ScopeTreasure Courses

MONGODB COURSES · LESSON MG.2 · CODE SCOPE SERIES

MongoDB Setup + Pehla Document

Your database on the internet for free: an Atlas cluster, the connection string, mongosh — and your first real document, born with its own ObjectId.

Introduction

A database that lives only inside a tutorial is a toy. Real apps keep their data on a server that never sleeps — and MongoDB gives you exactly that, free, through **MongoDB Atlas**: a cloud cluster you create in the browser in about five minutes. In this lesson you will build that free cluster, connect to it with **mongosh** (the MongoDB Shell), and insert your very first document — the same five students the SQL series used, reborn as documents.

(Roman Urdu: Atlas = MongoDB ka apna cloud ghar. Cluster banana free hai; pehla document aaj hi dalega.)

Key points

- **Atlas is MongoDB, hosted:** a **cluster** is your database server group on the internet. The free tier (M0, 512 MB) is honestly enough for learning and small apps — the same free cluster pattern this channel used for the SQL series' hosting lesson.
- **Two safety settings matter at signup:** a **database user** (a username + password that exist only for the database) and **Network Access** (which computers may knock). For learning, `0.0.0.0/0` (anywhere) is common — but know it means *the password is now the only lock*, so make it strong.
- **The connection string is the address + key in one line:** `mongodb+srv://user:password@cluster.../schoolDB`. Apps and tools both connect with it. It holds a password — it lives in an `.env` file, never in public code (Node apps in MG.5 do exactly this).
- **mongosh is your daily tool:** the MongoDB Shell — type a command, read the answer. `db` names the current database, `use schoolDB` switches to it, and a database springs into existence the moment its first document lands.
- **insertOne / insertMany are the birth commands:** `db.students.insertOne({...})` adds one document; `insertMany([...])` adds a list. MongoDB replies with an acknowledgement and the new `_id` values.
- **ObjectId is born, not made:** skip the `_id` field and MongoDB generates a 24-character `ObjectId` — unique across the world, with the birth time hidden inside it. The five students below keep the same ids all series, so you can follow each one.

Code examples

Example 1 — Connect, and look around your new home

```
mongosh "mongodb+srv://school-cluster.mongodb.net/schoolDB" --username dbuser
db
show collections
```

What it does: The first line is typed in your computer's terminal — it opens mongosh against your Atlas cluster, aimed at the `schoolDB` database (you will be asked for the database user's password). Inside the shell, `db` confirms where you are, and `show collections` lists what exists. Brand-new database: nothing — yet.

What you will see:

```
Current Mongosh Log ID: 650a1b2c3d4e5f60718293a4
Connecting to: mongodb+srv://<credentials>@school-cluster.mongodb.net/schoolDB
Using MongoDB: 7.0.14
Using Mongosh: 2.3.1
Atlas atlas-abc123-shard-0 [primary] schoolDB> schoolDB
Atlas atlas-abc123-shard-0 [primary] schoolDB>
```

That long prompt (`Atlas ... schoolDB>`) is the shell saying *connected, standing inside schoolDB*. `show collections` printed nothing after it — an empty home, waiting for its first document. (Connection lines are trimmed here; your cluster name and versions will differ slightly — that is normal.)

Example 2 — Your first document is born

```
db.students.insertOne({
  name: 'Ali Raza',
  class: '10-A',
  marks: 88,
  city: 'Sahiwal'
})
```

What it does: `db.students` names a collection that does not exist yet — MongoDB creates it on the spot. The object you pass is the document. You did **not** give an `_id`; watch what the reply hands back.

What you will see:

```
{
  acknowledged: true,
  insertedId: ObjectId('650000000000000000000001')
}
```

`acknowledged: true` is the database saying *received and saved*. And there it is: Ali's identity card, `ObjectId('...001')`, generated by MongoDB itself. From this line onward, `schoolDB` and its `students` collection truly exist — the first insert is what creates them.

Example 3 — The full class arrives: insertMany + find

```
db.students.insertMany([
  { name: 'Sara Khan', class: '10-A', marks: 92, city: 'Lahore' },
  { name: 'Bilal Ahmed', class: '10-B', marks: 76, city: 'Multan' },
  { name: 'Hira Shah', class: '10-B', marks: 95, city: 'Sahiwal' },
  { name: 'Usman Tariq', class: '9-A', marks: 64, city: 'Karachi' }
])
db.students.countDocuments()
```

What it does: One command, four documents — each gets its own ObjectId, listed in order in the reply. `countDocuments()` then counts the whole collection: Ali from Example 2 plus these four.

What you will see:

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('650000000000000000000002'),
    '1': ObjectId('650000000000000000000003'),
    '2': ObjectId('650000000000000000000004'),
    '3': ObjectId('650000000000000000000005')
  }
}
5
```

Five students, five ObjectIds (`...002` to `...005` — Sara, Bilal, Hira, Usman, in insert order), and the count proves the register is complete. Run `db.students.find()` and all five documents print in this same order — the exact dataset every later lesson in this series will query, so keep these numbers in mind: **Ali 88, Sara 92, Bilal 76, Hira 95, Usman 64.**

Quick recap

- **Atlas** = MongoDB's free cloud home; a cluster + a database user + network access is the whole setup.
- The **connection string** is address and key in one line — guard it like a password; it belongs in `.env`, not in code.
- **mongosh** commands to remember: `db` (where am I), `use <name>` (switch/create-on-first-write), `show collections`.
- **insertOne / insertMany** create documents; MongoDB answers with `acknowledged: true` and the new `ObjectId`s.

Common mistakes

- **Losing the connection string's password:** Atlas shows the user password once at creation. If you lose it, the fix is a new database user — so save credentials the moment the cluster is born, in your password notes, not in a chat message.
- **Network Access left closed:** the classic first-day failure — mongosh hangs or refuses because your own IP is not on the allow list. If connect fails, check Network Access before blaming the password.
  (Roman Urdu: Connect na ho to pehle Network Access dekho — aksar wahi darwaza band hota hai.)
- **Putting the real string in public code:** a connection string in a GitHub repo is a password posted in public — bots find them within hours. `.env` + `.gitignore`, always (G.3 in the Git series shows the ignore habit).
- **Waiting for the database to "be created" first:** beginners hunt for a Create Database button. In MongoDB the first insert *is* the creation — database and collection both appear with the data.

Practice

1. Create your own free Atlas cluster (M0), a database user, and Network Access — then connect with mongosh and run `db` and `show dbs`. How many databases exist before you insert anything, and why?
2. In a database named `meriDukaan`, insert one product document of your choice with `insertOne`, then prove with `show collections` and `countDocuments()` that it landed.
3. Insert the five students again in a *practice* database, then deliberately run `db.students.find()` — write down which ObjectId belongs to Bilal, and check it follows the insert order from Example 3.

Practice quiz

1. What is the Atlas free tier honestly enough for?

- Nothing, it is a demo only
- Learning and small apps: the M0 cluster with 512 MB
- Production apps with millions of users
- Only storing pictures

Answer: Learning and small apps: the M0 cluster with 512 MB

2. Where does a MongoDB connection string belong, and why?

- In public code, it is only an address
- In a .env file, never in public code, because it holds a password and bots find posted strings within hours
- In a chat message, for safekeeping
- In the database itself

Answer: In a .env file, never in public code, because it holds a password and bots find posted strings within hours

3. When do a MongoDB database and collection come into existence?

- When you click Create Database in Atlas
- With the first insert: the first document's arrival creates both
- When you install mongosh
- When you pay for a cluster

Answer: With the first insert: the first document's arrival creates both

---

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/mongodb-setup-first-document · More free tools and notes at https://scopetreasure.com