PRACTICAL 6

REPLICATION USING MONGODB

Implement Replication

You are a database administrator for a company, and you need to set up a MongoDB replica set to ensure high availability and data redundancy. Perform the following tasks:

· Initialize a replica set with three nodes on different ports (27017, 27018, 27019).

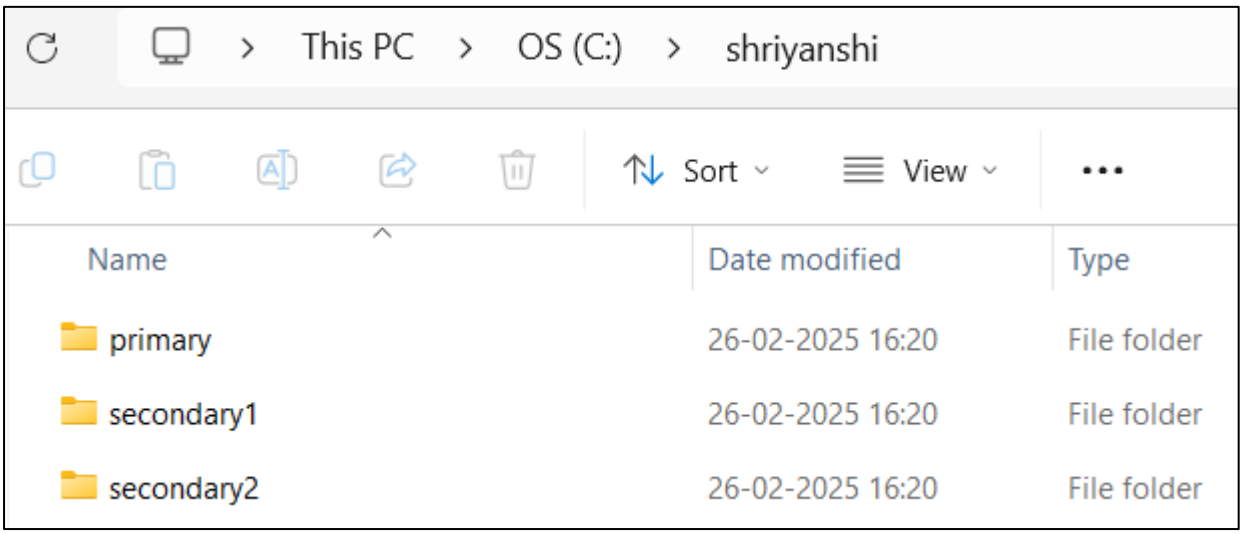
· Check the status of the replica set.

· Add a new secondary node to the existing replica set.

· Simulate a failover scenario by stepping down the primary and observing the election of a new primary.

· Check replication status and read from a secondary node using readPreference.

Create folders for the respective servers.



Now open Windows Powershell ISE and type the following commands.

mongod --port=2717 --dbpath="C:\shriyanshi\primary" --replSet="text-replica-set"

```
PS C:\Users\dell> mongod --port=2717 --dbpath="C:\shriyanshi\primary" --replSet="text-replica-set"
{"t":{"$date":"2025-02-26T16:34:46.018+05:30"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"thread1","msg":"Automatica
lly disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2025-02-26T16:34:46.019+05:30"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"thread1","msg":"Initialize
d wire specification","attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":17},"incomingInterna
lClient":{"minWireVersion":0,"maxWireVersion":17},"outgoing":{"minWireVersion":6,"maxWireVersion":17},"isInternalClient"
:true}}}
{"t":{"$date":"2025-02-26T16:34:46.979+05:30"},"s":"I",  "c":"NETWORK",  "id":4648602, "ctx":"thread1","msg":"Implicit T
CP FastOpen in use."}
{"t":{"$date":"2025-02-26T16:34:46.982+05:30"},"s":"I",  "c":"REPL",     "id":5123008, "ctx":"thread1","msg":"Successful
ly registered PrimaryOnlyService","attr":{"service":"TenantMigrationDonorService","namespace":"config.tenantMigrationDon
ors"}}
```

Shriyanshi Maurya | L024

mongod --port=2727 --dbpath="C:\shriyanshi\secondary1" --replSet="text-replica-set"

```
PS C:\Users\dell> mongod --port=2727 --dbpath="C:\shriyanshi\secondary1" --replSet="text-replica-set"
{"t":{"$date":"2025-02-26T16:35:28.414+05:30"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"-","msg":"Initialized wire
 specification","attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":17},"incomingInternalClien
t":{"minWireVersion":0,"maxWireVersion":17},"outgoing":{"minWireVersion":6,"maxWireVersion":17},"isInternalClient":true}
}}
{"t":{"$date":"2025-02-26T16:35:28.415+05:30"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"-","msg":"Automatically di
sabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2025-02-26T16:35:29.382+05:30"},"s":"I",  "c":"NETWORK",  "id":4648602, "ctx":"thread1","msg":"Implicit T
CP FastOpen in use."}
{"t":{"$date":"2025-02-26T16:35:29.383+05:30"},"s":"I",  "c":"REPL",     "id":5123008, "ctx":"thread1","msg":"Successful
ly registered PrimaryOnlyService","attr":{"service":"TenantMigrationDonorService","namespace":"config.tenantMigrationDon
ors"}}
```

mongod --port=2737 --dbpath="C:\shriyanshi\secondary2" --replSet="text-replica-set"

```
PS C:\Users\dell> mongod --port=2737 --dbpath="C:\shriyanshi\secondary2" --replSet="text-replica-set"
{"t":{"$date":"2025-02-26T16:36:05.471+05:30"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"-","msg":"Automatically di
sabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2025-02-26T16:36:06.457+05:30"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"thread1","msg":"Initialize
d wire specification","attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVersion":17},"incomingInterna
lClient":{"minWireVersion":0,"maxWireVersion":17},"outgoing":{"minWireVersion":6,"maxWireVersion":17},"isInternalClient"
:true}}}
{"t":{"$date":"2025-02-26T16:36:06.457+05:30"},"s":"I",  "c":"NETWORK",  "id":4648602, "ctx":"thread1","msg":"Implicit T
CP FastOpen in use."}
{"t":{"$date":"2025-02-26T16:36:06.460+05:30"},"s":"I",  "c":"REPL",     "id":5123008, "ctx":"thread1","msg":"Successful
ly registered PrimaryOnlyService","attr":{"service":"TenantMigrationDonorService","namespace":"config.tenantMigrationDon
ors"}}
```

mongosh --host="localhost:2717"

```
PS C:\Users\dell> mongosh --host="localhost:2717"
Current Mongosh Log ID: 67bef63a282e27987efa4213
Connecting to:          mongodb://localhost:2717/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.
4.0
Using MongoDB:          6.0.3
Using Mongosh:          2.4.0

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-02-26T16:34:47.083+05:30: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
   2025-02-26T16:34:47.084+05:30: This server is bound to localhost. Remote systems will be unable to connect to this se
rver. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or with --
bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable
 this warning
------
```

rs.initiate()

```
test> rs.initiate()
{
  info2: 'no configuration specified. Using a default configuration for the set',
  me: 'localhost:2717',
  ok: 1
}
```

rs.add({host:"localhost:2727"})

```
text-replica-set [direct: other] test> rs.add({host:"localhost:2727"})
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1740568193, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1740568193, i: 1 })
}
```

rs.add({host:"localhost:2737"})

```
text-replica-set [direct: primary] test> rs.add({host:"localhost:2737"})
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1740568230, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1740568230, i: 1 })
}
```

rs.status()

```
text-replica-set [direct: primary] test> rs.status()
{
  set: 'text-replica-set',
  date: ISODate('2025-02-26T11:11:02.889Z'),
  myState: 1,
  term: Long('1'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1740568257, i: 1 }), t: Long('1') },
    lastCommittedWallTime: ISODate('2025-02-26T11:10:57.762Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1740568257, i: 1 }), t: Long('1') },
    appliedOpTime: { ts: Timestamp({ t: 1740568257, i: 1 }), t: Long('1') },
    durableOpTime: { ts: Timestamp({ t: 1740568257, i: 1 }), t: Long('1') },
    lastAppliedWallTime: ISODate('2025-02-26T11:10:57.762Z'),
    lastDurableWallTime: ISODate('2025-02-26T11:10:57.762Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1740568217, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2025-02-26T11:09:27.648Z'),
    electionTerm: Long('1'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1740568167, i: 1 }), t: Long('-1') },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1740568167, i: 1 }), t: Long('-1') },
    numVotesNeeded: 1,
    priorityAtElection: 1,
    electionTimeoutMillis: Long('10000'),
    newTermStartDate: ISODate('2025-02-26T11:09:27.686Z'),
    wMajorityWriteAvailabilityDate: ISODate('2025-02-26T11:09:27.708Z')
  },
```

mongosh --host="localhost:2727"

```
PS C:\Users\dell> mongosh --host="localhost:2727"
Current Mongosh Log ID: 67bef72c15b5c14522fa4213
Connecting to:          mongodb://localhost:2727/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.
4.0
Using MongoDB:          6.0.3
Using Mongosh:          2.4.0

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-02-26T16:35:29.471+05:30: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
   2025-02-26T16:35:29.473+05:30: This server is bound to localhost. Remote systems will be unable to connect to this se
rver. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or with --
bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable
 this warning
------
```

mongosh --host="localhost:2737"

```
PS C:\Users\dell> mongosh --host="localhost:2737"
Current Mongosh Log ID: 67bef764380b920ed2fa4213
Connecting to:          mongodb://localhost:2737/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.
4.0
Using MongoDB:          6.0.3
Using Mongosh:          2.4.0

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-02-26T16:36:06.534+05:30: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
   2025-02-26T16:36:06.535+05:30: This server is bound to localhost. Remote systems will be unable to connect to this se
rver. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or with --
bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to disable
 this warning
------
```

Now, go to primary instance and type the command

show dbs;

```
text-replica-set [direct: primary] test> show dbs;
admin     80.00 KiB
config   160.00 KiB
local    404.00 KiB
text-replica-set [direct: primary] test>
```

Go to secondary1 instance and type command

show dbs;

```
text-replica-set [direct: secondary] test> show dbs;
admin     80.00 KiB
config   216.00 KiB
local    404.00 KiB
```

Go to secondary2 instance and type command

show dbs;

```
text-replica-set [direct: secondary] test> show dbs;
admin     80.00 KiB
config   220.00 KiB
local    404.00 KiB
```

Adding a database in primary server .

use practice

```
text-replica-set [direct: primary] test> use practice
switched to db practice
text-replica-set [direct: primary] practice>
```

db.users.insert({"name":"shriyanshi","age":21})

```
text-replica-set [direct: primary] practice> db.users.insert({"name":"shriyanshi","age":21})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('67bef8f0282e27987efa4215') }
}
```

db.users.find()

```
text-replica-set [direct: primary] practice> db.users.find()
[
  { _id: ObjectId('67bef8cb282e27987efa4214'), name: 'shriyanshi' },
  {
    _id: ObjectId('67bef8f0282e27987efa4215'),
    name: 'shriyanshi',
    age: 21
  }
]
```

Now, go to secondary1 and type command.

use practice

db.users.find()

```
text-replica-set [direct: secondary] test> use practice
switched to db practice
text-replica-set [direct: secondary] practice> db.users.find()
[
  { _id: ObjectId('67bef8cb282e27987efa4214'), name: 'shriyanshi' },
  {
    _id: ObjectId('67bef8f0282e27987efa4215'),
    name: 'shriyanshi',
    age: 21
  }
]
text-replica-set [direct: secondary] practice>
```

Now, go to secondary2 and type command.

use practice

db.users.find()

```
text-replica-set [direct: secondary] test> use practice
switched to db practice
text-replica-set [direct: secondary] practice> db.users.find()
[
  { _id: ObjectId('67bef8cb282e27987efa4214'), name: 'shriyanshi' },
  {
    _id: ObjectId('67bef8f0282e27987efa4215'),
    name: 'shriyanshi',
    age: 21
  }
]
```

Add more records in primary

db.users.insertMany([{"name":"Pranit","age":22},{"name":"Tanmay","age":25}])

```
text-replica-set [direct: primary] practice> db.users.insertMany([{"name":"Pranit","age":22},{"name":"Tanmay","age":25}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67bef9ff282e27987efa4216'),
    '1': ObjectId('67bef9ff282e27987efa4217')
  }
}
```

Check if records are visible in secondary1 and secondary2

```
text-replica-set [direct: secondary] practice> db.users.find();
[
  { _id: ObjectId('67bef8cb282e27987efa4214'), name: 'shriyanshi' },
  {
    _id: ObjectId('67bef8f0282e27987efa4215'),
    name: 'shriyanshi',
    age: 21
  },
  {
    _id: ObjectId('67bef9ff282e27987efa4216'),
    name: 'Pranit',
    age: 22
  },
  {
    _id: ObjectId('67bef9ff282e27987efa4217'),
    name: 'Tanmay',
    age: 25
  }
]
```

Update command in primary server.

db.users.update({"name":"Pranit"},{$set:{"email":"pranit43@gmail.com"}})

```
text-replica-set [direct: primary] practice> db.users.update({"name":"Pranit"},{$set:{"email":"pranit43@gmail.com"}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Check if update happened in secondary1 and secondary2

```
text-replica-set [direct: secondary] practice> db.users.find({"name":"Pranit"})
[
  {
    _id: ObjectId('67bef9ff282e27987efa4216'),
    name: 'Pranit',
    age: 22,
    email: 'pranit43@gmail.com'
  }
]
```

Delete command in primary server

db.users.deleteOne({"name":"Tanmay"})

db.users.find()

```
text-replica-set [direct: primary] practice> db.users.deleteOne({"name":"Tanmay"})
{ acknowledged: true, deletedCount: 1 }
text-replica-set [direct: primary] practice> db.users.find()
[
  { _id: ObjectId('67bef8cb282e27987efa4214'), name: 'shriyanshi' },
  {
    _id: ObjectId('67bef8f0282e27987efa4215'),
    name: 'shriyanshi',
    age: 21
  },
  {
    _id: ObjectId('67bef9ff282e27987efa4216'),
    name: 'Pranit',
    age: 22,
    email: 'pranit43@gmail.com'
  }
]
```

Check if delete happened in secondary1 and secondary2

```
text-replica-set [direct: secondary] practice> db.users.find()
[
  { _id: ObjectId('67bef8cb282e27987efa4214'), name: 'shriyanshi' },
  {
    _id: ObjectId('67bef8f0282e27987efa4215'),
    name: 'shriyanshi',
    age: 21
  },
  {
    _id: ObjectId('67bef9ff282e27987efa4216'),
    name: 'Pranit',
    age: 22,
    email: 'pranit43@gmail.com'
  }
]
```