**Unit-5**

**MongoDB Tutorial**

MongoDB is a No SQL database. It is an open-source, cross-platform, document-oriented database written in C++.

**What is MongoDB**

[MongoDB](https://www.javatpoint.com/mongodb-tutorial) is an open-source document database that provides high performance, high availability, and automatic scaling.

In simple words, you can say that - Mongo DB is a document-oriented database. It is an open source product, developed and supported by a company named 10gen.

MongoDB is available under General Public license for free, and it is also available under Commercial license from the manufacturer.

**The manufacturing company 10gen has defined MongoDB as:**

"MongoDB is a scalable, open source, high performance, document-oriented database." - 10gen

MongoDB was designed to work with commodity servers. Now it is used by the company of all sizes, across all industry.

**History of MongoDB**

The initial development of MongoDB began in 2007 when the company was building a platform as a service similar to window azure.

Window azure is a cloud computing platform and infrastructure, created by Microsoft, to build, deploy and manage applications and service through a global network.

MongoDB was developed by a NewYork based organization named 10gen which is now known as MongoDB Inc. It was initially developed as a PAAS (Platform as a Service). Later in 2009, it is introduced in the market as an open source database server that was maintained and supported by MongoDB Inc.

The first ready production of MongoDB has been considered from version 1.4 which was released in March 2010.

MongoDB2.4.9 was the stable version which was released on January 10, 2014.

**Purpose of building MongoDB**

All the modern applications require big data, fast features development, flexible deployment, and the older database systems not competent enough, so the MongoDB was needed.

**The primary purpose of building MongoDB is:**

* Scalability
* Performance
* High Availability
* Scaling from single server deployments to large, complex multi-site architectures.
* Develop Faster
* Deploy Easier
* Scale Bigger

**Why MongoDB is Document-Oriented?**

**MongoDB** is a document-oriented NoSQL database used for high volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB. Collections contain sets of documents and function which is the equivalent of relational database tables. MongoDB is a database which came into light around the mid-2000s.

**MongoDB Features**

1. Each database contains collections which in turn contains documents. Each document can be different with a varying number of fields. The size and content of each document can be different from each other.
2. The document structure is more in line with how developers construct their classes and objects in their respective programming languages. Developers will often say that their classes are not rows and columns but have a clear structure with key-value pairs.
3. The rows (or documents as called in MongoDB) doesn’t need to have a schema defined beforehand. Instead, the fields can be created on the fly.
4. The data model available within MongoDB allows you to represent hierarchical relationships, to store arrays, and other more complex structures more easily.
5. Scalability – The MongoDB environments are very scalable. Companies across the world have defined clusters with some of them running 100+ nodes with around millions of documents within the database

**Why Use MongoDB?**

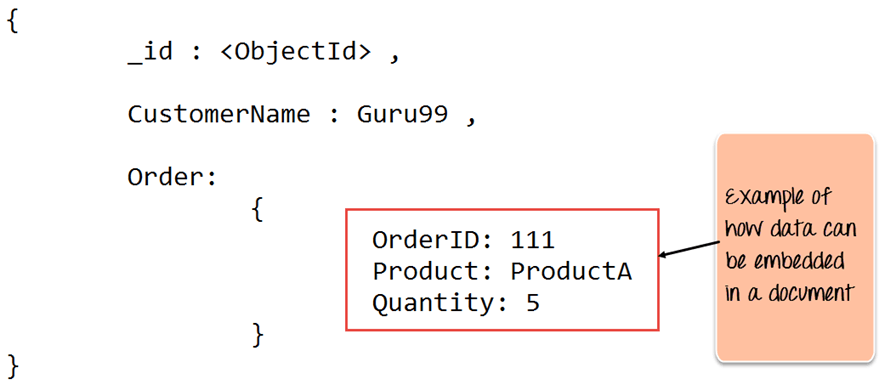
Below are the few of the reasons as to why one should start using MongoDB

1. **Document-oriented** – Since MongoDB is a NoSQL type database, instead of having data in a relational type format, it stores the data in documents. This makes MongoDB very flexible and adaptable to real business world situation and requirements.
2. **Ad hoc queries** – MongoDB supports searching by field, range queries, and regular expression searches. Queries can be made to return specific fields within documents.
3. **Indexing** – Indexes can be created to improve the performance of searches within MongoDB. Any field in a MongoDB document can be indexed.
4. **Replication** – MongoDB can provide high availability with replica sets. A replica set consists of two or more mongo DB instances. Each replica set member may act in the role of the primary or secondary replica at any time. The primary replica is the main server which interacts with the client and performs all the read/write operations. The Secondary replicas maintain a copy of the data of the primary using built-in replication. When a primary replica fails, the replica set automatically switches over to the secondary and then it becomes the primary server.
5. **Load balancing** – MongoDB can run over multiple servers, balancing the load and/or duplicating data to keep the system up and running in case of hardware failure.
6. **Auto-sharding:** The data is being distributed among several physical partitions known as shards. MongoDB has an in-built feature called automatic load balancing.
7. **Map Reduce:** It is a flexible aggregation tool that supports the Map Reduce function.
8. **Failure Handling:** In MongoDB, works effectively in case of failures such as multiple machine failures, data center failures by protecting data and making it available.
9. **GridFS:** This feature will allow the files to divide into smaller parts and store them in different documents without complicating the stack.
10. **Schema-less Database:** MongoDB is a schema-less database programmed in C++ language.
11. **Document-oriented Storage:** It uses BSON format which is similar to JSON
12. **Procedures:** MongoDB JavaScript works better than procedures as databases use the language more than procedures.
13. Provides high performance.

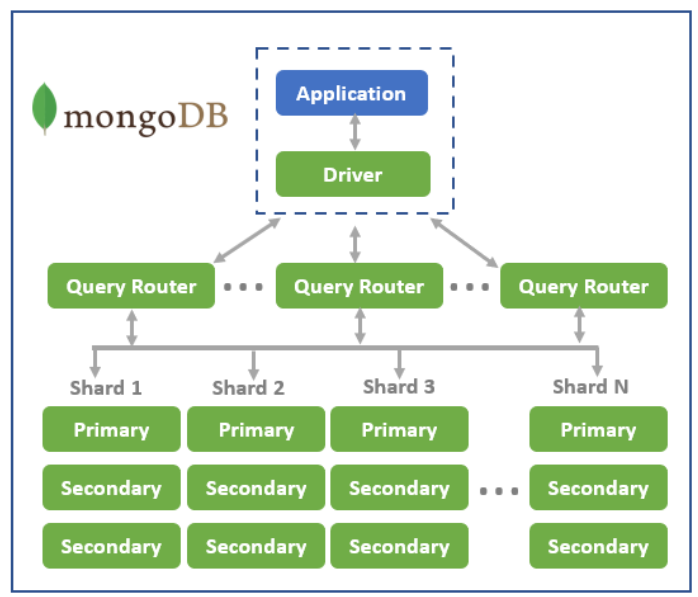
**MongoDB Example**

The below example shows how a document can be modeled in MongoDB.

1. The \_id field is added by MongoDB to uniquely identify the document in the collection.
2. What you can note is that the Order Data (OrderID, Product, and Quantity ) which in RDBMS will normally be stored in a separate table, while in MongoDB it is actually stored as an embedded document in the collection itself. This is one of the key differences in how data is modeled in MongoDB.



**MongoDB Architecture:**



**MongoDB Application:**

MongoDB is **an open source NoSQL database management program**. NoSQL is used as an alternative to traditional relational databases. NoSQL databases are quite useful for working with large sets of distributed data. MongoDB is a tool that can manage document-oriented information, store or retrieve information.

**MongoDB Driver:**

A database driver is **a computer program that implements a protocol (ODBC or JDBC) for a database connection**. The official MongoDB Node. js driver **allows Node.** **js applications to connect to MongoDB and work with data**. The driver features an asynchronous API which allows you to interact with MongoDB using traditional callbacks.

**Query Router:**

Query Router Sharding is transparent to applications; whether there is one or one hundred shards, the application code for querying MongoDB is the same. **Applications issue queries to a query router that dispatches the query to the appropriate shards.**

For key-value queries that are based on the shard key, the query router will dispatch the query to the shard that manages the document with the requested key. When using range-based sharding, queries that specify ranges on the shard key are only dispatched to shards that contain documents with values within the range.

For queries that don’t use the shard key, the query router will broadcast the query to all shards, aggregating and sorting the results as appropriate. Multiple query routers can be used with a MongoDB system, and the appropriate number is determined based on performance.

# Sharding

Sharding is the process of distributing data across multiple hosts. In MongoDB, sharding is achieved by splitting large data sets into small data sets across multiple MongoDB instances.

## How sharding works

When dealing with high throughput applications or very large databases, [the underlying hardware](https://www.bmc.com/blogs/it-infrastructure-management/) becomes the main limitation. High query rates can stress the CPU, RAM, and I/O capacity of disk drives resulting in a poor end-user experience.

To mitigate this problem, there are two types of scaling methods.

**Vertical scaling**

Vertical scaling is the traditional way of increasing the hardware capabilities of a single server. The process involves upgrading the CPU, RAM, and storage capacity. However, upgrading a single server is often challenged by technological limitations and cost constraints.

### Horizontal scaling

This method divides the dataset into multiple servers and distributes the database load among each server instance. Distributing the load reduces the strain on the required hardware resources and provides redundancy in case of a failure.

However, horizontal scaling increases the complexity of underlying architecture. MongoDB supports horizontal scaling through sharding.

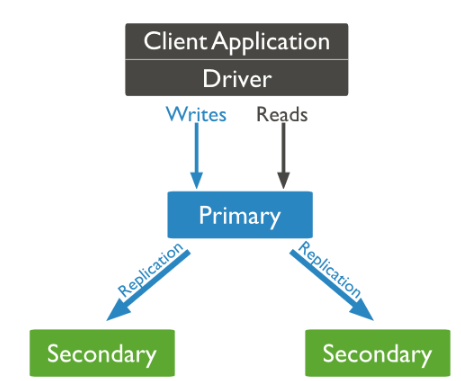
**Replication** as the name suggests is the replication or a copy of the same data onto multiple servers by Mongo DB. This is achieved by using a Replica Set. A replica set is basically a group of Mongo DB instances that maintain the same data set and pertain to any mongod process. When the primary node is down for some reason the other node becomes the primary node and thus making the operations smooth.

1. Replication Keeps our Data Safe.

2. Makes data available (24\*7)

3. Disaster recovery is made possible by Replication.

4. No downtime for Maintenance

.

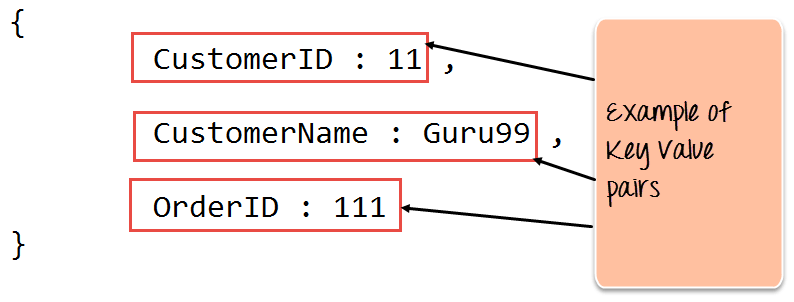
**Key Components of MongoDB Architecture**

Below are a few of the common terms used in MongoDB

1. **\_id** – This is a field required in every MongoDB document. The \_id field represents a unique value in the MongoDB document. The \_id field is like the document’s primary key. If you create a new document without an \_id field, MongoDB will automatically create the field. So for example, if we see the example of the above customer table, Mongo DB will add a 24 digit unique identifier to each document in the collection.

| **\_Id** | **CustomerID** | **CustomerName** | **OrderID** |
| --- | --- | --- | --- |
| 563479cc8a8a4246bd27d784 | 11 | Guru99 | 111 |
| 563479cc7a8a4246bd47d784 | 22 | Trevor Smith | 222 |
| 563479cc9a8a4246bd57d784 | 33 | Nicole | 333 |

1. **Collection** – This is a grouping of MongoDB documents. A collection is the equivalent of a table which is created in any other RDMS such as Oracle or MS SQL. A collection exists within a single database.
2. **Cursor** – This is a pointer to the result set of a query. Clients can iterate through a cursor to retrieve results.
3. **Database** – This is a container for collections like in RDMS wherein it is a container for tables. Each database gets its own set of files on the file system. A MongoDB server can store multiple databases.
4. **Document** – A record in a MongoDB collection is basically called a document. The document, in turn, will consist of field name and values.
5. **Field** – A name-value pair in a document. A document has zero or more fields. Fields are analogous to columns in relational databases.The following diagram shows an example of Fields with Key value pairs. So in the example below CustomerID and 11 is one of the key value pair’s defined in the document.



1. **JSON** – This is known as[JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html)Object Notation. This is a human-readable, plain text format for expressing structured data. JSON is currently supported in many programming languages.

Just a quick note on the key difference between the \_id field and a normal collection field. The \_id field is used to uniquely identify the documents in a collection and is automatically added by MongoDB when the collection is created.

**Data Modelling in MongoDB**

The data in MongoDB has a flexible schema. Unlike in[SQL](https://www.guru99.com/sql.html)databases, where you must have a table’s schema declared before inserting data, MongoDB’s collections do not enforce document structure. This sort of flexibility is what makes MongoDB so powerful.

When modeling data in Mongo, keep the following things in mind

1. What are the needs of the application – Look at the business needs of the application and see what data and the type of data needed for the application. Based on this, ensure that the structure of the document is decided accordingly.
2. What are data retrieval patterns – If you foresee a heavy query usage then consider the use of indexes in your data model to improve the efficiency of queries.
3. Are frequent inserts, updates and removals happening in the database? Reconsider the use of indexes or incorporate sharding if required in your data modeling design to improve the efficiency of your overall MongoDB environment.

**Differences between MongoDB & RDBMS**

Below are some of the key term differences between MongoDB and RDBMS

| **RDBMS** | **MongoDB** | **Difference** |
| --- | --- | --- |
| Table | Collection | In RDBMS, the table contains the columns and rows which are used to store the data whereas, in MongoDB, this same structure is known as a collection. The collection contains documents which in turn contains Fields, which in turn are key-value pairs. |
| Row | Document | In RDBMS, the row represents a single, implicitly structured data item in a table. In MongoDB, the data is stored in documents. |
| Column | Field | In RDBMS, the column denotes a set of data values. These in MongoDB are known as Fields. |
| Joins | Embedded documents | In RDBMS, data is sometimes spread across various tables and in order to show a complete view of all data, a join is sometimes formed across tables to get the data. In MongoDB, the data is normally stored in a single collection, but separated by using Embedded documents. So there is no concept of joins in MongoDB. |

Apart from the terms differences, a few other differences are shown below

1. Relational databases are known for enforcing data integrity. This is not an explicit requirement in MongoDB.
2. RDBMS requires that data be [normalized](https://www.guru99.com/database-normalization.html) first so that it can prevent orphan records and duplicates Normalizing data then has the requirement of more tables, which will then result in more table joins, thus requiring more keys and indexes. As databases start to grow, performance can start becoming an issue. Again this is not an explicit requirement in MongoDB. MongoDB is flexible and does not need the data to be normalized first.

**Conclusion:**

MongoDB is the most popular among the other NoSQL type of databases. All the big data applications make use of MongoDB as a default database. The future of MongoDB is bright and professionals possessing MongoDB skills are preferred in the IT market.

# Mongo DB - Operations

# MongoDB - Create Database

## The use Command

MongoDB **use DATABASE\_NAME** is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database.

### Syntax

Basic syntax of **use DATABASE** statement is as follows −

**use DATABASE\_NAME**

### Example

If you want to use a database with name **<mydb>**, then **use DATABASE** statement would be as follows −

**>use** mydb

switched to db mydb

To check your currently selected database, use the command **db**

**>db**

**mydb**

If you want to check your databases list, use the command **show dbs**.

**>show dbs**

**local 0.78125GB**

**test 0.23012GB**

Your created database (mydb) is not present in list. To display database, you need to insert at least one document into it.

**>**db**.**movie**.**insert**({"name":"tutorials point"})**

**>**show dbs

**local** **0.78125GB**

mydb **0.23012GB**

test **0.23012GB**

In MongoDB default database is test. If you didn't create any database, then collections will be stored in test database.

The following command **creates a record and inserts it into a newly created collection.** A **record** is a document with field names and values. As with a new database, if this record does not exist, MongoDB creates it:

db.Sample.insert({"SampleValue1" : 255, "SampleValue2" : "randomStringOfText"})

Let’s take a closer look at the syntax of the command to understand how the record and collection are created.

* **db.Sample.insert** – This is the **insert** command. In this example, the name **Sample** represents the name of the document you’re inserting data into. If the document does not already exist, MongoDB automatically creates it.
* **({  })** – The set of brackets encloses the data you are entering. Note that there’s a set of close-brackets at the end.
* **"SampleValue1" : 255, "SampleValue2" :  "randomStringOfText"** – Represents the format of the data we are entering into the record.

After you execute the command, you see the following response:

WriteResult({ "nInserted" : 1})

This response confirms that the system automatically **created a collection**and the **record was inserted**into said collection.

# MongoDB confirms that you have successfully inserted data to a new collection

**Query MongoDB with find Command**

You can verify that MongoDB saved the data from the insert command using the **find** command:

db.Sample.find()

As a result, the system displays the data located within the *Sample* collection.

find command prints information within the collection

# MongoDB - Drop Database

## The dropDatabase() Method:

MongoDB **db.dropDatabase()** command is used to drop a existing database.

### Syntax

Basic syntax of **dropDatabase()** command is as follows −

**db.dropDatabase()**

This will delete the selected database. If you have not selected any database, then it will delete default 'test' database.

### Example

First, check the list of available databases by using the command, **show dbs**.

**>**show dbs

**local** **0.78125GB**

mydb **0.23012GB**

test **0.23012GB**

If you want to delete new database **<mydb>**, then **dropDatabase()** command would be as follows −

**>use** mydb

switched to db mydb

**>**db**.**dropDatabase**()**

**>{** **"dropped"** **:** **"mydb",** **"ok"** **:** **1** **}**

**>**

Now check list of databases.

**>**show dbs

**local** **0.78125GB**

test **0.23012GB**

**>**

# MongoDB - Create Collection

## The createCollection() Method:

MongoDB **db.createCollection(name, options)** is used to create collection.

### Syntax

Basic syntax of **createCollection()** command is as follows −

**db.createCollection(name, options)**

In the command, **name** is name of collection to be created. **Options** is a document and is used to specify configuration of collection.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| Name | String | Name of the collection to be created |
| Options | Document | (Optional) Specify options about memory size and indexing |

Options parameter is optional, so you need to specify only the name of the collection. Following is the list of options you can use −

|  |  |  |
| --- | --- | --- |
| **Field** | **Type** | **Description** |
| capped | Boolean | (Optional) If true, enables a capped collection. Capped collection is a fixed size collection that automatically overwrites its oldest entries when it reaches its maximum size. **If you specify true, you need to specify size parameter also.** |
| autoIndexId | Boolean | (Optional) If true, automatically create index on \_id field.s Default value is false. |
| size | number | (Optional) Specifies a maximum size in bytes for a capped collection. **If capped is true, then you need to specify this field also.** |
| max | number | (Optional) Specifies the maximum number of documents allowed in the capped collection. |

While inserting the document, MongoDB first checks size field of capped collection, then it checks max field.

### Examples

Basic syntax of **createCollection()** method without options is as follows −

**>use** test

switched to db test

**>**db**.**createCollection**("mycollection")**

**{** **"ok"** **:** **1** **}**

You can check the created collection by using the command **show collections**.

**>**show collections

mycollection

system**.**indexes

The following example shows the syntax of **createCollection()** method with few important options −

**>** db**.**createCollection**("mycol",** **{** capped **:** **true,** autoIndexID **:** **true,** size **:** **6142800,** max **:** **10000** **}** **){**

**"ok"** **:** **0,**

**"errmsg"** **:** **"BSON field 'create.autoIndexID' is an unknown field.",**

**"code"** **:** **40415,**

**"codeName"** **:** **"Location40415"**

**}**

In MongoDB, you don't need to create collection. MongoDB creates collection automatically, when you insert some document.

**>**db**.**tutorialspoint**.**insert**({"name"** **:** **"tutorialspoint"}),**

**WriteResult({** **"nInserted"** **:** **1** **})**

**>**show collections

mycol

mycollection

system**.**indexes

tutorialspoint

# MongoDB - Drop Collection

## The drop() Method

MongoDB's **db.collection.drop()** is used to drop a collection from the database.

### Syntax

Basic syntax of **drop()** command is as follows −

db.COLLECTION\_NAME.drop()

### Example

First, check the available collections into your database **mydb**.

**>use** mydb

switched to db mydb

**>**show collections

mycol

mycollection

system**.**indexes

tutorialspoint

**>**

Now drop the collection with the name **mycollection**.

**>**db**.**mycollection**.**drop**()**

**true**

**>**

Again check the list of collections into database.

**>**show collections

mycol

system**.**indexes

tutorialspoint

**>**

drop() method will return true, if the selected collection is dropped successfully, otherwise it will return false.

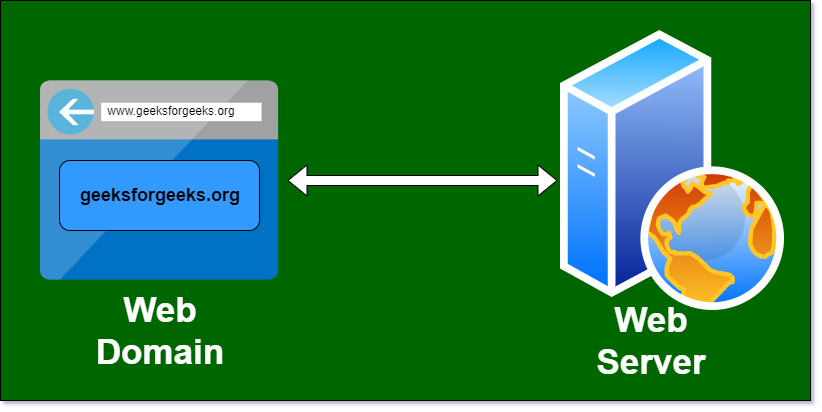
**Web Hosting Server and Web Domain**

When we want to host business online then the things that come in mind are:

* Which Domain name to buy?
* Which domain is good?
* What are its features?
* What is hosting?
* What is it’s use?
* And various other things?

Web-Domain or Domain name is the name used in a website’s URL to identify any particular Web page. For example: **www.geeksforgeeks.org**.

Web server is a program which processes the network requests of the users and serves them with files that create web pages. This exchange takes place using Hypertext Transfer Protocol (HTTP). In this virtual world, our website data is stored in the hosting server. Hosting Server is just like a computer that contains RAM, Processor, Hard Disk, Operating System etc that can be accessed remotely.



**Factors that Decide which Domain to Buy:**

* **Choose Simple Domain Name:** Choose a short domain name that is easy to spell, pronounce and memorable. Always choose a domain name related to your work and if domain name is already taken by other users then use synonyms word to register domain.
* **Avoid Hyphens and Numbers:** It is good practice to ignore hyphens and numbers from domain name.
* **Choose a best Domain Extension:** Domain extension matters for marketing and customers purpose. It is highly recommended to use “.com” domain extension because most of the user familiar with .com extension. You can also use “.net, .org, etc.
* **Customer Support:** A company with a good customer support will be at more ease to work.
* **Transfer Price:** If due to any reason you don’t like the domain company e.g. customer support is not good and if you wish to change your company then the extra amount paid to do that is transfer price.

**How Website Works with Domain and Hosting?**  
When we type URL in search box of browser then the browser send a request to its hosting provider and host send the response page to the browser and the website is downloaded by the browser and then it can be used further. So, the two things are uploading by the server and downloading by the browser.

**Types of Web Hosting Servers:**

* **Dedicated Server:** The Dedicated server means you are renting and using a complete server system. The cost of dedicated server is very high and the speed is also high. It is very secure because it is used by one company.
* **Virtual Private Server (VPS):** VPS share one physical server into multiple server and the shared server is acting like a separated server.
* **Shared Server:** The website is hosted on the server which is shared with other websites. The main advantage of shared hosting is its cheap price. The cost of shared hosting is very cheap in comparison of other hosting. The main drawback of shared hosting is its slow speed. The speed of website is very slow in shared hosting.
* **Cloud Based Web Hosting:** It is a hosting technology that lets hundreds of individual servers work together so that it looks like a big (giant) server. The main advantage of cloud-based web hosting is that its size can be modify according to server needs.

**Factors that Decide which Hosting to Choose:**

* **Port Speed:** It is another word for internet speed. Thus if the Internet Speed is high then the speed of uploading and downloading is also high.
* **Data Cap:** This is the transfer limit of the data provided by the server. After exceeding the limit charges are to be paid.
* **Operating System:** It has to be checked if there are some particular software’s used in website which are only available in selected operating system then opt for that operating system. Example: if you want to design a website in WordPress then LINUX OS works good and is also free.
* **RAM & Processing Power:** The requirements of RAM and processing power in a website can be done with the help of server. A shared server can handle about 1000 users at a time above it VPS server will be required and after some limit dedicated server will work.
* **Input Output operations:** In RAM, there is an input output operations so it has to be checked that at a time how many input output operations can the server handle. The more operations the server can handle the better.
* **Area of the market:** If the website is most likely to be used in India, it is preferred to buy a hosting of India.
* MongoDB is one of the most well-known and used databases out there.
* It comprises *Express*, *Angular*, and *Node*. It has mainly risen in popularity due to developers seeking a NoSQL database for data analytics and real-time web applications.
* Because it stores data in documents rather than relational tables.
* NoSQL databases come in several types, including pure document databases, key-value stores, wide-column databases, and graph databases.
* In general, we can divide hosting with MongoDB solutions into two broad categories, a self-managed MongoDB and a **Fully managed MongoDB hosting solutions**.
* Essentially, it all comes down to your specific needs and plans of use.
* The things to consider at this point are the price and features of the provider. However, you also need to assess how much time and knowledge you have to commit to the best MongoDB hosting provider.
* If you get into self-managed MongoDB, you’ll need some technical knowledge to back you up. In turn, if you get the managed tier, the hosting provider will do just about everything for you. And what do we mean by everything?
* The hosting provider will do the **initial setup**, as well as **the day-to-day maintenance**.
* In other words, you’ll always have someone watching over your shoulder.
* The support team will guide you if you run into an issue or just get stuck throughout your development process.
* Due to all this, you need to know what you are getting yourself into. To simplify, you need to estimate how much money you have to spend on hosting, and this will ultimately give you your clear-cut answer.

|  |  |
| --- | --- |
| **Self-Managed MongoDB Server** | **Fully Managed MongoDB Server** |
| You only get access to the raw hardware | MongoDB is pre-installed and pre-configured for you, so all you have to do is jump in there and develop |
| You will have to install MongoDB as well as any other app manually | If your server ever runs into any issues, you will have a support staff to assist you |
| It requires a lot of time and knowledge from your end, as you will receive no third party assistance | The web hosting provider team that manages your server can resolve the problem you are experiencing, freeing your time |
| You will have to monitor the server manually and take action if it encounters any downtime | You have consistent monitoring over your server, so it never ends up experiencing any downtime |

**Note**: Organizations like IBM, Twitter, Forbes, Facebook, Google, and many others use MongoDB as a backend data store.

**What Matters When Opting For A MongoDB Web Hosting Platform?**

* You might assume the price is the decisive factor when choosing a hosting provider.
* On the contrary, it should be the least of your concerns. Well, at least if you value quality and look for a superior provider.

**Performance**

* A MongoDB hosting platform should provide good performance.
* The speed factors such as *upload speed* and *download speed* rates.
* You can also use different insights or critical database-level metrics to assess the efficiency of your database, such as connections, cache hit ratio, CPU, load average, memory usage, and disk usage.
* Not all MongoDB hosting platforms provide for such analysis.

**Support**

* Most people choose to purchase hosting platforms as they lack the skills to work with MongoDB.
* If you encounter any issues with your site or database, the support team should be available 24/7/365 to assist you in your quest.
* Better yet, quality hosting provider teams don’t just wait around for you to ask for help. They troubleshoot and resolve issues on their own, even before they get to you.

**Backup Options and Automation**

* Each hosting company has its backup techniques and procedures. Some offer **unlimited storage**, whereas others limit your potential and even charge extra for these services.
* You should also consider automation. The question is, does your hosting provider allow you to schedule upgrades and backups, or should you take on this responsibility?
* If backups are done manually, it takes a lot of time. Also, you might easily forget to do so and lose valuable information in the process.

**Cloud Options:**

* Managed databases are commonly associated with the cloud.
* The feature allows for **easy access from different locations** and information sharing between the team.
* Some hosting companies allow you to bring your cloud plans, further saving on costs—however, others don’t.

**Security Features:** Your database might constantly be exposed to third parties on the Internet, especially on the cloud.

* So, it’s essential to find a hosting provider with **enhanced security features**, such as end-to-end encryption.
* You should be able to protect your data at the individual, divisional, or company-wide levels.

Here’s a list of the most popular hosting security features:

* + *DDoS(Distributed denial of Service) Protection*
  + *Virus Protection*
  + *Security Protection*
  + *SSL(Secure Socket Layer) Security Certificate*
  + *Domain Name Privacy*
  + *Firewalls*
* *Spam Filter*

When your amount of data grows and you don’t take any action, you will **harm your performance**. Thus, being able to scale in or out resources based on the workload dynamically is one of the core strengths of cloud architecture. Optimizing the resource usage eventually results in cutting the cloud expenditure.

**Deployment using Cloud Platforms:**

* AWS(Amazon Web Services) is a flexible, cost-effective, easy-to-use cloud computing platform.
* MongoDB is a popular NoSQL database that is widely deployed in the AWS cloud.
* Running your own MongoDB deployment on Amazon Elastic Cloud Compute (Amazon EC2) is a great solution for users whose applications require high-performance operations on large datasets.

**NoSQL on AWS**

* AWS provides an excellent platform for running many advanced data systems in the cloud.
* Some of the unique characteristics of the AWS cloud provide strong benefits for running NoSQL systems.
* AWS provides the following services for NoSQL and storage that do not require direct administration, and offers usage-based pricing.

Consider these options as possible alternatives to building your own system with open source software (OSS) or a commercial NoSQL product.

* With **Amazon DynamoDB,** you can offload the administrative burden of operating and scaling a highly available distributed database cluster while paying a low variable price for only the resources you consume.
* Amazon **Simple Storage Service (Amazon S3)** can store and retrieve any amount of data anytime from anywhere on the web. Amazon S3 gives developers access to the same highly scalable, reliable, secure, fast, and inexpensive infrastructure that Amazon uses to run its own global network of websites.
* MongoDB is a popular NoSQL document database that provides rich features, fast time-to-market, global scalability, and high availability, and is inexpensive to operate.
* As your deployments grow in terms of data volume and throughput, MongoDB scales easily with no downtime, and without changing your application.

**Storage Patterns**

* **WiredTiger storage engine:** For many applications, WiredTiger's granular concurrency control and native compression will provide the best all-around performance and storage efficiency for the broadest range of applications.
* **MMAPv1 engine**, which was the default storage engine before the latest MongoDB 3.2 release.
* **Encrypted storage engine,** which protects highly sensitive data without the performance or management overhead of separate file system encryption.
* **In-Memory storage engine**: which delivers extreme performance coupled with real-time analytics for the most demanding, latency-sensitive applications.

There are two popular approaches to managing MongoDB deployments:

* Using **MongoDB Cloud Manager**, which is a cloud service, or **MongoDB Ops Manager**, which is management software you can deploy, to provide monitoring, backup, and automation of MongoDB instances.
* Using a combination of tools and services such as AWS Cloud Formation and Amazon Cloud Watch along with other automation and monitoring tools.

**Using Cloud Manager or Ops Manager**

* Cloud Manager is a cloud service for managing MongoDB.
* By using Cloud Manager, you can deploy, monitor, back up, and scale MongoDB through the Cloud Manager interface or via an API call.
* Cloud Manager communicates with your infrastructure through agents installed on each of your servers, and coordinates critical operations across the servers in your MongoDB deployment.
* Ops Manager is similar to Cloud Manager. Ops Manager on AWS is used to manage, monitor, and back up their MongoDB deployments.
* You can deploy, monitor, back up, and scale MongoDB via the Cloud Manager or Ops Manager user interface directly, or invoke the RESTful APIs from existing enterprise tools.

**Deployment**

* With Cloud Manager or Ops Manager, you can deploy MongoDB replica sets, sharded clusters, and standalone instances quickly.
* They work by communicating with an automation agent installed on each server.
* The automation agent contacts Cloud Manager or Ops Manager and gets instructions on the goal state of your MongoDB deployment.

**Monitoring**

* Cloud Manager and Ops Manager monitoring provides real-time reporting, visualization, and alerting on key database and hardware indicators.
* A lightweight Monitoring Agent runs within your MongoDB deployment and collects statistics from the nodes in your MongoDB deployment.
* The agent transmits database statistics back to Cloud Manager or Ops Manager to provide real-time reporting. You can set alerts on the indicators you choose.

**Backup**

* Cloud Manager and Ops Manager are the only solutions that offer point-in-time backups of replica sets and cluster-wide snapshots of sharded clusters.
* A lightweight Backup Agent runs within your infrastructure. The agent conducts an initial sync and then tails the operation log (oplog) to provide a continuous backup.
* If the MongoDB cluster experiences a failure, the most recent backup is only moments behind, minimizing exposure to data loss.

**Deployment – AWS MongoDB Quick Start**

* AWS Quick Start reference deployments help you deploy fully functional enterprise software on the AWS cloud.
* Customization options include the MongoDB version you want to deploy, the number of replicas you want to launch to ensure high availability (1-3 replicas), and the number of shards you want to use to improve throughput and performance (0-3 shards). The Quick Start also provides microsharding options and lets you customize storage types and sizes.
* The MongoDB Quick Start takes approximately 15 minutes to deploy.
* You pay only for the AWS compute and storage resources you use—there is no additional cost for running the Quick Start.

**Monitoring – Amazon CloudWatch**

* used to collect and track metrics, to collect and monitor log files, and to set alarms.
* can send an alarm by SMS or email when user-defined thresholds are reached on individual AWS services.
* For example, you can set an alarm to warn of excessive storage throughput.

**Backup – Amazon EBS Snapshots**

* You can back up the data on your EBS volumes to Amazon S3 by taking point-intime snapshots.
* Snapshots are incremental backups, which means that only the blocks on the device that have changed after your most recent snapshot are saved.
* When you delete a snapshot, only the data exclusive to that snapshot is removed.
* Active snapshots contain all the information needed to restore your data to a new EBS volume.
* Taking a consistent Amazon EBS snapshot also depends upon the capabilities of a specific operating system or file system.
* An example of a file system that can flush its data for a consistent backup is XFS(Extents File System).
* To get a correct snapshot of a running mongod process:
* You must have journaling enabled. Otherwise, there is no guarantee that the snapshot will be consistent or valid.
* The journal should reside on the same logical volume as the data files, or you must use the MongoDB db.fsyncLock() method to capture a valid snapshot of your data.
* To get a consistent snapshot of a sharded system, you must disable the balancer and capture a snapshot from every shard and config server at approximately the same moment in time.

**Amazon Virtual Private Cloud (Amazon VPC):** enables you to create an isolated portion of the AWS cloud and launch Amazon EC2 instances that have private addresses in the range of your choice (e.g., 10.0.0.0/16).

* You can define subnets within your Amazon VPC, grouping similar kinds of instances based on IP address range, and then set up routing and security to control the flow of traffic in and out of the instances and subnets.
* If a subnet's traffic is routed to an Internet gateway, the subnet is known as a public subnet; otherwise, it’s called a private subnet.
* **Deploying Across Availability Zones**: You can create an Amazon VPC that spans multiple Availability Zones, but each subnet must reside entirely within one Availability Zone and cannot span zones.
* To deploy MongoDB across separate Availability Zones, you should create one Amazon VPC with private subnets for each zone.

A common security group can be used to allow the instances across the subnet to communicate.

* **Deploying Across Regions**: For MongoDB deployments that span regions, you would need to create an Amazon VPC in each region and allow communication on ports listed via security groups.
* Security groups are defined within the scope of an Amazon VPC, so you would need to use multiple security groups to secure access within and across regions.
* In this deployment, you should also secure the network communications between regions by using one of the following methods:
* Secure IPsec tunnel to connect multiple Amazon VPCs into a larger virtual private network that allows instances in each Amazon VPC to seamlessly connect to each other using private IP addresses.
* SSL to encrypt MongoDB’s entire network traffic and allow instances to connect to each other using public IP addresses.

The AWS cloud provides a unique platform for running NoSQL applications, including MongoDB.

* With capacities that can meet dynamic needs, costs based on use, and easy integration with other AWS products such as Amazon CloudWatch, AWS CloudFormation, and Amazon EBS, the AWS cloud enables you to run a variety of NoSQL applications without having to manage the hardware yourself.
* MongoDB, in combination with AWS, provides a robust platform for developing scalable, high-performance applications.