# MongoDB

A NoSQL Database

## Introduction

**Definition of MongoDB:**

MongoDB is a leading NoSQL database that employs a document-oriented data model.

Unlike traditional relational databases, MongoDB stores data in flexible, JSON-like documents, providing a scalable and agile approach to data management.

**Importance of NoSQL Databases in Modern Applications:**

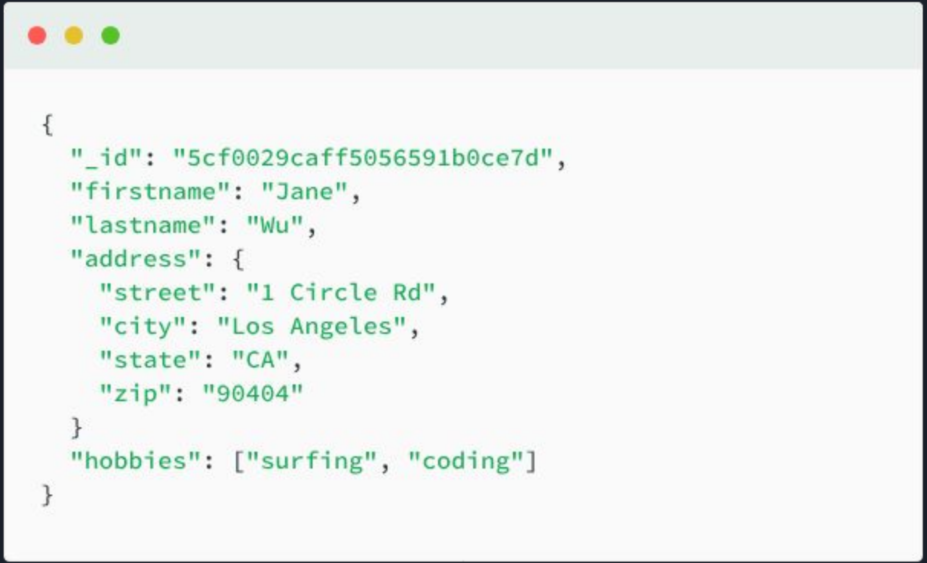
Traditional relational databases have limitations in handling unstructured and semi-structured data, such as  multimedia, and sensor data.

NoSQL databases, including MongoDB, offer flexible schemas and horizontal scalability, making them ideal for modern applications with dynamic and rapidly evolving data requirements.

The shift towards cloud computing, big data analytics, and real-time processing has increased the demand for NoSQL databases that can scale horizontally and handle diverse data types efficiently.

# Document-oriented Database

- MongoDB is classified as a document-oriented database, a type of NoSQL database that stores and retrieves data as documents. MongoDB employs a document-based data model.
- In MongoDB, data is stored in flexible, JSON-like documents. These documents are similar to the objects used in JavaScript Object Notation (JSON).



```
{
  "_id": "5cf0029caff5056591b0ce7d",
  "firstname": "Jane",
  "lastname": "Wu",
  "address": {
    "street": "1 Circle Rd",
    "city": "Los Angeles",
    "state": "CA",
    "zip": "90404"
  }
  "hobbies": ["surfing", "coding"]
}
```

## Comparison with Relational Databases

Relational Database (e.g., MySQL):

| CustomerID | Name | Age | Email | Address |
|---|---|---|---|---|
| 1 | John Smith | 30 | john@example.com | 123 Main St |
| 2 | Alice Brown | 25 | alice@example.com | 456 Elm St |
| 3 | Bob Johnson | 40 | bob@example.com | 789 Oak St |

In this example:

Each row represents a customer record.Columns represent specific attributes or fields of the customer data, such as CustomerID, Name, Age, Email, and Address.

All records within the customers table must adhere to the same schema, meaning that each row must contain values for all defined columns.

In MongoDB, customer data can be stored as flexible, JSON-like documents within a collection named customers. Each document can have its own structure, allowing for variation in fields and data types:

```
{
  "CustomerID": 1,
  "Name": "John Smith",
  "Age": 30,
  "Email": "john@example.com",
  "Address": "123 Main St"
 }
```

```
{
  "CustomerID": 2,
  "Name": "Alice Brown",
  "Age": 25,
  "Email": "alice@example.com",
  "Address": "456 Elm St",
  "Phone": "+1234567890",
 "Orders": [
   {
     "OrderID": 1001,
     "Product": "Laptop",
     "Quantity": 1
   }]
 }
```

In this MongoDB example:

Each document represents a customer record.

Documents can have varying structures and may contain different fields. For example, some documents may include additional fields like Phone or Orders.

Nested structures, such as the Orders array within the third document, allow for representing complex data relationships without the need for separate tables and foreign keys.

Advantages of MongoDB's Document Model:

MongoDB's document-oriented approach offers several advantages over relational databases:

**Schema Flexibility**: MongoDB documents do not enforce a fixed schema, allowing for dynamic and evolving data structures. Fields can be added, modified, or removed from documents without requiring alterations to the database schema.

**Hierarchical Data Representation**: MongoDB documents support nested structures and arrays, making it easy to represent complex data relationships and hierarchical data models.

## Features

**Scalability**: MongoDB's ability to scale horizontally refers to its capability to distribute data across multiple servers, also known as horizontal scaling or sharding.

Horizontal scaling in MongoDB is achieved through sharding, where data is partitioned and distributed across multiple shards (server instances).

**Flexibility**:MongoDB's schema flexibility refers to its ability to accommodate dynamic and evolving data structures without requiring a predefined schema.

**High Performance**: MongoDB's architecture and indexing capabilities contribute to its high performance, enabling fast read and write operations even at scale.

**Indexing:** MongoDB supports various types of indexes, including single-field, compound, and multi-key indexes, which facilitate fast data retrieval and query optimization.

# INDEXING

Indexing in MongoDB is a technique that allows the database to quickly locate and retrieve data without needing to search through every document in a collection. It functions similarly to an index in a book, helping the database engine to find data swiftly and efficiently.

- Indexes improve MongoDB query execution.
- Without indexes whole collection must be scanned(COLLSCAN).
- Index Stores sorted field values.
- If appropriate index exists MongoDB performs only index scans.

## Indexing Types

- **Single Field Indexes:** A single field index only includes data from a single field of the documents in a collection.
- **Compound Indexes :** A compound index includes more than one field of the documents in a collection.
- **Multikey Indexes:** A multikey index is an index on an array field, adding an index key for each value in the array.
- **Geospatial Indexes and Queries:** Geospatial indexes support location-based searches.
- **Text Indexes:** Text indexes support search of string content in documents.
- **Hashed Index:** Hashed indexes maintain entries with hashes of the values of the indexed field and are used with sharded clusters to support hashed shard keys.

## Index Creation

**Using CreateIndex**

- **Single: db.stud.createIndex( { zipcode: 1})**
- **Compound: db.stud.createIndex( { dob: 1, zipcode: -1 } )**
- **Unique: db.stud.createIndex( { rollno: 1 }, { unique: true } )**

**Using ensureIndex**

- **Single: db.stud.ensureIndex({“name":1})**
- **Compound: db.stud.ensureIndex ({“address":1,“name":-1})**

**Note:**

**1 for Ascending Sorting**

**-1 for Descending Sorting**

## Index Display

- **db.collection.getIndexes()**

  Returns an array that holds a list of documents that identify and describe the existing indexes on the collection.

- **db.collection.totalIndexSize()**

  Reports the total size used by the indexes on a collection  db.collection.

- **reIndex()**

  Rebuilds all existing indexes on a collection.

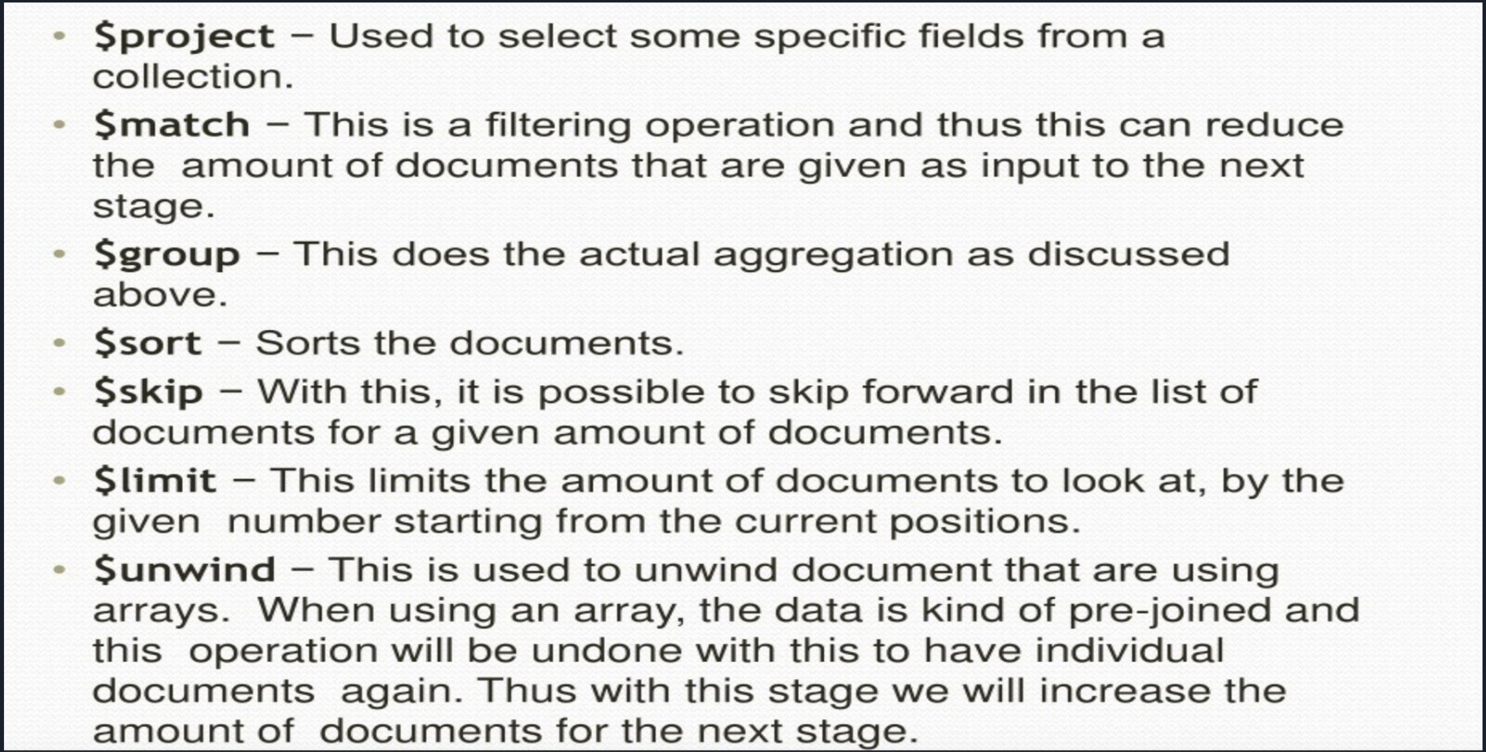
## Aggregations

- Aggregations operations process data records and return computed results.
- Aggregation operations group values from multiple documents together,and can perform a variety of operations on the grouped data.
- For aggregation in mongodb use aggregate() method.
- Syntax:
  >db.COLLECTION_NAME.aggregate(AGGREGATE_OPERATION)

## Aggregation

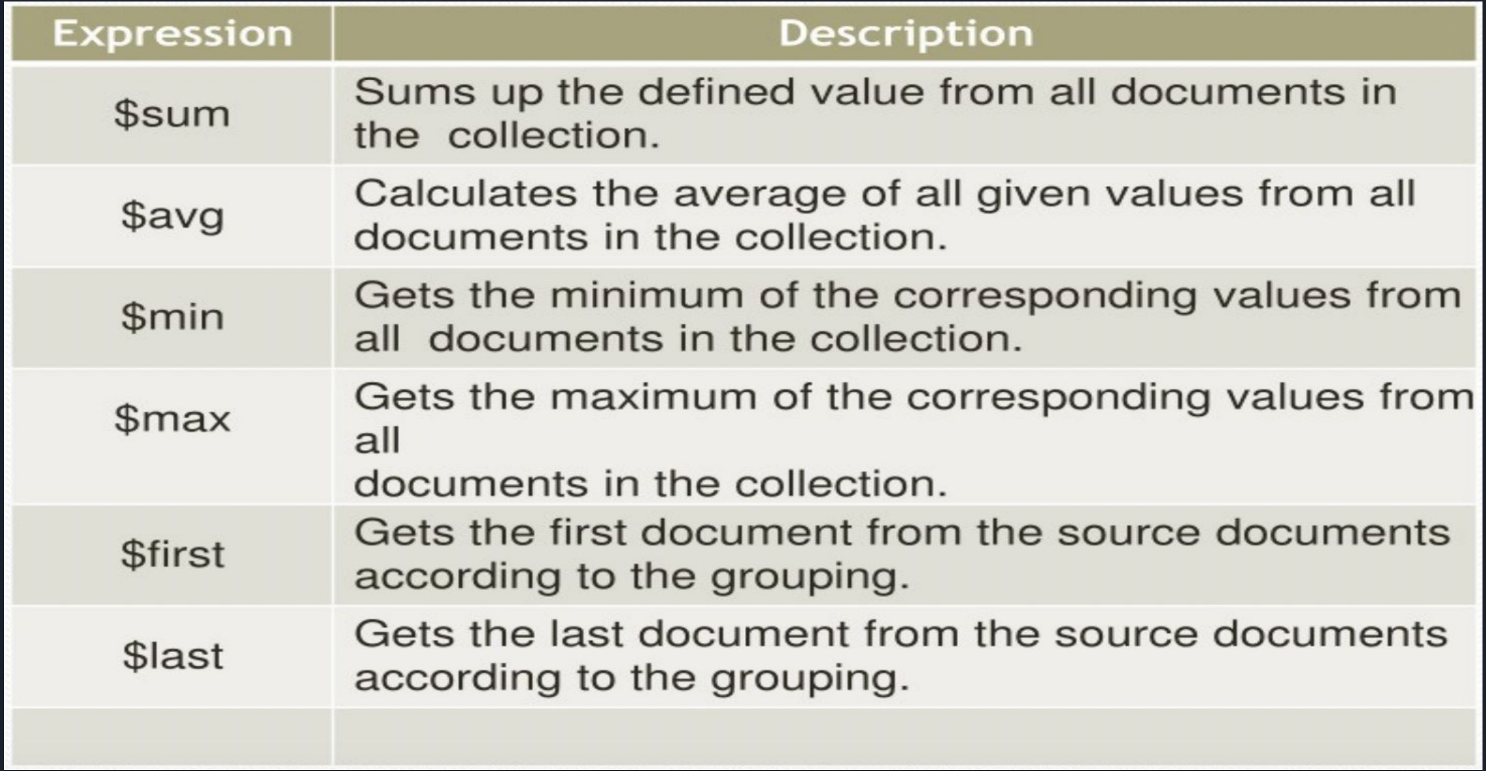
- MongoDB’s aggregation framework is modeled on the concept of data processing pipelines.
- Documents enter a multi-stage pipeline that transforms the documents into an aggregated result.
- Other pipeline operations provide tools for grouping and sorting documents by specific field or fields.
- In addition, pipeline stages can use operators for tasks such as calculating the average or concatenating a string.

## Pipeline Stages in aggregation



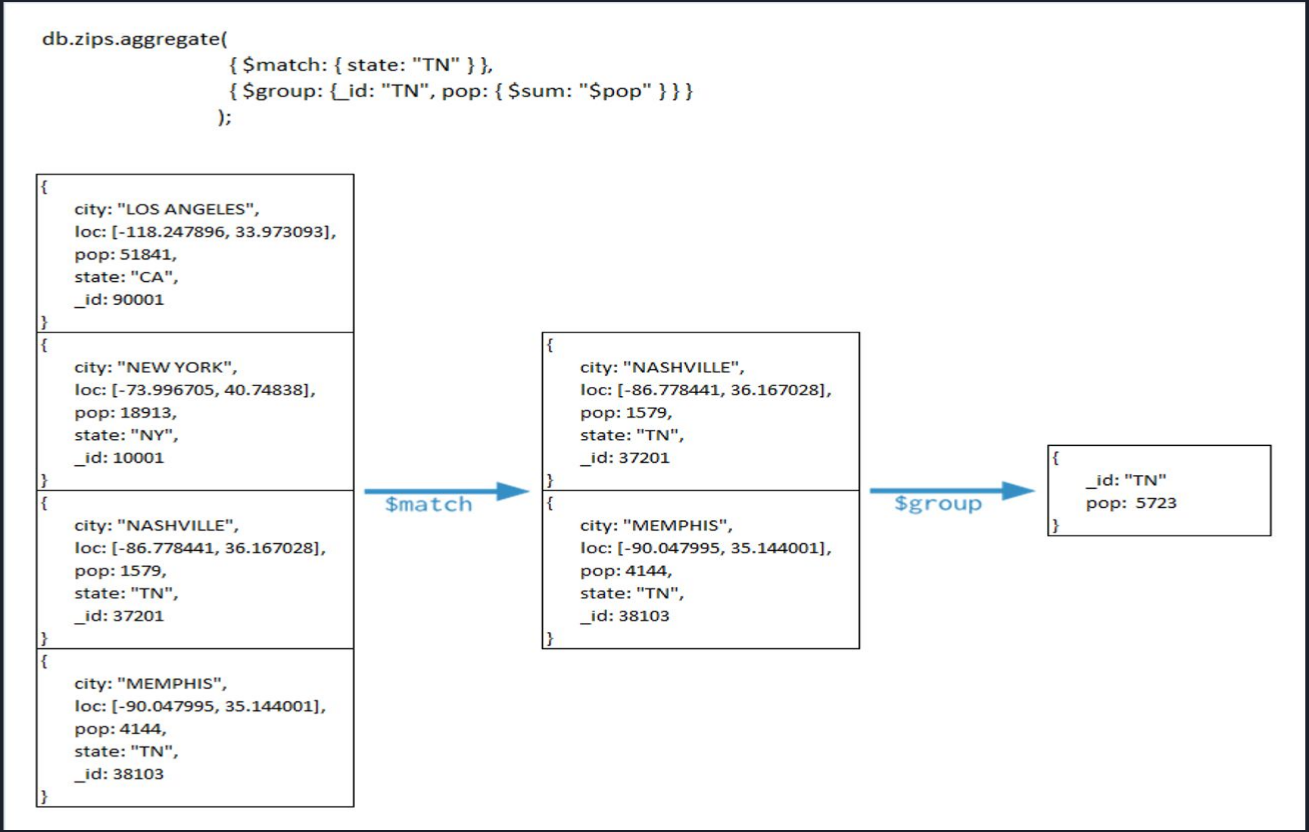
- **$project** – Used to select some specific fields from a collection.
- **$match** – This is a filtering operation and thus this can reduce the amount of documents that are given as input to the next stage.
- **$group** – This does the actual aggregation as discussed above.
- **$sort** – Sorts the documents.
- **$skip** – With this, it is possible to skip forward in the list of documents for a given amount of documents.
- **$limit** – This limits the amount of documents to look at, by the given number starting from the current positions.
- **$unwind** – This is used to unwind document that are using arrays. When using an array, the data is kind of pre-joined and this operation will be undone with this to have individual documents again. Thus with this stage we will increase the amount of documents for the next stage.

## aggregate() method



| Expression | Description |
|---|---|
| $sum | Sums up the defined value from all documents in the collection. |
| $avg | Calculates the average of all given values from all documents in the collection. |
| $min | Gets the minimum of the corresponding values from all documents in the collection. |
| $max | Gets the maximum of the corresponding values from all documents in the collection. |
| $first | Gets the first document from the source documents according to the grouping. |
| $last | Gets the last document from the source documents according to the grouping. |
| | |

# Example of aggregation



```
db.zips.aggregate(
                  { $match: { state: "TN" } },
                  { $group: {_id: "TN", pop: { $sum: "$pop" } } }
                 );
```

```
{
    city: "LOS ANGELES",
    loc: [-118.247896, 33.973093],
    pop: 51841,
    state: "CA",
    _id: 90001
}
{
    city: "NEW YORK",
    loc: [-73.996705, 40.74838],
    pop: 18913,
    state: "NY",
    _id: 10001
}
{
    city: "NASHVILLE",
    loc: [-86.778441, 36.167028],
    pop: 1579,
    state: "TN",
    _id: 37201
}
{
    city: "MEMPHIS",
    loc: [-90.047995, 35.144001],
    pop: 4144,
    state: "TN",
    _id: 38103
}
```

```
{
    city: "NASHVILLE",
    loc: [-86.778441, 36.167028],
    pop: 1579,
    state: "TN",
    _id: 37201
}
{
    city: "MEMPHIS",
    loc: [-90.047995, 35.144001],
    pop: 4144,
    state: "TN",
    _id: 38103
}
```

```
{
    _id: "TN"
    pop:  5723
}
```

# Authentication in MongoDB

**What is Authentication?**

Process of verifying the identity of a user or node connecting to the MongoDB server.

**MongoDB Authentication Mechanisms**

**SCRAM(Salted Challenge Response Authentication Mechanism) (Default)**: Secure, password-based authentication method.

**x.509 Certificates**: Utilizes client and server certificates for authentication, often used in high-security environments.

**LDAP Integration**: Connects MongoDB to an external LDAP service to manage user credentials centrally.

**Kerberos**: For enterprise environments, uses a ticket-based mechanism that does not transmit passwords over the network.

**MongoDB Atlas**: Offers integration with cloud identity providers through AWS IAM

## Authorization with RBAC in MongoDB

**What is Authorization?**

Process of determining if a user has the necessary permissions to perform an action or access resources.

**Role-Based Access Control (RBAC)**

Users are assigned roles that define their access to resources.

**Built-in Roles:** Pre-defined roles suitable for common use cases.

**Custom Roles:** Allows creation of bespoke roles with specific privileges.

**Example of Roles**

**read**: Allows data read operations.

**readWrite**: Allows data read and write operations.

**dbAdmin:** Provides administrative permissions on a database level.

**userAdmin**: Allows user management operations within a database.

## Best Practices for Securing MongoDB Deployments

**Encryption**

**In-Transit:** Use TLS/SSL to secure data as it moves between servers and applications.

**At-Rest**: Implement file system encryption or use MongoDB’s Encrypted Storage Engine.

**Network Access Control**

Restrict MongoDB instances to known IP addresses via firewalls or security groups.

Use bindIp configuration to limit which network interfaces can listen for MongoDB connections.

**Auditing and Monitoring**

Enable MongoDB's auditing features to log and monitor database activities.

Monitor these logs for unusual activities that might indicate a security threat.

**Regular Updates and Patching**

Keep MongoDB and its hosting environment up-to-date with the latest security patches.

## Replication for High Availability

What is Replication?

The process of synchronizing data across multiple servers.

Ensures high availability, data redundancy, and fault tolerance.

**MongoDB Replica Sets**

A replica set is a group of MongoDB servers that maintain the same data set, providing redundancy and high data availability.

**Primary Node**: Handles all write operations. All read operations can be configured to come from the primary or any secondary node.

**Secondary Nodes**: Replicate the primary's data set and can become the primary if it fails (automatic failover).

**Arbiters**: Optional nodes that participate in elections but do not hold data.

## Sharding for Horizontal Scaling

**Sharding:**

A method of distributing data across multiple machines to support very large datasets and high throughput operations.

MongoDB uses sharding to scale horizontally by dividing the data set and distributing the parts across multiple servers, or "shards".

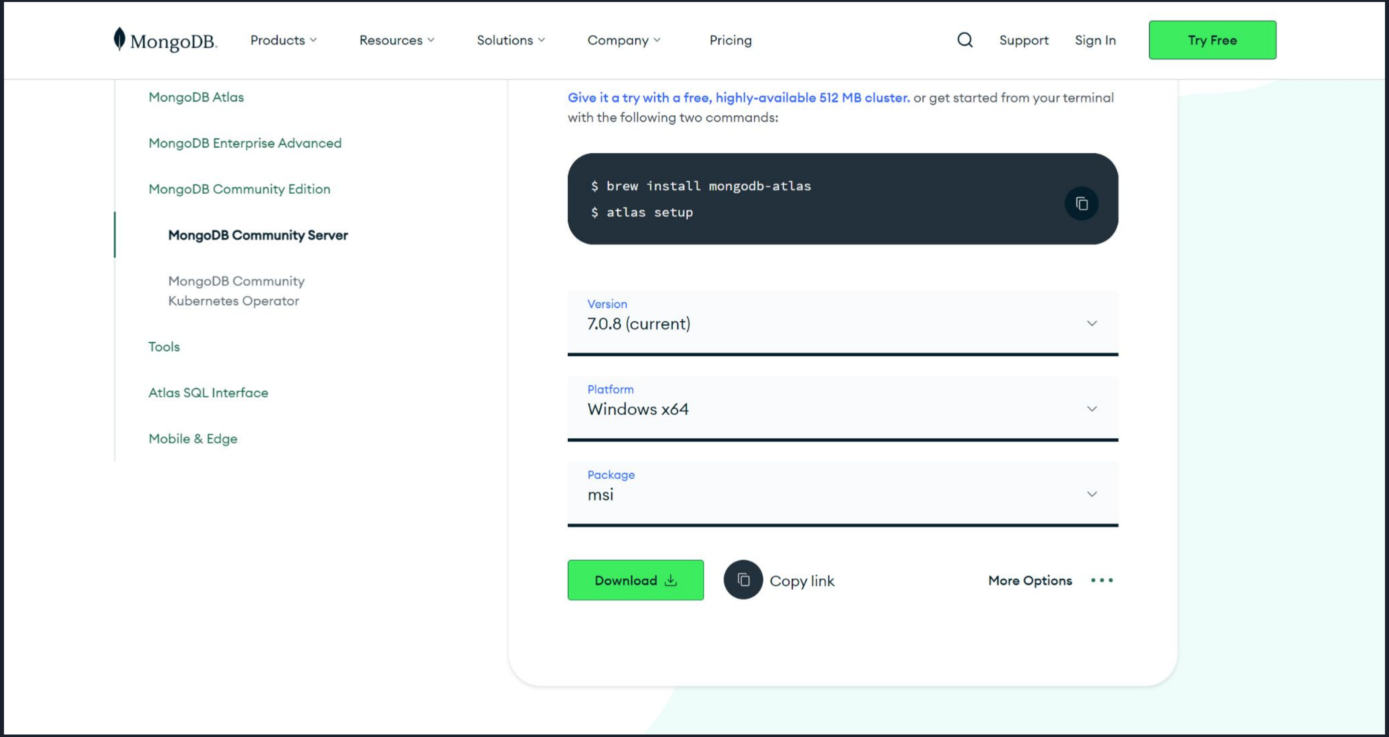
**Sharding Components**

**Shard**: Each holds a subset of the data.

**Config Servers**: Store the cluster’s metadata and configuration settings.

**Query Routers (mongos)**: Interface between client applications and the sharded cluster, directing operations to the appropriate shard(s).

## How to download MongoDB



Give it a try with a free, highly-available 512 MB cluster. or get started from your terminal with the following two commands:

```
$ brew install mongodb-atlas
$ atlas setup
```

Version
7.0.8 (current)

Platform
Windows x64

Package
msi

Link: https://www.mongodb.com/try/download/community

The wizard steps you through the installation of MongoDB and MongoDB Compass.

Choose Setup Type:

You can choose either the Complete (recommended for most users) or Custom setup type. The Complete setup option installs MongoDB and the MongoDB tools to the default location. The Custom setup option allows you to specify which executables are installed and where.

Service Configuration:

Starting in MongoDB 4.0, you can set up MongoDB as a Windows service during the install or just install the binaries

Install MongoDB Compass

Optional. To have the wizard install MongoDB Compass, select Install MongoDB Compass

## Mongosh

MongoDB Shell is the quickest way to connect to (and work with) MongoDB. Easily query data, configure settings, and execute other actions with this modern, extensible command-line interface — replete with syntax highlighting, intelligent autocomplete, contextual help, and error messages.

The .msi installer does not include mongosh

To use the MongoDB Shell, you must have a MongoDB deployment to connect to.

Link: https://www.mongodb.com/try/download/shell

THANK YOU