Міністерство освіти і науки України
НТУУ «Київський політехнічний інститут»
Фізико-технічний інститут

Протокол лабораторної роботи №4
з дисципліни Проектування високонавантажених систем
на тему: Налаштування реплікації та перевірка відмовостійкості MongoDB

Виконав: студент групи ФІ-21
Грунда Ярослав

Київ, 2025

Завдання

1. Налаштувати реплікацію в конфігурації: Primary with Two Secondary Members (P-S-S)
2. Спробувати зробити запис з однією відключеною нодою та write concern рівнім 3 та нескінченім таймаутом. Спробувати під час таймаута включити відключену ноду
3. Аналогічно попередньому пункту, але задати скінченний таймаут та дочекатись його закінчення. Перевірити чи данні записались і чи доступні на читання з рівнем readConcern: “majority”
4. Продемонстрував перевибори primary node відключивши поточний primary (Replica Set Elections) і що після відновлення роботи старої primary на неї реплікуються нові дані, які з’явилися під час її простою
5. Створити колекцію (таблицю) з каунтером лайків. Далі з 10 окремих клієнтів одночасно запустити інкерементацію каунтеру лайків по 10000 на кожного клієнта з різними опціями взаємодії з MongoDB.

Зміст

1 Частина 1

1.1 Запуск Docker-контейнерів

Docker-compose.yml :

```
services:
  mongo1:
    image: mongo:6
    container_name: mongo1
    hostname: mongo1
    ports:
      - 27017:27017
    command: ["mongod", "--replSet", "rs0", "--bind_ip_all"]

  mongo2:
    image: mongo:6
    container_name: mongo2
    hostname: mongo2
    ports:
      - 27018:27017
    command: ["mongod", "--replSet", "rs0", "--bind_ip_all"]

  mongo3:
    image: mongo:6
    container_name: mongo3
    hostname: mongo3
    ports:
      - 27019:27017
    command: ["mongod", "--replSet", "rs0", "--bind_ip_all"]

networks:
  default:
    name: mongo-net
```

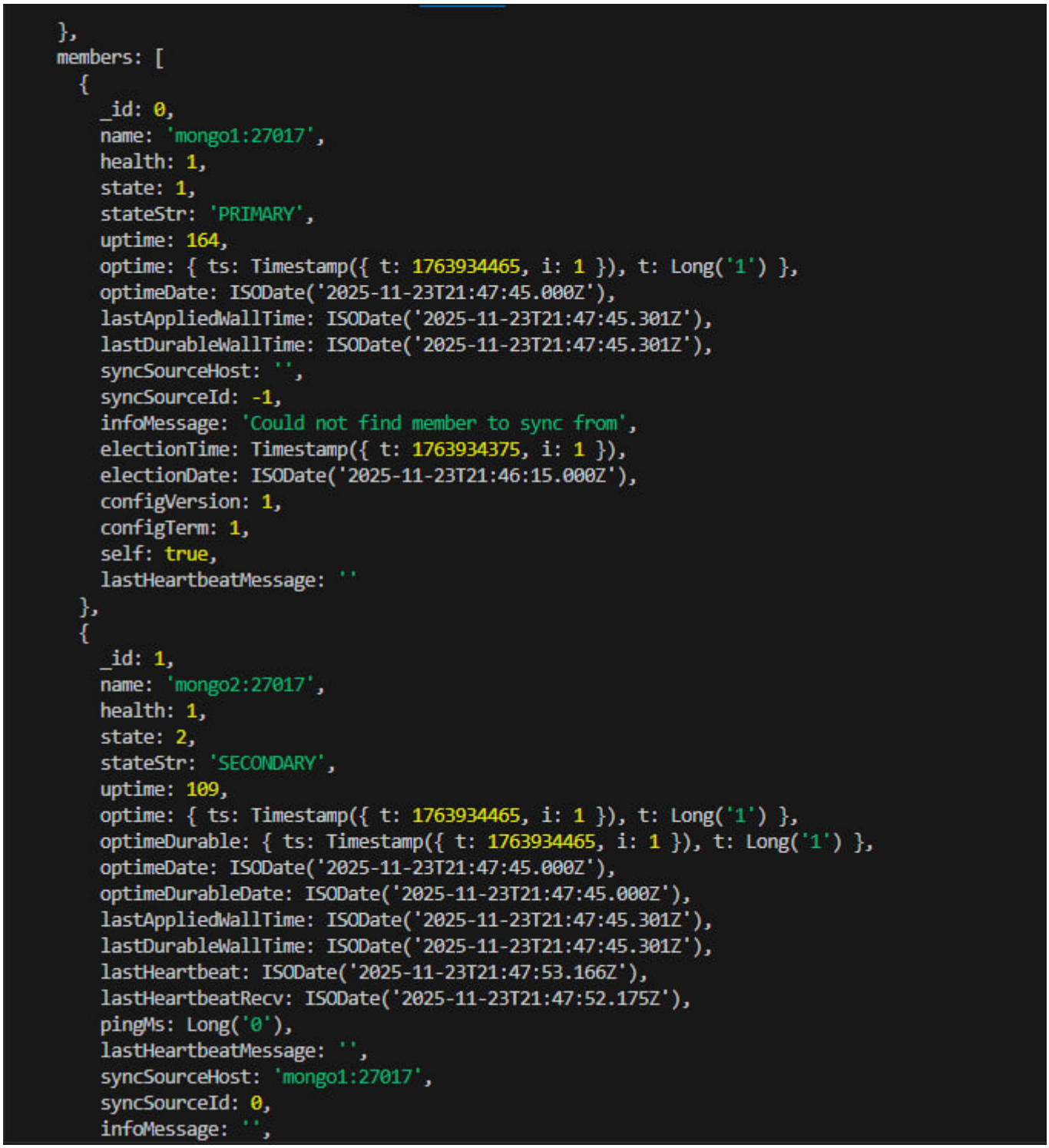
```
D:\University\semestr_7\High-Load_Systems\lab4>docker compose up -d
[+] Running 4/4
 ✓ Network mongo-net  Created
 ✓ Container mongo1   Started
 ✓ Container mongo2   Started
 ✓ Container mongo3   Started

D:\University\semestr_7\High-Load_Systems\lab4>docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS
2063a0ef47ec   mongo:6   "docker-entrypoint.s…"   10 seconds ago   Up 9 seconds    0.0.0.0:27019->27017
098563541e02   mongo:6   "docker-entrypoint.s…"   10 seconds ago   Up 9 seconds    0.0.0.0:27018->27017
9a8017979ec4   mongo:6   "docker-entrypoint.s…"   10 seconds ago   Up 9 seconds    0.0.0.0:27017->27017
```

1.2 Ініціалізація Replica Set

```
D:\University\semestr_7\High-Load_Systems\lab4>docker exec -it mongo1 mongosh
Current Mongosh Log ID: 6923808647e6f9ab98ce5f46
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=
Using MongoDB:          6.0.26
Using Mongosh:          2.5.8

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/


To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting
   2025-11-23T21:45:09.949+00:00: Using the XFS filesystem is strongly recommended with the WiredT
   2025-11-23T21:45:10.982+00:00: Access control is not enabled for the database. Read and write a
   2025-11-23T21:45:10.983+00:00: vm.max_map_count is too low
------

test> rs.initiate({
...   _id: "rs0",
...   members: [
...     { _id: 0, host: "mongo1:27017" },
...     { _id: 1, host: "mongo2:27017" },
...     { _id: 2, host: "mongo3:27017" }
...   ]
... })
{ ok: 1 }
rs0 [direct: other] test>

rs0 [direct: primary] test>
```

1.3 Перевірка статуса

```
rs0 [direct: primary] test> rs.status()
{
  set: 'rs0',
  date: ISODate('2025-11-23T21:47:53.182Z'),
  myState: 1,
  term: Long('1'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1763934465, i: 1 }), t: Long('1') },
    lastCommittedWallTime: ISODate('2025-11-23T21:47:45.301Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1763934465, i: 1 }), t: Long('1') },
    appliedOpTime: { ts: Timestamp({ t: 1763934465, i: 1 }), t: Long('1') },
    durableOpTime: { ts: Timestamp({ t: 1763934465, i: 1 }), t: Long('1') },
    lastAppliedWallTime: ISODate('2025-11-23T21:47:45.301Z'),
    lastDurableWallTime: ISODate('2025-11-23T21:47:45.301Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1763934415, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2025-11-23T21:46:15.070Z'),
    electionTerm: Long('1'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1763934363, i: 1 }), t: Long('-1') },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1763934363, i: 1 }), t: Long('-1') },
    numVotesNeeded: 2,
    priorityAtElection: 1,
    electionTimeoutMillis: Long('10000'),
    numCatchUpOps: Long('0'),
    newTermStartDate: ISODate('2025-11-23T21:46:15.238Z'),
    wMajorityWriteAvailabilityDate: ISODate('2025-11-23T21:46:15.864Z')
  },
  members: [
    {
      _id: 0,
      name: 'mongo1:27017',
      health: 1,
      state: 1,
```



```
  },
  members: [
    {
      _id: 0,
      name: 'mongo1:27017',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 164,
      optime: { ts: Timestamp({ t: 1763934465, i: 1 }), t: Long('1') },
      optimeDate: ISODate('2025-11-23T21:47:45.000Z'),
      lastAppliedWallTime: ISODate('2025-11-23T21:47:45.301Z'),
      lastDurableWallTime: ISODate('2025-11-23T21:47:45.301Z'),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: 'Could not find member to sync from',
      electionTime: Timestamp({ t: 1763934375, i: 1 }),
      electionDate: ISODate('2025-11-23T21:46:15.000Z'),
      configVersion: 1,
      configTerm: 1,
      self: true,
      lastHeartbeatMessage: ''
    },
    {
      _id: 1,
      name: 'mongo2:27017',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 109,
      optime: { ts: Timestamp({ t: 1763934465, i: 1 }), t: Long('1') },
      optimeDurable: { ts: Timestamp({ t: 1763934465, i: 1 }), t: Long('1') },
      optimeDate: ISODate('2025-11-23T21:47:45.000Z'),
      optimeDurableDate: ISODate('2025-11-23T21:47:45.000Z'),
      lastAppliedWallTime: ISODate('2025-11-23T21:47:45.301Z'),
      lastDurableWallTime: ISODate('2025-11-23T21:47:45.301Z'),
      lastHeartbeat: ISODate('2025-11-23T21:47:53.166Z'),
      lastHeartbeatRecv: ISODate('2025-11-23T21:47:52.175Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: 'mongo1:27017',
      syncSourceId: 0,
      infoMessage: '',
```

```
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: 'mongo1:27017',
      syncSourceId: 0,
      infoMessage: '',
      configVersion: 1,
      configTerm: 1
    },
    {
      _id: 2,
      name: 'mongo3:27017',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 109,
      optime: { ts: Timestamp({ t: 1763934465, i: 1 }), t: Long('1') },
      optimeDurable: { ts: Timestamp({ t: 1763934465, i: 1 }), t: Long('1') },
      optimeDate: ISODate('2025-11-23T21:47:45.000Z'),
      optimeDurableDate: ISODate('2025-11-23T21:47:45.000Z'),
      lastAppliedWallTime: ISODate('2025-11-23T21:47:45.301Z'),
      lastDurableWallTime: ISODate('2025-11-23T21:47:45.301Z'),
      lastHeartbeat: ISODate('2025-11-23T21:47:53.166Z'),
      lastHeartbeatRecv: ISODate('2025-11-23T21:47:52.175Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: 'mongo1:27017',
      syncSourceId: 0,
      infoMessage: '',
      configVersion: 1,
      configTerm: 1
    }
  ],
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1763934465, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1763934465, i: 1 })
}
rs0 [direct: primary] test>
```

1.4 Завдання 1

Відключення другої ноди

```
D:\University\semestr 7\High-Load Systems\lab4>docker stop mongo3
mongo3
```

Робимо запис з однією відключеною нодою, write concern рівнем 3 та нескінченим таймаутом.

```
rs0 [direct: primary] test> db.exp1.insertOne(
...
...   {exp: 1, status: "w3_no_timeout", ts: new Date()},
...
...   {writeConcern: {w: 3}}
...
... )
```

Отримаємо те що операція висить. Включаємо відключену ноду (docker start mongo3) і в результаті операція запису розблокується та успішно завершається.

```
rs0 [direct: primary] test> db.exp1.insertOne(
...
...   {exp: 1, status: "w3_no_timeout", ts: new Date()},
...
...   {writeConcern: {w: 3}}
...
... )
{
  acknowledged: true,
  insertedId: ObjectId('6923829f47e6f9ab98ce5f47')
}
rs0 [direct: primary] test>
```

1.5 Завдання 2

Знову зупиняємо ноду (docker stop mongo3). Пробуємо записати з таймаутом 2 секунди і після отримуємо помилку WriteConcernError:

```
rs0 [direct: primary] test> db.exp1.insertOne(
...   {exp: 2, status: "w3_timeout_2000ms", ts: new Date()},
...   {writeConcern: {w: 3, wtimeout: 2000}}
... )
Uncaught:
MongoWriteConcernError[WriteConcernFailed]: waiting for replication timed out
Additional information: {
  wtimeout: true,
  writeConcern: { w: 3, wtimeout: 2000, provenance: 'clientSupplied' }
}
Result: {
  n: 1,
  electionId: ObjectId('7fffffff000000000000001'),
  opTime: { ts: Timestamp({ t: 1763935092, i: 1 }), t: 1 },
  writeConcernError: {
    code: 64,
    codeName: 'WriteConcernFailed',
    errmsg: 'waiting for replication timed out',
    errInfo: {
      wtimeout: true,
      writeConcern: { w: 3, wtimeout: 2000, provenance: 'clientSupplied' }
    }
  },
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1763935092, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: 0
    }
  },
  operationTime: Timestamp({ t: 1763935092, i: 1 })
}
```

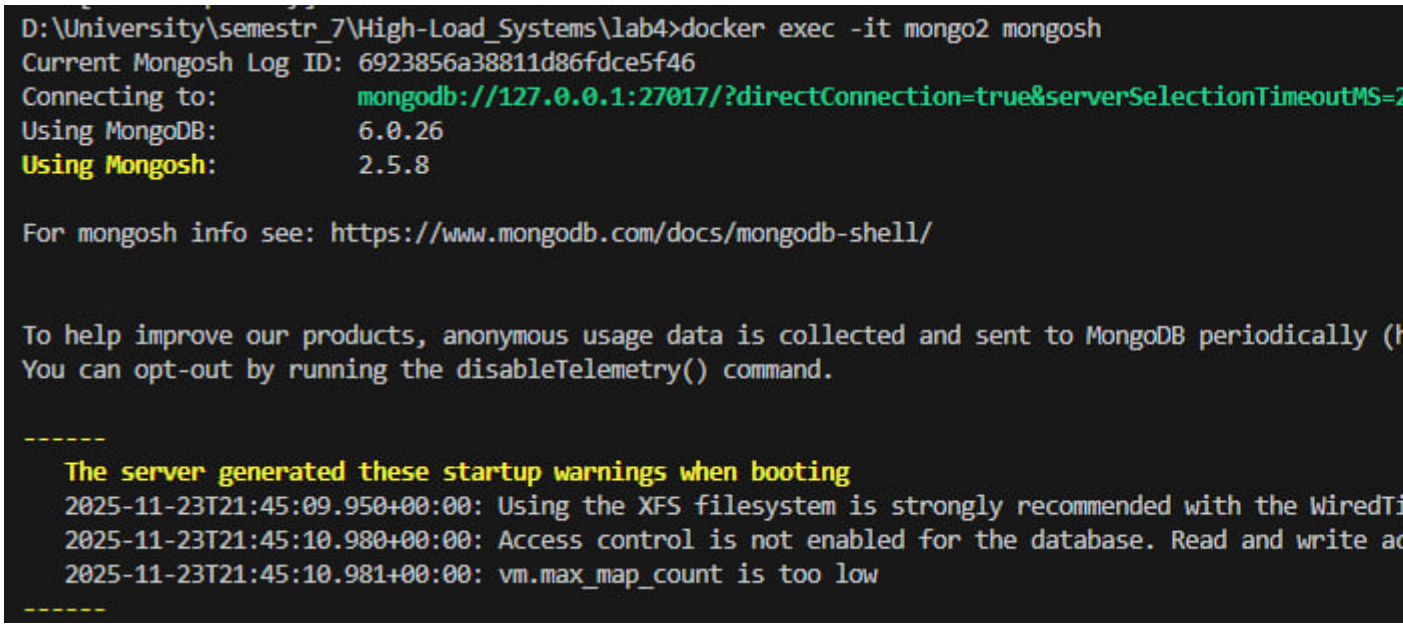
Незважаючи на отриману помилку WriteConcernError (через спливання таймауту wtimeout), дані були успішно збережені та є доступними для читання з рівнем readConcern: ’majority’.

Це пояснюється тим, що запис фактично відбувся на більшості вузлів кластера (2 з 3 нод: Primary та одна Secondary), що задовольняє умову консенсусу (majority). Помилка виникла виключно на етапі підтвердження: клієнт вимагав гарантію запису на всі три ноди (w: 3), але через відключену третю ноду кластер не зміг надати таке підтвердження у визначений час.

```
rs0 [direct: primary] test> db.exp1.find({ status: "w3_timeout_2000ms" }).readConcern("majority"
[
  {
    _id: ObjectId('6923837447e6f9ab98ce5f48'),
    exp: 2,
    status: 'w3_timeout_2000ms',
    ts: ISODate('2025-11-23T21:58:12.360Z')
  }
]
```

1.6 Завдання 3

Зупиняємо нашу primary ноду (docker stop mongo1). Підключаємось до ”живої”ноди.



Як бачимо відбувся перевибір головної ноди і нею стала друга.

```
members: [
  {
    _id: 0,
    name: 'mongo1:27017',
    health: 0,
    state: 8,
    stateStr: '(not reachable/healthy)',
    uptime: 0,
    optime: { ts: Timestamp({ t: 0, i: 0 }), t: Long('-1') },
    optimeDurable: { ts: Timestamp({ t: 0, i: 0 }), t: Long('-1') },
    optimeDate: ISODate('1970-01-01T00:00:00.000Z'),
    optimeDurableDate: ISODate('1970-01-01T00:00:00.000Z'),
    lastAppliedWallTime: ISODate('2025-11-23T22:06:03.098Z'),
    lastDurableWallTime: ISODate('2025-11-23T22:05:55.356Z'),
    lastHeartbeat: ISODate('2025-11-23T22:06:39.098Z'),
    lastHeartbeatRecv: ISODate('2025-11-23T22:06:03.613Z'),
    pingMs: Long('0'),
    lastHeartbeatMessage: "Couldn't get a connection within the time limit",
    syncSourceHost: '',
    syncSourceId: -1,
    infoMessage: '',
    configVersion: 1,
    configTerm: 1
  },
  {
    _id: 1,
    name: 'mongo2:27017',
    health: 1,
    state: 1,
    stateStr: 'PRIMARY',
    uptime: 1298,
    optime: { ts: Timestamp({ t: 1763935603, i: 1 }), t: Long('2') },
    optimeDate: ISODate('2025-11-23T22:06:43.000Z'),
    lastAppliedWallTime: ISODate('2025-11-23T22:06:43.101Z'),
    lastDurableWallTime: ISODate('2025-11-23T22:06:43.101Z'),
    syncSourceHost: '',
    syncSourceId: -1,
    infoMessage: '',
    electionTime: Timestamp({ t: 1763935563, i: 1 }),
    electionDate: ISODate('2025-11-23T22:06:03.000Z'),
    configVersion: 1,
    configTerm: 2,
```

Поки стара Primary (mongo1) мертва, запишемо нові дані на нову Primary (mongo2). Це ті дані, які mongo1 пропустить.

```
rs0 [direct: primary] test> db.exp1.insertOne({
...   status: "written_during_outage",
...   ts: new Date(),
...   note: "mongo1 was down"
... })
{
  acknowledged: true,
  insertedId: ObjectId('692385ba38811d86fdce5f47')
}
```

Запустимо стару Primary (docker start mongo1) та після запуска перевіряємо чи відбулася реплікація нових даних після відновлення роботи. Як бачимо mongo1 стала Secondary та вона реплікувала нові дані.

```
D:\University\semestr_7\High-Load_Systems\lab4>docker exec -it mongo1 mongosh
Current Mongosh Log ID: 69238601626bf12502ce5f46
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeou
Using MongoDB:          6.0.26
Using Mongosh:          2.5.8

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-11-23T22:08:17.796+00:00: Using the XFS filesystem is strongly recommended with the Wi
   2025-11-23T22:08:19.226+00:00: Access control is not enabled for the database. Read and wri
   2025-11-23T22:08:19.226+00:00: vm.max_map_count is too low
------

rs0 [direct: secondary] test> db.exp1.find({ status: "written_during_outage" })
[
  {
    _id: ObjectId('692385ba38811d86fdce5f47'),
    status: 'written_during_outage',
    ts: ISODate('2025-11-23T22:07:54.949Z'),
    note: 'mongo1 was down'
  }
]
rs0 [direct: secondary] test>
```

2 Частина 2

2.1 Запуск Python файла

```
(venv7s) D:\University\semestr_7\High-Load_Systems\lab4>docker exec -it python_client python main.
Виберіть тест (1-4):
1. WC=1 (Нормальний режим)
2. WC=majority (Нормальний режим)
3. WC=1 + Fail (Треба вручну вимкнути Primary!)
4. WC=majority + Fail (Треба вручну вимкнути Primary!)
```

2.2 Експеримент 1

```
============================================================
ЗАПУСК ТЕСТУ: WriteConcern = 1 (Normal)
Write Concern: 1
============================================================
Лічильник скинуто до 0.

--- РЕЗУЛЬТАТИ (WriteConcern = 1 (Normal)) ---
Час виконання: 135.90 сек
Кінцеве значення: 100000
Очікувано: 100000
УСПІХ: Дані цілісні.
```

2.3 Експеримент 2

```
============================================================
ЗАПУСК ТЕСТУ: WriteConcern = Majority (Normal)
Write Concern: majority
============================================================
Лічильник скинуто до 0.

--- РЕЗУЛЬТАТИ (WriteConcern = Majority (Normal)) ---
Час виконання: 316.06 сек
Кінцеве значення: 100000
Очікувано: 100000
УСПІХ: Дані цілісні.
```

2.4 Експеримент 3

```
============================================================
ЗАПУСК ТЕСТУ: WriteConcern = 1 (With FAILURE)
Write Concern: 1
============================================================
Лічильник скинуто до 0.

--- РЕЗУЛЬТАТИ (WriteConcern = 1 (With FAILURE)) ---
Час виконання: 236.36 сек
Кінцеве значення: 99997
Очікувано: 100000
ПОМИЛКА: Втрачено 3 оновлень!
```

Видно, що трохи данних загубилось через відключення Primary ноди.

2.5 Експеримент 4

```
============================================================
ЗАПУСК ТЕСТУ: WriteConcern = Majority (With FAILURE)
Write Concern: majority
============================================================
Лічильник скинуто до 0.

--- РЕЗУЛЬТАТИ (WriteConcern = Majority (With FAILURE)) ---
Час виконання: 363.34 сек
Кінцеве значення: 100000
Очікувано: 100000
УСПІХ: Дані цілісні.
```

Видно, що writeConcern = majority запобігло проблемі що виникла в третьому експерименті.

3 Код

Посилання на репозиторій: GitHub

```python
import time
import threading
from concurrent.futures import ThreadPoolExecutor
from pymongo import MongoClient, WriteConcern
from pymongo.errors import AutoReconnect, PyMongoError


URI = "mongodb://mongo1:27017,mongo2:27017,mongo3:27017/?replicaSet=rs0"

TOTAL_CLIENTS = 10
INCREMENTS_PER_CLIENT = 10000
DOC_ID = "likes_counter"

def worker_task(client_id, wc_mode):

    client = MongoClient(URI)

    wc = WriteConcern(w=wc_mode, wtimeout=5000)

    db = client.get_database("lab4_db", write_concern=wc)
    collection = db.likes

    for i in range(INCREMENTS_PER_CLIENT):
        collection.find_one_and_update(
            {"_id": DOC_ID},
            {"$inc": {"count": 1}},
            upsert=True
        )


    client.close()

```

```
def run_experiment(wc_mode, description):
    print(f"\n{'='*60}")
    print(f"ЗАПУСК ТЕСТУ: {description}")
    print(f"Write Concern: {wc_mode}")
    print(f"{'='*60}")

    init_client = MongoClient(URI)
    init_coll = init_client.get_database("lab4_db", write_concern=
       WriteConcern("majority")).likes
    init_coll.delete_many({})
    init_coll.insert_one({"_id": DOC_ID, "count": 0})
    print("Лічильник скинутодо  0.")
    init_client.close()

    start_time = time.time()

    with ThreadPoolExecutor(max_workers=TOTAL_CLIENTS) as executor:
        futures = [executor.submit(worker_task, i, wc_mode) for i in
           range(TOTAL_CLIENTS)]

        for future in futures:
            future.result()

    duration = time.time() - start_time

    check_client = MongoClient(URI)
    final_doc = check_client.lab4_db.likes.find_one({"_id": DOC_ID})
    check_client.close()

    final_count = final_doc['count']
    expected = TOTAL_CLIENTS * INCREMENTS_PER_CLIENT

    print(f"\n--- РЕЗУЛЬТАТИ ({description}) ---")
    print(f"Час виконання: {duration:.2f} сек")
    print(f"Кінцеве значення: {final_count}")
    print(f"Очікувано: {expected}")

    if final_count == expected:
        print("УСПІХ: Даніцілісні .")
    else:
        print(f"ПОМИЛКА: Втрачено {expected - final_count} оновлень!")

if __name__ == "__main__":
    print("Виберіть тест (1-4):")
    print("1. WC=1 Нормальний( режим)")
    print("2. WC=majority Нормальний( режим)")
    print("3. WC=1 + Fail Треба( вручнувимкнути  Primary!)")
    print("4. WC=majority + Fail Треба( вручнувимкнути  Primary!)")

    choice = input("Ваш вибір: ")

    if choice == '1':
        run_experiment(1, "WriteConcern = 1 (Normal)")
```

```
84      elif choice == '2':
85          run_experiment("majority", "WriteConcern = Majority (Normal)")
86      elif choice == '3':
87          print("\n!!! УВАГА: Підчасвиконанняскриптавиконайте     'docker
               stop <primary_node>' уіншомувікні   !\n")
88          time.sleep(3)
89          run_experiment(1, "WriteConcern = 1 (With FAILURE)")
90      elif choice == '4':
91          print("\n!!! УВАГА: Підчасвиконанняскриптавиконайте     'docker
               stop <primary_node>' уіншомувікні   !\n")
92          time.sleep(3)
93          run_experiment("majority", "WriteConcern = Majority (With FAILURE
               )")
```