Presentación

Nombre: Pablo Marvin Mendoza Coste

Matricula: 2021-0799

Materia: Programación 3

Tema: Tarea 3

Maestro/a: Kelyn Tejada

1-Que es Git?

Git es un sistema de control de versiones distribuido creado por Linus Torvalds, que permite a los desarrolladores registrar cambios en sus archivos y volver a versiones anteriores si es necesario. Es ampliamente utilizado en proyectos de código abierto y comerciales, y es considerado el estándar mundial para el control de versiones debido a su flexibilidad, popularidad y amplio soporte en entornos de desarrollo. Con Git, los desarrolladores pueden trabajar localmente sin conexión y luego sincronizar su trabajo con el servidor central, lo que facilita la colaboración en equipos y el mantenimiento eficiente de versiones en proyectos complejos.

2-Para que funciona el comando Git init?

El comando ´git init´ se utiliza para crear un nuevo repositorio de Git en un directorio específico. Este comando es fundamental para comenzar a utilizar Git en un proyecto. Al ejecutar ´git init´, se creará un subdirectorio oculto llamado ´.git´en el directorio actual. Este subdirectorio contiene toda la información y metadatos necesarios para el control de versiones de Git.

Algunas de las cosas que ocurren cuando se ejecuta ´git init´ son:

- Se crea el subdirectorio .git que almacena todos los datos y metadatos del repositorio, como objetos, referencias y archivos de configuración.
- Se inicializa el repositorio vacío, lo que significa que aún no tiene ningún archivo o commit.
- Se crea el archivo HEAD, que apunta a la rama actualmente extraída. Al comienzo, este archivo suele estar vacío, ya que no hay commits aún.
- Crea una rama maestra (normalmente llamada "master") como rama predeterminada para comenzar a trabajar.

Una vez que se ha inicializado el repositorio con ´git init´, puedes comenzar a agregar archivos, realizar cambios y confirmarlos para que formen parte del historial de versiones del proyecto. Es importante notar que la mayoría de los comandos de Git solo están disponibles dentro de un repositorio inicializado, por lo que este es generalmente el primer paso para comenzar a utilizar Git en un proyecto nuevo o existente.

4-Que es una rama?

En Git, una rama es una línea de desarrollo independiente que permite a los desarrolladores trabajar en nuevas características, correcciones de errores o experimentos sin afectar directamente el código base principal del proyecto. Una rama en Git es simplemente un puntero móvil que apunta a un commit específico en la cadena de confirmaciones.

Cuando se crea un repositorio de Git, se crea automáticamente una rama principal, comúnmente llamada "master" (aunque puede tener otros nombres). Esta rama principal es donde se encuentra el código base principal del proyecto.

Cuando un desarrollador quiere agregar una nueva función o corregir un error, crea una nueva rama a partir de la rama principal. Esta nueva rama contendrá una copia exacta del código de la rama principal en ese punto específico. A partir de ahí, el desarrollador puede realizar cambios en la nueva rama de forma independiente, sin afectar el código en la rama principal.

Una vez que los cambios en la nueva rama se consideran completos y probados, pueden fusionarse nuevamente en la rama principal. La fusión combinará los cambios realizados en la nueva rama con el código base principal y creará una nueva instantánea que incluya todos esos cambios.

3-Como saber es que rama estoy?

Para eso debemos de usar el comando ´git branch´, este comando te mostrará una lista de todas las ramas en tu repositorio y resaltará con un asterisco (*) la rama en la que te encuentras actualmente. También puedes usar el comando ´ git status ´, que te mostrará información sobre el estado actual de tu directorio de trabajo, incluida la rama en la que te encuentras.

5-Quien creo git?

Linus Torvalds fue el creador de Git. La creación de Git surgió como respuesta a la necesidad de encontrar una nueva herramienta de control de versiones después de que la relación entre la comunidad de desarrollo de Linux y la compañía que desarrollaba BitKeeper se rompiera.

Linus y la comunidad de desarrollo de Linux tenían requisitos específicos para una nueva herramienta, como velocidad, diseño sencillo, soporte para desarrollo no lineal y capacidad para manejar grandes proyectos como el núcleo de Linux de manera eficiente. Con estos objetivos en mente, Linus y su equipo crearon Git, que se ha convertido en uno de los sistemas de control de versiones más populares y ampliamente utilizados en el mundo del desarrollo de software. Su diseño distribuido y su capacidad para manejar proyectos grandes y complejos lo han hecho especialmente exitoso en la comunidad de desarrollo de software de código abierto.

6-Cuales son los comandos más esenciales de Git?

Los comandos más esenciales de Git que todo estudiante o desarrollador debería conocer son:

- **git init:** Inicializa un nuevo repositorio de Git en un directorio vacío o convierte un proyecto existente en un repositorio de Git.

- **git clone <URL>:** Clona un repositorio existente desde una URL remota a un repositorio local.

- **git add <archivo>:** Agrega los cambios realizados en un archivo específico al área de preparación (staging).

- **git commit -m "mensaje":** Guarda los cambios confirmados en el repositorio con un mensaje descriptivo.

- **git status:** Muestra el estado actual del repositorio, incluyendo los cambios realizados, los archivos agregados al área de preparación y los archivos sin seguimiento.

- **git log:** Muestra el historial de confirmaciones realizadas en el repositorio.

- **git pull:** Obtiene y fusiona los cambios desde un repositorio remoto al repositorio local.

- **git push:** Envía los cambios confirmados en el repositorio local a un repositorio remoto.

- **git branch:** Muestra una lista de ramas en el repositorio y resalta la rama actual.

- **git merge <rama>:** Fusiona los cambios de una rama específica en la rama actual.
- **git remote add <nombre> <URL>:** Agrega un repositorio remoto con un nombre específico.

- **git reset:** Deshace los cambios realizados en el área de preparación y el directorio de trabajo.

- **git stash:** Guarda temporalmente los cambios no comprometidos para trabajar en otra rama o tarea.

7-Que es git Flow?

GitFlow es una metodología alternativa para la creación de ramas en Git que utiliza tanto ramas de función como varias ramas principales. Fue introducido y popularizado por Vincent Driessen en nvie. En comparación con el desarrollo basado en troncos, GitFlow involucra ramas con una mayor duración y más confirmaciones. De acuerdo con este modelo, los desarrolladores crean una rama de función y la mantienen separada de la rama principal hasta que la función está completa. Estas ramas de función de larga duración requieren una mayor colaboración para su fusión y pueden presentar un mayor riesgo de desviarse de la rama principal, lo que podría ocasionar conflictos en las actualizaciones.

El flujo de trabajo de Git Flow se basa en dos ramas principales: la rama master y la rama develop. La rama master contiene el código estable y probado que se ha lanzado en producción, mientras que la rama develop es donde se integran y prueban todas las características nuevas antes de ser lanzadas.

Además de las ramas principales, Git Flow utiliza otras ramas específicas para diferentes propósitos. Las ramas de función se crean para desarrollar nuevas características o mejoras y se fusionan nuevamente en la rama develop una vez que la función está completa. Las ramas de publicación se utilizan para preparar una nueva versión antes de su lanzamiento y se crean a partir de la rama develop. Por último, las ramas de corrección se utilizan para solucionar problemas críticos en producción y se crean a partir de la rama master, luego se fusionan con la rama master y develop una vez que se corrige el error.

8-Que es trunk based development ?

El desarrollo basado en troncos (Trunk Based Development) es una práctica de gestión de control de versiones en la que los desarrolladores fusionan pequeñas actualizaciones de forma frecuente en una rama principal o tronco. Esta práctica ayuda a lograr la Integración Continua y el Despliegue Continuo, lo que aumenta la entrega de software y el rendimiento de la organización.

En este enfoque, se utiliza una sola rama principal (trunk) en el repositorio, y los desarrolladores trabajan en ramas de corta duración con pequeñas confirmaciones. Esto contrasta con otras estrategias como Git Flow, que involucran ramas de función de larga duración. La idea es mantener el flujo de publicación de la producción y simplificar las fases de fusión e integración.

El desarrollo basado en troncos requiere un alto grado de madurez por parte del equipo de desarrollo, ya que todos los desarrolladores trabajan sobre la misma rama principal. Cualquier error crítico puede afectar a todo el equipo, por lo que es esencial contar con buenas prácticas de versionado, pruebas de calidad y seguridad antes de promover cambios a la rama principal y desplegar en producción. Es una estrategia muy eficaz para equipos maduros y bien coordinados.