Advanced Databases

MongoDB CA

Student Number: d20125299

Student Name: Luke Hallinan

Programme Code: TU856

1. Setting up the cluster and replication
   A cluster of 3 nodes was used. These are shown below.

```
members: [
  {
    _id: 0,
    name: 'D20125299-1:27017',
    health: 1,
    state: 1,
    stateStr: 'PRIMARY',
    uptime: 928,
    optime: { ts: Timestamp({ t: 1671153396, i: 1 }), t: Long("1") },
    optimeDate: ISODate("2022-12-16T01:16:36.000Z"),
    lastAppliedWallTime: ISODate("2022-12-16T01:16:36.632Z"),
    lastDurableWallTime: ISODate("2022-12-16T01:16:36.632Z"),
    syncSourceHost: '',
    syncSourceId: -1,
    infoMessage: '',
    electionTime: Timestamp({ t: 1671153346, i: 1 }),
    electionDate: ISODate("2022-12-16T01:15:46.000Z"),
    configVersion: 1,
    configTerm: 1,
    self: true,
    lastHeartbeatMessage: ''
  },
  {
    _id: 1,
    name: 'D20125299-2:27017',
    health: 1,
    state: 2,
    stateStr: 'SECONDARY',
    uptime: 66,
    optime: { ts: Timestamp({ t: 1671153396, i: 1 }), t: Long("1") },
    optimeDurable: { ts: Timestamp({ t: 1671153396, i: 1 }), t: Long("1") },
    optimeDate: ISODate("2022-12-16T01:16:36.000Z"),
    optimeDurableDate: ISODate("2022-12-16T01:16:36.000Z"),
    lastAppliedWallTime: ISODate("2022-12-16T01:16:36.632Z"),
    lastDurableWallTime: ISODate("2022-12-16T01:16:36.632Z"),
    lastHeartbeat: ISODate("2022-12-16T01:16:40.557Z"),
    lastHeartbeatRecv: ISODate("2022-12-16T01:16:41.562Z"),
    pingMs: Long("0"),
    lastHeartbeatMessage: '',
    syncSourceHost: 'D20125299-1:27017',
    syncSourceId: 0,
    infoMessage: '',
    configVersion: 1,
    configTerm: 1
  },
```

```
{
  _id: 2,
  name: 'D20125299-3:27017',
  health: 1,
  state: 2,
  stateStr: 'SECONDARY',
  uptime: 66,
  optime: { ts: Timestamp({ t: 1671153396, i: 1 }), t: Long("1") },
  optimeDurable: { ts: Timestamp({ t: 1671153396, i: 1 }), t: Long("1") },
  optimeDate: ISODate("2022-12-16T01:16:36.000Z"),
  optimeDurableDate: ISODate("2022-12-16T01:16:36.000Z"),
  lastAppliedWallTime: ISODate("2022-12-16T01:16:36.632Z"),
  lastDurableWallTime: ISODate("2022-12-16T01:16:36.632Z"),
  lastHeartbeat: ISODate("2022-12-16T01:16:40.559Z"),
  lastHeartbeatRecv: ISODate("2022-12-16T01:16:41.562Z"),
  pingMs: Long("0"),
  lastHeartbeatMessage: '',
  syncSourceHost: 'D20125299-1:27017',
  syncSourceId: 0,
  infoMessage: '',
  configVersion: 1,
  configTerm: 1
}
```


The first is set as the primary with the other two being secondary. All are in a single cluster. Both shild nodes are linked to the first on syncSourceHost: 'D20125299-1:27017'.

2. Porting the data to Mongo

```
D20125299RepSet [direct: primary] test> use advanceddb
switched to db advanceddb
D20125299RepSet [direct: primary] advanceddb> db.factresults.find().pretty()
[
  {
    _id: ObjectId("639bcc9822f96becfa1115e7"),
    player_sk: 5,
    p_name: 'John',
    p_sname: 'McDonald',
    prize: 2000,
    year: 2014
  },
  {
    _id: ObjectId("639bcc9822f96becfa1115e8"),
    player_sk: 10,
    p_name: 'Martha',
    p_sname: 'Ross',
    prize: 8000,
    year: 2014
  },
  {
    _id: ObjectId("639bcc9822f96becfa1115e9"),
    player_sk: 1,
    p_name: 'Tiger',
    p_sname: 'Woods',
    prize: 16000,
    year: 2014
  },
  {
    _id: ObjectId("639bcc9822f96becfa1115ea"),
    player_sk: 8,
    p_name: 'Paul',
    p_sname: 'Bin',
    prize: 12000,
    year: 2014
  },
  {
    _id: ObjectId("639bcc9822f96becfa1115eb"),
    player_sk: 2,
    p_name: 'Jane',
    p_sname: 'Smith',
    prize: 9000,
    year: 2014
  },
```

```
{
  _id: ObjectId("639bcc9822f96becfa1115ec"),
  player_sk: 6,
  p_name: 'Mario',
  p_sname: 'Baggio',
  prize: 6000,
  year: 2014
},
{
  _id: ObjectId("639bcc9822f96becfa1115ed"),
  player_sk: 9,
  p_name: 'Peter',
  p_sname: 'Flynn',
  prize: 9400,
  year: 2014
}
```

Working with the Golf collection in MongoDB:

```
D20125299RepSet [direct: primary] advanceddb> db.factresults.find({p_name : "Paul"}).explain("executionStats")
{
  explainVersion: '1',
  queryPlanner: {
    namespace: 'advanceddb.factresults',
    indexFilterSet: false,
    parsedQuery: { p_name: { '$eq': 'Paul' } },
    queryHash: '9C5D431D',
    planCacheKey: '9C5D431D',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'COLLSCAN',
      filter: { p_name: { '$eq': 'Paul' } },
      direction: 'forward'
    },
    rejectedPlans: []
  },
  executionStats: {
    executionSuccess: true,
    nReturned: 1,
    executionTimeMillis: 0,
    totalKeysExamined: 0,
    totalDocsExamined: 7,
    executionStages: {
      stage: 'COLLSCAN',
      filter: { p_name: { '$eq': 'Paul' } },
      nReturned: 1,
      executionTimeMillisEstimate: 0,
      works: 9,
      advanced: 1,
      needTime: 7,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      direction: 'forward',
      docsExamined: 7
    }
  },
```

```
command: {
  find: 'factresults',
  filter: { p_name: 'Paul' },
  '$db': 'advanceddb'
},
serverInfo: {
  host: '18209ba622c6',
  port: 27017,
  version: '6.0.3',
  gitVersion: 'f803681c3ae19817d31958965850193de067c516'
},
serverParameters: {
  internalQueryFacetBufferSizeBytes: 104857600,
  internalQueryFacetMaxOutputDocSizeBytes: 104857600,
  internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
  internalDocumentSourceGroupMaxMemoryBytes: 104857600,
  internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
  internalQueryProhibitBlockingMergeOnMongoS: 0,
  internalQueryMaxAddToSetBytes: 104857600,
  internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
},
ok: 1,
'$clusterTime': {
  clusterTime: Timestamp({ t: 1671155336, i: 1 }),
  signature: {
    hash: Binary(Buffer.from("0000000000000000000000000000000000000000", "hex"), 0),
    keyId: Long("0")
  }
},
operationTime: Timestamp({ t: 1671155336, i: 1 })
```


This query will find anyone with the name paul in the data. it gives the number returned being 1, the total documents examines which is 7 and the timestamp of the operation. It does a collection scan with a filter on the whole document.

a. Adding a secondary index to golf data on a text field.

```
D20125299RepSet [direct: primary] advanceddb> db.factresults.find({p_name : "Paul"}).explain("executionStats")
{_name_1
  explainVersion: '1',
  queryPlanner: {
    namespace: 'advanceddb.factresults',
    indexFilterSet: false,
    parsedQuery: { p_name: { '$eq': 'Paul' } },
    queryHash: '9C5D431D',
    planCacheKey: 'CC6406ED',
    maxIndexedOrSolutionsReached: false,
    maxIndexedAndSolutionsReached: false,
    maxScansToExplodeReached: false,
    winningPlan: {
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { p_name: 1 },
        indexName: 'p_name_1',
        isMultiKey: false,
        multiKeyPaths: { p_name: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: { p_name: [ '["Paul", "Paul"]' ] }
      }
    },
    rejectedPlans: []
  },
```

```
executionStats: {
  executionSuccess: true,
  nReturned: 1,
  executionTimeMillis: 1,
  totalKeysExamined: 1,
  totalDocsExamined: 1,
  executionStages: {
    stage: 'FETCH',
    nReturned: 1,
    executionTimeMillisEstimate: 0,
    works: 2,
    advanced: 1,
    needTime: 0,
    needYield: 0,
    saveState: 0,
    restoreState: 0,
    isEOF: 1,
    docsExamined: 1,
    alreadyHasObj: 0,
    inputStage: {
      stage: 'IXSCAN',
      nReturned: 1,
      executionTimeMillisEstimate: 0,
      works: 2,
      advanced: 1,
      needTime: 0,
      needYield: 0,
      saveState: 0,
      restoreState: 0,
      isEOF: 1,
      keyPattern: { p_name: 1 },
      indexName: 'p_name_1',
      isMultiKey: false,
      multiKeyPaths: { p_name: [] },
      isUnique: false,
      isSparse: false,
      isPartial: false,
      indexVersion: 2,
      direction: 'forward',
      indexBounds: { p_name: [ '["Paul", "Paul"]' ] },
      keysExamined: 1,
      seeks: 1,
      dupsTested: 0,
      dupsDropped: 0
    }
  }
},
```

```
  command: {
    find: 'factresults',
    filter: { p_name: 'Paul' },
    '$db': 'advanceddb'
  },
  serverInfo: {
    host: '18209ba622c6',
    port: 27017,
    version: '6.0.3',
    gitVersion: 'f803681c3ae19817d31958965850193de067c516'
  },
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
  },
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1671155846, i: 1 }),
    signature: {
      hash: Binary(Buffer.from("0000000000000000000000000000000000000000", "hex"), 0),
      keyId: Long("0")
    }
  },
  operationTime: Timestamp({ t: 1671155846, i: 1 })
}
```

Added index to p_name field. This shows that an index was used by IXSCAN as well as the index called p_name_1. Also returned is the information from the non indexed such as number of values found and the time stamp. It has total keys examined which is 1.

3. Working with aggregation in MongoDB:
    a. Create an aggregation pipeline

```
D20125299RepSet [direct: primary] advanceddb> 
[ { _id: 2014, AvgPrizes: 8914.285714285714 } ]
```

```
D20125299RepSet [direct: primary] advanceddb> db.factresults.aggregate([{$group: { _id: "$year", AvgPrizes: { $avg: "$prize" } }}, {$sort: { AvgPrize: -1 }}]).explain("executionStats")
{ { _id: 2014, AvgPrizes: 8914.285714285714 } ]
  explainVersion: '2',
  stages: [
    {
      '$cursor': {
        queryPlanner: {
          namespace: 'advanceddb.factresults',
          indexFilterSet: false,
          parsedQuery: {},
          queryHash: '334AEDF2',
          planCacheKey: '334AEDF2',
          maxIndexedOrSolutionsReached: false,
          maxIndexedAndSolutionsReached: false,
          maxScansToExplodeReached: false,
          winningPlan: {
            queryPlan: {
              stage: 'GROUP',
              planNodeId: 2,
              inputStage: {
                stage: 'COLLSCAN',
                planNodeId: 1,
                filter: {},
                direction: 'forward'
              }
            },
            slotBasedPlan: {
              slots: '$$RESULT=s13 env: { s1 = TimeZoneDatabase(Atlantic/Canary...Factory) (timeZoneDB), s3 = Timestamp(1671156796, 1) (CLUSTER_TIME), s4 = 1671156801118 (NOW), s2 = Nothing (SEARCH_META) }',
              stages: '[2] mkobj s13 [_id = s8, AvgPrizes = s12] true false \n' +
                '[2] project [s12 = if (s11 == 0, null, doubleDoubleSumFinalize (s10) / s11)] \n' +
                '[2] group [s8] [s10 = aggDoubleDoubleSum (s9), s11 = sum (let [11.0 = s9] if (! exists (11.0) || typeMatch (11.0, 1088) || ! isNumber (11.0), 0, 1))] \n' +
                '[2] project [s9 = getField (s5, "prize")] \n' +
                '[2] project [s8 = fillEmpty (s7, null)] \n' +
                '[2] project [s7 = getField (s5, "year")] \n' +
                '[1] scan s5 s6 none none none none [] @"48a70215-a9ce-44b5-aba9-bbd6ae4dab37" true false '
            }
          },
          rejectedPlans: []
        },
```

```
executionStats: {
  executionSuccess: true,
  nReturned: 1,
  executionTimeMillis: 1,
  totalKeysExamined: 0,
  totalDocsExamined: 7,
  executionStages: {
    stage: 'mkobj',
    planNodeId: 2,
    nReturned: 1,
    executionTimeMillisEstimate: 0,
    opens: 1,
    closes: 1,
    saveState: 1,
    restoreState: 1,
    isEOF: 1,
    objSlot: 13,
    fields: [],
    projectFields: [ '_id', 'AvgPrizes' ],
    projectSlots: [ Long("8"), Long("12") ],
    forceNewObject: true,
    returnOldObject: false,
    inputStage: {
      stage: 'project',
      planNodeId: 2,
      nReturned: 1,
      executionTimeMillisEstimate: 0,
      opens: 1,
      closes: 1,
      saveState: 1,
      restoreState: 1,
      isEOF: 1,
      projections: {
        '12': 'if (s11 == 0, null, doubleDoubleSumFinalize (s10) / s11) '
      },
      inputStage: {
        stage: 'group',
        planNodeId: 2,
        nReturned: 1,
        executionTimeMillisEstimate: 0,
        opens: 1,
        closes: 1,
        saveState: 1,
        restoreState: 1,
        isEOF: 1,
        groupBySlots: [ Long("8") ],
```

```
expressions: {
  '10': 'aggDoubleDoubleSum (s9) ',
  '11': 'sum (let [l1.0 = s9] if (! exists (l1.0) || typeMatch (l1.0, 1088) || ! isNumber (l1.0), 0, 1)) '
},
usedDisk: false,
spilledRecords: 0,
spilledBytesApprox: 0,
inputStage: {
  stage: 'project',
  planNodeId: 2,
  nReturned: 7,
  executionTimeMillisEstimate: 0,
  opens: 1,
  closes: 1,
  saveState: 1,
  restoreState: 1,
  isEOF: 1,
  projections: { '9': 'getField (s5, "prize") ' },
  inputStage: {
    stage: 'project',
    planNodeId: 2,
    nReturned: 7,
    executionTimeMillisEstimate: 0,
    opens: 1,
    closes: 1,
    saveState: 1,
    restoreState: 1,
    isEOF: 1,
    projections: { '8': 'fillEmpty (s7, null) ' },
    inputStage: {
      stage: 'project',
      planNodeId: 2,
      nReturned: 7,
      executionTimeMillisEstimate: 0,
      opens: 1,
      closes: 1,
      saveState: 1,
      restoreState: 1,
      isEOF: 1,
      projections: { '7': 'getField (s5, "year") ' },
```

```
                    inputStage: {
                      stage: 'scan',
                      planNodeId: 1,
                      nReturned: 7,
                      executionTimeMillisEstimate: 0,
                      opens: 1,
                      closes: 1,
                      saveState: 1,
                      restoreState: 1,
                      isEOF: 1,
                      numReads: 7,
                      recordSlot: 5,
                      recordIdSlot: 6,
                      fields: [],
                      outputSlots: []
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    nReturned: Long("1"),
    executionTimeMillisEstimate: Long("0")
  },
  {
    '$sort': { sortKey: { AvgPrize: -1 } },
    totalDataSizeSortedBytesEstimate: Long("245"),
    usedDisk: false,
    spills: Long("0"),
    nReturned: Long("1"),
    executionTimeMillisEstimate: Long("0")
  }
],
serverInfo: {
  host: '18209ba622c6',
  port: 27017,
  version: '6.0.3',
  gitVersion: 'f803681c3ae19817d31958965850193de067c516'
},
```

This shows the method used for the aggregation pipeline. It uses a collection scan and a group. The slots based plan allows the aggregation to work using valuesrather than documents.

b. Add relevant indexes and reorder your stages.

```
D20125299RepSet [direct: primary] advanceddb> db.factresults.aggregate([{$group: { _id: "$year", AvgPrizes: { $avg: "$prize" } }}, {$sort: { AvgPrize: -1 }}]).explain("executionStats")
{ear_1
  explainVersion: '2',
  stages: [
    {
      '$cursor': {
       queryPlanner: {
         namespace: 'advanceddb.factresults',
         indexFilterSet: false,
         parsedQuery: {},
         queryHash: '334AEDF2',
         planCacheKey: '334AEDF2',
         maxIndexedOrSolutionsReached: false,
         maxIndexedAndSolutionsReached: false,
         maxScansToExplodeReached: false,
         winningPlan: {
           queryPlan: {
             stage: 'GROUP',
             planNodeId: 2,
             inputStage: {
               stage: 'COLLSCAN',
               planNodeId: 1,
               filter: {},
               direction: 'forward'
             }
           },
           slotBasedPlan: {
             slots: '$$RESULT=s13 env: { s3 = Timestamp(1671157156, 1) (CLUSTER_TIME), s1 = TimeZoneDatabase(Atlantic/Canary...Factory) (timeZoneDB), s4 = 1671157158399 (NOW), s2 = Nothing (SEARCH_META) }',
             stages: '[2] mkobj s13 [_id = s8, AvgPrizes = s12] true false \n' +
               '[2] project [s12 = if (s11 == 0, null, doubleDoubleSumFinalize (s10) / s11)] \n' +
               '[2] group [s8] [s10 = aggDoubleDoubleSum (s9), s11 = sum (let [l1.0 = s9] if (! exists (l1.0) || typeMatch (l1.0, 1088) || ! isNumber (l1.0), 0, 1))] \n' +
               '[2] project [s9 = getField (s5, "prize")] \n' +
               '[2] project [s8 = fillEmpty (s7, null)] \n' +
               '[2] project [s7 = getField (s5, "year")] \n' +
               '[1] scan s5 s6 none none none none [] @"48a70215-a9ce-44b5-aba9-bbd6ae4dab37" true false '
           }
         },
         rejectedPlans: []
       },
```

```
executionStats: {
  executionSuccess: true,
  nReturned: 1,
  executionTimeMillis: 0,
  totalKeysExamined: 0,
  totalDocsExamined: 7,
  executionStages: {
    stage: 'mkobj',
    planNodeId: 2,
    nReturned: 1,
    executionTimeMillisEstimate: 0,
    opens: 1,
    closes: 1,
    saveState: 1,
    restoreState: 1,
    isEOF: 1,
    objSlot: 13,
    fields: [],
    projectFields: [ '_id', 'AvgPrizes' ],
    projectSlots: [ Long("8"), Long("12") ],
    forceNewObject: true,
    returnOldObject: false,
    inputStage: {
      stage: 'project',
      planNodeId: 2,
      nReturned: 1,
      executionTimeMillisEstimate: 0,
      opens: 1,
      closes: 1,
      saveState: 1,
      restoreState: 1,
      isEOF: 1,
      projections: {
        '12': 'if (s11 == 0, null, doubleDoubleSumFinalize (s10) / s11) '
      },
      inputStage: {
        stage: 'group',
        planNodeId: 2,
        nReturned: 1,
        executionTimeMillisEstimate: 0,
        opens: 1,
        closes: 1,
        saveState: 1,
        restoreState: 1,
        isEOF: 1,
        groupBySlots: [ Long("8") ],
```

```
expressions: {
  '10': 'aggDoubleDoubleSum (s9) ',
  '11': 'sum (let [l1.0 = s9] if (! exists (l1.0) || typeMatch (l1.0, 1088) || ! isNumber (l1.0), 0, 1)) '
},
usedDisk: false,
spilledRecords: 0,
spilledBytesApprox: 0,
inputStage: {
  stage: 'project',
  planNodeId: 2,
  nReturned: 7,
  executionTimeMillisEstimate: 0,
  opens: 1,
  closes: 1,
  saveState: 1,
  restoreState: 1,
  isEOF: 1,
  projections: { '9': 'getField (s5, "prize") ' },
  inputStage: {
    stage: 'project',
    planNodeId: 2,
    nReturned: 7,
    executionTimeMillisEstimate: 0,
    opens: 1,
    closes: 1,
    saveState: 1,
    restoreState: 1,
    isEOF: 1,
    projections: { '8': 'fillEmpty (s7, null) ' },
    inputStage: {
      stage: 'project',
      planNodeId: 2,
      nReturned: 7,
      executionTimeMillisEstimate: 0,
      opens: 1,
      closes: 1,
      saveState: 1,
      restoreState: 1,
      isEOF: 1,
      projections: { '7': 'getField (s5, "year") ' },
```

```
                    projections: { _id: 0, getField(35, "year") },
                    inputStage: {
                      stage: 'scan',
                      planNodeId: 1,
                      nReturned: 7,
                      executionTimeMillisEstimate: 0,
                      opens: 1,
                      closes: 1,
                      saveState: 1,
                      restoreState: 1,
                      isEOF: 1,
                      numReads: 7,
                      recordSlot: 5,
                      recordIdSlot: 6,
                      fields: [],
                      outputSlots: []
                    }
                  }
                }
              }
            }
          }
        }
      }
    },
    nReturned: Long("1"),
    executionTimeMillisEstimate: Long("0")
  },
  {
    '$sort': { sortKey: { AvgPrize: -1 } },
    totalDataSizeSortedBytesEstimate: Long("245"),
    usedDisk: false,
    spills: Long("0"),
    nReturned: Long("1"),
    executionTimeMillisEstimate: Long("0")
  }
],
serverInfo: {
  host: '18209ba622c6',
  port: 27017,
  version: '6.0.3',
  gitVersion: 'f803681c3ae19817d31958965850193de067c516'
},
```

```
  serverParameters: {
    internalQueryFacetBufferSizeBytes: 104857600,
    internalQueryFacetMaxOutputDocSizeBytes: 104857600,
    internalLookupStageIntermediateDocumentMaxSizeBytes: 104857600,
    internalDocumentSourceGroupMaxMemoryBytes: 104857600,
    internalQueryMaxBlockingSortMemoryUsageBytes: 104857600,
    internalQueryProhibitBlockingMergeOnMongoS: 0,
    internalQueryMaxAddToSetBytes: 104857600,
    internalDocumentSourceSetWindowFieldsMaxMemoryBytes: 104857600
  },
  command: {
    aggregate: 'factresults',
    pipeline: [
      { '$group': { _id: '$year', AvgPrizes: { '$avg': '$prize' } } },
      { '$sort': { AvgPrize: -1 } }
    ],
    cursor: {},
    '$db': 'advanceddb'
  },
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1671157156, i: 1 }),
    signature: {
      hash: Binary(Buffer.from("0000000000000000000000000000000000000000", "hex"), 0),
      keyId: Long("0")
    }
  },
  operationTime: Timestamp({ t: 1671157156, i: 1 })
}
D20125299RepSet [direct: primary] advanceddb>
```

an index was added to the year column. In the case of this dataset there was not a lot of change but with larger more varied ones this can change the method and speed greatly.

<<use explain, capture the output and include it here, comment on what is happening>>

4. Replication working

```
{
  set: 'D20125299RepSet',
  date: ISODate("2022-12-16T02:23:54.404Z"),
  myState: 1,
  term: Long("2"),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long("2000"),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1671157430, i: 1 }), t: Long("2") },
    lastCommittedWallTime: ISODate("2022-12-16T02:23:50.219Z"),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1671157430, i: 1 }), t: Long("2") },
    appliedOpTime: { ts: Timestamp({ t: 1671157430, i: 1 }), t: Long("2") },
    durableOpTime: { ts: Timestamp({ t: 1671157430, i: 1 }), t: Long("2") },
    lastAppliedWallTime: ISODate("2022-12-16T02:23:50.219Z"),
    lastDurableWallTime: ISODate("2022-12-16T02:23:50.219Z")
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1671157420, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'stepUpRequestSkipDryRun',
    lastElectionDate: ISODate("2022-12-16T02:23:10.196Z"),
    electionTerm: Long("2"),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1671157386, i: 1 }), t: Long("1") },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1671157386, i: 1 }), t: Long("1") },
    numVotesNeeded: 2,
    priorityAtElection: 1,
    electionTimeoutMillis: Long("10000"),
    priorPrimaryMemberId: 0,
    numCatchUpOps: Long("0"),
    newTermStartDate: ISODate("2022-12-16T02:23:10.216Z"),
    wMajorityWriteAvailabilityDate: ISODate("2022-12-16T02:23:11.213Z")
  },
  electionParticipantMetrics: {
    votedForCandidate: true,
    electionTerm: Long("1"),
    lastVoteDate: ISODate("2022-12-16T01:15:46.537Z"),
    electionCandidateMemberId: 0,
    voteReason: '',
    lastAppliedOpTimeAtElection: { ts: Timestamp({ t: 1671153335, i: 1 }), t: Long("-1") },
    maxAppliedOpTimeInSet: { ts: Timestamp({ t: 1671153335, i: 1 }), t: Long("-1") },
    priorityAtElection: 1
  },
  members: [
```

```
members: [
  {
    _id: 0,
    name: 'D20125299-1:27017',
    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") },
    optimeDate: ISODate("1970-01-01T00:00:00.000Z"),
    optimeDurableDate: ISODate("1970-01-01T00:00:00.000Z"),
    lastAppliedWallTime: ISODate("2022-12-16T02:23:20.218Z"),
    lastDurableWallTime: ISODate("2022-12-16T02:23:20.218Z"),
    lastHeartbeat: ISODate("2022-12-16T02:23:44.217Z"),
    lastHeartbeatRecv: ISODate("2022-12-16T02:23:19.222Z"),
    pingMs: Long("0"),
    lastHeartbeatMessage: "Couldn't get a connection within the time limit",
    syncSourceHost: '',
    syncSourceId: -1,
    infoMessage: '',
    configVersion: 1,
    configTerm: 2
  },
  {
    _id: 1,
    name: 'D20125299-2:27017',
    health: 1,
    state: 1,
    stateStr: 'PRIMARY',
    uptime: 5318,
    optime: { ts: Timestamp({ t: 1671157430, i: 1 }), t: Long("2") },
    optimeDate: ISODate("2022-12-16T02:23:50.000Z"),
    lastAppliedWallTime: ISODate("2022-12-16T02:23:50.219Z"),
    lastDurableWallTime: ISODate("2022-12-16T02:23:50.219Z"),
    syncSourceHost: '',
    syncSourceId: -1,
    infoMessage: '',
    electionTime: Timestamp({ t: 1671157390, i: 1 }),
    electionDate: ISODate("2022-12-16T02:23:10.000Z"),
    configVersion: 1,
    configTerm: 2,
    self: true,
    lastHeartbeatMessage: ''
  },
```

```
    {
      _id: 2,
      name: 'D20125299-3:27017',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 4098,
      optime: { ts: Timestamp({ t: 1671157430, i: 1 }), t: Long("2") },
      optimeDurable: { ts: Timestamp({ t: 1671157430, i: 1 }), t: Long("2") },
      optimeDate: ISODate("2022-12-16T02:23:50.000Z"),
      optimeDurableDate: ISODate("2022-12-16T02:23:50.000Z"),
      lastAppliedWallTime: ISODate("2022-12-16T02:23:50.219Z"),
      lastDurableWallTime: ISODate("2022-12-16T02:23:50.219Z"),
      lastHeartbeat: ISODate("2022-12-16T02:23:54.230Z"),
      lastHeartbeatRecv: ISODate("2022-12-16T02:23:53.258Z"),
      pingMs: Long("0"),
      lastHeartbeatMessage: '',
      syncSourceHost: 'D20125299-2:27017',
      syncSourceId: 1,
      infoMessage: '',
      configVersion: 1,
      configTerm: 2
    }
  ],
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1671157430, i: 1 }),
    signature: {
      hash: Binary(Buffer.from("0000000000000000000000000000000000000000", "hex"), 0),
      keyId: Long("0")
    }
  },
  operationTime: Timestamp({ t: 1671157430, i: 1 })
}

C:\Users\lukeh\Documents\College\4th-Year\Databases\lab 11>
```

As can be seen above the first instance was stopped. When this happen the second instance was autimatically reassigned to PRIMARY while the first instance is now (not reachable/healthy). This is a great way to protect agains data loss and interuption by have an automatic switch if one fails.