

**Load Balanced Web API dengan Multi-Container Docker (Backend,
Load Balancer dan Redis Database)**



Disusun oleh:

Alayavaro Rachmadia	2410501095
Panji Anugerah Panengah	2410501105
Rakha Abyan Hertamtama	2410501089
Rizki Ramadhan	2410501091
Shidqi Athalla Arka Qafriyanto	2410501077

Jalan RS. Fatmawati Raya, Pd. Labu, Kec. Cilandak, Kota Jakarta Selatan,

Daerah Khusus Ibukota Jakarta 12450

D3 Sistem Informasi

1. Deskripsi Arsitektur

Project ini membangun sebuah sistem *multi-container* menggunakan Docker Compose yang terdiri dari empat service utama:

1. **Backend1 (Flask API instance 1)**
2. **Backend2 (Flask API instance 2)**
3. **Redis Database** untuk penyimpanan data (*total_visits*)
4. **Nginx Reverse Proxy** berfungsi sebagai load balancer

Ketika pengguna mengakses <http://localhost:8080>, request akan masuk ke Nginx → didistribusikan secara bergantian (*round-robin*) ke backend1 atau backend2. Setiap instance backend mencatat jumlah kunjungan ke Redis, sehingga data tetap tersimpan meskipun container backend dihentikan atau dihapus.

2. Diagram Arsitektur

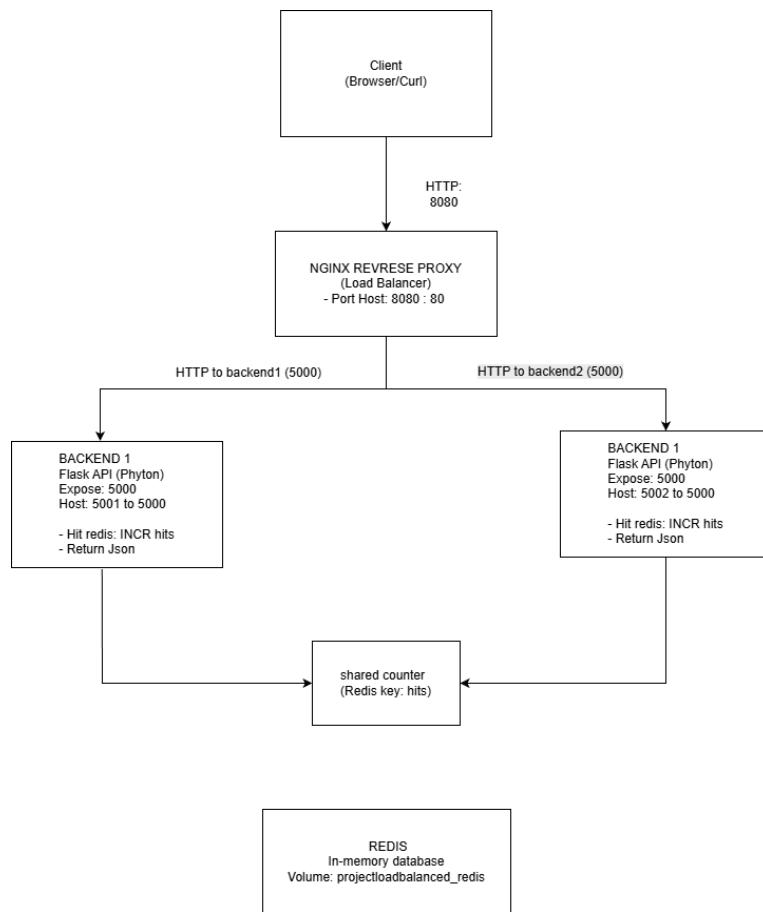
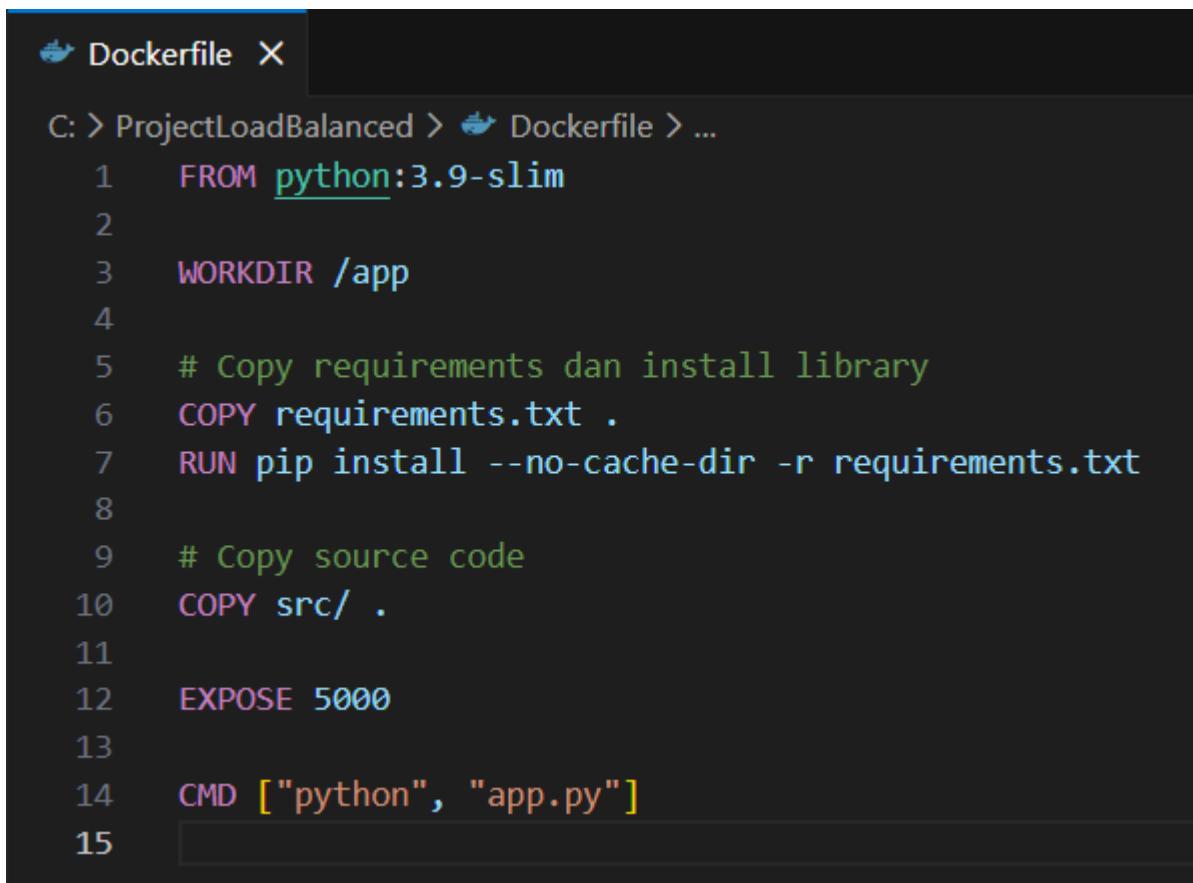


Diagram tersebut menunjukkan arsitektur Load Balanced Web API di mana client mengirim request ke Nginx (port 8080) sebagai load balancer. Nginx kemudian membagi request secara bergantian ke dua backend Flask API (Backend 1 dan Backend 2). Setiap backend mengakses Redis untuk menambah nilai counter (hits) yang digunakan bersama (shared). Hasilnya, beban kerja terbagi merata, performa lebih stabil, dan data tetap tersinkron antar backend.

3. Penjelasan Dockerfile



```
C: > ProjectLoadBalanced > Dockerfile > ...
1  FROM python:3.9-slim
2
3  WORKDIR /app
4
5  # Copy requirements dan install library
6  COPY requirements.txt .
7  RUN pip install --no-cache-dir -r requirements.txt
8
9  # Copy source code
10 COPY src/ .
11
12 EXPOSE 5000
13
14 CMD ["python", "app.py"]
15
```

Penjelasan:

- **Base image:** python:3.9-slim (ringan & stabil)
- **WORKDIR /app:** lokasi file aplikasi
- **COPY src/:** menyalin app.py
- **Install dependency:** Flask & Redis client
- **Expose 5000:** port API
- **CMD:** menjalankan Flask server

Dockerfile ini digunakan untuk membangun image backend1 dan backend2.

4. Penjelasan docker-compose.yml

```
  Dockerfile X  docker-compose.yml X
C: > ProjectLoadBalanced > docker-compose.yml
1   version: '3.8'
2
3   services:
4     redis:
5       image: redis:alpine
6       container_name: redis
7       ports:
8         - "6379:6379"
9       volumes:
10      - redis_data:/data
11
12    backend1:
13      build: ./src
14      container_name: backend1
15      ports:
16        - "5001:5000"
17      environment:
18        - REDIS_HOST=redis
19      depends_on:
20        - redis
21
22    backend2:
23      build: ./src
24      container_name: backend2
25      ports:
26        - "5002:5000"
27      environment:
28        - REDIS_HOST=redis
29      depends_on:
30        - redis
31
32    nginx:
33      image: nginx:alpine
34      container_name: nginx
35
36      ports:
37        - "8080:80"
38
39      volumes:
40        - ./configs/nginx.conf:/etc/nginx/nginx.conf:ro
41      depends_on:
42        - backend1
43        - backend2
44
45      volumes:
46        - redis_data:
```

Penjelasan Service

Service	Fungsi
redis	database penyimpanan total kunjungan
backend1	instance API Flask pertama

backend2 instance API Flask kedua

nginx reverse proxy + load balancer

Port Mapping

- Backend1 → localhost:5001 → port internal 5000
- Backend2 → localhost:5002 → port internal 5000
- Redis → localhost:6379
- Nginx → localhost:8080

Volume

- redis_data:/data (persistent storage)

Network

Semua service berada dalam 1 internal network bernama mynet.

5. Cara Menjalankan Project

Build Image Backend:

```
PS C:\ProjectLoadBalanced> docker build -t backend-image .
[+] Building 3.3s (11/11) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 275B
=> [internal] load metadata for docker.io/library/python:3.9-slim
=> [auth] library/python:pull token for registry-1.docker.io
=> [internal] load .dockerrcignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/python:3.9-slim@sha256:2d97f6910b16bd338d3060f261f53f144965f755599aab1acda1e13cf1731b1b
=> => resolve docker.io/library/python:3.9-slim@sha256:2d97f6910b16bd338d3060f261f53f144965f755599aab1acda1e13cf1731b1b
=> [internal] load build context
=> => transferring context: 90B
=> CACHED [2/5] WORKDIR /app
=> CACHED [3/5] COPY requirements.txt .
=> CACHED [4/5] RUN pip install --no-cache-dir -r requirements.txt
=> CACHED [5/5] COPY src/ .
=> exporting to image
=> => exporting layers
=> => exporting manifest sha256:1abeb6834327d5ff622e70892aef1960f9cce76e6219deb965f6544a62fd7309
=> => exporting config sha256:4fccfb993fb84e0cf3502db3c18a1c5ccae714949c4599d0940fb5bbc7106ab
=> => exporting attestation manifest sha256:448a10b7a84ed0a02f65bf03dbc1403ddf07e9de71058b86b6180c6a6e31df
=> => exporting manifest list sha256:4416b3e0f2dd9b3c47d3547ab098302254de889723ebd10328f92d12486d8075
=> => naming to docker.io/library/backend-image:latest
=> => unpacking to docker.io/library/backend-image:latest
PS C:\ProjectLoadBalanced>
```

Docker Compose Up -d:

```
PS C:\ProjectLoadBalanced> docker compose up -d
time="2025-12-03T21:40:36+07:00" level=warning msg="C:\\\\ProjectLoadBalanced\\\\docker-compose.yml: the attribute `version` is obsolete,
it will be ignored, please remove it to avoid potential confusion"
[+] Running 5/5
✓ Network projectloadbalanced_default Created
✓ Container redis Started
✓ Container backend1 Started
✓ Container backend2 Started
✓ Container nginx Started
PS C:\ProjectLoadBalanced>
```

Cek Kontainer Proses:

```
PS C:\ProjectLoadBalanced> docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS
de5e21e7597d nginx:alpine "/docker-entrypoint..." 32 seconds ago Up 30 seconds 0.0.0.0:8080->80/tcp, [::]:80
80->80/tcp      nginx
fe65b4364c13 projectloadbalanced-backend2 "python app.py" 32 seconds ago Up 30 seconds 0.0.0.0:5002->5000/tcp, [::]-
5002->5000/tcp backend2
d6f53c5499a2 projectloadbalanced-backend1 "python app.py" 32 seconds ago Up 30 seconds 0.0.0.0:5001->5000/tcp, [::]-
5001->5000/tcp backend1
8dec3b4ad075 redis:alpine "docker-entrypoint.s..." 32 seconds ago Up 31 seconds 0.0.0.0:6379->6379/tcp, [::]-
6379->6379/tcp redis
PS C:\ProjectLoadBalanced>
```

Log nginx:

```
PS C:\ProjectLoadBalanced> docker compose logs nginx
time="2025-12-03T21:41:43+07:00" level=warning msg="C:\\\\ProjectLoadBalanced\\\\docker-compose.yml: the attribute `version` is obsolete,
it will be ignored, please remove it to avoid potential confusion"
nginx | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
nginx | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
nginx | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
nginx | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
nginx | 10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
nginx | /docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
nginx | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
nginx | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
nginx | /docker-entrypoint.sh: Configuration complete; ready for start up
PS C:\ProjectLoadBalanced>
```

Test Api Via Load Balancer:

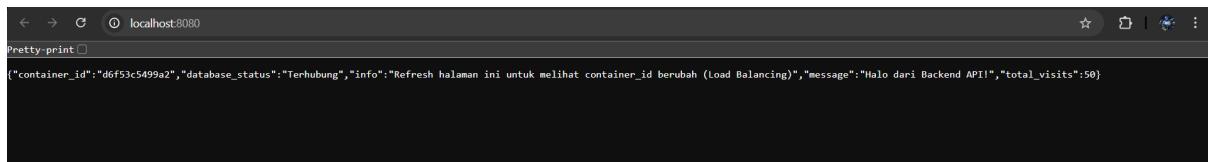
```
PS C:\ProjectLoadBalanced> curl http://localhost:8080

StatusCode : 200
StatusDescription : OK
Content : {"container_id": "d6f53c5499a2", "database_status": "Terhubung", "info": "Refresh halaman ini untuk melihat container_id berubah (Load Balancing)", "message": "Halo dari Backend API!", "total_visits": 46}

RawContent : HTTP/1.1 200 OK
              Connection: keep-alive
              Content-Length: 196
              Content-Type: application/json
              Date: Wed, 03 Dec 2025 14:42:26 GMT
              Server: nginx/1.29.3

              {"container_id": "d6f53c5499a2", "database_stat...
Forms : {}
Headers : [[Connection, keep-alive], [Content-Length, 196], [Content-Type, application/json], [Date, Wed, 03 Dec 2025 14:42:26 GMT]...]
Images : {}
InputFields : {}
Links : {}
ParsedHtml : mshtml.HTMLDocumentClass
RawContentLength : 196
```

6. Screenshot Hasil Berjalan



```

backend2 | * Serving Flask app 'app'
backend2 | * Debug mode: off
backend2 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
backend2 | * Running on all addresses (0.0.0.0)
backend2 | * Running on http://127.0.0.1:5000
backend2 | * Running on http://172.22.0.4:5000
backend2 | Press CTRL+C to quit
backend2 | 172.22.0.5 - - [03/Dec/2025 14:42:45] "GET / HTTP/1.0" 200 -
backend2 | 172.22.0.5 - - [03/Dec/2025 14:42:49] "GET / HTTP/1.0" 200 -
backend1 | * Serving Flask app 'app'
backend1 | * Debug mode: off
backend1 | WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
backend1 | * Running on all addresses (0.0.0.0)
backend1 | * Running on http://127.0.0.1:5000
backend1 | * Running on http://172.22.0.3:5000
backend1 | Press CTRL+C to quit
backend1 | 172.22.0.5 - - [03/Dec/2025 14:42:26] "GET / HTTP/1.0" 200 -
backend1 | 172.22.0.5 - - [03/Dec/2025 14:42:48] "GET / HTTP/1.0" 200 -
backend1 | 172.22.0.5 - - [03/Dec/2025 14:42:50] "GET / HTTP/1.0" 200 -
redis | Starting Redis Server
redis | 1:C 03 Dec 2025 14:40:37.447 * o000o000o000o Redis is starting o000o000o000o
redis | 1:C 03 Dec 2025 14:40:37.447 * Redis version=8.4.0, bits=64, commit=00000000, modified=1, pid=1, just started
redis | 1:C 03 Dec 2025 14:40:37.447 * Configuration loaded
redis | 1:M 03 Dec 2025 14:40:37.448 * monotonic clock: POSTIX_clock_gettime
redis | 1:M 03 Dec 2025 14:40:37.516 * <ReJSON> Version: 80400 git sha: unknown branch: unknown
redis | 1:M 03 Dec 2025 14:40:37.516 * <ReJSON> Exported RedisJSON_V1 API
redis | 1:M 03 Dec 2025 14:40:37.517 * <ReJSON> Exported RedisJSON_V2 API
redis | 1:M 03 Dec 2025 14:40:37.517 * <ReJSON> Exported RedisJSON_V3 API
redis | 1:M 03 Dec 2025 14:40:37.517 * <ReJSON> Exported RedisJSON_V4 API
redis | 1:M 03 Dec 2025 14:40:37.517 * <ReJSON> Exported RedisJSON_V5 API
redis | 1:M 03 Dec 2025 14:40:37.517 * <ReJSON> Exported RedisJSON_V6 API
redis | 1:M 03 Dec 2025 14:40:37.517 * <ReJSON> Enabled diskless replication
redis | 1:M 03 Dec 2025 14:40:37.517 * <ReJSON> Initialized shared string cache, thread safe: true.
redis | 1:M 03 Dec 2025 14:40:37.517 * Module 'ReJSON' loaded from /usr/local/lib/redis/modules//rejson.so
redis | 1:M 03 Dec 2025 14:40:37.517 * <search> Acquired RedisJSON_V6 API
redis | 1:M 03 Dec 2025 14:40:37.519 * Server initialized
redis | 1:M 03 Dec 2025 14:40:37.520 * <search> Loading event starts
redis | 1:M 03 Dec 2025 14:40:37.520 * <search> Changing workers threadpool size from 0 to 4
redis | 1:M 03 Dec 2025 14:40:37.520 * <search> Enabled workers threadpool of size 4
redis | 1:M 03 Dec 2025 14:40:37.521 * Loading RDB produced by version 8.4.0
redis | 1:M 03 Dec 2025 14:40:37.521 * RDB age 43 seconds
redis | 1:M 03 Dec 2025 14:40:37.521 * RDB memory usage when created 1.12 Mb
redis | 1:M 03 Dec 2025 14:40:37.521 * Done loading RDB, keys loaded: 1, keys expired: 0.
redis | 1:M 03 Dec 2025 14:40:37.521 * <search> Changing workers threadpool size from 4 to 0
redis | 1:M 03 Dec 2025 14:40:37.522 * <search> Disabled workers threadpool of size 4
redis | 1:M 03 Dec 2025 14:40:37.522 * <search> Loading event ends
redis | 1:M 03 Dec 2025 14:40:37.522 * DB loaded from disk: 0.002 seconds
redis | 1:M 03 Dec 2025 14:40:37.522 * Ready to accept connections tcp

```

Log di atas menunjukkan bahwa seluruh sistem **Nginx + Backend1 + Backend2 + Redis** telah berjalan dengan benar dan saling terhubung di dalam Docker Compose. Nginx berhasil memuat konfigurasi load balancer dan menerima request dari host (172.22.0.1), lalu meneruskannya bergantian ke Backend1 dan Backend2, terlihat dari log Flask yang menerima request bergantian.. Kedua backend berjalan pada alamat internal container (172.22.0.3 dan 172.22.0.4) serta merespons permintaan melalui port 5000 tanpa error. Redis juga berhasil berjalan dalam mode standalone dengan modul-modul tambahan aktif, menunjukkan bahwa database siap menerima dan menyimpan data secara persisten melalui volume yang telah dipasang.

7. Kendala dan Solusi

Kendala 1 Nginx 502 Bad Gateway

- Penyebab: backend berjalan di port 3000
- Solusi: mengubah app.run(... port=5000) sesuai dengan compose & Nginx

Kendala 2 Load balancer tidak merespons

Penyebab: salah upstream port di default.conf

Solusi: sesuaikan ke backend1:5000 dan backend2:5000

Kendala 3 Data tidak persistent

Penyebab: Redis belum memakai volume

Solusi: menambahkan

volumes:

- redis_data:/data
 - Data total_visits menjadi tetap setelah restart.

Kendala 4 Data tidak persistent

Request tidak bergantian ke Backend1/Backend2

Solusi: Memakai round_robin default pada Nginx upstream.

