Getting Started with MongoDB and the tools

Pierre Quentin Ngandjui Tiako

May 25, 2024

Abstract

This document provides a comprehensive guide to getting started with MongoDB and the tools present in the git. It covers prerequisites, installation steps, basic usage, and troubleshooting tips.

1 Introduction

Welcome to the MongoDB and the t getting started guide. This document is designed to help new users quickly and efficiently begin using MongoDB and the tools.

2 Prerequisites

Before you begin, ensure you have the following prerequisites:

- A computer with internet access
- An operating system (Windows, Linux)
- Basic knowledge of command line interface
- Requirements Latest versions (Min 7.0) `https://www.mongodb.com/docs/manual/administration/production-notes/`
  - Min 10 GB of free disk space
  - Min 4 GB RAM
  - minimum x86_64 microarchitectures processor
  - Min Windows server 2022 / windows 11 or Ubuntu 20.04

3 Installation

Follow these steps to install MongoDB on your system.

3.1 Windows

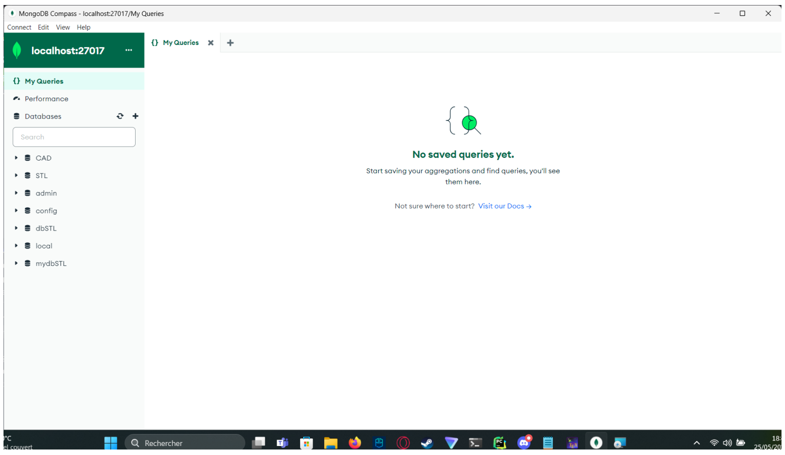
1. Download the installer from the MongoDB website.
2. Run the installer and follow the on-screen instructions.
3. Keep the default parameters so that Mongo server could run in background at the start of your machine
4. Add the path to the Mongo server bin files to your environnement variables
5. Make sure to install MongoDB Compass on your local computer to have a GUI to observe the database
6. For a more command line use (optionnal)
    - Download Mongo shell executable file (.msi)
    - Follow the installation steps (keep the default parameters)
7. Verify the installation by running `mongod --version` in the command prompt or open Mongo Compass and try to connect via the default address

3.2 Linux

1. Follow the steps of Mongo DB Linux installation depending on your Linux version

4 Create a database in the Mongo Server

1. Create a new database in your Mongo Server
    - Via Mongo Compass:

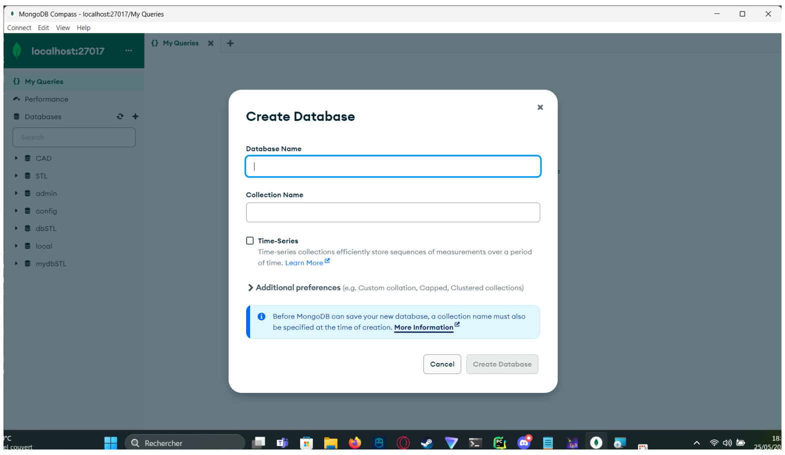


No saved queries yet.

Start saving your aggregations and find queries, you'll see them here.

Not sure where to start? Visit our Docs →

Figure 1: Mongo Compass Interface After Connection to the database



Create Database

Database Name

Collection Name

Time-Series

Time-series collections efficiently store sequences of measurements over a period of time. Learn More

Additional preferences (e.g. Custom collation, Capped, Clustered collections)

Before MongoDB can save your new database, a collection name must also be specified at the time of creation. More Information

Cancel Create Database

Figure 2: Create a new Database, need a first collection name

- via Mongossh:
  (a) Open MongoSB Shell in terminal:

```
mongo
```

  (b) Switch to a new database: replace myNewDatabase by the name of your database

```
use myNewDatabase
```

  (c) Verify the current database:

```
db
```

5 Tools and Database Initialization

1. Clone the tools from the Gitlab link

2. In your python environnement, Install Packages from requirements.txt in the python tools folder:

```
    pip install -r requirements.txt
```

3. Change the Mongo database parameters in the main.py with the parameters of the database you create before and the CAD_root_directory (the root path in the machine where the CAD are stored) in the MongoDBManagement.py

4. Create an instance of the object MongoDBManagement with the Mongo database parameters and use the appropriate method to initialize the database

```
    # Replace by Mongo database parameters
    hostname = "localhost"
    port = 27017
    dbname = "CAD"
    collection_name = 'files'

    # Create the object to communicate with the database
    mongo_db_object = MongoDBManagement(mongo_hostname=
       hostname, mongo_port=port, mongo_dbname=dbname)

    # Initialize an empty database with all the documents
        referencing the CAD (ONE TIME USE)
    mongo_db_object.initialize_database(dir_name=
       collection_name)
```

5. If the CAD files are stored on another machine than the one running this code, make sure to actualize the sftp parameters to connect remotely to this machine. Add ans change this in the precedent code:

```
    # Replace by sftp parameters if the CAD are stored remotely
    sftpHost = 'localhost'
    sftpPort = 2000
    uname = 'sftpuser'
    pwd = 'SFTP789'

    # Initialize an empty database with all the documents referencing the CAD (ONE TIME USE)
    mongo_db_object.initialize_database(dir_name=collection_name, isLocal="False")
```

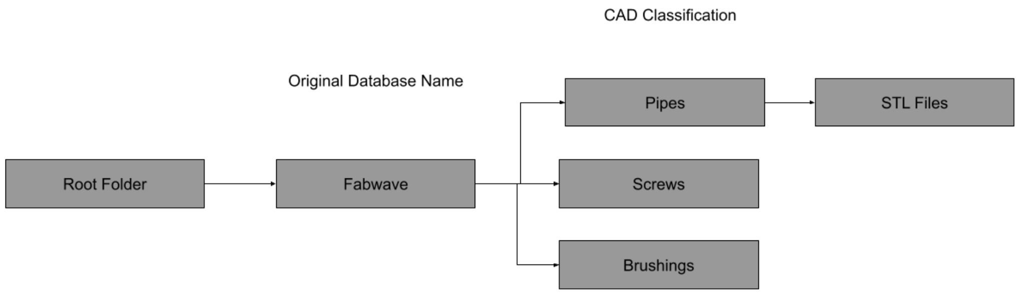


Figure 3: CAD files must be stored according to this organization. In the CAD root directory, there are folders named after the original CAD databases. And inside, if the original database provides a file classification, this is retained, as with Fabwave.

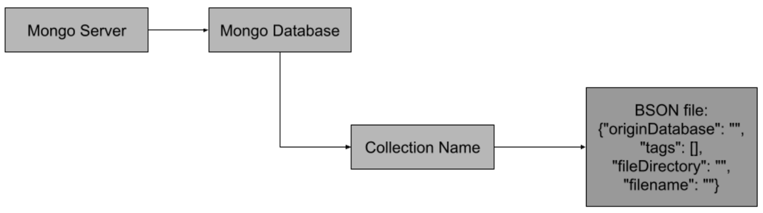


Figure 4: Database Structure. In a Mongo Server, there is a database that have on collection where all the data are stored. Those data are in the form of a BSON file where the reference of the CAD file are stored.

6 Python Code Usage

6.1 Overview

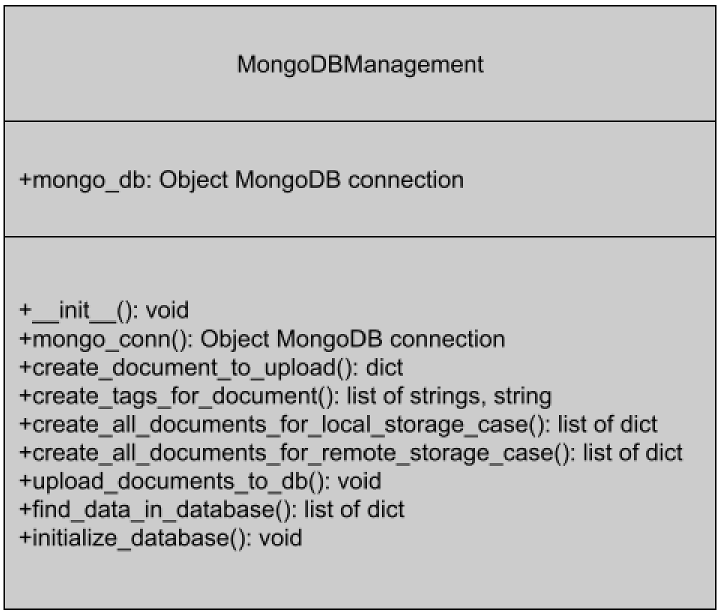

MongoDBManagement
+mongo_db: Object MongoDB connection
+__init__(): void
+mongo_conn(): Object MongoDB connection
+create_document_to_upload(): dict
+create_tags_for_document(): list of strings, string
+create_all_documents_for_local_storage_case(): list of dict
+create_all_documents_for_remote_storage_case(): list of dict
+upload_documents_to_db(): void
+find_data_in_database(): list of dict
+initialize_database(): void

Figure 5: UML representation of the MongoDBManagement object for python code. It contains a public attribute (represented by the + sign) mongo_db, which is an object that enables connection to a MongoDB database. It also contains 9 public methods, each of which returns a certain type of variable, void and if the method returns no variable.

6.2 Examples

Here are some examples of how to use the Python tools:

6.2.1 Example 1: Create the documents if the CAD are stored locally

To create a list of dictionaries, each one store the reference of one CAD file (the parameters of the Mongo database need to be change as told earlier:

```
# Replace by Mongo DB database parameters
hostname = "localhost"
port = 27017
dbname = "CAD"
collection_name = 'files'

# Create the object to communicate with the database
mongo_db_object = MongoDBManagement(mongo_hostname=hostname,
    mongo_port=port, mongo_dbname=dbname)

# Create the documents if the CAD are stored locally
docs = mongo_db_object.
    create_all_documents_for_local_storage_case()
```

6.2.2 Example 2: Upload a document / data in the database

Documents are BSON files which are JSON-like files. The representation of those files in python are in the form of dictionary. Here is an example of a document created in python and it is upload to the database:

```
# Replace by Mongo DB database parameters
hostname = "localhost"
port = 27017
dbname = "CAD"
collection_name = 'files'

# Create the object to communicate with the database
mongo_db_object = MongoDBManagement(mongo_hostname=hostname,
    mongo_port=port, mongo_dbname=dbname)

doc = {"originDatabase": "Fabwave", "tags": ["Pipes"], "
    fileDirectory": "D:\CAD␣files\Fabwave\CAD16-24\Pipes\STL",
        "filename": "0a79f67b-2028-46e7-899d-7c134f29079f.stl"
            }

# Upload the documents in the database
mongo_db_object.upload_documents_to_db(doc, dir_name=
    collection_name)
```

6.2.3 Example 3: Find / Load documents from the database

Here is an example, on how to extrat all the documents where in the field tags there is a value "Pipes". In the first initialization of the database without any modification, there are normally 95 documents with the tag "Pipes":

```python
# Replace by Mongo DB database parameters
hostname = "localhost"
port = 27017
dbname = "CAD"
collection_name = 'files'

# Create the object to communicate with the database
mongo_db_object = MongoDBManagement(mongo_hostname=hostname,
    mongo_port=port, mongo_dbname=dbname)

# Find documents in database
query = {"tags": "Pipes"}
found_docs = mongo_db_object.find_data_in_database(query=
    query, collection=collection_name)
print(len(found_docs))  # Should be 95 in this example
```

7 MATLAB Code

7.1 Overview

As the core of the pipeline is written in MATLAB, this part is the main tool for establishing communication between the database and the pipeline. Since MATLAB has tools for communicating with a database, at first sight these tools could be used without overlaying to achieve the desired result. However, between writing the query filter, and the way MATLAB imports data from the database (which we'll look at below), it's necessary not only to add overlays to MATLAB's existing methods, but also to have code that enables the re positioning step. This code consists of 5 objects: **QueryManager**, **JSONWriter**, **ImportData**, **AutomatePositionning** and **AddBoundariesLimitsToDataInDB**.

There is a need to modify as in the Python Code the parameters of the Mongo database:

```
%% Connect to database (change the appropriate database name
   dbname)
server = "localhost";
port = 27017;
dbname = "CAD";
conn = mongoc(server,port,dbname);

%% List of Collections
collection = "files";
```

7.2 Examples

Here are some examples of how to use the MATLAB tools:

7.2.1 Example 1: Compute the boundaries of each CAD and update the correponding document in the database

AddBoundariesLimitsToDataInDB is used here to add the CAD dimensions to the data in the database. Having the approximate dimensions of the object can be used as filter criteria for working specifically with a particular size of object. This is what **AddBoundariesLimitsToDataInDB** does: it loads all the data, calculates the object's boundaries on the X, Y and Z axes, adds a field boundaries to the data in the database and updates it with the calculated values.

```
%% Compute Boundaries (if new databases, compute the
   boundaries of the CAD and add new field boundaries to the
   datas)
update_boundaries = AddBoundariesLimitsToDataInDB(conn,
   collection);
docs = update_boundaries.list_documents;
boundaries = update_boundaries.list_documents_and_boundaries;
update_boundaries.updateBoundariesInDB();
```

7.2.2 Example 2: Write a Query

QueryManager is used to write MongoDB Query filters. In MATLAB, queries must be passed in the form of strings of characters in JSON format. To write this query filter:

$$\text{"\{"tags" : \{"\$all" : ["Mechanical component","Pipes"]\}\}"} \tag{1}$$

Done as it follows:

```
%% Define search Query (For a more complex database, it will
   need to learn how to write a query from scratch)

queryManager = QueryManager();

search_struct1.tags = struct("all", ["Pipes" "Mechanical
   component"]);

query1 = queryManager.convert_from_struct_to_query(
   search_struct1);
```

7.2.3 Example 3: Import data from the database

ImportData is an object for managing the import of data from the database. Here is an example to obtain a struture array that contains the documents that respect the conditions of the precedent query, it is supposed to give without modification of the database and the code an array with a length of 95.

```
%% Import Data when stored locally

data_import = ImportData();

data = data_import.import_from_one_collection(conn,
   collection, Query=query);
```

7.2.4 Example 4: Generate the JSON file with the steps to position the CAD

AutomatePositionning is used to reposition the object in relation to the fingerbase. A simple positioning have been decided. This positioning just consists of centering the object in relation to the fingerbase and aligning their main axis (the largest axis) and making it protrude slightly on the z axis. Either a JSON file with this positioning steps is generated, or it is possible to add positioning steps directly in the method and use those positioning steps in the JSON file. Here is the case for the computation of positioning steps:

```

aut_pos = AutomatePositionning();

% Name of the json genrated file
jsonfile_name = "Test";

aut_pos.json_operation_gen(jsonfile_name, data);
```

Here is the case if we add the positioning steps manually:

```

aut_pos = AutomatePositionning();

% Name of the json genrated file
jsonfile_name = "Test";

position_steps.indv_transf_steps_orientation = [[0 0 0], [0 0
    -90]];
position_steps.indv_transf_steps_translation = [[0 0 -1], [0
   0 0]];

aut_pos.json_operation_gen(jsonfile_name, pc_filename,
   position_steps=position_steps);
```

8 Next steps

To see more about the code, look into the code slides presentation.