IDENTIFICA HERRAMIENTAS DE VERSIONAMIENTO.
GA7-220501096-01-AA1-EV03

PRESENTADO POR
KELY JOHANA DUARTE BOLIVAR

INSTRUCTOR
FELIPE MARTINEZ LEIVA

SERVICIO NACIONAL DE APRENDIZAJE SENA
ANÁLISIS Y DESARROLLO DE SOFTWARE
FICHA 2721403
23 DE MARZO DEL 2024

INTRODUCCIÓN

El versionamiento es una práctica esencial en el desarrollo de software que permite gestionar y controlar cambios en el código fuente de manera sistemática. Herramientas de versionamiento facilitan este proceso al ofrecer funciones para el seguimiento, comparación y colaboración en el desarrollo de proyectos.

ACTIVIDAD

Tomando como referencia el componente formativo “Integración continua”, realice una tabla con las diferencias entre el sistema de control de versionamiento git local y git remoto.

Elementos para tener en cuenta en el documento:

1. Se deben seguir las normas básicas de presentación de un documento escrito, es decir el documento debe tener como mínimo una portada, introducción, objetivo, tablas con diferencias y características y comandos de git local y git remoto.
2. Realice una tabla con la descripción de los comandos básicos de git remoto y git local.

¿QUÉ ES GIT?

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. Este les permite a los desarrolladores mantener un historial de las confirmaciones, revertir cambios y compartir códigos con otras personas. Para poder usarlo se debe instalar en el computador. Esto quiere decir que las personas que lo usan no necesitan de Internet para usarlo o acceder a su repositorio local.

Es actualmente unas de las herramientas más conocidas y usado por empresas como Amazon, Microsoft, Facebook, por solo nombrar algunas.

¿QUÉ ES GITHUB?

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. Al ser una red, no está guardado en nuestro computador, sino en la nube. Lo que quiere decir que necesita de Internet y para usarlo o acceder a su biblioteca necesitamos una conexión.

Existen versiones de escritorio, pero esto no significa que esté instalado en nuestro computador, en su lugar es una herramienta que ayuda a sincronizar nuestro computador con el servidor.

Los desarrolladores pueden usarlo como backups para crear repositorios a los que acceden después y para compartirlo con otros; Actualmente tienen varios competidores; por ejemplo, Gitlab. Pero esta no tiene un repositorio tan extenso como Github.

DIFERENCIAS	
GIT	**GITHUB**
1. Es un software. 2. Está alojado localmente en el sistema. 3. Es una herramienta de linea de comandos. 4. Es una herramienta de maneja distintas versiones de las ediciones de los archivos. 5. Proporciona Sistema de Control de Versiones(VCS), la Gestión del Código Fuente(SCM). 6. Git puede ser usado sin GitHub. 7. Git es de Linux y fue lanzado en 2015	1. Es un servicio. 2. Está alojado en la web. 3. Proporciona una interfaz gráfica. 4. Es un espacio para subir una copia del repositorio Git. 5. Proporciona funcionalidades de Git como el Sistema de Control de Versiones(VCS), la Gestión del Código Fuente(SCM), además de añadir algunas características propias. 6. En cambio GitHub no puede ser usado sin Git. 7. Mientras que Github es de Microsoft.

CARACTERÍSTICAS	
GIT	**GITHUB**
Git tiene innumerables opciones que podemos realizar como: 1. Rastreo de cambios. 2. Gestión de ramas. 3. Fusiones y resolución de conflictos. 4. Control de versiones en local sin necesidad de internet. 5. Revertir cambios 6. Creación de etiquetas.	Algunas de las características que podemos utilizar con GitHub son las siguientes: 1. Alojar repositorios Git en la nube. 2. Colaboración en equipo. 3. Seguimiento de problemas. 4. Solicitudes de extracción (Pull request). 5. Revisión de código. 6. Integración continua. 7. Ramificaciones y fusiones de ramas. 8. Gestión de acceso. 9. Estadísticas y análisis.

COMANDOS BÁSICOS

1. **Git init:** Inicia un nuevo repositorio, esto creará el “staging” o área de ensayo y un repositorio local: git init

2. **Git clone:** Obtiene una copia de un proyecto que se encuentra en un repositorio público: git clone nombredeusuario@host:/path/to/repository
   A la inversa, ejecuta el siguiente comando básico para copiar un repositorio local: git clone /path/to/repository

3. **Git add:** Añade un archivo al área de ensayo. Ejemplo: git add [Nombre-archivo] o git add . para añadir todo.

4. **Git branch:** Permite crear, enumerar y eliminar ramas, así como cambiar su nombre. Por ejemplo, si quieres listar todas las ramas presentes en el repositorio, el comando debería verse así: git branch o si quieres borrar una rama, usa: git branch -d <branch-name>

5. **Git checkout:** Se utiliza principalmente para cambiarte de una rama a otra y para chequear archivos y commits. Por ejemplo, el siguiente comando crea una nueva y automáticamente se cambia a ella: command git checkout -b [branch-name] Para cambiar de una rama a otra, sólo usa: git checkout [branch-name]

6. **Git status:** Brinda toda la información necesaria sobre los archivos de la rama actual. git status

7. **Git commit:** Guarda los archivos en el repositorio local. git commit –m “El mensaje que acompaña al commit va aquí” Debes tener en cuenta que los cambios confirmados no llegarán al repositorio remoto.

8. **Git push:** Envía los commits al repositorio remoto. IMPORTANTE: Git push solamente carga los cambios que han sido confirmados. git push origin [master] Reemplaza [master] con la rama en la que quieres enviar los cambios cuando no quieras enviarlos a la rama maestra.

9. **Git pull:** Extrae y descarga contenido desde un repositorio remoto y actualiza al instante el repositorio local reflejando ese contenido. git pull

10. **Git merge:** Integra las características de tu rama con todos los commits realizados a las demás ramas del repositorio. git merge [branch-name]

11. **Git resert:** Deshace cambios efectuados en el historial de confirmaciones de un repositorio. git reset - -hard HEAD

12. **Git config:** Puede ser usado para establecer una configuración específica de usuario, como el email, nombre de usuario y tipo de formato, etc. Por ejemplo, el siguiente comando se usa para establecer un email: git config --global user.email ejemplo@tecsify.com La opción -global le dice a GIT que vas a usar ese correo electrónico para todos los repositorios locales. Si quieres utilizar diferentes correos electrónicos para diferentes repositorios, usa el siguiente comando: git config --local user.email [tuemail @ejemplo.com].

13. **Git merge:** Se usa para fusionar una rama con otra rama activa: `git merge [branch-name]`

14. **Git log:** se usa para ver el historial del repositorio listando ciertos detalles de la confirmación. Al ejecutar el comando se obtiene una salida como ésta:
```
commit 15f4b6c44b3c8344caasdac9e4be13246e21sadw
Author: Oscar Morales [contacto@tecsify.com]
Date:   Fri Oct 7 12:56:29 2022 -0600
```

15. **Git stash:** guardará momentáneamente los cambios que no están listos para ser confirmados. De esta manera, puedes volver al proyecto más tarde. `git stash`

16. **Git fetch:** Le permite al usuario buscar todos los objetos de un repositorio remoto que actualmente no se encuentran en el directorio de trabajo local. `git fetch origin`

CONCLUSIÓN

En resumen, las herramientas de versionamiento son fundamentales para garantizar la integridad y la colaboración efectiva en el desarrollo de software. Con opciones como Git, Subversion y Mercurial, los equipos pueden gestionar eficientemente el flujo de cambios, mejorar la trazabilidad y mantener la coherencia en sus proyectos.