Node.js MongoDB Tutorial

simplilearn

MERN Stack

simplilearn

What's in it for you?

What is MongoDB?

Why connect Node.js with MongoDB?

MERN Stack

simplilearn

MERN Stack

simplilearn

MERN Stack

- MERN Stack is a JavaScript Stack that is used for building modern single-page applications.
- MERN Stack comprises of 4 technologies namely: MongoDB, Express, React and Node.js. It is designed to make the development process smoother and easier.

MERN Stack

The phrase "MERN stack" refers to the following technologies:

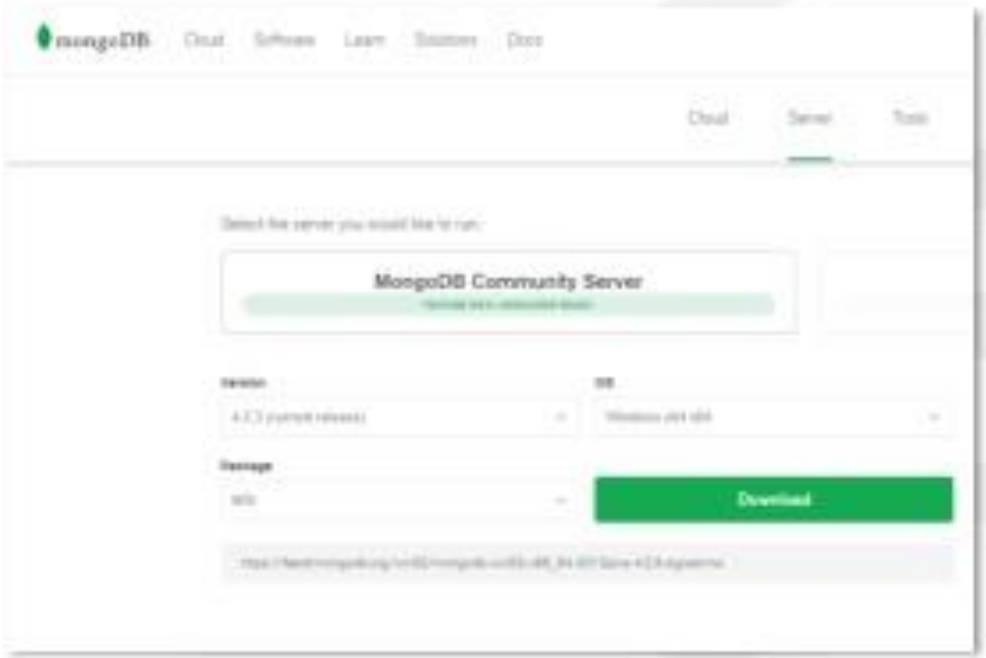
- **MongoDB**: MongoDB is a cross-platform document-oriented database program
- **Express.js**: Express.js, or simply Express, is a web application framework for Node.js
- **React**: React is a JavaScript library for building user interfaces.
- **Node.js**: Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser

PREREQUISITE

1. Database and its Structure
2. Concept of Queries
3. Schema

What is MongoDB?

- MongoDB is a cross-platform, document-oriented database that provides, high performance, high availability, and easy scalability
- MongoDB works on concept of collection and document
- It is a NoSQL database and is written in C++
- To be able to use MongoDB, download the free MongoDB database from the official website



INTRODUCTION

- MongoDB is an open source database that uses a document-oriented data model.

- MongoDB is one of several database types to arise in the mid-2000s under the NoSQLbanner. Instead of using tables and rows as in relational databases, MongoDB is built on an architecture of collections and documents.

CONTINUED....

- Documents comprise sets of key-value pairs and are the basic unit of data in MongoDB.

- Collections contain sets of documents and function as the equivalent of relational database tables.

ABOUT MongoDB

- Developed by 10gen
  - Founded in 2007
- Written in C++

FEATURES OF MongoDB ...

1

2

3

SCHEMA FREE

vs

NoSQL

RDBMS		MongoDB
Database	➡	Database
Table, View	➡	Collection
Row	➡	Document (BSON)
Column	➡	Field
Index	➡	Index
Join	➡	Embedded Document
Foreign Key	➡	Reference
Partition	➡	Shard

Collection is not strict about what it Stores

Schema-less

Hierarchy is evident in the design

Embed ded Docum ent ?

DOCUMENT

MongoDB is a **document**-oriented database. Instead of storing your data in tables **made** out of individual rows, like a relational database does, it stores your data in collections **made** out of individual **documents**.

In **MongoDB**, a **document** is a big JSON blob with no particular format or schema.N

A document in MongoDB is like a JSON.
Example:

```
{'name': 'Christiano',
  'language': 'Python',
  'country': 'Brazil'}
```

You can dynamically update your data

, example:

```
{'name': 'Christiano',
  'language': 'Python',
  'country': 'Brazil'}

{'name': 'Christiano',
  'language': 'Python',
  'country': 'Brazil',
  'event': 'PyConAr'}
```

Databases

In MongoDB, databases hold collections of documents.

use <db> **statement,**

use myDB

Creation of Database

- If a database does not exist, MongoDB creates the database when you first store data for that database.

SCHEMA FREE

- MongoDB does not need any pre-defined data schema
- Every document in a collection could have different data
  - Addresses NULL data fields

JSON FORMAT

- Data is in name / value pairs
- A name/value pair consists of a field name followed by a colon, followed by a value:
  - Example: “name”: “R2-D2”
- Data is separated by commas
  - Example: “name”: “R2-D2”, race : “Droid”
- Curly braces hold objects
  - Example: {“name”: “R2-D2”, race : “Droid”, affiliation: “rebels”}
- An array is stored in brackets []
  - Example [{“name”: “R2-D2”, race : “Droid”, affiliation: “rebels”},
  - {“name”: “Yoda”, affiliation: “rebels”}]

INSERTION OF DOCUMENTS

Inserting Single Documents

```
db.inventory.insertOne
({ item: "canvas", qty: 100,tags: ["cotton"],
size:{h: 28, w: 35.5, unit: "cm" } } )
```

Inserting Multiple Documents

db.collection.insertMany():
can insert *multiple* documents into a collection.
Pass an array of documents to the method.

EXAMPLE : INSERT MULTIPLE DOCUMENTS

```
db.inventory.insertMany
([{ item: “Freeze“,qty: 25, tags: ["blank", "red"],size:{ h: 14, w: 21, unit: "cm" }},
  { item: “TV“,qty: 85, tags: ["gray"],size:{ h: 27.9, w: 35.5, unit : "cm" } },
  { item: “AC“,qty:25,tags: [“white”], "blue"],size:{ h: 19, w: 22.85, unit:"cm" } }
])
```

Query Operators

Name	Description
$eq	Matches value that are equal to a specified value
$gt, $gte	Matches values that are greater than (or equal to a specified value
$lt, $lte	Matches values less than or (equal to) a specified value
$ne	Matches values that are not equal to a specified value
$in	Matches any of the values specified in an array
$nin	Matches none of the values specified in an array
$or	Joins query clauses with a logical OR returns all
$and	Join query clauses with a loginal AND
$not	Inverts the effect of a query expression
$nor	Join query clauses with a logical NOR
$exists	Matches documents that have a specified field

MongoDB sample Commands:

Ques. Show all databases:

```
show dbs
```

Ques. Create a Database Named 'workshop':

```
use workshop
```

Ques. Creating a collection/table named 'student' in 'workshop' database:

```
db.createCollection("student")
```

Ques. Inserting a document/record in 'student':

```
db.student.insertOne({RollNo:1, Name:"Aashna", Marks:80})
```

Ques. Inserting multiple documents/records in 'student':

```
db.student.insertMany( [{RollNo:2, Name:"Abhinav", Marks:70},
{RollNo:3, Name:"Ayesha", Marks:85},
{RollNo:4, Name:"Mohit", Marks:72} ] )
```

Ques. Display all records/documents from 'student':

```
db.student.find()
```

Ques. Display details of students having Marks more than 80:

```
db.student.find( { Marks : { $gt : 80} } )
```

Lets Relate the MongoDB Queries with SQL

- db.users.insert({a:1,b:2})

- Insert into Users Values(1,2)

- db.users.find( )

- Select * from users;

- db.users.find({age:33})

- Select * from users where age=33

- db.users.find({'name':'Manasvi',Age:12})

- Select * From Users where name='Manasvi' and Age=12;

- db.users.find({‘a:{$gt:22}})

- Select * from users where a>22;

- db.users.sort({name:-1})

- Select * from users order by name desc;

WHAT IS JSON?

JSON is acronym of Java Object Notation which is a format for structuring data to transmit data between server and web application

Why connect Node.js with MongoDB?

simplilearn

Why connect Node.js with MongoDB?

- All the modern applications require big data, fast features development, flexible deployment
- The older database systems can have issues competing with modern scalability needs
- MongoDB was needed to address both challenges

Why connect Node.js with MongoDB?

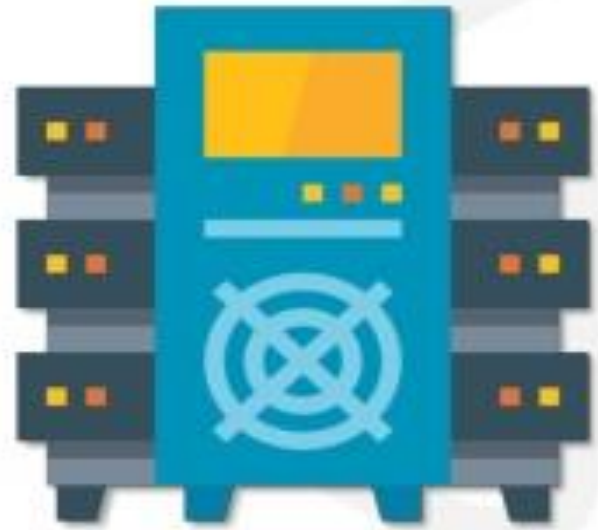
The primary purpose of building MongoDB is:

- Scalability
- Performance
- High Availability
- Scaling from single server deployments to large, complex multi-site architectures
- Easily deploy, operate, and scale the databases across the leading cloud platforms like Microsoft Azure, AWS, etc.

Why connect Node.js with MongoDB?

It is used in large implementation areas like:

- Big Data
- Content Management and Delivery
- Mobile and Social Infrastructure
- User Data Management
- Data Hub



Creating a MongoDB database in Node.js

To create a database in MongoDB:

- start by creating a MongoClient object
- then specify a connection URL with the correct ip address and the name of the database you want to create

```
var MongoClient = require('mongodb').MongoClient;
var url = "mongodb://localhost:27017/mydb";

MongoClient.connect(url, function(err, db) {
  if (err) throw err;
  console.log("Database created!");
  db.close();
});
```