**Київський політехнічний інститут імені Ігоря Сікорського**

**Фізико-технічний інститут**

**Проектування розподілених систем**

**Лабораторна робота №5**

Варіант 1

**Виконала:**

Студентка групи ФБ-42мп

Алькова Аліна

**Lab 5 - Мікросервіси з використанням Service Discovery та Config Server на базі Consul**

Завдання базується на функціоналі розробленому у попередньому завданні, і є його подальшим розвитком.

До системи необхідно додати *Consul*, який буде виконувати роль *Service Register*, *Service Discovery* та *Config Server*.

Це дасть змогу явно не вказувати адреси мікросервісів та конфігурації для Hazelcast та Message Queue

*Façade-service.py*

import random  
import requests  
from flask import Flask, request, jsonify  
from kafka import KafkaProducer  
from kafka.errors import NoBrokersAvailable  
import json  
import time  
import consul  
import os  
import atexit  
  
app = Flask(\_\_name\_\_)  
CONSUL\_HOST = os.getenv("CONSUL\_HOST", "consul")  
CONSUL\_PORT = int(os.getenv("CONSUL\_PORT", "8500"))  
INSTANCE\_ID = os.getenv("INSTANCE\_ID", "facade1")  
PORT = 8880  
  
  
consul\_client = consul.Consul(host=CONSUL\_HOST, port=CONSUL\_PORT)  
  
  
def register\_service():  
 consul\_client.agent.service.register(  
 service\_id=INSTANCE\_ID,  
 name="facade-service",  
 address="facade-service",  
 port=PORT,  
 check=consul.Check.http(f"http://facade-service:{PORT}/health", interval="10s", timeout="5s")  
 )  
 print(f"[Facade] Зареєстровано в Consul як {INSTANCE\_ID}")  
  
  
def deregister\_service():  
 consul\_client.agent.service.deregister(INSTANCE\_ID)  
 print(f"[Facade] Видалено з Consul")  
  
  
@app.route("/health", methods=["GET"])  
def health():  
 return jsonify({"status": "healthy"}), 200  
  
  
def get\_kafka\_config():  
 for \_ in range(10):  
 try:  
 \_, brokers\_data = consul\_client.kv.get("config/kafka/brokers")  
 if brokers\_data is None:  
 raise ValueError("Ключ config/kafka/brokers відсутній у Consul")  
 brokers = json.loads(brokers\_data["Value"].decode("utf-8"))  
 \_, topic\_data = consul\_client.kv.get("config/kafka/topic")  
 if topic\_data is None:  
 raise ValueError("Ключ config/kafka/topic відсутній у Consul")  
 topic = json.loads(topic\_data["Value"].decode("utf-8"))  
 return brokers, topic  
 except Exception as e:  
 print(f"[Facade] Помилка отримання конфігурації Kafka з Consul: {e}")  
 time.sleep(5)  
 raise Exception("[Facade] Не вдалося отримати конфігурацію Kafka після кількох спроб")  
  
  
KAFKA\_BROKERS, KAFKA\_TOPIC = get\_kafka\_config()  
for i in range(10):  
 try:  
 producer = KafkaProducer(  
 bootstrap\_servers=KAFKA\_BROKERS,  
 value\_serializer=lambda v: json.dumps(v).encode('utf-8')  
 )  
 print("[Facade] Підключено до Kafka", flush=True)  
 break  
 except NoBrokersAvailable:  
 print(f"[Facade] Kafka недоступна (спроба {i+1}/10), повтор через 5с...", flush=True)  
 time.sleep(5)  
else:  
 raise Exception("[Facade] Kafka недоступна після кількох спроб")  
  
def get\_service\_urls(service\_name):  
 try:  
 \_, services = consul\_client.catalog.service(service\_name)  
 return [f"http://{s['ServiceAddress']}:{s['ServicePort']}" for s in services if s["ServiceID"]]  
 except Exception as e:  
 print(f"[Facade] Помилка виявлення сервісу {service\_name}: {e}")  
 return []  
  
def send\_to\_random\_instance(service\_name, path, method="GET", data=None):  
 urls = get\_service\_urls(service\_name)  
 random.shuffle(urls)  
 for url in urls:  
 try:  
 full\_url = f"{url}{path}"  
 if method == "POST":  
 r = requests.post(full\_url, json=data, timeout=3)  
 else:  
 r = requests.get(full\_url, timeout=3)  
 if r.status\_code in (200, 201):  
 return r.json()  
 except Exception as e:  
 print(f"[Facade] Попередження: не вдалося звернутися до {url}: {e}")  
 return {"error": "Помилка всіх екземплярів"}, 500  
  
@app.route("/entry", methods=["POST"])  
def post\_entry():  
 msg = request.json.get("msg")  
 if msg:  
 producer.send(KAFKA\_TOPIC, {"msg": msg})  
 print(f"[Facade] Відправлено до Kafka: {msg}")  
 response = send\_to\_random\_instance("logging-service", "/tracker", method="POST", data={"msg": msg})  
 if "error" in response:  
 print(f"[Facade] Не вдалося надіслати до logging-service: {response['error']}")  
 return jsonify({"error": "Не вдалося зберегти в logging-service"}), 500  
 print(f"[Facade] Успішно надіслано до logging-service: {response}")  
 return jsonify({"status": "Відправлено до Kafka та logging-service"}), 201  
 print("[Facade] Помилка: Відсутній msg у запиті")  
 return jsonify({"error": "Відсутній msg"}), 400  
  
@app.route("/entry", methods=["GET"])  
def get\_logs():  
 return send\_to\_random\_instance("logging-service", "/tracker", method="GET")  
  
@app.route("/notify", methods=["GET"])  
def notify():  
 return send\_to\_random\_instance("messages-service", "/messages", method="GET")  
  
@app.route("/combined", methods=["GET"])  
def get\_combined\_messages():  
 logging\_response = send\_to\_random\_instance("logging-service", "/tracker", method="GET")  
 logging\_messages = logging\_response.get("messages", []) if "error" not in logging\_response else []  
 messages\_response = send\_to\_random\_instance("messages-service", "/messages", method="GET")  
 messages\_service\_messages = messages\_response.get("messages", []) if "error" not in messages\_response else []  
 combined\_messages = list(set(logging\_messages + messages\_service\_messages))  
 return jsonify({"messages": combined\_messages})  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 register\_service()  
 atexit.register(deregister\_service)  
 app.run(host='0.0.0.0', port=PORT)

*logging-service.py:*

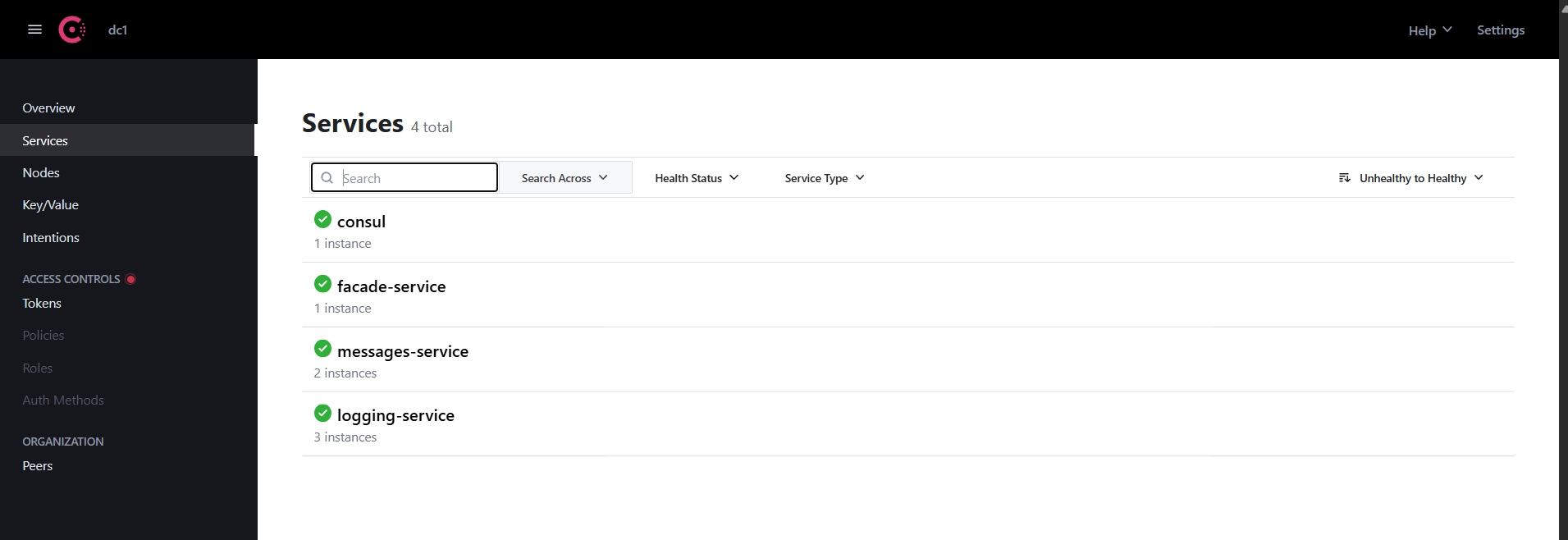
import atexit  
import time  
from flask import Flask, request, jsonify, json  
import hazelcast  
import os  
import logging  
import consul  
  
tracker = Flask(\_\_name\_\_)  
INSTANCE\_ID = os.getenv("INSTANCE\_ID", "default")  
PORT = int(os.getenv("PORT", "8881"))  
CONSUL\_HOST = os.getenv("CONSUL\_HOST", "consul")  
CONSUL\_PORT = int(os.getenv("CONSUL\_PORT", "8500"))  
  
tracker.logger.setLevel(logging.DEBUG)  
  
  
consul\_client = consul.Consul(host=CONSUL\_HOST, port=CONSUL\_PORT)  
  
  
def register\_service():  
 consul\_client.agent.service.register(  
 service\_id=INSTANCE\_ID,  
 name="logging-service",  
 address="logging-service" + INSTANCE\_ID[-1],  
 port=PORT,  
 check=consul.Check.http(f"http://logging-service{INSTANCE\_ID[-1]}:{PORT}/health", interval="10s", timeout="5s")  
 )  
 print(f"[{INSTANCE\_ID}] Зареєстровано в Consul як {INSTANCE\_ID}")  
  
  
def deregister\_service():  
 consul\_client.agent.service.deregister(INSTANCE\_ID)  
 print(f"[{INSTANCE\_ID}] Видалено з Consul")  
  
  
@tracker.route("/health", methods=["GET"])  
def health():  
 return jsonify({"status": "healthy"}), 200  
  
def get\_hazelcast\_config():  
 for attempt in range(10):  
 try:  
 \_, members\_data = consul\_client.kv.get("config/hazelcast/cluster\_members")  
 print(f"[{INSTANCE\_ID}] Attempt {attempt + 1}: members\_data = {members\_data}")  
 if members\_data is None:  
 raise ValueError("Ключ config/hazelcast/cluster\_members відсутній у Consul")  
 members = json.loads(members\_data["Value"].decode("utf-8"))  
 \_, cluster\_name\_data = consul\_client.kv.get("config/hazelcast/cluster\_name")  
 print(f"[{INSTANCE\_ID}] Attempt {attempt + 1}: cluster\_name\_data = {cluster\_name\_data}")  
 if cluster\_name\_data is None:  
 raise ValueError("Ключ config/hazelcast/cluster\_name відсутній у Consul")  
 cluster\_name = json.loads(cluster\_name\_data["Value"].decode("utf-8"))  
 print(f"[{INSTANCE\_ID}] Successfully retrieved Hazelcast config: members={members}, cluster\_name={cluster\_name}")  
 return members, cluster\_name  
 except Exception as e:  
 print(f"[{INSTANCE\_ID}] Attempt {attempt + 1} failed: {e}")  
 time.sleep(5)  
 raise Exception(f"[{INSTANCE\_ID}] Не вдалося отримати конфігурацію Kafka після 10 спроб")  
  
  
  
HAZELCAST\_MEMBERS, HAZELCAST\_CLUSTER\_NAME = get\_hazelcast\_config()  
hz = hazelcast.HazelcastClient(  
 cluster\_members=HAZELCAST\_MEMBERS,  
 cluster\_name=HAZELCAST\_CLUSTER\_NAME  
)  
records = hz.get\_map("logs").blocking()  
  
@tracker.route("/tracker", methods=["POST", "GET"])  
def handle():  
 if request.method == "POST":  
 if request.is\_json:  
 payload = request.get\_json()  
 identifier = payload.get("msg")  
 info = payload.get("msg")  
 else:  
 identifier = request.form.get("key")  
 info = request.form.get("data")  
  
 if not (identifier and info):  
 return jsonify({"error": "Відсутній ключ або дані"}), 400  
  
 if records.contains\_key(identifier):  
 tracker.logger.info(f"[{INSTANCE\_ID}] Пропущено: {identifier}")  
 return jsonify({"status": "Дублікат"}), 201  
  
 records.put(identifier, info)  
 tracker.logger.info(f"[{INSTANCE\_ID}] Збережено: {info} [Key: {identifier}]")  
 return jsonify({"status": "OK"}), 201  
  
 elif request.method == "GET":  
 values = records.values()  
 return jsonify({"messages": list(values)})  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 time.sleep(20)  
 register\_service()  
 atexit.register(deregister\_service)  
 tracker.logger.info(f"[{INSTANCE\_ID}] Запуск logging-service на порті {PORT}")  
 tracker.run(host='0.0.0.0', port=PORT)

*messages-service.py:*

from flask import Flask, jsonify  
from kafka import KafkaConsumer  
from kafka.errors import NoBrokersAvailable  
import json  
import threading  
import os  
import time  
import consul  
import atexit  
  
app = Flask(\_\_name\_\_)  
INSTANCE\_ID = os.getenv("INSTANCE\_ID", "default")  
PORT = int(os.getenv("PORT", "8890"))  
CONSUL\_HOST = os.getenv("CONSUL\_HOST", "consul")  
CONSUL\_PORT = int(os.getenv("CONSUL\_PORT", "8500"))  
  
messages = []  
  
consul\_client = consul.Consul(host=CONSUL\_HOST, port=CONSUL\_PORT)  
  
def register\_service():  
 consul\_client.agent.service.register(  
 service\_id=INSTANCE\_ID,  
 name="messages-service",  
 address="messages-service" + INSTANCE\_ID[-1],  
 port=PORT,  
 check=consul.Check.http(f"http://messages-service{INSTANCE\_ID[-1]}:{PORT}/health", interval="10s", timeout="5s")  
 )  
 print(f"[{INSTANCE\_ID}] Зареєстровано в Consul як {INSTANCE\_ID}")  
  
  
def deregister\_service():  
 consul\_client.agent.service.deregister(INSTANCE\_ID)  
 print(f"[{INSTANCE\_ID}] Видалено з Consul")  
  
  
@app.route("/health", methods=["GET"])  
def health():  
 return jsonify({"status": "healthy"}), 200  
  
def get\_kafka\_config():  
 for attempt in range(10):  
 try:  
 \_, brokers\_data = consul\_client.kv.get("config/kafka/brokers")  
 print(f"[{INSTANCE\_ID}] Attempt {attempt + 1}: brokers\_data = {brokers\_data}")  
 if brokers\_data is None:  
 raise ValueError("Ключ config/kafka/brokers відсутній у Consul")  
 brokers = json.loads(brokers\_data["Value"].decode("utf-8"))  
 \_, topic\_data = consul\_client.kv.get("config/kafka/topic")  
 if topic\_data is None:  
 raise ValueError("Ключ config/kafka/topic відсутній у Consul")  
 topic = json.loads(topic\_data["Value"].decode("utf-8"))  
 print(f"[{INSTANCE\_ID}] Successfully retrieved Kafka config: brokers={brokers}, topic={topic}")  
 return brokers, topic  
 except Exception as e:  
 print(f"[{INSTANCE\_ID}] Attempt {attempt + 1} failed: {e}")  
 time.sleep(5)  
 raise Exception(f"[{INSTANCE\_ID}] Не вдалося отримати конфігурацію Kafka після 10 спроб")  
  
def consume():  
 KAFKA\_BROKERS, KAFKA\_TOPIC = get\_kafka\_config()  
 for i in range(10):  
 try:  
 consumer = KafkaConsumer(  
 KAFKA\_TOPIC,  
 bootstrap\_servers=KAFKA\_BROKERS,  
 auto\_offset\_reset="earliest",  
 group\_id="group-messages",  
 value\_deserializer=lambda m: json.loads(m.decode('utf-8'))  
 )  
 print(f"[{INSTANCE\_ID}] Підключено до Kafka", flush=True)  
 break  
 except NoBrokersAvailable:  
 print(f"[{INSTANCE\_ID}] Kafka недоступна (спроба {i+1}/10), повтор через 5с...", flush=True)  
 time.sleep(5)  
 else:  
 raise Exception(f"[{INSTANCE\_ID}] Kafka недоступна після кількох спроб")  
  
 print(f"[{INSTANCE\_ID}] Запуск циклу споживача Kafka", flush=True)  
 for msg in consumer:  
 content = msg.value.get("msg")  
 if content:  
 messages.append(content)  
 print(f"[{INSTANCE\_ID}] Отримано з Kafka: {content}", flush=True)  
  
@app.route("/messages", methods=["GET"])  
def get\_messages():  
 return jsonify({"messages": messages})  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 time.sleep(20)  
 register\_service()  
 atexit.register(deregister\_service)  
 print(f"[{INSTANCE\_ID}] Запуск на порті {PORT}")  
 threading.Thread(target=consume, daemon=True).start()  
 app.run(host='0.0.0.0', port=PORT)

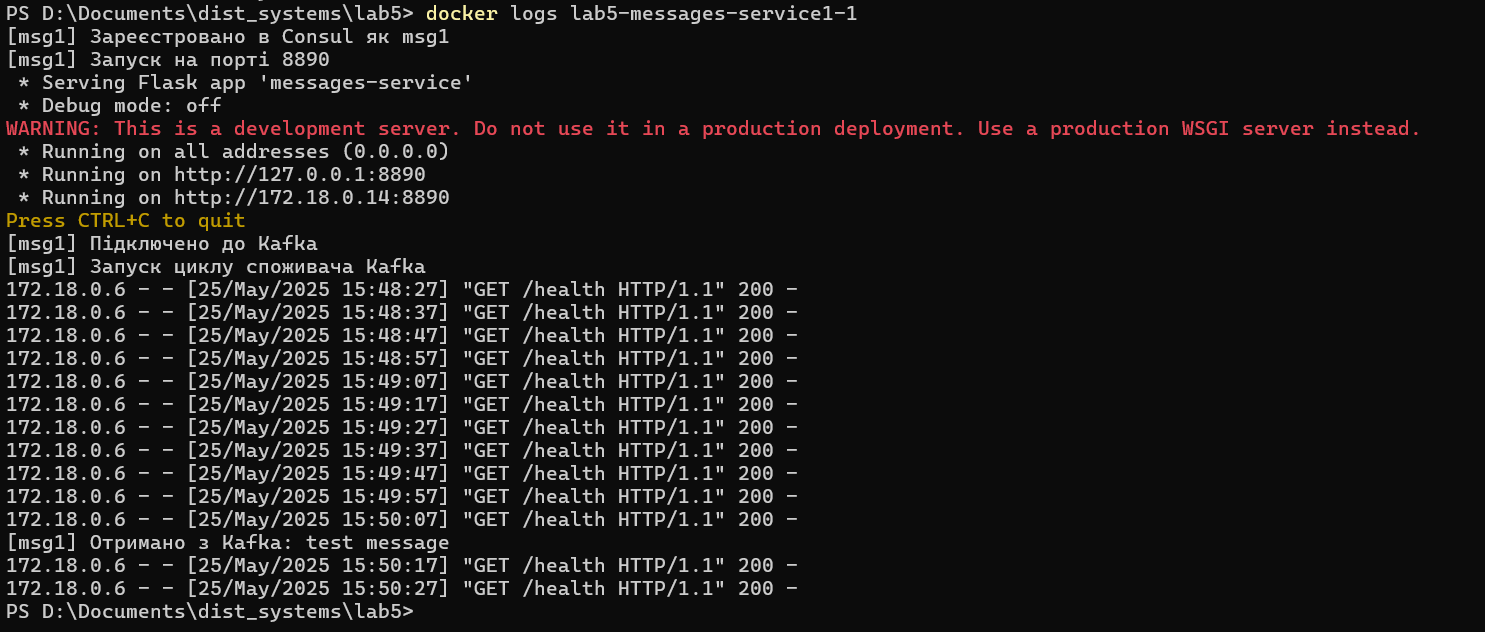
Вимоги:

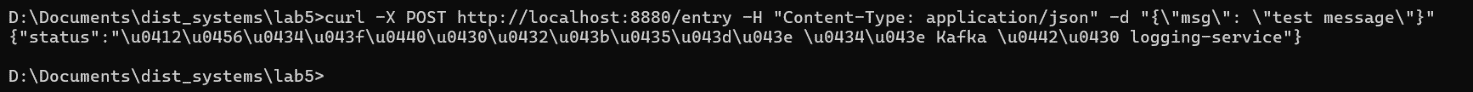
1. Всі мікросервіси мають самостійно динамічно реєструватись при старті у *Consul*, кожного з сервісів може бути запущено кілька екземплярів (це буде відображатись як кількість instances на UI):

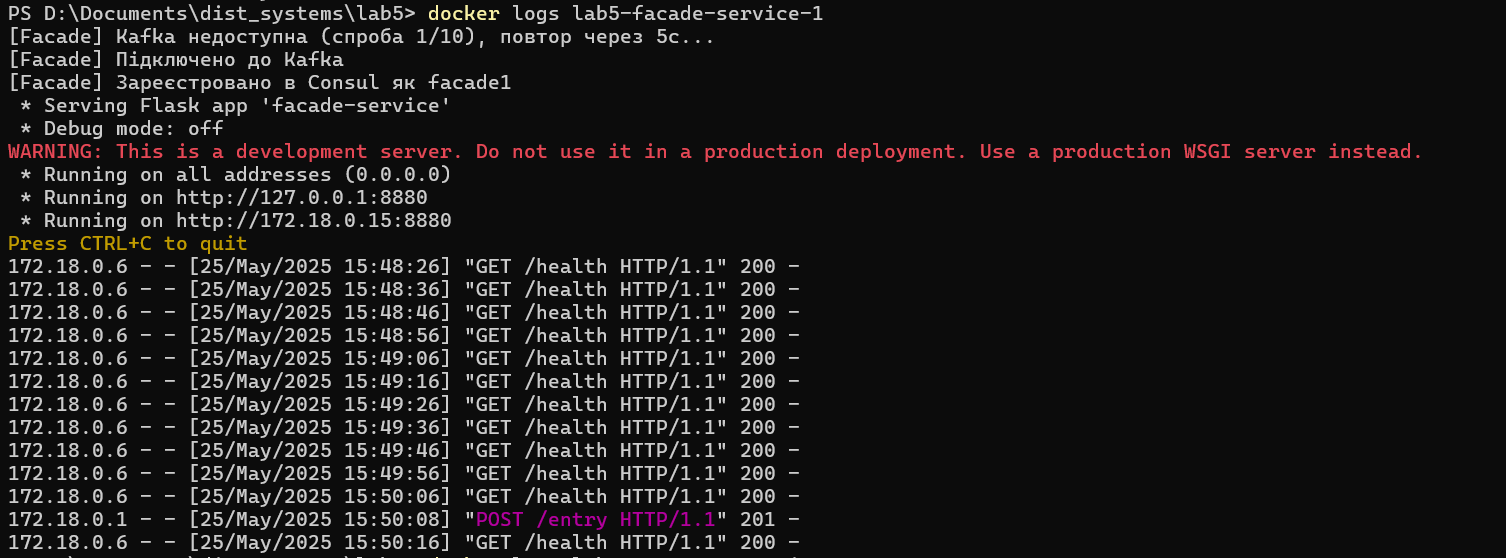
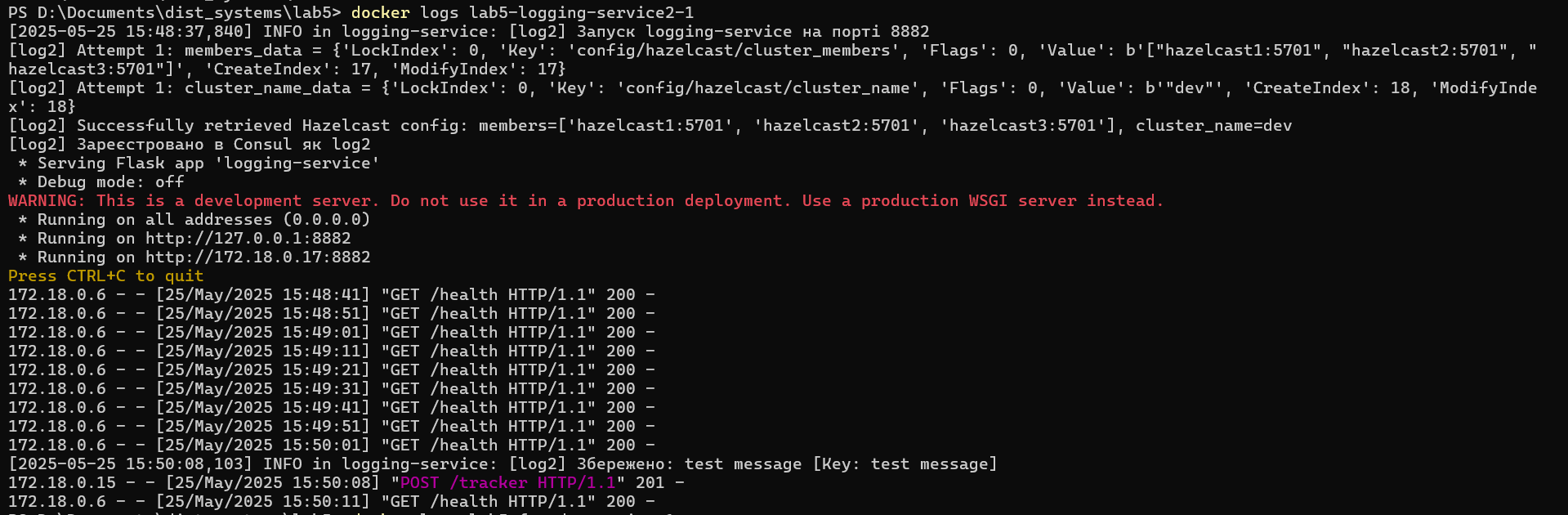
* ***facade-service***
* ***logging-service***
* ***messages-service***
* У facade-service.py, logging-service.py і messages-service.py є функції register\_service(), які викликаються при запуску (if \_\_name\_\_ == "\_\_main\_\_").
* У docker-compose.yml усі сервіси (facade-service, messages-service1/2, logging-service1/2/3) залежать від consul і consul-init, що гарантує їхній запуск після ініціалізації Consul.
* У init-consul.sh ключі для Hazelcast і Kafka ініціалізуються (config/hazelcast/cluster\_members, config/hazelcast/cluster\_name, config/kafka/brokers, config/kafka/topic).

2. При звертанні ***facade-service***до ***logging-service*** та ***messages-service***, IP-адреси (і порти) мають зчитуватись ***facade-service*** з *Consul.* Немає бути задано в коді чи конфігураціях статичних значень адрес.

 У facade-service.py функція get\_service\_urls(service\_name) використовує consul\_client.catalog.service(service\_name) для отримання списку екземплярів із Consul. Адреси (ServiceAddress) і порти (ServicePort) зчитуються динамічно.

 У send\_to\_random\_instance() вибір випадкового екземпляра і звернення до нього (наприклад, /tracker для logging-service або /messages для messages-service) базується на отриманих адресах.



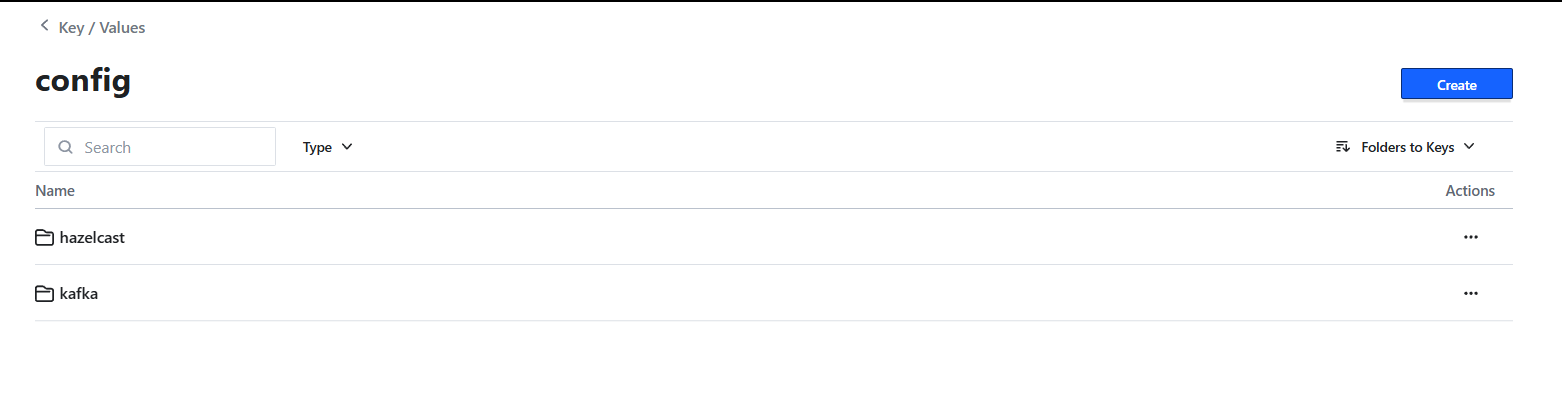


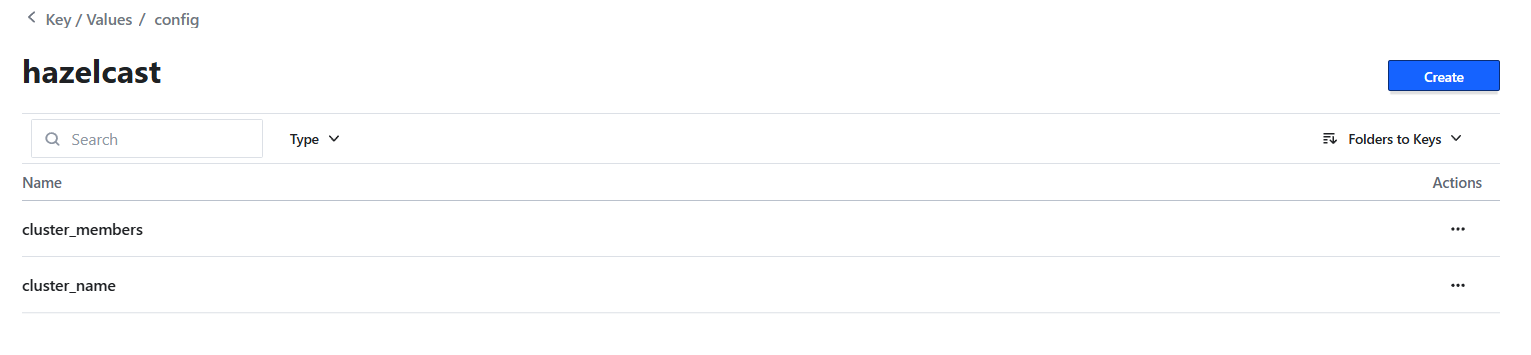
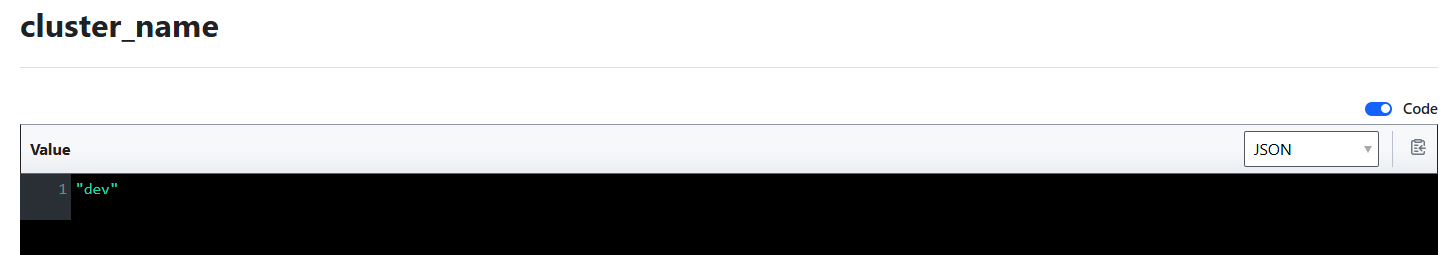
3. Налаштування для клієнтів Hazelcast мають зберігатись як key/value у *Consul* і зчитуватись ***logging-service***

 У logging-service.py функція get\_hazelcast\_config() зчитує ключі config/hazelcast/cluster\_members і config/hazelcast/cluster\_name з Consul.

 Логи показують успішне отримання конфігурації (наприклад, Successfully retrieved Hazelcast config: members=...).

 У init-consul.sh ці ключі ініціалізуються:

* config/hazelcast/cluster\_members = ["hazelcast1:5701", "hazelcast2:5701", "hazelcast3:5701"]
* config/hazelcast/cluster\_name = "dev".

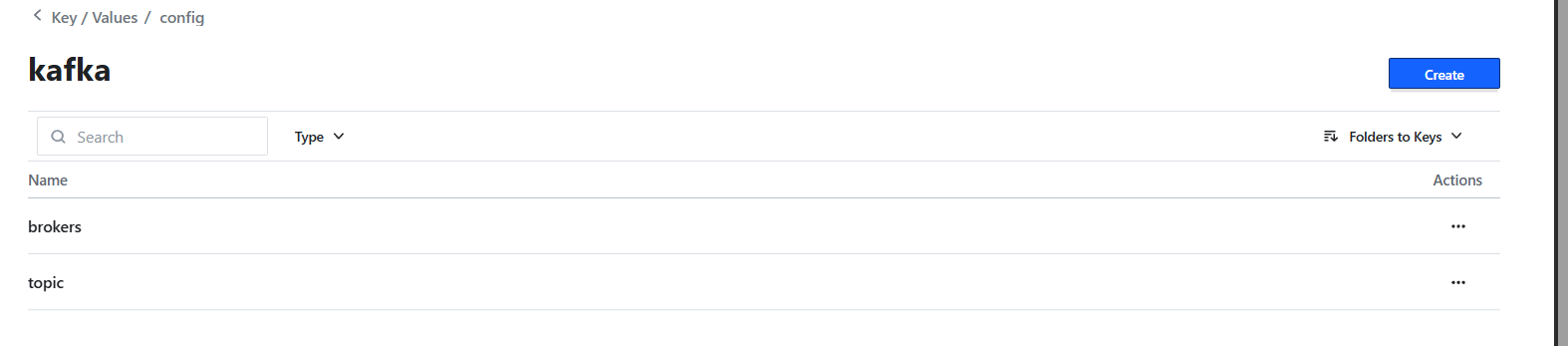


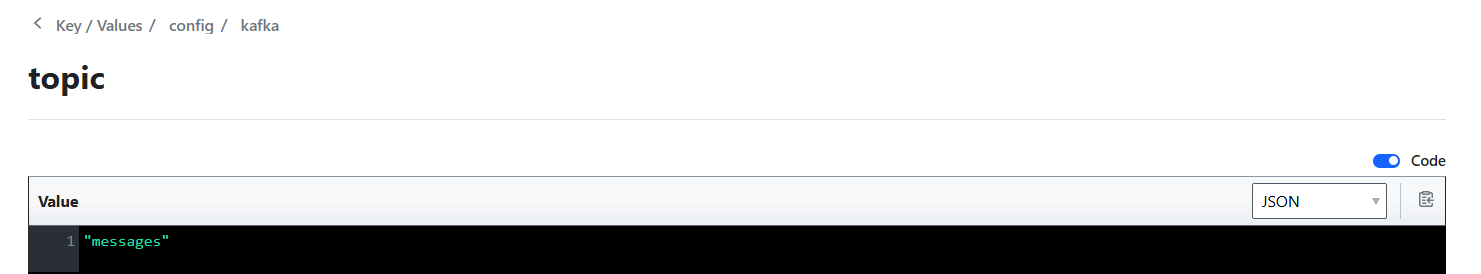
4. Налаштування для Message Queue (адреса, назва черги, …) мають зберігатись як key/value у *Consul* і зчитуватись ***facade-service*** та ***messages-service***

 У init-consul.sh ініціалізуються:

* config/kafka/brokers = ["kafka1:9092", "kafka2:9093", "kafka3:9094"]
* config/kafka/topic = "messages".

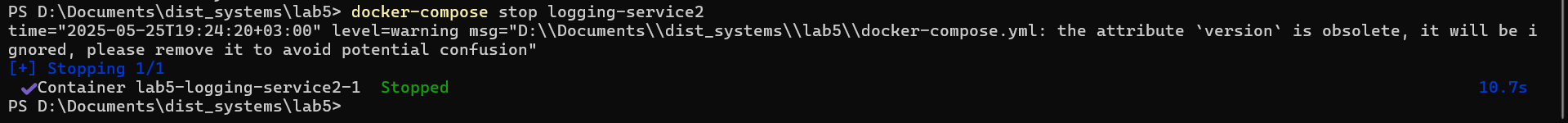
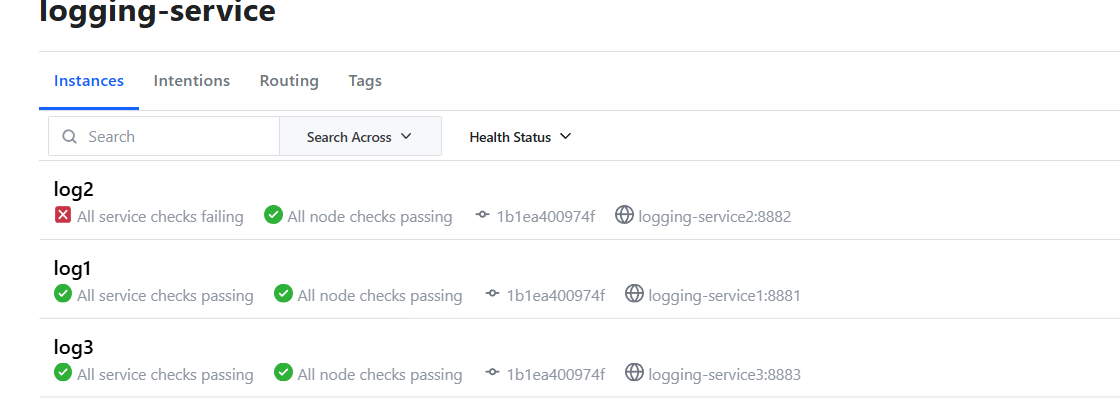
 У messages-service.py функція get\_kafka\_config() зчитує ці ключі.

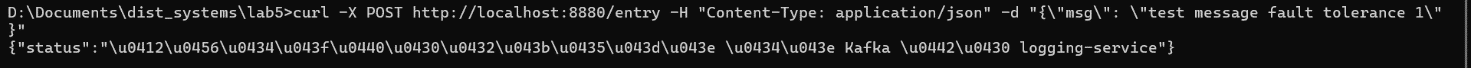
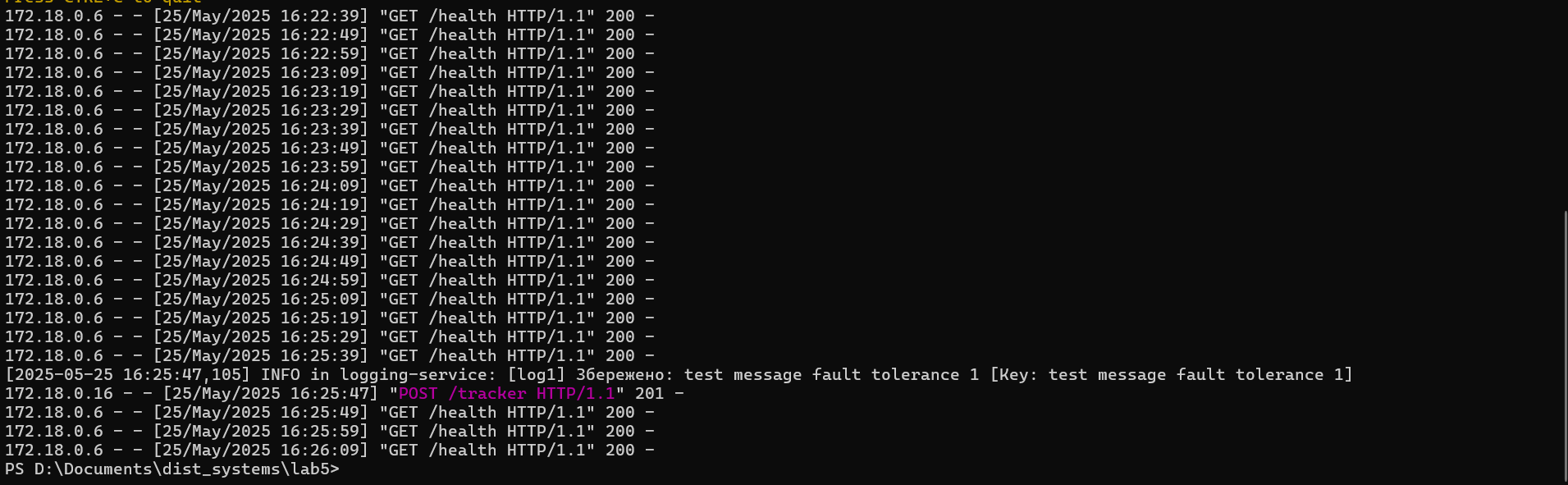
 У facade-service.py get\_kafka\_config() також зчитує ці ключі без статичних значень.

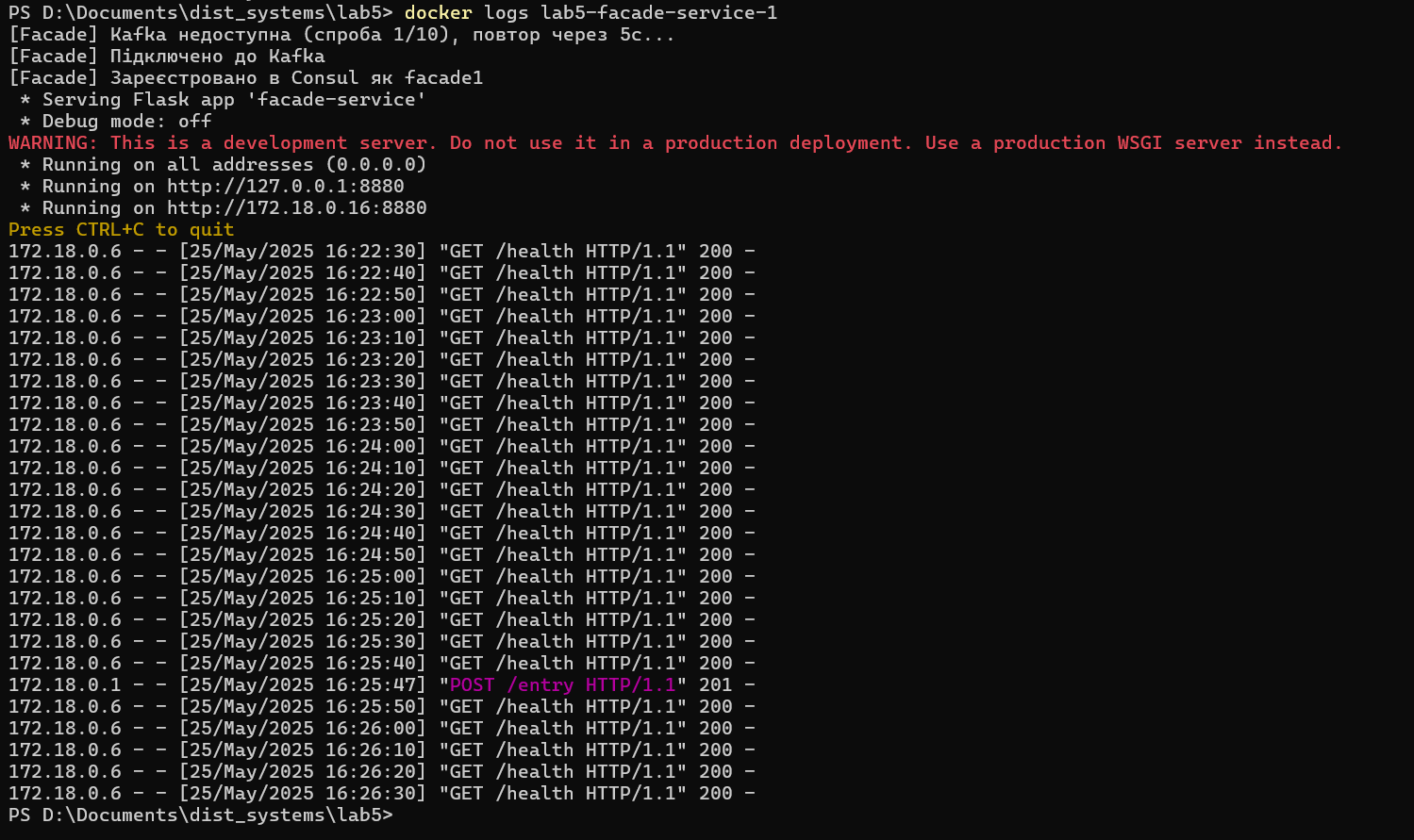


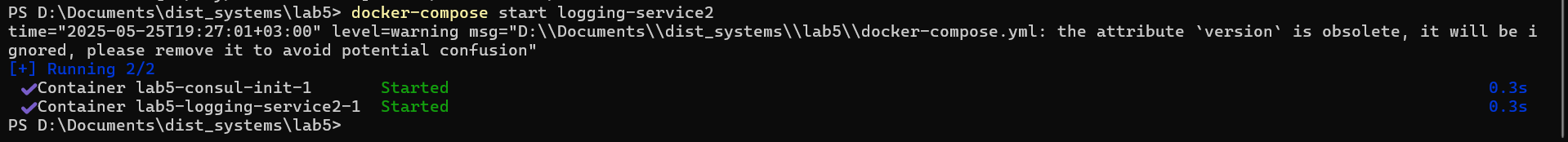
5. Продемонструвати, що у випадку відключення екземпляру певного мікросервісу, це буде відображатись у Consul (відключений екземпляр сервісу змінить статус) , а виклики будуть перенаправлятись до інших працюючих екземплярів.

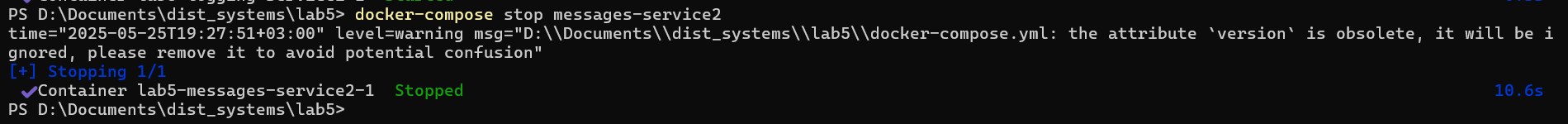
* Consul налаштований на перевірку стану сервісів через HTTP-ендпоінт /health кожні 10 секунд:
  + У facade-service.py: check=consul.Check.http(f"http://facade-service:{PORT}/health", interval="10s", timeout="5s").
  + У messages-service.py: check=consul.Check.http(f"http://messages-service{INSTANCE\_ID[-1]}:{PORT}/health", interval="10s", timeout="5s").
  + У logging-service.py: check=consul.Check.http(f"http://logging-service{INSTANCE\_ID[-1]}:{PORT}/health", interval="10s", timeout="5s").
* У facade-service функція send\_to\_random\_instance автоматично перенаправляє запити до доступних екземплярів, якщо один із них недоступний:

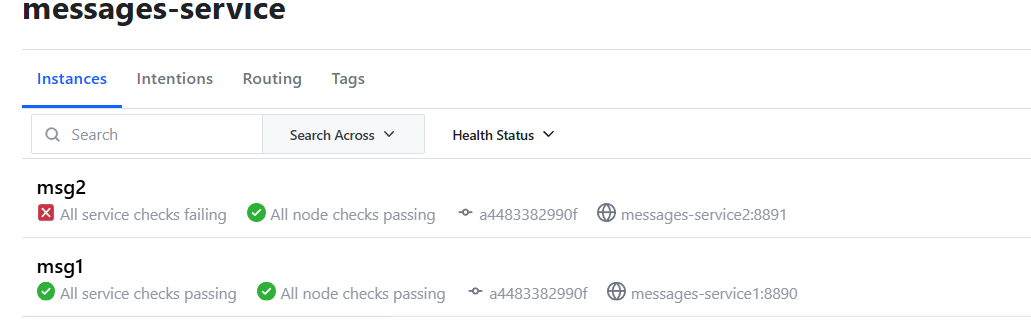
Зупинемо один з logging-service:

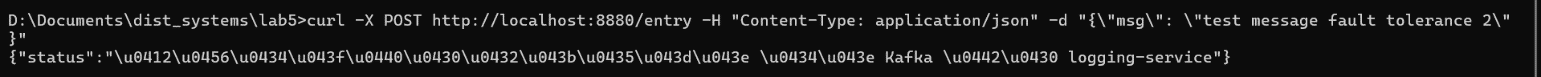
Надішлемо запит:

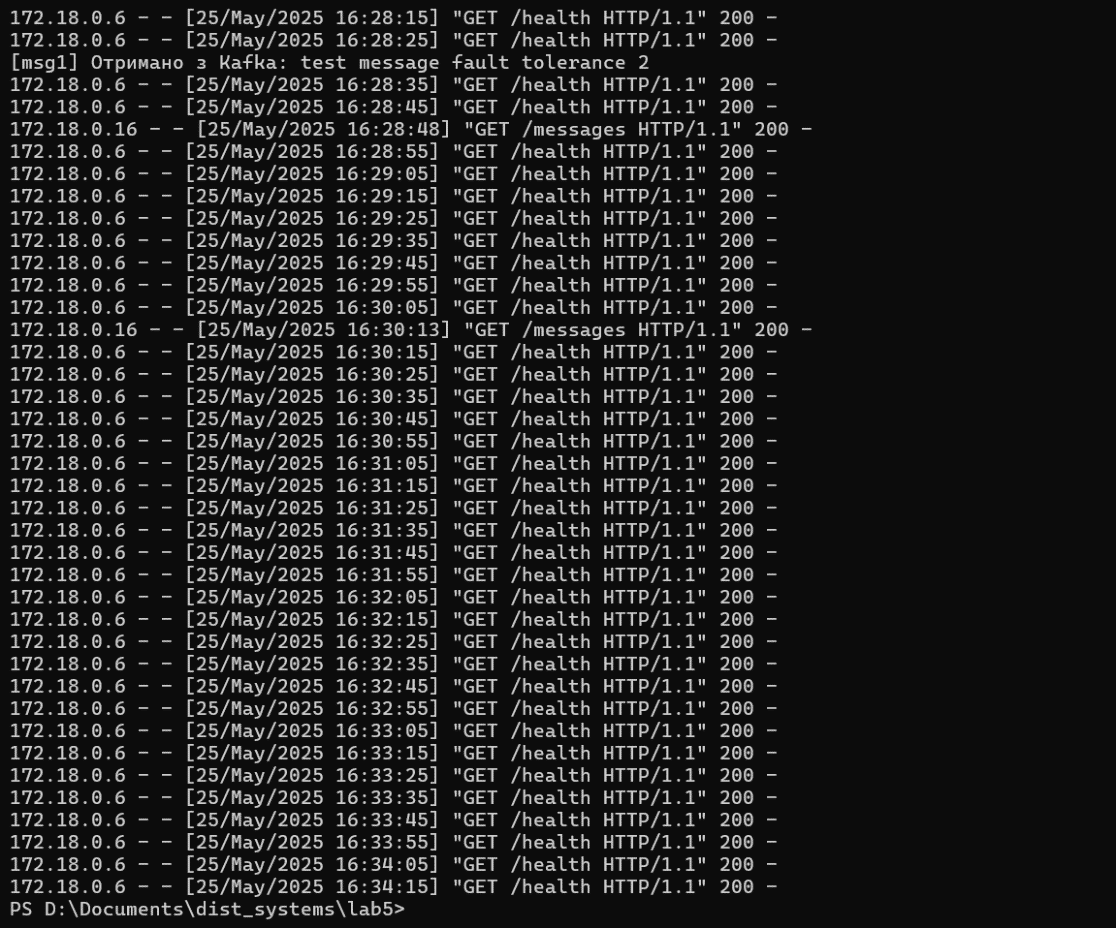
Помилок немає. Запит був надісланий до одного з доступних екземплярів (тут logging-service1). Повідомлення було збережено

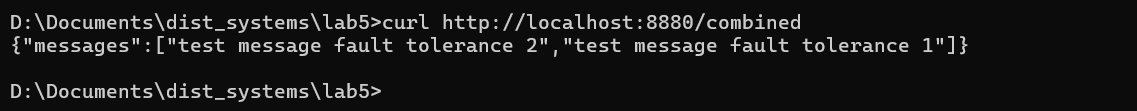
Запустимо заново:

Перевіримо те саме з одним з екземплярів messages-service:



Надсилаємо запит:

Логи з messages-service1:



Запити перенаправляються, а статус у Consul оновлюється.