Government College of Engineering, Jalgaon
(An Autonomous Institute of Government of Maharashtra)

Name : Pratiksha Rajendra Badgujar **Semester :** V **PRN :** 2241003
Class : T. Y. B.Tech Computer **Academic Year :** 2024-25 **Subject :** CO307U DBMS Lab
Course Teacher : Mr. Vinit Kakde
Date of Performance : **Date of Completion :**

Practical no. 11

Aim : Install MongoDB, run MongoDB on your OS and set up a python environment with MongoDB. Also Connect to MongoDB with python, get a Database Handle and Create a collection and insert a document.

How to Install and Configure MongoDB in Ubuntu?

Step 1: Importing MongoDB Repositories

To begin the installation process for MongoDB, you first need to import the Public key leveraged by the Package Management system associated with your Ubuntu installation.

Ubuntu's Package Management tools help ensure Package consistency and authenticity by cross verifying that these are signed using the GPG keys. You can import the MongoDB Public GPG key using the following line of code:

```
> sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10</span>
```

Once you've imported the GPGP key, you now need to create the Source list for your MongoDB installation. To do this, you can use the following line of code and create the "/etc/apt/sources.list.d/mongodb-org-3.4.list" list file as follows:

```
> echo "deb http://repo.mongodb.org/apt/ubuntu xenial/mongodb-org/3.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.4.list
```

With your list file now created, you can install the Local Package repository. To do this, you can use the following line of code:

```
> sudo apt-get update
```

This completes the first step to Install MongoDB on Ubuntu.

Step 2: Installing MongoDB Packages

You now need to install the latest stable version of MongoDB on your system. You can use the following command for the same:

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

Install a specific release of MongoDB:

In case you want to install a specific version of MongoDB on your system, you'll need to specify the version for each component package while installing them. You can refer the following example to implement this:

```
> sudo apt-get install -y mongodb-org=3.4 mongodb-org-server=3.4 mongodb-org-shell=3.4 mongodb-org-mongos=3.4 mongodb-org-tools=3.4
```

Step 3: Launching MongoDB as a Service on Ubuntu

With MongoDB up and running, you now need to create a Unit file, that can help your system understand the process of managing resources. For example, the most commonly leveraged Unit file helps determine how to start, stop or auto-manage a service.
To do this, you can create a configuration file, "mongodb.service in /etc/systemd/system", that will help manage the MongoDB system.

```
> sudo vim /etc/systemd/system/mongodb.service
```

Now, copy the following information in your configuration file:

```
#Unit contains the dependencies to be satisfied before the service is started.
[Unit]
Description=MongoDB Database
After=network.target
Documentation=https://docs.mongodb.org/manual
# Service tells systemd, how the service should be started.
# Key `User` specifies that the server will run under the mongodb user and
# `ExecStart` defines the startup command for MongoDB server.
```

```
[Service]
User=mongodb
Group=mongodb
ExecStart=/usr/bin/mongod --quiet --config /etc/mongod.conf
# Install tells systemd when the service should be automatically started.
# `multi-user.target` means the server will be automatically started during boot.
[Install]
WantedBy=multi-user.target
```

Once you've created your configuration file, you now need to update the system service using the following command:

```
> systemctl daemon-reload
```

Now, start/enable the updated systemd service for your MongoDB instance:

```
> sudo systemctl start mongodb
```

With your MongoDB instance now up, you now need to verify if MongoDB started on port 27017. To do this, you can use the "netstat" command as follows:

```
> netstat -plntu
```

To verify if your MongoDB instance started correctly, you can use the status command as follows:

```
> sudo systemctl status mongodb
```

You can now enable the auto-start functionality for your system as follows:

```
> sudo systemctl enable mongodb
```

In case you want to stop or restart your MongoDB instance running on your Ubuntu installation, you can use the following lines of code:

```
> sudo systemctl stop mongodb

> sudo systemctl restart mongodb
```

This is how you can launch your MongoDB service on Ubuntu and complete another step to successfully Install MongoDB on Ubuntu.

Step 4: Configuring and Connecting MongoDB

Once you've set up MongoDB as a service, you now need to launch your MongoDB installation. To do this, open the Mongo Shell and switch to the database admin mode using the following command:

```
> mongo
> use admin
```

Now, create a root user for your MongoDB installation and exit the Mongo Shell as follows:

```
> db.createUser({user:"admin", pwd:”password", roles:[{role:"root", db:"admin"}]})
```

You can now connect with your MongoDB, by first restarting MongoDB and then using the following line of code:

```
> mongo -u admin -p admin123 --authenticationDatabase admin
```

You'll now be able to see MongoDB set up a connection. You can use the "show dbs" command as follow to open a list of all available databases:

```
> show dbs
```

This is how you can successfully Install MongoDB on Ubuntu and successfully launch it.

Step 5: MongoDB Tuning

Scaling MongoDB is easy and can be done in both ways, horizontally and vertically. It is essential to ensure the optimal performance of the Database. Horizontal scaling is adding server resources such as RAM and CPUs while vertical scaling is adding servers to the configuration.

The performance of the MongoDB Database depends on several factors including Memory use, Number of concurrent connections, and WiredTiger Cache among others. The default storage engine of MongoDB is WiredTiger which preserves 50% memory. This means 8GB of RAM will have a 0.5*(8-1) memory preserver for WiredTiger.

For checking the usage stats and determining if changes are required use the following command given below.

```
> db.serverStatus().wiredTiger.cache
{
  'application threads page read from disk to cache count': 6,
  'application threads page read from disk to cache time (usecs)': 46,
  'application threads page write from cache to disk count': 184,
  'application threads page write from cache to disk time (usecs)': 10501,
  'bytes allocated for updates': 65768,
  'bytes belonging to page images in the cache': 30285,
  'bytes belonging to the history store table in the cache': 571,
  'bytes currently in the cache': 104652,
  'bytes dirty in the cache cumulative': 2813442,
  'bytes not belonging to page images in the cache': 74366,
  'bytes read into cache': 28042,
  'bytes written from cache': 1283385,
  'cache overflow score': 0,
  'checkpoint blocked page eviction': 0,
  'checkpoint of history store file blocked non-history store page eviction': 0,
  'eviction calls to get a page': 2,
  'eviction calls to get a page found queue empty': 2,
  'eviction calls to get a page found queue empty after locking': 0,
  'eviction currently operating in aggressive mode': 0,
  'eviction empty score': 0,
  'eviction passes of a file': 0,
  'eviction server candidate queue empty when topping up': 0,
  'eviction server candidate queue not empty when topping up': 0,
  'eviction server evicting pages': 0,
  'eviction server slept, because we did not make progress with eviction': 0,
  'eviction server unable to reach eviction goal': 0,
  'eviction server waiting for a leaf page': 0,
```

```
  'eviction state': 64,
  'eviction walk target pages histogram - 0-9': 0,
  'eviction walk target pages histogram - 10-31': 0,
  'eviction walk target pages histogram - 128 and higher': 0,
  'eviction walk target pages histogram - 32-63': 0,
  'eviction walk target pages histogram - 64-128': 0,
  'eviction walk target pages reduced due to history store cache pressure': 0,
  'eviction walk target strategy both clean and dirty pages': 0,
  'eviction walk target strategy only clean pages': 0,
  'eviction walk target strategy only dirty pages': 0,
  'eviction walks abandoned': 0,
..............
```

From the above result, some of the key points to note are listed below.

wiredTiger.cache.maximum bytes configure
wiredTiger.cache.bytes currently in the cache
wiredTiger.cache.pages read into cache
wiredTiger.cache.pages written from cache
wiredTiger.cache.tracked dirty bytes in the cache

To check the usage of WiredTiger Concurrency Read and Write Ticket, follow the command given below.

```
> db.serverStatus().wiredTiger.concurrentTransactions
{
  write: { out: 0, available: 128, totalTickets: 128 },
  read: { out: 1, available: 127, totalTickets: 128 }
}
```

Step 6: Uninstall MongoDB on Ubuntu (Optional)

Warning: All databases and their respective configurations would be removed after this process is put in place. Ensure that you back up all your data and configuration information before proceeding with this process, as it’s irreversible.

To uninstall MongoDB on Ubuntu, you first need to remove the MongoDB packages. To do this, you can stop the MongoDB service and execute the following command to remove the installed packages:

```
> sudo apt-get purge mongodb-org*
```

You can remove your created databases, log files and directories using the following command:

```
> sudo rm -r /var/log/mongodb
> sudo rm -r /var/lib/mongodb
```

This is how you can uninstall MongoDB on Ubuntu.

Mongo DB

MongoDB is a document database designed for ease of development and scaling. This Lab.Computer assignment comes running with a mongo database and we will learn basic concepts and querying with Mongo using its Python client library pymongo. This assignment has been preinstalled with PyMongo.

PyMongo

PyMongo is the client that implements the protocol that mongodb server implements and makes it available as a Python library. Lets see how we connect to our Mongo database using PyMongo. You can find more about it here: **https://github.com/mongodb/docs/blob/master/source/reference/mongodb-wire-protocol.txt**

Making a Connection with MongoClient

The first step when working with PyMongo is to create a MongoClient to the running mongod instance. Doing so is easy:

In [3]:

```
import pymongo
client = pymongo.MongoClient()
### GEt the version of mongo server and put it in variable ver
ver = None
ver = client.server_info()['version']
```

In [4]:

```
### BEGIN HIDDEN TEST
assert(ver == "4.2.23")
```

END HIDDEN TEST

Database in mongo

A single instance of MongoDB can support multiple independent databases. When working with PyMongo you access databases using attribute style access on MongoClient instances.

In [5]:
```
## Creating a database named test_db
test_db1 = client.test_db1

test_db2 = None
# Create a database named test_db2 and assign to the test_db2 variable
# YOUR CODE HERE
test_db2 = client.test_db2
```

Collection in MongoDB

A collection is a group of documents stored in MongoDB, and can be thought of as roughly the equivalent of a table in a relational database. Getting a collection in PyMongo works the same as getting a database:

In [6]:
```
test_collection1 = test_db1.test_collection1

test_collection2 = None
# Create a collection named test_collection2 and assign to the test_collection2 variable
# YOUR CODE HERE
test_collection2 = test_db1.test_collection2
```

Inserting data to a collection

Now we will insert some data to the collection. MongoDB is schemaless, which means that no schema is enforced by the database — you can add and remove fields the way you want and MongoDB will not complain. This makes life a lot easier in many regards, especially when there are frequent changes to the data model. Lets see how we insert data to a Mongo database

In [7]:

```
import datetime
test_collection1.delete_many({})
post1 = {"author": "Mike",
       "text": "My first blog post!",
       "tags": ["mongodb", "python", "pymongo"],
       "date": str(datetime.datetime.utcnow())}
test_collection1.insert_one(post1)

## Insert a second post whose "author" is "Jane" and has a tag named "C++" in the collection test_collection1
# YOUR CODE HERE
post2 = {"author" : "Jane",
      "text" : "My first blog post!",
      "tags" : ["mongodb", "python", "pymongo", "C++"],
      "date" : str(datetime.datetime.utcnow())}
test_collection1.insert_one(post2)
print(post1)
print(post2)
```

Output→{'author': 'Mike', 'text': 'My first blog post!', 'tags': ['mongodb', 'python', 'pymongo'], 'date': '2022-12-03 09:20:09.170494', '_id': ObjectId('638b14c9c2ec80590e417f61')}

{'author': 'Jane', 'text': 'My first blog post!', 'tags': ['mongodb', 'python', 'pymongo', 'C++'], 'date': '2022-12-03 09:20:09.171466', '_id': ObjectId('638b14c9c2ec80590e417f62')}

Conclusion:

MongoDB is a powerful and flexible NoSQL database that caters to the needs of modern applications requiring scalable, high-performance data management.

Question and Answers:-

1. **What is MongoDB and what are its key features?**

   Ans→ MongoDB is a free and open-source document-oriented database management system. It is a NoSQL database that uses JSON-like documents with optional schemas for storing and managing data. Its key features include high scalability, flexibility, and performance.

2. **How does MongoDB differ from traditional relational databases?**

   Ans→ MongoDB differs from traditional relational databases in several ways. First, it uses a document-oriented data model, which allows for greater flexibility and scalability compared to the rigid table-based structure of relational databases. Second, MongoDB uses dynamic schemas, which means that data can be added or modified without the need for predefined schema definitions. Third, MongoDB provides built-in support for sharding and replication, which enables it to handle large amounts of data and ensure high availability.

3. **What are the main advantages of using MongoDB?**

   Ans→Some of the main advantages of using MongoDB include its high scalability and performance, flexibility, and ability to handle complex data structures. It also offers a rich query language and powerful indexing capabilities, as well as built-in support for sharding and replication. Additionally, MongoDB is easy to use and has a large and active community of developers and users.

4. **What are the main limitations of MongoDB?**

   Ans→One of the main limitations of MongoDB is that it does not support the concept of transactions, which can be problematic for applications that require

strong data consistency and integrity. Additionally, MongoDB does not support joins, which can make it more difficult to query and manipulate data from multiple collections. Finally, MongoDB may not be the best choice for applications that require strict data schemas or complex data relationships.

5. Can MongoDB support transactions?

Ans→ No, MongoDB does not support the concept of transactions. Instead, it uses atomic operations at the document level, which provide a limited form of data consistency and integrity. This means that a write operation in MongoDB either succeeds or fails, but it cannot be partially completed or rolled back. This can be problematic for applications that require strong data consistency and integrity.

Name & Sign of Course Teacher

Mr. Vinit Kakde