Task 6 - Налаштування реплікації та перевірка відмовостійкості MongoDB

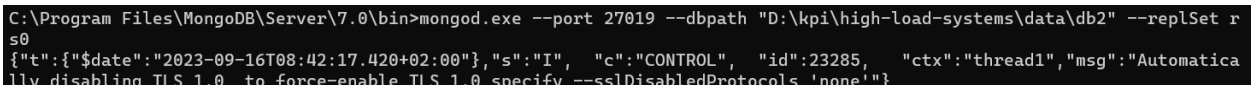
Виконав: Потужний Богдан (ФІ-03)

Ознайомтесь з реплікацію даних в MongoDB
http://docs.mongodb.org/manual/core/replication-introduction/

Завдання:

1) Налаштувати реплікацію в конфігурації: Primary with Two Secondary Members (всі ноди можуть бути запущені як окремі процеси або у Docker контейнерах) - http://docs.mongodb.org/manual/core/replica-set-architecture-three-members/
    - Deploy a Replica Set for Testing and Development- http://docs.mongodb.org/manual/tutorial/deploy-replica-set-for-testing/
    - http://www.tugberkugurlu.com/archive/setting-up-a-mongodb-replica-set-with-docker-and-connecting-to-it-with-a--net-core-app

    Запускаємо 3 ноди використовуючи наступну команду



    Під’єднуємося до головної ноди.

    І в ній налаштовуємо сет реплік

```
rs.initiate({
  _id: "rs0",
  members: [
    { _id: 0, host: "localhost:27018" },
    { _id: 1, host: "localhost:27019" },
    { _id: 2, host: "localhost:27020" }
  ]
})
```

    У відповідь отримали ОК.

    Подивимося на результат, ще за допомогою статусу

```
rs.status()
[
  {
    "$clusterTime": {
      "clusterTime": {"$timestamp": {"t": 1694846881, "i": 1}},
      "signature": {
        "hash": {"$binary": {"base64": "AAAAAAAAAAAAAAAAAAAAAAAAAAA=",
"subType": "00"}},
        "keyId": 0
      }
    },
    "date": {"$date": "2023-09-16T06:48:03.201Z"},
    "electionCandidateMetrics": {
      "lastElectionReason": "electionTimeout",
      "lastElectionDate": {"$date": "2023-09-16T06:44:31.497Z"},
      "electionTerm": 1,
      "lastCommittedOpTimeAtElection": {
        "ts": {"$timestamp": {"t": 1694846660, "i": 1}},
        "t": -1
```

```
      },
      "lastSeenOpTimeAtElection": {
        "ts": {"$timestamp": {"t": 1694846660, "i": 1}},
        "t": -1
      },
      "numVotesNeeded": 2,
      "priorityAtElection": 1,
      "electionTimeoutMillis": 10000,
      "numCatchUpOps": 0,
      "newTermStartDate": {"$date": "2023-09-16T06:44:31.563Z"},
      "wMajorityWriteAvailabilityDate": {"$date": "2023-09-
16T06:44:32.171Z"}
    },
    "heartbeatIntervalMillis": 2000,
    "lastStableRecoveryTimestamp": {"$timestamp": {"t": 1694846831,
"i": 1}},
    "majorityVoteCount": 2,
    "members": [
      {
        "_id": 0,
        "name": "localhost:27018",
        "health": 1,
        "state": 1,
        "stateStr": "PRIMARY",
        "uptime": 396,
        "optime": {
          "ts": {"$timestamp": {"t": 1694846881, "i": 1}},
          "t": 1
        },
        "optimeDate": {"$date": "2023-09-16T06:48:01.000Z"},
        "lastAppliedWallTime": {"$date": "2023-09-16T06:48:01.729Z"},
        "lastDurableWallTime": {"$date": "2023-09-16T06:48:01.729Z"},
        "syncSourceHost": "",
        "syncSourceId": -1,
        "infoMessage": "",
        "electionTime": {"$timestamp": {"t": 1694846671, "i": 1}},
        "electionDate": {"$date": "2023-09-16T06:44:31.000Z"},
        "configVersion": 1,
        "configTerm": 1,
        "self": true,
        "lastHeartbeatMessage": ""
      },
      {
        "_id": 1,
        "name": "localhost:27019",
        "health": 1,
        "state": 2,
        "stateStr": "SECONDARY",
        "uptime": 222,
        "optime": {
          "ts": {"$timestamp": {"t": 1694846881, "i": 1}},
          "t": 1
        },
        "optimeDurable": {
          "ts": {"$timestamp": {"t": 1694846881, "i": 1}},
          "t": 1
        },
        "optimeDate": {"$date": "2023-09-16T06:48:01.000Z"},
        "optimeDurableDate": {"$date": "2023-09-16T06:48:01.000Z"},
```

```
      "lastAppliedWallTime": {"$date": "2023-09-16T06:48:01.729Z"},
      "lastDurableWallTime": {"$date": "2023-09-16T06:48:01.729Z"},
      "lastHeartbeat": {"$date": "2023-09-16T06:48:02.431Z"},
      "lastHeartbeatRecv": {"$date": "2023-09-16T06:48:03.091Z"},
      "pingMs": 0,
      "lastHeartbeatMessage": "",
      "syncSourceHost": "localhost:27018",
      "syncSourceId": 0,
      "infoMessage": "",
      "configVersion": 1,
      "configTerm": 1
    },
    {
      "_id": 2,
      "name": "localhost:27020",
      "health": 1,
      "state": 2,
      "stateStr": "SECONDARY",
      "uptime": 222,
      "optime": {
        "ts": {"$timestamp": {"t": 1694846881, "i": 1}},
        "t": 1
      },
      "optimeDurable": {
        "ts": {"$timestamp": {"t": 1694846881, "i": 1}},
        "t": 1
      },
      "optimeDate": {"$date": "2023-09-16T06:48:01.000Z"},
      "optimeDurableDate": {"$date": "2023-09-16T06:48:01.000Z"},
      "lastAppliedWallTime": {"$date": "2023-09-16T06:48:01.729Z"},
      "lastDurableWallTime": {"$date": "2023-09-16T06:48:01.729Z"},
      "lastHeartbeat": {"$date": "2023-09-16T06:48:02.431Z"},
      "lastHeartbeatRecv": {"$date": "2023-09-16T06:48:01.559Z"},
      "pingMs": 0,
      "lastHeartbeatMessage": "",
      "syncSourceHost": "localhost:27018",
      "syncSourceId": 0,
      "infoMessage": "",
      "configVersion": 1,
      "configTerm": 1
    }
  ],
  "myState": 1,
  "ok": 1,
  "operationTime": {"$timestamp": {"t": 1694846881, "i": 1}},
  "optimes": {
    "lastCommittedOpTime": {
      "ts": {"$timestamp": {"t": 1694846881, "i": 1}},
      "t": 1
    },
    "lastCommittedWallTime": {"$date": "2023-09-16T06:48:01.729Z"},
    "readConcernMajorityOpTime": {
      "ts": {"$timestamp": {"t": 1694846881, "i": 1}},
      "t": 1
    },
    "appliedOpTime": {
      "ts": {"$timestamp": {"t": 1694846881, "i": 1}},
      "t": 1
    },
```

```
      "durableOpTime": {
        "ts": {"$timestamp": {"t": 1694846881, "i": 1}},
        "t": 1
      },
      "lastAppliedWallTime": {"$date": "2023-09-16T06:48:01.729Z"},
      "lastDurableWallTime": {"$date": "2023-09-16T06:48:01.729Z"}
    },
    "set": "rs0",
    "syncSourceHost": "",
    "syncSourceId": -1,
    "term": 1,
    "votingMembersCount": 3,
    "writableVotingMembersCount": 3,
    "writeMajorityCount": 2
  }
]
```


Бачимо, що усі ноди успішно підключені перейдемо до наступних завдань.

2) Продемонструвати Read Preference Modes: читання з *primary* і *secondary* node (http://docs.mongodb.org/manual/core/read-preference/)

```
db.collection.insertOne({data: "Some data"})

db.collection.find().readPref('primary')
db.collection.find().readPref('primaryPreferred')
db.collection.find().readPref('secondary')
db.collection.find().readPref('secondaryPreferred')
db.collection.find().readPref('nearest')
```

Result 26-2 | Result 26-3 | Result 26-4 | Result 26-5 | Result 26-6

	_id	data
1	6505558d217371775860e952	Some data

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

```
db.collection.insertOne({data: " Some data node 3 off"}, {writeConcern: {w: 3, wtimeout: 0}})
```

```
db.collection.insertOne({data: "Some data node 3 off"}, {writeConcern: {w: 3, wtimeout: 0}})  4 s
```

```
db.collection.find()
```

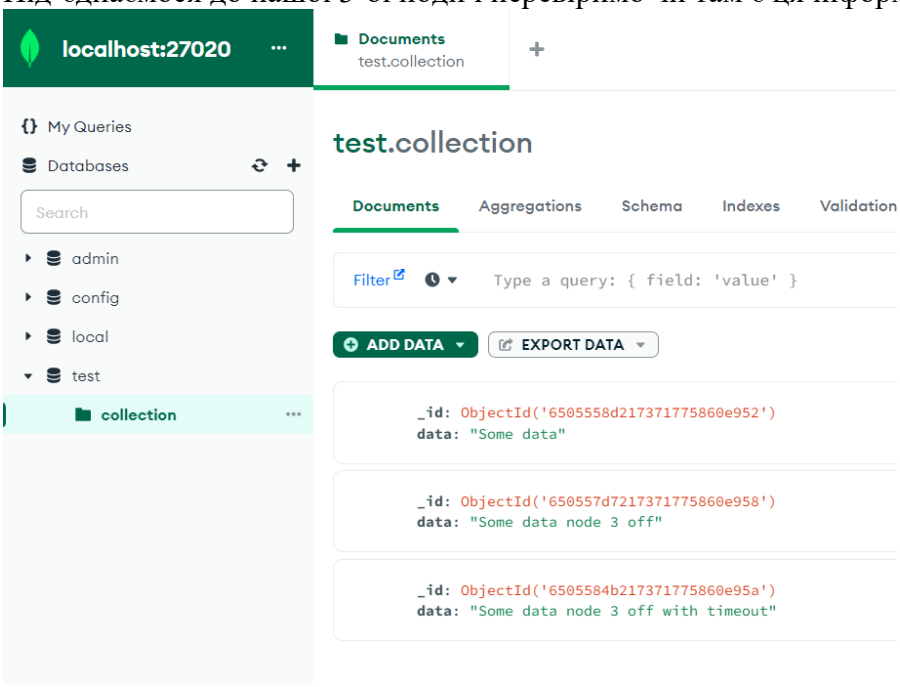
	_id	data
1	6505558d217371775860e952	Some data
2	650557d7217371775860e958	Some data node 3 off

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

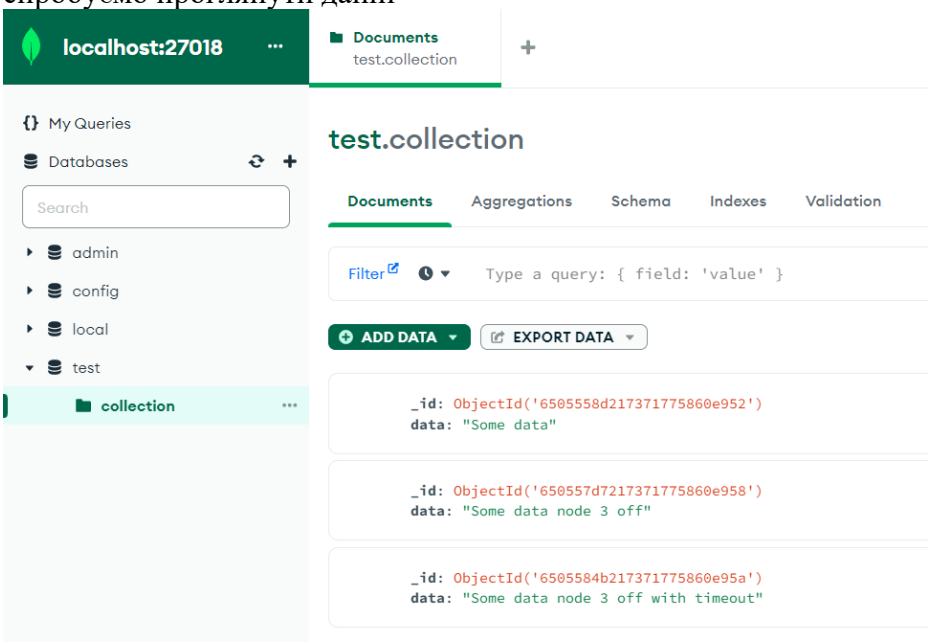
```
db.collection.insertOne({data: "Some data node 3 off with timeout"},
{writeConcern: {w: 3, wtimeout: 5000}})

waiting for replication timed out
```

Під’єднаємося до нашої 3-ої ноди і перевіримо чи там є ця інформація



Так вона була додана, тепер доєднаємося до головної ноди за наступним mongodb://localhost:27018/collection?readConcernLevel=majority доступом і спробуємо проглянути данні



Тут також усі дані відображаються

5) Продемонстрував перевибори primary node в відключивши поточний primary (Replica Set Elections) - http://docs.mongodb.org/manual/core/replica-set-elections/
Re-elections пройшли успішно, і був встановлений наступний порядок


```
[
  {
    "$clusterTime": {
      "clusterTime": {"$timestamp": {"t": 1694853417, "i": 1}},
      "signature": {
        "hash": {"$binary": {"base64": "AAAAAAAAAAAAAAAAAAAAAAAAAAA=",
"subType": "00"}},
        "keyId": 0
      }
    },
    "date": {"$date": "2023-09-16T08:37:06.678Z"},
    "electionCandidateMetrics": {
      "lastElectionReason": "stepUpRequestSkipDryRun",
      "lastElectionDate": {"$date": "2023-09-16T08:34:05.800Z"},
      "electionTerm": 2,
      "lastCommittedOpTimeAtElection": {
        "ts": {"$timestamp": {"t": 1694853237, "i": 1}},
        "t": 1
      },
      "lastSeenOpTimeAtElection": {
        "ts": {"$timestamp": {"t": 1694853237, "i": 1}},
        "t": 1
      },
      "numVotesNeeded": 2,
      "priorityAtElection": 1,
      "electionTimeoutMillis": 10000,
      "priorPrimaryMemberId": 0,
      "numCatchUpOps": 0,
      "newTermStartDate": {"$date": "2023-09-16T08:34:05.885Z"},
      "wMajorityWriteAvailabilityDate": {"$date": "2023-09-16T08:34:05.926Z"}
    },
    "electionParticipantMetrics": {
      "votedForCandidate": true,
      "electionTerm": 1,
      "lastVoteDate": {"$date": "2023-09-16T06:44:31.501Z"},
      "electionCandidateMemberId": 0,
      "voteReason": "",
      "lastAppliedOpTimeAtElection": {
        "ts": {"$timestamp": {"t": 1694846660, "i": 1}},
        "t": -1
      },
      "maxAppliedOpTimeInSet": {
        "ts": {"$timestamp": {"t": 1694846660, "i": 1}},
        "t": -1
      },
      "priorityAtElection": 1
    },
    "heartbeatIntervalMillis": 2000,
    "lastStableRecoveryTimestamp": {"$timestamp": {"t": 1694853396, "i": 1}},
```

```
    "majorityVoteCount": 2,
    "members": [
      {
        "_id": 0,
        "name": "localhost:27018",
        "health": 0,
        "state": 8,
        "stateStr": "(not reachable/healthy)",
        "uptime": 0,
        "optime": {
          "ts": {"$timestamp": {"t": 0, "i": 0}},
          "t": -1
        },
        "optimeDurable": {
          "ts": {"$timestamp": {"t": 0, "i": 0}},
          "t": -1
        },
        "optimeDate": {"$date": "1970-01-01T00:00:00.000Z"},
        "optimeDurableDate": {"$date": "1970-01-01T00:00:00.000Z"},
        "lastAppliedWallTime": {"$date": "2023-09-16T08:34:15.926Z"},
        "lastDurableWallTime": {"$date": "2023-09-16T08:34:15.926Z"},
        "lastHeartbeat": {"$date": "2023-09-16T08:37:04.430Z"},
        "lastHeartbeatRecv": {"$date": "2023-09-16T08:34:20.564Z"},
        "pingMs": 0,
        "lastHeartbeatMessage": "Error connecting to localhost:27018
(127.0.0.1:27018) :: caused by :: onInvoke :: caused by :: No connection
could be made because the target machine actively refused it.",
        "syncSourceHost": "",
        "syncSourceId": -1,
        "infoMessage": "",
        "configVersion": 1,
        "configTerm": 2
      },
      {
        "_id": 1,
        "name": "localhost:27019",
        "health": 1,
        "state": 1,
        "stateStr": "PRIMARY",
        "uptime": 6891,
        "optime": {
          "ts": {"$timestamp": {"t": 1694853417, "i": 1}},
          "t": 2
        },
        "optimeDate": {"$date": "2023-09-16T08:36:57.000Z"},
        "lastAppliedWallTime": {"$date": "2023-09-16T08:36:57.013Z"},
        "lastDurableWallTime": {"$date": "2023-09-16T08:36:57.013Z"},
        "syncSourceHost": "",
        "syncSourceId": -1,
        "infoMessage": "",
        "electionTime": {"$timestamp": {"t": 1694853245, "i": 1}},
        "electionDate": {"$date": "2023-09-16T08:34:05.000Z"},
        "configVersion": 1,
        "configTerm": 2,
        "self": true,
        "lastHeartbeatMessage": ""
      },
      {
        "_id": 2,
```

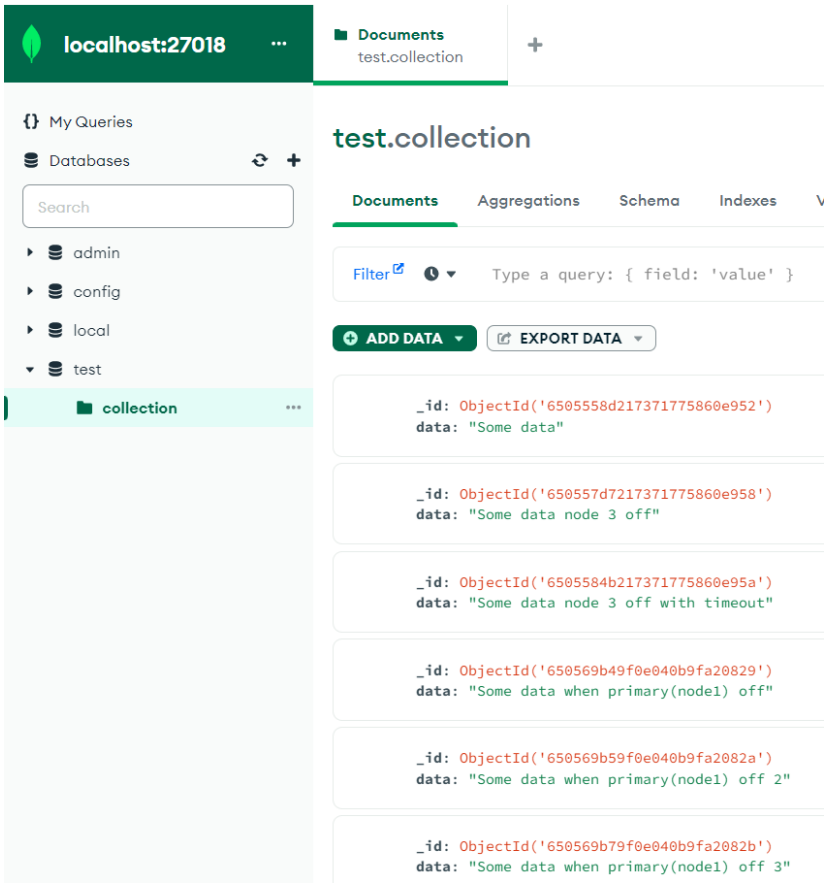
```
    "name": "localhost:27020",
    "health": 1,
    "state": 2,
    "stateStr": "SECONDARY",
    "uptime": 4268,
    "optime": {
      "ts": {"$timestamp": {"t": 1694853417, "i": 1}},
      "t": 2
    },
    "optimeDurable": {
      "ts": {"$timestamp": {"t": 1694853417, "i": 1}},
      "t": 2
    },
    "optimeDate": {"$date": "2023-09-16T08:36:57.000Z"},
    "optimeDurableDate": {"$date": "2023-09-16T08:36:57.000Z"},
    "lastAppliedWallTime": {"$date": "2023-09-16T08:36:57.013Z"},
    "lastDurableWallTime": {"$date": "2023-09-16T08:36:57.013Z"},
    "lastHeartbeat": {"$date": "2023-09-16T08:37:05.645Z"},
    "lastHeartbeatRecv": {"$date": "2023-09-16T08:37:06.102Z"},
    "pingMs": 0,
    "lastHeartbeatMessage": "",
    "syncSourceHost": "localhost:27019",
    "syncSourceId": 1,
    "infoMessage": "",
    "configVersion": 1,
    "configTerm": 2
  }
],
"myState": 1,
"ok": 1,
"operationTime": {"$timestamp": {"t": 1694853417, "i": 1}},
"optimes": {
  "lastCommittedOpTime": {
    "ts": {"$timestamp": {"t": 1694853417, "i": 1}},
    "t": 2
  },
  "lastCommittedWallTime": {"$date": "2023-09-16T08:36:57.013Z"},
  "readConcernMajorityOpTime": {
    "ts": {"$timestamp": {"t": 1694853417, "i": 1}},
    "t": 2
  },
  "appliedOpTime": {
    "ts": {"$timestamp": {"t": 1694853417, "i": 1}},
    "t": 2
  },
  "durableOpTime": {
    "ts": {"$timestamp": {"t": 1694853417, "i": 1}},
    "t": 2
  },
  "lastAppliedWallTime": {"$date": "2023-09-16T08:36:57.013Z"},
  "lastDurableWallTime": {"$date": "2023-09-16T08:36:57.013Z"}
},
"set": "rs0",
"syncSourceHost": "",
"syncSourceId": -1,
"term": 2,
"votingMembersCount": 3,
"writableVotingMembersCount": 3,
"writeMajorityCount": 2
```

```
  }
]
```

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

```
db.collection.insertOne({data: "Some data when primary(node1) off"});
db.collection.insertOne({data: "Some data when primary(node1) off 2"});
db.collection.insertOne({data: "Some data when primary(node1) off 3"});
```

Відновлюємо роботу попередньо головної ноди і перевіряємо чи відбувся запис



Усі данні було додано до попередньо головної ноди

6) Привести кластер до неконсистентного стану користуючись моментом часу коли *primary node* не відразу помічає відсутність *secondary node*
   - відключивши дві *secondary node* протягом 5 сек. на мастер записати значення (з w:1) і перевірити, що воно записалось
     Перевірили що головна нода знаходиться на порту 27018, а отже вимикаємо 27019 та 27020

```
> db.collection.insertOne({data: "2 nodes turned off"}, {writeConcern: {w: 1}})
  db.collection.find().readConcern("linearizable")
< {
    _id: ObjectId('6505558d217371775860e952'),
    data: 'Some data'
  }
```

…

```
  {
    _id: ObjectId('658d680bd7ffacc9cc4d5d66'),
    data: '2 nodes turned off'
  }
> db.collection.find().readConcern("local")
< {
    _id: ObjectId('6505558d217371775860e952'),
    data: 'Some data'
  }
```

…

```
  {
    _id: ObjectId('658d680bd7ffacc9cc4d5d66'),
    data: '2 nodes turned off'
  }
> db.collection.find().readConcern("majority")
< {
    _id: ObjectId('6505558d217371775860e952'),
    data: 'Some data'
  }
```

- спробувати зчитати це значення з різними рівнями *read concern* - `readConcern: {level: <"majority"|"local"| "linearizable">}`
  Вивід на попередніх слайдах
- включити дві інші ноди таким чином, щоб вони не бачили попереднього мастера (його можна відключити) і дочекатись поки вони оберуть нового мастера

```
{
  set: 'rs0',
  date: 2023-12-28T12:28:13.717Z,
  myState: 1,
  term: Long('12'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
```

```
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 0, i: 0 }), t:
Long('-1') },
    lastCommittedWallTime: 1970-01-01T00:00:00.000Z,
    appliedOpTime: { ts: Timestamp({ t: 1703766487, i: 1 }), t:
Long('12') },
    durableOpTime: { ts: Timestamp({ t: 1703766487, i: 1 }), t:
Long('12') },
    lastAppliedWallTime: 2023-12-28T12:28:07.207Z,
    lastDurableWallTime: 2023-12-28T12:28:07.207Z
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1703766027, i: 2
}),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: 2023-12-28T12:27:45.105Z,
    electionTerm: Long('12'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 0, i: 0
}), t: Long('-1') },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1703766027, i:
2 }), t: Long('11') },
    numVotesNeeded: 2,
    priorityAtElection: 1,
    electionTimeoutMillis: Long('10000'),
    numCatchUpOps: Long('0'),
    newTermStartDate: 2023-12-28T12:27:47.200Z
  },
  members: [
    {
      _id: 0,
      name: 'localhost:27018',
      health: 0,
      state: 8,
      stateStr: '(not reachable/healthy)',
      uptime: 0,
      optime: [Object],
      optimeDurable: [Object],
      optimeDate: 1970-01-01T00:00:00.000Z,
      optimeDurableDate: 1970-01-01T00:00:00.000Z,
      lastAppliedWallTime: 1970-01-01T00:00:00.000Z,
      lastDurableWallTime: 1970-01-01T00:00:00.000Z,
      lastHeartbeat: 2023-12-28T12:28:13.612Z,
      lastHeartbeatRecv: 1970-01-01T00:00:00.000Z,
      pingMs: Long('0'),
      lastHeartbeatMessage: 'Error connecting to localhost:27018
(127.0.0.1:27018) :: caused by :: onInvoke :: caused by :: No
```

```
connection could be made because the target machine actively
refused it.',
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      configVersion: -1,
      configTerm: -1
    },
    {
      _id: 1,
      name: 'localhost:27019',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 39,
      optime: [Object],
      optimeDurable: [Object],
      optimeDate: 2023-12-28T12:20:27.000Z,
      optimeDurableDate: 2023-12-28T12:20:27.000Z,
      lastAppliedWallTime: 2023-12-28T12:20:27.154Z,
      lastDurableWallTime: 2023-12-28T12:20:27.154Z,
      lastHeartbeat: 2023-12-28T12:28:13.216Z,
      lastHeartbeatRecv: 2023-12-28T12:28:11.717Z,
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: 'localhost:27020',
      syncSourceId: 2,
      infoMessage: '',
      configVersion: 4,
      configTerm: 12
    },
    {
      _id: 2,
      name: 'localhost:27020',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 41,
      optime: [Object],
      optimeDate: 2023-12-28T12:28:07.000Z,
      lastAppliedWallTime: 2023-12-28T12:28:07.207Z,
      lastDurableWallTime: 2023-12-28T12:28:07.207Z,
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: 'Could not find member to sync from',
      electionTime: Timestamp({ t: 1703766465, i: 1 }),
      electionDate: 2023-12-28T12:27:45.000Z,
      configVersion: 4,
      configTerm: 12,
      self: true,
      lastHeartbeatMessage: ''
```

```
    }
  ],
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1703766487, i: 1 }),
    signature: {
      hash:
Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1703766487, i: 1 })
}
```

Третю ноду обрано лідером

- підключити (включити) попередню primary-ноду до кластеру і подивитись, що сталось зі значенням яке було на неї записано

Зчитали з ноди 2 та бачимо присутність інформації

7) Земулювати eventual consistency за допомогою установки затримки реплікації для репліки https://docs.mongodb.com/manual/tutorial/configure-a-delayed-replica-set-member/

```
> rs.reconfig(cfg)
< {
    ok: 1,
    '$clusterTime': {
      clusterTime: Timestamp({ t: 1694883169, i: 1 }),
      signature: {
        hash: Binary(Buffer.from("0000000000000000000000000000000000000000", "hex"), 0),
        keyId: Long("0")
      }
    },
    operationTime: Timestamp({ t: 1694883169, i: 1 })
  }
```

8) Лишити *primary* та *secondary* для якої налаштована затримка реплікації. Записати декілька значень. Спробувати прочитати значення з `readConcern: {level: "linearizable"}`
Має бути затримка поки значення не реплікуються на більшість нод

```
> db.collection.find().readConcern(level = "linearizable")
  db.collection.find().readConcern(level = "linearizable")
  {
    _id: ObjectId("6505e06781040feab9f67fc4"),
    data: 'delayed1'
  }
```

Опис додаткових команд Replication Reference - http://docs.mongodb.org/manual/reference/replication/

Вимогу до оформлення протоколу:
Завдання здається особисто без протоколу, або надсилається протокол який має містити:

- команди та результати їх виконання