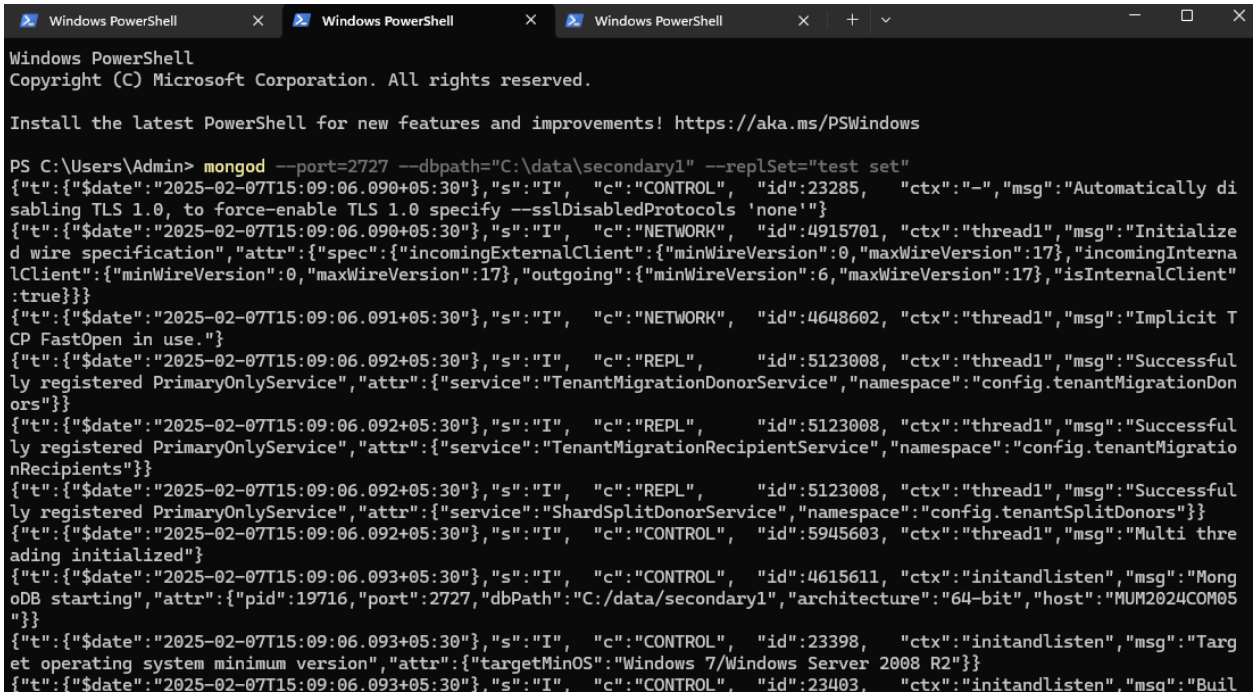
PRACTICAL 6
7-02-2025
L030
MONGODB REPLICATION

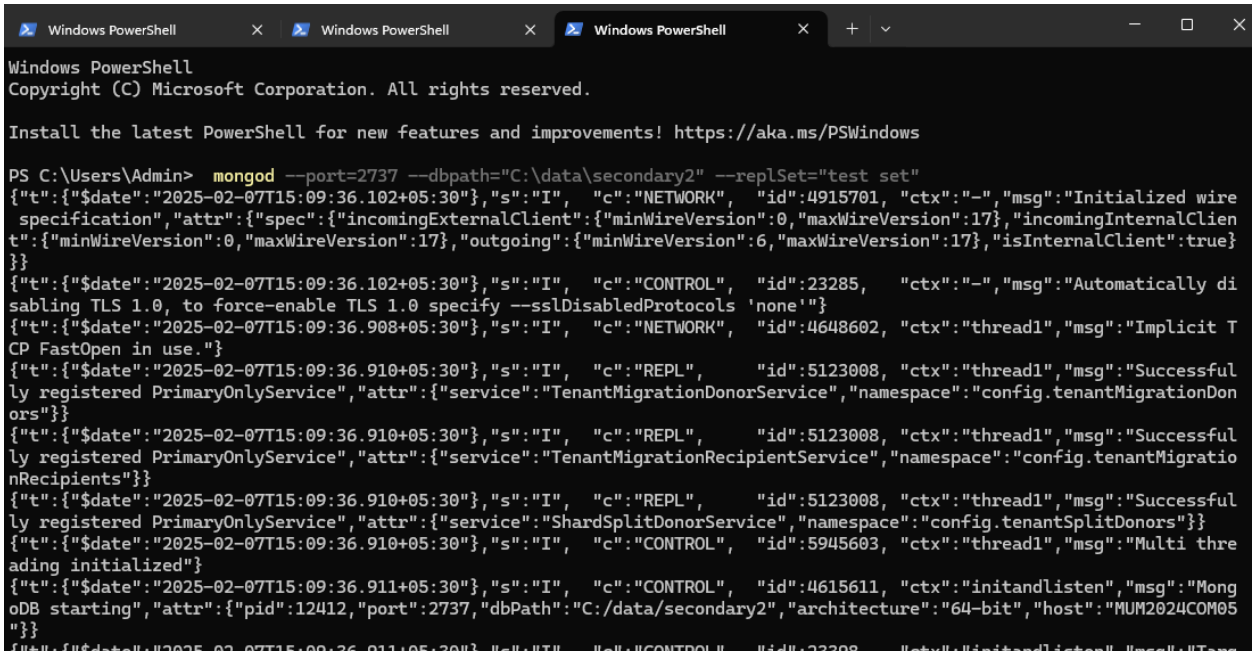
STEPS:

1. Create a folder named “data” in C drive
2. Inside this folder make 3 more folders named “primary” ,”Secondary1”,”Secondary2”.
3. Open Powershell
4. Type the following commands.
   mongod --port=2717 --dbpath ="C:\data\primary" --replSet ="test-set"

5. Open another window of Powershell
6. Type the following command
   mongod --port=2717 --dbpath ="C:\data\secondary1" --replSet ="test-set"



6. Repeat step 5 6 again and type the following command
mongod --port=2717 --dbpath ="C:\data\secondary1" --replSet ="test-set"



7. After this open another window od powershell and type the command
mongosh --host="localhost:2717"

Then type rs.initiate() to to start the connection

```
PS C:\Users\Admin> mongosh --host="localhost:2717"
Current Mongosh Log ID: 67a5dc9255b2bfee355d61e9
Connecting to:          mongodb://localhost:2717/?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
   2025-02-07T15:40:30.447+05:30: Access control is not enabled for the database. Read and write access to data and conf
iguration is unrestricted
   2025-02-07T15:40:30.447+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
------

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

9. Add the other 2 secondary ports by using the commands
 rs.add({host:"localhost:2727"})
 rs.add({host:"localhost:2737"})

```
test-set [direct: other] test> rs.add({host:"localhost:2727"})
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1738923219, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1738923219, i: 1 })
}
test-set [direct: primary] test> rs.add({host:"localhost:2737"})
{
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1738923225, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1738923225, i: 1 })
}
test-set [direct: primary] test>
```

10.rs.status()

```
test-replica-set [direct: primary] test> rs.status()
{
  set: 'test-replica-set',
  date: ISODate('2025-02-07T09:55:29.709Z'),
  myState: 1,
  term: Long('1'),
  syncSourceHost: '',
  syncSourceId: -1,
  heartbeatIntervalMillis: Long('2000'),
  majorityVoteCount: 2,
  writeMajorityCount: 2,
  votingMembersCount: 3,
  writableVotingMembersCount: 3,
  optimes: {
    lastCommittedOpTime: { ts: Timestamp({ t: 1738922125, i: 1 }), t: Long('1') },
    lastCommittedWallTime: ISODate('2025-02-07T09:55:25.338Z'),
    readConcernMajorityOpTime: { ts: Timestamp({ t: 1738922125, i: 1 }), t: Long('1')
    appliedOpTime: { ts: Timestamp({ t: 1738922125, i: 1 }), t: Long('1') },
    durableOpTime: { ts: Timestamp({ t: 1738922125, i: 1 }), t: Long('1') },
    lastAppliedWallTime: ISODate('2025-02-07T09:55:25.338Z'),
    lastDurableWallTime: ISODate('2025-02-07T09:55:25.338Z')
  },
  lastStableRecoveryTimestamp: Timestamp({ t: 1738922105, i: 1 }),
  electionCandidateMetrics: {
    lastElectionReason: 'electionTimeout',
    lastElectionDate: ISODate('2025-02-07T09:46:14.955Z'),
    electionTerm: Long('1'),
    lastCommittedOpTimeAtElection: { ts: Timestamp({ t: 1738921574, i: 1 }), t: Long('
    lastSeenOpTimeAtElection: { ts: Timestamp({ t: 1738921574, i: 1 }), t: Long('-1')
    numVotesNeeded: 1,
    priorityAtElection: 1,
    electionTimeoutMillis: Long('10000'),
    newTermStartDate: ISODate('2025-02-07T09:46:14.975Z'),
    wMajorityWriteAvailabilityDate: ISODate('2025-02-07T09:46:14.988Z')
  },
  members: [
    {
      _id: 0,
      name: 'localhost:2717',
      health: 1,
```

12. **Create a new instance**

mongosh --host="localhost:2727"

```
PS C:\Users\Admin> mongosh --host="localhost:2727"
Current Mongosh Log ID: 67a5d908de29b1414d8bf201
Connecting to:          mongodb://localhost:2727/?directConnection=true&serverSelectionTimeoutMS=2000&
appName=mongosh+2.2.0
Using MongoDB:          7.0.6
Using Mongosh:          2.2.0
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
   2025-02-07T15:09:09.713+05:30: Access control is not enabled for the database. Read and write acces
s to data and configuration is unrestricted
   2025-02-07T15:09:09.714+05:30: This server is bound to localhost. Remote systems will be unable to
connect to this server. Start the server with --bind_ip <address> to specify which IP addresses it sho
uld 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
------
```

13. Create a new instance

mongosh --host="localhost:2737"

```
PS C:\Users\Admin>  mongosh --host="localhost:2737"
Current Mongosh Log ID: 67a5d95b0b96eab9ed8bf201
Connecting to:          mongodb://localhost:2737/?directConnection=true&serverSelectionTimeoutMS=2000&
appName=mongosh+2.2.0
Using MongoDB:          7.0.6
Using Mongosh:          2.2.0
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
   2025-02-07T15:11:13.734+05:30: Access control is not enabled for the database. Read and write acces
s to data and configuration is unrestricted
   2025-02-07T15:11:13.734+05:30: This server is bound to localhost. Remote systems will be unable to
connect to this server. Start the server with --bind_ip <address> to specify which IP addresses it sho
uld 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
```

14. Open the instance with the primary server

Run the cmd

show dbs

```
test-replica-set [direct: primary] test> show dbs
admin    80.00 KiB
config  212.00 KiB
local   404.00 KiB
test-replica-set [direct: primary] test>
```

The same databases should be reflecting in our secondary servers

Secondary1:

```
test-replica-set [direct: secondary] test> show dbs
admin    80.00 KiB
config  276.00 KiB
local   404.00 KiB
test-replica-set [direct: secondary] test>
```

Secondary2:

```
test-replica-set [direct: secondary] test> show dbs
admin    80.00 KiB
config  276.00 KiB
local   404.00 KiB
test-replica-set [direct: secondary] test>
```

15. Open the primary server instance then create database

use practical

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

16. Insert some records

db.users.insertMany([{ name: "Huda", roll_no: 17, age: 21 }, { name: "Shifa", roll_no: 18, age: 22 }, { name: "Tasmiya", roll_no: 21, age: 23, address: "Mumbai" }, { name: "Samiya", roll_no: 12, address: "Delhi" }]);

```
test-replica-set [direct: primary] practical> db.users.insertMany([ { name: "Huda", roll_no: 17, age:
21 }, { name: "Shifa", roll_no: 18, age: 22 }, { name: "Tasmiya", roll_no: 21, age: 23, address: "Mumb
ai" }, { name: "Samiya", roll_no: 12, address: "Delhi" } ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67a5dbee5d78a0ea418bf202'),
    '1': ObjectId('67a5dbee5d78a0ea418bf203'),
    '2': ObjectId('67a5dbee5d78a0ea418bf204'),
    '3': ObjectId('67a5dbee5d78a0ea418bf205')
  }
}
```

17. Reading the entries

db.users.find()

```
test-replica-set [direct: primary] practical> db.users.find()
[
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf202'),
    name: 'Huda',
    roll_no: 17,
    age: 21
  },
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf203'),
    name: 'Shifa',
    roll_no: 18,
    age: 22
  },
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf204'),
    name: 'Tasmiya',
    roll_no: 21,
    age: 23,
    address: 'Mumbai'
  },
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf205'),
    name: 'Samiya',
    roll_no: 12,
    address: 'Delhi'
  }
```

18. Now switch to your secondary server

use practical

Now try to read the data

db.users.find()

First Secondary server:

```
test-replica-set [direct: secondary] test> use practical
switched to db practical
```

```
test-replica-set [direct: secondary] practical> db.users.find()
[
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf204'),
    name: 'Tasmiya',
    roll_no: 21,
    age: 23,
    address: 'Mumbai'
  },
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf203'),
    name: 'Shifa',
    roll_no: 18,
    age: 22
  },
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf202'),
    name: 'Huda',
    roll_no: 17,
    age: 21
  },
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf205'),
    name: 'Samiya',
    roll_no: 12,
    address: 'Delhi'
  }
]
```

Second Secondary server:

```
test-replica-set [direct: secondary] test> use practical
switched to db practical
```

```
test-replica-set [direct: secondary] practical> db.users.find()
[
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf204'),
    name: 'Tasmiya',
    roll_no: 21,
    age: 23,
    address: 'Mumbai'
  },
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf203'),
    name: 'Shifa',
    roll_no: 18,
    age: 22
  },
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf202'),
    name: 'Huda',
    roll_no: 17,
    age: 21
  },
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf205'),
    name: 'Samiya',
    roll_no: 12,
    address: 'Delhi'
  }
]
```

19. Performing more CRUD Operations

a. Insert a single record

db.users.insertOne({ name: "Nina", roll_no: 19, age: 20 });

```
test-replica-set [direct: primary] practical> db.users.insertOne({ name: "Nina", roll_no: 19, age: 20
});
{
  acknowledged: true,
  insertedId: ObjectId('67a5de905d78a0ea418bf206')
}
```

b. Insert Multiple records

db.users.insertMany([{ name: "Sana", roll_no: 20, age: 22 },{ name: "Emad", roll_no: 21, age: 24 }]);

```
test-replica-set [direct: primary] practical> db.users.insertMany([{ name: "Sana", roll_no: 20, age: 2
2 },{ name: "Emad", roll_no: 21, age: 24 }]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67a5df055d78a0ea418bf207'),
    '1': ObjectId('67a5df055d78a0ea418bf208')
  }
}
```

c. Find record where name = “Huda”

db.users.find({ name: "Huda" }).pretty();

```
test-replica-set [direct: primary] practical> db.users.find({ name: "Huda" }).pretty();
[
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf202'),
    name: 'Huda',
    roll_no: 17,
    age: 21
  }
]
```

d. Update a document

db.users.updateOne({ name: "Shifa" }, { $set: { age: 23 } });

```
test-replica-set [direct: primary] practical> db.users.updateOne( { name: "Shifa" }, { $set: { age: 23
 } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

e. Update multiple documents

db.users.updateMany({ age: { $lt: 23 } }, { $set: { status: "young" } });

```
test-replica-set [direct: primary] practical> db.users.updateMany( { age: { $lt: 23 } }, { $set: { sta
status: "young" } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 3,
  modifiedCount: 3,
  upsertedCount: 0
}
```

All the changes we made are reflected in the secondary server

First secondary server

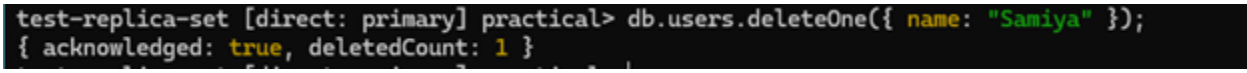
```
test-replica-set [direct: secondary] practical> db.users.find()
[
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf204'),
    name: 'Tasmiya',
    roll_no: 21,
    age: 23,
    address: 'Mumbai'
  },
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf203'),
    name: 'Shifa',
    roll_no: 18,
    age: 23
  },
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf202'),
    name: 'Huda',
    roll_no: 17,
    age: 21,
    status: 'young'
  },
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf205'),
    name: 'Samiya',
    roll_no: 12,
    address: 'Delhi'
  },
  {
    _id: ObjectId('67a5de905d78a0ea418bf206'),
    name: 'Nina',
    roll_no: 19,
    age: 20,
    status: 'young'
  },
  {
    _id: ObjectId('67a5df055d78a0ea418bf207'),
    name: 'Sana',
    roll_no: 20,
    age: 22,
    status: 'young'
  },
  {
    _id: ObjectId('67a5df055d78a0ea418bf208'),
    name: 'Emad',
    roll_no: 21,
    age: 24
  }
]
```

Second Secondary server

```
test-replica-set [direct: secondary] practical> db.users.find()
[
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf204'),
    name: 'Tasmiya',
    roll_no: 21,
    age: 23,
    address: 'Mumbai'
  },
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf203'),
    name: 'Shifa',
    roll_no: 18,
    age: 23
  },
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf202'),
    name: 'Huda',
    roll_no: 17,
    age: 21,
    status: 'young'
  },
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf205'),
    name: 'Samiya',
    roll_no: 12,
    address: 'Delhi'
  },
  {
    _id: ObjectId('67a5de905d78a0ea418bf206'),
    name: 'Nina',
    roll_no: 19,
    age: 20,
    status: 'young'
  },
  {
    _id: ObjectId('67a5df055d78a0ea418bf208'),
    name: 'Emad',
    roll_no: 21,
    age: 24
  },
  {
    _id: ObjectId('67a5df055d78a0ea418bf207'),
    name: 'Sana',
    roll_no: 20,
    age: 22,
    status: 'young'
  }
]
```

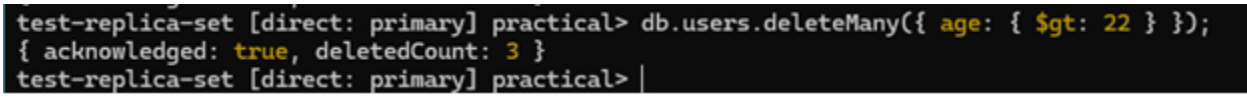
f. Delete a single record

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



g. Delete multiple records

db.users.deleteMany({ age: { $gt: 22 } });



Changes successfully reflected in the secondary servers

```
test-replica-set [direct: secondary] practical> db.users.find()
[
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf202'),
    name: 'Huda',
    roll_no: 17,
    age: 21,
    status: 'young'
  },
  {
    _id: ObjectId('67a5de905d78a0ea418bf206'),
    name: 'Nina',
    roll_no: 19,
    age: 20,
    status: 'young'
  },
  {
    _id: ObjectId('67a5df055d78a0ea418bf207'),
    name: 'Sana',
    roll_no: 20,
    age: 22,
    status: 'young'
  }
]
```

```
test-replica-set [direct: secondary] practical> db.users.find()
[
  {
    _id: ObjectId('67a5dbee5d78a0ea418bf202'),
    name: 'Huda',
    roll_no: 17,
    age: 21,
    status: 'young'
  },
  {
    _id: ObjectId('67a5de905d78a0ea418bf206'),
    name: 'Nina',
    roll_no: 19,
    age: 20,
    status: 'young'
  },
  {
    _id: ObjectId('67a5df055d78a0ea418bf207'),
    name: 'Sana',
    roll_no: 20,
    age: 22,
    status: 'young'
  }
]
```