



Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Фізико-технічний інститут

Лабораторна робота №5
з дисципліни
«Проектування розподілених систем»
на тему:
«Мікросервиси з використанням Service Discovery та Config Server на базі
Consul»

Виконав:
студент групи ФБ-31мп
Щур Павло
Перевірив:
Родіонов А. М.





Посилання на GitHub: https://github.com/ShchurPavlo/distributed-systems-design-2024/tree/micro_consul/lab5

Виконання завдань:

1) Розгорнемо Consul як окремий Docker контейнер:

```
C:\Users\Pavlo Shchur>docker run -d -p 8500:8500 -p 8600:8600/udp --name=consul_node consul:1.15.4 agent -server -ui -no-de=server-1 -bootstrap-expect=1 -client=0.0.0.0
b666b5a77496f57c21118d630e14b588f168aa200d41e83182b957074eb963dc

C:\Users\Pavlo Shchur>
```

<input type="checkbox"/>	Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	 consul_node b666b5a77496	consul:1.15.4	Running	8500:8500 Show all ports (2)	0.51%	3 minutes ago	  

2) Реалізуємо окрему “бібліотеку” `consul_lib.py` з функціями роботи з consul. Там реалізуємо функції реєстрації/дереєстрації сервісу, запису та зчитування даних в форматі key/value :

```
from random import choice
import consul
import json
import uuid

consul_client = consul.Consul(host="192.168.0.107", port=8500)
def Register_service(name, port):
    id=str(uuid.uuid4())
    consul_client.agent.service.register(name,id,address="192.168.0.107",port=port)
    return id
def Deregister_service(id):
    consul_client.agent.service.deregister(id)
def Get_port(service_name):
    ports=[]
    for _, service_info in consul_client.agent.services().items():
        if service_info['Service'] == service_name:
            ports.append(service_info['Port'])
    result=f"http://127.0.0.1:{choice(ports)}/data"
    return result
def Add_value(key, value):
    value_json = json.dumps(value)
    consul_client.kv.put(key, value_json)
def Get_value(key):
    _, config = consul_client.kv.get(key)
    return config['Value']
```

3) Задамо значення для клієнтів Hazelcast як key/value:

< Key / Values

HZ_config

Value

```
1 {
2   "cluster_name": "ps_cluster",
3   "map_name": "messages",
4   "queue_name": "queue"
5 }
```

Save

Cancel

3) Додамо функціонал реєстрації/дереєстрації в код існуючих мікросервісів:

```
12
13     service_id = Register_service(name: 'logging-service', args.logport)

13
14     service_id = Register_service(name: 'message-service', args.port)
15

7
8     service_id = Register_service(name: 'facade-service', port: 5000)
9
```

4) Реалізуємо отримання даних про Hazelcast з Consul:

```
hz_config = json.loads(Get_value('HZ_config'))
print("Hazelcast config: ", hz_config)

hz = hazelcast.HazelcastClient(cluster_name=hz_config['cluster_name'], cluster_members=[])
messages_queue = hz.get_queue(hz_config['queue_name']).blocking()
```

5) Запустимо наші мікросервіси та перевіримо реєстрацію:

Services 4 total	
Q Search	Search Across Health Status Service Type Unhealthy to Healthy
✓ consul	1 instance
✓ facade-service	1 instance
✓ logging-service	3 instances
✓ message-service	2 instances

6) Пересвідчимося в коректності роботи мікросервісів, запустимо client.py та зробимо декілька записів даних:

```
PS D:\KPI Styding\DistSystems\Lab5> python .\client.py
1)Sent POST-request
2)Sent GET-request
3)Exit
Your choose:1
Message:msg2
POST відповідь: Success!
1)Sent POST-request
2)Sent GET-request
3)Exit
Your choose:1
Message:msg3
POST відповідь: Success!
1)Sent POST-request
2)Sent GET-request
3)Exit
Your choose:1
Message:msg4
POST відповідь: Success!
1)Sent POST-request
2)Sent GET-request
3)Exit
Your choose:2
GET відповідь: {'Log data': 'msg2\nmsg1\nmsg4\nmsg3', 'Message data': ''}
1)Sent POST-request
2)Sent GET-request
3)Exit
```

The screenshot displays three overlapping Windows PowerShell windows. The top window shows the execution of `python .\logging_service.py --hzport 5703 --logport 5003`, which starts a Flask app on `http://127.0.0.1:5003` and logs incoming data. The bottom-left window shows `python .\logging_service.py --hzport 5702 --logport 5002`, starting another instance on `http://127.0.0.1:5002`. The bottom-right window shows the execution of `python .\logging_service.py --hzport 5702 --logport 5002` again, but with a different Hazelcast config, and it shows the receipt of a GET request for `/data HTTP/1.1`.