INSTITUTO TECNOLÓGICO DE LAS AMÉRICAS (ITLA)

Asignatura: Control de Versiones

Respuestas Teóricas sobre Git

Nombre: Angela Todd

Matrícula: 20209974

Fecha: 05/04/2025

Índice

Respuestas

1. ¿Qué es Git?

Git es un sistema de control de versiones distribuido y de código abierto, creado para manejar todo tipo de proyectos de desarrollo de software, desde pequeños hasta muy grandes, con velocidad y eficiencia. Git permite a múltiples desarrolladores trabajar en el mismo proyecto sin interferencias, registrando los cambios realizados en los archivos, permitiendo así un historial completo y detallado del proyecto. Con Git, los desarrolladores pueden colaborar, crear ramas de desarrollo, fusionar cambios y revertir errores con facilidad.

2. ¿Para qué sirve el comando git init?

El comando `git init` se utiliza para inicializar un nuevo repositorio de Git. Este comando crea un nuevo subdirectorio llamado `.git` que contiene todos los archivos necesarios del repositorio —estructura de control de versiones, configuraciones y registros de cambios. Este comando se ejecuta una sola vez al comenzar un nuevo proyecto con Git, y transforma cualquier carpeta en un repositorio que puede empezar a ser versionado.

3. ¿Qué es una rama en Git?

Una rama en Git es una versión paralela del repositorio. Se utiliza para desarrollar funcionalidades de manera aislada del resto del código. Por ejemplo, se puede crear una rama para una nueva característica, trabajar en ella, y luego fusionarla con la rama principal (usualmente `main` o `master`) una vez finalizada. Las ramas permiten trabajar en múltiples líneas de desarrollo de forma simultánea sin afectar la estabilidad del proyecto principal.

4. ¿Cómo saber en cuál rama estoy trabajando?

Para saber en cuál rama estás trabajando actualmente en Git, puedes usar el comando `git branch`. Este comando lista todas las ramas del repositorio, y marcará con un asterisco (*) la rama en la que estás actualmente. También puedes usar `git status`, que indica claramente la rama activa al inicio del resultado.

5. ¿Quién creó Git?

Git fue creado por Linus Torvalds, el mismo creador del núcleo del sistema operativo Linux. Fue desarrollado en 2005 como una alternativa al sistema BitKeeper, con el objetivo de crear un sistema de control de versiones rápido, eficiente y completamente distribuido para manejar el desarrollo del núcleo de Linux.

6. ¿Cuáles son los comandos esenciales de Git?

Los comandos esenciales de Git incluyen:

- `git init`: Inicializa un nuevo repositorio.
- `git clone`: Clona un repositorio existente.
- `git status`: Muestra el estado de los archivos en el repositorio.
- `git add`: Agrega archivos al área de preparación (staging).
- `git commit`: Registra los cambios en el repositorio.
- `git push`: Envía los cambios al repositorio remoto.
- `git pull`: Trae y fusiona cambios desde un repositorio remoto.
- `git branch`: Muestra o gestiona ramas.
- `git merge`: Fusiona ramas.
- `git log`: Muestra el historial de confirmaciones.

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

El desarrollo basado en trunk (Trunk Based Development) es una práctica en la cual todos los desarrolladores trabajan directamente sobre una única rama principal (trunk), haciendo commits frecuentes y pequeños. En lugar de trabajar en ramas largas y separadas, los desarrolladores integran sus cambios continuamente. Esta metodología promueve la colaboración constante, reduce conflictos y acelera la entrega de software, siendo una base clave para las prácticas modernas como DevOps y CI/CD.

7. ¿Qué es Git Flow?

Git Flow es una estrategia de ramificación para Git que define una serie de reglas para manejar el desarrollo de software. Propone ramas específicas para funcionalidades (`feature`), versiones de producción (`release`), mantenimiento (`hotfix`) y desarrollo general (`develop`). Esta metodología organiza el flujo de trabajo del equipo, mejora la calidad del código y facilita la integración continua y entrega continua (CI/CD).

Bibliografía

1. Chacon, S., & Straub, B. (2014). *Pro Git* (2nd ed.). Apress.
2. Git-scm.com. (https://git-scm.com/)
3. Atlassian Git Tutorials. (https://www.atlassian.com/git/tutorials)
4. Vincent Driessen - Git Flow. (https://nvie.com/posts/a-successful-git-branching-model/)