

EASE MAPE System

User manual

TABLES OF CONTENTS

TABLES OF CONTENTS	0
INTRODUCTION	1
Framework components	1
PROJECT DESCRIPTIONS	2
Project hierarchy	2
Imports requirements	2
Files descriptions	3
.env	3
/src/ease/mape/main.py	3
/src/ease/mape/monitoring/monitoring.py	3
/src/ease/mape/monitoring/average_cpu.py	3
/src/ease/mape/analysis/threshold_analysis.py	3
/src/ease/mape/analysis/linear_model_analysis.py	3
/src/ease/mape/planning/planning.py	3
/src/ease/mape/execution/execution.py	3
Class diagram	4
HOW TO RUN EXISTING MAPE LOOP	5
Docker client	5
Before starting :	5
Get started :	5
Debugging :	5
BUILD YOUR OWN IMPLEMENTATION	6
Monitoring	6
Analysis	7
Planning	7
Execution	7
Main	7
ERRORS YOU MAY ENCOUNTER	8

INTRODUCTION

The framework is developed in Python 3.6

Only Docker deployment is available in the system for the moment, and it only can be used with MongoDB NoSQL database. But you can develop your own deployment for custom clients.

Framework components

1. Monitoring

Monitoring needs to be connected to a docker client (like docker environment) and a database. It is supposed to get different data from the client like CPU %, memory %, disk I/O and network throughput of components, and store them into the NoSQL database in JSON document format. It can also get and store current date and for example the number of containers running on the client.

The main point is that the monitoring needs to run independently of the other modules. Because if something happens in other parts of the system the monitoring should continue to get information from the client.

2. Analysis (*Threshold analysis*)

Analysis needs to be connected to the database because he will periodically read the new information stored. Because of the threshold-based, the analysis module should concentrate his analyze on the CPU and compare it with the threshold of our choice. It should return a list of information to the planning module.

Example :

- return 1 if CPU > upper_threshold
- return 2 if CPU < lower_threshold
- return 0 in other case

3. Planning

Planning is a bit similar to the analysis but he only has to get information from analysis and plan the adaptive action. For example, add or remove containers in the case of Docker monitoring. And then send the decision to the execution module.

4. Execution

The execution module receives decisions from the planning module and execute the decision. He has to use the corresponding API and make a break in the MAPE loop until the system stability is reached.

Analysis, Planning and Execution need to run together.

PROJECT DESCRIPTIONS

Project hierarchy

* `__init__.py` files are just used to consider the folder as a package in the python project.

** `/src/old` folder is excluded

*** `/test` folder is empty

EASE MAPE System

 `.env`

/src

/ease/mape


 `main.py`

/monitoring

 `monitoring.py`

 `average_cpu.py`

/analysis

 `threshold_analysis.py`

 `linear_model_analysis.py`

/planning

 `planning.py`

/execution

 `execution.py`

Imports requirements

(listed in the **requirements.txt** file)

pymongo MongoClient

dotenv load_dotenv

abc ABC, abstractmethod

docker

time

os

sys

datetime

Files descriptions

`.env`

This text file set some environment variable. It can be used as a configuration file for maybe, the database URI, path or threshold of your choice.

`/src/ease/mape/main.py`

This python file is the main program to run **Analysis**, **Planning** and **Execution** module. It instantiate 3 objects and begins a while True loop to execute each function of the **-APE loop**.

`/src/ease/mape/monitoring/monitoring.py`

This file contain one abstract class **Monitoring**. In this class we have different abstract methods to implement. There is also a **DockerMonitoring** class for a docker client.

`/src/ease/mape/monitoring/average_cpu.py`

This is a try to have a average value for CPU usage because the CPU usage fluctuates.

`/src/ease/mape/analysis/threshold_analysis.py`

This is one of the different analysis files. This one is about threshold analysis. There is one abstract class with one abstract method. Same as monitoring there is the docker implementation.

`/src/ease/mape/analysis/linear_model_analysis.py`

Not yet implemented

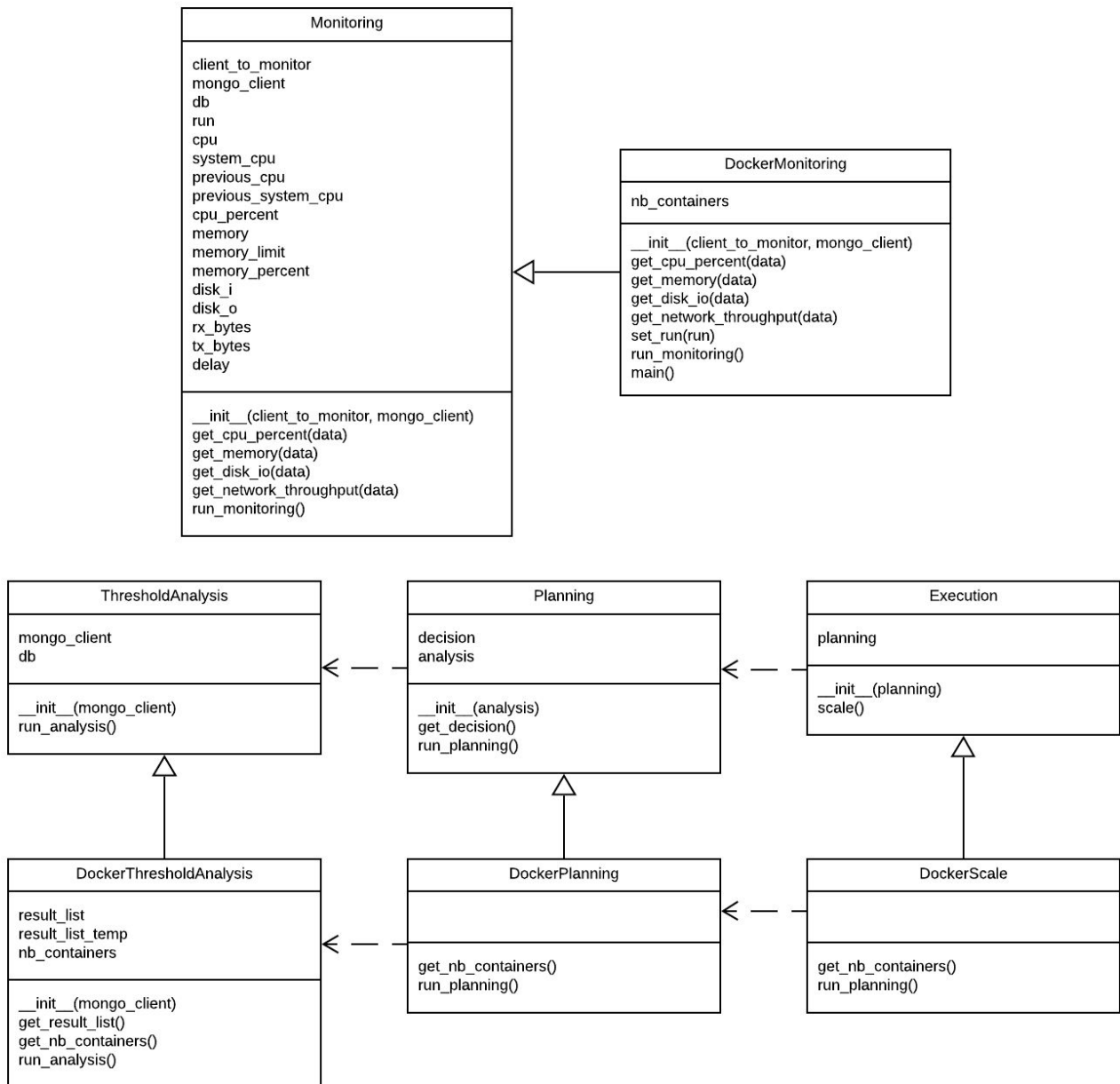
`/src/ease/mape/planning/planning.py`

The planning file contains one abstract class **Planning** and one abstract method. Some method calls analysis methods, for getting the number of containers in the docker implementation.

`/src/ease/mape/execution/execution.py`

The execution file contains one abstract class **Execution** and one abstract method. The **DockerScale** class use `os.system()` method to send bash command to the system to scale up and down the number of containers.

Class diagram



HOW TO RUN EXISTING MAPE LOOP

Docker client

Before starting :

- Make sure you have a MongoDB database running and the good URI mentioned in the program.
- Make sure you have mentioned the path where the *docker-compose.yml* is located.
- For the threshold analysis, please choose your thresholds in the .env file.

Get started :

1. Configure your .env file
2. Run the **/src/ease/mape/monitoring/monitoring.py** file into a terminal
 - a. You should see "*Monitoring is running*"
 - b. Wait few seconds until the monitoring is done
 - c. You should see "*Containers data stored in ... sec*"
(If not the monitoring will return an error.)
 - d. Wait ... sec and it will redo until you stop it

As mentioned previously monitoring run independently than the other modules of the MAPE loop so you can run only the **monitoring.py** file to just monitor your client.

3. Run the **/src/ease/mape/main.py** into another terminal
 - a. You should see "*THRESHOLD ANALYSIS ...*" (In our case for the threshold based)
 - b. If it works then you should see some information about thresholds.
 - c. After the number of containers will be printed and the planning decision.
("NTR" means Nothing To Report)
 - d. And in case of a scale the docker-compose prompt will appears
 - e. Wait ... sec and it will redo until you stop

Warning : You cannot run the main.py if there is no data into the database. Wait for the first insert before launch this -APE loop.

Debugging :

- If the docker client is on your computer, you can launch docker stats command to have a stream of your current environment and see if changes well done.
- Maybe you can use MongoDB Charts for plotting some data.

BUILD YOUR OWN IMPLEMENTATION

Monitoring

If you want to implement a monitoring for custom clients, you have to create a subclass of the abstract class **Monitoring** in the `/src/ease/mape/monitoring/monitoring.py` file. The next step is to implement all abstract method.

The `__init__()` method initiate the client with the `client_to_monitor` parameter and the MongoDB client with the `mongo_client` parameter. It also initiate lot of variable that you can use if you need for your monitoring's needs.

The best way to implement the abstract methods is to give them a JSON or dictionary parameter. If you do not want to implement method like `get_network_throughput()` because you do not need it, you are forced to write them into your subclass but you can just write *pass* and they will be skipped.

ex :

```
def get_network_throughput(self, data):
    pass
```

Remember that if you want our **MAPE** loop works you must regroup all your data into a dictionary and store it into the database. Because the analysis will always read it in the same way.

Here is an example of this type of document for docker clients :

```
{
  "date":  ISODate("2019-09-04T19:22:21.695  Z"),
  "nb_of_containers":1,
  "iot-docker-mongodb_web_2":{
    "short_id":"e281a15ae0",
    "cpu":{
      "cpu_usage":84.8679906742857
    },
    "memory":{
      "memory":53420032,
      "memory_limit":16547303424,
      "memory_percent":0.3228322502536592
    }
  }
}
```

Analysis

The threshold analysis is quite simple to implement because you only need to read the database where you stored the cpu data and compare them to the thresholds. In order to do that you need to enter the same mongo client as the monitoring and get the last update data from it. The best way to connect the analysis to the planning is to store analysis into a list that the planning can read and interpret them. Be careful to never empty this list, so to make sure this never happened, create a temporary list that you can copy into the main list when all data were analysed.

Planning

The `__init__()` method of the **Planning** class needs an analysis object parameter, in order to get the analysis list and plan some actions in consequence. The decision in our case is the information which will be send to the execution module, so you need to implement the `get_decision()` method.

Execution

The `__init__()` method of the **Execution** class needs a planning object parameter, in order to get the decision and execute it. In other words, the execution will scale up or done if needed for the client. The only method to implement is the `scale()` method which is the link between the **MAPE** loop and the client.

Main

In the `/src/ease/mape/main.py` file you will instantiate all objects you need like the analysis, planning and execution and you will call all method you need from those three classes. In order to run it continuously, you can use a `while True` loop.

ERRORS YOU MAY ENCOUNTER

In the docker implementation which already exist there is not the possibility to stop the monitoring until the system reach stability again. So sometimes when the the web containers is scaled down the monitoring throw an error and you have to relaunch the file.li