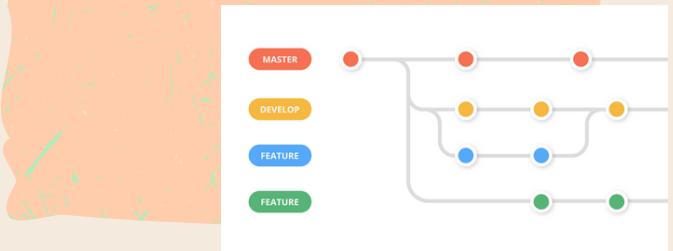




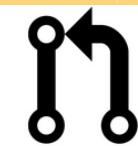
## Branches

Para añadir una nueva función o solucionar algún error, generamos una nueva rama para alojar estos cambios. Esto hace sea más complicado que el código inestable se fusione con el código base principal, y nos da la oportunidad de limpiar el historial futuro antes de fusionarlo con la rama principal. Para generar una nueva rama usamos `git branch [branch-name]`.



## PULL REQUEST

Crea una solicitud de incorporación de cambios para proponer cambios en un repositorio y colaborar con ellos. Estos cambios se proponen en una rama, lo cual garantiza que la rama predeterminada contenga únicamente trabajo finalizado y aprobado.



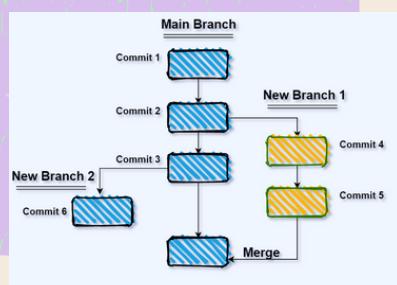
## Fork

Fork en nuestro idioma significa "bifurcar". Lo que hacemos con fork es crear una copia del repositorio original, se crea un repositorio nuevo en nuestra cuenta y entonces lo que hacemos es trabajar sobre esa copia. Todos los cambios que hagamos serán sobre nuestra copia del repositorio original y no en el repositorio original.



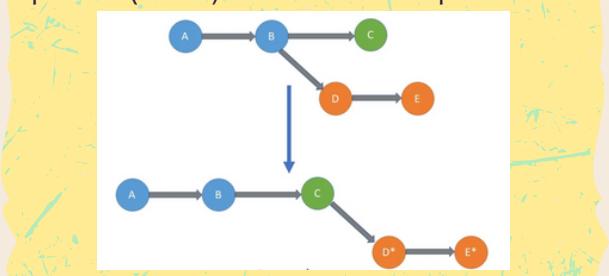
## MERGE

La fusión es la forma que tiene Git de volver a unir un historial bifurcado. El comando `git merge` permite tomar las líneas independientes de desarrollo creadas por `git branch` e integrarlas en una sola rama. Esto lo hace sin eliminar el historial de la rama secundaria.



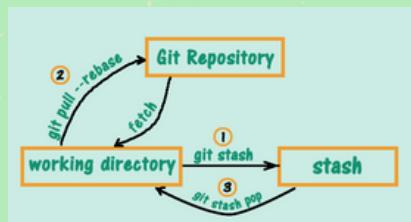
## Rebase

No mantiene el historial de la rama secundaria, sino que "re-escribe" la historia de la rama principal integrando los commits de la rama secundaria en la rama principal, no crea un commit de unión adicional, sino que cambia el puntero (HEAD) al último commit que ubica.



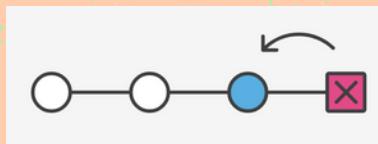
## Stash

Recoge los cambios sin confirmar (tanto los que están preparados como los que no), los guarda aparte para usarlos más adelante y, acto seguido, los deshace en el código en el que estás trabajando. De esta forma podemos trabajar en alguna otra tarea en el código en el que estamos y más adelante regresar a lo que estábamos con el stash.

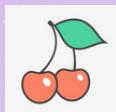


## Clean

Es un comando para deshacer archivos sin seguimiento, es decir, aquellos que se han creado en el directorio de trabajo del repositorio, pero que no se han añadido al índice de seguimiento del repositorio con `git add`.



## Cherry pick



Es el acto de elegir una confirmación de una rama y aplicarla a otra, ejemplo:

Rama main antes de aplicar cherry-pick:

```
a - b - c - d  Main  
 \  
 e - f - g Feature
```

Rama main después de aplicar cherry-pick indicando que recoja la confirmación f:

```
a - b - c - d - f  Main  
 \  
 e - f - g Feature
```

## Conflicts

Las herramientas vistas son algunas de toda la gama que nos ofrece git para el tratamiento de conflictos al momento de trabajar de manera colaborativa en un repositorio en el que existen diversos cambios constantes. La creación de ramas para trabajar en diferentes actualizaciones o tratar errores es una de las formas de manejar los conflictos. Los pasos base a seguir son:

1. Crear una rama basada en otra (usualmente la main como partida).
2. Trabajar en la nueva rama
3. Guardar los cambios
4. Ubicarse en la rama que queremos unir los cambios
5. Fusionar los cambios

O lo que enlenguaje git sería:

1. `git checkout -b feature`
2. Realizamos los cambios pertinentes
3. `git commit -m "feature realizada"`
4. `git checkout master`
5. `git merge feature`



## REST

Es una interfaz que sirve para conectar varios sistemas basados en el protocolo HTTP. Sirve para obtener y generar datos y operaciones, devolviendo esos datos en formatos muy específicos, como XML y JSON.

Elaborado por: Angeles Mena

## GET

Es el verbo que usamos para hacer consultas de recursos. No debe causar efectos secundarios en el servidor, no deben producir nuevos registros, ni modificar los ya existentes.

No importa cuantas veces ejecutemos el GET, los resultados que obtendremos serán los mismos.

## POST

Con POST creamos recursos nuevos. Cada llamada con POST debería producir un nuevo recurso.

Algunos escenarios más complejos para el uso de POST son los inicios de sesión, agregar a un carrito de compras, procesar un pago nuevo, etc

## PATCH

También lo utilizamos cuando queremos actualizar un recurso, pero a diferencia de PUT, PATCH actualiza algunos elementos del recurso mismo, sin sustituirlo por completo..

## PUT

Con esta petición lo que hacemos es actualizar el recurso. Indica que vamos a sustituir por completo un recurso.

## DELETE

Con este verbo eliminamos recursos. Puede ser uno solo o un conjunto de ellos, ejemplo:

Eliminar un recurso individual como en:  
DELETE /cursos/backend-profesional

O para eliminar una colección completa:  
DELETE /cursos