# Mongo Database And API Setup

https://docs.mongodb.com/manual/tutorial/query-documents/



## Overview

MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling.

### Document Database

A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.


## Field Data Types

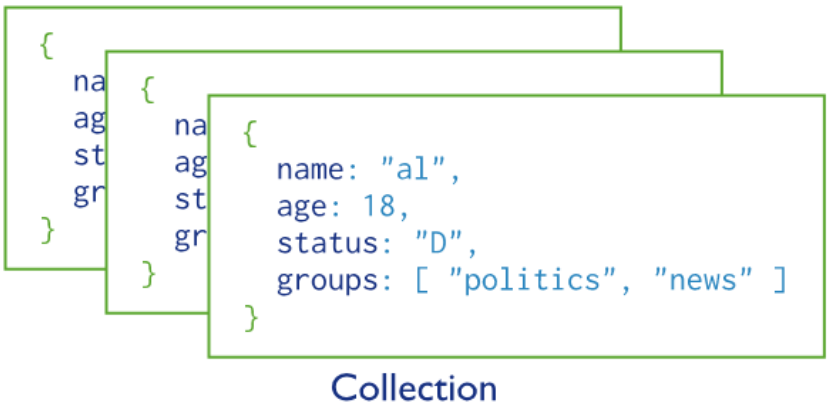
```
var mydoc = {

    _id: ObjectId("5099803df3f4948bd2f98391"),

    name: { first: "Alan", last: "Turing" },

    birth: new Date('Jun 23, 1912'),

    death: new Date('Jun 07, 1954'),

    contribs: [ "Turing machine", "Turing test", "Turingery" ],

    views : NumberLong(1250000)

}
```

The above fields have the following data types:

- _id holds an ObjectId.
- name holds an *embedded document* that contains the fields first and last.
- birth and death hold values of the *Date* type.
- contribs holds an *array of strings*.
- views holds a value of the *NumberLong* type.

## Databases And Collections

MongoDB stores BSON documents, i.e. data records, in collections; the collections in databases. BSON is a binary representation of JSON documents, though it contains more data types than JSON.




## Installation

MongoDB is currently installed on compute902.
For the best installation experience, MongoDB provides packages for popular Linux distributions. These packages are the preferred way to run MongoDB.

### Using .rpm Packages (Recommended)

To install using the package management system yum, first configure the system by adding a repo file that identifies the redhat repository containing the mongodb rpm.

ssh to the server where the new mongodb instance will be created.

1. Create a /etc/yum.repos.d/mongodb-org-4.0.repo file so that you can install MongoDB directly using yum.

   Copy the configuration below into the new file:
   [mongodb-org-4.0]
   name=MongoDB Repository
   baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/4.0/x86_64/
   gpgcheck=1
   enabled=1
   gpgkey=https://www.mongodb.org/static/pgp/server-4.0.asc

   You can specify any available version of MongoDB.

2. Install the mongoDB packages
   To install the latest stable version of MongoDB, issue the following command:

   sudo yum install -y mongodb-org

## MongoDB Configuration

### Default Configuration

The config file for your mongoDB instance will be supplied by:

```
/etc/mongod.conf
```

This file will contain the default port mongoDB:

```
# network interfaces
net:
        port: 27017
```

note**: Before you can access this port remotely, you will need to configure the ssh server on the management node to forward request for port 27017 to your mongoDB instance. In the example below compute902 has an ip-address of 192.168.8.252. So we can ssh to the management node and issue a command with the following format:

ssh -nfNT -L us0789lnxp.america.apci.com:27017:192.168.8.252:27017 root@192.168.8.252

## Start/Stop/Restart MongDB

To Start mongoDB issue the following command:

```
service mongod start
```

To Sop mongoDB issue the following command:

```
service mongod stop
```

To Restart mongoDB issue the following command:

```
service mongod restart
```

## Log File

You can follow the state of the process for errors or important messages by watching the output in the `/var/log/mongodb/mongod.log` file.

## Mongo Shell

The mongo shell is an interactive JavaScript interface to MongoDB. You can use the mongo shell to query and update data as well as perform administrative operations.

The mongo shell is a component of the MongoDB distributions. Once you have installed and have started MongoDB, connect the mongo shell to your running MongoDB instance.

- ssh to your mongoDB server
- type the ‘mongo’ command on the command line

```
[johnsol3@us0789lnxp ~]$ ssh root@compute902
Last login: Mon Aug 13 12:44:41 2018 from us0789lnxp
[root@compute902 ~]# mongo
MongoDB shell version v3.4.9
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 3.4.9
>
```

When you run mongo without any arguments, the mongo shell will attempt to connect to the MongoDB instance running on the localhost interface on port 27017.

You can use the --host --port arguments to connect to a remote instance.

Non-root users can supply the --username --password arguments to pass these values for authentication.

## Creating Collections

Records in mongoDB are JSON (in binary) structures called documents, which are stored in Collections. A collection is a grouping of documents. A collection is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields.

### Creating a new Database

Mongo will create a database implicitly when the database is first referenced.

To create a new database first logon to the mongo shell via the ‘mongo’ command.

```
>mongo
```

The ‘show databases’ command will list available databases.

```
>show databases
```

To switch to a particular database, issue the ‘use <database name>’ command

```
>use psamd
```

If the database does not exists it will be created.

## Creating a new Collection

Mongo will create a collection implicitly when the collection is first referenced.

Create or insert operations add new documents to a collection. If the collection does not currently exist, insert operations will create the collection.

MongoDB provides the following methods to insert documents into a collection:

- db.collection.insertOne()
- db.collection.insertMany()

In MongoDB, insert operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.

So to create a new collection we first switch(or create) the database via the ‘use’ command, i.e. ‘use users’. Then reference the new collection passing a JSON structure as a document in the collection:


You can also pass an array of JSON objects to the insertMany command to create multiple documents in one command.

The following example inserts three new documents into the inventory collection.

```
db.inventory.insertMany([
  { item: "journal", qty: 25, tags: ["blank", "red"], size: { h: 14, w: 21, uom: "cm" } },
  { item: "mat", qty: 85, tags: ["gray"], size: { h: 27.9, w: 35.5, uom: "cm" } },
  { item: "mousepad", qty: 25, tags: ["gel", "blue"], size: { h: 19, w: 22.85, uom: "cm" } }
])
```

$set

The $set operator replaces the value of a field with the specified value.

The $set operator expression has the following form:

copy

copied

```
{ $set: { <field1>: <value1>, ... } }
```

To specify a <field> in an embedded document or in an array, use dot notation.

## Behavior

If the field does not exist, $set will add a new field with the specified value, provided that the new field does not violate a type constraint. If you specify a dotted path for a non-existent field, $set will create the embedded documents *as needed* to fulfill the dotted path to the field.

If you specify multiple field-value pairs, $set will update or create each field.

## Examples

```
db.products.update(
  { _id: 100 },
```

```
  { $set: { "details.make": "zzz" } }
)
```

## Importing a collection