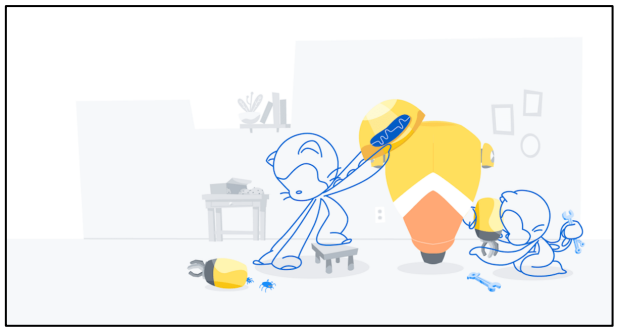
Integrantes: Fernando Huilca, Gregory Salazar, Mateo Simbaña

Curso: GR2SW

Fecha: 8 de noviembre de 2025

Caída de GitHub – Octubre de 2018



1. Resumen del caso

Fuente: https://github.blog/news-insights/company-news/oct21-post-incident-analysis/

El 21 de octubre de 2018, GitHub, una de las plataformas más importantes para el alojamiento de código y proyectos colaborativos, sufrió una caída global que afectó sus principales servicios durante casi 24 horas. Los usuarios no podían realizar operaciones como “push”, “pull”, ni acceder a repositorios o páginas alojadas.

El problema se originó durante una tarea programada de mantenimiento en la infraestructura de red. Durante el reemplazo de un componente principal, se produjo una desconexión momentánea que provocó una desincronización entre las bases de datos de réplica. Esto causó errores en cascada que impactaron directamente la disponibilidad del servicio y la integridad temporal de los datos.

2. Clasificación del mantenimiento

Este caso representa una combinación de **mantenimiento preventivo** y **mantenimiento correctivo**.

Inicialmente, la actividad planificada por el equipo de GitHub era un **mantenimiento preventivo**, dado que consistía en sustituir hardware crítico para evitar fallas futuras. Sin embargo, al generarse una interrupción inesperada en el proceso, fue necesario aplicar **mantenimiento correctivo** para restaurar el funcionamiento del sistema y sincronizar las bases de datos afectadas.

Por lo tanto, se trató de un mantenimiento preventivo que derivó en un correctivo debido a una falla inesperada en el proceso de actualización.

3. Procesos SCM involucrados

La Gestión de Configuración de Software (SCM) fue importante para manejar este incidente.

- **Control de versiones:** El equipo utilizó el mismo sistema de control (Git) para identificar rápidamente los cambios más recientes realizados en la infraestructura y determinar en qué momento exacto se produjo la desincronización. Esto permitió aislar el problema y garantizar que los datos del sistema pudieran recuperarse sin pérdidas.
- **Gestión de cambios:** Se activó un protocolo de emergencia que detuvo cualquier modificación adicional en la red y en los servicios dependientes. Además, se revisaron los procedimientos de despliegue y se documentaron las acciones tomadas para mejorar el flujo de aprobación de cambios futuros.

Gracias a estos procesos, el equipo pudo restaurar gradualmente los servicios sin comprometer la estabilidad del sistema ni la seguridad de la información.

4. Impacto en el Ciclo de Vida del Desarrollo de Software (SDLC)

El incidente afectó principalmente las fases de **implementación, pruebas y mantenimiento**.

Durante la **implementación**, el fallo en la red provocó una interrupción directa en la ejecución de los sistemas de producción. Después, se realizaron **pruebas** para asegurar la consistencia de los datos y verificar que todas las funcionalidades volvieran a operar correctamente. Finalmente, en la fase de **mantenimiento**, se tomaron medidas para prevenir que un evento similar ocurra de nuevo, ajustando protocolos de despliegue y de monitoreo de red.

También fue necesario replanificar algunas tareas y priorizar la recuperación del servicio sobre otras actividades de desarrollo que estaban programadas.

5. Beneficios del SCM

Gracias a un sólido sistema de Gestión de Configuración de Software, GitHub pudo:

- Identificar rápidamente las causas del incidente por al registro detallado de los cambios.
- Recuperar versiones estables del sistema sin pérdida de datos.
- Mantener trazabilidad completa de las acciones ejecutadas durante la crisis.
- Reforzar la documentación y los procedimientos internos para evitar errores similares.

El caso demuestra cómo una correcta aplicación de SCM permite responder de forma organizada ante imprevistos y garantizar la continuidad del servicio en plataformas de alta demanda.