Utilizando CI/CD desplegar una aplicación a un appservices ya sea .NET o NODE

Van a desplegar una aplicación a un appservices (Pueden utilizar la organización de esmarlin para crear el appservices)

Dockerización (Utilizar la app de Python)

- Crear un Dockerfile que genere una imagen lo más ligera posible y correr esta imagen en un contenedor.
  Dockerizar la aplicacion de Python que les envio

SonarQube

- Levantar una instancia local de SonarQube (en Minikube)
- Configurar un análisis de calidad en el pipeline:
    - Hacer un análisis de tu repositorio.

RabbitMQ

- Desplegar una instancia de RabbitMQ (utilizando Helm en su cluster de minikube).
- Crear un vhost llamado devops_vhost.
- Crear un usuario devops_user con una contraseña segura, y asignarle permisos completos sobre devops_vhost.
- RabbitMQ debe estar expuesto localmente en su pc.

Kubernetes

- Desplegar la aplicación en Kubernetes:
    - Crear un Deployment con al menos 2 réplicas.
    - Crear un Service tipo ClusterIP para exponer la aplicación dentro del cluster.
    - Utilizar ConfigMaps/Secrets para inyectar las variables de entorno necesarias (ejemplo: credenciales de RabbitMQ, URL de SonarQube, etc.).
    - Configurar un HPA a su Deployment
    - Asignarle recursos a su deployments y que este en QoS Class Guaranteed.

CI/CD (Pipelines Template)

- Configurar un pipeline de CI/CD con la herramienta que prefieras (Utilizando Azure DevOps):
    - **Fase de Build y Test: (Pueden simiular los test con prints)**
        - Ejecutar los tests unitarios de la aplicación.
        - Si fallan, el pipeline se detiene. (Usar condiciones)

- **Fase de Análisis SonarQube:**
    - Ejecutar el SonarQube Scanner.
- **Fase de Build de la Imagen Docker:**
    - Construir la imagen Docker y subirla a un registro privado o público (Docker Hub).

Los pipelines deben ser manejados como templates, variables y parámetros. Y deben ser llamados.