Ejercicio Práctico 1: Evaluación de Conceptos DevOps y su Aplicación en Proyectos

Lorena Soto San Martin

Parte 1: Preguntas teóricas

1. Fundamentos de DevOps

- ¿Qué es DevOps y cuál es su propósito principal?

El termino DevOps nace como el acrónimo de los términos en ingles development (desarrollo) y operations (operaciones) y hace referencia a la metodología de desarrollo de software que combina, integra y automatiza el trabajo de los equipos de desarrollo de software y operaciones de TI, con el objetivo de acelerar la entrega de aplicaciones y servicios de mayor calidad. Este movimiento surge en el año 2007 como una nueva propuesta al modelo tradicional en donde estas dos áreas trabajaban por separado lo cual, generaba una mayor cantidad de errores debido a la falta de coordinación entre ambos equipos y retrasaba los tiempos de entrega de los sistemas.

- Explica el modelo CAMS y su importancia en la cultura DevOps.

El modelo CAMS es un marco que describe los principios fundamentales que deben ser implementados en las prácticas de DevOps, su importancia radica en que la incorporación de estos principios en las organizaciones permite mejorar la colaboración, la automatización y la entrega continua de software de alta calidad.

CAMS significa Cultura, Automatización, Medición y Compartir y son los valores fundamentales de este modelo, los cuales se describen a continuación:

- **Cultura:** Cambiar la cultura empresarial y lograr que los equipos de desarrollado y operaciones estén en sintonía trabajando en conjunto para alcanzar en objetivos comunes es uno de los objetivos principales en DevOps.
- **Automatización:** Debido a la velocidad y la alta tasa de cambio, la automatización resulta muy útil y es parte fundamental de la cultura DevOps. La automatización ahorra tiempo y mejora la eficiencia, como la integración continua/distribución continua (CI/CD) y la entrega continua (IaaS).
- **Medición:** Para mejorar la medición es fundamental. En DevOps Se utilizan métricas para identificar áreas de mejora y monitorear el progreso, esto permite mejorar la toma de decisiones ya que se estará basada en datos concretos.
- **Compartir:** Los procesos DevOps, al igual que Agile y Scrum, priorizan la transparencia y la apertura. Difundir el conocimiento ayuda a fortalecer los ciclos de retroalimentación y permite a la organización mejorar continuamente. Esta inteligencia colectiva convierte al equipo en una unidad más eficiente y le permite ser más que la suma de sus partes.

2. Integración y Entrega Continua

- ¿Cuál es la diferencia entre Integración Continua y Entrega Continua?

La integración continua y la entrega continua son practicas fundamentales en los procesos DevOps, pero cuentan con objetivos y procesos diferentes. La integración continua se puede describir como una práctica de desarrollo de software que consiste en integrar cambios de código en un repositorio múltiples veces al día de forma automatizada. En cambio, la integración continua e la práctica posterior a la integración continua y consiste en preparar automáticamente los cambios en el código y se entregan a la fase de producción, esto permite que el código siempre este en un estado listo para ser desplegado en producción, luego el proceso de despliegue es automatizado.

Ambas prácticas se complementan, y juntas ayudan a mejorar la calidad, eficiencia y velocidad del desarrollo de software.

- ¿Qué beneficios aporta la Integración Continua al proceso de desarrollo de software?

La integración continua es una de principales prácticas recomendadas de DevOps, ya que permite a los desarrolladores fusionar con frecuencia los cambios de código en un repositorio central donde luego se ejecutan las compilaciones y pruebas, lo cual permite validar el nuevo código antes de la integración. Esto permite que la integración continua tenga múltiples beneficios, los cuales se detallan a continuación:

- Mejora en la productividad de desarrollo
- Mayor rapidez en la detección de errores
- Reducción de tiempos de integración
- Facilita la gestión de versión
- Facilita la integración con otras prácticas como el desarrollo iterativo y la entrega continua.

3. Contenedores y Docker

- ¿Qué es un contenedor y en qué se diferencia de una máquina virtual?

Un contenedor es un tipo de tecnología que permite agrupar el código de una aplicación con las bibliotecas y los archivos de configuración asociados, junto con las dependencias necesarias para que la aplicación se ejecute. Esto permite a los desarrolladores y profesionales de TI implementar aplicaciones sin problemas en todos los entornos. A diferencia de las máquinas virtuales que virtualizan el hardware subyacente para que se puedan ejecutar varias instancias de sistemas operativos (SO) en el hardware un contenedor contiene solo la aplicación y sus dependencias necesarias para ejecutarse, pero no incluye un sistema operativo completo. Esto los hace más eficientes, más rápidos de arrancar y utilizar una menor cantidad de recursos.

Sin embargo, a pesar de estas considerables diferencias, ambos son útiles y presentan ventajas de acuerdo al contexto en que se aplican, siendo los contenedores más apropiados para aplicaciones web, microservicios o servicios en la nube y las maquinas

virtuales para aplicaciones tradicionales que requieran para su implementación un sistema operativo completo.

- ¿Cuáles son los beneficios del uso de Docker en entornos DevOps?

Docker es una herramienta que permite empaquetar aplicaciones junto con todas sus dependencias en contenedores, su uso, otorga múltiples beneficios ya que facilita la distribución y ejecución de imágenes de aplicaciones en diferentes entornos y permite asegurar que las aplicaciones se ejecuten de manera consistente en todos los entornos, lo cual, reduce el tiempo y los esfuerzos necesarios para su implementación. Además, Docker permite automatizar la creación y el despliegue de contenedores, reduciendo tiempos de implementación. Entre sus otros beneficios se encuentra la reducción de conflictos de dependencias, por el aislamiento con el que contara la aplicación y la simplificación en la administración de los entornos.

4. Pruebas y Automatización en CI/CD

- ¿Cuáles son los tipos de pruebas más importantes en un pipeline de CI/CD?

Los pipelines permiten automatizar la integración y pruebas continuas en una aplicación. Dentro de las pruebas más importantes y que se recomienda realizar en un proceso CI/CD se encuentran:

- **Pruebas unitarias:** Aquellas que verifican que cada unidad del código (clases, funciones, entre otros) funcione correctamente.
- **Pruebas de integración:** Aquellas que asegurar que los diferentes módulos o servicios del código funcionen correctamente al interactuar entre sí.
- **Pruebas funcionales:** Permiten asegurar que las funcionalidades clave del sistema estén operando según lo esperado por el usuario.
- **Pruebas de aceptación:** Verifica que los usuarios puedan completar un flujo correctamente.
- **Pruebas de rendimiento:** La aplicación cumple con los requisitos de rendimiento, por ejemplo, tiempos de respuesta esperados.
- **Pruebas de resiliencia:** Estas pruebas inyectan fallas en los entornos para identificar las áreas de riesgo.
- **Prueba de seguridad:** Estas pruebas identifican vulnerabilidades o infracciones de seguridad en el código, lo cual, ayuda a prevenir riesgos y asegurar la protección contra ataques cibernéticos.

Cabe mencionar que no siempre se ejecutan todas estas pruebas, sin embargo, como mínimo se deberían ejecutar pruebas unitarias y de seguridad en el código, así como pruebas de integración y aceptación en un entorno de pruebas.

- Explica en qué consiste el desarrollo guiado por pruebas (TDD) y su impacto en CI/CD.

TDD es una metodología en el desarrollo de software que consiste en escribir primero las pruebas, después escribir el código fuente que pase la prueba satisfactoriamente y, por último, refactorizar el código escrito, el objetivo es garantizar que el código cumpla

con los requisitos y sea probado de manera continua durante el desarrollo, lo cual, permite reducir errores y mejorar la fiabilidad del software.

TDD impacta de manera significativa en los flujos de trabajo que utilizan CI/CD, y es una buena práctica utilizar ambos en conjunto ya que permite un ciclo de desarrollo ágil, con validación constante, detección temprana de errores, y despliegues más rápidos y estables, ya que mientras TDD se centra en garantizar que el código sea probado desde el principio, CI/CD automatiza la integración y despliegue del código para que sea validado constantemente y entregado sin inconvenientes.

5. Infraestructura y Monitoreo en DevOps

- ¿Qué es Infraestructura como Código (IaC) y qué ventajas ofrece?

La IaC es una práctica en la gestión de infraestructura TI que codifica los recursos de infraestructura de una organización en archivos de texto, en lugar de realizar configuraciones manuales. Estos archivos de infraestructura se envían a un sistema de control de versiones el cual permite los flujos de trabajo de ramas de función y de solicitudes de extracción, que son dependencias fundamentales de la CI y la CD.

IaC proporciona múltiples ventajas, ya que proporciona una manera eficiente y confiable de gestionar la infraestructura TI de forma automatizada, eliminando gran parte del trabajo manual, lo cual implica una disminución en los costos, tiempos de implementación y una reducción en la posibilidad de que se cometan errores humanos. Además, fomenta la consistencia, la transparencia y la facilidad de administración en entornos de infraestructura a gran escala.

- ¿Por qué es importante el monitoreo en DevOps? Menciona al menos dos herramientas de monitoreo utilizadas en entornos CI/CD.

La monitorización en DevOps se refiere al proceso de supervisar, recopilar y analizar datos sobre el estado y el rendimiento de sistemas, aplicaciones y la infraestructura. Este proceso es indispensable en materia de observabilidad ya que proporciona a las organizaciones la información necesaria para gestionar y operar su infraestructura de TI y sus aplicaciones dentro del dinámico panorama digital. Garantizando que las organizaciones cumplan con sus objetivos estratégicos y mitiguen los riesgos operativos.

Algunas de las herramientas de monitoreo más utilizadas en la actualidad en los entonos CI/CD se encuentran:

- **Prometheus:** Sistema de monitoreo completo de extremo a extremo con administrador de alertas. Entre sus ventajas se encuentra, que no tiene que buscar ninguna integración de terceros para los mecanismos de alerta. Es una herramienta de monitoreo autosuficiente.
- **DataDog:** Servicio de monitoreo de infraestructura basado en SaaS con cientos de integraciones. Permite a los equipos de DevOps controlar los entornos dinámicos de nube, lo cual, facilita la visualización del estado de su infraestructura a un alto nivel por ubicación, aplicación o servicio.

- **ELK Stack (Elasticsearch, Logstash, Kibana):** Conjunto de herramientas para la recopilación, análisis y visualización de logs. Elasticsearch es el motor de búsqueda, Logstash procesa y transporta los logs, y Kibana visualiza los datos de manera gráfica.

6. Orquestación y Kubernetes

- ¿Cuál es el propósito de un orquestador de contenedores como Kubernetes?

Las herramientas de orquestación de contenedores, como Kubernetes, tienen como objetivo simplificar la administración de la infraestructura de contenedores mediante la automatización de todo su ciclo de vida, desde el aprovisionamiento y la programación hasta el despliegue y la eliminación. Antes de que existieran las plataformas de orquestación de contenedores, las organizaciones utilizaban scripting complejo (secuencias de comandos) para administrar el despliegue, la programación y la eliminación de contenedores en varios equipos. El mantenimiento de estos scripts creaba desafíos, como el control de versiones, y la configuración era difícil de escalar. La orquestación de contenedores automatiza y resuelve estas complejidades, eliminando los desafíos asociados con la administración manual.

- Explica cómo Kubernetes facilita la escalabilidad y gestión de aplicaciones en producción.

Kubernetes es una plataforma para la organización de contenedores que automatiza muchos de los procesos manuales involucrados en la implementación, la gestión y el ajuste de las aplicaciones que se alojan allí. Dentro de sus beneficios se encuentran que facilitan considerablemente la escalabilidad de los contenedores y la gestión de aplicaciones en producción, lo cual, se explica a continuación:

- **Escalabilidad:** Kubernetes cuenta con escalabilidad automática, lo que permite a los clústeres ajustar dinámicamente la asignación de recursos a las aplicaciones en función de la demanda del tráfico. Esto significa que los kubernetes pueden aumentar o disminuir automáticamente la capacidad de los recursos para garantizar un rendimiento óptimo de las aplicaciones en momentos de alta carga. Esta característica es una de las principales de Kubernetes y permite optimizar el uso de recursos y mejorar la eficiencia operativa.
- **Gestión de aplicaciones:** Kubernetes facilita la gestión de aplicaciones en producción de varias maneras. Su capacidad para gestionar el despliegue y actualizaciones automáticas, garantizar la alta disponibilidad mediante la replicación de pods y la recuperación ante fallos, facilitar la escalabilidad según la demanda, además, provee mecanismos de gestión de configuraciones y secretos como contraseñas, claves de API o configuraciones específicas del entorno. Estas características convierten a kubernetes en una herramienta clave para mantener aplicaciones estables y seguras en producción.