

Сербіненко Олексій 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)

```
mongo3
mongo
27019:27017

mongo2
mongo
27018:27017

mongo1
mongo
27017:27017

6,"codeName":"NamespaceNotFound","errmsg":"Unable to retrieve storageStats in $collStats stage ::
caused by :: Collection [config.image_collection] not found."}, "stats": {}, "cmd": {"aggregate": "imag
e_collection", "cursor": {}, "pipeline": [{"$collStats": {"storageStats": {"waitForLock": false, "numericO
nly": true}}}], "$db": "config"}}}
2024-12-28 20:11:11 mongo2 | {"t":{"$date":"2024-12-28T18:11:11.002+00:00"},"s":"W", "c":"QUERY"
, "id":23799, "ctx":"ftdc","msg":"Aggregate command executor error","attr":{"error":{"code":2
6,"codeName":"NamespaceNotFound","errmsg":"Unable to retrieve storageStats in $collStats stage ::
caused by :: Collection [config.transactions] not found."}, "stats": {}, "cmd": {"aggregate": "transact
ions", "cursor": {}, "pipeline": [{"$collStats": {"storageStats": {"waitForLock": false, "numericOnly": tru
e}}}], "$db": "config"}}}
2024-12-28 20:11:11 mongo3 | {"t":{"$date":"2024-12-28T18:11:11.001+00:00"},"s":"W", "c":"QUERY"
, "id":23799, "ctx":"ftdc","msg":"Aggregate command executor error","attr":{"error":{"code":2
6,"codeName":"NamespaceNotFound","errmsg":"Unable to retrieve storageStats in $collStats stage ::
caused by :: Collection [local.oplog.rs] not found."}, "stats": {}, "cmd": {"aggregate": "oplog.rs", "cu
rsor": {}, "pipeline": [{"$collStats": {"storageStats": {"waitForLock": false, "numericOnly": true}}}], "$d
b": "local"}}}
2024-12-28 20:11:11 mongo2 | {"t":{"$date":"2024-12-28T18:11:11.002+00:00"},"s":"W", "c":"QUERY"
, "id":23799, "ctx":"ftdc","msg":"Aggregate command executor error","attr":{"error":{"code":2
6,"codeName":"NamespaceNotFound","errmsg":"Unable to retrieve storageStats in $collStats stage ::
caused by :: Collection [config.image_collection] not found."}, "stats": {}, "cmd": {"aggregate": "imag
e_collection", "cursor": {}, "pipeline": [{"$collStats": {"storageStats": {"waitForLock": false, "numericO
nly": true}}}], "$db": "config"}}}
```

## Replica set

```
alekseyserbinenko@MacBook-Air-Aleksej ~ % mongosh --host localhost --port 27017
Current Mongosh Log ID: 67704159e7478251f829239e
Connecting to: mongod://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.3.7
Using MongoDB: 8.0.4
Using Mongosh: 2.3.7

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
2024-12-28T18:10:50.220+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
2024-12-28T18:10:50.220+00:00: For customers running the current memory allocator, we suggest changing the contents of the following sysfsFile
2024-12-28T18:10:50.220+00:00: We suggest setting the contents of sysfsFile to 0.
2024-12-28T18:10:50.220+00:00: Your system has glibc support for rseq built in, which is not yet supported by tcmalloc-google and has critical performance imp
t the environment variable GLIBC_TUNABLES=glibc.pthread.rseq=0
2024-12-28T18:10:50.220+00:00: vm.max_map_count is too low
2024-12-28T18:10:50.220+00:00: We suggest setting swappiness to 0 or 1, as swapping can cause performance problems.
-----

test> rs.initiate({
...   _id: "rs0",
...   members: [
...     { _id: 0, host: "mongo1:27017" },
...     { _id: 1, host: "mongo2:27017" },
...     { _id: 2, host: "mongo3:27017" }
...   ]
... });
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1735410022, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAAA', 0),
      keyId: Long('0')
    },
    operationTime: Timestamp({ t: 1735410022, i: 1 })
  }
}
rs0 [direct: secondary] test> █
```

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

зупиню другу ноду

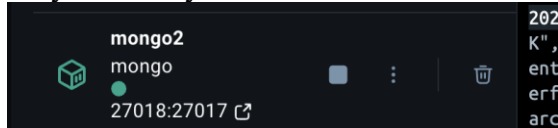
```
Last login: Sat Dec 28 20:10:33 on ttys012
alekseyserbinenko@MacBook-Air-Aleksej ~ % docker stop mongo2
mongo2
```

Виконаю команду `db.test.insertOne({name: "test_1", time: new Date()}, {writeConcern:{w:3, timeout: 0}});`;

Бачу що команда зависла, адже чекає на запуск ноди

```
2 |
rs0 [direct: primary] test> db.test.insertOne({name: "test_1", time: new Date()}, {writeConcern:{w:3, timeo
ut: 0}});
```

запустив ноду



Все працює

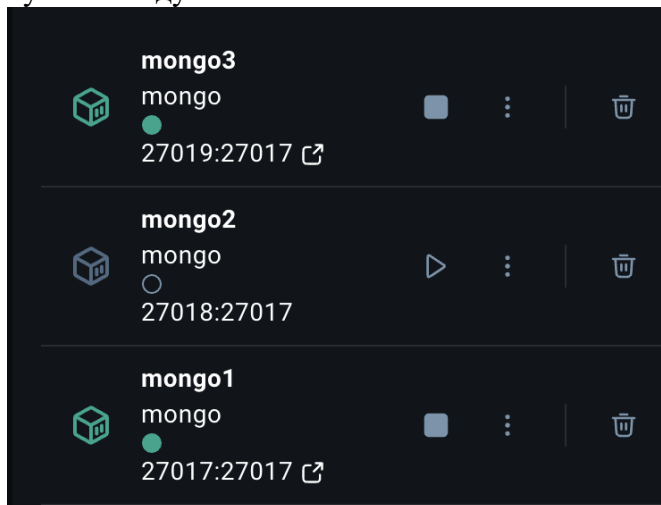
```
[rs0 [direct: primary] test> db.test.insertOne({name: "test_1", time: new Date()}, {writeConcern:{w:3, timeout: 0}});
{
  acknowledged: true,
  insertedId: ObjectId('6770442ce7478251f029239f')
}
[rs0 [direct: primary] test>
```

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"*

`db.test.insert({name:"task3_1", time: new Date()}, {writeConcern:{w:3,timeout:5000}})`

```
[rs0 [direct: primary] test> db.test.insert({name:"task3_1", time: new Date()}, {writeConcern:{w:3,timeout:5000}})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('67704bc8151134c10eb96dc7') }
}
```

Зупиню ноду



```
[rs0 [direct: primary] test> db.test.insert({name:"task3_2", time: new Date()}, {writeConcern:{w:3,timeout:10000}})
Uncaught:
MongoBulkWriteError: waiting for replication timed out
Result: BulkWriteResult {
  insertedCount: 1,
  matchedCount: 0,
  modifiedCount: 0,
  deletedCount: 0,
  upsertedCount: 0,
  upsertedIds: {},
  insertedIds: { '0': ObjectId('67704c4a151134c10eb96dc8') }
}
Write Errors: []
```

Бачу, що запис зберігся(task3\_2)

```

[rs0 [direct: primary] test> db.test.find();
[
  {
    _id: ObjectId('6770442ce7478251f029239f'),
    name: 'test_1',
    time: ISODate('2024-12-28T18:32:12.325Z')
  },
  {
    _id: ObjectId('67704bc8151134c10eb96dc7'),
    name: 'task3_1',
    time: ISODate('2024-12-28T19:04:40.556Z')
  },
  {
    _id: ObjectId('67704c4a151134c10eb96dc8'),
    name: 'task3_2',
    time: ISODate('2024-12-28T19:06:50.333Z')
  }
]
rs0 [direct: primary] test>

```

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 primary на неї реплікуються нові дані, які з'явилися під час її простою

вимкну поточну primary

```

{
  _id: 2,
  name: 'mongo3:27017',
  health: 1,
  state: 1,
  stateStr: 'PRIMARY',
  uptime: 2335
}

```

Mongo3 недоступна.

тепер primary – mongo1

```

{
  _id: 2,
  name: 'mongo3:27017',
  health: 0,
  state: 8,
  stateStr: '(not reachable/healthy)',
  uptime: 0,
  optime: { ts: Timestamp({ t: 0, i: 0 }) },
  optime_d: 0
}

```

```

{
  _id: 0,
  name: 'mongo1:27017',
  health: 1,
  state: 1,
  stateStr: 'PRIMARY',
  uptime: 3908,
  optime: { ts: Timestamp({ t: 1725412250, i: 1 }) },
  optime_d: 2024-12-28T19:10:50.000Z
}

```

```

rs0 [direct: primary] test> db.test.insert({name:"task4", time: new Date()});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('67704f4c7bd3a37731589ad2') }
}
rs0 [direct: primary] test>

```

Запустив монго3

Запис на місці

```

    },
    {
      _id: ObjectId('67704f4c7bd3a37731589ad2'),
      name: 'task4',
      time: ISODate('2024-12-28T19:19:40.389Z')
    }
  ]
]

[rs0 [direct: primary] test> db.test.find().readConcern("majority");
[
  {
    _id: ObjectId('6770442ce7478251f029239f'),
    name: 'test_1',
    time: ISODate('2024-12-28T18:32:12.325Z')
  },
  {
    _id: ObjectId('67704bc8151134c10eb96dc7'),
    name: 'task3_1',
    time: ISODate('2024-12-28T19:04:40.556Z')
  },
  {
    _id: ObjectId('67704c4a151134c10eb96dc8'),
    name: 'task3_2',
    time: ISODate('2024-12-28T19:06:50.333Z')
  },
  {
    _id: ObjectId('67704f4c7bd3a37731589ad2'),
    name: 'task4',
    time: ISODate('2024-12-28T19:19:40.389Z')
  }
]
]

```

## 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\(\)](#)

```

rs0 [direct: primary] test> db.likes_counter.insertOne({_id:1, likes:0});
{ acknowledged: true, insertedId: 1 }
rs0 [direct: primary] test> db.likes_counter.find()

rs0 [direct: primary] test> db.likes_counter.find()
[ { _id: 1, likes: 0 } ]
rs0 [direct: primary] test>

```

1. Вказавши у параметрах `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 - 100K

2. Вказавши у параметрах `findOneAndUpdate` `writeConcern = majority` (це буде означати, що Primary чекає поки значення запишеться 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 100K



Колекція створена. Поточний лічильник: 0  
Тест з writeConcern=1 (без відключень)  
Фінальний лічильник (writeConcern=1): 100000  
Time: 21.878232955932617 s.

Лічильник оновлено!  
Тест з writeConcern=majority (без відключень)  
Фінальний лічильник (writeConcern=1): 100000  
Time: 8.175148010253906 s.

Лічильник оновлено!  
Тест з writeConcern=1 (з відключенням Primary)  
Вимикаємо Primary на порту 27017...  
Нода на порту 27017 вимкнена (PID: 61081).  
Фінальний лічильник (writeConcern=1): 99997  
Time: 26.521278142929077 s.

Очікуємо поки буде увімкнено вимкнену ноду на порту 27017...  
Користувач увімкнув ноду!

Лічильник оновлено!  
Тест з writeConcern=majority (з відключенням Primary)  
Вимикаємо Primary на порту 27018...  
Нода на порту 27018 вимкнена (PID: 61357).  
Фінальний лічильник (writeConcern=1): 100000  
Time: 37.96386504173279 s.