1.1 MongoDB databases: Installation & Configuration

1.1.1 Introduction

Why to use database for this project?
By default Rasa chatbot stores message history only in system memory during operation time. Once the chatbot is restarted all the messages stored by memory tracker-store gets lost. To retain the conversation history even after at later times, Rasa framework allows use of tracker-store to use MongoDB database for storing conversations automatically.

What is MongoDB?
MongoDB database is a NoSQL database which stores data as collections and each collections consists of documents. Documents are equivalent of rows or tuples and collections equivalent to tables in relational databases. Therefore, MongoDB is a document-oriented database and schema-less structure of the document allows storing of unstructured data.

MongoDB by default runs on port number **27017**.

Why to use MongoDB for this project?
The reason for choosing MongoDB database with this project is that it is one of the supported by tracker-store in Rasa framework. It supports feature called “Change Stream” which allows the application listening for changes in databases to receive changes occurring at database at particular reference point. This feature is very handy while it comes to visualizing sentiments for user sent messages at real time.

1.1.2 Change Stream: MongoDB

Change Stream in MongoDB is a feature that allows MongoDB database as a real-time database. With change stream feature applied, changes (CRUD operations) happening in databases are propagated to applications which are listening for changes in that database. Change Stream avoids the need for applications to make queries to database in certain intervals repeatedly looking for changes in the database. Applications can subscribe to changes in database over a particular collection and even whole database.

Change Stream feature is only supported by MongoDB version 3.6 or later. Change Stream feature requires deployment of **replicaset. Replicaset** is cluster of MongoDB servers where one of the server acts as a primary server and the rest act as secondary server. Primary server is both readable and writable. However, secondary server is only readable. Secondary servers in replicaset hold replica of databases present on Primary server. More, precisely, secondary servers are just replicas of primary server. The purpose of those replica secondary servers is to serve as back up of data present on primary servers and support availability of data via fault tolerance.

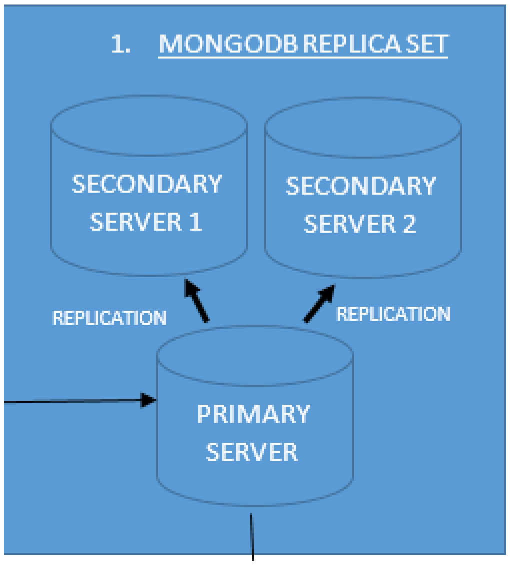


Figure 1 MongoDB replicaset example

[7] Whenever, data are changed on primary MongoDB server in replicaset, primary server maintains log of it inside operation log e.g. oplog. The secondary server reads this operation log and reflects the same changes in their databases which resembles with the database on primary server. The change stream is also propagated to those applications who listen for the changes.

1.1.3 Installation: MongoDB Compass

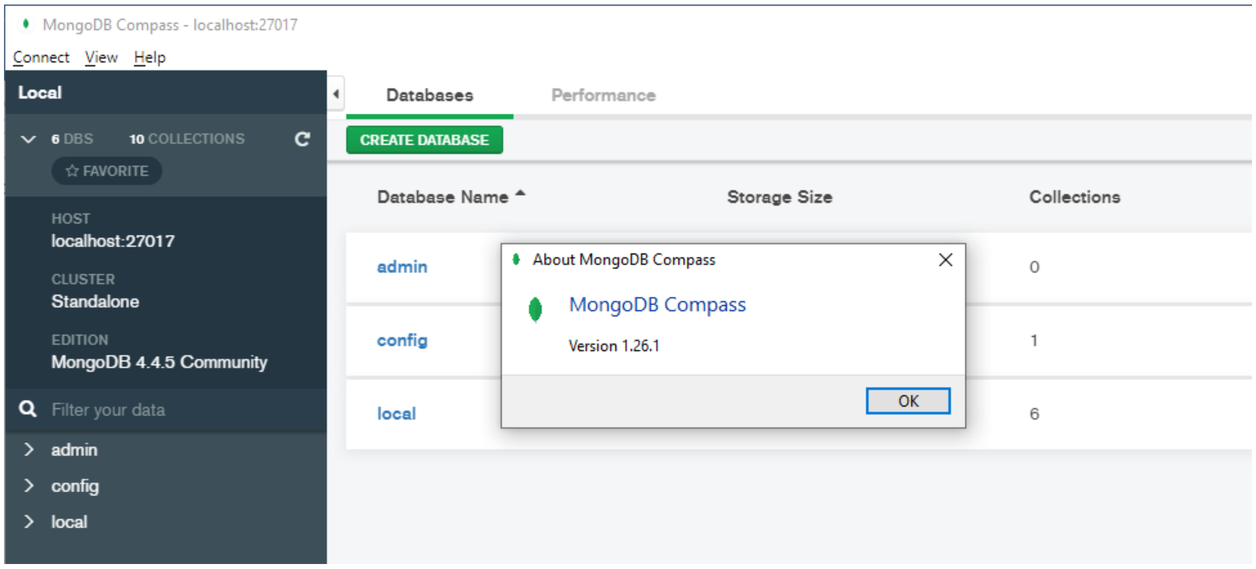
MongoDB Compass which is GUI along with underlying local MongoDB database has been used for the project. Main reason behind using MongoDB Compass is that it comes with the GUI that helps with the following:

- Visualizing the data in database
- Performing database operations easily (e.g. deleting documents, creating databases, creating aggregating pipelines)
- Visualizing performances etc.

This provides interaction with the database more convenient.
MongoDB Compass for Windows can be downloaded and installed using the following link.

https://www.mongodb.com/try/download/compass

Installation of MongoDB Compass is easy and straightforward. The version of MongoDB Compass and underlying MongoDB database software used for this project is shown below.



Database Name	Storage Size	Collections
admin		0
config		1
local		6

Figure 2 MongoDB Compass user interface

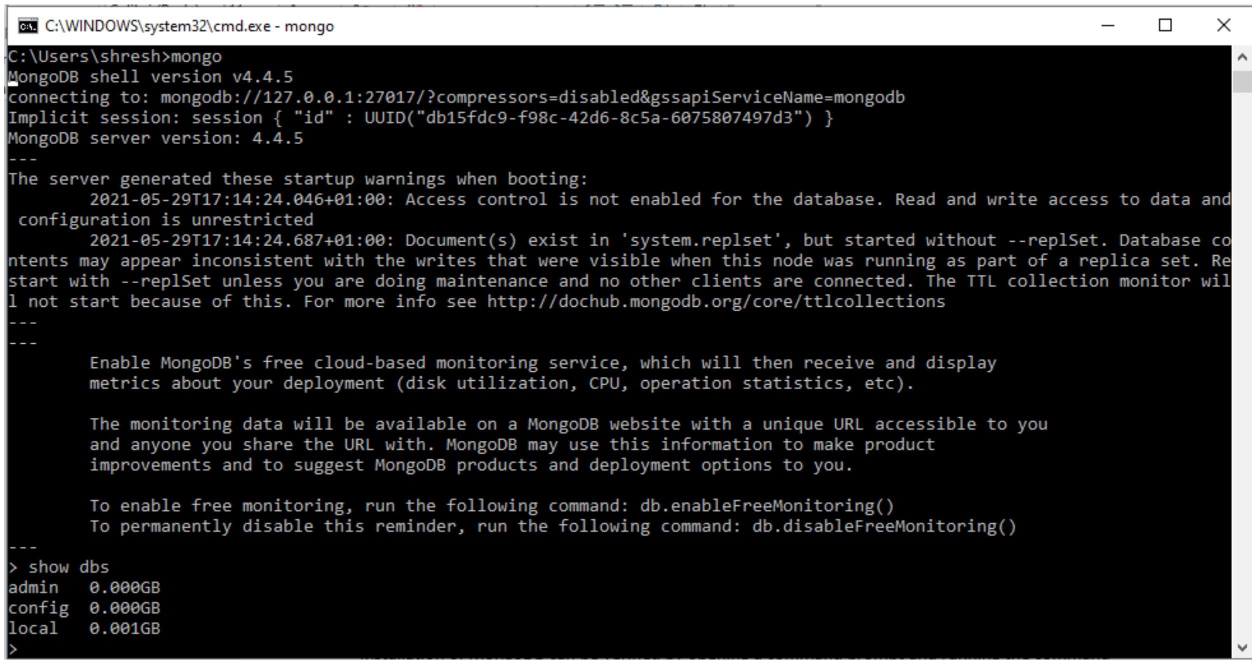
MongoDB version: 4.4.5

MongoDB Compass: 1.26.1

MongoDB is installed as standalone local installation on the PC used.

1.1.4 Verifying Installation: MongoDB

Installation of MongoDB can be verified by opening a command prompt and running the command 'mongo' command as shown below. This method can be used if path variable for environment variable was created during the installation of MongoDB compass.



```
C:\Users\shresh>mongo
MongoDB shell version v4.4.5
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("db15fdc9-f98c-42d6-8c5a-6075807497d3") }
MongoDB server version: 4.4.5
---
The server generated these startup warnings when booting:
        2021-05-29T17:14:24.046+01:00: Access control is not enabled for the database. Read and write access to data and
 configuration is unrestricted
        2021-05-29T17:14:24.687+01:00: Document(s) exist in 'system.replset', but started without --replSet. Database co
ntents may appear inconsistent with the writes that were visible when this node was running as part of a replica set. Re
start with --replSet unless you are doing maintenance and no other clients are connected. The TTL collection monitor wil
l not start because of this. For more info see http://dochub.mongodb.org/core/ttlcollections
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> show dbs
admin   0.000GB
config  0.000GB
local   0.001GB
>
```

Figure 3 verifying installation of MongoDB using 'mongo' command

If MongoDB is installed then we are connected to MongoDB database and we can run some commands as shown in the diagram above.

1.1.5 Configuring: MongoDB Replicaset

The replicaset shown on diagram above has been implemented for this project i.e. the cluster consists of

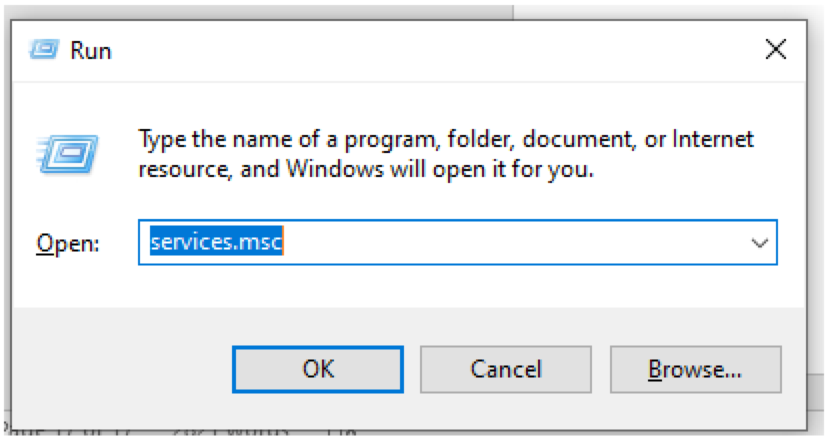
- 1 primary server
- 2 secondary servers

Following steps have been taken for configuring the replicaset of MongoDB databases.

1.1.5.1 Stopping MongoDB service

If MongoDB is running, the process needs to be sopped.

'services.msc' can be typed in Run dialog box on Windows and the MongoDB service can be disabled as shown below.

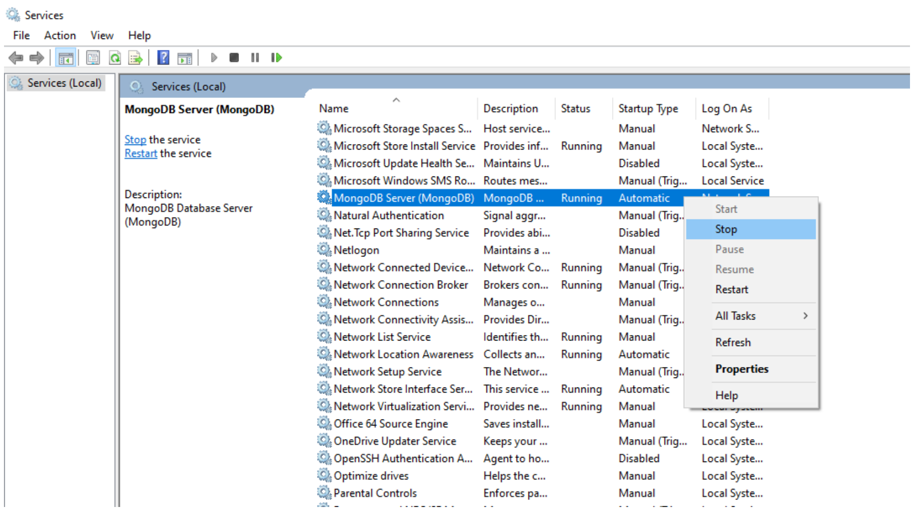


Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: services.msc

Figure 4 Opening Services window



Services (Local)

MongoDB Server (MongoDB)

Stop the service
Restart the service

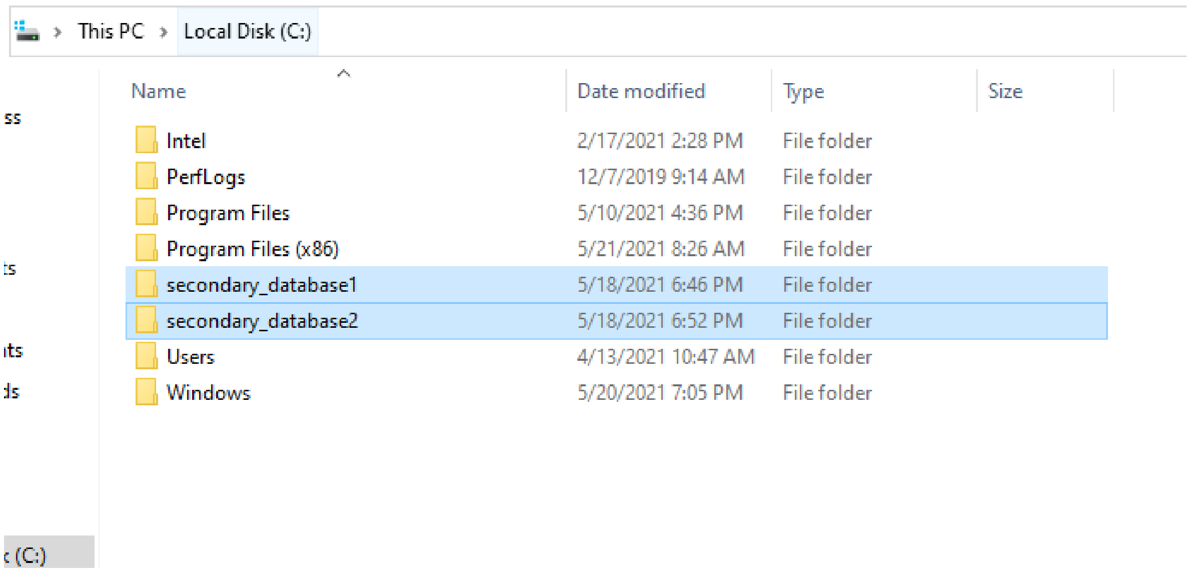
Description:
MongoDB Database Server (MongoDB)

Name	Description	Status	Startup Type	Log On As
Microsoft Storage Spaces S...	Host service...		Manual	Network S...
Microsoft Store Install Service	Provides inf...	Running	Manual	Local Syste...
Microsoft Update Health Se...	Maintains U...		Disabled	Local Syste...
Microsoft Windows SMS Ro...	Routes mes...		Manual (Trig...	Local Service
MongoDB Server (MongoDB)	MongoDB ...	Running	Automatic	
Natural Authentication	Signal aggr...		Manual (Trig...	
Net.Tcp Port Sharing Service	Provides abi...		Disabled	
Netlogon	Maintains a ...		Manual	
Network Connected Device...	Network Co...	Running	Manual (Trig...	
Network Connection Broker	Brokers con...	Running	Manual (Trig...	
Network Connections	Manages o...		Manual	
Network Connectivity Assis...	Provides Dir...		Manual (Trig...	
Network List Service	Identifies th...	Running	Manual	
Network Location Awareness	Collects an...	Running	Automatic	
Network Setup Service	The Networ...		Manual (Trig...	
Network Store Interface Ser...	This service ...	Running	Automatic	
Network Virtualization Servi...	Provides ne...	Running	Manual	
Office 64 Source Engine	Saves install...		Manual	Local Syste...
OneDrive Updater Service	Keeps your ...		Manual (Trig...	Local Syste...
OpenSSH Authentication A...	Agent to ho...		Disabled	Local Syste...
Optimize drives	Helps the c...		Manual	Local Syste...
Parental Controls	Enforces pa...		Manual	Local Syste...

Figure 5 Stopping MongoDB service

1.1.5.2 Defining root folders for Secondary MongoDB Servers

We aim to choose, MongoDB instance running at port number 27017 as a primary server. For creating two secondary servers root folders for those databases needs to be created separately as shown on the diagram below.



This PC › Local Disk (C:)

Name	Date modified	Type	Size
Intel	2/17/2021 2:28 PM	File folder	
PerfLogs	12/7/2019 9:14 AM	File folder	
Program Files	5/10/2021 4:36 PM	File folder	
Program Files (x86)	5/21/2021 8:26 AM	File folder	
secondary_database1	5/18/2021 6:46 PM	File folder	
secondary_database2	5/18/2021 6:52 PM	File folder	
Users	4/13/2021 10:47 AM	File folder	
Windows	5/20/2021 7:05 PM	File folder	

Figure 6 defining root folders for secondary databases

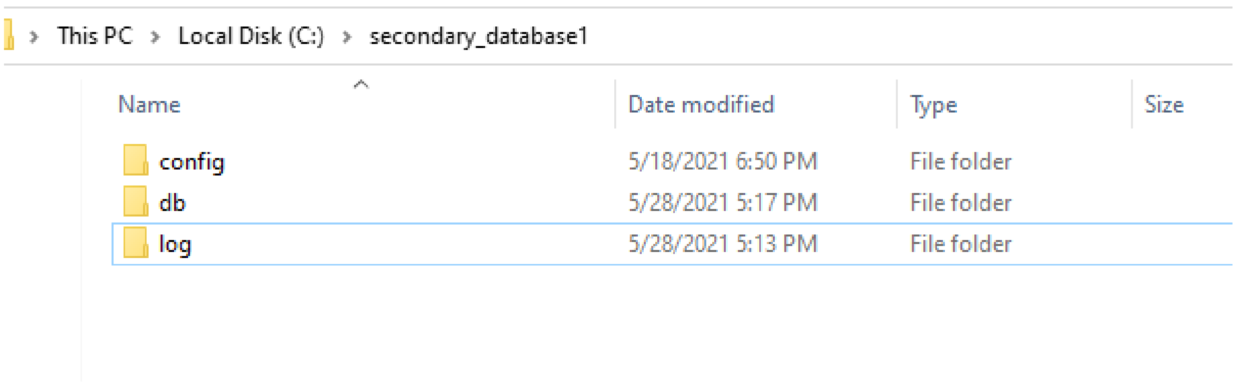
- Since 2 secondary databases are about to be used, two folders i.e. **secondary_database1** and **secondary_database2** are created in C drive.

1.1.5.3 Defining sub-folders: db, log and config folders for each secondary servers

Inside each root folders, the folders: db, config and log need to be created. Following are few details about the significance of those folders.

- db folder as a Databases for storing collection and documents and other related data
- log folder for storing information regarding different operations performed
- config folder for defining and storing configuration information about ports and database and log files

This can be done as shown below.



This PC › Local Disk (C:) › secondary_database1

Name	Date modified	Type	Size
config	5/18/2021 6:50 PM	File folder	
db	5/28/2021 5:17 PM	File folder	
log	5/28/2021 5:13 PM	File folder	

Figure 7 creating db, config and log folder for secondary database

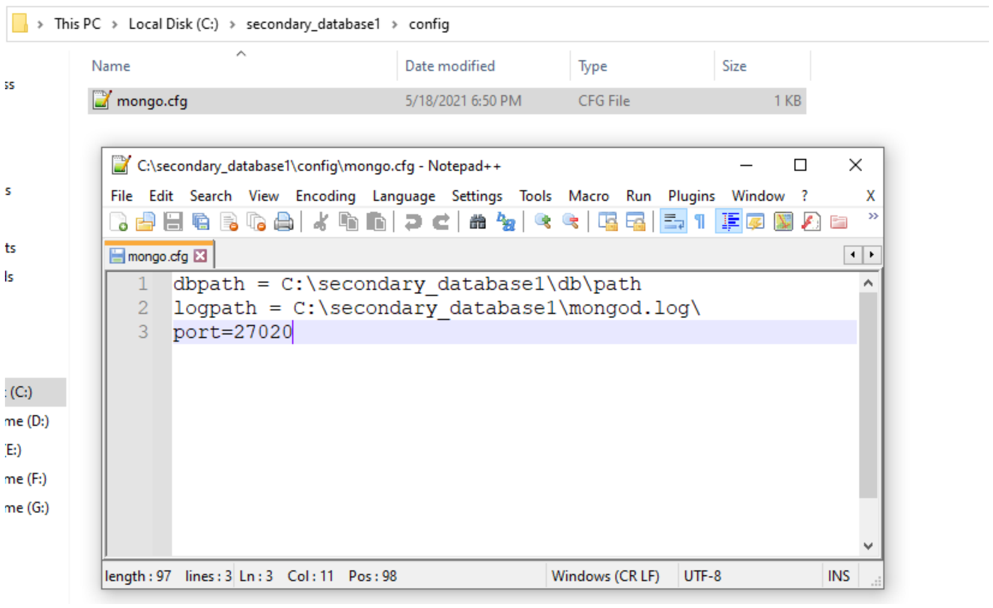
The same folders need to be created inside **secondary_database2** as well.

1.1.5.4 Creating 'mongo.cfg' file

Inside config folder, a configuration file called 'mongo.cfg' file needs to be created. This file is referred to as configuration file by the MongoDB server to which that root folder is assigned to during runtime. The 'mongo.cfg' file consists of following details:

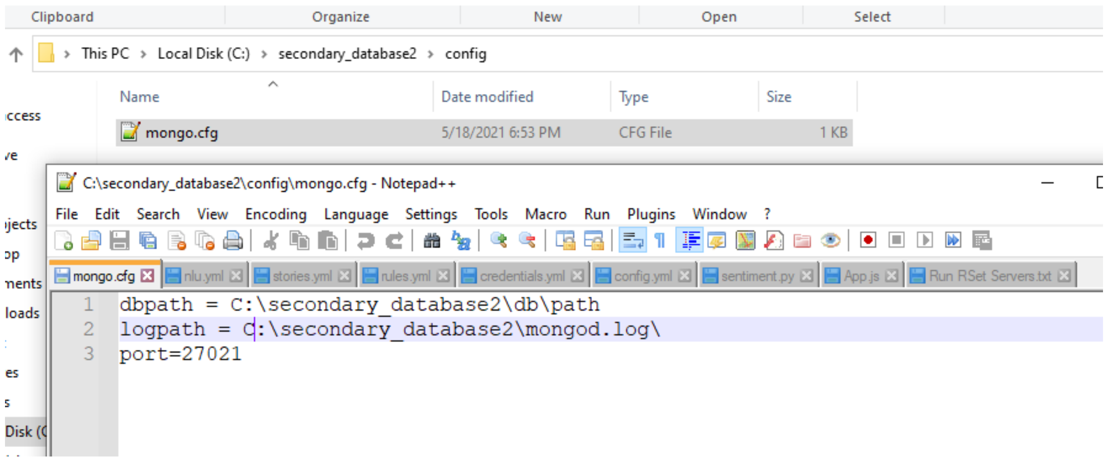
- dbpath : specifies path to the database where data are stored for that secondary server
- logpath: specifies path to the log folder where log generated are written during the operation for the secondary server
- port: specifies one of the registered port number on which that secondary server will run.

Following is the snapshot showing how it is done.


This PC › Local Disk (C:) › secondary_database1 › config
Name | Date modified | Type | Size
mongo.cfg | 5/18/2021 6:50 PM | CFG File | 1 KB
C:\secondary_database1\config\mongo.cfg - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
mongo.cfg
length : 97 lines : 3 Ln : 3 Col : 11 Pos : 98 Windows (CR LF) UTF-8 INS

```
dbpath = C:\secondary_database1\db\path
logpath = C:\secondary_database1\mongod.log\
port=27020
```

Figure 8 creating config file for secondary server 1

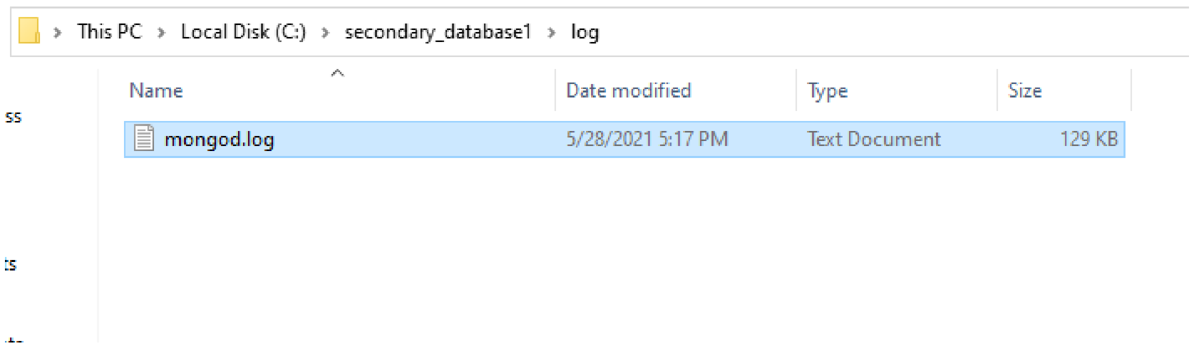

Clipboard Organize New Open Select
This PC › Local Disk (C:) › secondary_database2 › config
Name | Date modified | Type | Size
mongo.cfg | 5/18/2021 6:53 PM | CFG File | 1 KB
C:\secondary_database2\config\mongo.cfg - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
mongo.cfg nlu.yml stories.yml rules.yml credentials.yml config.yml sentiment.py App.js Run RSet Servers.txt

```
dbpath = C:\secondary_database2\db\path
logpath = C:\secondary_database2\mongod.log\
port=27021
```

Figure 9 creating config file for secondary server 2

1.1.5.5 Creating ‘mongod.log’ file

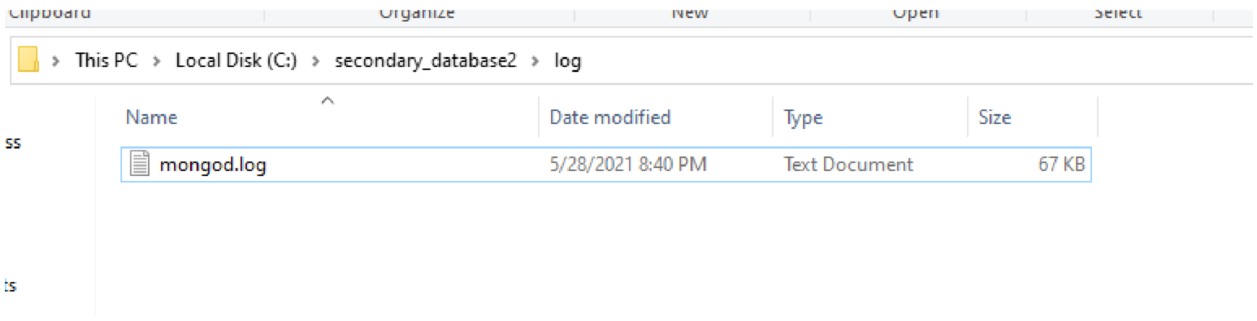
Inside ‘log’ folder of both root directories created, an empty log file named ‘mongod.log’ needs to be created. This file will be written with all the logs generated during server run later when MongoDB servers are started.



This PC › Local Disk (C:) › secondary_database1 › log

Name	Date modified	Type	Size
mongod.log	5/28/2021 5:17 PM	Text Document	129 KB

Figure 10 creating 'mongod.log' file inside log folder inside secondary database1 folder



This PC › Local Disk (C:) › secondary_database2 › log

Name	Date modified	Type	Size
mongod.log	5/28/2021 8:40 PM	Text Document	67 KB

Figure 11 creating 'mongod.log' file inside log folder inside secondary database2 folder

1.1.5.6 Creating and Running 3 instances of MongoDB

Now, three different instances of MongoDB needs to be created. One is for primary server that will run on port number 27017 and two for secondary servers as replica servers which will operate on port numbers 27020 and 27021 as stated inside ‘mongod.cfg’ files above.

Following 3 commands are run individually for creating and running 3 instances of MongoDB databases.

```
mongod --dbpath "C:\Program Files\MongoDB\Server\4.4\data" --logpath "C:\Program Files\MongoDB\Server\4.4\log\mongo.log" --port 27017 --storageEngine=wiredTiger --journal --replSet rSet
```

```
mongod --dbpath "C:\secondary_database1\db" --logpath "C:\secondary_database1\log\mongod.log" --port 27020 --storageEngine=wiredTiger --journal --replSet rSet
```

```
mongod --dbpath "C:\secondary_database2\db" --logpath "C:\secondary_database2\log\mongod.log" --port 27021 --storageEngine=wiredTiger --journal --replSet rSet
```

Each of the commands has same parameters, they can be described as follows:

- dbpath flag specifies path the database to be used by the instance of MongoDB being created.
- logpath flag specifies path to log file for writing log information by the MongoDB instance being created during running operations.
- port flag binds the instance of MongoDB being created to the port specified.
- storageEngine specifies as default storage engine to be used while storing the document and collection in database. The default storageEngine is wiredTiger for MongoDB.
- replSet flag indicates that those instances are not going to be started as standalone instances but rather as a part of replicaset called 'rSet' in this case. The name of the replicaset can be any string value.

To run these commands windows command prompt needs to be opened with administrative privilege.

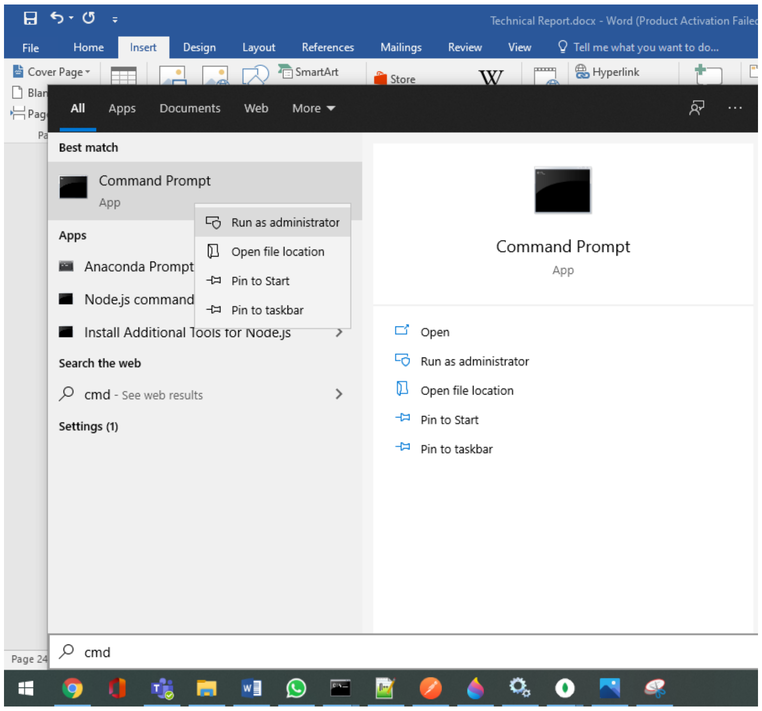
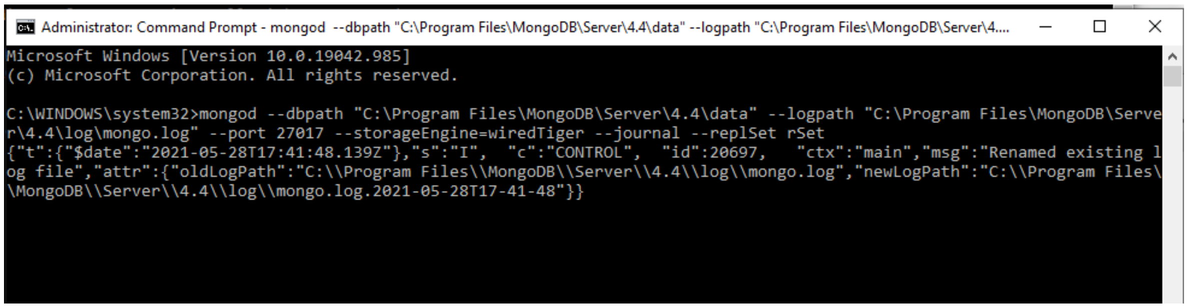


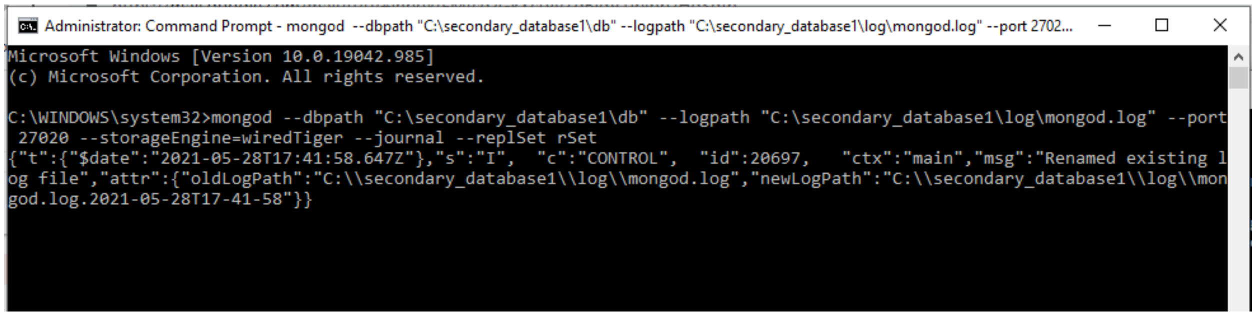
Figure 12 opening command prompt with administrative privilege

Those 3 different commands shown above can be run inside 3 different command prompts as shown below.



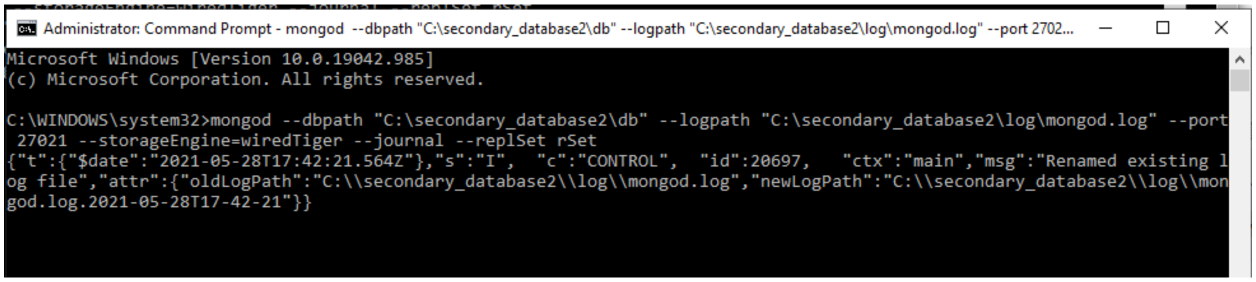
```
Administrator: Command Prompt - mongod  --dbpath "C:\Program Files\MongoDB\Server\4.4\data" --logpath "C:\Program Files\MongoDB\Server\4....
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>mongod --dbpath "C:\Program Files\MongoDB\Server\4.4\data" --logpath "C:\Program Files\MongoDB\Serve
r\4.4\log\mongo.log" --port 27017 --storageEngine=wiredTiger --journal --replSet rSet
{"t":{"$date":"2021-05-28T17:41:48.139Z"},"s":"I",  "c":"CONTROL",  "id":20697,   "ctx":"main","msg":"Renamed existing l
og file","attr":{"oldLogPath":"C:\\Program Files\\MongoDB\\Server\\4.4\\log\\mongo.log","newLogPath":"C:\\Program Files\
\MongoDB\\Server\\4.4\\log\\mongo.log.2021-05-28T17-41-48"}}
```

Figure 13 Starting MongoDB replicaset instance on port 27017



```
Administrator: Command Prompt - mongod  --dbpath "C:\secondary_database1\db" --logpath "C:\secondary_database1\log\mongod.log" --port 2702...
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>mongod --dbpath "C:\secondary_database1\db" --logpath "C:\secondary_database1\log\mongod.log" --port
 27020 --storageEngine=wiredTiger --journal --replSet rSet
{"t":{"$date":"2021-05-28T17:41:58.647Z"},"s":"I",  "c":"CONTROL",  "id":20697,   "ctx":"main","msg":"Renamed existing l
og file","attr":{"oldLogPath":"C:\\secondary_database1\\log\\mongod.log","newLogPath":"C:\\secondary_database1\\log\\mon
god.log.2021-05-28T17-41-58"}}
```

Figure 14 Starting MongoDB replicaset instance on port 27020



```
Administrator: Command Prompt - mongod  --dbpath "C:\secondary_database2\db" --logpath "C:\secondary_database2\log\mongod.log" --port 2702...
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>mongod --dbpath "C:\secondary_database2\db" --logpath "C:\secondary_database2\log\mongod.log" --port
 27021 --storageEngine=wiredTiger --journal --replSet rSet
{"t":{"$date":"2021-05-28T17:42:21.564Z"},"s":"I",  "c":"CONTROL",  "id":20697,   "ctx":"main","msg":"Renamed existing l
og file","attr":{"oldLogPath":"C:\\secondary_database2\\log\\mongod.log","newLogPath":"C:\\secondary_database2\\log\\mon
god.log.2021-05-28T17-42-21"}}
```

Figure 15 Starting MongoDB replicaset instance on port 27021

Now 3 different instances of MongoDB are running in our single localhost machine.

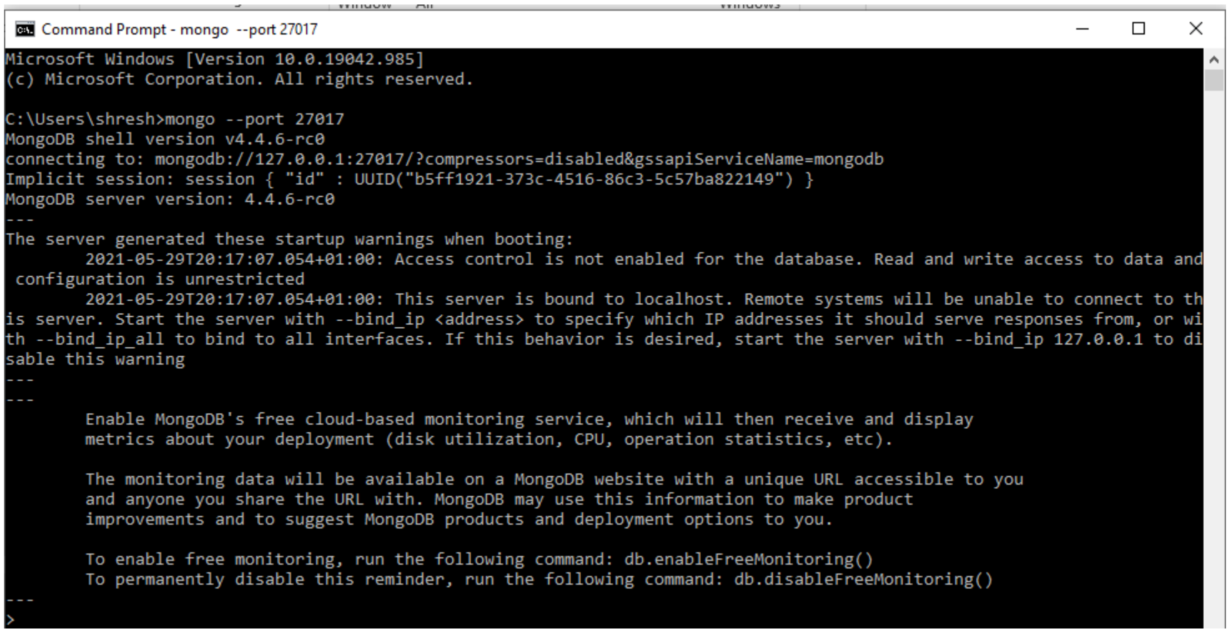
1.1.5.7 Configuring Replicaset: 'rSet'

Now to configure, replicaset 'rSet' with details about those running instances, we need to connect to MongoDB instance running on port number 27017 i.e. the server process running on default MongoDB port.

The following command can be used to connect to MongoDB instance running at particular port.

mongo --port [port number]

The diagram bellow shows running of the command.



```
Command Prompt - mongo  --port 27017
Microsoft Windows [Version 10.0.19042.985]
(c) Microsoft Corporation. All rights reserved.

C:\Users\shresh>mongo --port 27017
MongoDB shell version v4.4.6-rc0
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("b5ff1921-373c-4516-86c3-5c57ba822149") }
MongoDB server version: 4.4.6-rc0
---
The server generated these startup warnings when booting:
        2021-05-29T20:17:07.054+01:00: Access control is not enabled for the database. Read and write access to data and
 configuration is unrestricted
        2021-05-29T20:17:07.054+01:00: This server is bound to localhost. Remote systems will be unable to connect to th
is server. Start the server with --bind_ip <address> to specify which IP addresses it should serve responses from, or wi
th --bind_ip_all to bind to all interfaces. If this behavior is desired, start the server with --bind_ip 127.0.0.1 to di
sable this warning
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
>
```

Figure 16 connecting to MongoDB instance running on port 27017

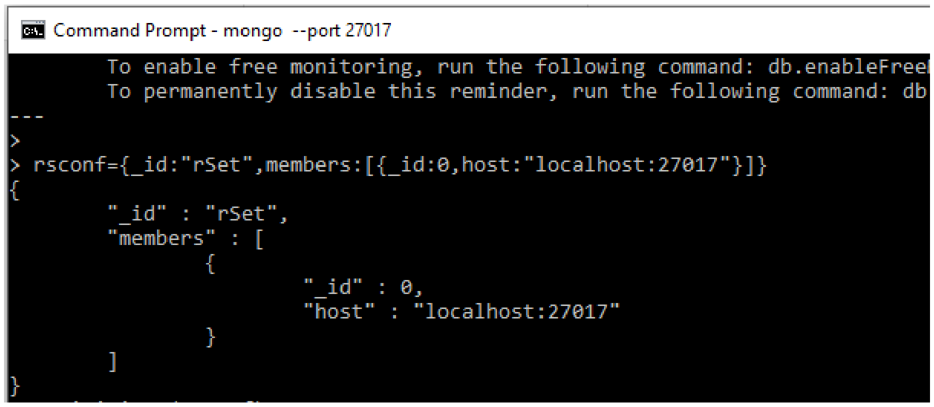
Now, replicaset configurations needs to be written into replicaset variable 'rsconf' with replicaset ID as the name of replicaset we specified while running those 3 different instances of MongoDB. The following command is used for defining replicaset variable.

rsconf={_id:"rSet",members:[{_id:0,host:"localhost:27017"}]}

where,

- 'rsconf' is a dictionary variable which holds replicaset information
- id holds the name for the replicaset
- host specifies the domain along with port number for primary server

The command is executed to produce the output as shown below.



```
Command Prompt - mongo  --port 27017
        To enable free monitoring, run the following command: db.enableFree
        To permanently disable this reminder, run the following command: db
---
>
> rsconf={_id:"rSet",members:[{_id:0,host:"localhost:27017"}]}
{
        "_id" : "rSet",
        "members" : [
                {
                        "_id" : 0,
                        "host" : "localhost:27017"
                }
        ]
}
```

Figure 17 writing rsconf variable

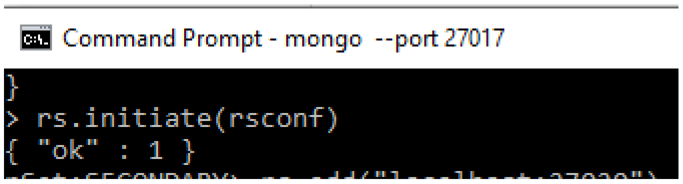
Next, we need to initiate or activate the replicaset we just created using the following command.

rs.initiate(rsconf)

where,

- rs.initiate() is a function that takes replicaset configuration variable as input and activates the replicaset.

The command is executed to give following output.



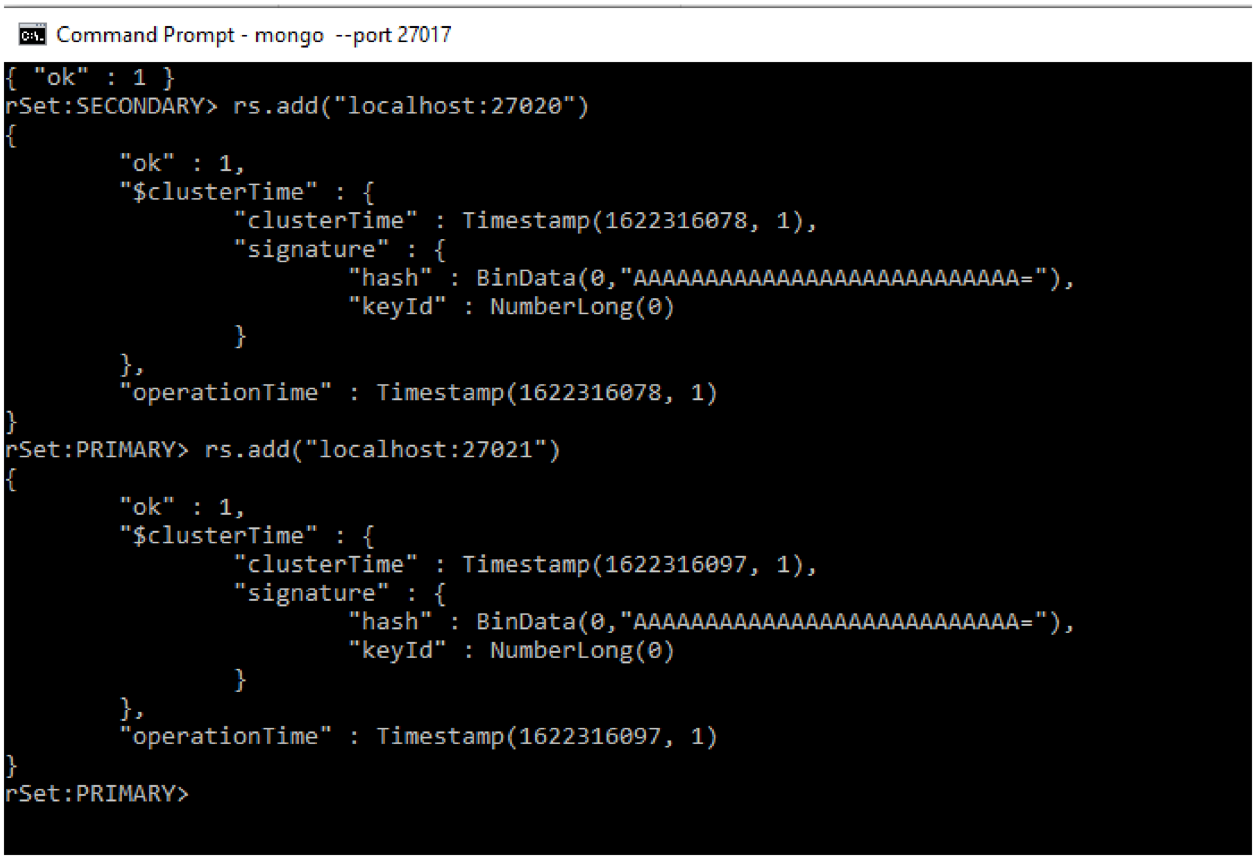
Command Prompt - mongo --port 27017

```
}
> rs.initiate(rsconf)
{ "ok" : 1 }
```

Figure 18 initiating replcaset

Finally, we need to add two other instances of MongoDB databases to the same replcaset configuration variable using the command below.

rs.add("localhost:27020")
rs.add("localhost:27021")
These command adds those 2 instances to the replicaset individually.
The command is executed and following output is obtained.



Command Prompt - mongo --port 27017

```
{ "ok" : 1 }
rSet:SECONDARY> rs.add("localhost:27020")
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1622316078, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1622316078, 1)
}
rSet:PRIMARY> rs.add("localhost:27021")
{
        "ok" : 1,
        "$clusterTime" : {
                "clusterTime" : Timestamp(1622316097, 1),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        },
        "operationTime" : Timestamp(1622316097, 1)
}
rSet:PRIMARY>
```

Figure 19 Adding remaining instances to the replicast

That is all we need to do while configuring the replicaset of different MongoDB instances.

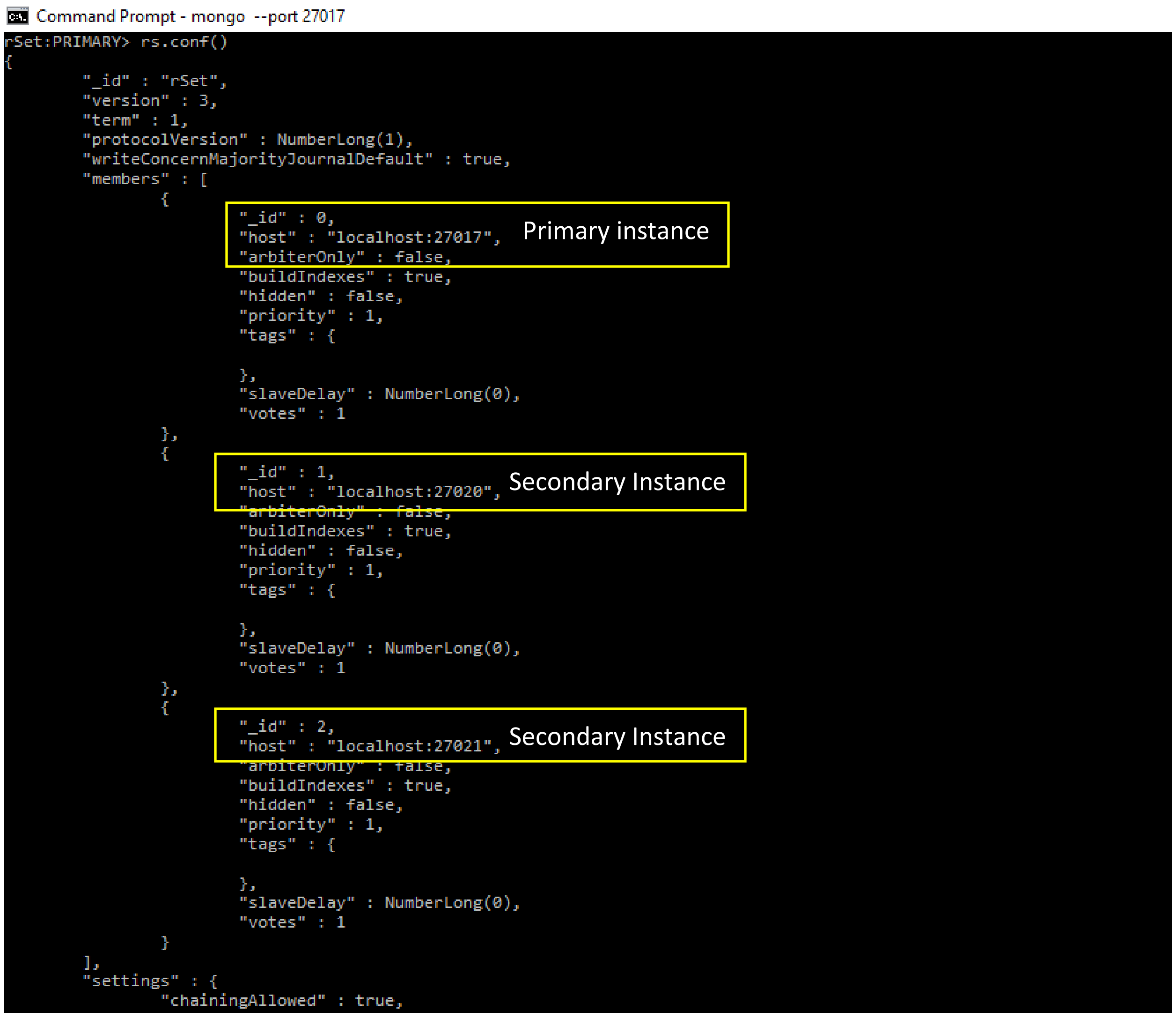
Note: All those servers need to be up and running for the project to work while we turn on every system components.

1.1.5.8 Checking replicaset configuration

Following command can be run into the same command prompt to view configuration written to replicaset variable.

rs.conf()

The command displays the following output.



```
rSet:PRIMARY> rs.conf()
{
        "_id" : "rSet",
        "version" : 3,
        "term" : 1,
        "protocolVersion" : NumberLong(1),
        "writeConcernMajorityJournalDefault" : true,
        "members" : [
                {
                        "_id" : 0,
                        "host" : "localhost:27017",
                        "arbiterOnly" : false,
                        "buildIndexes" : true,
                        "hidden" : false,
                        "priority" : 1,
                        "tags" : {

                        },
                        "slaveDelay" : NumberLong(0),
                        "votes" : 1
                },
                {
                        "_id" : 1,
                        "host" : "localhost:27020",
                        "arbiterOnly" : false,
                        "buildIndexes" : true,
                        "hidden" : false,
                        "priority" : 1,
                        "tags" : {

                        },
                        "slaveDelay" : NumberLong(0),
                        "votes" : 1
                },
                {
                        "_id" : 2,
                        "host" : "localhost:27021",
                        "arbiterOnly" : false,
                        "buildIndexes" : true,
                        "hidden" : false,
                        "priority" : 1,
                        "tags" : {

                        },
                        "slaveDelay" : NumberLong(0),
                        "votes" : 1
                }
        ],
        "settings" : {
                "chainingAllowed" : true,
```

Figure 20 checking replicaset configuration

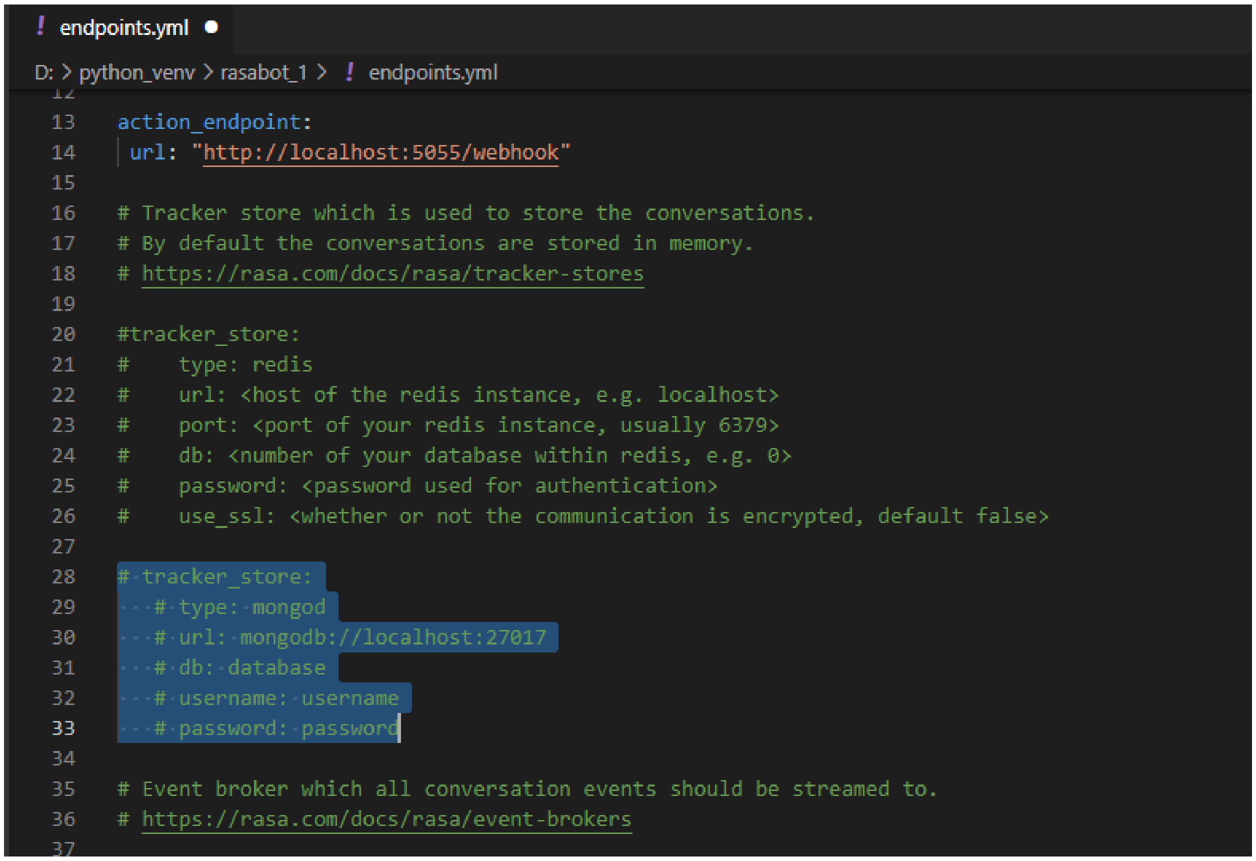
Since each replicaset can have only one Primary server at most and the MongoDB prompt on current command prompt shows the label "PRIMARY" which indicates that the MongoDB instance running on port 27017 is the primary server and hence rest of the two instances become secondary servers automatically.

1.1.6 Defining MongoDB Trackerstore inside Rasa chatbot

Now MongoDB database is installed and running. Now we need to define MongoDB Trackerstore in ‘endpoints.yml’ which is present inside root directory of Rasa chatbot we created earlier. Following snapshots displays required information changes while defining MongoDB Trackerstore. Following details need to be supplied while defining MongoDB Trackerstore.

- ‘type’ specifying driver for the database used which is ‘mongd’ in this case
- ‘url’ specifying complete IP of the primary MongoDB server
- ‘db’ specifying name of the database. If database with supplied name does not already exists in database then it will be created automatically.
- ‘username’ specifying username that can be used for logging into the database
- ‘password’ specifying password as login credential for logging into the database

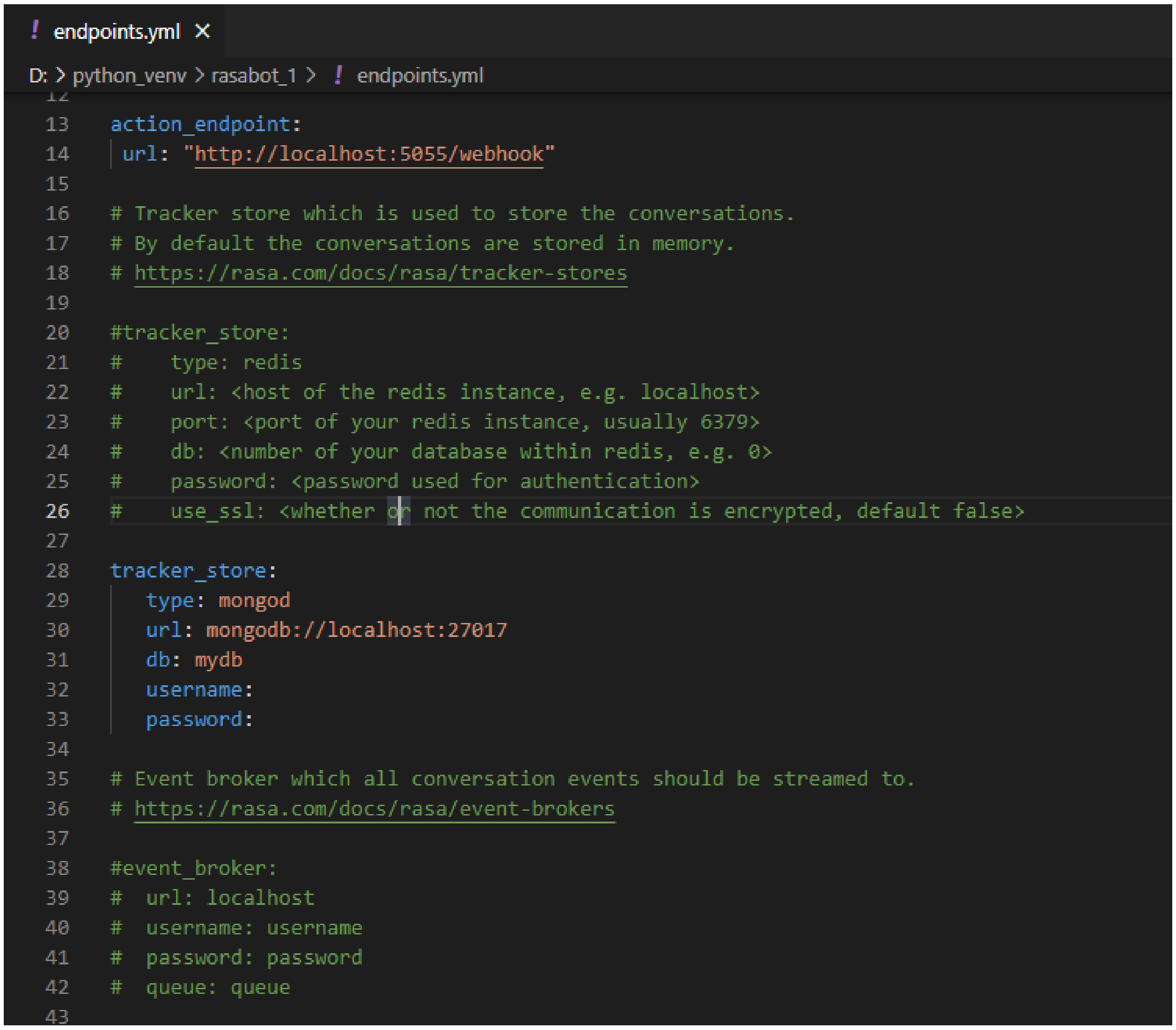
After locating ’endpoints.yml’ file, the file is opened on Visual Studion Code. The code requiring modification is selected as shown below.



```
action_endpoint:
  url: "http://localhost:5055/webhook"

# Tracker store which is used to store the conversations.
# By default the conversations are stored in memory.
# https://rasa.com/docs/rasa/tracker-stores

#tracker_store:
#    type: redis
#    url: <host of the redis instance, e.g. localhost>
#    port: <port of your redis instance, usually 6379>
#    db: <number of your database within redis, e.g. 0>
#    password: <password used for authentication>
#    use_ssl: <whether or not the communication is encrypted, default false>

# tracker_store:
   # type: mongod
   # url: mongodb://localhost:27017
   # db: database
   # username: username
   # password: password

# Event broker which all conversation events should be streamed to.
# https://rasa.com/docs/rasa/event-brokers
```

Figure 21 highlighting the code to be modified

- while uncommenting the code the resulting indentation should be preserved and necessary details should be supplied
- CTRL + K + U could be pressed simultaneously to uncomment those lines.



endpoints.yml

D: > python_venv > rasabot_1 > endpoints.yml

```yaml
action_endpoint:
 url: "http://localhost:5055/webhook"

# Tracker store which is used to store the conversations.
# By default the conversations are stored in memory.
# https://rasa.com/docs/rasa/tracker-stores

#tracker_store:
#    type: redis
#    url: <host of the redis instance, e.g. localhost>
#    port: <port of your redis instance, usually 6379>
#    db: <number of your database within redis, e.g. 0>
#    password: <password used for authentication>
#    use_ssl: <whether or not the communication is encrypted, default false>

tracker_store:
   type: mongod
   url: mongodb://localhost:27017
   db: mydb
   username:
   password:

# Event broker which all conversation events should be streamed to.
# https://rasa.com/docs/rasa/event-brokers

#event_broker:
#  url: localhost
#  username: username
#  password: password
#  queue: queue
```

Figure 22 Updating MongoDB TrackerStore in endpoints.yml file

- url defined is the IP address of the primary MongoDB server we created earlier.
- db defined is 'mydb'.
- username and password is not used hence left as blank

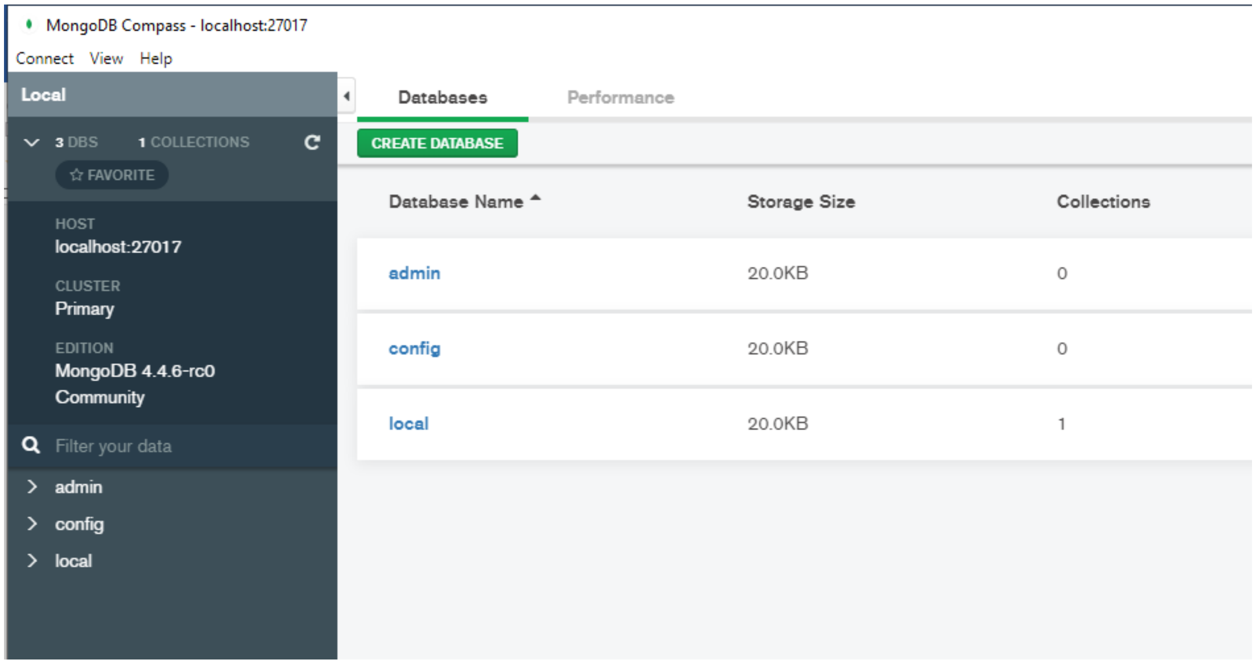
The file is saved after editing.

1.1.7 Verifying conversation being stored in MongoDB

Pre-condition for this step to work correctly is we need to have chatbot already running with endpoints APIs enabled, MongoDB also running and endpoint already configured with MongoDB Trackerstore. Next, we need to verify that the MongoDB Trackersotre is storing conversation inside primary MongoDB server.

However, since we still have not developed any UI for interacting with Rasa chatbot we interact wth Rasa chatbot using Postman to communicate as we did earlier to verify two way communication between the chatbot and the client.

MongoDB Compass is connected to MongoDB instance running on port 27017.



HOST
localhost:27017
CLUSTER
Primary
EDITION
MongoDB 4.4.6-rc0
Community

Database Name	Storage Size	Collections
admin	20.0KB	0
config	20.0KB	0
local	20.0KB	1

Figure 23 MongoDB compass connected to Primary MongoDB server

- The Compass is connected to MongoDB server on 27017, the IP which we defined on ‘endpoints.yml’ file.

There is no database ‘mydb’ created yet.