Materia

Programación 3

Profesor

Kelyn Tejada Belliard

Tema

Tarea 3 Cuestionario

Estudiante

Ivan Fernando Rodríguez Gautreau

Matricula

2021-1299

Fecha de Entrega

31/3/2023

Santo Domingo, República Dominicana, Distrito Nacional

1-¿Que es Git?

Git es un proyecto de código abierto maduro y mantenido activamente desarrollado originalmente en 2005. Una cantidad asombrosa de proyectos de software confían en Git para el control de versiones, incluidos proyectos comerciales y de código abierto. Los desarrolladores que han trabajado con Git están bien representados en el grupo de talentos de desarrollo de software disponible y funciona bien en una amplia gama de sistemas operativos e IDE (entornos de desarrollo integrado).

Al tener una arquitectura distribuida, Git es un ejemplo de un DVCS (por lo tanto, Sistema de control de versiones distribuido). En lugar de tener un solo lugar para el historial de versiones completo del software, como es común en los sistemas de control de versiones que alguna vez fueron populares, como CVS o Subversion (también conocido como SVN), en Git, la copia de trabajo del código de cada desarrollador también es un repositorio. que puede contener el historial completo de todos los cambios.

2-¿Para que funciona el comando Git init?

El comando git init crea un nuevo repositorio Git. Se puede usar para convertir un proyecto no versionado existente en un repositorio de Git o inicializar un nuevo repositorio vacío. La mayoría de los demás comandos de Git no están disponibles fuera de un repositorio inicializado, por lo que este suele ser el primer comando que ejecutará en un nuevo proyecto.

Al ejecutar git init, se crea un subdirectorio git en el directorio de trabajo actual, que contiene todos los metadatos de Git necesarios para el nuevo repositorio. Estos metadatos incluyen subdirectorios para objetos, referencias y archivos de plantilla.

4-¿Que es una rama?

Una rama representa una línea independiente de desarrollo. Las ramas sirven como una abstracción para el proceso de edición/etapa/confirmación. Puede pensar en ellos como una forma de solicitar un directorio de trabajo, un área de preparación y un historial de proyectos completamente nuevos. Las nuevas confirmaciones se registran en el historial de la rama actual, lo que da como resultado una bifurcación en el historial del proyecto.

3-¿Como saber es que rama estoy?

Una forma de revisar la rama actual en git es el comando git branch –show-current. Tambien se puede revisar las ramas existentes utilizando únicamente el git branch.

5-¿Quien creo git?

Git fue creado por Linus Torvalds, un ingeniero de software tambien reconocido por desarrollar el kernel. Git se creo en el 2005 como una herramienta de gestión de versiones en solo 2 semanas, con el objetivo de ofrecer eficiencia, confiabilidad y

compatibilidad para el mantenimiento de aplicaciones que contienen múltiples archivos de código fuente. Su función principal consiste en registrar los cambios realizados en los archivos de una computadora y coordinar el trabajo colaborativo de varias personas que trabajan en archivos compartidos en un repositorio de código.

6-¿Cuales son los comandos más esenciales de Git?

Los comandos mas esenciales de Git son:

git init: Crea un nuevo repositorio de Git vacío.

git clone: Crea una copia local de un repositorio remoto.

git add: Agrega cambios al área de preparación o "staging".

git commit: Registra cambios en el repositorio.

git push: Envía los cambios locales al repositorio remoto.

git pull: Descarga y fusiona los cambios del repositorio remoto al repositorio local.

git status: Muestra el estado actual de los archivos en el árbol de trabajo y en el área de preparación.

git log: Muestra una lista de los commits en el repositorio.

git branch: Muestra una lista de ramas en el repositorio y permite crear, eliminar y cambiar entre ellas.

git checkout: Cambia entre diferentes ramas o versiones de los archivos en el árbol de trabajo.

git merge: Combina los cambios de una rama en otra rama.

7-¿Que es git Flow?

Gitflow es un modelo de ramificación alternativo de Git que implica el uso de ramas de características y múltiples ramas principales. Fue publicado por primera vez y popularizado por Vincent Driessen en nvie. En comparación con el desarrollo basado en Trunk, Gitflow tiene ramas de mayor duración y compromisos más grandes. Bajo este modelo, los desarrolladores crean una rama de feature y retrasan su fusión en la rama principal del tronco hasta que se completa la característica. Estas ramas de características de larga duración requieren más colaboración para fusionarse y tienen un mayor riesgo de desviarse de la rama del tronco. También pueden introducir actualizaciones en conflicto.

Gitflow se puede utilizar para proyectos que tienen un ciclo de lanzamiento programado y para la práctica recomendada de DevOps de entrega continua. Este flujo de trabajo no agrega nuevos conceptos o comandos más allá de lo que se requiere para el Flujo de trabajo de ramas de características. En cambio, asigna roles muy específicos a

diferentes ramas y define cómo y cuándo deben interactuar. Además de las ramas de características, utiliza ramas individuales para preparar, mantener y registrar lanzamientos. Por supuesto, también se aprovechan todos los beneficios del Flujo de trabajo de ramas de características: solicitudes de extracción, experimentos aislados y una colaboración más eficiente.

8-Que es trunk based development ?

El desarrollo basado en trunk es una práctica de gestión de control de versiones en la que los desarrolladores fusionan actualizaciones pequeñas y frecuentes en un "trunk" o rama principal del código. Es una práctica común entre los equipos de DevOps y forma parte del ciclo de vida de DevOps, ya que simplifica las fases de fusión e integración. De hecho, el desarrollo basado en trunk es una práctica obligatoria de CI/CD. Los desarrolladores pueden crear ramas de corta duración con unos pocos pequeños compromisos en comparación con otras estrategias de ramificación de funciones de larga duración. A medida que la complejidad de la base de código y el tamaño del equipo aumentan, el desarrollo basado en trunk ayuda a mantener las versiones de producción en curso.