



Technology Series

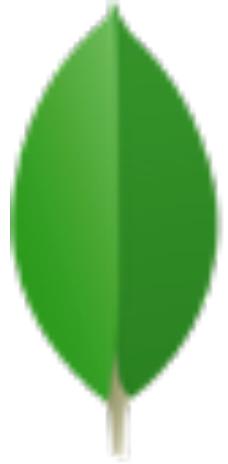
MongoDB

01

Presented by
Subhash EP



Subhash



mongoDB

{ name: mongo, type: DB }

Installing MongoDB

- MongoDB website supplies binaries to install on Linux, Mac OS X, and Windows.
- If you use Windows or Linux
 - download either the 32-bit or 64-bit version
- Mac users are safe to download the 64-bit version.

Installation ...

- After downloading and extracting the archive file,
- locate the mongod binary,
- usually located in the bin folder.
- The mongod process runs the main MongoDB server process
 - as a standalone server or
 - a single node of a MongoDB replica set

Installation ...

- In our case, we will use MongoDB as a standalone server.
- The mongod process requires
 - a folder to store the database files in (the default folder is /data/db) and
 - a port to listen to (the default port is 27017).

MongoDB on Windows

- Once downloaded,
 - unpack the archive file,
 - and move the folder to C:\mongodb.

MongoDB on Windows

- MongoDB default folder n Windows,
 - C:\data\db
- Start Command Window and
 - C:\md data\db
- You can tell the mongod service to use an alternative path for the data files using the
 - --dbpath command-line flag

Running MongoDB manually

- To run MongoDB manually,
- open the command prompt:
 - C:\> C:\mongodb\bin\mongod.exe
- will run the main MongoDB service
 - listening to the default 27017 port

Running MongoDB server on Windows

```
Windows\system32\cmd.exe - C:\mongodb\bin\mongod.exe
:\\\mongodb\\bin\\mongod.exe
mongod\\bin\\mongod.exe --help for help and startup options
07-05T12:40:29.140+0300
07-05T12:40:29.140+0300 warning: 32-bit servers don't have journaling enabled by default.
Please use --journal if you want durability.
07-05T12:40:29.140+0300
07-05T12:40:29.140+0300 [initandlisten] MongoDB starting : pid=976 port=27017 dbpath=C:\data\db\ 32-bit host=IE9Win7
07-05T12:40:29.140+0300 [initandlisten]
07-05T12:40:29.140+0300 [initandlisten] ** NOTE: This is a 32 bit MongoDB binary.
07-05T12:40:29.140+0300 [initandlisten] **           32 bit builds are limited to less than 2GB of data (or less with --journal).
07-05T12:40:29.140+0300 [initandlisten] **           Note that journaling defaults to off for 32 bit and is currently off.
07-05T12:40:29.140+0300 [initandlisten] **           See http://dochub.mongodb.org/core/32bit
07-05T12:40:29.140+0300 [initandlisten]
07-05T12:40:29.140+0300 [initandlisten] targetMinOS: windows XP SP3
07-05T12:40:29.140+0300 [initandlisten] db version v2.6.3
07-05T12:40:29.140+0300 [initandlisten] git version: 255f67a66f9603c59380b2a389e38e2cb
07-05T12:40:29.140+0300 [initandlisten] build info: windows sys.getwindowsversion(minor=1, build=7601, platform=2, service_pack='Service Pack 1') BOOST_LIB_VERSION=1.52.0
07-05T12:40:29.140+0300 [initandlisten] allocator: system
07-05T12:40:29.140+0300 [initandlisten] options: {}
07-05T12:40:29.156+0300 [initandlisten] waiting for connections on port 27017
```

Connect to MongoDB.

- To connect to MongoDB through the mongo.exe shell,
 - open another Command Prompt.
- C:\mongodb\bin\mongo.exe

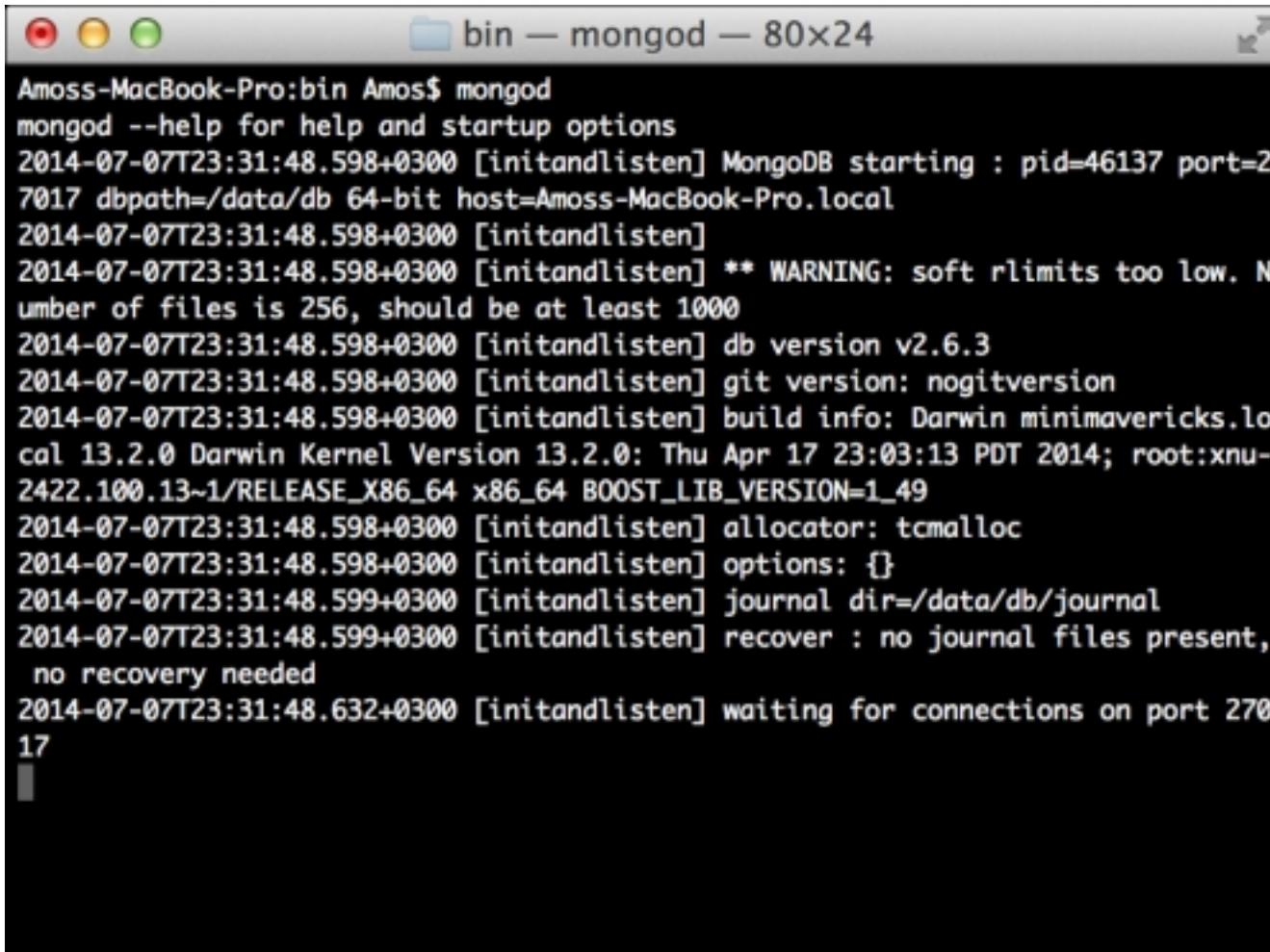
MongoDB shell Mongo running

```
c:\mongodb\bin>mongo.exe
MongoDB shell version: 2.4.8
connecting to: test
>
```

Installing MongoDB on Mac OS X and Linux

- curl -O [http://downloads.mongodb.org/...](http://downloads.mongodb.org/)
- tar -zxvf mongodb-.....tgz
- mv mongoDB-..... mongoDB
- **sudo mkdir -p /data/db**
- **sudo chown -R \$USER /data/db**
- cd mongoDB/bin
- mongod

Running MongoDB on Mac



```
Amoss-MacBook-Pro:bin Amos$ mongod
mongod --help for help and startup options
2014-07-07T23:31:48.598+0300 [initandlisten] MongoDB starting : pid=46137 port=2
7017 dbpath=/data/db 64-bit host=Amoss-MacBook-Pro.local
2014-07-07T23:31:48.598+0300 [initandlisten]
2014-07-07T23:31:48.598+0300 [initandlisten] ** WARNING: soft rlimits too low. N
umber of files is 256, should be at least 1000
2014-07-07T23:31:48.598+0300 [initandlisten] db version v2.6.3
2014-07-07T23:31:48.598+0300 [initandlisten] git version: nogitversion
2014-07-07T23:31:48.598+0300 [initandlisten] build info: Darwin minimavericks.lo
cal 13.2.0 Darwin Kernel Version 13.2.0: Thu Apr 17 23:03:13 PDT 2014; root:xnu-
2422.100.13~1/RELEASE_X86_64 x86_64 BOOST_LIB_VERSION=1_49
2014-07-07T23:31:48.598+0300 [initandlisten] allocator: tcmalloc
2014-07-07T23:31:48.598+0300 [initandlisten] options: {}
2014-07-07T23:31:48.599+0300 [initandlisten] journal dir=/data/db/journal
2014-07-07T23:31:48.599+0300 [initandlisten] recover : no journal files present,
no recovery needed
2014-07-07T23:31:48.632+0300 [initandlisten] waiting for connections on port 270
17
```

MongoDB shell Mongo running

```
Last login: Sun Jun 28 16:47:48 on ttys000
my:~ subhash$ mongo
MongoDB shell version: 3.0.1
connecting to: test
Server has startup warnings:
2015-06-28T16:47:55.261+0530 I CONTROL  [initandlisten]
2015-06-28T16:47:55.261+0530 I CONTROL  [initandlisten] ** WARNING: soft rlimits
too low. Number of files is 256, should be at least 1000
> |
```

Test your database

- run the following command in shell:
 - > db.articles.insert({title: "Hello World"})
- This will create a new **article** collection and insert a **JSON** object
- To retrieve the article object:
 - > db.articles.find()

Getting Started

- A document is the basic unit of data for MongoDB and is roughly equivalent to a row in a relational database management system
- Similarly, a collection can be thought of as a table with a dynamic schema

Collection

Document

Name: "Rick"

Age: 23

Document

Item ID: 12

Amount: 45

Getting Started

- A single instance of MongoDB can host multiple independent databases, each of which can have its own collections.
- Every document has a special key, "_id", that is unique within a collection.

Getting Started

- MongoDB comes with a simple but powerful JavaScript shell, which is useful for the administration of MongoDB instances and data manipulation.

Documents

- At the heart of MongoDB
- an ordered set of keys with associated values.
- The representation of a document varies by programming language,
 - but most languages have a data structure that is a natural fit,
 - such as a map, hash, or dictionary

Documents

- In JavaScript, for example, documents are represented as objects:
- {"greeting" : "Hello, world!", "foo" : 3}

MongoDB type-sensitive & case-sensitive

- type-sensitive

- {"foo" : 3}
- {"foo" : "3"}

- case-sensitive

- {"foo" : 3}
- {"Foo" : 3}

Step # 1 – define document

```
var new_entry = {  
    firstname: "Subhash",  
    lastname: "Edakkuda",  
    age: 25,  
    address: {  
        street: "OMR, Perungudi",  
        city: "Chennai",  
        state: "TN",  
        postalcode: 600096  
    }  
}
```

Step # 2 – Save

```
db.addressBook.save(new_entry)
```

Step # 3 – Read

`db.addressBook.find()`

```
{"greeting" : "Hello, world!", "greeting" : "Hello, MongoDB!"}
```



Collections

- A collection is a group of documents.
 - document is the MongoDB analog of a row in a relational database,
 - then a collection can be thought of as the analog to a table.

Dynamic Schemas

- Collections have dynamic schemas.
- can have any number of different “shapes.”
- For example, both documents stored in a single collection:
 - {"greeting" : "Hello, world!"}
 - {"foo" : 5}

Dynamic Documents

- The documents can not only have
 - different types for their values (string versus integer)
 - but also have entirely different keys.

Dynamic Collections?

- Because any document can be put into any collection,
- “Why do we need separate collections at all?”

Many Collections

- Keeping different kinds of documents in the same collection can be a nightmare for developers and admins.
- Developers need to make sure that each query is only returning documents of a certain type or that the application code performing a query can handle documents of different shapes.
- If we're querying for blog posts, it's a hassle to weed out documents containing author data.

Many Collections

- It is much faster to get a list of collections than to extract a list of the types in a collection.
- For example, if we had a "type" field in each document that specified whether the document was a “skim,” “whole,” or “chunky monkey,” it would be much slower to find those three values in a single collection than to have three separate collections and query the correct collection.

Many Collections

- Grouping documents of the same kind together in the same collection allows for data locality.
- Getting several blog posts from a collection containing only posts will likely require fewer disk seeks than getting the same posts from a collection containing posts and author data.

Many Collections

- We begin to impose some structure on our documents when we create indexes. (This is especially true in the case of unique indexes.)
- These indexes are defined per collection. By putting only documents of a single type into the same collection, we can index our collections more efficiently.

That's all



End of Session

©

subhash.ep@gmail.com
linkedin.com/in/subhashep