Firdaus Mulla | L007

Mithibai College Department of Computer Science
Msc(Data Science and AI)

Practical 6: Replication using Mongodb

Date: 07/02/2025 **Submission Date: 14/02/2025**

Write-up: -

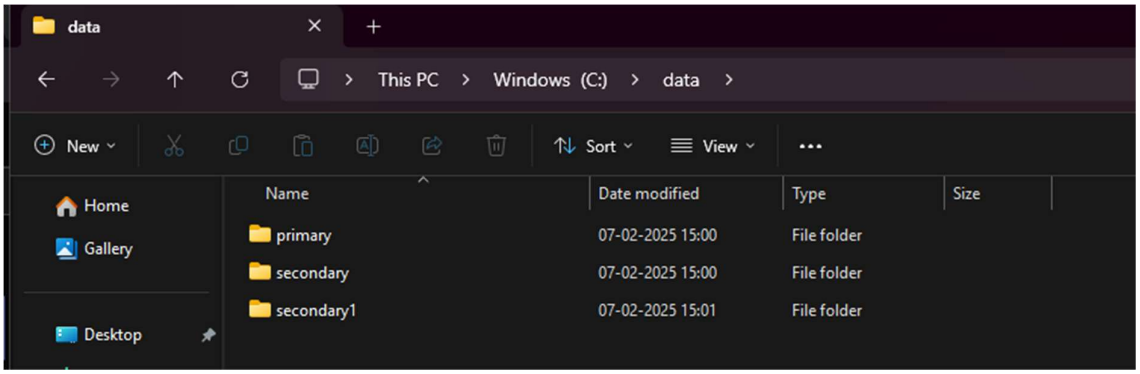
- Replication
- Replication Architecture
- Replication benefits and limitations

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:

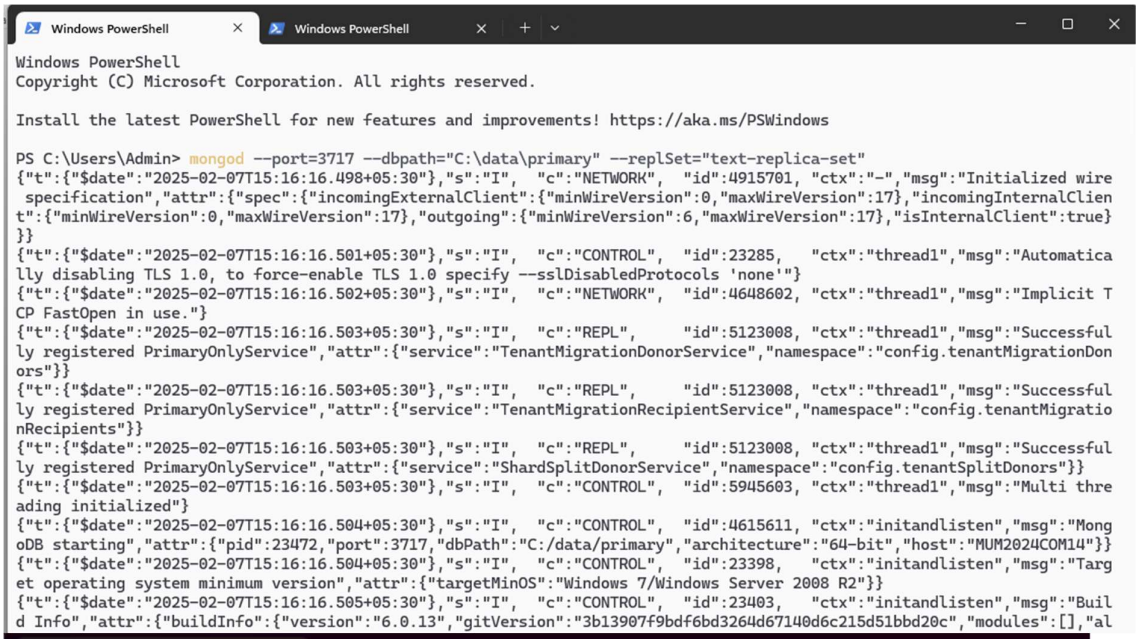
- Initialize a replica set with three nodes on different ports (2717, 2727, 2737).
- Check the status of the replica set.
- Add a new secondary node to the existing replica set.
- Simulate a failover scenario by stepping down the primary and observing the election of a new primary.
- Check replication status and read from a secondary node using readPreference

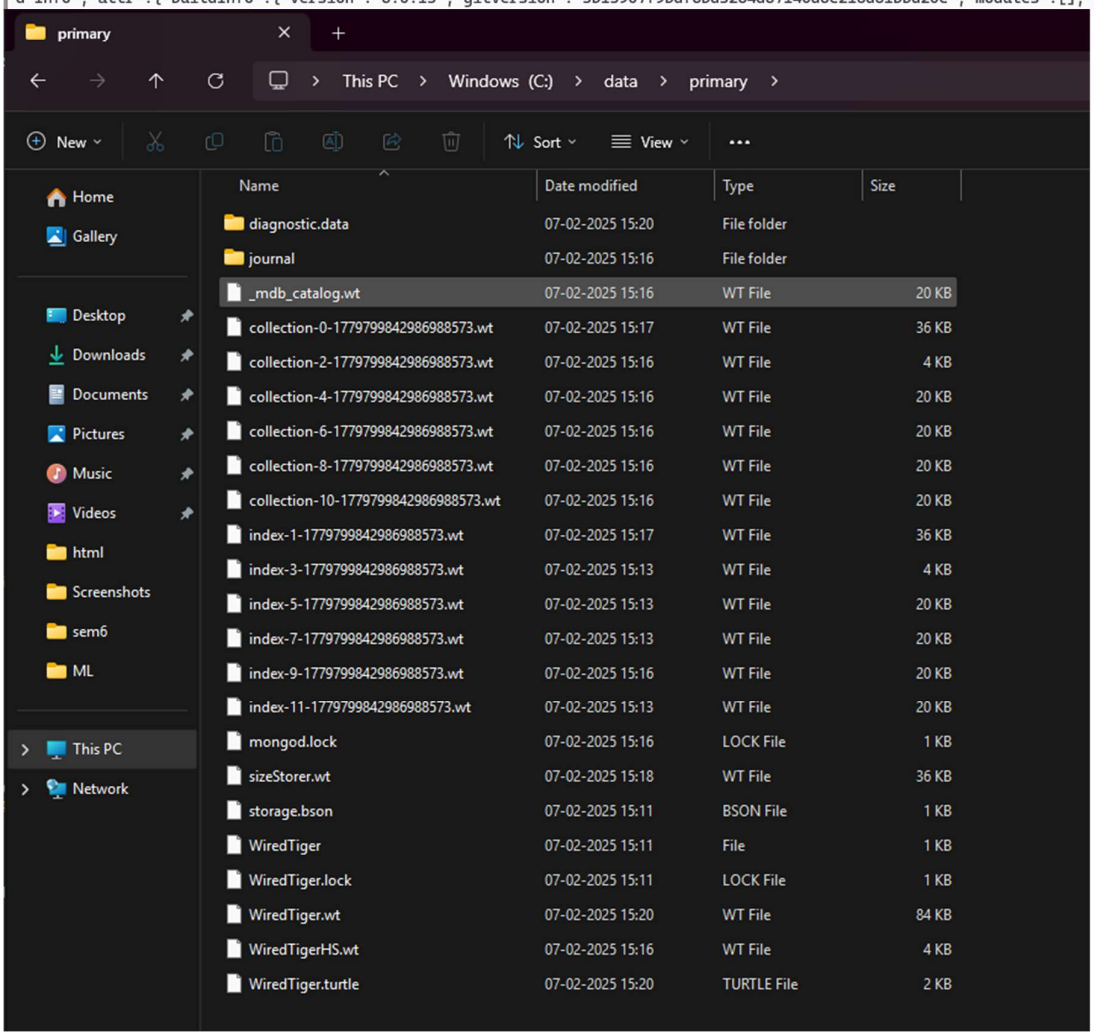
Create a data folder in C drive and then create 3 folders in it.



Open Command Prompt or Powershell and do the following commands.
Note: All the instances should be done in a new instance
mongod --port=3717 --dbpath="C:\data\primary" --replSet="text-replica-set"

Firdaus Mulla | L007

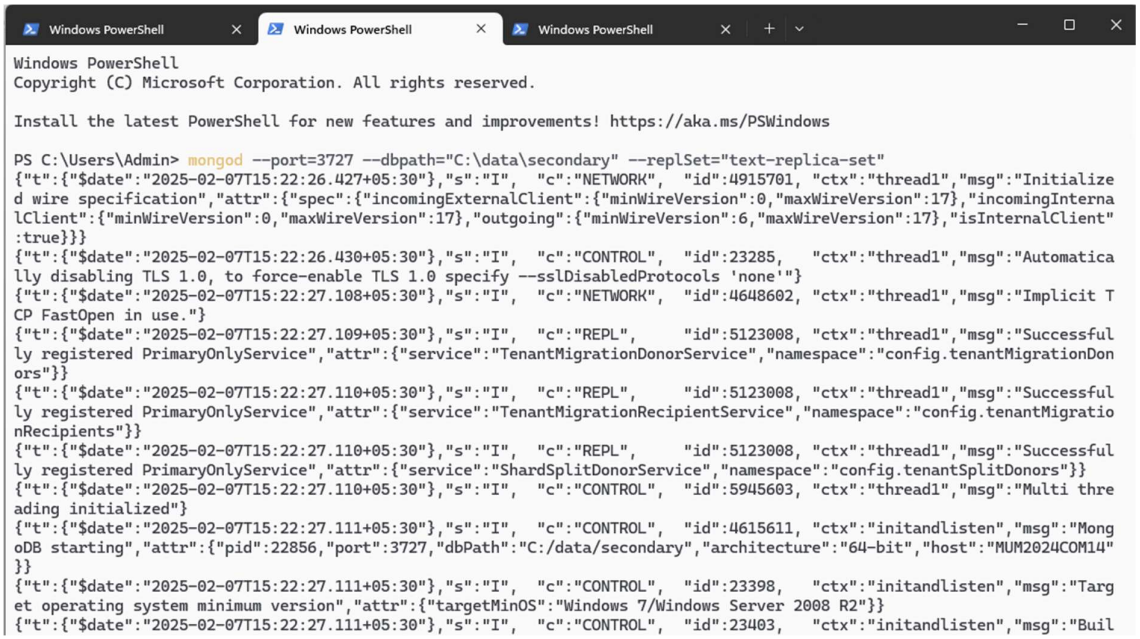




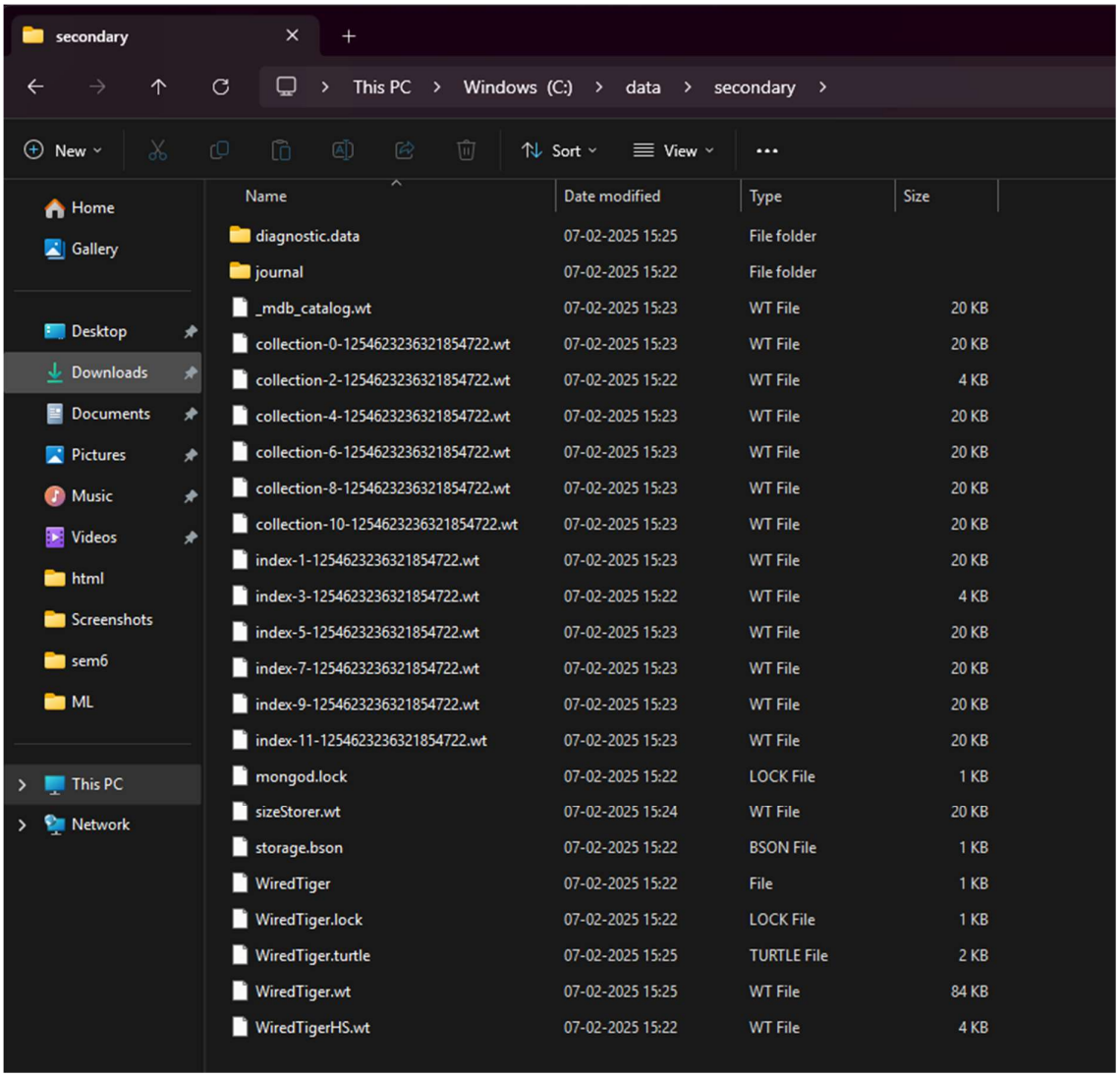
For secondary

mongod --port=3727 --dbpath="C:\data\secondary" --replSet="text-replica-set"

Firdaus Mulla | L007



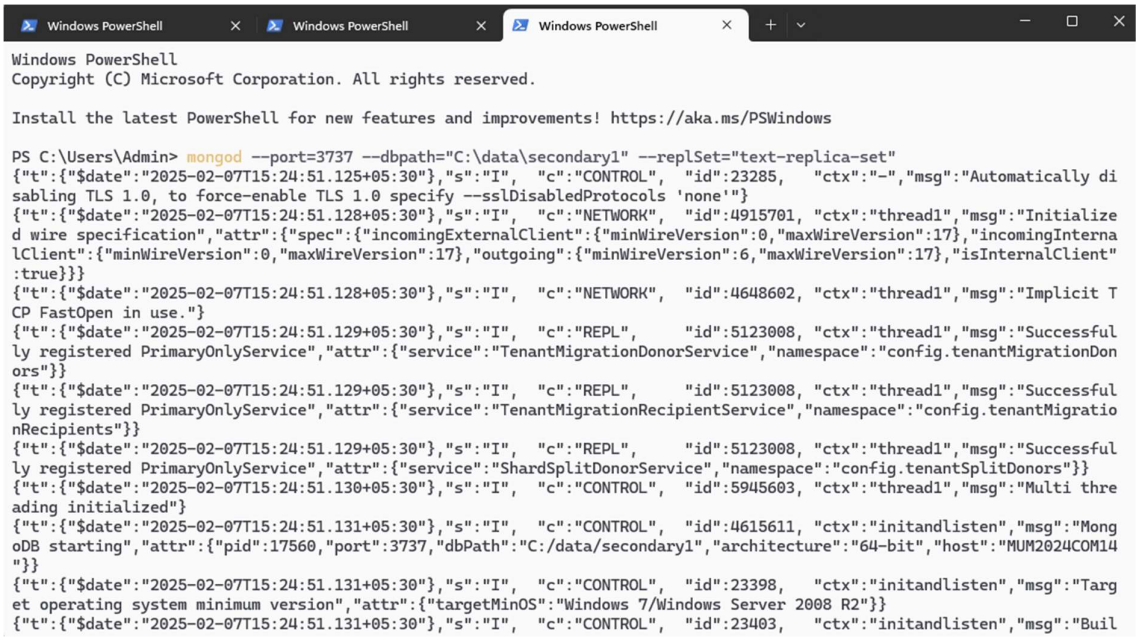
Firdaus Mulla | L007



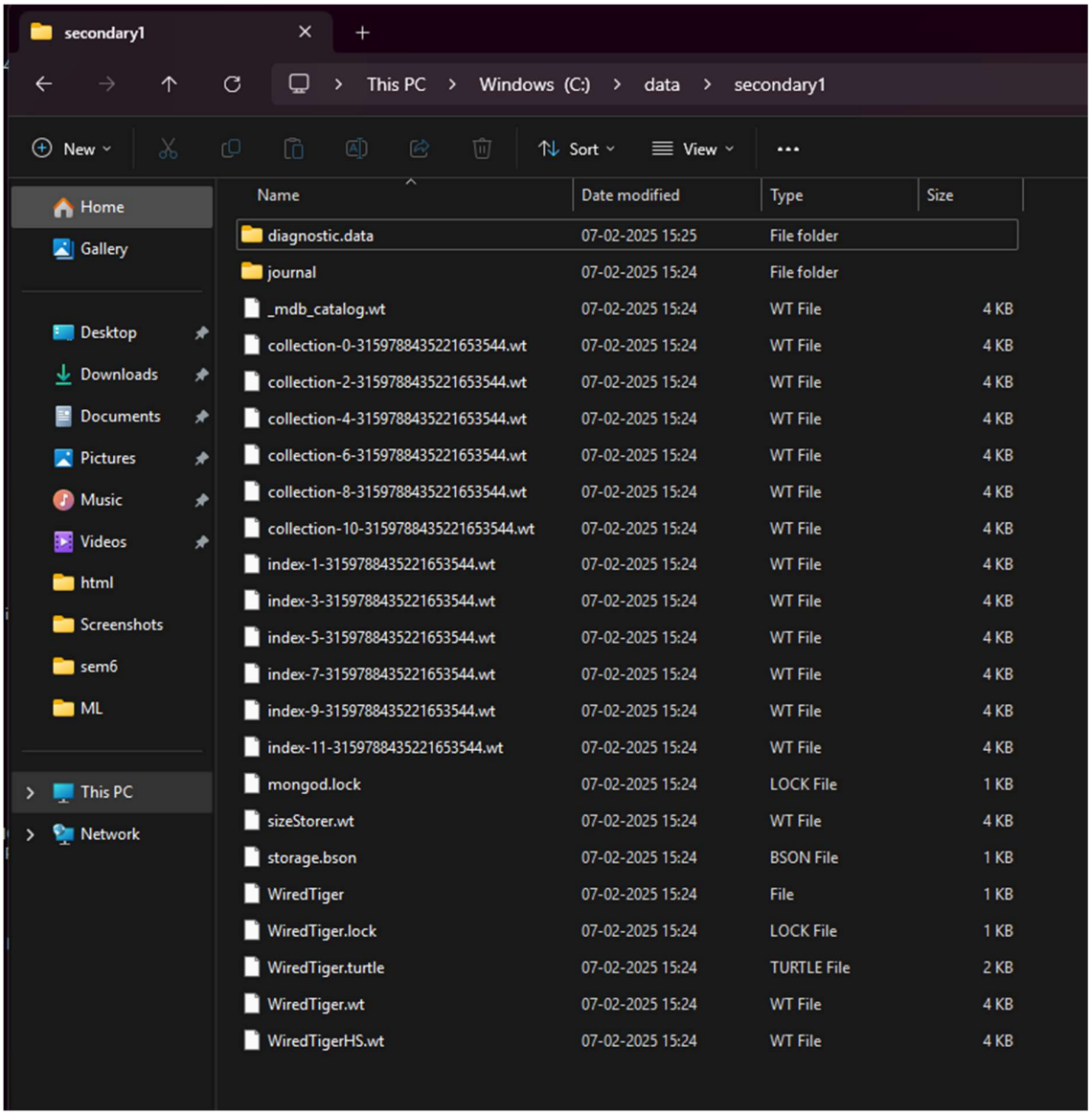
For secondary1

```
mongod --port=3737 --dbpath="C:\data\secondary1" --replSet="text-replica-set"
```

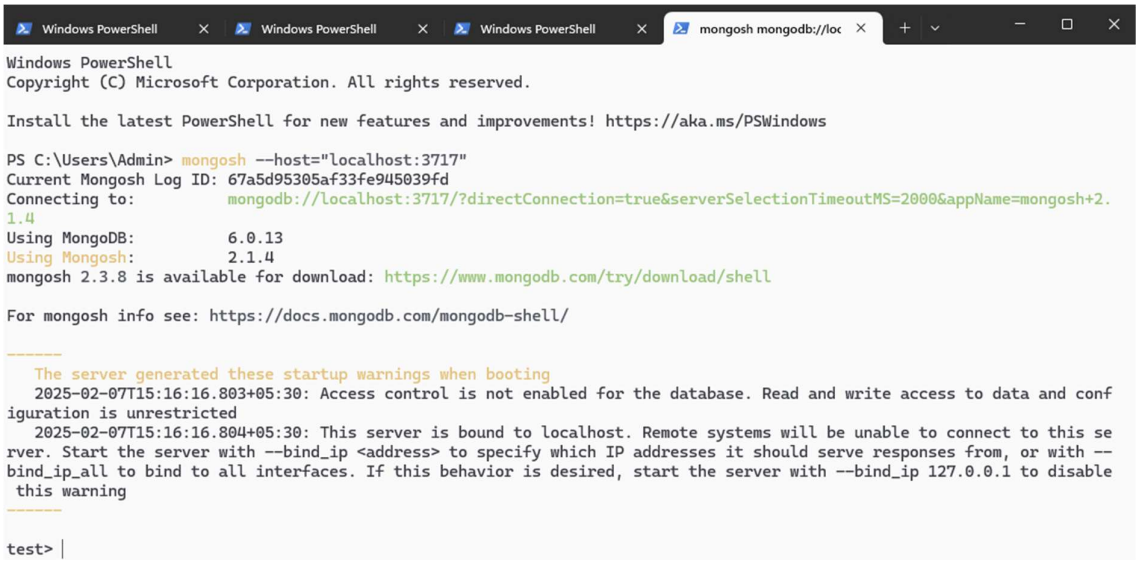
Firdaus Mulla | L007



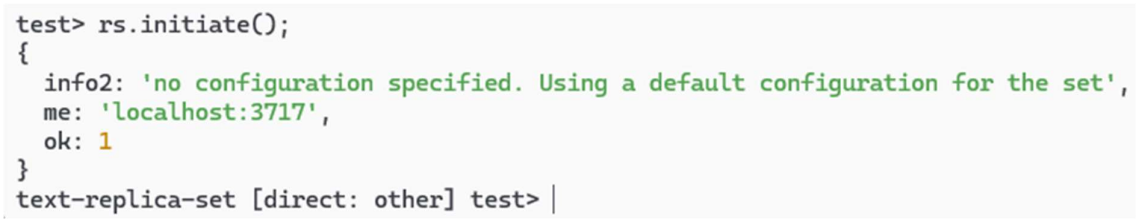
Firdaus Mulla | L007



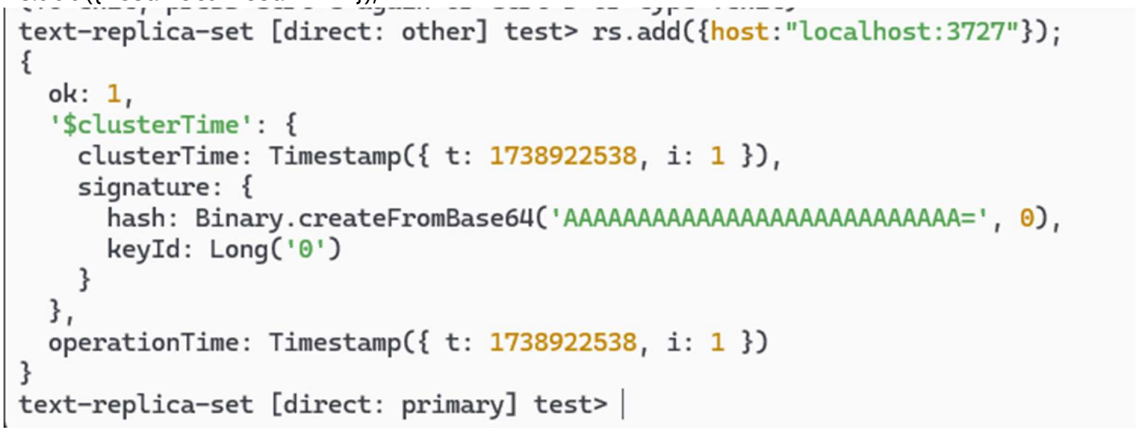
Open a new powershell window and perform the following command
mongosh --host="localhost:3717"



a test prompt will be open, perform the following commands in the test prompt.
rs.initiate();



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



rs.add({host:"localhost:3737"});

```
text-replica-set [direct: primary] test> rs.add({host:"localhost:3737"});
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1738923228, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1738923228, i: 1 })
}
```

Check the status : rs.status()

```
text-replica-set [direct: primary] test> rs.status();
{
  set: 'text-replica-set',
  date: ISODate('2025-02-07T10:04:33.470Z'),
  myState: 1,
  term: Long('1'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 2,
  writableVotingMembersCount: 2,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1738922665, i: 1 }), t: Long('1') },
    lastCommittedWallTime: ISODate('2025-02-07T10:04:25.465Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1738922665, i: 1 }), t: Long('1') },
    appliedOpTime: { ts: Timestamp({ t: 1738922665, i: 1 }), t: Long('1') },
    durableOpTime: { ts: Timestamp({ t: 1738922665, i: 1 }), t: Long('1') },
    lastAppliedWallTime: ISODate('2025-02-07T10:04:25.465Z'),
    lastDurableWallTime: ISODate('2025-02-07T10:04:25.465Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1738922615, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2025-02-07T10:00:45.413Z'),
    electionTerm: Long('1'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1738922445, i: 1 }), t: Long('-1') },
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1738922445, i: 1 }), t: Long('-1') },
    numVotesNeeded: 1,
    priorityAtElection: 1,
    electionTimeoutMillis: Long('10000'),
    newTermStartDate: ISODate('2025-02-07T10:00:45.436Z'),
    wMajorityWriteAvailabilityDate: ISODate('2025-02-07T10:00:45.450Z')
  },
  members: [
    {
      _id: 0,
      name: 'localhost:3717',
      health: 1,
      state: 1,
      stateStr: 'PRIMARY',
      uptime: 1098,
      optime: { ts: Timestamp({ t: 1738922665, i: 1 }), t: Long('1') },
      optimeDate: ISODate('2025-02-07T10:04:25.000Z'),
      lastAppliedWallTime: ISODate('2025-02-07T10:04:25.465Z'),
      lastDurableWallTime: ISODate('2025-02-07T10:04:25.465Z'),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
```

Firdaus Mulla | L007

```
      lastAppliedWallTime: ISODate('2025-02-07T10:04:25.465Z'),
      lastDurableWallTime: ISODate('2025-02-07T10:04:25.465Z'),
      syncSourceHost: '',
      syncSourceId: -1,
      infoMessage: '',
      electionTime: Timestamp({ t: 1738922445, i: 2 }),
      electionDate: ISODate('2025-02-07T10:00:45.000Z'),
      configVersion: 3,
      configTerm: 1,
      self: true,
      lastHeartbeatMessage: ''
    },
    {
      _id: 1,
      name: 'localhost:3727',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 135,
      optime: { ts: Timestamp({ t: 1738922665, i: 1 }), t: Long('1') },
      optimeDurable: { ts: Timestamp({ t: 1738922665, i: 1 }), t: Long('1')
      optimeDate: ISODate('2025-02-07T10:04:25.000Z'),
      optimeDurableDate: ISODate('2025-02-07T10:04:25.000Z'),
      lastAppliedWallTime: ISODate('2025-02-07T10:04:25.465Z'),
      lastDurableWallTime: ISODate('2025-02-07T10:04:25.465Z'),
      lastHeartbeat: ISODate('2025-02-07T10:04:32.440Z'),
      lastHeartbeatRecv: ISODate('2025-02-07T10:04:32.942Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: 'localhost:3717',
      syncSourceId: 0,
      infoMessage: '',
      configVersion: 3,
      configTerm: 1
    }
  ],
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1738922665, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1738922665, i: 1 })
}
```

```
    {
      _id: 2,
      name: 'localhost:3737',
      health: 1,
      state: 2,
      stateStr: 'SECONDARY',
      uptime: 73,
      optime: { ts: Timestamp({ t: 1738923291, i: 2 }), t: Long('1') },
      optimeDurable: { ts: Timestamp({ t: 1738923291, i: 2 }), t: Long('1') },
      optimeDate: ISODate('2025-02-07T10:14:51.000Z'),
      optimeDurableDate: ISODate('2025-02-07T10:14:51.000Z'),
      lastAppliedWallTime: ISODate('2025-02-07T10:14:51.495Z'),
      lastDurableWallTime: ISODate('2025-02-07T10:14:51.495Z'),
      lastHeartbeat: ISODate('2025-02-07T10:15:00.990Z'),
      lastHeartbeatRecv: ISODate('2025-02-07T10:15:01.508Z'),
      pingMs: Long('0'),
      lastHeartbeatMessage: '',
      syncSourceHost: 'localhost:3727',
      syncSourceId: 1,
      infoMessage: '',
      configVersion: 5,
      configTerm: 1
    }
  ],
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1738923291, i: 2 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1738923291, i: 2 })
}
```

For secondary open a new powershell window and perform the following command
mongosh --host="localhost:3727"

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Admin> mongosh --host="localhost:3727"
Current Mongosh Log ID: 67a5db983c06264de4ae005f
Connecting to:          mongodb://localhost:3727/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.1.4
Using MongoDB:          6.0.13
Using Mongosh:          2.1.4
mongosh 2.3.9 is available for download: https://www.mongodb.com/try/download/shell

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

------
   The server generated these startup warnings when booting
   2025-02-07T15:22:27.150+05:30: Access control is not enabled for the database. Read and write access to data and configurati
   2025-02-07T15:22:27.151+05:30: This server is bound to localhost. Remote systems will be unable to connect to this server. S
erve responses from, or with --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind
------

text-replica-set [direct: secondary] test>
```

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://

PS C:\Users\Admin>  mongosh --host="localhost:3737"
Current Mongosh Log ID: 67a5ddaa14c6a29ddb91d656
Connecting to:          mongodb://localhost:3737/?directConnection=true&s
Using MongoDB:          6.0.13
Using Mongosh:          2.1.4
mongosh 2.3.9 is available for download: https://www.mongodb.com/try/down

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

------
   The server generated these startup warnings when booting
   2025-02-07T15:24:51.181+05:30: Access control is not enabled for the d
   2025-02-07T15:24:51.182+05:30: This server is bound to localhost. Remo
erve responses from, or with --bind_ip_all to bind to all interfaces. If
------

text-replica-set [direct: secondary] test>
```

Go to primary powershell window and perform the dbms commands

```
text-replica-set [direct: primary] test> show dbs;
admin    80.00 KiB
config  176.00 KiB
local   436.00 KiB
```

Secondary

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

Secondary1

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

Now go to primary powershell window and perform the crud operations

use pracical6;

```
text-replica-set [direct: primary] test> use pracical6;
switched to db pracical6
```

db.users.insert({name:"Firdaus"});

```
text-replica-set [direct: primary] pracical6> db.users.insert({name:"Firdaus"});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('67a5df9d05af33fe945039fe') }
}
```

text-replica-set [direct: primary] pracical6> db.users.insertMany([{name:"Nida"},

{name:"John"}]);

```
text-replica-set [direct: primary] pracical6> db.users.insertMany([{name:"Nida"}, {name:"John"}]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67a5e16405af33fe945039ff'),
    '1': ObjectId('67a5e16405af33fe94503a00')
  }
}
```

Go to secondary shell and perform the db.users.find()

```
text-replica-set [direct: secondary] pracical6> db.users.find();
MongoServerError[NotPrimaryNoSecondaryOk]: not primary - consider using db.getMongo().setReadPref() or readPreference in the connection string
text-replica-set [direct: secondary] pracical6> db.getMongo();
mongodb://localhost:37277?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.1.4
```

If get the above error than perform the following command
db.getMongo().setReadPref("secondaryPreferred") and
again use db.users.find();

```
text-replica-set [direct: secondary] pracical6> db.getMongo().setReadPref("secondaryPreferred")

text-replica-set [direct: secondary] pracical6> db.users.find();
[
  { _id: ObjectId('67a5df9d05af33fe945039fe'), name: 'Firdaus' },
  { _id: ObjectId('67a5e16405af33fe945039ff'), name: 'Nida' },
  { _id: ObjectId('67a5e16405af33fe94503a00'), name: 'John' }
]
```

In secondary1

```
text-replica-set [direct: secondary] pracical6> db.getMongo().setReadPref("secondaryPreferred")

text-replica-set [direct: secondary] pracical6> db.users.find();
[
  { _id: ObjectId('67a5df9d05af33fe945039fe'), name: 'Firdaus' },
  { _id: ObjectId('67a5e16405af33fe945039ff'), name: 'Nida' },
  { _id: ObjectId('67a5e16405af33fe94503a00'), name: 'John' }
]
```

Update

In primary:

```
text-replica-set [direct: primary] pracical6> db.users.updateOne({name:"John"}, {$set:{name:"Sushmita"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
text-replica-set [direct: primary] pracical6> db.users.find();
[
  { _id: ObjectId('67a5df9d05af33fe945039fe'), name: 'Firdaus' },
  { _id: ObjectId('67a5e16405af33fe945039ff'), name: 'Nida' },
  { _id: ObjectId('67a5e16405af33fe94503a00'), name: 'Sushmita' }
]
text-replica-set [direct: primary] pracical6>
```

In secondary:

```
text-replica-set [direct: secondary] pracical6> db.users.find();
[
  { _id: ObjectId('67a5df9d05af33fe945039fe'), name: 'Firdaus' },
  { _id: ObjectId('67a5e16405af33fe945039ff'), name: 'Nida' },
  { _id: ObjectId('67a5e16405af33fe94503a00'), name: 'Sushmita' }
]
```

In secondary1:

```
text-replica-set [direct: secondary] pracical6> db.users.find();
[
  { _id: ObjectId('67a5df9d05af33fe945039fe'), name: 'Firdaus' },
  { _id: ObjectId('67a5e16405af33fe945039ff'), name: 'Nida' },
  { _id: ObjectId('67a5e16405af33fe94503a00'), name: 'Sushmita' }
]
```

Delete

In primary:

db.users.deleteOne({name:"Sushmita"});

```
text-replica-set [direct: primary] pracical6> db.users.deleteOne({name:"Sushmita"});
{ acknowledged: true, deletedCount: 1 }
text-replica-set [direct: primary] pracical6> db.users.find();
[
  { _id: ObjectId('67a5df9d05af33fe945039fe'), name: 'Firdaus' },
  { _id: ObjectId('67a5e16405af33fe945039ff'), name: 'Nida' }
]
text-replica-set [direct: primary] pracical6>
```

In secondary

```
text-replica-set [direct: secondary] pracical6> db.users.find()
[
  { _id: ObjectId('67a5df9d05af33fe945039fe'), name: 'Firdaus' },
  { _id: ObjectId('67a5e16405af33fe945039ff'), name: 'Nida' }
]
```

In secondary1:

```
text-replica-set [direct: secondary] pracical6> db.users.find();
[
  { _id: ObjectId('67a5df9d05af33fe945039fe'), name: 'Firdaus' },
  { _id: ObjectId('67a5e16405af33fe945039ff'), name: 'Nida' }
]
```