Autor: Alexis Pérez

Introducción a Git

¿Qué es el sistema de control de versiones (VCS)?

El sistema control de versiones (VCS) es un programa o conjunto de programas que realizara un seguimiento de los cambios en una colección de archivos que se están trabajando de acuerdo a los proyectos de interés.

¿Qué es Git?

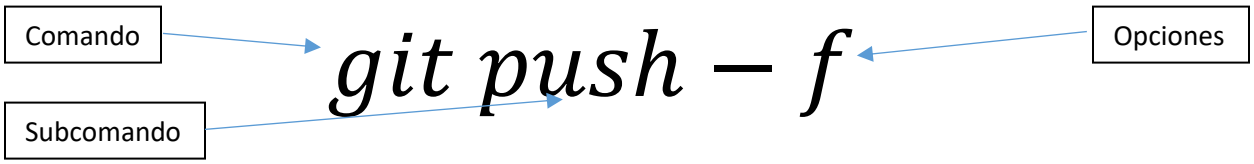
Git es un VCS de código abierto, fue creado por Linux Torvalds. También es un sistema distribuido, lo que significa que cualquier proyecto puede estar en un cliente <-> servidor

Comprendiendo Git

Para comprender su funcionamiento nos remontamos al siguiente ejemplo. Cada vez que guardas un trabajo en tu ordenador, Git crea una confirmación. Una confirmación es instantánea de todos los archivos que trabajes, para todo esto se cree debemos conocer todos sus comandos básicos para tener una interacción del controlador de versiones centralizado.

Configuración de entorno

Este programa cuando se instala en tu equipo, puedes abrir una ventana parecida a una terminal de comandos de tu sistema operativo, en donde puedes configurar inicialmente tu sesión, aunque debemos conocer la estructura de los comandos de Git, la cual su nomenclatura es la siguiente:



Una vez comprendiendo esto ahora debemos a proceder a configurar un usuario y correo asociado para que Git pueda monitorear quien y cuando se realizaron cambios en los proyectos que se están trabajando. Utilizando lo siguiente:

$$git\ config\ --global\ user.name\ "(Inserta\ tu\ nombre)"$$

$$git\ config\ --global\ user.email\ "(Tu\ correo\ electronico)"$$

¿Qué es un repositorio?

Podremos definirlo como un directorio en donde almacenamos los proyectos creados, también se considera o asocia un nombre en particular para ubicar nuestro “repo” que está alojada desde el lado de cliente (nuestra maquina) o servidor (algún sitio de internet).

Creando un repositorio

Una vez teniendo configurado nuestro usuario y correo, podemos crear un repositorio local en nuestro equipo, para eso realizaremos lo siguiente:

1. En la ventana de Git, escribimos $mkdir\ (nombre\ de\ la\ carpeta)$
2. Seguido de eso, abrimos esa carpeta con el siguiente comando: $cd\ (nombre\ de\ la\ carpeta)$
3. Vamos a inicializar nuestro proyecto con este comando: $git\ init$
4. Para verificar siempre tenemos creado o hay cambios, utilizaremos $git\ status$

Importancia del repositorio

Un repositorio permite al desarrollador(es) aplicar múltiples cambios al código de un proyecto en el cual se trabaja, también se considera que los repositorios pueden tener unas ventajas, ya que cuentan la formas de poder modificar cosas que se consideren importantes, introducir características o corregir errores.

¿Qué son las ramas en Git?

Las ramas son punteros de versión en donde se van a guardando y trabajando en el proyecto, se asocia como una rama de árbol porque todo el proyecto que se transforma como producto o servicio actúa como el árbol que se conforma por varias ramas. Siguiendo con el ejemplo de nuestro repositorio haremos lo siguiente:

1. En la terminal crearemos un archivo y usaremos el comando $touch\ (nombre\ del\ archivo.extension\ del\ archivo)$
2. Usamos lo siguiente para agregar nuestro archivo al repo $git\ add\ (nombre\ del\ archivo.extension\ del\ archivo)$
3. Finalmente realizaremos un commit (confirmación) de esta forma $git\ commit -m\ (El\ mensaje\ que\ quieras\ indicar)$

¿Qué es GitHub?

Es un repositorio basado en la nube que permite a los desarrolladores o personas almacenar y trabajar los proyectos de forma organizada, cabe decir que GitHub está basado en Git e incluye características adicionales con Git. Para saber más de eso puedes acceder a su página en línea y realizar una cuenta para poder crear tus repositorios en la plataforma.

Continuando con el ejercicio podemos realizar lo siguiente para configurar nuestro usuario de GitHub en la configuración del mismo:

1. Necesitamos crearnos una cuenta en GitHub https://github.com/ una vez teniendo la cuenta, necesitamos el nombre de usuario para usar el siguiente comando $git\ config --global\ user.username\ (tu\ nombre\ de\ usuario)$
2. Ahora vamos a Github, en el apartado de repositorios y creamos un nuevo repositorio

3. Nos dara un link en formato SSH o HTTP, copiamos ese link y nos conectamos mediante la terminal de Git con el comando
   $git\ remote\ add\ origin\ (URL\ de\ GitHub)$
4. Vamos a subir nuestros archivos al repositorio de Github con este comando
   $git\ push\ origin\ master$
5. Listo tenemos nuestro proyecto en GitHub

Configurando las ramas (branches)

Previamente mencionamos que los proyectos son como un árbol entero y que se conforman por ramas, gracias a Git, podemos definir ramas que se asocian como versiones distintas de la rama principal. Para eso haremos otra rama del proyecto

1. Abre la terminal de Git y escribe lo siguiente
   $git\ branch\ (nombre\ de\ la\ rama\ a\ crear)$
2. Para acceder a esa rama ejecutamos lo siguiente $git\ checkout\ (rama\ creada)$
3. Estaremos en la rama nueva y podemos seguir los mismos pasos para crear y subir un nuevo archivo al repositorio, recordando estos comandos
   $touch, git\ add, commit\ y\ push\ (rama\ creada)$
4. Y listo, tenemos nuestra rama creada.

Mas comandos a considerar

Existen más comandos a considerar, con lo explicado y aprendido podemos tener garantizado que tenemos conocimientos básicos en Git con GitHub. Sin embargo, hay una cantidad de comandos adicionales, veremos a continuación lo siguientes:

Merge

Este comando fusiona cualquier cambio que se haya hecho en la base del proyecto en una rama separada de la rama actual como si de un nuevo commit se tratara. La sintaxis del comando es la siguiente: $git\ merge\ (Nombre\ de\ la\ rama)$

Hablando en un caso, si estamos trabajando actualmente en una rama llamada *dev* y queremos fusionar los nuevos cambios que se hayan realizado en una rama llamada *new-features* usamos ese comando

Rebase

En lo particular la reorganización sí es lo que buscamos cuando tenemos diversas ramas es ideal usar el rebase. Como en el caso de merge es para fusionar, también sucede con rebase solo que hay diferencias. En rebase es utilizado para integrar cambios de una rama a otra de una manera más limpia y lineal, permite reescribir el historial de la rama aplicando un commit y este es su sintaxis: $git\ rebase\ (rama\ a\ apuntar)$

Conflicts

Normalmente pueden suceder errores o conflictos en Git, esto puede ocurrir cuando hay cambios en diferentes partes del mismo proyecto en ramas que se están fusionado o rebaseando y por lo general Git no puede saber con automáticamente como combinar estos cambios.

La solución se basa en hacerlo manualmente indicados por Git y luego marcalos como resueltos con *git add* (*archivo modificado*) antes de finalizar cada merge o rebase. Despuse se podría aplicar los comandos de merge y rebase agrando la opción a los comandos al final con –*continue*

Referencias Bibliográficas:

Carrillo, J. (07 de febrero de 2021). La Guía Definitiva para Git Merge y Git Rebase. freeCodeCamp. https://www.freecodecamp.org/espanol/news/la-guia-definitiva-para-git-merge-y-git-rebase/

Staff IONOS. (09 de noviembre de 2022). Git rebase: cómo poner los cambios al principio de una rama. IONOS. https://www.ionos.mx/digitalguide/paginas-web/desarrollo-web/git-rebase/

mijacobs, LizCasey1 & EdKaim. (04 de Octubre de 2023). ¿Qué es Git? Microsoft Learn. https://learn.microsoft.com/es-es/devops/develop/git/what-is-git