



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

Лабораторна робота №3
з дисципліни
«Проектування розподілених систем»
на тему:
«Мікросервиси з використанням Hazelcast Distributed Map»

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

Посилання на GitHub: https://github.com/ShchurPavlo/distributed-systems-design-2024/tree/micro_basics/lab3

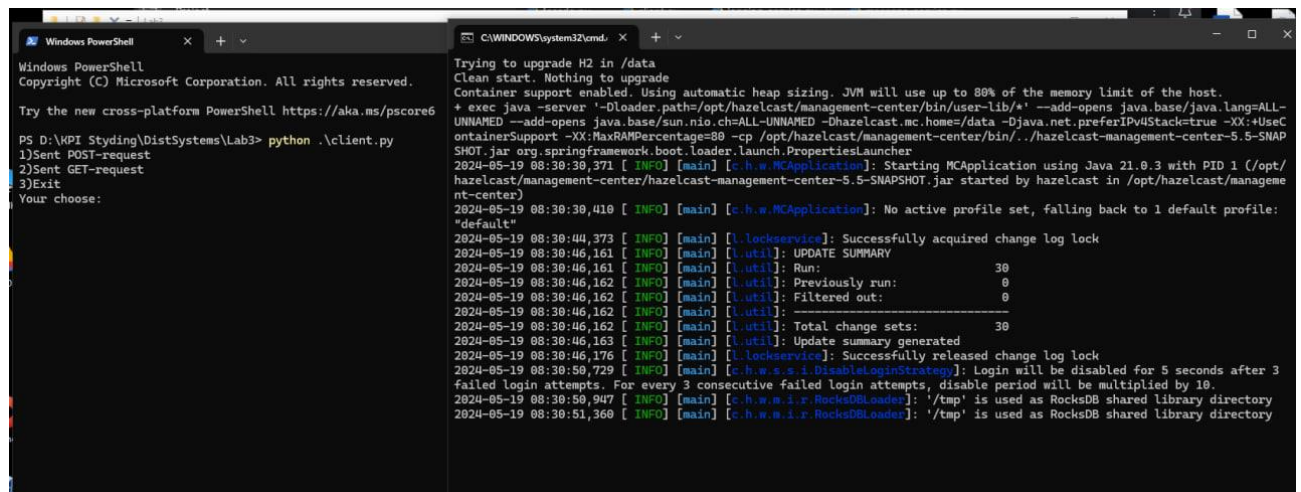
Виконання завдань:

1) Запустимо client.py який в свою чергу пропонує користувацьке меню та запускає management center в окремому контейнері docker, та прокидує веб-доступ на нього по 8080 порту:

```
import requests
import subprocess

command = f'start cmd /k "docker run --network hazelcast-network -p 8080:8080 hazelcast/management-center:latest-snapshot"'
subprocess.run(command, shell=True, check=True)

while(True):
    print("1)Sent POST-request \n2)Sent GET-request \n3)Exit")
    choose=int(input("Your choose:"))
    if choose==3:
        exit()
    elif choose==1:
        data = input("Message:")
        response_post = requests.post('http://127.0.0.1:5000/data', data=data)
        print('POST відповідь:', response_post.text)
    elif choose==2:
        # Відправка GET запиту
        response_get = requests.get('http://127.0.0.1:5000/data')
        print('GET відповідь:', response_get.json())
    else:
        print("Retry")
```



The screenshot shows a Windows terminal window with two panes. The left pane shows the execution of a Python script named client.py. The user enters '1' to send a POST request, then '2' to send a GET request, and finally '3' to exit. The right pane shows the output of a Docker container running the hazelcast/management-center:latest-snapshot image. The output includes logs for the application startup, JVM configuration, and the successful acquisition and release of a change log lock. The logs also show the application's status and the use of the /tmp directory as a shared library directory.

2) Тепер можна запусити facade-service:

```
from flask import Flask, request, jsonify
import random
import requests

def generate_unique_key():
    return ''.join(random.choices('123456789', k=4))

app = Flask(__name__)
@app.route('/data', methods=['GET', 'POST'])
def handle_data():
    if request.method == 'GET':
        port = random.randint(5001, 5003)
        response_logging = requests.get(f'http://127.0.0.1:{port}/data',
timeout=5)
        response_logging.raise_for_status()
        response_message = requests.get(f'http://127.0.0.1:5005/data').text
        return jsonify({'Message data': response_message, 'Log data':
response_logging.text})
    elif request.method == 'POST':
        message = request.get_data().decode('utf-8')
        key=generate_unique_key()
        port = random.randint(5001, 5003)
        data = {"key": key, "msg": message}
        print(data)
        response = requests.post(f'http://127.0.0.1:{port}/data', data=data)
        print("Response:", response.text)
        data = response.text
        return data

if __name__ == '__main__':
    app.run(debug=True, port=5000)
```

Ta messages-service:

```
from flask import Flask, jsonify, request
import requests

messages = {}

app = Flask(__name__)
@app.route('/data', methods=['GET', 'POST'])
def data():
    if request.method == 'GET':
        msg="Not implemented yet"
        return msg
    else:
        return jsonify({'error': 'Bad request'})

if __name__ == '__main__':
    app.run(debug=True, port=5005)
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\KPI Styding\DistSystems\Lab3> python .\facade.py
* Serving Flask app 'facade'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 318-908-639

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\KPI Styding\DistSystems\Lab3> python .\message_service.py
* Serving Flask app 'message_service'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5005
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 318-908-639
```

3) Тепер можна запустити logging-service:

```
from flask import Flask, jsonify, request
import requests
import hazelcast
import argparse
import subprocess

parser = argparse.ArgumentParser()
parser.add_argument("--logport", type=int, required=True)
parser.add_argument("--hzport", type=int, required=True)
args = parser.parse_args()

port = args.hzport
command = f'start cmd /k "docker run -it --name {port}-member --network hazelcast-network --rm -e HZ_NETWORK_PUBLICADDRESS=192.168.0.107:{port} -e HZ_CLUSTERNAME=ps_cluster -p {port}:5701 hazelcast/hazelcast:5.4.0"'
subprocess.run(command, shell=True, check=True)

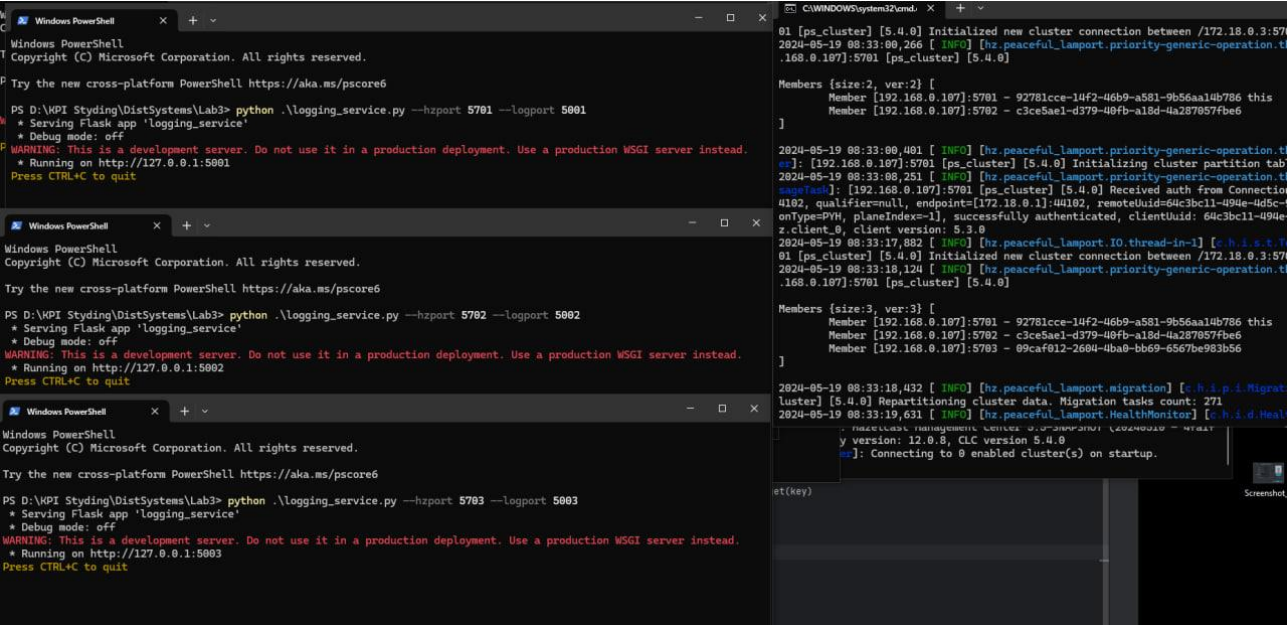
app = Flask(__name__)

hz = hazelcast.HazelcastClient(cluster_name="ps_cluster", cluster_members=[])
messages_map = hz.get_map("messages").blocking()

@app.route('/data', methods=['GET', 'POST'])
def data():
    if request.method == "POST":
        key = request.form['key']
        message = request.form['msg']
        messages_map.put(key, message)
        print("New writing data:", key, message)
        return ("Success!")
    elif request.method == "GET":
        keys = messages_map.key_set()
        messages = []
        for key in keys:
            message = messages_map.get(key)
            messages.append(message)
        return "\n".join(messages)

if __name__ == '__main__':
    app.run(port=args.logport)
```

logging.py також запускає Docker контейнер з Hazelcast, а в якості аргументів приймає номери портів ноди та самого logging-service:



В Docker отримаємо такий набір контейнерів:

<input type="checkbox"/>	Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	eager_ride	hazelcast/ma	Running	8080:8080	1.71%	47 minutes ago	
<input type="checkbox"/>	5701-member	hazelcast/haz	Running	5701:5701	1.33%	29 minutes ago	
<input type="checkbox"/>	5702-member	hazelcast/haz	Running	5702:5701	1.41%	28 minutes ago	
<input type="checkbox"/>	5703-member	hazelcast/haz	Running	5703:5701	1.83%	28 minutes ago	

4) Відкриємо management center та переконаємось в роботоздатності кластера:

Members										
<div>Search Default View </div>										
Member	Additi...	Script...	Cons...	Strea...	Slow ...	Hazel...	Owned Partitions	OS Committed Vi...	OS CP...	
192.168.0.107:5701		Disabled	Disabled	Enabled	No	5.4.0	91	6.35 GB	2.91 %	
192.168.0.107:5702		Disabled	Disabled	Enabled	No	5.4.0	90	6.36 GB	5.42 %	
192.168.0.107:5703		Disabled	Disabled	Enabled	No	5.4.0	90	6.36 GB	5.93 %	
1 - 3 of 3 Rows										

Також можна побачити список клієнтів – 3 екземпляри logging service + management center:

Name	Address	Type	Member Connection	Hazelcast Client Version	UUID
hz.client_0	172.18.0.1	Python	ALL	5.3.0	260d453d-a251-4bc8-9cc7-...
hz.client_0	172.18.0.1	Python	ALL	5.3.0	e55c9d33-1cfd-4304-af8b-...
hz.client_0	172.18.0.1	Python	ALL	5.3.0	4c4cc7e5-1729-4ef6-9748-...
MC-Client-ps_cluster	172.18.0.1	Management Center	ALL	5.5.0-SNAPSHOT	3ab95b20-fe5d-4c08-a839-...
1 - 4 of 4 Rows					

5) Запишемо 10 повідомлень:

```
ory (Outliers)
Windows PowerShell
PS D:\KPI Styding\DistSystems\Lab3> python .\client.py
1)Sent POST-request
2)Sent GET-request
3)Exit
Your choose:1
Message:1
POST відповідь: Success!
1)Sent POST-request
2)Sent GET-request
3)Exit
Your choose:1
Message:2
POST відповідь: Success!
1)Sent POST-request
2)Sent GET-request
3)Exit
Your choose:1
Message:3
POST відповідь: Success!
1)Sent POST-request
2)Sent GET-request
3)Exit
Your choose:1
Message:4
POST відповідь: Success!
1)Sent POST-request
2)Sent GET-request
3)Exit
Your choose:1
Message:5
```

6) Подивимось розподіл даних по нодах:

Map Statistics (In-Memory Format: BINARY)

RESET TIME 1 minute ago → now Default View

Member	Entries	Gets	Puts	Removals	Sets	Entry Memory
192.168.0.107:5701	2	0	2	0	0	258.00 B
192.168.0.107:5702	3	0	3	0	0	387.00 B
192.168.0.107:5703	5	0	5	0	0	646.00 B
TOTAL	10	0	10	0	0	1.26 kB

1 - 3 of 3 Rows 10

Та по кожному з Logging service:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\KPI Styding\DistSystems\Lab3> python .\logging_service.py --hport 5703
* Serving Flask app 'logging_service'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5003
Press CTRL+C to quit
New writing data: 7966 3
127.0.0.1 - - [19/May/2024 12:28:46] "POST /data HTTP/1.1" 200 -
New writing data: 5617 7
127.0.0.1 - - [19/May/2024 12:28:55] "POST /data HTTP/1.1" 200 -
New writing data: 5323 9
127.0.0.1 - - [19/May/2024 12:29:00] "POST /data HTTP/1.1" 200 -
New writing data: 5757 10
127.0.0.1 - - [19/May/2024 12:29:02] "POST /data HTTP/1.1" 200 -

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\KPI Styding\DistSystems\Lab3> python .\logging_service.py --hport 5702
* Serving Flask app 'logging_service'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5002
Press CTRL+C to quit
New writing data: 5955 2
127.0.0.1 - - [19/May/2024 12:28:44] "POST /data HTTP/1.1" 200 -
New writing data: 3624 5
127.0.0.1 - - [19/May/2024 12:28:51] "POST /data HTTP/1.1" 200 -

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\KPI Styding\DistSystems\Lab3> python .\logging_service.py --hport 5701
* Serving Flask app 'logging_service'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5001
Press CTRL+C to quit
New writing data: 2644 1
127.0.0.1 - - [19/May/2024 12:28:38] "POST /data HTTP/1.1" 200 -
New writing data: 7283 4
127.0.0.1 - - [19/May/2024 12:28:49] "POST /data HTTP/1.1" 200 -
New writing data: 3995 6
127.0.0.1 - - [19/May/2024 12:28:54] "POST /data HTTP/1.1" 200 -
New writing data: 5782 8
127.0.0.1 - - [19/May/2024 12:28:58] "POST /data HTTP/1.1" 200 -
```

7) Виконаємо GET-запит:

Його отримав logging-service який працює на 5002 порті:

```
PS D:\KPI Styding\DistSystems\Lab3> python .\logging_service.py --hzport 5702 --logport 5002
* Serving Flask app 'logging_service'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5002
Press CTRL+C to quit
New writing data: 5955 2
127.0.0.1 - - [19/May/2024 12:28:44] "POST /data HTTP/1.1" 200 -
New writing data: 3624 5
127.0.0.1 - - [19/May/2024 12:28:51] "POST /data HTTP/1.1" 200 -
127.0.0.1 - - [19/May/2024 12:35:03] "GET /data HTTP/1.1" 200 -
```

У відповідь отримаємо всі 10 записів:

```
1)Sent POST-request
2)Sent GET-request
3)Exit
Your choose:2
GET відповідь: {'Log data': '6\n8\n1\n3\n4\n7\n2\n10\n5\n9', 'Message data': 'Not implemented yet'}
1)Sent POST-request
```

8) Відключаємо один екземпляр logging service та повторимо Get запит:

```
Your choose:2
GET відповідь: {'Log data': '6\n8\n3\n4\n7\n2\n10\n5\n9\n1', 'Message data': 'Not implemented yet'}
1)Sent POST-request
2)Sent GET-request
3)Exit
Your choose:|
```

Його отримав logging-service який працює на 5001 порті:

```
PS D:\KPI Styding\DistSystems\Lab3> python .\logging_service.py --hzport 5701 --logport 5001
* Serving Flask app 'logging_service'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5001
Press CTRL+C to quit
New writing data: 2644 1
127.0.0.1 - - [19/May/2024 12:28:38] "POST /data HTTP/1.1" 200 -
New writing data: 7283 4
127.0.0.1 - - [19/May/2024 12:28:49] "POST /data HTTP/1.1" 200 -
New writing data: 3995 6
127.0.0.1 - - [19/May/2024 12:28:54] "POST /data HTTP/1.1" 200 -
New writing data: 5782 8
127.0.0.1 - - [19/May/2024 12:28:58] "POST /data HTTP/1.1" 200 -
127.0.0.1 - - [19/May/2024 12:40:36] "GET /data HTTP/1.1" 200 -
```

9) Якщо залишити тільки один екземпляр logging-service (5003 порт) після Get запиту всеодно буде отримано всі дані (відрізняється тільки порядок їх отримання):

```
Your choose:2
GET відповідь: {'Log data': '7\n2\n10\n5\n9\n1\n6\n8\n3\n4', 'Message data': 'Not implemented yet'}
1)Sent POST-request
2)Sent GET-request
3)Exit
Your choose:
```

```
PS D:\KPI Styding\DistSystems\Lab3> python .\logging_service.py --hzport 5703 --logport 5003
* Serving Flask app 'logging_service'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5003
Press CTRL+C to quit
New writing data: 7966 3
127.0.0.1 - - [19/May/2024 12:28:46] "POST /data HTTP/1.1" 200 -
New writing data: 5617 7
127.0.0.1 - - [19/May/2024 12:28:55] "POST /data HTTP/1.1" 200 -
New writing data: 5323 9
127.0.0.1 - - [19/May/2024 12:29:00] "POST /data HTTP/1.1" 200 -
New writing data: 5757 10
127.0.0.1 - - [19/May/2024 12:29:02] "POST /data HTTP/1.1" 200 -
127.0.0.1 - - [19/May/2024 12:45:49] "GET /data HTTP/1.1" 200 -
```