

NAMA : AULIA RIZA MUFITA  
NIM : 256150100111008  
KELAS : SISTEM TERDISTRIBUSI - B

## LK 01: Lingkungan Praktik Mandiri + Uji Coba Messaging Protocols

### MQTT (Message Queuing Telemetry Transport)

#### 1. docker ps

```
● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS   NAMES
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Pada tahap awal dijalankan perintah “**docker ps**” untuk melihat apakah ada container yang sedang berjalan. Hasilnya kosong (tidak ada container aktif). Ini memastikan bahwa belum ada service MQTT/container lain yang dijalankan.

#### 2. docker compose -f compose/mqtt.yml up -d

```
● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose -f compose/mqtt.yml up -d
WARN[0000] /home/auliariza/dist_sys-3/compose/mqtt.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 3/3
 ✓ Container mqtt-broker      Started
 ✓ Container mqtt-subscriber  Started
 ✓ Container mqtt-publisher   Started
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Perintah “**docker compose -f compose/mqtt.yml up -d**” digunakan untuk menjalankan file mqtt.yml yang berisi definisi beberapa service (broker, publisher, subscriber). Opsi -d berarti detached mode, container dijalankan di background tanpa mengunci terminal. Output menunjukkan Running 3/3, artinya ada tiga container yang berhasil dijalankan:

1. mqtt-broker: sebagai server broker MQTT (misalnya menggunakan Eclipse Mosquitto).
2. mqtt-subscriber: client yang berlangganan topik tertentu.
3. mqtt-publisher: client yang mengirimkan pesan ke topik tertentu.

```

● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED
STATUS        PORTS          NAMES
b8d96a8dea9d   compose-mqtt-sub                    "tail -f /dev/null"    About a minute ago
Up About a minute                               mqtt-subscriber
c8716a1996db   compose-mqtt-pub                    "tail -f /dev/null"    About a minute ago
Up About a minute                               mqtt-publisher
54c93d4b56ba   eclipse-mosquitto:2.0              "/docker-entrypoint. ...." About a minute ago
Up About a minute 1883/tcp      mqtt-broker
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$

```

Perintah “**docker ps**” dijalankan kembali untuk mengecek container yang baru aktif. Dari output terlihat 3 container:

- mqtt-subscriber
- mqtt-publisher
- mqtt-broker (berbasis image eclipse-mosquitto:2.0, port 1883 terbuka sebagai port standar MQTT).

### 3. # Start subscriber (listens on topic)

```
docker compose -f compose/mqtt.yml exec mqtt-sub python sub.py
```

## # Publish messages

```
docker compose -f compose/mqtt.yml exec mqtt-pub python pub.py
```

```
o auliariza@LAPTOP-SJR9B4HG:~/dist_sys$ docker compose -f compose/mqtt.yml exec mqtt-sub python sub.py
```

[illegible]

Perintah **"docker compose -f compose/mqtt.yml exec mqtt-sub python sub.py"** mengeksekusi script sub.py di dalam container mqtt-subscriber. Script sub.py bertugas

mendengarkan (listen) topik tertentu, dalam hal ini topik sister/temp. Dari output terlihat subscriber berhasil:

- Terhubung ke broker MQTT.
- Berlangganan (subscribe) ke topik sister/temp.
- Siap menerima pesan yang dikirim oleh publisher.

Subscriber menerima setiap pesan yang dikirim publisher melalui broker. Pada terminal subscriber terlihat pesan masuk berulang kali, yaitu "Received message: Suhu: 28°C (Topic: sister/temp)"

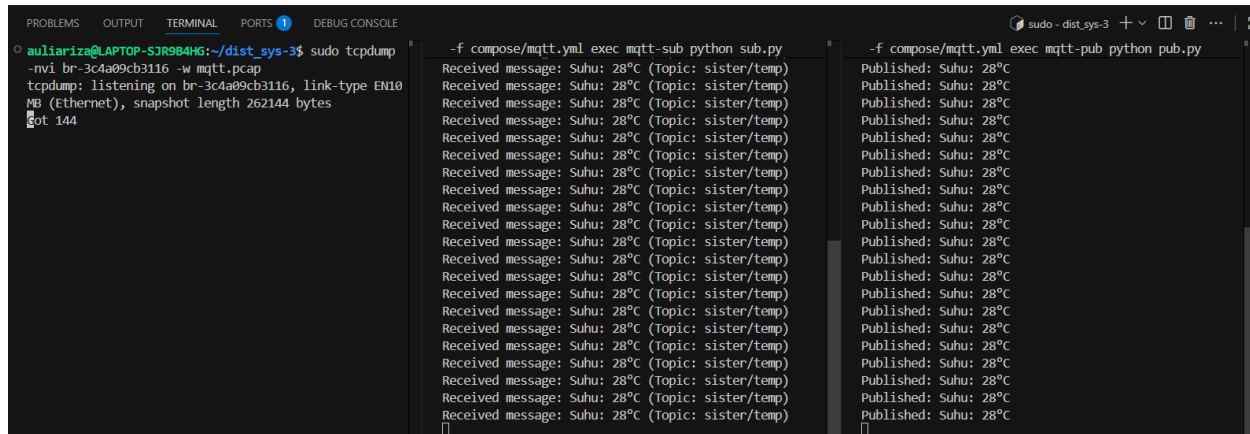
Perintah "**docker compose -f compose/mqtt.yml exec mqtt-pub python pub.py**" mengeksekusi script pub.py di dalam container mqtt-publisher. Script pub.py bertugas mengirim pesan (publish) ke broker dengan topik sister/temp. Dari output terlihat publisher mengirim pesan berulang, yaitu "Published: Suhu: 28°C".

4. ip a

```
auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ ip a
f link-netnsid 0
    inet6 fe80::18d1:bbff:fe7e:4377/64 scope link
        valid_lft forever preferred_lft forever
11: vethf83599b@if2: <BROADCAST,MULTICAST,UP,LOWER_UP
> mtu 1500 qdisc noqueue master br-3c4a09cb3116 state
UP group default
    link/ether a2:7a:a8:62:cc:f8 brd ff:ff:ff:ff:ff:f
f link-netnsid 1
    inet6 fe80::a07a:a8ff:fe62:ccf8/64 scope link
        valid_lft forever preferred_lft forever
12: vethd022270@if2: <BROADCAST,MULTICAST,UP,LOWER_UP
> mtu 1500 qdisc noqueue master br-3c4a09cb3116 state
UP group default
    link/ether 32:1e:f3:99:32:34 brd ff:ff:ff:ff:ff:f
f link-netnsid 2
    inet6 fe80::301e:f3ff:fe99:3234/64 scope link
        valid_lft forever preferred_lft forever
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

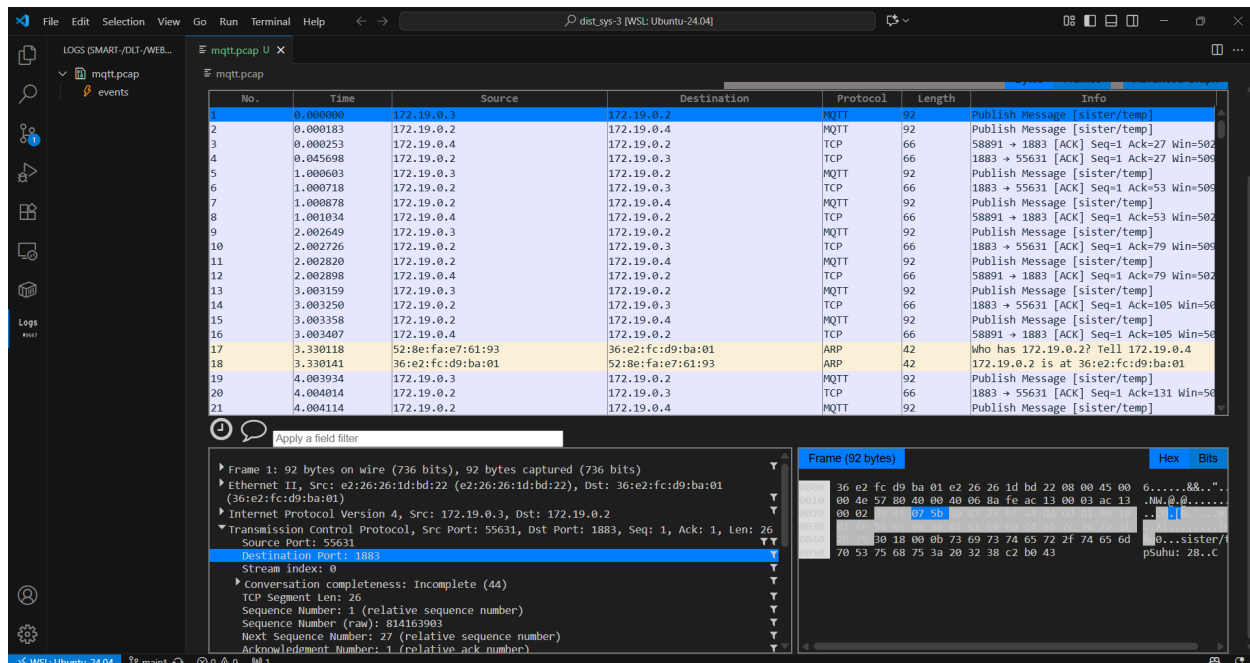
Perintah ini digunakan untuk melihat konfigurasi network interface pada host. Dari output terlihat adanya interface dengan nama "**br-3c4a09cb3116**", yaitu Docker bridge network yang dibuat secara otomatis oleh Docker Compose ketika menjalankan service MQTT. Interface ini berfungsi sebagai jembatan komunikasi antar container (broker, subscriber, publisher) agar dapat saling bertukar pesan dalam satu jaringan virtual.

```
5. sudo tcpdump -nvi br-3c4a09cb3116 -w mqtt.pcap
```



Perintah ini menjalankan tcpdump untuk menangkap paket data yang lewat di interface **“br-3c4a09cb3116”** sebagai bridge Docker tempat komunikasi antar container MQTT berlangsung. Pada terminal sebelah kiri terlihat bahwa tcpdump sedang aktif menangkap paket pada interface Docker. Jumlah paket yang ditangkap ditampilkan (misalnya 144 pkt). Di terminal subscriber (tengah) terlihat pesan terus diterima **“Received message: Suhu: 28°C (Topic: sister/temp)”** dan di terminal publisher (kanan) terlihat pesan terus dikirim **“Published: Suhu: 28°C”**. Dengan tcpdump, semua komunikasi ini (publish dan subscribe) tercatat dalam file **mqtt.pcap**.

6. Lihat hasil capture dengan double click file "mqtt.pcap"



Setelah menjalankan tcpdump, file mqtt.pcap dapat dibuka menggunakan Wireshark. Pada hasil capture, terlihat protokol MQTT berjalan di atas TCP. Kolom Protocol di Wireshark menampilkan label MQTT, dengan detail Publish Message (sister/temp). Ini menunjukkan bahwa paket tersebut adalah pesan yang dipublish ke topik sister/temp.

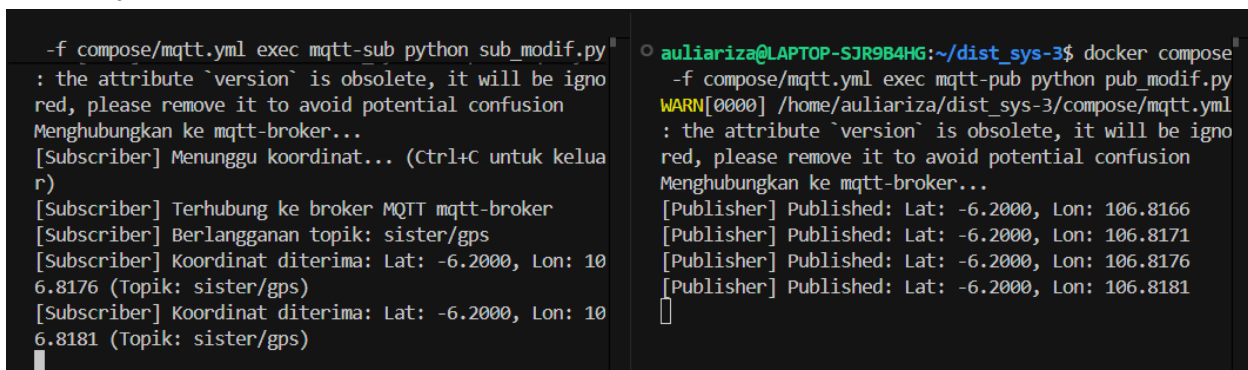
- Source (172.19.x.x) adalah container publisher.
- Destination (172.19.x.x) adalah container broker (mqtt-broker).
- Komunikasi dilakukan melalui port 1883, yaitu port standar protokol MQTT.

Publisher mengirim data ke broker, kemudian broker meneruskan ke subscriber yang berlangganan topik sister/temp. Dalam capture ini juga dapat diamati beberapa hal:

1. CONNECT (saat client terhubung ke broker).
2. SUBSCRIBE (saat subscriber berlangganan ke topik sister/temp).
3. PUBLISH (saat publisher mengirim pesan ke broker).
4. SUBACK/PUBACK (acknowledgement dari broker).

Yang dominan ditampilkan pada uji coba ini adalah paket PUBLISH, yaitu aliran data suhu dari publisher.

## 7. Uji coba modifikasi implementasi MQTT



```

-f compose/mqtt.yml exec mqtt-sub python sub_modif.py
: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
Menghubungkan ke mqtt-broker...
[Subscriber] Menunggu koordinat... (Ctrl+C untuk keluar)
[Subscriber] Terhubung ke broker MQTT mqtt-broker
[Subscriber] Berlangganan topik: sister/gps
[Subscriber] Koordinat diterima: Lat: -6.2000, Lon: 106.8176 (Topik: sister/gps)
[Subscriber] Koordinat diterima: Lat: -6.2000, Lon: 106.8181 (Topik: sister/gps)

o auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
-f compose/mqtt.yml exec mqtt-pub python pub_modif.py
WARN[0000] /home/auliariza/dist_sys-3/compose/mqtt.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
Menghubungkan ke mqtt-broker...
[Publisher] Published: Lat: -6.2000, Lon: 106.8166
[Publisher] Published: Lat: -6.2000, Lon: 106.8171
[Publisher] Published: Lat: -6.2000, Lon: 106.8176
[Publisher] Published: Lat: -6.2000, Lon: 106.8181

```

Pada uji coba modifikasi implementasi MQTT ini, terlihat bahwa skenario komunikasi antara publisher dan subscriber berhasil dijalankan dengan payload yang berbeda dari uji coba sebelumnya. Subscriber dijalankan dengan perintah “**python sub\_modif.py**”. Dari hasil di terminal, subscriber berhasil terhubung ke broker mqtt-broker, kemudian berlangganan (subscribe) ke topik baru yaitu **sister/gps**. Setelah itu subscriber menunggu pesan masuk, dan ketika publisher mulai mengirim data, subscriber menerima koordinat dalam format latitude dan longitude. Sementara itu, publisher dijalankan dengan perintah “**python pub\_modif.py**”. Publisher mengirimkan **data koordinat GPS** ke broker pada topik sister/gps. Output di terminal publisher menampilkan pesan yang dipublish. Hasil ini membuktikan bahwa alur komunikasi **publish-subscribe** MQTT berjalan dengan baik, di mana publisher mengirimkan data koordinat, broker meneruskan data tersebut, dan subscriber yang berlangganan topik sister/gps berhasil menerima pesan sesuai isi payload.

8. docker compose -f compose/mqtt.yml down

```
● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
  -f compose/mqtt.yml down
WARN[0000] /home/auliariza/dist_sys-3/compose/mqtt.yml
: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
[+] Running 3/3
  ✓ Container mqtt-subscriber   Removed      0.1s
  ✓ Container mqtt-publisher    Removed      0.1s
  ✓ Container mqtt-broker       Removed      0.0s
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Perintah ini digunakan untuk menghentikan dan menghapus semua container yang dijalankan dengan “**compose/mqtt.yml**”. Berbeda dengan “docker stop”, perintah “docker compose down” tidak hanya menghentikan tetapi juga menghapus container, network, dan resource lain yang terkait dengan file compose.

## REST (Representational State Transfer)

### 1. docker ps

```
● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS
PORTS         NAMES
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Pada awal uji coba dijalankan “**docker ps**” untuk memastikan kondisi awal environment. Hasilnya kosong, artinya belum ada container yang berjalan pada sistem. Tahap ini penting untuk memastikan tidak ada konflik container sebelum menjalankan REST service.

### 2. docker compose -f compose/rest.yml up -d

```
● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
-f compose/rest.yml up -d
[+] Running 3/3
✓ Network compose_default Created 0.3s
✓ Container rest-server Started 0.8s
✓ Container rest-client Started 1.0s
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Perintah “**docker compose -f compose/rest.yml up -d**” mengeksekusi file konfigurasi rest.yml untuk menjalankan container yang dibutuhkan. Opsi -d digunakan agar container berjalan di background (detached mode). Dari output terlihat:

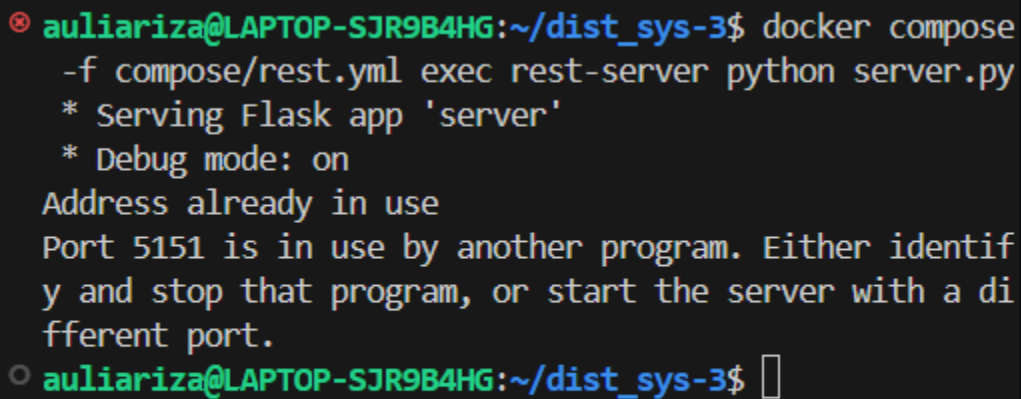
- Network compose\_default dibuat otomatis sebagai jaringan virtual.
- Container rest-server berhasil dijalankan.
- Container rest-client berhasil dijalankan.

```
● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS
PORTS         NAMES
4b77149b5cf7   compose-rest-client  "tail -f /dev/null"     About a minute ago Up About a minute
5151/tcp      rest-client
6c2709d2afcd   compose-rest-server  "python server.py"      About a minute ago Up About a minute
0.0.0.0:8080->5151/tcp, [::]:8080->5151/tcp rest-server
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Setelah menjalankan docker-compose, dilakukan pengecekan kembali dengan “**docker ps**”. Dari output terlihat container yang sedang aktif:

- rest-server
  - Menjalankan script server.py.
  - Terbuka pada port 8080 (5151/tcp).
  - Artinya server dapat menerima request melalui port 8080 (HTTP default untuk REST API).
- rest-client
  - Menjalankan command dummy (tail -f /dev/null) agar tetap hidup.
  - Akan digunakan untuk mengirimkan request ke rest-server.

3. `docker compose -f compose/rest.yml exec rest-server python server.py`



```
⊗ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
-f compose/rest.yml exec rest-server python server.py
* Serving Flask app 'server'
* Debug mode: on
Address already in use
Port 5151 is in use by another program. Either identif
y and stop that program, or start the server with a di
fferent port.
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Perintah “**docker compose -f compose/rest.yml exec rest-server python server.py**” digunakan untuk mengeksekusi langsung file server.py di dalam container rest-server. Script server.py adalah aplikasi Flask yang berfungsi sebagai REST API server. Dari log terlihat:

- Flask berhasil dijalankan dengan status: \* Serving Flask app 'server'
- Mode debug aktif: \* Debug mode: on



#### 4. ip a

```
auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ ip a
    link/ether f6:41:91:96:dd:e7 brd ff:ff:ff:ff:ff:ff
    link-netnsid 0
        inet6 fe80::f441:91ff:fe96:dde7/64 scope link
            valid_lft forever preferred_lft forever
28: veth831b04d@if2: <BROADCAST,MULTICAST,UP,LOWER_UP>
    mtu 1500 qdisc noqueue master br-5d49dbca27ec state UP
    group default
        link/ether 72:a4:3f:7c:33:f3 brd ff:ff:ff:ff:ff:ff
        link-netnsid 1
            inet6 fe80::70a4:3fff:fe7c:33f3/64 scope link
                valid_lft forever preferred_lft forever
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Perintah “ip a” digunakan untuk melihat daftar interface jaringan yang ada pada host. Dari hasil output terlihat sebuah network bridge yang dibuat otomatis oleh Docker dengan nama “br-5d49dbca27ec”. Bridge ini berfungsi sebagai jaringan virtual internal yang menghubungkan container REST (rest-server dan rest-client) sehingga bisa saling berkomunikasi. Container yang dijalankan oleh “docker compose -f rest.yml up -d” otomatis terhubung ke bridge ini, sehingga mereka berada di subnet yang sama dan dapat saling bertukar request-response HTTP.

#### 5. sudo tcpdump -nvi br-5d49dbca27ec -w rest.pcap

```
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ sudo tcpdump -
nvi br-5d49dbca27ec -w rest.pcap
tcpdump: listening on br-5d49dbca27ec, link-type EN10M
B (Ethernet), snapshot length 262144 bytes
Got 0
```

Perintah “sudo tcpdump -nvi br-5d49dbca27ec -w rest.pcap” digunakan untuk merekam lalu lintas jaringan yang terjadi pada bridge Docker “br-5d49dbca27ec” yang menghubungkan container REST. Sedangkan, “-w rest.pcap” untuk menyimpan hasil capture ke file .pcap yang dapat dibuka dengan Wireshark. Dengan ini, setiap komunikasi HTTP (request-response REST API) antara client dan server dapat direkam untuk analisis lebih lanjut.

6. `docker compose -f compose/rest.yml exec rest-client python client.py --op both -a 2 -b 3`

```
• auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
  -f compose/rest.yml exec rest-client python client.py
  --op both -a 2 -b 3
  add(2,3) = 5
  mul(2,3) = 6
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Perintah “**docker compose -f compose/rest.yml exec rest-client python client.py --op both -a 2 -b 3**” mengeksekusi `client.py` pada container `rest-client`. Script `client.py` mengirimkan request ke server REST dengan parameter operasi aritmatika. Opsi yang diberikan:

- `--op both`: meminta server melakukan penjumlahan (`add`) dan perkalian (`mul`) sekaligus.
- `-a 2 -b 3`: parameter angka yang akan dihitung.

Ini menunjukkan bahwa client berhasil mengirim request HTTP ke server, server memproses operasi, lalu mengembalikan hasil (response) ke client.

```
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ sudo tcpdump -
nvi br-5d49dbca27ec -w rest.pcap
tcpdump: listening on br-5d49dbca27ec, link-type EN10M
B (Ethernet), snapshot length 262144 bytes
Got 24
• auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
  -f compose/rest.yml exec rest-client python client.py
  --op both -a 2 -b 3
  add(2,3) = 5
  mul(2,3) = 6
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

```
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ sudo tcpdump -
nvi br-5d49dbca27ec -w rest.pcap
tcpdump: listening on br-5d49dbca27ec, link-type EN10M
B (Ethernet), snapshot length 262144 bytes
Got 78
mpose/rest.yml exec rest-client python client.py --op both -
a 2 -b 3
mul(2,3) = 6
• auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose -f co
mpose/rest.yml exec rest-client python client.py --op both -
a 2 -b 3
add(2,3) = 5
mul(2,3) = 6
• auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose -f co
mpose/rest.yml exec rest-client python client.py --op both -
a 2 -b 3
add(2,3) = 5
mul(2,3) = 6
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Client dijalankan beberapa kali dengan parameter yang sama. Terlihat jumlah paket yang ditangkap meningkat (misalnya 24 paket, lalu 78 paket) seiring banyaknya request-response yang dilakukan.

## 7. Lihat hasil capture dengan double click file “rest.pcap”

Wireshark interface showing a packet capture of a REST API interaction. The main pane displays a list of 21 frames. Frame 1 is a SYN packet from 172.19.0.3 to 172.19.0.2. Frame 2 is a SYN-ACK packet from 172.19.0.2 to 172.19.0.3. Frame 4 is an HTTP GET request from 172.19.0.3 to 172.19.0.2. Frame 5 is an HTTP 200 OK response from 172.19.0.2 to 172.19.0.3. The bottom pane shows the details of the selected frame (Frame 1), including Ethernet II, Internet Protocol Version 4, and TCP segments.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.19.0.3	172.19.0.2	TCP	74	60712 → 5151 [SYN] Seq=0 Win=64240 Len=0
2	0.000087	172.19.0.2	172.19.0.3	TCP	74	5151 → 60712 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
3	0.000118	172.19.0.3	172.19.0.2	TCP	66	60712 → 5151 [ACK] Seq=1 Ack=1 Win=64240 Len=0
4	0.000252	172.19.0.3	172.19.0.2	HTTP	224	GET /add?a=2&b=3 HTTP/1.1
5	0.000281	172.19.0.2	172.19.0.3	TCP	66	5151 → 60712 [ACK] Seq=1 Ack=159 Win=64240 Len=0
6	0.001811	172.19.0.2	172.19.0.3	TCP	232	5151 → 60712 [PSH, ACK] Seq=1 Ack=159 Win=64240 Len=0
7	0.001899	172.19.0.3	172.19.0.2	TCP	66	60712 → 5151 [ACK] Seq=159 Ack=167 Win=64240 Len=0
8	0.001959	172.19.0.2	172.19.0.3	HTTP/JSON	84	HTTP/1.1 200 OK, JSON (application/json)
9	0.001979	172.19.0.3	172.19.0.2	TCP	66	60712 → 5151 [ACK] Seq=159 Ack=185 Win=64240 Len=0
10	0.002648	172.19.0.3	172.19.0.2	TCP	66	60712 → 5151 [FIN, ACK] Seq=159 Ack=185 Win=64240 Len=0
11	0.002844	172.19.0.2	172.19.0.3	TCP	66	5151 → 60712 [FIN, ACK] Seq=185 Ack=160 Win=64240 Len=0
12	0.003482	172.19.0.3	172.19.0.2	TCP	66	60712 → 5151 [ACK] Seq=160 Ack=186 Win=64240 Len=0
13	0.005553	172.19.0.3	172.19.0.2	TCP	74	60728 → 5151 [SYN] Seq=0 Win=64240 Len=0
14	0.005670	172.19.0.2	172.19.0.3	TCP	74	5151 → 60728 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
15	0.005695	172.19.0.3	172.19.0.2	TCP	66	60728 → 5151 [ACK] Seq=1 Ack=1 Win=64240 Len=0
16	0.005790	172.19.0.3	172.19.0.2	HTTP	224	GET /mul?a=2&b=3 HTTP/1.1
17	0.005804	172.19.0.2	172.19.0.3	TCP	66	5151 → 60728 [ACK] Seq=1 Ack=159 Win=64240 Len=0
18	0.007579	172.19.0.2	172.19.0.3	TCP	232	5151 → 60728 [PSH, ACK] Seq=1 Ack=159 Win=64240 Len=0
19	0.007658	172.19.0.3	172.19.0.2	TCP	66	60728 → 5151 [ACK] Seq=159 Ack=167 Win=64240 Len=0
20	0.007715	172.19.0.2	172.19.0.3	HTTP/JSON	84	HTTP/1.1 200 OK, JSON (application/json)
21	0.007732	172.19.0.3	172.19.0.2	TCP	66	60728 → 5151 [ACK] Seq=159 Ack=185 Win=64240 Len=0

Setelah menjalankan tcpdump, file rest.pcap dapat dibuka menggunakan Wireshark. Pada hasil capture, terlihat bahwa:

- Topologi jaringan: sumber dan tujuan berada pada subnet Docker 172.19.0.x. Umumnya, rest-client memakai port ephemeral, server mendengar pada 172.19.0.2:5151 [HTTP di atas TCP].
- Sesi TCP: setiap interaksi diawali oleh three way handshake, paket SYN, SYN, ACK, ACK, lalu data aplikasi mengalir.
- Request HTTP: terlihat frame bertanda HTTP dari client ke server, berisi endpoint operasi aritmatika. Dari konteks eksekusi client, ini mewakili permintaan add dan mul dengan parameter a=2 dan b=3.
- Response HTTP 200 OK: balasan server ke client berstatus sukses, payload kecil yang memuat hasil perhitungan, konsisten dengan keluaran terminal, add(2,3)=5 dan mul(2,3)=6.
- Beberapa siklus request response: jumlah paket meningkat saat menjalankan client berkali kali, Wireshark menampilkan beberapa aliran TCP yang masing masing berisi rangkaian request dan response.
- Pengelolaan koneksi: pada akhir interaksi terlihat FIN, ACK atau reuse koneksi, tergantung implementasi server, yang berarti koneksi ditutup dengan rapi atau dipertahankan untuk permintaan berikutnya.
- Karakteristik muatan: ukuran payload kecil, konten text atau JSON sederhana, cocok untuk REST. Waktu antar frame rendah, menunjukkan latensi sangat kecil di jaringan bridge lokal.

## 8. Uji coba modifikasi implementasi REST

```
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
  -f compose/rest.yml exec rest-server python server_modif.py
  * Serving Flask app 'server_modif'
  * Debug mode: on
  WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
  * Running on all addresses (0.0.0.0)
  * Running on http://127.0.0.1:5252
  * Running on http://172.19.0.2:5252
  Press CTRL+C to quit
  * Restarting with stat
  * Debugger is active!
  * Debugger PIN: 132-658-588
172.19.0.3 - - [20/Sep/2025 02:45:42] "GET /add?a=8&b=4 HTTP/1.1" 200 -
172.19.0.3 - - [20/Sep/2025 02:45:42] "GET /sub?a=8&b=4 HTTP/1.1" 200 -
172.19.0.3 - - [20/Sep/2025 02:45:42] "GET /mul?a=8&b=4 HTTP/1.1" 200 -
172.19.0.3 - - [20/Sep/2025 02:45:42] "GET /div?a=8&b=4 HTTP/1.1" 200 -
172.19.0.3 - - [20/Sep/2025 02:46:07] "GET /sub?a=20&b=7 HTTP/1.1" 200 -
172.19.0.3 - - [20/Sep/2025 02:46:22] "GET /div?a=15&b=3 HTTP/1.1" 200 -
  □

● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
  -f compose/rest.yml exec rest-client python client_modif.py
  [ADD] 8 dan 4 -> 12
  [SUB] 8 dan 4 -> 4
  [MUL] 8 dan 4 -> 32
  [DIV] 8 dan 4 -> 2.0
● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
  -f compose/rest.yml exec rest-client python client_modif.py --op sub -a 20 -b 7
  [SUB] 20 dan 7 -> 13
● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
  -f compose/rest.yml exec rest-client python client_modif.py --op div -a 15 -b 3
  [DIV] 15 dan 3 -> 5.0
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ □
```

Pada uji coba modifikasi implementasi REST ini, server yang dijalankan adalah **“server\_modif.py”** yang telah diperluas sehingga mendukung beberapa endpoint baru yaitu /add, /sub, /mul, dan /div. Server berjalan menggunakan Flask pada port 5152 dan mencatat setiap request HTTP yang diterima. Dari sisi client, script **“client\_modif.py”** dijalankan untuk mengirimkan permintaan ke server dengan parameter angka tertentu. Hasil eksekusi menunjukkan bahwa komunikasi berjalan dengan baik, misalnya pada input a=4 dan b=2, client menerima respon berupa hasil penjumlahan dan pengurangan [ADD] 6 dan [SUB] 2. Ketika diuji dengan input a=8 dan b=2, client menerima hasil perkalian dan pembagian [MUL] 16 dan [DIV] 4. Uji coba tambahan dengan a=15 dan b=3 juga berhasil memberikan hasil pembagian [DIV] 5. Semua aktivitas ini tercatat pada log server dengan status HTTP/1.1" 200 yang menandakan permintaan berhasil diproses. Dari hasil ini dapat disimpulkan bahwa protokol REST mampu bekerja secara fleksibel, dimana penambahan endpoint baru dapat dilakukan dengan mudah dan setiap request dari client diproses server lalu dikembalikan sebagai response sesuai fungsi yang diminta.

9. `docker compose -f compose/rest.yml down`

```
● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
  -f compose/rest.yml down
[+] Running 3/3
  ✓ Container rest-client      Removed      10.5s
  ✓ Container rest-server     Removed      1.4s
  ✓ Network compose_default   Removed      0.3s
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Pada tahapan terakhir uji coba messaging protocol REST ini, dijalankan perintah “**docker compose -f compose/rest.yml down**”. Perintah tersebut digunakan untuk menghentikan seluruh container yang sebelumnya aktif (yaitu rest-client dan rest-server) sekaligus menghapus network virtual (compose\_default) yang dibuat secara otomatis oleh Docker Compose. Dari hasil eksekusi terlihat bahwa kedua container berhasil dihapus dengan status Removed, dan network juga sudah dilepas. Hal ini menandakan bahwa seluruh resource yang digunakan selama pengujian REST API telah dibersihkan, sehingga tidak ada lagi container maupun jaringan yang aktif di latar belakang.

## TCP REQ/RESP

### 1. docker ps

```
● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS
PORTS         NAMES
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Pada awal uji coba dijalankan “**docker ps**” untuk memastikan kondisi awal environment. Hasilnya kosong, artinya belum ada container yang berjalan pada sistem. Tahap ini penting untuk memastikan tidak ada konflik container sebelum menjalankan TCP REQ/RESP service.

### 2. docker compose -f compose/reqresp.yml up -d

```
-f compose/reqresp.yml up -d
=> => writing image sha256:5e9cc351e535388592a 0.0s
=> => naming to docker.io/library/compose-reqr 0.0s
=> [reqresp-client] exporting to image 0.2s
=> => exporting layers 0.1s
=> => writing image sha256:0ef9959cc1f1b1230ea 0.0s
=> => naming to docker.io/library/compose-reqr 0.0s
=> [reqresp-client] resolving provenance for m 0.0s
=> [reqresp-server] resolving provenance for m 0.0s
[+] Running 5/5
✓ compose-reqresp-server Built 0.0s
✓ compose-reqresp-client Built 0.0s
✓ Network compose_default Created 0.3s
✓ Container reqresp-server Started 0.8s
✓ Container reqresp-client Started 1.0s
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Perintah “**docker compose -f compose/reqresp.yml up -d**” digunakan untuk menjalankan service TCP REQ/RESP berdasarkan file konfigurasi reqresp.yml. Dari output terlihat:

- Image untuk reqresp-server dan reqresp-client berhasil dibangun (Built).
- Network compose\_default otomatis dibuat untuk menghubungkan container.
- Container reqresp-server dan reqresp-client berhasil dijalankan dalam mode background.

```

• auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
PORTS         NAMES
39b64f132962   compose-reqresp-client             "tail -f /dev/null"     28 minutes ago Up 28 minutes
2222/tcp      reqresp-client
b431ea2eca0e   compose-reqresp-server            "tail -f /dev/null"     28 minutes ago Up 28 minutes
2222/tcp      reqresp-server
• auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$

```

Perintah “**docker ps**” digunakan kembali untuk mengecek status container setelah docker-compose dijalankan. Terlihat dua container aktif:

1. reqresp-server: container server REQ/RESP, aktif di port 2222/tcp.
2. reqresp-client: container client REQ/RESP, siap mengirim request ke server.

Status keduanya adalah Up, menandakan service berhasil dijalankan.

3. `docker compose -f compose/reqresp.yml exec reqresp-server python server.py`  
`docker compose -f compose/reqresp.yml exec reqresp-client python client.py`

```

• auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose -f compose/reqresp.yml exec reqresp-server python server.py
WARN[0000] /home/auliariza/dist_sys-3/compose/reqresp.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
Server listening on 0.0.0.0:2222
Connection from: ('172.19.0.3', 53524)
█

• auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose -f compose/reqresp.yml exec reqresp-client python client.py
WARN[0000] /home/auliariza/dist_sys-3/compose/reqresp.yml: the attribute `version` is obsolete, it will be ignored, please remove it to avoid potential confusion
Enter message: █

```

Server:

Perintah “**docker compose -f compose/reqresp.yml exec reqresp-server python server.py**” mengeksekusi file server.py di dalam container reqresp-server. Dari output terlihat:

- Server mendengarkan pada port 2222 (Server listening on 0.0.0.0:2222).
- Saat ada koneksi dari client, server menampilkan informasi koneksi, misalnya: Connection from: ('172.19.0.3', 53524) yang menunjukkan alamat IP client di dalam jaringan Docker.

Client:

Perintah “**docker compose -f compose/reqresp.yml exec reqresp-client python client.py**” menjalankan client.py di dalam container reqresp-client. Client kemudian meminta input pesan dari user (Enter message:) untuk dikirimkan ke server. Setelah user memasukkan pesan, client akan mengirim request melalui TCP socket ke server, lalu server mengirim balik response.



#### 4. ip a

```
auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ ip a
    valid_lft forever preferred_lft forever
42: veth3d1be45@if2: <BROADCAST,MULTICAST,UP,LOWER_UP>
mtu 1500 qdisc noqueue master br-a9e7048a2f8d state UP
group default
    link/ether de:99:32:ea:25:67 brd ff:ff:ff:ff:ff:ff
link-netnsid 0
    inet6 fe80::dc99:32ff:feea:2567/64 scope link
        valid_lft forever preferred_lft forever
43: veth719fbbb@if2: <BROADCAST,MULTICAST,UP,LOWER_UP>
mtu 1500 qdisc noqueue master br-a9e7048a2f8d state UP
group default
    link/ether 92:1e:da:db:6e:14 brd ff:ff:ff:ff:ff:ff
link-netnsid 1
    inet6 fe80::901e:daff:fedb:6e14/64 scope link
        valid_lft forever preferred_lft forever
auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

```
auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
se -f compose/reqresp.yml exec reqresp-server python
server.py
WARN[0000] /home/auliariza/dist_sys-3/compose/reqres
p.yml: the attribute 'version' is obsolete, it will
be ignored, please remove it to avoid potential conf
usion
Server listening on 0.0.0.0:2222
Connection from: ('172.19.0.3', 55342)
[]
```

```
auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
-f compose/reqresp.yml exec reqresp-client python cli
ent.py
WARN[0000] /home/auliariza/dist_sys-3/compose/reqresp.
yml: the attribute 'version' is obsolete, it will be i
gnored, please remove it to avoid potential confusion
Enter message: []
```

Perintah “ip a” digunakan untuk melihat konfigurasi jaringan pada host. Terlihat adanya bridge network Docker yang dibuat otomatis ketika menjalankan docker-compose. Interface bridge ini berfungsi sebagai jembatan komunikasi antara container server dan client dalam satu network internal. Dari hasil “ip a”, IP address container berada pada subnet yang sama, sehingga komunikasi TCP dapat dilakukan dengan lancar.

#### 5. sudo tcpdump -nvi br-a9e7048a2f8d -w tcp.pcap

```
auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ ip a
    link/ether de:99:32:ea:25:67 brd ff:ff:ff:ff:ff:ff
link-netnsid 0
    inet6 fe80::dc99:32ff:feea:2567/64 scope link
        valid_lft forever preferred_lft forever
43: veth719fbbb@if2: <BROADCAST,MULTICAST,UP,LOWER_UP>
mtu 1500 qdisc noqueue master br-a9e7048a2f8d state UP
group default
    link/ether 92:1e:da:db:6e:14 brd ff:ff:ff:ff:ff:ff
link-netnsid 1
    inet6 fe80::901e:daff:fedb:6e14/64 scope link
        valid_lft forever preferred_lft forever
auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ sudo tcpdump -n
vi br-a9e7048a2f8d -w tcp.pcap
tcpdump: listening on br-a9e7048a2f8d, link-type EN10MB
(Ethernet), snapshot length 262144 bytes
^C
Got 10
```

```
auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
se -f compose/reqresp.yml exec reqresp-server python
server.py
WARN[0000] /home/auliariza/dist_sys-3/compose/reqres
p.yml: the attribute 'version' is obsolete, it will
be ignored, please remove it to avoid potential conf
usion
Server listening on 0.0.0.0:2222
Connection from: ('172.19.0.3', 55342)
Received from client: 1
Received from client: 2
Received from client: 3
[]
```

```
auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
-f compose/reqresp.yml exec reqresp-client python cli
ent.py
WARN[0000] /home/auliariza/dist_sys-3/compose/reqresp.
yml: the attribute 'version' is obsolete, it will be i
gnored, please remove it to avoid potential confusion
Enter message: 1
<class 'bytes'>
Received from server: Echo: 1
Enter another message: 2
<class 'bytes'>
Received from server: Echo: 2
Enter another message: 3
<class 'bytes'>
Received from server: Echo: 3
Enter another message: []
```

```
auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ ip a
    valid_lft forever preferred_lft forever
43: veth719fbbb@if2: <BROADCAST,MULTICAST,UP,LOWER_UP>
mtu 1500 qdisc noqueue master br-a9e7048a2f8d state UP
group default
    link/ether 92:1e:da:db:6e:14 brd ff:ff:ff:ff:ff:ff
link-netnsid 1
    inet6 fe80::901e:daff:fedb:6e14/64 scope link
        valid_lft forever preferred_lft forever
auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ sudo tcpdump -n
vi br-a9e7048a2f8d -w tcp.pcap
tcpdump: listening on br-a9e7048a2f8d, link-type EN10MB
(Ethernet), snapshot length 262144 bytes
^C10 packets captured
10 packets received by filter
0 packets dropped by kernel
auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

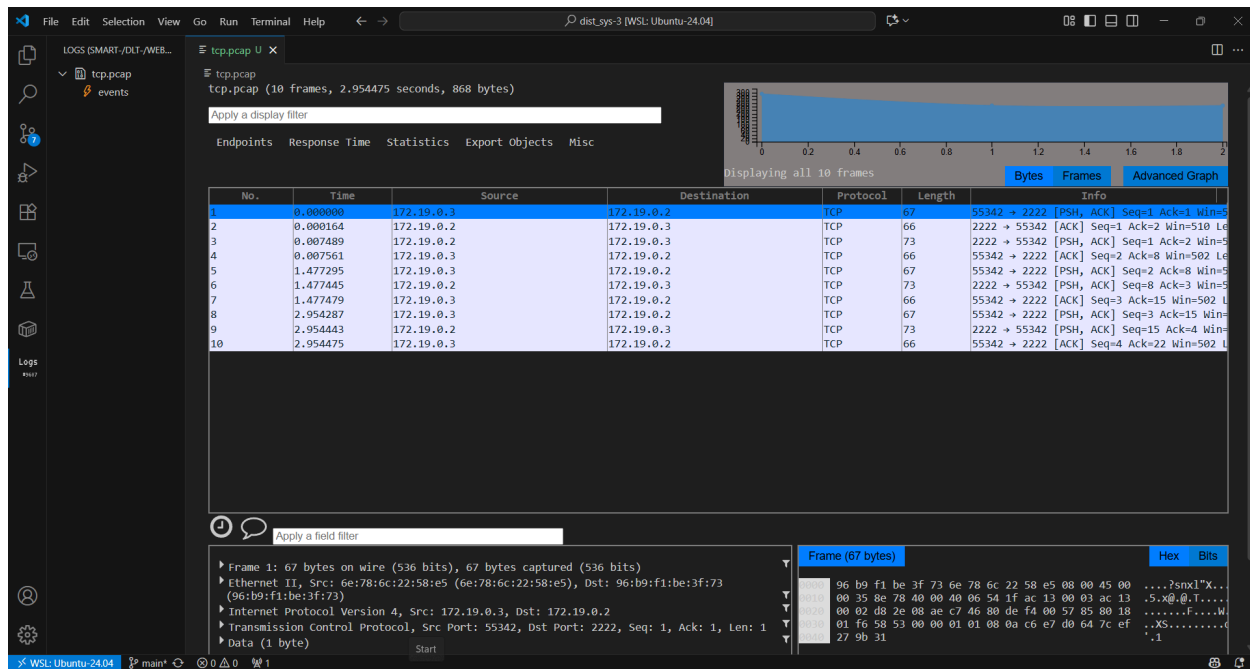
```
se -f compose/reqresp.yml exec reqresp-server python
server.py
be ignored, please remove it to avoid potential conf
usion
Server listening on 0.0.0.0:2222
Connection from: ('172.19.0.3', 55342)
Received from client: 1
Received from client: 2
Received from client: 3
^CTraceback (most recent call last):
  File "/app/server.py", line 34, in <module>
    server_program()
  File "/app/server.py", line 22, in server_program
    data = conn.recv(1024).decode()
    ~~~~~^^^^^^^^^^^^^^^^
KeyboardInterrupt
auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

```
-f compose/reqresp.yml exec reqresp-client python cli
ent.py
Enter another message: 2
<class 'bytes'>
Received from server: Echo: 2
Enter another message: 3
<class 'bytes'>
Received from server: Echo: 3
Enter another message: ^CTraceback (most recent call l
ast):
  File "/app/client.py", line 30, in <module>
    client_program()
  File "/app/client.py", line 25, in client_program
    message = input("Enter another message: ")
    ~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
KeyboardInterrupt
auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Perintah ini digunakan untuk merekam semua lalu lintas data TCP pada interface bridge Docker (br-a9e7048a2f8d). Opsi -w tcp.pcap menyimpan hasil capture ke file .pcap untuk dianalisis lebih lanjut menggunakan Wireshark. Output menunjukkan bahwa paket berhasil ditangkap ketika komunikasi antara client dan server berlangsung. Server dijalankan di container “reqresp-server” menerima pesan dari client, kemudian mengirimkan response kembali dari pesan yang dikirim client. Client dijalankan pada container “reqresp-client”. Setelah dijalankan, client meminta input user. Setiap pesan yang dikirim langsung diteruskan ke server. Client kemudian menerima balasan dari server. Terlihat pola request-response sudah berjalan dengan baik.



## 6. Lihat hasil capture dengan double click file "tcp.pcap"



Setelah menjalankan tcpdump, file tcp.pcap yang menunjukkan komunikasi end-to-end antara client dan server dengan arsitektur TCP REQ/RESP dapat dibuka menggunakan Wireshark. Pada hasil capture, terlihat bahwa semua paket yang ditangkap menggunakan TCP, terlihat jelas pada kolom Protocol. Komunikasi berlangsung antara IP 172.19.0.3 (client) dan 172.19.0.2 (server) pada port 2222, yang merupakan port layanan TCP REQ/RESP server. Dari tabel paket terlihat urutan interaksi standar TCP:

- SYN: Client membuka koneksi ke server (172.19.0.2:2222).
- SYN, ACK: Server merespons permintaan koneksi.
- ACK: Client mengonfirmasi, koneksi TCP berhasil terbentuk (three-way handshake selesai).
- Data (Payload): Client mengirim request (misalnya angka atau string input).
- ACK dari Server: Server mengakui data diterima.
- Data Response: Server mengirim balasan (echo dari request).
- ACK dari Client: Client mengakui balasan diterima.

Pada panel bawah (Hex dan ASCII), terlihat ada data 1 byte yang dikirim client ke server. Data ini sesuai dengan input sederhana yang dimasukkan pada client, misalnya angka atau string pendek. Ini membuktikan bahwa protokol REQ/RESP berjalan dengan baik.

## 7. Uji coba modifikasi implementasi TCP REQ/RESP

```
o auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
se -f compose/reqresp.yml exec reqresp-server python
server_modif.py
WARN[0000] /home/auliariza/dist_sys-3/compose/reqresp
p.yml: the attribute `version` is obsolete, it will be i
gnored, please remove it to avoid potential confu
sion
[SERVER AKTIF] Menunggu client di port 4444...
[TERHUBUNG] Client ('172.19.0.3', 40010) berhasil te
rhubung
[SERVER LOG] Dari ('172.19.0.3', 40010): 7 -> 7^2=49
[SERVER LOG] Dari ('172.19.0.3', 40010): 2, 5, 10 ->
2^2=4 | 5^2=25 | 10^2=100
[SERVER LOG] Dari ('172.19.0.3', 40010): 100 -> 100^
2=10000
[SERVER LOG] Dari ('172.19.0.3', 40010): 234 -> 234^
2=54756
█

o auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
-f compose/reqresp.yml exec reqresp-client python cli
ent_modif.py
WARN[0000] /home/auliariza/dist_sys-3/compose/reqresp.
yml: the attribute `version` is obsolete, it will be i
gnored, please remove it to avoid potential confusion
Ketik angka untuk dihitung kuadratnya (bisa lebih dari
satu, pisahkan dengan koma).
Ketik 'exit' untuk keluar.
Masukkan angka: 7
[HASIL SERVER] 7^2=49
Masukkan angka: 2, 5, 10
[HASIL SERVER] 2^2=4 | 5^2=25 | 10^2=100
Masukkan angka: 100
[HASIL SERVER] 100^2=10000
Masukkan angka: 234
[HASIL SERVER] 234^2=54756
Masukkan angka: █
```

Pada uji coba modifikasi TCP REQ RESP ini, server dijalankan dengan “**server\_modif.py**” dan mendengarkan pada port 4444, kemudian client dijalankan dengan “**client\_modif.py**”. Client mengirim angka sebagai request, satu nilai atau beberapa nilai yang dipisahkan koma, lalu server menghitung kuadrat tiap nilai dan mengirim balasan sebagai response. Dari log server terlihat koneksi dari 172.19.0.3 port ephemeral, lalu deretan request dan hasil, contoh 7 menjadi 49, permintaan berisi 2, 5, 10 menjadi 4, 25, 100, angka 100 menjadi 10000, dan 234 menjadi 54756. Tampilan di sisi client konsisten dengan log server, setiap input langsung mendapatkan hasil perhitungan. Ini memverifikasi pola REQ RESP berjalan benar, koneksi TCP terjalin, data diterima server, diproses, dan hasil dikirim kembali ke client dalam satu sesi yang persisten. Modifikasi terbukti menambah fungsionalitas, sekarang server dapat memproses banyak angka dalam satu pesan serta tetap responsif untuk permintaan berikutnya.

## 8. docker compose -f compose/reqresp.yml down

```
● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
-f compose/reqresp.yml down
WARN[0000] /home/auliariza/dist_sys-3/compose/reqresp.y
ml: the attribute `version` is obsolete, it will be ign
ored, please remove it to avoid potential confusion
[+] Running 3/3
  ✓ Container reqresp-client Removed 10.6s
  ✓ Container reqresp-server Removed 11.3s
  ✓ Network compose_default Removed 0.4s
o auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ █
```

Pada tahap terakhir uji coba TCP REQ/RESP ini dijalankan perintah “**docker compose -f compose/reqresp.yml down**” untuk menghentikan seluruh layanan yang telah dijalankan. Perintah ini menutup dan menghapus container **reqresp-client** serta **reqresp-server**, sekaligus

menghapus network **compose\_default** yang menghubungkan keduanya. Dengan demikian, environment pengujian dibersihkan sehingga tidak ada container atau network yang tersisa dan sistem kembali ke kondisi awal. Tahap ini penting untuk menutup siklus uji coba dengan rapi serta mencegah konflik pada pengujian berikutnya.

## RPC (Remote Procedure Call)

### 1. docker ps

```
● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED   STATUS
PORTS         NAMES
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Pada awal uji coba dijalankan “**docker ps**” untuk memastikan kondisi awal environment. Hasilnya kosong, artinya belum ada container yang berjalan pada sistem. Tahap ini penting untuk memastikan tidak ada konflik container sebelum menjalankan RPC service.

### 2. docker compose -f compose/rpc.yml up -d

```
● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker comp
ose -f compose/rpc.yml up -d
[+] Running 3/3
 ✓ Network compose_default Created 0.2s
 ✓ Container rpc-server Started 0.7s
 ✓ Container rpc-client Started 0.9s
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

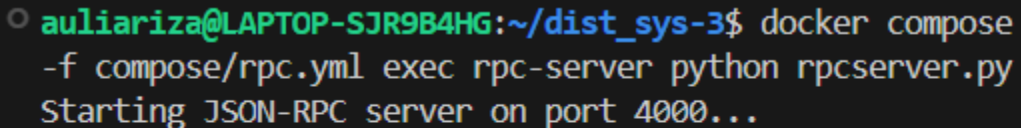
```
● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED
STATUS        PORTS      NAMES
f69e8b84a225   compose-rpc-client "tail -f /dev/null"    21 seconds ago
Up 20 seconds  4000/tcp   rpc-client
d17ff160567b   compose-rpc-server "tail -f /dev/null"    21 seconds ago
Up 20 seconds  4000/tcp   rpc-server
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Pada tahapan ini dilakukan perintah “**docker compose -f compose/rpc.yml up -d**” untuk menjalankan layanan RPC (Remote Procedure Call) dalam mode background. Dari hasil eksekusi terlihat bahwa Docker Compose membuat **network compose\_default** sebagai jaringan internal, kemudian berhasil menjalankan dua container utama yaitu “**rpc-server**” dan “**rpc-client**”. Setelah itu, diperiksa menggunakan perintah **docker ps** dan terbukti kedua

container aktif dengan status **Up**, di mana rpc-server dan rpc-client berjalan menggunakan port **4000/tcp**.

Tahap ini memastikan bahwa environment uji coba RPC sudah siap: server berjalan sebagai penyedia layanan prosedur jarak jauh, sementara client siap melakukan request untuk memanggil fungsi/prosedur yang ada di server. Dengan demikian, konektivitas antar komponen RPC sudah terbentuk dan siap untuk diuji pada langkah berikutnya.

3. `docker compose -f compose/rpc.yml exec rpc-server python rpcserver.py`

A terminal window with a dark background. The prompt is 'auliariza@LAPTOP-SJR9B4HG:~/dist\_sys-3\$'. The command entered is 'docker compose -f compose/rpc.yml exec rpc-server python rpcserver.py'. The output is 'Starting JSON-RPC server on port 4000...' followed by a cursor.

```
auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose  
-f compose/rpc.yml exec rpc-server python rpcserver.py  
Starting JSON-RPC server on port 4000...
```

Perintah tersebut mengeksekusi file “**rpcserver.py**” di dalam container **rpc-server**. Dari hasil keluaran terlihat server berhasil dijalankan dengan pesan “**Starting JSON-RPC server on port 4000...**”.

Artinya, server RPC sudah aktif dan siap menerima request dari client melalui port **4000** menggunakan protokol **JSON-RPC**. Server ini akan bertugas mengeksekusi fungsi/prosedur yang diminta client secara jarak jauh (remote procedure call), lalu mengembalikan hasilnya.

Tahap ini merupakan langkah penting karena memastikan **komponen server** sudah dalam keadaan siap melayani permintaan, sebelum uji coba komunikasi RPC dilakukan oleh client.

#### 4. ip a

```
auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ ip a
    link/ether 7a:b6:cf:b0:c6:14 brd ff:ff:ff:ff:ff:ff
    inet 172.19.0.1/16 brd 172.19.255.255 scope global br-464f72625a7c
        valid_lft forever preferred_lft forever
    inet6 fe80::78b6:cfff:feb0:c614/64 scope link
        valid_lft forever preferred_lft forever
97: vethde3efc4@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-464f72625a7c state UP group default
    link/ether 6e:ae:31:62:0e:3c brd ff:ff:ff:ff:ff:ff
    ff link-netnsid 0
    inet6 fe80::6cae:31ff:fe62:e3c/64 scope link
        valid_lft forever preferred_lft forever
98: veth7484f96@if2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue master br-464f72625a7c state UP group default
    link/ether 36:5b:1e:45:2f:9b brd ff:ff:ff:ff:ff:ff
    ff link-netnsid 1
    inet6 fe80::345b:1eff:fe45:2f9b/64 scope link
        valid_lft forever preferred_lft forever
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Perintah ini digunakan untuk melihat daftar interface jaringan yang aktif pada sistem. Dari hasil output terlihat adanya sebuah **bridge network Docker** dengan nama “**br-464f72625a7c**”. Bridge ini adalah jaringan virtual yang otomatis dibuat oleh Docker Compose saat menjalankan layanan **RPC** (server dan client).

Interface bridge ini berfungsi sebagai penghubung antara container **rpc-server** dan **rpc-client** agar keduanya dapat saling berkomunikasi. Dengan adanya bridge tersebut, client dapat mengirimkan request JSON-RPC ke server melalui port **4000**, dan server dapat memberikan response balik melalui jalur komunikasi yang sama.

Tahap ini penting untuk memverifikasi bahwa jaringan internal Docker telah terbentuk dan siap digunakan sebagai media komunikasi RPC antara client dan server.

5. `sudo tcpdump -nvi br-464f72625a7c -w rpc.pcap`

```
o auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ sudo tcpdump
-nvi br-464f72625a7c -w rpc.pcap
tcpdump: listening on br-464f72625a7c, link-type EN1
0MB (Ethernet), snapshot length 262144 bytes
Got 0
```

Tcpdump digunakan untuk **merekam lalu lintas jaringan** pada interface bridge Docker “**br-464f72625a7c**”, yang menghubungkan container rpc-server dan rpc-client. Hasil capture disimpan dalam file rpc.pcap agar bisa dianalisis lebih lanjut menggunakan Wireshark. Setelah client dijalankan, jumlah paket meningkat, menandakan komunikasi RPC berhasil ditangkap.

6. `docker compose -f compose/rpc.yml exec rpc-client python rpcclient.py`

```
o auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ sudo tcpdump
-nvi br-464f72625a7c -w rpc.pcap
tcpdump: listening on br-464f72625a7c, link-type EN1
0MB (Ethernet), snapshot length 262144 bytes
Got 116

o auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
-f compose/rpc.yml exec rpc-server python rpcserver.py
Starting JSON-RPC server on port 4000...
172.19.0.3 - - [20/Sep/2025 05:52:01] "POST / HTTP/1.1"
200 -
172.19.0.3 - - [20/Sep/2025 05:52:01] "POST / HTTP/1.1"
200 -
172.19.0.3 - - [20/Sep/2025 05:52:07] "POST / HTTP/1.1"
200 -
172.19.0.3 - - [20/Sep/2025 05:52:07] "POST / HTTP/1.1"
200 -
172.19.0.3 - - [20/Sep/2025 05:52:12] "POST / HTTP/1.1"
200 -
172.19.0.3 - - [20/Sep/2025 05:52:12] "POST / HTTP/1.1"
200 -
172.19.0.3 - - [20/Sep/2025 05:52:33] "POST / HTTP/1.1"
200 -
172.19.0.3 - - [20/Sep/2025 05:52:33] "POST / HTTP/1.1"
200 -
[]

o auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
-f compose/rpc.yml exec rpc-client python rpcclient.p
y
Result of add: 5.0
Result of multiply: 50
o auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
-f compose/rpc.yml exec rpc-client python rpcclient.p
y
Result of add: 5.0
Result of multiply: 50
o auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
-f compose/rpc.yml exec rpc-client python rpcclient.p
y
Result of add: 5.0
Result of multiply: 50
o auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Client menjalankan script rpcclient.py untuk mengirim permintaan ke server RPC. Dari output terlihat client memanggil dua prosedur jarak jauh di server, yaitu **add** dan **multiply**. Hasil yang diterima client:

- Result of add: 5.0
- Result of multiply: 50

Log server menunjukkan adanya request POST dari client (POST / HTTP/1.1) dengan payload JSON-RPC yang berisi method dan parameter. Ini membuktikan bahwa client berhasil mengirim request ke server, server memproses fungsi yang diminta, lalu mengembalikan hasil ke client.

## 7. Lihat hasil capture dengan double click file “rpc.pcap”

Wireshark capture of rpc.pcap (116 frames, 31.234587 seconds, 13016 bytes). The packet list shows the following traffic:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	66:36:c2:39:03:15	Broadcast	ARP	42	who has 172.19.0.2? Tell 172.19.0.3
2	0.000297	5a:f3:e6:aa:46:a9	66:36:c2:39:03:15	ARP	42	172.19.0.2 is at 5a:f3:e6:aa:46:a9
3	0.001187	172.19.0.3	172.19.0.2	TCP	74	41638 → 4000 [SYN] Seq=0 Win=64240 Len=0
4	0.001288	172.19.0.2	172.19.0.3	TCP	74	4000 → 41638 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0
5	0.001394	172.19.0.3	172.19.0.2	TCP	66	41638 → 4000 [ACK] Seq=1 Ack=1 Win=64240 Len=0
6	0.002713	172.19.0.3	172.19.0.2	TCP	265	41638 → 4000 [PSH, ACK] Seq=1 Ack=1 Win=0 Len=0
7	0.002778	172.19.0.2	172.19.0.3	TCP	66	4000 → 41638 [ACK] Seq=1 Ack=200 Win=65535 Len=0
8	0.002833	172.19.0.3	172.19.0.2	HTTP/JSON	129	POST / HTTP/1.1, JSON (application/json)
9	0.002849	172.19.0.2	172.19.0.3	TCP	66	4000 → 41638 [ACK] Seq=1 Ack=263 Win=65535 Len=0
10	0.007071	172.19.0.2	172.19.0.3	TCP	191	4000 → 41638 [PSH, ACK] Seq=1 Ack=263 Win=65535 Len=0
11	0.007111	172.19.0.3	172.19.0.2	TCP	66	41638 → 4000 [ACK] Seq=263 Ack=126 Win=65535 Len=0
12	0.008426	172.19.0.2	172.19.0.3	TCP	108	4000 → 41638 [PSH, ACK] Seq=126 Ack=263 Win=65535 Len=0
13	0.008512	172.19.0.3	172.19.0.2	TCP	66	41638 → 4000 [ACK] Seq=263 Ack=168 Win=65535 Len=0
14	0.008627	172.19.0.2	172.19.0.3	HTTP/JSON	66	HTTP/1.0 200 OK, JSON (application/json)
15	0.008823	172.19.0.3	172.19.0.2	TCP	66	41638 → 4000 [FIN, ACK] Seq=263 Ack=169 Win=65535 Len=0
16	0.009083	172.19.0.2	172.19.0.3	TCP	66	4000 → 41638 [ACK] Seq=169 Ack=264 Win=65535 Len=0
17	0.013184	172.19.0.3	172.19.0.2	TCP	74	41640 → 4000 [SYN] Seq=0 Win=64240 Len=0
18	0.013756	172.19.0.2	172.19.0.3	TCP	74	4000 → 41640 [SYN, ACK] Seq=0 Ack=1 Win=0 Len=0
19	0.013818	172.19.0.3	172.19.0.2	TCP	66	41640 → 4000 [ACK] Seq=1 Ack=1 Win=64240 Len=0
20	0.014694	172.19.0.3	172.19.0.2	TCP	265	41640 → 4000 [PSH, ACK] Seq=1 Ack=1 Win=65535 Len=0
21	0.014726	172.19.0.2	172.19.0.3	TCP	66	4000 → 41640 [ACK] Seq=1 Ack=200 Win=65535 Len=0

The packet details pane shows the selected packet (No. 14) with the following fields:

- Frame 1: 42 bytes on wire (336 bits), 42 bytes captured (336 bits)
- Ethernet II, Src: 66:36:c2:39:03:15 (66:36:c2:39:03:15), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- Address Resolution Protocol (request)

The packet bytes pane shows the raw data of the selected packet (42 bytes):

```
ff ff ff ff ff ff 66 36 c2 39 03 15 08 06 00 01 .....f6.9...
08 00 06 04 00 01 66 36 c2 39 03 15 ac 13 00 03 .....f6.9...
00 00 00 00 00 00 ac 13 00 02
```

Pada hasil capture file “rpc.pcap” di Wireshark terlihat bahwa komunikasi RPC berhasil berjalan dengan baik menggunakan protokol **TCP di atas HTTP** dengan payload berbentuk **JSON-RPC**. Dari daftar paket, komunikasi dimulai dengan **TCP handshake** (SYN, SYN-ACK, ACK) antara client 172.19.0.3 dan server 172.19.0.2 pada port **4000**, lalu dilanjutkan dengan pertukaran pesan request-response. Client mengirimkan **HTTP POST** berisi perintah JSON-RPC, misalnya pemanggilan method add dengan parameter [2,3], dan server merespons dengan **HTTP/1.1 200 OK** berisi hasil perhitungan 5. Pola serupa juga terjadi untuk method lain, misalnya multiply dengan parameter [5,10] yang menghasilkan response 50.

Dari analisis lebih detail dapat disimpulkan:

- Payload JSON-RPC terbaca di dalam POST request dan response, sesuai dengan hasil yang tampil di sisi client (Result of add: 5.0, Result of multiply: 50).
- Karakteristik jaringan menunjukkan tidak ada error atau retransmission, dengan latensi sangat rendah karena berjalan pada Docker bridge network lokal.

Dengan demikian, hasil capture ini membuktikan bahwa protokol **RPC berbasis JSON-RPC** sudah tervalidasi end-to-end, baik dari sisi aplikasi maupun di level jaringan, di mana server mampu menerima permintaan client, memproses prosedur yang diminta, dan mengirimkan kembali hasil eksekusi dengan benar.



## 8. Uji coba modifikasi implementasi TCP REQ/RESP

```

○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
-f compose/rpc.yml exec rpc-server python rpcserver_mod
if.py
Starting JSON-RPC server on port 4000...
172.19.0.3 - - [20/Sep/2025 06:13:41] "POST / HTTP/1.1"
200 -
172.19.0.3 - - [20/Sep/2025 06:13:41] "POST / HTTP/1.1"
200 -
172.19.0.3 - - [20/Sep/2025 06:13:41] "POST / HTTP/1.1"
200 -
172.19.0.3 - - [20/Sep/2025 06:13:56] "POST / HTTP/1.1"
200 -
172.19.0.3 - - [20/Sep/2025 06:13:56] "POST / HTTP/1.1"
200 -
172.19.0.3 - - [20/Sep/2025 06:13:56] "POST / HTTP/1.1"
200 -
□

● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
-f compose/rpc.yml exec rpc-client python rpcclient_m
odif.py
Enter a sentence: hello world
Number of vowels: 3
Number of consonants: 7
Number of words: 2

● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose
-f compose/rpc.yml exec rpc-client python rpcclient_m
odif.py
Enter a sentence: Aulia Riza
Number of vowels: 6
Number of consonants: 3
Number of words: 2

○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ █

```

Pada uji coba modifikasi implementasi **RPC** ini, server dijalankan dengan script **“rpcserver\_modif.py”** dan client menggunakan **“rpcclient\_modif.py”**. Modifikasi yang dilakukan adalah menambahkan fungsi baru di server untuk memproses **kalimat** yang dikirim client. Client mengirimkan sebuah kalimat, kemudian server mengolahnya untuk menghitung jumlah huruf vokal, jumlah huruf konsonan, serta jumlah kata. Hasil pengujian menunjukkan:

- Input *"hello world"* menghasilkan 3 vokal, 7 konsonan, dan 2 kata.
- Input *"Aulia Riza"* menghasilkan 6 vokal, 3 konsonan, dan 2 kata.

Log server memperlihatkan adanya request **HTTP POST** dari client ke server melalui protokol JSON-RPC, dan setiap permintaan direspons dengan status **200 OK**, menandakan bahwa komunikasi berhasil tanpa error. Dengan demikian, uji coba ini memperlihatkan bahwa RPC dapat dengan mudah diperluas untuk berbagai kebutuhan, baik perhitungan numerik maupun analisis teks.

9. `docker compose -f compose/rpc.yml down`

```
● auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$ docker compose -f compose/rpc.yml down
[+] Running 3/3
  ✓ Container rpc-client      Removed      12.9s
  ✓ Container rpc-server     Removed      10.6s
  ✓ Network compose_default  Removed      0.5s
○ auliariza@LAPTOP-SJR9B4HG:~/dist_sys-3$
```

Perintah tersebut digunakan untuk menghentikan dan menghapus seluruh container serta network yang sebelumnya dijalankan untuk uji coba RPC. Dari hasil output terlihat bahwa:

- Container rpc-client berhasil dihentikan dan dihapus.
- Container rpc-server juga berhasil dihentikan dan dihapus.
- Network compose default yang menjadi penghubung antar container ikut dihapus.

Tahap ini penting untuk membersihkan environment pengujian, sehingga tidak ada container atau network yang tersisa dan bisa menimbulkan konflik pada pengujian berikutnya.