Elección de arquitectura: DooD vs. DinD

1) Qué implementamos: Docker-outside-of-Docker (DooD)

Los 'Healthcheckers' corren en un contenedor que monta el socket del daemon Docker del host y habla con él vía 'DOCKER_HOST=unix:///var/run/docker.sock'. De esa forma, los healthcheckers controlan los mismos nodos que componen el sistema (Nodo_1..n).

Configuración vigente (DooD)

docker-compose.yml

```
services:
  healther:
    build: .
    container_name: healther
    environment:
      - MODE=${MODE:-manual}                  # manual | auto
      - DOCKER_HOST=unix:///var/run/docker.sock
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock   # <-- daemon del host
      - ./logs:/app/logs
      - ./.env:/app/.env:ro
    tty: true
    stdin_open: true
```

Dockerfile

```
FROM python:3.11-slim
WORKDIR /app
COPY requirements.txt /app/requirements.txt
RUN pip install --no-cache-dir -r /app/requirements.txt
COPY app.py /app/app.py
ENTRYPOINT ["python", "/app/app.py"]
```

app.py (fragmento relevante)

```
def ensure_docker_host():
    if not os.environ.get("DOCKER_HOST"):
        os.environ["DOCKER_HOST"] = "unix:///var/run/docker.sock"
    # ...
client = docker.from_env()
```

Consecuencia directa

Los 'Healthcheckers' "ven" y operan (start/stop/inspect) sobre Nodo_1..n del host.

2) Otra alternativa: Docker-in-Docker (DinD)

DinD levanta un daemon Docker adentro de un contenedor (servicio docker:dind). Los 'Healthcheckers' se conectan a ese daemon (por ejemplo, *tcp://dind:2375*).

Consideración importante: Los nodos que vería/arrancaría son los del daemon interno, no los del host.

Ejemplo de docker-compose.dind.yml:

```
services:
  dind:
    image: docker:26-dind
    privileged: true
    container_name: dind
    environment:
      - DOCKER_TLS_CERTDIR=
    volumes:
      - dind-data:/var/lib/docker
    tty: true
  healther:
    build: .
    container_name: healther
    environment:
      - MODE=${MODE:-manual}
      - DOCKER_HOST=tcp://dind:2375          # <-- se conecta al daemon de dind
    depends_on:
      - dind
    volumes:
      - ./logs:/app/logs
      - ./.env:/app/.env:ro
volumes:
  dind-data:
```

Limitación central para el TP

Con DinD, los 'Healtcheckers' no controlarían a Nodo_1..n (del daemon del host), sino a nodos *dentro* de DinD. Para "revivir" los workers reales habría que migrar todo el stack (cliente/servidor/nodos) a ese daemon interno, lo que rompe la topología y es completamente impracticable en entornos reales, donde los datacenters se encuentran en distintos puntos físicos del planeta.

3) Tabla comparativa

Criterio	DooD (implementado)	DinD (alternativa)
Objetivo del Proyecto (Revivir workers reales)	✅ Controla *Nodo_1..n* del host.	❌ Solo ve los nodos del daemon interno.
Complejidad operativa	Baja: Basta con montar /var/run/docker.sock.	Alta: Requiere servicio dind, modo *privileged*, redes y volúmenes propios.
Overhead / espacio	Mínimo: Sin daemon extra ni capas duplicadas.	Mayor: Mantiene su propio daemon, caché e imágenes duplicadas.
Observabilidad	Unificada: docker ps y logs del host muestran todo.	Fragmentada: El host y DinD tienen vistas separadas de los nodos.
Rendimiento	Sin salto adicional: Usa el daemon del host directamente.	Más overhead por ejecutar un daemon adicional dentro del contenedor.
Seguridad	Acceso al socket del host (privilegios altos, pero controlados).	Requiere *privileged* y expone un daemon adicional, con mayor superficie de ataque.
Paridad con entornos reales	Más realista: Administra nodos del sistema real.	Menos realista: Simula un cluster aislado dentro de un contenedor.
Cuándo conviene usarlo	Ideal para herramientas que operan sobre el cluster local (como *healthcheckers*).	Útil para entornos de CI/CD aislados o pruebas efímeras donde no debe tocarse el host.

4) Seguridad y control del alcance

Ambas opciones implican privilegios altos (DooD por el socket del host; DinD por privileged). En este caso:

- Limitamos el alcance con una lista explícita de targets en el '.env' (HEALTH_TARGETS=...).
- Los healthcheckers operan sólo sobre esos nombres.
- No exponemos el daemon por TCP; usamos socket UNIX local (evita superficie de red).

5) "Vida real" vs. Laboratorio

En producción, los nodos de un sistema distribuido normalmente están en hosts diferentes (cada uno con su IP y su propio runtime). No se "anidan" todos los nodos dentro de un único contenedor/daemon como hace DinD. Elegir DooD nos acerca más al modelo real.