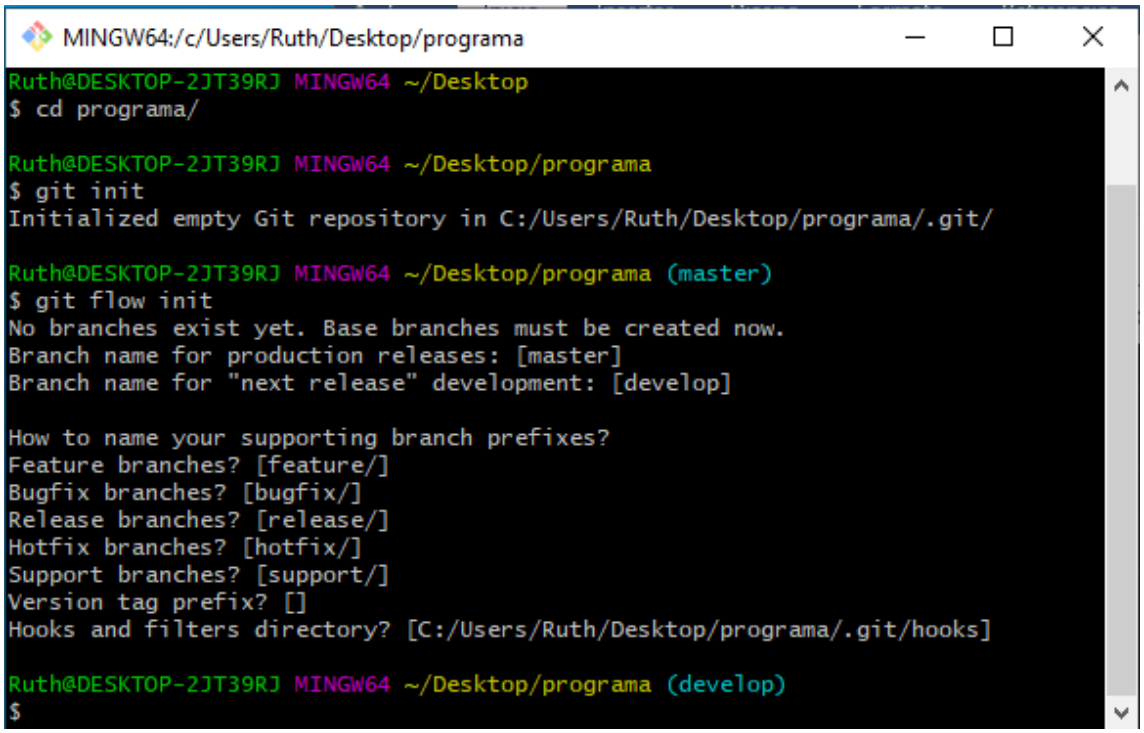
SEMANA 04: GIT FLOW

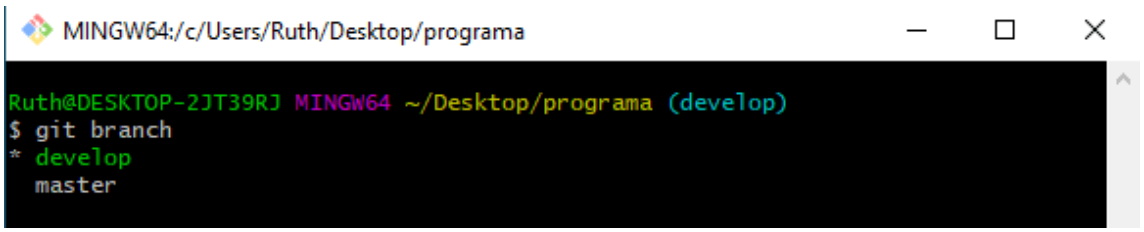
1. Inicializando un proyecto

En nuestra consola bash se crea una carpeta y se escribe los siguientes comandos git:

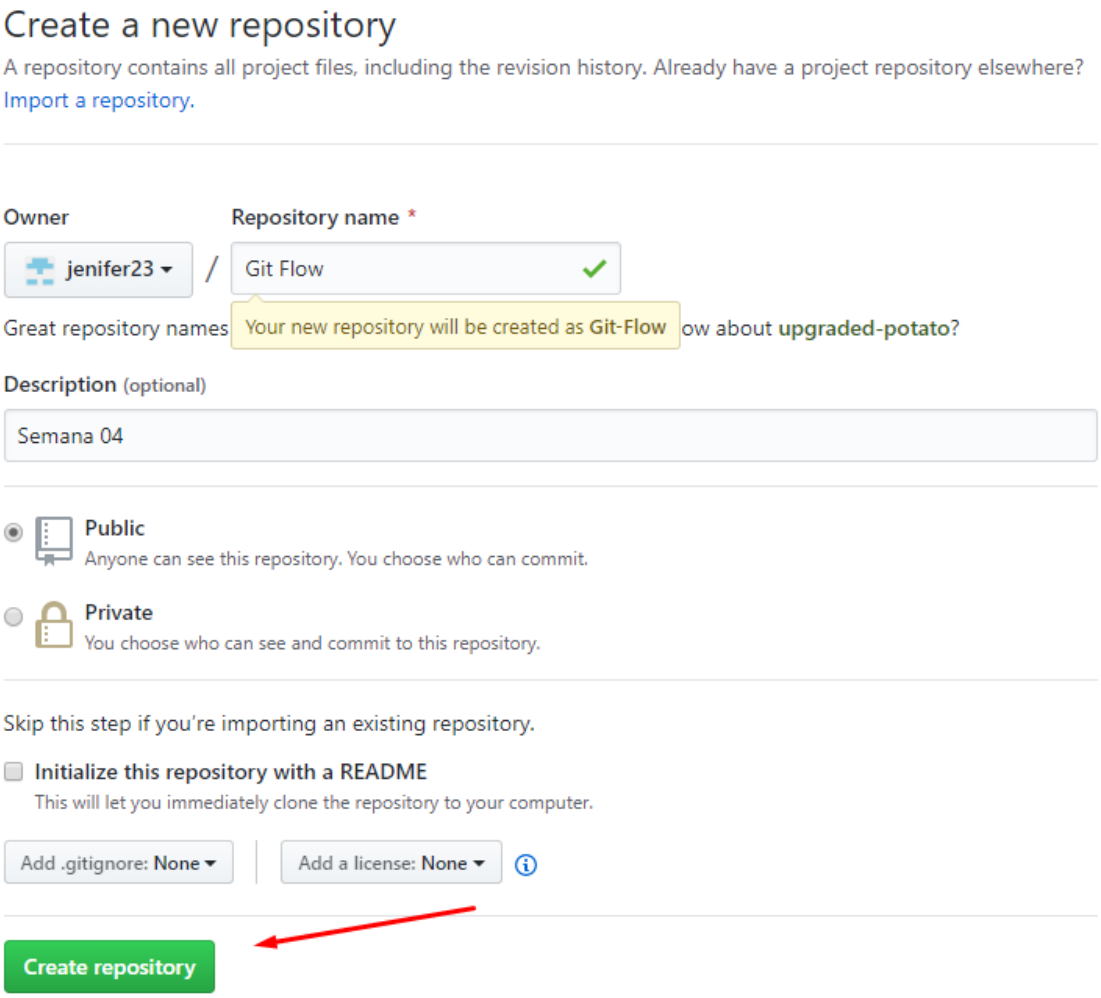
```
$git init
$git flow init
$git branch
```

Git flow crea todas las *branches* necesarias.





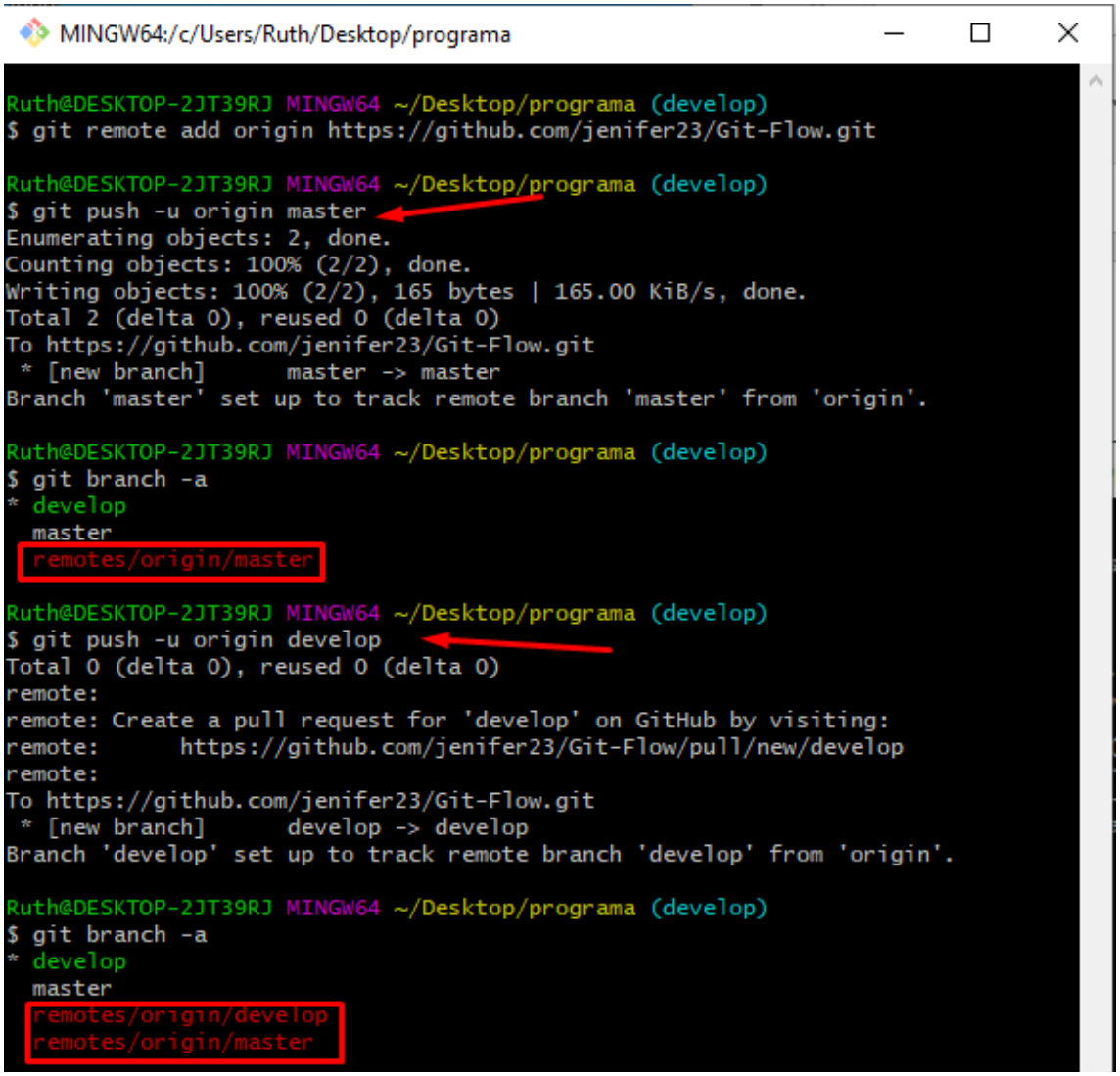
Luego se crea el repositorio en GitHub



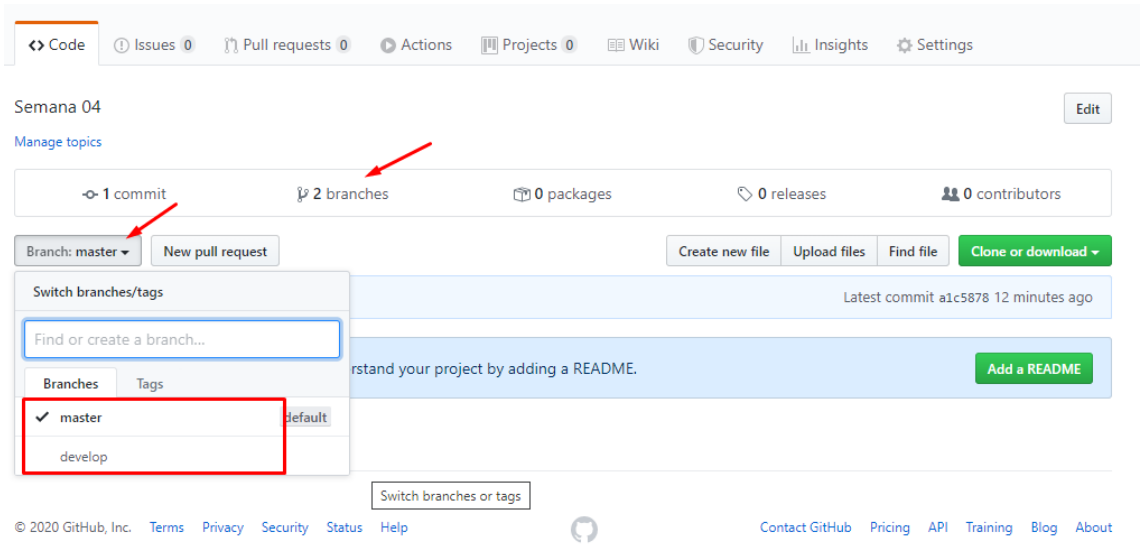
En nuestra consola bash se escribe los comandos:

```
$git remote add origin direccion
$git push –u origin master
$git branch -a
$git push –u origin develop
$git branch -a
```

Se sube nuestro proyecto y sus nuevas ramas (***master y develop***) a un servidor remoto creado en Github



En nuestro repositorio remoto se mostraría así:



2. **Creando una rama feature**

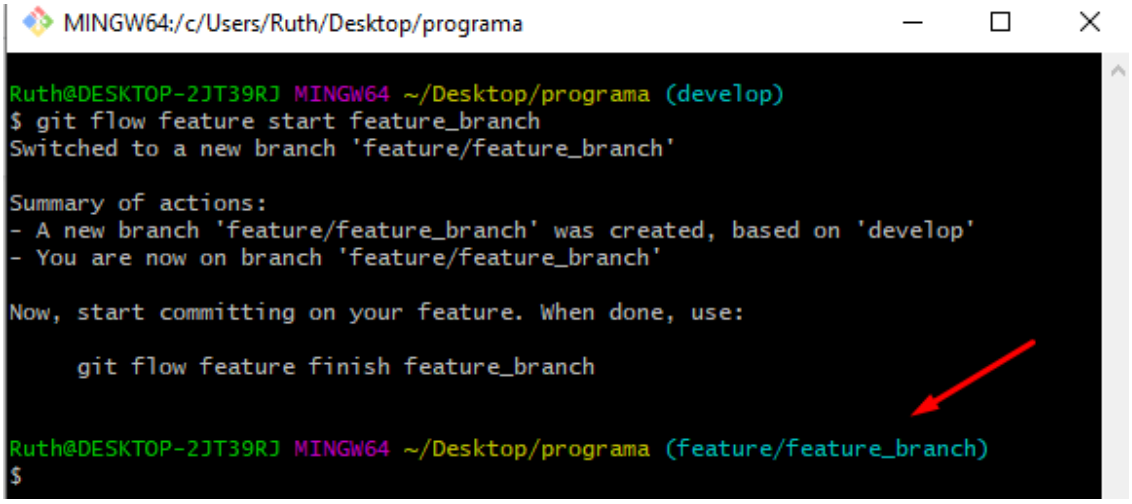
Con el uso de los comandos Git Flow en nuestra consola bash se escribe:

```
$git flow feature start feature_branch
```

Sin git flow el flujo sería el siguiente:

```
$git checkout develop
$git checkout –b feature_branch
```

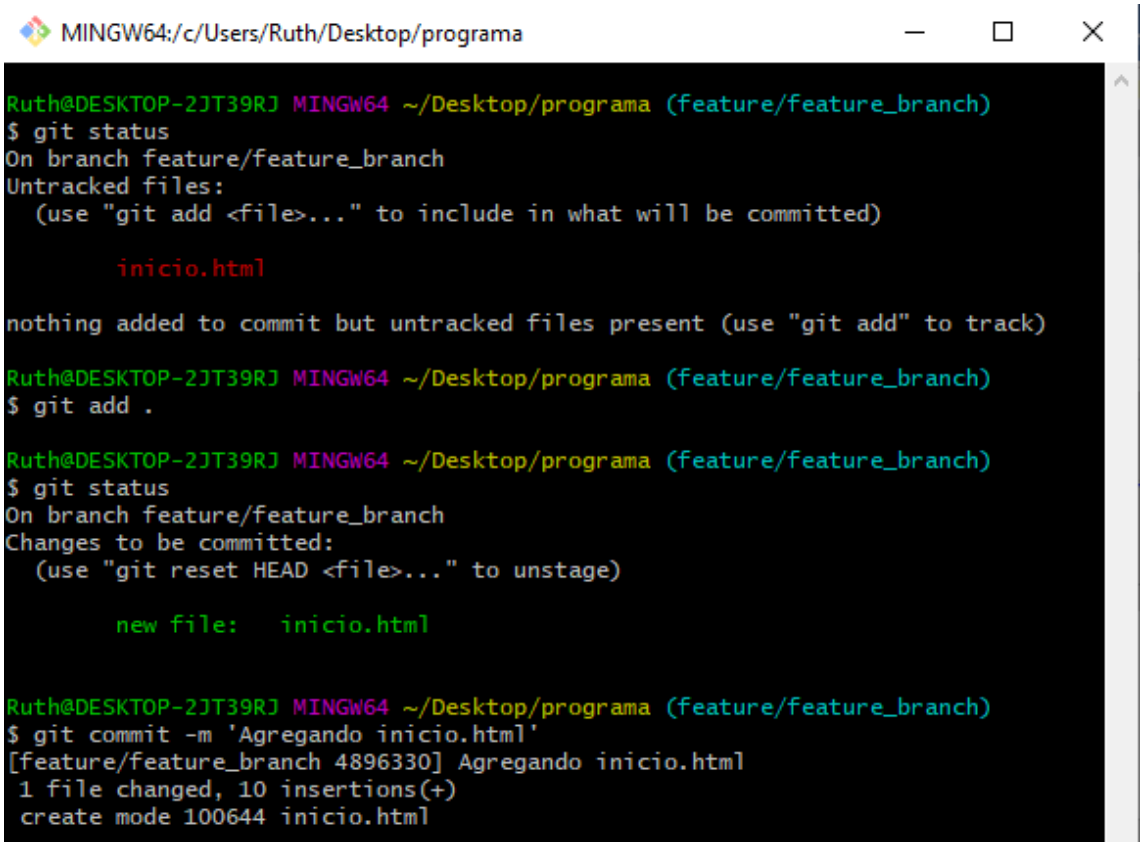
Git flow automáticamente crea la nueva rama e hizo un checkout a ésta.



En nuestro repositorio creamos un archivo ***inicio.html***

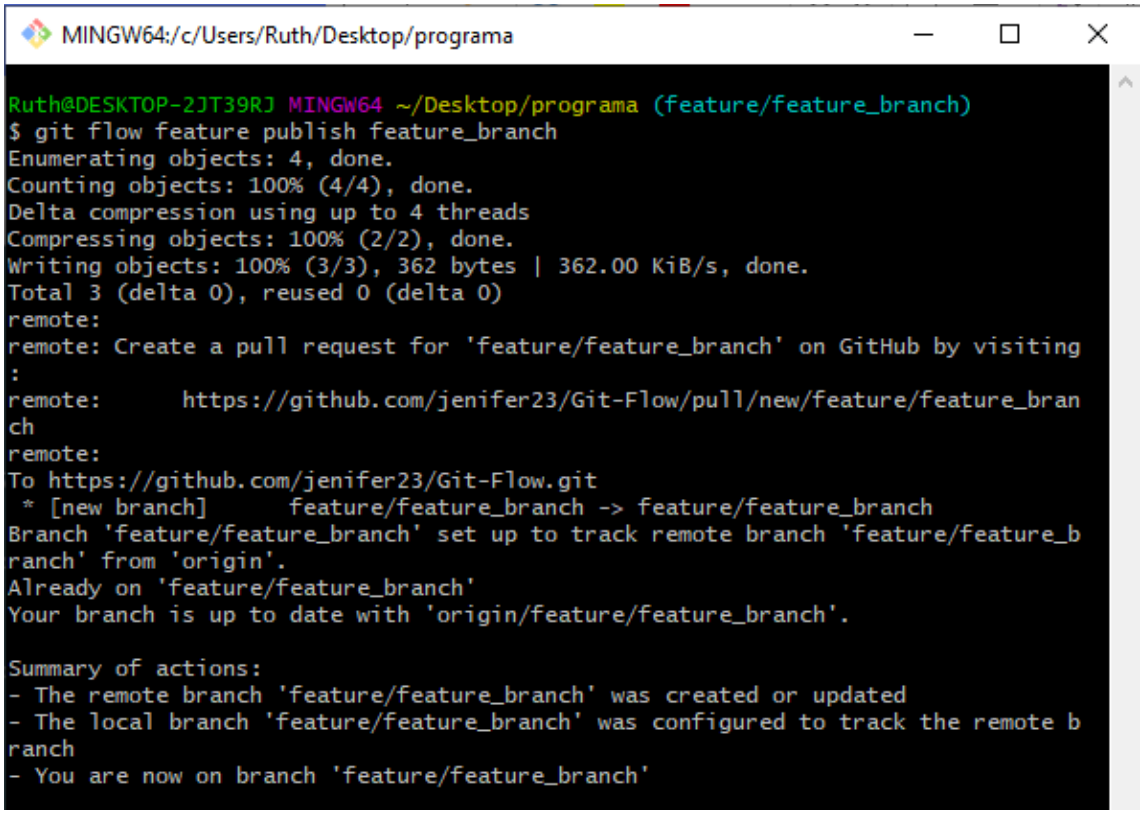
Guardamos los cambios realizados y hacemos un commit.

```
$git status
$git add .
$git status
$git commit –m ‘Agregando inicio.html’
```



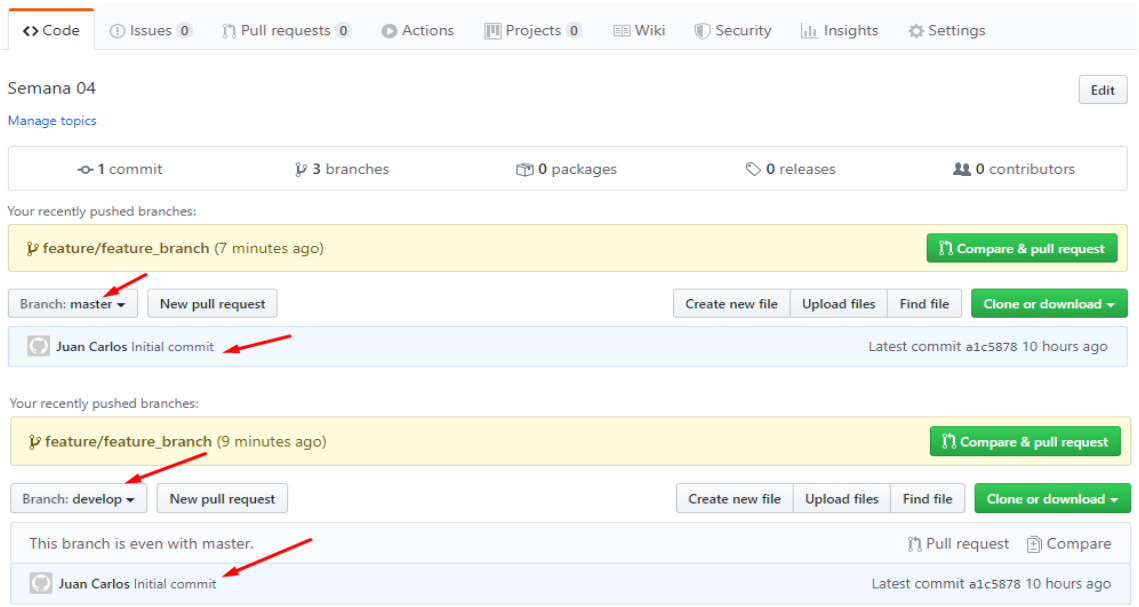
Luego publicaremos la nueva rama para su colaboración, para ello en nuestra consola bash escribimos:

```
$git flow feature publish feature_branch
```

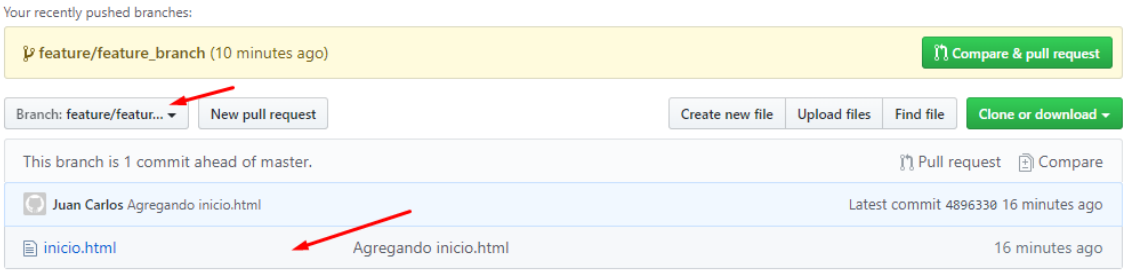


Revisamos GitHub para ver los cambios:

Rama master y develop:



Rama feature_branch:



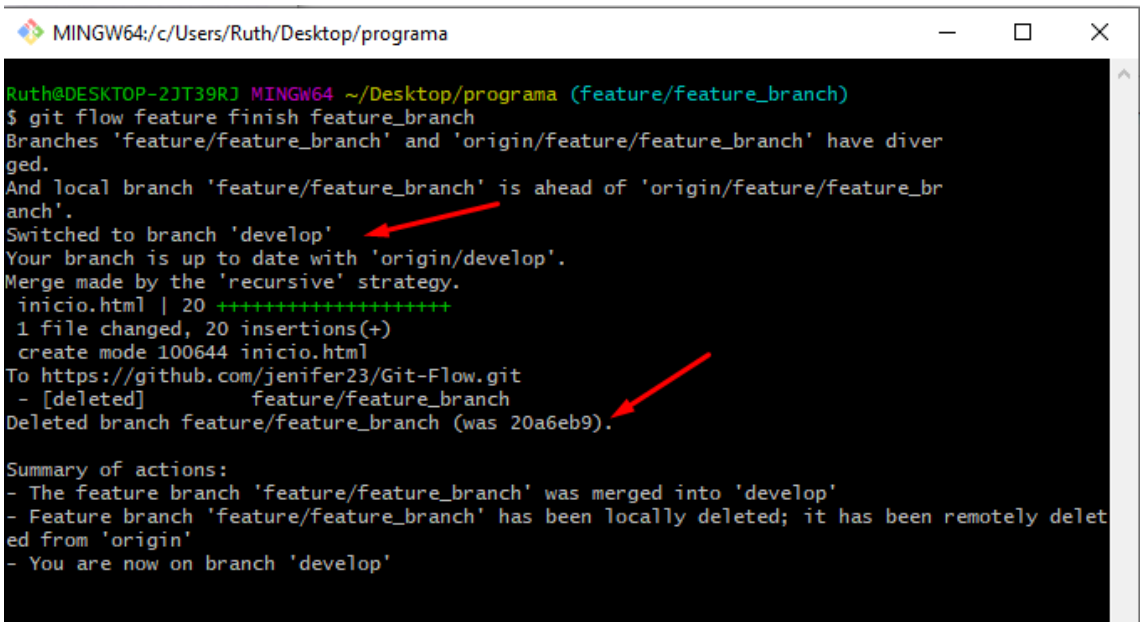
Cuando se termine el trabajo de desarrollo de la característica, el siguiente paso es fusionar el ***feature*** con la rama ***develop*** escribimos en la consola el siguiente comando:

```
$git flow feature finish feature_branch
```

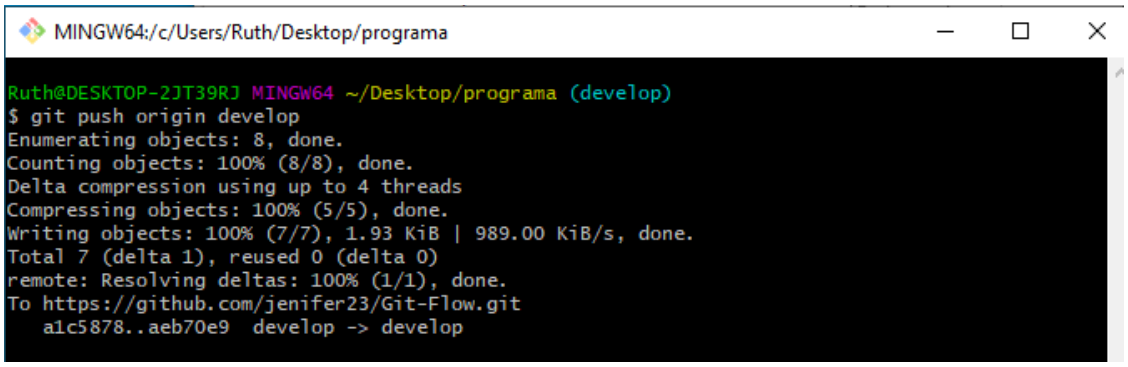
Nos abrirá el editor de texto de Git que por defecto es Vim, nos pedirá ingresar o modificar el commit, estos son los pasos para salir de Vim:

```
Primero se pulsa la Tecla ESC, para situarnos en el buffer inferior

:q → sale del archivo, si no tenemos cambios sin guardar sale sin más.

:q! → sale del archivo descartando los cambios no guardados.

:w → guarda los cambios del archivo, pero no sale de Vim para seguir editando el archivo.

:wq → guarda los cambios y sale de Vim.

:x → igual que el anterior.
```

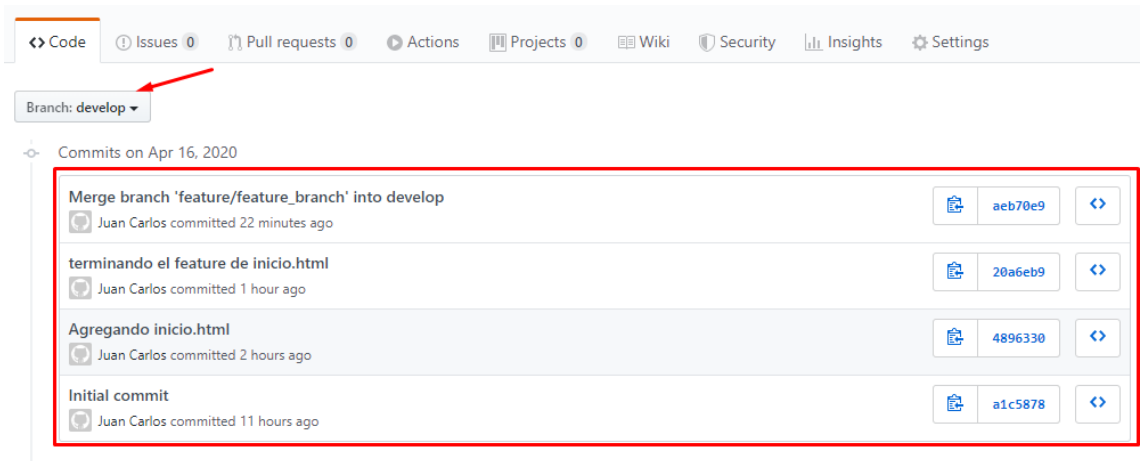
Git Flow automáticamente fusiona la rama ***feature_branch*** a ***develop*** y lo elimina, también elimina la rama del servidor remoto y cambia a la rama ***develop***.



Guardamos los cambios en el servidor remoto.



En GitHub nos muestra la rama develop:

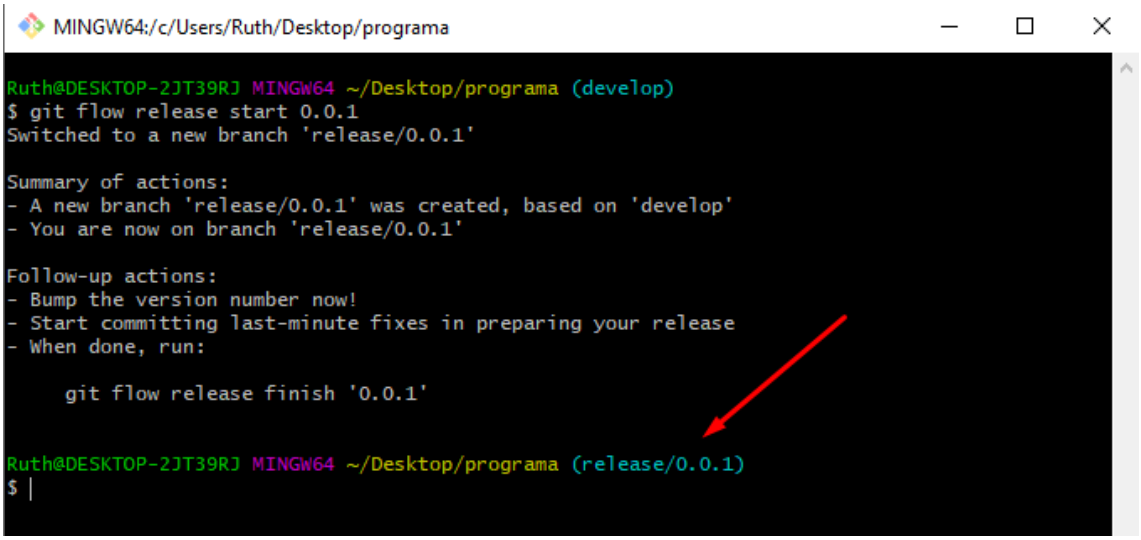


3. Creando una rama release

Con el uso de los comandos Git Flow en nuestra consola bash se escribe:

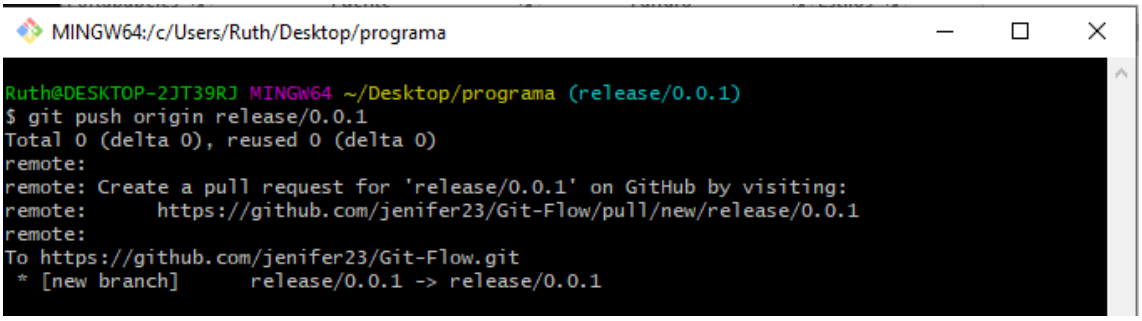
```
$git flow release start 0.0.1
```

Git flow automáticamente crea la nueva rama e hizo un checkout a ésta.

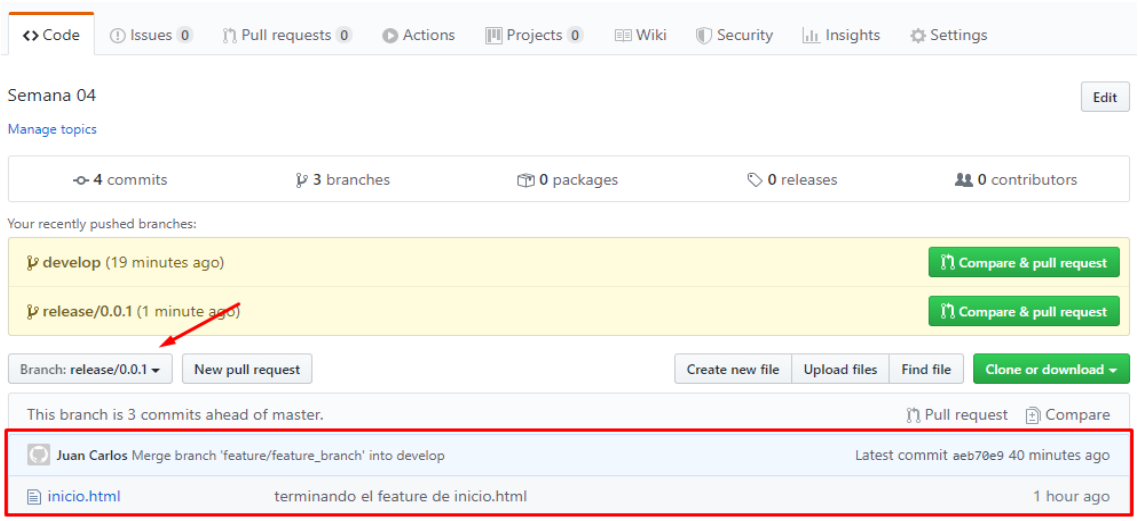


Los cambios se pueden enviar al servidor remoto para que los colaboradores puedan descargar:

```
$git push origin release/0.0.1
```

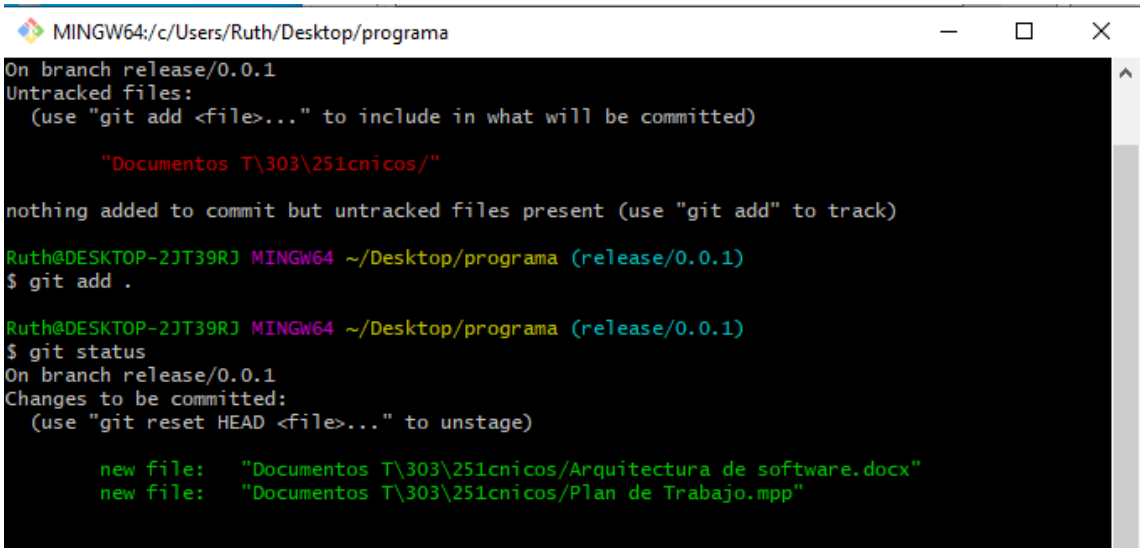


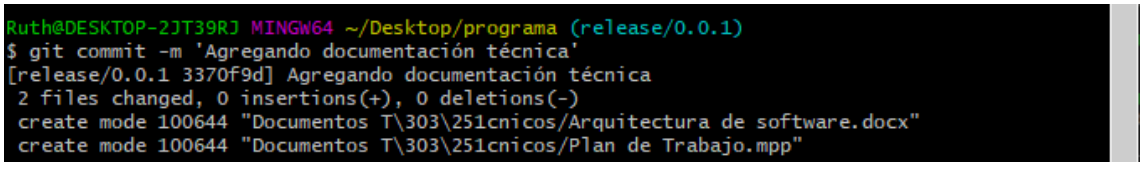
Se crea una copia de todo el contenido de la rama ***develop*** a la rama ***0.0.1***



Creamos la carpeta donde se guardará la documentación técnica del proyecto, luego guardamos los cambios.

```
$git status

$git add .

$git status

$git commit –m 'Agregando documentación técnica'
```





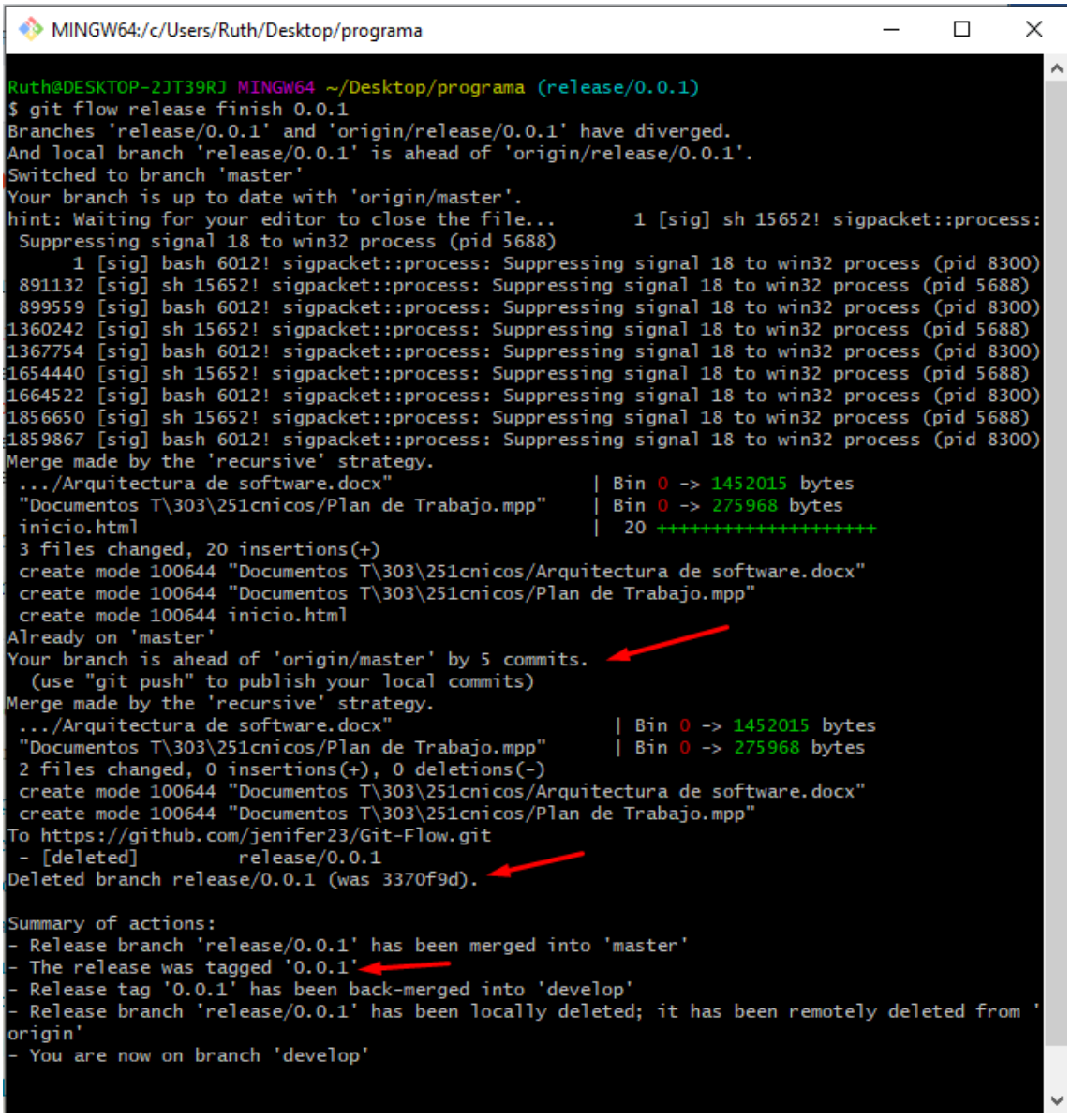
En ese punto se poder hacer Quality Assurance, todo lo que tenga que ver con testing y aseguramiento de calidad.

Para finalizar el relèase se escribe el siguiente comando:

```
$git flow release finish 0.0.1
```

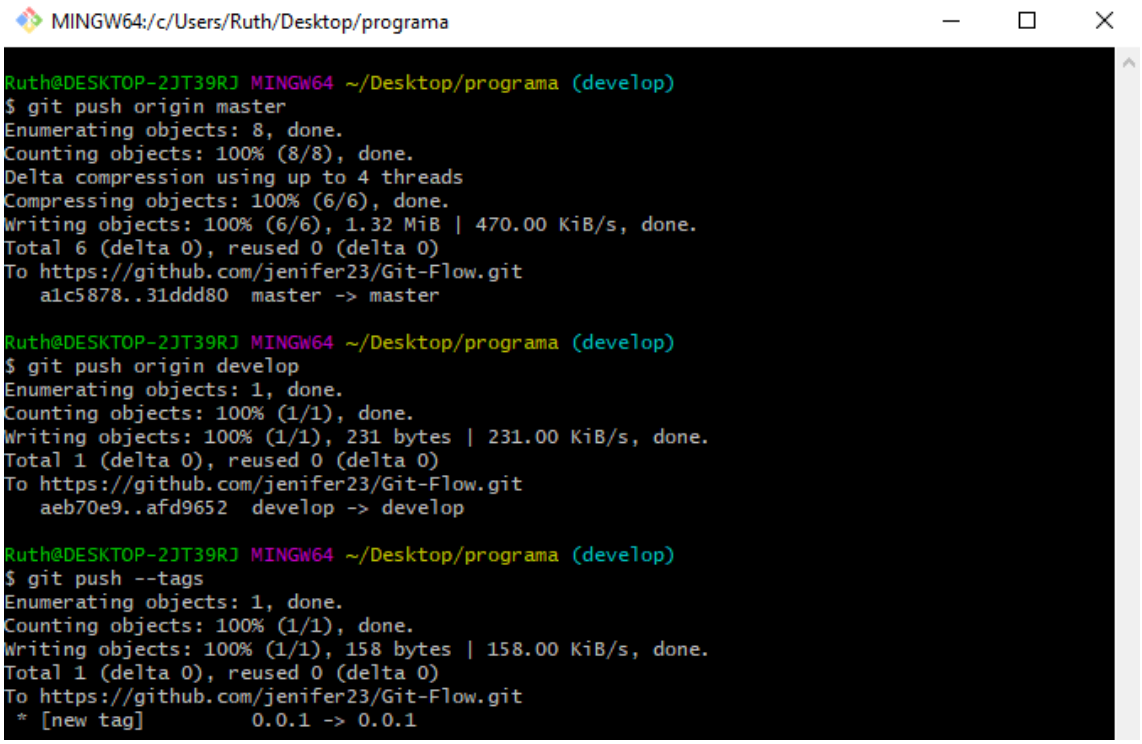
Nos muestra el editor de Vim, primero pedirá la descripción del commit y luego el número del tag para la rama ***master*** y luego el commit para la rama ***develop.***

Git flow, automáticamente fusiona la rama ***0.0.1*** a la rama ***master,*** le asigna un ***tag*** y elimina de forma local y remota la rama ***0.0.1.***

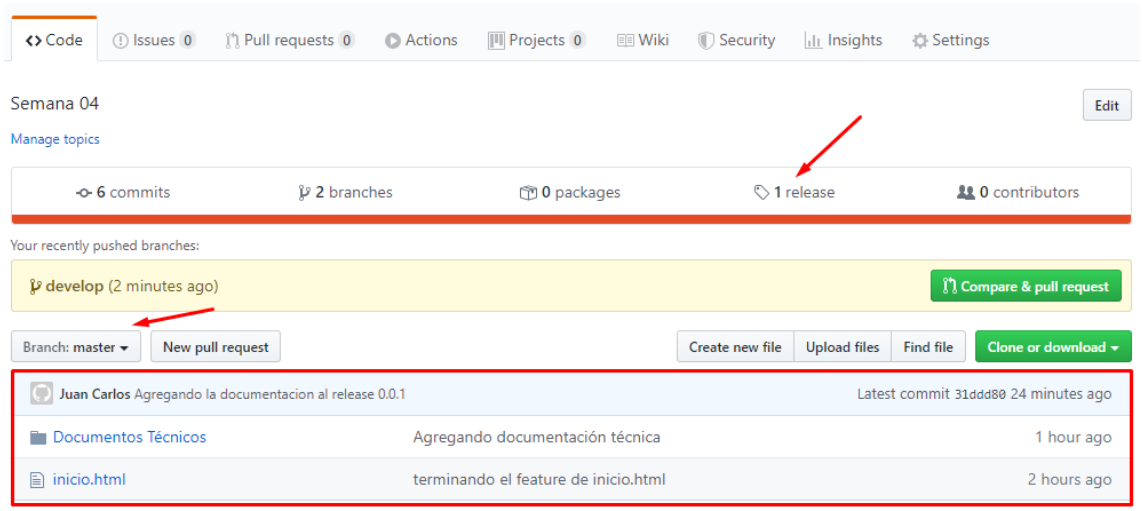


Para guardar los cambios en el servidor remoto:

```
$git push origin master

$git push origin develop

$git push --tags
```



En GitHub, nos mostraría lo siguiente:

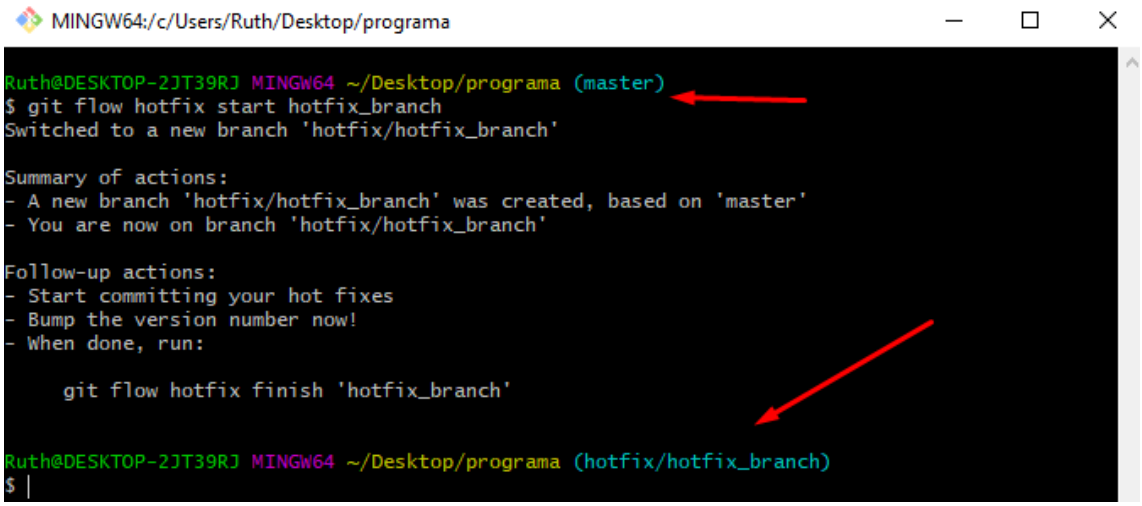


4. Creando una rama hotfix

Con el uso de los comandos Git Flow en nuestra consola bash desde la rama ***master*** se escribe:

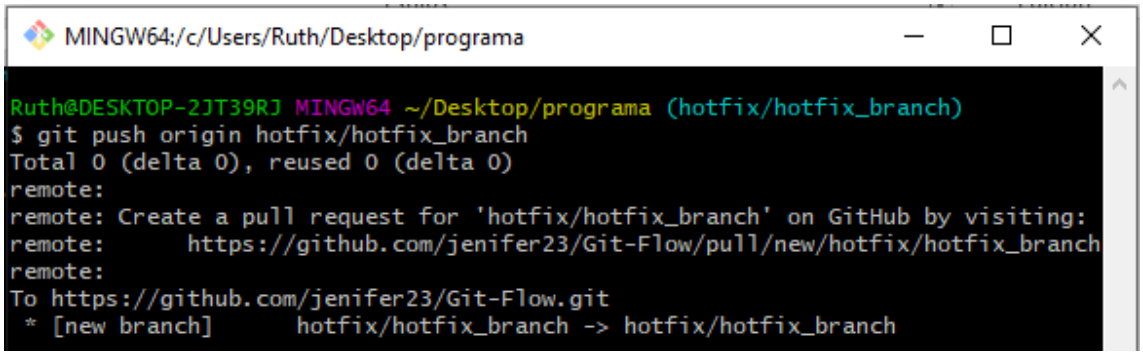
```
$git flow hotfix start hotfix_branch
```

Git flow automáticamente crea la nueva rama e hizo un checkout a ésta.

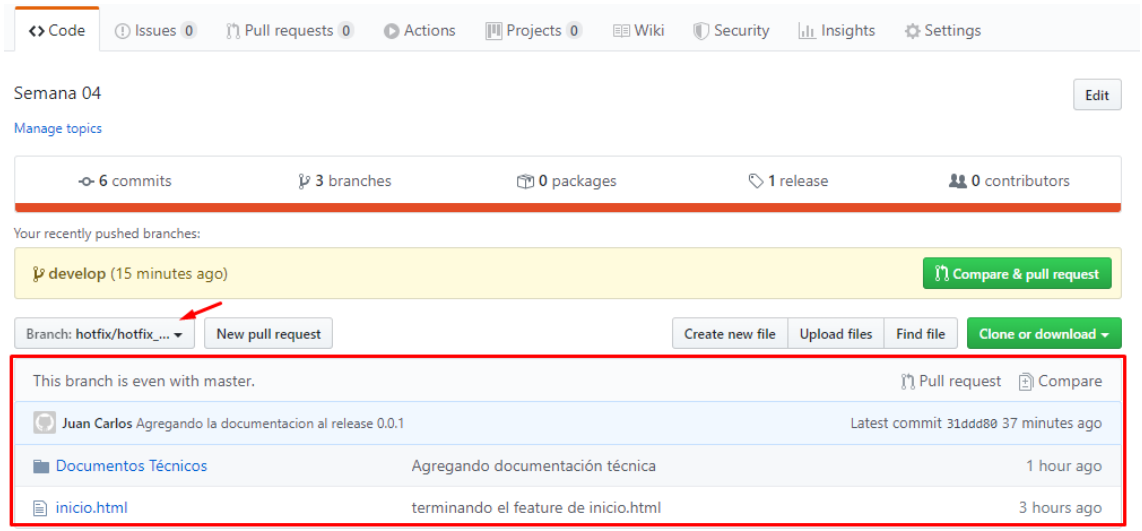


Los cambios se pueden enviar al servidor remoto para que los colaboradores puedan descargar:

```
$git push origin hotfix/hotfix_branch
```

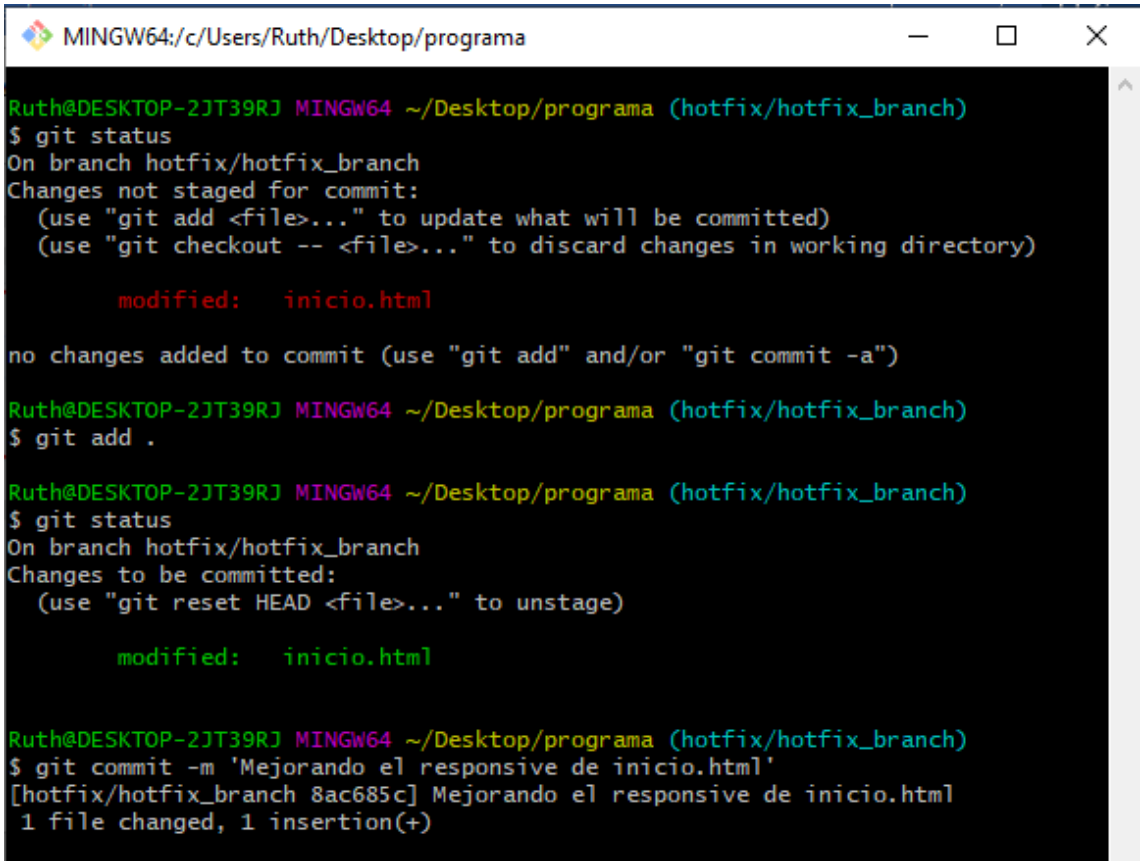


Se crea una copia de todo el contenido de la rama ***master*** a la rama ***hotfix_branch***



Modificamos nuestro ***inicio.html*** para que pueda ser responsive.

```
$git status
$git add .
$git status
$git commit –m ‘Mejorando el responsive de inicio.html’
```

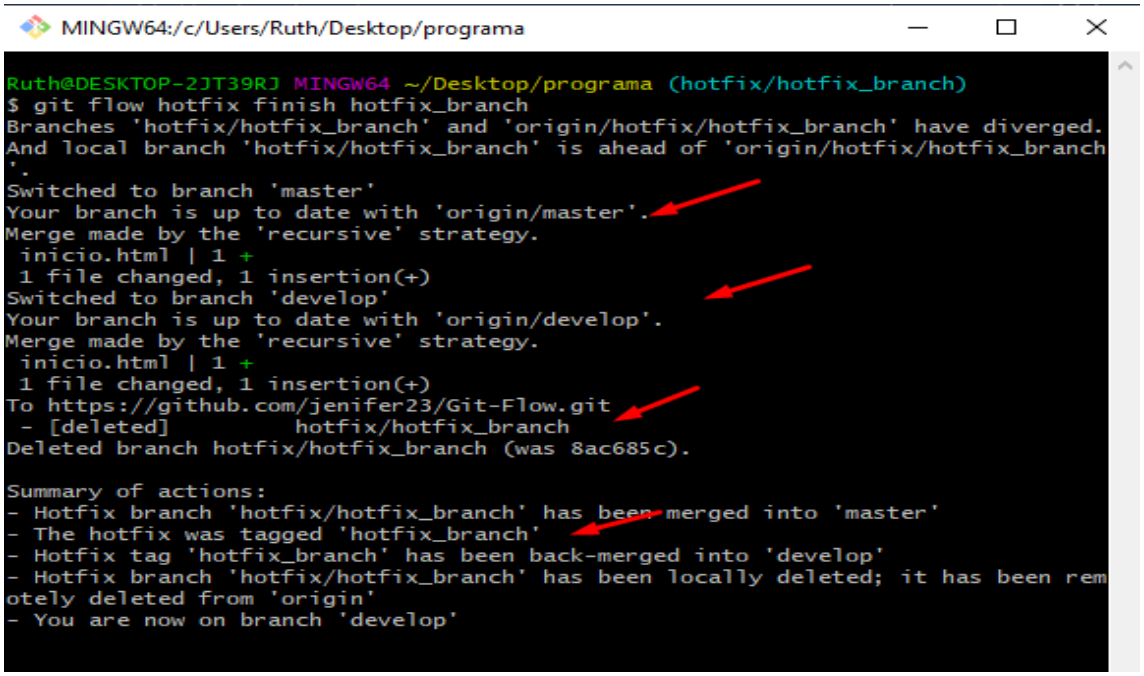


Para finalizar el hotfix se escribe el siguiente comando:

```
$git flow hotfix finish hotfix_branch
```

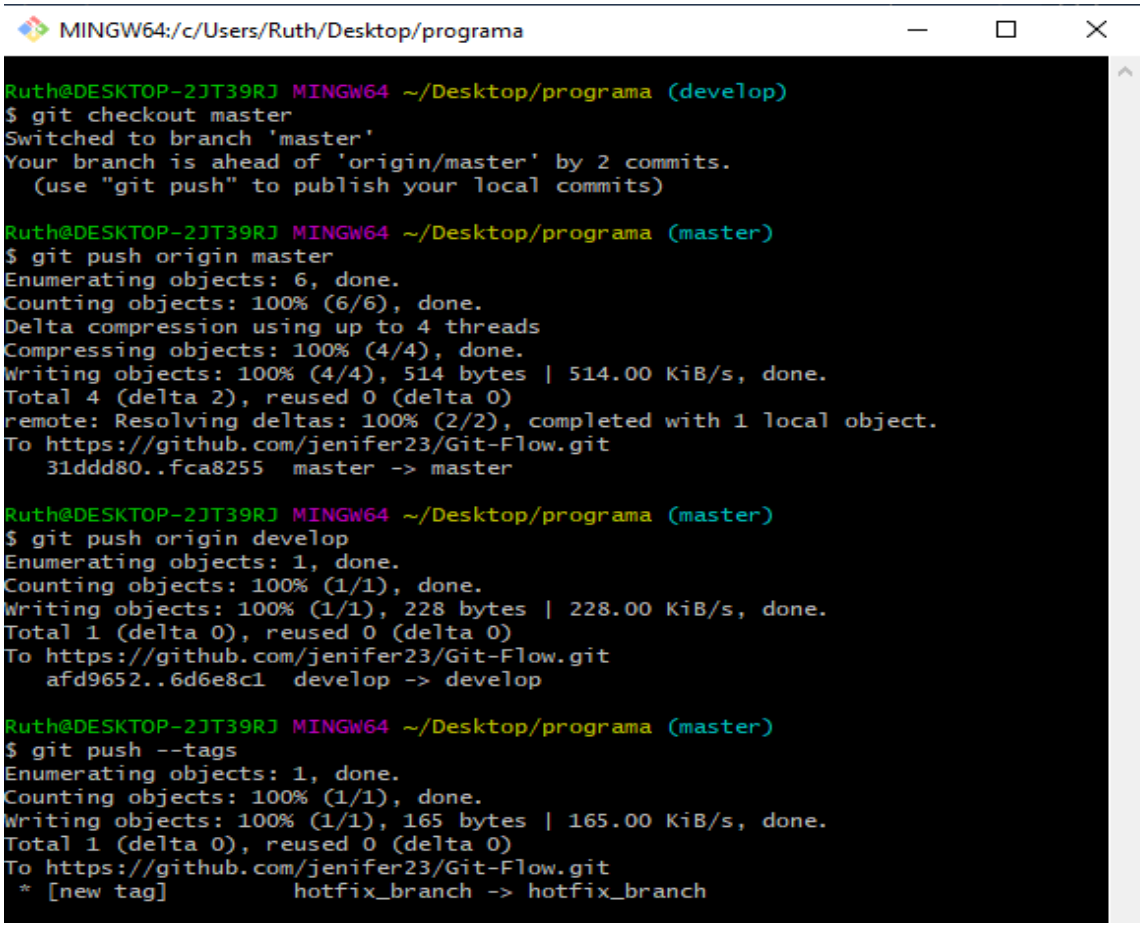
Nos muestra el editor de Vim, primero pedirá la descripción del commit y luego el número del tag para la rama ***master*** y luego el commit para la rama ***develop,*** esto con el fin de no perder en ***develop*** los arreglos hechos en ***master.***

Git flow, automáticamente fusiona la rama ***hotfix_branch*** a la rama ***master,*** le asigna un ***tag*** y elimina de forma local y remota la rama ***hotfix_branch.***



Para guardar los cambios en el servidor remoto desde la rama ***master***:

```
$git push origin master

$git push origin develop

$git push --tags
```



En GitHub, nos mostraría lo siguiente: