

MANGODB ENVIRONMENT SETUP

MongoDB Environment Setup

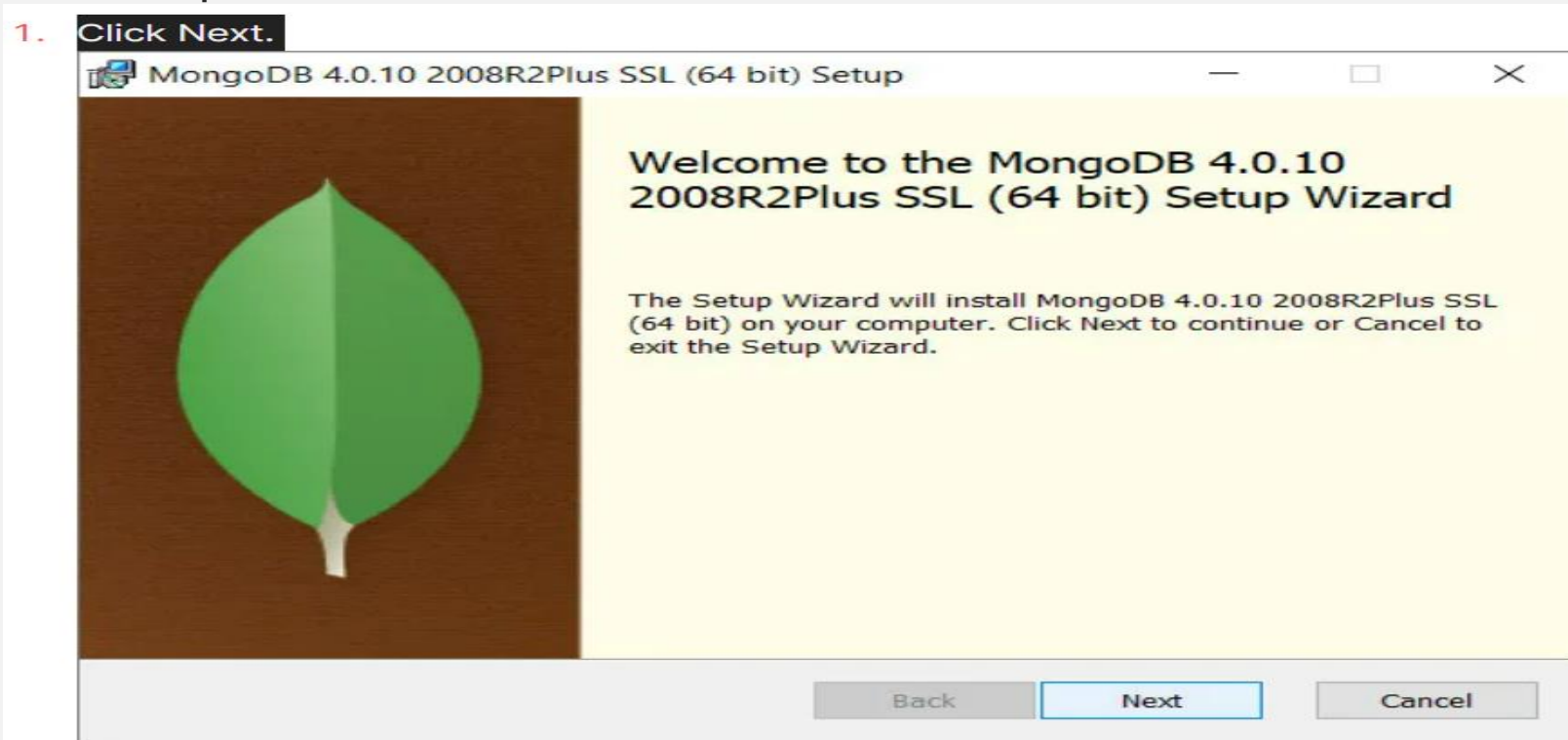
- To get started with MongoDB, you have to install it in your system. You need to find and download the latest version of MongoDB, which will be compatible with your computer system.
- You can use this (<http://www.mongodb.org/downloads>) link and follow the instruction to install MongoDB in your PC.

MongoDB Environment Setup

- The website of MongoDB provides all the installation instructions, and MongoDB is supported by Windows.
- It is to be noted that, MongoDB will not run in Windows XP; so you need to install higher versions of windows to use this database.

MongoDB Environment Setup

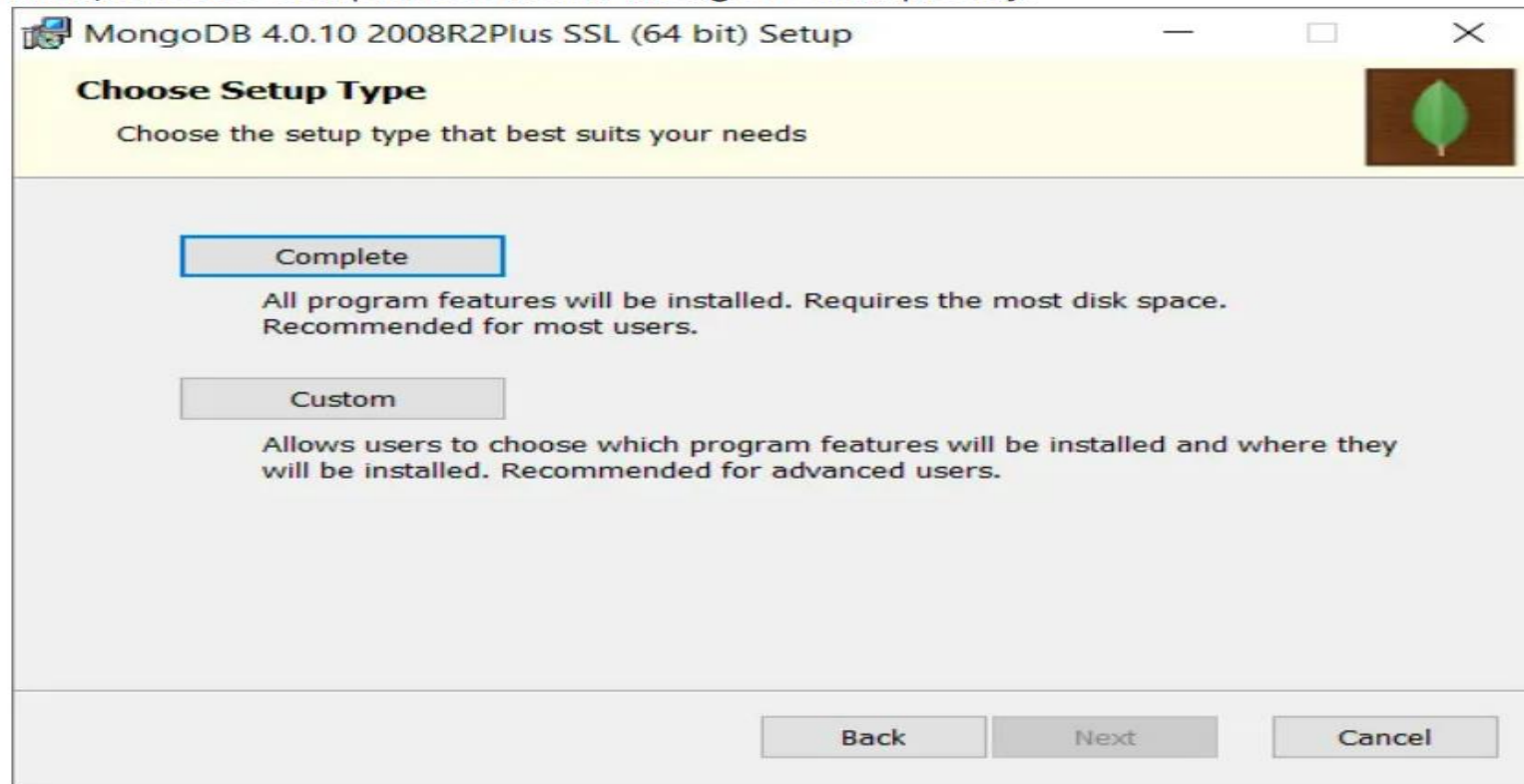
- Once the download is complete, double click this setup file to install it. Follow the steps:



MongoDB Environment Setup

- Follow the steps:

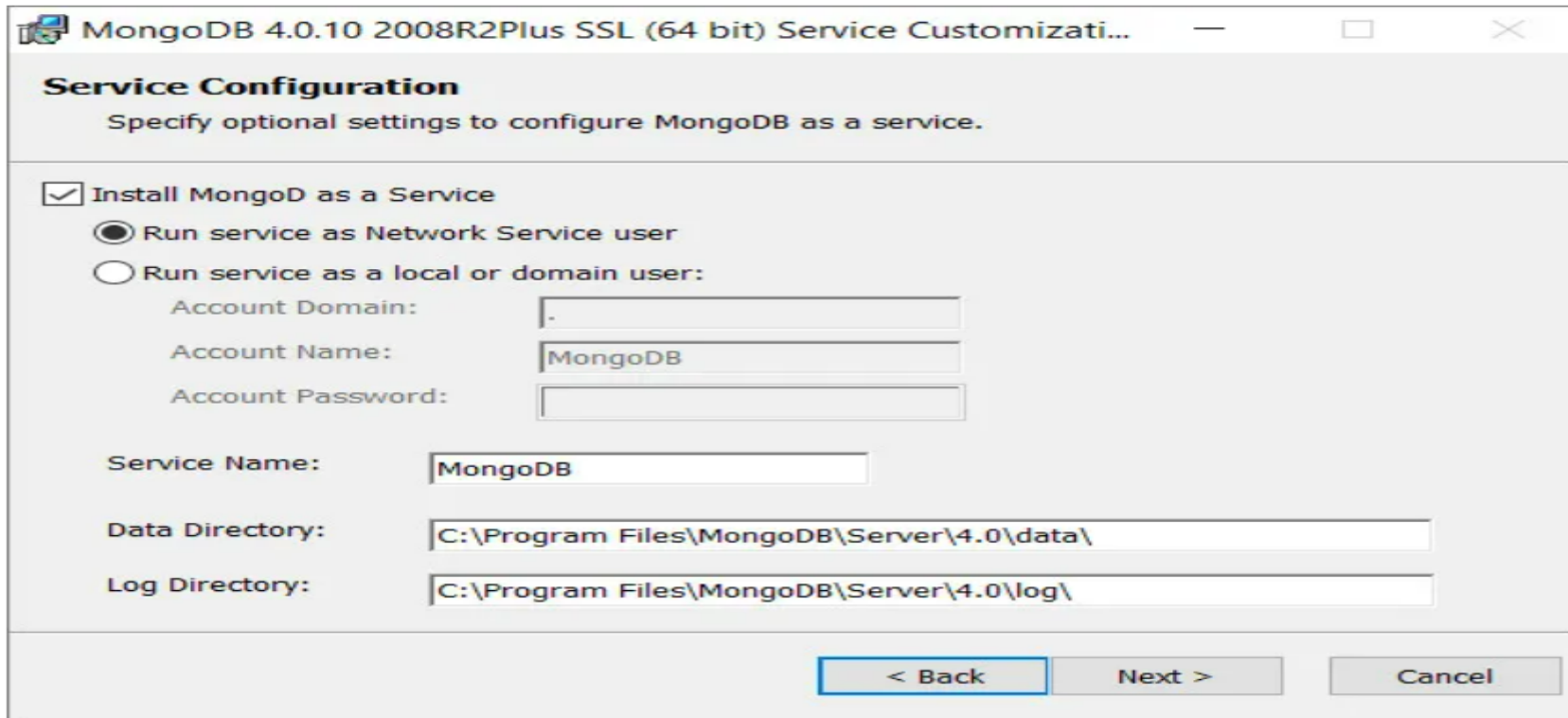
2. Now, choose Complete to install MongoDB completely.



MongoDB Environment Setup

- Follow the steps:

3. Then, select the radio button "Run services as Network service user."



The screenshot shows the "Service Configuration" window for MongoDB 4.0.10. The window title is "MongoDB 4.0.10 2008R2Plus SSL (64 bit) Service Customizati...". The main heading is "Service Configuration" with the subtitle "Specify optional settings to configure MongoDB as a service.".

The "Install MongoDB as a Service" checkbox is checked. Below it, the "Run service as Network Service user" radio button is selected. The "Run service as a local or domain user:" option is unselected, and its associated fields (Account Domain, Account Name, and Account Password) are empty.

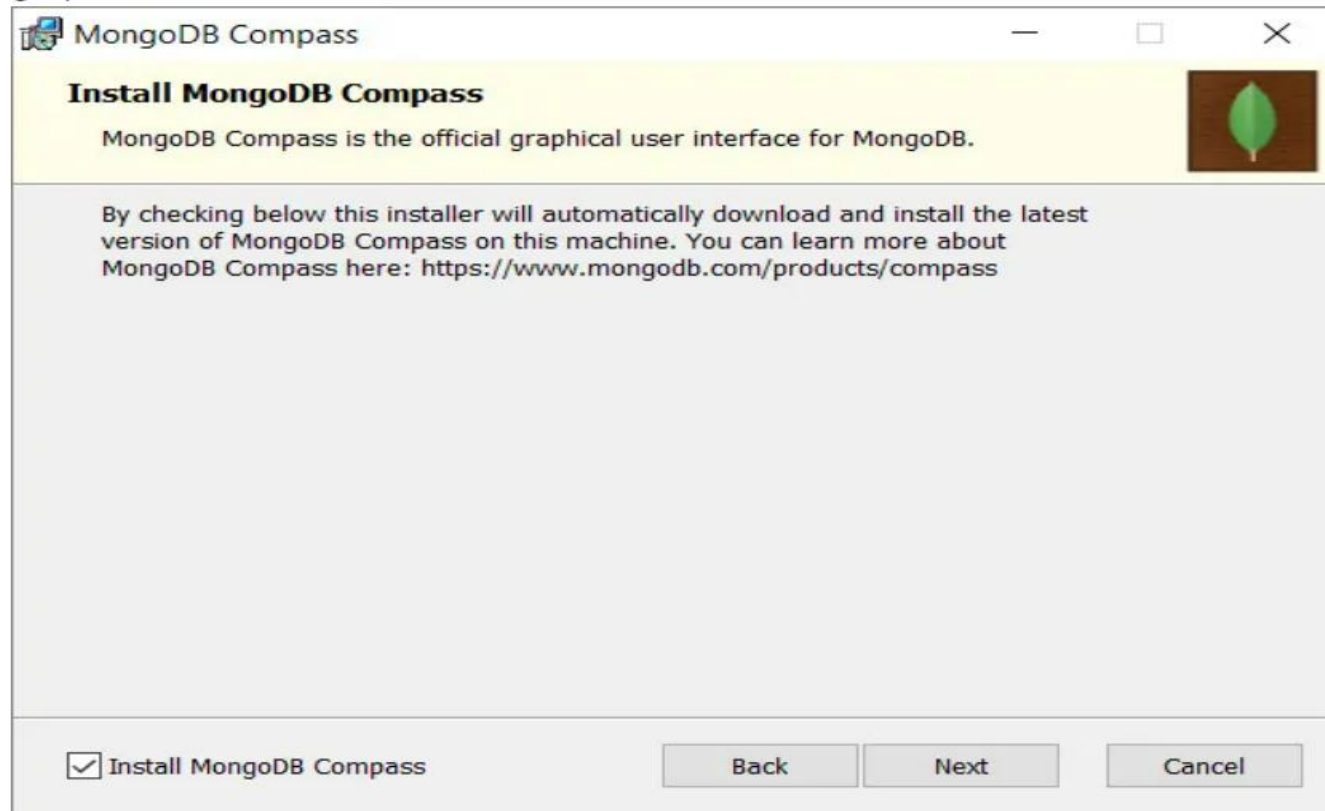
The "Service Name:" field contains "MongoDB". The "Data Directory:" field contains "C:\Program Files\MongoDB\Server\4.0\data\". The "Log Directory:" field contains "C:\Program Files\MongoDB\Server\4.0\log\".

At the bottom right, there are three buttons: "< Back" (highlighted with a blue border), "Next >", and "Cancel".

MongoDB Environment Setup

- Follow the steps:

4. The setup system will also prompt you to install MongoDB Compass, which is MongoDB official graphical user interface (GUI). You can tick the checkbox to install that as well.



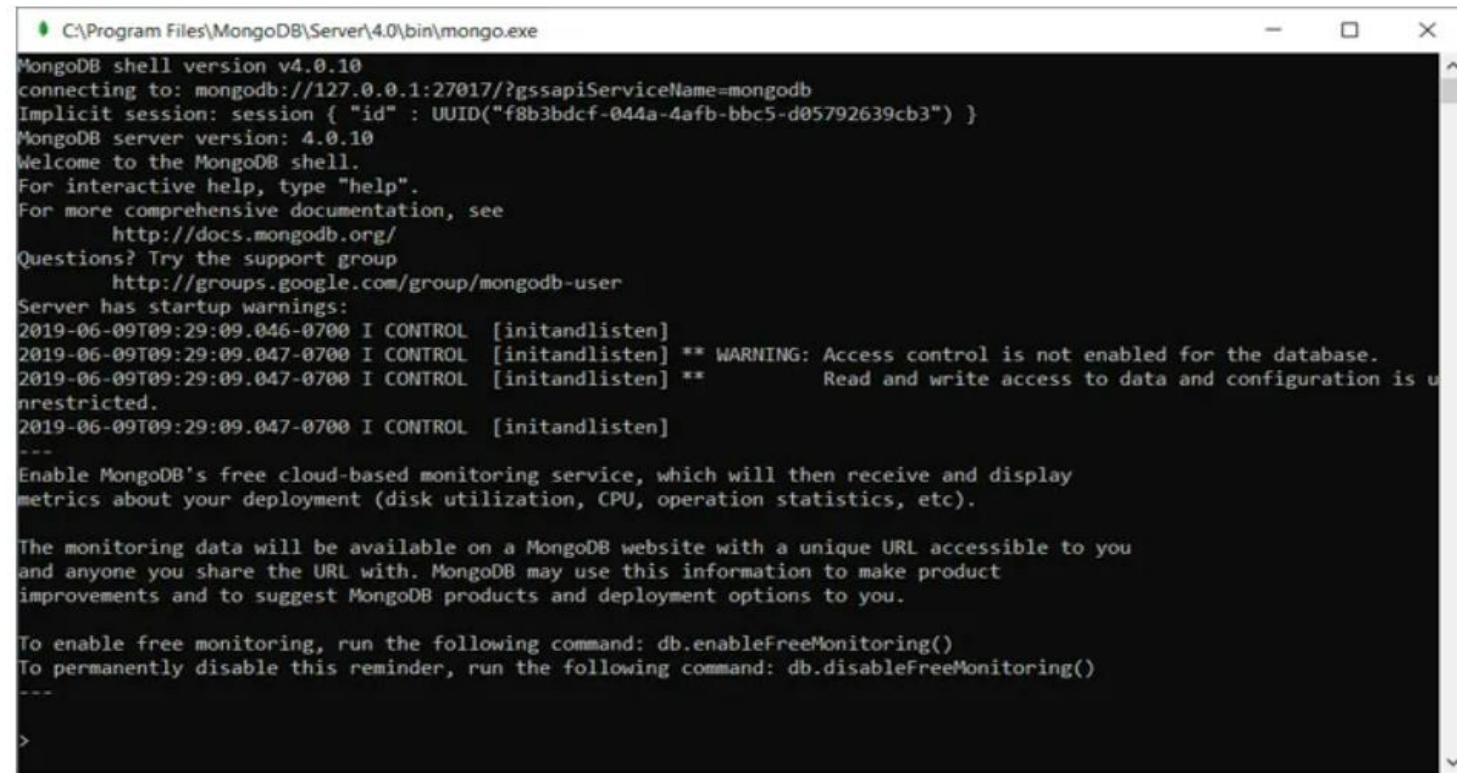
MongoDB Environment Setup

- Follow the steps:
- Once the installation is done completely, you need to start MongoDB and to do so follow the process:
 1. Open Command Prompt.
 2. Type: C:\Program Files\MongoDB\Server\4.0\bin
 3. Now type the command simply: **mongod** to run the server.

MongoDB Environment Setup

In this way, you can start your MongoDB database. Now, for running MongoDB primary client system, you have to use the command:

```
C:\Program Files\MongoDB\Server\4.0\bin>mongo.exe
```



```
C:\Program Files\MongoDB\Server\4.0\bin\mongo.exe
MongoDB shell version v4.0.10
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("f8b3bdcf-044a-4afb-bbc5-d05792639cb3") }
MongoDB server version: 4.0.10
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
  http://docs.mongodb.org/
Questions? Try the support group
  http://groups.google.com/group/mongodb-user
Server has startup warnings:
2019-06-09T09:29:09.046-0700 I CONTROL [initandlisten]
2019-06-09T09:29:09.047-0700 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
2019-06-09T09:29:09.047-0700 I CONTROL [initandlisten] **           Read and write access to data and configuration is u
nrestricted.
2019-06-09T09:29:09.047-0700 I CONTROL [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
```

MongoDB Environment Setup

- For storing data in a MongoDB, you need to create a database first.
- It will allow you to systematically organize your data so that it can be retrieved as per requirement.
- If you wish to delete a database, MongoDB also allows you to delete that.
- Today's lab, you will learn how to create and delete a database in MongoDB.

CREATING A DB IN MONGODB

- For storing data in a MongoDB, you need to create a database first.
- It will allow you to systematically organize your data so that it can be retrieved as per requirement.
- If you wish to delete a database, MongoDB also allows you to delete that.
- Today's lab, you will learn how to create and delete a database in MongoDB.

CREATING A DB IN MONGODB

- For storing data in a MongoDB, you need to create a database first.
- It will allow you to systematically organize your data so that it can be retrieved as per requirement.
- If you wish to delete a database, MongoDB also allows you to delete that.
- Today's lab, you will learn how to create and delete a database in MongoDB.

CREATING A DB IN MONGODB

- In this MongoDB tool, you need not have to produce, or it is optional to create a database manually.
- This is because MongoDB has the feature of automatically creating it for the first time for you, once you save your value in that collection.
- So, explicitly, you do not need to mention or put a command to create a database; instead it will be created automatically once the collection is filled with values.

CREATING A DB IN MONGODB

- You can make use of the "use" command followed by the database_name for creating a database.
- This command will tell the MongoDB client to create a database by this name if there is no database exists by this name.
- Otherwise, this command will return the existing database that has the name.
- Syntax
- `use my_project_db`

CREATING A DB IN MONGODB

Syntax:

```
use my_project_db
```

Example:

```
> use my_project_db  
switched to db my_project_db  
>
```

Here 'use' is the command for creating a new database in MongoDB and '**my_project_db**' is the name of the database. This will prompt you with a message that it has switched to a new DB (database) name 'my_project_db'.

CREATING A DB IN MONGODB

View the List of Databases in MongoDB

If you are eager to check the list of database that is residing with MongoDB, you can use the **show dbs** command. By default, it may not show your created database, and MongoDB's default database is a test.

```
show dbs
```

```
> show dbs
admin    0.000GB
config   0.000GB
```

In order to make a list show your database name, you have to make use of the command:

CREATING A DB IN MONGODB

Example:

```
db.movie.insert({"name":"Avengers: Endgame"})
```

Now, when you again use the **show dbs** command, it will now show your created database name in the list.

```
> show dbs
admin    0.000GB
config   0.000GB
> db.movie.insert({"name":"Avengers: Endgame"})
WriteResult({ "nInserted" : 1 })
> show dbs
admin          0.000GB
config         0.000GB
my_project_db  0.000GB
>
```

CREATING A DB IN MONGODB

It is to be noted that, for checking your currently selected database, you can use the command:

```
db
```

```
> db  
my_project_db  
>
```

CREATING A DB IN MONGODB

It is to be noted that, for checking your currently selected database, you can use the command:

```
db
```

```
> db  
my_project_db  
>
```

CREATING A DB IN MONGODB

- In MongoDB, the dropDatabase command is implemented for a similar purpose. This also helps in deleting the connected data files of that database.
- For operating this command, you have to **reside on the current database**.

CREATING A DB IN MONGODB

Example:

```
db.dropDatabase()  
{ "dropped": "my_project_db", "ok": 1 }
```

```
> db  
my_project_db  
> db.dropDatabase()  
{ "ok" : 1 }  
> show dbs  
admin    0.000GB  
config   0.000GB  
>
```

This will drop the existing database in which you are residing and will show the above message.

MONGODB DATA TYPES

- Data type is an essential component of a language or script that is used to define the type of data being used in framing the database.
- It is important for you to know that MongoDB stores data in BSON format.
- Today you will learn about the different data types that exist in MongoDB, along with their implementation techniques.

WHAT ARE JSON AND BSON?

- JSON based databases usually return query results which can be effortlessly parsed, having modest or nix transformation, straightforwardly by the use of JavaScript along with most well-liked programming languages - dropping the quantity of logic one needs for building your application layer.

WHAT ARE JSON AND BSON?

- In the case of MongoDB, data representation is done in JSON document format, but here the JSON is binary-encoded, which is termed as BSON.
- BSON is the extended version of the JSON model, which is providing additional data types, makes performance to be competent to encode and decode in diverse languages and ordered fields.

DIFFERENT MONGO DB DATA TYPES

- Remote procedure calls in MongoDB can be made by using the BSON format.
- MongoDB has a unique way of representing data types in which each data type is associated with an alias as well as a number that is usually used to search or find any specific record within a MongoDB database.
- MongoDB allows its users to implement different variations of data types:

DIFFERENT MONGO DB DATA TYPES

Integer

Integer is a data type that is used for storing a numerical value, i.e., integers as you can save in other programming languages. 32 bit or 64-bit integers are supported, which depends on the server.

Example:

```
db.TestCollection.insert({"Integer example": 62})
```

Output:

```
> use my_project_db
switched to db my_project_db
> db.TestCollection.insert({"Integer example": 62})
WriteResult({ "nInserted" : 1 })
> db.TestCollection.find()
{ "_id" : ObjectId("5d08724bce0d9292a5121a78"), "Integer example" : 62 }
>
```

DIFFERENT MONGO DB DATA TYPES

Boolean

Boolean is implemented for storing a Boolean (i.e., true or false) values.

Example:

```
db.TestCollection.insert({"Nationality Indian": true})
```

Output::

```
> db.TestCollection.insert({"Nationality Indian": true})
WriteResult({ "nInserted" : 1 })
> db.TestCollection.find()
{ "_id" : ObjectId("5d08724bce0d9292a5121a78"), "Integer example" : 62 }
{ "_id" : ObjectId("5d087412ce0d9292a5121a79"), "Nationality Indian" : true }
>
```

DIFFERENT MONGO DB DATA TYPES

Double

Double is implemented for storing floating-point data in MongoDB.

Example:

```
db.TestCollection.insert({"double data type": 3.1415})
```

Output:

```
> db.TestCollection.insert({"double data type": 3.1415})
WriteResult({ "nInserted" : 1 })
> db.TestCollection.find()
{ "_id" : ObjectId("5d08724bce0d9292a5121a78"), "Integer example" : 62 }
{ "_id" : ObjectId("5d087412ce0d9292a5121a79"), "Nationality Indian" : true }
{ "_id" : ObjectId("5d0874dcce0d9292a5121a7a"), "double data type" : 3.1415 }
```

DIFFERENT MONGO DB DATA TYPES

String

String is one of the most frequently implemented data type for storing the data.

Example:

```
db.TestCollection.insert({"string data type" : "This is a sample message."})
```

Output:

```
> db.TestCollection.insert({"string data type" : "This is a sample message."})
WriteResult({ "nInserted" : 1 })
> db.TestCollection.find()
{ "_id" : ObjectId("5d08724bce0d9292a5121a78"), "Integer example" : 62 }
{ "_id" : ObjectId("5d087412ce0d9292a5121a79"), "Nationality Indian" : true }
{ "_id" : ObjectId("5d0874dcce0d9292a5121a7a"), "double data type" : 3.1415 }
{ "_id" : ObjectId("5d087547ce0d9292a5121a7b"), "string data type" : "This is
sample message." }
>
```

DIFFERENT MONGO DB DATA TYPES

Arrays

Arrays are implemented for storing arrays or list type or several values under a single key.

Example:

```
var degrees = ["BCA", "BS", "MCA"]
db.TestCollection.insert({" Array Example" : " Here is an example of array",
" Qualification" : degrees})
```

Output:

```
> var degrees = ["BCA", "BS", "MCA"]
> db.TestCollection.insert({" Array Example" : " Here is an example of array",
" db.TestCollection.insert({" Array Example" : " Here is an example of array",
" Qualification" : degrees})
WriteResult({ "nInserted" : 1 })
> db.TestCollection.find()
{ "_id" : ObjectId("5d0a6bc0940707e25658e62c"), "Integer example" : 62 }
{ "_id" : ObjectId("5d0a6c2e28323ade3e6287f9"), "Nationality Indian" : true }
{ "_id" : ObjectId("5d0a6c3528323ade3e6287fa"), "double data type" : 3.1415 }
{ "_id" : ObjectId("5d0a6c3d28323ade3e6287fb"), "string data type" : "This is a
sample message." }
{ "_id" : ObjectId("5d0a6ee828323ade3e628800"), " Array Example" : " Here is an
example of array", " Qualification" : [ "BCA", "BS", "MCA" ] }
>
```


DIFFERENT MONGO DB DATA TYPES

Object

Object is implemented for embedded documents.

Example:

```
var embeddedObject={"English" : 94, "ComputerSc." : 96, "Maths" : 80,
                    "GeneralSc." : 85}
db.TestCollection.insert({"Object data type" : "This is Object",
                          "Marks" : embeddedObject})
```

Output:

```
> var embeddedObject = {"English" : 94, "ComputerSc." : 96, "Maths" : 80, "GeneralSc." : 85}
> db.TestCollection.insert({"Object data type" : "This is Object", "Marks" : embeddedObject})
WriteResult({ "nInserted" : 1 })
> db.TestCollection.find()
{ "_id" : ObjectId("5d0a6bc0940707e25658e62c"), "Integer example" : 62 }
{ "_id" : ObjectId("5d0a6c2e28323ade3e6287f9"), "Nationality Indian" : true }
{ "_id" : ObjectId("5d0a6c3528323ade3e6287fa"), "double data type" : 3.1415 }
{ "_id" : ObjectId("5d0a6c3d28323ade3e6287fb"), "string data type" : "This is a sample message." }
{ "_id" : ObjectId("5d0a6ee828323ade3e628800"), " Array Example" : " Here is an example of array", " Qualification" : [ "BCA", "BS", "MCA" ] }
{ "_id" : ObjectId("5d0a715828323ade3e628801"), "Object data type" : "This is Object", "Marks" : { "English" : 94, "ComputerSc." : 96, "Maths" : 80, "GeneralSc." : 85 } }
>
```