Київський політехнічний інститут імені Ігоря Сікорського

Фізико-технічний інститут

Проектування розподілених систем

Проект

Replicated log task

Виконали:

Студенти групи ФБ-42мп

Алькова Аліна, Легойда Юлія, Осіпчук Антон

Iteration 0.

Choose a desirable language for implementation and try to implement (or find the implementation) a simple *Echo Client-Server* application.

Для цієї частини ми реалізували клієнт-серверну програму мовою Python, використовуючи бібліотеку Flask для серверної частини та requests для клієнтської частини. Програма складається з клієнта і сервера, які взаємодіють через HTTP-запити

Серверна частина (server.py):

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

app = Flask(__name__)

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %
(message)s')
logger = logging.getLogger(__name__)

messages = []

@app.route('/echo', methods=['POST'])
def echo_post():
    data = request.get_json()
    if not data or 'message' not in data:
        logger.error("Invalid request: 'message' field is required")
        return jsonify({"error": "Message is required"}), 400

    message = data['message']
    messages.append(message)
    logger.info(f"Received and stored message: {message}")
    return jsonify({"received": message}), 200

@app.route('/echo', methods=['GET'])
def echo_get():
    logger.info("Returning all messages")
    return jsonify({"messages": messages}), 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

- Сервер приймає HTTP-запити на ендпоінт /echo з методами POST і GET.
- **POST /echo**: Отримує JSON-об'єкт із полем message, зберігає повідомлення в список messages і повертає отримане повідомлення у відповіді з кодом 200. Якщо поле message відсутнє, повертається помилка 400.
- **GET /echo**: Повертає всі збережені повідомлення у вигляді JSON.
- Реалізовано логування за допомогою модуля logging для відстеження отриманих повідомлень та помилок.

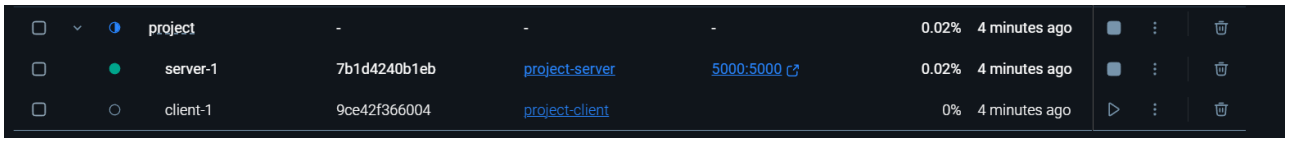
Клієнтська частина (client.py):

```
import requests
import logging

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %
(message)s')
logger = logging.getLogger(__name__)
```

```python
def send_message(message):
    url = 'http://server:5000/echo'
    data = {'message': message}
    try:
        response = requests.post(url, json=data)
        response.raise_for_status()
        logger.info(f"Sent message: {message}, Received: {response.json()}")
    except requests.RequestException as e:
        logger.error(f"Failed to send message: {e}")

def get_messages():
    url = 'http://server:5000/echo'
    try:
        response = requests.get(url)
        response.raise_for_status()
        logger.info(f"Retrieved messages: {response.json()}")
    except requests.RequestException as e:
        logger.error(f"Failed to retrieve messages: {e}")

if __name__ == '__main__':
    send_message("Hello, Echo Server!")
    get_messages()
```

- Функція send_message: Надсилає повідомлення на сервер через POST-запит до /echo.
- Функція get_messages: Отримує всі повідомлення з сервера через GET-запит до /echo.
- Обробка помилок: Використовується try-except для обробки виключень, пов'язаних із мережевими запитами.
- **Логування**: Реалізовано логування для відстеження надісланих і отриманих даних, а також помилок.
- Клієнт автоматично надсилає тестове повідомлення "Hello, Echo Server!" і викликає функцію для отримання повідомлень.

Перевірка запуска:

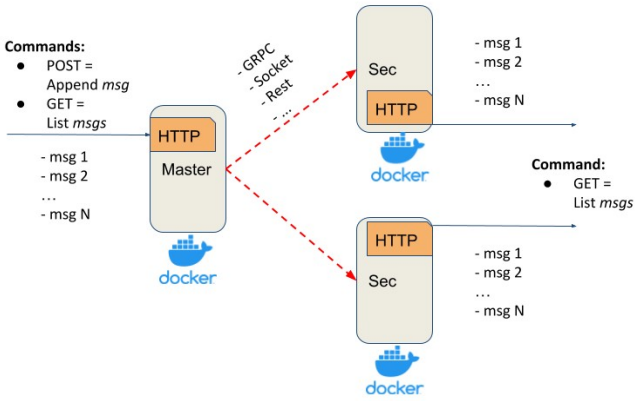



```
[+] Running 5/5
 ✔client                      Built                                    0.0s
 ✔server                      Built                                    0.0s
 ✔Network project_echo-network Created                                 0.1s
 ✔Container project-server-1  Created                                 0.1s
 ✔Container project-client-1  Created                                 0.1s
Attaching to client-1, server-1
server-1  |  * Serving Flask app 'server' (lazy loading)
server-1  |  * Environment: production
server-1  |    WARNING: This is a development server. Do not use it in a production deployment.
server-1  |    Use a production WSGI server instead.
server-1  |  * Debug mode: off
server-1  | 2025-06-02 11:32:59,743 - WARNING -  * Running on all addresses.
server-1  |    WARNING: This is a development server. Do not use it in a production deployment.
server-1  | 2025-06-02 11:32:59,743 - INFO -  * Running on http://172.18.0.2:5000/ (Press CTRL+C to quit)
server-1  | 2025-06-02 11:32:59,888 - INFO - Received and stored message: Hello, Echo Server!
server-1  | 2025-06-02 11:32:59,889 - INFO - 172.18.0.3 - - [02/Jun/2025 11:32:59] "POST /echo HTTP/1.1" 200 -
client-1  | 2025-06-02 11:32:59,890 - INFO - Sent message: Hello, Echo Server!, Received: {'received': 'Hello, Echo Server!'}
server-1  | 2025-06-02 11:32:59,894 - INFO - Returning all messages
client-1  | 2025-06-02 11:32:59,897 - INFO - Retrieved messages: {'messages': ['Hello, Echo Server!']}
server-1  | 2025-06-02 11:32:59,895 - INFO - 172.18.0.3 - - [02/Jun/2025 11:32:59] "GET /echo HTTP/1.1" 200 -
client-1 exited with code 0
```

Iteration 1.

- **5 points**

The Replicated Log should have the following deployment architecture: one ***Master*** and any number of ***Secondaries***.



Master should expose a simple HTTP server (or alternative service with a similar API) with:

- *POST method* - appends a message into the in-memory list
- *GET method* - returns all messages from the in-memory list

Secondary should expose a simple HTTP server(or alternative service with a similar API) with:

- *GET method* - returns all replicated messages from the in-memory list

Properties and assumptions:

- after each POST request, the message should be replicated on every *Secondary* server
- *Master* should ensure that *Secondaries* have received a message via *ACK*
- *Master's POST request* should be finished only after receiving *ACKs* from all *Secondaries* (blocking replication approach)
- to test that the replication is blocking, introduce the delay/sleep on the *Secondary*
- at this stage assume that the communication channel is a perfect link (no failures and messages lost)
- any RPC framework can be used for *Master-Secondary* communication (Sockets, language-specific RPC, HTTP, Rest, gRPC, …)
- your implementation should support logging
- *Master* and *Secondaries* should run in Docker

У цій частині ми налаштували один майстер і два вторинних сервера (secondary1, secondary2) у docker-compose.yml. Архітектура підтримує можливість додавання нових вторинних серверів шляхом зміни конфігурації.

Майстер (master.py):

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

app = Flask(__name__)

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %
(message)s')
logger = logging.getLogger(__name__)

messages = []

SECONDARIES = ['http://secondary1:5001', 'http://secondary2:5001']

@app.route('/messages', methods=['POST'])
```

```
def post_message():
    data = request.get_json()
    if not data or 'message' not in data:
        logger.error("Invalid request: 'message' field is required")
        return jsonify({"error": "Message is required"}), 400

    message = data['message']
    messages.append(message)
    logger.info(f"Stored message: {message}")

    acks = []
    for secondary in SECONDARIES:
        try:
            start_time = time.time()
            response = requests.post(f"{secondary}/replicate", json={'message':
message}, timeout=10)
            response.raise_for_status()
            acks.append(True)
            logger.info(f"Received ACK from {secondary} in {time.time() -
start_time:.2f}s")
        except requests.RequestException as e:
            logger.error(f"Failed to replicate to {secondary}: {e}")
            return jsonify({"error": f"Failed to replicate to {secondary}"}),
500

    if all(acks):
        logger.info("All Secondaries acknowledged")
        return jsonify({"status": "Message replicated"}), 200
    else:
        logger.error("Not all Secondaries acknowledged")
        return jsonify({"error": "Replication failed"}), 500

@app.route('/messages', methods=['GET'])
def get_messages():
    logger.info("Returning all messages")
    return jsonify({"messages": messages}), 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

- Реалізовано HTTP-сервер за допомогою Flask, який працює на порту 5000.
- **Ендпоінти**:
    - **POST /messages**: Приймає JSON-запит із полем message, зберігає повідомлення у локальний список messages і відправляє його на всі вторинні сервери (Secondaries) через POST-запит до їхнього ендпоінта /replicate. Повертає відповідь лише після отримання підтверджень (ACK) від усіх вторинних серверів.
    - **GET /messages**: Повертає всі збережені повідомлення у вигляді JSON.
- Використовується модуль logging для запису інформації про збережені повідомлення, успішні реплікації та помилки.
- **Реплікація**: Майстер відправляє повідомлення на два вторинні сервери (secondary1:5001, secondary2:5001) і чекає підтвердження (ACK) від кожного. Якщо хоча б один сервер не підтверджує, повертається помилка 500.

Вторинні сервери (secondary.py):

```
from flask import Flask, request, jsonify
import logging
import time
```

```
app = Flask(__name__)

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %
(message)s')
logger = logging.getLogger(__name__)

messages = []

@app.route('/replicate', methods=['POST'])
def replicate_message():
    data = request.get_json()
    if not data or 'message' not in data:
        logger.error("Invalid request: 'message' field is required")
        return jsonify({"error": "Message is required"}), 400

    message = data['message']
    time.sleep(2)
    messages.append(message)
    logger.info(f"Replicated message: {message}")
    return jsonify({"status": "ACK"}), 200

@app.route('/messages', methods=['GET'])
def get_messages():
    logger.info("Returning all messages")
    return jsonify({"messages": messages}), 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5001)
```

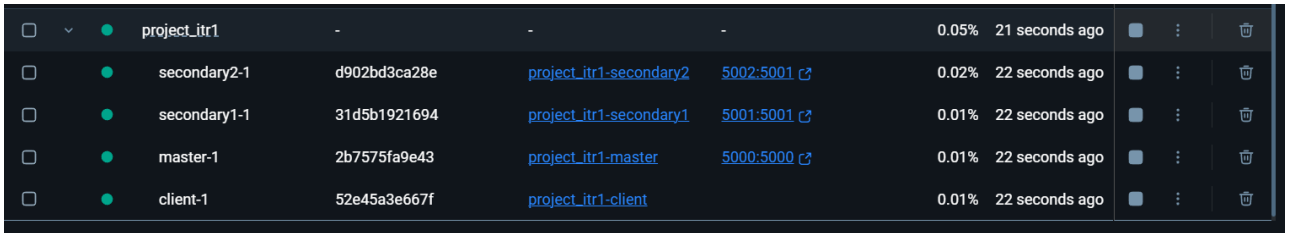
- Реалізовано HTTP-сервер за допомогою Flask, який працює на порту 5001 (для secondary1 і secondary2, хоча для другого порт мапиться як 5002 зовні через Docker Compose).
- **Ендпоінти**:
    - **POST /replicate**: Приймає повідомлення від майстра, додає затримку в 2 секунди (time.sleep(2)), зберігає повідомлення у локальний список messages і повертає підтвердження ({"status": "ACK"}).
    - **GET /messages**: Повертає всі збережені повідомлення.
- Логуються отримані повідомлення та запити.

Клієнт (client.py):

Незважаючи на назву "client", файл фактично дублює код вторинного сервера (secondary.py) і працює як ще один HTTP-сервер на порту 5001. Він має ті ж ендпоінти (/replicate і /messages) і ту ж логіку, що й вторинні сервери. Ми ніяк не використовуємо його повноцінно, ми залишили client.py як резервну копію вторинного сервера. Усі подальші тести ми проводимо вручну але можна було б в цьому клієнті реалізувати автоматичне надсилання потрібних запитів.

Тестування роботи:

```
Attaching to client-1, master-1, secondary1-1, secondary2-1
secondary2-1  |  * Serving Flask app 'secondary' (lazy loading)
secondary2-1  |  * Environment: production
secondary2-1  |    WARNING: This is a development server. Do not use it in a production deployment.
secondary2-1  |    Use a production WSGI server instead.
secondary2-1  |  * Debug mode: off
secondary2-1  | 2025-06-02 12:10:52,566 - WARNING -  * Running on all addresses.
secondary2-1  |    WARNING: This is a development server. Do not use it in a production deployment.
secondary2-1  | 2025-06-02 12:10:52,566 - INFO -  * Running on http://172.18.0.2:5001/ (Press CTRL+C to quit)
secondary1-1  |  * Serving Flask app 'secondary' (lazy loading)
secondary1-1  |  * Environment: production
secondary1-1  |    WARNING: This is a development server. Do not use it in a production deployment.
secondary1-1  |    Use a production WSGI server instead.
secondary1-1  |  * Debug mode: off
secondary1-1  | 2025-06-02 12:10:52,769 - WARNING -  * Running on all addresses.
secondary1-1  |    WARNING: This is a development server. Do not use it in a production deployment.
secondary1-1  | 2025-06-02 12:10:52,769 - INFO -  * Running on http://172.18.0.3:5001/ (Press CTRL+C to quit)
master-1      |  * Serving Flask app 'master' (lazy loading)
master-1      |  * Environment: production
master-1      |    WARNING: This is a development server. Do not use it in a production deployment.
master-1      |    Use a production WSGI server instead.
master-1      |  * Debug mode: off
master-1      | 2025-06-02 12:10:52,837 - WARNING -  * Running on all addresses.
master-1      |    WARNING: This is a development server. Do not use it in a production deployment.
master-1      | 2025-06-02 12:10:52,838 - INFO -  * Running on http://172.18.0.4:5000/ (Press CTRL+C to quit)
client-1      |  * Serving Flask app 'client' (lazy loading)
client-1      |  * Environment: production
client-1      |    WARNING: This is a development server. Do not use it in a production deployment.
client-1      |    Use a production WSGI server instead.
client-1      |  * Debug mode: off
client-1      | 2025-06-02 12:10:53,009 - WARNING -  * Running on all addresses.
client-1      |    WARNING: This is a development server. Do not use it in a production deployment.
client-1      | 2025-06-02 12:10:53,010 - INFO -  * Running on http://172.18.0.5:5001/ (Press CTRL+C to quit)
```



Тестування POST-запиту до Master:

Надсилаємо POST-запит до Master (http://localhost:5000/messages) із повідомленням, щоб перевірити, чи воно зберігається та реплікується на Secondary-сервери.

```
D:\Documents\dist_systems\project_itr1>curl -X POST http://localhost:5000/messages -H "Content-Type: application/json" -d "{\"message\":\"Test message 1\"}"

{"status":"Message replicated"}
```

```
client-1      | 2025-06-02 12:10:53,009 - WARNING -  * Running on all addresses.
client-1      |    WARNING: This is a development server. Do not use it in a production deployment.
client-1      | 2025-06-02 12:10:53,010 - INFO -  * Running on http://172.18.0.5:5001/ (Press CTRL+C to quit)
master-1      | 2025-06-02 12:11:54,246 - INFO - Stored message: Test message 1
secondary1-1  | 2025-06-02 12:11:56,256 - INFO - Replicated message: Test message 1
secondary1-1  | 2025-06-02 12:11:56,257 - INFO - 172.18.0.4 - - [02/Jun/2025 12:11:56] "POST /replicate HTTP/1.1" 200 -
master-1      | 2025-06-02 12:11:56,258 - INFO - Received ACK from http://secondary1:5001 in 2.01s
secondary2-1  | 2025-06-02 12:11:58,265 - INFO - Replicated message: Test message 1
secondary2-1  | 2025-06-02 12:11:58,267 - INFO - 172.18.0.4 - - [02/Jun/2025 12:11:58] "POST /replicate HTTP/1.1" 200 -
master-1      | 2025-06-02 12:11:58,268 - INFO - Received ACK from http://secondary2:5001 in 2.01s
master-1      | 2025-06-02 12:11:58,268 - INFO - All Secondaries acknowledged
master-1      | 2025-06-02 12:11:58,269 - INFO - 172.18.0.1 - - [02/Jun/2025 12:11:58] "POST /messages HTTP/1.1" 200 -
```

Перевірка блокуючої реплікації:

```
PS D:\Documents\dist_systems\project_itr1> Measure-Command { Invoke-RestMethod -Uri http://localhost:5000/messages -Method Post -Headers @{"Content-Type"="application/json"} -Body '{"message":"Time Test message"}' }


Days              : 0
Hours             : 0
Minutes           : 0
Seconds           : 4
Milliseconds      : 208
Ticks             : 42083061
TotalDays         : 4,87072465277778E-05
TotalHours        : 0,00116897391666667
TotalMinutes      : 0,070138435
TotalSeconds      : 4,2083061
TotalMilliseconds : 4208,3061
```

Вимірювання часу виконання POST-запиту підтверджує, що Master чекає на ACK від усіх Secondary, і затримка (2 секунди на кожен Secondary) впливає на загальний час. Це

демонструє сильну консистентність: повідомлення не вважається збереженим, доки всі сервери його не підтвердять.

Перевірка GET-запитів:

```
D:\Documents\dist_systems\project_itr1>curl http://localhost:5000/messages
{"messages":["Test message 1","Time Test message"]}

D:\Documents\dist_systems\project_itr1>curl http://localhost:5001/messages
{"messages":["Test message 1","Time Test message"]}

D:\Documents\dist_systems\project_itr1>curl http://localhost:5002/messages
{"messages":["Test message 1","Time Test message"]}

D:\Documents\dist_systems\project_itr1>
```

Запити до Master і Secondary перевіряють, що повідомлення коректно зберігаються і синхронізуються між усіма серверами. Це підтверджує, що реплікація працює правильно.

Перевірка логів:

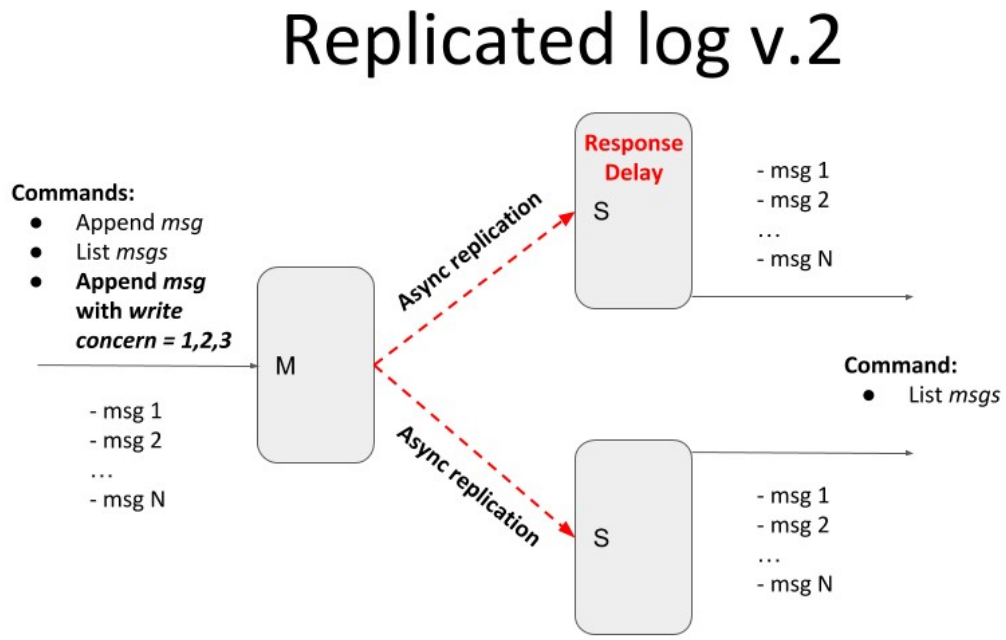
Логи містять інформацію про збереження повідомлень, реплікацію та ACK.

```
client-1      |    WARNING: This is a development server. Do not use it in a production deployment.
client-1      | 2025-06-02 12:10:53,010 - INFO -  * Running on http://172.18.0.5:5001/ (Press CTRL+C to quit)
master-1      | 2025-06-02 12:11:54,246 - INFO - Stored message: Test message 1
secondary1-1  | 2025-06-02 12:11:56,256 - INFO - Replicated message: Test message 1
secondary1-1  | 2025-06-02 12:11:56,257 - INFO - 172.18.0.4 - - [02/Jun/2025 12:11:56] "POST /replicate HTTP/1.1" 200 -
master-1      | 2025-06-02 12:11:56,258 - INFO - Received ACK from http://secondary1:5001 in 2.01s
secondary2-1  | 2025-06-02 12:11:58,265 - INFO - Replicated message: Test message 1
secondary2-1  | 2025-06-02 12:11:58,267 - INFO - 172.18.0.4 - - [02/Jun/2025 12:11:58] "POST /replicate HTTP/1.1" 200 -
master-1      | 2025-06-02 12:11:58,268 - INFO - Received ACK from http://secondary2:5001 in 2.01s
master-1      | 2025-06-02 12:11:58,268 - INFO - All Secondaries acknowledged
master-1      | 2025-06-02 12:11:58,269 - INFO - 172.18.0.1 - - [02/Jun/2025 12:11:58] "POST /messages HTTP/1.1" 200 -
master-1      | 2025-06-02 12:16:37,343 - INFO - Stored message: Time Test message
secondary1-1  | 2025-06-02 12:16:39,352 - INFO - Replicated message: Time Test message
secondary1-1  | 2025-06-02 12:16:39,352 - INFO - 172.18.0.4 - - [02/Jun/2025 12:16:39] "POST /replicate HTTP/1.1" 200 -
master-1      | 2025-06-02 12:16:39,354 - INFO - Received ACK from http://secondary1:5001 in 2.01s
secondary2-1  | 2025-06-02 12:16:41,361 - INFO - Replicated message: Time Test message
secondary2-1  | 2025-06-02 12:16:41,362 - INFO - 172.18.0.4 - - [02/Jun/2025 12:16:41] "POST /replicate HTTP/1.1" 200 -
master-1      | 2025-06-02 12:16:41,363 - INFO - Received ACK from http://secondary2:5001 in 2.01s
master-1      | 2025-06-02 12:16:41,363 - INFO - All Secondaries acknowledged
master-1      | 2025-06-02 12:16:41,364 - INFO - 172.18.0.1 - - [02/Jun/2025 12:16:41] "POST /messages HTTP/1.1" 200 -
master-1      | 2025-06-02 12:18:26,987 - INFO - Returning all messages
master-1      | 2025-06-02 12:18:26,988 - INFO - 172.18.0.1 - - [02/Jun/2025 12:18:26] "GET /messages HTTP/1.1" 200 -
secondary1-1  | 2025-06-02 12:18:35,415 - INFO - Returning all messages
secondary1-1  | 2025-06-02 12:18:35,415 - INFO - 172.18.0.1 - - [02/Jun/2025 12:18:35] "GET /messages HTTP/1.1" 200 -
secondary2-1  | 2025-06-02 12:18:40,462 - INFO - Returning all messages
secondary2-1  | 2025-06-02 12:18:40,462 - INFO - 172.18.0.1 - - [02/Jun/2025 12:18:40] "GET /messages HTTP/1.1" 200 -
```

Iteration 2.

- **5 points**

In the previous iteration, the replication was blocking for all secondaries, i.e. to return a response to the client we should receive acknowledgements (ACK) from all secondaries.



Current iteration should provide tunable semi-synchronicity for replication, by defining *write concern* parameters.

- client POST request in addition to the message should also contain *write concern* parameter *w=1,2,3,..,n*
- *w* value specifies how many ACKs the master should receive from secondaries before responding to the client
  w = 1 - only from master
  w = 2 - from master and one secondary
  w = 3 - from master and two secondaries

Please emulate the replica's inconsistency (and eventual consistency) with the master by introducing the artificial delay on the secondary node. In this case, the master and secondary should temporarily return different lists of messages.
Add logic for messages deduplication and to guarantee the total ordering of messages.

1. Оновлення master.py Додаємо підтримку параметра w, асинхронну реплікацію для демонстрації неконсистентності, дедуплікацію (за допомогою унікального ID повідомлення) і тотальний порядок:

```
from flask import Flask, request, jsonify
import aiohttp
import asyncio
import logging
import time
import uuid
import threading

app = Flask(__name__)
```

```python
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %
(message)s')
logger = logging.getLogger(__name__)

messages = []

SECONDARIES = ['http://secondary1:5001', 'http://secondary2:5001']

async def replicate_to_secondary(secondary, message_data, timeout=15):
    try:
        start_time = time.time()
        logger.info(f"Attempting to replicate to {secondary}")
        async with aiohttp.ClientSession() as session:
            async with session.post(f"{secondary}/replicate", json=message_data,
timeout=timeout) as response:
                response.raise_for_status()
                logger.info(f"Received ACK from {secondary} in {time.time() -
start_time:.2f}s")
                return True
    except Exception as e:
        logger.error(f"Failed to replicate to {secondary}: {e}")
        return False

def run_async_tasks_in_background(tasks):
    loop = asyncio.new_event_loop()
    asyncio.set_event_loop(loop)
    try:
        loop.run_until_complete(asyncio.gather(*tasks))
    finally:
        loop.close()

@app.route('/messages', methods=['POST'])
def post_message():
    data = request.get_json()
    if not data or 'message' not in data or 'w' not in data:
        logger.error("Invalid request: 'message' and 'w' fields are required")
        return jsonify({"error": "Message and write concern (w) are required"}),
400

    message = data['message']
    w = data['w']
    message_id = str(uuid.uuid4())

    if not isinstance(w, int) or w < 1 or w > len(SECONDARIES) + 1:
        logger.error(f"Invalid write concern: w={w}, must be between 1 and
{len(SECONDARIES) + 1}")
        return jsonify({"error": f"Write concern must be between 1 and
{len(SECONDARIES) + 1}"}), 400

    message_entry = {"id": message_id, "message": message, "order": len(messages)}
    if any(m["id"] == message_id for m in messages):
        logger.warning(f"Duplicate message ID {message_id}, ignoring")
        return jsonify({"status": "Message already exists"}), 200
```

```
    messages.append(message_entry)
    logger.info(f"Stored message: {message} with ID {message_id}")

    tasks = [replicate_to_secondary(secondary, {"id": message_id, "message":
message, "order": message_entry["order"]}) for secondary in SECONDARIES]

    if w == 1:
        threading.Thread(target=run_async_tasks_in_background, args=(tasks,),
daemon=True).start()
        logger.info(f"Received 1 ACKs, satisfying w={w}")
        return jsonify({"status": "Message replicated", "message_id": message_id}),
200

    async def replicate_with_concern():
        acks = [True]
        required_acks = w - 1

        if w == len(SECONDARIES) + 1:
            results = await asyncio.gather(*tasks, return_exceptions=True)
            for result in results:
                acks.append(result if isinstance(result, bool) and result else
False)
            return acks

        for task in asyncio.as_completed(tasks, timeout=15):
            try:
                result = await task
                acks.append(result if isinstance(result, bool) and result else
False)
                if sum(acks) >= w:
                    return acks
            except Exception as e:
                logger.error(f"Task failed: {e}")
                acks.append(False)
                if sum(acks) >= w:
                    return acks
        return acks

    loop = asyncio.new_event_loop()
    asyncio.set_event_loop(loop)
    try:
        acks = loop.run_until_complete(replicate_with_concern())
    finally:
        loop.close()

    if sum(acks) >= w:
        logger.info(f"Received {sum(acks)} ACKs, satisfying w={w}")
        return jsonify({"status": "Message replicated", "message_id": message_id}),
200
    else:
        logger.error(f"Not enough ACKs: received {sum(acks)}, required w={w}")
        return jsonify({"error": "Not enough ACKs received"}), 500
```

```
@app.route('/messages', methods=['GET'])
def get_messages():
    logger.info("Returning all messages")
    sorted_messages = sorted(messages, key=lambda x: x["order"])
    return jsonify({"messages": [m["message"] for m in sorted_messages]}), 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

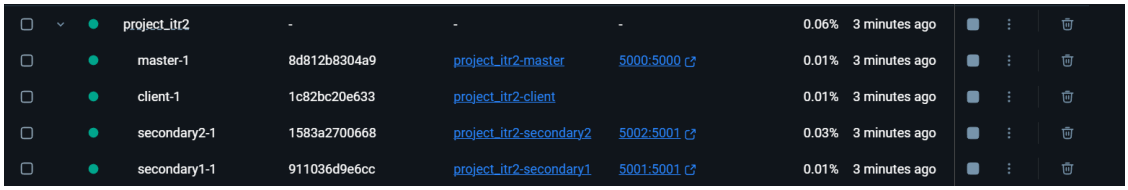
2. Оновлення secondary.py Додаємо підтримку дедуплікації та тотального порядку, а також затримку для емуляції неконсистентності:

```
from flask import Flask, request, jsonify
import logging
import time

app = Flask(__name__)

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %
(message)s')
logger = logging.getLogger(__name__)

messages = []

@app.route('/replicate', methods=['POST'])
def replicate_message():
    data = request.get_json()
    if not data or 'id' not in data or 'message' not in data or 'order' not in
data:
        logger.error("Invalid request: 'id', 'message', and 'order' fields are
required")
        return jsonify({"error": "ID, message, and order are required"}), 400

    message_id = data['id']
    message = data['message']
    order = data['order']

    if any(m["id"] == message_id for m in messages):
        logger.warning(f"Duplicate message ID {message_id}, ignoring")
        return jsonify({"status": "ACK"}), 200

    time.sleep(5)
    messages.append({"id": message_id, "message": message, "order": order})
    logger.info(f"Replicated message: {message} with ID {message_id}")
    return jsonify({"status": "ACK"}), 200

@app.route('/messages', methods=['GET'])
def get_messages():
    logger.info("Returning all messages")
    sorted_messages = sorted(messages, key=lambda x: x["order"])
```

```
    return jsonify({"messages": [m["message"] for m in sorted_messages]}), 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5001)
```

3. Оновлення client.py. Оскільки client.py виконує роль Secondary, оновлюємо його аналогічно до secondary.py:

```
from flask import Flask, request, jsonify
import logging
import time

app = Flask(__name__)

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %
(message)s')
logger = logging.getLogger(__name__)

messages = []

@app.route('/replicate', methods=['POST'])
def replicate_message():
    data = request.get_json()
    if not data or 'id' not in data or 'message' not in data or 'order' not in
data:
        logger.error("Invalid request: 'id', 'message', and 'order' fields are
required")
        return jsonify({"error": "ID, message, and order are required"}), 400

    message_id = data['id']
    message = data['message']
    order = data['order']

    if any(m["id"] == message_id for m in messages):
        logger.warning(f"Duplicate message ID {message_id}, ignoring")
        return jsonify({"status": "ACK"}), 200

    time.sleep(8)
    messages.append({"id": message_id, "message": message, "order": order})
    logger.info(f"Replicated message: {message} with ID {message_id}")
    return jsonify({"status": "ACK"}), 200

@app.route('/messages', methods=['GET'])
def get_messages():
    logger.info("Returning all messages")
    sorted_messages = sorted(messages, key=lambda x: x["order"])
    return jsonify({"messages": [m["message"] for m in sorted_messages]}), 200

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5001)
```



project_itr2	-	-	-	0.06%	3 minutes ago	
master-1	8d812b8304a9	project_itr2-master	5000:5000	0.01%	3 minutes ago	
client-1	1c82bc20e633	project_itr2-client		0.01%	3 minutes ago	
secondary2-1	1583a2700668	project_itr2-secondary2	5002:5001	0.03%	3 minutes ago	
secondary1-1	911036d9e6cc	project_itr2-secondary1	5001:5001	0.01%	3 minutes ago	

```
Attaching to client-1, master-1, secondary1-1, secondary2-1
master-1       |  * Serving Flask app 'master' (lazy loading)
master-1       |  * Environment: production
master-1       |    WARNING: This is a development server. Do not use it in a production deployment.
master-1       |    Use a production WSGI server instead.
master-1       |  * Debug mode: off
master-1       | 2025-06-02 12:57:23,449 - WARNING -  * Running on all addresses.
master-1       |    WARNING: This is a development server. Do not use it in a production deployment.
master-1       | 2025-06-02 12:57:23,449 - INFO -  * Running on http://172.18.0.2:5000/ (Press CTRL+C to quit)
secondary2-1   |  * Serving Flask app 'secondary' (lazy loading)
secondary2-1   |  * Environment: production
secondary2-1   |    WARNING: This is a development server. Do not use it in a production deployment.
secondary1-1   |  * Serving Flask app 'secondary' (lazy loading)
secondary2-1   |    Use a production WSGI server instead.
secondary1-1   |  * Environment: production
secondary2-1   |  * Debug mode: off
secondary1-1   |    WARNING: This is a development server. Do not use it in a production deployment.
secondary2-1   | 2025-06-02 12:57:23,531 - WARNING -  * Running on all addresses.
secondary1-1   |    Use a production WSGI server instead.
secondary2-1   |    WARNING: This is a development server. Do not use it in a production deployment.
secondary1-1   |  * Debug mode: off
secondary2-1   | 2025-06-02 12:57:23,532 - INFO -  * Running on http://172.18.0.4:5001/ (Press CTRL+C to quit)
secondary1-1   | 2025-06-02 12:57:23,543 - WARNING -  * Running on all addresses.
secondary1-1   |    WARNING: This is a development server. Do not use it in a production deployment.
secondary1-1   | 2025-06-02 12:57:23,543 - INFO -  * Running on http://172.18.0.3:5001/ (Press CTRL+C to quit)
client-1       |  * Serving Flask app 'client' (lazy loading)
client-1       |  * Environment: production
client-1       |    WARNING: This is a development server. Do not use it in a production deployment.
client-1       |    Use a production WSGI server instead.
client-1       |  * Debug mode: off
client-1       | 2025-06-02 12:57:23,781 - WARNING -  * Running on all addresses.
client-1       |    WARNING: This is a development server. Do not use it in a production deployment.
client-1       | 2025-06-02 12:57:23,782 - INFO -  * Running on http://172.18.0.5:5001/ (Press CTRL+C to quit)
```

Тестування POST-запиту з різними значеннями w:

W=1

Демонструючи неконсистентність і остаточну консистентність

```
PS D:\Documents\dist_systems\project_itr2> .\test.ps1
Testing w=1
POST to Master (w=1):

message_id                           status
----------                           ------
0f082857-b904-4b27-8598-fbecec03148c Message replicated


GET from Master:

messages
--------
{Test message 1}


GET from Secondary1:

messages
--------
{}


GET from Secondary2:

messages
--------
{}


Waiting 10 seconds...
GET from Secondary1 after delay:

messages
--------
{Test message 1}


GET from Secondary2 after delay:

messages
--------
{Test message 1}
```

W=2

```
Testing w=2
POST to Master (w=2):

message_id                           status
----------                           ------
d3c47682-6f0c-437a-9149-be3e3daea84b Message replicated


GET from Master:

messages
--------
{Test message 1, Test message 2}


GET from Secondary1:

messages
--------
{Test message 1, Test message 2}


GET from Secondary2:

messages
--------
{Test message 1, Test message 2}
```

W=3

```
Testing w=3
POST to Master (w=3):

message_id                           status
----------                           ------
9a1a47d3-d971-4d78-9550-921715c4535b Message replicated


GET from Master:

messages
--------
{Test message 1, Test message 2, Test message 3}


GET from Secondary1:

messages
--------
{Test message 1, Test message 2, Test message 3}


GET from Secondary2:

messages
--------
{Test message 1, Test message 2, Test message 3}
```

```
client-1      | 2025-06-02 16:23:04,493 - WARNING -  * Running on all addresses.
client-1      |    WARNING: This is a development server. Do not use it in a production deployment.
client-1      | 2025-06-02 16:23:04,493 - INFO -  * Running on http://172.18.0.5:5001/ (Press CTRL+C to quit)
master-1      | 2025-06-02 16:24:20,128 - INFO - Stored message: Test message 1 with ID 0f082857-b904-4b27-8598-fbecec03148c
master-1      | 2025-06-02 16:24:20,129 - INFO - Attempting to replicate to http://secondary1:5001
master-1      | 2025-06-02 16:24:20,129 - INFO - Received 1 ACKs, satisfying w=1
master-1      | 2025-06-02 16:24:20,130 - INFO - Attempting to replicate to http://secondary2:5001
master-1      | 2025-06-02 16:24:20,131 - INFO - 172.18.0.1 - - [02/Jun/2025 16:24:20] "POST /messages HTTP/1.1" 200 -
master-1      | 2025-06-02 16:24:20,253 - INFO - Returning all messages
master-1      | 2025-06-02 16:24:20,253 - INFO - 172.18.0.1 - - [02/Jun/2025 16:24:20] "GET /messages HTTP/1.1" 200 -
secondary1-1  | 2025-06-02 16:24:20,261 - INFO - Returning all messages
secondary1-1  | 2025-06-02 16:24:20,262 - INFO - 172.18.0.1 - - [02/Jun/2025 16:24:20] "GET /messages HTTP/1.1" 200 -
secondary2-1  | 2025-06-02 16:24:20,270 - INFO - Returning all messages
secondary2-1  | 2025-06-02 16:24:20,271 - INFO - 172.18.0.1 - - [02/Jun/2025 16:24:20] "GET /messages HTTP/1.1" 200 -
secondary2-1  | 2025-06-02 16:24:25,157 - INFO - Replicated message: Test message 1 with ID 0f082857-b904-4b27-8598-fbecec03148c
secondary1-1  | 2025-06-02 16:24:25,156 - INFO - Replicated message: Test message 1 with ID 0f082857-b904-4b27-8598-fbecec03148c
secondary2-1  | 2025-06-02 16:24:25,157 - INFO - 172.18.0.4 - - [02/Jun/2025 16:24:25] "POST /replicate HTTP/1.1" 200 -
master-1      | 2025-06-02 16:24:25,158 - INFO - Received ACK from http://secondary1:5001 in 5.03s
secondary1-1  | 2025-06-02 16:24:25,157 - INFO - 172.18.0.4 - - [02/Jun/2025 16:24:25] "POST /replicate HTTP/1.1" 200 -
master-1      | 2025-06-02 16:24:25,159 - INFO - Received ACK from http://secondary2:5001 in 5.03s
secondary1-1  | 2025-06-02 16:24:30,289 - INFO - Returning all messages
secondary1-1  | 2025-06-02 16:24:30,290 - INFO - 172.18.0.1 - - [02/Jun/2025 16:24:30] "GET /messages HTTP/1.1" 200 -
secondary2-1  | 2025-06-02 16:24:30,297 - INFO - Returning all messages
secondary2-1  | 2025-06-02 16:24:30,297 - INFO - 172.18.0.1 - - [02/Jun/2025 16:24:30] "GET /messages HTTP/1.1" 200 -
master-1      | 2025-06-02 16:24:30,305 - INFO - Stored message: Test message 2 with ID d3c4768Z-6f0c-437a-9149-be3e3daea84b
master-1      | 2025-06-02 16:24:30,306 - INFO - Attempting to replicate to http://secondary2:5001
master-1      | 2025-06-02 16:24:30,307 - INFO - Attempting to replicate to http://secondary1:5001
secondary2-1  | 2025-06-02 16:24:35,311 - INFO - Replicated message: Test message 2 with ID d3c47682-6f0c-437a-9149-be3e3daea84b
secondary2-1  | 2025-06-02 16:24:35,311 - INFO - 172.18.0.4 - - [02/Jun/2025 16:24:35] "POST /replicate HTTP/1.1" 200 -
master-1      | 2025-06-02 16:24:35,312 - INFO - Received ACK from http://secondary2:5001 in 5.01s
secondary1-1  | 2025-06-02 16:24:35,314 - INFO - Replicated message: Test message 2 with ID d3c47682-6f0c-437a-9149-be3e3daea84b
master-1      | 2025-06-02 16:24:35,313 - INFO - Received 2 ACKs, satisfying w=2
secondary1-1  | 2025-06-02 16:24:35,315 - INFO - 172.18.0.4 - - [02/Jun/2025 16:24:35] "POST /replicate HTTP/1.1" 200 -
master-1      | 2025-06-02 16:24:35,314 - INFO - 172.18.0.1 - - [02/Jun/2025 16:24:35] "POST /messages HTTP/1.1" 200 -
master-1      | 2025-06-02 16:24:35,437 - INFO - Returning all messages
master-1      | 2025-06-02 16:24:35,437 - INFO - 172.18.0.1 - - [02/Jun/2025 16:24:35] "GET /messages HTTP/1.1" 200 -
secondary1-1  | 2025-06-02 16:24:35,445 - INFO - Returning all messages
secondary1-1  | 2025-06-02 16:24:35,446 - INFO - 172.18.0.1 - - [02/Jun/2025 16:24:35] "GET /messages HTTP/1.1" 200 -
secondary2-1  | 2025-06-02 16:24:35,454 - INFO - Returning all messages
master-1      | 2025-06-02 16:24:35,464 - INFO - Stored message: Test message 3 with ID 9a1a47d3-d971-4d78-9550-921715c4535b
secondary2-1  | 2025-06-02 16:24:35,454 - INFO - 172.18.0.1 - - [02/Jun/2025 16:24:35] "GET /messages HTTP/1.1" 200 -
master-1      | 2025-06-02 16:24:35,464 - INFO - Attempting to replicate to http://secondary1:5001
master-1      | 2025-06-02 16:24:35,465 - INFO - Attempting to replicate to http://secondary2:5001
secondary2-1  | 2025-06-02 16:24:40,471 - INFO - Replicated message: Test message 3 with ID 9a1a47d3-d971-4d78-9550-921715c4535b
secondary2-1  | 2025-06-02 16:24:40,471 - INFO - 172.18.0.4 - - [02/Jun/2025 16:24:40] "POST /replicate HTTP/1.1" 200 -
master-1      | 2025-06-02 16:24:40,472 - INFO - Received ACK from http://secondary2:5001 in 5.01s
secondary1-1  | 2025-06-02 16:24:40,475 - INFO - Replicated message: Test message 3 with ID 9a1a47d3-d971-4d78-9550-921715c4535b
master-1      | 2025-06-02 16:24:40,476 - INFO - Received ACK from http://secondary1:5001 in 5.01s
secondary1-1  | 2025-06-02 16:24:40,476 - INFO - 172.18.0.4 - - [02/Jun/2025 16:24:40] "POST /replicate HTTP/1.1" 200 -
master-1      | 2025-06-02 16:24:40,477 - INFO - Received 3 ACKs, satisfying w=3
master-1      | 2025-06-02 16:24:40,477 - INFO - 172.18.0.1 - - [02/Jun/2025 16:24:40] "POST /messages HTTP/1.1" 200 -
master-1      | 2025-06-02 16:24:40,587 - INFO - Returning all messages
master-1      | 2025-06-02 16:24:40,587 - INFO - 172.18.0.1 - - [02/Jun/2025 16:24:40] "GET /messages HTTP/1.1" 200 -
secondary1-1  | 2025-06-02 16:24:40,594 - INFO - Returning all messages
secondary1-1  | 2025-06-02 16:24:40,595 - INFO - 172.18.0.1 - - [02/Jun/2025 16:24:40] "GET /messages HTTP/1.1" 200 -
secondary2-1  | 2025-06-02 16:24:40,602 - INFO - Returning all messages
secondary2-1  | 2025-06-02 16:24:40,603 - INFO - 172.18.0.1 - - [02/Jun/2025 16:24:40] "GET /messages HTTP/1.1" 200 -
```

Тестування дедуплікації

Тепер ми можемо надіслати два POST-запити з однаковим id, щоб перевірити, що Master виявляє дублювання і не додає повідомлення вдруге.

```
PS D:\Documents\dist_systems\project_itr2> Invoke-RestMethod -Uri http://localhost:5000/messages -Method Post -Headers @{"Content-Type"="application/json"}
-Body '{"message":"Test message 4", "w":1, "id":"test-id-123"}'

message_id  status
----------  ------
test-id-123 Message replicated


PS D:\Documents\dist_systems\project_itr2> Invoke-RestMethod -Uri http://localhost:5000/messages -Method Post -Headers @{"Content-Type"="application/json"}
-Body '{"message":"Test message 4", "w":1, "id":"test-id-123"}'

message_id  status
----------  ------
test-id-123 Message already exists
```

Виконали GET-запити, щоб переконатися, що повідомлення не дублюються:

```
PS D:\Documents\dist_systems\project_itr2> Invoke-RestMethod -Uri http://localhost:5000/messages -Method Get

messages
--------
{Test message 4}


PS D:\Documents\dist_systems\project_itr2> Invoke-RestMethod -Uri http://localhost:5001/messages -Method Get

messages
--------
{Test message 4}


PS D:\Documents\dist_systems\project_itr2> Invoke-RestMethod -Uri http://localhost:5002/messages -Method Get

messages
--------
{Test message 4}
```

```
client-1      | 2025-06-02 13:25:57,689 - WARNING -  * Running on all addresses.
client-1      |    WARNING: This is a development server. Do not use it in a production deployment.
client-1      | 2025-06-02 13:25:57,689 - INFO -  * Running on http://172.18.0.5:5001/ (Press CTRL+C to quit)
master-1      | 2025-06-02 13:26:15,723 - INFO - Stored message: Test message 4 with ID test-id-123
secondary1-1  | 2025-06-02 13:26:17,731 - INFO - Replicated message: Test message 4 with ID test-id-123
secondary1-1  | 2025-06-02 13:26:17,732 - INFO - 172.18.0.3 - - [02/Jun/2025 13:26:17] "POST /replicate HTTP/1.1" 200 -
master-1      | 2025-06-02 13:26:17,733 - INFO - Received ACK from http://secondary1:5001 in 2.01s
secondary2-1  | 2025-06-02 13:26:19,739 - INFO - Replicated message: Test message 4 with ID test-id-123
secondary2-1  | 2025-06-02 13:26:19,740 - INFO - 172.18.0.3 - - [02/Jun/2025 13:26:19] "POST /replicate HTTP/1.1" 200 -
master-1      | 2025-06-02 13:26:19,742 - INFO - Received ACK from http://secondary2:5001 in 2.01s
master-1      | 2025-06-02 13:26:19,742 - INFO - Received 3 ACKs, satisfying w=1
master-1      | 2025-06-02 13:26:19,743 - INFO - 172.18.0.1 - - [02/Jun/2025 13:26:19] "POST /messages HTTP/1.1" 200 -
master-1      | 2025-06-02 13:26:34,112 - WARNING - Duplicate message ID test-id-123, ignoring
master-1      | 2025-06-02 13:26:34,113 - INFO - 172.18.0.1 - - [02/Jun/2025 13:26:34] "POST /messages HTTP/1.1" 200 -
master-1      | 2025-06-02 13:26:45,825 - INFO - Returning all messages
master-1      | 2025-06-02 13:26:45,826 - INFO - 172.18.0.1 - - [02/Jun/2025 13:26:45] "GET /messages HTTP/1.1" 200 -
secondary1-1  | 2025-06-02 13:26:50,908 - INFO - Returning all messages
secondary1-1  | 2025-06-02 13:26:50,908 - INFO - 172.18.0.1 - - [02/Jun/2025 13:26:50] "GET /messages HTTP/1.1" 200 -
secondary2-1  | 2025-06-02 13:26:56,011 - INFO - Returning all messages
secondary2-1  | 2025-06-02 13:26:56,012 - INFO - 172.18.0.1 - - [02/Jun/2025 13:26:56] "GET /messages HTTP/1.1" 200 -
```

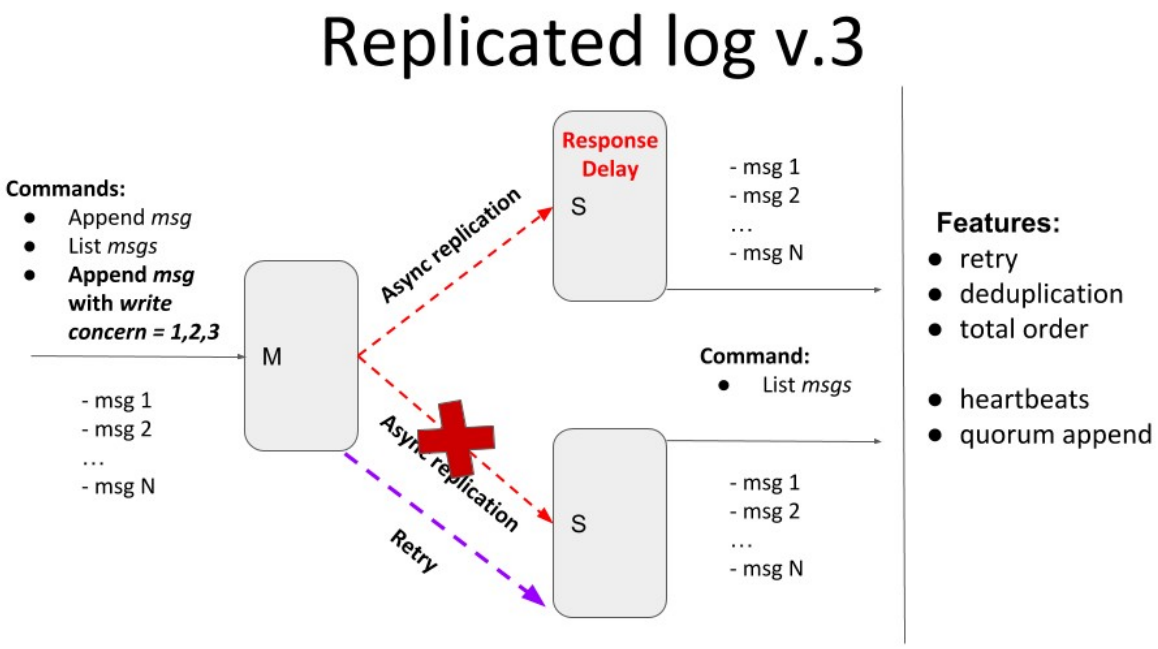
Тестування тотального порядку:

```
PS D:\Documents\dist_systems\project_itr2> Invoke-RestMethod -Uri http://localhost:5000/messages -Method Post -Headers @{"Content-Type"="application/json"} -Body '{"message":"Test message
4", "w":3}'

message_id                           status
----------                           ------
fa8ffad1-b685-447b-abef-37adca90c095 Message replicated


PS D:\Documents\dist_systems\project_itr2> Invoke-RestMethod -Uri http://localhost:5000/messages -Method Post -Headers @{"Content-Type"="application/json"} -Body '{"message":"Test message
5", "w":3}'

message_id                           status
----------                           ------
8573b194-55d2-427e-a228-7663fe3bf703 Message replicated


PS D:\Documents\dist_systems\project_itr2> Invoke-RestMethod -Uri http://localhost:5001/messages -Method Get

messages
--------
{Test message 1, Test message 2, Test message 3, Test message 4...}


PS D:\Documents\dist_systems\project_itr2> Invoke-RestMethod -Uri http://localhost:5002/messages -Method Get

messages
--------
{Test message 1, Test message 2, Test message 3, Test message 4...}


PS D:\Documents\dist_systems\project_itr2>
```

Отож, було виконано напівсинхронну реплікацію з параметром w, емуляцію неконсистентності, дедуплікацію повідомлень також організували гарантію тотального порядку: поле order у повідомленнях і сортування в GET- запитах забезпечують одинаковий порядок.

Iteration 3.

- **15 points**



The current iteration should provide tunable semi-synchronicity for replication with a *retry* mechanism that should deliver all messages *exactly-once* in total order.

Main features:

- If message delivery fails (due to connection, or internal server error, or secondary is unavailable) the delivery attempts should be repeated - *retry*
    - If one of the secondaries is down and *w=3*, the client should be blocked until the node becomes available. Clients running in parallel shouldn't be blocked by the blocked one.
    - If *w>1* the client should be blocked until the message will be delivered to all secondaries required by the write concern level. Clients running in parallel shouldn't be blocked by the blocked one.
    - All messages that secondaries have missed due to unavailability should be replicated after (re)joining the master
    - Retries can be implemented with an unlimited number of attempts but, possibly, with some "smart" delays logic
    - You can specify a *timeout* for the master in case if there is no response from the secondary
- All messages should be present exactly once in the secondary log - *deduplication*
    - To test deduplication you can generate some random internal server error response from the secondary after the message has been added to the log
- The order of messages should be the same in all nodes - *total order*
    - If secondary has received messages *[msg1, msg2, msg4]*, it shouldn't display the message *'msg4'* until the *'msg3'* will be received
    - To test the total order, you can generate some random internal server error response from the secondaries

У цій роботі реалізовано семі-синхронну реплікацію повідомлень між одним Master-вузлом і двома Secondary-вузлами. Ключові функції:

- Write-concern (w) для блокування відповіді клієнта до отримання потрібної кількості ACK.
- Retry-механізм із нескінченними повторними спробами доставки до вторинок.
- Дедуплікація у Secondary, щоб жодне повідомлення не з'явилося двічі.
- Total-order через буферизацію «out-of-order» повідомлень до отримання пропущених.
- Initial-sync у Secondary після запуску, щоб підтягнути всі пропущені записи із Master.

1. master.py

Основні нові елементи в master.py:

- Retry-механізм – якщо під час синхронної реплікації якийсь сервер недоступна або повернув помилку, цикл while True в sync_replicate_to_secondary починає нескінченні спроби з паузою 2 сек. Поки останній сервер не «підніметься», master не відправить клієнту 200.

- Метод /full_messages – віддає увесь масив повідомлень із полями id, message, order. Це потрібно для механізму initial sync у вторинних вузлах.

- Метод /clean – потрібен для коректного функціонування автоматичних тестів, цей виклик очистить messages.

2. secondary.py

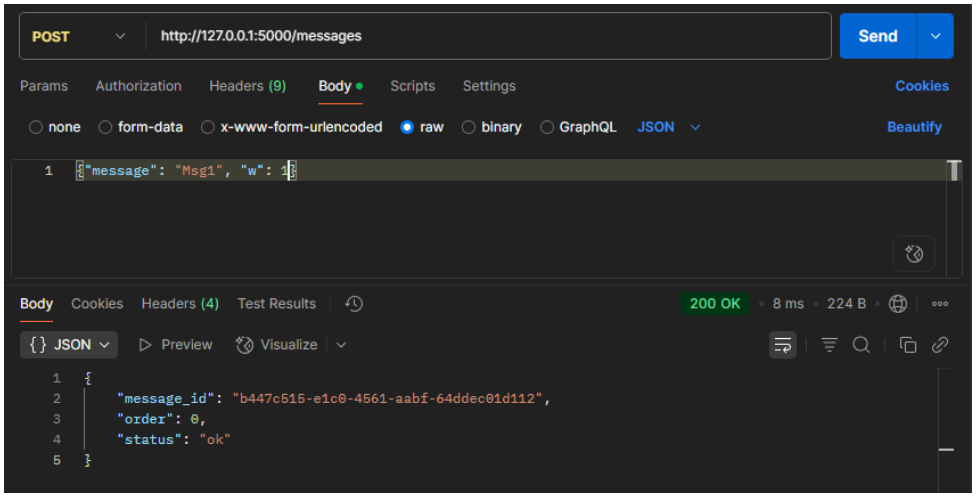
- Initial Sync
-- При старті сервіс робить GET /full_messages до master.
-- Отримує масив { id, message, order }, який сортує за order і додає в локальний список messages.
-- Після цього expected_order = len(messages) – всі нові POST /replicate будуть додаватися «з місця, де зупинилися».

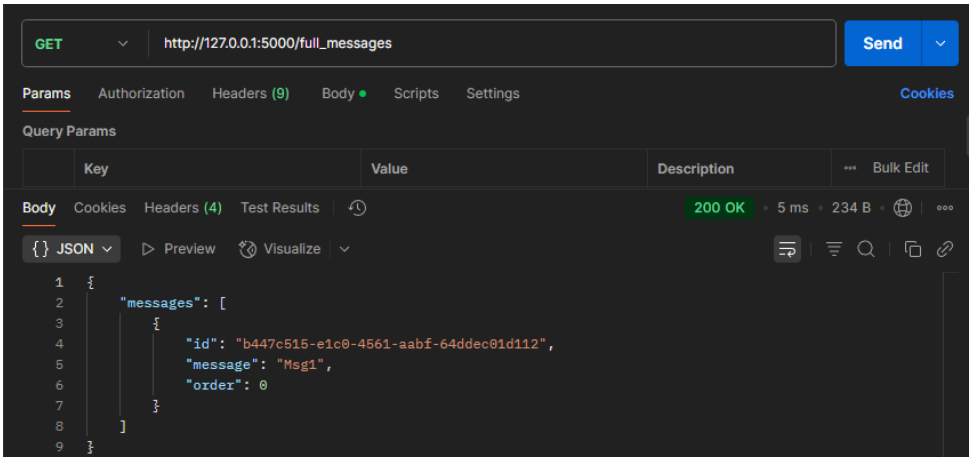
Self-check acceptance test:

- Start M + S1
- send (Msg1, W=1) - Ok
- send (Msg2, W=2) - Ok
- send (Msg3, W=3) - Wait
- send (Msg4, W=1) - Ok
- Start S2
- Check messages on S2 - [Msg1, Msg2, Msg3, Msg4]

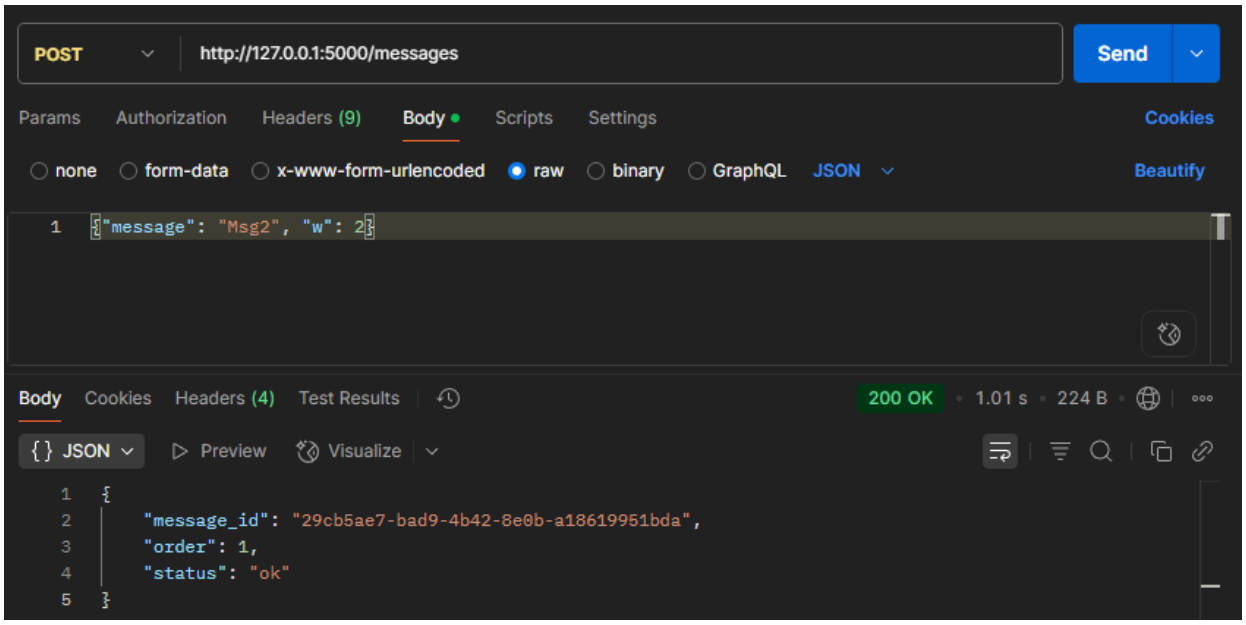
Start M + S1

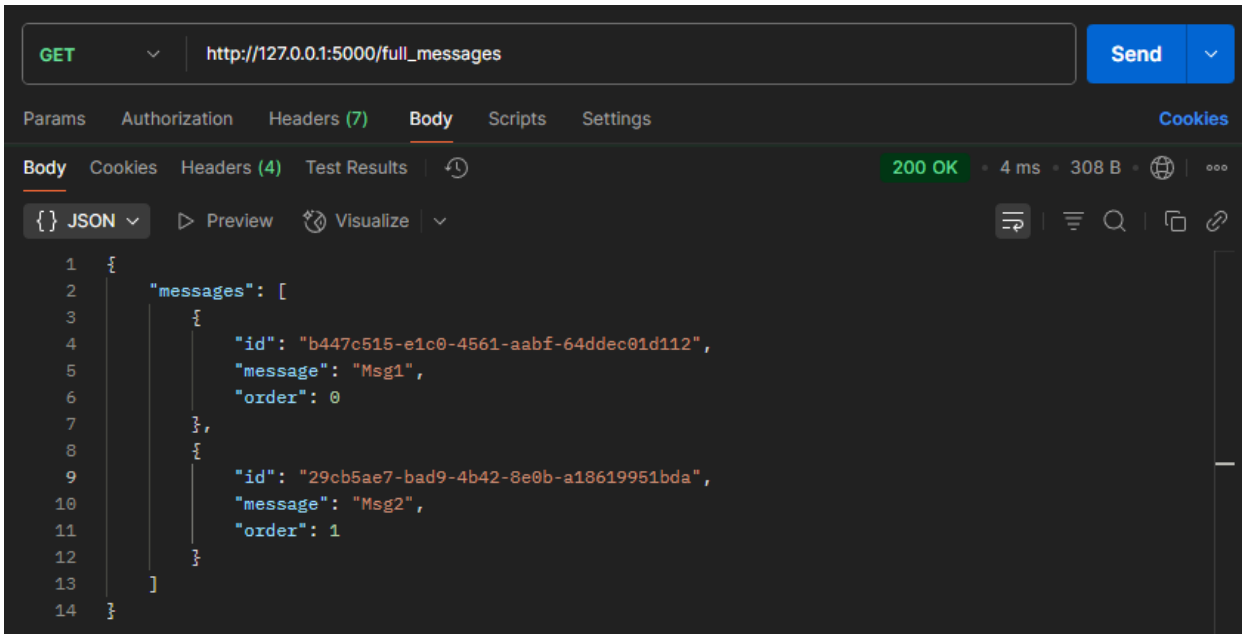
send (Msg1, W=1) — Ok

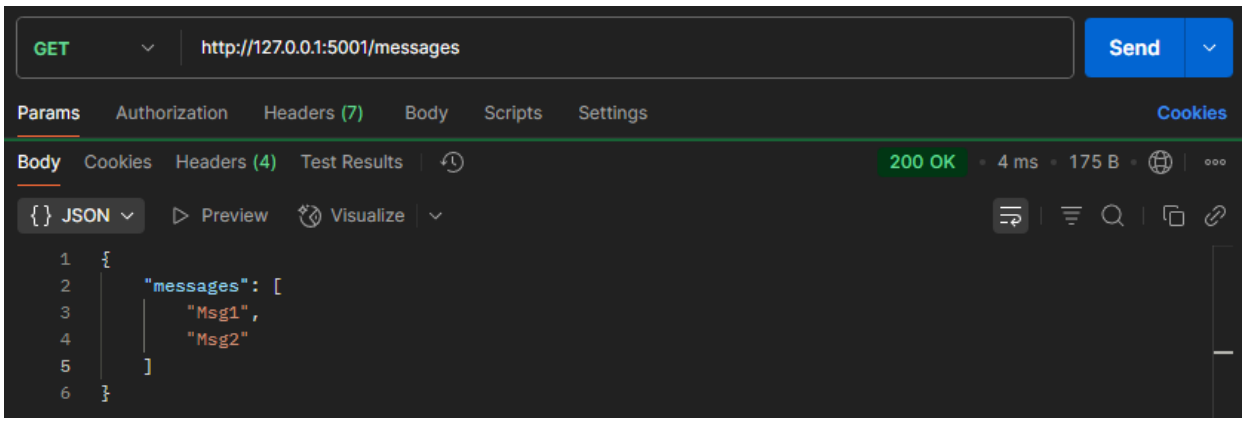




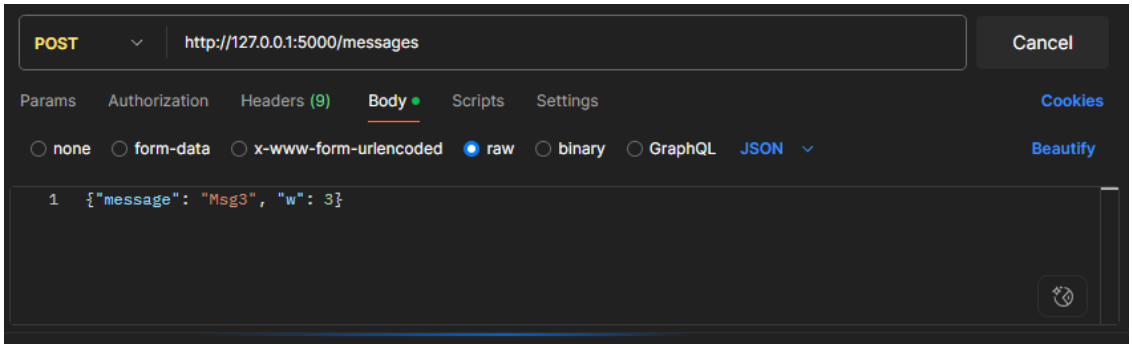
send (Msg2, W=2) — Ok



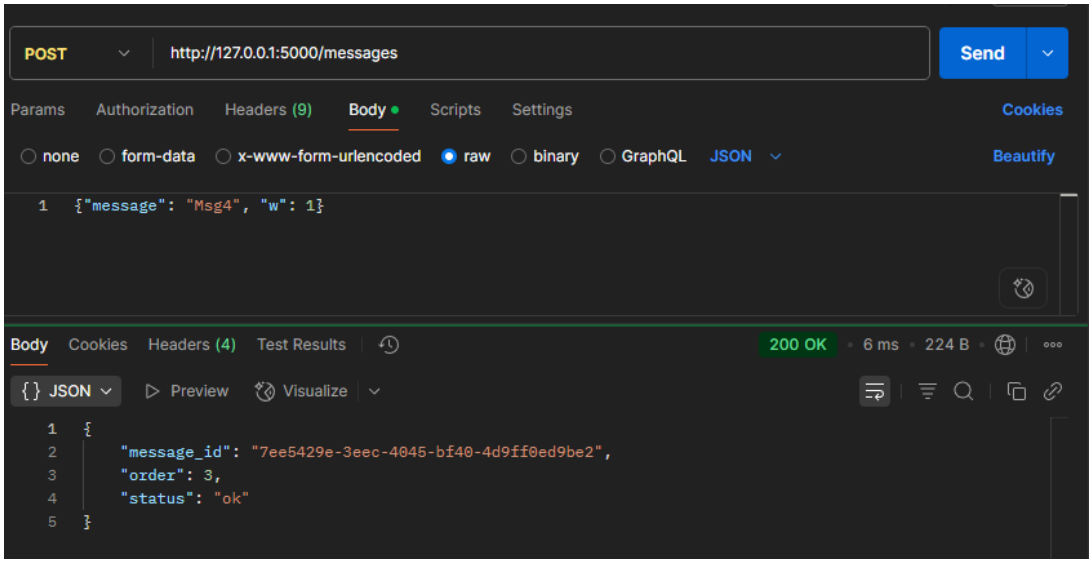


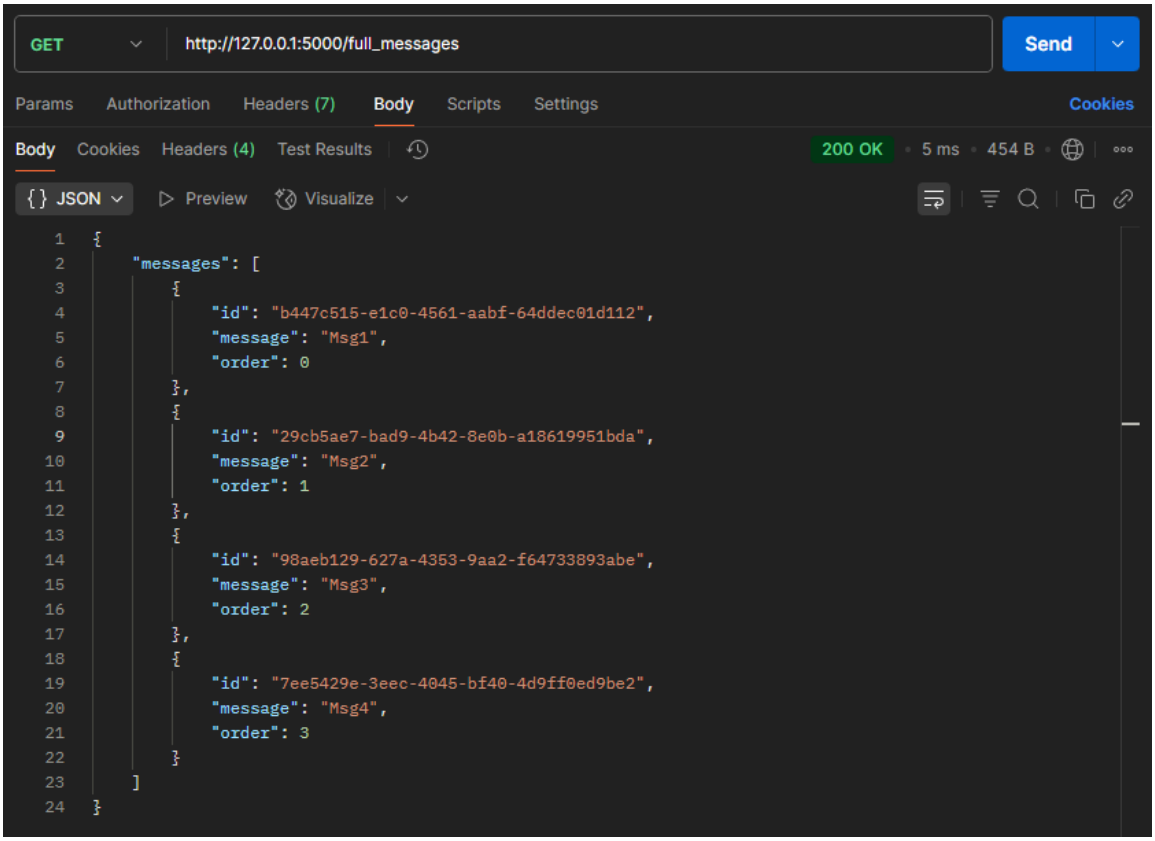


send (Msg3, W=3) - Wait

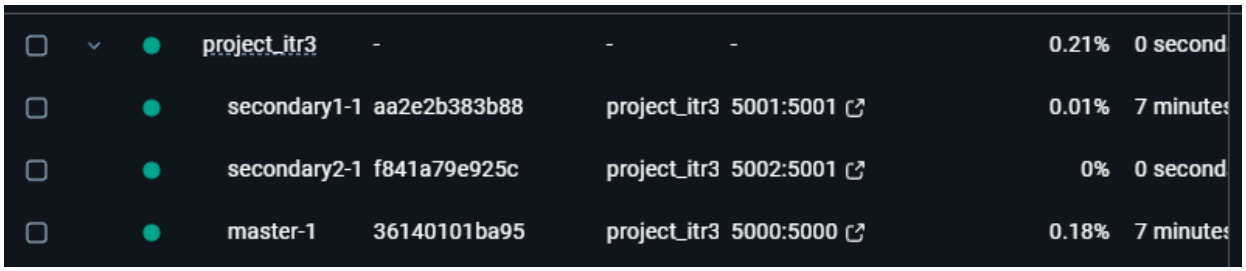


send (Msg4, W=1) - Ok

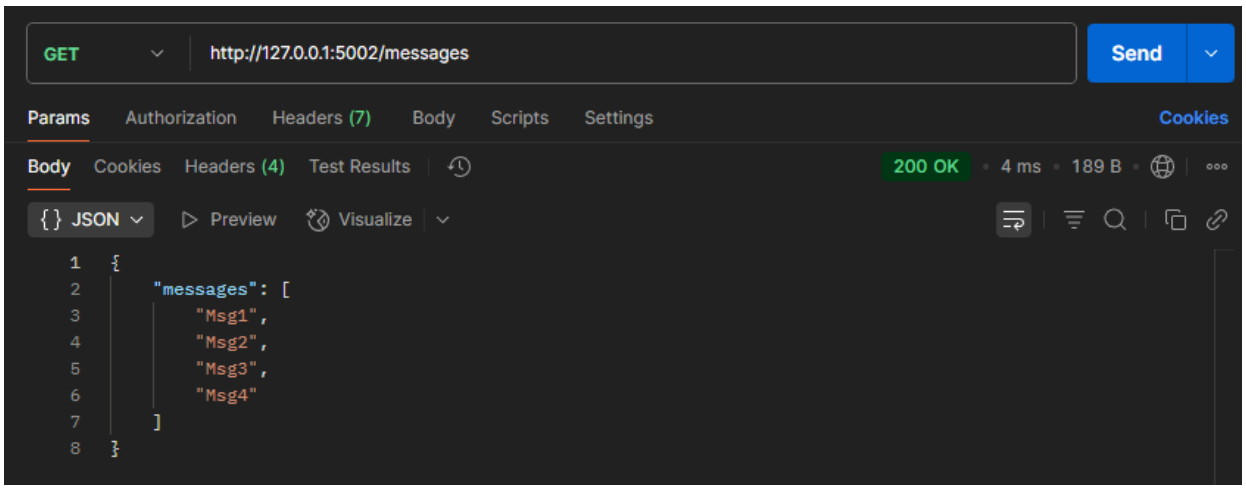




Start S2



Check messages on S2 - [Msg1, Msg2, Msg3, Msg4]



Перевіряю тепер автоматичними тестами, в них перевіряється в більшості все те саме, але автоматично.

Тести:
1) перевірка що при w=3 master буде чекати запуску усіх secondary, перевірка доставки після запуску, перевірка що сервіс не блокується для інших запитів
2) перевірка що нода отримала всі повідомлення після запуску
3) перевірка на подвійне отримання
4) перевірка на правильний порядок при помилці в доставці