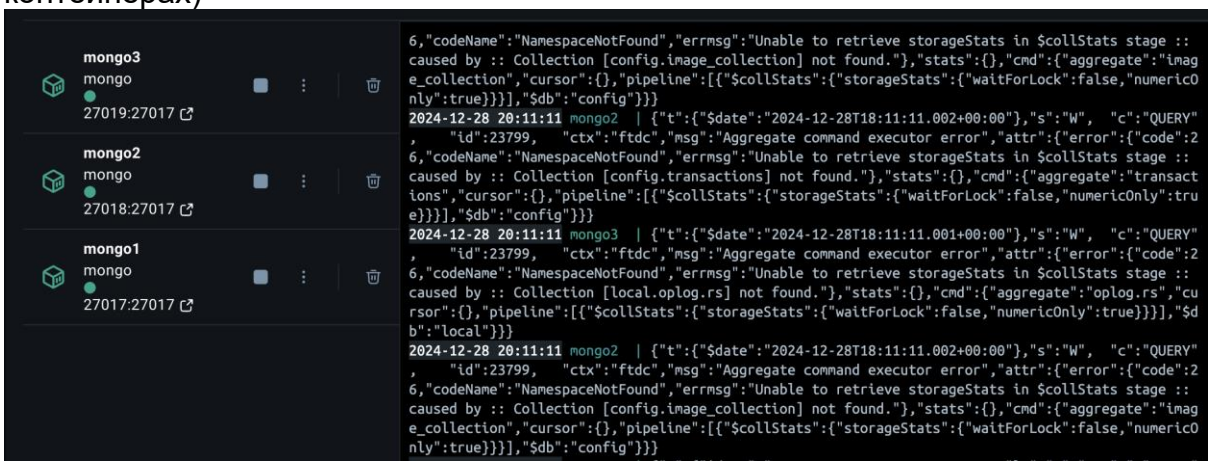
Сербіненко Олексій lab4

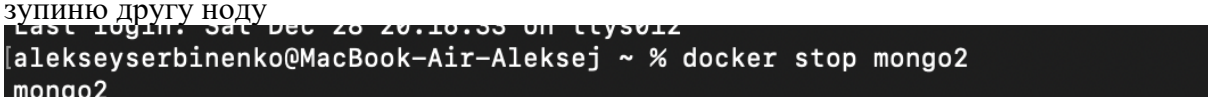
1. Налаштувати реплікацію в конфігурації: Primary with Two Secondary Members (P-S-S) (всі ноди можуть бути запущені як окремі процеси або у Docker контейнерах)



Replica set

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

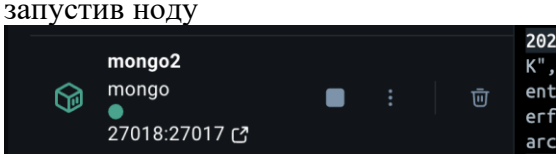
зупиню другу ноду



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

запустив ноду



Все працює

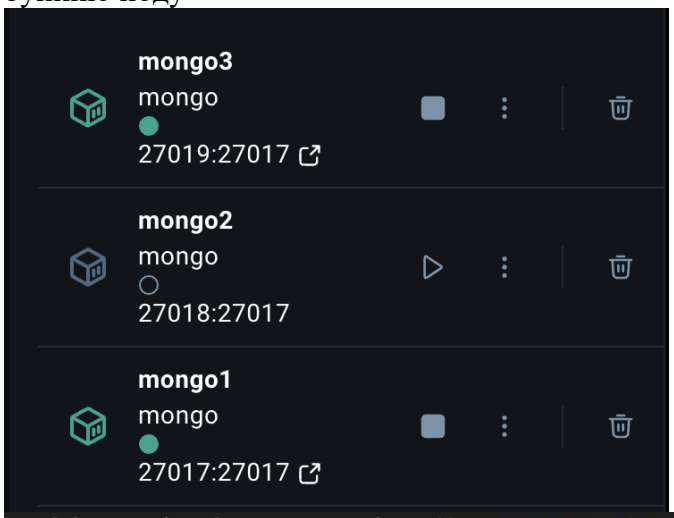
```
[rs0 [direct: primary] test> db.test.insertOne({name: "test_1", time: new Date()}, {writeConcern:{w:3, t
ut: 0}});
{
  acknowledged: true,
  insertedId: ObjectId('6770442ce7478251f029239f')
}
rs0 [direct: primary] test>
```

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,wtimeout:5000}})

```
rs0 [direct: primary] test> db.test.insert({name:"task3_1", time: new Date()}, {writeConcern:{w:3,wtimeout:
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,wtimeout:
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: []
```

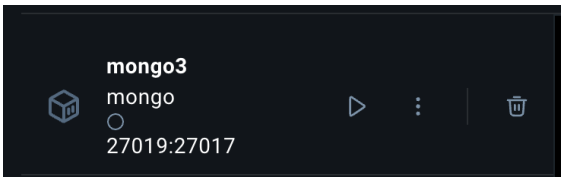
Бачу, що запис зберігся(task3_2)

```
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>
```

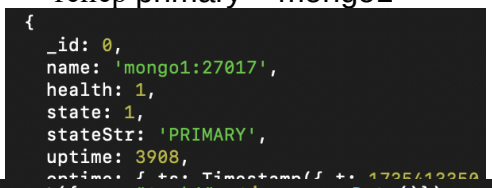
4.Продемонстрував перевибори primary node відключивши поточний primary (Replica Set Elections) - http://docs.mongodb.org/manual/core/replica-set-elections/

- і що після відновлення роботи старої primary на неї реплікуються нові дані, які з'явилися під час її простою

вимкну поточну primary

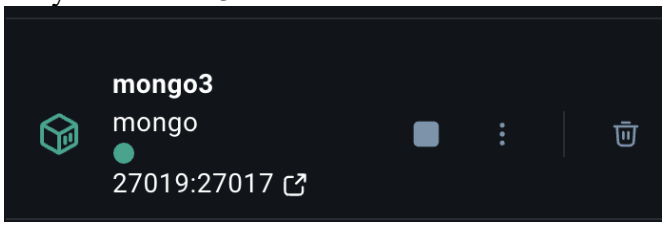


Mongo3 недоступна. тепер primary – mongo1




```
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_conter.find()

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

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

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

```
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.
```