

## Flujo Detallado

### 1. Crear la Branch de Proyecto (Hotfix):

Cuando se necesita desarrollar un proyecto, se crea una branch de hotfix desde master.

- git checkout master
- git pull origin master
- git checkout -b hotfix/nombre-del-proyecto
- git add .
- git commit -m 'comentario con estructura'
- git push

### 2. Desarrollar y Probar en Branch de Desarrollo (Develop):

Realiza los cambios necesarios en la branch de hotfix y realiza commits. Fusiona la branch de hotfix con develop para pruebas preliminares.

- git checkout develop
- git pull origin develop
- git merge hotfix/nombre-del-proyecto
- git push origin develop

### 3. Probar en Branch de QA:

Después de las pruebas preliminares, fusiona la branch de hotfix con qa para pruebas de calidad.

- git checkout qa
- git pull origin qa
- git merge hotfix/nombre-del-proyecto
- git push origin qa

### 4. Validar en Branch de Pre-Producción:

Una vez que las pruebas de calidad son satisfactorias, fusiona la branch de hotfix con pre-production para pruebas finales.

- git checkout pre-production
- git pull origin pre-production
- git merge hotfix/nombre-del-proyecto
- git push origin pre-production

### 5. Fusionar con Master:

Una vez que el proyecto ha sido validado en pre, se fusiona con master.

- git checkout master
- git pull origin master
- git merge hotfix/nombre-del-proyecto
- git push origin master

### 6. Sincronizar Pre, QA y Develop con Master:

- git checkout pre-production

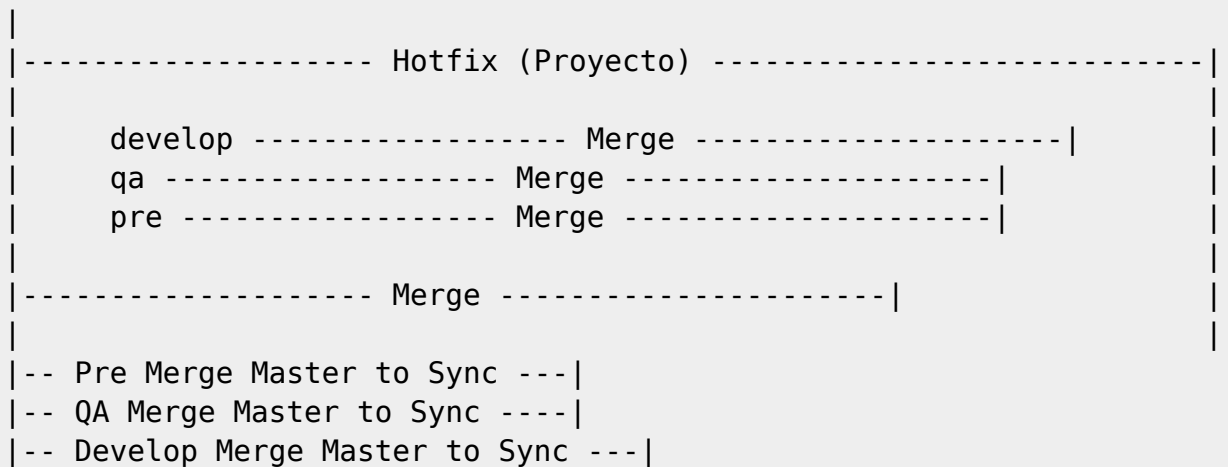
- git pull origin pre-production
- git merge master
- git push origin pre-production
  
- git checkout qa
- git pull origin qa
- git merge master
- git push origin qa
  
- git checkout develop
- git pull origin develop
- git merge master
- git push origin develop

## 7. Eliminar la Branch de Proyecto:

Después de que el proyecto ha sido desplegado a producción y validado, puedes eliminar la branch de hotfix.

- git branch -d hotfix/nombre-del-proyecto
- git push origin --delete hotfix/nombre-del-proyecto

master



## Beneficios de Este Enfoque

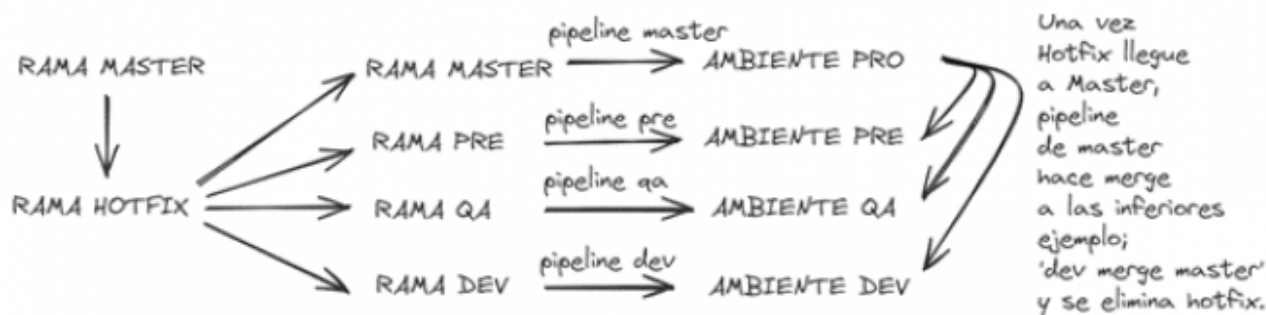
**Multinivel de Pruebas:** Permite realizar pruebas en diferentes niveles (develop, qa, pre) antes de desplegar a producción.

**Consistencia:** Asegura que master, pre, qa y develop estén siempre sincronizadas. **Flexibilidad y Control:** Proporciona un control granular sobre el flujo de trabajo y las pruebas en diferentes etapas.

Este flujo de trabajo proporciona un enfoque estructurado y robusto para gestionar proyectos urgentes, garantizando que los cambios sean probados exhaustivamente en diferentes entornos antes de ser desplegados a producción.

Este enfoque está basado en Git Flow y GitLab Flow, también se analizó GitHub Flow. Se puede

consultar más información para comparar las tres en el siguiente artículo: Choosing the Right Git Branching Strategy: A Comparative Analysis.



### Buenas prácticas en desarrollo cuando 2 o más usuarios están tocando el mismo proyecto al mismo tiempo.

Cuando un desarrollador esté trabajando en un Hotfix dentro de un proyecto, antes de mandar su Hotfix a mezclar con cualquier rama (DEV, QA, PRE, PRO), debe subir a master y hacer git pull por si otro desarrollador envió cambios. Se recomienda antes de enviar el git pull en master, de ser necesario, hacer un git stash para limpiar cambios y finalmente cuando finalice, volver a la rama Hotfix y hacer git merge master. Si sale un mensaje de diferencia, escribir en consola :wq

. Si requiere intervención por conflictos, usar Visual Studio o el IDE de su preferencia para solucionarlos.

### Creación de primera rama Hotfix por un desarrollador

- git pull origin master # Pull para actualizar con remoto
- git checkout -b hotfix/primerproyecto # Se crea hotfix
- git add . # Se agregan cambios
- git commit -m 'comentario con estructura' # Se agrega comentarios
- git push # Se mandan cambios al remoto primera vez
- git checkout master # Se pasa a la rama master
- git stash # Limpio cambios locales de ser necesario
- git pull # Pull para actualizar con remoto
- git checkout -b hotfix/primerproyecto # Se pasa al hotfix
- git merge master # Se mezcla hotfix con master
- :wq # Si consola muestra algún mensaje usar :wq
- git checkout master # Se pasa a master
- git pull # Se actualiza de ser necesario usar stash
- git merge -b hotfix/primerproyecto # Y ya se puede hacer la mezcla del hotfix

### Creación de segunda rama Hotfix por segundo desarrollador

- git pull origin master # Pull para actualizar con remoto
- git checkout -b hotfix/segundoproyecto # Se crea hotfix
- git add . # Se agregan cambios
- git commit -m 'comentario con estructura' # Se agrega comentarios
- git push # Se mandan cambios al remoto primera vez

- git checkout master # Se pasa a la rama master
- git stash # Limpio cambios locales de ser necesario
- git pull # Pull para actualizar con remoto
- git checkout -b hotfix/segundoproyecto # Se pasa al hotfix
- git merge master # Se mezcla hotfix con master
- :wq # Si consola muestra algún mensaje usar :wq
- git checkout master # Se pasa a master
- git pull # Se actualiza de ser necesario usar stash
- git merge -b hotfix/segundoproyecto # Y ya se puede hacer la mezcla del hotfix

[←Regresar](#)

From:  
<http://wiki.adacsc.co/> - Wiki

Permanent link:  
<http://wiki.adacsc.co/doku.php?id=ada:howto:sicoferp:factory:new-migracion-sicoferp:front:flujogit&rev=1718986633>

Last update: **2024/06/21 16:17**

