МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ім. ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
КАФЕДРА ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

Проектування високонавантажених систем

Лабораторна робота №4

Налаштування реплікації та перевірка відмовостійкості MongoDB

Перевірив:
Родіонов А. М.

Виконав:
студент І курсу
групи ФБ-41мп
Сахній Н. Р.

Київ 2024

Мета роботи: Налаштувати реплікацію в MongoDB для забезпечення відмовостійкості, перевірити її функціональність у різних сценаріях, а також потрібно реалізувати механізм одночасного оновлення лічильника з використанням функції "`findOneAndUpdate`" та оцінити його ефективність.

Завдання до виконання:

0. INITIALIZATION of MongoDB:

1) `cat ./docker-compose.yml`

```yaml
services:
    # MongoDB Node 1
    mongodb-node1:
        ports:
            - "27017:27017"
        container_name: mongodb-node1
        networks:
            - "mongodb-network"
        image: mongo:latest
        command: mongod --replSet labReplicaSet --bind_ip_all

    # MongoDB Node 2
    mongodb-node2:
        ports:
            - "27018:27017"
        container_name: mongodb-node2
        networks:
            - "mongodb-network"
        image: mongo:latest
        command: mongod --replSet labReplicaSet --bind_ip_all

    # MongoDB Node 3
    mongodb-node3:
        ports:
            - "27019:27017"
        container_name: mongodb-node3
        networks:
            - "mongodb-network"
        image: mongo:latest
        command: mongod --replSet labReplicaSet --bind_ip_all

networks:
    mongodb-network:
        driver: bridge
```

2) `docker ps`

```
t-1000@DESKTOP-DRIOPBB MINGW64 /d/KPI/5 курс Магістрат/Проектування ВС/Task_4-MongoDB_Replication
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                      NAMES
cdceed2255bf   mongo:latest   "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   0.0.0.0:27017->27017/tcp   mongodb-node1
377cd70d39f1   mongo:latest   "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   0.0.0.0:27018->27017/tcp   mongodb-node2
93cfa71ba33d   mongo:latest   "docker-entrypoint.s…"   4 minutes ago   Up 4 minutes   0.0.0.0:27019->27017/tcp   mongodb-node3
```

I. SETUP of REPLICATION

1) Налаштувати реплікацію в конфігурації: “Primary with Two Secondary Members” (**P-S-S**) (всі ноди будуть запущені в **docker**-контейнерах)

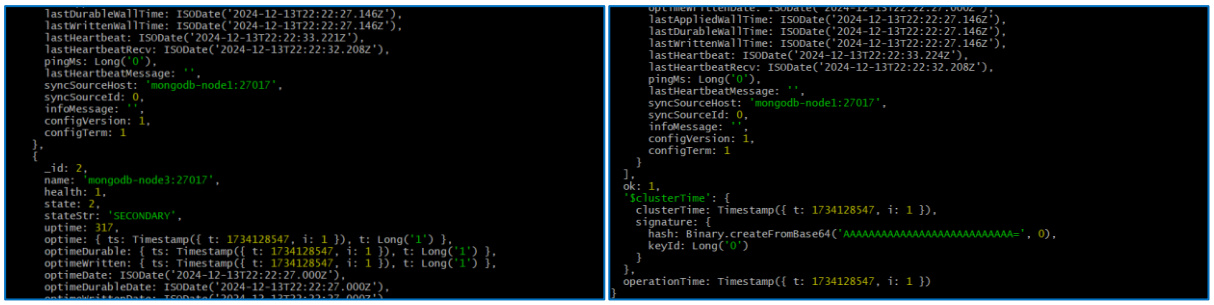
- `rs.initiate( { _id: "labReplicaSet", members: [ { _id: 0, host: "mongodb-node1:27017" }, { _id: 1, host: "mongodb-node2:27017" }, { _id: 2, host: "mongodb-node3:27017" } ] } )`

```
t-1000@DESKTOP-DRIOPBB MINGW64 /d/KPI/5 курс Магістрат/Проектування ВС/Task_4-MongoDB_Replication
$ winpty docker exec -it mongodb-node1 bash
root@cdceed2255bf:/# mongosh
Current Mongosh Log ID: 675cb18f5545e09228e94969
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
   2024-12-13T22:01:22.402+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http:
   2024-12-13T22:01:22.958+00:00: Access control is not enabled for the database. Read and write access to data and configuratio
   2024-12-13T22:01:22.958+00:00: For customers running the current memory allocator, we suggest changing the contents of the fo
   2024-12-13T22:01:22.958+00:00: We suggest setting the contents of sysfsFile to 0.
   2024-12-13T22:01:22.958+00:00: Your system has glibc support for rseq built in, which is not yet supported by tcmalloc-google
IBC_TUNABLES=glibc.pthread.rseq=0
   2024-12-13T22:01:22.958+00:00: vm.max_map_count is too low
   2024-12-13T22:01:22.958+00:00: We suggest setting swappiness to 0 or 1, as swapping can cause performance problems.
------

test> use lab
switched to db lab
lab> rs.initiate(
...   {
...     _id: "labReplicaSet",
...     members: [
...       { _id: 0, host: "mongodb-node1:27017" },
...       { _id: 1, host: "mongodb-node2:27017" },
...       { _id: 2, host: "mongodb-node3:27017" }
    ]
  }
...     ]
...   }
... )
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1734128236, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1734128236, i: 1 })
}
```

- `rs.status()`

```
labReplicaSet [direct: secondary] lab> rs.status()
{
  set: 'labReplicaSet',
  date: ISODate('2024-12-13T22:22:33.372Z'),
  myState: 1,
  term: Long('1'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1734128547, i: 1 }), t: Long('1') },
    lastCommittedWallTime: ISODate('2024-12-13T22:22:27.146Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1734128547, i: 1 }), t: Long('1') },
    appliedOpTime: { ts: Timestamp({ t: 1734128547, i: 1 }), t: Long('1') },
    durableOpTime: { ts: Timestamp({ t: 1734128547, i: 1 }), t: Long('1') },
    writtenOpTime: { ts: Timestamp({ t: 1734128547, i: 1 }), t: Long('1') },
    lastAppliedWallTime: ISODate('2024-12-13T22:22:27.146Z'),
    lastDurableWallTime: ISODate('2024-12-13T22:22:27.146Z'),
    lastWrittenWallTime: ISODate('2024-12-13T22:22:27.146Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1734128527, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2024-12-13T22:17:27.025Z'),
    electionTerm: Long('1'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1734128236, i: 1 }), t: Long('-1') },
    lastSeenWrittenOpTimeAtElection: { ts: Timestamp({ t: 1734128236, i: 1 }), t: Long('-1') },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1734128236, i: 1 }), t: Long('-1') },
    numVotesNeeded: 2,
    priorityAtElection: 1,
    electionTimeoutMillis: Long('10000'),
    numCatchUpOps: Long('0'),
    newTermStartDate: ISODate('2024-12-13T22:17:27.101Z'),
    wMajorityWriteAvailabilityDate: ISODate('2024-12-13T22:17:27.559Z')
  },
```

```
  members: [
    {
      _id: 0,
      name: 'mongodb-node1:27017',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 1271,
      optime: { ts: Timestamp({ t: 1734128547, i: 1 }), t: Long('1') },
      optimeDate: ISODate('2024-12-13T22:22:27.000Z'),
      optimeWritten: { ts: Timestamp({ t: 1734128547, i: 1 }), t: Long('1') },
      optimeWrittenDate: ISODate('2024-12-13T22:22:27.000Z'),
      lastAppliedWallTime: ISODate('2024-12-13T22:22:27.146Z'),
      lastDurableWallTime: ISODate('2024-12-13T22:22:27.146Z'),
      lastWrittenWallTime: ISODate('2024-12-13T22:22:27.146Z'),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1734128247, i: 1 }),
      electionDate: ISODate('2024-12-13T22:17:27.000Z'),
      configVersion: 1,
      configTerm: 1,
      self: true,
      lastHeartbeatMessage: ''
    },
    {
      _id: 1,
      name: 'mongodb-node2:27017',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 317,
      optime: { ts: Timestamp({ t: 1734128547, i: 1 }), t: Long('1') },
      optimeDurable: { ts: Timestamp({ t: 1734128547, i: 1 }), t: Long('1') },
      optimeWritten: { ts: Timestamp({ t: 1734128547, i: 1 }), t: Long('1') },
      optimeDate: ISODate('2024-12-13T22:22:27.000Z'),
      optimeDurableDate: ISODate('2024-12-13T22:22:27.000Z'),
      optimeWrittenDate: ISODate('2024-12-13T22:22:27.000Z'),
      lastAppliedWallTime: ISODate('2024-12-13T22:22:27.146Z'),
```



2) Спробувати зробити запис з однією відключеною нодою та **writeConcern** рівним “**3**” та нескінченним таймаутом. Спробувати під час таймаута включити відключену ноду.

❖ Якщо нода вимкнена:

- `docker stop mongodb-node2 && docker ps`

```
t-1000@DESKTOP-DRIOPBB MINGW64 /d/KPI/5 курс Маістрат/Проектування ВС/Task_4-MongoDB_Replication
$ docker stop mongodb-node2
mongodb-node2

t-1000@DESKTOP-DRIOPBB MINGW64 /d/KPI/5 курс Маістрат/Проектування ВС/Task_4-MongoDB_Replication
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED             STATUS             PORTS                      NAMES
cdceed2255bf   mongo:latest   "docker-entrypoint.s…"   About an hour ago   Up About an hour   0.0.0.0:27017->27017/tcp   mongodb-node1
93cfa71ba33d   mongo:latest   "docker-entrypoint.s…"   About an hour ago   Up About an hour   0.0.0.0:27019->27017/tcp   mongodb-node3
```

- `db.MongoDB_DocS.insertOne( { name: 'mongoff' }, { writeConcern: { w: 3 }, wtimeout: 0 } )`

```
labReplicaSet [direct: primary] lab> db.MongoDB_DocS.insertOne( { name: 'mongoff' }, { writeConcern: { w: 3 }, wtimeout: 0 } )
... I'm hung up -_- ...
```

- `db.getCollection( "MongoDB_DocS" ).find( {} ).readPref( "primary" )`

```
labReplicaSet [direct: primary] lab> db.getCollection( "MongoDB_DocS" ).find( {} ).readPref( "primary" )
[ { _id: ObjectId('675cc0e06d85340012e9496c'), name: 'mongoff' } ]
```

❖ Якщо ввімкнути ноду:

- `docker start mongodb-node2 && docker ps`

```
t-1000@DESKTOP-DRIOPBB MINGW64 /d/KPI/5 курс Маістрат/Проектування ВС/Task_4-MongoDB_Replication
$ docker start mongodb-node2
mongodb-node2

t-1000@DESKTOP-DRIOPBB MINGW64 /d/KPI/5 курс Маістрат/Проектування ВС/Task_4-MongoDB_Replication
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED             STATUS             PORTS                      NAMES
cdceed2255bf   mongo:latest   "docker-entrypoint.s…"   About an hour ago   Up About an hour   0.0.0.0:27017->27017/tcp   mongodb-node1
377cd70d39f1   mongo:latest   "docker-entrypoint.s…"   About an hour ago   Up 3 seconds       0.0.0.0:27018->27017/tcp   mongodb-node2
93cfa71ba33d   mongo:latest   "docker-entrypoint.s…"   About an hour ago   Up About an hour   0.0.0.0:27019->27017/tcp   mongodb-node3
```

- `db.MongoDB_DocS.insertOne( { name: 'mongone' }, { writeConcern: { w: 3 }, wtimeout: 0 } )`

```
labReplicaSet [direct: primary] lab> db.MongoDB_DocS.insertOne( { name: 'mongone' }, { writeConcern: { w: 3 }, wtimeout: 0 } )
{
  acknowledged: true,
  insertedId: ObjectId('675cc18a6d85340012e9496d')
}
```

- `db.getCollection( "MongoDB_DocS" ).find( {} ).readPref( "primary" )`

```
labReplicaSet [direct: primary] lab> db.getCollection( "MongoDB_DocS" ).find( {} ).readPref( "primary" )
[
  { _id: ObjectId('675cc0e06d85340012e9496c'), name: 'mongoff' },
  { _id: ObjectId('675cc18a6d85340012e9496d'), name: 'mongone' }
]
```

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

- `$ docker stop mongodb-node2 && docker ps`

```
t-1000@DESKTOP-DRIOPBB MINGW64 /d/KPI/5 курс Магістрат/Проектування ВС/Task_4-MongoDB_Replication
$ docker stop mongodb-node2
mongodb-node2

t-1000@DESKTOP-DRIOPBB MINGW64 /d/KPI/5 курс Магістрат/Проектування ВС/Task_4-MongoDB_Replication
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED             STATUS             PORTS                     NAMES
cdceed2255bf   mongo:latest   "docker-entrypoint.s…"   About an hour ago   Up About an hour   0.0.0.0:27017->27017/tcp   mongodb-node1
93cfa71ba33d   mongo:latest   "docker-entrypoint.s…"   About an hour ago   Up About an hour   0.0.0.0:27019->27017/tcp   mongodb-node3
```

- `db.MongoDB_DocS.insertOne( { name: 'mongott' }, { writeConcern: { w: 3 }, wtimeout: 1000 } )`

```
labReplicaSet [direct: primary] lab> db.MongoDB_DocS.insertOne( { name: 'mongott' }, { writeConcern: { w: 3 }, wtimeout: 1000 } )
⨯ There should have been TimeoutError
```

- `db.getCollection("MongoDB_DocS").find({}).readConcern("majority")`

```
labReplicaSet [direct: primary] lab> db.getCollection("MongoDB_DocS").find({}).readConcern("majority")
[
  { _id: ObjectId('675cc0e06d85340012e9496c'), name: 'mongoff' },
  { _id: ObjectId('675cc18a6d85340012e9496d'), name: 'mongone' },
  { _id: ObjectId('675cc2736d85340012e9496e'), name: 'mongott' }
]
```

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.

- `docker stop mongodb-node1 && docker ps`

```
t-1000@DESKTOP-DRIOPBB MINGW64 /d/KPI/5 курс Маістрат/Проектування ВС/Task_4-MongoDB_Replication
$ docker stop mongodb-node1
mongodb-node1

t-1000@DESKTOP-DRIOPBB MINGW64 /d/KPI/5 курс Маістрат/Проектування ВС/Task_4-MongoDB_Replication
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED       STATUS         PORTS                     NAMES
377cd70d39f1   mongo:latest   "docker-entrypoint.s…"   2 hours ago   Up 8 minutes   0.0.0.0:27018->27017/tcp   mongodb-node2
93cfa71ba33d   mongo:latest   "docker-entrypoint.s…"   2 hours ago   Up 2 hours     0.0.0.0:27019->27017/tcp   mongodb-node3
```

- `winpty docker exec -it mongodb-node2 bash`

```
t-1000@DESKTOP-DRIOPBB MINGW64 /d/KPI/5 курс Маістрат/Проектування ВС/Task_4-MongoDB_Replication
$ winpty docker exec -it mongodb-node2 bash
root@377cd70d39f1:/# mongosh
Current Mongosh Log ID: 675cc671cedf113f90e94969
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/


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

------
   The server generated these startup warnings when booting
   2024-12-13T23:33:38.069+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http:
   2024-12-13T23:33:39.077+00:00: Access control is not enabled for the database. Read and write access to data and configuratio
   2024-12-13T23:33:39.077+00:00: For customers running the current memory allocator, we suggest changing the contents of the fo
   2024-12-13T23:33:39.077+00:00: We suggest setting the contents of sysfsFile to 0.
   2024-12-13T23:33:39.077+00:00: Your system has glibc support for rseq built in, which is not yet supported by tcmalloc-google
```

- `rs.status()`

```
  _id: 0,
  name: 'mongodb-node1:27017',
  health: 0,
  state: 8,
  stateStr: '(not reachable/healthy)',
  uptime: 0,
```

```
  _id: 1,
  name: 'mongodb-node2:27017',
  health: 1,
  state: 1,
  stateStr: 'PRIMARY',
  uptime: 555,
```

- `db.MongoDB_DocS.insertOne( { name: 'mongodb'}, { writeConcern: { j: true } } )`

```
labReplicaSet [direct: primary] lab> db.MongoDB_DocS.insertOne( { name: 'mongodb'}, { writeConcern: { j: true } } )
{
  acknowledged: true,
  insertedId: ObjectId('675ccd26c682b5733ae9496a')
}
```

- `docker start mongodb-node1 && docker ps`

```
t-1000@DESKTOP-DRIOPBB MINGW64 /d/KPI/5 курс Маістрат/Проектування ВС/Task_4-MongoDB_Replication
$ docker start mongodb-node1
mongodb-node1

t-1000@DESKTOP-DRIOPBB MINGW64 /d/KPI/5 курс Маістрат/Проектування ВС/Task_4-MongoDB_Replication
$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED       STATUS         PORTS                     NAMES
cdceed2255bf   mongo:latest   "docker-entrypoint.s…"   2 hours ago   Up 10 minutes   0.0.0.0:27017->27017/tcp   mongodb-node1
377cd70d39f1   mongo:latest   "docker-entrypoint.s…"   2 hours ago   Up 21 minutes   0.0.0.0:27018->27017/tcp   mongodb-node2
93cfa71ba33d   mongo:latest   "docker-entrypoint.s…"   2 hours ago   Up 21 minutes   0.0.0.0:27019->27017/tcp   mongodb-node3
```

- `db.MongoDB_DocS.find()`

```
labReplicaSet [direct: secondary] lab> db.MongoDB_DocS.find()
[
  { _id: ObjectId('675cc0e06d85340012e9496c'), name: 'mongoff' },
  { _id: ObjectId('675cc18a6d85340012e9496d'), name: 'mongone' },
  { _id: ObjectId('675cc2736d85340012e9496e'), name: 'mongott' },
  { _id: ObjectId('675ccd26c682b5733ae9496a'), name: 'mongodb' }
]
```

II. COUNTER ESTIMATION

0) Необхідно створити колекцію (таблицю) з лічильником лайків.

- `pip install pymongo`

- `MongoClient(uri)`

```
# Параметри з'єднання з MongoDB
uri = "mongodb://localhost:27017"

client = MongoClient(uri)
db = client["lab"]
collection = db["likes_counter"]

client
```

```
MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True)
```

- `initialize_field()`

```
# Ініціалізація колекції та лічильника "likes"
def initialize_field():
    collection.delete_many({})
    collection.insert_one({"_id": 1, "likes": 0})
    collection.insert_one({"_id": 2, "likes": 0})
    collection.insert_one({"_id": 3, "likes": 0})
    collection.insert_one({"_id": 4, "likes": 0})
```

```
labReplicaSet [direct: primary] lab> show collections
likes_counter
labReplicaSet [direct: primary] lab> db.likes_counter.find()
[
  { _id: 1, likes: 0 },
  { _id: 2, likes: 0 },
  { _id: 3, likes: 0 },
  { _id: 4, likes: 0 }
]
```

- `increment_likes(id, write_concern_option)`

```
# Інкремент лічильника на '+1' через findOneAndUpdate
def increment_likes(id, write_concern_option):
    for _ in range(10000):
        collection.find_one_and_update(
            {"_id": id}, {"$inc": {"likes": 1}},
            write_concern=write_concern_option)
```

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

- Snippet of Code – **writeConcern=1**

```
print(f"\n{'-' * 50}\nExecuting task with 'writeConcern=1'...\n{'-' * 50}")
execute_task(id=1, write_concern=1, disconnect_primary=False)
```

- Executing Result – **writeConcern=1**

```
--------------------------------------------------
Executing task with 'writeConcern=1'...
--------------------------------------------------
Task executed in: 324.74 seconds
Final counter value: 100000
```

```
labReplicaSet [direct: primary] lab> db.likes_counter.find()
[
  { _id: 1, likes: 100000 },
  { _id: 2, likes: 0 },
  { _id: 3, likes: 0 },
  { _id: 4, likes: 0 }
]
```

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.

- Snippet of Code – **writeConcern=majority**

```
print(f"\n{'-' * 50}\nExecuting task with 'writeConcern=majority'...\n{'-' * 50}")
execute_task(id=2, write_concern="majority", disconnect_primary=False)
```

- Executing Result – **writeConcern=majority**

```
--------------------------------------------------
Executing task with 'writeConcern=majority'...
--------------------------------------------------
Task executed in: 786.36 seconds
Final counter value: 100000
```

```
labReplicaSet [direct: primary] lab> db.likes_counter.find()
[
  { _id: 1, likes: 100000 },
  { _id: 2, likes: 100000 },
  { _id: 3, likes: 0 },
  { _id: 4, likes: 0 }
]
```

3) Повторно запустіть 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.

- Snippet of Code – **writeConcern=1 and disconnect_primary**

```
print(f"\n{'-' * 50}\nExecuting task with 'writeConcern=1' and 'disconnect_primary'...\n{'-' * 50}")
execute_task(id=3, write_concern=1, disconnect_primary=True)
```

- Executing Result – **writeConcern=1 and disconnect_primary**

```
--------------------------------------------------
Executing task with 'writeConcern=1' and 'disconnect_primary'...
--------------------------------------------------
Manually disconnect PrimaryNode!

Task executed in: 286.34 seconds
Final counter value: 84615
```

```
labReplicaSet [direct: primary] lab> db.likes_counter.find()
[
  { _id: 1, likes: 100000 },
  { _id: 2, likes: 100000 },
  { _id: 3, likes: 84615 },
  { _id: 4, likes: 0 }
]
```

4) Повторно запустіть 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.

- `Snippet of Code` – **writeConcern=majority and discon_prime**

```
print(f"\n{'-' * 50}\nExecuting task with 'writeConcern=majority' and 'disconnect_primary'...\n{'-' * 50}")
execute_task(id=4, write_concern="majority", disconnect_primary=True)
```

- `Executing Result` – **writeConcern=majority and discon_prime**

```
--------------------------------------------------
Executing task with 'writeConcern=majority' and 'disconnect_primary'...
--------------------------------------------------
Manually disconnect PrimaryNode!

Task executed in: 412.06 seconds
Final counter value: 100000
```

```
labReplicaSet [direct: primary] lab> db.likes_counter.find()
[
  { _id: 1, likes: 100000 },
  { _id: 2, likes: 100000 },
  { _id: 3, likes: 84615 },
  { _id: 4, likes: 100000 }
]
```