Pull Request

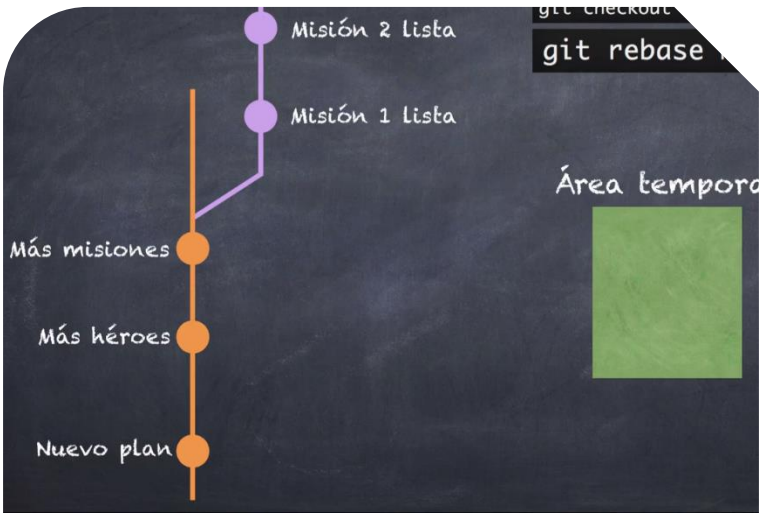
Un *Pull Request* es cuando queremos realizar aportaciones como: subir cambios a un proyecto en un repositorio, se inicia el *pull request* que para el dueño o líder del proyecto pueda observar los cambios que se desean incorporar al proyecto, si esto cambios no son aprobados por el líder del proyecto no afectara al código fuente del proyecto, cunado los cambios son aceptados y aprobados, estos cambios se verán reflejados en proyecto fuente.

Fork

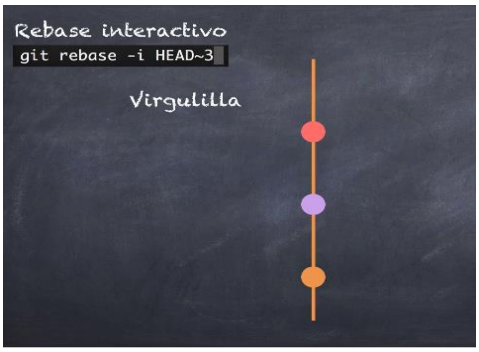
Un **fork** no sirve para realizar una copia de algún proyecto en *github* asía nuestra cuenta de *github*, de tal forma que podemos realizar los cambios que se deseemos sin afectar al proyecto fuente.

Rebace

Cuando en la rama main un compañero crea un commit y nosotros queremos que nuestro commit queden antes de los que hizo nuestro compañero, el git rebase mueve nuestros commit a un área temporal y los dejara al final de los commit de nuestro compañero.



Tenemos otro tipo de rebase como el rebase interactivo con el cual podemos indicar el número de commits antes para realizar mover nuestro commit



El rebase nos sirve para ordenas commits, unir commits, y separar commits

Stash

El *stash* es un espacio en el cual podemos poner todos los cambios temporales para dejar la rama en el último commit con los cambios funcionales y aprobados y subirlos

Cuando ya han subido los cambios podemos obtener del *stash* los cambios que se estaban realizando con el comando *git stash pop*

Hay que tener cuidado cuando trabajamos con el *stash* dado que puede causar conflictos y si git no puede realizar el auto *merge* nosotros tendremos que solucionar los conflictos indicando los cambios que queremos que se guarden.

Clean

git-clean: elimina archivos sin seguimiento del árbol de trabajo.

Limpia el árbol de trabajo eliminando recursivamente archivos que no están bajo control de versiones, comenzando desde el directorio actual.

Normalmente, sólo se eliminan los archivos desconocidos para Git, pero si -x se especifica la opción, también se eliminan los archivos ignorados. Esto puede resultar útil, por ejemplo, para eliminar todos los productos de compilación.

Cherry-pick

Dadas una o más confirmaciones existentes, aplique el cambio que introduce cada una, registrando una nueva confirmación para cada una. Esto requiere que su árbol de trabajo esté limpio (sin modificaciones del compromiso HEAD).

Cuando no es obvio cómo aplicar un cambio, sucede lo siguiente:

La rama actual y HEAD el puntero permanecen en la última confirmación realizada con éxito.

El CHERRY_PICK_HEAD árbitro señalará el compromiso que introdujo el cambio que es difícil de aplicar.

Las rutas en las que el cambio se aplicó limpiamente se actualizan tanto en el archivo de índice como en su árbol de trabajo.

Para rutas conflictivas, el archivo de índice registra hasta tres versiones, como se describe en la sección "TRUE MERGE" de git-merge[1] . Los archivos del árbol de trabajo incluirán una descripción del conflicto entre corchetes por los marcadores de conflicto habituales <<<<<<<y >>>>>>>.