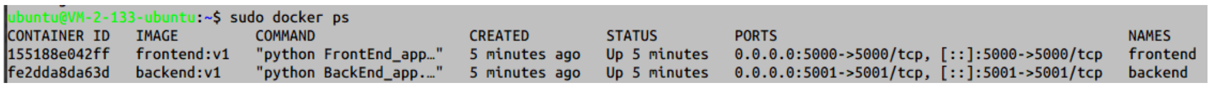
아미루딘 자피르

2022320061

과제 2

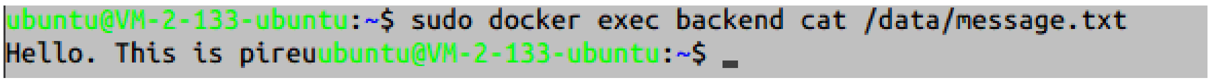
PART B – Screenshots (Report)

1. **Docker ps showing both containers running**



```
ubuntu@VM-2-133-ubuntu:~$ sudo docker ps
CONTAINER ID   IMAGE         COMMAND                  CREATED         STATUS         PORTS                                         NAMES
155188e042ff   frontend:v1   "python FrontEnd_app…"   5 minutes ago   Up 5 minutes   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp   frontend
fe2dda8da63d   backend:v1    "python BackEnd_app.…"   5 minutes ago   Up 5 minutes   0.0.0.0:5001->5001/tcp, [::]:5001->5001/tcp   backend
```

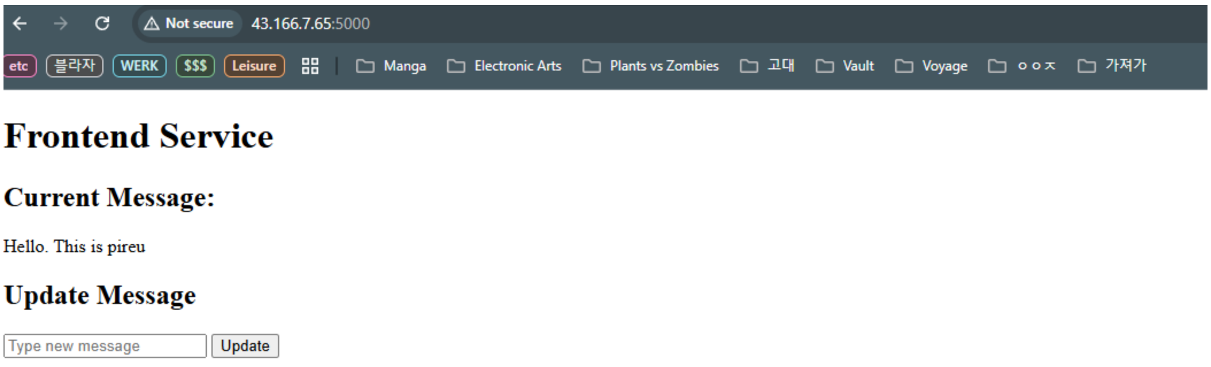
2. **Volume content:**



```
ubuntu@VM-2-133-ubuntu:~$ sudo docker exec backend cat /data/message.txt
Hello. This is pireuubuntu@VM-2-133-ubuntu:~$ _
```

3. **Frontend webpage showing:**

   V1 message:


Not secure 43.166.7.65:5000
etc 블라자 WERK $$$ Leisure Manga Electronic Arts Plants vs Zombies 고대 Vault Voyage ㅇㅇㅈ 가져가
Type new message
Update

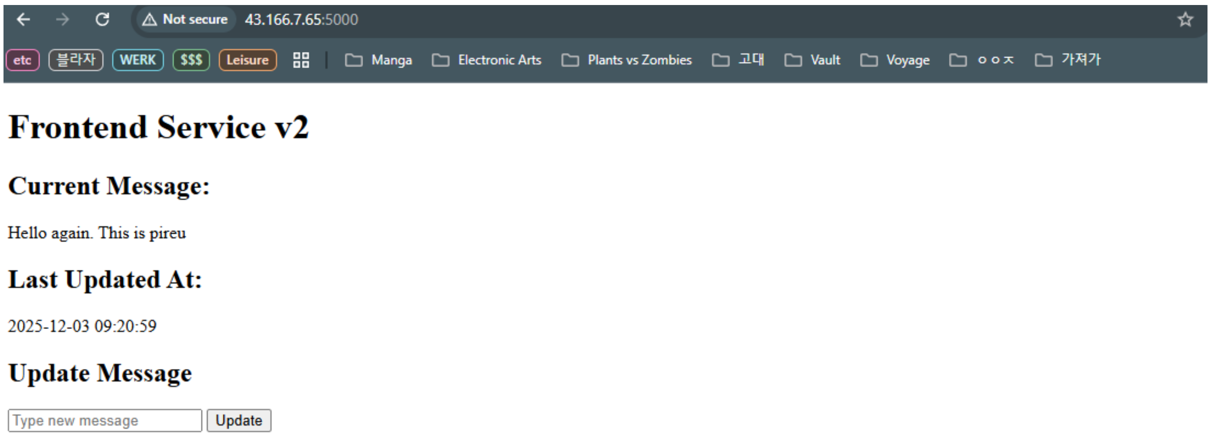
Frontend Service

Current Message:

Hello. This is pireu

Update Message

V2 updated message:


Not secure 43.166.7.65:5000
etc 블라자 WERK $$$ Leisure Manga Electronic Arts Plants vs Zombies 고대 Vault Voyage ㅇㅇㅈ 가져가
Type new message
Update

Frontend Service v2

Current Message:

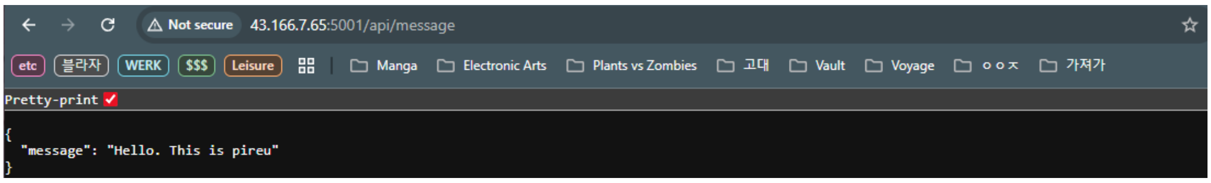
Hello again. This is pireu

Last Updated At:

2025-12-03 09:20:59

Update Message

4. **Browser screenshot hitting backend API directly:**
   a. **GET /api/message (v1 and v2)**

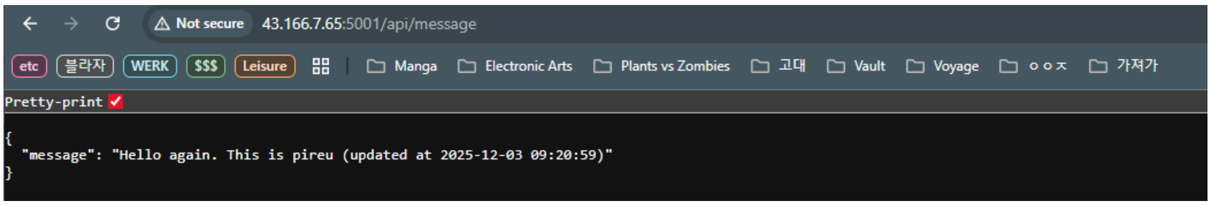


Not secure 43.166.7.65:5001/api/message

etc 블라자 WERK $$$ Leisure Manga Electronic Arts Plants vs Zombies 고대 Vault Voyage ㅇㅇㅈ 가져가

Pretty-print ✓

```
{
  "message": "Hello. This is pireu"
}
```



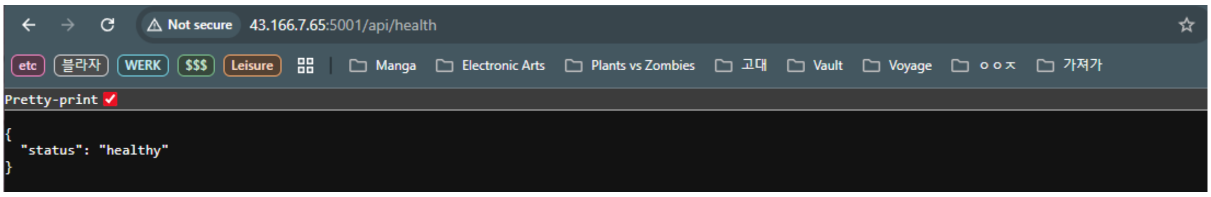
Not secure 43.166.7.65:5001/api/message

etc 블라자 WERK $$$ Leisure Manga Electronic Arts Plants vs Zombies 고대 Vault Voyage ㅇㅇㅈ 가져가

Pretty-print ✓

```
{
  "message": "Hello again. This is pireu (updated at 2025-12-03 09:20:59)"
}
```

   b. **GET /api/health (v2)**



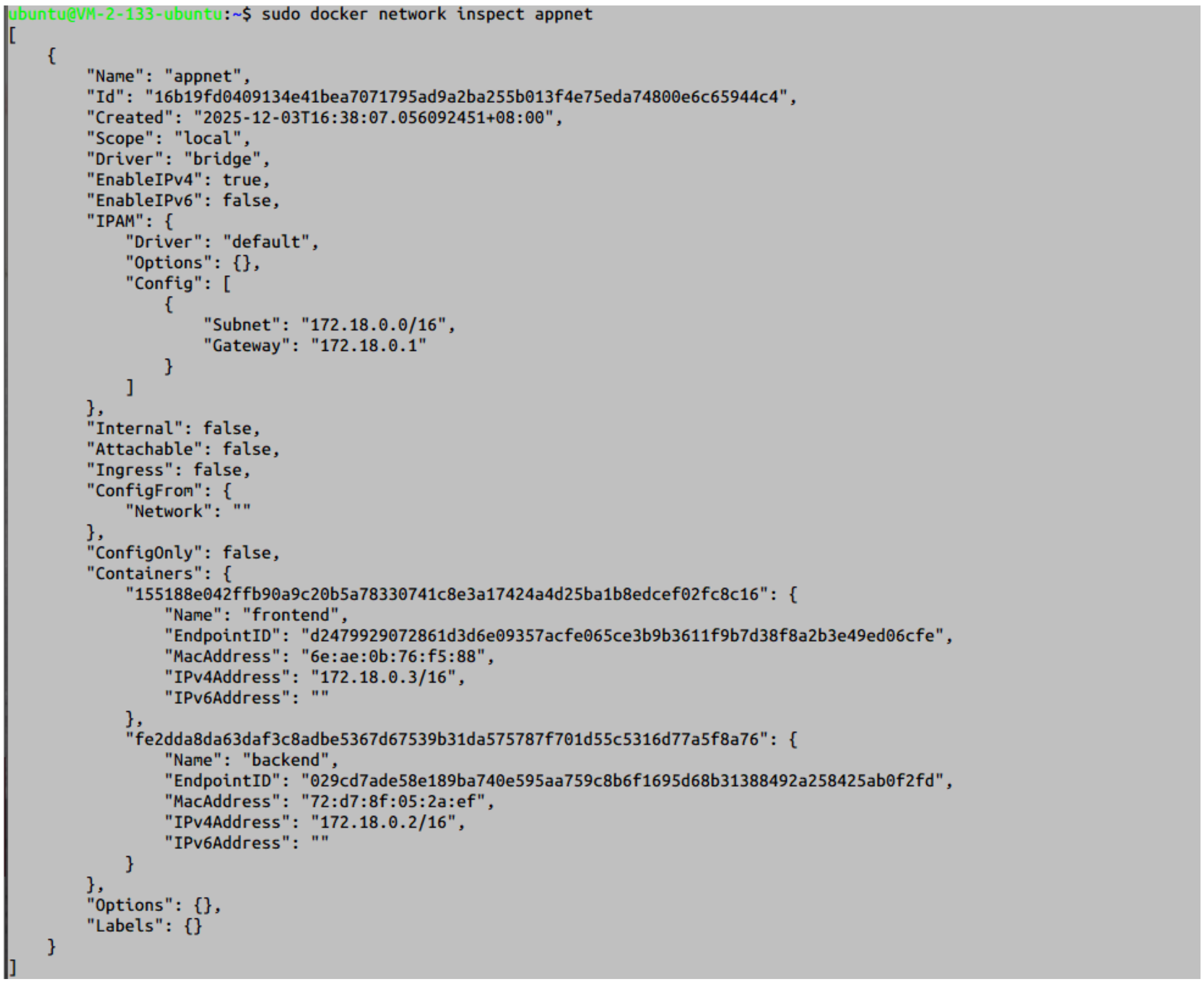
Not secure 43.166.7.65:5001/api/health

etc 블라자 WERK $$$ Leisure Manga Electronic Arts Plants vs Zombies 고대 Vault Voyage ㅇㅇㅈ 가져가

Pretty-print ✓

```
{
  "status": "healthy"
}
```

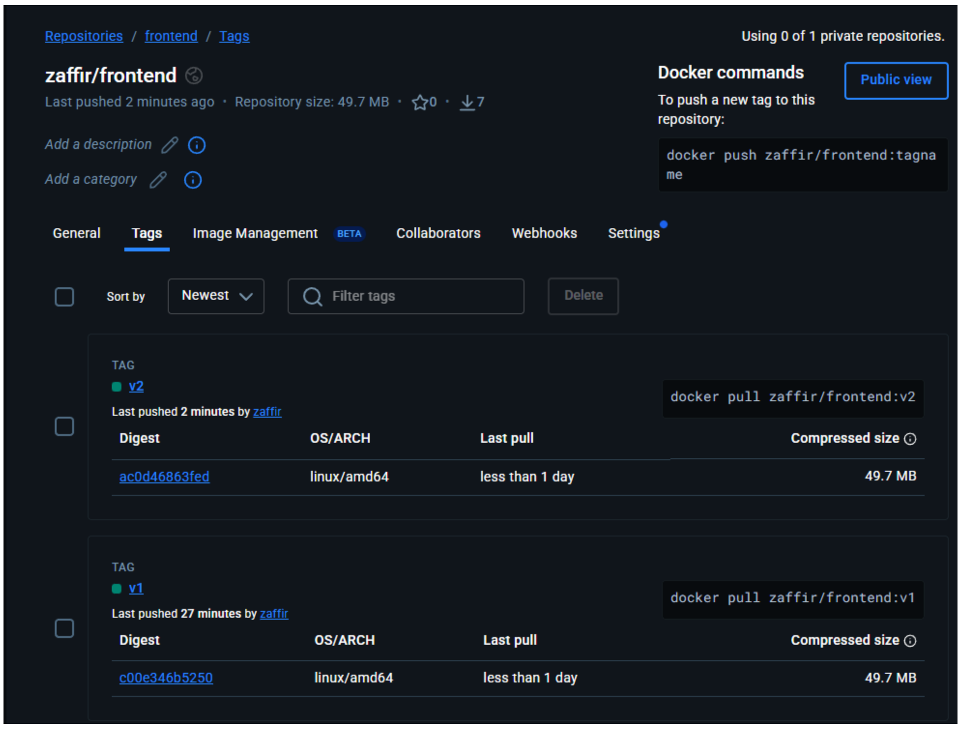
5. Network appnet with the running containers:



```
ubuntu@VM-2-133-ubuntu:~$ sudo docker network inspect appnet
[
    {
        "Name": "appnet",
        "Id": "16b19fd0409134e41bea7071795ad9a2ba255b013f4e75eda74800e6c65944c4",
        "Created": "2025-12-03T16:38:07.056092451+08:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv4": true,
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "155188e042ffb90a9c20b5a78330741c8e3a17424a4d25ba1b8edcef02fc8c16": {
                "Name": "frontend",
                "EndpointID": "d2479929072861d3d6e09357acfe065ce3b9b3611f9b7d38f8a2b3e49ed06cfe",
                "MacAddress": "6e:ae:0b:76:f5:88",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
            },
            "fe2dda8da63daf3c8adbe5367d67539b31da575787f701d55c5316d77a5f8a76": {
                "Name": "backend",
                "EndpointID": "029cd7ade58e189ba740e595aa759c8b6f1695d68b31388492a258425ab0f2fd",
                "MacAddress": "72:d7:8f:05:2a:ef",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            }
        },
        "Options": {},
        "Labels": {}
    }
]
```

6. **Docker Hub pages for:**

Frontend:


Repositories / frontend / Tags

Using 0 of 1 private repositories.

zaffir/frontend

Last pushed 2 minutes ago · Repository size: 49.7 MB · ☆0 · ↓7

Add a description

Add a category

Docker commands

To push a new tag to this repository:

```
docker push zaffir/frontend:tagname
```

General | Tags | Image Management BETA | Collaborators | Webhooks | Settings

Sort by Newest | Filter tags | Delete

TAG
v2
Last pushed 2 minutes by zaffir

```
docker pull zaffir/frontend:v2
```

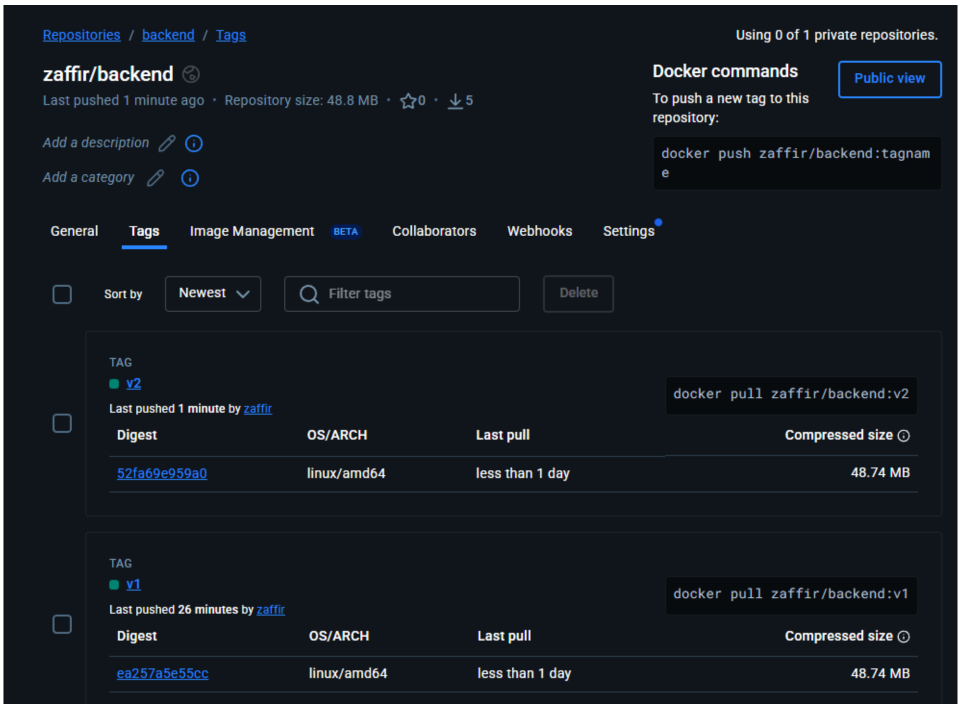
Digest	OS/ARCH	Last pull	Compressed size
ac0d46863fed	linux/amd64	less than 1 day	49.7 MB

TAG
v1
Last pushed 27 minutes by zaffir

```
docker pull zaffir/frontend:v1
```

Digest	OS/ARCH	Last pull	Compressed size
c00e346b5250	linux/amd64	less than 1 day	49.7 MB

Backend:


Repositories / backend / Tags

Using 0 of 1 private repositories.

zaffir/backend

Last pushed 1 minute ago · Repository size: 48.8 MB · ☆0 · ↓5

Add a description

Add a category

Docker commands

To push a new tag to this repository:

```
docker push zaffir/backend:tagname
```

General | Tags | Image Management BETA | Collaborators | Webhooks | Settings

Sort by Newest | Filter tags | Delete

TAG
v2
Last pushed 1 minute by zaffir

```
docker pull zaffir/backend:v2
```

Digest	OS/ARCH	Last pull	Compressed size
52fa69e959a0	linux/amd64	less than 1 day	48.74 MB

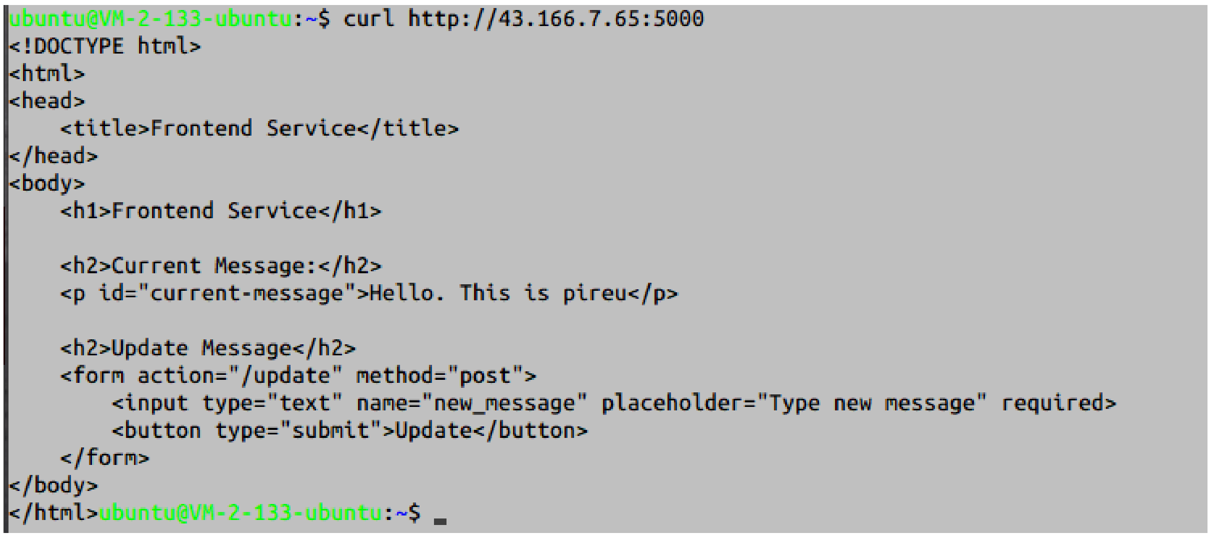
TAG
v1
Last pushed 26 minutes by zaffir

```
docker pull zaffir/backend:v1
```

Digest	OS/ARCH	Last pull	Compressed size
ea257a5e55cc	linux/amd64	less than 1 day	48.74 MB

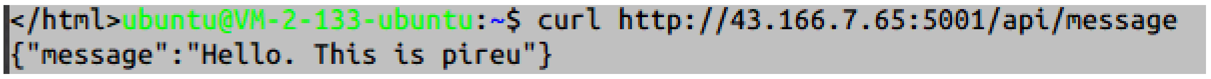
PART C - TEST OUTPUT

curl http://<VM_IP>:5000



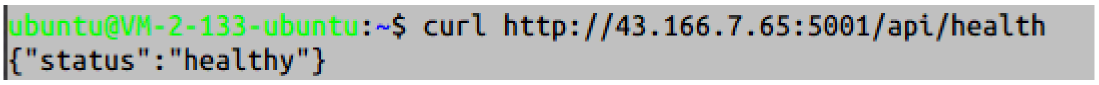
```
ubuntu@VM-2-133-ubuntu:~$ curl http://43.166.7.65:5000
<!DOCTYPE html>
<html>
<head>
    <title>Frontend Service</title>
</head>
<body>
    <h1>Frontend Service</h1>

    <h2>Current Message:</h2>
    <p id="current-message">Hello. This is pireu</p>

    <h2>Update Message</h2>
    <form action="/update" method="post">
        <input type="text" name="new_message" placeholder="Type new message" required>
        <button type="submit">Update</button>
    </form>
</body>
</html>ubuntu@VM-2-133-ubuntu:~$ _
```

curl http://<VM_IP>:5000/api/message



```
</html>ubuntu@VM-2-133-ubuntu:~$ curl http://43.166.7.65:5001/api/message
{"message":"Hello. This is pireu"}
```

curl http://<VM_IP>:5000/api/health



```
ubuntu@VM-2-133-ubuntu:~$ curl http://43.166.7.65:5001/api/health
{"status":"healthy"}
```

PART D – Short Explanation

1. How the frontend communicates with the backend

The frontend communicates with the backend using HTTP requests over the Docker internal network. Specifically, the frontend sends a GET request to http://backend:5001/api/message to retrieve the current message and a POST request to the same endpoint to save a new message. Since both containers are attached to the same Docker network (appnet), the frontend can resolve the backend service using its container name, backend, simplifying inter-service routing. The backend processes the request and returns the data or status as a JSON object.

2. Why Docker needs a shared network

Docker containers are isolated environments by default, meaning they cannot directly reference each other by name or local IP address across standard isolation boundaries. A shared bridge network (like appnet) connects these isolated containers, allowing them to communicate seamlessly. This shared network uses a virtual subnet where Docker assigns an IP and ensures that the service name (backend) resolves to the correct container IP within that network. This enables the microservice architecture, allowing independent components to interact without knowing the host machine's configuration.

3. What the volume is used for

The volume is used to provide persistent storage for the backend service's data, specifically the message stored in /data/message.txt. Without a volume, any data written inside the container (to the writable container layer) would be lost immediately when the container is stopped or deleted. By mounting a Docker-managed volume (like backend_data) to the container's /data directory, the message is stored on the Docker Host machine, ensuring the data survives the container's lifecycle. This is crucial for stateful applications like databases or, in this case, a simple message store.

4. What you changed for v2

For the Backend v2, I implemented two changes: updating the message storage logic to append a timestamp in the format (updated at YYYY-MM-DD HH:MM:SS) and adding a new /api/health endpoint that returns {"status": "healthy"} . For the Frontend v2, I changed the HTML template to have the page title "Frontend Service v2" and added logic in the Python code to parse the timestamp from the backend message. Finally, the frontend displays this extracted timestamp as "Last updated at: <timestamp>" on the webpage.