



INTRODUCTION

MongoDB is a popular NoSQL database known for its flexibility, scalability, and high performance. It stores data in JSON-like documents, allowing for dynamic schemas and complex data structures.

Key features include horizontal scaling through sharding, high availability with replica sets, powerful indexing, and a rich query language. MongoDB is widely used in content management systems, real-time analytics, and mobile applications due to its ability to handle large volumes of data and support for diverse and evolving data models.

What is it

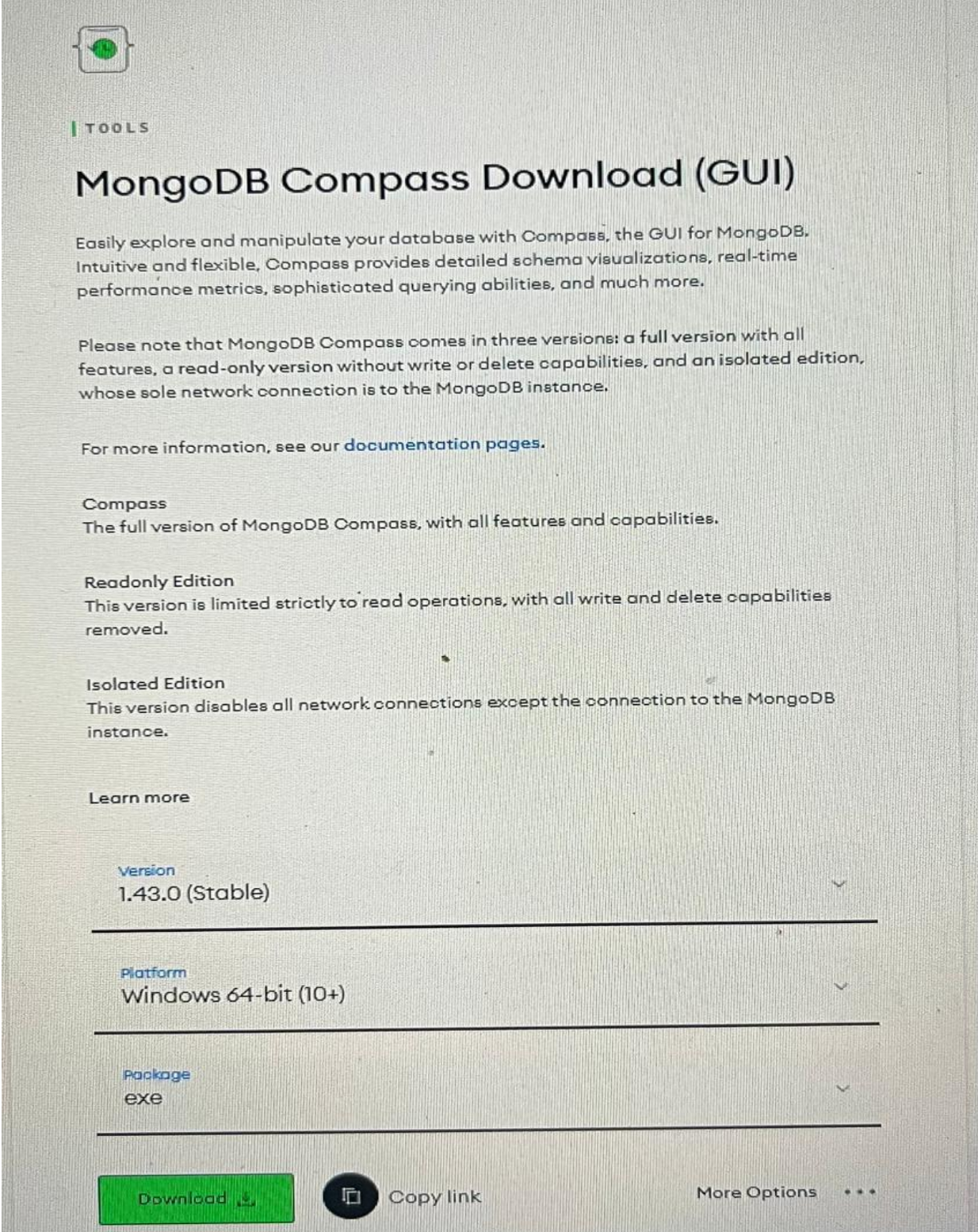
- A document-oriented database – documents encapsulate and encode data in some standard formats
- NoSQL database – non-adherence to the widely used relational database systems – highly optimized for retrieve and append operations
- It uses BSON format
 - It is schema-less – No more configuring database columns with types – Documents are self-describing – Documents are analogous to structures in programming languages that associate keys with values – The values of a field can be any of the BSON data types, including other documents, arrays, and arrays of documents
- No transaction


MONGODB INSTALLATION

You need to install mongodb compass and mongodb shell

STEP 1: just search mongodb compass download

STEP 2: mongodb compass download



 **TOOLS**

MongoDB Compass Download (GUI)

Easily explore and manipulate your database with Compass, the GUI for MongoDB. Intuitive and flexible, Compass provides detailed schema visualizations, real-time performance metrics, sophisticated querying abilities, and much more.

Please note that MongoDB Compass comes in three versions: a full version with all features, a read-only version without write or delete capabilities, and an isolated edition, whose sole network connection is to the MongoDB instance.


For more information, see our [documentation pages](#).


Compass
The full version of MongoDB Compass, with all features and capabilities.


Readonly Edition
This version is limited strictly to read operations, with all write and delete capabilities removed.




Isolated Edition
This version disables all network connections except the connection to the MongoDB instance.

[Learn more](#)

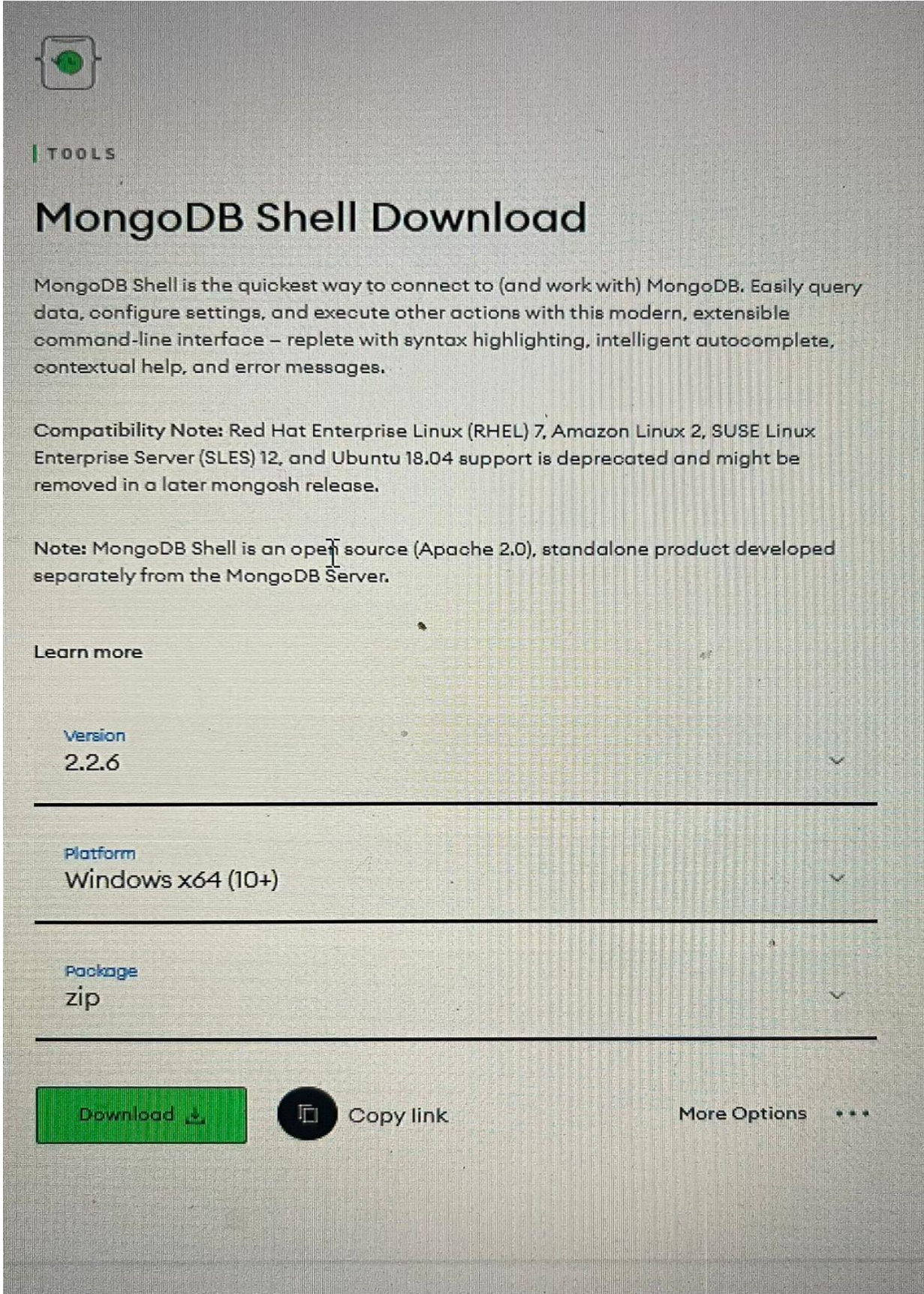
Version
1.43.0 (Stable) 

Platform
Windows 64-bit (10+) 


Package
exe 

[Download](#)   [Copy link](#) [More Options](#) 

STEP 3: install mongodb shell



The image shows the MongoDB Shell Download page. At the top left is the MongoDB logo. Below it is a 'TOOLS' section. The main heading is 'MongoDB Shell Download'. A paragraph describes the shell as the quickest way to connect to and work with MongoDB, highlighting its modern, extensible command-line interface with features like syntax highlighting, intelligent autocomplete, contextual help, and error messages. A 'Compatibility Note' states that support for Red Hat Enterprise Linux (RHEL) 7, Amazon Linux 2, SUSE Linux Enterprise Server (SLES) 12, and Ubuntu 18.04 is deprecated and might be removed in a later mongosh release. A 'Note' mentions that MongoDB Shell is an open source (Apache 2.0) standalone product developed separately from the MongoDB Server. Below this is a 'Learn more' link. The download configuration section has three rows: 'Version' set to '2.2.6', 'Platform' set to 'Windows x64 (10+)', and 'Package' set to 'zip'. Each row has a dropdown arrow. At the bottom, there are three buttons: a green 'Download' button with a download icon, a 'Copy link' button with a copy icon, and a 'More Options' button with three dots.



TOOLS

MongoDB Shell Download


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.


Compatibility Note: Red Hat Enterprise Linux (RHEL) 7, Amazon Linux 2, SUSE Linux Enterprise Server (SLES) 12, and Ubuntu 18.04 support is deprecated and might be removed in a later mongosh release.


Note: MongoDB Shell is an open source (Apache 2.0), standalone product developed separately from the MongoDB Server.

[Learn more](#)

Version	2.2.6	▼
Platform	Windows x64 (10+)	▼
Package	zip	▼

Download 

 Copy link

More Options 

DATABASE

- MongoDB groups collections into databases
- A single instance of MongoDB can host several databases
- Each database groups together zero or more collections
- A database has its own permissions, users and roles and each database is stored in separate files on disk
- A good rule is storing all data for a single application in the same database
- Reserved database names:
 - admin: the “root” database, in terms of authentication. If a user is added to the admin database, the user automatically inherits permissions for all databases.
 - Config: used for storing information of sharded setup

DATABASE SCHEMA

- Documents have flexible schema
 - Collections do not enforce specific data structure
- Key decision of data modeling:
 - References vs. embedded documents
 - In other words: Where to draw lines between aggregates
 - *Structure of data
 - *Relationships between data
- Embedded Documents
 - Related data in a single document structure
- Documents can have subdocuments (in a field or array)

STRUCTURE OF DATABASE

The information is typically organized in a specific format, often using tables with rows and columns. This makes it easier to search, filter, and analyze the data.

DATABASE MANAGEMENT SYSTEM

This is the software that acts like the filing cabinet manager. It allows you to store, retrieve, update, and manage all the data within the database.

DATA TYPE

There are several components that make up the structure of a typical database:

1. **Tables:** Tables are the basic building blocks of a database. They consist of rows and columns, where each row represents a record and each column represents a specific attribute or field of that record.
2. **Columns:** Columns define the type of data that can be stored in a particular field of a table. For example, a column might be defined as storing text, numbers, dates, or other data types.
3. **Rows:** Rows, also known as records or tuples, represent individual instances of data stored in a table. Each row contains a set of values corresponding to the columns defined in the table.
4. **Keys:** Keys are used to uniquely identify each row in a table. The primary key is a column or set of columns that uniquely identifies each record in the table. Foreign keys are used to establish relationships between tables.
5. **Indexes:** Indexes are data structures that improve the speed of data retrieval operations on a database table. They are created on one or more columns of a table and provide a quick way to look up data based on the values in those columns.
6. **Constraints:** Constraints are rules that enforce data integrity within the database. Common types of constraints include primary key constraints, foreign key constraints, unique constraints, and check constraints.
7. **Relationships:** Relationships define how data in one table is related to data in another table. Common types of relationships include one-to-one, one-to-many, and many-to-many relationships.

SQL AND NO SQL

SQL (Structured Query Language) and NoSQL (Not OnlySQL) databases differ in several aspects, including their data model, schema flexibility, scalability, and querying capabilities. Here's a breakdown of the key differences:

1. **Data Model:**

- **SQL (Relational Databases):** Relational databases organize data into tables with rows and columns. Tables have predefined schemas, meaning the structure of the data (column names, data types, constraints) must be defined before data can be inserted.
- **NoSQL Databases:** NoSQL databases use various data models, such as document, key-value, column-family, or graph. The most common NoSQL data model is the document model, where data is stored in flexible, JSON-like documents within collections (equivalent to tables in SQL).

2. **Schema Flexibility:**

- **SQL:** Relational databases enforce a rigid schema, meaning all rows in a table must have the same structure. Any changes to the schema require altering the table, which can be cumbersome.
- **NoSQL:** NoSQL databases offer schema flexibility. Each document or record can have its own structure within the same collection, allowing for dynamic schemas and easier evolution of the data model.

3. **Scalability:**

- **SQL:** Traditional relational databases typically scale vertically, meaning you need to upgrade hardware (CPU, RAM) to handle increased load. Horizontal scaling (scaling out across multiple machines) can be challenging and often requires complex sharding techniques.
- **NoSQL:** NoSQL databases are designed for horizontal scalability. They can easily scale out across multiple servers, distributing data and workload efficiently. This makes them well-suited for handling large volumes of data and high traffic.

4. **Querying:**

- **SQL:** Relational databases use SQL for querying data. SQL provides a powerful, standardized language for querying, filtering, aggregating, and manipulating data across multiple tables using joins.
- **NoSQL:** NoSQL databases have different querying mechanisms based on their data model. For example, document databases like MongoDB use query languages tailored to their data model, allowing for efficient retrieval and manipulation of JSON-like documents.

5. Transactions and ACID Properties:

- **SQL:** Relational databases typically support ACID (Atomicity, Consistency, Isolation, Durability) transactions, ensuring data integrity and reliability.
- **NoSQL:** NoSQL databases may offer varying levels of transaction support, with some sacrificing strong consistency for scalability. Many NoSQL databases provide eventual consistency guarantees rather than strong consistency.

6. Use Cases:

- **SQL:** Relational databases are well-suited for applications with complex queries, multi-row transactions, and strong consistency requirements. They are commonly used in traditional enterprise applications, finance, and applications with complex data relationships.
- **NoSQL:** NoSQL databases excel in scenarios requiring high scalability, flexible schemas, and rapid development. They are commonly used in web applications, mobile apps, real-time analytics, IoT, and situations with large volumes of unstructured or semi-structured data.

