Міністерство освіти і науки України Національний технічний університет України «Київський політехнічний інститут ім. І. Сікорського » Фізико-технічний інститут

Проектування високонавантажених систем

Лабораторна робота №4

Налаштування реплікації та перевірка відмовостійкості MongoDB

Виконав:

Студент групи ФБ-41мп

Заріцький О. В.

Київ 2025 р.

Налаштування реплікації

compose.yml

```
services:
  mongodb-node1:
    image: mongo:latest
    container_name: mongodb-node1
    ports:
      - "27017:27017"
    networks:
      - mongo-cluster
    command: ["mongod", "--replSet", "lab4ReplicaSet", "--bind_ip", "localhost,mongodb-
node1", "--setParameter", "diagnosticDataCollectionEnabled=false"]

  mongodb-node2:
    image: mongo:latest
    container_name: mongodb-node2
    ports:
      - "27018:27017"
    networks:
      - mongo-cluster
    command: ["mongod", "--replSet", "lab4ReplicaSet", "--bind_ip", "localhost,mongodb-
node2", "--setParameter", "diagnosticDataCollectionEnabled=false"]

  mongodb-node3:
    image: mongo:latest
    container_name: mongodb-node3
    ports:
      - "27019:27017"
    networks:
      - mongo-cluster
    command: ["mongod", "--replSet", "lab4ReplicaSet", "--bind_ip", "localhost,mongodb-
node3", "--setParameter", "diagnosticDataCollectionEnabled=false"]

networks:
  mongo-cluster:
    driver: bridge
```

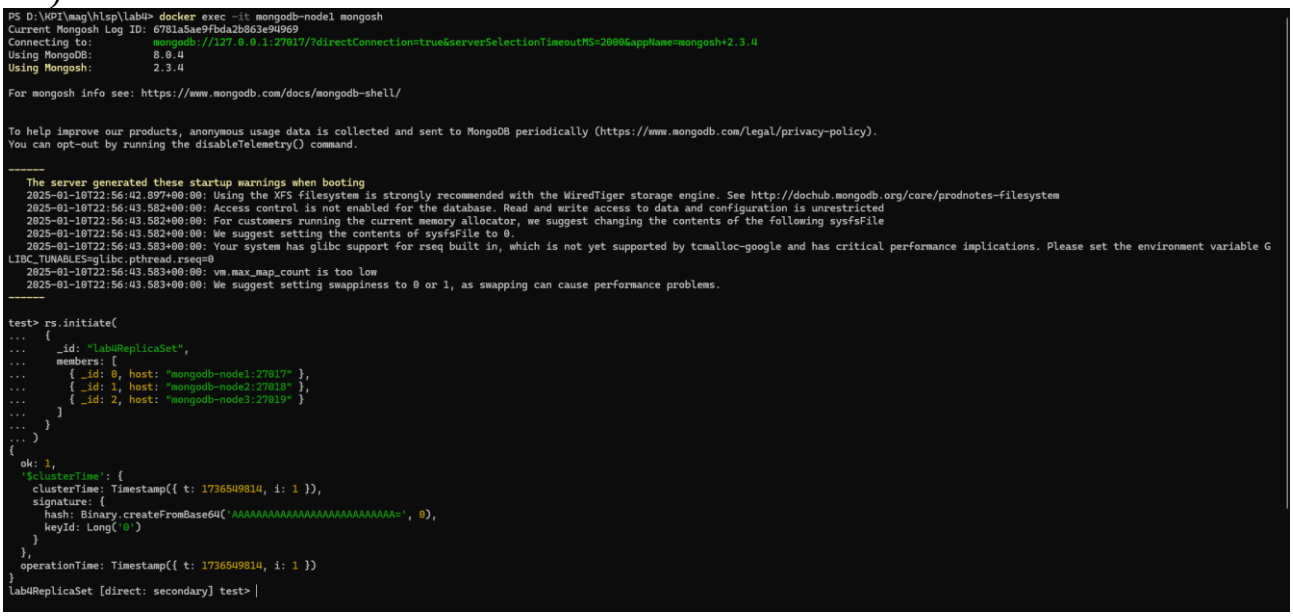
docker ps

```
PS D:\KPI\mag\hlsp\lab4> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                      NAMES
899ef11ad8e2   mongo:latest   "docker-entrypoint.s…"   8 minutes ago   Up 8 minutes   0.0.0.0:27017->27017/tcp   mongodb-node1
1e41c59b1041   mongo:latest   "docker-entrypoint.s…"   8 minutes ago   Up 8 minutes   0.0.0.0:27019->27017/tcp   mongodb-node3
01c0149d0d6a   mongo:latest   "docker-entrypoint.s…"   8 minutes ago   Up 8 minutes   0.0.0.0:27018->27017/tcp   mongodb-node2
PS D:\KPI\mag\hlsp\lab4>
```

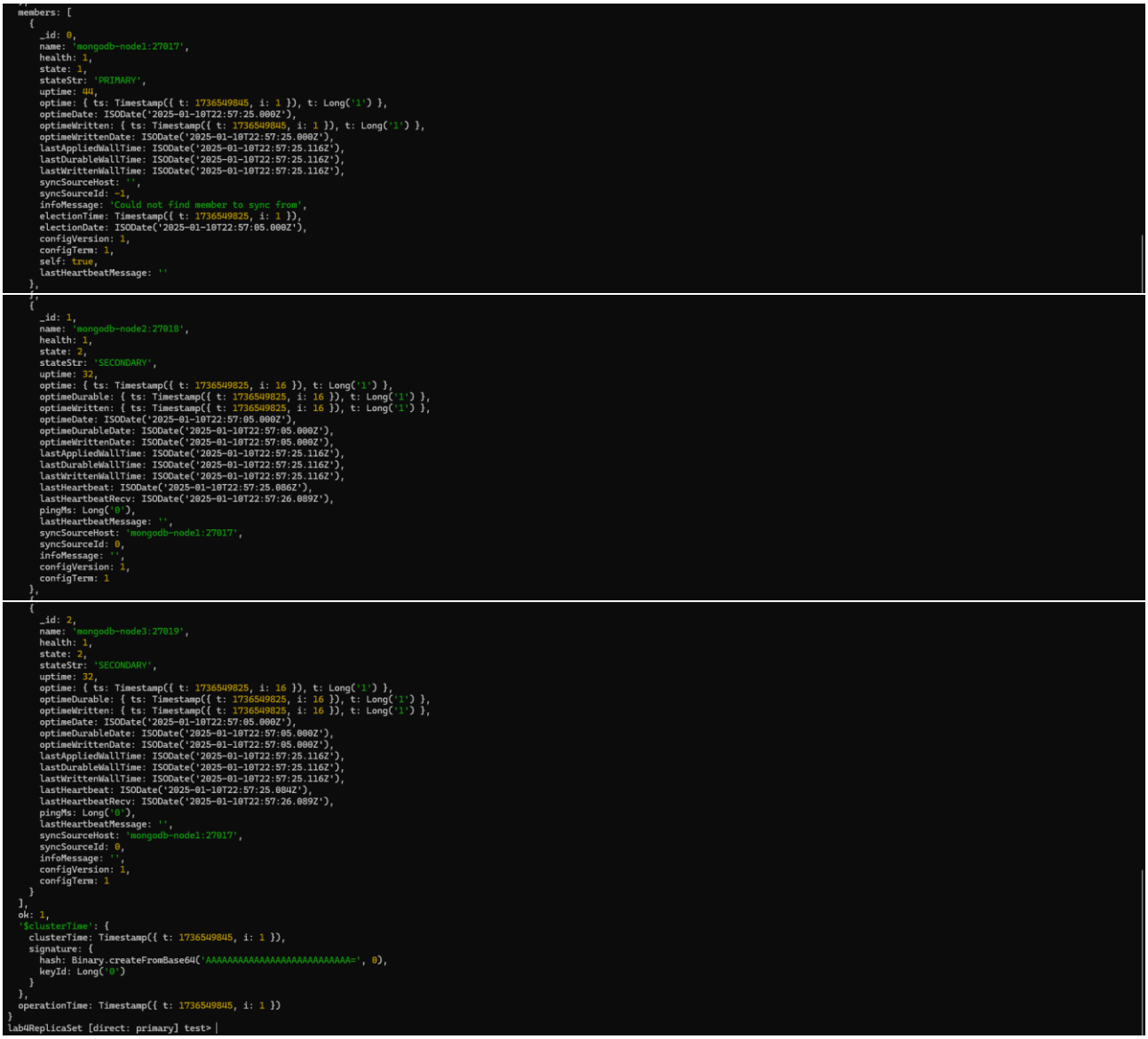
1. Налаштувати реплікацію в конфігурації: Primary with Two Secondary Members (P-S-S) (всі ноди можуть бути запущені як окремі процеси або у Docker контейнерах)

docker exec -it mongodb-node1 mongosh

```
rs.initiate(
  {
    _id: "lab4ReplicaSet",
    members: [
      { _id: 0, host: "mongodb-node1:27017" },
      { _id: 1, host: "mongodb-node2:27018" },
      { _id: 2, host: "mongodb-node3:27019" }
    ]
  }
)
```



rs.status()



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

Вимкнена нода

```
PS D:\KPI\mag\hlsp\lab4> docker stop mongodb-node1
mongodb-node1
PS D:\KPI\mag\hlsp\lab4> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                     NAMES
b09152e2c063   mongo:latest   "docker-entrypoint.s…"   10 minutes ago   Up 10 minutes   0.0.0.0:27018->27017/tcp   mongodb-node2
6d74bf198584   mongo:latest   "docker-entrypoint.s…"   10 minutes ago   Up 10 minutes   0.0.0.0:27019->27017/tcp   mongodb-node3
PS D:\KPI\mag\hlsp\lab4>
```

db.test.insertOne({ name: "test1_mngOff" },{ writeConcern: { w: 3 }, wtimeout: 0 })

```
lab4ReplicaSet [direct: primary] test> db.test.insertOne( { name: "test1_mngOff" }, { writeConcern: { w: 3 }, wtimeout: 0 } )
```

Увімкнена нода

```
PS D:\KPI\mag\hlsp\lab4> docker start mongodb-node1
mongodb-node1
PS D:\KPI\mag\hlsp\lab4> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                      NAMES
d42b073063a6   mongo:latest   "docker-entrypoint.s…"   12 minutes ago   Up 2 seconds    0.0.0.0:27017->27017/tcp   mongodb-node1
b09152e2c063   mongo:latest   "docker-entrypoint.s…"   12 minutes ago   Up 12 minutes   0.0.0.0:27018->27017/tcp   mongodb-node2
6d74bf198584   mongo:latest   "docker-entrypoint.s…"   12 minutes ago   Up 12 minutes   0.0.0.0:27019->27017/tcp   mongodb-node3
PS D:\KPI\mag\hlsp\lab4>
```

db.test.insertOne({ name: "test1_mngOff" },{ writeConcern: { w: 3 }, wtimeout: 0 })

```
lab4ReplicaSet [direct: primary] test> db.test.insertOne( { name: "test1_mngOff" }, { writeConcern: { w: 3 }, wtimeout: 0 } )
{
  acknowledged: true,
  insertedId: ObjectId('67803165e5c581db7ee9496c')
}
lab4ReplicaSet [direct: primary] test>
```

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”

```
PS D:\KPI\mag\hlsp\lab4> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                      NAMES
b09152e2c063   mongo:latest   "docker-entrypoint.s…"   40 minutes ago   Up 40 minutes   0.0.0.0:27018->27017/tcp   mongodb-node2
6d74bf198584   mongo:latest   "docker-entrypoint.s…"   40 minutes ago   Up 40 minutes   0.0.0.0:27019->27017/tcp   mongodb-node3
PS D:\KPI\mag\hlsp\lab4>
```

db.test.insertOne({ name: " test3_mngOff " },{ writeConcern: { w: 3 }, wtimeout: 5 })

```
lab4ReplicaSet [direct: primary] test> db.test.insertOne({ name: " test3_mngOff " },{ writeConcern: { w: 3 }, wtimeout: 5 })
^CStopping execution...
```

```
lab4ReplicaSet [direct: primary] test> db.test.find().readConcern("majority")
[
  { _id: ObjectId('6780311ae5c581db7ee9496a'), name: 'test1_mngOff' },
  { _id: ObjectId('6780311ce5c581db7ee9496b'), name: 'test1_mngOff' },
  { _id: ObjectId('67803165e5c581db7ee9496c'), name: 'test1_mngOff' },
  { _id: ObjectId('678033a7e5c581db7ee9496d'), name: ' test2_mngOff ' },
  { _id: ObjectId('678038b1adda0d980ae9496a'), name: ' test3_mngOff ' },
  { _id: ObjectId('678038bdadda0d980ae9496b'), name: ' test3_mngOff ' },
  { _id: ObjectId('6780399eadda0d980ae9496c'), name: ' test3_mngOff ' },
  { _id: ObjectId('678039dcadda0d980ae9496d'), name: ' test3_mngOff ' },
  { _id: ObjectId('678039e6adda0d980ae9496e'), name: ' test3_mngOff ' }
]
lab4ReplicaSet [direct: primary] test>
```

4. Продемонструвати 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.

```
PS D:\KPI\mag\hlsp\lab4> docker stop mongodb-node3
mongodb-node3
PS D:\KPI\mag\hlsp\lab4> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS                      NAMES
d42b073063a6   mongo:latest   "docker-entrypoint.s…"   53 minutes ago   Up 25 seconds   0.0.0.0:27017->27017/tcp   mongodb-node1
b09152e2c063   mongo:latest   "docker-entrypoint.s…"   53 minutes ago   Up 53 minutes   0.0.0.0:27018->27017/tcp   mongodb-node2
PS D:\KPI\mag\hlsp\lab4>
```

```
PS D:\KPI\mag\hlsp\lab4> docker exec -it mongodb-node3 mongosh
Error response from daemon: container 6d74bf198584483d03d23c35c27faad3094d45c2dcdf7819397adcc8f2a3d21f0 is not running
PS D:\KPI\mag\hlsp\lab4> docker exec -it mongodb-node1 mongosh
Current Mongosh Log ID: 67803b7728a9b1f9d0e94969
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.4
Using MongoDB:          8.0.4
Using Mongosh:          2.3.4

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

------
   The server generated these startup warnings when booting
   2025-01-09T21:09:02.992+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
   2025-01-09T21:09:03.706+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2025-01-09T21:09:03.706+00:00: For customers running the current memory allocator, we suggest changing the contents of the following sysfsFile
   2025-01-09T21:09:03.706+00:00: We suggest setting the contents of sysfsFile to 0.
   2025-01-09T21:09:03.706+00:00: Your system has glibc support for rseq built in, which is not yet supported by tcmalloc-google and has critical performance implications. Please set the environment variable G
LIBC_TUNABLES=glibc.pthread.rseq=0
   2025-01-09T21:09:03.706+00:00: vm.max_map_count is too low
   2025-01-09T21:09:03.706+00:00: We suggest setting swappiness to 0 or 1, as swapping can cause performance problems.
------

lab4ReplicaSet [direct: primary] test> rs.status()
{
  set: 'lab4ReplicaSet',
  date: ISODate('2025-01-09T21:11:24.378Z'),
  myState: 1,
  term: Long('2'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
  members: [
    {
      _id: 0,
      name: 'mongodb-node1:27017',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 142,
      optime: { ts: Timestamp({ t: 1736457084, i: 1 }), t: Long('2') },
      optimeDate: ISODate('2025-01-09T21:11:24.000Z'),
      optimeWritten: { ts: Timestamp({ t: 1736457084, i: 1 }), t: Long('2') },
      optimeWrittenDate: ISODate('2025-01-09T21:11:24.000Z'),
      lastAppliedWallTime: ISODate('2025-01-09T21:11:24.355Z'),
      lastDurableWallTime: ISODate('2025-01-09T21:11:24.355Z'),
      lastWrittenWallTime: ISODate('2025-01-09T21:11:24.355Z'),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1736456954, i: 1 }),
      electionDate: ISODate('2025-01-09T21:09:14.000Z'),
      configVersion: 1,
      configTerm: 2,
      self: true,
      lastHeartbeatMessage: ''
    },
    {
      _id: 1,
      name: 'mongodb-node2:27017',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 140,
      optime: { ts: Timestamp({ t: 1736457074, i: 1 }), t: Long('2') },
      optimeDurable: { ts: Timestamp({ t: 1736457074, i: 1 }), t: Long('2') },
      optimeWritten: { ts: Timestamp({ t: 1736457074, i: 1 }), t: Long('2') },
      optimeDate: ISODate('2025-01-09T21:11:14.000Z'),
      optimeDurableDate: ISODate('2025-01-09T21:11:14.000Z'),
      optimeWrittenDate: ISODate('2025-01-09T21:11:14.000Z'),
      lastAppliedWallTime: ISODate('2025-01-09T21:11:24.355Z'),
      lastDurableWallTime: ISODate('2025-01-09T21:11:24.355Z'),
      lastWrittenWallTime: ISODate('2025-01-09T21:11:24.355Z'),
      lastHeartbeat: ISODate('2025-01-09T21:11:22.397Z'),
      lastHeartbeatRecv: ISODate('2025-01-09T21:11:22.930Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: 'mongodb-node1:27017',
      syncSourceId: 0,
      infoMessage: '',
      configVersion: 1,
      configTerm: 2
    },
  {
    _id: 2,
    name: 'mongodb-node3:27017',
    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') },
    optimeWritten: { 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'),
    optimeWrittenDate: ISODate('1970-01-01T00:00:00.000Z'),
    lastAppliedWallTime: ISODate('2025-01-09T21:09:24.355Z'),
    lastDurableWallTime: ISODate('2025-01-09T21:09:24.355Z'),
    lastWrittenWallTime: ISODate('2025-01-09T21:09:24.355Z'),
    lastHeartbeat: ISODate('2025-01-09T21:11:24.362Z'),
    lastHeartbeatRecv: ISODate('2025-01-09T21:09:22.867Z'),
    pingMs: Long('0'),
    lastHeartbeatMessage: "Couldn't get a connection within the time limit",
    syncSourceHost: '',
    syncSourceId: -1,
    infoMessage: '',
    configVersion: 1,
    configTerm: 2
  }
```

db.test.insertOne({ name: " test4_mngOff " },{ writeConcern: { w: 3 }, wtimeout: 1000 })

```
lab4ReplicaSet [direct: primary] test> db.test.insertOne({ name: " test4_mngOff " },{ writeConcern: { w: 3 }, wtimeout: 1000 })
^CStopping execution...
```

```
PS D:\KPI\mag\hlsp\lab4> docker start mongodb-node3
mongodb-node3
PS D:\KPI\mag\hlsp\lab4> docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED             STATUS             PORTS                      NAMES
d42b073063a6   mongo:latest   "docker-entrypoint.s…"   About an hour ago   Up 13 minutes      0.0.0.0:27017->27017/tcp   mongodb-node1
b09152e2c063   mongo:latest   "docker-entrypoint.s…"   About an hour ago   Up About an hour   0.0.0.0:27018->27017/tcp   mongodb-node2
6d74bf198584   mongo:latest   "docker-entrypoint.s…"   About an hour ago   Up 1 second        0.0.0.0:27019->27017/tcp   mongodb-node3
PS D:\KPI\mag\hlsp\lab4>
```

```
lab4ReplicaSet [direct: primary] test> db.test.insertOne({ name: " test4_mngOff " },{ writeConcern: { w: 3 }, wtimeout: 1000 })
{
  acknowledged: true,
  insertedId: ObjectId('67803e7c28a9b1f9d0e9496b')
}
```

db.test.find()

```
lab4ReplicaSet [direct: primary] test> db.test.find()
[
  { _id: ObjectId('6780311ce5c581db7ee9496b'), name: 'test1_mngOff' },
  { _id: ObjectId('6780311ae5c581db7ee9496a'), name: 'test1_mngOff' },
  { _id: ObjectId('67803165e5c581db7ee9496c'), name: 'test1_mngOff' },
  { _id: ObjectId('678033a7e5c581db7ee9496d'), name: ' test2_mngOff ' },
  { _id: ObjectId('6780399eadda0d980ae9496c'), name: ' test3_mngOff ' },
  { _id: ObjectId('678038bdadda0d980ae9496b'), name: ' test3_mngOff ' },
  { _id: ObjectId('678039e6adda0d980ae9496e'), name: ' test3_mngOff ' },
  { _id: ObjectId('678038b1adda0d980ae9496a'), name: ' test3_mngOff ' },
  { _id: ObjectId('678039dcadda0d980ae9496d'), name: ' test3_mngOff ' },
  { _id: ObjectId('67803dad28a9b1f9d0e9496a'), name: ' test4_mngOff ' },
  { _id: ObjectId('67803e7c28a9b1f9d0e9496b'), name: ' test4_mngOff ' }
]
lab4ReplicaSet [direct: primary] test>
```

Аналіз продуктивності та перевірка цілісності

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

db.createCollection("users")

db.users.insertOne({ user_name: "User1", likes_count: 0 })

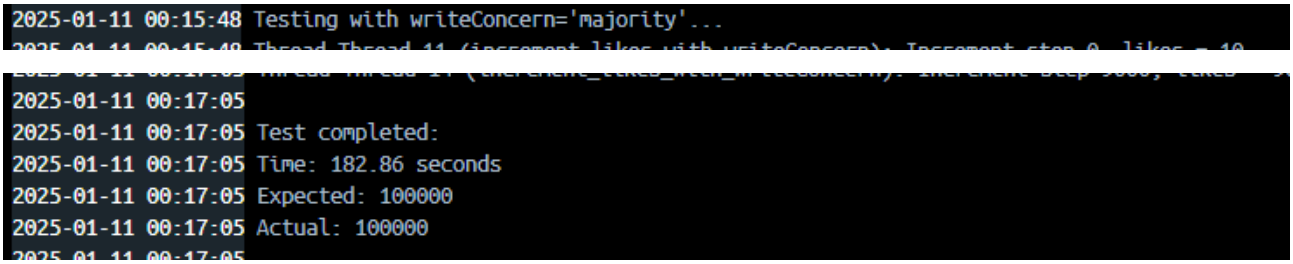
```
URI = "mongodb://mongodb-node1:27017,mongodb-node2:27018,mongodb-node3:27019/?replicaSet=lab4ReplicaSet"
client = MongoClient(URI)

db = client["lab4ReplicaSet"]
collection = db["counter"]
```

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

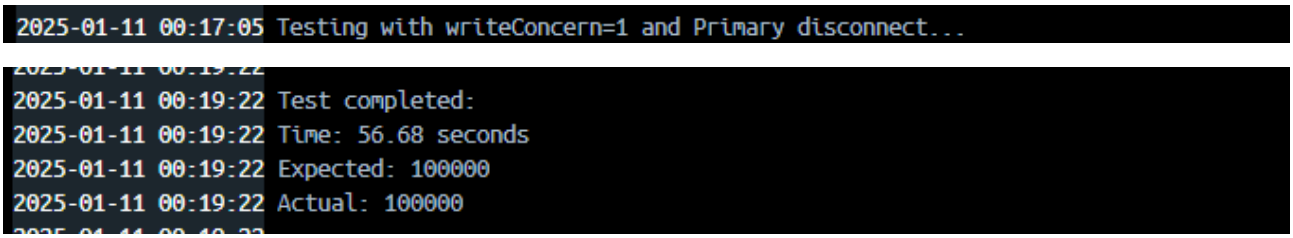
```
2025-01-11 00:13:16 Testing with writeConcern=1...
```

```
2025-01-11 00:15:48
2025-01-11 00:15:48 Test completed:
2025-01-11 00:15:48 Time: 59.79 seconds
2025-01-11 00:15:48 Expected: 100000
2025-01-11 00:15:48 Actual: 100000
```

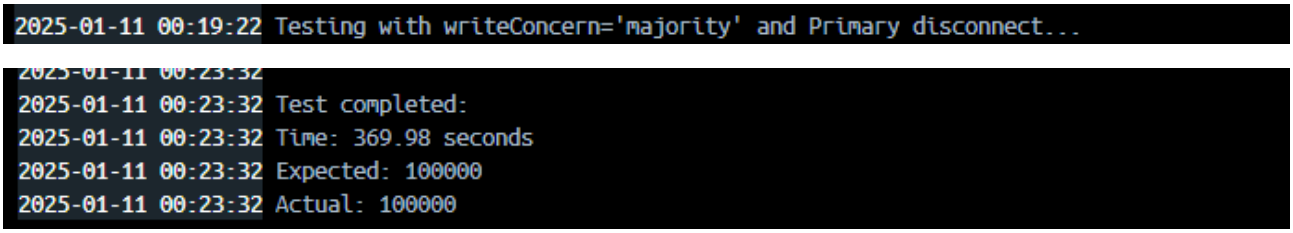
2. Вказавши у парметрах findOneAndUpdate writeConcern = majority (це буде означати, що Primary чекає поки значення запишется на більшість нод), запустіть 10 клієнтів з інкрементом по 10_000 на кожному з них. Виміряйте час виконання та перевірте чи кінцеве значення буде дорівнювати очікуваному - 100К



3. 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.



4. 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.



Код виконання:

```
import time
import threading
from pymongo import MongoClient, WriteConcern

URI = "mongodb://mongodb-node1:27017,mongodb-node2:27018,mongodb-
node3:27019/?replicaSet=lab4ReplicaSet"
client = MongoClient(URI)
```

```python
db = client["lab4ReplicaSet"]
collection = db["counter"]

increment_value = 10000
num_threads = 10
user_name = "User1"

def initialize_counter():
    collection.find_one_and_update(
        {"user_name": user_name},
        {"$set": {"likes_count": 0}},
        upsert=True
    )

def increment_likes_with_writeConcern(user_name, increment_value, write_concern_level):
    write_concern = WriteConcern(write_concern_level)
    for i in range(increment_value):
        collection.with_options(write_concern=write_concern).find_one_and_update(
            {"user_name": user_name},
            {"$inc": {"likes_count": 1}},
            return_document=True
        )

        if i % 1000 == 0:
            current_likes = collection.find_one({"user_name": user_name})["likes_count"]
            print(f"Thread {threading.current_thread().name}: Increment step {i}, likes =
{current_likes}")

def run_test(write_concern_level, disconnect_primary=False):
    initialize_counter()
    threads = []

    start_time = time.time()
    for _ in range(num_threads):
        thread = threading.Thread(
            target=increment_likes_with_writeConcern,
            args=(user_name, increment_value, write_concern_level)
        )
        threads.append(thread)
        thread.start()

    if disconnect_primary:
        print("Disconnecting Primary...")
        time.sleep(2)

    for thread in threads:
        thread.join()

    end_time = time.time()

    user = collection.find_one({"user_name": user_name})
    print(f"\nTest completed:")
    print(f"Time: {end_time - start_time:.2f} seconds")
    print(f"Expected: {num_threads * increment_value}")
    print(f"Actual: {user['likes_count']}")

if __name__ == "__main__":
    print("Testing with writeConcern=1...")
    run_test(1)

    print("\nTesting with writeConcern='majority'...")
    run_test("majority")

    print("\nTesting with writeConcern=1 and Primary disconnect...")
    run_test(1, disconnect_primary=True)

    print("\nTesting with writeConcern='majority' and Primary disconnect...")
    run_test("majority", disconnect_primary=True)
```