Cuestionario sobre Git

Nombre del estudiante: Luis Elian Pérez Carpio

Matrícula: 2021-0119

Profesor: Kelyn Tejada

Materia: Programación III

Índice

Introducción

Este documento responde detalladamente las preguntas teóricas sobre Git, un sistema de control de versiones ampliamente utilizado en el desarrollo de software.

Preguntas y respuestas

1. ¿Qué es Git?

Git es un sistema de control de versiones distribuido, diseñado para manejar proyectos de software de manera eficiente y con rapidez. Fue creado por Linus Torvalds en 2005 para el desarrollo del kernel de Linux y desde entonces ha sido adoptado en una variedad de proyectos. Git permite a los desarrolladores rastrear cambios en el código, colaborar con otros y revertir modificaciones cuando sea necesario. A diferencia de los sistemas de control de versiones centralizados, Git almacena una copia completa del repositorio en cada computadora de los usuarios, lo que mejora la velocidad y la seguridad de los datos.

2. ¿Para qué sirve el comando git init?

El comando git init se usa para inicializar un nuevo repositorio de Git en un directorio. Al ejecutarlo, Git crea una carpeta oculta llamada .git que contiene todos los archivos y metadatos necesarios para el control de versiones. Este comando se usa cuando se quiere empezar a gestionar un proyecto con Git desde cero. Si el directorio ya contiene archivos, git init no los modifica, pero permite empezar a hacer seguimiento de ellos mediante otros comandos como git add y git commit.

3. ¿Qué es una rama en Git?

Una rama en Git es una línea independiente de desarrollo dentro de un proyecto. Permite trabajar en nuevas características, corregir errores o probar cambios sin afectar el código principal. La rama principal en la mayoría de los repositorios es main o master, pero se pueden crear tantas ramas como sean necesarias.

Las ramas se pueden fusionar (merge) con otras cuando los cambios han sido probados y validados, permitiendo así un desarrollo flexible y organizado. Para crear una rama, se usa git branch nombre_de_la_rama y para cambiar de rama, git checkout nombre_de_la_rama o git switch nombre_de_la_rama.

4. ¿Cómo saber en cuál rama estoy trabajando?

Para saber en qué rama se está trabajando, se puede usar el comando git branch. Este comando muestra una lista de las ramas disponibles y marca con un asterisco (*) la rama actual. Otra opción es utilizar git status, que además de mostrar la rama actual, indica si hay archivos sin seguimiento, modificaciones pendientes de confirmación y otros detalles del estado del repositorio.

5. ¿Quién creó Git?

Git fue creado por Linus Torvalds en 2005. Torvalds es también conocido por haber desarrollado el kernel de Linux. Git nació de la necesidad de encontrar un sistema de control de versiones eficiente, rápido y distribuido para gestionar el desarrollo del código del kernel de Linux. Desde su creación, Git ha evolucionado y se ha convertido en una herramienta fundamental en el desarrollo de software moderno.

6. ¿Cuáles son los comandos esenciales de Git?

Algunos de los comandos esenciales de Git son:

- git init: Inicializa un nuevo repositorio de Git.
- git clone URL: Clona un repositorio remoto en la máquina local.
- git status: Muestra el estado actual del repositorio.

- git add archivo: Agrega un archivo al área de preparación (staging area).
- git commit -m "mensaje": Guarda los cambios en el repositorio con un mensaje descriptivo.
- git pull: Obtiene y fusiona cambios desde un repositorio remoto.
- git push: Envía los cambios locales a un repositorio remoto.
- git branch: Muestra las ramas existentes.
- git checkout nombre_de_rama o git switch nombre_de_rama: Cambia a otra rama.
- git merge nombre_de_rama: Fusiona una rama con la actual.
- git log: Muestra el historial de commits.

7. ¿Qué es Git Flow?

Git Flow es un modelo de flujo de trabajo basado en Git que define un proceso estructurado para gestionar ramas en un proyecto de desarrollo de software. Propone una estrategia clara dividiendo las ramas en diferentes categorías:

- main: Contiene el código estable listo para producción.
- develop: Rama principal donde se integran nuevas funcionalidades antes de ser liberadas.
- feature: Ramas utilizadas para desarrollar nuevas características, creadas a partir de develop y fusionadas de nuevo al completarse.
- release: Ramas creadas para preparar una nueva versión antes de fusionarse en main.
- hotfix: Ramas para corregir errores críticos en producción, creadas a partir de main y fusionadas de vuelta una vez resuelto el problema.

8. ¿Qué es el desarrollo basado en trunk (Trunk Based Development)?

El desarrollo basado en trunk (Trunk Based Development) es una estrategia en la que todos los desarrolladores trabajan directamente sobre una rama principal (trunk o main), en lugar de utilizar ramas largas o separadas para nuevas funcionalidades. Se basa en realizar commits frecuentes y pequeños en la rama principal, evitando la creación de muchas ramas divergentes.

Este enfoque reduce la complejidad de las fusiones (merge) y facilita la entrega continua. Es especialmente utilizado en entornos de desarrollo ágil y DevOps, donde se prioriza la integración rápida y la estabilidad del código.

Bibliografía

- Chacon, S., & Straub, B. (2014). *Pro Git*. Apress.
- Sitio oficial de Git: https://git-scm.com/
- Atlassian Git Tutorials: https://www.atlassian.com/git