Lectura 30/07/2025

Lectura Chapter 1:

¿Qué es DevOps y por qué es importante?
DevOps es un conjunto de prácticas cuyo objetivo principal es **reducir el tiempo entre la realización de un cambio en un sistema y su puesta en producción normal**, al tiempo que se **garantiza una alta calidad**. Si estás involucrado en la construcción de sistemas de software y tu organización busca reducir el tiempo de comercialización de nuevas características, DevOps es relevante. Influye en la forma en que se organizan los equipos, se construyen los sistemas y la estructura de los mismos.

Motivación de DevOps:
DevOps surge como respuesta al problema de las **liberaciones lentas** (releases lentos). Cuanto más tiempo tarda una liberación en llegar al mercado, menos ventajas se obtienen de sus nuevas características o mejoras de calidad. El objetivo es liberar de manera continua, lo que a menudo se denomina "entrega continua" o "despliegue continuo". Los despliegues incorrectos pueden tener costos financieros sustanciales, como demuestran ejemplos como Knight Capital y Goldman Sachs. Las encuestas indican que la velocidad y calidad de los despliegues son una preocupación para muchas empresas.

Barreras a las liberaciones lentas:
Tradicionalmente, el proceso de liberación implica pasos manuales y una gran coordinación, lo que lo hace laborioso, lento y propenso a errores. Las principales razones para esto incluyen:

- **Mala coordinación entre desarrollo y operaciones:** A menudo, los desarrolladores tienen la actitud de "terminé el desarrollo, ahora ve y ejecútalo", lo que lleva a la falta de comunicación sobre compatibilidades, recursos necesarios, capacitación del personal de soporte, etc..
- **Capacidad limitada del personal de operaciones:** El personal de operaciones tiene una amplia gama de responsabilidades complejas, y encontrar y retener personal calificado es un desafío.

La perspectiva DevOps: Dos temas clave
La respuesta de DevOps a estos problemas se basa en dos temas consistentes:

1. **Automatización:**
    - Las herramientas pueden realizar acciones, verificar la validez, informar errores y mantener un historial para control de calidad y auditoría.
    - Permiten **aplicar políticas organizacionales** (ej., requerir una justificación para cada cambio).
    - Las herramientas y sus archivos de configuración deben tratarse con el mismo control de calidad que el código de la aplicación, lo que se denomina "**infraestructura como código**".
2. **Responsabilidades del equipo de desarrollo:**
    - Si el equipo de desarrollo asume las responsabilidades de entrega, soporte y mantenimiento del servicio, **se reduce la necesidad de transferir conocimiento** al personal de operaciones, acortando así el tiempo de despliegue.

Prácticas DevOps:
El texto identifica cinco categorías de prácticas DevOps que cumplen con la definición orientada a objetivos:

- **Tratar a Ops como ciudadanos de primera clase en los requisitos:** Incorporar las necesidades de operaciones (ej., logging y monitoreo) desde el inicio del desarrollo.
- **Hacer a Dev más responsable del manejo de incidentes:** Acortar el tiempo entre la observación y la reparación de un error, a menudo asignando la responsabilidad primaria a Dev para nuevos despliegues.
- **Hacer cumplir el proceso de despliegue utilizado por todos:** Asegurar despliegues de mayor calidad y evitar errores por procesos ad hoc, facilitando el rastreo del historial de un artefacto.
- **Usar el despliegue continuo:** Acortar el tiempo desde que un desarrollador comete código hasta que se despliega, enfatizando las pruebas automatizadas para la calidad.
- **Desarrollar código de infraestructura (scripts de despliegue) con las mismas prácticas que el código de aplicación:** Garantizar alta calidad en

las aplicaciones desplegadas y que los despliegues se realicen según lo planeado, aplicando control de calidad.

DevOps y Agile:
DevOps complementa las prácticas ágiles, impactando las tres fases de Disciplined Agile Delivery (Incepción, Construcción y Transición/Despliegue):

- **Incepción:** Considera los requisitos de Ops y la coordinación de la planificación de la liberación.
- **Construcción:** Se enfoca en la gestión de ramas de código, integración y despliegue continuo, y pruebas automatizadas.
- **Transición (Despliegue):** El equipo de desarrollo es responsable del despliegue, monitoreo, decisiones de rollback y monitoreo de la ejecución post-despliegue, actuando como "ingeniero de fiabilidad".

Estructura del equipo en DevOps:

- **Tamaño del equipo:** Los equipos deben ser relativamente pequeños (ej., regla de las "dos pizzas" de Amazon) para tomar decisiones rápidamente, formar unidades coherentes y facilitar la expresión de opiniones. Un equipo pequeño impulsa una arquitectura basada en microservicios para manejar tareas más grandes.
- **Roles del equipo (además de Team Lead y Team Member):**
  - **Service Owner:** Responsable de la coordinación externa, prioriza el trabajo y comunica la visión del servicio.
  - **Reliability Engineer (Ingeniero de Fiabilidad):** Monitorea el servicio post-despliegue (ej., usando canaries), es el punto de contacto para problemas en ejecución, realiza análisis a corto y largo plazo (ej., "5 Whys" para la causa raíz) y automatiza tareas repetitivas.
  - **Gatekeeper (Portero):** Decide si una versión de un servicio avanza al siguiente paso en el pipeline de despliegue, basándose en resultados de pruebas y listas de verificación.
  - **DevOps Engineer (Ingeniero DevOps):** Responsable del "cuidado y alimentación" de las herramientas utilizadas en la cadena de herramientas DevOps (ej., gestión de configuración, CI/CD), incluyendo su configuración y monitoreo de uso.

Coordinación en DevOps:
Uno de los objetivos de DevOps es
minimizar la coordinación para reducir el tiempo de comercialización.

- Las formas de coordinación varían (directa/indirecta, persistente/efímera, síncrona/asíncrona).
- Las **herramientas DevOps incorporan mecanismos de coordinación** (ej., sistemas de control de versiones evitan sobrescrituras, CI coordina pruebas).
- Demasiada comunicación y colaboración manual, especialmente síncrona, puede ir en contra del objetivo de DevOps.
- Se busca minimizar la **coordinación entre equipos**, que es el factor más lento en el proceso de liberación.
- La arquitectura del software ayuda a que las piezas de código funcionen juntas.
- DevOps argumenta que la duplicación de esfuerzo es un costo necesario para un tiempo de comercialización más corto, especialmente cuando las tareas del equipo son pequeñas.

Barreras para la adopción de DevOps:
A pesar de los beneficios, existen obstáculos significativos:

- **Cultura y tipo de organización:** La aversión al riesgo varía según el dominio (ej., regulados como finanzas vs. ágiles como análisis de negocios). Las organizaciones con costos altos por tiempo de inactividad o largos plazos de entrega pueden ser reacias a adoptar DevOps rápidamente. La adopción debe considerar las prácticas implícitas y el impacto en la cultura organizacional.
- **Tipo de departamento:** Desarrolladores están incentivados a cambiar (liberar código), mientras que operaciones está incentivado a minimizar el tiempo de inactividad y resistir el cambio, lo que crea choques culturales.
- **Mentalidad de silo:** La lealtad a los equipos individuales puede dificultar la colaboración y la adopción de objetivos comunes a nivel organizacional.
- **Soporte de herramientas:** Requiere experiencia en instalación, configuración y uso de herramientas, así como definir y hacer cumplir procesos comunes a través de las herramientas.

- **Problemas de personal:** Mover tareas de operaciones a desarrollo puede aumentar los costos de personal debido a los salarios más altos de los ingenieros de software, lo que requiere que la automatización genere ahorros de tiempo significativos. La alta demanda de desarrolladores también puede dificultar la adición de nuevas tareas.

Conclusiones y compensaciones:
DevOps busca
reducir el tiempo desde la concepción de una característica hasta su despliegue. Implica varias **compensaciones** (tradeoffs):

- Necesidad de **soporte para herramientas DevOps** versus el acortamiento del tiempo de comercialización.
- **Mover responsabilidades de IT/Ops a desarrolladores:** Considera el costo y el tiempo de la tarea por cada grupo, la disponibilidad de personal y el tiempo de reparación de errores (los desarrolladores tienen el contexto inmediato).
- **Eliminar la supervisión de nuevas características y despliegues** versus la autonomía de los equipos de desarrollo. La eficiencia de los equipos autónomos debe superar la duplicación de esfuerzos.

DevOps tiene el potencial de llevar a TI a un nuevo terreno, con alta frecuencia de innovación y ciclos rápidos para mejorar la experiencia del usuario.

Lectura Chapters 2 y 3:

Capítulo 2: Capacidades de DevOps
El Capítulo 2 explora las
capacidades que componen DevOps, las cuales abarcan todo el ciclo de vida de la entrega de software. La adopción de DevOps por parte de una organización depende de sus objetivos de negocio, los desafíos que busca abordar y las deficiencias en sus capacidades de entrega de software. La lectura propone una **arquitectura de referencia de DevOps** con cuatro conjuntos de rutas de adopción: **Steer, Develop/Test, Deploy y Operate**.

- **Steer (Dirigir)**: Esta ruta se enfoca en la **planificación de negocio continua**, estableciendo y ajustando objetivos empresariales basándose en el

feedback del cliente. Busca conciliar la necesidad de agilidad con la confianza en que lo que se entrega es lo correcto, gestionando costos e identificando el desperdicio.

- **Develop/Test (Desarrollo/Pruebas)**: Implica el **desarrollo colaborativo** y las **pruebas continuas**.
    - El desarrollo colaborativo permite que equipos multifuncionales (negocio, analistas, arquitectos, desarrolladores, QA, operaciones, seguridad, proveedores, etc.) trabajen juntos en una plataforma común, incluyendo la **integración continua** (CI). La CI consiste en que los desarrolladores integren su trabajo de forma continua o frecuente con el de otros, descubriendo riesgos tempranamente.
    - Las pruebas continuas significan **probar antes y de forma continua** a lo largo del ciclo de vida, lo que se conoce como "shift-left testing". Esto reduce costos, acorta ciclos de prueba y proporciona feedback continuo sobre la calidad. Se facilita mediante **pruebas automatizadas y virtualización de servicios**.
- **Deploy (Despliegue)**: Esta ruta es donde se originaron la mayoría de las capacidades centrales de DevOps. La **entrega y el despliegue continuos** extienden el concepto de integración continua, facilitando el despliegue continuo de software a QA y luego a producción de manera eficiente y automatizada. El objetivo es liberar nuevas características a los clientes lo antes posible.
- **Operate (Operación)**: Incluye el **monitoreo continuo** y la **retroalimentación continua del cliente y optimización**.
    - El monitoreo continuo proporciona datos y métricas a todas las partes interesadas en diferentes etapas del ciclo de entrega, no solo en producción, para reaccionar y mejorar las características o planes de negocio.
    - La retroalimentación continua del cliente implica capturar el comportamiento y los puntos débiles de los clientes para mejorar las aplicaciones y la experiencia del cliente. Este bucle de retroalimentación es esencial para que las empresas sean más ágiles y receptivas.

Capítulo 3: Adopción de DevOps
La adopción de DevOps requiere un plan que abarque

personas, procesos y tecnología. DevOps es una capacidad empresarial que involucra a todas las partes interesadas, desde los propietarios de negocios hasta operaciones y seguridad.

- **Personas en DevOps**:
  - En su raíz, DevOps es un **movimiento cultural** que se centra en las personas. Una **cultura DevOps** se caracteriza por un alto grado de colaboración, enfoque en objetivos de negocio (no departamentales), confianza y valor en el aprendizaje mediante experimentación.
  - Requiere que los líderes trabajen con sus equipos para fomentar la **colaboración y el intercambio**, eliminando barreras autoimpuestas y reemplazando incentivos individuales conflictivos con la responsabilidad compartida de entregar capacidades de forma rápida y segura.
  - La discusión sobre tener un **equipo DevOps separado** es continua; puede funcionar como un centro de excelencia para facilitar la colaboración, pero debe evitar añadir burocracia o asumir todos los problemas relacionados con DevOps.
- **Proceso en DevOps**: Los procesos definen lo que hacen las personas.
  - **DevOps como proceso de negocio** afecta a toda la empresa, haciéndola más ágil en la entrega de capacidades desde la idea hasta la producción.
  - La **gestión de cambios** es inherente al flujo del proceso DevOps, controlando, gestionando y rastreando los cambios, y debe incluir la gestión de elementos de trabajo, flujos de trabajo configurables y control de acceso basado en roles.
  - Las **técnicas DevOps específicas** incluyen:
    - **Mejora continua**: Un proceso continuo de identificación de áreas de mejora, a menudo a través de "retrospectivas" para aprender de lo que salió bien y mal.
    - **Planificación de la liberación**: Función de negocio crítica que impulsa las hojas de ruta y los cronogramas de entrega, buscando lanzamientos más pequeños y frecuentes.
    - **Integración continua**: Reduce el riesgo e identifica problemas temprano al integrar y validar continuamente el trabajo de los

desarrolladores.

        - **Entrega continua**: Automatiza el despliegue de software a entornos de prueba, staging y producción, y es considerada por muchos como la parte más crítica de DevOps.
        - **Pruebas continuas**: Requiere procesos para el aprovisionamiento y configuración de entornos de prueba, gestión de datos de prueba y diferentes tipos de pruebas (integración, función, rendimiento, seguridad). La **virtualización de servicios** es clave para simular componentes no disponibles.
        - **Monitoreo y retroalimentación continuos**: Procesos para absorber feedback de diversas fuentes (clientes, herramientas de monitoreo) e incorporarlo en planes de entrega de software.
- **Tecnología en DevOps**: La tecnología permite a las personas enfocarse en el trabajo creativo de alto valor, delegando tareas rutinarias a la automatización y escalando las habilidades de los equipos.
    - **Infraestructura como código**: Capacidad central de DevOps para gestionar la escala y velocidad del aprovisionamiento y configuración de entornos. Evoluciona hacia **entornos definidos por software** que definen sistemas completos (nodos, topologías, roles, etc.). Existen herramientas especializadas, de entorno y despliegue, y genéricas para su gestión.
    - **Pipeline de entrega**: Consiste en las etapas por las que pasa una aplicación desde el desarrollo hasta la producción. Incluye:
        - **Entorno de desarrollo**: Herramientas para escribir y probar código, gestión de control de código fuente, gestión de elementos de trabajo, pruebas unitarias.
        - **Etapa de construcción (build)**: Compila el código, crea y prueba binarios.
        - **Repositorio de paquetes**: Almacenamiento común para binarios y activos asociados (archivos de configuración, scripts).
        - **Entorno de pruebas**: Donde los equipos de QA realizan pruebas, con herramientas para gestión de entornos y datos de prueba, y varios tipos de pruebas.

        - **Entornos de staging y producción**: Donde se despliegan las aplicaciones, con herramientas de gestión de entorno y monitoreo.
    - **Automatización del despliegue y gestión de liberaciones**: Herramientas especializadas son esenciales para orquestar y rastrear despliegues, gestionar configuraciones y coordinar planes de liberación con todas las partes interesadas.