



*Las Americas Institute of Technology*

**Asignatura: Programación #3**

**Nombre:** wesly José Estévez bautista 2022-1065

**Cuatrimestre-** C-1

**Kelyn Tejada Belliard - Docente ITLA**

**Introducción**

Git es un sistema de control de versiones distribuido ampliamente utilizado en el desarrollo de software. Permite a los desarrolladores trabajar de manera colaborativa, rastrear cambios y gestionar versiones de código de manera eficiente. En este documento se responderán diversas preguntas clave sobre Git para proporcionar una comprensión sólida de sus fundamentos y buenas prácticas.

**1- ¿Qué es Git?**

Git es un sistema de control de versiones distribuido que permite a los desarrolladores gestionar y rastrear cambios en archivos de código fuente de manera eficiente. Fue diseñado para optimizar la colaboración en proyectos de software, permitiendo trabajar en paralelo sin conflictos.

**2-¿Para qué sirve el comando git init?**

El comando `git init` se utiliza para inicializar un nuevo repositorio de Git en un directorio. Al ejecutarlo, se crea una carpeta oculta llamada `.git`, que contiene toda la información necesaria para gestionar el historial de versiones del proyecto.

### **3-¿Qué es una rama en Git?**

Una rama en Git es una versión paralela del código principal que permite desarrollar nuevas características o corregir errores sin afectar la versión estable. Las ramas facilitan el trabajo colaborativo y la integración de cambios de manera controlada.

### **4-¿Cómo saber en cuál rama estoy trabajando?**

Para saber la rama que estamos usando podemos usar los siguientes comandos.

`git Branch`

este muestra las ramas que tenemos disponibles y marca con un asterisco\* la rama en donde nos encontramos.

Otro comando que podemos usar es.

`git status`

que muestra la rama activa en el momento.

### **5-¿Quién creó Git?**

Git fue creado por Linus Torvalds en 2005, el mismo creador del sistema operativo Linux. Lo diseñó para gestionar el

desarrollo del kernel de Linux de manera eficiente y distribuida.

## 6- ¿Cuáles son los comandos esenciales de Git?

Estos son comandos esenciales de Git incluyen:

`git init` - Inicializa un repositorio.

`git clone <URL>` - Clona un repositorio existente.

`git add <archivo>` - Agrega archivos al área de preparación.

`git commit -m "Mensaje"` - Guarda los cambios en el historial.

`git status` - Muestra el estado del repositorio.

`git branch` - Lista las ramas disponibles.

`git checkout <rama>` - Cambia de rama.

`git merge <rama>` - Fusiona una rama con la actual.

`git pull` - Descarga cambios desde un repositorio remoto.

`git push` - Sube cambios a un repositorio remoto.

## 7- ¿Qué es Git Flow?

Git Flow es una metodología de desarrollo basada en el uso de múltiples ramas para organizar el desarrollo de software. Se basa en las siguientes ramas principales:

`main`: Contiene la versión estable del código.

`develop`: Rama principal para el desarrollo.

`feature`: Para el desarrollo de nuevas funcionalidades.

`release`: Para preparar nuevas versiones.

**hotfix:** Para correcciones urgentes en producción.

## **8- ¿Qué es el desarrollo basado en trunk (Trunk Based Development)?**

El desarrollo basado en trunk (Trunk Based Development) es una estrategia en la que los desarrolladores trabajan directamente en una única rama principal (trunk o main), en lugar de utilizar múltiples ramas a largo plazo. Se centra en realizar integraciones frecuentes y minimizar la fragmentación del código.

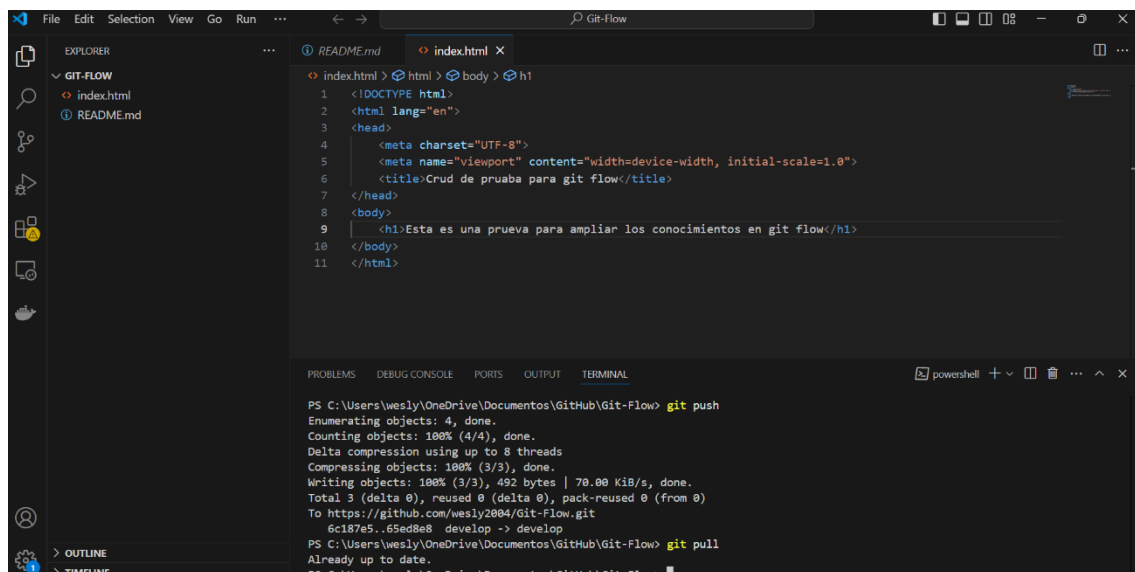
## Bibliografía

Chacon, S., & Straub, B. (2014). Pro Git. Apress.

Official Git Documentation: <https://git-scm.com/doc>

Atlassian Git Tutorials:

<https://www.atlassian.com/git/tutorials>



```
PS C:\Users\wesly\OneDrive\Documents\GitHub\Git-Flow> git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 492 bytes | 70.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/wesly2804/Git-Flow.git
 6c187e5..65ed8e8 develop -> develop
PS C:\Users\wesly\OneDrive\Documents\GitHub\Git-Flow> git pull
Already up to date.
PS C:\Users\wesly\OneDrive\Documents\GitHub\Git-Flow>
```

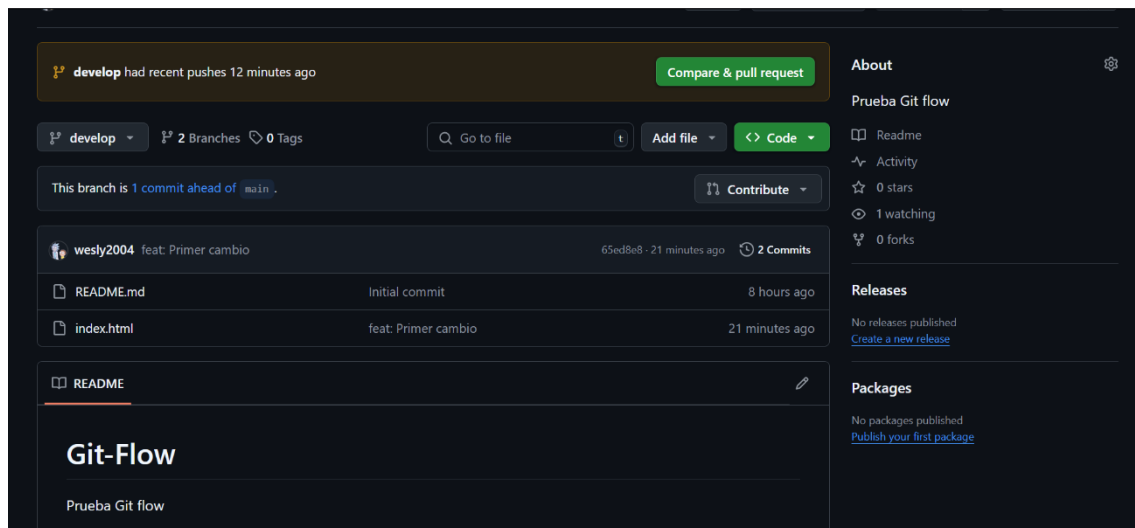
En esta imagen representa el método git Flow prácticamente he creado un repositorio en github luego le he agregado en readme file y cree una rama llamada develop, luego he iniciado el método git Flow usando `git flow init` y ya luego continúe con la creación del crud para así tener el cambio con control de versiones, luego cree las features y les di un nombre `git flow feature start nuevos-archivos`

`git add -A` para agregar los cambios al repositorio

agrego un comentario `git commit -m "feat: Primer cambio"`

luego lo finalizo `git flow feature finish nuevos-archivos`

y ya solo queda subirlo `git push`



Link del repositorio: <https://github.com/wesly2004/Git-Flow>