EASE MAPE System

User manual

version : 0.2

author : Valentin LAQUIT

date : 2019/11/5

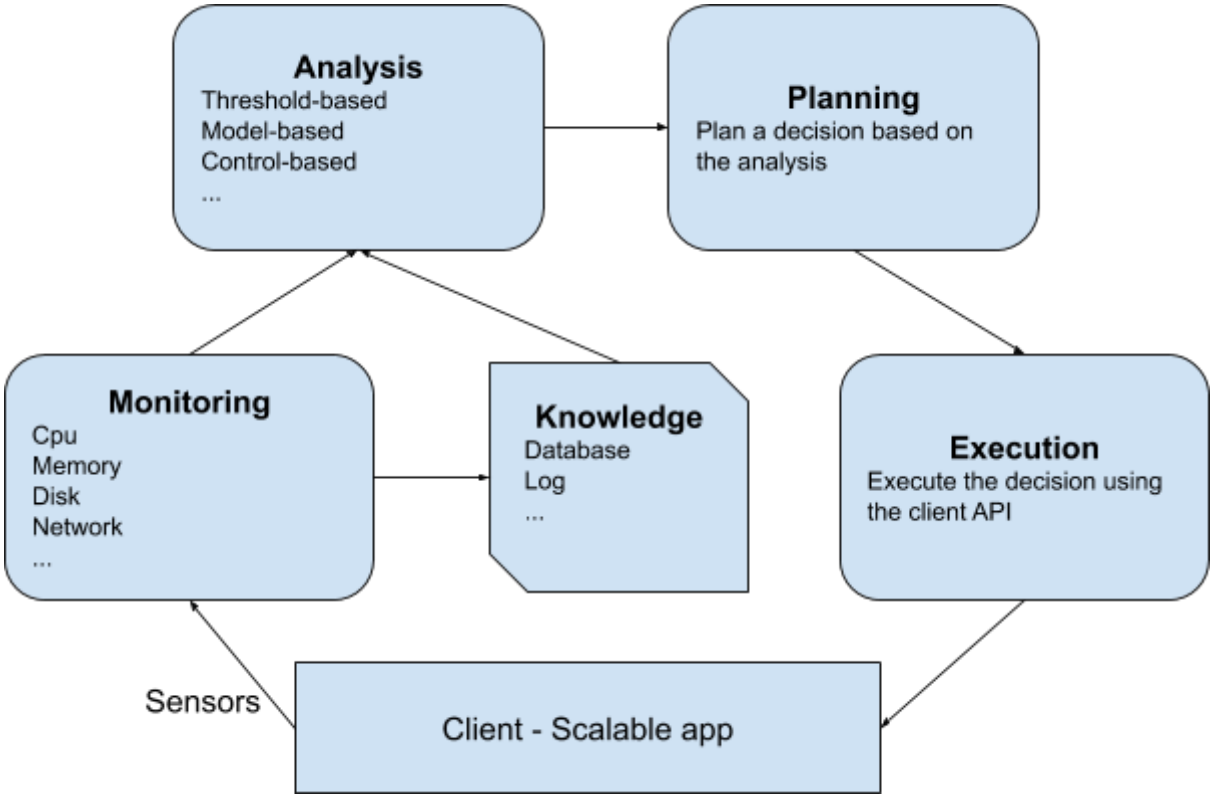
TABLES OF CONTENTS

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.

What is a MAPE architecture ?



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.

T**he 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 :

```
EASE-MAPE-System
 ├──── build.py
 ├──── .env
 ├──── requirements.txt
 ├──── db
 │     └──── docker-compose.yml
 ├──── mape
 │     ├──── dockercompose_autoscale.py
 │     ├──── analysis
 │     │     ├──── analysis.py
 │     │     └──── docker_threshold_analysis.py
 │     ├──── execution
 │     │     ├──── docker_execution.py
 │     │     └──── execution.py
 │     ├──── monitoring
 │     │     ├──── docker_monitoring.py
 │     │     ├──── monitoring.py
 │     └──── planning
 │           ├──── planning.py
 │           └──── threshold_planning.py
 └──── scripts
       ├──── autoscale.sh
       ├──── db.sh
       └──── monitoring.sh
```

⚠ *This is not the complete list of files and folders in the GitHub repository, but these are the important and necessary files for the project to work.*

All requirements are indicated in requirement.txt. Before you run anything, you should install all requirements with `pip3 install` or `pip install` command*.
** Using of PyBuilder build tool is in developpement.*

Files descriptions

/Build.py
This file is the PyBuilder* setup file. You should use it for installing dependencies.

/.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. (This file is hidden, do `ls -a` to see it)

/requirements.txt
All requirements are indicated in requirement.txt. Before you run anything, you should install all requirements with `pip3 install` or `pip install` command*. Or use the PyBuilder module and use `pyb` to install dependencies.

/db/docker-compose.yml
It is a docker-compose file to create a local mongodb database (optional)

/mape/dockercompose_autoscale.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 the analysis.

/mape/monitoring/monitoring.py
This file contains one abstract class **Monitoring.** In this class you have the `__init__()` method that get parameter for the client to monitor and the database client. And you have a `database_insertion()` method that you can call in your program to insert data into the database.

/mape/monitoring/docker_monitoring.py
This is the concrete implementation of the monitoring module for Docker Client. You have a class with several methods of calculation. The most important is that this class use Docker remote API to get information about containers running in the client environment.

/mape/analysis/analysis.py
This file contains one abstract class **Analysis.** There is several method, including one abstractmethod `update`. Notice that you have *Observer relation.

/mape/analysis/docker_threshold_analysis.py
This is the concrete implementation of the analysis module for threshold based analysis. There is an *Observer relation with Planning. The goal is to call Planning method when the analysis detect something.

`/mape/planning/planning.py`
This file contains one abstract class **Planning.** There is several method, including one abstractmethod `update`. Notice that you have *Observer relation.

`/mape/planning/threshold_planning.py`
This is the concrete implementation of the planning module. There an *Observer relation with Analysis and Execution. When he analysis call Planning method, the planning will plan some decision and call the Execution module.

`/mape/execution/execution.py`
This file contains one abstract class **Planning.** There is several method, including one abstractmethod `update`.

`/mape/execution/docker_execution.py`
The **DockerExecution** class use `os.system()` method to send bash command to the system to scale up and down the number of containers. There is an *Observer relation with Planning.

`/scripts/autoscale.sh`
This bash script run the autoscaling.

`/scripts/db.sh`
This bash script create a MongoDB database and a database manager.

`/scripts/docker_monitoring.sh`
This bash script run the monitoring.

** Using of PyBuilder build tool is in developpement.*

HOW TO RUN EXISTING MAPE LOOP

Docker client

Before starting :

You need a sample application to test this system. It requires a web application and a workload generator to simulate a change in workload.

- Install docker :

Official docker documentation: https://docs.docker.com/

- Make sure you have a MongoDB database running and the good URI mentioned in the program.

Official mongoDB manual: https://docs.mongodb.com/manual/

- Make sure you have mentioned the path where the *docker-compose.yml* is located.

Install docker compose: https://docs.docker.com/compose/install/

- For the threshold analysis, please choose your thresholds in the .env file.

Install PyBuilder : http://pybuilder.github.io/

- You should read about the build.py file and the command pyb to use PyBuilder.

Get started :

1. Configure your **.env** file (It is an hidden file so use a terminal to edit it)
2. Install all requirements from requirements.txt. (via: Pybuilder, .sh file or pip command)
3. Run the **monitoring.sh** file into a terminal with `sh monitoring.sh`
    a. You should see "*Monitoring V2 \n Using: Docker remote API*"
    b. Wait few seconds until the monitoring is done
    c. You should see "Time to insert into the database … "
    (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.

4. Run the **auoscale.sh** into another terminal
    a. You should see "*Analyse : …*"
    b. After some informations will be printed and the planning decision.
    (If nothing is printed, this mean that the analysis does not detects something wrong)
    c. And in case of a scale the docker-compose prompt will appears
    d. Wait … sec and it will redo until you stop

Warning : You cannot run the autoscale 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.

You can follow this tutorial :
https://www.mongodb.com/blog/post/visualizing-your-data-with-mongodb-charts

BUILD YOUR OWN IMPLEMENTATION

It is recommended to use PyCharm or an equivalent IDE.

Monitoring

If you want to implement a monitoring for custom clients, you have to create a subclass of the abstract class **Monitoring** into a new file in the same folder. The next step is to implement all method you need.
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 idea is to connect the algorithm to the client with an API. For example a REST API. The best way to implement methods is to give them a JSON or dictionary parameter taken from the client's API.
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.

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 establish an *Observer relation with the Planning. So the Analysis will call the Planning when it is necessary.

Planning

The `__init__()` method of the **Planning** class needs to observe the Analysis and plan decision. The decision in our case is the information which will be send to the execution module.

Execution

The `__init__()` method of the **Execution** class needs to observe the Analysis, 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 **/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.

Dependencies

To manage dependencies, you must list all the necessary packages in the requirements.txt file.

ERROR YOU MAY ENCOUNTER

- Maybe you will have trouble installing dependencies with Pybuilder. This may be due to an access or python version problem.

Appendix

Observer relation