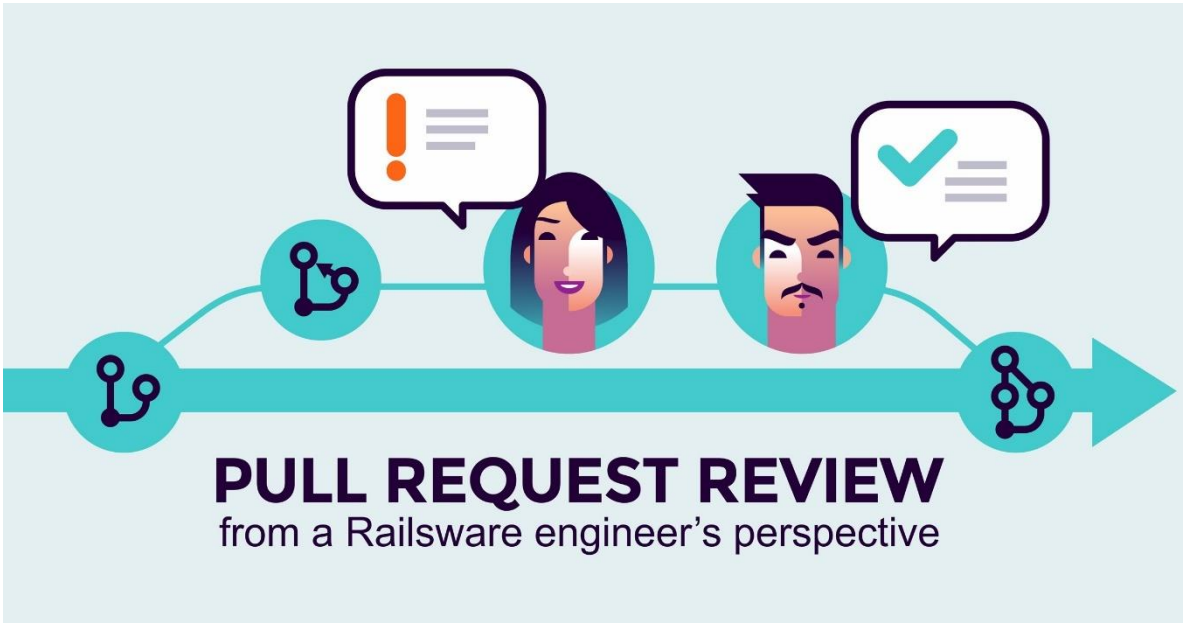
# Git Pull Request

Un pull request es una petición que el propietario de un fork de un repositorio hace al propietario del repositorio original para que este último incorpore los commits que están en el fork.




**Comandos:**

**$ git checkout master**

**Switched to branch 'master'**

**$ git checkout -b mycommits**

**Switched to a new branch 'mycommits'**

**$ git checkout mycommits**

**$ git push -u origin mycommits**

**Total 0 (delta 0), reused 0 (delta 0)**

**To git@github-aprendegit-userjess:aprendegit-userjess/fork**

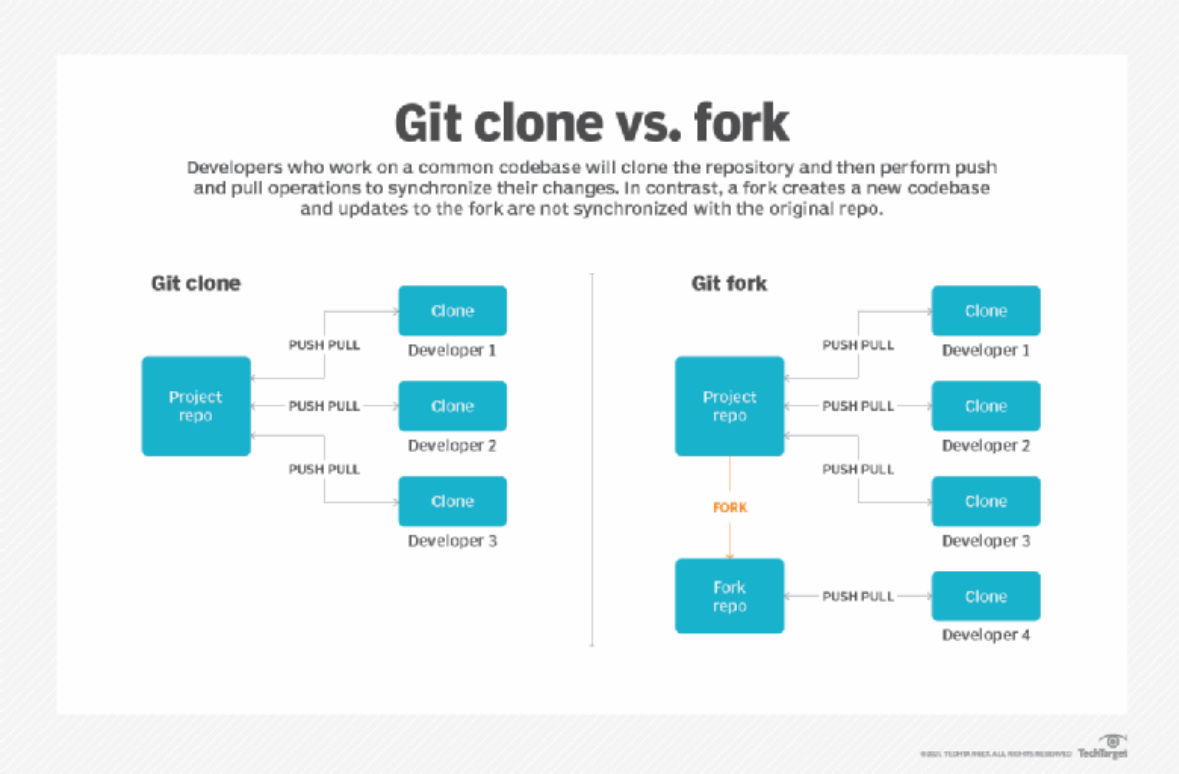
*** [new branch] mycommits -> mycommits**

**Branch mycommits set up to track remote branch mycommits from origin.**

Cuando aprendegit-userjess envía el pull request, Github envía al propietario un email avisándole de que tiene un pull request listo para revisar.

# Git Fork

Cuando hacemos un fork de un repositorio, se hace una copia exacta en crudo (en inglés «bare») del repositorio original que podemos utilizar como un repositorio git cualquiera. Después de hacer fork tendremos dos repositorios git idénticos, pero con distinta URL. Justo después de hacer el fork, estos dos repositorios tienen exactamente la misma historia, son una copia idéntica. Finalizado el proceso, tendremos dos repositorios independientes que pueden cada uno evolucionar de forma totalmente autónoma. De hecho, los cambios que se hacen el repositorio original NO se transmiten automáticamente a la copia (fork). Esto tampoco ocurre a la inversa: las modificaciones que se hagan en la copia (fork) NO se transmiten automáticamente al repositorio original.




Comandos:

```
$ gh repo fork https://github.com/learn-devops-fast/spock-lizard-docker.git --cl
one
✓ Created fork cameronmcnz/spock-lizard-docker
Cloning into 'spock-lizard-docker'...
remote: Enumerating objects: 628, done.
remote: Counting objects: 100% (177/177), done.
remote: Compressing objects: 100% (133/133), done.
remote: Total 628 (delta 67), reused 71 (delta 10), pack-reused 451
Receiving objects: 100% (628/628), 35.28 MiB | 10.57 MiB/s, done.
Resolving deltas: 100% (191/191), done.
```

# Git Rebase

Realizar un rebase a una rama (branch) en Git es una forma de mover la totalidad de una rama a otro punto del árbol. Ejemplo:

Imagen antes del rebase:

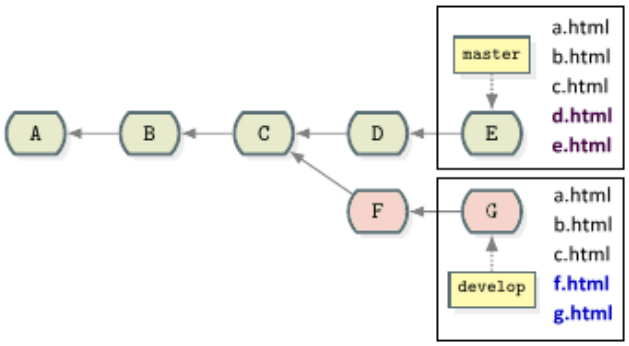
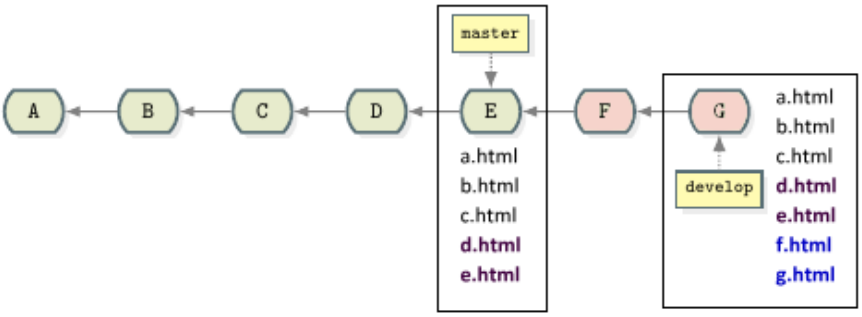



Imagen después de aplicar un rebas:




**Comandos:**

**git fetch origin main**

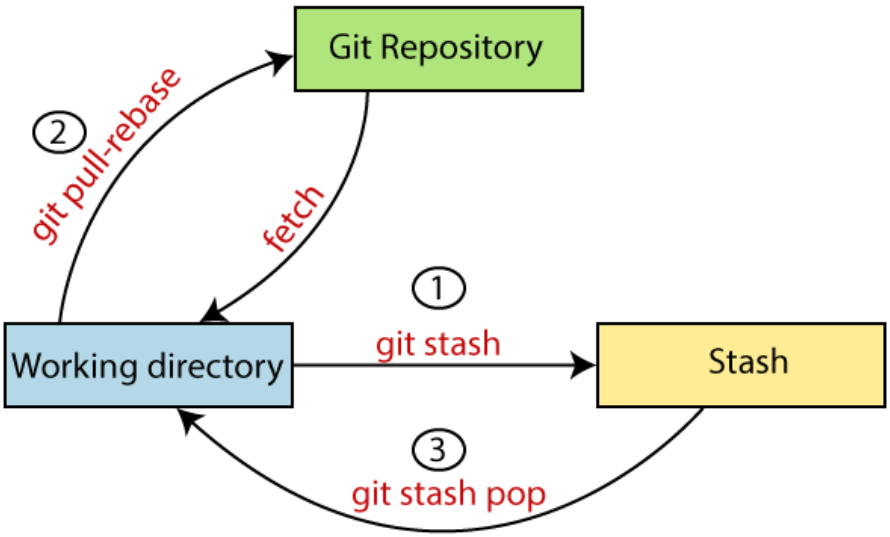
**git checkout my-feature-branch**

**git rebase origin/main**

# Git Stash

Git tiene un área llamada "stash" donde puedes almacenar temporalmente una captura de tus cambios sin enviarlos al repositorio. Está separada del directorio de trabajo (working directory), del área de preparación (staging area), o del repositorio.

Esta funcionalidad es útil cuando has hecho cambios en una rama que no estás listo para realizarle commit, pero necesitas cambiar a otra rama.




Para guardar tus cambios en el stash, ejecuta el comando:

**git stash save "mensaje opcional para ti"**

Esto guarda los cambios y revierte el directorio de trabajo a como se veía en tu último commit. Los cambios guardados están disponibles en cualquier rama de ese repositorio.

Ten en cuenta que los cambios que quieras guardar deben estar en los archivos rastreados. Si has creado un nuevo archivo e intentas guardar tus cambios, puede que obtengas el error No local changes to save (No hay cambios locales que guardar).

Para ver lo que hay en tu stash, ejecuta el comando:

**git stash list**

Esto devuelve una lista de tus capturas guardadas en el formato stash@{0}: RAMA-STASHED-CAMBIOS-SON-PARA: MESSAGE. La parte de stash@{0} es el nombre del stash, y el número en las llaves ({}) es el índice (index) del stash. Si tienes múltiples conjuntos de cambios guardados en stash, cada uno tendrá un índice diferente.

Si olvidaste los cambios que hiciste en el stash, puedes ver un resumen de ellos con el comando **git stash show NOMBRE-DEL-STASH**. Si quieres ver el típico diseño "diff" (con los + y - en las líneas con los cambios), puedes incluir la opción -p (para parche o patch).

Recuperar Cambios en Stash

Para recuperar los cambios del stash y aplicarlos a la rama actual en la que estás, hay dos opciones:

**git stash apply NOMBRE-DEL-STASH** aplica los cambios y deja una copia en el stash

**git stash pop NOMBRE-DEL-STASH** aplica los cambios y elimina los archivos del stash

Para borrar los cambios guardados en Stash se ejecuta el comando:

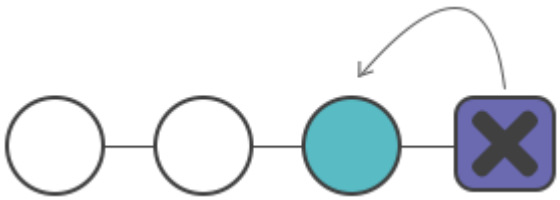
**git stash drop NOMBRE-DEL-STASH**

Para limpiar todo del stash, se ejecuta el comando:

**git stash clear**

# Git Clean

El comando Git Clean se usa para limpiar los archivos sin seguimiento en el repositorio. Si queremos eliminar los archivos no deseados, podemos usar el comando de limpieza en Git.



**Comandos:**

**git clean -n:** para ejecutar en seco.

**git clean -f:** eliminación forzada de archivos.

**git clean -f -x:** eliminar archivos .gitignore

**git clean -f -d:** elimina los directorios sin seguimiento.

Ejemplo:

**git clean -f**

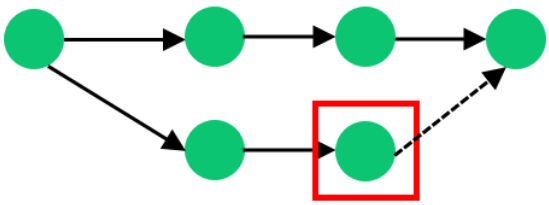
#Removing test_untracked_file

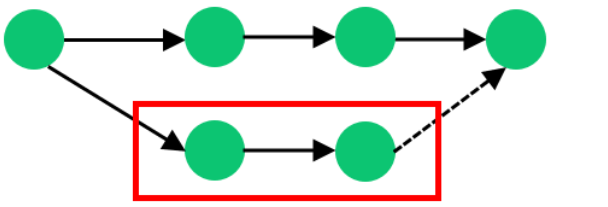
Por defecto, no eliminará:

- Los archivos .gitignore
- Nuevos directorios creados recientemente
- Archivos de índice.
- Archivos de confirmación existentes

# Git Cherry-pick

git cherry-pick es un potente comando que permite que las confirmaciones arbitrarias de Git se elijan por referencia y se añadan al actual HEAD de trabajo. La ejecución de cherry-pick es el acto de elegir una confirmación de una rama y aplicarla a otra. git cherry-pick puede ser útil para deshacer cambios.



Por ejemplo, supongamos que una confirmación se aplica accidentalmente en la rama equivocada. Puedes cambiar a la rama correcta y ejecutar cherry-pick en la confirmación para aplicarla a donde debería estar. Cherry-pick, por otro lado, le permite seleccionar compromisos individuales para la integración.

**Comandos:**

**git cherry-pick commitSha**

**commitSha** es una referencia de confirmación. Puedes encontrar una referencia de confirmación con el comando **git log.**

**$ git cherry-pick af02e0b**

De esta manera, la revisión especificada se confirmará directamente en su rama actualmente desprotegida. Si desea realizar más modificaciones, también puede indicar a Git que solo agregue los cambios de la confirmación a su copia de trabajo, sin confirmarlos directamente:

**$ git cherry-pick af02e0b --no-commit**