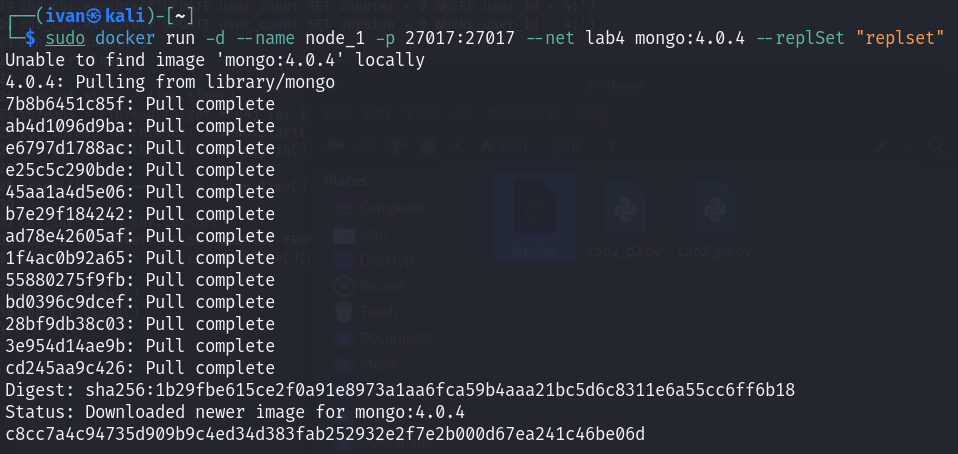
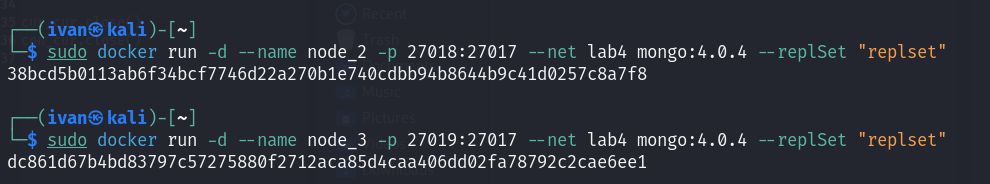
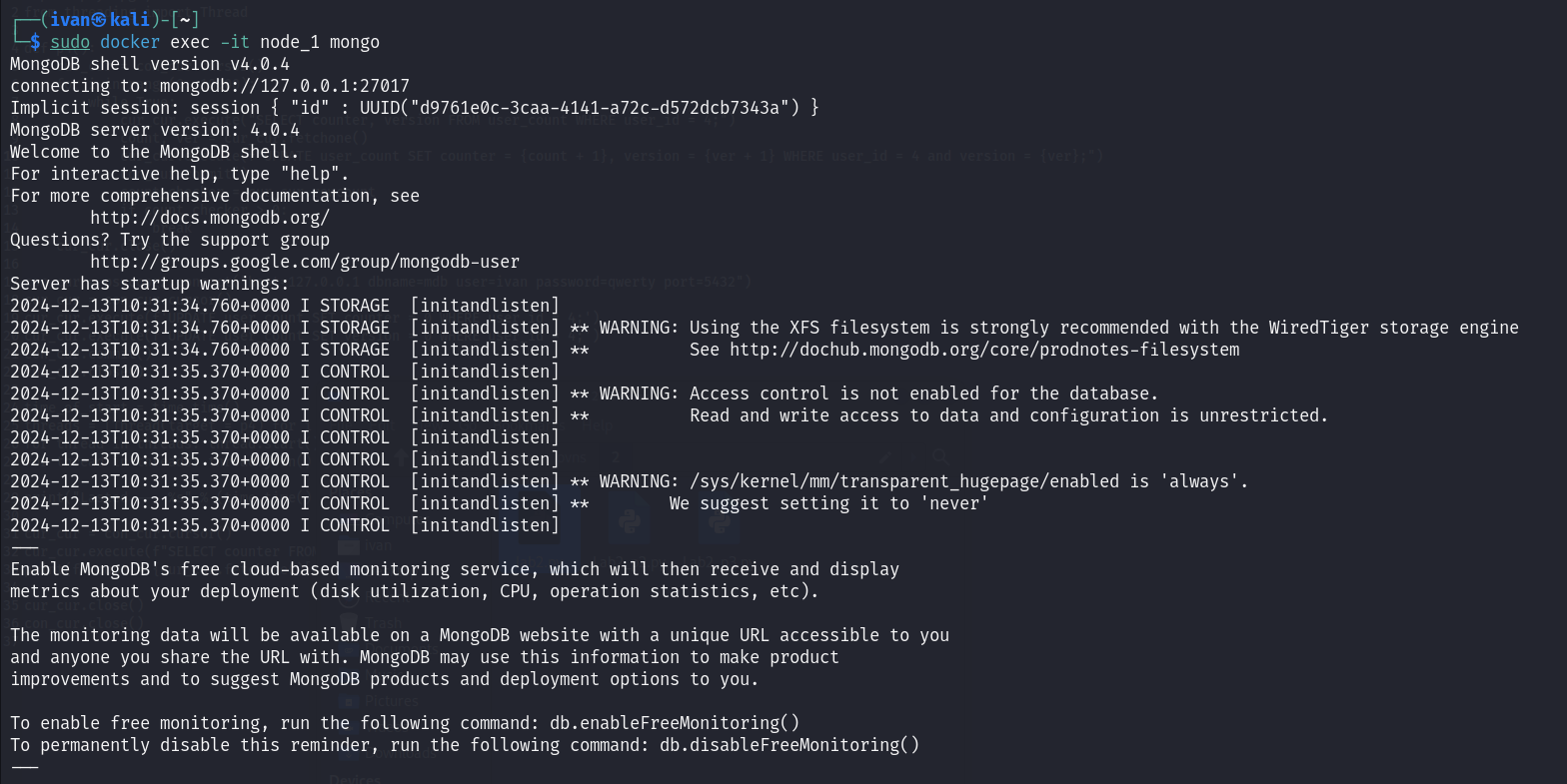
**Завдання:**

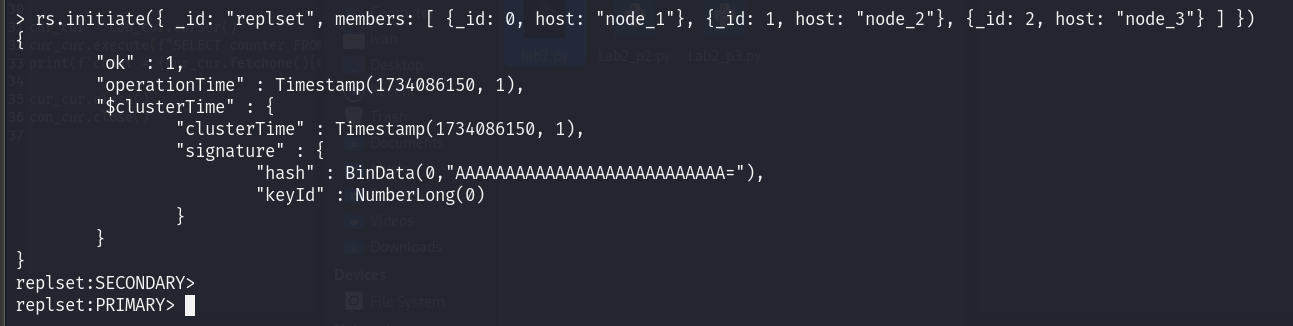
**I Налаштування реплікації**

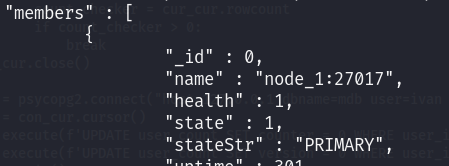
1. Налаштувати реплікацію в конфігурації: Primary with Two Secondary Members (P-S-S) (всі ноди можуть бути запущені як окремі процеси або у Docker контейнерах) - <http://docs.mongodb.org/manual/core/replica-set-architecture-three-members/>
   * Deploy a Replica Set for Testing and Development- <http://docs.mongodb.org/manual/tutorial/deploy-replica-set-for-testing/>
   * <http://www.tugberkugurlu.com/archive/setting-up-a-mongodb-replica-set-with-docker-and-connecting-to-it-with-a--net-core-app>

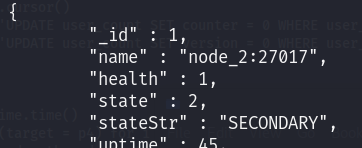


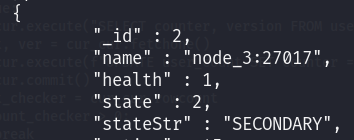




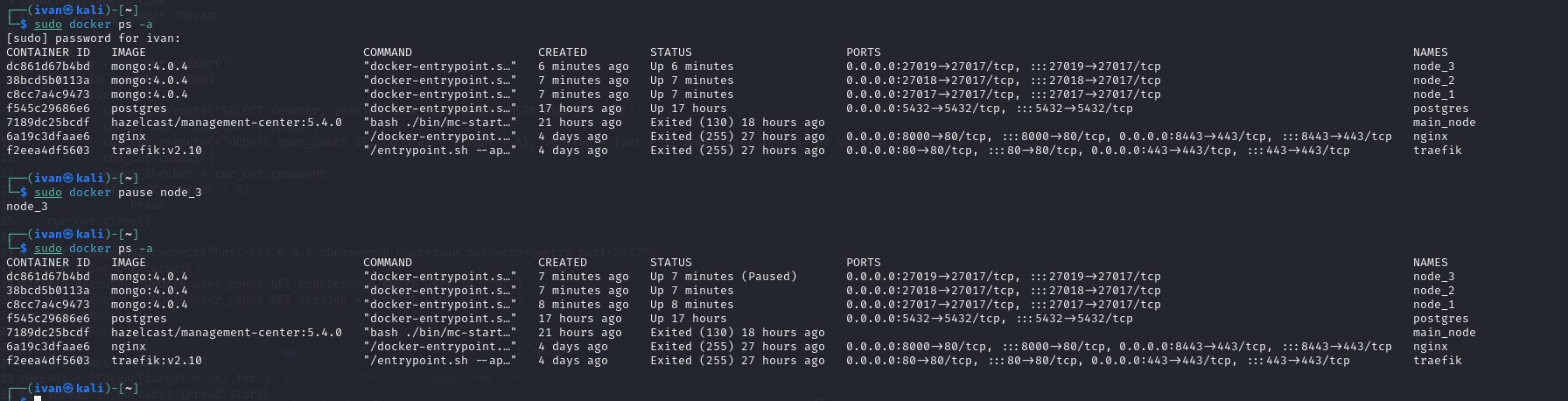


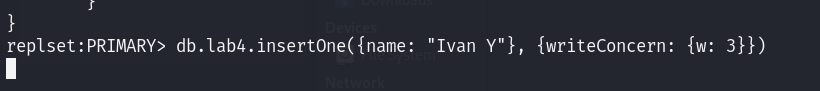




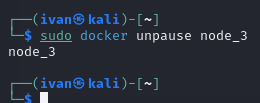


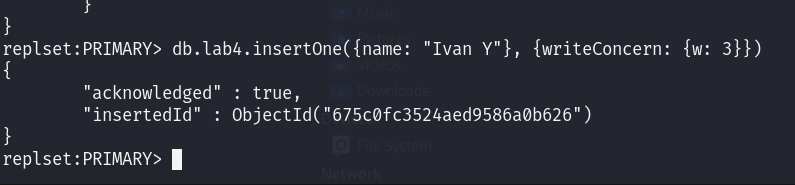
1. Спробувати зробити запис з однією відключеною нодою та *write concern* рівнім 3 та нескінченім таймаутом. Спробувати під час таймаута включити відключену ноду



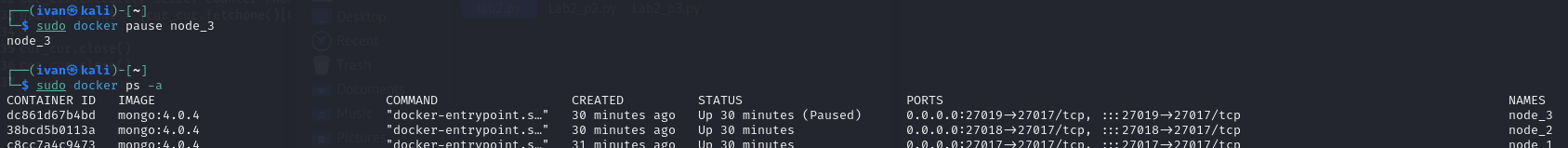


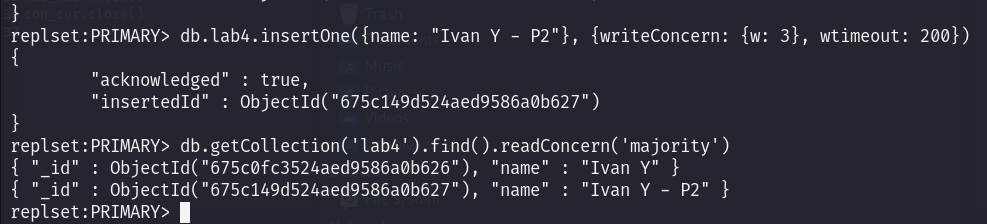
Після включення:



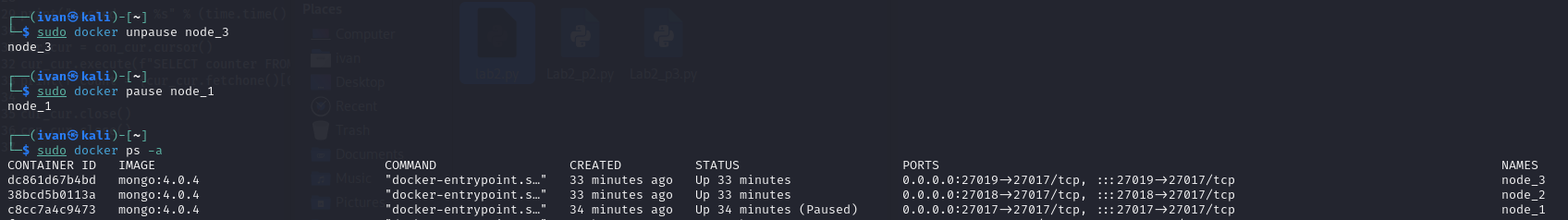


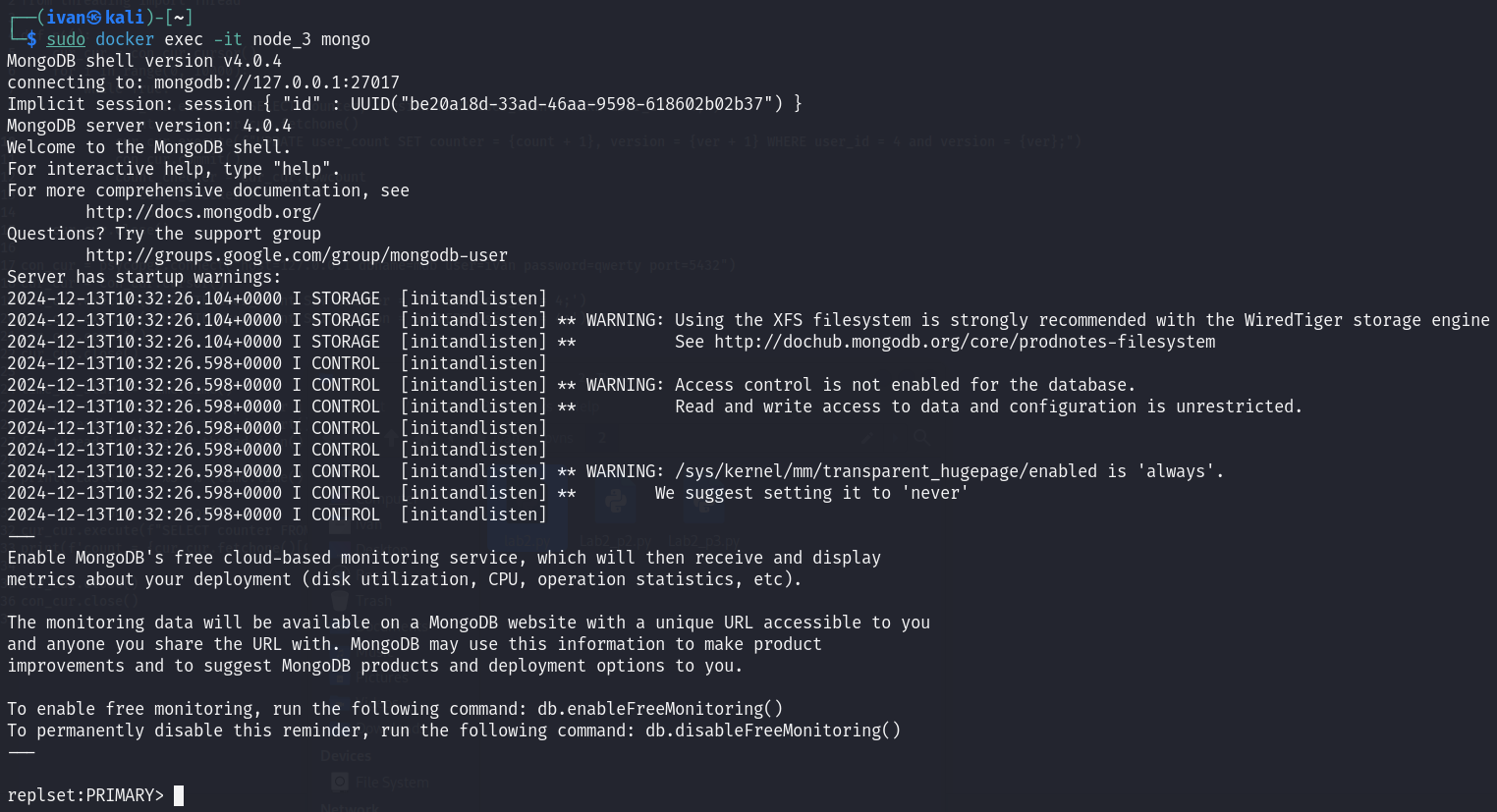
1. Аналогічно попередньому пункту, але задати скінченний таймаут та дочекатись його закінчення. Перевірити чи данні записались і чи доступні на читання з рівнем *readConcern: “majority”*

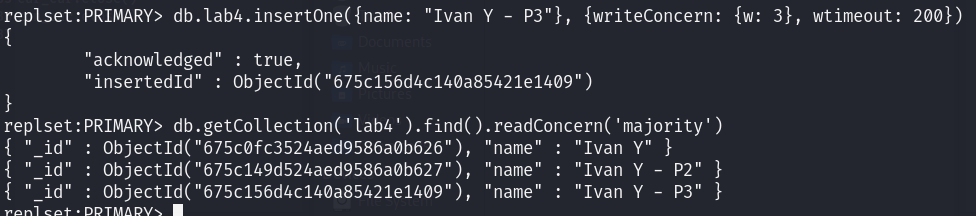


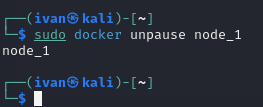


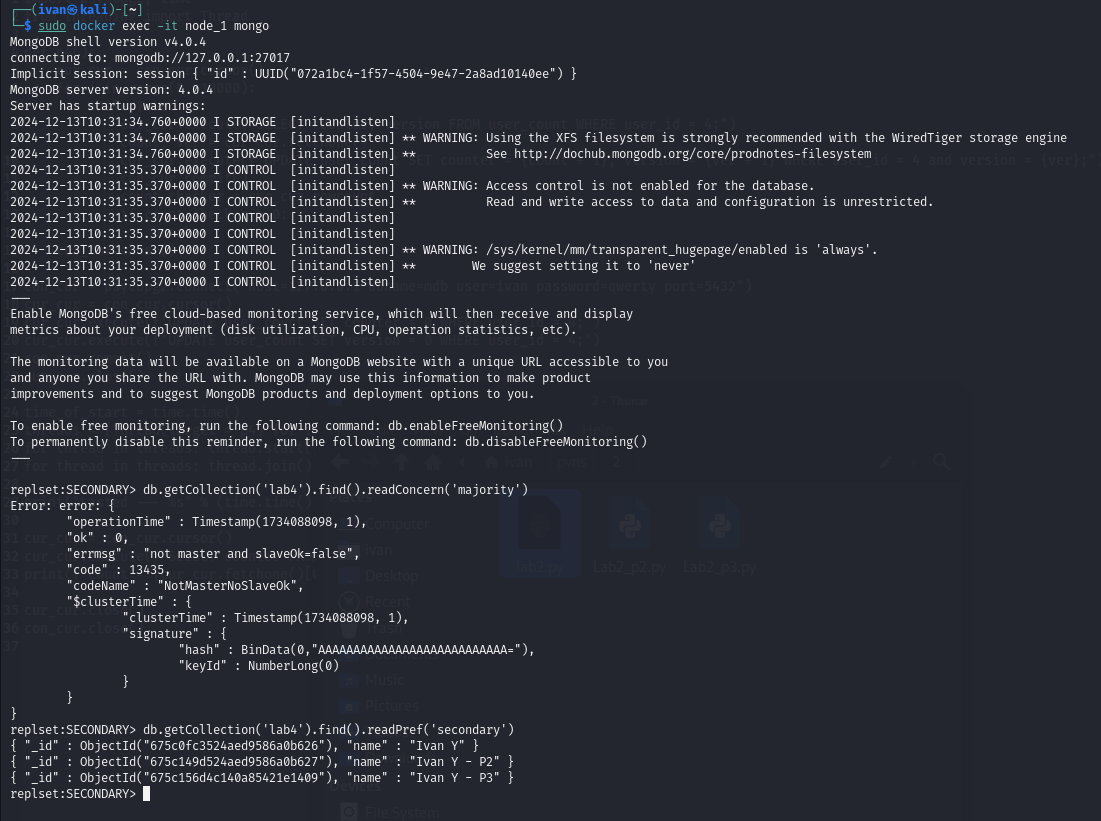
1. Продемонстрував перевибори primary node відключивши поточний primary (Replica Set Elections) - <http://docs.mongodb.org/manual/core/replica-set-elections/>
   * і що після відновлення роботи старої primary на неї реплікуються нові дані, які з'явилися під час її простою











**II Аналіз продуктивності та перевірка цілісності**

Аналогічно попереднім завданням, необхідно буде створити колекцію (таблицю) з каунтером лайків. Далі з 10 окремих клієнтів одночасно запустити інкерементацію каунтеру лайків по 10\_000 на кожного клієнта з різними опціями взаємодії з MongoDB.

Для того, щоб не було lost updates, для оновлення каунтера необхідно використовувати функцію [findOneAndUpdate()](https://www.mongodb.com/docs/manual/reference/method/db.collection.findOneAndUpdate/#db.collection.findoneandupdate--)

Приклад використання:

*db.grades.findOneAndUpdate(*

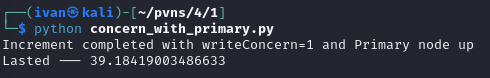
*{ "name" : "R. Stiles" },*

*{ $inc: { "points" : 5 } }*

*)*

<https://www.mongodb.com/docs/manual/reference/method/db.collection.findOneAndUpdate/#update-a-document>

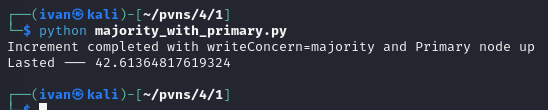
1. Вказавши у парметрах *findOneAndUpdate writeConcern = 1* (це буде означати, що запис іде тільки на Primary ноду і не чекає відповіді від Secondary), запустіть 10 клієнтів  з інкрементом по 10\_000 на кожному з них. Виміряйте час виконання та перевірте чи кінцеве значення буде дорівнювати очікуваному - 100К



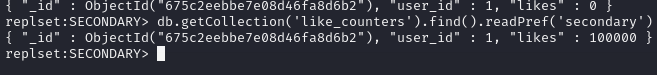




1. Вказавши у парметрах *findOneAndUpdate writeConcern = majority* (це буде означати, що Primary чекає поки значення запишется на більшість нод), запустіть 10 клієнтів  з інкрементом по 10\_000 на кожному з них. Виміряйте час виконання та перевірте чи кінцеве значення буде дорівнювати очікуваному - 100К

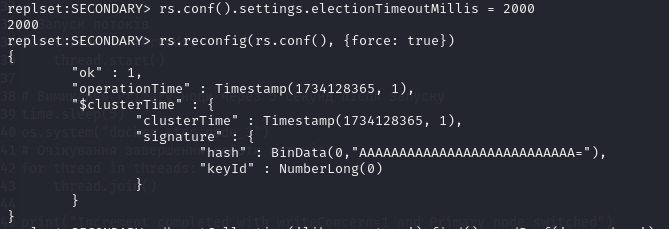


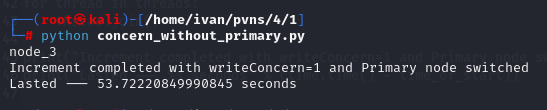




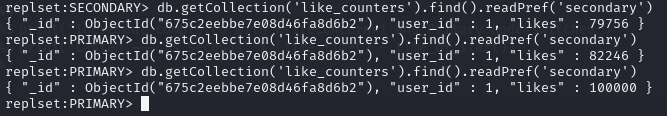
1. Повторно запустить код при *writeConcern = 1*, але тепер під час роботи відключіть Primary ноду і подивитись що буде обрана інша Primary нода, яка продовжить обробку запитів, і чи кінцевий результат буде коректним.

Попередньо зменшив electionTimeoutMillis на всіх нодах:

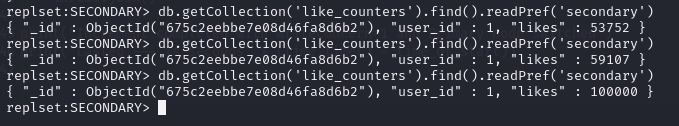




Нода, що світчнулась:

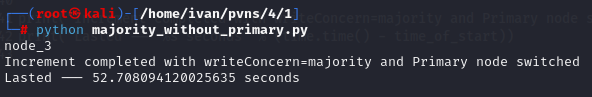


Інша жива нода:

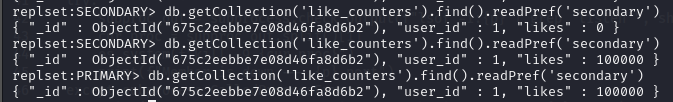


1. Повторно запустить код при *writeConcern = majority*, але тепер під час роботи відключіть Primary ноду і подивитись що буде обрана інша Primary нода, яка продовжить обробку запитів, і чи кінцевий результат буде коректним.

При *writeConcern = 1* деякі записи можуть губитись під час раптового відключення. При *writeConcern = majority* має виходити очікуваний результат.



Нода, яка свічнулась:



В моєму випадку (принаймні для одного запуску), і при *writeConcern = 1*, і при *writeConcern = majority* кінцеві значення каунтерів були коректними.