Implementation tutorial

Prerequisites:

- PC with Linux
- Internet connection to download dependencies
- The latest version of the project
- A test application
- Using PyCharm is recommended
- Know the Python language

To install the tools:

- Docker: https://docs.docker.com/install/linux/docker-ce/ubuntu/
- Docker Compose : https://docs.docker.com/compose/install/
- MongoDB Charts : https://docs.mongodb.com/charts/current/installation/
    - This tool does not work on Chrome. Use Firefox.

Get Started :

1. Open a terminal (you will have to open several terminals)

2. To install Python dependencies several solutions are possible.
    a. Pybuilder
    b. Travis
    c. Manually
    d. With PyCharm (it is done automatically)

To be sure that it will work for this demonstration it is recommended to do like this:

In a terminal navigate to the root of the project.
Execute the command:

pip install -r requirements.txt

This corresponds to the manual method.

3. Set up the database:

To do this, please configure the **docker-compose.yml** file located in the **/db** folder. You will be able to modify the database port (if not necessary, please leave port *27017* by default). Then the identifiers (*root* and *password*).

Once configured, navigate to the directory where this file is located and run:

docker-compose build
docker-compose up

Now the MongoDB database is ready to be used. You can verify it at localhost:8081 which correspond to Mongo Express interface of your local database. See more: https://hub.docker.com/_/mongo-express

4. Configure your monitoring part:

The most important part is monitoring. To do this, you need to know how to access information from your target application such as the percentage of CPU usage. This is possible with APIs.

For a docker environment you can refer to the existing implementation in the **docker_monitoring.py** file.

You must create a new file in which you create a class inherited from the *Monitoring* class of the **Monitoring.py** file.

Follow the structure of the existing docker_monitoring.py class to create your own algorithm.

At the end of your monitoring loop you should insert the data into the database using the *database_insertion(data)* method of *Monitoring* class.

You must ensure that the information is easily usable. For this purpose, it is preferable to insert information as JSON as MongoDB allows.

5. Create your analysis, planning and execution algorithms:

For each module (analysis, planning and execution) you must implement a subclass of each abstract method. To do that please follow the structure of the existing algorithms and create your own classes. Try to follow the observer method to use the 3 modules together. See more: https://refactoring.guru/design-patterns/observer/python/example

There is one point you should always do the same: The analysis have to get the information from the database periodically.

6. Run the test application:

Run your test app before starting the MAPE loop.

7. Run your MAPE loop:

You must run the monitoring part independently. The latter must retrieve the information periodically and insert it into the database.

Then run the rest of the modules with a hand file that performs the necessary functions you have defined. (You can view the file **dockercompose_autoscale.py)**

To see the changes, you can use MongoDB Charts. This tool allows you to connect directly to the database and create graphs using the stored data.

See more :

- Install : https://docs.mongodb.com/charts/current/installation/
- Use : https://docs.mongodb.com/charts/current/dashboards/