Mongodb

What is Mongodb 🙂

MongoDB is an open-source NoSQL document-oriented database developed by MongoDB Inc.

🗒️ What is NoSQL?

NoSQL stands for "Not Only SQL". It refers to a group of databases that do not use the traditional relational database model (tables and rows and SQL language).

📁 How MongoDB Stores Data in the Following Format

- **Database** → like in RDBMS.
- **Collection** → equivalent to tables.
- **Document** → equivalent to rows. A document is a JSON-like structure (BSON).

```
{
  "name": "Alice",
  "email": "alice@example.com",
  "age": 28,
  "skills": ["Node.js", "MongoDB", "React"]
}
```

✅ Advantages of MongoDB / NoSQL (Point-wise)

1. **Schema-less**
    - No fixed schema is required.
    - Allows storing different structures in the same collection.
2. **Scalability**
    - Supports horizontal scaling using sharding.
    - Ideal for handling large-scale, distributed data systems.
3. **High Performance**
    - Fast read and write operations.
    - Especially efficient with unstructured or semi-structured data.
4. **Data Flexibility**
    - Supports storing arrays, nested documents, and complex data types.

5. **Developer-Friendly**

   - Uses JSON-like syntax.
   - Very intuitive for developers, especially in JavaScript/Node.js environments.

6. **Ease of Use**

   - MongoDB Query Language (MQL) is simple and resembles JSON syntax.
   - Easier to learn and use compared to SQL for some tasks.

7. **Cloud-Native Compatibility**

   - Works seamlessly with cloud platforms and microservices architectures.
   - Suitable for modern web and mobile applications.

❌ Disadvantages of MongoDB / NoSQL (Point-wise)

1. **No Joins**
    - Traditional SQL-style JOIN operations are not supported.
    - May require data duplication or manual embedding, which can complicate data consistency.
2. **Schema Control Limitations**
    - The schema-less nature can lead to inconsistent or unpredictable data structures.
    - Developers need to implement extra validation to maintain data integrity.
3. **Less Mature Ecosystem**
    - May have fewer tools, less documentation, or less community support in some areas.

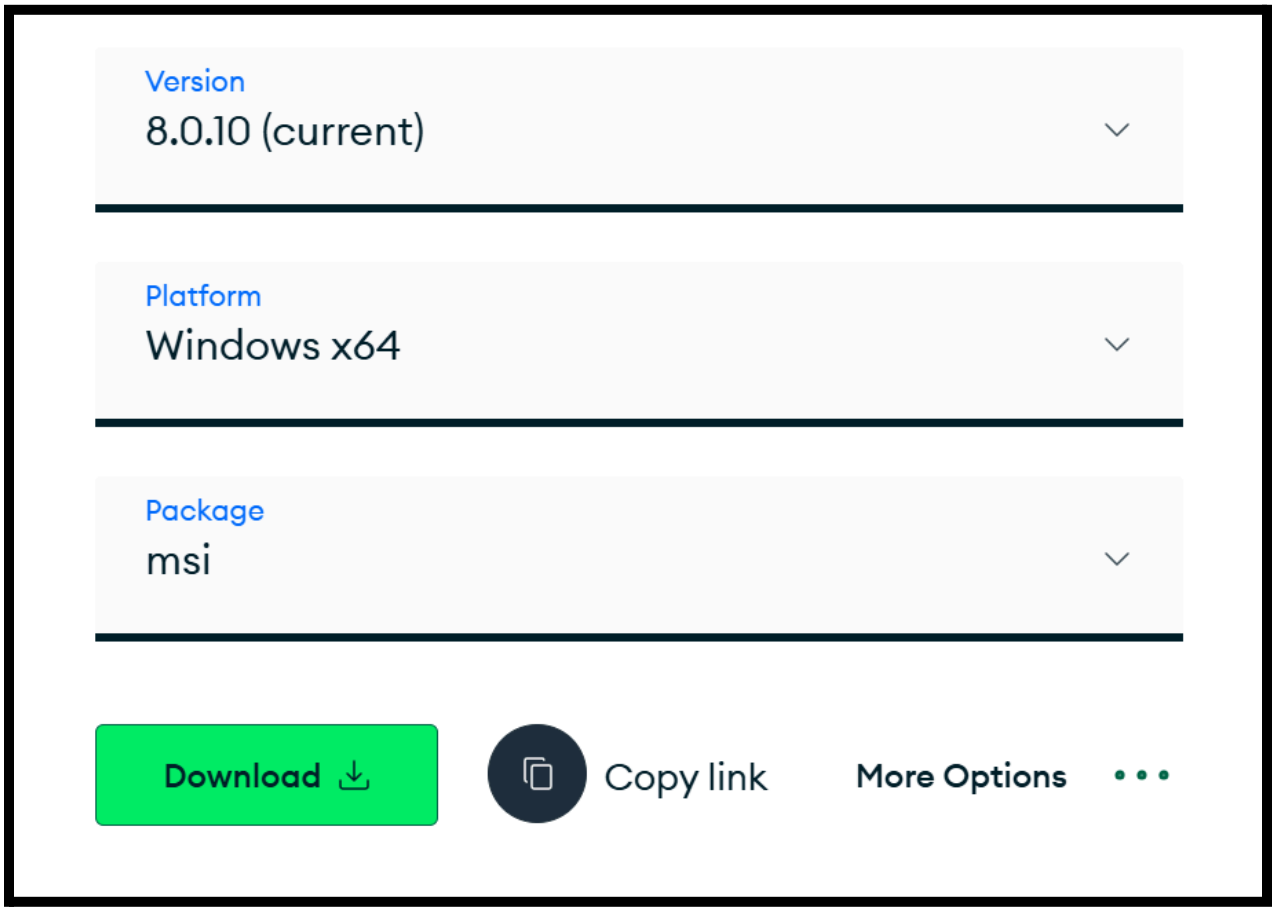
Installation of Mongodb Server 👋

✅ Step-by-Step Installation of MongoDB on Windows

🔽 1. Download MongoDB MSI Installer

- Go to the official MongoDB website:
  https://www.mongodb.com/try/download/community
- Choose the following options:
  - **Version**: Choose the latest stable release.
  - **Platform**: Windows
  - **Package**: MSI
- Click on **Download**.

📸 ***Screenshot Description*:**



👨‍💻 2. Run the Installer

- After downloading the `.msi` file, **double-click** to start the installation.

🛠️ 3. Accept the License Agreement

- Check the checkbox for **“I accept the terms…”**
- Click **Next**.

⚙️ 4. Choose Setup Type

- Select **"Complete"** (recommended for most users).
- Click **Next**.

🧰 5. Service Configuration

- Keep defaults:
    - **Install MongoDB as a Service** → ✅
    - **Service Name**: MongoDB
    - **Run service as Network Service user**
- Click **Next**.

📁 6. Install MongoDB Compass (Optional)

- Optionally, you can install **MongoDB Compass**, the GUI tool.
    - If you need it, leave the box checked.
    - **Otherwise, uncheck and proceed.**

🔧 7. Click Install

- Click the **Install** button and wait for the installation to complete.

✅ 8. Installation Completed

- Once the installation finishes, click **Finish**.

Installation of Mongo Client

✅ What is MongoDB Client (mongosh)?

The **MongoDB Shell (mongosh)** is the **command-line client** used to interact with a MongoDB database.

🔽 Step-by-Step Guide to Install MongoDB Client (mongosh) on Windows

---

🔹 1. Visit the Official MongoDB Shell Download Page

Go to:
🔗 https://www.mongodb.com/try/download/shell

🔹 2. Choose the Right Options

In the download dropdowns:

- **Select Version**: Latest (default)
- **Platform**: Windows
- **Package**: MSI

Then click on **Download**.

🔹 3. Run the Installer

After downloading the `.msi` file:

- Double-click it to launch the installer.
- Click **Next** through the setup steps.
- Accept the license agreement.
- Use **default installation path** or choose custom.
- Click **Install**.

🔹 4. Finish Installation

After the install completes, click **Finish**.

🔹 5. Add MongoDB Shell to PATH (if not done automatically)

To run `mongosh` from any folder in CMD:

Go to:

C:\Program Files\MongoDB\mongosh\<version>\bin

1. Copy the full path.
2. Open **System Environment Variables**:
    - Press `Win + S`, search for **Environment Variables**, and open it.
3. Click **Path** > Edit > **New** → paste the path.
4. Click **OK** to save.

🔹 6. Verify Installation

Open **Command Prompt** and type:

mongosh

You should see a MongoDB Shell prompt like:

Current Mongosh Log ID: 66f3...
Connecting to: mongodb://localhost:27017

🔹 7. Connect to a MongoDB Server

If MongoDB server is running locally:

mongosh

To connect to a remote server:

mongosh "mongodb://<hostname>:<port>"

🔷 1. Database Commands

✅ Create or Switch to a Database

```
use myDatabase
```

- If `myDatabase` does not exist, MongoDB creates it when you first store data in it.

📂 Show All Databases

```
show dbs
```

- Lists all existing databases (only those with at least 1 collection and data).

❌ Drop/Delete a Database

```
use myDatabase
db.dropDatabase()
```

- Deletes the currently selected database.

🔷 2. Collection Commands

✅ Create a Collection

```
db.createCollection("myCollection")
```

📂 Show All Collections in Current DB

```
show collections
```

❌ Drop/Delete a Collection

```
db.myCollection.drop()
```

🔷 3. CRUD Operations

Let's assume you are working with:

```
use myDatabase
db.createCollection("students")
```

🟢 Create (Insert)

Insert One Document

```
db.students.insertOne({ name: "Kabir", age: 24, course: "MongoDB" })
```

Insert Multiple Documents

```
db.students.insertMany([
  { name: "Asha", age: 22, course: "React" },
  { name: "Raj", age: 25, course: "Node.js" }
])
```

🟡 Read (Find)

Find All Documents

```
db.students.find()
```

Find One Document

```
db.students.findOne({ name: "Kabir" })
```

🟠 Update

Update One Document

```
db.students.updateOne(
  { name: "Kabir" },
  { $set: { course: "Full Stack" } }
)
```

Update Multiple Documents

```
db.students.updateMany(
  { course: "React" },
  { $set: { course: "Frontend" } }
)
```

🔴 Delete

Delete One Document

```
db.students.deleteOne({ name: "Raj" })
```

Delete Multiple Documents

```
db.students.deleteMany({ course: "Frontend" })
```

Here are 2 real-world examples using MongoDB CRUD operations with practical use cases:

✅ Example 1: Student Management System

🔹 Use Case:

You are building a web application to manage student records for a college.

📁 Collection: `students`

🔧 Operations:

1. Insert a new student

```
db.students.insertOne({
  name: "Priya Sharma",
  rollNo: "CS101",
  age: 20,
  department: "Computer Science",
  subjects: ["DBMS", "OS", "Networking"],
  isActive: true
})
```

2. Find all active students in Computer Science

```
db.students.find({ department: "Computer Science", isActive: true })
```

3. Update subject list for a student

```
db.students.updateOne(
  { rollNo: "CS101" },
  { $push: { subjects: "AI" } }
)
```

4. Remove a student record after graduation

```
db.students.deleteOne({ rollNo: "CS101" })
```

✅ Example 2: E-commerce Order System

🔹 Use Case:

You manage product orders in an online shopping platform.

📁 Collection: `orders`

🔧 Operations:

1. Insert a new order

```
db.orders.insertOne({
  orderId: "ORD123",
  customerName: "Amit Kumar",
  items: [
    { productId: "P1001", name: "T-shirt", quantity: 2, price: 499 },
    { productId: "P1023", name: "Shoes", quantity: 1, price: 1999 }
  ],
  totalAmount: 2997,
  orderDate: new Date(),
  status: "Pending"
})
```

2. Find all pending orders

```
db.orders.find({ status: "Pending" })
```

3. Update order status to "Shipped"

```
db.orders.updateOne(
  { orderId: "ORD123" },
  { $set: { status: "Shipped" } }
)
```

4. Delete canceled orders

```
db.orders.deleteMany({ status: "Canceled" })
```