MongoDB AdministratorTools_Lab

- mongosh
- mongodump, mongorestore
- mongotop, mongostat
- Compass & Atlas Metrics
- Lab: Use admin tools on a dataset

CSC Scenario (Use Case)

"CSC Compliance Platform"
Bangalore team manages the **primary MongoDB cluster** (Atlas or on-prem).
Mumbai team needs:

- **Backups** before a schema change
- A **restored copy** for testing
- Basic **health monitoring** during peak filing season

We'll simulate:

1. Admin in **Bangalore** checks the cluster & dataset using mongosh.
2. They take a **backup** with mongodump.
3. They **restore** the dump into a new database (for Mumbai testing).
4. During a simulated load, they monitor with mongotop & mongostat.
5. They also check **Compass & Atlas Metrics** for performance graphs.

You can run this lab on:

- A **local mongod** (simpler for class), or
- An **Atlas cluster** (tools connect via connection string).

I'll write commands assuming **local** for simplicity. For Atlas, you mostly just replace --host/--uri.

---

0. Prerequisites

Before lab:

1. **Install tools** on each participant machine:
    - MongoDB Database Tools (includes mongodump, mongorestore, mongotop, mongostat)
    - mongosh
    - MongoDB Compass
2. Have one database running (local or Atlas), with:
    - DB: csc_compliance

    - Collections: entities, filings
3. Optionally pre-load 1–5K sample documents so tools show meaningful stats.

Part 1 – mongosh: “Talking to the Cluster”

🎯 **Use Case**

Bangalore DBA wants to:

- Confirm MongoDB is up.
- See which databases exist.
- Check stats for csc_compliance.
- Verify indexes for performance.

🔷 **Step 1.1 – Connect with mongosh**

Local:

```
mongosh "mongodb://localhost:27017"
```

Atlas (example):

```
mongosh "mongodb+srv://csc_app_user@csc-compliance-m0.xxxxxx.mongodb.net"
```

💬 Explain: This is the **admin shell** – equivalent of “SSH into your database brain.”

🔷 **Step 1.2 – Switch to CSC DB & Inspect**

Inside mongosh:

```
show dbs          // see all databases
use csc_compliance // switch to our CSC database
show collections   // should show entities, filings
```

Check stats for filings:

```
db.filings.stats()
```

Explain key fields:

- count – number of docs (filings)
- size – data size
- storageSize – allocated on disk
- nindexes – number of indexes

📌 CSC story: Bangalore DBA verifies that filings has the expected volume before executing backups or schema changes.

🔷 Step 1.3 – Check Indexes and Create One (Quick Admin Action)

db.filings.getIndexes()

If you don't see an index on state:

db.filings.createIndex({ state: 1, status: 1, due_date: 1 })
🧠 Use Case: Admin adds an index to speed up compliance dashboards showing **open filings per state** during peak season.

2️⃣ Part 2 – mongodump & mongorestore : Backup & Restore

🎯 Use Case

Before Bangalore team deploys a change to filings document structure, they must:

1. Take a **backup** of csc_compliance.
2. Restore it into a **test DB** (csc_compliance_mumbai) for the Mumbai team to validate changes.

🔷 Step 2.1 – Take a Backup with mongodump

From terminal (not inside mongosh):

```
mongodump \
  --db=csc_compliance \
  --out=./csc_backup_2025_01_01
```

This creates a folder:

```
./csc_backup_2025_01_01/csc_compliance/
    entities.bson
    filings.bson
    ... metadata.json
```
💬 Explain:

- This is a **logical backup** – BSON files per collection.
- You can zip and send to Mumbai or store in S3/backup system.

Step 2.2 – Restore into a New Database (csc_compliance_mumbai)

Simulate Mumbai's test environment using mongorestore:

```
mongorestore \
  --nsFrom='csc_compliance.*' \
  --nsTo='csc_compliance_mumbai.*' \
  ./csc_backup_2025_01_01
```

This copies all collections from DB csc_compliance → csc_compliance_mumbai.

Verify in mongosh:

```
show dbs
use csc_compliance_mumbai
show collections
db.filings.countDocuments()
```

Use Case Explanation:

- Bangalore created a backup.
- Mumbai QA team now has an **isolated copy** to test new features, indexes, or schema changes without impacting production.

3️⃣ Part 3 – mongotop & mongostat : Live Monitoring

Now we pretend peak load is happening (many filings being inserted/queried).

3.1 mongotop: Collection-Level Read/Write Time

🎯 Use Case

During US year-end filings, Bangalore DBA wants to see **which collections are busiest** – filings vs entities.

🔷 Step 3.1.1 – Run mongotop

From a new terminal:

mongotop 3

(3 = refresh every 3 seconds.)

Output example:

```
ns                          total    read    write
csc_compliance.filings        102ms    70ms    32ms
csc_compliance.entities        10ms    10ms    0ms
admin.system.version           0ms     0ms     0ms
...
```

Explain columns:

- ns – namespace (db.collection)
- total – total time spent in read + write since last sample
- read – time spent reading
- write – time spent writing

💬 CSC Story:
Admin sees csc_compliance.filings dominating I/O during peak – correct, as filings are the hot table. If something unexpected (e.g., system.profile) shows high usage, that's a red flag.

3.2 mongostat: Server-Level Health Snapshot

🎯 Use Case

Mumbai support team gets a complaint: **"The compliance dashboard is slow."**
They use mongostat to see server health: connections, ops/sec, memory, etc.

🔷 Step 3.2.1 – Run mongostat

mongostat 3

Sample output:

```
insert query update delete getmore command dirty  used flushes vsize   res   qr|qw
ar|aw  netIn netOut  conn   time
   10    30      5      2       0     50     0%   20%       0  1.5G  300M   0|0    5|0   10kB
20kB    30  12:30:01
...
```

Key columns to explain:

- insert/query/update/delete – operations per second
- command – includes things like isMaster, auth, etc.
- conn – number of connected clients
- qr|qw – queued reads/writes
- netIn/netOut – network flow

💬 CSC Story:

- If conn suddenly spikes or qr|qw grows large, DB is under pressure.
- Admins might decide to scale up Atlas tier, add indexes, or investigate noisy clients.

4️⃣ Part 4 – Compass & Atlas Metrics

We've used CLI; now let's use the UI tools.

4.1 MongoDB Compass – Local Insight

🎯 Use Case

DBA quickly wants to **inspect stats**, **indexes**, and **estimated document counts** without remembering all shell commands.

🔷 Step 4.1.1 – Connect from Compass

1. Open **MongoDB Compass**.
2. Use your connection string (local or Atlas).
3. Click **Connect**.

🔷 Step 4.1.2 – View Collection Stats

1. Select DB csc_compliance.
2. Click filings collection.
3. Navigate to **Indexes** tab – see all indexes.
4. Navigate to **Schema** / **Documents** – visually inspect data.

💬 Explanation:
Compass is a GUI wrapper around many admin operations: profiling queries, checking sample docs, building filters, and even showing index usage suggestions in some versions.

4.2 Atlas Metrics – Cloud Monitoring Panel

(Relevant if your CSC cluster is on Atlas)

🎯 Use Case

Bangalore SRE team needs **historical performance**: CPU, Memory, Connections, Opcounters, and response times over last 6–24 hours.

🔷 Step 4.2.1 – Open Metrics in Atlas

1. Log in to **Atlas**.
2. Go to your project: csc-compliance-sandbox.
3. Click the cluster (e.g., csc-compliance-m0 or csc-compliance-prod).
4. Click **Metrics** tab.

You'll see charts like:

- **Connections** – spikes when many app servers / devs connect
- **Operation Execution Time**, **Opcounters** (insert/query/update, etc.)
- **CPU**, **Memory**, **Disk I/O** (for larger tiers)

🔷 Step 4.2.2 – Correlate with Your Lab Activity

While running:

- mongostat, mongotop
- Heavy shell script inserting filings
- Application test traffic

…you will see:

- Opcounters go up (insert/query).
- Connections grow if many clients connect.
- Latency charts reflect load.

💬 CSC Story:

- Bangalore SRE checks Atlas Metrics to see if a Bangalore promotion run or Mumbai UAT run caused performance issues.
- Historical charts help answer: "Was the DB slow yesterday around 4 PM IST?"

5️⃣ Final Lab: End-to-End Checklist for Participants

Here's a **compact lab flow** you can give to CSC Bangalore & Mumbai teams.

Step 1 – Connect & Inspect (mongosh)

- Connect to the DB using mongosh.
- show dbs, use csc_compliance, show collections.
- Run db.filings.stats() and interpret document count & indexes.

Step 2 – Backup (mongodump)

- Run mongodump --db=csc_compliance --out=./csc_backup_<date>.
- Verify the backup folder & BSON files.

Step 3 – Restore (mongorestore)

- Restore into csc_compliance_mumbai using mongorestore --nsFrom='csc_compliance.*' --nsTo='csc_compliance_mumbai.*' ./csc_backup_<date>.
- Confirm in mongosh that new DB exists and has correct counts.

Step 4 – Monitor Load (mongotop & mongostat)

- In one terminal, run mongotop 3.
- In another, run mongostat 3.
- In a third, run a script or loop that inserts/queries filings.
- Observe which collections are hot and how ops/sec & connections behave.

Step 5 – Compass & (Optionally) Atlas Metrics

- Connect from MongoDB Compass, browse csc_compliance and csc_compliance_mumbai.
- Check indexes, counts, schema.
- If using Atlas, open cluster **Metrics** and correlate graphs with your test load.