[MongoDB-Let's learn!](http://sankethtsagar.blogspot.com/)

[MongoDB](http://sankethtsagar.blogspot.com/2016/12/mongodb.html)

MongoDB is

* Open source document database
* NoSQL database
* Highly scalable
* Schemaless

Data is stored in JSON format and internally supported by BSON format.

The **document** is the unit of storing data in a MongoDB database

A simple example of a JSON document is as follows :

A **collection** is the group of documents.

[](https://1.bp.blogspot.com/-fRRmhwM_qTg/WFjUkoRw06I/AAAAAAAAI88/QyfL1QC_ONsG_a_v0MXw-ql8r8FeimcVQCLcB/s1600/mongo.png)

MongoDB installation  on Ubuntu

**Step #1 Set up the package database**  
In this step, we will import the MongoDB GPG public key.

MongoDB is already included in Ubuntu package repositories, but the official MongoDB repository provides most up-to-date version and is the recommended way of installing the software. Ubuntu ensures the authenticity of software packages by verifying that they are signed with GPG keys, so we first have to import they key for the official MongoDB repository.

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10

Then we’ll create a list file for MongoDB:

echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' | sudo tee /etc/apt/sources.list.d/mongodb.list

Now reload the package database:

sudo apt-get update

**Step #2: Install Latest Stable Version MongoDB**

At this point, installing MongoDB is as simple as running just one command:

sudo apt-get install -y mongodb-org

**Step #3: Get MongoDB Running**

Start-Up MongoDB

sudo service mongod start

Enter the MongoDB Command Line

mongo

Shutdown MongoDB

sudo service mongod stop

Once the installation is done we can procede with the following  
 **Create Database**

*>use DB\_NAME*

Example: *use mydb*

To display the database you are using, type db  
*>db*

**Check the list of databases**

>*show dbs*

Note: To show your db in the list it should contain at-least one document.

          Test is the default database

**Dropping database**

Drops the current database

>*db.dropDatabase()*

**Creating a collection**

>*db.createCollection(name, option)*

"option" field can have 4 different values. They are

* capped
* autoIndexID
* size
* max

|  |
| --- |
| [https://1.bp.blogspot.com/-7YoqHmINHtA/WGtOBLjMBdI/AAAAAAAAI_E/hRJrJ5nPuxsjlj2ShSQjOidl-Ib65hAMwCLcB/s320/mongo1.png](https://1.bp.blogspot.com/-7YoqHmINHtA/WGtOBLjMBdI/AAAAAAAAI_E/hRJrJ5nPuxsjlj2ShSQjOidl-Ib65hAMwCLcB/s1600/mongo1.png) |
| creating database and collection |

**Check the list of collection**

*>show collections*

**Dropping a collection**

*>db.COLLECTION\_NAME.drop()*

·         **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.

·         **Timestamp** − ctimestamp. This can be handy for recording when a document has been            modified or added.

·         **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.

·         **Date**− This datatype is used to store the current date or time in UNIX time format. You           can specify your own date time by creating object of Date and passing day, month, year into     it.

·         **Object ID** − This datatype is used to store the document’s ID.

·         **Binary data** − This datatype is used to store binary data.

·         **Code** − This datatype is used to store JavaScript code into the document.

·         **Regular expression** − This datatype is used to store regular expression.

**Insert a document**

>*db.COLLECTION\_NAME.insert(document)*

Example: db.mycol.insert({

                    \_id: ObjectId(7df78ad89c),

                    title: 'MongoDB Overview',

                    description: 'NoSQL',})

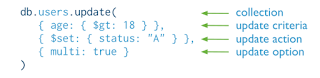
|  |
| --- |
| [https://2.bp.blogspot.com/-DYTQI6QtDWk/WGyJyL3Q6iI/AAAAAAAAI_c/NIEXZu0CZdAuqM5r4GvDwfTHG9VzAlVTwCLcB/s320/mongo3.png](https://2.bp.blogspot.com/-DYTQI6QtDWk/WGyJyL3Q6iI/AAAAAAAAI_c/NIEXZu0CZdAuqM5r4GvDwfTHG9VzAlVTwCLcB/s1600/mongo3.png) |
| Inserting a document |

If \_id fiels is not specified then MongoDB assigns an unique object id to the document.

|  |
| --- |
| [https://2.bp.blogspot.com/-5b5sJmhRTL0/WGyKQWqIKOI/AAAAAAAAI_g/tVgEEFBxhF4qnOtROrTWiLpWibl1-n_yACLcB/s320/mongo4.png](https://2.bp.blogspot.com/-5b5sJmhRTL0/WGyKQWqIKOI/AAAAAAAAI_g/tVgEEFBxhF4qnOtROrTWiLpWibl1-n_yACLcB/s1600/mongo4.png) |
| Inserting multiple documents |

**Update a document**

*>db.COLLECTION\_NAME.update(Selection\_Criteria, Updated\_data)*

[](https://1.bp.blogspot.com/-D__sQcO1FJU/WHc5tVqACTI/AAAAAAAAJB8/YjncjKjtE9ct9_LiaJnx2pgdkFHafMYqACLcB/s1600/sa.png)

**Delete document**

Here Remove() method is used to delete a document.  
  
>*db.COLLECTION\_NAME.remove(DELETION\_CRITERIA)*  
  
  
**Query Document**  
find() method is used to query data.

*>db.COLLECTION\_NAME.find()*

|  |
| --- |
| [https://3.bp.blogspot.com/-7HWYhCXSfTo/WGyLw0kk6lI/AAAAAAAAI_w/gSNKKcypdsAAJmpQO8w0j3j0f1q27x8RACEw/s400/mongo5.png](https://3.bp.blogspot.com/-7HWYhCXSfTo/WGyLw0kk6lI/AAAAAAAAI_w/gSNKKcypdsAAJmpQO8w0j3j0f1q27x8RACEw/s1600/mongo5.png) |
| using find() query to get all the documents with status "p" |

We can use pretty() method to get the result in the formatted way.

>*db.mycol.find().pretty()*

**Limit() method**

Accepts number as argument, and displays that much of documents.

*>db.COLLECTION\_NAME.find().limit(NUMBER)*  
If you don't specify the number argument in **limit()** method then it will display all documents from the collection.  
 **Bulk Write Operation**  
MongoDB provides clients the ability to perform write operations in bulk. Bulk write operations affect a *single*collection. MongoDB allows applications to determine the acceptable level of  acknowledgement required for bulk write operations.

The db.collection.bulkWrite() method provides the ability to perform bulk insert, update, and remove operations. MongoDB also supports bulk insert through the db.collection.insertMany()

bulkWrite() supports the following write operations:

* [insertOne](https://docs.mongodb.com/manual/reference/method/db.collection.bulkWrite/#bulkwrite-write-operations-insertone)
* [updateOne](https://docs.mongodb.com/manual/reference/method/db.collection.bulkWrite/#bulkwrite-write-operations-updateonemany)
* [updateMany](https://docs.mongodb.com/manual/reference/method/db.collection.bulkWrite/#bulkwrite-write-operations-updateonemany)
* [replaceOne](https://docs.mongodb.com/manual/reference/method/db.collection.bulkWrite/#bulkwrite-write-operations-replaceone)
* [deleteOne](https://docs.mongodb.com/manual/reference/method/db.collection.bulkWrite/#bulkwrite-write-operations-deleteonemany)
* [deleteMany](https://docs.mongodb.com/manual/reference/method/db.collection.bulkWrite/#bulkwrite-write-operations-deleteonemany)

**Skip() method**

It also accepts number as an argument which is used to skip document or documents.

*>db.COLLECTION\_NAME.find().limit(NUMBER).skip(NUMBER)*

**Sort() method**

It is used to sort the documents in ascending or in descending order. To specify sorting order 1 and -1 are used. 1 is used for ascending order while -1 is used for descending order.

*>db.COLLECTION\_NAME.find().sort({KEY:1})*

**Indexing**  
Indexes support the efficient resolution of queries. A scan without indexes is inefficient.To create an index you need to use ensureIndex() method of MongoDB.

*>db.COLLECTION\_NAME.ensureIndex({KEY:1})*

Here key is the name of the field on which you want to create index and 1 is for ascending order. To create index in descending order you need to use -1.  
  
**Aggregate() method**  
Aggregations operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. In SQL count(\*) and with group by is an equivalent of mongodb aggregation.

*>db.COLLECTION\_NAME.aggregate(AGGREGATE\_OPERATION)*

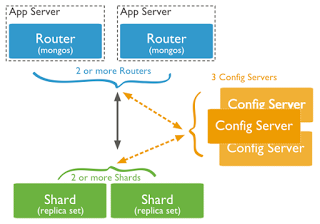
**Replication**  
  
Replication is the process of synchronizing data across multiple servers. Replication provides redundancy and increases data availability with multiple copies of data on different database servers. Replication protects a database from the loss of a single server. Replication also allows you to recover from hardware failure and service interruptions. **Working**

Replica set is a group of two or more nodes (generally minimum 3 nodes are required).  
In a replica set, one node is primary node and remaining nodes are secondary.  
All data replicates from primary to secondary node.  
At the time of automatic failover or maintenance, election establishes for primary and a new primary node is elected.  
After the recovery of failed node, it again join the replica set and works as a secondary node.

|  |
| --- |
| [https://4.bp.blogspot.com/-yy9EeBoF9q0/WGdOLG00ojI/AAAAAAAAI-U/L67prBnLtpoFyAOuQVAKDsVf7csiqfpWwCLcB/s320/1.png](https://4.bp.blogspot.com/-yy9EeBoF9q0/WGdOLG00ojI/AAAAAAAAI-U/L67prBnLtpoFyAOuQVAKDsVf7csiqfpWwCLcB/s1600/1.png) |
| Working of replica |

**Sharding**

Sharding is the process of storing data records across multiple machines and it is MongoDB's approach to meeting the demands of data growth. As the size of the data increases, a single machine may not be sufficient to store the data nor provide an acceptable read and write throughput. Sharding solves the problem with horizontal scaling. With sharding, you add more machines to support data growth and the demands of read and write operations.

[](https://3.bp.blogspot.com/-sSfmiHz9qm8/WGdPpL3tpdI/AAAAAAAAI-g/h8QJ_WlMCfA9FpruuYDHADOObkXmWP5ngCLcB/s1600/2.png)

In the following diagram, there are three main components −

* **Shards** − Shards are used to store data. They provide high availability and data consistency. In production environment, each shard is a separate replica set.
* **Config Servers** − Config servers store the cluster's metadata. This data contains a mapping of the cluster's data set to the shards. The query router uses this metadata to target operations to specific shards. In production environment, sharded clusters have exactly 3 config servers.
* **Query Routers** − Query routers are basically mongo instances, interface with client applications and direct operations to the appropriate shard. The query router processes and targets the operations to shards and then returns results to the clients. A sharded cluster can contain more than one query router to divide the client request load. A client sends requests to one query router. Generally, a sharded cluster have many query routers.