Assignment 2

Cloud Computing

2020320052 김도윤

Part B – Screenshots

1.docker ps showing both containers running

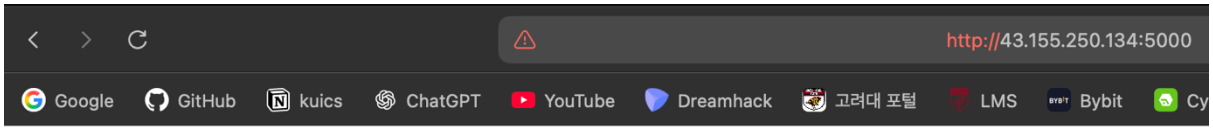
```
ubuntu@VM-2-60-ubuntu:~$ sudo docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS                                                NAMES
da4c26d99155   qefoi/frontend:v1  "python app_front.py"    37 seconds ago   Up 36 seconds   0.0.0.0:5000->5000/tcp, [::]:5000->5000/tcp          frontend
289e9e637067   qefoi/backend:v1   "python app_back.py"     47 seconds ago   Up 47 seconds   0.0.0.0:5001->5001/tcp, [::]:5001->5001/tcp          backend
```

2. Volume content

```
ubuntu@VM-2-60-ubuntu:~$ sudo docker exec backend cat /data/message.txt
hello, this is v1 versionubuntu@VM-2-60-ubuntu:~$
```

3. Frontend webpage showing:

- v1 message



Frontend Service

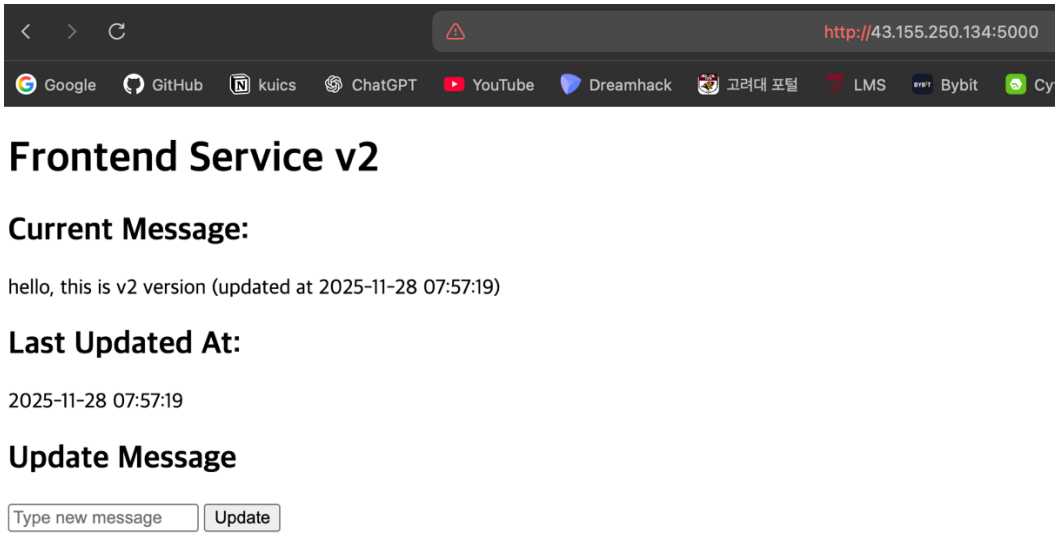
Current Message:

hello, this is v1 version

Update Message

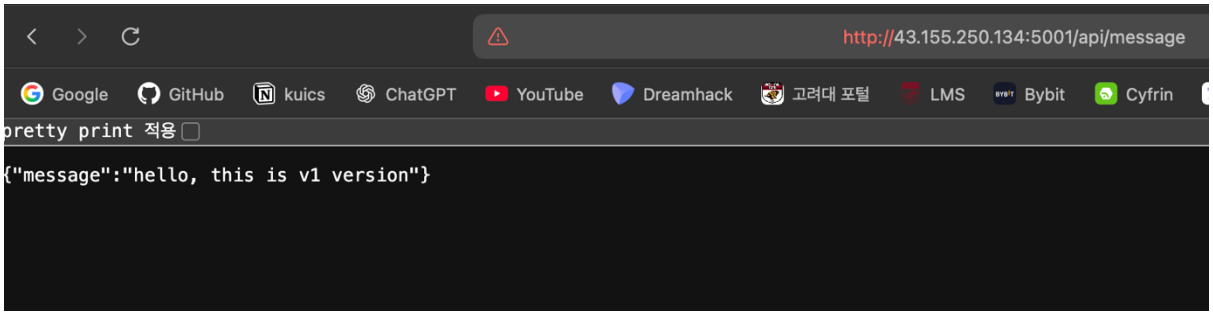
Type new message | Update

- v2 updated message

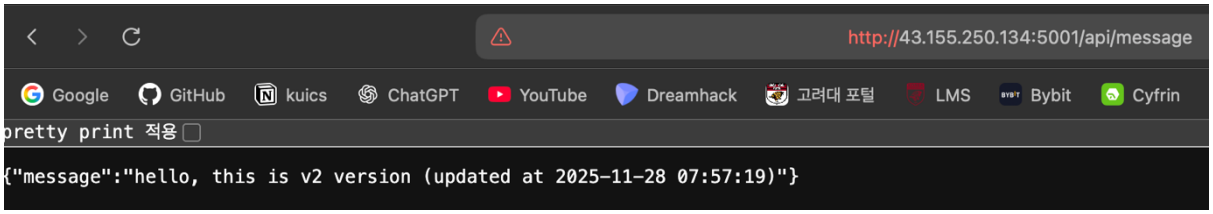


4. Browser screenshot hitting backend API directly:

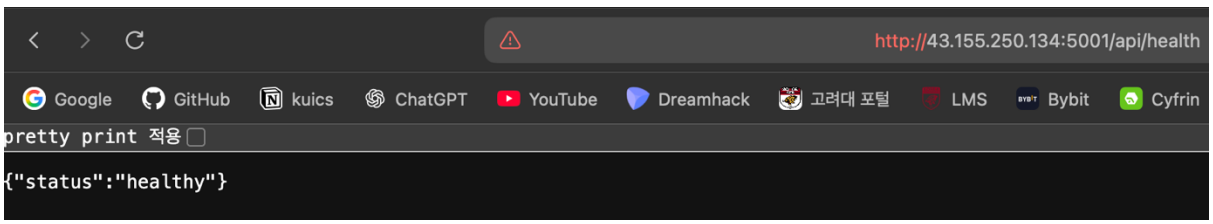
- GET /api/message v1



- GET /api/message v2



- GET /api/health v2

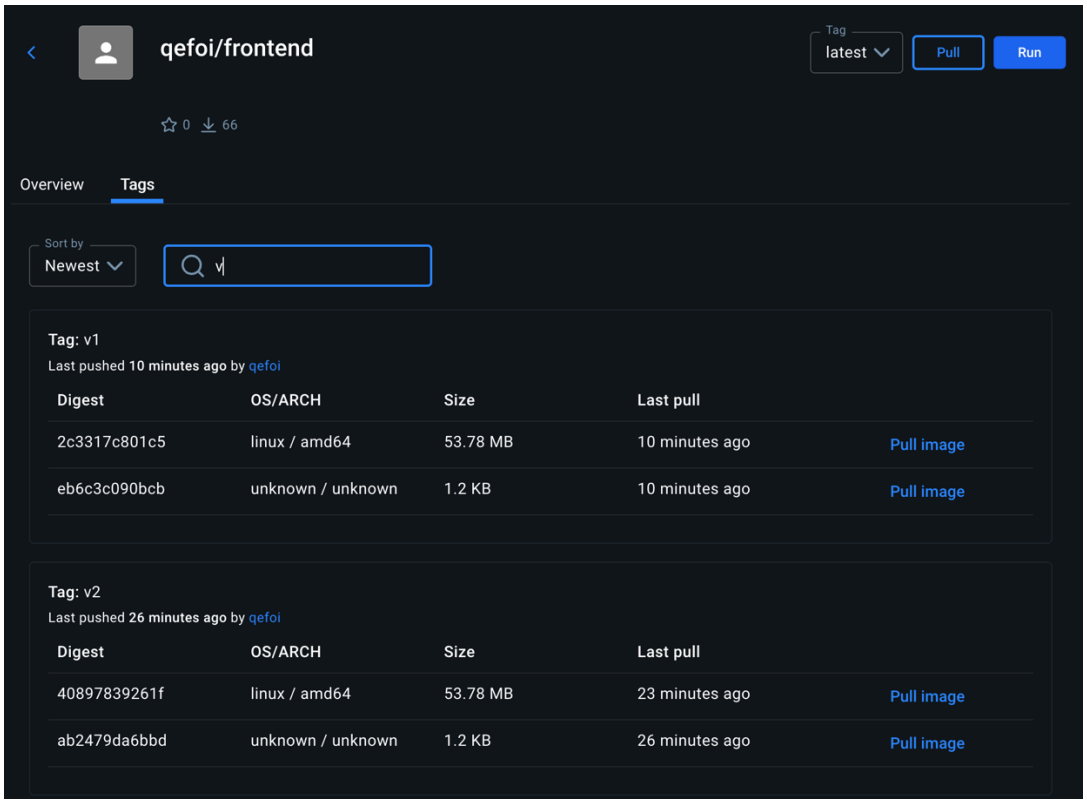


5. Network appnet with the running containers

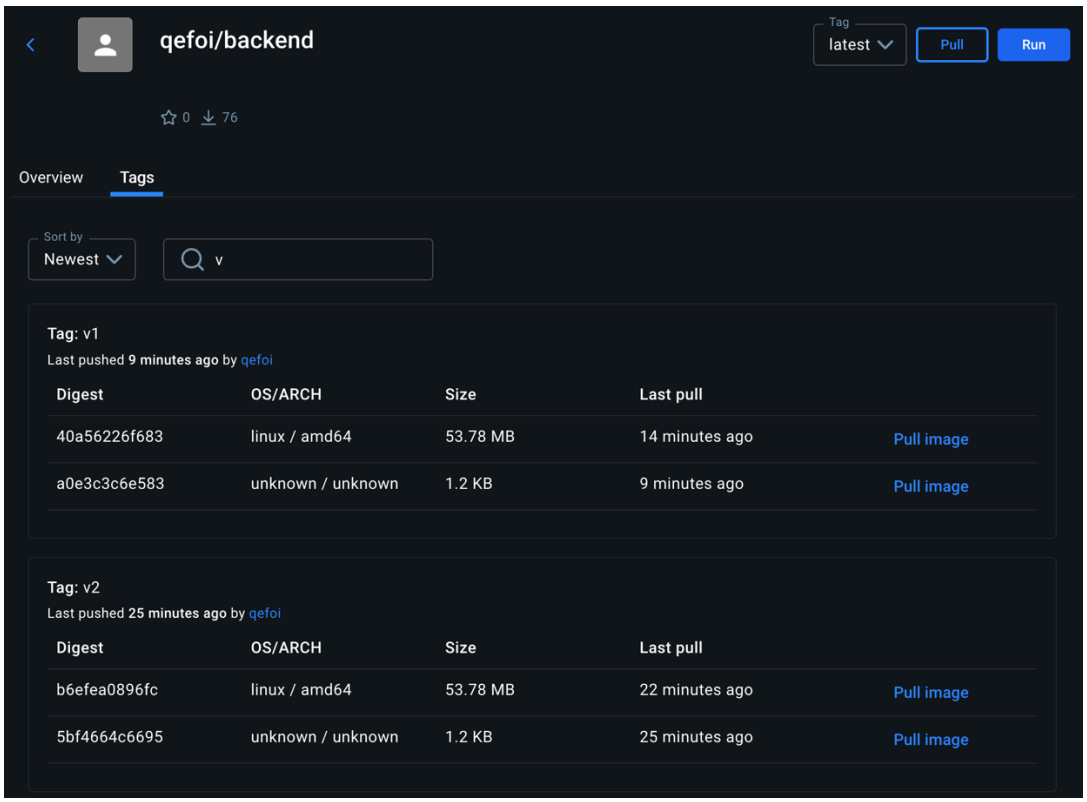
```
ubuntu@VM-2-60-ubuntu:~$ sudo docker network inspect appnet
[
    {
        "Name": "appnet",
        "Id": "85c52adc799199b34e4b183428ad445e539ced1de0d09a9397849059ad560e30",
        "Created": "2025-11-28T15:18:45.329551978+08:00",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv4": 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": {
            "289e9e637067575fa6d8c7cd892acb5037ec19c5683db675fa45513eb8545c08": {
                "Name": "backend",
                "EndpointID": "0450811d81e5f88c09068061b7a823ff3a1863d51d4e7891d416dc6a6032f479",
                "MacAddress": "06:fa:4c:7f:6c:53",
                "IPv4Address": "172.18.0.2/16",
                "IPv6Address": ""
            },
            "da4c26d99155aa8dcdc76ed6cdb259baf2400356d91eb4557a9872dacd741806": {
                "Name": "frontend",
                "EndpointID": "892ac8b67fb59fc649edbe5cf0101476038802b1a83baca6e5de7a52fc5fe3ae",
                "MacAddress": "2e:61:e4:a1:03:75",
                "IPv4Address": "172.18.0.3/16",
                "IPv6Address": ""
            }
        },
        "Status": {
            "IPAM": {
                "Subnets": {
                    "172.18.0.0/16": {
                        "IPsInUse": 5,
                        "DynamicIPsAvailable": 65531
                    }
                }
            }
        }
    }
]
```

6. Docker Hub pages

-frontend



- backend



Part C - Test Output

```
ubuntu@VM-2-60-ubuntu:~$ curl http://43.155.250.134:5000
<!DOCTYPE html>
<html>
<head>
    <title>Frontend Service</title>
</head>
<body>
    <h1>Frontend Service v2</h1>

    <h2>Current Message:</h2>
    <p id="current-message">hello, this is v2 version (updated at 2025-11-28 07:57:19)</p>

    <h2>Last Updated At:</h2>
    <p id="update_time"> 2025-11-28 07:57:19</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-2-60-ubuntu:~$ curl http://43.155.250.134:5001/api/message
{"message":"hello, this is v2 version (updated at 2025-11-28 07:57:19)"}
ubuntu@VM-2-60-ubuntu:~$ curl http://43.155.250.134:5001/api/health
{"status":"healthy"}
```

Part D - Short Explanation

1. How the frontend communicates with the backend

frontend는 HTTP 요청을 통해 백엔드의 API 엔드포인트에 접근한다. 도커 환경에서는 frontend와 backend가 서로 다른 컨테이너로 실행되지만, 둘이 같이 도커 네트워크 내에 있다면 백엔드 컨테너의 이름을 호스트명으로 사용하여 호출 할 수 있다. 그래서 IP 주소를 몰라도 이름으로 통신이 가능하다.

2. Why Docker needs a shared network

frontend가 backend를 컨테이너 이름으로 호출하려면 두 컨테이너가 같은 도커 네트워크에 존재해야 한다. appnet은 컨테이너 간 이름 기반 통신을 지원한다. 만약 공유 네트워크를 사용하지 않는다면 frontend는 backend의 컨테이너 주소를 해석할 수 없어 api 호출이 실패하게 된다. 따라서 서비스들이 정상적으로 통신하기 위해 appnet 같은 shared network가 필요로 한다.

3. What the volume is used for

컨테이너는 기본적으로 일시적인 파일 시스템을 사용하기 때문에 컨테이너가 재시작되거나 삭제되면 내부에 저장된 데이터가 사라지게 된다. 도커 volume은 이런 문제를 해결해준다. 도커 volume은 컨테이너와 무관하게 데이터를 유지하고, 같은 volume을 여러 컨테이너에 마운트 하여 데이터 공유도 가능하게 한다.

4. What you changed for v2

v2에서는 backend에서 메시지를 저장할 때 현재 시간을 포함하도록 수정하고, 새로운 /api/health 엔드포인트를 추가하여 서비스 상태를 json으로 반환하게 만들었다. 또한 html을 수정하여 페이지 제목을 Frontend Service v2로 변경하고 timestamp를 파싱하여 frontend에 표시하도록 기능을 추가했다.