Evidencia de conocimiento: GA7-220501096-AA1-EV03 identifica herramientas de versionamiento

Aprendiz: José Carlos Pinzón Ferrer

Servicio Nacional de Aprendizaje Tecnólogo en Análisis y Desarrollo de Software

Instructor: ***Cristian Camilo Arias***

Bogotá, Colombia 23 de marzo de 2024

INTRODUCCIÓN

GitLab y GitHub son sistemas de control de versiones para la gestión del código fuente en el desarrollo de software. En el ámbito del desarrollo de software, la gestión eficiente del código fuente es fundamental para el éxito de un proyecto. En este sentido, herramientas como Git y GitHub han revolucionado la forma en que los equipos de desarrollo colaboran y gestionan sus proyectos. Si bien ambos términos suelen utilizarse indistintamente, es importante comprender la diferencia principal entre Git y GitHub para aprovechar al máximo su potencial. En este texto, exploraremos en detalle esta distinción y cómo cada una de estas herramientas contribuye al proceso de desarrollo de software.

OBJETIVO

En este tema busco clarificar la diferencia fundamental entre Git y GitHub, dos herramientas esenciales en el desarrollo de software moderno. A través de esta distinción, buscamos proporcionar una comprensión más profunda de cómo estas herramientas se complementan entre sí y cómo pueden utilizarse de manera efectiva en proyectos de desarrollo. Al comprender la naturaleza de Git como un sistema de control de versiones de código abierto instalado localmente y GitHub como un servicio en línea para alojar repositorios de Git, los desarrolladores podrán tomar decisiones informadas sobre la implementación y gestión de sus proyectos, mejorando así la eficiencia y la colaboración en el proceso de desarrollo de software.

Diferencias entre Git y Github

GitHub simplifica la colaboración mediante el uso de Git, proporcionando una plataforma que permite almacenar repositorios de código en la nube. Esto posibilita que múltiples desarrolladores trabajen en un mismo proyecto, visualizando en tiempo real las ediciones realizadas por cada uno.

Git	GitHub
• Git es un sistema de control de versiones distribuido. • Se instala localmente en la máquina de cada desarrollador. • Permite el seguimiento de cambios en los archivos de un proyecto a lo largo del tiempo. • Facilita la gestión de ramas para trabajar en paralelo en diferentes características o versiones del proyecto. • Proporciona herramientas para fusionar cambios de diferentes ramas de manera eficiente. • No depende de una conexión a Internet para funcionar, ya que todos los cambios se realizan localmente.	• GitHub es una plataforma de alojamiento de repositorios de código basada en la web. • Permite a los desarrolladores almacenar y gestionar repositorios Git de manera remota. • Facilita la colaboración entre equipos de desarrollo, ya que varios desarrolladores pueden trabajar en un mismo proyecto. • Ofrece herramientas para gestionar problemas (issues), solicitudes de extracción (pull requests), y discusiones en torno al código. • Proporciona funciones adicionales como la integración continua (CI), despliegue continuo (CD) y la automatización de flujos de trabajo a través de GitHub Actions. • Permite a los desarrolladores acceder y contribuir a proyectos de código abierto alojados en la plataforma.

Git local y Git remoto se utilizan y se comportan en un entorno de integración continua, mostrando las ventajas y desventajas de cada uno en ese contexto específico.

Git local		Git remote	
Descripción	Comando	Descripción	**Comando**
Para crear un nuevo repositorio en un directorio ya existente, es necesario utilizar el comando "Git init".	**$ git init**	Se puede visualizar los repositorios remotos configurados mediante la ejecución del comando.	**$ git remote**
Este comando generará un recién creado subdirectorio denominado git, el cual incluirá todos los archivos esenciales para el repositorio. En caso de querer obtener una réplica de un repositorio ya existente, se debe emplear el comando de clonación.	**$ git clone https://url_del_repositorio**	**Para establecer un repositorio remoto y asignarle un nombre para su identificación, se emplea el comando siguiente →**	**git remote add [nombre-remoto] [url]**
Esta función permite observar la situación presente de tus archivos, señalando si están siendo seguidos por Git o no.	**$ git status**	**En este contexto, "nombre-remoto" se refiere al nombre mediante el cual se hace referencia al repositorio, mientras que "URL" representa la dirección lógica del repositorio en una red o en Internet. A continuación, se muestra un ejemplo donde se define un repositorio alojado en un servidor de GitHub con el nombre de referencia "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 addDirectorio**	**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 noestaba en la última versión	**$ git addDirectorio**	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 nuestrarama 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.	**$ git commit -m "En esta versión se arreglo el archive W"**		

También es posible ejecutar una operación de confirmación quesalte el paso de preparación quese logra con la ejecución del comando add. Es decir, la operación de confirmación seencarga de preparar todos los archivos rastreados y luegoconfirmar. 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**		