GIT Y GITHUB

Introducción

Git

Git es una herramienta conocida como "repositorio", el cual es un espacio centralizado donde se almacena, organiza, mantiene y difunde información, en este caso programas y código. Es una plataforma que de forma local va a reuniendo y organizando la información que nosotros vamos generando, para ir generando versiones de nuestros programas. La finalidad de las versiones consiste en nosotros poder ir hacia atrás en el tiempo y recuperar software sin cambios o recuperar versiones descartadas por las razones que sean.

Una manera en que se suele explicar el tema de las versiones y es justamente verlo como un árbol; donde existe un tronco principal generalmente conocido como MAIN el cual generalmente es la rama principal y en muchos casos el software "en producción".

Cada rama puede ser una propuesta inconclusa, un parche, una actualización o lo que la necesidad haya demandado en su momento.

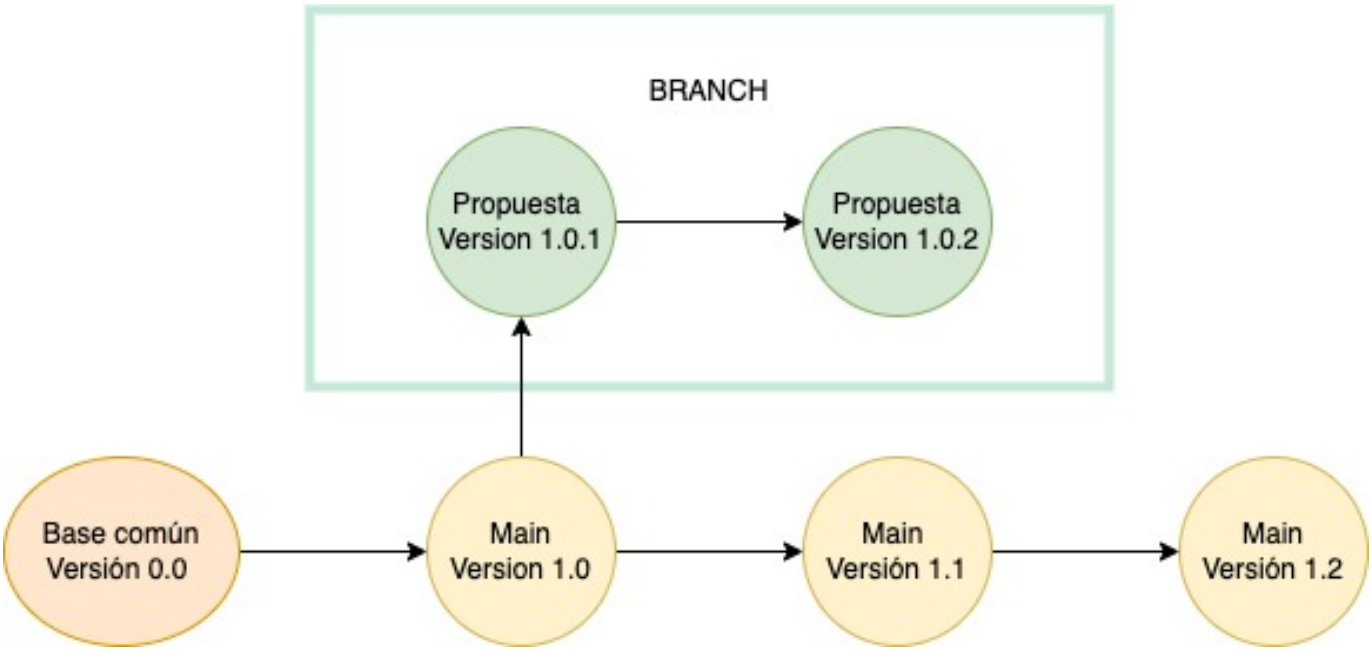
¿Por qué usar repositorios?
El desarrollo del software rara mente es lineal y no es siempre ir hacia delante es necesario a veces regresar en el tiempo revisar todo lo que se ha trabajado a lo largo del tiempo. En un repositorio se puede consultar todas las versiones, saber quién las hizo y cuando las hizo, e incluso podemos guardar la documentación correspondiente de una manera más ordenada.

.Git Hub.

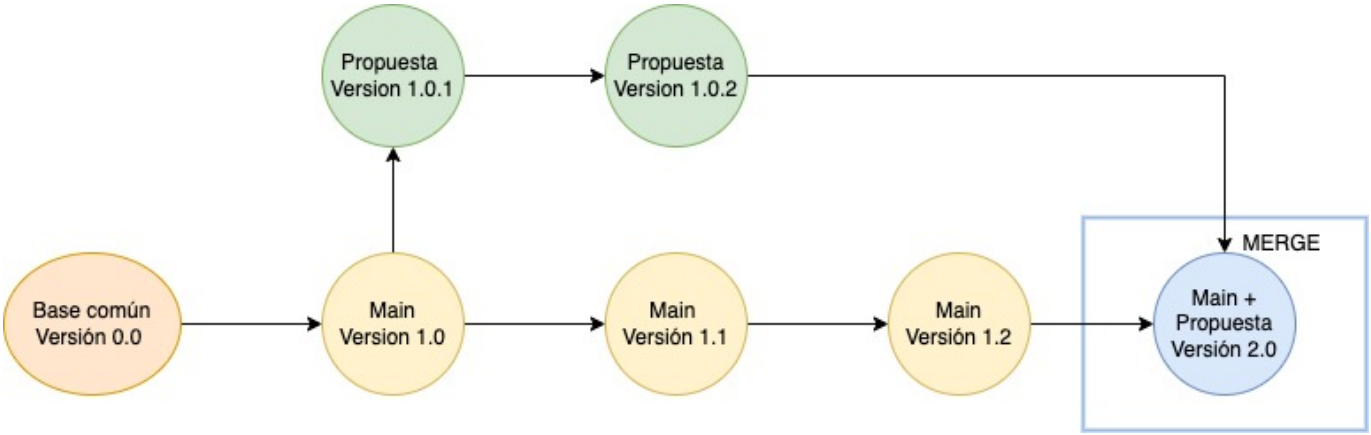
Es la plataforma de Git en que se alojan proyectos, usando el sistema de controlador de versiones, éste se comparte en modalidad "nube" y sea accesible desde cualquier punto del mundo; ya sea de manera privada o pública. De esta manera se facilita el trabajo en equipo dado que cualquiera puede colaborar en los proyectos, revisar la documentación, realizar nuevas propuestas y discutir sobre cualquier aspecto del proyecto desde cualquier parte del mundo.

Merge, branch, conflicts.

En este contexto, se escucha hablar de las "ramas" o branch en inglés y se refiere a una vertiente que toma el desarrollo de software que sigue una línea de versiones como en el ejemplo del árbol.



Algunas veces, las versiones deberán ser fusionadas en una sola rama y lo que normalmente ocurre es que se fusionan las 2 extremidades de dicho branch o “rama” para generar una nueva versión.



Sin embargo en algunas ocasiones esto no es tan simple y se pueden generar choques en las versiones, por lo que se debe de conocer el tipo de merge que se está intentando llevar a cabo y por ende, el conflicto que está presentando. Una vez teniéndolos en cuenta, se ejecutan comandos específicos que ayudan a resolver la situación.

Comandos Básicos			
git add	Agrega los cambios desde el directorio de archivos al área del repositorio.		
git branch	Se usa para administrar las ramas del software que se está trabajando. Tiene variantes para administrarlas	git branch / git branch —list	Enumera todas las ramas del repositorio
		git branch <branch>	Crea una nueva rama (en este caso llamada “branch”)
		git branch -d <branch>	Elimina la rama de forma segura
		git branch -D <branch>	Elimina la rama de forma forzada.
		git branch -m <branch>	Cambia el nombre de la rama
		git branch -a	Enumera todas las ramas remotas.
git checkout	Permite desplazarse por las ramas creadas por el git branch. E igualmente tiene varias variantes.	git checkout -b <new-branch>	Se crea y se extrae la rama llamada “new-branch” simultáneamente
		git checkout -b <name-branch>	Se extrae la rama “name-branch” en el HEAD actual.
		git checkout -b <name-branch> <existing-branch>	A diferencia de los anteriores, aquí se indica en qué HEAD se encuentra ese branch
git fetch	Descarga una rama de otro repositorio junto con todas sus confirmaciones y archivos asociados		
git merge	Se utiliza para combinar dos ramas, generalmente el “último punto” de ambas y los fusiona en una sola		
git clean	Elimina los archivos sin seguimiento de tu directorio de trabajo. Es la contraparte lógica de git reset, que normalmente solo funciona en archivos con seguimiento.		
git clone	Crea una copia de un repositorio de Git existente. La clonación es la forma más habitual de que los desarrolladores obtengan una copia de trabajo de un repositorio central.		
git commit	Confirma la instantánea preparada en el historial del proyecto. En combinación con git add, define el flujo de trabajo básico de todos los usuarios de Git.		
git config	Este comando va bien para establecer las opciones de configuración para instalar Git.		

Comandos Básicos			
git init	Inicializa un nuevo repositorio de Git.		
git log	Este comando muestra el historial de confirmaciones completo con el formato predeterminado	git log -n <limit>	Esta opción limita el número de confirmaciones
		git log --author="<pattern>"	Busca de acuerdo a un solo autor
		git log <since>..<until>	Solo muestra las confirmaciones que se producen entre < since > y < until >.
git pull	Este comando es la versión automatizada de git fetch. Descarga una rama de un repositorio remoto e inmediatamente la fusiona en la rama actual.		
git push	Permite mover una o varias ramas a otro repositorio, lo que es una buena forma de publicar contribuciones		
git reset	Deshace los cambios en los archivos del directorio de trabajo. El restablecimiento permite limpiar o eliminar por completo los cambios que no se han enviado a un repositorio público		
git status	Muestra el estado del directorio en el que estás trabajando y la instantánea preparada.		