

Introduction to MongoDB

MongoDB is an open-source, cross-platform, distributed document database. MongoDB is developed by [MongoDB Inc.](https://www.mongodb.com/) and categorized as a NoSQL database.

1) Easy to use

MongoDB is a document-oriented database. It uses the concept of the document to store data, which is more flexible than the row concept in the relational database management system (RDBMS).

A document allows you to represent complex hierarchical relationships with a single record.

MongoDB doesn't require predefined schemas that allow you to add to or remove fields from documents more quickly.

2) Designed to scale out

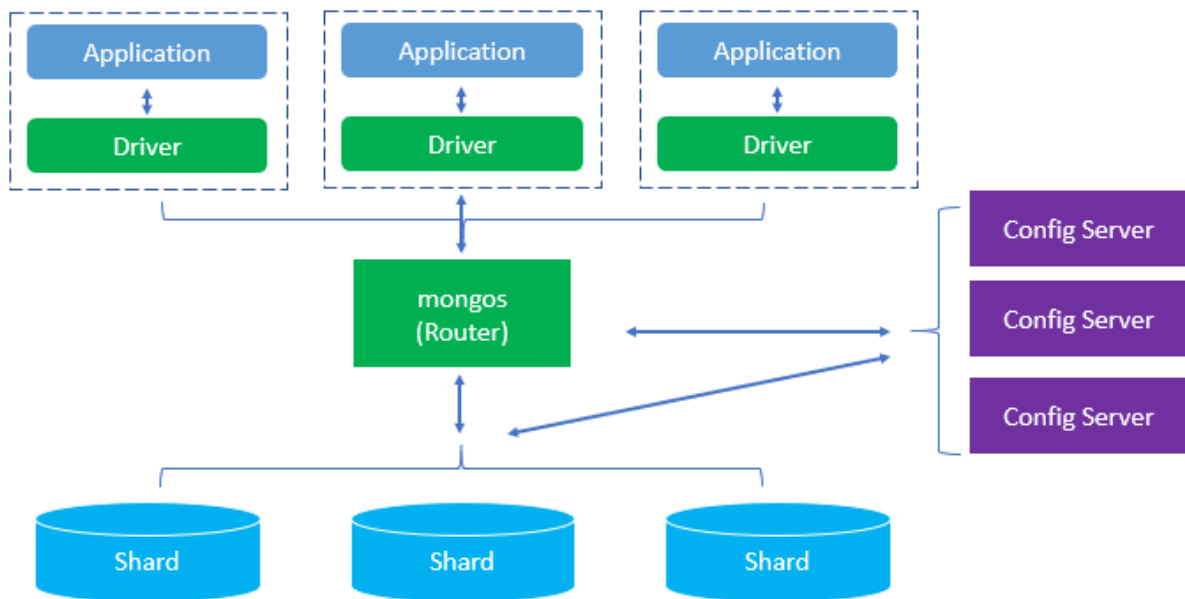
When the database grows, you'll have a challenge of how to scale it. There are two common ways:

- Scaling up – upgrade the current server to a bigger one with more resources (CPU, RAM, etc). However, getting a bigger server means increasing more costs.
- Scaling out – purchase additional servers and add them to the cluster. This is cheaper and more scalable than scaling up. The downside is that it takes more effort to manage multiple servers than a big one.

MongoDB was designed to scale out.

MongoDB allows you to split data across many servers. It also automatically manages the load balancing across the cluster, redistributing data, and routing updates to the correct servers.

The following picture illustrates how MongoDB scale-out using sharding across multiple servers:



3) Rich features

Like any database system, MongoDB allows you to insert, update, and delete, and select data. In addition, it supports other features including:

- Indexing
- Aggregation
- Specify collection and index types
- File Storage

Note that you will learn more about these features in detail in the next tutorial.

4) High performance

MongoDB was designed to maintain the high performance from both architecture and feature perspectives.

The philosophy of MongoDB is to create a full-featured database that is scalable, flexible, and fast.

MongoDB editions

MongoDB has three editions: community server, enterprise server, and atlas.

1) MongoDB Community Server

The MongoDB Community Edition is free and available on Windows, Linux, and macOS.

MongoDB Community Edition uses the [Server Side Public License](#) (SSPL). It means that if you offer MongoDB as a service to the public, you need to open the source code of the software

that makes the service works e.g., administration and monitoring tools. Otherwise, you need to pay for the enterprise subscription.

If you use the MongoDB Community Edition as a component of your application, not the final product, you are free to use it.

2) MongoDB Enterprise Server

MongoDB Enterprise Server is a commercial edition of MongoDB as a part of the MongoDB Enterprise Advanced subscription.

3) MongoDB Atlas

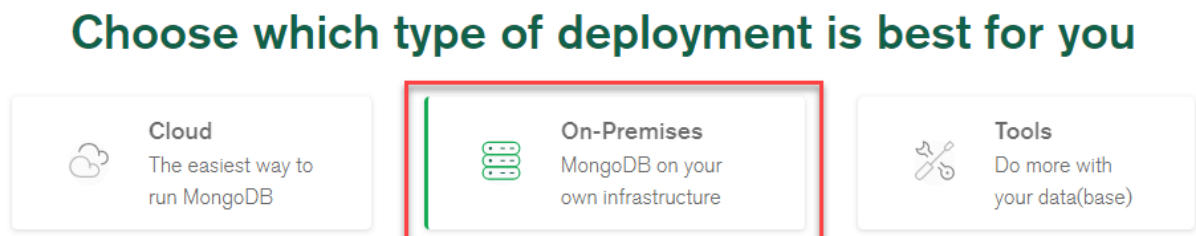
MongoDB Atlas is a global cloud database service. It is a database as a service that allows you to focus on building apps rather than spending time managing the databases.

MongoDB Atlas is available on common cloud platforms such as AWS, Azure, and GCP. MongoDB Atlas has a free tier for your experiments.

Download MongoDB Community Server

First, visiting the [download page](#) on the mongodb.com website.

Second, click the On-Premises (MongoDB on your own infrastructure tab)



Third, select the MongoDB Community Server.


Finally, select the version, platform, and click the download button to download the installation file.

Available Downloads ▼

Version
4.2.8 ▼

Platform
Windows ▼

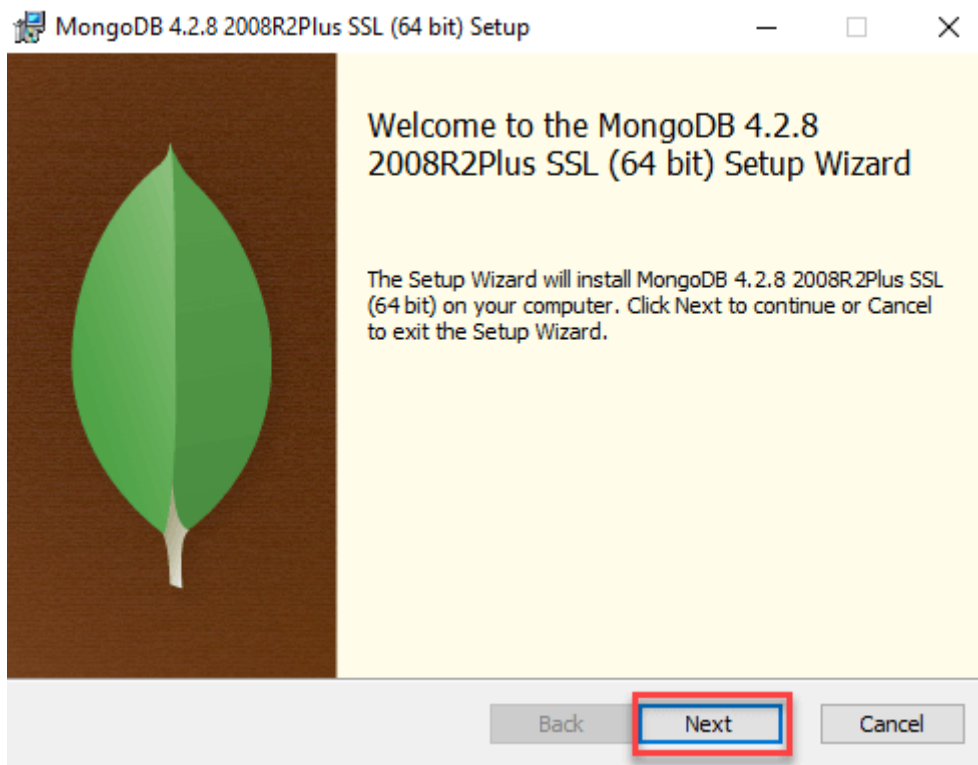
Package
msi ▼

 **Download** [Copy Link](#)

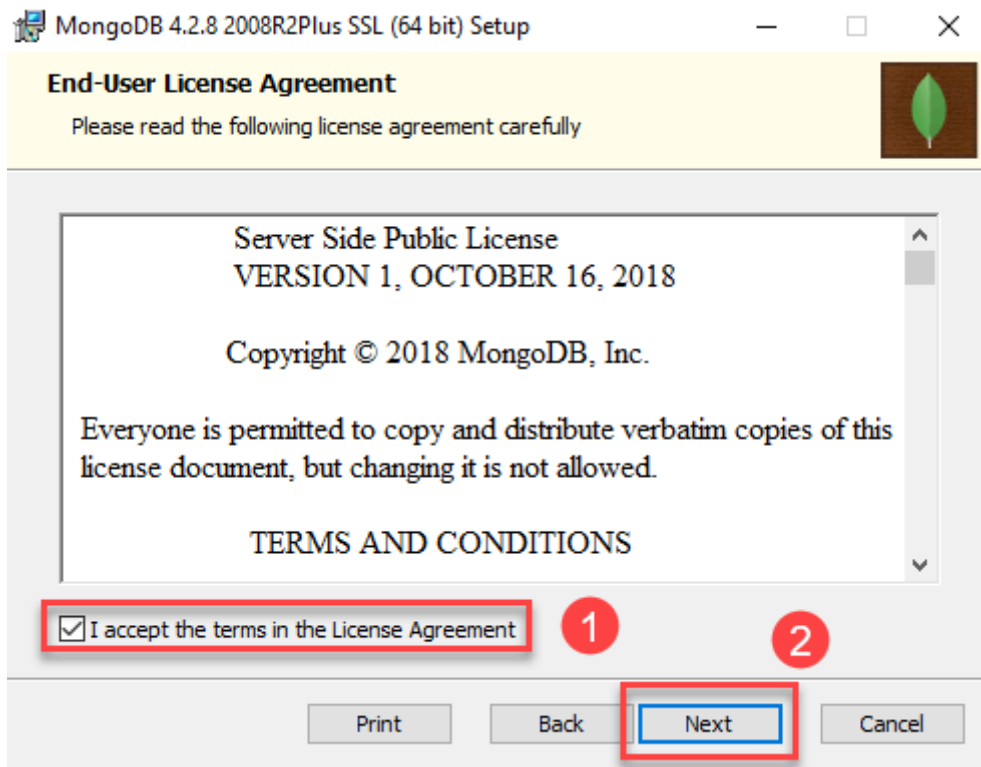
[Changelog](#)
[Release Notes](#)

Install MongoDB Community Server on the local machine

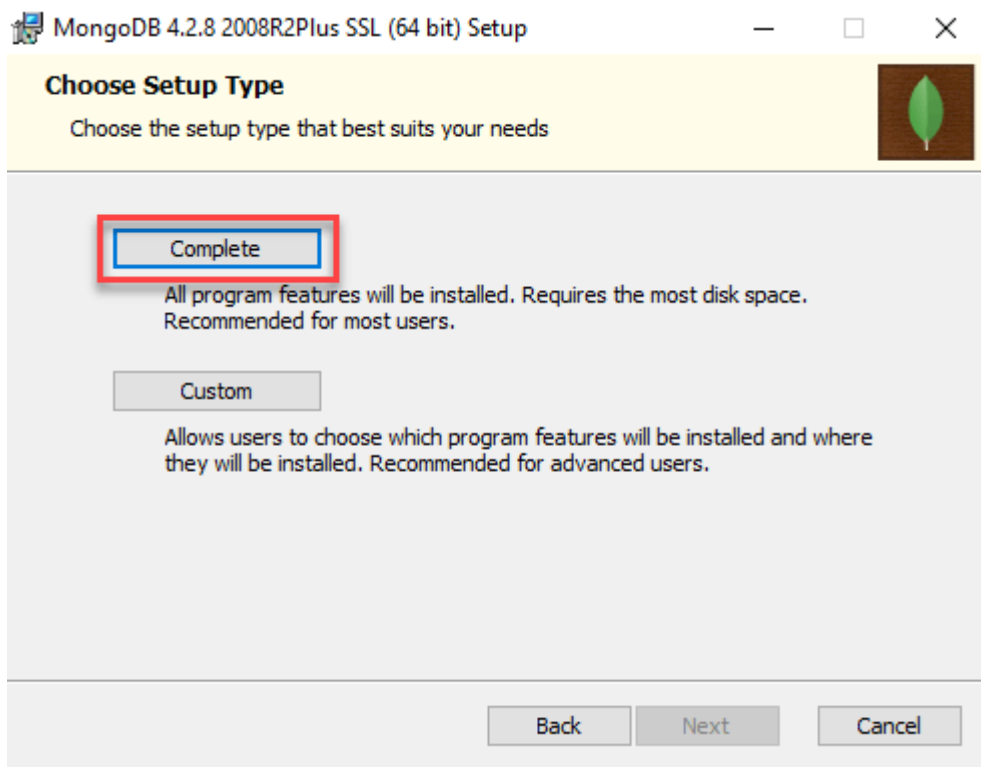
First, double-click the downloaded file to launch the setup wizard. Click the Next button to start setting up MongoDB.



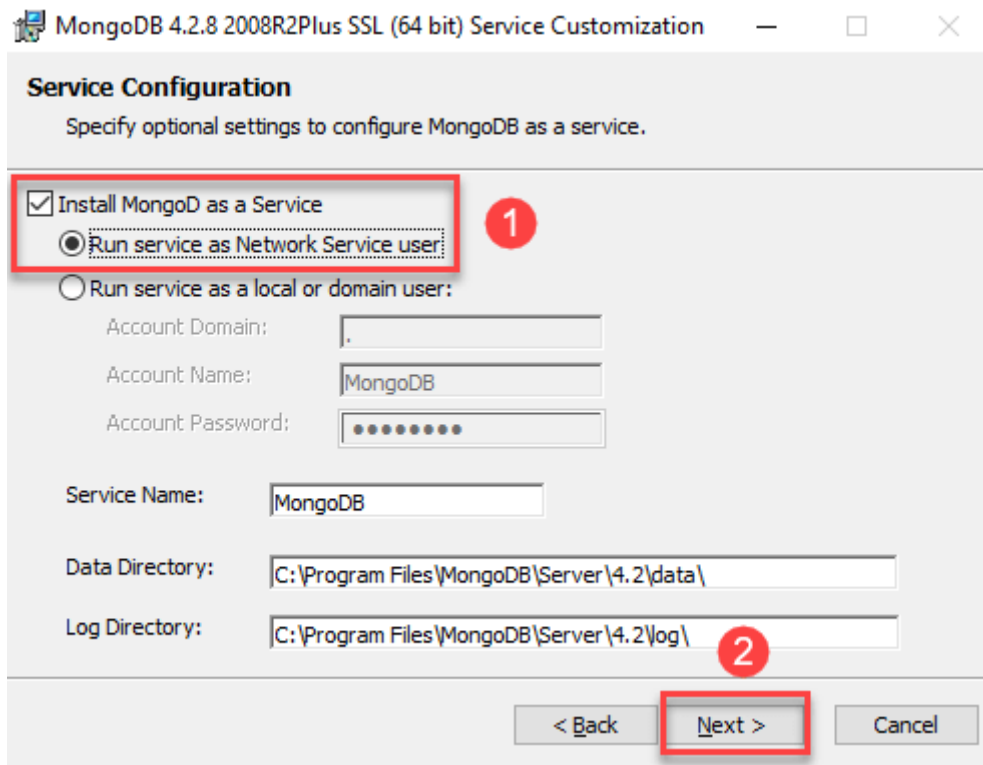
Second, accept the terms in the License Agreement and click the Next button:



Third, click the **Complete** button to install all program features. If you want to select which features to install and where they will be installed, select the Custom button. However, this option is recommended for advanced users only.



Fourth, select Install MongoDB as a service and click the Next button.

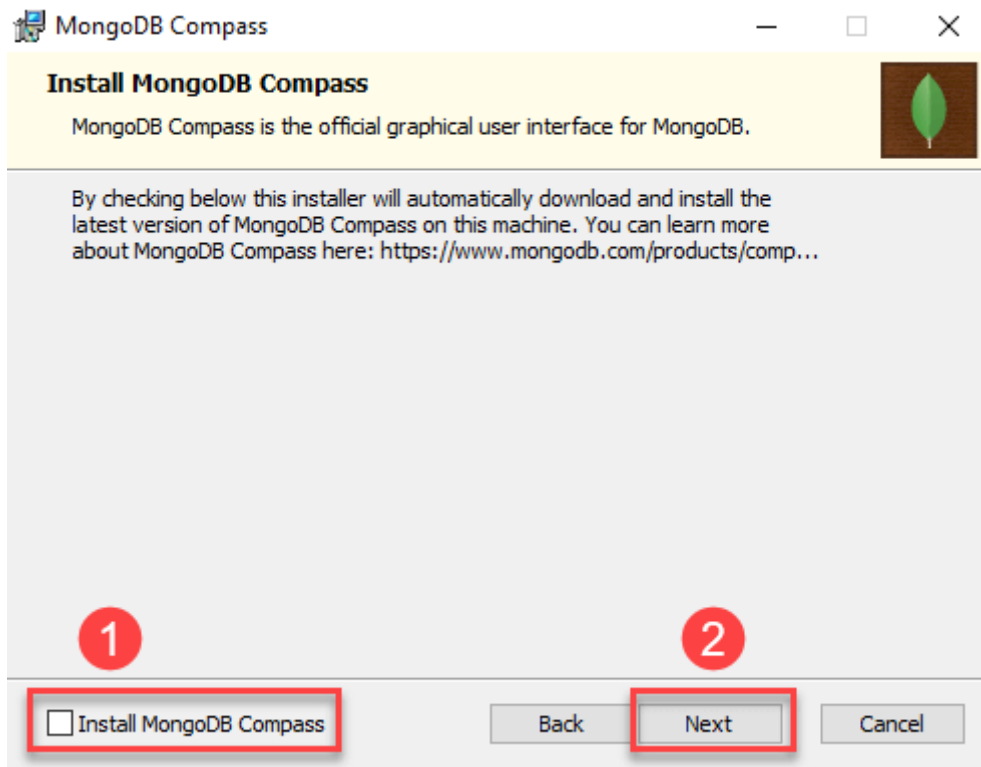


Fifth, uncheck the **Install MongoDB Compass** checkbox and click the **Next** button. If you select it, the setup wizard will also install the MongoDB Compass with the community version.

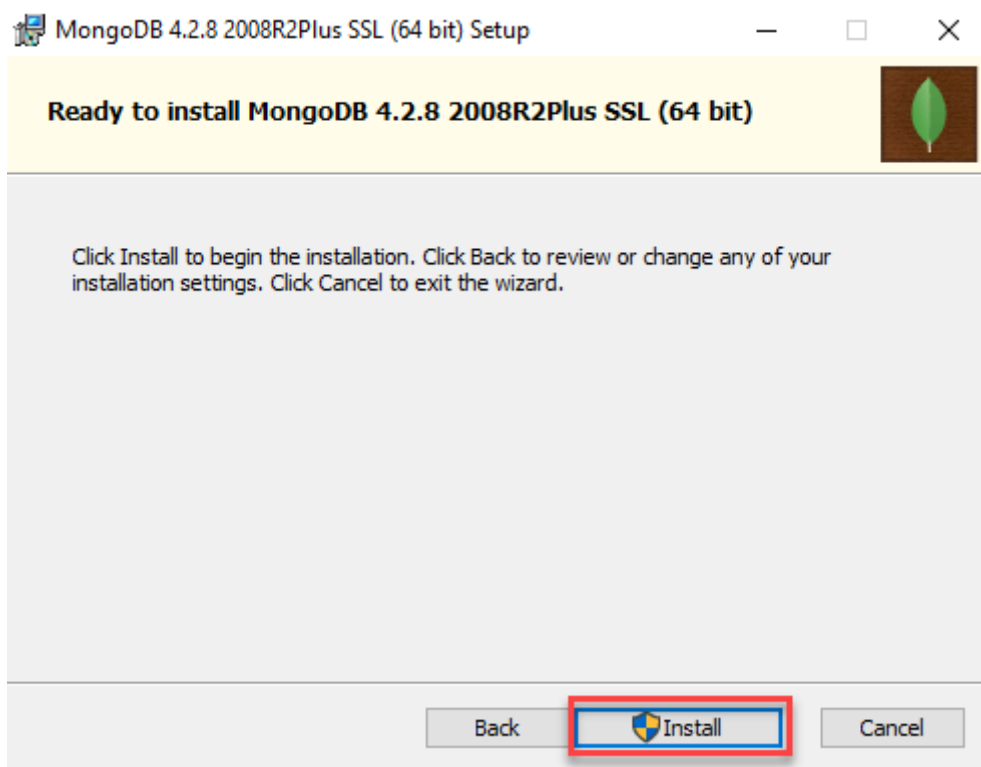
MongoDB Compass is a GUI tool that allows you to interact with MongoDB server including querying, indexing, document validation, and more.

MongoDB Compass has several versions. Like MongoDB, the community version always is free, but with limited features.

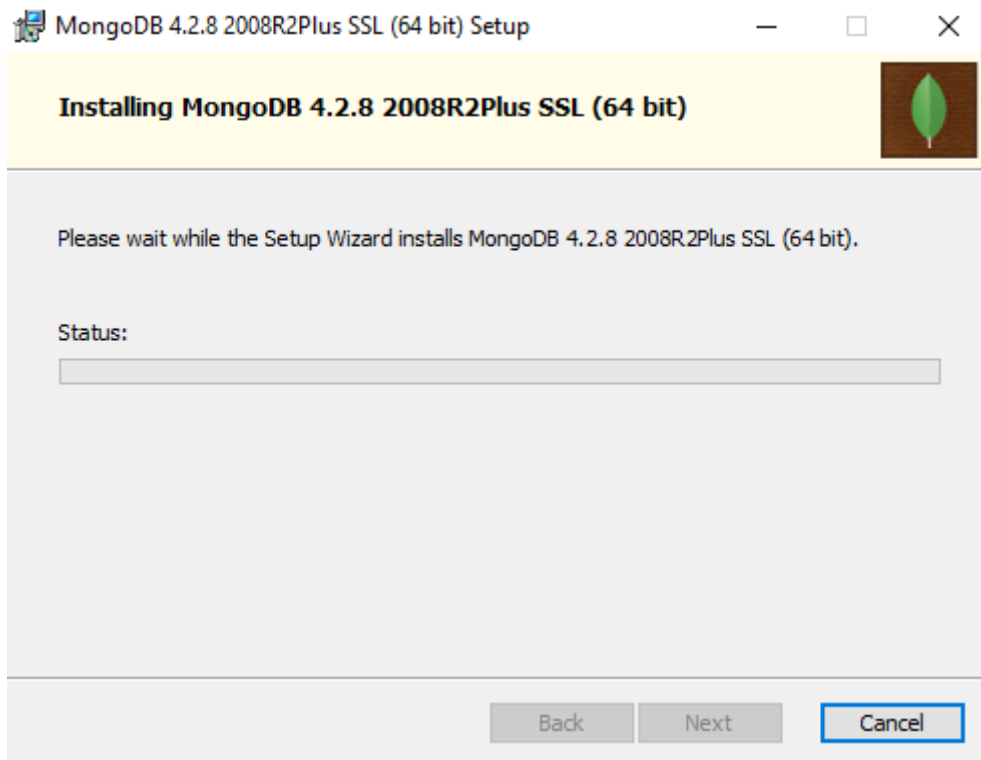
The good news is that [starting from April 30, 2020, the full version of MongoDB Compass is free for all](#). You will learn how to download and install MongoDB Compass in the next section.



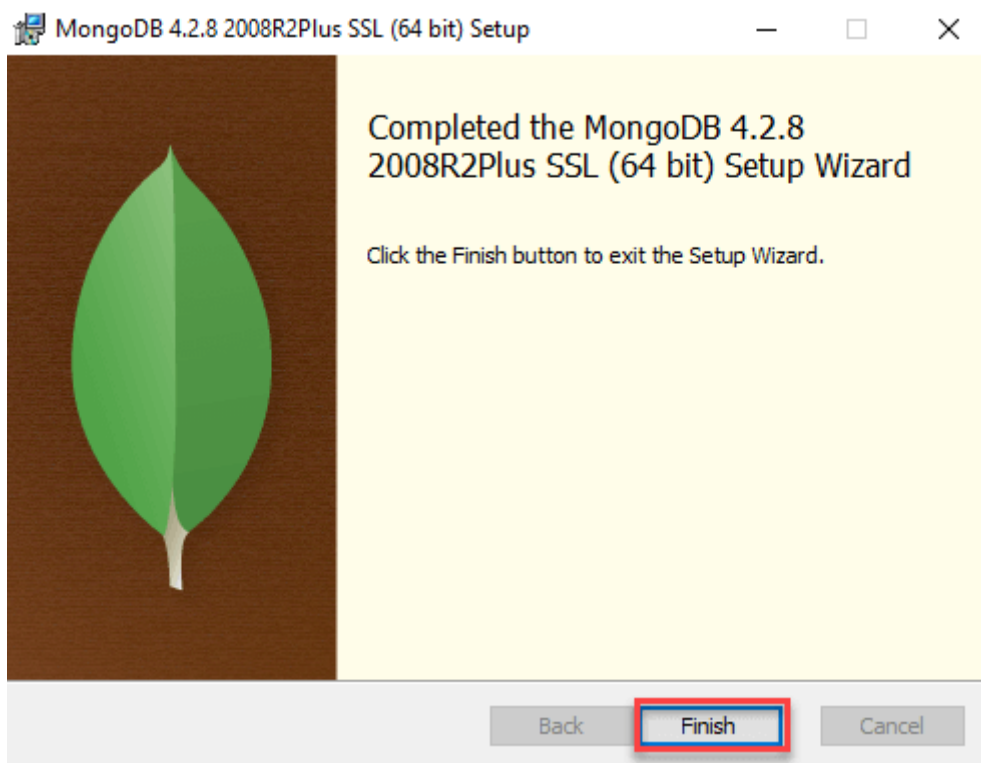
Sixth, click the Install button to start the installation.



The install will take some minutes to complete.



Seventh, click the **Finish** button to exit the setup wizard.



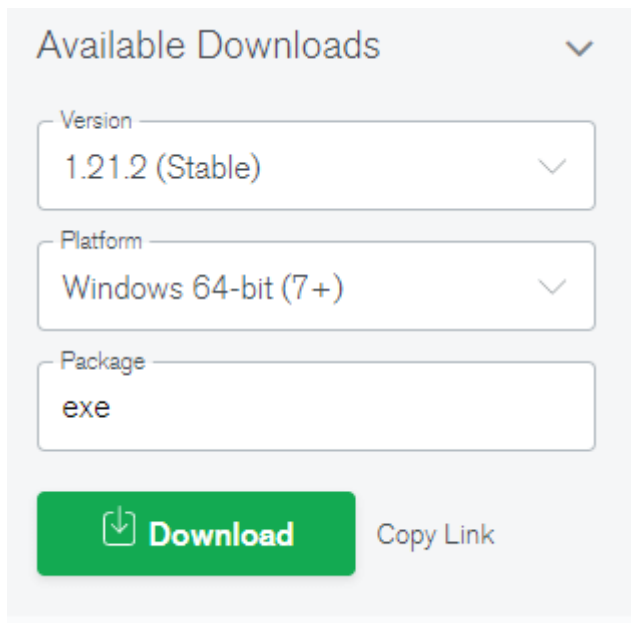
Download MongoDB Compass

You follow these steps to download MongoDB Compass:

First, visit the [download page on the mongodb.com website](#).

Second, select the MongoDB Compass tool.

Third, select the stable version to download. Note that the version that you see on the website may be higher than the version on the following screenshot:




Available Downloads

Version
1.21.2 (Stable)

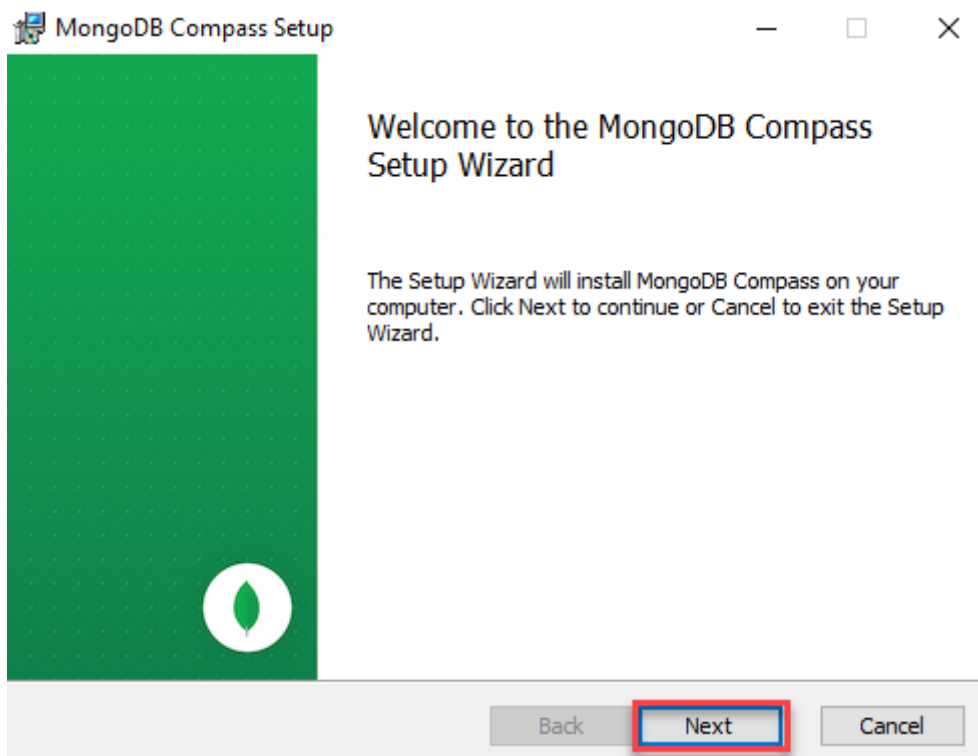
Platform
Windows 64-bit (7+)

Package
exe

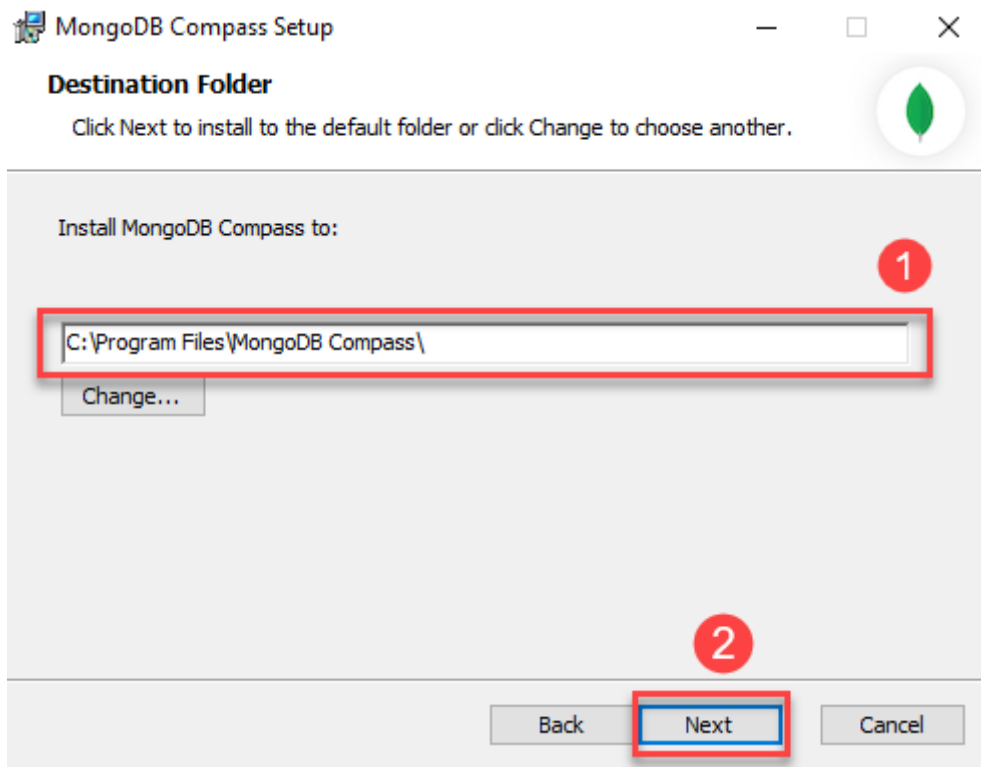
 **Download** [Copy Link](#)

Install MongoDB Compass

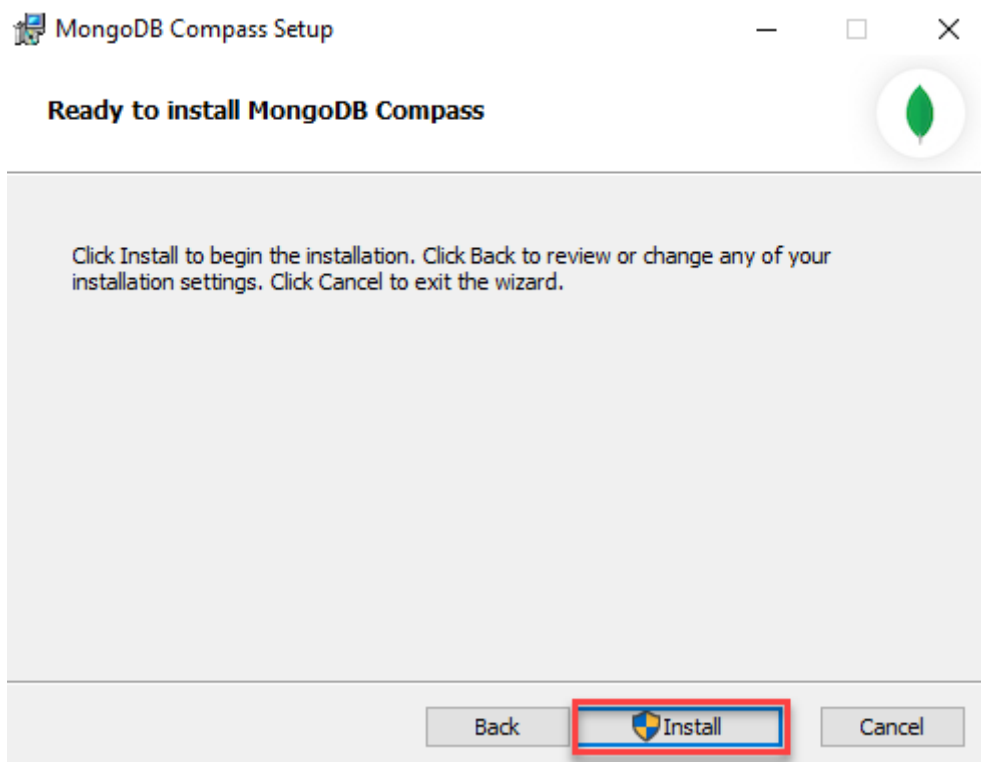
First, double-click the installer file to launch the MongoDB Compass Setup Wizard and click the Next button.



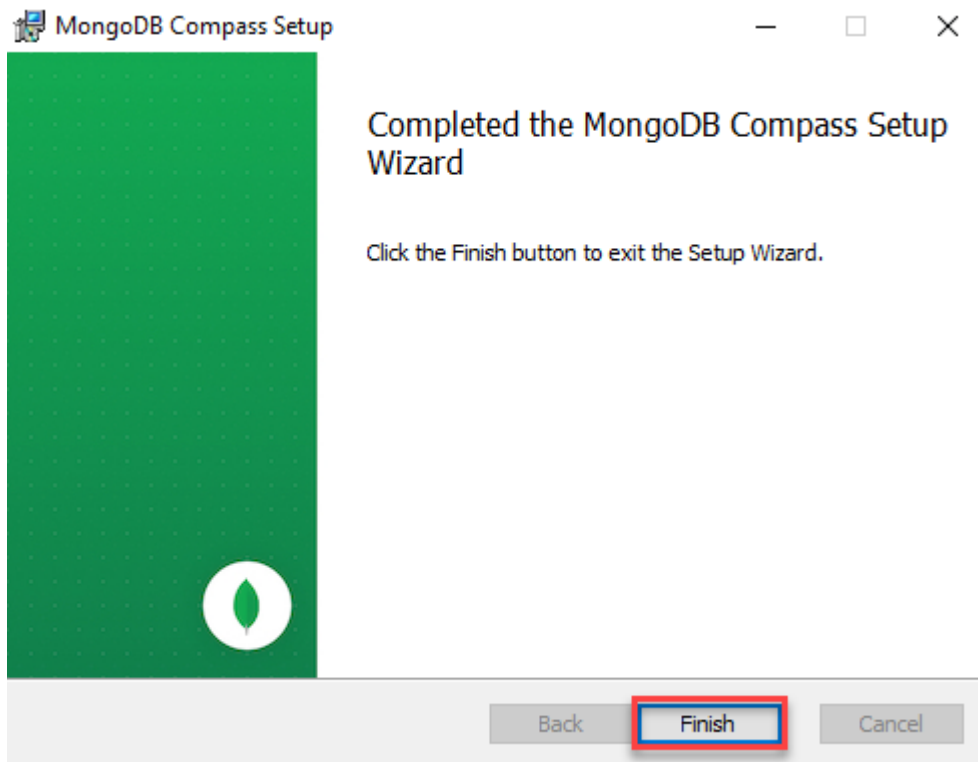
Second, select the destination folder and click Next to install to the default folder which is C:\Program Files\MongoDB Compass



Third, click the Install button to begin the installation.



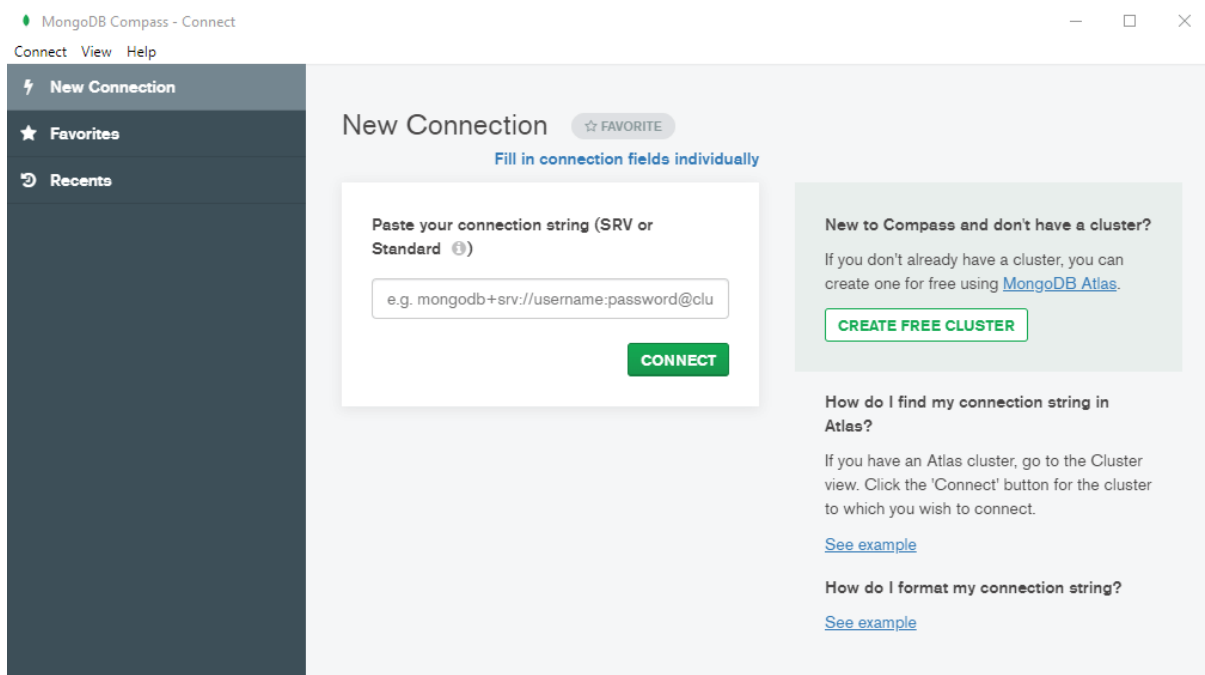
Fourth, once the installation is completed, click the Finish button to exit the Setup Wizard.



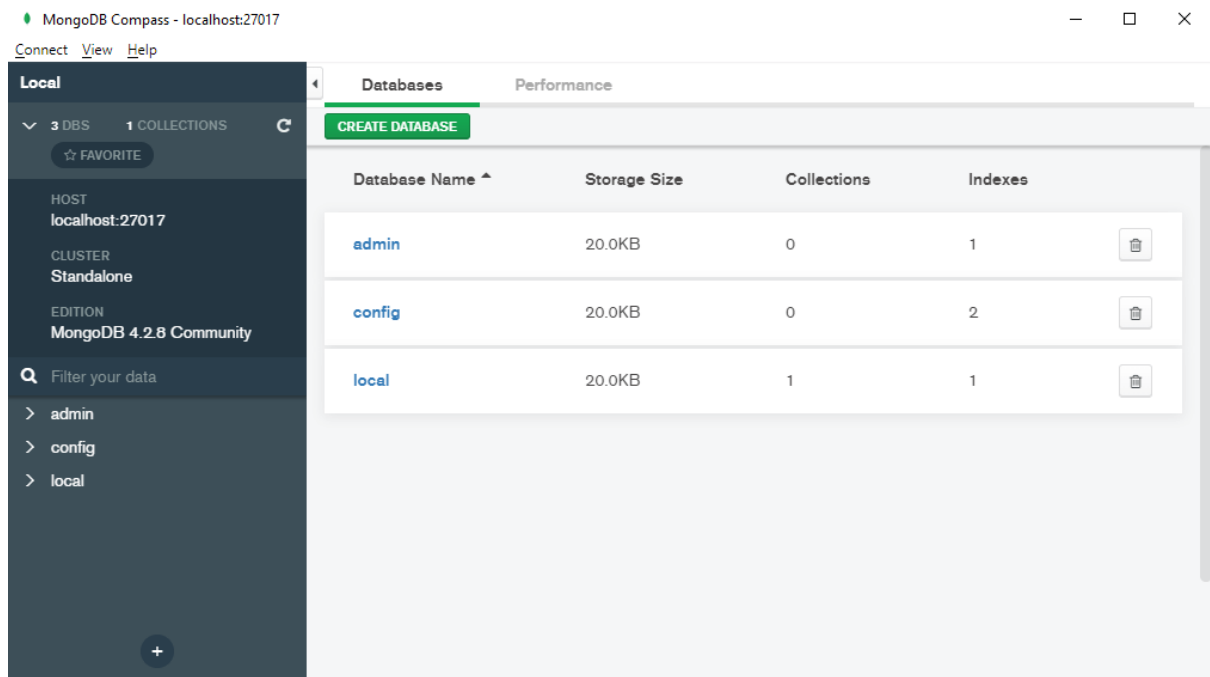
You'll find the MongoDB Compass in the Start menu.

To launch the MongoDB Compass, click the MongoDB Compass icon. The following shows the main screen of the MongoDB Compass.

To connect to the local MongoDB Server, click the **Connect** button:



If everything is fine, you will see the following screen:



In this tutorial, you've learned step by step how to install the MongoDB server and MongoDB Compass on your computer. Now, you're ready to learn about MongoDB.

Data formats

In MongoDB, you will often deal with JSON and BSON formats. Therefore, it's important to fully understand them.

JSON

JSON stands for JavaScript Object Notation. JSON syntax is based on a subset of JavaScript ECMA-262 3rd edition.

A JSON document is a collection of fields and values in a structured format. For example:

```
{  
  "first_name": "John",  
  "last_name": "Doe",  
  "age": 22,  
  "skills": ["Programming", "Databases", "API"]  
}
```

Code language: JSON / JSON with Comments (json)

BSON

BSON stands for Binary JSON, which is a binary-coded serialization of JSON-like documents.

Documents

MongoDB stores data records as BSON documents, which are simply called documents.

```
{
  _id: ObjectId("5f339953491024badf1138ec"),
  title: "MongoDB Tutorial",
  isbn: "978-4-7766-7944-8",
  published_date: new Date('June 01, 2020'),
  author: {
    first_name: "John",
    last_name: "Doe"
  }
}
```

A document is a set of field-and-value pairs with the following structure:

```
{
  field_name1: value1,
  field_name2: value2,
  field_name3: value3,
  ...
}
```

In this syntax, the field names are strings and values can be numbers, strings, objects, arrays, etc. For example:

```
{
  _id: ObjectId("5f339953491024badf1138ec"),
  title: "MongoDB Tutorial",
  isbn: "978-4-7766-7944-8",
  published_date: new Date('June 01, 2020'),
  author: { first_name: "John"
, last_name: "Doe"}
}
```

Code language: CSS (css)

This document has the following field-and-value pairs:

- `_id` holds an ObjectId
- `title` holds a string.
- `isbn` holds a string.
- `published_date` holds a value of the Date type.
- `author` holds an embedded document that contains two fields `first_name` and `last_name`.

If you are familiar with a relational database management system (RDBMS), you will find that a document is similar to a row in a table, but it is much more expressive.

Field names have the following restrictions:

- MongoDB reserves the field `_id` and uses it to uniquely identify the document.
- Field names cannot contain null characters.
- Top-level field names cannot start with the dollar sign (\$) character.

Collections

MongoDB stores documents in a collection. A collection is a group of documents.



A collection is analogous to a table in an RDBMS.

MongoDB	RDBMS
Documents	Rows
Collections	Tables

Unlike a table that has a fixed schema, a collection has a dynamic schema.

It means that a collection may contain documents that have any number of different “shapes”. For example, you can store the following documents in the same collection:

```
{
  title: "MongoDB Tutorial",
  published_date: new Date('June 01, 2020')
}
```

```
{
  title: "MongoDB Basics",
  published_date: new Date('Jan 01, 2021'),
  isbn: "978-4-7766-7944-8"
}
```

Code language: PHP (php)

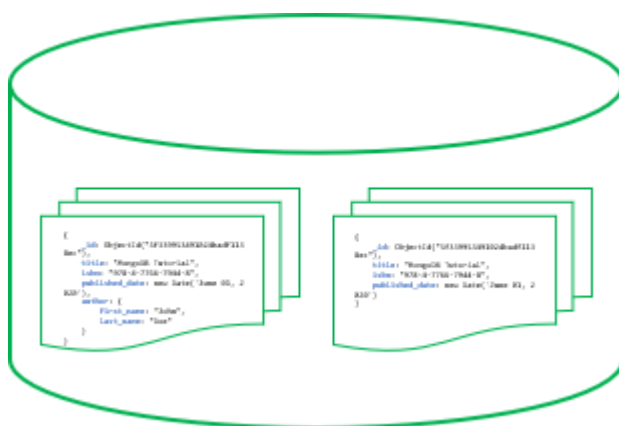
Note that the second document has one more field than the first one. In theory, you can have completely different fields for every document.

A collection has a name e.g., books. The collection name cannot:

- contain the dollar sign (\$)
- contain the null character (\0).
- be an empty string.
- begin with the system because MongoDB reserves system* for internal collection names.

Databases

MongoDB stores collections into a database. A single instance of MongoDB can host multiple databases.



A database can be referenced by a name for example bookdb. The database names cannot:

- Be an empty string ("").
- Contain any of these characters: /, \, ., ", *, <, >, :, /, ?, \$, (a single space), or \0 (the null character).
- Exceed the maximum size which is 64 bytes.

MongoDB also has some reserved database names such as admin, local, and config that you cannot use to create new databases.

Namespace

A namespace is a concatenation of the database name with a collection in that database. Namespaces allow you to fully qualify collections.

For example, if the collection name is books and database name is bookdb, the namespace of the books collection would be bookdb.books.

Summary

- MongoDB stores data records as BSON documents. A document is a set of field-and-value pairs.
- MongoDB stores documents in a collection and collections in a database.
- A namespace is a concatenation of the database name and the collection name (database_name.collection_name) to fully qualify the collection.

Null

The null type is used to represent a null and a field that does not exist. For example:

```
{  
  "isbn": null  
}
```

Code language: JSON / JSON with Comments (json)

Boolean

The boolean type has two values true and false. For example:

```
{  
  "best_seller": true  
}
```


Code language: JSON / JSON with Comments (json)

Number

By default, the mongo shell uses the 64-bit floating-point numbers. For example:

```
{
  "price": 9.95,
  "pages": 851
}
```

Code language: JSON / JSON with Comments (json)

The NumberInt and NumberLong classes represent 4-byte and 8-byte integers respectively. For example:

```
{
  "year": NumberInt("2020"),
  "words": NumberLong("95403")
}
```

Code language: JSON / JSON with Comments (json)

String

The string type represents any string of UTF-8 characters. For example:

```
{
  "title": "MongoDB Data Types"
}
```

Code language: JSON / JSON with Comments (json)

Date

The date type stores dates as 64-bit integers that represents milliseconds since the Unix epoch (January 1, 1970). It does not store the time zone. For example:

```
{
  "updated_at": new Date()
}
```

Code language: JSON / JSON with Comments (json)

In JavaScript, the Date class is used to represent the date type in MongoDB.

Note that you should always call the new Date(), not just Date() when you create a new Date object because the Date() returns a string representation of the date, not the Date object.

The mongo shell displays dates using local time zone settings. However, MongoDB does not store date with the time zone. To store the time zone, you can use another key e.g., timezone.

Regular Expression

MongoDB allows you to store [JavaScript regular expressions](#). For example:

```
{
  "pattern": /\d+/
}
```

Code language: JSON / JSON with Comments (json)

In this example, /\d+/ is a regular expression that matches one or more digits.

Array

The array type allows you to store a list of values of any type. The values do not have to be in the same type, for example:

```
{
  "title": "MongoDB Array",
  "reviews": ["John", 3.5, "Jane", 5]
}
```

Code language: JSON / JSON with Comments (json)

The good thing about arrays in the document is that MongoDB understands their structures and allows you to carry operations on their elements.

For example, you can query all documents where 5 is an element of the reviews array. Also, you can create an index on the reviews array to improve the query performance.

Embedded Document

A value of a document can be another document that is often referred to as an embedded document.

The following example shows a book document that contains the author document as an embedded document:

```
{
```

```
"title": "MongoDB Tutorial",  
"pages": 945,  
"author": {  
  "first_name": "John",  
  "last_name": "Doe"  
}  
}
```

Code language: JSON / JSON with Comments (json)

In this example, the author document has its own key/value pairs including `first_name` and `last_name`.

Object ID

In MongoDB, every document has an `"_id"` key. The value of the `"_id"` key can be any type. However, it defaults to an `ObjectId`.

The value of the `"_id"` key must be unique within a collection so that MongoDB can identify every document in the collection.

The `ObjectId` class is the default type for `"_id"`. It is used to generate unique values globally across servers. Since MongoDB is designed to be distributed, it is important to ensure the identifiers are unique in the shared environment.

The `ObjectId` uses 12 bytes for storage, where each byte represents 2 hexadecimal digits. In total, an `ObjectId` is 24 hexadecimal digits.

The 12-byte `ObjectId` value consists of:

- A 4-byte timestamp value that represents the `ObjectId`'s generated time measured in seconds since the Unix epoch.
- A 5-byte random value.
- A 3-byte increment counter, initialized to a random value.

These first 9 bytes of an `ObjectId` guarantee its uniqueness across servers and processes for a single second. The last 3 bytes guarantee uniqueness within a second in a single process.

As a result, these 12-bytes allow for up to 256^3 (16,777,216) unique `ObjectId`s values to be generated per process in a single second.

When you insert a document without specifying a value for the `"_id"` key, MongoDB automatically generates a unique id for the document. For example:

```
db.books.insertOne({  
  "title": "MongoDB Basics"  
});
```

Code language: JavaScript (javascript)

Output:

```
{  
  "acknowledged" : true,  
  "insertedId" : ObjectId("5f2fcae09b58c38603442a4f")  
}
```

Code language: JSON / JSON with Comments (json)

MongoDB generated the id with the value `ObjectId("5f2fcae09b58c38603442a4f")`. You can view the inserted document like this:

```
db.books.find().pretty()
```

Code language: CSS (css)

Output:

```
{  
  "_id" : ObjectId("5f2fcae09b58c38603442a4f"),  
  "title" : "MongoDB Basics"  
}
```

Code language: JSON / JSON with Comments (json)

In this tutorial, you have learned the most commonly used MongoDB data types including null, number, string, array, regular expression, date, and ObjectId.