14-7-2023

Identifica herramientas de versionamiento

GA7-220501096-AA1-EV03

Juan Sebastian Rojas Amaya
SENA CENTRO DE GESTION ADMINISTRATIVA

Contenido

INTRODUCCIÓN

La Integración Continua es una práctica de desarrollo de software que se ha vuelto cada vez más popular en los últimos años. Esta práctica implica la automatización del proceso de construcción, prueba y despliegue de software, es por eso que en el control versionamiento encontramos a Git y Github para realizar todo tipo de trabajo.

OBJETIVO

Conocer las diferencias, características y comandos de git local y git remoto para entender cómo funciona cada uno por separado y en conjunto.

DESARROLLO DE LA ACTIVIDAD

Diferencias entre Git y Github facilita la colaboración con git. Es una plataforma que puede mantener repositorios de código en almacenamiento basado en la nube para que varios desarrolladores puedan trabajar en un solo proyecto y ver las ediciones de cada uno en tiempo real:

Git	Github

Git es un sistema de control de versiones distribuido (DVCS) de software de código abierto y gratuito diseñado para administrar todo el historial del código fuente. Esto les permite a los desarrolladores mantener un historial de las confirmaciones, revertir cambios y compartir códigos con otras personas.	Github es un repositorio de Git, ofrece control de versiones y administración de códigos entre otras funcionalidades. Por ejemplo, la creación de proyectos colaborativos y seguimiento de fallas o errores, además los desarrolladores pueden usarlo como backups para crear repositorios a los que acceden después y para compartirlo con otros.

Diferencias entre Git local y Git remote

Git local		**Git remote**	
Descripción	Comando	Descripción	Comando
inicializa un repositorio desde un directorio existente se debe usar el comando Git init	**$ git init**	Para ver los repositorios remotos configurados se puede ejecutar el comando	**$ git remote**
Este comando creará un nuevo subdirectorio llamado .git, el cual contendrá todos los archivos necesarios para el repositorio. Si se desea obtener una copia desde un repositorio existente se debe utilizar el comando de clonación.	**$ git clone https://url_del_repositorio**	Para definir un repositorio remoto y asociarlo a un nombre para su referenciación se utiliza el siguiente comando:	**git remote add [nombre-remoto] [url]**

Sirve para visualizar el estado actual de tus archivos indicando si están o no rastreados por Git	**$ git status**	Donde nombre-remoto corresponde al nombre con que se referencia el repositorio y URL es la ubicación lógica del mismo en un entorno de red o en una dirección de internet. En el siguiente ejemplo se define un repositorio ubicado en un servidor de Github y cuyo nombre de referencia es ref.	**$ git remote add ref https://github.com/paulboone/ticgit**
Para todos los archivos nuevos que se deseen ser rastreados por Git, se debe indicar el siguiente comando:	**$ git add Nombre_archivo**	Una vez definido el remote se pueden extraer los datos utilizando el siguiente comando:	**$ git fetch [nombre-remoto]**
También es posible indicar a Git que haga rastreo de un directorio, lo cual implica que recursivamente se hace rastreo de todos los archivos en el interior del directorio.	**$ git add Directorio**	El anterior comando se conecta al repositorio remoto y trae todos los datos de los cuales aún no se tiene copia en tu repositorio local. Para enviar información desde el repositorio local hacia el servidor remoto, se utiliza el siguiente comando:	**git push [nombre-remoto] [nombre-rama]**
El comando git add además de servir para iniciar el rastreo de un archivo o directorio que no estaba en la última versión,	**$ git add Directorio**	Recordemos que, si se ha clonado un repositorio desde alguna ubicación, Git asigna el nombre origin al servidor del que se ha realizado la clonación. Así, si por ejemplo queremos enviar nuestra rama master al servidor origin se debe ejecutar el comando de la siguiente forma:	**$ git push origin master**

También es posible indicar a Git que haga rastreo de un directorio, lo cual implica que recursivamente se hace rastreo de todos los archivos en el interior del directorio.	**$ git commit**		
Al ejecutar la confirmación el sistema desplegará un mecanismo para recibir un mensaje de confirmación que será asociado a esta operación de commit. También es posible agregar el comentario explícitamente en la ejecución	**$ git commit -m "En esta versión se arregló el archive W"**		
de la confirmación usando la opción -m.			
También es posible ejecutar una operación de confirmación que salte el paso de preparación que se logra con la ejecución del comando add. Es decir, la operación de confirmación se encarga de preparar todos los archivos rastreados y luego confirmar. Esto es posible agregando la opción -a.	**$ git commit -a -m 'comentario de esta confirmación'**		
Para visualizar el histórico de las confirmaciones desde la más reciente hasta la más antigua realizadas sobre un repositorio se ejecuta el comando:	**$ git log**		

CONCLUSION

Git y GitHub son herramientas complementarias pero diferentes en su enfoque. Git es el sistema de control de versiones subyacente que permite el seguimiento de los cambios en el código fuente, mientras que GitHub es una plataforma en línea que aprovecha Git para facilitar la colaboración, la gestión de proyectos y el alojamiento de repositorios remotos. Tanto Git como GitHub son esenciales para los desarrolladores, y su combinación ofrece un flujo de trabajo más eficiente y una mayor visibilidad en el desarrollo de software.