Documentation

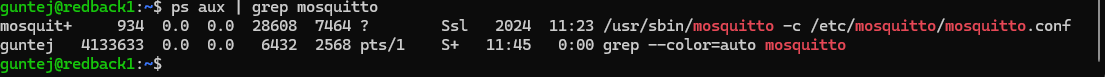
* Ls redback-data-warehouse
* Cd redback-data-warehouse
* Docker ps
* Cd “MongoDB Connection”
* Cd “Project1”
* docker-compose-up –build
* docker-compose up --build -d
* cd "MongoDB Connection/Project1"
* docker –version
* nano docker-compose.yaml
* docker pull mongo
* docker ps -a
* docker inspect mongodb-container
* docker stop mongodb-container
* docker rm mongodb-container
* ps aux | grep mosquito
* mosquitto\_sub -h mqtt.example.address -p 1883 -t "#" -v
* mosquitto\_sub -h localhost -p 1883 -t "#" -v
* mosquitto\_pub -h localhost -p 1883 -t "bike/000001/incline/control" -m "increase"
* mosquitto\_sub -h localhost -p 1883 -t "bike/#" -v
* mosquitto\_pub -h localhost -p 1883 -t "bike/000001/incline/control" -m "increase"
* bike -> MQTT -> HTTP request -> MongoDB API -> MongoDB

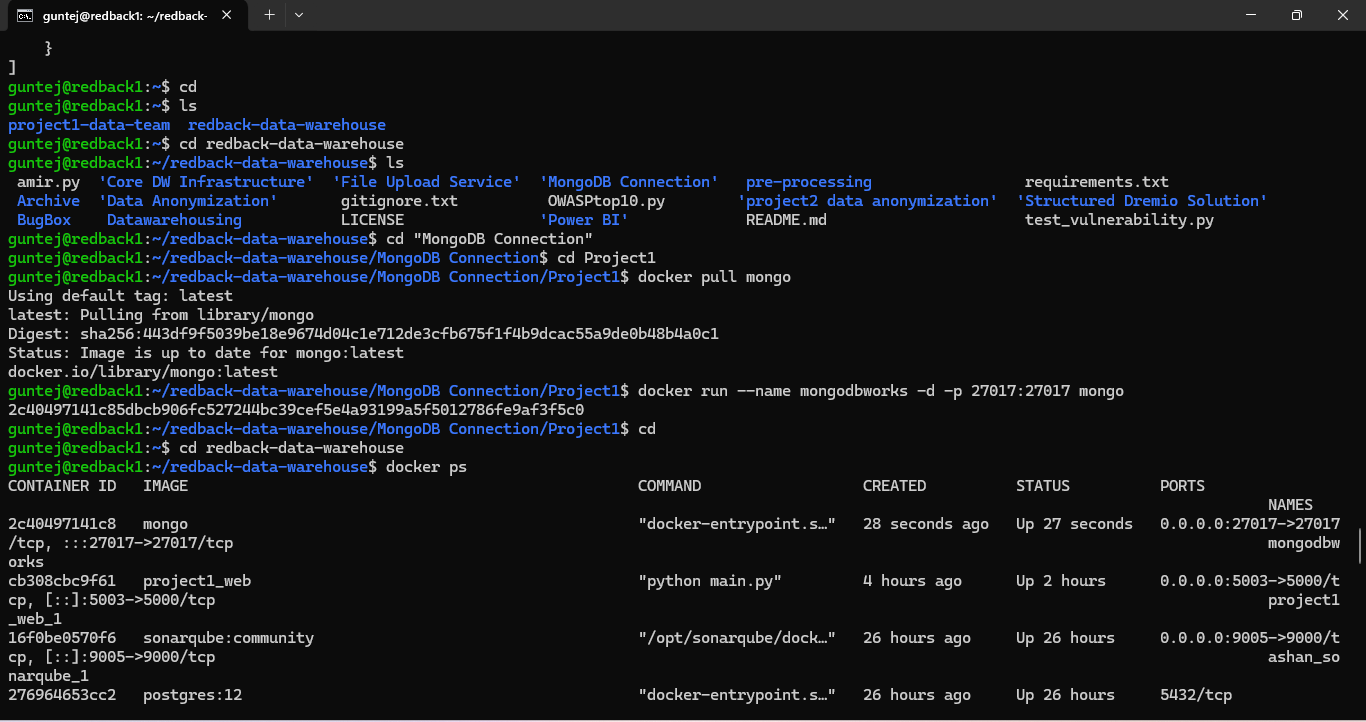
Important information:

Some of the things I have do are setup a mqtt broker. Good thing is at we have a mqtt server setup on the vm (esclipse mosquito) I’ve been playing around with that and seeing what’s setup in the vm.

* ps aux | grep mosquito

This command allows me to run mosquito on the vm. This pic shows my work.



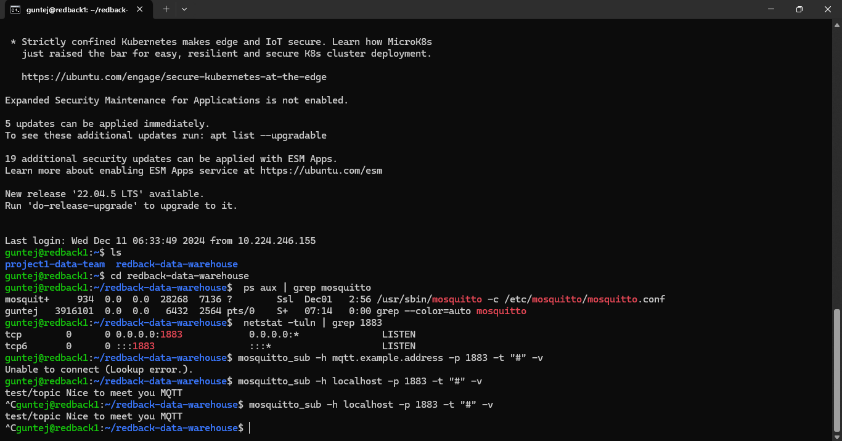


Another thing off course is setting up a mongodb database. ( I got the container up running that Ben d did previously) however upon further inspection it’s not really a mongodb container it’s more of a python web application container. (like the sole purpose…. Is not for mongo)

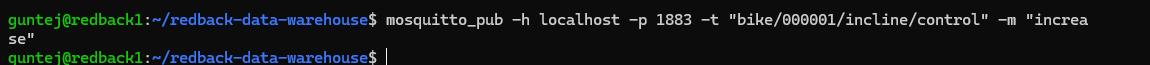
I just used these commands that are in the pic and we got a mongo running in the container.

Since the MQTT server is already set up, I am able to focus on creating the MQTT client to subscribe to topics and store messages in MongoDB.

I have tried to essentially trial a publish and subscribe kind of method running in two tabs.(This is an example of it.) (Ctrl z to stop. Or ctrl c)



With the help from project 1 team they’ve given me naming conventions on how the mqtt stream should follow. bike/{DEVICE\_ID}/[topic name]/[sub-topic]

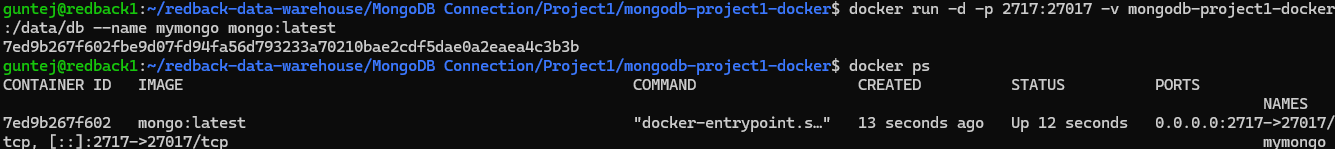


A screen shot of a computer code

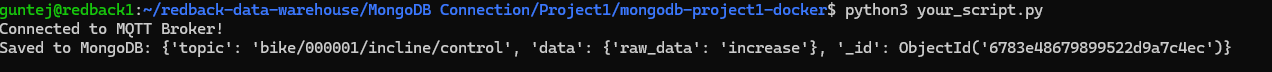
Description automatically generated

This was a trial run using their conventions and it worked.

Making a directory inside the root folder.

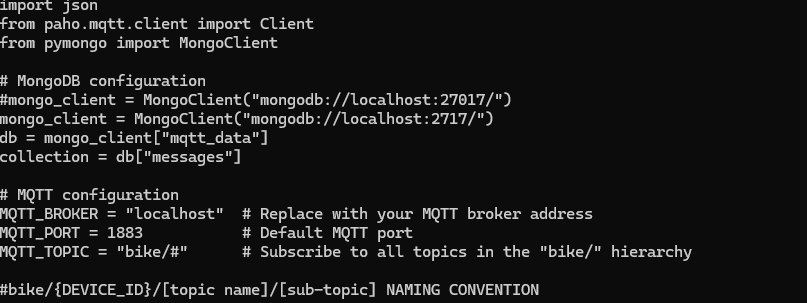


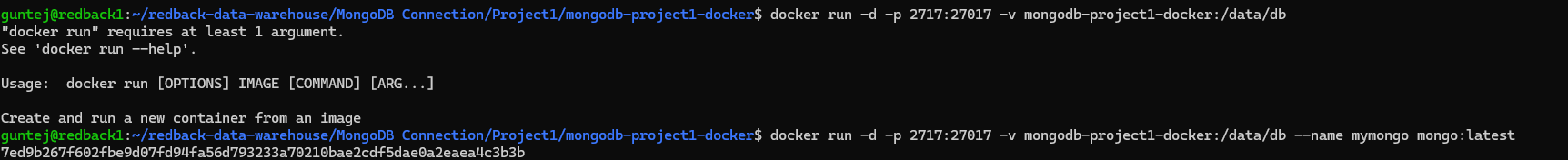
Just had to make some quick updates regarding the mongo.



This means the script has been successfully connected to MongoDB and that the test messaged was also saved correctly.

Without this script not possible

* I needed 3 tabs open to run the final test at the end. The key thing is that they are all under the same file.
* (guntej@redback1:~/redback-data-warehouse/MongoDB Connection/Project1/mongodb-project1-docker )
* This above in brackets is what I used for the 3 tabs.
* One tab I had to do nano your script.py and add the python script in the file in this nano file.
* This was all used in the nano script.
* Then run the script as I added above python3 your\_script.py and we received the output showing us we successfully ran it.
* Other tab was simply used to publish messages. An example I have given in the first page (mosquitto\_pub -h localhost -p 1883 -t "bike/000001/incline/control" -m "increase")
* Extremely important part of this task as this ensures you have done it successfully.
* The 3rd tab is optional but I recommend it so the mongodb database is open and running.



* These commands ensured that the MongoDB is running.
* With the success of previous 2 steps this ensure data flowing into mongodb database. I gave some updated steps on how I fixed up the mongodb database.