클라우드 컴퓨팅 실습 과제 2

Objective

Build a small two-service web application using Docker:

- **Frontend (Flask)** → Simple HTML page that shows a message and lets you update it
- **Backend (Flask)** → JSON API that stores & returns a message (persisted through a Docker volume)

You will:

- Write **Dockerfiles**
- Build **custom images** (`v1`)
- Use a **volume** to persist backend data
- Connect the services using a **Docker network**
- Publish images to **Docker Hub**
- Modify the services to create **v2** images (defined below)

We will provide (a ZIP file with):

- Frontend HTML skeleton code (**index.html** - inside the `/templates` folder)
- Frontend Flask skeleton code (**app_front.py**)
- Backend Flask skeleton code (**app_back.py**)

Explanation

1. Application Requirements

Backend Service (Flask)

Runs on port **5001**

Required Endpoints

GET /api/message

Returns JSON:

```
{ "message": "<stored message>" }
```

POST /api/message

Accepts JSON:

```
{ "message": "New message here" }
```

Writes the message to:

```
 /data/message.txt
```

(`data/` must be a Docker volume.)

Frontend Service (Flask)

Runs on port **5000**

Must:

- Render a single HTML page
- Display the current message (fetched from backend)
- Provide a form to update the message
- Send updates to:
  `http://backend:5001/api/message`

Frontend HTML Skeleton (use the `index.html` skeleton code)

```
<!DOCTYPE html>
<html>
<head>
    <title>Frontend Service</title>
</head>
<body>
    <h1>Frontend Service</h1>

    <h2>Current Message:</h2>
    <p id="current-message">{{ current_message }}</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>
```

---

2. Build Docker Images

```
docker build -t frontend:v1 .
docker build -t backend:v1 .
```

---

3. Create a Shared Docker Network

```
docker network create appnet
```

This allows containers to communicate by service **name** (e.g., `backend`).

- Containers must be connected to this network - do not use the default `bridge` network

---

4. Run the Services

Backend (with volume)

```
docker run -d --name backend \
  --network appnet \
```

```
  -v backend_data:/data \
  -p 5001:5001 backend:v1
```

Frontend

```
docker run -d --name frontend \
  --network appnet \
  -p 5000:5000 frontend:v1
```

Visit in browser:

```
http://<YOUR_VM_IP>:5000
```

---

5. Push Images to Docker Hub

```
docker tag frontend:v1 <hub-id>/frontend:v1
docker tag backend:v1 <hub-id>/backend:v1

docker push <hub-id>/frontend:v1
docker push <hub-id>/backend:v1
```

---

6. Create v2 Versions (Required Modifications)

Backend v2 (Mandatory Changes)

1. When updating the message, append a timestamp.
   Final stored format:
   ```
   <message> (updated at YYYY-MM-DD HH:MM:SS)
   ```
2. Add a new endpoint:
   ```
   GET /api/health
   ```
   Must return:
   ```
   { "status": "healthy" }
   ```

Frontend v2 (Mandatory Changes)

1. Change page title to:

```
Frontend Service v2
```

2. Add new line on page:

```
Last updated at: <timestamp>
```

The timestamp must come from the message returned by the backend.

Building and Pushing v2

You may use docker commit or rebuild normally.

```
docker commit frontend <hub-id>/frontend:v2
docker commit backend <hub-id>/backend:v2

docker push <hub-id>/frontend:v2
docker push <hub-id>/backend:v2
```

Submission Checklist (제출물)

> **Requirements**
>
> Submit `Part A` in **one ZIP (.zip) file**
> Submit `Part B`, `Part C`, `Part D` in **one report (.pdf)** file
>
> Any other file formats will result in **deduction to score!**

Part A — Source Code (submit files)

- `frontend/app.py`
- `frontend/templates/index.html`
- `frontend/Dockerfile`
- `backend/app.py`
- `backend/Dockerfile`

Part B — Screenshots (Report)

1. `docker ps` showing both containers running (frontend and backend - v1 or v2 either is fine)
2. **Volume content**:

```
docker exec backend cat /data/message.txt
```

3. **Frontend webpage** showing:
   - v1 message
   - v2 updated message
4. **Browser screenshot** hitting backend API directly:
   - `GET /api/message` (v1 and v2)
   - `GET /api/health` (v2)
5. **Network** appnet with the running containers:

```
docker network inspect appnet
```

6. **Docker Hub pages** for:

```
frontend:v1
frontend:v2
```

```
backend:v1
backend:v2
```

Part C — Test Output (Report)

- Copy/paste or screenshot results from:

```
curl http://<VM_IP>:5000
curl http://<VM_IP>:5001/api/message
curl http://<VM_IP>:5001/api/health
```

Only provide for v2 (no need to include v1)

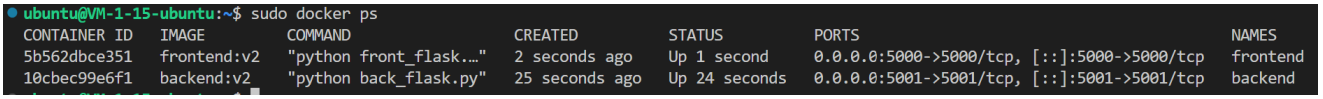
Part D — Short Explanation (~4 sentences - Report)

Explain:

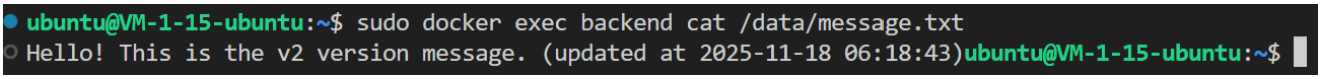
1. How the frontend communicates with the backend
2. Why Docker needs a shared network
3. What the volume is used for
4. What you changed for v2

Example (Screenshots)

1. docker ps



2. docker volume



3. Frontend webpage

Frontend Service

Current Message:

Hello! This is the v1 version message.

Update Message

Type new message Update

Frontend Service v2

Current Message:

Hello! This is the v2 version message.

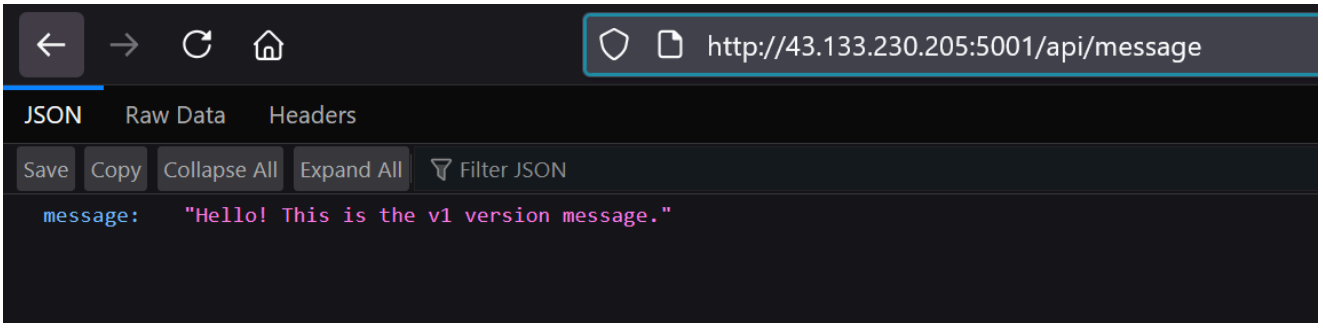
Last Updated At:

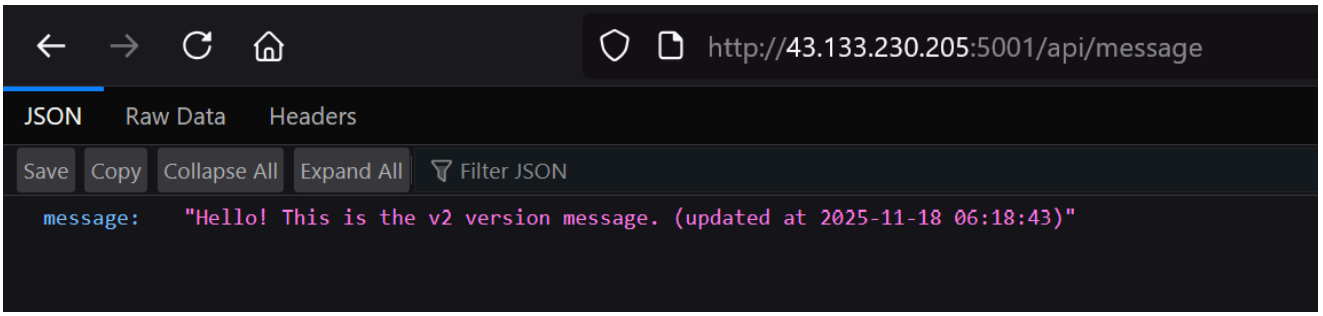
2025-11-18 06:18:43

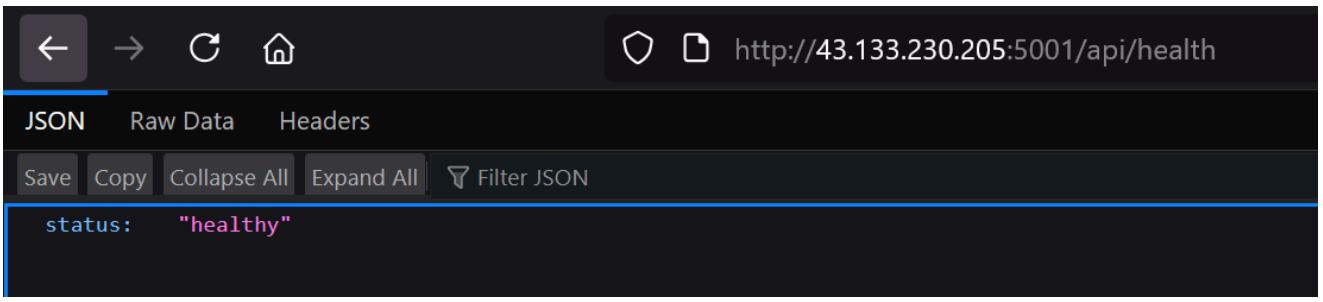
Update Message

Type new message Update

4. Backend API (webpage)



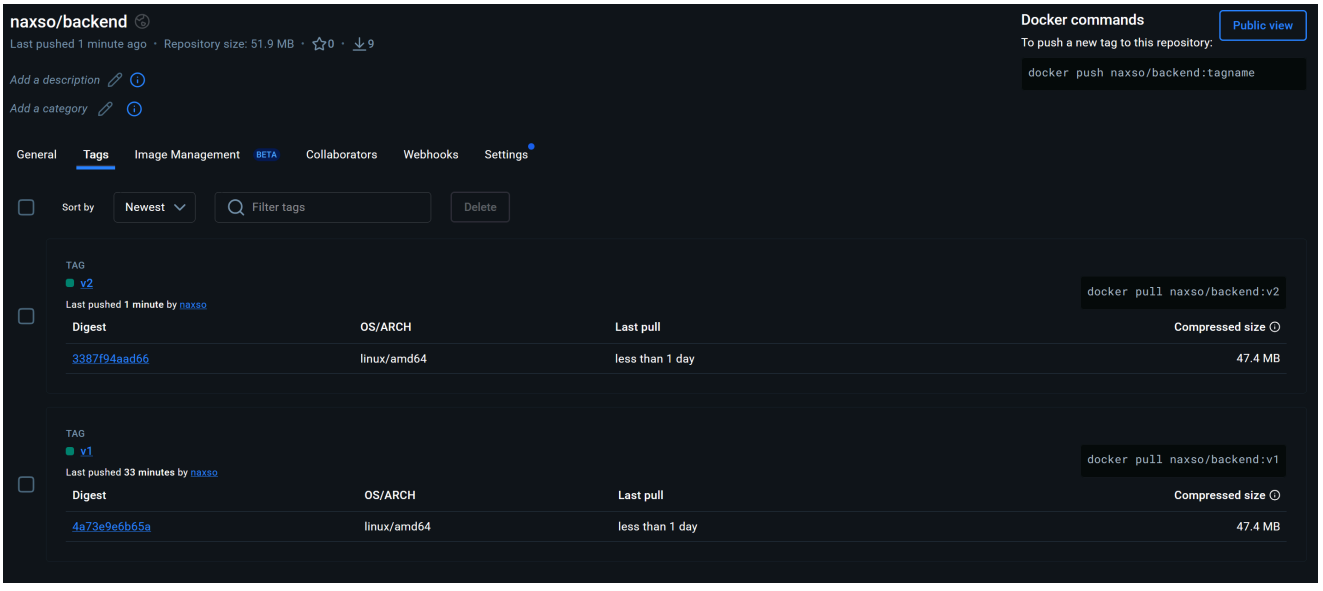


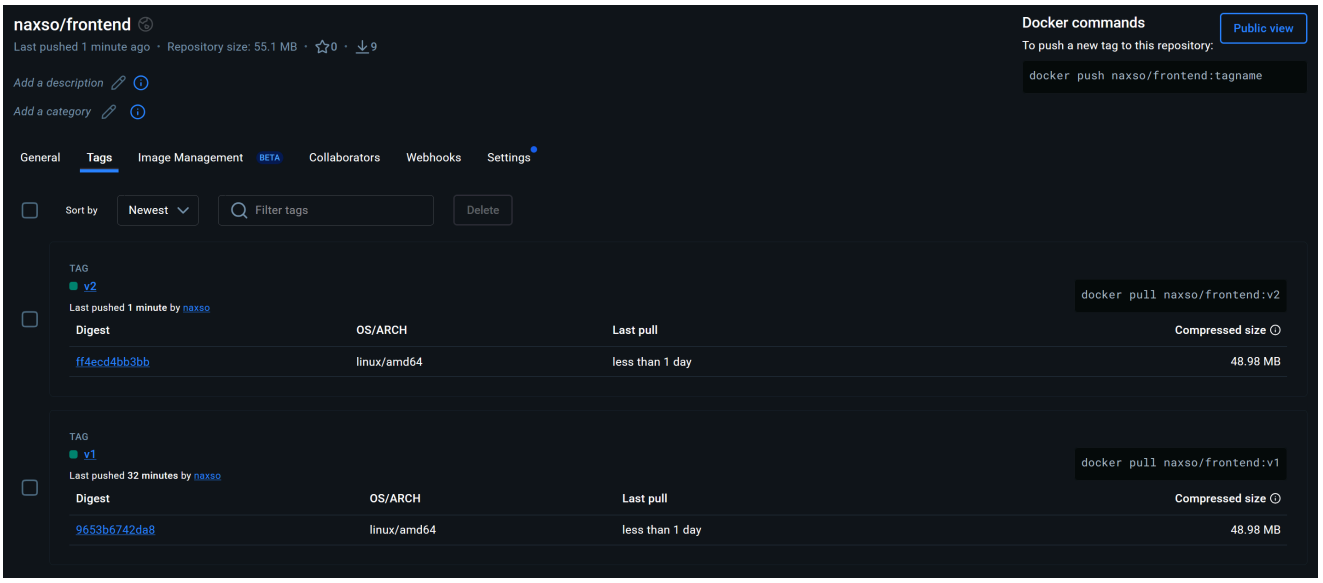


5. docker network

```
"Name": "appnet",
"Id": "9d150b7fe7ee0f6291e1d6aafe4a9afe3e61a9fdab8a827f9cafccdf47382ac2",
"Created": "2025-11-18T12:38:51.416874904+08:00",
"Scope": "local",
"Driver": "bridge",
"EnableIPv4": true,
"EnableIPv6": false,
"IPAM": {
    "Driver": "default",
    "Options": {},
    "Config": [
        {
            "Subnet": "172.18.0.0/16",
            "IPRange": "",
            "Gateway": "172.18.0.1"
        }
    ]
},
"Internal": false,
"Attachable": false,
"Ingress": false,
"ConfigFrom": {
    "Network": ""
},
"ConfigOnly": false,
"Options": {},
"Labels": {},
"Containers": {
    "10cbec99e6f1ad82b2682e7e794e2838a1146c897d54564d2acdbe1dbfd4785d": {
        "Name": "backend",
        "EndpointID": "4f1b870e943dbc0c9c89676e9ba031c0a71706961efde2b4cd9ee1b67ffc4e33",
        "MacAddress": "5a:f8:a0:36:3b:4e",
        "IPv4Address": "172.18.0.2/16",
        "IPv6Address": ""
    },
    "5b562dbce351b39d0ab34d66b920c5c0155a9c57ecefa037e50e27b83b4e314e": {
        "Name": "frontend",
        "EndpointID": "cd2002ef057047fe9fd795c2e0a47e1139c4e37e84e8600d57ebfe7cb1a91e6f",
        "MacAddress": "42:1c:06:73:db:ed",
        "IPv4Address": "172.18.0.3/16",
        "IPv6Address": ""
```


6. Docker Hub repositories





7. Test output (`curl` command - `curl port 5000` not included in example)

```
ubuntu@VM-1-15-ubuntu:~$ curl http://43.133.230.205:5001/api/message
{"message":"Hello! This is the v2 version message. (updated at 2025-11-18 06:18:43)"}
ubuntu@VM-1-15-ubuntu:~$ curl http://43.133.230.205:5001/api/health
{"status":"healthy"}
```

📅 Deadline

Date: 2025.12.03(수) - 2 weeks

Late submissions will not be accepted!

Submit files in the required format! (source code - ZIP file / report - PDF)