INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS GUADALAJARA



Caso #16-2.

Microservicios, multicontenedores y redes con Docker Compose

Objetivo:

Que el estudiante comprenda y aplique microservicios, multicontenedores y redes de contenedores con Docker Compose

Recordar que el modelo de microservicios considera que las aplicaciones se dividen en sus elementos más pequeños e independientes entre sí. Los microservicios representan componentes independientes que ejecutan siempre la misma tarea cuando son llamados. Una aplicación con microservicios consiste en diferentes microservicios que se ejecutan dentro de una aplicación, además estos microservicios pueden ser compartidos por otras aplicaciones. Es un tema fundamental en el desarrollo de aplicaciones en la nube.

Vamos a realizar una práctica que realiza lo siguiente:

Se tiene una ecuación:

***f(a,b,c,d)=(a+b) × (c−d)***

donde ***f*** recibe los valores de ***a, b, c y d,*** y después dependiendo de estos valores se obtiene el resultado de la suma de (a+b) y se multiplica por el valor resultante de la resta (c-d). El resultado de las operaciones será el resultado de ***f(a,b,c,d).***

Para que puedan tener una mejor comprensión de los microservicios y después los puedan aplicar en problemas reales, vamos a desarrollar **la solución de la ecuación f(a,b,c,d).**

Vamos a diseñar la aplicación con tres microservicios, los cuales serán ejecutados en contenedores independientes:

**1. suma →** Calcula a + b

**2. resta →** Calcula c - d

**3. ecuacion →** Va a llamar a los microservicios suma y resta, les va a pasar sus valores respectivos, y después va a recibir los resultados y los va a multiplicar.

Ejemplo: Si a = 5, b = 3, c = 10, d = 2

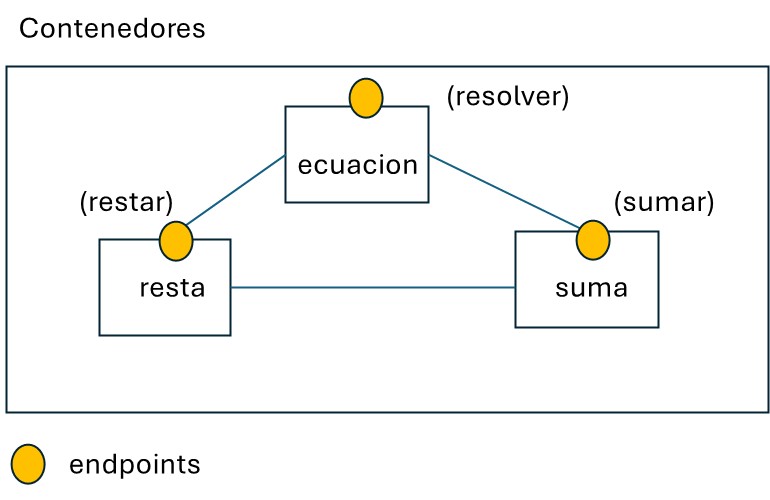
Entonces el resultado es el siguiente:

suma (a+b) = 5 + 3 = 8

resta (c – d) = 10 – 2 = 8

multiplica = 8 \* 8 = 64

Y los tres contenedores los vamos a conectar en red.



Paso 1. Definir la estructura de trabajo:

microservicios

|\_\_\_ suma

| |\_ suma.py

| |\_ Dockerfile

|

|\_\_\_ resta

| |\_ resta.py

| |\_ Dockerfile

|

|\_\_\_ ecuacion

| |\_ multiplica.py

| |\_ Dockerfile

|

|\_\_\_\_compose.yaml

2. Construcción del microservicio ***suma.***

**suma.py**

from fastapi import FastAPI

from pydantic import BaseModel

app = FastAPI()

class Input(BaseModel):

a: float

b: float

@app.post("/sumar")

def sumar(valores: Input):

resultado = valores.a + valores.b

return {"resultado": resultado}

El texto subrayado en amarillo será el ***endpoint*** disponible para conectarse al microservicio suma.

El texto subrayado en azul hace referencia a BaseModel: de la librería Pydantic, que permite definir el esquema de los datos que espera recibir el ***endpoint.***

Para ejecutar el microservicio ***suma*** en un contenedor, hay que definir su dockerfile:

**Dockerfile**

FROM python:3.10-slim

WORKDIR /microservicios

COPY suma.py .

RUN pip install fastapi uvicorn pydantic requests

CMD ["uvicorn", "suma:app", "--host", "0.0.0.0", "--port", "8000"]

3. Construcción del microservicio resta

**resta.py**

from fastapi import FastAPI

from pydantic import BaseModel

app = FastAPI()

class Input(BaseModel):

c: float

d: float

@app.post("/restar")

def restar(valores: Input):

resultado = valores.c - valores.d

return {"resultado": resultado}

El ***endpoint*** del microservicio se encuentra en amarillo.

De igual forma necesitamos el Dockerfile del microservicio ***resta.***

**Dockerfile**

FROM python:3.10-slim

WORKDIR /microservicios

COPY resta.py .

RUN pip install fastapi uvicorn pydantic requests

CMD ["uvicorn", "resta:app", "--host", "0.0.0.0", "--port", "8000"]

4. Construcción del microservicio ecuación.

**multiplica.py**

from fastapi import FastAPI

from pydantic import BaseModel

import requests

app = FastAPI()

class Input(BaseModel):

a: float

b: float

c: float

d: float

@app.post("/resolver")

def resolver(valores: Input):

suma\_resp = requests.post("http://suma:8000/sumar", json={"a": valores.a, "b": valores.b})

resta\_resp = requests.post("http://resta:8000/restar", json={"c": valores.c, "d": valores.d})

suma = suma\_resp.json()["resultado"]

resta = resta\_resp.json()["resultado"]

resultado = suma \* resta

return {"resultado": resultado}

El ***endpoint*** del microservicio se encuentra en amarillo.

Observen en la línea verde que el microservicio ***ecuacion*** por medio de FastAPI, llama a los microservicios ***suma*** y ***resta*** por su ***endpoint***, no es necesario utilizar alguna URL. Esto gracias a que los contenedores se encuentran conectados en red.

***requests:*** se utiliza para hacer peticiones HTTP a otros servicios externos (en este caso, los microservicios suma y resta.

**Dockerfile**

FROM python:3.10-slim

WORKDIR /microservicios

COPY multiplica.py .

RUN pip install fastapi uvicorn pydantic requests

CMD ["uvicorn", "multiplica:app", "--host", "0.0.0.0", "--port", "8000"]

1. Construcción del archivo compose.yaml

version: "3.9"

services:

  suma:

    build: ./suma

    ports:

      - "8001:8000"

    networks:

      - shape-net

  resta:

    build: ./resta

    ports:

      - "8002:8000"

    networks:

      - shape-net

  ecuacion:

    build: ./ecuacion

    ports:

      - "8003:8000"

    networks:

      - shape-net

    depends\_on:

      - suma

      - resta

networks:

  shape-net:

    driver: bridge

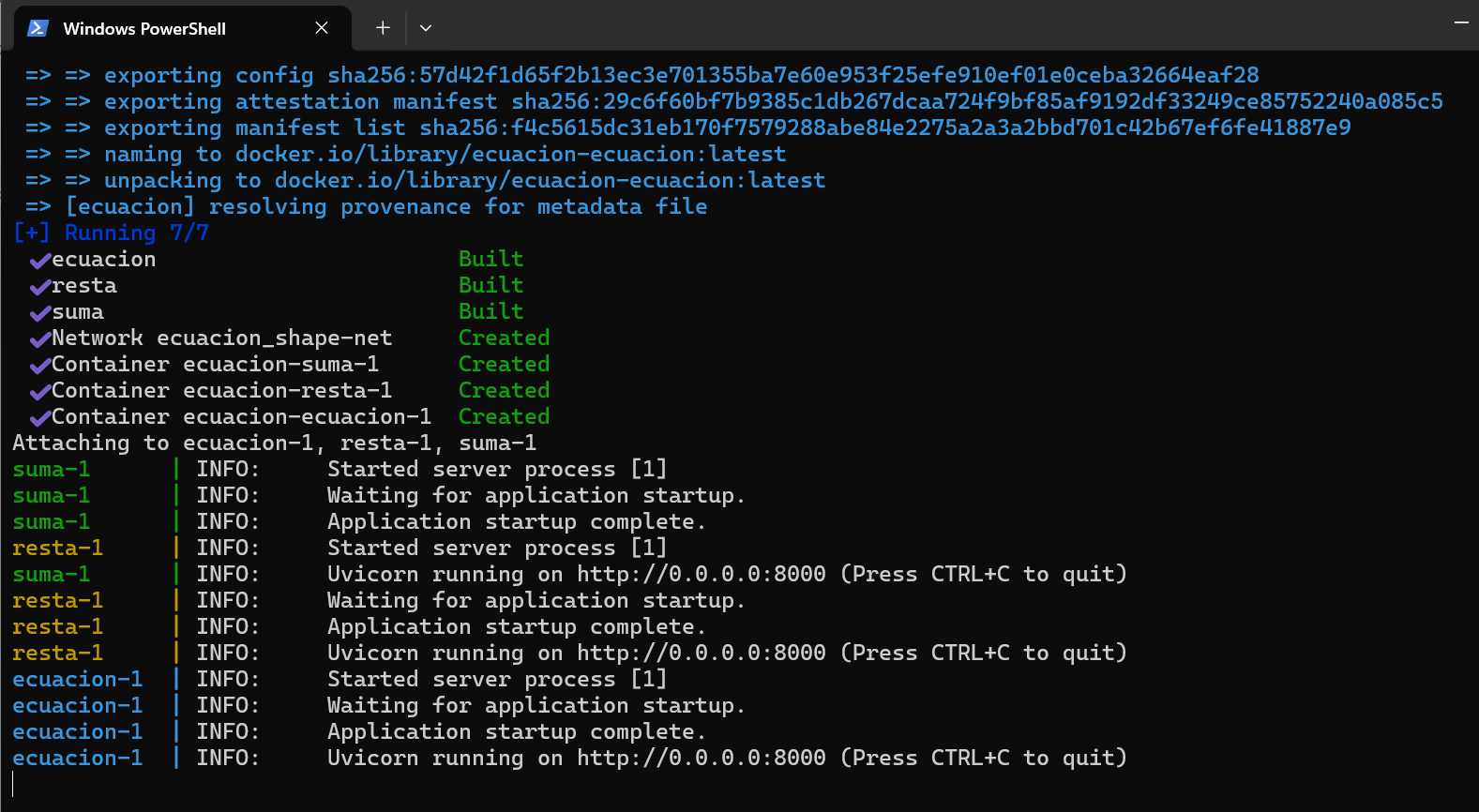
Observen en el yaml la forma de conectar los contenedores en red.

El puerto 8000 del contenedor será el puerto expuesto.

1. Ejecución de la aplicación

***docker-compose up --build***

Este comando levanta los microservicios (suma, resta y multiplica) en contenedores.



Si miramos nuestro Docker Desktop podremos ver los contenedores ejecutándose.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

1. Ahora vamos a pasar datos a los microservicios. Recordar que el microservicio ecuación se accede por medio del ***endpoint resolver***. En este caso el microservicio va a recibir los valores de ***a,b,c y d,*** los va a repartir a los miscroservicios ***suma*** y ***resta,*** los cuales le regresan sus resultados respectivos, y después realiza la multiplicación para resolver la ecuación y presenta el resultado.

Hay varias formas de enviar datos a los contenedores: HTTP, CURL, POSTMAN. En este caso voy a usar POSTMAN.

En la parte del URL teclemaos:

<http://localhost:8003/resolver>

después, en pestaña ***body*** y después en pestaña ***raw***, ponemos los valores de ***a,b,c,d*** en formato json.

{

    “a”: 5,

    “b”:7,

    “c”:3,

    “d”:8

}

Y finalmente damos ***Send,*** y recibimos el resultado en formato json.

{

    "resultado": -60.0

}

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

1. También puedo mandar llamar los microservicios ***suma*** y ***resta*** de manera independiente de la siguiente forma:

**Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.**

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.