Informe DevOps – Caso TransChile

Alumno: César Herrera González

Introducción y análisis del estado actual

Desarrollo aislado sin repositorio centralizado

El modelo actual en TransChile se basa en desarrollos independientes sin un repositorio común, integrando cambios manualmente una vez al mes mediante FTP, sin validaciones automatizadas.

Problemas y consecuencias del modelo actual

La ausencia de control de versiones, CI/CD, pruebas automatizadas y revisiones de seguridad genera baja calidad, riesgos operacionales, retrasos en entregas y dificulta la mejora continua.

Propuesta de implementación de Git y control de versiones

Recomendación de Github flow

Se recomienda por su simplicidad y enfoque en la colaboración continua. Es ideal para equipos que despliegan con frecuencia, ya que permite trabajar con ramas ligeras para nuevas funcionalidades, integrarlas mediante pull requests y desplegar rápidamente tras la aprobación. Favorece la integración continua, revisiones de código efectivas y un ciclo de entrega ágil

Uso de Git con GitHub

Se propone Git junto con GitHub para gestión de repositorios, colaboración y CI/CD, usando Pull Requests y revisiones para asegurar calidad.

Implementación de integración continua (CI/CD)

Elección entre GitHub Actions

GitHub Actions permiten automatizar construcción, pruebas y despliegue, integrándose nativamente en sus plataformas respectivas.

Diferencias clave entre las herramientas

GitLab CI/CD soporta despliegues avanzados y mejor integración de seguridad, mientras GitHub Actions ofrece un amplio ecosistema de acciones reutilizables.

Pipeline de despliegue automático a staging

Incluye build, pruebas unitarias, análisis estático y despliegue automático a staging, con despliegue a producción manual tras aprobación.

Seguridad y análisis estático del código

SonarQube para vulnerabilidades

SonarQube es una plataforma de código abierto que realiza análisis estático para detectar fallos y vulnerabilidades en el código. Se integra con CI/CD para automatizar revisiones y mejorar la calidad y seguridad del software.

Auditoría de dependencias

Las auditorías de dependencias usan herramientas como Dependabot y Snyk para identificar paquetes inseguros y vulnerabilidades, integrándose en pipelines de desarrollo para mantener la seguridad y la integridad del software.

Beneficios esperados tras la implementación

Mejora en la calidad del código

La implementación de DevOps con control de versiones y CI/CD mejora la calidad del código mediante automatización de pruebas, revisiones obligatorias y análisis estático con SonarQube, asegurando código limpio y mantenible.

Incremento en la seguridad del software

La seguridad se fortalece con auditorías automáticas y manuales, escaneo de dependencias y políticas estrictas de revisión y control de accesos, previniendo vulnerabilidades y aumentando la confiabilidad del software.

Plan de adopción y capacitación

Capacitación en Git y CI/CD

La formación abarca desde conceptos básicos hasta avanzados de Git, incluyendo gestión de ramas, fusiones y conflictos, además del uso de plataformas como GitHub y GitLab para control de versiones y revisiones de código.

Automatización y calidad en CI/CD

Se enseña el flujo completo del pipeline CI/CD, desde commits hasta despliegues, utilizando herramientas como GitLab CI/CD y GitHub Actions para automatizar pruebas, análisis estático con SonarQube y despliegues seguros.

Implementación gradual para adopción efectiva

La adopción se realiza en fases: primero control de versiones, luego pipelines de integración continua, y finalmente análisis estático y auditoría de dependencias, facilitando la adaptación y promoviendo mejores prácticas DevOps.

Consideraciones finales y recomendaciones

Disciplina en el uso de ramas

La implementación efectiva de Git requiere disciplina en el uso de ramas, recomendando desarrollo basado en trunk con ramas cortas para features, facilitando integración continua y ciclos de desarrollo cortos.

Revisión de código y colaboración

Es fundamental respetar prácticas de revisión mediante pull requests y revisiones entre pares para asegurar calidad, detectar errores y fomentar aprendizaje y colaboración en el equipo.

Promoción de cultura DevOps

Fomentar una cultura DevOps que integre desarrollo, operaciones y seguridad, con automatización, integración continua, análisis estático y políticas claras para mejorar calidad, seguridad y confiabilidad.

Conclusión

Clave para superar problemas actuales

La adopción de prácticas DevOps y herramientas como Git, GitHub Flow, CI/CD y SonarQube es esencial para resolver los problemas de desarrollo aislado, baja calidad del software y falta de automatización. GitHub Flow, en particular, permite una colaboración ágil, integración continua y despliegues más seguros, facilitando un ciclo de desarrollo moderno y eficiente.

Avance hacia un desarrollo más ágil y seguro

Implementar Trunk-based Development y automatización con GitLab CI/CD o GitHub Actions mejora la integración continua, la seguridad y la entrega rápida y confiable del software.

Muchas Gracias!

Contacto: cesarherr321@gmail.com