# Estrategia de Branching: GitHub Flow + Backup Persistente

Este documento formaliza la estrategia de branching basada en GitHub Flow, complementada con una rama de respaldo persistente (backup) que garantiza la disponibilidad de una copia exacta del último estado estable del sistema. Esta combinación ofrece agilidad en el desarrollo y seguridad ante posibles fallos en despliegue.

## 🎯 Objetivo

Mantener un flujo de trabajo ágil y simple (GitHub Flow), asegurando al mismo tiempo un punto de restauración constante mediante una rama backup que se actualiza con cada despliegue exitoso.

## ⚙️ Flujo de Trabajo Detallado

1. 1. Partir desde la rama principal (main), que refleja el estado actual en producción.
2. 2. Crear una rama de funcionalidad (feature/nueva-funcionalidad) para desarrollar cambios o nuevas características.
3. 3. Desarrollar, probar localmente y subir los cambios al repositorio remoto.
4. 4. Abrir un Pull Request (PR) hacia main para revisión y validación de CI/CD.
5. 5. Realizar el merge de la rama feature a main una vez aprobado y validado.
6. 6. El pipeline CI/CD despliega automáticamente los cambios en producción.
7. 7. Confirmar que el despliegue fue exitoso y la aplicación funciona correctamente.
8. 8. Actualizar la rama backup para reflejar el nuevo estado estable del main.
9. 9. En caso de fallo en despliegue, restaurar main desde backup y realizar la corrección en una nueva rama hotfix.

## 🔁 Actualización de la Rama Backup

Una vez confirmado que el despliegue ha sido exitoso, la rama backup debe sincronizarse con main. Esto garantiza que backup siempre refleje el último estado estable en producción.

Comandos sugeridos:

git checkout main

git pull origin main

git checkout backup

git merge main

git push origin backup

## 🚨 Restauración ante Fallos en Despliegue

Si el despliegue falla o introduce errores críticos, no se debe actualizar la rama backup. En su lugar, se debe restaurar main al estado estable almacenado en backup mediante los siguientes pasos:

git checkout main

git reset --hard origin/backup

git push origin main --force

## ✅ Beneficios de la Estrategia

• Combina simplicidad y seguridad.

• Permite un flujo ágil con ramas cortas por funcionalidad.

• Garantiza un punto de restauración estable y actualizado.

• Facilita la recuperación rápida ante fallos en producción.

• Evita revertir múltiples commits o merges complejos.

## 💡 Recomendaciones

• Proteger la rama backup en el repositorio para evitar commits directos.

• Actualizar la rama backup únicamente tras un despliegue exitoso.

• Nombrar las ramas feature de forma descriptiva (feature/login, feature/ajuste-ui).

• Configurar CI/CD para que los despliegues automáticos ocurran desde main.

• Agregar una rama staging opcional si se desea probar antes de producción.