1. Diagrama de Arquitectura del Pipeline GreenDelivery

El flujo de datos sigue un camino claro: **Edge $\rightarrow$ Ingesta $\rightarrow$ Procesamiento $\rightarrow$ Almacenamiento $\rightarrow$ Decisión**.

1. **Edge (Python/Simulador):** Un *script* de **Python** genera datos de telemetría (temperatura, fuerza g, ubicación) y los publica.
2. **Broker de Mensajes (MQTT/Docker):** El simulador envía los datos al **Broker MQTT** (ej. Mosquitto, contenido en **Docker**), que desacopla al publicador del suscriptor.
3. **Motor de Ingesta (Node-RED):** Este motor se suscribe al *Broker*. Su función es validar la **Integridad (I)** del dato y ejecutar la lógica de **Reintentos con Backoff** para garantizar la **Disponibilidad (A)**.
4. **API de Ingesta (FastAPI/Docker):** El motor llama a esta API (contenida en **Docker**) para servir como una capa de seguridad y abstracción para la base de datos.
5. **Almacenamiento (PostgreSQL/Docker):** La base de datos persiste toda la telemetría histórica y los datos maestros de `shipments`.
6. **Acción y Decisión (Discord/BI Tool):**
    a. Si el motor detecta una anomalía sostenida, genera una **Alerta** enviada por **Webhook a Discord**.
    b. La base de datos se conecta a un **BI Tool** (Metabase/Looker Studio) para la visualización de **KPIs** y el *dashboard*.

2. Fichas de Decisión Arquitectónica (Justificación)

A. Lenguaje de Simulación y Análisis: Python

Elegimos **Python** para el simulador IoT y los *scripts* de evaluación analítica. Esto se justifica por su **Portabilidad** y el principio de **Recursos Compartidos** (Rasgos NIST), ya que el código se ejecuta de manera consistente en el *Edge*, en contenedores y en el entorno de *analytics*. Además, facilita la **Velocidad** (Big Data V) de desarrollo y la experimentación rápida. El *trade-off* es una menor eficiencia en el rendimiento concurrente puro en comparación con otros lenguajes.

B. Protocolo de Comunicación: MQTT

Implementamos el protocolo **MQTT** para el envío de datos desde el *Edge*. Esta es una decisión de **Desacoplamiento** y **Resiliencia**. El modelo Publicador/Suscriptor es

crucial para la **Disponibilidad (A)**; si el sistema de ingesta falla, el *Broker* retiene los mensajes. MQTT optimiza el **Volumen** y la **Velocidad** (Big Data V's) al ser ligero y adecuado para redes inestables de IoT.

C. Herramienta de Contenedorización: Docker

Utilizamos **Docker** (y Compose) para empaquetar servicios como el Broker, la API de Ingesta y PostgreSQL. Esto garantiza la **Veracidad** (Big Data V) del entorno: lo que funciona en local funcionará en cualquier despliegue Cloud. Cumple con el rasgo de **Recursos Compartidos** (NIST) al permitir el uso eficiente de una máquina virtual para alojar múltiples servicios aislados. El *trade-off* es la complejidad adicional de gestión de redes en contenedores.

D. Motor de Ingesta y Procesamiento: Node-RED

Node-RED sirve como nuestro motor *low-code* de recepción y procesamiento. Su principal justificación es la **Disponibilidad (A)**, ya que permite implementar la lógica de **reintentos con *backoff*** de manera visual y robusta contra fallos temporales de la base de datos. Además, facilita el **Autoservicio Bajo Demanda** (NIST) para la rápida prototipación de la lógica de detección. El *trade-off* a largo plazo es una menor **Escalabilidad** en comparación con soluciones puras de *stream processing*.