DESPLIEGUE DE MICROSERVICIOS DE ALTO TRÁFICO

Sidorowicz | Pecchio | Ceballos | Ghibaudo | Villegas | Angonese | Wendler | Fernandez | Delgado | Scrosati

SISTEMAS
UTN - FRC

INTRODUCCION

Las **plataformas de streaming** manejan millones de usuarios activos simultáneamente y se basan en arquitecturas de microservicios altamente interconectadas. Cada servicio (catálogo, pagos, recomendaciones, historial, etc.) debe actualizarse sin interrumpir la disponibilidad del sistema.

Desafío: es implementar un plan de despliegue que asegure compatibilidad, escalabilidad y cero downtime, aplicando estrategias DevOps y CI/CD con mecanismos de rollback y monitoreo continuo.

Objetivo: Diseñar un plan de despliegue seguro, automatizado y escalable para microservicios de alto tráfico.

HERRAMIENTAS Y ESTRATEGIAS

Herramientas

- ***Contenedores***: Docker
- ***Orquestación***: Kubernetes
- ***Integración/Entrega continua:*** Jenkins / GitHub Actions
- ***Control de versiones (SCM):*** Git
- ***Seguridad***: OAuth 2.0, HTTPS, escaneo de vulnerabilidades (Trivy)
- ***Monitoreo***: Prometheus + Grafana

Estrategias de despliegue

- ***Rolling Deployment:*** reemplazo gradual de instancias.
- ***Blue/Green Deployment:*** coexistencia de dos versiones (producción y nueva).
- ***Canary Deployment:*** liberación controlada a una parte de los usuarios.
- ***Auto Scaling:*** aumento/disminución automática de recursos según la demanda.

RESULTADOS

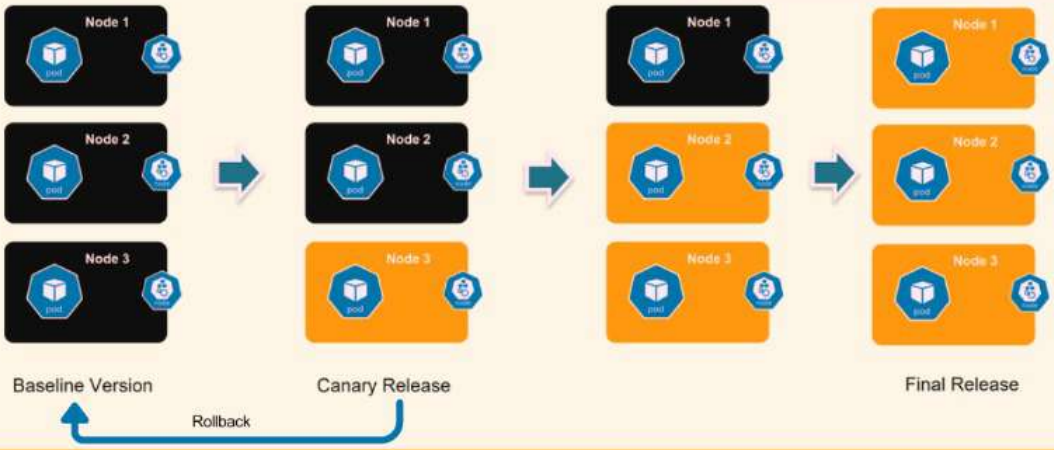
- ***Entorno***: Clúster Kubernetes en la nube con escalado automático.
- ***Tipo***: Canary deployment
- ***Infraestructura***: Contenedores Docker versionados.
- ***Dependencias***: Verificación previa y backups automáticos
- ***Horario***: Ejecución en horas de bajo tráfico.
- ***Automatización***: Pipeline CI/CD con pruebas y monitoreo.
- ***Monitoreo***: Prometheus y Grafana en tiempo real.
- ***Aprobación***: QA y DevOps validan cada release.
- ***Validación***: Smoke tests post-despliegue.
- ***Contingencia***: Rollback inmediato ante errores.

CONCLUSIONES

El **despliegue de microservicios de alto tráfico** requiere una arquitectura resiliente, automatización completa y observabilidad continua.

La combinación de DevOps, CI/CD y Kubernetes permite lograr actualizaciones sin downtime, **garantizando** compatibilidad entre servicios y una rápida recuperación ante fallos.

La seguridad, el monitoreo y las estrategias progresivas (Canary deployment) consolidan un **flujo de despliegue confiable y sostenible** en el tiempo.



REFERENCIAS

- Newman, S. (2015). Building Microservices. O'Reilly Media.
- Dragoni, N. et al. (2017). Microservices: Yesterday, Today, and Tomorrow.
- Fowler, M. (2019). Microservices Resource Guide.
- Documentation: kubernetes.io, aws.amazon.com, istio.io