UNIVERSIDAD TECNOLOGICA NACIONAL
FACULTAD REGIONAL CORDOBA

GRUPO 10:
82603 - MALDONADO, FRANCISCO
86495 - LYPNIK, ZOI
89398 - SISCA, TOMAS NAZARENO
90230 - OCAMPO VYSIN, ARIAN JAVIER
90317 - BLENCIO, JUAN MATEO
91140 - SALUM, MAURICIO
90023 - CARRERAS, NICOLÁS

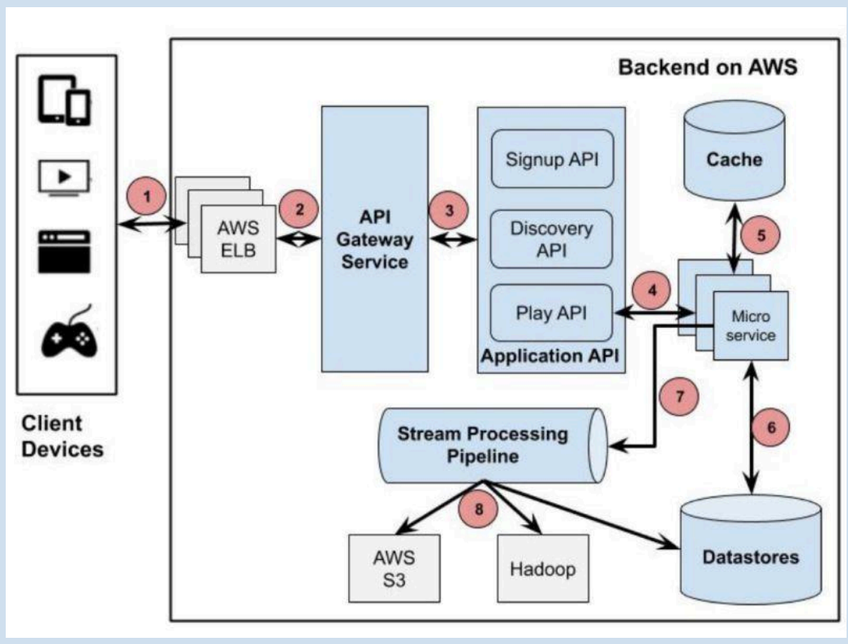
GRUPO 10 - 4K2
INGENIERIA Y CALIDAD DE SOFTWARE

DESPLIEGUE DE MICROSERVICIOS DE ALTO TRÁFICO 24/7

INTRODUCCIÓN

El escenario propuesto es una plataforma de streaming 24/7. Su arquitectura de microservicios, de alta interdependencia y complejidad, presenta el desafío de aplicar cambios y nuevas funcionalidades de forma gradual y sin generar caídas, asegurando la compatibilidad de los contratos de API.

A continuación, presentamos un plan de despliegue que minimiza el riesgo, se alinea con prácticas modernas y garantiza la continuidad del servicio. Para ello, seleccionamos y justificamos la estrategia de Canary Release como solución óptima.



MATERIALES Y MÉTODOS

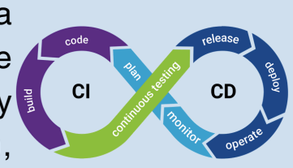
SISTEMA DE CONTROL DE VERSIONES

GitHub

Es la "fuente única de verdad". Todo despliegue es gatillado por un cambio en el SCM

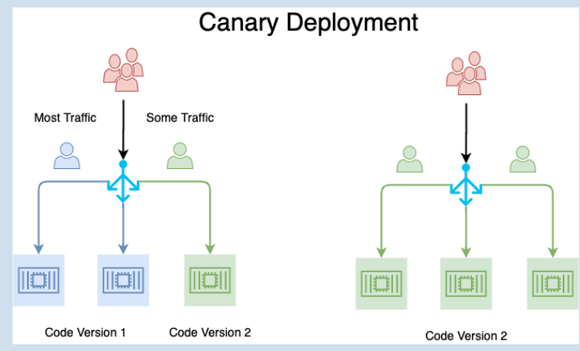
PRÁCTICAS CONTINUAS (CI/CD)

El proceso se automatiza en un pipeline de Integración Continua (CI) y Despliegue Continuo (CD), que prueba y empaqueta el artefacto (imagen Docker)



ESTRATEGIA DE CANARY RELEASE

Se liberan nuevas versiones a un pequeño porcentaje de usuarios, reduciendo el riesgo y permitiendo rollback inmediato



CULTURA DE TRABAJO

Adopción de una cultura donde los equipos son dueños del ciclo de vida completo. Los roles colaboran para automatizar y monitorear el despliegue.

SEGURIDAD

La seguridad se integra en el pipeline (ej. escaneo de vulnerabilidades del artefacto) en lugar de ser un paso final

HERRAMIENTAS

Istio
Jenkins
Prometheus
Docker
Kubernetes

RESULTADOS

Se presenta el plan aplicado al despliegue del microservicio catalogo:v1.2 (el artefacto), coexistiendo con v1.1.

Fase	Etapa (CI/CD)	Estrategia / Herramients Clave	Acción y Criterio de Decisión
1	Preparación (CI)	SCM (Git), Pruebas Unitarias y de Contrato, Docker	El pipeline es gatillado por Git. Se ejecutan prerrequisitos: pruebas unitarias y pruebas de contrato y se asegura la calidad del artefacto. Se construye y escanea de la imagen Docker.
2	Despliegue Canario (CD)	Kubernetes, Istio	Instalación de v1.2 en la infraestructura Kubernetes (entorno destino). Istio dirige solo el 1% del tráfico a v1.2. 99% permanece en v1.1.
3	Monitoreo y Decisión	Prometheus, Métricas (Tasa de Error y Latencia)	Monitoreo en tiempo real comparando v1.2 vs. v1.1. Criterio de Aprobación: Métricas estables, estas son las pruebas post despliegue.
	Plan Rollback	Istio	Rollback Automático: Si falla (supera umbral de las métricas), el tráfico se revierte a 0% en v1.2 (100% en v1.1).
4	Promoción	Istio	Aumento Gradual del Tráfico a v1.2. v1.1 es retirado una vez finalizada la transición.

CONCLUSIONES

En este Plan de Despliegue, resolvemos el requisito de aplicar cambios graduales, permitiendo la evolución de la plataforma de streaming sin caídas de servicio.

El hallazgo clave es que la mitigación de riesgos ya no depende de "ventanas de mantenimiento", sino del monitoreo y la capacidad de rollback instantáneo, lo que transforma el despliegue en un proceso de bajo impacto.

En este enfoque vemos la cultura DevOps, utilizando herramientas de orquestación y control de tráfico para automatizar decisiones basadas en métricas. Así, la estrategia Canary se demuestra como la solución superior para arquitecturas de microservicios complejas y de alta disponibilidad.

REFERENCIAS Y AGRADECIMIENTOS

1. Google (2016). Site Reliability Engineering. O'Reilly.
2. Humble, J. & Farley, D. (2010). Continuous Delivery.
3. Fowler, M. (2010). "Canary Release". martinfowler.com
4. Kubernetes.io. (Documentación oficial)
5. Istio.io. (Documentación oficial)
6. Prometheus.io. (Documentación oficial)