

**Проектування високонавантажених систем**  
**Лабораторна робота №4**  
**Налаштування реплікації та перевірка відмовостійкості**  
**MongoDB**  
**Мартиненко Денис ФБ-42мп**

## **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)

- Файл `docker-compose.yml`:

```
version: '3'

services:
  mongo1:
    ports:
      - 27017:27017
    container_name: mongo1
    networks:
      - mongoCluster
    image: mongo:latest
    command: mongod --replSet myReplicaSet --bind_ip localhost,mongo1
  mongo2:
    ports:
      - 27018:27017
    container_name: mongo2
    networks:
      - mongoCluster
    image: mongo:latest
    command: mongod --replSet myReplicaSet --bind_ip localhost,mongo2
  mongo3:
    ports:
      - 27019:27017
    container_name: mongo3
    networks:
      - mongoCluster
    image: mongo:latest
    command: mongod --replSet myReplicaSet --bind_ip localhost,mongo3

networks:
  mongoCluster:
    driver: bridge
```

## Запущені контейнери:

dataintensive\_lab4

C:\Users\denis\OneDrive\Рабочий стол\dataintensive\_lab4

View Configurations

mongo3

mongo:latest

27019:27017

mongo2

mongo:latest

27018:27017

mongo1

mongo:latest

27017:27017

age\_collection","cursor":{},"pipeline":[{"\$collStats":{"storageStats":{"waitFo  
rLock":false,"numericOnly":true}}}],"\$db":"config"}}}  
2024-12-13 23:45:52 mongo3 | {"t":{"\$date":"2024-12-13T21:45:52.000+00:00"},"  
s":"W", "c":"QUERY", "id":23799, "ctx":"ftdc","msg":"Aggregate command e  
xecutor error","attr":{"error":{"code":26,"codeName":"NamespaceNotFound","errm  
sg":"Unable to retrieve storageStats in \$collStats stage :: caused by :: Colle  
ction [local.oplog.rs] not found."},"stats":{},"cmd":{"aggregate":"oplog.rs","  
cursor":{},"pipeline":[{"\$collStats":{"storageStats":{"waitForLock":false,"num  
ericOnly":true}}}],"\$db":"local"}}}  
2024-12-13 23:45:52 mongo3 | {"t":{"\$date":"2024-12-13T21:45:52.000+00:00"},"  
s":"W", "c":"QUERY", "id":23799, "ctx":"ftdc","msg":"Aggregate command e  
xecutor error","attr":{"error":{"code":26,"codeName":"NamespaceNotFound","errm  
sg":"Unable to retrieve storageStats in \$collStats stage :: caused by :: Colle  
ction [config.transactions] not found."},"stats":{},"cmd":{"aggregate":"transa  
ctions"},"cursor":{},"pipeline":[{"\$collStats":{"storageStats":{"waitForLock":f

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

SERIAL MONITOR

PS C:\Users\denis\OneDrive\Рабочий стол\dataintensive\_lab4> docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
5a64bc918f02	mongo:latest	"docker-entrypoint.s..."	23 minutes ago	Up 7 minutes	0.0.0.0:27019->27017/tcp	mongo3
463c88cc45e3	mongo:latest	"docker-entrypoint.s..."	23 minutes ago	Up 7 minutes	0.0.0.0:27018->27017/tcp	mongo2
14e5cfc3f800	mongo:latest	"docker-entrypoint.s..."	23 minutes ago	Up 7 minutes	0.0.0.0:27017->27017/tcp	mongo1

PS C:\Users\denis\OneDrive\Рабочий стол\dataintensive\_lab4>

## Ініціалізація реплікації. mongo1 - Primary Member:

```
docker exec -it mongo1 mongosh --eval "rs.initiate({
  _id: 'myReplicaSet',
  members: [
    { _id: 0, host: 'mongo1:27017' },
    { _id: 1, host: 'mongo2:27017' },
    { _id: 2, host: 'mongo3:27017' }
  ]
})"
```

```
PS C:\Users\denis\OneDrive\Рабочий стол\dataintensive_lab4> docker exec -it mongo1 mongosh --eval "rs.initiate({
>>   _id: 'myReplicaSet',
>>   members: [
>>     { _id: 0, host: 'mongo1:27017' },
>>     { _id: 1, host: 'mongo2:27017' },
>>     { _id: 2, host: 'mongo3:27017' }
>>   ]
>> })"
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1734127015, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1734127015, i: 1 })
}
```

## Статус набору реплік:

```
Learn more at https://docs.docker.com/go/debug-cli/
PS C:\Users\denis\OneDrive\Рабочий стол\dataintensive_lab4> docker exec -it mongo1 mongosh --eval "rs.status()"
{
  set: 'myReplicaSet',
  date: ISODate('2024-12-13T21:58:38.608Z'),
  myState: 2,
  term: Long('1'),
  syncSourceHost: 'mongo2:27017',
  syncSourceId: 1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1734127117, i: 1 }), t: Long('1') },
    lastCommittedWallTime: ISODate('2024-12-13T21:58:37.106Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1734127117, i: 1 }), t: Long('1') },
    appliedOpTime: { ts: Timestamp({ t: 1734127117, i: 1 }), t: Long('1') },
    durableOpTime: { ts: Timestamp({ t: 1734127117, i: 1 }), t: Long('1') },
    writtenOpTime: { ts: Timestamp({ t: 1734127117, i: 1 }), t: Long('1') },
    lastAppliedWallTime: ISODate('2024-12-13T21:58:37.106Z'),
    lastDurableWallTime: ISODate('2024-12-13T21:58:37.106Z'),
    lastWrittenWallTime: ISODate('2024-12-13T21:58:37.106Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1734127067, i: 1 }),
  electionParticipantMetrics: {
    votedForCandidate: true,
    electionTerm: Long('1'),
    lastVoteDate: ISODate('2024-12-13T21:57:06.992Z'),
    electionCandidateMemberId: 1,
    voteReason: '',
    lastWrittenOpTimeAtElection: { ts: Timestamp({ t: 1734127015, i: 1 }), t: Long('-1') },
    maxWrittenOpTimeInSet: { ts: Timestamp({ t: 1734127015, i: 1 }), t: Long('-1') },
    lastAppliedOpTimeAtElection: { ts: Timestamp({ t: 1734127015, i: 1 }), t: Long('-1') },
    maxAppliedOpTimeInSet: { ts: Timestamp({ t: 1734127015, i: 1 }), t: Long('-1') },
    priorityAtElection: 1,
    newTermStartDate: ISODate('2024-12-13T21:57:07.070Z'),
    newTermAppliedDate: ISODate('2024-12-13T21:57:07.618Z')
  },
}
```

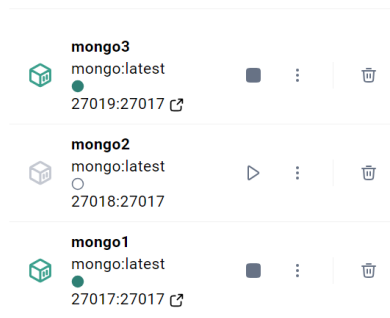
```
members: [
  {
    _id: 0,
    name: 'mongo1:27017',
    health: 1,
    state: 2,
    stateStr: 'SECONDARY',
    uptime: 1176,
    optime: { ts: Timestamp({ t: 1734127117, i: 1 }), t: Long('1') },
    optimeDate: ISODate('2024-12-13T21:58:37.000Z'),
    optimeWritten: { ts: Timestamp({ t: 1734127117, i: 1 }), t: Long('1') },
    optimeWrittenDate: ISODate('2024-12-13T21:58:37.000Z'),
    lastAppliedWallTime: ISODate('2024-12-13T21:58:37.106Z'),
    lastDurableWallTime: ISODate('2024-12-13T21:58:37.106Z'),
    lastWrittenWallTime: ISODate('2024-12-13T21:58:37.106Z'),
    syncSourceHost: 'mongo2:27017',
    syncSourceId: 1,
    infoMessage: '',
    configVersion: 1,
    configTerm: 1,
    self: true,
    lastHeartbeatMessage: ''
  },
]
```

```
{
  _id: 2,
  name: 'mongo3:27017',
  health: 1,
  state: 2,
  stateStr: 'SECONDARY',
  uptime: 102,
  optime: { ts: Timestamp({ t: 1734127117, i: 1 }), t: Long('1') },
  optimeDurable: { ts: Timestamp({ t: 1734127117, i: 1 }), t: Long('1') },
  optimeWritten: { ts: Timestamp({ t: 1734127117, i: 1 }), t: Long('1') },
  optimeDate: ISODate('2024-12-13T21:58:37.000Z'),
  optimeDurableDate: ISODate('2024-12-13T21:58:37.000Z'),
  optimeWrittenDate: ISODate('2024-12-13T21:58:37.000Z'),
  lastAppliedWallTime: ISODate('2024-12-13T21:58:37.106Z'),
  lastDurableWallTime: ISODate('2024-12-13T21:58:37.106Z'),
  lastWrittenWallTime: ISODate('2024-12-13T21:58:37.106Z'),
  lastHeartbeat: ISODate('2024-12-13T21:58:38.056Z'),
  lastHeartbeatRecv: ISODate('2024-12-13T21:58:38.055Z'),
  pingMs: Long('0'),
  lastHeartbeatMessage: '',
  syncSourceHost: 'mongo2:27017',
  syncSourceId: 1,
  infoMessage: '',
  configVersion: 1,
  configTerm: 1
}
```

```
{
  _id: 1,
  name: 'mongo2:27017',
  health: 1,
  state: 1,
  stateStr: 'PRIMARY',
  uptime: 102,
  optime: { ts: Timestamp({ t: 1734127117, i: 1 }), t: Long('1') },
  optimeDurable: { ts: Timestamp({ t: 1734127117, i: 1 }), t: Long('1') },
  optimeWritten: { ts: Timestamp({ t: 1734127117, i: 1 }), t: Long('1') },
  optimeDate: ISODate('2024-12-13T21:58:37.000Z'),
  optimeDurableDate: ISODate('2024-12-13T21:58:37.000Z'),
  optimeWrittenDate: ISODate('2024-12-13T21:58:37.000Z'),
  lastAppliedWallTime: ISODate('2024-12-13T21:58:37.106Z'),
  lastDurableWallTime: ISODate('2024-12-13T21:58:37.106Z'),
  lastWrittenWallTime: ISODate('2024-12-13T21:58:37.106Z'),
  lastHeartbeat: ISODate('2024-12-13T21:58:38.055Z'),
  lastHeartbeatRecv: ISODate('2024-12-13T21:58:37.044Z'),
  pingMs: Long('0'),
  lastHeartbeatMessage: '',
  syncSourceHost: '',
  syncSourceId: -1,
  infoMessage: '',
  electionTime: Timestamp({ t: 1734127026, i: 1 }),
  electionDate: ISODate('2024-12-13T21:57:06.000Z'),
  configVersion: 1,
  configTerm: 1
},
```

```
,
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1734127117, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1734127117, i: 1 })
}
```

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



- Зупиняємо контейнер

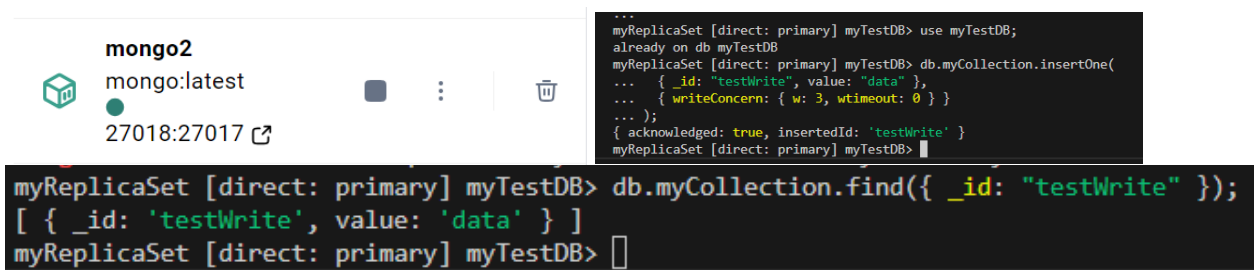
```
myReplicaSet [direct: primary] test> show dbs;
admin      80.00 KiB
config     164.00 KiB
local      444.00 KiB
myReplicaSet [direct: primary] test> use myTestDB;
switched to db myTestDB
myReplicaSet [direct: primary] myTestDB> db.myCollection.insertOne({ test: "value" });
{
  acknowledged: true,
  insertedId: ObjectId('675cba3daa65102742e9496a')
}
myReplicaSet [direct: primary] myTestDB> show dbs;
admin      80.00 KiB
config     164.00 KiB
local      444.00 KiB
myTestDB   8.00 KiB
myReplicaSet [direct: primary] myTestDB> 
```

- Додаємо БД для експерименту

```
myReplicaSet [direct: primary] myTestDB> use myTestDB;
already on db myTestDB
myReplicaSet [direct: primary] myTestDB> db.myCollection.insertOne(
...   { _id: "testWrite", value: "data" },
...   { writeConcern: { w: 3, wtimeout: 0 } }
... );

```

- З відключеною ногою, команда впаде у нескінченне виконання, оскільки один із вузлів (mongo2) недоступний, і запис не може бути підтверджений на всіх трьох вузлах



- Після вмикання ноди команда виконується

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"*

```
myReplicaSet [direct: primary] myTestDB> db.myCollection.insertOne(
...   { _id: "testTimeout", value: "data" },
...   { writeConcern: { w: 3, wtimeout: 5000 } }
... );

Uncaught:
MongoWriteConcernError[WriteConcernFailed]: waiting for replication timed out
Additional information: {
  wtimeout: true,
  writeConcern: { w: 3, wtimeout: 5000, provenance: 'clientSupplied' }
}
Result: {
  n: 1,
  electionId: ObjectId('7fffffff000000000000000000000000'),
  opTime: { ts: Timestamp({ t: 1734130730, i: 1 }), t: Long('2') },
  writeConcernError: {
    code: 64,
    codeName: 'WriteConcernFailed',
    errmsg: 'waiting for replication timed out',
    errInfo: {
      wtimeout: true,
      writeConcern: { w: 3, wtimeout: 5000, provenance: 'clientSupplied' }
    }
  },
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1734130730, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1734130730, i: 1 })
}
myReplicaSet [direct: primary] myTestDB>
myReplicaSet [direct: primary] myTestDB> []
```












- Отримали абсолютно очікуваний результат, команда завершилася через 5 секунд із помилкою таймауту, оскільки один із вузлів недоступний

```
myReplicaSet [direct: primary] myTestDB> db.myCollection.find({ _id: "testTimeout" });
[ { _id: 'testTimeout', value: 'data' } ]
myReplicaSet [direct: primary] myTestDB> []
```

```
myReplicaSet [direct: primary] myTestDB> db.myCollection.find({ _id: "testTimeout" }).readConcern("majority");
[ { _id: 'testTimeout', value: 'data' } ]
myReplicaSet [direct: primary] myTestDB> []
```

4. 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 прімару на неї реплікуються нові дані, які з'явилися під час її простою

mongo3			
	mongo:latest		
	27019:27017		
mongo2			
	mongo:latest		
	27018:27017		
mongo1			
	mongo:latest		
	27017:27017		

- Вимикаємо Primary ноду

```

PS C:\Users\denis\OneDrive\Рабочий стол\dataintensive_lab4> docker exec -it mongo2 mongosh
Current Mongosh Log ID: 675cbe2e276c104f86e94969
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/privacy-policy).
You can opt-out by running the disableTelemetry() command.

-----
The server generated these startup warnings when booting
2024-12-13T23:05:51.811+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/advise-xfs
2024-12-13T23:05:53.198+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2024-12-13T23:05:53.199+00:00: For customers running the current memory allocator, we suggest changing the contents of the following sysfs file to 0: /sys/kernel/mm/transparent_hugepage/enabled
2024-12-13T23:05:53.199+00:00: We suggest setting the contents of sysfsFile to 0.
2024-12-13T23:05:53.199+00:00: Your system has glibc support for rseq built in, which is not yet supported by tcmalloc-google and has critical race conditions.
2024-12-13T23:05:53.199+00:00: Environment variable GLIBC_TUNABLES=glibc.pthread.rseq=0
2024-12-13T23:05:53.199+00:00: vm.max_map_count is too low
2024-12-13T23:05:53.199+00:00: We suggest setting swappiness to 0 or 1, as swapping can cause performance problems.

-----

myReplicaSet [direct: primary] test> rs.status();
{
  "set": "myReplicaSet",
  "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") },
      "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("2024-12-13T23:06:27.854Z"),
      "lastDurableWallTime": ISODate("2024-12-13T23:06:27.854Z"),
      "lastWrittenWallTime": ISODate("2024-12-13T23:06:27.854Z"),
      "lastHeartbeat": ISODate("2024-12-13T23:07:33.913Z"),
      "lastHeartbeatRecv": ISODate("2024-12-13T23:06:36.365Z"),
      "pingMs": Long("0"),
      "lastHeartbeatMessage": "Error connecting to mongo1:27017 :: caused by :: Could not find address for mongo1:27017: SocketException: onInv",
      "syncSourceHost": "",
      "syncSourceId": -1,
      "infoMessage": "",
      "configVersion": 1,
      "configTerm": 3
    },
    {
      "_id": 1,
      "name": "mongo2:27017",
      "health": 1,
      "state": 1,
      "stateStr": "PRIMARY",
      "uptime": 103,
      "optime": { "ts": Timestamp({ t: 1734131247, i: 1 }), "t": Long("3") },
      "optimeDate": ISODate("2024-12-13T23:07:27.000Z"),
      "optimeWritten": { "ts": Timestamp({ t: 1734131247, i: 1 }), "t": Long("3") },
      "optimeWrittenDate": ISODate("2024-12-13T23:07:27.000Z"),
      "lastAppliedWallTime": ISODate("2024-12-13T23:07:27.855Z"),
    }
  ]
}

```

- Secondary нода 2 стала Primary

```

myReplicaSet [direct: primary] test> use myTestDB;
myReplicaSet [direct: primary] myTestDB> db.myCollection.insertOne({ _id: "newData", value: "added during failover" });
{ acknowledged: true, insertedId: 'newData' }
myReplicaSet [direct: primary] myTestDB> 

```

- Додаємо нові дані до БД через нову Primary ноду

```

learn more at https://docs.docker.com/go/deleg-cri/
PS C:\Users\denis\OneDrive\Рабочий стол\dataintensive_lab4> docker exec -it mongo1 mongosh
Current Mongosh Log ID: 675cbf0ae13399e307e94969
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
2024-12-13T23:10:43.415+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodn
2024-12-13T23:10:45.033+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2024-12-13T23:10:45.034+00:00: For customers running the current memory allocator, we suggest changing the contents of the following sysfsFile
2024-12-13T23:10:45.035+00:00: We suggest setting the contents of sysfsFile to 0.
2024-12-13T23:10:45.035+00:00: Your system has glibc support for rseq built in, which is not yet supported by tcmalloc-google and has critical performance im
nment variable GLIBC_TUNABLES=glibc.pthread.rseq=0
2024-12-13T23:10:45.035+00:00: vm.max_map_count is too low
2024-12-13T23:10:45.037+00:00: We suggest setting swappiness to 0 or 1, as swapping can cause performance problems.
-----

myReplicaSet [direct: secondary] test> rs.status();
{
  set: 'myReplicaSet',
  date: ISODate('2024-12-13T23:11:09.528Z'),
  myState: 2,
  term: Long('3'),
  syncSourceHost: 'mongo3:27017',
  syncSourceId: 2,
  heartbeatIntervalMillis: Long('2000'),
},
lastStableRecoveryTimestamp: Timestamp({ t: 1734131156, i: 1 }),
members: [
  {
    _id: 0,
    name: 'mongo1:27017',
    health: 1,
    state: 2,
    stateStr: 'SECONDARY',
    uptime: 26,
    optime: { ts: Timestamp({ t: 1734131467, i: 1 }), t: Long('3') },
    optimeDate: ISODate('2024-12-13T23:11:07.000Z'),
    optimeWritten: { ts: Timestamp({ t: 1734131467, i: 1 }), t: Long('3') },
    optimeWrittenDate: ISODate('2024-12-13T23:11:07.000Z'),
    lastAppliedWallTime: ISODate('2024-12-13T23:11:07.855Z'),
    lastDurableWallTime: ISODate('2024-12-13T23:11:07.855Z'),
    lastWrittenWallTime: ISODate('2024-12-13T23:11:07.855Z'),
    syncSourceHost: 'mongo3:27017',
    syncSourceId: 2,
    infoMessage: '',
    configVersion: 1,
    configTerm: 3,
    self: true,
    lastHeartbeatMessage: ''
  },
  {
    _id: 1,
    name: 'mongo2:27017',
    health: 1,
    state: 1,
    stateStr: 'PRIMARY',
    uptime: 24,
    optime: { ts: Timestamp({ t: 1734131467, i: 1 }), t: Long('3') },
    optimeDurable: { ts: Timestamp({ t: 1734131467, i: 1 }), t: Long('3') },
    optimeWritten: { ts: Timestamp({ t: 1734131467, i: 1 }), t: Long('3') },
    optimeDate: ISODate('2024-12-13T23:11:07.000Z'),
    optimeDurableDate: ISODate('2024-12-13T23:11:07.000Z'),
  }
]

```

- Знову запускаємо нашу стару Primary ноду і бачимо, що після перезапуску вона перейшла у статус Secondary

```

myReplicaSet [direct: secondary] test> use myTestDB;
switched to db myTestDB
myReplicaSet [direct: secondary] myTestDB> db.myCollection.find({ _id: "newData" });
[ { _id: 'newData', value: 'added during failover' } ]
myReplicaSet [direct: secondary] myTestDB> 

```

- Нові дані з реплікації з'явилися на старій Primary



## II 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.

Для того, щоб не було lost updates, для оновлення каунтера необхідно використовувати функцію [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>

```
myReplicaSet [direct: primary] likesDB> db.likesCounter.insertOne({ _id: "likeCounter", count: 0 });  
{ acknowledged: true, insertedId: 'likeCounter' }  
myReplicaSet [direct: primary] likesDB>   
  
myReplicaSet [direct: primary] likesDB> db.likesCounter.find();  
[ { _id: 'likeCounter', count: 0 } ]  
myReplicaSet [direct: primary] likesDB>   

```

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

```
root@04fb5d42831b:/app# node increment.js  
Запускаємо клієнтів...  
Клієнт 3 завершив роботу.  
Клієнт 8 завершив роботу.  
Клієнт 4 завершив роботу.  
Клієнт 1 завершив роботу.  
Клієнт 10 завершив роботу.  
Клієнт 7 завершив роботу.  
Клієнт 9 завершив роботу.  
Клієнт 5 завершив роботу.  
Клієнт 6 завершив роботу.  
Клієнт 2 завершив роботу.  
Усі клієнти завершили роботу.  
Час виконання: 140.268 секунд  
Кінцеве значення каунтера: 100000  
root@04fb5d42831b:/app#
```

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

```
root@04fb5d42831b:/app# node increment.js  
Запускаємо клієнтів...  
Клієнт 6 завершив роботу.  
Клієнт 4 завершив роботу.  
Клієнт 2 завершив роботу.  
Клієнт 1 завершив роботу.  
Клієнт 3 завершив роботу.  
Клієнт 10 завершив роботу.  
Клієнт 7 завершив роботу.  
Клієнт 5 завершив роботу.  
Клієнт 9 завершив роботу.  
Клієнт 8 завершив роботу.  
Усі клієнти завершили роботу.  
Час виконання: 289.555 секунд  
Кінцеве значення каунтера: 200000  
root@04fb5d42831b:/app#
```

- Значення каунтера 200к, але це очікуваний результат, бо кількість лайків при виконанні коду з *ws=1* ми не скидали



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

[illegible]

- Так, ноду 1 знову було обрано Primary

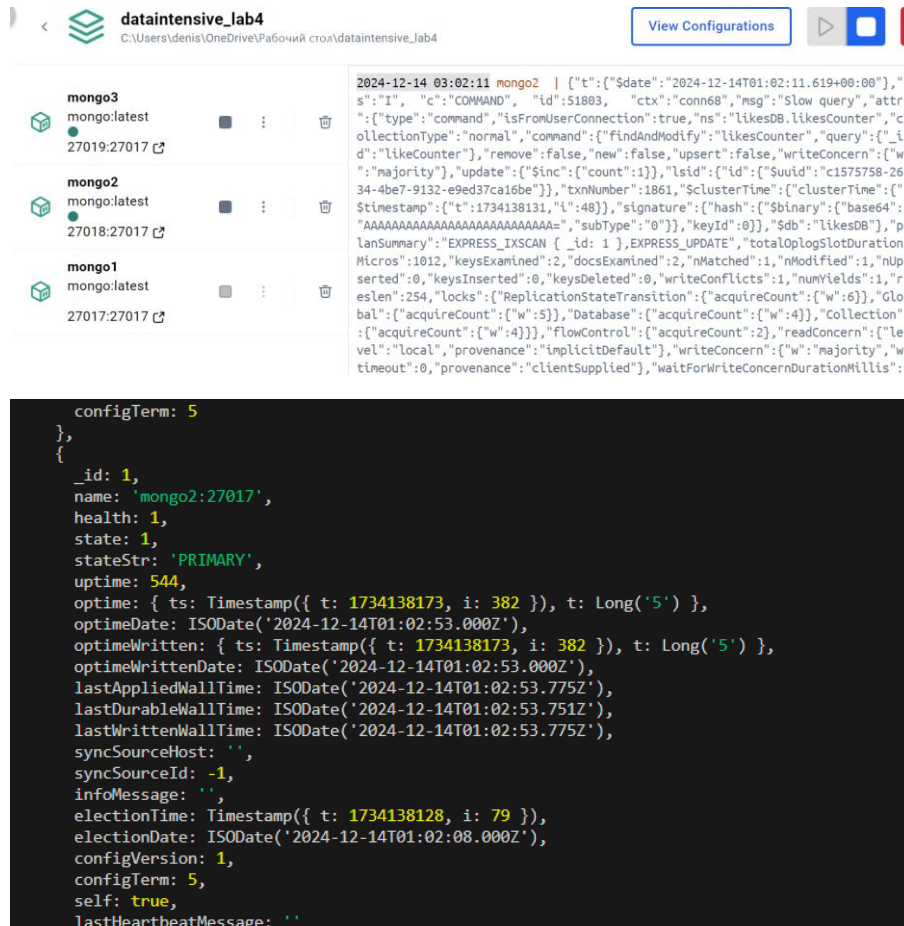
```
root@04fb5d42831b:/app# node increment.js
Запускаємо клієнтів...
Клієнт 3 завершив роботу.
Клієнт 4 завершив роботу.
Клієнт 8 завершив роботу.
Клієнт 7 завершив роботу.
Клієнт 5 завершив роботу.
Клієнт 2 завершив роботу.
Клієнт 9 завершив роботу.
Клієнт 6 завершив роботу.
Клієнт 10 завершив роботу.
Клієнт 1 завершив роботу.
Усі клієнти завершили роботу.
Час виконання: 132.035 секунд
root@04fb5d42831b:/app#

myReplicaSet [direct: secondary] likesDB> db.likesCounter.find();
[ { _id: 'likeCounter', count: 300000 } ]
```

- Отримали очікуваний результат

8. Повторно запустити код при `writeConcern = majority`, але тепер під час роботи відключити 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.



The screenshot shows the MongoDB Atlas interface for a cluster named 'dataintensive\_lab4'. On the left, a list of nodes is shown: 'mongo3' (27019:27017), 'mongo2' (27018:27017), and 'mongo1' (27017:27017). The 'mongo2' node is highlighted. On the right, a log entry from 'mongo2' is displayed, showing a 'findAndModify' operation on the 'likesCounter' collection with a 'writeConcern' of 'majority'. Below the log entry, a configuration snippet for 'mongo2:27017' is shown, indicating its state as 'PRIMARY'.

- Так, ноду 2 знову було обрано Primary

```
root@04fb5d42831b:/app# node increment.js
Запускаємо клієнтів...
Клієнт 3 завершив роботу.
Клієнт 7 завершив роботу.
Клієнт 5 завершив роботу.
Клієнт 9 завершив роботу.
Клієнт 10 завершив роботу.
Клієнт 8 завершив роботу.
Клієнт 6 завершив роботу.
Клієнт 4 завершив роботу.
Клієнт 1 завершив роботу.
Клієнт 2 завершив роботу.
Усі клієнти завершили роботу.
Час виконання: 310.375 секунд
Кінцеве значення каунтера: 400000
```

```
myReplicaSet [direct: primary] likesDB> db.likesCounter.find();
[ { _id: 'likeCounter', count: 400000 } ]
myReplicaSet [direct: primary] likesDB>
```

- Отримали очікуваний результат

## Код

```
const { MongoClient } = require("mongodb");

// Налаштування підключення до Primary (mongo2)
const uri =
  "mongodb://mongo1:27017,mongo2:27017,mongo3:27017/?replicaSet=myReplicaSet";
const dbName = "likesDB";
const collectionName = "likesCounter";

// Функція для інкрементування каунтера
async function incrementCounter(clientId, iterations = 10000) {
  const client = new MongoClient(uri);
  await client.connect();

  const db = client.db(dbName);
  const collection = db.collection(collectionName);

  for (let i = 0; i < iterations; i++) {
    await collection.findOneAndUpdate(
      { _id: "likeCounter" },
      { $inc: { count: 1 } },
      { writeConcern: { w: "majority" } } // або 1 для writeconserne=1
    );
  }

  console.log(`Клієнт ${clientId} завершив роботу.`);
  await client.close();
}

// Функція для запуску 10 клієнтів одночасно
async function runConcurrentClients() {
  const clients = [];
  const clientCount = 10;

  console.log("Запускаємо клієнтів...");

  const startTime = Date.now(); // Початок вимірювання часу

  for (let i = 0; i < clientCount; i++) {
    clients.push(incrementCounter(i + 1));
  }

  await Promise.all(clients);

  const endTime = Date.now(); // Кінець вимірювання часу
  console.log(`Усі клієнти завершили роботу.`);
  console.log(`Час виконання: ${(endTime - startTime) / 1000} секунд`);

  // Після завершення перевіряємо значення каунтера
}
```

```
const client = new MongoClient(uri);
await client.connect();
const db = client.db(dbName);
const collection = db.collection(collectionName);

const result = await collection.findOne({ _id: "likeCounter" });
console.log(`Кінцеве значення каунтера: ${result.count}`);
await client.close();
}

// Запускаємо тест
runConcurrentClients().catch(console.error);
```