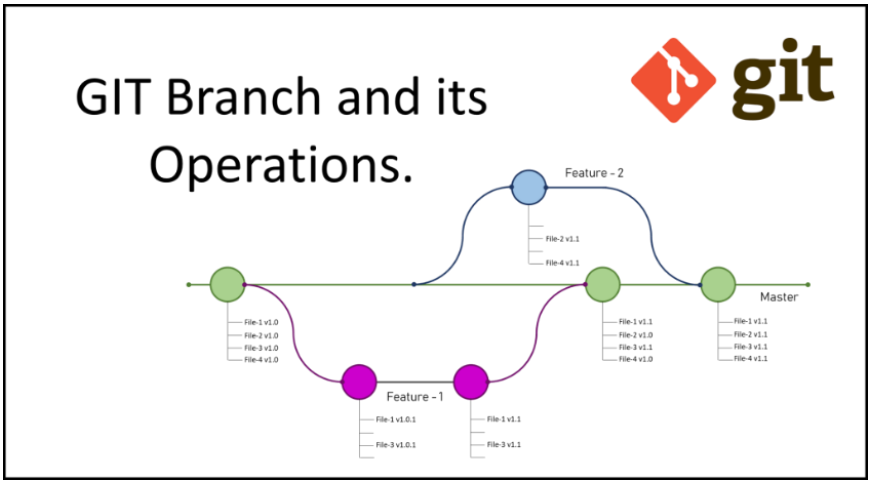
UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
ESCUELA DE ESTUDIOS DE POSTGRADO
MAESTRÍA EN INGENIERIA PARA LA INDUSTRIA
CON ESPECIALIZACIÓN EN CIENCIAS DE LA COMPUTACIÓN
CURSO: FUNDAMENTOS DE PROGRAMACIÓN Y SCRIPTING
NOMBRE: BERNY ANDREÉ CARDONA RAMOS

¿Qué es GIT?

Es un sistema de control de versiones de software, el cual se usa para trabajos en equipo donde se puede almacenar información, comúnmente es utilizada por desarrolladores para almacenar su código fuente. La información puede ser compartida y editada de manera simultánea. Una de las características de Git es que nos ofrece un historial en cada cambio que se realice, con el fin de que se puedan revertir cambios, colaborar de manera eficiente y sobre todo fusionar versiones. Esto ultimo es una de las características más importantes de GIT ya que nos permite realizar ramificaciones, con el fin de crear un nuevo ambiente, realizar cambios y luego poder fusionarlo con la rama principal.

Control de versiones con GIT

El control de versiones con GIT, como se menciono anteriormente, nos permite tener un repositorio centralizado y realizar un seguimiento de los cambios a medida que se agrega o crece el proyecto, esto con el fin de mantener el código organizado. Para ello utilizamos ramas, la cual nos permite trabajar de manera simultanea sin afectar el trabajo de los demás y al finalizar poder fusionar el código, en caso se haya editado el mismo documento y existan conflictos, GIT nos permite fusionar los cambios en una versión final, según las preferencias del equipo.



Estados de un archivo en GIT

Estado untracked: Estos son los nuevos archivos, los cuales GIT no tiene registros de su existencia.

Estado Unstaged: Al contrario de anterior estado, GIT si tiene registro de su existencia, pero fue modificado y su última versión esta únicamente guardada en el disco duro o localmente.

Estado Staged: Este nos indica que el archivo ya ha sido agregado y modificado para que vaya en la siguiente confirmación

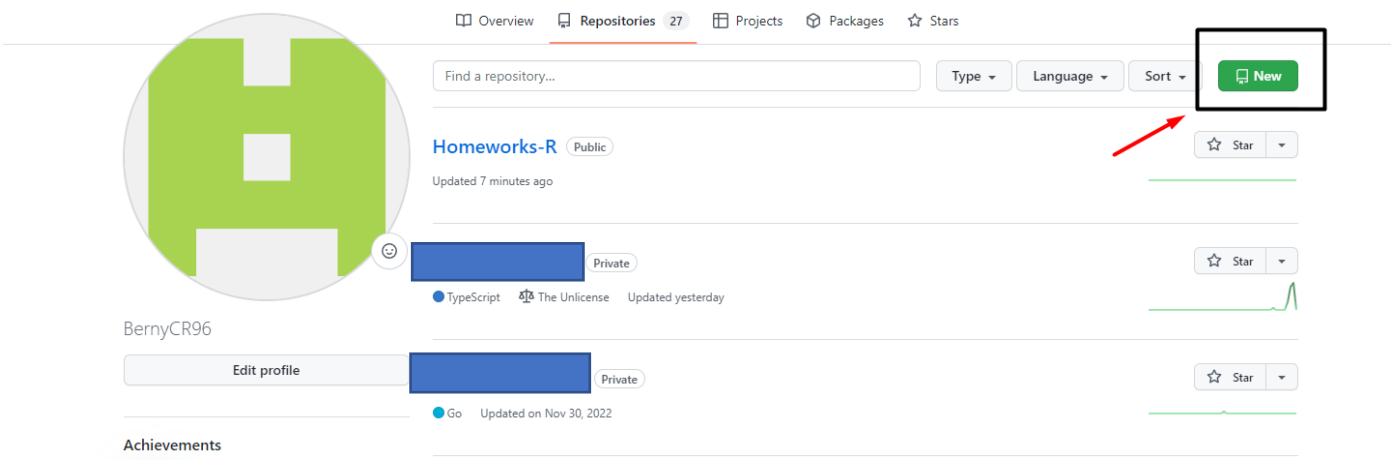
Estado Tracked: Es cuando confirmamos los cambios y estos son almacenados de manera segura en nuestra base de datos local.

Como se configura un repositorio

1. Para ello, deberán de tener una cuenta de Github e instalar GitHub CLI, la cual nos permitirá utilizar la línea de comandos en el equipo.

Para crearla una cuenta, puedes ingresar al siguiente enlace: https://github.com/signup?source=login

2. Luego, crear un nuevo Repositorio.



3. Luego crearemos un nuevo repositorio con el comando "**git init**" utilizando nuestra terminal y posicionándonos dentro de la carpeta del proyecto.
4. En caso no tener configurados nuestro nombre y correo, configurar los mismos utilizando los siguientes comandos:

   git config --global user.name "nombre de usuario

   git config –global user.email "correo del usuario"

5. Ahora agregaremos los archivos al repositorio con el siguiente comando: **git add <nombre del archivo>**

   O **git add .**

   El punto indica que agregará todos los archivos que se encuentren dentro de la carpeta actual.

6. Luego haremos la confirmación de los archivos con el siguiente comando: **git commit -m "descripción de la confirmación"**
7. Luego insertaremos nuestro repositorio local, al creado en el paso 1 con los siguientes comandos: **git remote add origin <REMOTE_URL>**
8. Por último, insertaremos los cambios en el repositorio remoto con el siguiente comando: **git push origin main**

Comandos en GIT:

Anteriormente usamos comandos de git, para crear y configurar un repositorio. A continuación, se presenta un resumen de los comandos más utilizados en Git:

git init: Inicializará un nuevo repositorio en la carpeta contenedora.

git add: Este nos indica que el archivo ya ha sido agregado y modificado para que vaya en la siguiente confirmación.

git commit: realizá la confirmación de los cambios.

git push: envía los cambios del repositorio local, al remoto

git remote: permite administrar las conexiones a repositorios remotos.

git pull: obtiene los últimos cambios del repositorio remoto y los agrega al repositorio local.

git branch: nos muestra las ramas del repositorio.

git checkout <nombre de la rama>: nos permite cambiar entre ramas.