Proyecto de GitFlow

Jesus Manuel Terán Vergara

Ficha: 2928818

SENA | CTMA

22 Octubre del 2025

1. ¿Qué es Gitflow?

Gitflow es una estrategia de ramificación para Git que organiza el desarrollo de software mediante el uso de ramas específicas (como main, develop, feature, release y hotfix). Permite que varios desarrolladores trabajen al mismo tiempo de forma ordenada y controlada, evitando conflictos y errores en el código.

2. ¿Qué ramas principales se usan en Gitflow? Describa cada una.

Las ramas principales que podemos encontrar en un proyecto de Gitflow son:

Principales

-Master

Es la rama principal del proyecto, donde se guarda el código completamente estable y funcional.
Todo lo que está en main debe ser una versión que ya pasó por pruebas y está lista para los usuarios.
En otras palabras, main es como la “línea oficial del producto”: solo contiene versiones publicadas o preparadas para producción.

Mantener el historial de versiones finales del software.
Cada vez que se lanza una nueva versión, se marca en main con una etiqueta (tag), por ejemplo v1.0, v1.1, etc.
Así se puede volver fácilmente a cualquier versión anterior si es necesario.

- Develop

Segundarias

- Feature
- Release
- Hotfix

3. ¿Cuál es el propósito de cada tipo de rama en Gitflow?

Tipo de Rama	Propósito principal
Main (o master)	Guardar el código **estable y listo para producción**. Representa el historial oficial de versiones publicadas.
Develop	Servir como la **base de desarrollo** donde se integran todas las nuevas funciones antes de lanzarlas.
Feature	Permitir que los desarrolladores trabajen **de forma aislada** en nuevas funcionalidades sin afectar el resto del proyecto.
Release	Facilitar la **preparación de una nueva versión**, realizando pruebas finales y ajustes antes de publicarla.
Hotfix	Permitir **correcciones rápidas** de errores graves detectados en producción sin interrumpir el desarrollo en curso.

4. ¿Qué ventajas ofrece Gitflow frente al uso libre de ramas?

Organización clara del trabajo
Cada tipo de tarea (nueva función, corrección, versión) tiene su propia rama y reglas.
➜ Evita confusiones sobre dónde trabajar o fusionar.

Trabajo en equipo sin conflictos
Todos pueden trabajar al mismo tiempo sin pisarse el código, gracias a las ramas *feature*.

Control de versiones estable
Main siempre contiene una versión lista para producción, y *develop* las versiones en desarrollo.
➜Nunca se mezcla código inestable con el que ya funciona.

Facilita lanzamientos y mantenimiento
Las ramas *release* y *hotfix* permiten preparar versiones o corregir errores urgentes sin afectar el desarrollo normal.

Historial más limpio y entendible
El flujo de ramas muestra claramente la evolución del proyecto: qué se desarrolló, cuándo y por qué.

5. ¿Cuáles son los pasos para implementar Gitflow en un proyecto real?

Pasos para implementar Gitflow en un proyecto real

Crear o clonar un repositorio

Primero necesitas un proyecto en GitHub o GitLab (puede ser nuevo o uno existente).

git clone <url-del-repositorio>
cd <nombre-del-proyecto>

Inicializar Gitflow en el repositorio

Aquí se configura la estructura de ramas que usarás:

git flow init

Durante la inicialización, Git te preguntará los nombres de las ramas principales.
Generalmente dejas los valores por defecto:

- Rama principal: main
- Rama de desarrollo: develop

Crear la rama *develop*

Si el repositorio solo tiene main, Gitflow creará develop automáticamente (o puedes hacerlo tú):

git branch develop
git push -u origin develop

Iniciar una rama *feature* (para una nueva función)

Cuando comiences una nueva funcionalidad, creas una rama de feature desde develop:

git flow feature start nombre-feature

Ejemplo:

git flow feature start login

Ahí trabajas, haces commits, y cuando termines:

git flow feature finish login

Esto hace merge automático con develop.

Crear una rama release

Cuando el proyecto ya está listo para lanzar una nueva versión:

git flow release start 1.0

Aquí se hacen pruebas y pequeños ajustes.
Cuando todo está correcto:

git flow release finish 1.0

Esto fusiona la release con main y develop, y marca el lanzamiento con un tag (por ejemplo, v1.0).

Crear una rama hotfix (si hay errores en producción)

Si surge un error urgente en main:

git flow hotfix start fix-login

Después de arreglarlo:

git flow hotfix finish fix-login

Esto fusiona los cambios en main y develop para mantener todo sincronizado.

6. ¿Qué es un pull request? ¿Qué es un merge?

Pull Request (PR): solicitud para revisar y fusionar los cambios de una rama (por ejemplo feature/login) con otra (como develop).

Merge: acción de combinar los cambios entre ramas. Puede hacerse al aprobar un PR o con el comando:

git merge nombre-rama

Parte 2:

Ejercicio Práctico Paso 1:

Crear un Proyecto

• Elijan un nombre y propósito simple para un proyecto (por ejemplo: "Lista de tareas html", "Calculadora JavaScript", "Blog básico HTML").

- Nombre elegido para el repositorio: proyecto-gitflow
- Proyecto simple a realizar una lista de tarea sencilla

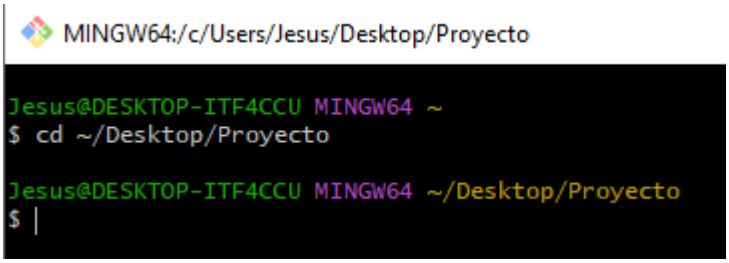
• Crear un repositorio desde GitHub o GitLab.

Paso 2: Configurar Gitflow

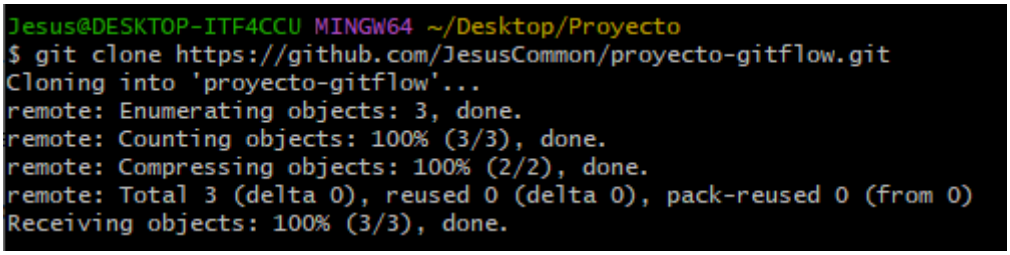
• Configurar el repositorio local con Gitflow (git flow init).

- Empezamos con nuestro segundo paso clonando nuestro repositorio de github, abrimos Git Bash y ponemos la carpeta en donde vamos a trabajar el proyecto

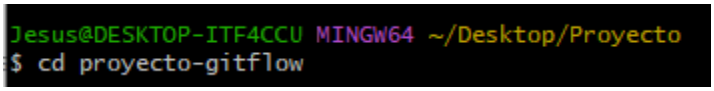
1. Creamos una carpeta en el escritorio con cualquier nombre e ingresamos a ella



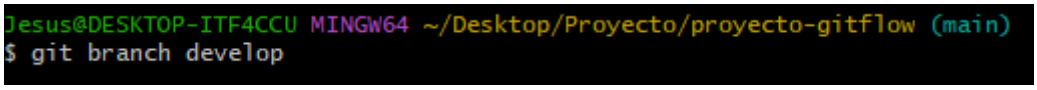
estando dentro de la carpeta nuestro siguiente paso es clonar nuestro repositorio



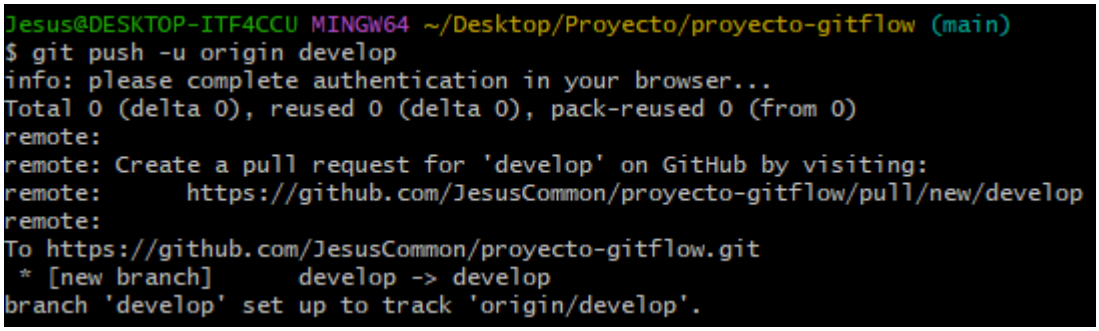
para clonar el repositorio ponemos el comando git clone y seguido va la url de nuestro repositorio creado con github



Nuestro siguiente paso es ingresar o acceder al repositorio



Creamos nuestra rama develop



Subimos la rama a nuestro repositorio

• Asegurarse de tener las ramas principales: main (o master) y develop

```
Jesus@DESKTOP-ITF4CCU MINGW64 ~/Desktop/Proyecto/proyecto-gitflow (main)
$ git branch -a
  develop
* main
  remotes/origin/HEAD -> origin/main
  remotes/origin/develop
  remotes/origin/main
```

De esta forma con el comando git branch -a no aseguramos que este las ramas principales, que se nos indica que son: main y develop

despues de esto podemos cerrar la terminal de bash y abrir nuestro proyecto en vs code

Paso 3: Flujos de Trabajo Realicen los siguientes flujos en el repositorio colaborativo:

• Crear una rama de feature y desarrollar una funcionalidad básica.

Aqui empezamos creando los archivos dentro de la carpeta de nuestro repositorio que tendrá nuestro proyecto

```
PS C:\Users\Jesus\Desktop\Proyecto> cd proyecto-gitflow
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git checkout develop
Already on 'develop'
Your branch is up to date with 'origin/develop'.
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git checkout -b feature/estructura-base
Switched to a new branch 'feature/estructura-base'
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow>
```

Accedemos al repositorio, y cambiamos el al entorno de trabajo de develop y creamos un nuevo archivo en nuestra rama feature

```
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git add .
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git commit -m "Feature: agregar funcionalidad básica de lista de tareas"
[feature/estructura-base e072720] Feature: agregar funcionalidad básica de lista de tareas
 3 files changed, 40 insertions(+)
 create mode 100644 index.html
 create mode 100644 script.js
 create mode 100644 styles.css
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git push origin feature/estructura-base
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 944 bytes | 314.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature/estructura-base' on GitHub by visiting:
remote:      https://github.com/JesusCommon/proyecto-gitflow/pull/new/feature/estructura-base
remote:
To https://github.com/JesusCommon/proyecto-gitflow.git
 * [new branch]      feature/estructura-base -> feature/estructura-base
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow>
```

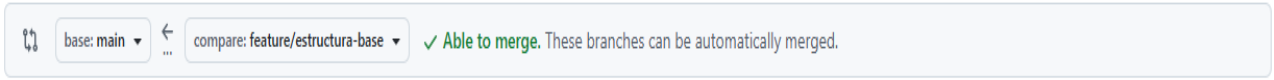
Nuestro siguiente paso es agregar todos los archivos modificados, para que sean listo para confirmar sus cambios, hacemos un commit que guarde los cambios y subimos la rama a nuestro repositorio

git add . : actualizar o agregar
git commit -m "Feature: agregar funcionalidad básica de lista de tareas" : guardar los cambios
git push origin feature/estructura-base : subir todo al repositorio

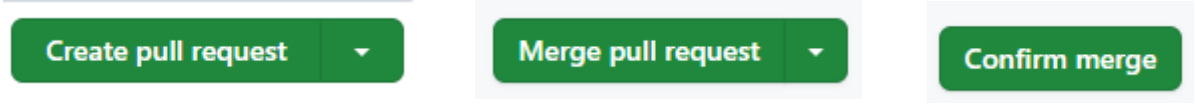
• Hacer pull requests o merge requests desde feature → develop.

Para hacer los pull nos dirigimos al repositorio y seguimos una serie de pasos

Primero le damos al boton que dice compare & pull request



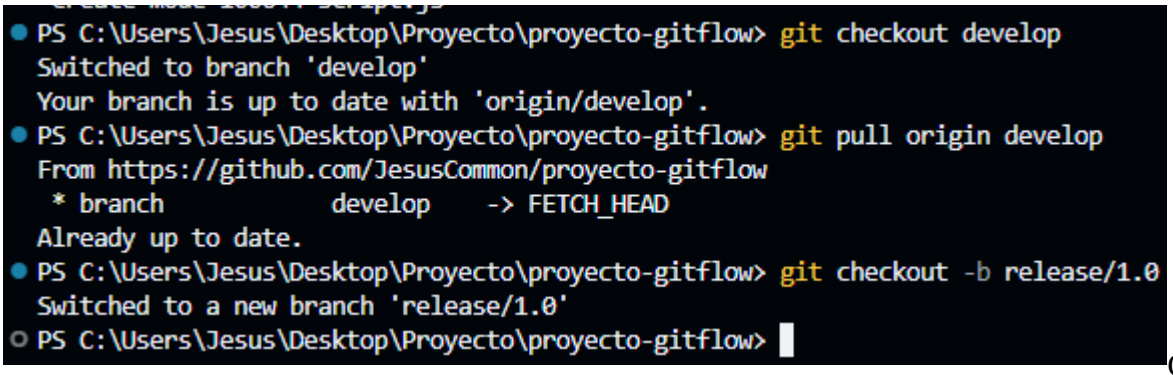
Despues verificamos que este la comparacion correcta



Despues creamos el pull con el marge y confirmamos este ultimo

Y por ultimo borramos el branch

• Simular una release creando una rama release/x.x desde develop, hacer ajustes y fusionar en main y develop.



Cambiamos nuevamente a la rama develop y actualizamos los cambios, por ultimo creamos nuestra rama dentro develop

```
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git add index.html
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git commit -m "Release 1.0: versión estable con estructura base"
[release/1.0 000fc5e] Release 1.0: versión estable con estructura base
 1 file changed, 19 insertions(+)
 create mode 100644 index.html
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow>
```

hacemos correcion de uno de nuestros archivos y guardamos los cambios

```
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git pull origin main
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (1/1), 933 bytes | 155.00 KiB/s, done.
From https://github.com/JesusCommon/proyecto-gitflow
 * branch            main       -> FETCH_HEAD
   9f269e7..408fcb9  main       -> origin/main
Updating 9f269e7..408fcb9
Fast-forward
 index.html | 17 +++++++++++++++++
 script.js  | 11 +++++++++++
 styles.css | 12 ++++++++++++
 3 files changed, 40 insertions(+)
 create mode 100644 index.html
 create mode 100644 script.js
 create mode 100644 styles.css
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git merge release/1.0
Auto-merging index.html
CONFLICT (add/add): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow>
```

Pasamos a la rama de main y actualizamos para que los cambios se apliquen correctamente por ultimo publicamos la version 1.0 de nuestro proyecto… este nos dejara un conflicto que se soluciona modificando el archivo el cual editamos anteriormente

```
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git add index.html
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git commit -m "Merge: resolver conflicto al fusionar release/1.0 en main"
[main ce33c4d] Merge: resolver conflicto al fusionar release/1.0 en main
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git push origin main
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 4 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 697 bytes | 348.00 KiB/s, done.
Total 6 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 2 local objects.
To https://github.com/JesusCommon/proyecto-gitflow.git
   408fcb9..ce33c4d  main -> main
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow>
```

Despues de hacer la solucion de conflicto agregamos el archivo, guardamos los cambios y por ultimo lo subimos a nuestra rama main

• Crear una rama hotfix desde main, realizar una corrección y fusionar en main y develop.

```
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git checkout main
Already on 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git pull origin main
From https://github.com/JesusCommon/proyecto-gitflow
 * branch            main       -> FETCH_HEAD
Already up to date.
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git checkout -b hotfix/correccion-texto
Switched to a new branch 'hotfix/correccion-texto'
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow>
```

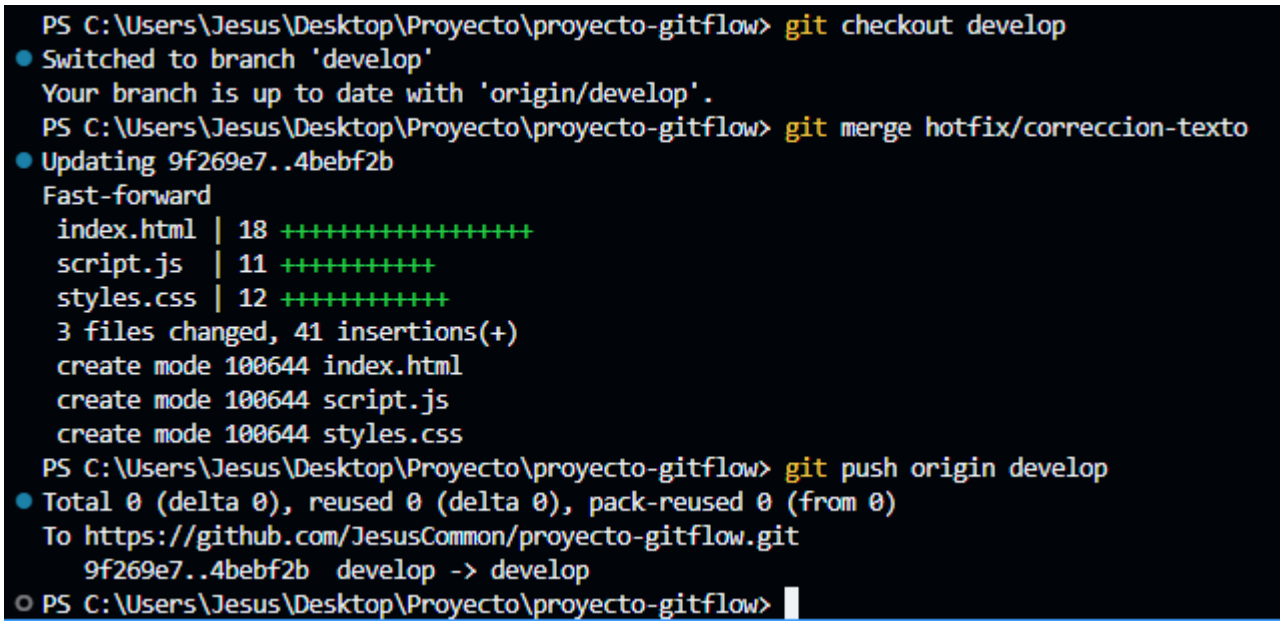
Cambiamos a nuestra rama main, actualizamos para que se apliquen los cambios y por ultimo creamos nuestra rama de reparaciones

```
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git add index.html
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git commit -m "Hotfix: corrección de texto en versión 1.0.1"
[hotfix/correccion-texto 4bebf2b] Hotfix: corrección de texto en versión 1.0.1
 1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git push origin hotfix/correccion-texto
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 343 bytes | 343.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
remote:
remote: Create a pull request for 'hotfix/correccion-texto' on GitHub by visiting:
remote:      https://github.com/JesusCommon/proyecto-gitflow/pull/new/hotfix/correccion-texto
remote:
To https://github.com/JesusCommon/proyecto-gitflow.git
 * [new branch]      hotfix/correccion-texto -> hotfix/correccion-texto
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow>
```

Hacemos cambio en el archivo index… lo agregamos para confirmar los cambios, guardamos los cambiamos y subimos nuestra rama al repositorio lista para ser integrada

```
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git merge hotfix/correccion-texto
Updating ce33c4d..4bebf2b
Fast-forward
 index.html | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow> git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/JesusCommon/proyecto-gitflow.git
   ce33c4d..4bebf2b  main -> main
PS C:\Users\Jesus\Desktop\Proyecto\proyecto-gitflow>
```

Cambiamos al main nuevamente y fusionamos los cambios hechos con hotfix en el main y subimos los cambios al repositorio



Por utlimo cambiamos nuestra rama develop, y fusionamos la correcion que se hizo aquí para que no se pierda si se actualiza cualquier otra cosa y por ultimo subimos la rama al repositorio

Referencias:

Gitflow: https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow

Gitflow: https://openwebinars.net/blog/git-flow-tipos-de-ramas/