

Install MongoDB

On your virtual machine, first update the APT package repository cache with the following command:

```
sudo apt update
```

Now install *wget* and *gnupg*

```
sudo apt install wget gnupg
```

Add the GPG key of the MongoDB official package repository to your Debian installation.

```
wget -qO - https://www.mongodb.org/static/pgp/server-4.2.asc | sudo apt-key add -
```

Add the official MongoDB 4 package repository for Debian 9 on Debian 10

```
echo "deb http://repo.mongodb.org/apt/debian buster/mongodb-org/4.2 main" | \  
sudo tee /etc/apt/sources.list.d/mongodb-org-4.2.list
```

```
echo "deb http://deb.debian.org/debian/ stretch main" | \  
sudo tee /etc/apt/sources.list.d/debian-stretch.list
```

Install MongoDB 4.

```
sudo apt update  
sudo apt-get install -y mongodb-org
```

Enable **MongoDB**, which will cause it to automatically start when the virtual machine is booted.

```
sudo systemctl enable mongod
```

```

rashid@rashid-debian:~$ sudo apt-get install -y mongodb-org
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libcurl4 mongodb-org-mongos mongodb-org-server mongodb-org-shell mongodb-org-tools
The following NEW packages will be installed:
  libcurl4 mongodb-org mongodb-org-mongos mongodb-org-server mongodb-org-shell mongodb-org-tools
0 upgraded, 6 newly installed, 0 to remove and 3 not upgraded.
Need to get 98.1 MB of archives.
After this operation, 297 MB of additional disk space will be used.
Get:1 http://deb.debian.org/debian buster/main amd64 libcurl4 amd64 7.64.0-4+deb10u1 [331 kB]
Get:2 http://repo.mongodb.org/apt/debian buster/mongodb-org/4.2/main amd64 mongodb-org-shell amd64 4.2.10 [12.1 MB]
Get:3 http://repo.mongodb.org/apt/debian buster/mongodb-org/4.2/main amd64 mongodb-org-server amd64 4.2.10 [18.5 MB]
Get:4 http://repo.mongodb.org/apt/debian buster/mongodb-org/4.2/main amd64 mongodb-org-mongos amd64 4.2.10 [10.2 MB]
Get:5 http://repo.mongodb.org/apt/debian buster/mongodb-org/4.2/main amd64 mongodb-org-tools amd64 4.2.10 [57.0 MB]
Get:6 http://repo.mongodb.org/apt/debian buster/mongodb-org/4.2/main amd64 mongodb-org amd64 4.2.10 [3,524 B]
Fetched 98.1 MB in 24s (4,071 kB/s)
Selecting previously unselected package libcurl4:amd64.
(Reading database ... 71223 files and directories currently installed.)
Preparing to unpack .../0-libcurl4_7.64.0-4+deb10u1_amd64.deb ...
Unpacking libcurl4:amd64 (7.64.0-4+deb10u1) ...
Selecting previously unselected package mongodb-org-shell.
Preparing to unpack .../1-mongodb-org-shell_4.2.10_amd64.deb ...
Unpacking mongodb-org-shell (4.2.10) ...
Selecting previously unselected package mongodb-org-server.
Preparing to unpack .../2-mongodb-org-server_4.2.10_amd64.deb ...
Unpacking mongodb-org-server (4.2.10) ...
Selecting previously unselected package mongodb-org-mongos.
Preparing to unpack .../3-mongodb-org-mongos_4.2.10_amd64.deb ...
Unpacking mongodb-org-mongos (4.2.10) ...
Selecting previously unselected package mongodb-org-tools.
Preparing to unpack .../4-mongodb-org-tools_4.2.10_amd64.deb ...
Unpacking mongodb-org-tools (4.2.10) ...
Selecting previously unselected package mongodb-org.
Preparing to unpack .../5-mongodb-org_4.2.10_amd64.deb ...
Unpacking mongodb-org (4.2.10) ...
Setting up mongodb-org-tools (4.2.10) ...
Setting up libcurl4:amd64 (7.64.0-4+deb10u1) ...
Setting up mongodb-org-server (4.2.10) ...
Adding system user 'mongodb' (UID 108) ...
Adding new user 'mongodb' (UID 108) with group 'nogroup' ...
Not creating home directory '/home/mongodb'.
Adding group 'mongodb' (GID 116) ...
Done.
Adding user 'mongodb' to group 'mongodb' ...
Adding user mongodb to group mongodb
Done.
Setting up mongodb-org-shell (4.2.10) ...
Setting up mongodb-org-mongos (4.2.10) ...
Setting up mongodb-org (4.2.10) ...
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for libc-bin (2.28-10) ...

```

Now check whether MongoDB 4 is working correctly.

```
mongod --version
```

```

rashid@rashid-debian:~$ mongod --version
db version v4.2.10
git version: 88276238fa97b47c0ef14362b343c5317ecbd739
OpenSSL version: OpenSSL 1.1.1d 10 Sep 2019
allocator: tcmalloc
modules: none
build environment:
  distmod: debian10
  distarch: x86_64
  target_arch: x86_64

```

Finally, start the MongoDB service and check its status.

```
sudo service mongod start
```

```
sudo service mongod status
```

```

rashid@rashid-debian:~$ sudo service mongod status
● mongod.service - MongoDB Database Server
   Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor preset: enabled)
   Active: active (running) since Sat 2020-11-14 21:24:10 GMT; 6s ago
     Docs: https://docs.mongodb.org/manual
  Main PID: 1937 (mongod)
    Memory: 79.3M
   CGroup: /system.slice/mongod.service
           └─1937 /usr/bin/mongod --config /etc/mongod.conf

Nov 14 21:24:10 rashid-debian systemd[1]: Started MongoDB Database Server.

```

Accessing your Database remotely

Edit the `/etc/mongod.conf` file and set your `bind_ip = 0.0.0.0` in order to be able to connect from outside the virtual machine.

`sudo nano /etc/mongod.conf`

```
GNU nano 3.2 /etc/mongod.conf

# mongod.conf
# for documentation of all options, see:
# http://docs.mongodb.org/manual/reference/configuration-options/

# Where and how to store data.
storage:
  dbPath: /var/lib/mongodb
  journal:
    enabled: true
# engine:
# mmapv1:
# wiredTiger:

# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log

# network interfaces
net:
  port: 27017
  bindIp: 0.0.0.0

# how the process runs
processManagement:
  timeZoneInfo: /usr/share/zoneinfo

#security:
#operationProfiling:
#replication:
#sharding:

## Enterprise-Only Options:
#auditLog:
#snmp:
```

Restart MongoDB: `sudo service mongod restart`

Install pymongo

We will now install `pymongo` – which contains tools for interacting with MongoDB database from Python

Open the **anaconda terminal on your main/host computer** and run the following command:

`pip install pymongo`

```
(base) C:\Users\rashi>pip install pymongo
Collecting pymongo
  Downloading pymongo-3.11.0-cp38-cp38-win_amd64.whl (382 kB)
    | 382 kB 3.3 MB/s
Installing collected packages: pymongo
Successfully installed pymongo-3.11.0

(base) C:\Users\rashi>
```

PyMongo

November 14, 2020

<https://api.mongodb.com/python/current/tutorial.html>

1 Making a Connection with MongoClient

```
[3]: from pymongo import MongoClient
      client = MongoClient('192.168.56.30', 27017)
```

2 Getting a Database

```
[13]: db = client.test_database
```

3 Getting a Collection

```
[5]: collection = db.test_collection
```

4 Documents

```
[6]: import datetime
      post = {"author": "Mike",
              "text": "My first blog post!",
              "tags": ["mongodb", "python", "pymongo"],
              "date": datetime.datetime.utcnow()}
```

5 Inserting a Document

```
[7]: posts = db.posts
      post_id = posts.insert_one(post).inserted_id
      post_id
```

```
[7]: ObjectId('5fb04fc934aa675cb92f1370')
```

6 listing all of the collections in our database

```
[8]: db.list_collection_names()
```

```
[8]: ['posts']
```

7 Getting a Single Document With find_one()

The most basic type of query that can be performed in MongoDB is `find_one()`. This method returns a single document matching a query (or `None` if there are no matches). It is useful when you know there is only one matching document, or are only interested in the first match. Here we use `find_one()` to get the first document from the `posts` collection:

```
[11]: import pprint
      pprint.pprint(posts.find_one())

{'_id': ObjectId('5fb04fc934aa675cb92f1370'),
 'author': 'Mike',
 'date': datetime.datetime(2020, 11, 14, 21, 44, 17, 843000),
 'tags': ['mongodb', 'python', 'pymongo'],
 'text': 'My first blog post!'}
```

`find_one()` also supports querying on specific elements that the resulting document must match. To limit our results to a document with author “Mike” we do:

```
[10]: pprint.pprint(posts.find_one({"author": "Mike"}))

{'_id': ObjectId('5fb04fc934aa675cb92f1370'),
 'author': 'Mike',
 'date': datetime.datetime(2020, 11, 14, 21, 44, 17, 843000),
 'tags': ['mongodb', 'python', 'pymongo'],
 'text': 'My first blog post!'}
```

If we try with a different author, like “Eliot”, we’ll get no result

```
[12]: posts.find_one({"author": "Eliot"})
```

8 Querying By ObjectId

We can also find a post by its `_id`, which in our example is an `ObjectId`:

```
[14]: post_id

[14]: ObjectId('5fb04fc934aa675cb92f1370')

[15]: pprint.pprint(posts.find_one({"_id": post_id}))

{'_id': ObjectId('5fb04fc934aa675cb92f1370'),
 'author': 'Mike',
 'date': datetime.datetime(2020, 11, 14, 21, 44, 17, 843000),
```

```
'tags': ['mongodb', 'python', 'pymongo'],
'text': 'My first blog post!']}
```

9 Bulk Inserts

In order to make querying a little more interesting, let's insert a few more documents. In addition to inserting a single document, we can also perform bulk insert operations, by passing a list as the first argument to `insert_many()`. This will insert each document in the list, sending only a single command to the server:

```
[17]: new_posts = [{"author": "Mike",
                  "text": "Another post!",
                  "tags": ["bulk", "insert"],
                  "date": datetime.datetime(2009, 11, 12, 11, 14)},
                {"author": "Eliot",
                  "title": "MongoDB is fun",
                  "text": "and pretty easy too!",
                  "date": datetime.datetime(2009, 11, 10, 10, 45)}]
result = posts.insert_many(new_posts)
result.inserted_ids
```

```
[17]: [ObjectId('5fb0515034aa675cb92f1373'), ObjectId('5fb0515034aa675cb92f1374')]
```

There are a couple of interesting things to note about this example:

The result from `insert_many()` now returns two `ObjectId` instances, one for each inserted document. `new_posts[1]` has a different “shape” than the other posts - there is no “tags” field and we’ve added a new field, “title”. This is what we mean when we say that MongoDB is schema-free

10 Querying for More Than One Document

To get more than a single document as the result of a query we use the `find()` method. `find()` returns a `Cursor` instance, which allows us to iterate over all matching documents. For example, we can iterate over every document in the `posts` collection:

```
[18]: all_posts = posts.find()
      for post in all_posts:
          pprint.pprint(post)
```

```
{'_id': ObjectId('5fb04fc934aa675cb92f1370'),
 'author': 'Mike',
 'date': datetime.datetime(2020, 11, 14, 21, 44, 17, 843000),
 'tags': ['mongodb', 'python', 'pymongo'],
 'text': 'My first blog post!'}
{'_id': ObjectId('5fb0514434aa675cb92f1371'),
 'author': 'Mike',
 'date': datetime.datetime(2009, 11, 12, 11, 14),
 'tags': ['bulk', 'insert'],
 'text': 'Another post!'}
```

```
{'_id': ObjectId('5fb0514434aa675cb92f1372'),
  'author': 'Eliot',
  'date': datetime.datetime(2009, 11, 10, 10, 45),
  'text': 'and pretty easy too!',
  'title': 'MongoDB is fun'}
{'_id': ObjectId('5fb0515034aa675cb92f1373'),
  'author': 'Mike',
  'date': datetime.datetime(2009, 11, 12, 11, 14),
  'tags': ['bulk', 'insert'],
  'text': 'Another post!'}
{'_id': ObjectId('5fb0515034aa675cb92f1374'),
  'author': 'Eliot',
  'date': datetime.datetime(2009, 11, 10, 10, 45),
  'text': 'and pretty easy too!',
  'title': 'MongoDB is fun'}
```

Just like we did with `find_one()`, we can pass a document to `find()` to limit the returned results. Here, we get only those documents whose author is “Mike”:

```
[19]: for post in posts.find({"author": "Mike"}):
      pprint.pprint(post)
```

```
{'_id': ObjectId('5fb04fc934aa675cb92f1370'),
  'author': 'Mike',
  'date': datetime.datetime(2020, 11, 14, 21, 44, 17, 843000),
  'tags': ['mongodb', 'python', 'pymongo'],
  'text': 'My first blog post!'}
{'_id': ObjectId('5fb0514434aa675cb92f1371'),
  'author': 'Mike',
  'date': datetime.datetime(2009, 11, 12, 11, 14),
  'tags': ['bulk', 'insert'],
  'text': 'Another post!'}
{'_id': ObjectId('5fb0515034aa675cb92f1373'),
  'author': 'Mike',
  'date': datetime.datetime(2009, 11, 12, 11, 14),
  'tags': ['bulk', 'insert'],
  'text': 'Another post!'}
```

11 Counting

If we just want to know how many documents match a query we can perform a `count_documents()` operation instead of a full query. We can get a count of all of the documents in a collection:

```
[20]: posts.count_documents({})
```

```
[20]: 5
```

or just of those documents that match a specific query:

```
[21]: posts.count_documents({"author": "Mike"})
```

```
[21]: 3
```

12 Range Queries

MongoDB supports many different types of advanced queries. As an example, let's perform a query where we limit results to posts older than a certain date, but also sort the results by author. Here we use the special “\$lt” operator to do a range query, and also call `sort()` to sort the results by author.

```
[22]: d = datetime.datetime(2009, 11, 12, 12)
      for post in posts.find({"date": {"$lt": d}}).sort("author"):
          pprint.pprint(post)
```

```
{'_id': ObjectId('5fb0514434aa675cb92f1372'),
  'author': 'Eliot',
  'date': datetime.datetime(2009, 11, 10, 10, 45),
  'text': 'and pretty easy too!',
  'title': 'MongoDB is fun'}
{'_id': ObjectId('5fb0515034aa675cb92f1374'),
  'author': 'Eliot',
  'date': datetime.datetime(2009, 11, 10, 10, 45),
  'text': 'and pretty easy too!',
  'title': 'MongoDB is fun'}
{'_id': ObjectId('5fb0514434aa675cb92f1371'),
  'author': 'Mike',
  'date': datetime.datetime(2009, 11, 12, 11, 14),
  'tags': ['bulk', 'insert'],
  'text': 'Another post!'}
{'_id': ObjectId('5fb0515034aa675cb92f1373'),
  'author': 'Mike',
  'date': datetime.datetime(2009, 11, 12, 11, 14),
  'tags': ['bulk', 'insert'],
  'text': 'Another post!'}
```

13 Indexing

Adding indexes can help accelerate certain queries and can also add additional functionality to querying and storing documents. In this example, we'll demonstrate how to create a unique index on a key that rejects documents whose value for that key already exists in the index.

First, we'll need to create the index:

```
[23]: result = db.profiles.create_index([('user_id', pymongo.ASCENDING)], unique=True)
      sorted(list(db.profiles.index_information()))
```

```
[23]: ['_id_', 'user_id_1']
```


Notice that we have two indexes now: one is the index on `_id` that MongoDB creates automatically, and the other is the index on `user_id` we just created.

Now let's set up some user profiles:

```
[24]: user_profiles = [{'user_id': 211, 'name': 'Luke'}, {'user_id': 212, 'name': 'Ziltoid'}]
      result = db.profiles.insert_many(user_profiles)
```

The index prevents us from inserting a document whose `user_id` is already in the collection:

```
[27]: new_profile = {'user_id': 213, 'name': 'Drew'}
      duplicate_profile = {'user_id': 212, 'name': 'Tommy'}
      result = db.profiles.insert_one(new_profile) # This is fine.
      #result = db.profiles.insert_one(duplicate_profile) ## Try to run this and
      ↳ you should get an exception!
```

Install Neo4j

```
wget -O - https://debian.neo4j.com/neotechnology.gpg.key | sudo apt-key add -  
echo 'deb https://debian.neo4j.com stable latest' | sudo tee -a /etc/apt/sources.list.d/neo4j.list  
sudo apt update  
sudo apt install -y neo4j  
sudo service neo4j start
```

```
done.  
done.  
Setting up openjdk-11-jre-headless:amd64 (11.0.9+11-1~deb10u1) ...  
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/rmid to provide /usr/bin/rmid (rmid) in auto mode  
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/java to provide /usr/bin/java (java) in auto mode  
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/keytool to provide /usr/bin/keytool (keytool) in auto mode  
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jjs to provide /usr/bin/jjs (jjs) in auto mode  
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/pack200 to provide /usr/bin/pack200 (pack200) in auto mode  
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/rmiregistry to provide /usr/bin/rmiregistry (rmiregistry) in auto mode  
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/unpack200 to provide /usr/bin/unpack200 (unpack200) in auto mode  
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/bin/jfr to provide /usr/bin/jfr (jfr) in auto mode  
update-alternatives: using /usr/lib/jvm/java-11-openjdk-amd64/lib/jexec to provide /usr/bin/jexec (jexec) in auto mode  
Setting up cypher-shell (4.1.3) ...  
Setting up neo4j (1:4.1.3) ...  
Processing triggers for systemd (241-7~deb10u4) ...  
rashid@rashid-debian:~$ sudo service neo4j start
```

Check Installation:

```
neo4j --version
```

```
sudo service neo4j status
```

```
rashid@rashid-debian:~$ neo4j --version  
neo4j 4.1.3  
rashid@rashid-debian:~$ sudo service neo4j status  
● neo4j.service - Neo4j Graph Database  
   Loaded: loaded (/lib/systemd/system/neo4j.service; disabled; vendor preset: enabled)  
   Active: active (running) since Sat 2020-11-14 22:11:38 GMT; 8min ago  
     Main PID: 3633 (java)  
       Tasks: 45 (limit: 4915)  
      Memory: 572.0M  
     CGroup: /system.slice/neo4j.service  
             └─3633 /usr/bin/java -cp /var/lib/neo4j/plugins:/etc/neo4j:/usr/share/neo4j/lib/*:/var/lib/neo4j/plugins/* -XX:+UseG1GC -XX:-OmitStackTraceInFastThrow -XX:+AlwaysPreTouch -XX:+UnlockExperimentalV  
Nov 14 22:11:42 rashid-debian neo4j[3633]: 2020-11-14 22:11:42.634+0000 INFO ===== Neo4j 4.1.3 =====  
Nov 14 22:11:44 rashid-debian neo4j[3633]: 2020-11-14 22:11:44.691+0000 INFO Initializing system graph model for component 'security-users' with version -1 and status UNINITIALIZED  
Nov 14 22:11:44 rashid-debian neo4j[3633]: 2020-11-14 22:11:44.700+0000 INFO Setting up initial user from default: neo4j  
Nov 14 22:11:44 rashid-debian neo4j[3633]: 2020-11-14 22:11:44.700+0000 INFO Creating new user 'neo4j' (passwordChangeRequired=true, suspended=false)  
Nov 14 22:11:44 rashid-debian neo4j[3633]: 2020-11-14 22:11:44.710+0000 INFO Setting version for 'security-users' to 2  
Nov 14 22:11:44 rashid-debian neo4j[3633]: 2020-11-14 22:11:44.714+0000 INFO After initialization of system graph model component 'security-users' have version 2 and status CURRENT  
Nov 14 22:11:44 rashid-debian neo4j[3633]: 2020-11-14 22:11:44.719+0000 INFO Performing postinitialization step for component 'security-users' with version 2 and status CURRENT  
Nov 14 22:11:45 rashid-debian neo4j[3633]: 2020-11-14 22:11:45.121+0000 INFO Bolt enabled on localhost:7687.  
Nov 14 22:11:46 rashid-debian neo4j[3633]: 2020-11-14 22:11:46.471+0000 INFO Remote interface available at http://localhost:7474/  
Nov 14 22:11:46 rashid-debian neo4j[3633]: 2020-11-14 22:11:46.472+0000 INFO Started.  
[linux 3-19/19] (END)
```

Edit the file `/etc/neo4j/neo4j.conf` using a suitable editor such as `vi` or `nano` and uncomment the following lines (remove the `#`). You will need to run this as superuser (`sudo`).

```
sudo nano /etc/neo4j/neo4j.conf
```

```
dbms.default_listen_address=0.0.0.0  
dbms.security.auth_enabled=false
```

Enable Neo4j, which will cause it to automatically start when the virtual machine is booted.

```
sudo systemctl enable neo4j
```

Restart Database

```
sudo service neo4j restart
```

Install neo4j

The Neo4j Python driver is officially supported by Neo4j and connects to the database using the binary protocol. Use pip to install the driver.

pip install neo4j

```
(base) C:\Users\rashi>pip install neo4j
Collecting neo4j
  Downloading neo4j-4.1.1.tar.gz (67 kB)
    |#####| 67 kB 1.6 MB/s
Requirement already satisfied: pytz in c:\users\rashi\anaconda3\lib\site-packages (from neo4j) (2020.1)
Building wheels for collected packages: neo4j
  Building wheel for neo4j (setup.py) ... done
  Created wheel for neo4j: filename=neo4j-4.1.1-py3-none-any.whl size=94672 sha256=52c481c0bba1ee3c1b7de2db04399de0d1eee844f3aca74fd21c6823aea1701
  Stored in directory: c:\users\rashi\appdata\local\pip\cache\wheels\47\91\bf\efc7f154c2f0d12fc45817f35aa503ac35cd547091c26e4ff9
Successfully built neo4j
Installing collected packages: neo4j
Successfully installed neo4j-4.1.1

(base) C:\Users\rashi>
```

The database can also be accessed through a web browser at: <http://192.168.56.30:7474/>

The screenshot displays the Neo4j Browser interface. On the left sidebar, the 'Database Information' panel shows the following details:

- Use database:** neo4j - default
- Node Labels:** (9) Dude, Person
- Relationship Types:** (5) KNOWS
- Property Keys:** name
- DBMS:**
 - Version: 4.1.3
 - Edition: Community
 - Name: neo4j
 - Databases: ☒ dbs
 - Information: ☒ sysinfo
 - Query List: ☒ queries

The main workspace shows the results of two Cypher queries:

Query 1: `neo4j$ MATCH (n:Person) RETURN n LIMIT 25`

The results are displayed in a graph view, showing 7 nodes (Person(7)) and 5 relationships (KNOWS(5)). The graph shows a central node 'Arthur' connected to 'Merlin', 'Lancelot', and 'Guinevere'. To the right, 'Bob' is connected to 'Alice', who is connected to 'Cari'.

Query 2: `neo4j$ MATCH (n:Dude) RETURN n LIMIT 25`

The results show 2 nodes (Dude(2)) in a graph view, represented by orange circles.