Explicaciones:

## AWS CI/CD:

Para desplegar la solución en AWS utilizando Elastic Beanstalk y CodePipeline, sigue estos pasos:

1. Preparar el Entorno:

    - Elastic Beanstalk: Levanta una instancia de Elastic Beanstalk configurada para el tipo de aplicación que deseas desplegar en este caso Docker.
    - CodePipeline: Configura un pipeline de CI/CD en AWS CodePipeline que automatizará el proceso de construcción y despliegue de tu aplicación.

2. Configurar CodePipeline:

    - Utiliza el archivo AWS-CI-CD.yml para definir las fases del pipeline, incluyendo la construcción, pruebas y despliegue de la aplicación. Este archivo debe estar en el repositorio de tu aplicación.

3. Desplegar la Solución:

    - Cuando se ejecute el pipeline, el código será construido y desplegado automáticamente en la instancia de Elastic Beanstalk configurada previamente.

## GCP CI/CD:

Para desplegar la solución en Google Cloud Run utilizando Google Cloud Build, sigue estos pasos:

1. Preparar el Entorno:

    - Repositorio: Asegúrate de que el código de tu aplicación, incluyendo el Dockerfile, esté subido a un repositorio (por ejemplo, en GitHub).
    - Trigger de Cloud Build: Crea un trigger en Google Cloud Build que se active cada vez que haya cambios en el repositorio.

2. Configurar Cloud Build:

    - Utiliza el archivo GCP-CI-CD.yaml para definir los pasos necesarios para construir y desplegar tu aplicación en Cloud Run. Este archivo debe estar en el repositorio de tu aplicación.

3. Desplegar la Solución:

    - Al activarse el trigger, Cloud Build ejecutará los pasos definidos en GCP-CI-CD.yaml, construyendo la imagen Docker, empujándola a Google Container Registry y desplegándola en Cloud Run.

## Azure DevOps:

Para desplegar la solución utilizando Azure DevOps, sigue estos pasos:

1. Preparar el Entorno:

    - Conexiones de Servicio: Crea las conexiones de servicio necesarias en Azure DevOps para autenticar y autorizar el acceso a los recursos de Azure.
    - Pipeline: Configura un pipeline en Azure DevOps que se relacione con el repositorio donde está el código de tu aplicación.

2. Configurar el Pipeline:

    - Utiliza el archivo Azure-CI-CD.yml para definir el flujo de CI/CD, que incluye las etapas de construcción, pruebas y despliegue de tu aplicación. Este archivo debe estar en el repositorio de tu aplicación.

3. Desplegar la Solución:

    - Cuando se ejecute el pipeline, el código será construido y desplegado automáticamente en los recursos de Azure configurados (por ejemplo, App Service, AKS, etc.).

# 4. Detallar cómo pondrías en producción la solución (Ejercicio del avance de trabajo final) [5pts]

### 1. Preparar el Entorno de Desarrollo:

- Configuración del Repositorio: Asegurar que el código esté en un repositorio de control de versiones, como GitHub o GitLab.

- Dockerización: Asegurar que haya un Dockerfile bien configurado para facilitar el despliegue en diferentes entornos.

### 2. Automatización del Despliegue (CI/CD):

- AWS Elastic Beanstalk y CodePipeline:

    - Levantar Instancia de Elastic Beanstalk: Crear una instancia de Elastic Beanstalk configurada para la aplicación.

    - Configurar CodePipeline: Configurar CodePipeline utilizando el archivo AWS-CI-CD.yml para automatizar la construcción y despliegue de la aplicación.

- Google Cloud Run y Cloud Build:

- Subir Código a un Repositorio: Subir el código de la aplicación a un repositorio (por ejemplo, GitHub).

- Crear Trigger en Cloud Build: Configurar un trigger en Google Cloud Build que se active con los cambios en el repositorio.

- Configurar Cloud Build: Utilizar el archivo GCP-CI-CD.yaml para definir los pasos necesarios para construir y desplegar la aplicación en Cloud Run.

- Azure DevOps:

  - Crear Conexiones de Servicio: Configurar las conexiones de servicio necesarias en Azure DevOps.

  - Configurar Pipeline: Configurar un pipeline en Azure DevOps y usar Azure-CI-CD.yml para automatizar el despliegue.

3. **Monitoreo y Logging:**

   - Monitoreo del Tiempo de Ejecución:

     - Implementar Herramienta de Monitoreo: Utilizar herramientas como Prometheus o CloudWatch para monitorear el tiempo de ejecución de las funciones de generación y clasificación de imágenes.

     - Añadir Código para Medir Tiempos: Integrar en el código el registro del tiempo de inicio y fin de cada operación.

   - Logging:

     - Configurar Herramienta de Logging: Usar herramientas como ELK Stack, Cloud Logging o Application Insights para almacenar logs detallados de cada ejecución.

4. **Almacenamiento de Imágenes y Clasificaciones:**

   - Almacenamiento de Imágenes:
     - Configurar Servicio de Almacenamiento: Utilizar servicios como Amazon S3, Google Cloud Storage o Azure Blob Storage para almacenar las imágenes.

     - Modificar Código para Subir Imágenes: Ajustar el código para subir las imágenes generadas y subidas a estos servicios.

- Almacenamiento de Clasificaciones:

    - Configurar Base de Datos: Utilizar una base de datos como DynamoDB, Firestore o Cosmos DB para almacenar las clasificaciones y el tiempo de ejecución.

    - Modificar Código para Guardar Clasificaciones: Ajustar el código para guardar las clasificaciones y tiempos de ejecución en la base de datos.

5. **Seguridad y Autenticación:**

    - Gestión de API Keys y Tokens: Almacenar claves y tokens de forma segura usando gestores de secretos como AWS Secrets Manager, Google Secret Manager o Azure Key Vault.

    - Implementar Autenticación y Autorización: Utilizar OAuth, JWT o IAM roles para gestionar la seguridad de la aplicación**.**