# 4.3 Langkah deploy aplikasi multi-container menggunakan Docker Compose dan NGINX sebagai reverse proxy

# 4.3.1 Kandungan docker-compose.yml

Fail docker-compose.yml menyenaraikan semua servis yang anda ingin jalankan. Contohnya:

|  |
| --- |
| version: '3.8'  services:  product-svc:  container\_name: productservice  build: ./productSvc  ports:  - "8020:8020"  environment:  APP\_INTERNAL\_PORT: 8020  DB\_ADDRESS: mysql  DB\_PORT: 3306  DB\_NAME: productmngm  DB\_USERNAME: root  DB\_PASSWORD: S3cret  depends\_on:  - mysql  networks:  - my-network  mysql:  container\_name: mysql  image: mysql:latest  restart: always  environment:  MYSQL\_ROOT\_PASSWORD: S3cret  MYSQL\_DATABASE: productmngm  ports:  - "3306:3306"  volumes:  - mysql-data:/var/lib/mysql  networks:  - my-network  nginx:  container\_name: nginx  image: nginx:latest  ports:  - "80:80"  volumes:  - ./nginx.conf:/etc/nginx/nginx.conf:ro  depends\_on:  - product-svc  networks:  - my-network  product-ui:  container\_name: product-ui  build: ./productApp # path to your React app Dockerfile folder  ports:  - "8030:80"  networks:  - my-network  volumes:  mysql-data:  networks:  my-network:  driver: bridge |

# 4.3.2 NGINX sebagai reverse proxy

Apa Itu **Reverse Proxy**?

Secara ringkas:

**Reverse proxy** ialah *“orang tengah”* antara client (browser) dengan backend server (API, database, frontend app dll).

Analogi Mudah:  
Bayangkan anda pergi ke kaunter servis pelanggan di pejabat kerajaan:

* Anda (client) tak terus jumpa pegawai belakang.
* Anda jumpa **kaunter depan** dulu (reverse proxy).
* Kaunter ini tengok permintaan anda, dan **salurkan** ke jabatan yang betul (backend server).

Fungsi Reverse Proxy:

* **Terima permintaan dari client** (browser).
* **Salurkan ke backend** server tertentu (Spring Boot, Node.js, React, dsb).
* Hantar semula **jawapan dari backend** ke client.

Kandungan nginx.conf:

|  |
| --- |
| events {}  http {  server {  listen 80;  location /product-service/ {  proxy\_pass http://productservice:8020/product-service/;  }  location /product-ui/ {  proxy\_pass http://product-ui:80/;  }  }  }  } |

# 4.3.3 Bina dan Jalankan Semua Servis

Dalam terminal, jalankan arahan:

|  |
| --- |
| docker-compose up -d --build |

* **Build** semua servis (kalau ada Dockerfile),
* Jalankan dalam **background (-d)**

# 4.3.4 Periksa Sama Ada Semua Container Berjalan

Gunakan perintah:

|  |
| --- |
| docker ps |

Pastikan semua container seperti productservice, mysql, product-ui, dan nginx aktif. Seperti:

# 4.3.5 Akses Aplikasi Anda di Pelayar (Browser)

* Frontend (React App)  
  <http://localhost/product-ui>
* API (Swagger UI)  
  <http://localhost/product-service/swagger-ui/index.html>