Usar una herramienta como GitHub, GitLab, BitBucket, entre otros trae muchas ventajas para un proyecto. Estas herramientas son de control de versiones lo que permite ver que ha cambiado en el código y quien lo ha hecho. Además, si el código principal se rompe se puede volver a una versión estable y seguir trabajando a partir de ahí. Por otro lado, implica que todos los que desarrollan tienen el código actualizado al alcance de la mano y poder crear cuantas branches se quieran para poder trabajar y mantener un orden adecuado.

Para manejar un repositorio en una herramienta de control de versiones donde trabajan varias personas, es importante establecer reglas para el mismo. Sino se establecen reglas claras es muy fácil que se pierda el control y todos los participantes trabajen de formas distintas.

Antes de iniciar se debe seleccionar el tipo de repositorio que se utilizará ya sea poly o mono repo, donde el primero es donde se usa un solo repositorio por proyecto mientras que en mono en un mismo repositorio se tienen todos los proyectos.

Una de las mejores formas de usar git es por medio de branches y existen muchas estrategias para manejarlas y controlarlas durante el proceso de desarrollo. Esto ayuda a definir como bugs, features, optimizaciones entre otros se manejarán y como se pondrán en el Branch principal todo esto sin afectar el código que se encuentra en producción.

En git existen los submódulos que representan una gran ventaja dado que permiten agregar algún otro proyecto dentro de otro. Y la ventaja es que no es necesario estar actualizando a la última versión únicamente se actualiza el repositorio y este se encarga de lo que haga falta. Generalmente se usan cuando se desean utilizar librerías creadas por terceros. Además, se trabajan por separado en los commits.

Uno de los modelos más conocidos para administrar git es el llamado “gitflow”. En este modelo se tiene una Branch principal y se sacan branches del mismo donde se trabajan las features o algún bug que se este arreglando y una vez que esté totalmente listo entonces se le hace el merge con el master (este es el Branch principal). Un solo usuario puede tener muchos branches donde trabaja con distintas actualizaciones al master, pero no interactúan entre sí.

En el mundo de git cuando se trabaja localmente existe el concepto de upstream que se refiere a la ubicación a través de la cual se clona el repositorio. Para usar esto se utiliza el comando “git –set-upstream origin nombreBranch” esto lo que hace es subir el Branch local al repositorio remoto para que también exista ahí. Por otro lado, remote es lo que no se tiene en el repositorio clonado localmente, ósea que solo existe en el repositorio remote y se debe hacer un pull para poder actualizar el local.

Por lo general, cuando se desea unir un Branch de un desarrollador con el master se debe de hacer un pull request. Esto implica generalmente que al código que ingresa se le hará un code review, inclusive si así se tiene definido se pondría a correr un pipeline para asegurarse que el nuevo código no va a romper el código actual. Al ser una petición de unión debe existir algún desarrollador que no sea el que pide el pull request que lo apruebe. Esta aprobación implica de cierta forma que ambos son responsables de los cambios que se harán.