## MongoDB Mongo shell and Data base creation

MongoDB Mongo shell is an interactive JavaScript interface that allows you to interact with MongoDB instances through the command line. The shell can be used for:

* Data manipulation
* Administrative operations such as maintenance of database instances

## MongoDB Mongo shell features

MongoDB Mongo shell is the default client for the MongoDB database server. It’s a command-line interface (CLI), where the input and output are all console-based. The Mongo shell is a good tool to manipulate small sets of data.

Here are the top features that Mongo shell offers:

* Run all MongoDB queries from the Mongo shell.
* Manipulate data and perform administration operations.
* Mongo shell uses JavaScript and a related API to issue commands.
* See previous commands in the mongo shell with up and down arrow keys.
* View possible command completions using the tab button after partially entering a command.
* Print error messages, so you know what went wrong with your commands.

MongoDB has recently introduced a new mongo shell known as **mongosh**. It has some additional features, such as extensibility and embeddability—that is, the ability to use it inside other products such as VS Code.

## Installing the mongo shell

The mongo shell gets installed when you install the MongoDB server. It is installed in the same location as the MongoDB server binary.

If you want to install it separately, you can visit the [MongoDB download center](https://www.mongodb.com/try/download/shell), from there select the version and package you need, download the archive, and copy it to a location in your file system.

Mongo shell is available for all main operating systems, including:

* Windows
* Linux
* Mac OS

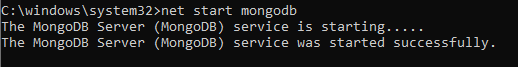
## Connect to MongoDB database

Once you’ve [downloaded and installed MongoDB](https://www.bmc.com/blogs/how-to-install-mongodb-ubuntu-mac/),  you can use the mongo shell to connect with a MongoDB server that is up and running.

**Note:** It is required that your server is already running before you connect with it through the shell. You can start the server in CMD using the following command.

Copy

net start MongoDB



## Basic commands for Mongo shell

Now it’s time to work with the Mongo shell. First, we will learn some basic commands that will help you to get started with using MongoDB.

Run the **db** command to see the database you are currently working with

db

Run the **use** command to switch to a different database. If you don’t have a database,

use company

Run the **use** command to switch to a different database. If you don’t have a database, [learn how to create a new database](https://www.bmc.com/blogs/mongodb-create-database/).

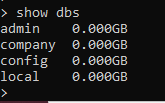
use company

Use the **find** method to fetch data in a collection. The **forEach(printjson)** method will print them with JSON formatting

db.employee.find().forEach(printjson)

  
Use the show dbs command to Show all databases

show dbs



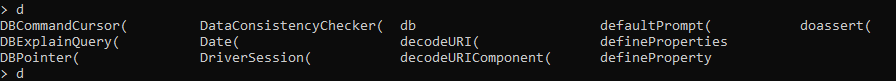
One important command will help you work with the Mongo shell easily: the **help** command. Run the help command to get a list of help options available in the mongo shell.

To get a full list of commands that you can execute on the current database, type **db.help()**

## Mongo shell keyboard shortcuts

There are two important keyboard shortcuts that you should know:

1. **Use up and down arrows** to go back and forth in the commands history.
2. **Press the tab key** to get a full list of possible commands. For example, type**d** and press tab twice. You will get the following output.



## Disadvantages of the mongo shell

Although the Mongo shell is an excellent tool for learning and testing the MongoDB server, it is difficult to be used in a production environment. Being a shell inherently carries certain disadvantages. Let’s see what they are:

* The Mongo shell is strictly a console centric method of data manipulation. While some find it easy and quick, others might not find those characteristics appealing.
* If you are working on multiple sessions, you need multiple terminals.
* If the results are too long, they scroll away.
* Repetitive commands or debugging a function need the programmer to traverse the long command line history manually.

## Alternatives to MongoDB mongo shell

So now you know the mongo shell has some disadvantages. At this point, you may want to know what other options are available. MongoDB developers have introduced drivers specific to each programming language to connect with the MongoDB databases when using MongoDB in your applications. You can find them [here](https://docs.mongodb.com/drivers/).

Additionally, many people prefer to use GUIs to work with databases nowadays.  One of the best GUI tools for MongoDB is the [MongoDB Compass](https://www.mongodb.com/products/compass). Some other useful GUI tools are:

* [NoSQLBooster](https://nosqlbooster.com/)
* [Mongo Management Studio](https://mms.litixsoft.de/)
* [Robo 3T](https://robomongo.org/)

We connect to the Mongo terminal using the mongo command

mongo

By default Mongo will connect to localhost.

We can connect to a remote server by passing arguments, like so:

mongo connection.mongolab.com:45352 -u username -p passw0rd

Once we connect to a Mongo instance we can type JavaScript directly into the console. We can create variables, do maths, write JSON.

## Exercise - connect to a console

Connect to the console at localhost. Try typing some JavaScript expressions.

* Tell me how many seconds there are in a week
* Tell me how many weeks there are in a human lifetime of 80 years.

## Creating a database

We can switch to a database in Mongo with the use command.

use petshop

This will switch to writing to the petshop database. It doesn't matter if the database doesn't exist yet. It will be brought into existence when you first write a document to it.

You can find which database you are using simply by typing db. You can drop the current database and everything in it using db.dropDatabase.

db

> petshop

db.dropDatabase()

## Exercise - Create a database

* Use the use command to connect to a new database (If it doesn't exist, Mongo will create it when you write to it).

That was easy wasn't it. Don't worry, it gets a bit harder.

## Collections

Collections are sets of (usually) related documents. Your database can have as many collections as you like.

Because Mongo has no joins, a Mongo query can pull data from only one collection at a time. You will need to take this into account when deciding how to arrange your data.

You can create a collection using the createCollection command.

use petshop

db.createCollection('mammals')

Collections will also be created automatically. If you write a document to a collection that doesn't exist that collection will be brought into being for you.

View your databases and collections using the show command, like this:

show dbs

show collections

## Exercise - Create a collection

* Use db.createCollection to create a collection. I'll leave the subject up to you.
* Run show dbs and show collections to view your database and collections.

## Documents

Documents are JSON objects that live inside a collection. They can be any valid JSON format, with the caveat that they can't contain functions.

The size limit for a document is 16Mb which is more than ample for most use cases.

### Creating a document

You can create a document by inserting it into a collection

db.mammals.insert({name: "Polar Bear"})

db.mammals.insert({name: "Star Nosed Mole"})

## Exercise - Create some documents

* Insert a couple of documents into your collection. I'll leave the subject matter up to you, perhaps cars or hats.

### Finding a document

You can find a document or documents matching a particular pattern using the find function.

If you want to find all the mammals in the mammals collection, you can do this easily.

db.mammals.find()