IT Bootcamp

Branches (Ramas)

DigitalHouse>

Índice

1. Branches
2. Conflictos trabajando con Branches

1 | Branches

¿Qué es una Rama/Branch?

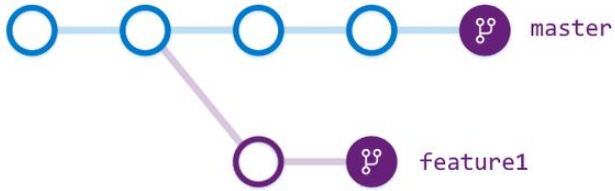
Una rama en Git es un "espacio" dentro del repositorio donde se almacenan nuestros archivos.

Por defecto en Git, la rama principal se llama **main**.

¿Para qué se usan las Branches?

En Git las Ramas son espacios o entornos independientes para que podamos trabajar sobre un mismo proyecto en forma más independiente, sin cometer errores o borrar el conjunto de archivos originales del proyecto.

Esto permite dar flexibilidad para desarrollar el proyecto de manera más organizada, donde solo se incorpora una rama a la rama principal cuando se está seguro que lo desarrollado funciona correctamente y no rompe lo que ya se tiene en la misma.



Supongamos que tenemos una receta...

- Como no nos acordamos bien los pasos de la misma, comenzamos por ir probando distintos ingredientes. Por cada ingrediente que agreguemos vamos a crear una rama.

- En caso de que nos salga bien seguimos avanzando, y en caso de que nos equivoquemos, podemos volver a la rama anterior y seguir trabajando sobre esa.

- Una vez que probamos y encontramos la receta final, unificamos esa rama a nuestra “principal” (main o master).

Comandos para Branches

Conocer en qué rama estamos:

```
>_ git branch
```

Crear una nueva rama:

```
>_ git branch nombreBranch
```

Comandos para Branches

Cambiar de rama:

```
>_  git checkout nombreBranch
```

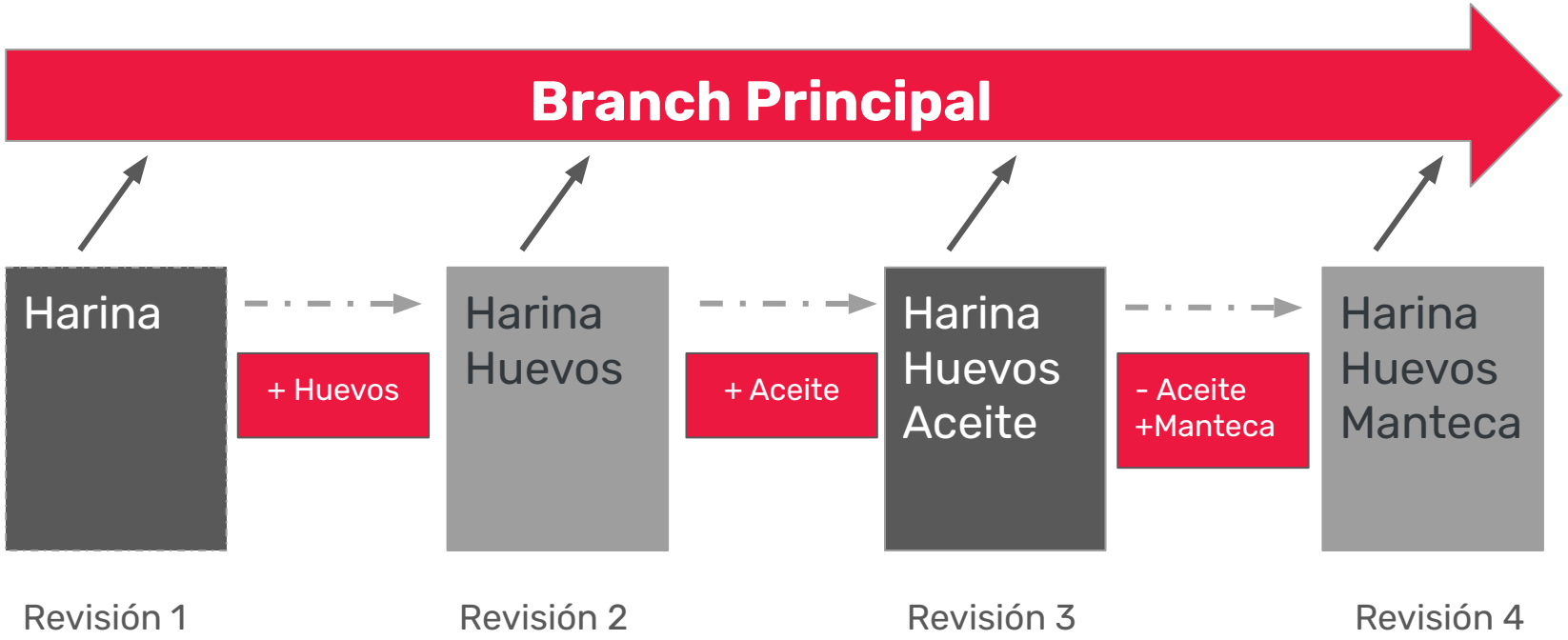
Eliminar una rama:

```
>_  git branch -d nombreBranch
```

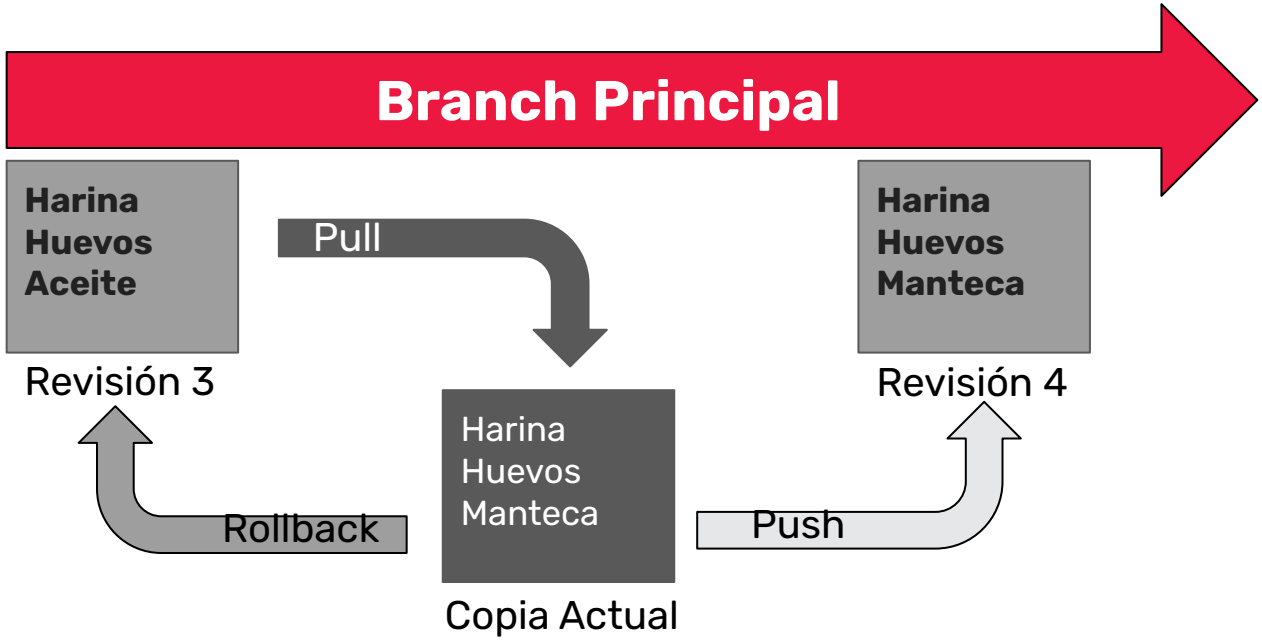
Copiar una rama:

```
>_  git branch -c nombreBranch nombreCopia
```

Flujo de trabajo



Flujo de trabajo



2 Conflictos trabajando con branches

¿Qué es un Conflicto?

Sea que usemos la consola de comandos o un GUI, el error más frecuente es cuando dos o más usuarios tocan el mismo archivo en la misma línea de código.

Para evitar esto, la recomendación más eficiente es dividir bien el trabajo y no tocar al mismo tiempo con varias personas el mismo archivo.

Resolviendo Conflictos

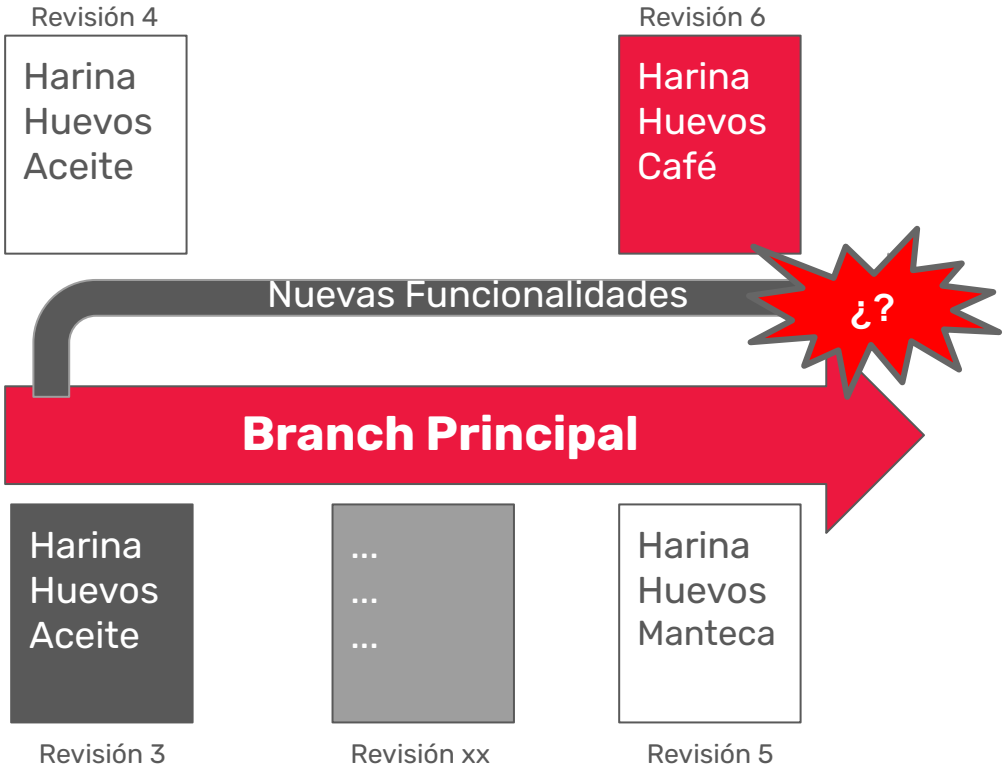
Cuando estamos trabajando con varios colaboradores es posible que tengamos conflictos cuando actualicemos o bajemos los archivos que tengamos en el repositorio remoto a nuestro repositorio local. Esto ocurre por lo general cuando se tocan uno o varios archivos en diferentes commits.

- Git a menudo puede resolver diferencias entre ramas y fusionarlas automáticamente. Por lo general, los cambios se realizan en diferentes líneas, o incluso en diferentes archivos, lo que hace que la combinación sea fácil de entender para Git.
- A veces hay cambios en competencia que Git no puede resolver sin nuestra ayuda.

Conflictos


Revisión 4
Harina
Huevos
Aceite
Revisión 6
Harina
Huevos
Nuevas Funcionalidades
Branch Principal
Harina
Huevos
Aceite
Revisión 3
Harina
Huevos
Manteca
Revisión 5

Generando Conflictos



Merge

Si el conflicto es causado por cambios en la misma línea del mismo archivo los mismos se pueden resolver utilizando el editor de código y decidiendo cuáles cambios se quedan y cuales no. Por ejemplo tenemos conflictos en el archivo.txt

```
> git status
  # On branch branch-b
  # You have unmerged paths.
  #   (fix conflicts and run "git commit")
  #
  # Unmerged paths:
  #   (use "git add ..." to mark resolution)
  #
  # both modified:      archivo.txt
  #
  no changes added to commit (use "git add" and/or
"git commit -a")
```

Resolviendo el conflicto

Debemos en el editor de código buscar el marcador de conflicto <<<<<<<

Los cambios de la rama HEAD o base aparecerán después de la línea <<<<<<< HEAD. A continuación, veremos =======, que divide nuestros cambios de los cambios en la otra rama, seguido de >>>>>>> BRANCH-NAME. Lo que resta es decidir qué se queda y qué debemos sacar.

Luego agregar los cambios al stage `git add .`, los confirmamos `git commit -m “solucionando conflictos”` y por último los subimos `git push origin main`

{}

```
Este es un ejemplo de conflictos
en un archivo
<<<<<<< HEAD
cambios desde otro usuario
=======
mis cambios
>>>>>>> branch-a
```

Resolviendo el conflicto

Debemos abrir el editor de código y buscar el marcador de conflicto <<<<<<<

Los cambios de la rama HEAD o base aparecerán después de la línea <<<<<<< HEAD. A continuación, veremos =======, que divide nuestros cambios de los cambios en la otra rama, seguido de >>>>>>> BRANCH-NAME.

{}

```
Este es un ejemplo de conflictos en un archivo
<<<<<<< HEAD
cambios desde otro usuario
=======
otros cambios desde la rama ‘a’
>>>>>>> branch-a
```

DigitalHouse>