

UJICoba MESSAGING PROTOCOLS SISTEM TERDISTRIBUSI

“Disusun dalam rangka memenuhi salah satu tugas individu
Mata Kuliah Sistem Terdistribusi”

Dosen Pengampu :

Achmad Basuki, ST., MMG., Ph.D.



Disusun Oleh :

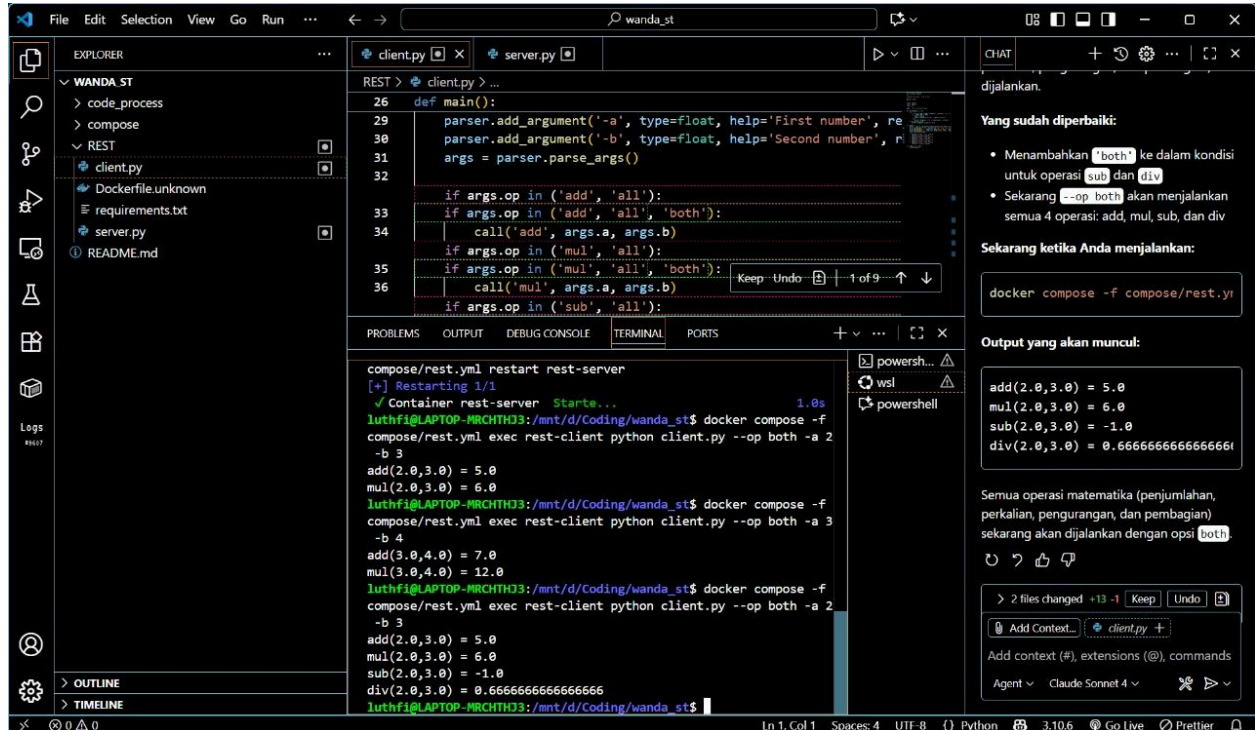
Wanda Azizah Subekti

2546000082

**PROGRAM STUDI MAGISTER ILMU KOMPUTER
DEPARTEMEN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA**

2025

REST



```
client.py
26 def main():
27     parser.add_argument('-a', type=float, help='First number', required=True)
28     parser.add_argument('-b', type=float, help='Second number', required=True)
29     args = parser.parse_args()
30
31     if args.op in ('add', 'all'):
32         call('add', args.a, args.b)
33     if args.op in ('sub', 'all'):
34         call('sub', args.a, args.b)
35     if args.op in ('mul', 'all'):
36         call('mul', args.a, args.b)
37     if args.op in ('div', 'all'):
38         call('div', args.a, args.b)
```

```
server.py
1 from flask import Flask, request, jsonify
2 app = Flask(__name__)
3
4 @app.route('/add', methods=['GET'])
5 def add():
6     a = request.args.get('a')
7     b = request.args.get('b')
8     result = a + b
9     return jsonify({'result': result})
10
11 @app.route('/sub', methods=['GET'])
12 def sub():
13     a = request.args.get('a')
14     b = request.args.get('b')
15     result = a - b
16     return jsonify({'result': result})
17
18 @app.route('/mul', methods=['GET'])
19 def mul():
20     a = request.args.get('a')
21     b = request.args.get('b')
22     result = a * b
23     return jsonify({'result': result})
24
25 @app.route('/div', methods=['GET'])
26 def div():
27     a = request.args.get('a')
28     b = request.args.get('b')
29     if b == 0:
30         return jsonify({'error': 'Division by zero'})
31     result = a / b
32     return jsonify({'result': result})
```

```
terminal
compose/rest.yml restart rest-server
[+] Restarting 1/1
Container rest-server Starting... 1.0s
luthfi@LAPTOP-MRCHTH33:/mnt/d/Coding/wanda_st$ docker compose -f
compose/rest.yml exec rest-client python client.py --op both -a 2
-b 3
add(2.0,3.0) = 5.0
mul(2.0,3.0) = 6.0
luthfi@LAPTOP-MRCHTH33:/mnt/d/Coding/wanda_st$ docker compose -f
compose/rest.yml exec rest-client python client.py --op both -a 3
-b 4
add(3.0,4.0) = 7.0
mul(3.0,4.0) = 12.0
luthfi@LAPTOP-MRCHTH33:/mnt/d/Coding/wanda_st$ docker compose -f
compose/rest.yml exec rest-client python client.py --op both -a 2
-b 3
add(2.0,3.0) = 5.0
mul(2.0,3.0) = 6.0
sub(2.0,3.0) = -1.0
div(2.0,3.0) = 0.6666666666666666
luthfi@LAPTOP-MRCHTH33:/mnt/d/Coding/wanda_st$
```

Kode program pada folder **REST** ini merupakan implementasi sederhana layanan **REST API kalkulator**.

- **server.py** berperan sebagai *backend* menggunakan framework Flask. Terdapat empat endpoint, yaitu:
 - **/add** untuk penjumlahan,
 - **/sub** untuk pengurangan,
 - **/mul** untuk perkalian,
 - **/div** untuk pembagian (dilengkapi validasi pembagian dengan nol).Setiap endpoint menerima parameter **a** dan **b** melalui *query string*, kemudian mengembalikan hasil perhitungan dalam format JSON melalui metode HTTP GET. Server dijalankan pada port 5151.

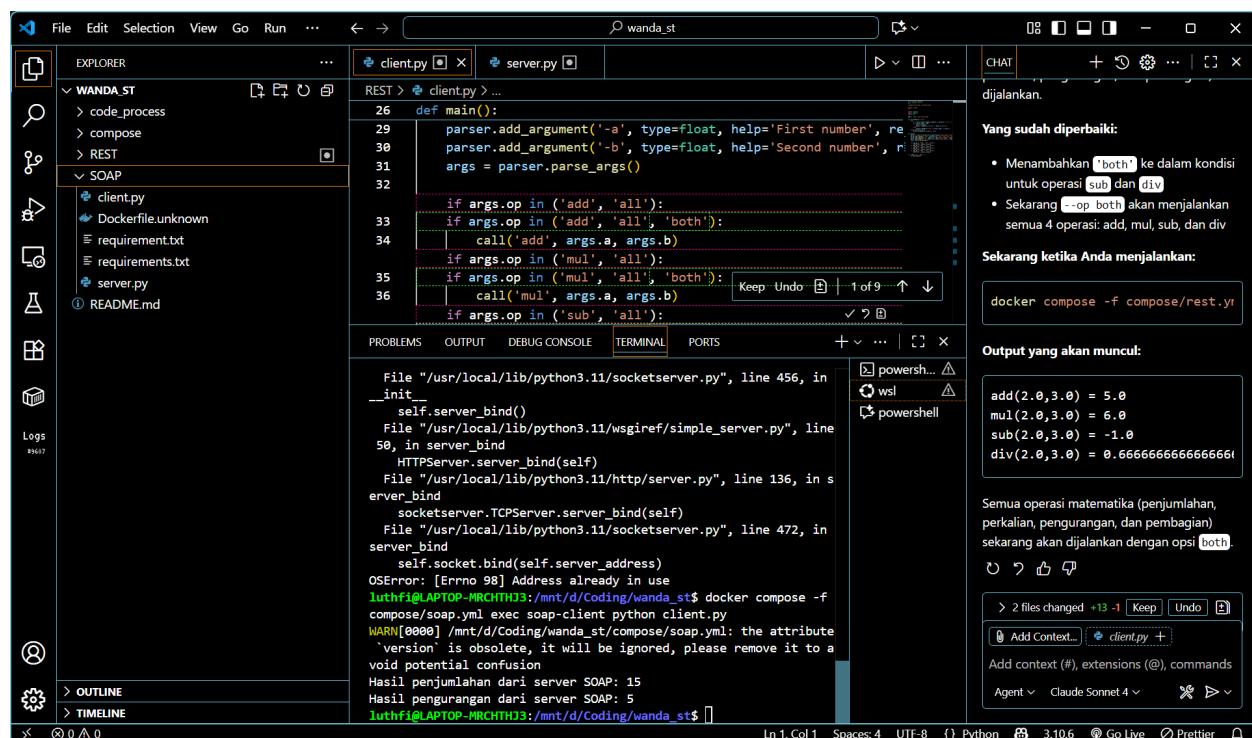
- **client.py** berfungsi sebagai *frontend client* berbasis CLI (Command Line Interface). Program ini mengirimkan permintaan ke server sesuai dengan operasi yang dipilih (**add**, **sub**, **mul**, **div**, atau **all**) menggunakan parameter **-a** dan **-b**. Hasil perhitungan dari

server ditampilkan kembali pada terminal.

- **requirements.txt** berisi daftar pustaka yang diperlukan, yaitu **Flask** sebagai penyedia layanan web dan **Requests** sebagai library HTTP client.
- **Dockerfile** digunakan untuk mengemas aplikasi server ke dalam *container*. File ini mengatur lingkungan Python, melakukan instalasi dependensi, menyalin kode program, membuka port 5151, serta mengeksekusi **server.py**.

Secara alur, server menyediakan layanan perhitungan aritmatika melalui API, kemudian client melakukan pemanggilan API tersebut, server memproses operasi, dan hasilnya dikirim kembali dalam format JSON untuk ditampilkan oleh client.

SOAP



Kode program pada folder **SOAP** merupakan implementasi layanan kalkulator berbasis **SOAP Web Service** menggunakan pustaka **Spyne** sebagai server dan **Zeep** sebagai client.

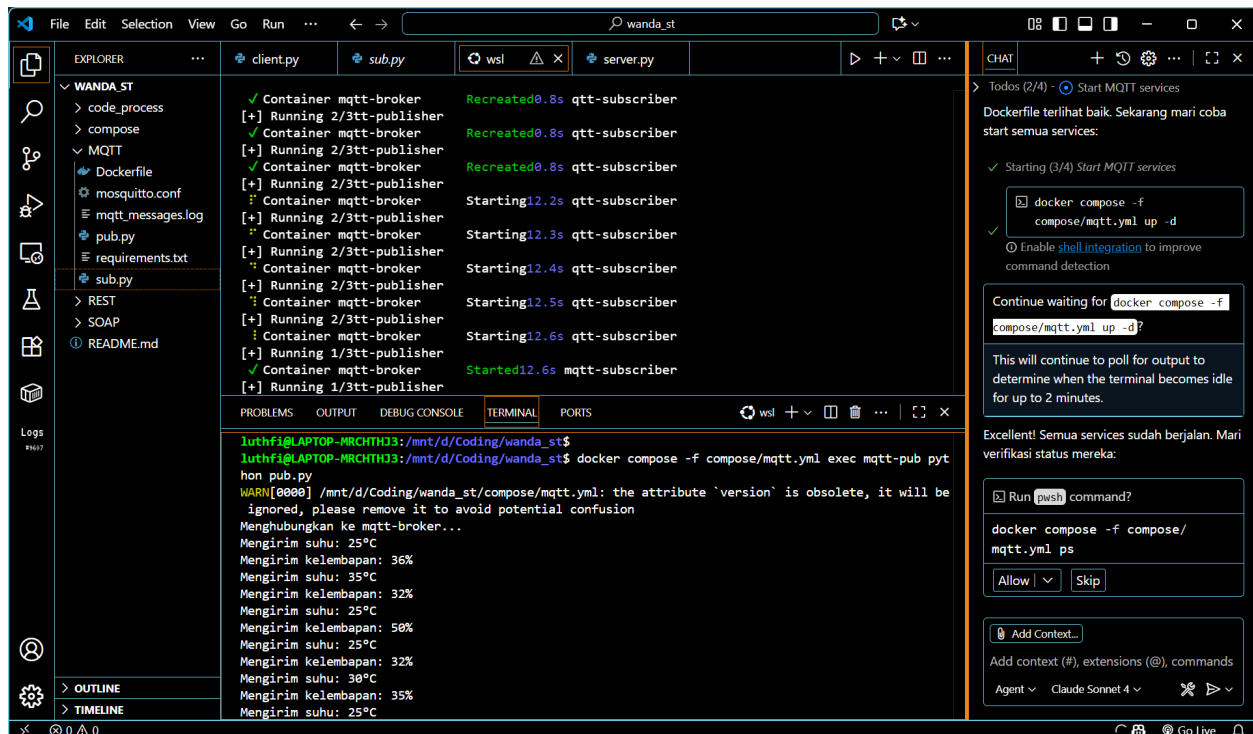
Pada **server.py**, didefinisikan kelas **CalculatorService** yang berisi dua metode, yaitu **add** untuk penjumlahan dan **subtract** untuk pengurangan. Keduanya menerima dua parameter bertipe integer dan mengembalikan hasil dalam bentuk integer. Layanan ini dijalankan dengan protokol **SOAP 1.1** melalui **WsgiApplication**, dan server dibuka pada alamat **0.0.0.0** port **8000** sehingga dapat diakses dari container lain.

Pada **client.py**, dibuat objek **Client** berdasarkan URL WSDL dari server SOAP. Melalui objek ini, client dapat memanggil metode yang tersedia, misalnya **add(10, 5)** dan **subtract(10, 5)**. Hasil perhitungan yang diterima dari server kemudian ditampilkan di terminal.

File **requirements.txt** berisi dependensi utama, yaitu **spyne** untuk membangun server SOAP, **zeep** untuk client SOAP, serta **requests** dan **lxml** untuk mendukung komunikasi dan pemrosesan XML. Sementara itu, **Dockerfile** digunakan untuk mengatur lingkungan container Python, menginstal dependensi, menyalin kode program, serta menyiapkan server pada port 8000.

Secara keseluruhan, alur program ini adalah server menyediakan layanan SOAP dengan operasi penjumlahan dan pengurangan, kemudian client memanggil layanan tersebut dan menampilkan hasil yang dikirimkan kembali oleh server.

MQTT



Kode program pada direktori **MQTT** ini merupakan implementasi komunikasi berbasis **Message Queuing Telemetry Transport (MQTT)**, yaitu protokol *publish/subscribe* yang umum digunakan pada Internet of Things (IoT).

File **pub.py** berfungsi sebagai *publisher* yang mengirimkan data secara periodik ke broker MQTT. Program ini menghasilkan nilai acak suhu (20–35 °C) dan kelembapan (30–50%), kemudian mempublikasikan data tersebut ke dua topik berbeda, yaitu **sister/temp** dan **sister/humidity**. Data dikirim setiap satu detik.

File **sub.py** berperan sebagai *subscriber*. Program ini terhubung ke broker, berlangganan pada topik `sister/temp` dan `sister/humidity`, lalu menampilkan pesan yang diterima di terminal. Selain itu, setiap pesan yang diterima juga disimpan ke dalam file `mqtt_messages.log` sebagai catatan (*logging*).

File **mosquitto.conf** digunakan untuk mengonfigurasi broker MQTT (dalam hal ini Mosquitto) agar berjalan pada port 1883, mengizinkan koneksi anonim, dan tidak menggunakan *persistence*.

File **requirements.txt** berisi dependensi utama yaitu `paho-mqtt`, pustaka Python untuk mengimplementasikan client MQTT.

Sedangkan **Dockerfile** berfungsi menyiapkan lingkungan eksekusi di dalam container Python, menginstal dependensi, menyalin kode program dari direktori MQTT, serta menjaga container tetap aktif.

Secara keseluruhan, alur kerja program adalah *publisher* mengirimkan data suhu dan kelembapan ke broker, kemudian *subscriber* menerima data tersebut melalui topik yang sesuai dan menampilkannya di terminal sekaligus menyimpannya ke file log.

Upcall

The screenshot displays the VS Code interface for the 'Upcall' project. The Explorer panel on the left shows the project structure with files like `clientcall.py`, `servercall.py`, `Dockerfile`, and `README.md`. The `servercall.py` file is open in the editor, showing a Python script for a MQTT server. The script defines a `server_program()` function that binds to `0.0.0.0:4141` and listens for incoming connections. The Output panel at the bottom shows the Docker build process, including the installation of dependencies and the execution of the `upcall` client. The Chat panel on the right shows a message from Claude Sonnet 4, providing instructions on how to run the client.

```
11 def server_program():
12     server_socket = socket.socket()
13     # Bind to all interfaces so client container can reach it
14     server_socket.bind(('0.0.0.0', 4141))
15
16     server_socket.listen(1)
17     print("Upcall server listening on 0.0.0.0:4141 (DNS: upcall-server:4141)")
18     conn, address = server_socket.accept()
19     print("Connection from:", address)
20
21     while True:
22         data = conn.recv(1024).decode()
23         if not data:
24             break
25         print("Received from client:", data)
26
27 # Simulate upcall (feeding an upcall message to client)
```

Output:

```
✓ upcall-server Built 0.0s
✓ upcall-client Built 0.0s
✓ Container upcall-server Started 1.0s
✓ Container upcall-client Started 1.2s
luthfi@LAPTOP-MRCHTHJ3:/mnt/d/Coding/wanda_st$ docker compose -f compose/upcall.yml
exec upcall-server python servercall.py
WARN[0000] /mnt/d/Coding/wanda_st/compose/upcall.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
Upcall server listening on 0.0.0.0:4141 (DNS: upcall-server:4141)
Connection from: ('172.18.0.6', 42002)
Received from client: wanda
Received from client: riri
```

Chat:

Todos (4/4)

terminal saat ini.

✓ Starting (4/4) Run upcall client

Solusi untuk menjalankan perintah kedua:

Server sudah berjalan dengan baik di terminal pertama dan listening pada port 4141. Untuk menjalankan client, Anda perlu:

1. Buka terminal baru (jangan tutup yang lama)
2. Di terminal baru, jalankan client:

cd /mnt/d/Coding/wanda_st
docker compose -f compose/upcall

Atau jika ingin menjalankan client di background:

```
# Di terminal yang sama dengan s
docker compose -f compose/upcall
```

Add context (#), extensions (@), commands

Agent Claude Sonnet 4

Kode program pada direktori **upcall** merupakan implementasi komunikasi berbasis **socket programming** antara client dan server.

File **servercall.py** berperan sebagai server. Program ini membuat socket yang terhubung pada alamat **0.0.0.0** port **4141**, sehingga dapat menerima koneksi dari client. Setelah koneksi terbentuk, server menunggu pesan dari client, lalu mencetak pesan tersebut ke terminal. Setiap pesan yang diterima akan direspons dengan mengirimkan kembali *upcall message* berupa teks **"Upcall event: Processing <data>"**.

File **clientcall.py** berfungsi sebagai client. Program ini mencoba terhubung ke server dengan timeout 10 detik. Setelah terhubung, client meminta input dari pengguna berupa beberapa pesan sekaligus yang dipisahkan dengan koma. Setiap pesan kemudian diproses secara terpisah: dikirim ke server, ditunggu balasannya, lalu hasil respons ditampilkan di terminal. Client juga mendukung perintah khusus **bye** untuk menutup koneksi. Semua aktivitas (pesan terkirim, pesan diterima, error, maupun penutupan koneksi) dicatat dalam file **client_log.txt** menggunakan modul logging.

File **Dockerfile** digunakan untuk menyalin seluruh kode dalam direktori **upcall** ke dalam container Python 3.11, menjaga container tetap berjalan, dan memungkinkan koneksi antar-service melalui Docker Compose.

Secara keseluruhan, modifikasi utama pada program client adalah penambahan fitur pemrosesan **multi-pesan dengan pemisah koma**, sehingga setiap pesan yang dimasukkan pengguna dikirim ke server satu per satu dan ditampilkan sebagai pesan terpisah di sisi server.