**Instituto Tecnológico de las Américas**


**Nombre:**

Roniell Pérez Bello

**Matricula**:

2021-0032

**Sección:**

#6

**Santo Domingo, 26 de noviembre del 2022**

1- **¿Qué es Git?**

Desde la página oficial de git lo definen como un sistema que sirve para llevar un control de versiones, distribuido, gratuito y de código abierto, diseñado para gestionar desde proyectos pequeños a muy grandes con rapidez y eficacia.

Personalmente, de manera simplificada podría decir que git es un software que le brinda a sus usuarios la posibilidad de llevar un registro muy completo de los cambios en que suceden en los archivos de un repositorio que se encuentra en el computador, permitiendo así un óptimo control a la hora de trabajar manejando muchos archivos.

Git se ejecuta localmente en un computador, para complementar un trabajo en equipo con una alta coordinación existen diversas plataformas que se pueden utilizar como hosts en línea, por ejemplo, como GitHub, Bitbucket, etc. GIT permite obtener los cambios realizados en un host en línea y poder fusionarlos en una maquina local, permitiendo así que un grupo de personas puedan trabajar en conjunto en un mismo archivo.

2- **¿Para que funciona el comando Git init?**

De manera resumida la funcionalidad que cumple el comando Git init es que, a partir de un directorio, este al ser ejecutado convierte dicho directorio en un nuevo repositorio de Git.

Desde un punto de vista más técnico y profundo, cuando el comando Git init es ejecutado en un directorio lo que sucede es que crea una zona llamada staging dentro de la memoria RAM donde se irán guardando los cambios que ocurran sobre los archivos del directorio, pasando a

crear el repositorio /.git/, que será donde se guardara todos los cambios.

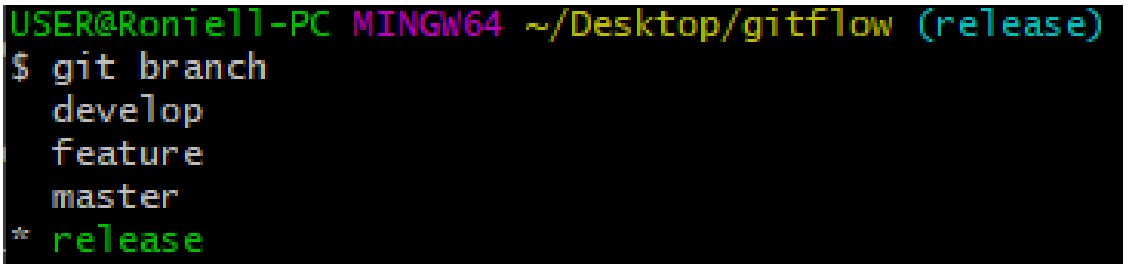
**3- ¿Qué es una rama?**

Una rama en git se puede visualizar como una división del estado de los archivos, con la finalidad de crear nuevos caminos a favor de la evolución de los archivos, en Git las ramas son parte diaria del desarrollo, son una guía instantánea para los cambios realizados.

Las ramas actúan de manera independiente en el desarrollo, pero se pueden combinar en caso de así desearse, son de bastante ayuda a la hora de generar cierta abstracción en los procesos de edición, preparación y confirmación. Todas esas nuevas confirmaciones son registradas en el historial de la rama principal, creando una bifurcación en el historial del proyecto.

**4- ¿Como saber es que rama estoy?**

Con el comando 'git branch' podemos visualizar todas las ramas que existen en nuestro repositorio y nos resalta la rama actual en la que nos encontramos, en la siguiente imagen tenemos un ejemplo:



Si solo queremos ver la rama actual sin que se listen las demás, podemos ejecutar 'git branch --show-current', en la siguiente imagen podemos ver un ejemplo:


**5- ¿Quién creó git?**

El creador de git se llama Linus Torvalds, la razón fue debido a que en el 2005, la relación entre la comunidad que desarrolló el núcleo de Linux y la empresa comercial que desarrolló BitKeeper se rompió, por lo que BitKeeper ya no se podía usar más de manera gratuita, esto fue lo que llevó a a Linus Torvalds a dirigir la comunidad de desarrolladores de Linux para desarrollar su propia herramienta, por lo que basándose en BitKeeper, se creó GIT.

**6- ¿Cuáles son los comandos más esenciales de Git?**

Personalmente opino que los comandos más esenciales de git son los siguientes:

- **Git init:** Permite crear e inicializar un repositorio git.
- **Git status:** Nos permite ver toda la información necesaria sobre la rama actual.
- **Git add:** Permite incluir los cambios de uno o varios archivos en nuestro próximo commit.

**Git add <nombre del archivo>**

- **Git commit:** Permite guardar los cambios estableciendo un punto de control en el proceso de desarrollo, es posible volver a dicho punto más tarde si es necesario, se necesita escribir un mensaje corto para explicar lo que ha sido creado o modificado en el commit.

**git commit -m "Mi mensaje"**

- **Git clone:** Permite descargar el código fuente existente desde un repositorio remoto haciendo una copia idéntica de la última versión de un proyecto en un repositorio.

**git clone <url del repositorio>**

- **Git branch:** Permite para crear, enumerar y eliminar ramas en un repositorio.

**git branch < nombre de la rama >** *(crea rama)*

**git branch -d <nombre de la rama>** *(elimina rama)*

**git branch** *(lista las ramas)*

- **Git push:** Este comando es el que nos permite subir nuestros cambios a un repositorio remoto al cual se esté conectado.

**git push <repositorio> <rama>**

- **Git pull:** Este comando nos permite obtener actualizaciones del repositorio remoto.

**git pull <repositorio>**

- **Git merge:** Este comando sirve para integrar todos los cambios y características de una rama a otra.

**git merge <nombre de la rama>**

- **Git revert:** Permite revertir los cambios efectuados en el historial de commits dentro de un repositorio.

  **git revert <número del commit>**

- **Git checkout:** Este comando es el que nos permite cambiar de una rama a otra.

  **git checkout <nombre de la rama>**

**7- ¿Qué es git Flow?**

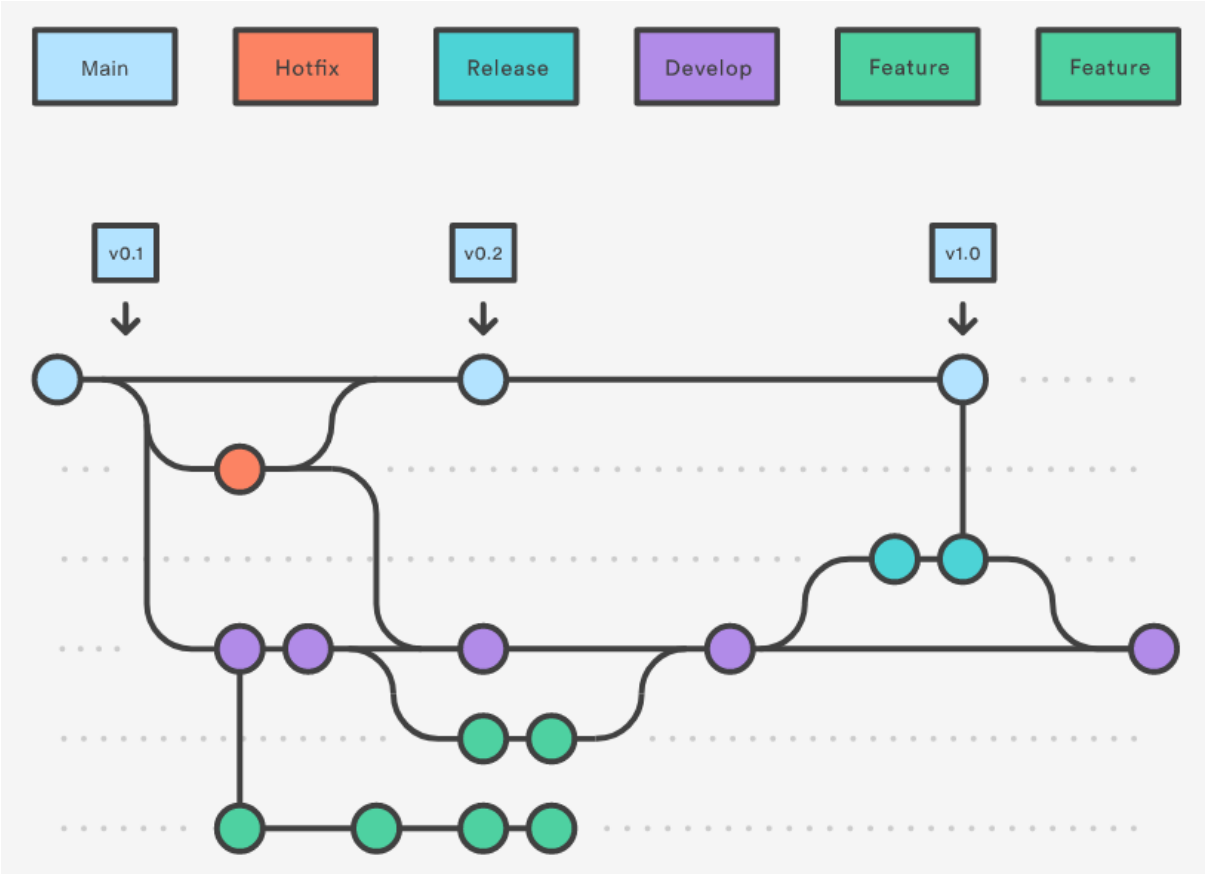
"Gitflow es un modelo alternativo de creación de ramas en Git en el que se utilizan ramas de función y varias ramas principales. Fue Vincent Driessen en nvie quien lo publicó por primera vez y quien lo popularizó" (Atlassian Bitbucket).

En gitflow se trabaja con ramas de función y retrasan su fusión con la rama principal del tronco hasta que todas las funcionalidades esperadas están completadas.

En lugar de tabajar una única rama, en gitflow se utiliza diversas ramas para registrar el historial del proyecto. La rama principal, la cual es comúnmente llamada **main** o **master**, en esta se almacena el historial de publicación oficial; la rama **develop** que sirve como rama de integración para las funciones; la rama **feature** que sirve para crear funcionalidades específicas, esta rama utiliza la rama develop como rama primaria, cuando una función está terminada, se vuelve a fusionar en develop; la rama **release** es la que se usa para preparar una publicación a la rama principal una vez que la rama develop cumple con las tareas propuestas; la rama **hotfix** que se usa para depurar el código que venga de producción después de que se haya detectado un fallo

crítico en producción que deba resolverse, al que se le va a hacer una release puntual para corregirlo.

La siguiente imagen representa el flujo que existe en gitflow:



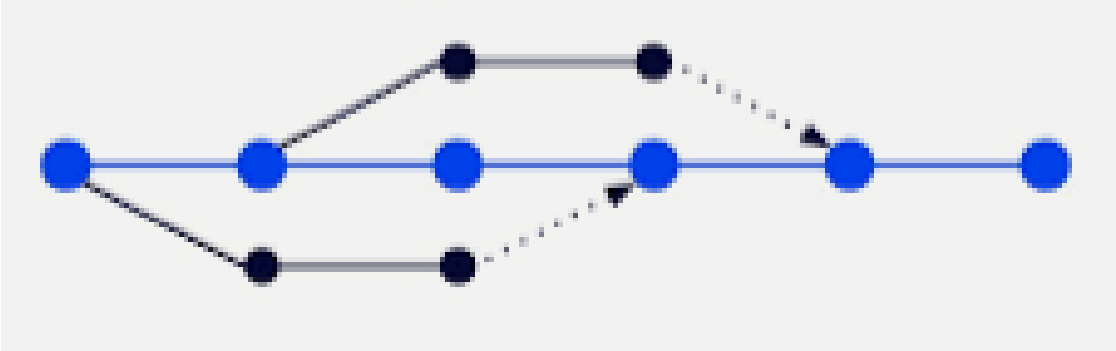

**8- ¿Que es trunk based development?**

Trunk based development es una estrategia de Git en la que se trabaja en una sola rama, por el eso el nombre de 'basado en tronco', en esta estrategia todo el equipo colabora integrando directamente los cambios en la única rama existente.

Con esta estrategia se lleva una integración continua, se disminuye la fricción que se puede dar en la integración cambios, agilizando las fases de fusión e integración.

También hay que destacar que en esta estrategia se crean ramas de corta duración con unos pocos commits en comparación con la estrategia de git flow donde se lleva una ramificación de características de larga duración.

La siguiente imagen representa como se trabaja en esta estrategia:



## Bibliografía


**Git. Página web, recuperado de: https://git-scm.com/**

**Lorenzo, A. (2020). ¿Qué sucede cuando hacemos git init? Artículo, recuperado de: https://platzi.com/tutoriales/1557-git-github/7414-que-sucede-cuando-hacemos-git-init/**

**KeepCoding Team. (2022). ¿Qué son las ramas en git y para qué sirven? Artículo, recuperado: https://keepcoding.io/blog/que-son-las-ramas-en-git/**

**Git. A Short History of Git. Artículo, recuperado de: https://git-scm.com/book/en/v2/Getting-Started-A-Short-History-of-Git**

**Atlassian Bitbucket. Flujo de trabajo de git Flow. Página web, recuperada de: https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow#:~:text=%C2%BFQu%C3%A9%20es%20Gitflow%3F,vez%20y%20quien%20lo%20populariz%C3%B3.**