Ejercicio 9: Ejemplo utilizando Docker

Materia: Computación tolerante a fallas

Universidad de Guadalajara

Profesor: Michel Emanuel López Franco

22/4/2024

José Maximiliano Díaz Méndez

Introducción

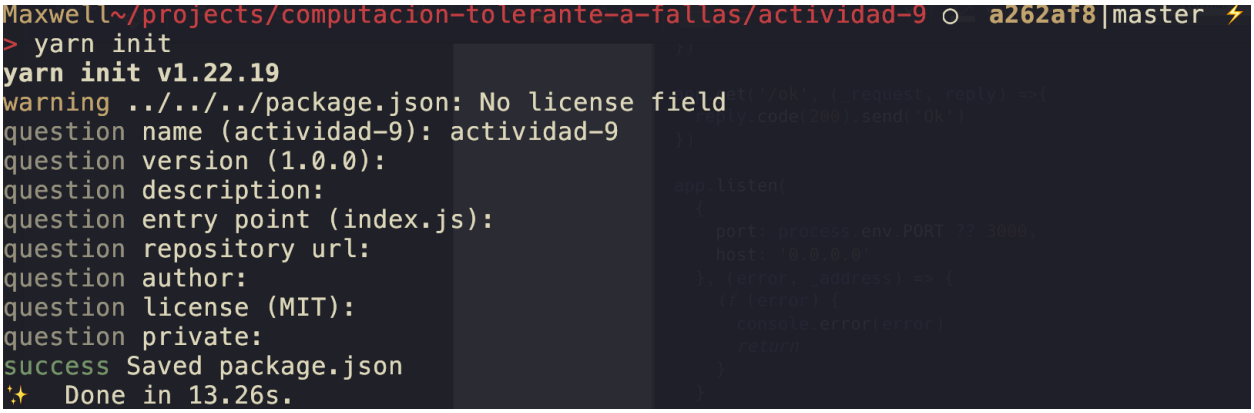
A continuación, se utilizará Docker para crear un servidor de Fastify junto a Nginx para crear un balanceador de carga que distribuya la carga entra dos instancias del servidor.

Requisitos

- Docker

Crear servidor

1. Se crea un directorio a nuestra elección sobre el cual se trabajará.
2.
3. Dentro de este directorio abrimos una terminal y ejecutamos el comando **yarn init** y se llenan los datos a conveniencia.



4. Creamos el subdirectorio **src** y dentro escribimos lo siguiente en el archivo **server.js**.

```
const fastify = require('fastify')

const app = fastify({
  logger: {
    enabled: 'true',
    level: 'debug',
    serializers: {
      req (request) {
        return {
          method: request.method,
          url: request.url,
          headers: request.headers,
          hostname: request.hostname,
          remoteAddress: request.ip,
          remotePort: request.socket.remotePort
        }
      }
    }
  },
})

app.get('/ok', (_request, reply) =>{
  reply.code(200).send('Ok')
})

app.listen(
  {
    port: process.env.PORT ?? 3000,
    host: '0.0.0.0'
  }, (error, _address) => {
    if (error) {
      console.error(error)
      return
    }
  }
)
```

Crear imagen del servidor

1. Se crea el archivo server.dockerfile con el siguiente contenido que indica los pasos para construir la imagen.

```
FROM node:20-alpine
WORKDIR /usr/src/app

COPY package.json .
COPY src src/

RUN yarn install

CMD ["node", "src/server.js"]
```

2. Finalmente usamos el comando **docker build -t node-app:1.0 .** para validar que la imagen se genere correctamente.

Configurar Nginx

1. Se crea el archivo nginx.conf con el siguiente contenido que indica como debe balancear la carga.

```
upstream loadbalancer {
server 172.17.0.1:3001 weight=6;
server 172.17.0.1:3002 weight=6;
}
server {
location / {
proxy_pass http://loadbalancer;
}
```

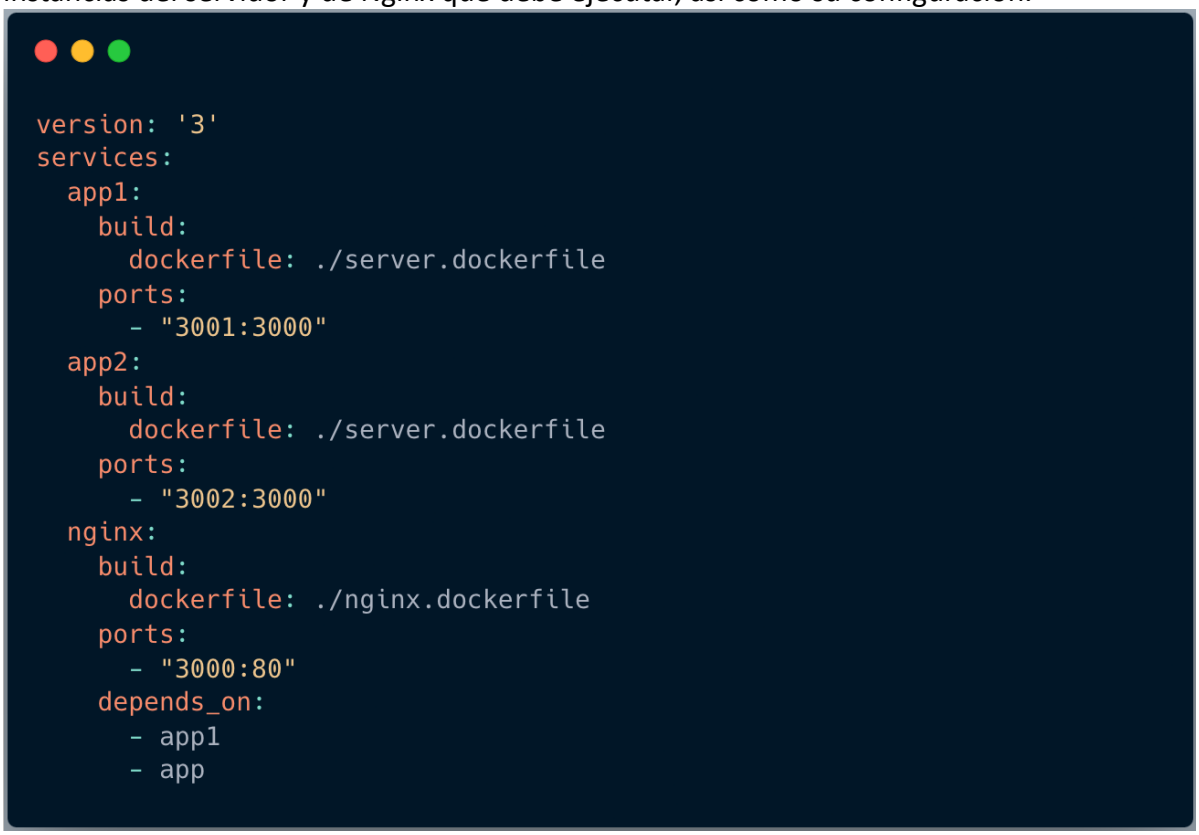
2. Después se crea el archivo nginx.dockerfile con el siguiente contenido indicando como debe construir la imagen de Nginx.

```
FROM nginx
RUN rm /etc/nginx/conf.d/default.conf
COPY nginx.conf /etc/nginx/conf.d/default.conf
```

3. Con el comando **docker build -t load-balancer:1.0 .** podemos comprobar que se construya de forma correcta.

Como ejecutar el proyecto con el balanceador de carga.

1. Se crea el archivo docker-compose.yml con el siguiente contenido que describe las instancias del servidor y de Nginx que debe ejecutar, así como su configuración.



```yaml
version: '3'
services:
  app1:
    build:
      dockerfile: ./server.dockerfile
    ports:
      - "3001:3000"
  app2:
    build:
      dockerfile: ./server.dockerfile
    ports:
      - "3002:3000"
  nginx:
    build:
      dockerfile: ./nginx.dockerfile
    ports:
      - "3000:80"
    depends_on:
      - app1
      - app
```

2. Con el comando **docker compose up** se ejecutar el archivo anterior ejecutando los servicios configurados anteriormente.

```
Maxwell~/projects/computacion-tolerante-a-fallas/actividad-9 ○  a262af8|master ⚡
> docker compose up
WARN[0000] /Users/max021311/projects/computacion-tolerante-a-fallas/actividad-9/docker-compose.yml: `version` is obsolete
[+] Running 3/0
 ✔ Container actividad-9-app2-1   Created
 ✔ Container actividad-9-app1-1   Created
 ✔ Container actividad-9-nginx-1  Created
Attaching to app1-1, app2-1, nginx-1
nginx-1  | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
nginx-1  | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
nginx-1  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
nginx-1  | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
nginx-1  | 10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf differs from the packaged version
nginx-1  | /docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
nginx-1  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
nginx-1  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
nginx-1  | /docker-entrypoint.sh: Configuration complete; ready for start up
nginx-1  | 2024/04/23 01:31:32 [notice] 1#1: using the "epoll" event method
nginx-1  | 2024/04/23 01:31:32 [notice] 1#1: nginx/1.25.5
nginx-1  | 2024/04/23 01:31:32 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
nginx-1  | 2024/04/23 01:31:32 [notice] 1#1: OS: Linux 6.6.22-linuxkit
nginx-1  | 2024/04/23 01:31:32 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
nginx-1  | 2024/04/23 01:31:32 [notice] 1#1: start worker processes
nginx-1  | 2024/04/23 01:31:32 [notice] 1#1: start worker process 28
nginx-1  | 2024/04/23 01:31:32 [notice] 1#1: start worker process 29
nginx-1  | 2024/04/23 01:31:32 [notice] 1#1: start worker process 30
nginx-1  | 2024/04/23 01:31:32 [notice] 1#1: start worker process 31
nginx-1  | 2024/04/23 01:31:32 [notice] 1#1: start worker process 32
nginx-1  | 2024/04/23 01:31:32 [notice] 1#1: start worker process 33
nginx-1  | 2024/04/23 01:31:32 [notice] 1#1: start worker process 34
nginx-1  | 2024/04/23 01:31:32 [notice] 1#1: start worker process 35
app2-1   | {"level":30,"time":1713835892595,"pid":1,"hostname":"16eb67686f89","msg":"Server listening at http://0.0.0.0:3000"}
app1-1   | {"level":30,"time":1713835892595,"pid":1,"hostname":"ddf7f27d20ae","msg":"Server listening at http://0.0.0.0:3000"}

v View in Docker Desktop   w Enable Watch
   nvim src/server.js      docker compose up      …/projects/computacion-tolerante-a-fallas/actividad-9
```

3. Si hacemos suficientes peticiones al servidor de Nginx con el comando **curl http//localhost:3000/ok** se podrá apreciar como Nginx dirige las solicitudes a uno de los dos servidores.

```
nginx-1  | 192.168.65.1 - - [23/Apr/2024:01:33:20 +0000] "GET /ok HTTP/1.1" 200 2 "-" "curl/8.4.0" "-"
app1-1   | {"level":30,"time":1713836000852,"pid":1,"hostname":"ddf7f27d20ae","reqId":"req-6","res":{"statusCode":200},"responseTime":0.47520800
00034766,"msg":"request completed"} Watch
app1-1   | {"level":30,"time":1713836001258,"pid":1,"hostname":"ddf7f27d20ae","reqId":"req-7","req":{"method":"GET","url":"/ok","headers":{"host
":"loadbalancer","connection":"close","user-agent":"curl/8.4.0","accept":"*/*"},"hostname":"loadbalancer","remoteAddress":"172.19.0.4","remotePo
rt":53720},"msg":"incoming request"}
nginx-1  | 192.168.65.1 - - [23/Apr/2024:01:33:21 +0000] "GET /ok HTTP/1.1" 200 2 "-" "curl/8.4.0" "-"
app1-1   | {"level":30,"time":1713836001259,"pid":1,"hostname":"ddf7f27d20ae","reqId":"req-7","res":{"statusCode":200},"responseTime":0.85691600
00042757,"msg":"request completed"} Watch
app1-1   | {"level":30,"time":1713836007096,"pid":1,"hostname":"ddf7f27d20ae","reqId":"req-8","req":{"method":"GET","url":"/ok","headers":{"host
":"loadbalancer","connection":"close","user-agent":"curl/8.4.0","accept":"*/*"},"hostname":"loadbalancer","remoteAddress":"172.19.0.4","remotePo
rt":50684},"msg":"incoming request"}
nginx-1  | 192.168.65.1 - - [23/Apr/2024:01:33:27 +0000] "GET /ok HTTP/1.1" 200 2 "-" "curl/8.4.0" "-"
app1-1   | {"level":30,"time":1713836007097,"pid":1,"hostname":"ddf7f27d20ae","reqId":"req-8","res":{"statusCode":200},"responseTime":1.35099999
99951106,"msg":"request completed"} Watch
app2-1   | {"level":30,"time":1713836007664,"pid":1,"hostname":"16eb67686f89","reqId":"req-1","req":{"method":"GET","url":"/ok","headers":{"host
":"loadbalancer","connection":"close","user-agent":"curl/8.4.0","accept":"*/*"},"hostname":"loadbalancer","remoteAddress":"172.19.0.4","remotePo
rt":45090},"msg":"incoming request"}
nginx-1  | 192.168.65.1 - - [23/Apr/2024:01:33:27 +0000] "GET /ok HTTP/1.1" 200 2 "-" "curl/8.4.0" "-"
app2-1   | {"level":30,"time":1713836007671,"pid":1,"hostname":"16eb67686f89","reqId":"req-1","res":{"statusCode":200},"responseTime":6.77662499
9998603,"msg":"request completed"}e Watch
app1-1   | {"level":30,"time":1713836008099,"pid":1,"hostname":"ddf7f27d20ae","reqId":"req-9","req":{"method":"GET","url":"/ok","headers":{"host
":"loadbalancer","connection":"close","user-agent":"curl/8.4.0","accept":"*/*"},"hostname":"loadbalancer","remoteAddress":"172.19.0.4","remotePo
rt":50698},"msg":"incoming request"}
nginx-1  | 192.168.65.1 - - [23/Apr/2024:01:33:28 +0000] "GET /ok HTTP/1.1" 200 2 "-" "curl/8.4.0" "-"
app1-1   | {"level":30,"time":1713836008101,"pid":1,"hostname":"ddf7f27d20ae","reqId":"req-9","res":{"statusCode":200},"responseTime":1.74908400
00100434,"msg":"request completed"} Watch
app2-1   | {"level":30,"time":1713836008550,"pid":1,"hostname":"16eb67686f89","reqId":"req-2","req":{"method":"GET","url":"/ok","headers":{"host
":"loadbalancer","connection":"close","user-agent":"curl/8.4.0","accept":"*/*"},"hostname":"loadbalancer","remoteAddress":"172.19.0.4","remotePo
rt":45104},"msg":"incoming request"}
nginx-1  | 192.168.65.1 - - [23/Apr/2024:01:33:28 +0000] "GET /ok HTTP/1.1" 200 2 "-" "curl/8.4.0" "-"
app2-1   | {"level":30,"time":1713836008552,"pid":1,"hostname":"16eb67686f89","reqId":"req-2","res":{"statusCode":200},"responseTime":1.28125,"m
sg":"request completed"}   w Enable Watch

v View in Docker Desktop   w Enable Watch
```

4.

Conclusión

La práctica me resulto fácil de hacer debido a que ya he usado bastante Docker por mi trabajo, pero fue interesante tener que configurar manualmente un balanceador de carga en un entorno de desarrollo local ya que nunca he hecho esto, sino que usado el balanceador de carga de AWS para dirigir y dividir el tráfico de distintas partes de la app web a diferentes microservicios.

Referencias

Ouassini, A. (2023, 16 diciembre). Sample load balancing solution with Nginx and docker | Medium. *Medium*. Recuperado 22 de abril de 2024, de https://medium.com/@ouassini/sample-load-balancing-solution-with-docker-and-nginx-cf1ffc60e644