Docker, assessment task

|  |  |
| --- | --- |
| **ASMT number** | 01 |
| **For module** | Docker |
| **Start date** | 21.09.2021 |
| **End date** | 30.09.2021 EOD |

It is the final docker task and challenge!

# Task

Prepare docker image and docker-compose which purpose is to deploy a simple web application along with Redis data structure store server.

The application is written in Python language with Flask framework support, and it has been designed to store data in Redis.

# Files

Files are zipped and available in the task’s directory

The structure:

.

├── README.md

├── main.py - Python web application

├── requirements.txt - List of all application dependencies

└── templates

    ├── reset.html - HTML template for /visitor

    └── visitor.html - HTML template for /visitor/reset

# Main Goals

Task must meet below requirements

* Dockerize web application. Pass below environments to the container:
  + REDIS\_HOSTNAME
  + REDIS\_PORT
* Create docker-compose configuration that easily deploy application and Redis in declarative way.
* All data stored in Redis should be preserved across containers recreation.
* Use alpine or alpine based images in this challenge.

# Score

You can use whatever technique you wish. Please note that you can use images from registries, or you can create your own images.

The following conditions will be taken into account during the evaluation:

* Approach to the application deployment – less steps is better.
* Usage of multi containers approach.
* Code formatting (Dockerfile and docker-compose.yml)
* How you resolved application dependencies.

# What will be checked to assess your work

* Can I see the webpage?
* Can I reset the number of visits?
* Can I recreate all containers without losing data?
* How code was written
  + Code formatting
  + Is the code parametrized? (i.e. Redis version, Python version, etc)
  + Use of templates and variables for parametrization
* Documentation (can be very brief but useful)

Additional points will be given for optional task, based on:

* % of completion
* If what you have… works J

# Delivery

Please send access to repo (as usual) to [Pawel\_Piwosz@epam.com](mailto:Pawel_Piwosz@epam.com).

# Optional task 1

Create HAProxy in front of Python web application.

* Use multi containers approach.
* Configure HA proxy to listen on HTTP protocol.

All traffic between a user and HA proxy should be handled by HTTP protocol

Traffic between HA proxy and Python App should be handled by HTTP protocol.

client <--> (HTTP) <--> HA Proxy <--> (HTTP) <--> Python App

# Optional task 2

Implement HTTPS protocol for HA proxy.

* Reconfigure HA proxy to listen only on HTTPS protocol.
* Generate your own CA, generate and sign certificate for HA proxy.

All traffic between a user and HA proxy should be handled by HTTPS protocol

Traffic between HA proxy and Python App should be handled by HTTP protocol.

client <--> (HTTPS) <--> HA Proxy <--> (HTTP) <--> Python App

# Optional task 3

Use a password with Redis:

* Modify main.py to include and use password.
* Modify Redis image to use passwords.

# Helping hand

Please complete obligatory task first. Make sure, that you have working solution, then create branch and work on optional tasks.

You can use below hints:

## Hint 1: Python version

Use python 3.7

## Hint 2: Python web server

Use gunicorn web server.

## Hint 3: Images that you can use

redis:alpine for Redis container

python:3.7-alpine for the application container

## Hint 4: Connect to Redis from your host

Obviously, your Redis container must expose the port. The default port is 6379.

To connect to Redis you have to use redis-cli tool. In orger to install it, run

**$ sudo apt-get install redis-tools**

In case that you have running container with default configuration of redis, it will be enough for you to run

**$ redis-cli**

From your command line (default connection is going to 127.0.0.1:6379)

Here is documentation, if you wish to learn more: <https://redis.io/topics/rediscli>

## Hint 5: CMD for Python app

CMD ["gunicorn", "-w 4", "-b", "0.0.0.0:8000", "main:app"]

# Steps

* Dockerize the application.
* Create docker-compose.yaml.
* On a new branch start working on optional tasks (not required).

# Help needed?

Please contact Lukasz Wojdyla, Sylwia Semper and Pawel Piwosz

# Don’t be shy J

**I will recreate machines for you as many times as needed J**