Task4 – MongoDB Replication

Buchko Anna

Source code on
https://github.com/AnnaBuchko/DistributedDatabases/tree/master/Task4-MongoDb

Running 3 instances of mongoDb in docker:

```
D:\Anna\DE UKU\DistributedDatabases\Task4-MongoDb>docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED       STATUS       PORTS                                                  NAMES
7014fc6b048a   mongo:8        "docker-entrypoint.s…"   3 hours ago   Up 3 hours   127.0.10.1:27017->27017/tcp                            task4-mongodb-mongo1-1
e2103843e906   mongo:8        "docker-entrypoint.s…"   3 hours ago   Up 3 hours   127.0.10.3:27017->27017/tcp                            task4-mongodb-mongo3-1
a184b4845393   mongo:8        "docker-entrypoint.s…"   3 hours ago   Up 3 hours   127.0.10.2:27017->27017/tcp                            task4-mongodb-mongo2-1
a7dc144cdf01   neo4j:latest   "tini -g -- /startup…"   2 weeks ago   Up 3 hours   0.0.0.0:7474->7474/tcp, 7473/tcp, 0.0.0.0:7687->7687/tcp   ne4j
```

1. Created 20 items in collection products
2. Query all products:

```
> use homework
< switched to db homework
> db.products.find({})
< {
    _id: ObjectId('67dab6d50526f82bd9235222'),
    category: 'Phone',
    brand: 'Apple',
    model: 'iPhone 13',
    os: 'iOS',
    price: 999
  }
  {
    _id: ObjectId('67dab6d50526f82bd9235223'),
    category: 'Phone',
    brand: 'Samsung',
    model: 'Galaxy S22',
    os: 'Android',
    price: 899
  }
  {
    _id: ObjectId('67dab6d50526f82bd9235224'),
    category: 'Phone',
    brand: 'Google',
    model: 'Pixel 6',
    os: 'Android',
    price: 799
  }
```

```
  {
    _id: ObjectId('67dab6d50526f82bd9235225'),
    category: 'Phone',
    brand: 'OnePlus',
    model: 'OnePlus 10 Pro',
    os: 'Android',
    price: 850
  }
  {
    _id: ObjectId('67dab6d50526f82bd9235226'),
    category: 'Phone',
    brand: 'Xiaomi',
    model: 'Mi 12',
    os: 'Android',
    price: 750
  }
  {
    _id: ObjectId('67dab6d50526f82bd9235227'),
    category: 'TV',
    brand: 'LG',
    screen_size: '65 inches',
    resolution: '4K',
    smart_tv: true,
    price: 1300
  }
```

...

3. Підрахуйте скільки є різних категорій товарів

```
> db.products.distinct("category").length
< 4
> db.products.distinct("category")
< [ 'Laptop', 'Phone', 'Smart Watch', 'TV' ]
```

4. Виведіть список всіх виробників товарів без повторів

```
> db.products.distinct("brand")
< [
    'Apple',   'Asus',
    'Dell',    'Fitbit',
    'Garmin',  'Google',
    'HP',      'Huawei',
    'LG',      'Lenovo',
    'OnePlus', 'Samsung',
    'Sony',    'TCL',
    'Vizio',   'Xiaomi'
  ]
```

5. Обновить певні товари, змінивши існуючі значення і додайте нові властивості (характеристики) усім товарам за певним критерієм

```
> db.products.updateMany({"category" : { $in: ["Phone", "Smart Watch"] } }, { $mul: { price: 1.05 }, $set: { warranty: "2 years" } })
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 10,
    modifiedCount: 10,
    upsertedCount: 0
  }
```

```
> db.products.find({"category" : { $in: ["Phone", "Smart Watch"]}})
< {
    _id: ObjectId('67dab6d50526f82bd9235222'),
    category: 'Phone',
    brand: 'Apple',
    model: 'iPhone 13',
    os: 'iOS',
    price: 1048.95,
    warranty: '2 years'
  }
  {
    _id: ObjectId('67dab6d50526f82bd9235223'),
    category: 'Phone',
    brand: 'Samsung',
    model: 'Galaxy S22',
    os: 'Android',
    price: 943.95,
    warranty: '2 years'
  }
  {
    _id: ObjectId('67dab6d50526f82bd9235224'),
    category: 'Phone',
    brand: 'Google',
    model: 'Pixel 6',
    os: 'Android',
    price: 838.95,
    warranty: '2 years'
  }
```

```
{
  _id: ObjectId('67dab6d50526f82bd923522c'),
  category: 'Smart Watch',
  brand: 'Apple',
  model: 'Apple Watch Series 8',
  features: [
    'Heart Rate Monitor',
    'GPS'
  ],
  price: 577.5,
  warranty: '2 years'
}
{
  _id: ObjectId('67dab6d50526f82bd923522d'),
  category: 'Smart Watch',
  brand: 'Samsung',
  model: 'Galaxy Watch 5',
  features: [
    'ECG',
    'Sleep Tracking'
  ],
  price: 420,
  warranty: '2 years'
}
```

6. Знайдіть товари у яких є (присутнє поле) певні властивості

```
> db.products.find({ "ram": { $exists: true } })
< {
    _id: ObjectId('67dab6d50526f82bd9235231'),
    category: 'Laptop',
    brand: 'Apple',
    model: 'MacBook Air M2',
    ram: '16GB',
    storage: '512GB SSD',
    price: 1400
  }
  {
    _id: ObjectId('67dab6d50526f82bd9235232'),
    category: 'Laptop',
    brand: 'Dell',
    model: 'XPS 13',
    ram: '16GB',
    storage: '1TB SSD',
    price: 1600
  }
```

```
{
  _id: ObjectId('67dab6d50526f82bd9235233'),
  category: 'Laptop',
  brand: 'HP',
  model: 'Spectre x360',
  ram: '16GB',
  storage: '512GB SSD',
  price: 1500
}
{
  _id: ObjectId('67dab6d50526f82bd9235234'),
  category: 'Laptop',
  brand: 'Lenovo',
  model: 'ThinkPad X1 Carbon',
  ram: '32GB',
  storage: '1TB SSD',
  price: 1800
}
{
  _id: ObjectId('67dab6d50526f82bd9235235'),
  category: 'Laptop',
  brand: 'Asus',
  model: 'ROG Zephyrus G15',
  ram: '32GB',
  storage: '2TB SSD',
  price: 2000
}
```

1) Створіть кілька замовлень з різними наборами товарів, але так щоб один з товарів був у декількох замовленнях
   Check file insert_data.txt for full script which insert orders. See 2 orders with same product:

```
_id: ObjectId('67dbd94e79b3ade8cb6b135b')
order_number : 1
date : 2025-03-15T00:00:00.000+00:00
▸ customer : Object
▸ payment : Object
▾ items_id : Array (2)
    0: ObjectId('67dab6d50526f82bd9235222')
    1: ObjectId('67dab6d50526f82bd923522c')
```

```
_id: ObjectId('67dbd94e79b3ade8cb6b135c')
order_number : 2
date : 2025-03-16T00:00:00.000+00:00
▸ customer : Object
▸ payment : Object
▾ items_id : Array (1)
    0: ObjectId('67dab6d50526f82bd9235222')
```

2) Виведіть всі замовлення
   Shown only query and one order, there is 5 more as well

```
> db.orders.find({})
< {
    _id: ObjectId('67dbd94e79b3ade8cb6b135b'),
    order_number: 1,
    date: 2025-03-15T00:00:00.000Z,
    customer: {
      name: 'Olena',
      surname: 'Shevchenko',
      phones: [
        9876549,
        1234568
      ],
      address: 'Kiev, Lva Tolstoho 8, UA'
    },
    payment: {
      card_owner: 'Olena Shevchenko',
      cardId: 98765432
    },
    items_id: [
      ObjectId('67dab6d50526f82bd9235222'),
      ObjectId('67dab6d50526f82bd923522c')
    ]
  }
```

3) Знайдіть всі замовлення з певним товаром (товарами) (шукати можна по ObjectId)

```
> db.orders.find({ items_id: ObjectId("67dab6d50526f82bd9235222") })
< {
    _id: ObjectId('67dbd94e79b3ade8cb6b135b'),
    order_number: 1,
    date: 2025-03-15T00:00:00.000Z,
    customer: {
      name: 'Olena',
      surname: 'Shevchenko',
      phones: [
        9876549,
        1234568
      ],
      address: 'Kiev, Lva Tolstoho 8, UA'
    },
    payment: {
      card_owner: 'Olena Shevchenko',
      cardId: 98765432
    },
    items_id: [
      ObjectId('67dab6d50526f82bd9235222'),
      ObjectId('67dab6d50526f82bd923522c')
    ]
  }
```

```
{
  _id: ObjectId('67dbd94e79b3ade8cb6b135c'),
  order_number: 2,
  date: 2025-03-16T00:00:00.000Z,
  customer: {
    name: 'Yuriy',
    surname: 'Dovzhenko',
    phones: [
      9876540,
      1234560
    ],
    address: 'Lviv, Chornovola 12, UA'
  },
  payment: {
    card_owner: 'Yuriy Dovzhenko',
    cardId: 12398765
  },
  items_id: [
    ObjectId('67dab6d50526f82bd9235222')
  ]
}
```

```
{
  _id: ObjectId('67dbd94e79b3ade8cb6b135f'),
  order_number: 5,
  date: 2025-03-18T00:00:00.000Z,
  customer: {
    name: 'Dmytro',
    surname: 'Shevchenko',
    phones: [
      9870001,
      1478523
    ],
    email: 'dmytro.shevchenko@example.com',
    address: 'Dnipro, Pushkina 10, UA'
  },
  payment: {
    card_owner: 'Dmytro Shevchenko',
    cardId: 77665544,
    payment_method: 'Bank Transfer'
  },
  items_id: [
    ObjectId('67dab6d50526f82bd9235227'),
    ObjectId('67dab6d50526f82bd9235232'),
    ObjectId('67dab6d50526f82bd9235222')
  ],
  status: 'Delivered',
  discount: 5
}
```

4) Додайте в усі замовлення з певним товаром ще один товар і збільште існуючу вартість замовлення на деяке значення X

```
> let newProductId = ObjectId("67dab6d50526f82bd923522e");

  let newProduct = db.products.findOne({ _id: newProductId });
  let newPrice = newProduct ? newProduct.price : 0;

  db.orders.updateMany(
      { items_id: ObjectId("67dab6d50526f82bd9235227") },
      {
          $push: { items_id: newProductId },
          $inc: { total_sum: newPrice }
      }
  );
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 2,
    modifiedCount: 2,
    upsertedCount: 0
  }
```

5) Виведіть тільки інформацію про кастомера і номери кредитної карт, для замовлень вартість яких перевищує певну суму

```
> db.orders.find(
      { total_sum: { $gt: 2500 } },
      { "customer": 1, "payment": 1, "_id": 0 }
  );
< {
    customer: {
      name: 'Dmytro',
      surname: 'Shevchenko',
      phones: [
        9870001,
        1478523
      ],
      email: 'dmytro.shevchenko@example.com',
      address: 'Dnipro, Pushkina 10, UA'
    },
    payment: {
      card_owner: 'Dmytro Shevchenko',
      cardId: 77665544,
      payment_method: 'Bank Transfer'
    }
  }
```

6) Знайдіть замовлення зроблені одним замовником, і виведіть тільки інформацію про кастомера та товари у замовлені підставивши замість ObjectId("***") назви товарів та їх вартість (аналог join-а між таблицями orders та items).

```
db.orders.aggregate([
 {
  $match: { "customer.name": "Olena", "customer.surname": "Shevchenko" }
 },
 {
  $lookup: {
   from: "products",
   localField: "items_id",
   foreignField: "_id",
   as: "ordered_products"
  }
 },
 {
  $project: {
   _id: 0,
   order_number: 1,
   date: 1,
   "customer.name": 1,
   "customer.surname": 1,
   "customer.phones": 1,
   "customer.address": 1,
   ordered_products: 1
  }
 }
]);
```

Створіть [Capped collection](#) яка б містила 5 останніх відгуків на наш інтернет-магазин. Структуру запису визначіть самостійно.

1) Перевірте що при досягненні обмеження старі відгуки будуть затиратись

```
> db.createCollection("reviews", {
    capped: true,
    size: 5120,
    max: 5
});
< { ok: 1 }
```

```
> db.reviews.insertMany([
    { user_id: 1, product_id: ObjectId("67dab6d50526f82bd9235222"), review: "Excellent phone!" },
    { user_id: 2, product_id: ObjectId("67dab6d50526f82bd9235222"), review: "Great camera quality." },
    { user_id: 3, product_id: ObjectId("67dab6d50526f82bd9235227"), review: "Lightweight and powerful." },
    { user_id: 4, product_id: ObjectId("67dab6d50526f82bd9235232"), review: "Perfect." },
    { user_id: 5, product_id: ObjectId("67dab6d50526f82bd923522e"), review: "Amazing!" }
]);
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('67dbeb0a79b3ade8cb6b1361'),
      '1': ObjectId('67dbeb0a79b3ade8cb6b1362'),
      '2': ObjectId('67dbeb0a79b3ade8cb6b1363'),
      '3': ObjectId('67dbeb0a79b3ade8cb6b1364'),
      '4': ObjectId('67dbeb0a79b3ade8cb6b1365')
    }
}
```

```
> db.reviews.insertOne({ user_id: 6, product_id: ObjectId("67dab6d50526f82bd923522c"), review: "Excellent." });
< {
    acknowledged: true,
    insertedId: ObjectId('67dbeb8279b3ade8cb6b1366')
}
> db.reviews.find({})
< {
    _id: ObjectId('67dbeb0a79b3ade8cb6b1362'),
    user_id: 2,
    product_id: ObjectId('67dab6d50526f82bd9235222'),
    review: 'Great camera quality.'
}
{
    _id: ObjectId('67dbeb0a79b3ade8cb6b1363'),
    user_id: 3,
    product_id: ObjectId('67dab6d50526f82bd9235227'),
    review: 'Lightweight and powerful.'
}
{
    _id: ObjectId('67dbeb0a79b3ade8cb6b1364'),
    user_id: 4,
    product_id: ObjectId('67dab6d50526f82bd9235232'),
    review: 'Perfect.'
}
```

```
{
    _id: ObjectId('67dbeb0a79b3ade8cb6b1365'),
    user_id: 5,
    product_id: ObjectId('67dab6d50526f82bd923522e'),
    review: 'Amazing!'
}
{
    _id: ObjectId('67dbeb8279b3ade8cb6b1366'),
    user_id: 6,
    product_id: ObjectId('67dab6d50526f82bd923522c'),
    review: 'Excellent.'
}
```

II Налаштування реплікації

1) Налаштувати реплікацію в конфігурації: Primary with Two Secondary Members (P-S-S) (всі ноди запущені у Docker контейнерах) -

	Name	Container ID	Image	Port(s)	CPU (%)	Last started
▾	task4-mongodb	-	-	-	4.49%	2 hours ago
	mongo1-1	7014fc6b048a	mongo:8	27017:27017	1.43%	2 hours ago
	mongo3-1	e2103843e906	mongo:8	27017:27017	1.47%	2 hours ago
	mongo2-1	a184b4845393	mongo:8	27017:27017	1.59%	2 hours ago

```
> db.isMaster()
< {
    topologyVersion: {
      processId: ObjectId('67dbd4a8dfa421902b38d69b'),
      counter: Long('6')
    },
    hosts: [ 'mongo1:27017', 'mongo2:27017', 'mongo3:27017' ],
    setName: 'rs0',
    setVersion: 1,
    ismaster: true,
    secondary: false,
    primary: 'mongo3:27017',
    me: 'mongo3:27017',
    electionId: ObjectId('7fffffff0000000000000003'),
```

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

```
> var status = rs.status();
  status.members.forEach(function(member) {
      print("Host: " + member.name + " | State: " + member.stateStr + " | Uptime: " + member.uptime + " seconds");
  });
< Host: mongo1:27017 | State: (not reachable/healthy) | Uptime: 0 seconds
< Host: mongo2:27017 | State: SECONDARY | Uptime: 7308 seconds
< Host: mongo3:27017 | State: PRIMARY | Uptime: 7310 seconds
```

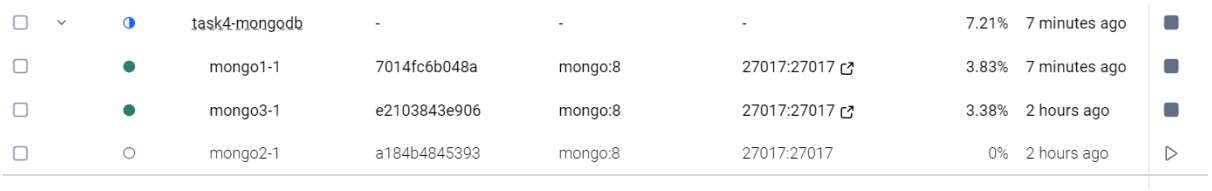
running

```
> db.replicaSetTest.insertOne({ message: "Message 2"}, { writeConcern: { w: 3 }});
```

after enabling mongo1 node the insert has finished but status was not updated:

```
> var status = rs.status();
  status.members.forEach(function(member) {
      print("Host: " + member.name + " | State: " + member.stateStr + " | Uptime: " + member.uptime + " seconds");
  });
< Host: mongo1:27017 | State: (not reachable/healthy) | Uptime: 0 seconds
< Host: mongo2:27017 | State: SECONDARY | Uptime: 7706 seconds
< Host: mongo3:27017 | State: PRIMARY | Uptime: 7708 seconds
```

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”*



		task4-mongodb	-	-	-	7.21%	7 minutes ago
		mongo1-1	7014fc6b048a	mongo:8	27017:27017	3.83%	7 minutes ago
		mongo3-1	e2103843e906	mongo:8	27017:27017	3.38%	2 hours ago
		mongo2-1	a184b4845393	mongo:8	27017:27017	0%	2 hours ago

```
> db.replicaSetTest.insertOne({ message: "Message 5"}, { writeConcern: { w: 3 , wtimeout: 5000}});
✖ ▸ MongoWriteConcernError[WriteConcernFailed]: waiting for replication timed out
> db.replicaSetTest.find(
    { message: "Message 5"},
    { readConcern: { level: "majority" } }
  );
< {
    _id: ObjectId('67dbf75679b3ade8cb6b136d'),
    readConcern: {
      level: 'majority'
    }
  }
> db.replicaSetTest.find(
    { message: "Message 5"},
    { readConcern: { level: 3 } }
  );
< {
    _id: ObjectId('67dbf75679b3ade8cb6b136d')
  }
```

4) Продемонстрував перевибори primary node відключивши поточний primary (Replica Set Elections) -

```
> var status = rs.status();
  status.members.forEach(function(member) {
      print("Host: " + member.name + " | State: " + member.stateStr + " | Uptime: " + member.uptime + " seconds");
  });
< Host: mongo1:27017 | State: SECONDARY | Uptime: 1 seconds
< Host: mongo2:27017 | State: SECONDARY | Uptime: 447 seconds
< Host: mongo3:27017 | State: PRIMARY | Uptime: 449 seconds
```

stop mongo3:

```
> var status = rs.status();
  status.members.forEach(function(member) {
      print("Host: " + member.name + " | State: " + member.stateStr + " | Uptime: " + member.uptime + " seconds");
  });
< Host: mongo1:27017 | State: PRIMARY | Uptime: 312 seconds
< Host: mongo2:27017 | State: SECONDARY | Uptime: 311 seconds
< Host: mongo3:27017 | State: (not reachable/healthy) | Uptime: 0 seconds
```

insert data and check secondary info:

```
> db.replicaSetTest.insertOne({ message: "Message 10"}, { writeConcern: { w: 1 , wtimeout: 5000}});
< {
    acknowledged: true,
    insertedId: ObjectId('67dbfe078c9ffd5fc94eb081')
  }
> rs.printSecondaryReplicationInfo()
<
  source: mongo2:27017

  {
    syncedTo: 'Thu Mar 20 2025 13:37:54 GMT+0200 (Eastern European Standard Time)',
    replLag: '0 secs (0 hrs) behind the primary '
  }
  ---------------------------------------------------------------------------

  source: mongo3:27017

  {
    'no replication info, yet.  State': '(not reachable/healthy)'
  }
```

start mongo3 and check secondary info:

```
> var status = rs.status();
  status.members.forEach(function(member) {
      print("Host: " + member.name + " | State: " + member.stateStr + " | Uptime: " + member.uptime + " seconds");
  });
< Host: mongo1:27017 | State: PRIMARY | Uptime: 428 seconds
< Host: mongo2:27017 | State: SECONDARY | Uptime: 427 seconds
< Host: mongo3:27017 | State: SECONDARY | Uptime: 0 seconds
```

```
> rs.printSecondaryReplicationInfo()
<
  source: mongo2:27017

  {
    syncedTo: 'Thu Mar 20 2025 13:39:14 GMT+0200 (Eastern European Standard Time)',
    replLag: '0 secs (0 hrs) behind the primary '
  }
  ---------------------------------------------------------------------------

  source: mongo3:27017

  {
    syncedTo: 'Thu Mar 20 2025 13:39:14 GMT+0200 (Eastern European Standard Time)',
    replLag: '0 secs (0 hrs) behind the primary '
  }
```

II Аналіз продуктивності та перевірка цілісності

Run application on 3 running nodes with writeComcern 1 and majority. See log with result below:

```
2025-03-20 15:22:19,905 - INFO - Running test with WriteConcern = 1
2025-03-20 15:23:36,295 - INFO - Execution Time: 76.389971 seconds
2025-03-20 15:23:36,327 - INFO - {'_id': ObjectId('67dbef3b79b3ade8cb6b1367'), 'message': 'Message 1', 'counter': 100000}
2025-03-20 15:23:36,361 - INFO - Running test with WriteConcern = majority
2025-03-20 15:26:25,631 - INFO - Execution Time: 169.269738 seconds
2025-03-20 15:26:25,655 - INFO - {'_id': ObjectId('67dbef3b79b3ade8cb6b1367'), 'message': 'Message 1', 'counter': 100000}
```

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.

```
> rs.isMaster()
< {
    topologyVersion: {
      processId: ObjectId('67dbfcb2f73d0f8a180abad2'),
      counter: Long('7')
    },
    hosts: [ 'mongo1:27017', 'mongo2:27017', 'mongo3:27017' ],
    setName: 'rs0',
    setVersion: 1,
    ismaster: true,
    secondary: false,
    primary: 'mongo1:27017',
    me: 'mongo1:27017',
    electionId: ObjectId('7fffffff0000000000000008'),
```

stop mongo1

```
> rs.isMaster()
< {
    topologyVersion: {
      processId: ObjectId('67dbf97f66143116d94932b1'),
      counter: Long('10')
    },
    hosts: [ 'mongo1:27017', 'mongo2:27017', 'mongo3:27017' ],
    setName: 'rs0',
    setVersion: 1,
    ismaster: true,
    secondary: false,
    primary: 'mongo2:27017',
    me: 'mongo2:27017',
```

Looks like the stop of primary has no effect. See log:

```
2025-03-20 15:33:58,130 - INFO - Running test with WriteConcern = 1
2025-03-20 15:35:11,773 - INFO - Execution Time: 73.643247 seconds
2025-03-20 15:35:11,798 - INFO - {'_id': ObjectId('67dbef3b79b3ade8cb6b1367'), 'message': 'Message 1', 'counter': 100000}
```

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.

```
> rs.isMaster()
< {
    topologyVersion: {
      processId: ObjectId('67dc1b778452090a824e92f6'),
      counter: Long('7')
    },
    hosts: [ 'mongo1:27017', 'mongo2:27017', 'mongo3:27017' ],
    setName: 'rs0',
    setVersion: 1,
    ismaster: true,
    secondary: false,
    primary: 'mongo3:27017',
    me: 'mongo3:27017',
```

stop mongo3 and mongo2 was selected

```
> rs.isMaster()
< {
    topologyVersion: {
      processId: ObjectId('67dbf97f66143116d94932b1'),
      counter: Long('17')
    },
    hosts: [ 'mongo1:27017', 'mongo2:27017', 'mongo3:27017' ],
    setName: 'rs0',
    setVersion: 1,
    ismaster: true,
    secondary: false,
    primary: 'mongo2:27017',
    me: 'mongo2:27017',
```

```
2025-03-20 15:53:10,738 - INFO - Running test with WriteConcern = majority
2025-03-20 15:55:43,581 - INFO - Execution Time: 152.842895 seconds
2025-03-20 15:55:43,595 - INFO - {'_id': ObjectId('67dbef3b79b3ade8cb6b1367'), 'message': 'Message 1', 'counter': 100000}
```