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

Requisitos:
Parte 1: Preguntas Teóricas (3 puntos)
Responde brevemente las siguientes preguntas relacionadas con los temas vistos en clase:
1. Fundamentos de DevOps (0.5 pts)
¿Qué es DevOps y cuál es su propósito principal?
DevOps es una filosofía de trabajo que combina las prácticas y herramientas de los equipos de desarrollo (Dev) y operaciones (Ops) para mejorar la colaboración, acelerar la entrega de software y automatizar procesos críticos como la integración y el despliegue de aplicaciones. Su propósito principal es reducir el ciclo de desarrollo y garantizar la estabilidad, escalabilidad y calidad del software a través de una entrega continua.

El modelo DevOps es clave en la Integración Continua (CI) y la Entrega Continua (CD), ya que ambos equipos trabajan juntos para implementar cambios de código de manera eficiente, confiable y frecuente. DevOps no es solo una práctica técnica, sino una cultura de colaboración.

Explica el modelo CAMS y su importancia en la cultura DevOps.
El modelo CAMS establece los cuatro pilares esenciales de DevOps:

1. Culture (Cultura): Fomenta la colaboración entre equipos, eliminando los silos.
2. Automation (Automatización): Automatiza tareas manuales repetitivas, como las pruebas y el despliegue.
3. Measurement (Medición): Monitorea y mide continuamente el rendimiento del sistema y los equipos.
4. Sharing (Compartir): Fomenta la transparencia y el intercambio de conocimiento entre los miembros del equipo.

2. Integración y Entrega Continua (0.5 pts)
¿Cuál es la diferencia entre Integración Continua y Entrega Continua?
En Despliegue Continuo y Entrega Continua, ambos conceptos están relacionados con la automatización y la entrega de software, pero tienen diferencias importantes:
- Entrega Continua (Continuous Delivery): El objetivo de la entrega continua es tener el código listo para ser desplegado en producción en cualquier momento. Esto significa que el código pasa por todo el ciclo de pruebas y validación, pero la decisión de cuándo desplegarlo en producción es manual.
- Despliegue Continuo (Continuous Deployment): En el despliegue continuo, cualquier cambio que pase todas las fases de pruebas y validación se despliega automáticamente en producción sin intervención manual. Es una extensión de la entrega continua con un paso automatizado adicional: el despliegue.
¿Qué beneficios aporta la Integración Continua al proceso de desarrollo de software?
La Integración Continua (CI) es una práctica fundamental en el desarrollo de software que aporta varios beneficios clave al proceso de desarrollo. Algunos de los más importantes son:

3. Contenedores y Docker (0.5 pts)

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

Un contenedor de aplicaciones es una unidad estandarizada que empaqueta todo el código de una aplicación y sus dependencias, de forma que se pueda ejecutar de manera consistente en cualquier entorno. A diferencia de los entornos tradicionales, donde las aplicaciones dependen del sistema operativo subyacente, los contenedores incluyen todos los binarios, bibliotecas y configuraciones necesarios para que la aplicación funcione.

Los contenedores son ligeros y rápidos de implementar, ya que comparten el núcleo del sistema operativo subyacente, en lugar de virtualizar todo un sistema operativo como lo hacen las máquinas virtuales.

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

- Portabilidad: Los contenedores Docker permiten ejecutar aplicaciones de manera consistente en diferentes entornos, desde la máquina del desarrollador hasta los servidores de producción.
- Eficiencia: Los contenedores son ligeros y utilizan menos recursos que las máquinas virtuales, ya que no virtualizan todo el sistema operativo.
- Velocidad: Los contenedores se inician en segundos, lo que mejora la eficiencia en los flujos de trabajo de CI/CD.
- Aislamiento: Cada contenedor está aislado, lo que permite ejecutar múltiples aplicaciones en la misma máquina sin conflictos entre ellas.

4. Pruebas y Automatización en CI/CD (0.5 pts)

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

En un **pipeline de CI/CD (Integración Continua / Despliegue Continuo)**, se ejecutan varios tipos de pruebas para garantizar la calidad y el buen funcionamiento del código en cada etapa del proceso de desarrollo y despliegue. Estos son los **tipos de pruebas más importantes** que suelen integrarse en un pipeline de CI/CD:

- *Pruebas unitarias*
- *Pruebas de integración*
- *Pruebas de sistema*
    a) *Funcionales*
    b) *Rendimiento*
- *Pruebas de aceptación*
- *Pruebas de humo*

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

Test-Driven Development (TDD), o desarrollo guiado por pruebas, es una práctica de desarrollo donde los desarrolladores escriben primero una prueba que falla y luego implementan el código necesario para que la prueba pase. Este ciclo se repite hasta que se completa la funcionalidad.

El enfoque de TDD tiene un impacto directo en los procesos de **Integración Continua (CI)** y **Despliegue Continuo (CD)**, dos prácticas fundamentales en el desarrollo de software moderno.

- **Mejora de la calidad del código**
- **Mayor cobertura de pruebas**
- **Integración y despliegue más confiables**
- **Feedback rápido**
- **Reducción de errores en producción**

5. Infraestructura y Monitoreo en DevOps (0.5 pts)

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

La Infraestructura como Código (IaC) permite definir infraestructura en archivos de configuración, facilitando la automatización y escalabilidad.

Beneficios de IaC:

- Despliegues más rápidos y repetibles.
- Evita configuraciones manuales.
- Escalabilidad dinámica.
- Mejora la colaboración entre equipos.

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

El monitoreo continuo es esencial en el ciclo DevOps, ya que proporciona retroalimentación inmediata sobre el rendimiento y estado del sistema después de cada despliegue. Esto asegura que los cambios en el código no introduzcan errores o reduzcan la eficiencia del sistema.

Porque es importante el monitoreo DevOps

- Detectar fallos antes de que afecten a los usuarios.
- Medir el rendimiento de la aplicación.
- Prevenir problemas con alertas tempranas.

6. Orquestación y Kubernetes (0.5 pts)

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

Kubernetes es una plataforma de orquestación de contenedores de código abierto que automatiza la implementación, escalado y gestión de aplicaciones en contenedores. Facilita la distribución automática de cargas de trabajo entre contenedores y asegura que las aplicaciones se mantengan disponibles y escalables.

Explica cómo Kubernetes facilita la escalabilidad y gestión de aplicaciones en producción.
Kubernetes facilita la escalabilidad y la gestión de aplicaciones en producción al automatizar tareas complejas como el escalado, la gestión de recursos, la recuperación ante fallos, y la actualización de aplicaciones, todo ello de manera eficiente y con alta disponibilidad. Esto permite a las organizaciones mantener aplicaciones resilientes y escalables sin tener que gestionar manualmente cada uno de los componentes de infraestructura.

Parte 2: Informe Aplicado a un Proyecto (4 puntos)
Los estudiantes deben redactar un informe en el que expliquen cómo aplicarían los conceptos vistos en un proyecto de desarrollo de software.
El informe debe incluir:
1. Introducción (0.5 pts)
Breve descripción del proyecto en el que se aplicarán los conceptos de DevOps.
Proyecto de un sistema de hospital, en el que los pacientes podrán tomar horas y ver sus fichas medicas, y los doctores podrán ver sus horas y fichas de pacientes.

2. Aplicación de Integración y Entrega Continua (1 pt)
- Explicación de cómo configurarían un pipeline de CI/CD utilizando herramientas como Jenkins, GitLab CI o CircleCI.
- Pasos para integrar pruebas automatizadas en el pipeline.

1. **Definir los Requisitos del Proyecto**
2. **Preparación del Proyecto:** Con el código del sistema hospitalario en un repositorio Git (GitHub).
3. **Despliegue Continuo en Entornos de Pruebas/Staging:** Construir imagen del proyecto con Docker
4. **Integración Continua (CI):** Configurar un servidor de **CI con Jetkins**
5. **Entrega Continua (CD):** Automatizar el proceso de **despliegue** utilizando herramientas de CD como Docker.
6. **Despliegue Continuo en Producción**
7. **Monitoreo y Retroalimentación:** usando herramientas de monitoreo como **Prometheus** y **Grafana** para visualizar el rendimiento y el estado de la aplicación.

3. Uso de Contenedores y Orquestación (1 pt)
-Cómo implementarían Docker en el proyecto.
-Ventajas de utilizar Docker y Kubernetes en el despliegue del sistema.
Implementar Docker en tu proyecto hospitalario te permite gestionar de manera eficiente los entornos de desarrollo, pruebas y producción, y te ayuda a garantizar que tu aplicación funcione de manera consistente sin importar el entorno. Docker también facilita la escalabilidad y el mantenimiento, lo que es esencial en aplicaciones críticas como las de un hospital.
Pasos para implementar Docker

1. **Instalar Docker**
2. Crear un Dockerfile
3. **Crear un archivo requirements.txt**
4. Construir la Imagen Docker
5. **Ejecutar el Contenedor**
6. Docker Compose (Opcional)
7. Pruebas y Despliegue

4. Monitoreo y Seguridad en DevOps (1.5 pts)
-Estrategias para monitorear logs y métricas del sistema.
-Uso de herramientas de monitoreo como ELK Stack o Prometheus para asegurar la estabilidad del proyecto.

Monitorear los **logs** y **métricas del sistema** es crucial para el funcionamiento y la estabilidad de cualquier proyecto, especialmente en un entorno tan crítico como un **sistema hospitalario**.

Algunos de los puntos de la iImportancia de Monitorear Logs y Métricas del Sistema son

1. **Detección Proactiva de Errores y Fallos**
2. **Mejora Continua del Rendimiento**
3. **Garantizar la Seguridad y Cumplimiento**
4. **Prevención de Desastres y Recuperación ante Fallos**
5. **Optimización de Costos y Recursos**
6. **Soporte a la Toma de Decisiones**

Implementar herramientas de monitoreo como **ELK Stack** y **Prometheus** en tu proyecto hospitalario te ayudará a asegurar la estabilidad y el rendimiento de la aplicación, permitiendo detectar y solucionar problemas de manera proactiva. **ELK Stack** es excelente para la gestión de logs y el análisis de errores, mientras que **Prometheus** ofrece monitoreo en tiempo real y alertas sobre el estado del sistema y la infraestructura. Juntas, estas herramientas te brindan visibilidad completa sobre la salud de tu aplicación, lo cual es esencial en entornos críticos como los hospitales.