



AGENDA

01

Data base

02

What is SQL?

03

What is NoSQL?

04

Types of NoSQL

05

What is MongoDB

06

Features of MongoDB

07

MongoDB Data types

08

Installation

09

Commands

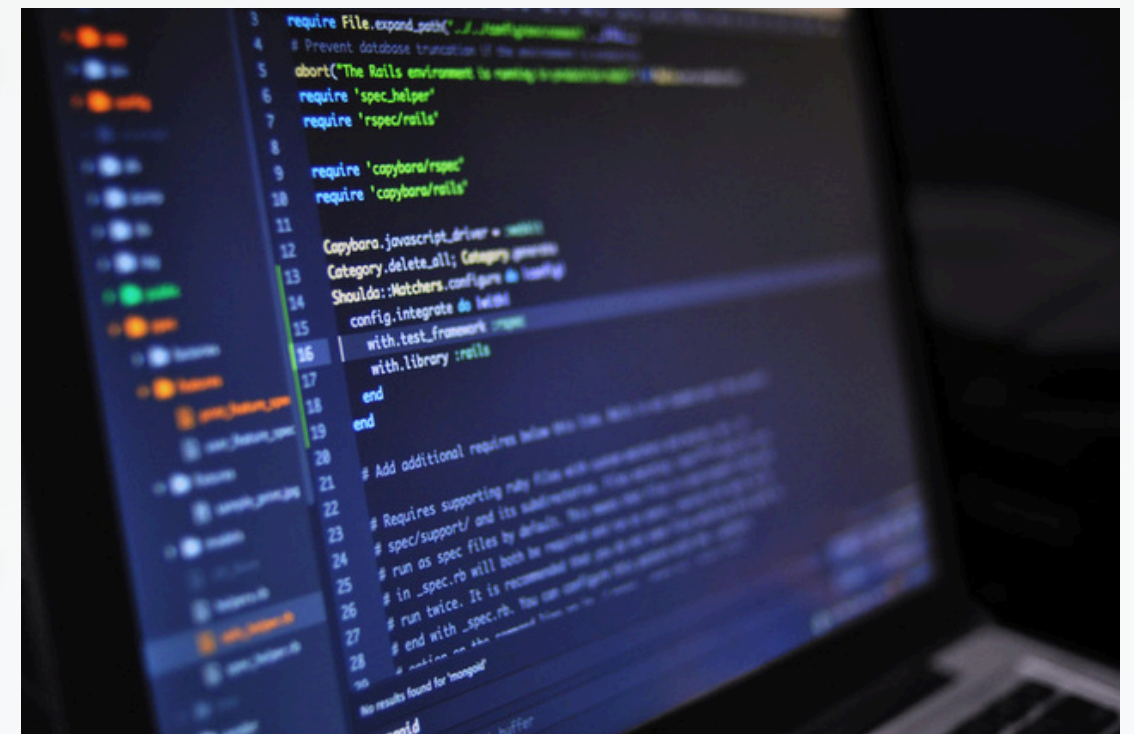
10

MongoDB compass & MongoDB shell

DATA BASE

A database is an organized collection of structured or unstructured information stored electronically on machine locally or in a cloud. These are managed using Database Management System (DBMS).

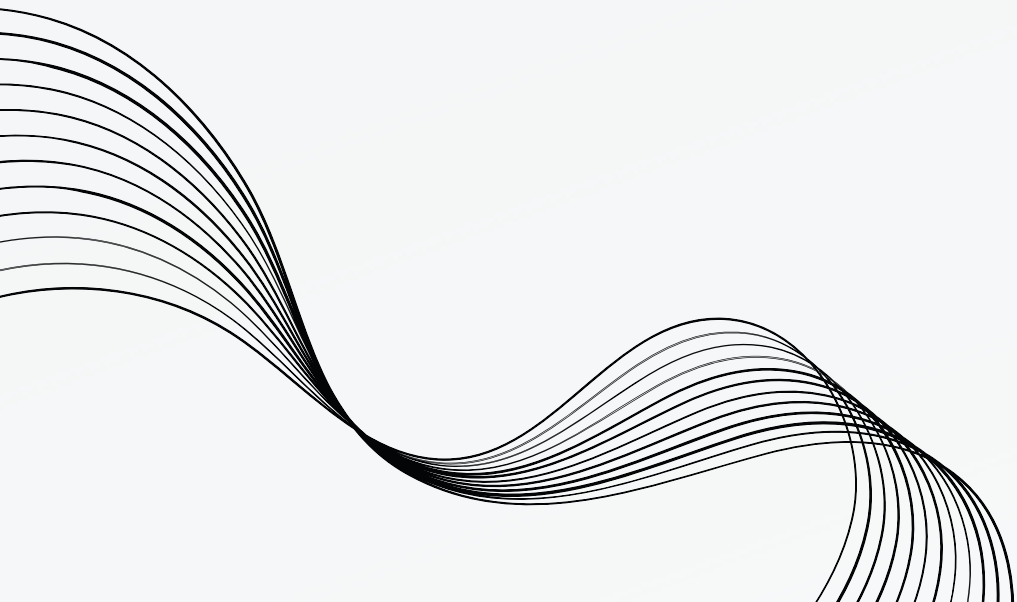
Databases are used for various data processing operations, the most basic being the Create, Update, Read, and Delete (CURD Operation).



```
1 require File.expand_path("../support/../../spec_helper", __FILE__)
2 # Prevent database truncation if the environment is production
3 abort("The Rails environment is running in production")
4 require 'spec_helper'
5 require 'rspec/rails'
6
7 require 'capybara/rspec'
8 require 'capybara/rails'
9
10 Capybara.javascript_driver = :webkit
11 Category.delete_all; Category.create
12 Shoulda::Matchers.configure do |config|
13   config.integrate do |with|
14     with.test_framework :rspec
15     with.library :rails
16   end
17 end
18
19 # Add additional requires below this line. See the documentation for more
20 #
21 # Requires supporting ruby files with spec_helper, this file, and the
22 # spec/support/ and its subdirectories. This file is required by default.
23 #
24 # run as spec files by default. This file is required by default.
25 # in _spec.rb will both be required. This file is required by default.
26 # run twice. It is recommended that you configure the application
27 # end with _spec.rb. You can configure the application
28 # end with _spec.rb. You can configure the application
```

SQL Database

SQL stands for Structured Query Language. SQL lets you access and manipulate database. It can execute queries against a database. It can retrieve data ,insert ,update ,delete records in a database.



NoSQL Database

- A NoSQL database has a dynamic schema for unstructured data. Data is stored in many ways which means it can be document-oriented, column-oriented, graph-based, or organized as a key value store. This flexibility means that documents can be created without having a defined structure first. Also, each document can have its own unique structure. The syntax varies from database to database, and you can add fields as you go.



Difference between SQL and NoSQL

SQL vs NoSQL

SQL	NoSQL
Data uses Schema	Schema-less (Schema Agnostic)
Maintain Relationship	No relations– though you can design relationship
Data distributed in multiple tables	Data in one table (embedded)
Monolithic, you can easily Scale-Up. Scale out is also possible but difficult (e.g. Azure Elastic Database tools)	Scale up and scale out- Globally distributed

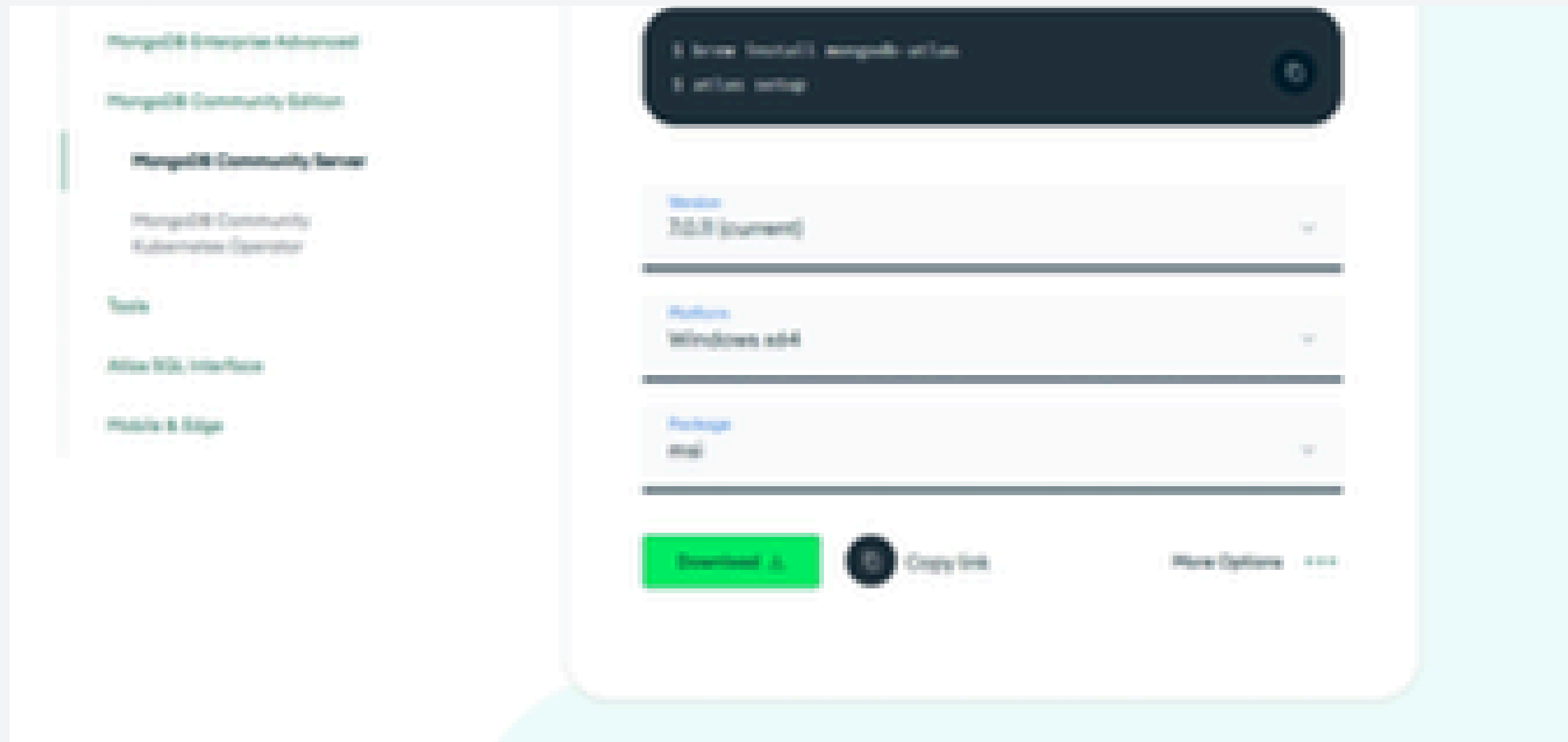
What is MongoDB

MongoDB community server: The source-available, free-to-use, and self-managed version of MongoDB.

>MongoDB shell : The MongoDB shell, mongosh is a Javascript and Node.js REPL environment for interacting with MongoDB development in Atlas

Install MongoDB Compass

**The very first step is to install the MongoDB compass into your system.
Go to the official site of MongoDB compass and click the download now button**



Installing MongoDB Shell (mongosh)

There are many ways to connect to your MongoDB database. We will start by using the MongoDB Shell, mongosh.

Use the official instructions to install mongosh on your operating system.

To verify that it has been installed properly, open your terminal and type: `mongosh --version` You should see that the latest version is installed.

The version used in this tutorial is v1.3.1.

MongoDB Atlas

MongoDB Enterprise Advanced

MongoDB Community Edition

Tools

MongoDB Shell

MongoDB Compass (GUI)

Atlas CLI

Atlas Kubernetes Operator

MongoDB CLI for Cloud Manager and Ops Manager

MongoDB Cluster-to-Cluster Sync

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

CREATE DATABASE USING MONGOSH

After connecting to your database using mongosh, you can see which database you are using by typing db in your terminal.

If you have used the connection string provided from the MongoDB Atlas dashboard, you should be connected to the myFirstDatabase database.

SHOW ALL DATABASES

To see all available databases, in your terminal type show dbs.

Notice that myFirstDatabase is not listed. This is because the database is empty. An empty database is essentially non-existent.



MongoDB Data types

- **String** - This is the most commonly used datatype to store the data. String in MongoDB must be UTF-8 valid.
- **Integer** - This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.
- **Boolean** - This type is used to store a boolean (true/ false) value.
- **Double** - This type is used to store floating point values.
- **Min/ Max keys** - This type is used to compare a value against the lowest and highest BSON elements.
- **Arrays** - This type is used to store arrays or list or multiple values into one key.
- **Object** - This datatype is used for embedded documents.
- **Null** - This type is used to store a Null value.
- **Symbol** - This datatype is used identically to a string; however, it's generally reserved for languages that use a specific symbol type.
- **Object ID** - This datatype is used to store the document's ID.
- **Binary data** - This datatype is used to store binary data.
- **Regular expression** - This datatype is used to store regular expression



MongoDB Commands

```
mongo
```

Open a terminal and start the MongoDB shell by typing mongo.

Create and Use a Database

```
use blog
```

Create (if not exists) and use the 'blog' database

CREATE COLLECTIONS

```
// Create a 'posts' collection  
db.createCollection("posts")
```

```
// Create a 'users' collection  
db.createCollection("users")
```

Create two collections: posts for storing blog posts and users for storing user information.

Insert Operations

Insert a single document into 'posts' collection

```
db.posts.insertOne({  
  title: "Introduction to MongoDB",  
  content: "MongoDB is a NoSQL database.",  
  author: "John Doe",  
  tags: ["mongodb", "nosql", "database"]  
})
```

Insert multiple documents into 'users' collection

```
db.users.insertMany([
  {
    username: "johndoe",
    email: "johndoe@example.com",
    age: 30
  },
  {
    username: "janedoe",
    email: "janedoe@example.com",
    age: 28
  }
])
```

Update Operations

Update a document in 'users' collection

```
db.users.updateOne(
  { username: "johndoe" },
  { $set: { age: 31 } }
)
```

```
db.posts.updateMany(  
  { tags: "mongodb" },  
  { $addToSet: { tags: "database" } }  
)
```

Delete Operations

Delete a document from 'users' collection

```
db.users.deleteOne({ username: "janedoe" })
```

Delete multiple documents from 'posts' collection

Thank
you!