MEMBUAT APLIKASI TO-DO LIST DENGAN BERBASIS DOCKER

LAPORAN MINGGU KE-13

Dosen Pengampu : Ferdi Chahyadi, S.Kom., M.Cs

Disusun Untuk Memenuhi Tugas Proyek

Mata Kuliah Sistem Operasi

DISUSUN OLEH :

1. **ADHIE MULIA SEMBIRING (2401020165)**
2. **MISYE KHALINA APRILIA MARBUN (2401020147)**
3. **RENDA KURNIA MANIK (2301020003)**
4. **YUDHA EKAPUTRA (2301020019)**

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS TEKNIK DAN TEKNOLOGI KEMARITIMAN

UNIVERSITAS MARITIM RAJA ALI HAJI

2025

1. AKTIVITAS YANG DILAKUKAN

1.1 Pengembangan Aplikasi Flask

Todo-App (port 5000)

- Frameworks : Pyhton Flask
- Endpoints : ?todos, /add, /complete
- Fungsi : Backend untuk manajemen to-do list dengan CRUD operations

Stats-App (port 5001)

- Framework: Python Flask
- Endpoints: /stats, /productivity
- Fungsi: Analisis statistik tasks dari todo-app
- Komunikasi: HTTP request ke todo-app melalui Docker network

1.2 Pembuatan Dockerfile

Dockerfile dibuat untuk kedua aplikasi dengan konfigurasi:

- Base Image: python:3.9-slim
- Working Directory: /app
- Install dependencies dari requirements.txt
- Expose port 5000 (todo-app) dan 5001 (stats-app)
- CMD untuk menjalankan aplikasi

1.3 Build dan Testing

- Build image untuk todo-app dan stats-app
- Testing individual container
- Konfigurasi Docker Compose untuk multi-container
- Integration testing komunikasi antar container

2. HASIL YANG DICAPAI

➢ Aplikasi Flask

- Todo-app dan stats-app berfungsi dengan baik
- Semua endpoints tested dan working
- Inter-container communication berhasil

➢ Docker Images

- todo-app: 200MB
- stats-app: 204MB
- Build process tanpa error

➢ Multi-Container

- Docker Compose orchestration berhasil
- Kedua container running stabil
- Network communication berfungsi

3. BUKTI IMPLEMENTASI

3.1 Struktur Project

- Gambar 1 : Struktur Folder

```
PS C:\Users\MSI Thin\to-do-list> dir


    Directory: C:\Users\MSI Thin\to-do-list


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         12/6/2025  12:50 PM                stats-app
d-----         12/6/2025  12:49 PM                todo-app
-a----        12/12/2025   4:42 PM            163 docker-compose.yml
```

3.2 Docker Image

- Gambar 2 : Docker Images List

```
PS C:\Users\MSI Thin\to-do-list\stats-app> docker images
                                                  i Info →  U  In Use
IMAGE                ID             DISK USAGE   CONTENT SIZE   EXTRA
hello-world:latest   f7931603f70e       20.3kB         3.96kB    U
python:3.9-slim      2d97f6910b16        183MB           45MB
stats-app:latest     e34bfcce7bee        204MB         49.7MB
todo-app:latest      1087ed102f71        200MB         48.8MB
```

3.3 Dockerfile Configurations

- Gambar 3 : Dockerfile Todo-App

```
PS C:\Users\MSI Thin\to-do-list> cd todo-app
PS C:\Users\MSI Thin\to-do-list\todo-app> type dockerfile
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY app.py .
EXPOSE 5000
CMD ["python", "app.py"]
```

- Gambar 4 : Fockerfile Stats-App

```
PS C:\Users\MSI Thin\to-do-list\stats-app> type dockerfile
FROM python:3.9-slim
WORKDIR /app
COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt
COPY app.py .
EXPOSE 5001
CMD ["python", "app.py"]
PS C:\Users\MSI Thin\to-do-list\stats-app>
```

3.4 Source Code

- Gambar 5 : app.py Todo-App

```
PS C:\Users\MSI Thin\to-do-list\todo-app> type app.py
from flask import Flask, jsonify, request

app = Flask(__name__)

todos = [
    {"id": 1, "task": "Belajar Docker", "completed": False},
    {"id": 2, "task": "Buat laporan", "completed": True},
]

@app.route('/')
def home():
    return jsonify({"message": "Todo App is Running!", "status": "OK"})

@app.route('/todos')
def get_todos():
    return jsonify({"todos": todos, "total": len(todos)})

@app.route('/add/<task>')
def add_todo(task):
    new_id = max([t['id'] for t in todos]) + 1 if todos else 1
    new_todo = {"id": new_id, "task": task, "completed": False}
    todos.append(new_todo)
    return jsonify({"message": "Todo added", "todo": new_todo})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

- Gambar 6 : requirements.txt Todo-App

```
PS C:\Users\MSI Thin\to-do-list\todo-app> type requirements.txt
flask==2.3.3
```

- Gambar 7 : app.py Stats-App

```
PS C:\Users\MSI Thin\to-do-list\stats-app> type app.py
from flask import Flask, jsonify
import requests

app = Flask(__name__)

@app.route('/')
def home():
    return jsonify({"message": "Stats App is Running!", "status": "OK"})

@app.route('/stats')
def get_stats():
    try:
        response = requests.get('http://todo-app:5000/todos')
        todos = response.json()['todos']

        total = len(todos)
        completed = len([t for t in todos if t['completed']])

        return jsonify({
            "total_tasks": total,
            "completed": completed,
            "pending": total - completed,
            "completion_rate": f"{(completed/total*100):.1f}%" if total > 0
else "0%"
        })
    except Exception as e:
        return jsonify({"error": "Cannot connect to todo-app", "details": st
r(e)})

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5001)
```

➢ Gambar 8 : requirements.txt Stats-App

```
PS C:\Users\MSI Thin\to-do-list\stats-app> type requirements.txt
flask==2.3.3
requests==2.31.0
```

3.5 Docker Compose Configuration

➢ Gambar 9 : docker-compose.yml

```
PS C:\Users\MSI Thin\to-do-list> type docker-compose.yml
version: '3.8'

services:
  todo-app:
    build: ./todo-app
    ports:
      - "5000:5000"

  stats-app:
    build: ./stats-app
    ports:
      - "5001:5001"
```

Konfigurasi multi-container dengan:

- Version 3.8
- Service todo-app (port 5000)
- Service stats-app (port 5001)

- Automatic network creation

3.6 Container Deployment

➢ Gambar 10 : Docker Compose Up

```
PS C:\Users\MSI Thin\to-do-list> docker-compose up
time="2025-12-12T23:09:12+07:00" level=warning msg="C:\\Users\\MSI Thin\\to-
do-list\\docker-compose.yml: the attribute `version` is obsolete, it will be
 ignored, please remove it to avoid potential confusion"
Attaching to stats-app-1, todo-app-1
todo-app-1  |  * Serving Flask app 'app'
todo-app-1  |  * Debug mode: off
todo-app-1  | WARNING: This is a development server. Do not use it in a prod
uction deployment. Use a production WSGI server instead.
todo-app-1  |  * Running on all addresses (0.0.0.0)
todo-app-1  |  * Running on http://127.0.0.1:5000
todo-app-1  |  * Running on http://172.18.0.3:5000
todo-app-1  | Press CTRL+C to quit
stats-app-1  |  * Serving Flask app 'app'
stats-app-1  |  * Debug mode: off
stats-app-1  | WARNING: This is a development server. Do not use it in a pro
duction deployment. Use a production WSGI server instead.
stats-app-1  |  * Running on all addresses (0.0.0.0)
stats-app-1  |  * Running on http://127.0.0.1:5001
stats-app-1  |  * Running on http://172.18.0.2:5001
stats-app-1  | Press CTRL+C to quit
```

,

Output menunjukkan:

- Kedua container berhasil running
- todo-app pada http://172.18.0.3:5000
- stats-app pada http://172.18.0.2:5001
- Flask development server aktif

➢ Gambar 11 : Docker PS

```
PS C:\Users\MSI Thin> docker ps
CONTAINER ID   IMAGE                  COMMAND           CREATED       STATUS
         PORTS                                       NAMES
5863e046c780   to-do-list-stats-app   "python app.py"   6 hours ago   Up 3 m
inutes   0.0.0.0:5001->5001/tcp, [::]:5001->5001/tcp   to-do-list-stats-app-
1
cdafed9f4085   to-do-list-todo-app    "python app.py"   6 hours ago   Up 3 m
inutes   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp   to-do-list-todo-app-1
PS C:\Users\MSI Thin>
```

Container status menampilkan:

- todo-app container (Up 3 minutes)
- stats-app container (Up 3 minutes)
- Port mapping: 5000 dan 5001

3.7 Testing

➢ Gambar 12 : Test Endpoint /todos

```
PS C:\Users\MSI Thin> curl http://localhost:5000/todos


StatusCode        : 200
StatusDescription : OK
Content           : {"todos":[{"completed":false,"id":1,"task":"Belajar
                    Docker"},{"completed":true,"id":2,"task":"Buat
                    laporan"}],"total":2}

RawContent        : HTTP/1.1 200 OK
                    Connection: close
                    Content-Length: 121
                    Content-Type: application/json
                    Date: Fri, 12 Dec 2025 16:10:00 GMT
                    Server: Werkzeug/3.1.4 Python/3.9.25

                    {"todos":[{"completed":false,"id":...
Forms             : {}
Headers           : {[Connection, close], [Content-Length, 121],
                    [Content-Type, application/json], [Date, Fri, 12 Dec
                    2025 16:10:00 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 121
```

Response:

- StatusCode: 200 OK
- Content: 2 tasks (Belajar Docker, Buat laporan)
- Server: Werkzeug/3.1.4 Python/3.9.25

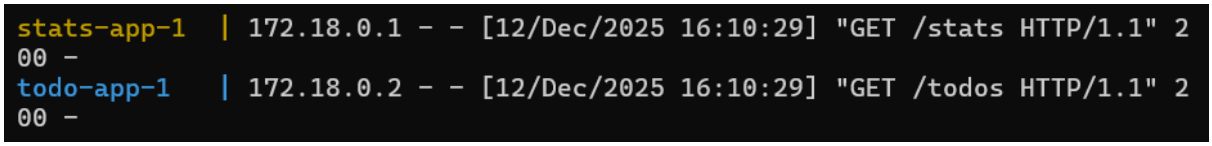
➢ Gambar 13 : Test Endpoint /stats

```
PS C:\Users\MSI Thin> curl http://localhost:5001/stats


StatusCode        : 200
StatusDescription : OK
Content           : {"completed":1,"completion_rate":"50.0%","pending":1,"t
                    otal_tasks":2}

RawContent        : HTTP/1.1 200 OK
                    Connection: close
                    Content-Length: 70
                    Content-Type: application/json
                    Date: Fri, 12 Dec 2025 16:10:29 GMT
                    Server: Werkzeug/3.1.4 Python/3.9.25

                    {"completed":1,"completion_rate":"5...
Forms             : {}
Headers           : {[Connection, close], [Content-Length, 70],
                    [Content-Type, application/json], [Date, Fri, 12 Dec
                    2025 16:10:29 GMT]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 70
```

Response:

- total_tasks: 2
- completed: 1
- pending: 1
- completion_rate: 50.0%

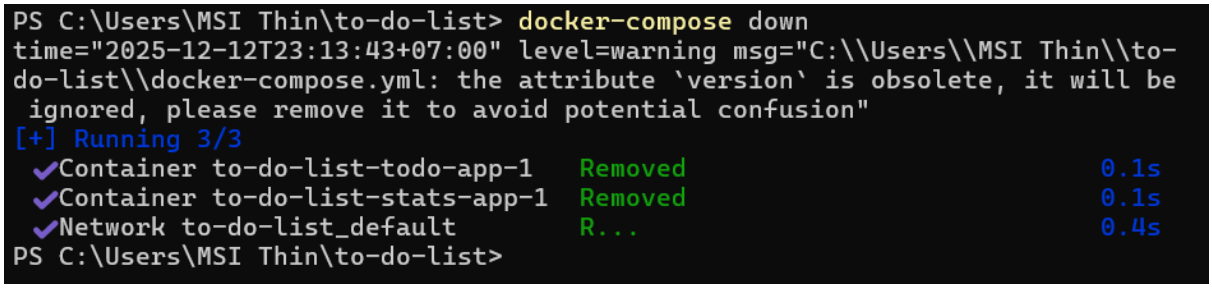
➢ Gambar 14 : Container Logs



Log menunjukkan:

- GET request ke /todos (200)
- GET request ke /stats (200)
- Inter-container communication berhasil

➢ Gambar 15 : Docker Compose Down



4. ANALISIS

4.1 Keberhasilan

- Aplikasi Flask berhasil dikembangkan dengan fitur lengkap
- Containerization menggunakan Docker berhasil diimplementasikan
- Multi-container communication berfungsi dengan baik
- Build process efisien dengan layer caching

4.2 Tantangan & Solusi

Challenge: Inter-container communication error
Solution: Menggunakan Docker Compose untuk shared network

4.3 Pembelajaran

- Pemahaman konsep containerization dan Docker
- Multi-container orchestration dengan Docker Compose
- Network isolation dan service discovery
- Best practices: slim images, layer caching

5. KESIMPULAN

Minggu ke-13 berhasil menyelesaikan target:

- **Pengembangan Aplikasi:** Todo-app dan stats-app dengan Flask
- **Containerization:** 2 Dockerfile dan images berhasil dibuat
- **Multi-Container:** Docker Compose orchestration berfungsi
- **Testing:** Semua endpoints tested dan working

-