Міністерство освіти і науки України

Національний технічний університет України

"Київський політехнічний інститут імені Ігоря Сікорського"

Фізико-технічний інститут

ПРОЕКТУВАННЯ РОЗПОДІЛЕНИХ СИСТЕМ

Проект

Ітерація №3

Виконав студент гр. ФБ-31мп:

Шевченко С. Ш.

Київ – 2024

# Загальна інформація:

[Репозиторій](https://github.com/dedkuzmich/distributed-systems-design-2024) Github з кодом (папка [project/v3](https://github.com/dedkuzmich/distributed-systems-design-2024/tree/main/project/v3))

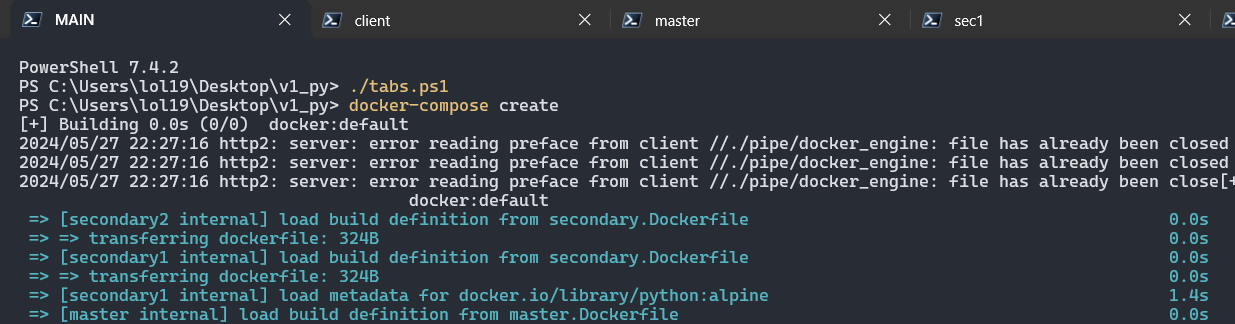
# Демонстрація роботи:

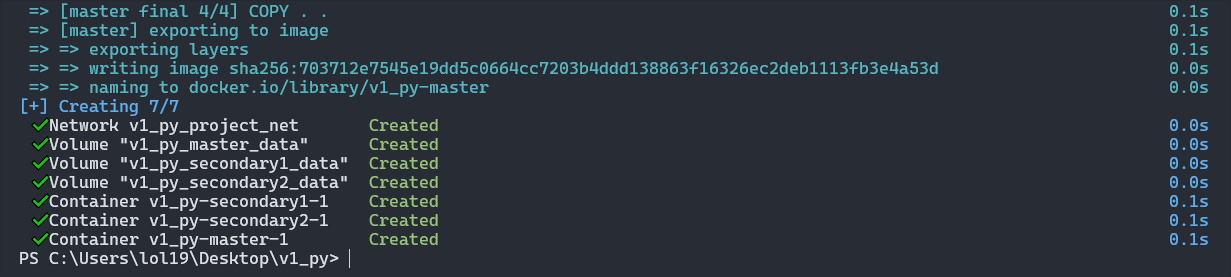
Створимо 5 вкладок у Windows Terminal за допомогою PowerShell 7 скрипту tab.ps1:

**$ ./tabs.ps1**

Створимо Docker контейнери:

**$ docker-compose create**





Видалити контейнери можна так:

**$ docker-compose down --rmi all --volumes**

Запустимо всі сервіси **окрім secondary2** в такій послідовності у відповідних вкладках:

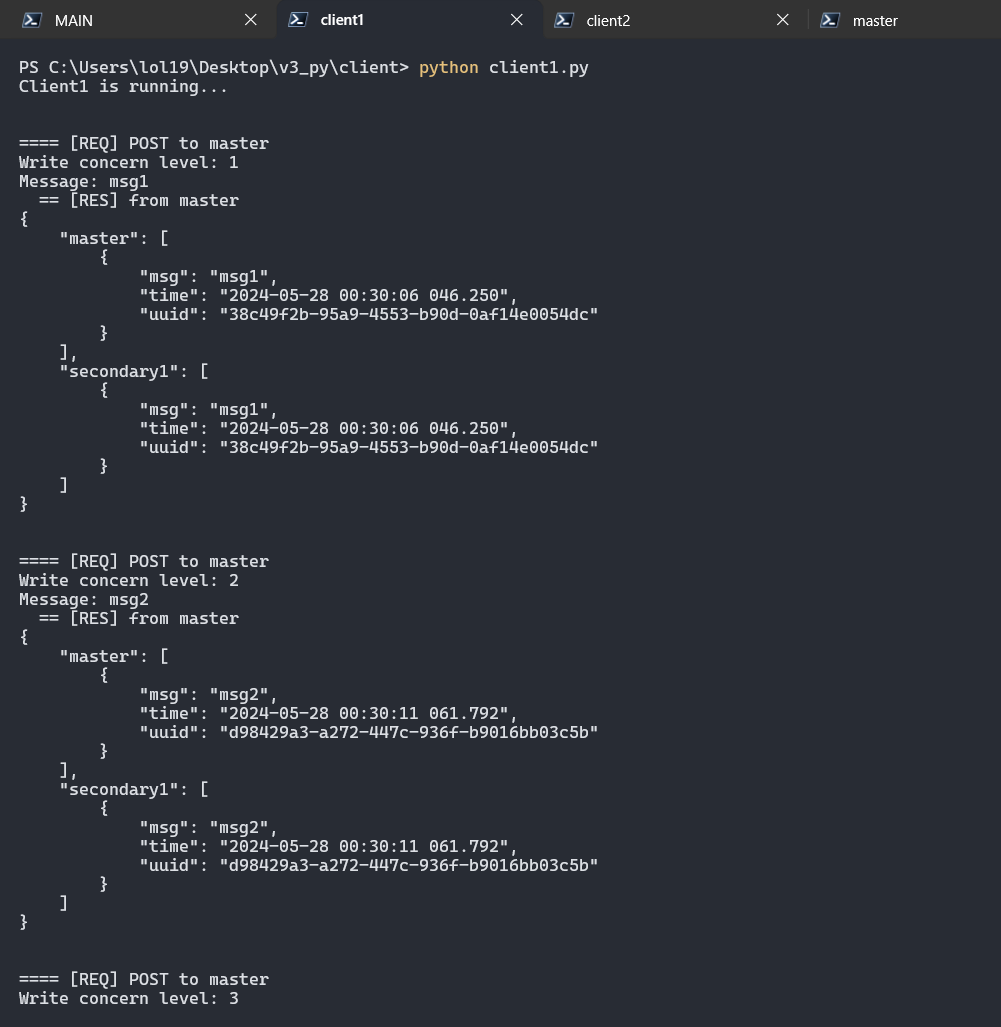
**$ docker-compose up master --no-recreate**

**$ docker-compose up secondary1 --no-recreate**

**$ python client1.py**

**$ python client2.py**

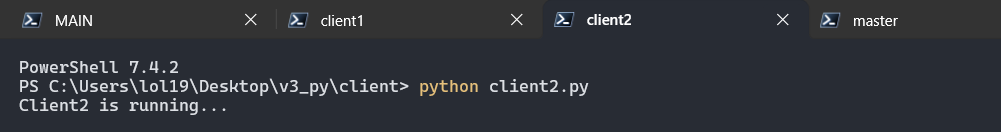
## Вивід client1

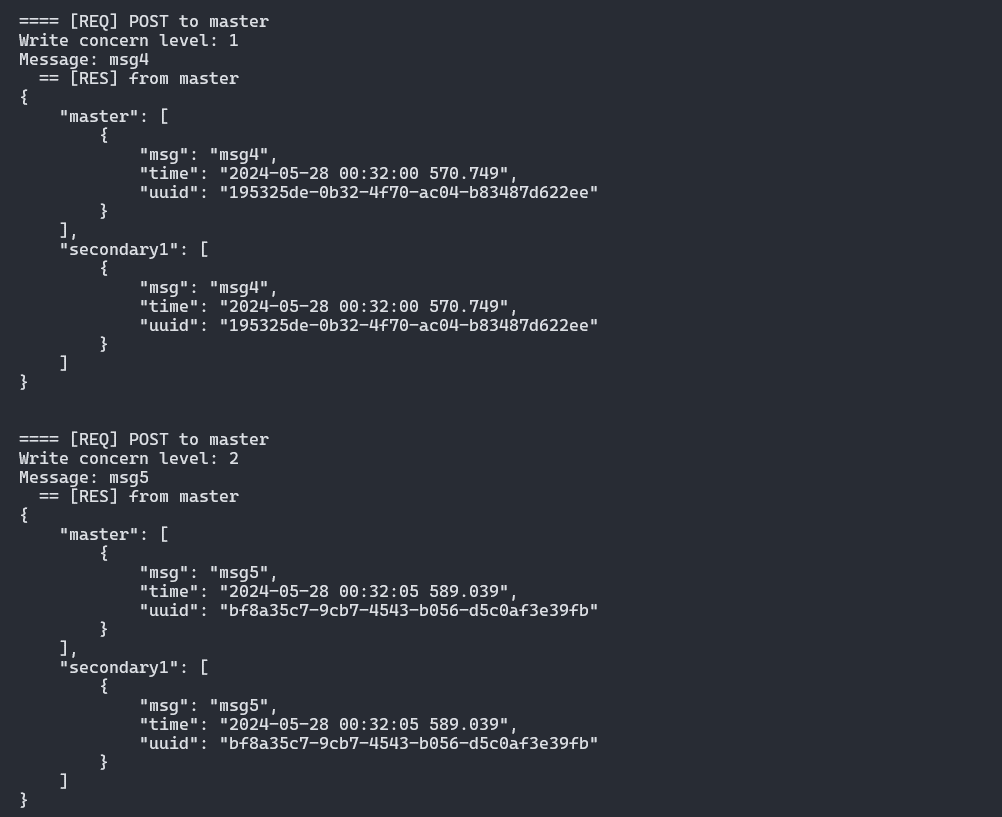


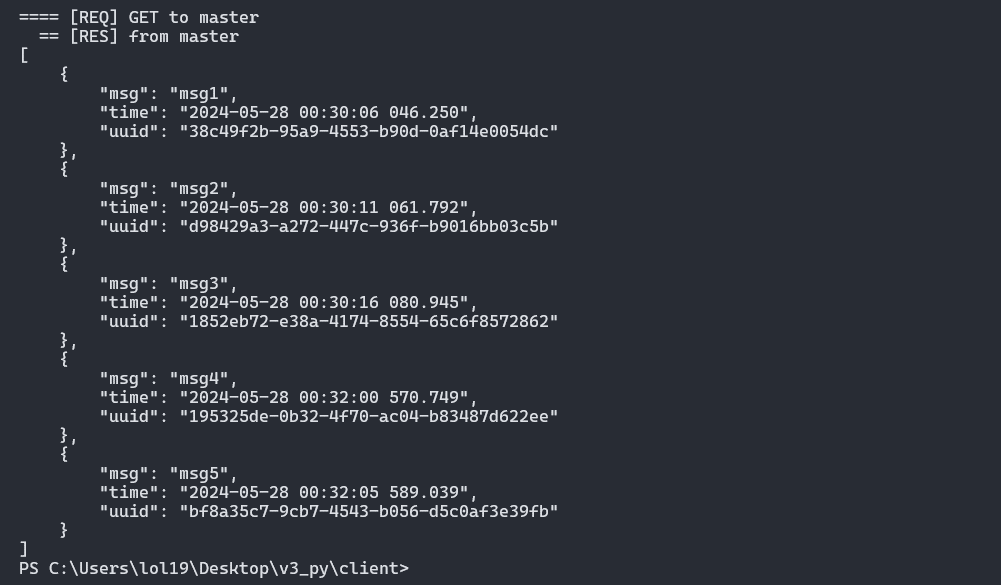
1. POST-запитом відправляються 3 повідомлення "msg1"-"msg3" на master. Master повертає ті повідомленя, які є новими для кожного сервісу. При write concern = 3, client1 блокується, тобто очікує відповіді master.

2. GET-запитом клієнт зчитує всі збережені повідомлення.

## Вивід client2

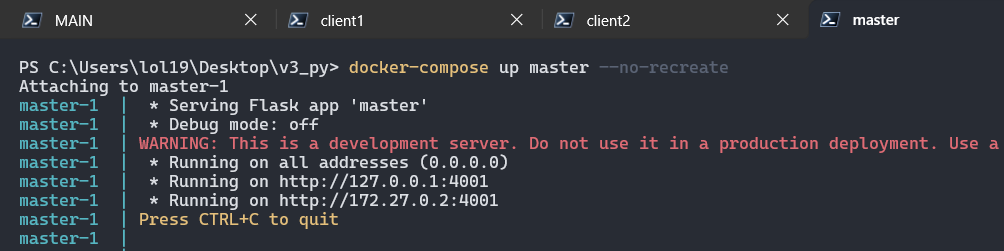




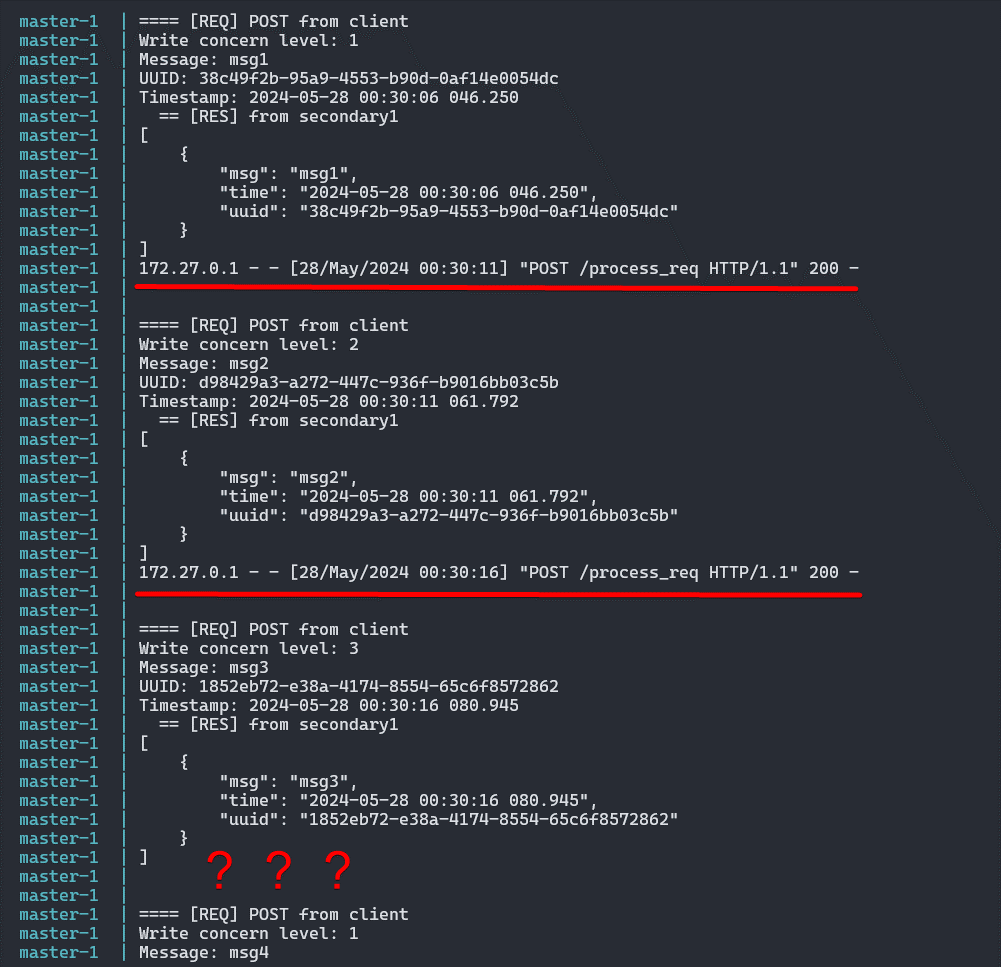


Client2 успішно завершується, так як не здійснює POST-запитів з write concern = 3.

## Вивід master



Обробка запитів client1:



На скріншоті видно, що останній POST-запит з "msg3" ще не завершився, а наступний POST-запит з "msg4" вже йде на обробку.

1. При надходженні POST-запиту з повідомленням, master генерує для нього UUID, мітку часу. Всі ці дані додаються у список master'а, а потім цей список відправляється на всі secondaries:

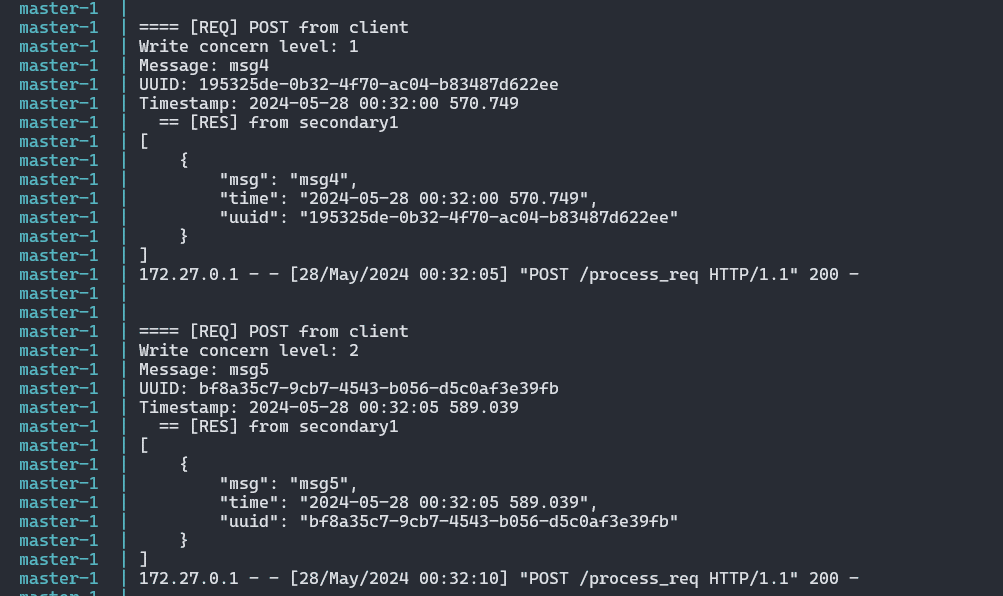
1.1. Якщо к-сть відповідей від secondaries >= рівня write concern, то POST-запит вважається успішниим.

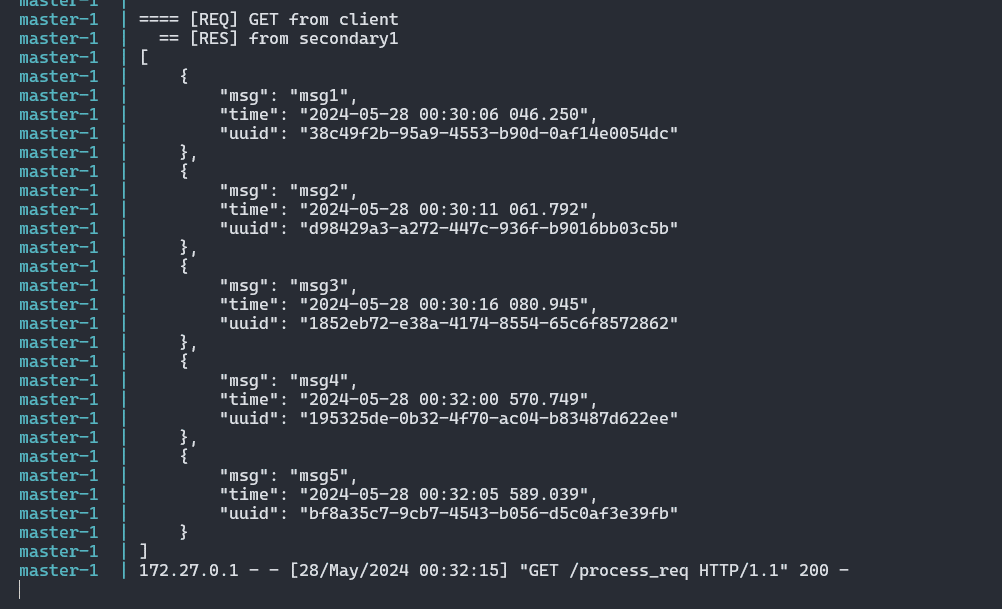
1.2. Якщо к-сть відповідей від secondaries < рівня write concern, то POST-запит до "проблемних" secondaries повторюється через 2 секунди. Master знає, від яких secondaries він вже отримав відповідь, отже запити йдуть тільки на ті secondaries, від яких відповіді ще не було.

[!] Клієнт, з якого надійшов POST-запит, блокується допоки write concern не буде задовільнено. Інші клієнти можуть і далі взаємодіяти з master.

2. При надходженні GET-запиту, master зчитує повідомлення із secondaries, проводить дедублікацію, сортує за часом та надсилає їх клієнту. Якщо жоден secondary не повертає відповідь, то master надсилає клієнту повідомлення зі свого списку.

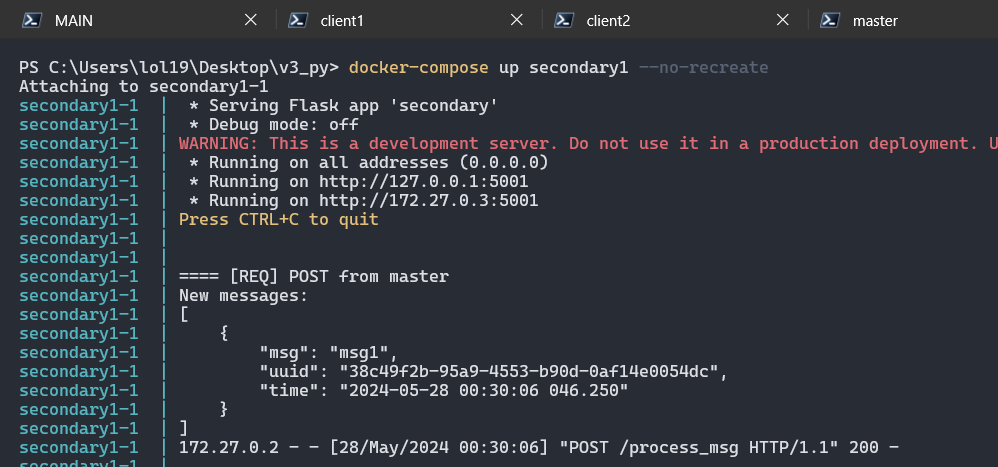
Обробка запитів client2:

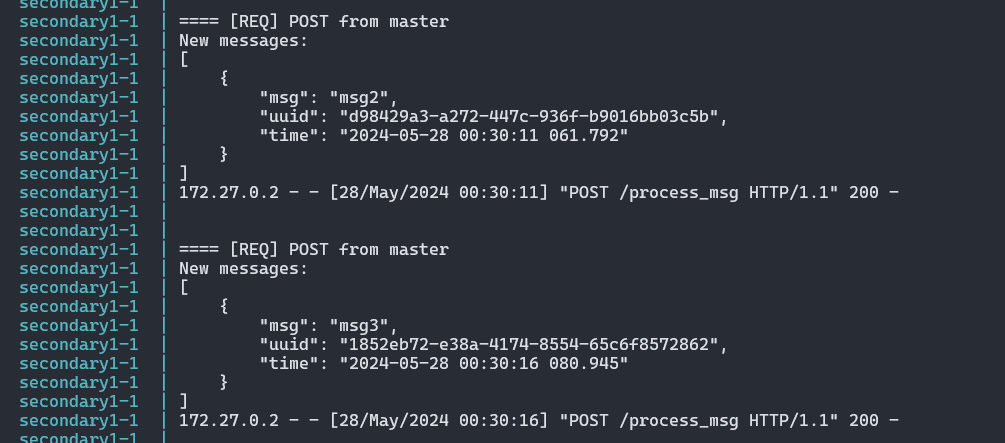


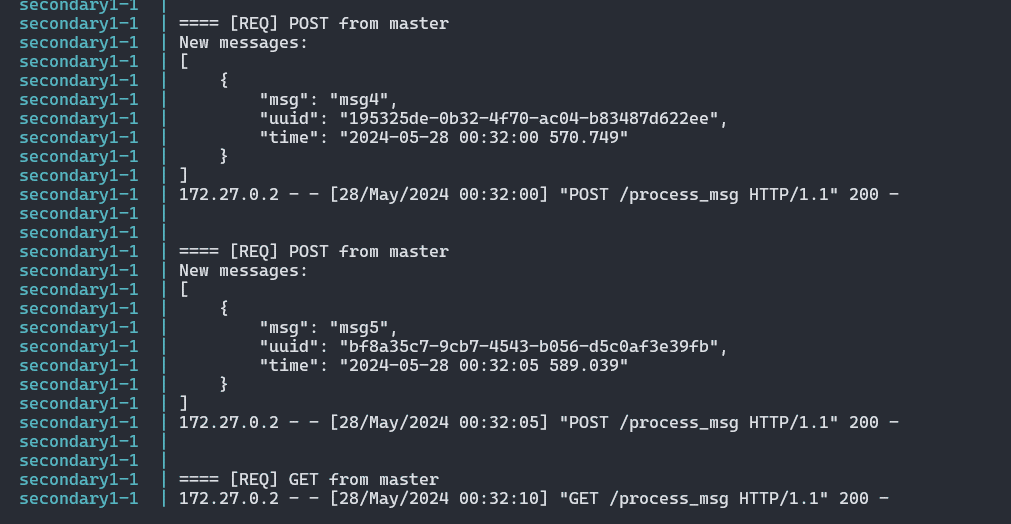


Всі запити від client2 завершуються успішно, так як не містять write concern = 3.

## Вивід secondary1





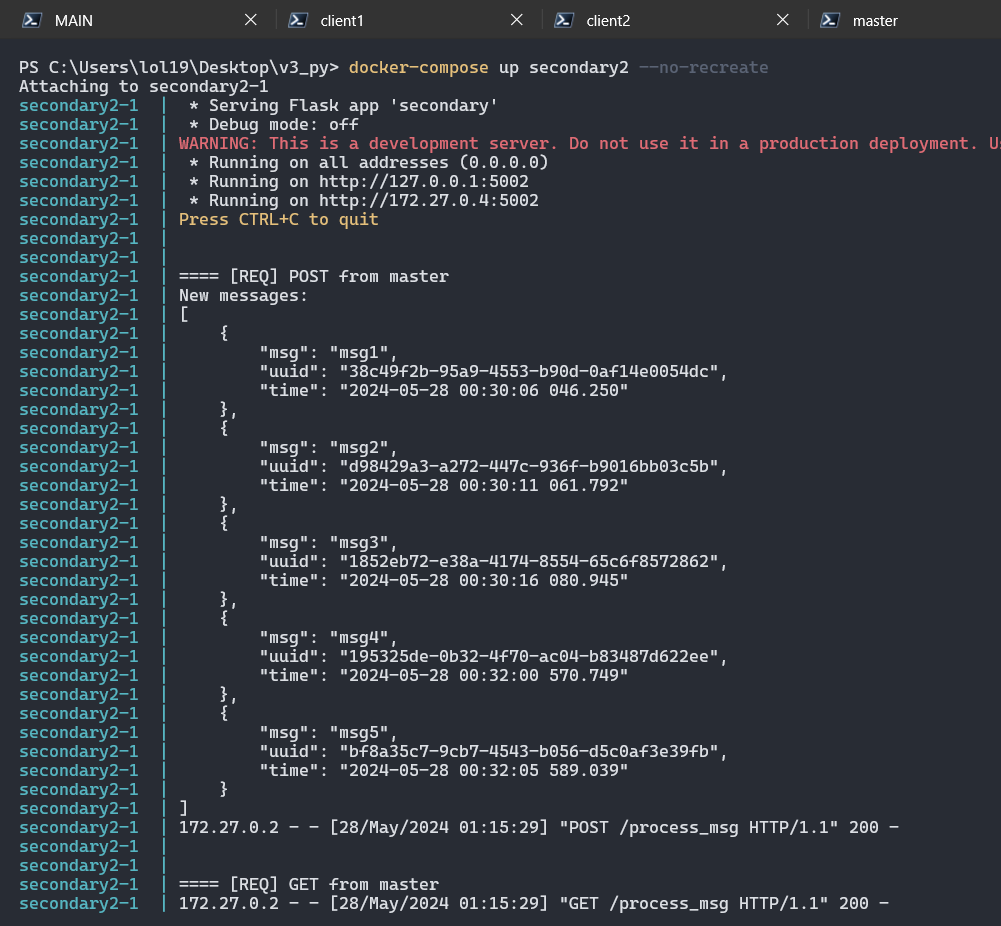


1. При надходженні POST-запиту, secondary отримує не 1 повідомлення, а цілий список. Secondary самостійно визначає нові повідомлення, додає їх до свого списку та сортує список за часом.

2. При надходженні GET-запиту, secondary надсилає вміст свого списку.

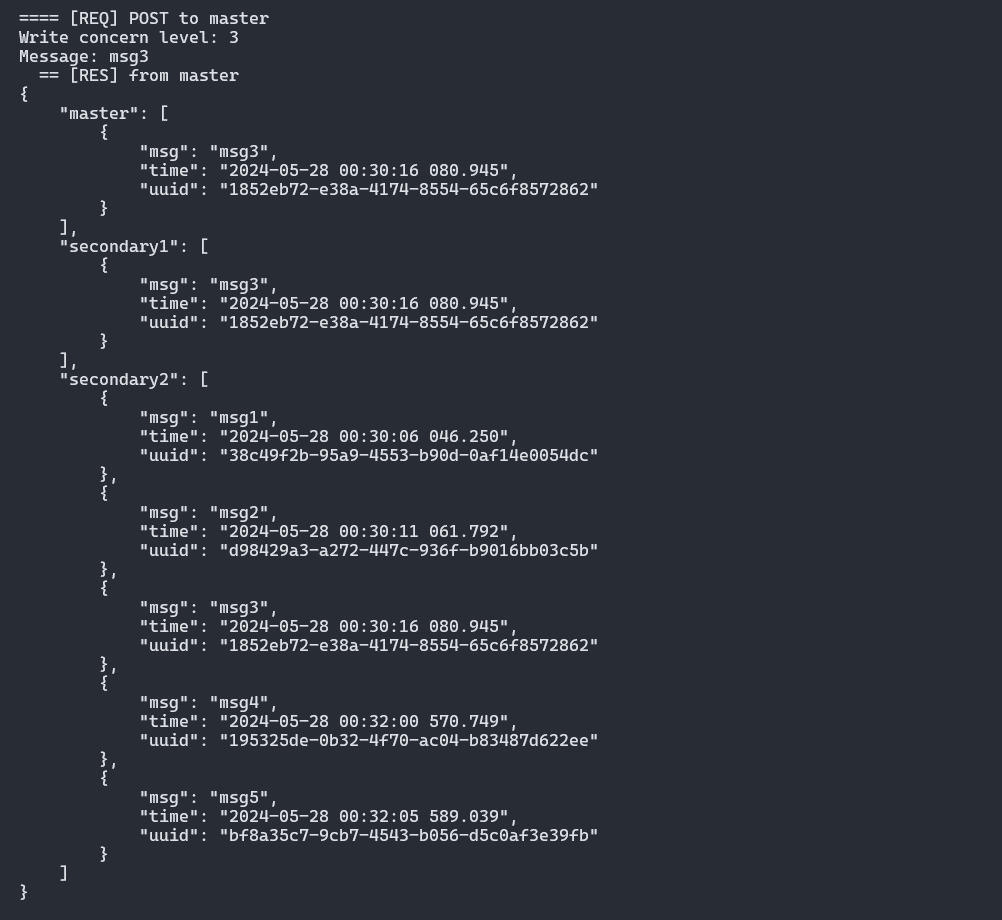
Тепер запустимо secondary2:

**$ docker-compose up secondary2 --no-recreate**



Бачимо, як реплікуються всі 5 повідомлень, включаючи "msg3".

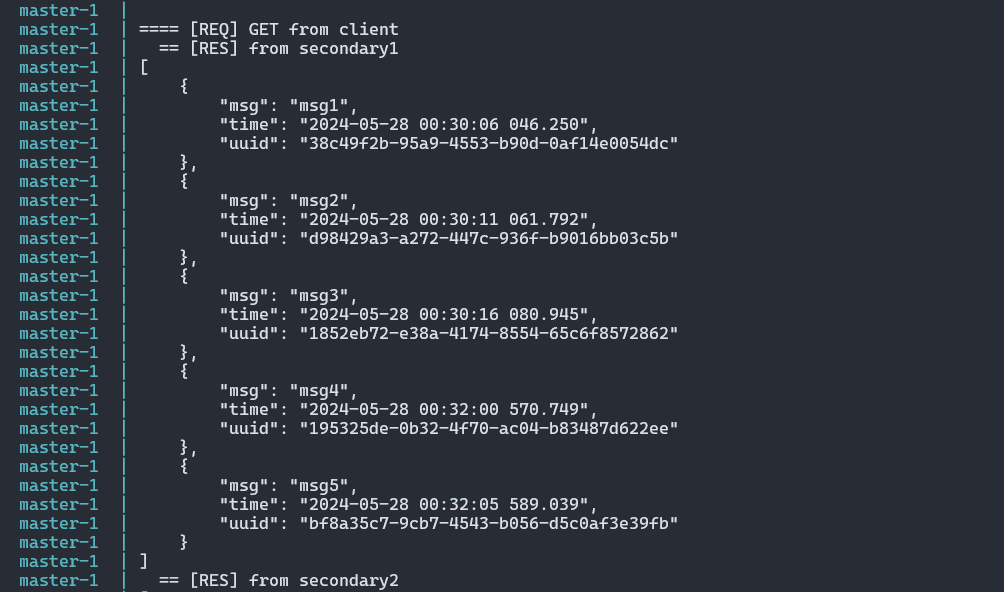
Повернемося до client1:

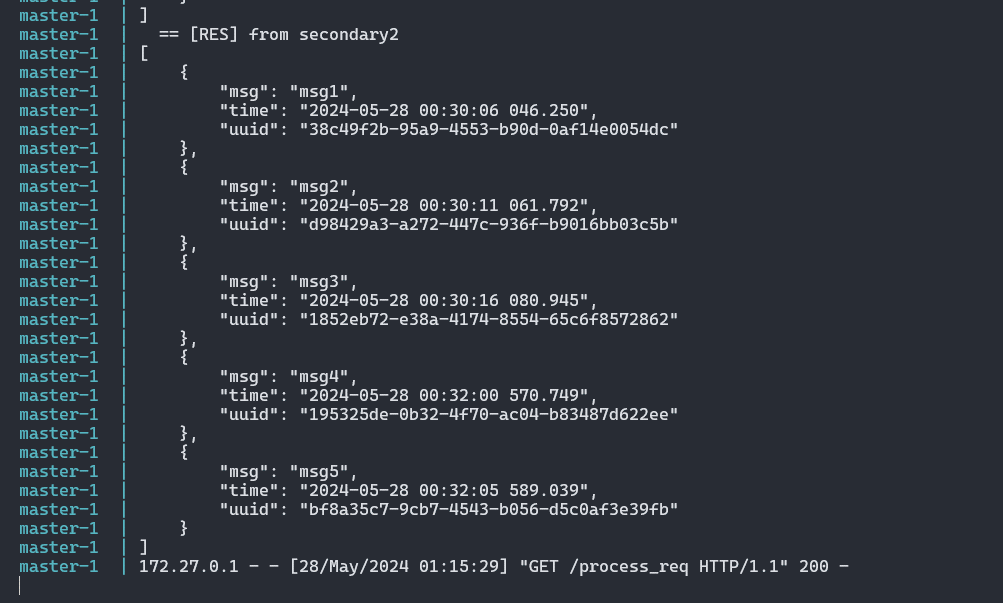


POST-запит з "msg3" завершився коректно, write concern = 3 задовільнено.

У відповіді видно, які повідомлення для яких сервісів є новими. Якщо master та secondary1 за результатами POST-запиту отримали тільки "msg3", то нещодавно запущений secondary2 отримав всі повідомлення, які він пропустив.

## Вивід master





При обробці GET запиту видно, що secondary1 та secondary2 повертають однакові списки. При запуску secondary2, master відправляє йому актуальний список.