

Guía Ramas hotfix- feature BitBucket

Manejo de Incidentes y
requerimientos

Empresa: CMPC



Autores: Oscar Salgado

Luis Sandoval

Versión del documento: 2.0

Fecha Primera Versión: 23/12/2021

1 INTRODUCCIÓN	2
2 ESQUEMA GENERAL DE TRABAJO	3
2.1. ESQUEMA GENERAL HOTFIX (INCIDENTE).....	3
2.2. ESQUEMA GENERAL FEATURE (REQUERIMIENTO).....	4
3 PROCEDIMIENTO	5
3.1. REGLAS PARA LA UTILIZACIÓN DE LAS RAMAS.....	5
3.2. PROCEDIMIENTO	5
3.2.1. <i>Procedimiento hotfix (incidente)</i>	5
Crear Rama	5
Pull request desde hotfix a develop	10
Pull request desde hotfix a master (paso a producción)	13
Aprobación pull request y merge a master (Responsable Jefe de Proyecto CMPC).....	14
Eliminación rama Hotfix (paso final).....	16
3.2.2. <i>Procedimiento feature (requerimiento)</i>	18
Crear Rama	18
Pull request desde feature a develop.....	22
Pull request desde develop a master (paso a producción) + Aprobación del Jefe de proyecto.....	26
Eliminación rama feature (paso final).....	27

1 Introducción

El presente documento es una guía práctica del manejo de ramas feature y hotfix que será de utilidad en la medida que se trabaja con requerimientos e incidentes, respectivamente.

En primer lugar, se realizará una descripción general del manejo de rama que ha definido CMPC, para posteriormente pasar en la segunda parte del documento a una descripción detallada del paso a paso necesario para:

Manejo de hotfix (incidentes).

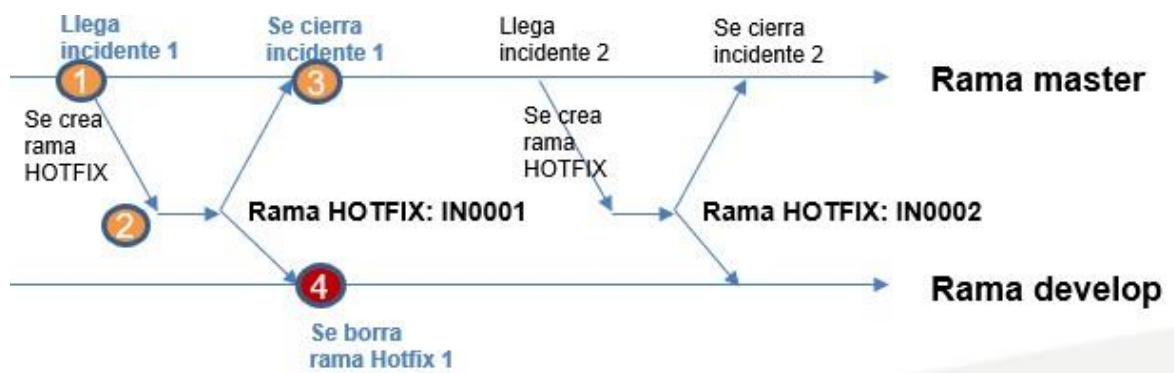
Manejo de feature (requerimientos).

2 Esquema General de trabajo

Este documento persigue aclarar el procedimiento y pasos que se deben seguir para llevar a cabo un requerimiento o un incidente (feature o hotfix, respectivamente).

A continuación, se presentan dos esquemas que muestran los movimientos que se realizan al manejar ramas hotfix y ramas feature.

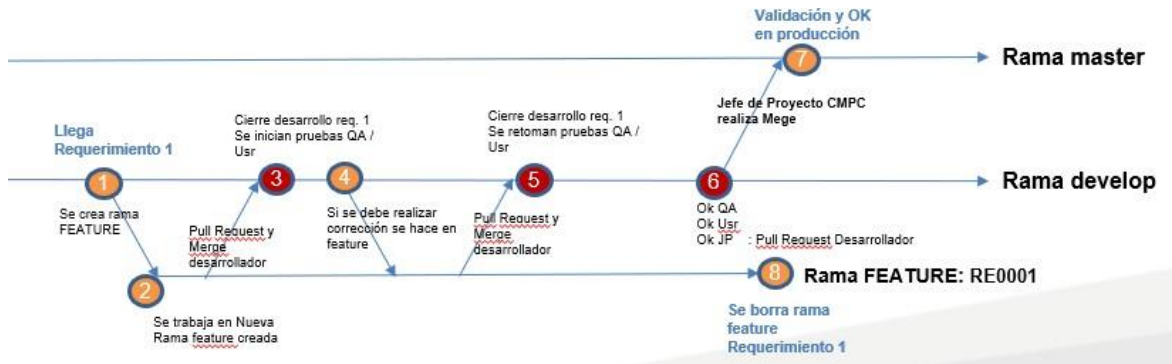
2.1. Esquema general hotfix (incidente).



1. Al llegar un Incidente se debe proceder a la creación de una rama hotfix
2. Una vez creada la rama hotfix, se debe trabajar el incidente.
3. Una vez terminada la solución del incidente se debe realizar un pull request para unirse a la rama master nuevamente y llevar los códigos finales, el jefe de proyecto debe realizar el pull request, quien además deberá aprobar y realizar merge hacia la rama master, para que los códigos efectivamente sean traspasados¹.
4. posteriormente se debe realizar un pull request a la rama develop, si esto no se realiza los requerimientos que fluyen desde la rama master podrían verse afectados al no contar con los últimos cambios. Este paso requiere vital atención dado que al encontrarse con cambios sobre los componentes modificados, producto de otros desarrollos en curso, se deberá realizar una **coordinación** con el o los desarrolladores involucrados, para alertar del nuevo cambio realizado y para asegurar que siempre sea considerado, por todos aquellos requerimientos que puedan estar al mismo tiempo en pruebas de usuario o de QA.

¹ Por definición es responsabilidad del JP de CMPC realizar el pull request y merge hacia la master. Antes la solicitud de pull request hacia la master la realizaba el desarrollador conjuntamente con las gestiones de paso a producción, y era el Jefe de proyecto quien aprobaba y realizaba merge. Ahora es responsabilidad del desarrollador hacer las gestiones de paso a producción enviando la Excel de paso, y el JP realiza el pull request para dejar los códigos en la rama master. El desarrollador es quien hace el pull request a la rama develop.

2.2. Esquema general feature (requerimiento)



1. Al llegar un requerimiento se debe proceder a crear una rama feature.
 2. Una vez creada la rama feature, se debe trabajar el requerimiento.
 3. Una vez terminada la solución del requerimiento se debe realizar un pull request para unirse a la rama develop nuevamente y llevar los códigos finales, posterior a esto se podrán realizar las pruebas de QA, usuarios o del JP de CMPC según sea requerido.
- En este punto es posible que otros desarrolladores hayan realizado cambios sobre la rama develop, por lo cual se deberá realizar una coordinación con ellos para asegurar no se vean afectados y no se pierda código de ninguna de las partes.
4. En caso de existir observaciones, errores a los códigos revisados, se deberá regresar a la rama feature, para trabajar los cambios, los cambios NUNCA deben ser realizados directamente en una rama distinta a feature².
 5. Una vez corregidos los errores, lo cual podrá ocurrir mas de una vez, llevándose a cabo los pasos 4 y 5 cuantas veces sea necesario, se realizarán las pruebas finales de QA, Usuario o JP.
 6. En caso de existir aprobación del requerimiento, el desarrollador deberá informar al Jefe de proyectos, para que este realice pull request hacia la rama master, haciendo el merge final de los códigos a la rama master³.
 7. Una vez realizado el merge a la rama master, el JP debe llevar los fuentes al servidor respectivo y el usuario deberá dar el ok en producción.
 8. Una vez recibido ok del usuario y JP, el desarrollador deberá cerrar la rama feature que fue utilizada para resolver el requerimiento.

² Lógicamente según el paso descrito, nos referimos a rama feture, pero nunca deben ser realizado cambios en ramas distintas a feature o hotfix, según corresponda.

³ Esto es un cambio a la definición antigua, donde el desarrollador realizaba el pull request y el JP de CMPC realizaba el approve y el merge, ahora es solo el JP de CMPC quien realiza el pull request, a solicitud de CMPC.

3 Procedimiento

3.1. Reglas para la utilización de las ramas

1. La rama **master** es una rama que está estrechamente controlada y a la cual solo debe llegar código estable correspondiente a producción.
2. La rama **develop** es una rama utilizada para construir ramas **feature**, en donde se implementan las nuevas funcionalidades o cambios. Pero no es una rama sobre la cual se deba trabajar.
3. La rama **hotfix** surge para reparar rápidamente errores que hayan llegado a producción, es decir, se generan desde la rama **master**.
4. La rama **feature** es utilizada para llevar a cabo requerimientos (no errores urgentes), esta rama se crea desde la rama **develop**.

La nomenclatura para creación de ramas **feature** y **hotfix** debe ser con el número de ticket que gatilla la modificación o la resolución de un incidente/requerimiento productivo y su respectivo prefijo.

Ejemplo:

```
hotfix/IN1318865  
feature/RE1318864
```

3.2. Procedimiento

1. Se recibe un ticket de Requerimiento o Incidente de un sistema determinado.

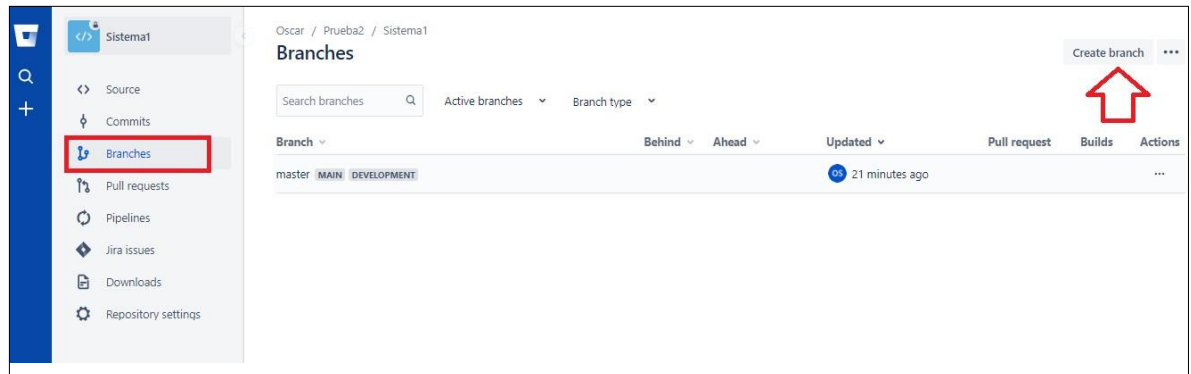
3.2.1. Procedimiento hotfix (incidente)

Crear Rama

2. **Si es un incidente**, se crea una rama **hotfix** con el prefijo en minúscula “hotfix” + nombre del tipo de ticket (IN) + número incidente (1318865), dando como resultado la rama: “**hotfix/IN1318865**”.

Para crear la rama hotfix desde bitbucket(repositorio remoto):

2.1. Posicionados en el repositorio del proyecto: en el menú lateral izquierdo seleccionar “Branches” y luego click al botón superior derecho de la pantalla “Create Branch”, se desplegará una ventana “pop-up”.



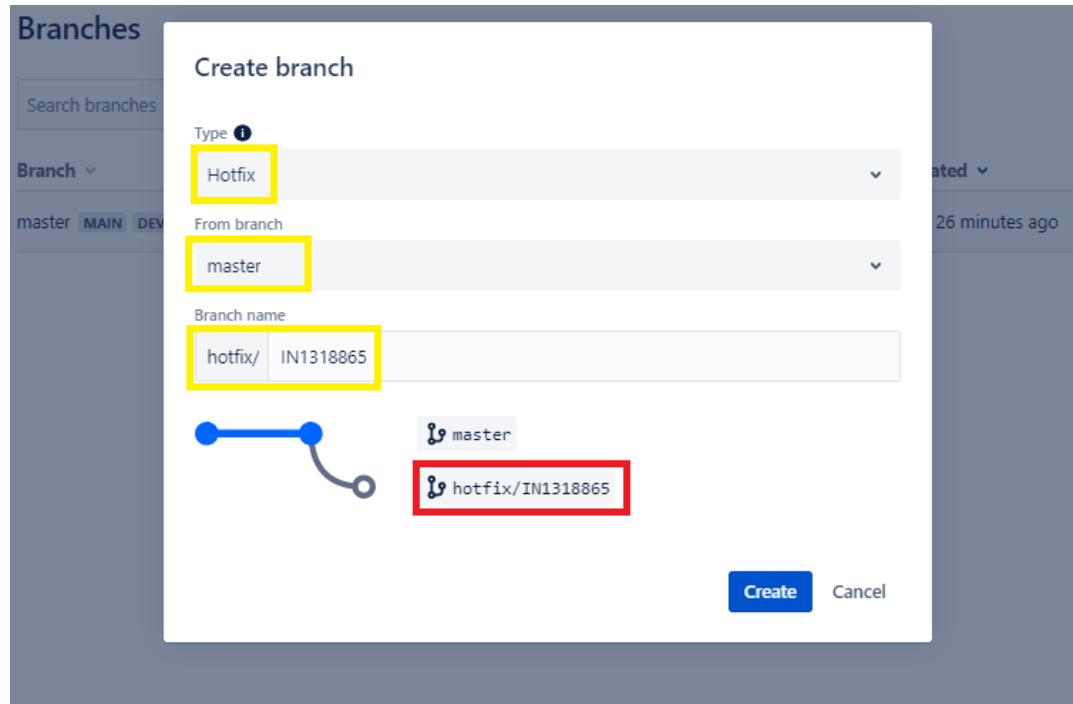
2.2. Seleccionar las opciones en ventana pop-up, según se muestra en la imagen:

Type: Hotfix

From Branch: master (ya que una rama hotfix por definición surge a partir de la master)

Branch name: IN+<número de incidente>

Como se aprecia en la imagen, se genera de manera automática el prefijo **hotfix**, luego dar click en create para crear la rama.

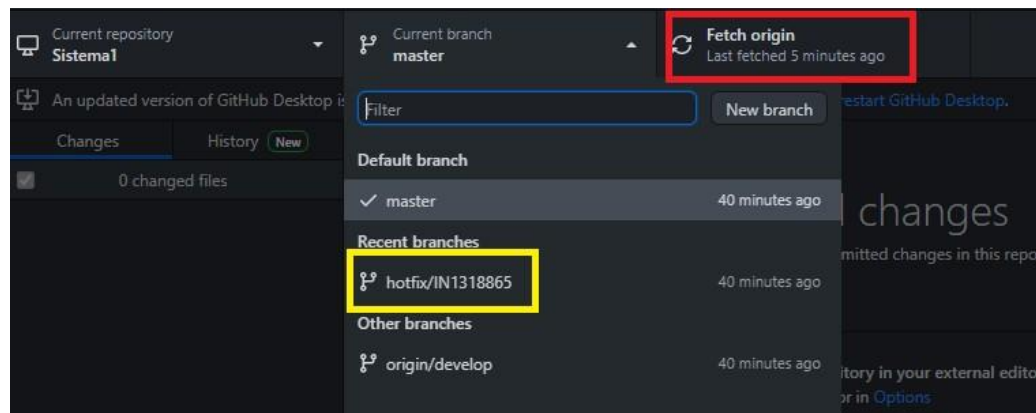


2.3. Una vez creada la rama hotfix en bitbucket, es necesario traerla a nuestro equipo (repositorio local) para trabajar sobre ella.

Esto se puede hacer a través de la aplicación “Github Desktop” (detallada en guías anteriores de subidas a bitbucket⁴).

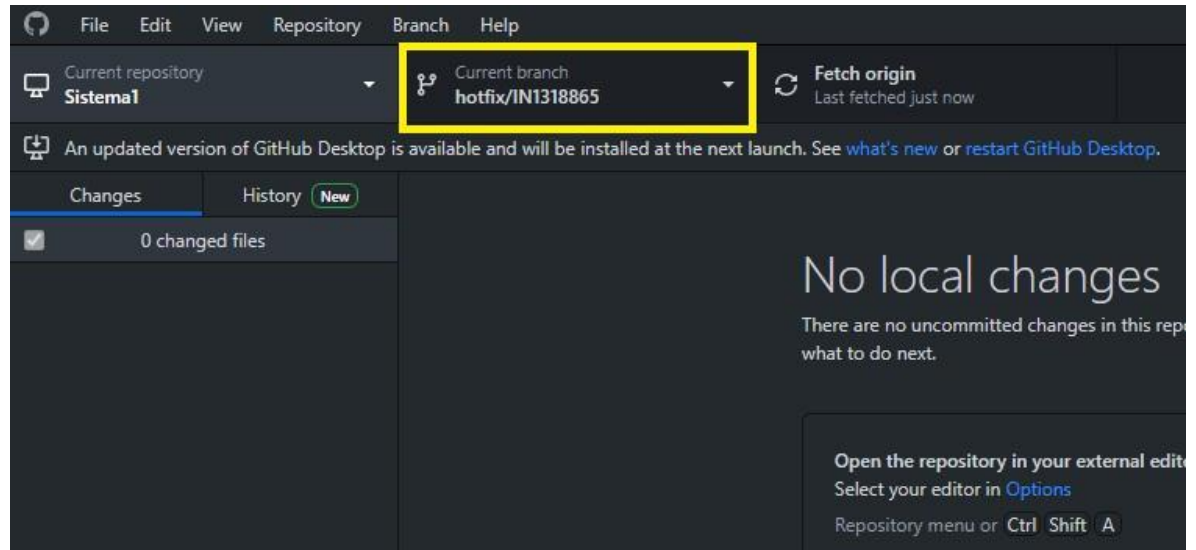
Presionar botón “Fetch origin” para traer a nuestro repositorio local los nuevos cambios desde repositorio remoto de bitbucket.

Luego en “Current Branch” seleccionar la rama hotfix creada, tal como se muestra en la imagen:



Resultado: debe quedar seleccionada la rama hotfix:

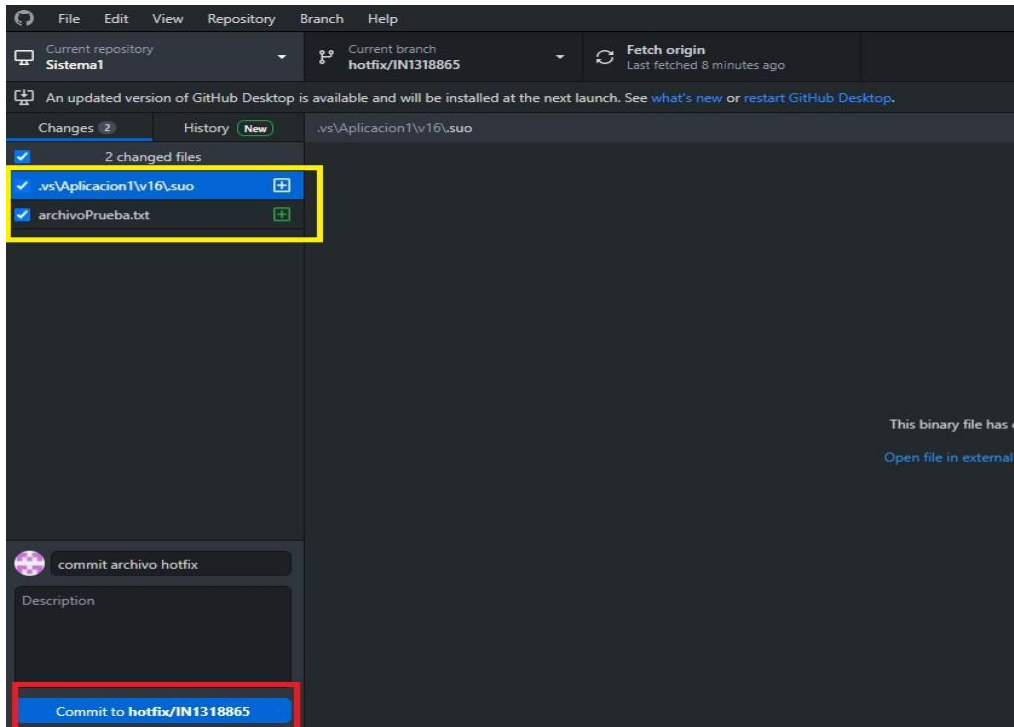
⁴ Para más detalles ir a PASO 1. Clonar Repositorio Existente en BitBucket, de la Guía práctica Bitbucket.



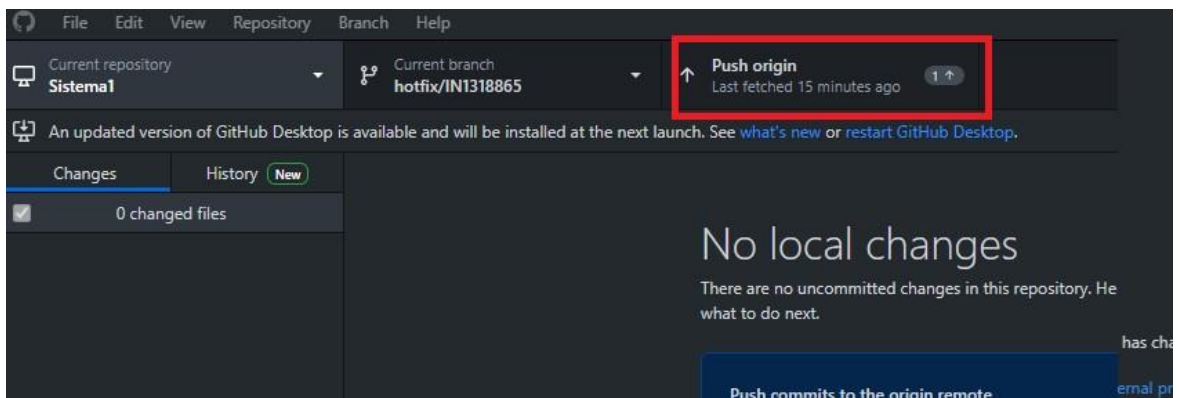
2.4. Una vez realizado los cambios de manera local, estos deben ser enviados a la rama repositorio remoto bitbucket:

“Github Desktop” detectará los cambios en archivos de manera automática (como se aprecia en la pestaña “Changes” en la imagen), para subirlos a bitbucket es necesario presionar el botón del panel derecho “commit to hotfix/IN1318865”, previo llenado del campo obligatorio “Summary”:

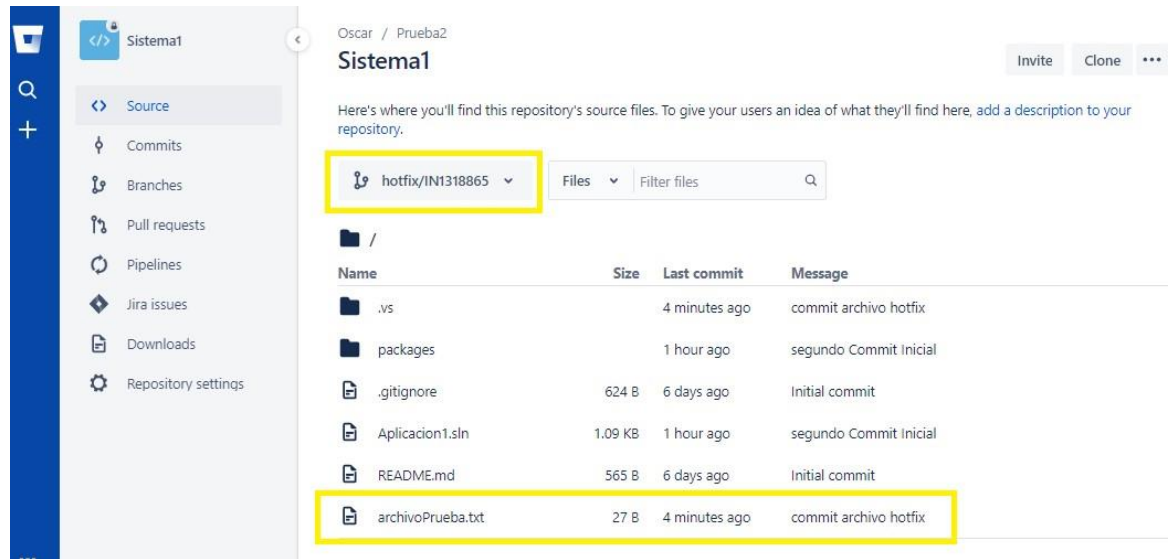
Para el ejemplo se creó un archivo llamado “archivoPrueba.txt”



Luego del commit anterior presionar botón “Push origin” para de esta forma enviar nuestros cambios locales al repositorio remoto.



2.5. Validando en repositorio remoto Bitbucket: Seleccionando la rama “hotfix/IN1318865”, vemos que se cargó el archivo “archivoPrueba.txt”:



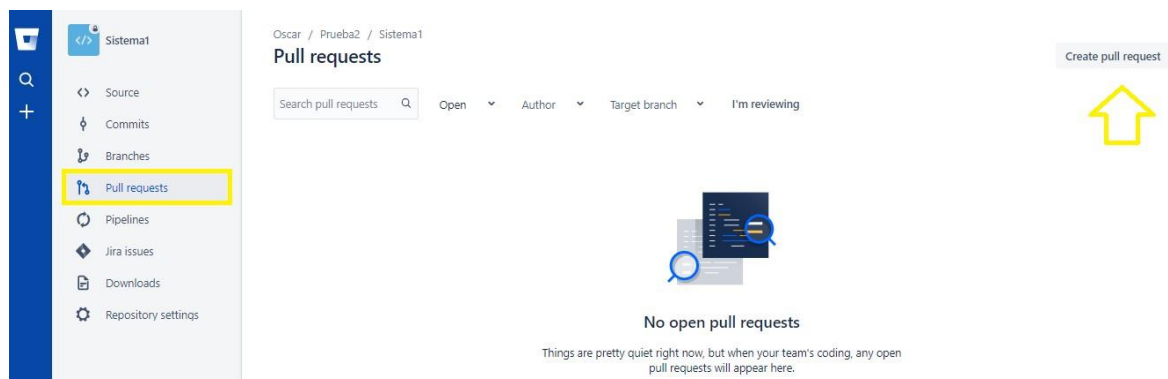
Pull request desde hotfix a develop

- Una vez resuelto el incidente, los respectivos cambios realizados y subidos a rama **hotfix en bitbucket**, se deben llevar hacia las ramas **develop** y **master**, esto se debe complementar con la planilla Excel de paso a producción y la coordinación debida con el jefe de proyecto CMPC.

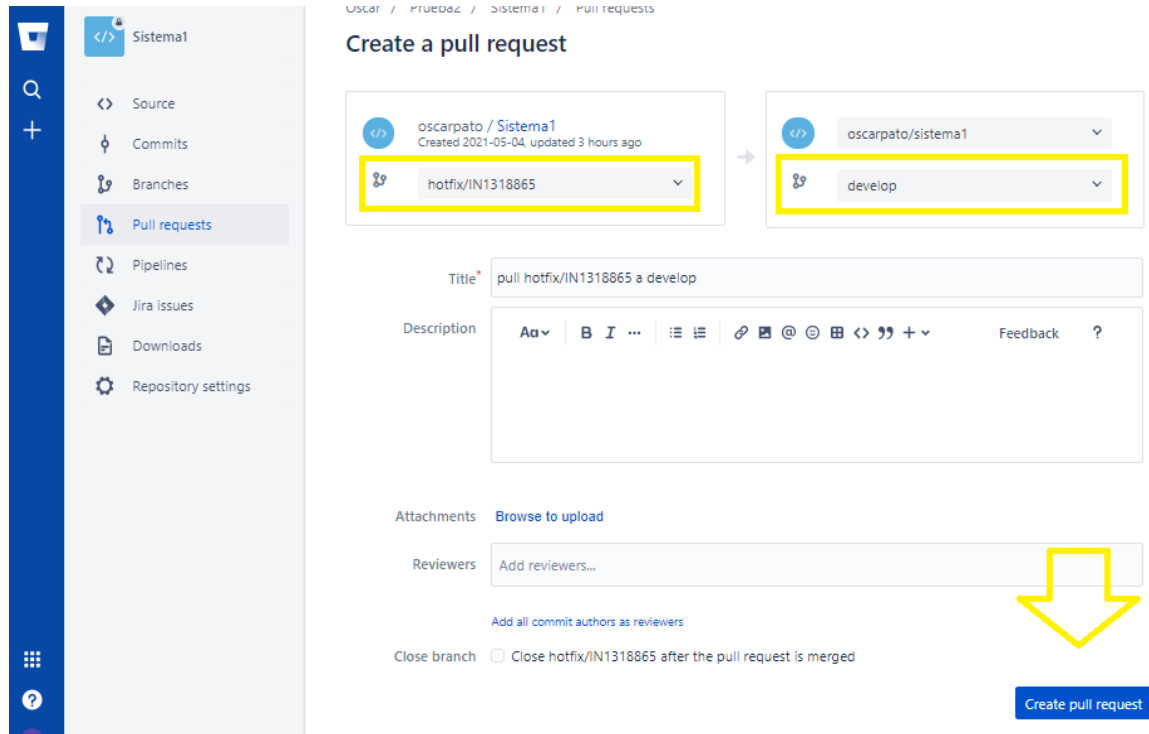
Para pasar la rama **hotfix a develop** y **master**, se debe realizar en forma coordinada y preferentemente el mismo día:

3.1. **Pull request** desde rama **hotfix a develop** (a cargo del desarrollador)

3.1.1. Nuevamente vamos al menú izquierdo y seleccionamos “**Pull requests**” y click al botón “**Create pull request**”

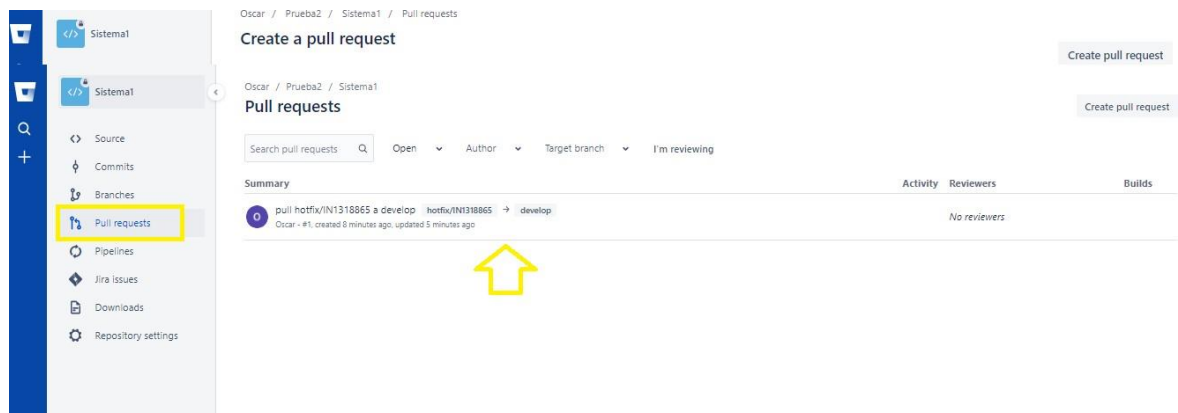


3.1.2. Se desplegará una ventana y seleccionamos: **hotfix/IN1318865** a **develop** como se muestra en la imagen, luego click a botón **“Create Pull request”**



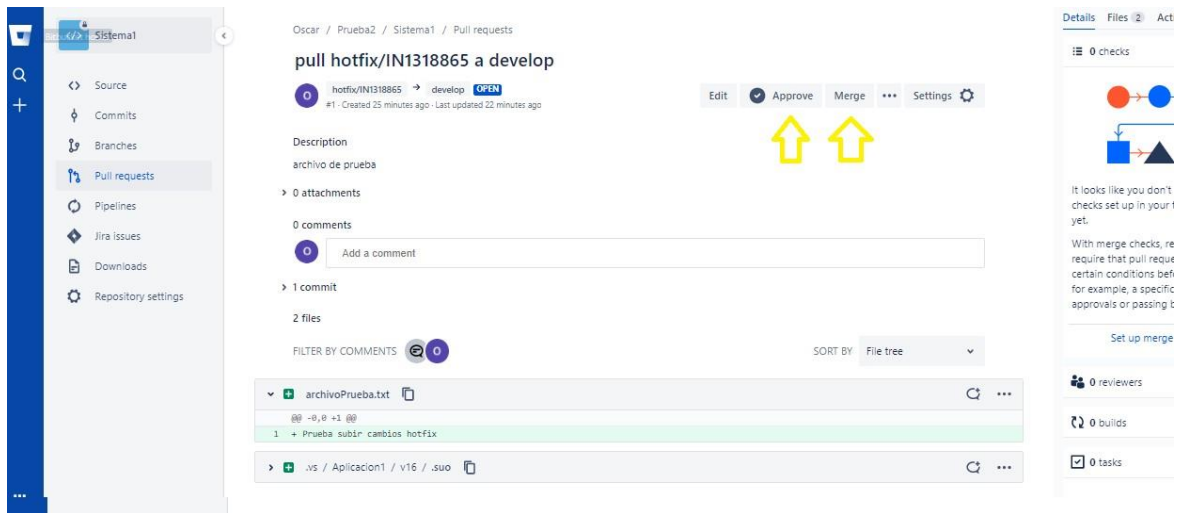
Nota: No se debe ingresar un **Reviewers** para **pull request** hacia **develop**, esto sólo es requerido para hacer **Pull Request** a la rama **Master**.

3.1.3. Se valida en menú lateral izquierdo seleccionando **“Pull request”**, que efectivamente se generó el pull request desde hotfix hacia develop (A la espera de aprobación y Merge con rama develop)

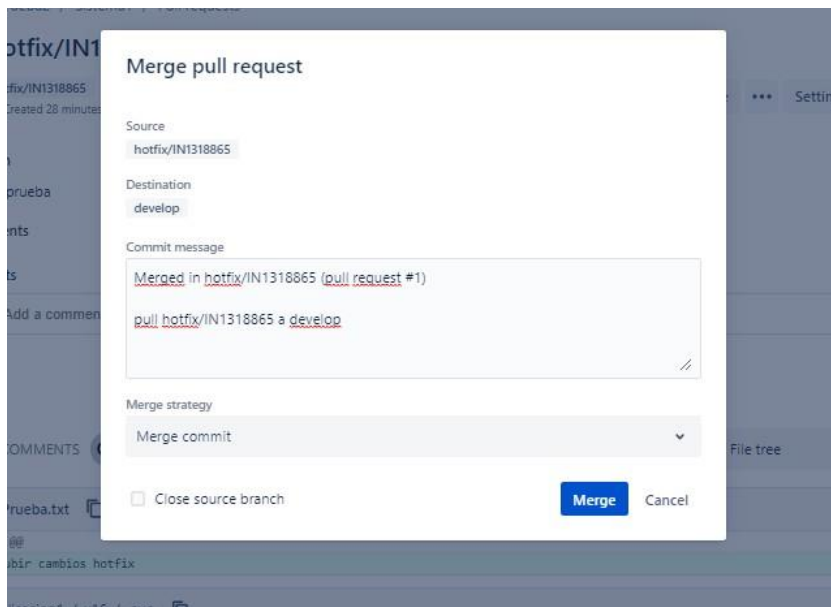


3.1.4. Merge de rama **hotfix** a **develop**: Ingresar al **pull request** que figura en el listado de la imagen anterior (a cargo del desarrollador)

Seleccionando los botones **“Approve”** y **“Merge”** (en ese orden), se realizará el merge hacia develop, por lo tanto, los cambios realizados en la rama hotfix pasarán a la develop.

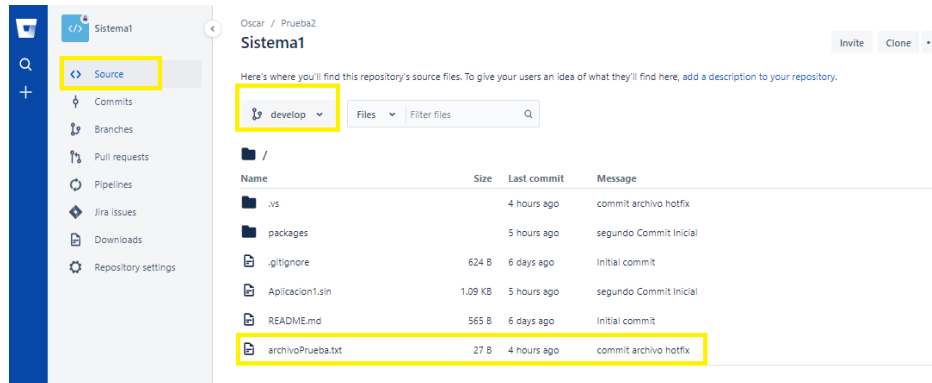


Luego de presionar **“Merge”**, se desplegará otra pantalla pop-up, hacer click en **“Merge”**:



Nota: no se debe seleccionar **“Close source Branch”**, dado que esto eliminaría la rama hotfix.

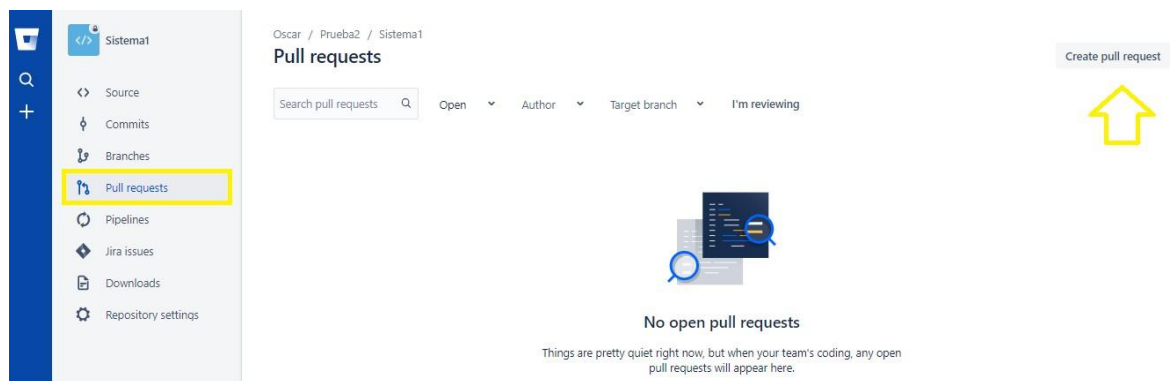
Posteriormente se valida que rama develop contiene los cambios generados en la rama hotfix:



Pull request desde hotfix a master (paso a producción)

3.2. Pull request desde rama hotfix a Master (a cargo del Jefe de Proyecto)

3.2.1. Nuevamente vamos al menú izquierdo y seleccionamos **“Pull requests”** y click al botón **“Create pull request”**

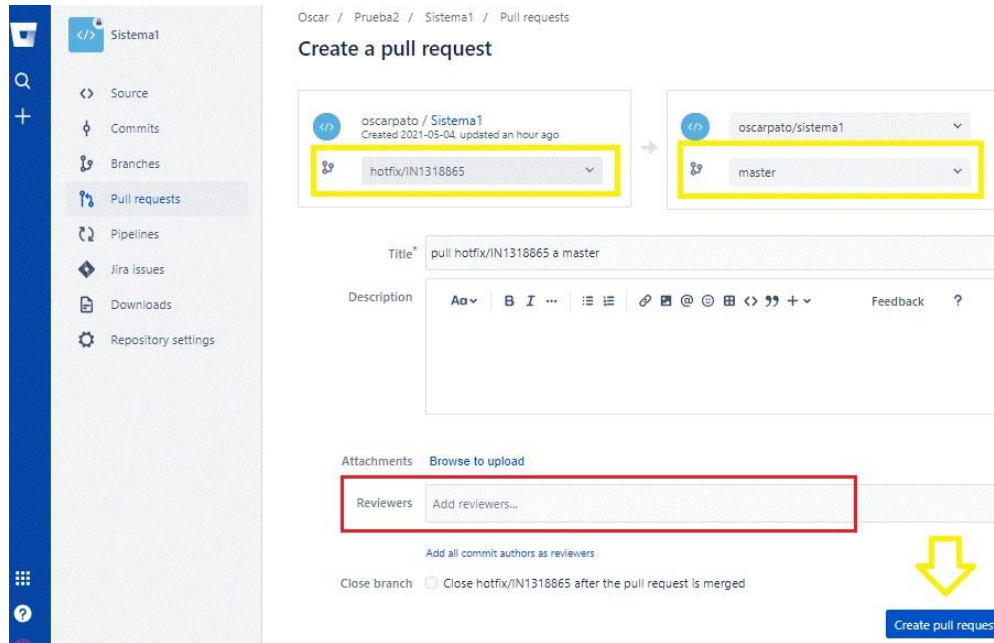


3.2.2. Se desplegará una ventana y seleccionamos: **hotfix/IN1318865⁵** a **master** como se muestra en la imagen.

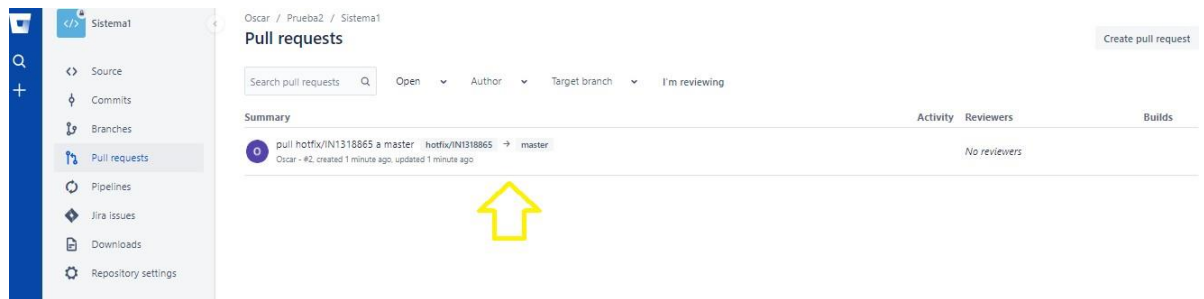
El desarrollador no hace el Pull Request a la master, Por definición de CMPC, solo el jefe de proyecto CMPC debe realizar pull request, aprobar y hará Merge a la Master (el desarrollador es responsable de realizar esto a la rama develop).

Luego click a botón **“Create Pull request”** y habiendo agregado al JP CMPC al campo **reviewers**, le llegará un correo informativo con la solicitud del Pull Request para su aprobación y Merge.

⁵ El número IN1318865, es sólo un ejemplo, se debe seleccionar el incidente en el cual se esté trabajando.



3.2.3. Se valida en menú lateral izquierdo seleccionando “Pull request”, que efectivamente se generó el pull request desde hotfix hacia master.



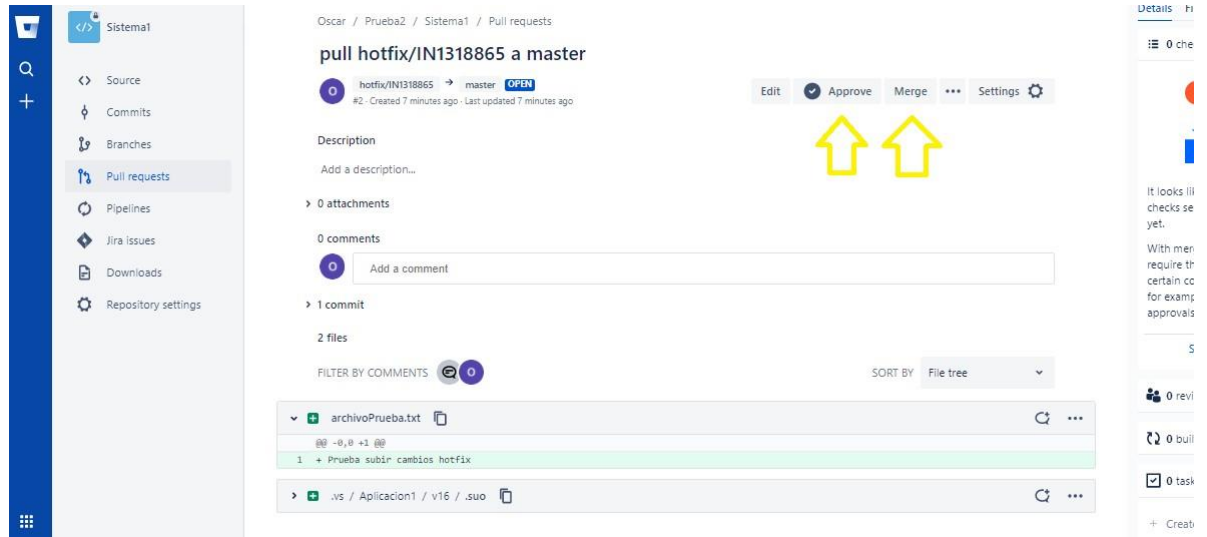
Aprobación pull request y merge a master (Responsable Jefe de Proyecto CMPC)

3.2.4. Aprobación de Pull Request y Merge hacia master:

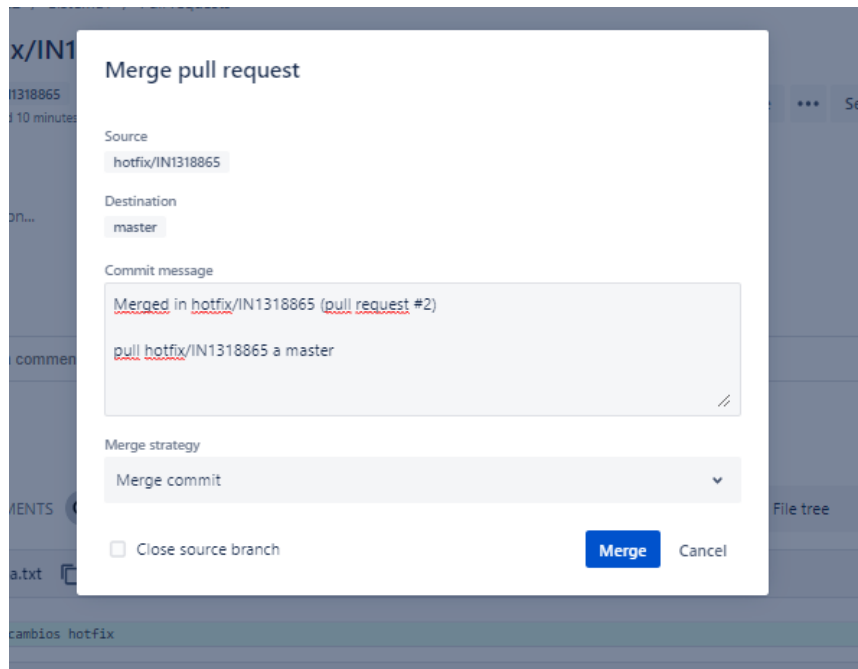
Importante: Por definición el Merge hacia la rama master debe ser realizado y aprobado por JP CMPC.

Ingresa al **pull request** que figura en el listado de la imagen anterior.

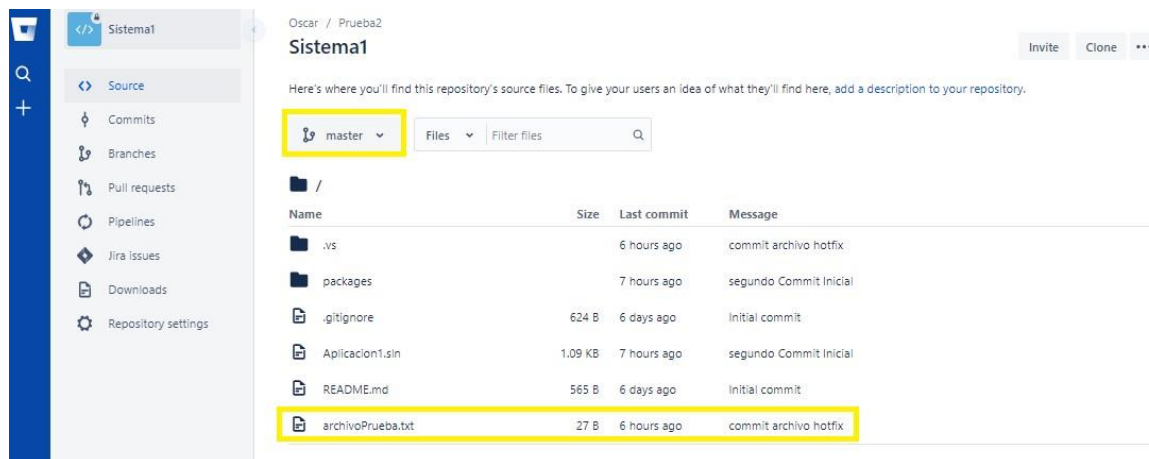
Seleccionando los botones “**Approve**” y “**Merge**” (en ese orden), se realizará el merge hacia master, por lo tanto, los cambios realizados en la rama hotfix pasarán a la master.



Luego de presionar “Merge”, se desplegará otra pantalla pop-up, hacer click en “Merge”:



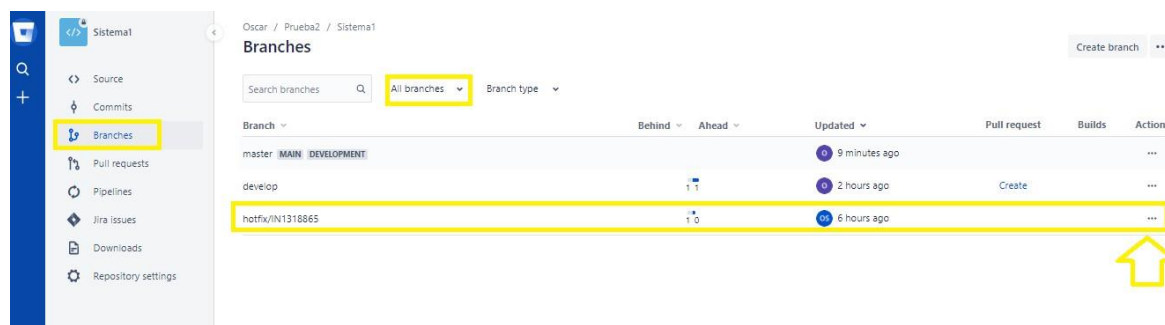
Posteriormente se valida que rama **master** contiene los cambios generados en la rama hotfix:



Eliminación rama Hotfix (paso final)

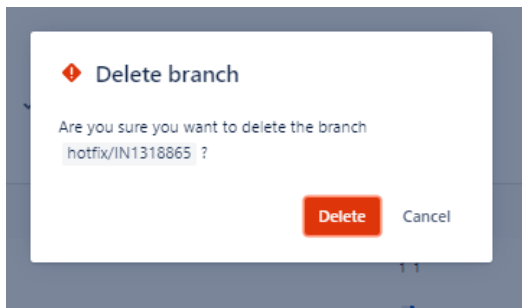
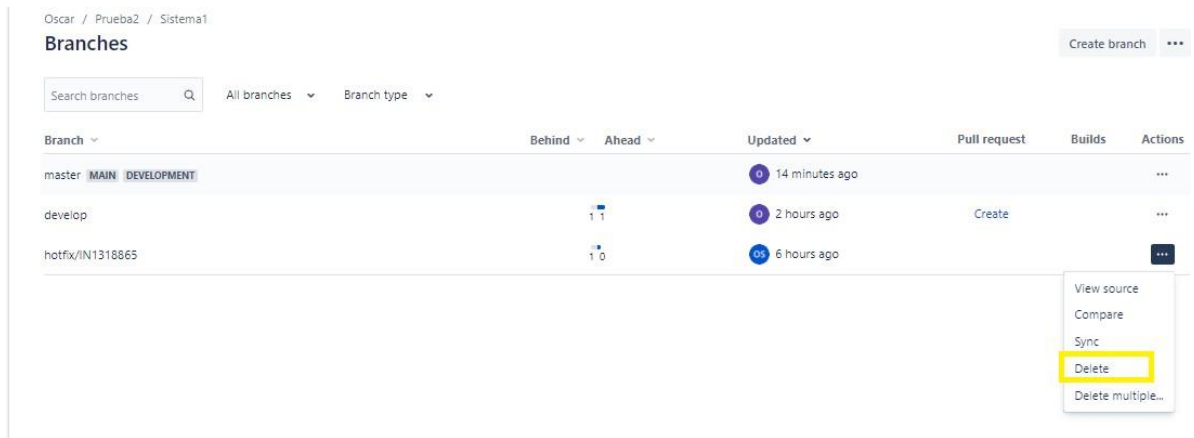
3.2.5. **Eliminación rama hotfix⁶**: Una vez realizado el paso a producción y habiendo validado que los cambios generados en **hotfix**, estén disponibles en **develop** y **master**, se puede proceder a eliminar la rama **hotfix**:

Desde Bitbucket en menú lateral izquierdo seleccionar “Branches”, luego “All branches” en selector superior y click a los tres puntos (...) de la rama hotfix a eliminar, en este caso a “hotfix/IN1318865” como se muestra en la imagen:



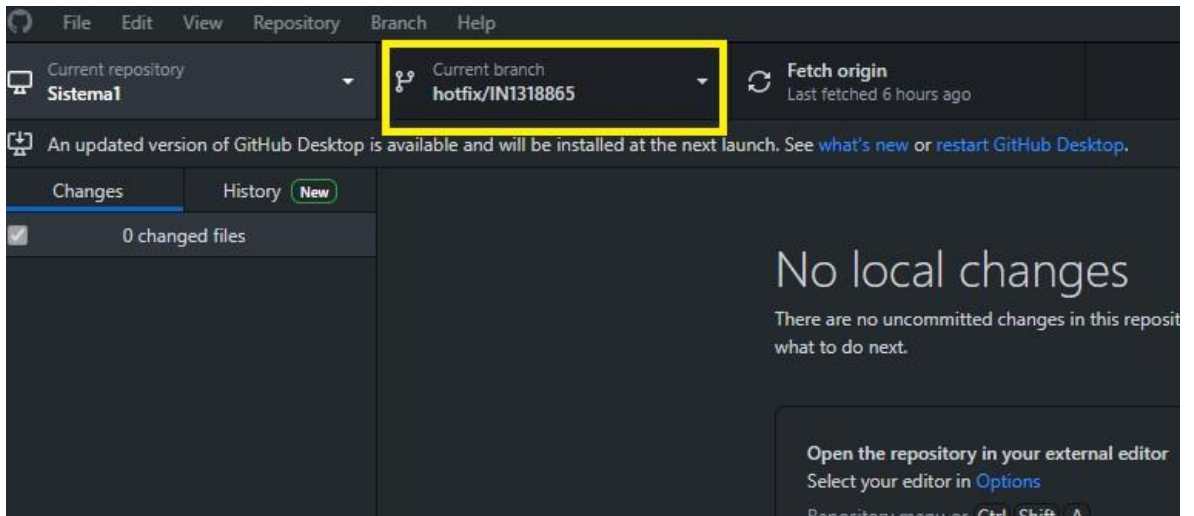
A continuación, se desplegará un pequeño menú del cual se debe seleccionar la opción “delete”:

⁶ Importante considerar, que la eliminación de la rama hotfix, se debe realizar cuando ya ha terminado la actividad, y los códigos han sido traspasados correctamente a las ramas develop y master. Todo previas validaciones y aprobaciones necesarias, según procedimiento CMPC.

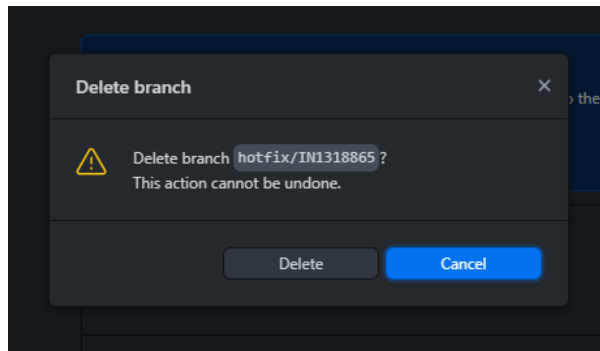
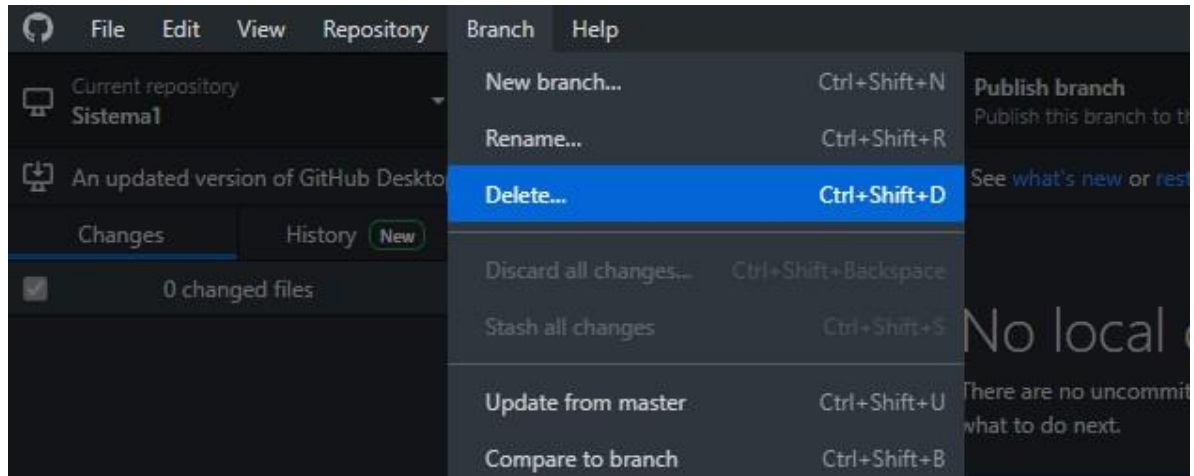


Una vez eliminada la rama en el repositorio remoto de bitbucket, debemos eliminarla de manera local, esto se puede hacer a través de la aplicación de escritorio “Github Desktop”:

Nos posicionamos en la rama a borrar en este caso la hotfix:



Y luego en el menú superior seleccionar **branch, delete**



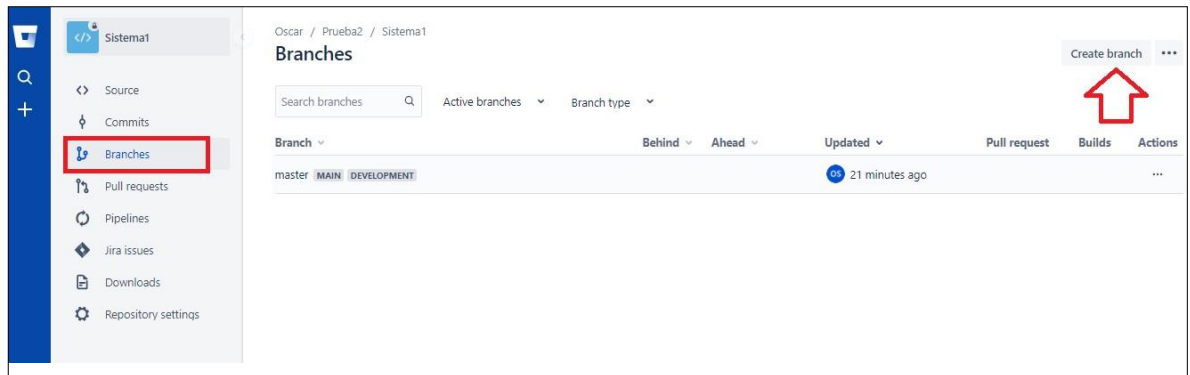
3.2.2. Procedimiento feature (requerimiento)

Crear Rama

4. **Si es un requerimiento**, se crea una rama **feature** con el prefijo en minúscula “feature” + nombre del tipo de ticket (RE) + número incidente (1318864), dando como resultado la rama: “feature/RE1318864”.

Para crear la rama feature desde bitbucket(repositorio remoto):

- 4.1. Posicionados en el repositorio del proyecto: en el menú lateral izquierdo seleccionar “Branches” y luego click al botón superior derecho de la pantalla “Create Branch”, se desplegará una ventana “pop-up”.



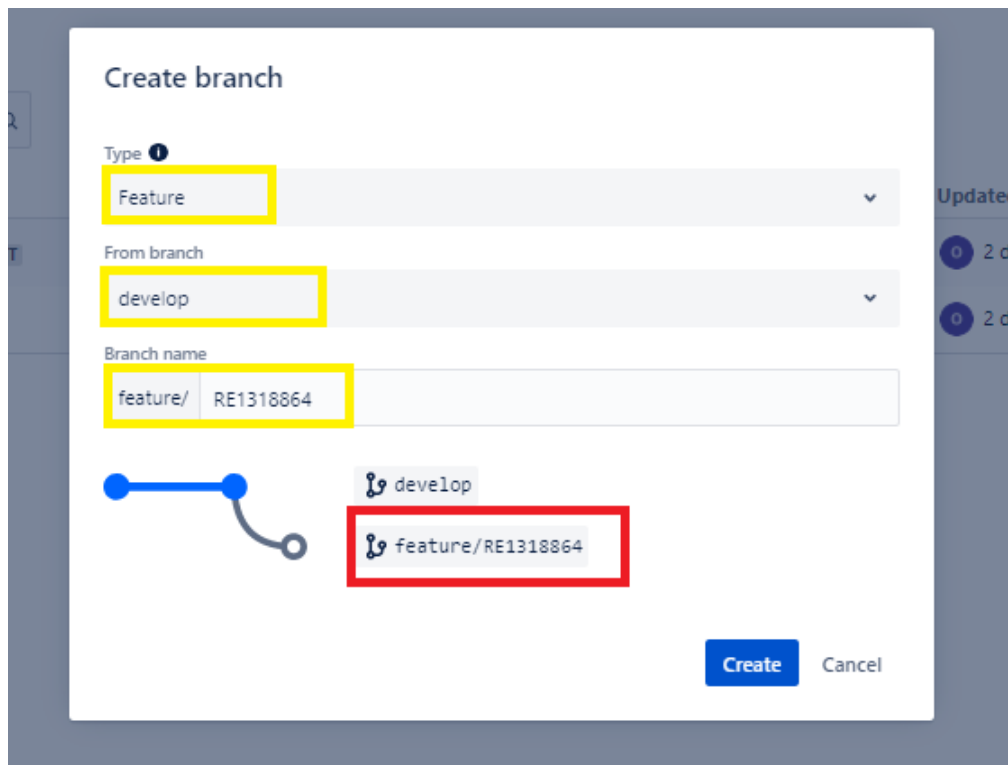
4.2. Seleccionar las opciones en ventana pop-up, según se muestra en la imagen:

Type: feature

From Branch: develop (ya que una rama feature por definición surge a partir de la develop)

Branch name: RE+<número de incidente>

Como se aprecia en la imagen, se genera de manera automática el prefijo **feature**, luego dar click en create para crear la rama.



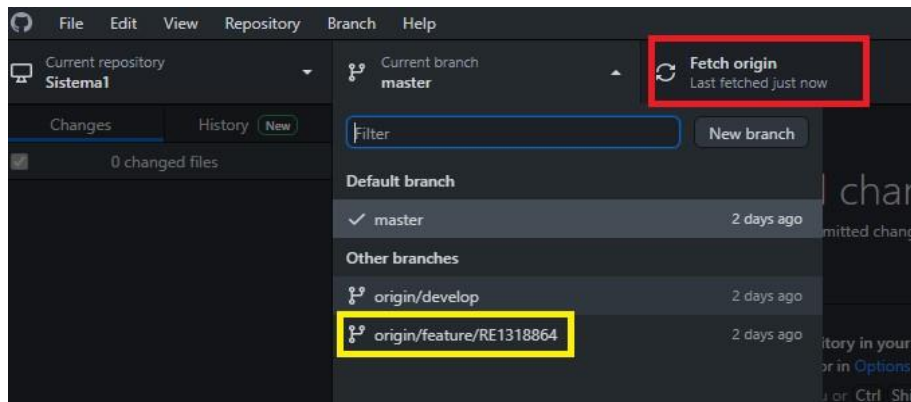
4.3. Una vez creada la rama **feature** en bitbucket, es necesario traerla a nuestro equipo (repositorio local) para trabajar sobre ella.

Esto se puede hacer a través de la aplicación “Github Desktop” (detallada en guías anteriores de subidas a bitbucket⁷).

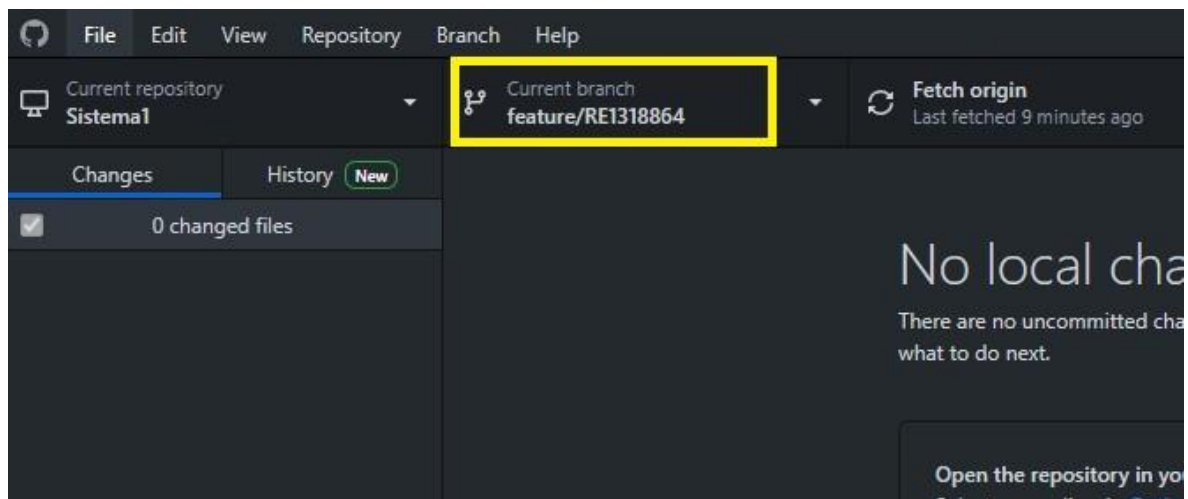
Presionar botón “Fetch origin” para traer a nuestro repositorio local los nuevos cambios desde repositorio remoto de bitbucket.

Importante: Hacer fetch regularmente para revisar si hay nuevos cambios en rama develop que deba incorporar hacia su rama local

Luego en “Current Branch” seleccionar la rama hotfix creada, tal como se muestra en la imagen:



Resultado: debe quedar seleccionada la rama feature:

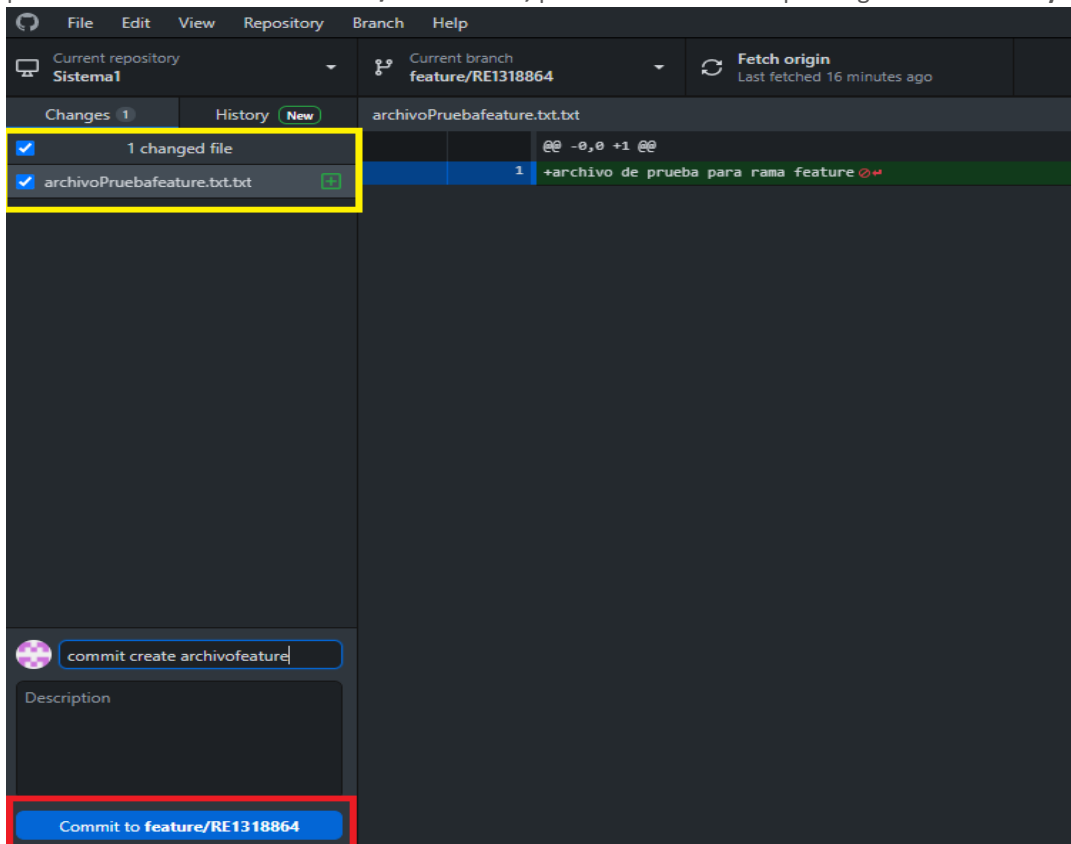


4.4. Una vez realizado los cambios de manera local, estos deben ser enviados a la rama repositorio remoto bitbucket:

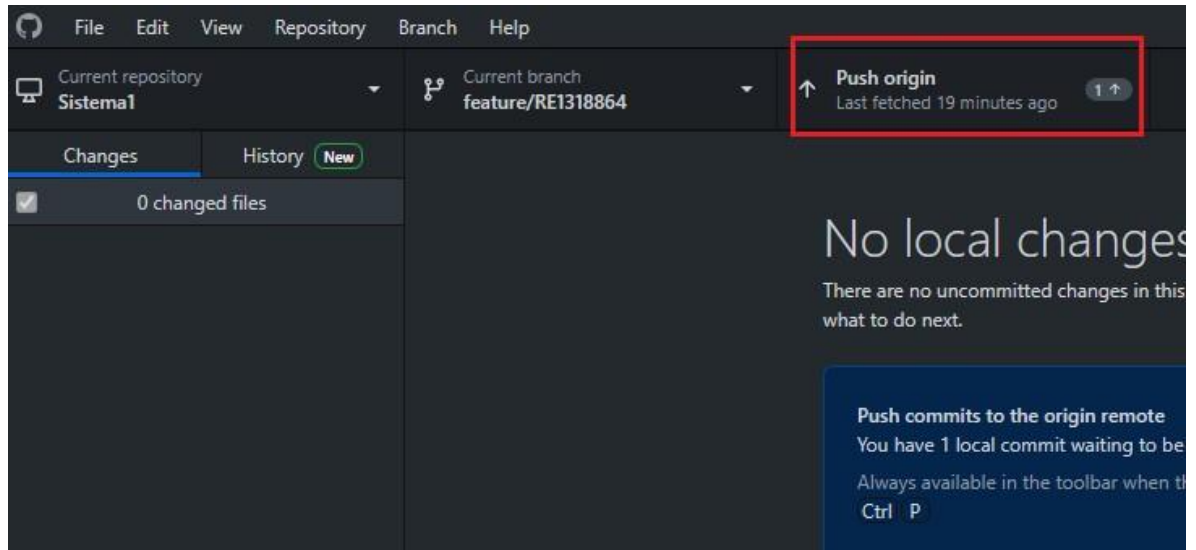
⁷ Para más detalles ir a PASO 1. Clonar Repositorio Existente en BitBucket, de la Guía práctica Bitbucket.

Para el ejemplo se creó un archivo llamado “**archivoPruebafeature.txt**”.

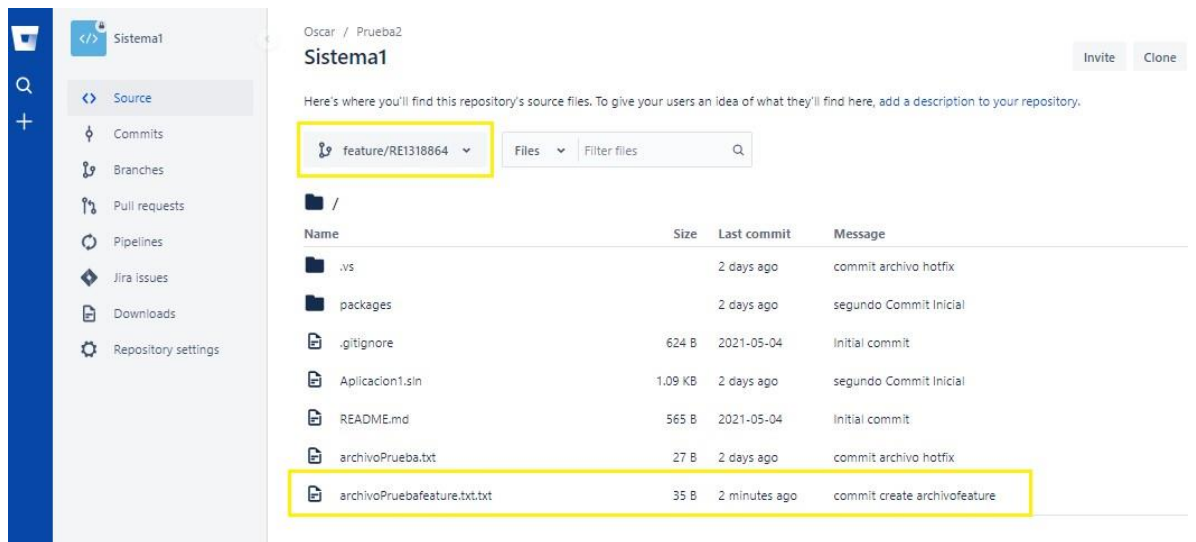
“Github Desktop” detectará los cambios en archivos de manera automática (como se aprecia en la pestaña “Changes” en la imagen), para subirlos a bitbucket es necesario presionar el botón del panel derecho “**commit to feature/RE1318864**”, previo llenado del campo obligatorio “**Summary**”:



Luego del commit anterior presionar botón “Push origin” para de esta forma enviar nuestros cambios locales al repositorio remoto.



4.5. Validando en repositorio remoto Bitbucket: Seleccionando la rama “feature/IN1318864”, vemos que se cargó el archivo “archivoPruebafeature.txt”:



Nota: Traspasar al repositorio remoto del tipo feature, se puede realizar tantas veces como el desarrollador estime necesario, se debe considerar que esta rama feature, será la mayoría de las veces del uso de sólo una persona, por lo que puede subir a repositorio cuantas veces estime necesario, en caso de que el feature, para proyectos mas grandes, sea utilizado por mas de un desarrollador, subir al repositorio deberá ser coordinado por todos quienes trabajan en el mismo feature, a fin de no entorpecer el trabajo del resto de los desarrolladores.

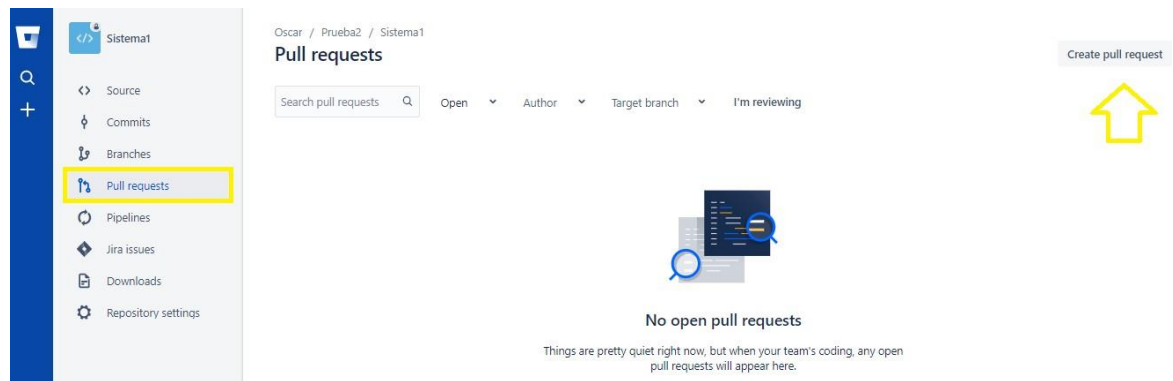
Pull request desde feature a develop

- Una vez resuelto el requerimiento, los respectivos cambios realizados y subidos a rama **feature en bitbucket**, se deben llevar hacia la rama **develop**.

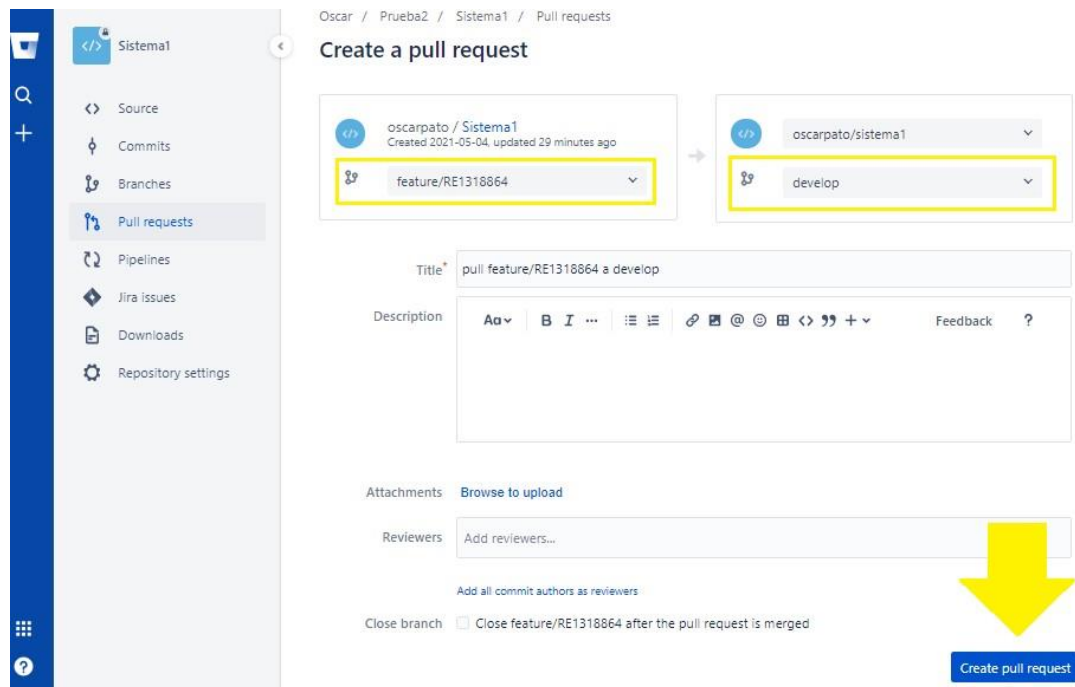
Para pasar la rama **feature** a **develop**, se debe realizar a través de un **Pull Request**.

5.1. Pull request desde rama **feature** a **Develop** (a cargo del desarrollador)

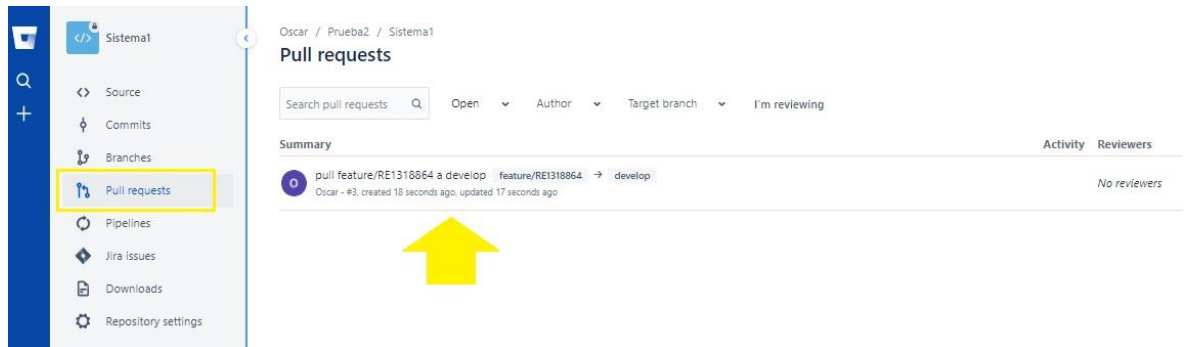
5.1.1. Nuevamente vamos al menú izquierdo y seleccionamos **“Pull requests”** y click al botón **“Create pull request”**



5.1.2. Se desplegará una ventana y seleccionamos: **feature/RE1318864** a **develop** como se muestra en la imagen, luego click a botón **“Create Pull request”**

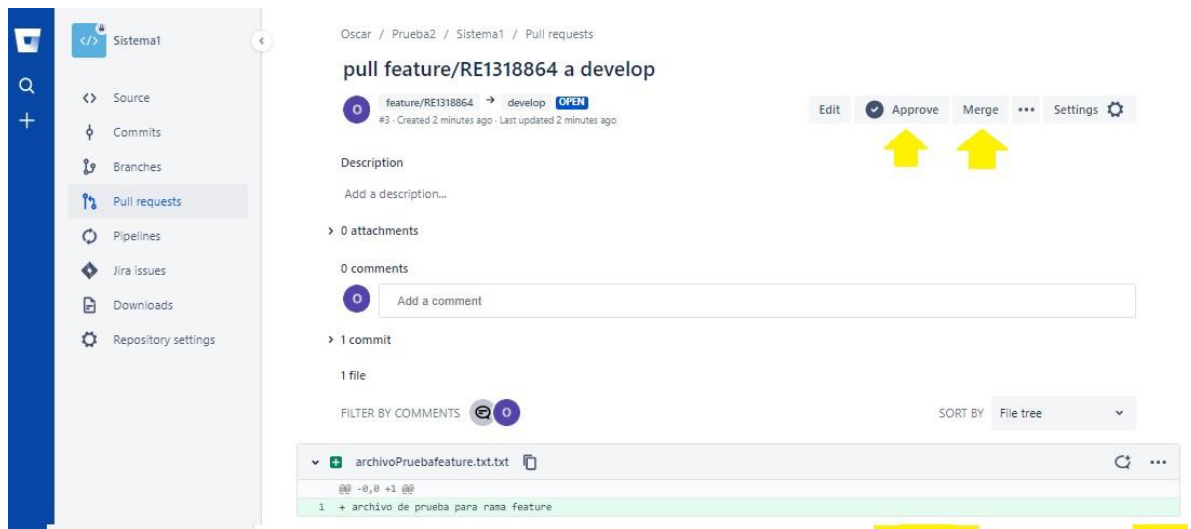


5.1.3. Se valida en menú lateral izquierdo seleccionando “Pull request”, que efectivamente se generó el pull request desde feature hacia develop (A la espera de aprobación y Merge con rama develop)

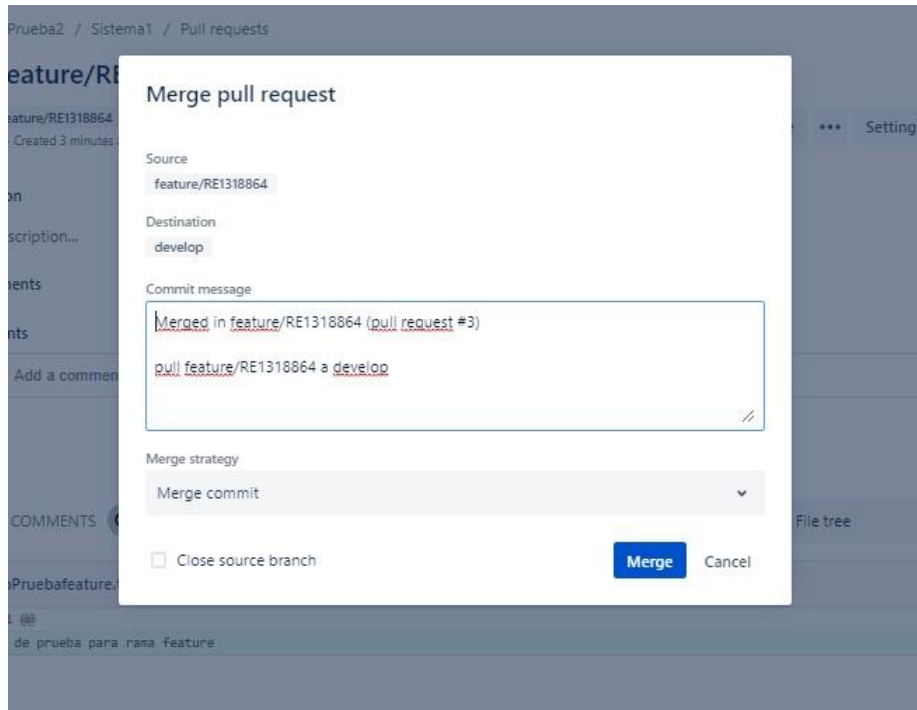


5.1.4. Merge de rama **feature** con **develop**: Ingresar al **pull request** que figura en el listado de la imagen anterior (a cargo del desarrollador)

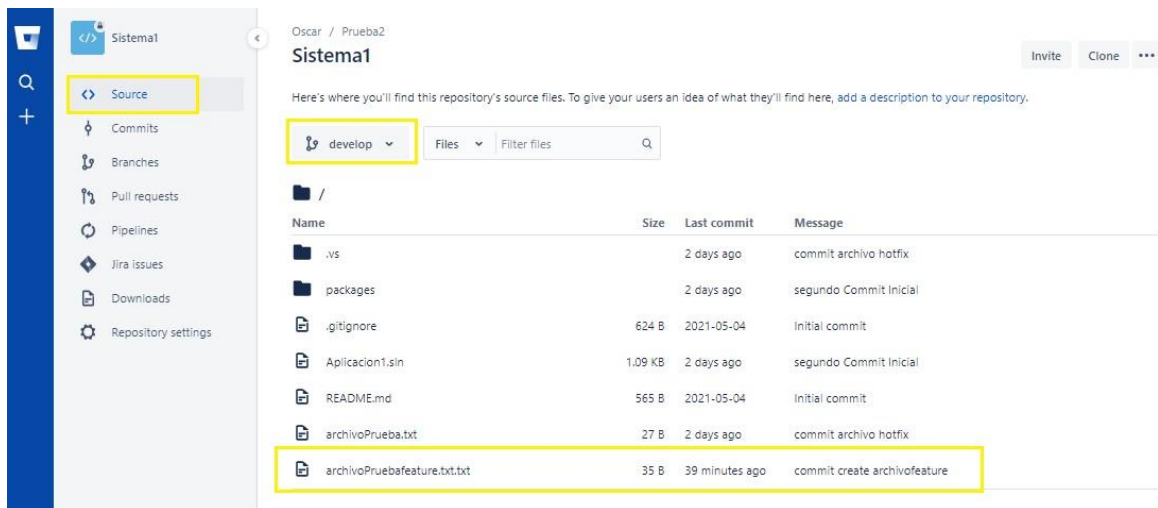
Seleccionando los botones “**Approve**” y “**Merge**” (en ese orden), se realizará el merge hacia develop, por lo tanto, los cambios realizados en la rama hotfix pasarán a la develop.



Luego de presionar “Merge”, se desplegará otra pantalla pop-up, hacer click en “Merge”:



Posteriormente se valida que rama develop contiene los cambios generados en la rama feature:



Nota: una vez traspasados los códigos a la rama develop, el requerimiento podrá ser validado y probada tanto por el usuario, por el área de QA de Gensys o por el Jefe de proyecto, si así se estima necesario.

- Si QA o el usuario, tienen observaciones a la aplicación que impliquen corregir código, esta modificación se debe realizar en los archivos fuentes de la rama feature, para una vez que estén

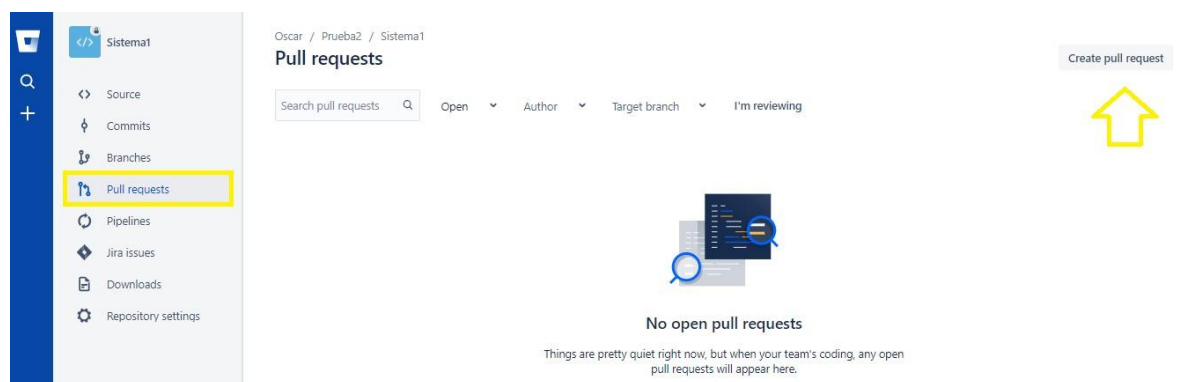
corregidos, volver a pasar los fuentes a la rama develop, tal como ya se realizó en el paso 5 de esta guía.

Pull request desde develop a master (paso a producción) - Responsabilidad del Jefe de proyecto

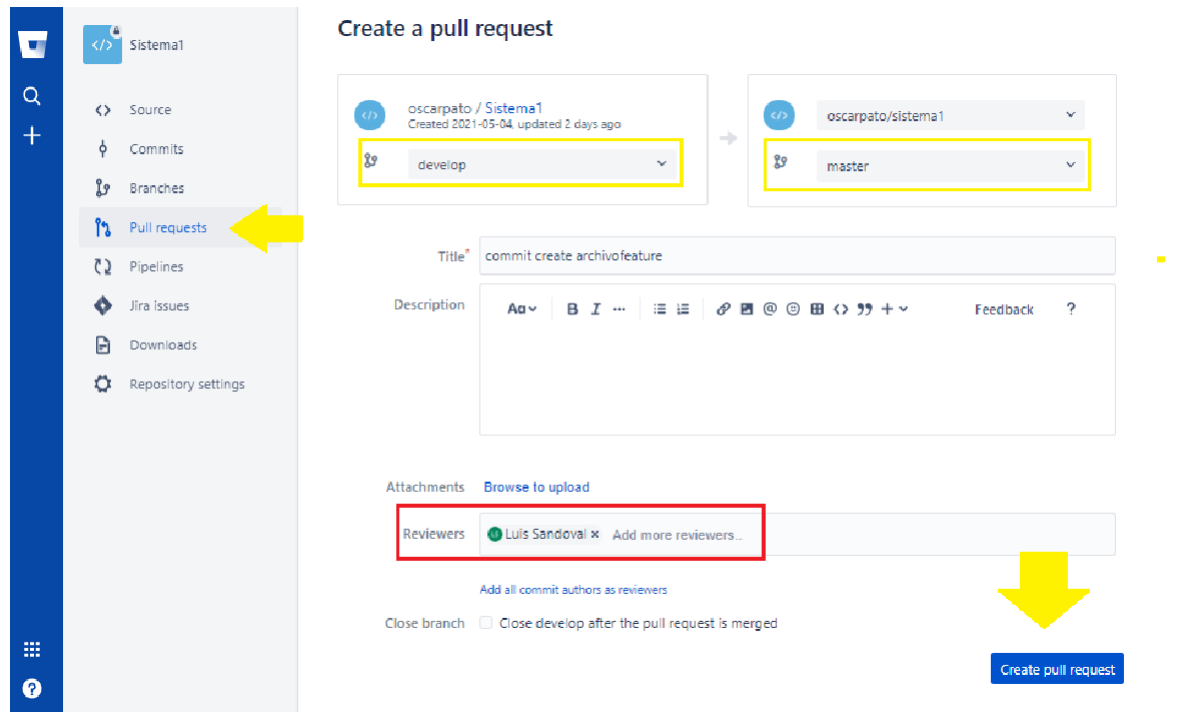
7. Si QA y el usuario aprueban la aplicación en develop para pasar a producción, con el V.ºB.º del jefe de proyecto respectivo, se deberá enviar documentación para el paso a producción (no bitbucket), luego el Jefe de Proyecto de CMPC deberá generar el Pull Request respectivo hacia la Master, y finalmente aprobar y realizar Merge hacia la master.

Pull Request develop a master

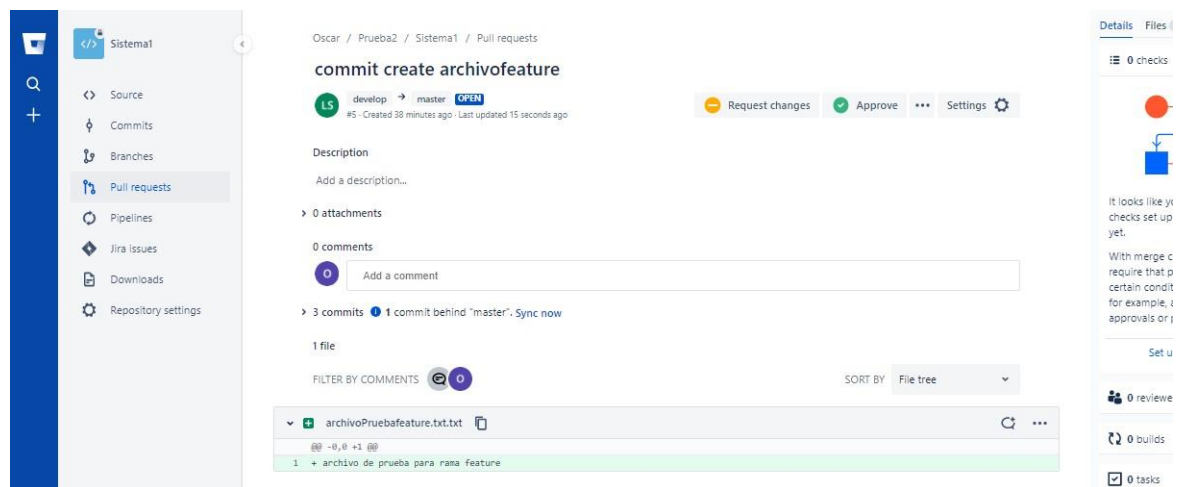
Siguiendo la lógica vista anteriormente en esta guía, en el menú lateral izquierdo seleccionar “Pull request” y luego el botón “Create pull request”:



- Posteriormente seleccionar rama develop hacia master (como se muestra en la imagen)
- No es necesario agregar un “Reviewers”, dado que antiguamente no era el Jefe de proyecto de CMPC quien hacía el pull request, sino que tan solo lo aprobaba, ahora el pull request es realizado por el Jefe de proyecto de CMPC, por lo que deberá presionar “Create pull request”.



Una vez realizado lo anterior, verán una imagen como ésta, donde el jefe de proyecto CMPC aprobará y hará el **Merge** hacia master, tal como se describe en el paso 3.2.4 de esta guía (ver sección aprobación pull request y merge a master) .



Eliminación rama feature (paso final)

8. Eliminación rama **feature**: Bajo la misma lógica del punto 3.2.5 de esta guía, por parte del desarrollador, posterior a realizadas las publicaciones y disponer de la aprobación de QA, usuario o Jefe de proyecto, según corresponda al procedimiento acordado con CMPC.