B00836202 Poojan Patel

Assignment – 3

Job 1 :

According to the given architecture, User will able to perform search based on the book name or author name. Also user will be able to submit and retrieve the notes for a successful search. The backend of that html is written In flask(python). That backend python file will interact with main service which will interact with other services. Other services includes,

1. Notes -> it will store the note in text file(Json format) and notes can be retrieved based on the keyword
2. log -> it will store the searched keyword, time and the frequency of the keyword searched in text file(Json format)
3. catalog -> it will maintain a collection in mongodb where it will first look whether the information is available in the collection or not. If it is not there, it will search in the database, store that results in catalog collection and gives the results.

Mongodb -> 3 collections

1. Author_Book -> which contains all the author and book data scraped from the files
2. Catalog -> as discussed above, it will store successful results of searched keyword
3. Time -> it will store the file processing start time, end time and file name

API calls

APIs -> Main_API, log, catalog, notes
Flow -> when search button is pressed, Main_API will be called based on the author or book, the Main_API will call log and catalog API. Catalog API will return the results and log api will log the details. For note submission and retrieval, Main_API will be called and Main_API will call the Notes API. Only catalog API will interact with mongodb while log and notes will interact with text file(json format).

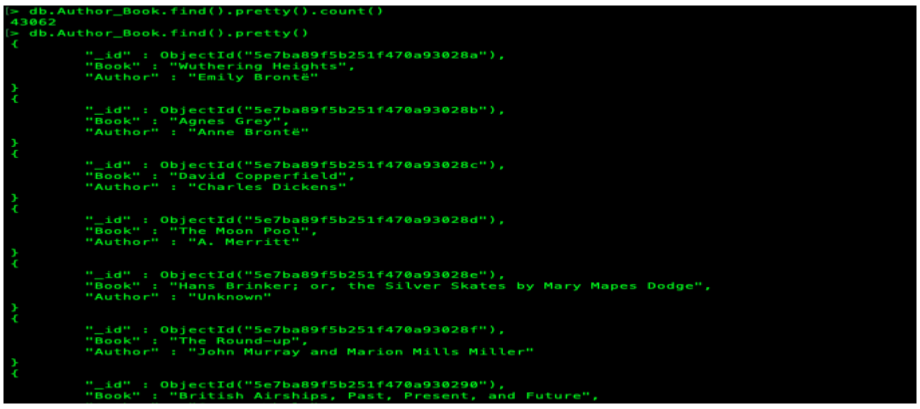
Containers in Docker

All the APIs are placed in different containers in same docker. All the APIs runs on different ports. The container has yml file which is responsible to run the container, Requirement file in which we have to specifies the modules require to run the python file, and Docker file which will install the modules specified in the requirement file, run the specified command, run the python file on specified port.

Job 2 :

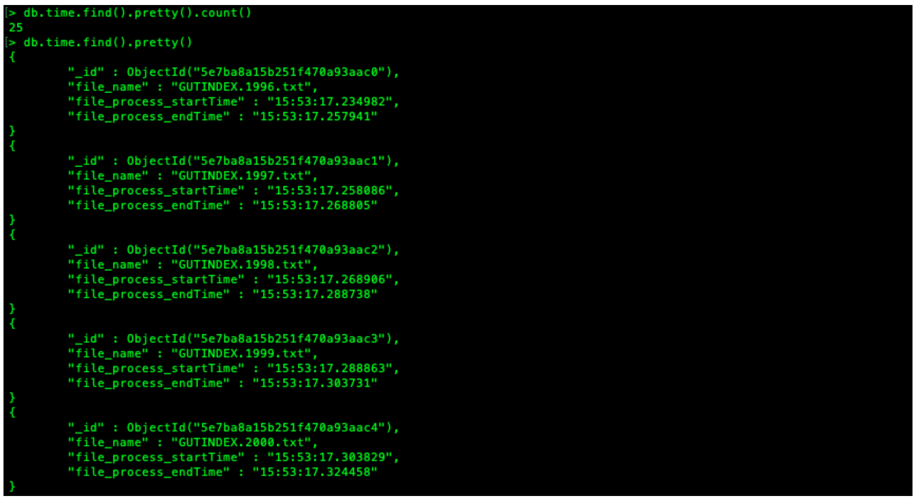
Extraction Engine

1. First, all the files were downloaded from the given website[4].
2. Python script was written to extract the data from all the files downloaded.
3. In the same file, cleaning was performed like removing the special characters, removing the non-English books etc.
4. After extracting books and authors from the files, all the information were stored in the mongodb[3] collection named Author_Book.
5. Start and end time to process each file with name are also stored in the mongodb collection named time.



```
[> db.Author_Book.find().pretty().count()
43062
[> db.Author_Book.find().pretty()
{
        "_id" : ObjectId("5e7ba89f5b251f470a93028a"),
        "Book" : "Wuthering Heights",
        "Author" : "Emily Brontë"
}
{
        "_id" : ObjectId("5e7ba89f5b251f470a93028b"),
        "Book" : "Agnes Grey",
        "Author" : "Anne Brontë"
}
{
        "_id" : ObjectId("5e7ba89f5b251f470a93028c"),
        "Book" : "David Copperfield",
        "Author" : "Charles Dickens"
}
{
        "_id" : ObjectId("5e7ba89f5b251f470a93028d"),
        "Book" : "The Moon Pool",
        "Author" : "A. Merritt"
}
{
        "_id" : ObjectId("5e7ba89f5b251f470a93028e"),
        "Book" : "Hans Brinker; or, the Silver Skates by Mary Mapes Dodge",
        "Author" : "Unknown"
}
{
        "_id" : ObjectId("5e7ba89f5b251f470a93028f"),
        "Book" : "The Round-up",
        "Author" : "John Murray and Marion Mills Miller"
}
{
        "_id" : ObjectId("5e7ba89f5b251f470a930290"),
        "Book" : "British Airships, Past, Present, and Future",
```

Figure 1 Insertion of data into mongodb and data count



```
[> db.time.find().pretty().count()
25
[> db.time.find().pretty()
{
        "_id" : ObjectId("5e7ba8a15b251f470a93aac0"),
        "file_name" : "GUTINDEX.1996.txt",
        "file_process_startTime" : "15:53:17.234982",
        "file_process_endTime" : "15:53:17.257941"
}
{
        "_id" : ObjectId("5e7ba8a15b251f470a93aac1"),
        "file_name" : "GUTINDEX.1997.txt",
        "file_process_startTime" : "15:53:17.258086",
        "file_process_endTime" : "15:53:17.268805"
}
{
        "_id" : ObjectId("5e7ba8a15b251f470a93aac2"),
        "file_name" : "GUTINDEX.1998.txt",
        "file_process_startTime" : "15:53:17.268906",
        "file_process_endTime" : "15:53:17.288738"
}
{
        "_id" : ObjectId("5e7ba8a15b251f470a93aac3"),
        "file_name" : "GUTINDEX.1999.txt",
        "file_process_startTime" : "15:53:17.288863",
        "file_process_endTime" : "15:53:17.303731"
}
{
        "_id" : ObjectId("5e7ba8a15b251f470a93aac4"),
        "file_name" : "GUTINDEX.2000.txt",
        "file_process_startTime" : "15:53:17.303829",
        "file_process_endTime" : "15:53:17.324458"
}
```

Figure 2 Start and End time for each file to process

[Note : It is written that after processing each file, 5 minute delay should be introduced but as there are 25 files, it will take around 2 hours so it is not feasible to do so. However it can be done easily by just writing 1 line in python. So it is not that I can’t do that but it is not feasible to do so.]

[Note : all the downloaded files and script to scrap, clean and loading data in mongodb are available in Scraping folder. All the screenshots can be found in **Screen shots** folder]

Job 3 :

Web Page

The web page was created using html, CSS, and Bootstrap. For scripting, Python flask was used. The web page has dropdown which has 2 options Book, and Author. One can select any of the option and can search according to that. For successful search, note submission and note retrieval option will come for that searched keyword.

Screen shot of web pages are available in **screen shot/front end** folder and html file for website and backend can be found in folder **Website**

It can be accessed on http://100.24.21.95:1999/

Job 4 :

Docker Containers

To know the flow of API, refer to Job 1 section -> API calls.

[Note: All the APIs and text files of notes and log can be found in **API** folder]

[Note: as there is no option to retrieve the logs, sample file which was created for testing is attached. To verify it, you can refer log.py file which contains the logic to write the log file]

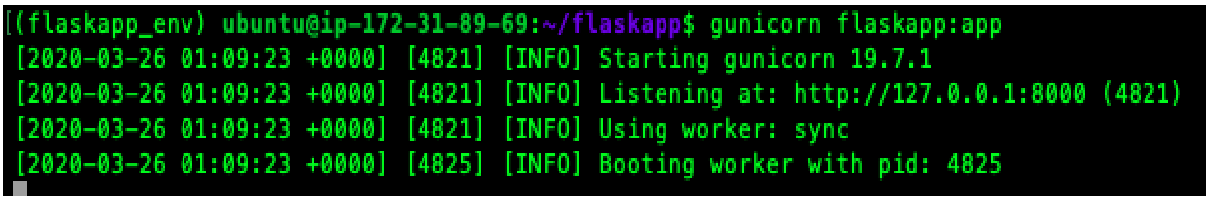
Screen shot for each docker file running separately can be found in **Screen shots/individual docker** folder

Also all the containerised services are kept in **Containers** folder to verify the container files like yml file, Docker file etc.

Job 5 :

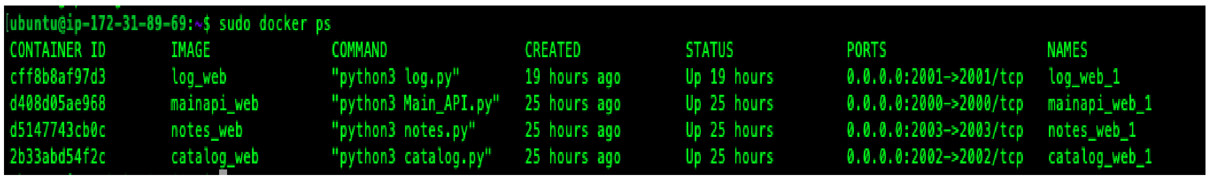
Deployment

As per the requirement specified in the document, docker was installed[2], microservices were containerised and deployed on ec2 instance. To host the website[1], nginx and gunicorn were used.



```
[(flaskapp_env) ubuntu@ip-172-31-89-69:~/flaskapp$ gunicorn flaskapp:app
 [2020-03-26 01:09:23 +0000] [4821] [INFO] Starting gunicorn 19.7.1
 [2020-03-26 01:09:23 +0000] [4821] [INFO] Listening at: http://127.0.0.1:8000 (4821)
 [2020-03-26 01:09:23 +0000] [4821] [INFO] Using worker: sync
 [2020-03-26 01:09:23 +0000] [4825] [INFO] Booting worker with pid: 4825
```

Figure 3 Hosting of website



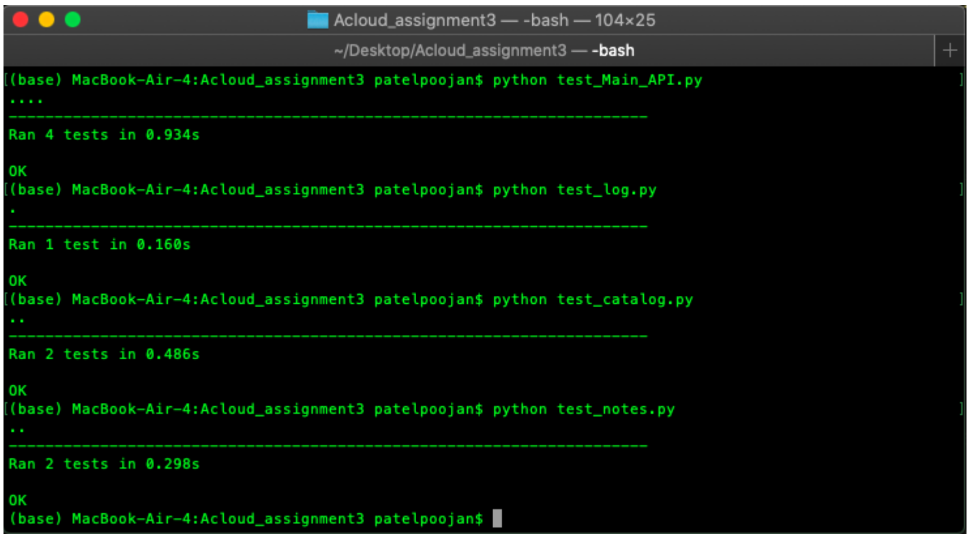
```
[ubuntu@ip-172-31-89-69:~$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND                CREATED             STATUS              PORTS                    NAMES
cff8b8af97d3        log_web             "python3 log.py"       19 hours ago        Up 19 hours         0.0.0.0:2001->2001/tcp   log_web_1
d408d05ae968        mainapi_web         "python3 Main_API.py"  25 hours ago        Up 25 hours         0.0.0.0:2000->2000/tcp   mainapi_web_1
d5147743cb0c        notes_web           "python3 notes.py"     25 hours ago        Up 25 hours         0.0.0.0:2003->2003/tcp   notes_web_1
2b33abd54f2c        catalog_web         "python3 catalog.py"   25 hours ago        Up 25 hours         0.0.0.0:2002->2002/tcp   catalog_web_1
```

Figure 4 Docker containers running

Job 6 :

Testing

API Testing [5] : APIs were tested using python testing framework "unittest". Different tests like empty string, keyword not present in database etc. were done.



```
[(base) MacBook-Air-4:Acloud_assignment3 patelpoojan$ python test_Main_API.py
....
----------------------------------------------------------------------
Ran 4 tests in 0.934s

OK
[(base) MacBook-Air-4:Acloud_assignment3 patelpoojan$ python test_log.py
.
----------------------------------------------------------------------
Ran 1 test in 0.160s

OK
[(base) MacBook-Air-4:Acloud_assignment3 patelpoojan$ python test_catalog.py
..
----------------------------------------------------------------------
Ran 2 tests in 0.486s

OK
[(base) MacBook-Air-4:Acloud_assignment3 patelpoojan$ python test_notes.py
..
----------------------------------------------------------------------
Ran 2 tests in 0.298s

OK
(base) MacBook-Air-4:Acloud_assignment3 patelpoojan$
```

Figure 5 Testing of all the components

Front-end Testing : As there were not more input field in front-end, manual testing was done. Search text box was set required field so empty string cannot be submitted and for note submission, blank note can be submitted but it is handled in the API and it will not added in notes file.

All the testing files can be found in **Testing** folder

References :

[1] Tran, C., 2020. *How To Deploy A Flask App On An AWS EC2 Instance*. [online] Chris Tran. Available at: <https://chrisdtran.com/2017/deploy-flask-on-ec2/> [Accessed 24 March 2020].

[2] Bogotobogo.com. 2020. *Docker Compose With Two Containers - Flask REST API Service Container And An Apache Server Container - 2020*. [online] Available at: <https://www.bogotobogo.com/DevOps/Docker/Docker-Compose-FlaskREST-Service-Container-and-Apache-Container.php> [Accessed 22 March 2020].

[3] “Welcome to the MongoDB Docs,” MongoDB Documentation. [Online].
Available: https://docs.mongodb.com/ [Accessed: 21-Mar-2020].

[4] “Offline Catalogs,” Project Gutenberg. [Online].
Available: http://www.gutenberg.org/wiki/Gutenberg:Offline_Catalogs [Accessed: 15-Mar-2020].

[5] “Unit Testing a Flask Application,” Patricks Software Blog. [Online].
Available: https://www.patricksoftwareblog.com/unit-testing-a-flask-application/ [Accessed: 24-Mar-2020].