Lab 3 - Мікросервіси з використанням Hazelcast Distributed Map

Завдання базується на першому завданні і є його розвитком.
До ***logging-service*** необхідно додати:

- у якості сховища повідомлень **Hazelcast Distributed Map**

```
try:
    client = hazelcast.HazelcastClient(
        cluster_name="dev",
        cluster_members=hazelcast_members,
        reconnect_mode="ASYNC"
    )
    msg_map = client.get_map("messages").blocking()
    logger.info(f"[{instance}] Connected to Hazelcast cluster")
```

Збереження повідомлень у Hazelcast Map у методі handle_logging для POST-запиту:

```
try:
    msg_map.put(id, f"{txt} (processed by {instance})")
    logger.info(f"[{instance}] Logged message: {txt} with ID
    return jsonify({'message': 'Message logged', 'id': id}),
```

Отримання повідомлень із Hazelcast Map для GET-запиту:

```
elif request.method == 'GET':
    try:
        all_messages = list(msg_map.values())
        logger.info(f"[{instance}] Retrieved {len(all_messages)} message
        return jsonify({
            'instance': instance,
            'messages': all_messages
        }), 200
```

- можливість запускати одночасно декілька копій ***logging-service***

```
logging1:
    build:
      context: ./logging-service
      dockerfile: ./Dockerfile
    container_name: logging1
    environment:
      - PORT=50051
      - HZ_ADDRESS=hazelcast1:5701,hazelcast2:5701,hazelcast3:5701
    command: python logging-service.py
    depends_on:
      hazelcast1:
        condition: service_healthy
      hazelcast2:
```

```
      condition: service_healthy
    hazelcast3:
      condition: service_healthy
  ports:
    - "50051:50051"
  networks:
    - microservices-net

logging2:
  build:
    context: ./logging-service
    dockerfile: ./Dockerfile
  container_name: logging2
  environment:
    - PORT=50052
    - HZ_ADDRESS=hazelcast1:5701,hazelcast2:5701,hazelcast3:5701
  command: python logging-service.py
  depends_on:
    hazelcast1:
      condition: service_healthy
    hazelcast2:
      condition: service_healthy
    hazelcast3:
      condition: service_healthy
  ports:
    - "50052:50052"
  networks:
    - microservices-net

logging3:
  build:
    context: ./logging-service
    dockerfile: ./Dockerfile
  container_name: logging3
  environment:
    - PORT=50053
    - HZ_ADDRESS=hazelcast1:5701,hazelcast2:5701,hazelcast3:5701
  command: python logging-service.py
  depends_on:
    hazelcast1:
      condition: service_healthy
    hazelcast2:
      condition: service_healthy
    hazelcast3:
      condition: service_healthy
  ports:
    - "50053:50053"
  networks:
    - microservices-net
```

- ***facade-service*** випадковим чином обирає до якої копії ***logging-service*** звертатись для запису та читання повідомлень

```python
def send_with_retry(services, data):
    tried = set()
    while len(tried) < len(services):
        service = random.choice(services)
        if service in tried:
            continue
```

Додатковий функціонал системи (+5 балів)

У зв'язку з тим, що тепер може бути декілька запущених екземплярів ***logging-service***, то ***facade-service*** має знати про їх IP-адреси, для доступу до них. Перелік IP-адрес, може міститись у коді самого ***facade-service***, чи передаватись у нього при старті, проте такий підхід є не гнучким, у разі динамічного призначення чм зміни цих адрес.
Тому, пропонується винести інформацію про IP-адреси інших мікросервісів у окремий реєстр, за який буде відповідати ***config-server*** (пізніше він буде замінений на *Service registry and discovery*).
Тепер, перед зверненням до ***logging-service*** та ***messages-service, facade-service*** має робить запит до ***config-server***, де по імені сервіса йому повертається перелік всіх IP-адрес екземплярів даного сервіса.
Перелік IP-адрес мікросервісів на ***config-server***, може братись з конфігураційного файлу чи передаватись з командного рядка під час його запуску.

Винесення інформації про IP-адреси в config-server
У файлі config-server.py:
Адреси сервісів отримуються із змінних середовища:

```python
# Отримуємо адреси сервісів із змінних середовища
logging_services = os.environ.get("LOGGING_SERVICES", "logging1:50051,logging2:50052,logging3:50053").split(',')
messages_services = os.environ.get("MESSAGES_SERVICES", "messages:8882").split(',')
```

Формування URL для сервісів:

```python
# Формуємо повні URL для logging-service
logging_services = [f"http://{addr}/logging" for addr in logging_services]
messages_services = [f"http://{addr}/messages" for addr in messages_services]
```

Ендпоінт для повернення адрес за ім'ям сервісу:

```python
@app.route('/services/<service_name>', methods=['GET'])
def get_service_addresses(service_name):
    if service_name == "logging":
        return jsonify({"addresses": logging_services})
    elif service_name == "messages":
        return jsonify({"addresses": messages_services})
    else:
        return jsonify({"error": "Service not found"}), 404
```

Запит facade-service до config-server для отримання адрес

Функція get_service_addresses робить запит до config-server:

```python
def get_service_addresses(service_name):
    logger.info(f"Fetching addresses for {service_name} from {config_server}")
    try:
        response = requests.get(f"{config_server}/services/{service_name}", timeout=3)
        if response.status_code == 200:
            addresses = response.json().get("addresses", [])
            logger.info(f"Received addresses for {service_name}: {addresses}")
            return addresses
```

Виклик цієї функції перед обробкою POST-запиту для logging-service:

```python
logging_services = get_service_addresses("logging")
```

І для GET-запиту, де отримуються адреси як logging-service, так і messages-service:

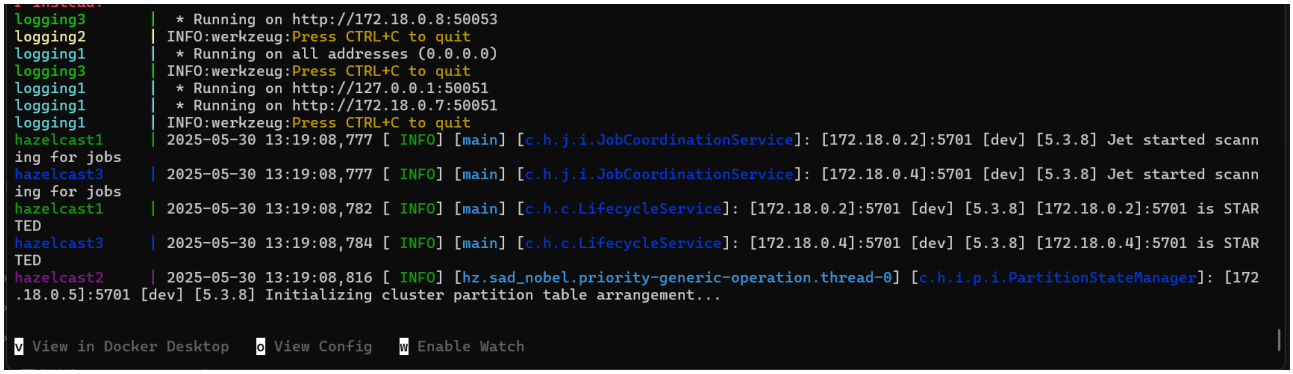
```python
logging_services = get_service_addresses("logging")
messages_services = get_service_addresses("messages")
```

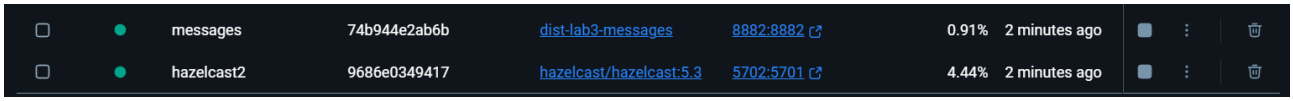
Передача адрес через змінні середовища в config-server
У файлі docker-compose.yml для сервісу config-server:

```yaml
config-server:
  build:
    context: ./config-server
    dockerfile: ./Dockerfile
  container_name: config-server
  environment:
    - LOGGING_SERVICES=logging1:50051,logging2:50052,logging3:50053
    - MESSAGES_SERVICES=messages:8882
```

ЗАВДАННЯ

- **Запустити три екземпляра *logging-service* (локально їх можна запустити на різних портах), відповідно мають запуститись також три екземпляра Hazelcast**





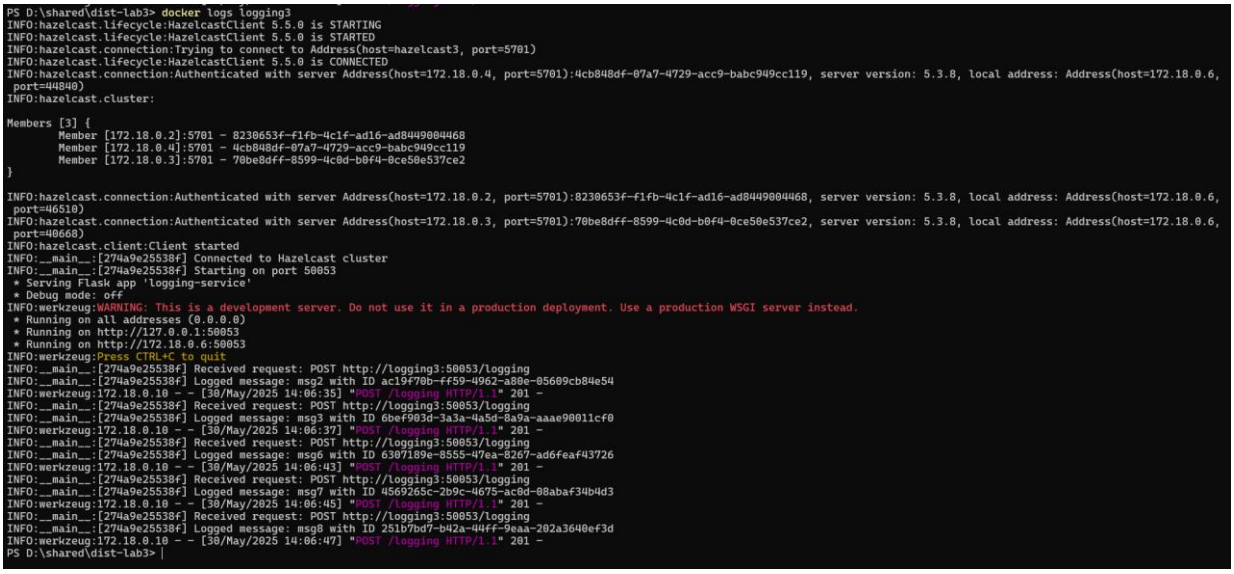
- **Через HTTP POST записати 10 повідомлень *msg1-msg10* через *facade-service***

```
D:\shared\dist-lab3>curl -X POST http://localhost:8880/messages -d "txt=msg1"
{"id":"c4e7ef88-6ddf-4767-991c-0c7933033c8a","txt":"msg1"}

D:\shared\dist-lab3>curl -X POST http://localhost:8880/messages -d "txt=msg2"
{"id":"ac19f70b-ff59-4962-a80e-05609cb84e54","txt":"msg2"}

D:\shared\dist-lab3>curl -X POST http://localhost:8880/messages -d "txt=msg3"
{"id":"6bef903d-3a3a-4a5d-8a9a-aaae90011cf0","txt":"msg3"}

D:\shared\dist-lab3>curl -X POST http://localhost:8880/messages -d "txt=msg4"
{"id":"960f973e-9f18-4d6b-a690-bf7a56b4e66e","txt":"msg4"}

D:\shared\dist-lab3>curl -X POST http://localhost:8880/messages -d "txt=msg5"
{"id":"8a29863e-50e1-48a9-9b71-1d0397e64ce3","txt":"msg5"}

D:\shared\dist-lab3>curl -X POST http://localhost:8880/messages -d "txt=msg6"
{"id":"6307189e-8555-47ea-8267-ad6feaf43726","txt":"msg6"}

D:\shared\dist-lab3>curl -X POST http://localhost:8880/messages -d "txt=msg7"
{"id":"4569265c-2b9c-4675-ac0d-08abaf34b4d3","txt":"msg7"}

D:\shared\dist-lab3>curl -X POST http://localhost:8880/messages -d "txt=msg8"
{"id":"251b7bd7-b42a-44ff-9eaa-202a3640ef3d","txt":"msg8"}

D:\shared\dist-lab3>curl -X POST http://localhost:8880/messages -d "txt=msg9"
{"id":"67077f48-9f1f-45d4-b5e6-e9b54552e53d","txt":"msg9"}

D:\shared\dist-lab3>curl -X POST http://localhost:8880/messages -d "txt=msg10"
{"id":"02f9373c-4c29-48b9-a710-5f5fec03839c","txt":"msg10"}

D:\shared\dist-lab3>
```

- **Показати які повідомлення отримав кожен з екземплярів *logging-service* (це має бути видно у логах сервісу)**

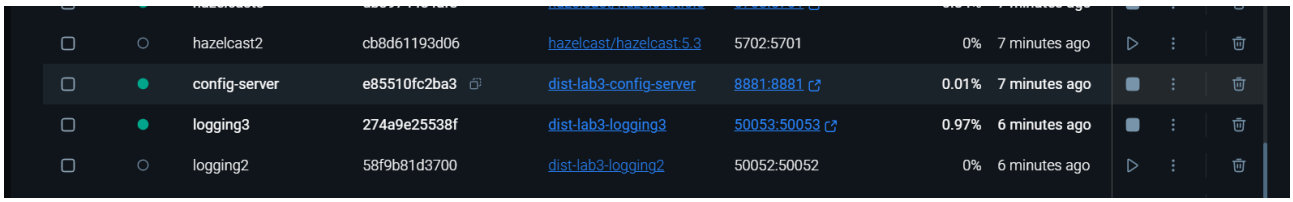
```
PS D:\shared\dist-lab3> docker logs logging1
INFO:hazelcast.lifecycle:HazelcastClient 5.5.0 is STARTING
INFO:hazelcast.lifecycle:HazelcastClient 5.5.0 is STARTED
INFO:hazelcast.connection:Trying to connect to Address(host=hazelcast3, port=5701)
INFO:hazelcast.lifecycle:HazelcastClient 5.5.0 is CONNECTED
INFO:hazelcast.connection:Authenticated with server Address(host=172.18.0.4, port=5701):4cb848df-07a7-4729-acc9-babc949cc119, server version: 5.3.8, local address: Address(host=172.18.0.8
 port=40362)
INFO:hazelcast.cluster:

Members [3] {
        Member [172.18.0.2]:5701 - 8230653f-f1fb-4c1f-ad16-ad8449004468
        Member [172.18.0.4]:5701 - 4cb848df-07a7-4729-acc9-babc949cc119
        Member [172.18.0.3]:5701 - 70be8dff-8599-4c0d-b0f4-0ce50e537ce2
}

INFO:hazelcast.connection:Authenticated with server Address(host=172.18.0.2, port=5701):8230653f-f1fb-4c1f-ad16-ad8449004468, server version: 5.3.8, local address: Address(host=172.18.0.8
 port=59170)
INFO:hazelcast.connection:Authenticated with server Address(host=172.18.0.3, port=5701):70be8dff-8599-4c0d-b0f4-0ce50e537ce2, server version: 5.3.8, local address: Address(host=172.18.0.8
 port=35264)
INFO:hazelcast.client:Client started
INFO:__main__:[5aa3d4c576c6] Connected to Hazelcast cluster
INFO:__main__:[5aa3d4c576c6] Starting on port 50051
 * Serving Flask app 'logging-service'
 * Debug mode: off
INFO:werkzeug: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://127.0.0.1:50051
 * Running on http://172.18.0.8:50051
INFO:werkzeug:Press CTRL+C to quit
INFO:__main__:[5aa3d4c576c6] Received request: POST http://logging1:50051/logging
INFO:__main__:[5aa3d4c576c6] Logged message: msg9 with ID 67077f48-9f1f-45d4-b5e6-e9b54552e53d
INFO:werkzeug:172.18.0.10 - - [30/May/2025 14:06:49] "POST /logging HTTP/1.1" 201 -
INFO:__main__:[5aa3d4c576c6] Received request: POST http://logging1:50051/logging
INFO:__main__:[5aa3d4c576c6] Logged message: msg10 with ID 02f9373c-4c29-48b9-a710-5f5fec03839c
INFO:werkzeug:172.18.0.10 - - [30/May/2025 14:06:51] "POST /logging HTTP/1.1" 201 -
```

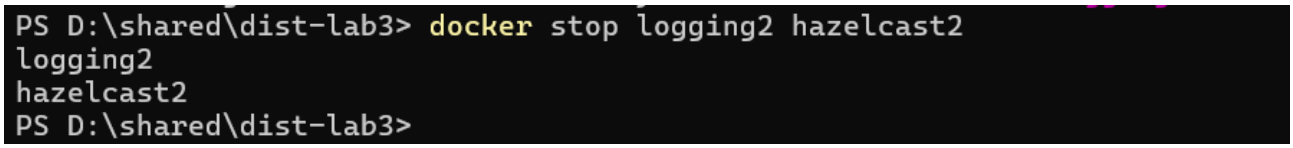
```
PS D:\shared\dist-lab3> docker logs logging2
INFO:hazelcast.lifecycle:HazelcastClient 5.5.0 is STARTING
INFO:hazelcast.lifecycle:HazelcastClient 5.5.0 is STARTED
INFO:hazelcast.connection:Trying to connect to Address(host=hazelcast3, port=5701)
INFO:hazelcast.lifecycle:HazelcastClient 5.5.0 is CONNECTED
INFO:hazelcast.connection:Authenticated with server Address(host=172.18.0.4, port=5701):4cb848df-07a7-4729-acc9-babc949cc119, server version: 5.3.8, local address: Address(host=172.18.0.7,
 port=55598)
INFO:hazelcast.cluster:

Members [3] {
        Member [172.18.0.2]:5701 - 8230653f-f1fb-4c1f-ad16-ad8449004468
        Member [172.18.0.4]:5701 - 4cb848df-07a7-4729-acc9-babc949cc119
        Member [172.18.0.3]:5701 - 70be8dff-8599-4c0d-b0f4-0ce50e537ce2
}

INFO:hazelcast.connection:Authenticated with server Address(host=172.18.0.2, port=5701):8230653f-f1fb-4c1f-ad16-ad8449004468, server version: 5.3.8, local address: Address(host=172.18.0.7,
 port=50870)
INFO:hazelcast.connection:Authenticated with server Address(host=172.18.0.3, port=5701):70be8dff-8599-4c0d-b0f4-0ce50e537ce2, server version: 5.3.8, local address: Address(host=172.18.0.7,
 port=39014)
INFO:hazelcast.client:Client started
INFO:__main__:[58f9b81d3700] Connected to Hazelcast cluster
INFO:__main__:[58f9b81d3700] Starting on port 50052
 * Serving Flask app 'logging-service'
 * Debug mode: off
INFO:werkzeug: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://127.0.0.1:50052
 * Running on http://172.18.0.7:50052
INFO:werkzeug:Press CTRL+C to quit
INFO:__main__:[58f9b81d3700] Received request: POST http://logging2:50052/logging
INFO:__main__:[58f9b81d3700] Logged message: msg1 with ID c4e7ef88-6ddf-4767-991c-0c7933033c8a
INFO:werkzeug:172.18.0.10 - - [30/May/2025 14:06:29] "POST /logging HTTP/1.1" 201 -
INFO:__main__:[58f9b81d3700] Received request: POST http://logging2:50052/logging
INFO:__main__:[58f9b81d3700] Logged message: msg4 with ID 960f973e-9f18-4d6b-a690-bf7a56b4e66e
INFO:werkzeug:172.18.0.10 - - [30/May/2025 14:06:39] "POST /logging HTTP/1.1" 201 -
INFO:__main__:[58f9b81d3700] Received request: POST http://logging2:50052/logging
INFO:__main__:[58f9b81d3700] Logged message: msg5 with ID 8a29863e-50e1-48a9-9b71-1d0397e64ce3
INFO:werkzeug:172.18.0.10 - - [30/May/2025 14:06:41] "POST /logging HTTP/1.1" 201 -
```



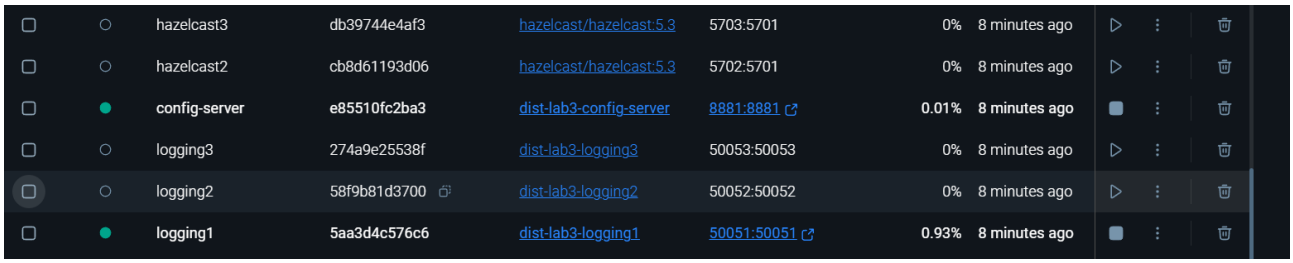
- **Через HTTP GET з *facade-service* прочитати повідомлення**
  На скріні видно що повідомлення з ідентичними id (наприклад, 6807890-messages, msg0, msg1 тощо) з'являються двічі, але обробляються різними екземплярами logging-service (наприклад, logging1, logging2, logging3). Це через механізм повторних спроб у façade.service.py(send_with_retry), де використовується бібліотека tenacity для повторних спроб надсилання запиту до logging-service

Вимкнути один/два екземпляри *logging-service* (разом з ним мають вимикатись й ноди Hazelcast) та перевірити чи зможемо прочитати повідомлення





```
PS D:\shared\dist-lab3> docker stop logging3 hazelcast3
logging3
hazelcast3
PS D:\shared\dist-lab3>
```



☐	○	hazelcast3	db39744e4af3	hazelcast/hazelcast:5.3	5703:5701	0%	8 minutes ago
☐	○	hazelcast2	cb8d61193d06	hazelcast/hazelcast:5.3	5702:5701	0%	8 minutes ago
☐	●	**config-server**	**e85510fc2ba3**	dist-lab3-config-server	8881:8881	**0.01%**	**8 minutes ago**
☐	○	logging3	274a9e25538f	dist-lab3-logging3	50053:50053	0%	8 minutes ago
☐	○	logging2	58f9b81d3700	dist-lab3-logging2	50052:50052	0%	8 minutes ago
☐	●	**logging1**	**5aa3d4c576c6**	dist-lab3-logging1	50051:50051	**0.93%**	**8 minutes ago**

```
D:\shared\dist-lab3>curl http://localhost:8880/messages
{"all_messages_from_hazelcast":[{"id":"67077f48-9f1f-45d4-b5e6-e9b54552e53d","text":"msg9 (processed by 5aa3d4c576c6)"},{"id":"251b7bd7-b42a-44ff-9eaa-202a3
640ef3d","text":"msg8 (processed by 274a9e25538f)"},{"id":"ac19f70b-ff59-4962-a80e-05609cb84e54","text":"msg2 (processed by 274a9e25538f)"},{"id":"02f9373c-
4c29-48b9-a710-5f5fec03839c","text":"msg10 (processed by 5aa3d4c576c6)"},{"id":"960f973e-9f18-4d6b-a690-bf7a56b4e66e","text":"msg4 (processed by 58f9b81d370
0)"},{"id":"6307189e-8555-47ea-8267-ad6feaf43726","text":"msg6 (processed by 274a9e25538f)"},{"id":"6bef903d-3a3a-4a5d-8a9a-aaae90011cf0","text":"msg3 (proc
essed by 274a9e25538f)"},{"id":"c4e7ef88-6ddf-4767-991c-0c7933033c8a","text":"msg1 (processed by 58f9b81d3700)"},{"id":"8a29863e-50e1-48a9-9b71-1d0397e64ce3
","text":"msg5 (processed by 58f9b81d3700)"},{"id":"4569265c-2b9c-4675-ac0d-08abaf34b4d3","text":"msg7 (processed by 274a9e25538f)"}],"logging_services":[{"
instance":"5aa3d4c576c6","messages":["msg9 (processed by 5aa3d4c576c6)","msg8 (processed by 274a9e25538f)","msg2 (processed by 274a9e25538f)","msg10 (proces
sed by 5aa3d4c576c6)","msg4 (processed by 58f9b81d3700)","msg6 (processed by 274a9e25538f)","msg3 (processed by 274a9e25538f)","msg1 (processed by 58f9b81d3
700)","msg5 (processed by 58f9b81d3700)","msg7 (processed by 274a9e25538f)"],"service":"http://logging1:50051/logging"},{"error":"HTTPConnectionPool(host='l
ogging2', port=50052): Max retries exceeded with url: /logging (Caused by NameResolutionError(\"<urllib3.connection.HTTPConnection object at 0x7f8277d03820>
: Failed to resolve 'logging2' ([Errno -2] Name or service not known)\"))","service":"http://logging2:50052/logging"},{"error":"HTTPConnectionPool(host='logg
ing3', port=50053): Max retries exceeded with url: /logging (Caused by NameResolutionError(\"<urllib3.connection.HTTPConnection object at 0x7f8277d01e10>:
Failed to resolve 'logging3' ([Errno -2] Name or service not known)\"))","service":"http://logging3:50053/logging"}]}
```