MongoDB Atlas Cluster_Lab

Use Case:

"Set up a secure *free* MongoDB Atlas cluster to store **CSC client entities & filings data**, and connect to it from developer laptops in Bangalore & Mumbai."

---

1. Create a Free Tier MongoDB Atlas Cluster (M0)

🎯 Lab Objective (Explain to CSC Participants)

"We're going to create a free MongoDB Atlas cluster in the cloud where we'll later store CSC client entities, filings, and reminders. This is our sandbox, not production."

Step 1.1 – Sign up / Sign in

1. Open browser: **https://www.mongodb.com/atlas**
2. Click **Sign In** or **Try Free**.
3. Sign in using:
   - Work email (preferred), or
   - Google / SSO (if allowed by CSC IT policies).
4. Once logged in, you'll land on the **Atlas UI** (Project / Organization view).
5.

💡 *Explain:* Atlas = MongoDB's fully managed cloud. No need to install MongoDB server, just use the cluster.

Step 1.2 – Create a New Project for CSC

1. On the top-left, click **"Projects"** → **"New Project"**.
2. Name it:

   csc-compliance-sandbox

3. Add teammates (optional) – e.g. other CSC devs from Bangalore / Mumbai.
4. Click **Create Project** → then **Build a Database**.

💡 *Explain:* A **Project** groups clusters, similar to a "folder" for one application, e.g. *CSC Compliance Portal.*

Step 1.3 – Choose Free Tier Cluster

1. On "Create a Database" page, pick:
   - **Deployment**: *Dedicated / Serverless / Shared*

    - Choose **Shared** → this includes **Free M0**.
2. Select **M0 Free Tier**.
3. Choose Cloud Provider & Region:
    - Example: AWS and a region near India (e.g., ap-south-1 Mumbai) if available in the list.
4. Keep default cluster name or rename to:

   Cluster0 → rename to csc-compliance-m0.

5. Click **Create Deployment** (or **Create Cluster** depending on UI).

💡 *Explain:*

- **M0** = free shared cluster, perfect for training and POCs.
- Region choice matters for **latency –** choose a region close to Bangalore/Mumbai users.

Atlas will now start provisioning your cluster (1–5 minutes in background).

---

2. Cluster Deployment Options (Teach While It Builds)

While the cluster is being created, explain deployment options in simple words:

2.1 Shared (M0/M2/M5)

- Multi-tenant; small workloads
- Good for **training, dev, prototypes**
- **Free (M0)** – limited storage & performance, but enough for learning.

2.2 Dedicated (M10+)

- Dedicated virtual machines
- Better performance & SLA
- For **production** CSC applications with predictable load.

2.3 Serverless

- You are billed based on **read/write/compute usage**, not fixed size.
- Great for **spiky** or low-volume workloads.

🧠 For CSC Bangalore/Mumbai devs:

- Use **M0/M10** for development / lower envs.
- **Dedicated clusters (M30+)** for real compliance workloads across clients.

3. Network & Security Configuration

Now secure access so CSC devs (Bangalore & Mumbai) can connect safely.

3.1 Create Database User (NOT your Atlas login)

1. Go to **Database** → click your cluster csc-compliance-m0.
2. Click **"Connect"** → you'll see a 3-step wizard.
3. Under **"Choose a connection method"**, pick something like **"MongoDB Shell"** or **"Compass"** (we will still create user here).
4. In **"Create a Database User"** section:
   - Username: csc_app_user
   - Password: generate a strong one or click **Autogenerate Secure Password**.
5. Copy/save the password somewhere safe (for the lab).
6. Give appropriate roles:
   - For lab: **Read and write to any database** (built-in role) is OK.
   - For real CSC app: restrict to specific DB.

💡 *Explain:*
This **database user** is what your applications and tools use to connect. It's separate from your MongoDB Atlas login.

3.2 Configure Network Access (IP Whitelisting)

We need to allow developer machines in **Bangalore & Mumbai**.

1. Still in the **Connect** wizard, find **"Add IP Address"**.
2. Options:
   - **My Current IP Address** → auto-detect (great in a classroom).
   - For remote teams, you can:
     - Add office IP ranges (from CSC IT)
     - Or 0.0.0.0/0 for quick learning (⚠️ **only for lab**, not production).
3. For lab simplicity:
   - Click **"Add My Current IP Address"** on Bangalore machines.
   - Mumbai users do the same from their location.
4. Click **Confirm** / **Save**.

⚠️ *Important message for CSC teams:*
For production, never leave 0.0.0.0/0. Use:

- Office static IPs
- VPN ranges
- VPC Peering / PrivateLink for secure connectivity.

4. Lab: Deploy and Connect to Atlas Cluster (CSC Hands-On)

Now we make it real: create CSC-style database & collections.

4.1 Connect Using MongoDB Compass (GUI)

Step 4.1.1 – Get Connection String

1. Click **Connect** → choose **“MongoDB Compass”**.
2. Copy the **connection string**, it will look like:

```
mongodb+srv://csc_app_user:<password>@csc-compliance-
m0.xxxxxx.mongodb.net/
```

3. Replace <password> with the actual password you created.

Step 4.1.2 – Open Compass and Connect

1. Start **MongoDB Compass** on your laptop.
2. In the **Connection String** input box, paste:

```
mongodb+srv://csc_app_user:<password>@csc-compliance-
m0.xxxxxx.mongodb.net/csc_compliance?retryWrites=true&w=majori
ty
```

3. Click **Connect**.

💡 *Explain:*

- mongodb+srv means Atlas connection using DNS SRV records.
- csc_compliance is the default database we’ll use for this lab.

If connection fails:

- Check IP whitelist
- Check username/password
- Ensure there’s no VPN blocking outbound 27017/443 depending on config.

4.2 Create CSC Collections and Insert Sample Data

Once connected in Compass:

Step 4.2.1 – Create Database & Collections

1. In Compass, click **“Create Database”**:
    - Database name: csc_compliance
    - Collection name: entities
2. Click **Create Database**.
3. Now click **“Create Collection”** inside csc_compliance:
    - filings
    - reminders

Step 4.2.2 – Insert Sample CSC Data (Entities)

In entities, click **Insert Document** and use:

```
{
  "_id": 1001,
  "name": "Acme Holdings Inc.",
  "clientId": "CLI-001",
  "jurisdiction": "DE",
  "status": "Active",
  "industry": "FinTech",
  "serviceTypes": ["registered_agent", "annual_report"],
  "city": "Bangalore",
  "country": "India"
}
```

Add another:

```
{
  "_id": 1002,
  "name": "Sunrise Logistics LLC",
  "clientId": "CLI-002",
  "jurisdiction": "CA",
  "status": "Active",
  "industry": "Logistics",
  "serviceTypes": ["registered_agent"],
  "city": "Mumbai",
  "country": "India"
}
```

💡 *Explain:*
These represent **CSC client entities** managed from Bangalore & Mumbai offices.

Step 4.2.3 – Insert Sample Filings

Switch to filings collection, **Insert Document**:

```
{
  "entity_id": 1001,
  "filing_type": "Annual Report",
  "state": "DE",
  "due_date": { "$date": "2025-03-01T00:00:00Z" },
  "filed_date": null,
  "status": "OPEN",
  "amount": 5000
}
```

Another:

```
{
  "entity_id": 1002,
  "filing_type": "Annual Report",
  "state": "CA",
  "due_date": { "$date": "2025-02-10T00:00:00Z" },
  "filed_date": { "$date": "2025-02-01T00:00:00Z" },
  "status": "FILED",
  "amount": 1500
}
```

4.3 Verify Queries from Compass (or mongosh)

Example 1 – All entities managed from Bangalore

In entities → **Filter**:

```
{ "city": "Bangalore" }
```

Example 2 – Open filings

In filings → **Filter**:

```
{ "status": "OPEN" }
```

💡 *Explain:*
Now the CSC team is working fully on **Atlas**, no local MongoDB server, but still doing the same operations.

4.4 Optional: Connect via mongosh

If you want devs to try the shell:

1. In Atlas **Connect** wizard, choose **“Connect with MongoDB Shell”**.
2. Copy the command Atlas gives, e.g.:

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

3. Run it in terminal, enter password when prompted.
4. Test:

```
show dbs
use csc_compliance
db.entities.find()
db.filings.find({ status: "OPEN" })
```

---

5. Use Case Explanation – Tie It All Together for CSC

Business Story

1. **CSC wants an easy way to spin up databases** for new apps without waiting for infra teams.

   → Atlas free cluster (M0) is perfect for **prototyping**.

2. **Developers in Bangalore & Mumbai need secure cloud access** to shared data.

   → We used **IP whitelisting** and **database users** to secure access.

3. **Compliance app needs to store entities and filings**.

   → We created csc_compliance database with entities and filings collections.

4. **They need to validate connectivity & basic CRUD** before building microservices.

   → We connected via **MongoDB Compass** and **mongosh** to insert & query records.

5. Later (future modules), they can:
   - Add **indexes** for performance
   - Use **Triggers**, **Charts**, **Realm**, **Backups**
   - Promote from **M0** → **Dedicated** when going to UAT/Prod.