Despliegue de MongoDB con Docker

Gabriel Rodríguez Flores

February 13, 2024

ADA ITS
INSTITUTO TECNOLÓGICO SUPERIOR

    - Docker-compose
  - Persistencia de los volumenes
  - Automatizar operaciones (up/down) con los scripts
  - Networks

Contents

1 Teoría

1.1 Fichero docker-compose

1.1.1 Servicios

- `image` (obligatorio) - Imagen a usar
- `name` - Nombre del servicio
- `ports` - Puertos a exponer (host:container)
- `volumes` - Volumenes a montar (host:container)
- `environment` - Variables de entorno
- `networks` - Redes a las que conectarse
- `restart` - Política de reinicio
- `depends_on` - Dependencias entre servicios
- `command` - Comando a ejecutar
- `entrypoint` - Entrypoint a ejecutar

```yaml
version: "3"
services:
  web:
    image: "nginx:alpine"
    ports:
      - "8080:80"
    volumes:
      - ./html:/usr/share/nginx/html
    environment:
      - NGINX_PORT=80
    networks:
      - frontend
    restart: always
    depends_on:
      - api
  api:
    image: "node:alpine"
    ports:
      - "3000:3000"
    volumes:
      - ./app:/app
    environment:
      - NODE_ENV=production
    networks:
      - frontend
      - backend
    restart: always
```

1.1.2 Volúmenes

- `driver` - Driver de almacenamiento
  - `local` - Almacenamiento local
  - `named` - Volumen con nombre
  - `anonymous` - Volumen anónimo
- `name` - Nombre del volumen
- `external` - Volumen externo

```
version: "3"
services:
  web:
    image: "nginx:alpine"
    ports:
      - "8080:80"
    volumes:
      - html:/usr/share/nginx/html
volumes:
  html:
    driver: local
```

1.1.3 Redes

- `driver` - Driver de red
  - `bridge` - Red por defecto
  - `host` - Red del host
  - `none` - Sin red
  - `overlay` - Red de overlay
- `name` - Nombre de la red
- `external` - Red externa
- `attachable` - Red adjunta
- `internal` - Red interna
- `labels` - Etiquetas

```
version: "3"
services:
  web:
    image: "nginx:alpine"
    ports:
      - "8080:80"
    networks:
```

```
      - frontend
  api:
    image: "node:alpine"
    ports:
      - "3000:3000"
    networks:
      - frontend
      - backend
networks:
  frontend:
    driver: bridge
  backend:
    driver: bridge
```

1.1.4 Build

- `context` - Contexto de construcción
- `dockerfile` - Fichero Dockerfile
- `args` - Argumentos de construcción
- `cache_from` - Imágenes de cache
- `labels` - Etiquetas
- `target` - Etiqueta de construcción

```
version: "3"
services:
  web:
    build:
      context: .
      dockerfile: Dockerfile
      args:
        - NODE_ENV=production
      cache_from:
        - node:alpine
      labels:
        - "com.example.description=Web"
        - "com.example.department=IT"
      target: builder
```

1.1.5 Profiles

- profile

Parámetros:

- **`extends`** - Perfiles a extender

- `file` - Fichero de perfiles
- `service` - Servicios a extender
- `network` - Redes a extender
- `volume` - Volúmenes a extender
- `config` - Configuraciones a extender
- `secrets` - Secretos a extender
- `secrets_file` - Fichero de secretos

1.2 Ejecución docker-compose

- `up` - Levantar los servicios
- `down` - Parar los servicios
- `logs` - Ver los logs
- `exec` - Ejecutar un comando en un servicio
- `ps` - Ver los servicios
- `build` - Construir las imágenes
- `pull` - Descargar las imágenes
- `push` - Subir las imágenes
- `images` - Ver las imágenes
- `networks` - Ver las redes
- `volumes` - Ver los volúmenes
- `prune` - Limpiar los recursos
- `stop` - Parar los servicios
- `start` - Iniciar los servicios
- `restart` - Reiniciar los servicios
- `rm` - Eliminar los servicios
- `rmi` - Eliminar las imágenes
- `network rm` - Eliminar las redes
- `volume rm` - Eliminar los volúmenes
- `exec` - Ejecutar un comando en un servicio
- `run` - Ejecutar un comando en un contenedor
- `cp` - Copiar archivos entre el host y el contenedor
- `top` - Ver los procesos de un servicio
- `stats` - Ver las estadísticas de un servicio
- `inspect` - Inspeccionar un servicio
- `events` - Ver los eventos del sistema
- `version` - Ver la versión de Docker
- `info` - Ver la información del sistema

2 Ejemplos

2.1 Servicios listos para desplegar

2.1.1 Docker-compose para MongoDB

```yaml
version: "3"
services:
  mongodb:
    image: "mongo"
    ports:
      - "27017:27017"
    volumes:
      - mongo-data:/data/db
volumes:
 mongo-data:
    driver: local
```

2.1.2 Docker-compose para Redis

```yaml
version: "3"
services:
  redis:
    image: "redis"
    ports:
      - "6379:6379"
    volumes:
      - redis-data:/data
volumes:
  redis-data:
    driver: local
```

2.1.3 Docker-compose para RabbitMQ

```yaml
version: "3"
services:
  rabbitmq:
    image: "rabbitmq:3-management"
    ports:
      - "5672:5672"
      - "15672:15672"
    volumes:
      - rabbitmq-data:/var/lib/rabbitmq
volumes:
  rabbitmq-data:
```

```
    driver: local
```

2.1.4 Docker-compose para Postgres

```
version: "3"
services:
  postgres:
    image: "postgres"
    ports:
      - "5432:5432"
    volumes:
      - postgres-data:/var/lib/postgresql/data
volumes:
  postgres-data:
    driver: local
```

2.1.5 Docker-compose para MailHog

```
version: "3"
services:
  mailhog:
    image: "mailhog/mailhog"
    ports:
      - "1025:1025"
      - "8025:8025"
```

2.1.6 Docker-compose para Centrifugo

```
version: "3"
services:
  centrifugo:
    image: centrifugo/centrifugo:latest
    ports:
      - "8000:8000"
    volumes:
      - centrifugo-data:/centrifugo
volumes:
  centrifugo-data:
    driver: local
```

2.1.7 Docker-compose para SonarQube

```
version: "3"
services:
  sonarqube:
    image: sonarqube:latest
    ports:
      - "9000:9000"
    volumes:
      - sonarqube-data:/opt/sonarqube/data
volumes:
  sonarqube-data:
    driver: local
```

2.1.8 Docker-compose para WireMock

```
version: "3"
services:
  wiremock:
    image: "rodolpheche/wiremock"
    ports:
      - "8080:8080"
    volumes:
      - wiremock-data:/home/wiremock
      - ./mappings:/home/wiremock/mappings
volumes:
  wiremock-data:
    driver: local
```

3 Tips

3.1 Acceder a la máquina host desde un contenedor

- Usar `host.docker.internal` en lugar de `localhost`
- Hay que configurar dicho alias. En docker-compose:

```
    extra_hosts:
      - "host.docker.internal:host-gateway"
```

4 Ejercicios

5 Entregables

5.1 En clase

5.2 Tarea