Cloud and Edge Computing – Project assignment

Introduction

In this project your task is to write and run a few simple applications in a docker environment. The project is split into two stages, which are described below. Overall, both stages implement the same functionality, but with a different load balancing approach. After deploying the different stages your next objective is to benchmark the two variants and finally plot a comparison of them.

All steps are assumed to be deployed inside a virtual machine (VM) we provide. It runs Debian 10 with all necessary software pre-installed. To run it you need to install VirtualBox. This project requires some basic bash scripting skills, as well as basics in Node.JS/Python/Ruby or some other easy-to-approach "web-compatible" language.

Resources

Docker documentation: https://docs.docker.com/
Host VM image: https://www.cs.helsinki.fi/u/owaltari/cec_vm/
Login credentials for the VM:

User: **cecuser** (sudo is at your disposal)
Password: **cecuser**

Inside the Host VM image you will find the following files/paths:

/var/cec/redis.rdb

This is a Redis database snapshot you will be using in both stages of this project.

/home/cecuser/Project/

This is a folder in your home directory inside the VM. Keep all your project related files here.

/home/cecuser/cec_benchmark.py

This is a python program that benchmarks latency to your containers with different parameters.

TASKS

Tasks are split into smaller steps. They must be completed in the order they are given.

Stage 1

A) Deploy and run the provided VM image. Build and run a "bulletinboard" app in a container inside the VM. (Follow "*Containerizing an application*" in the official documentation) (1pt)

B) Run a Redis server in a container and make it provide the snapshot stored in /var/cec/redis.rdb (1pt)

C) Write an HTTP server that pulls one value from the database through your Redis container. Calculate the factorial of that number and operation in response, i.e. in the HTTP document body. (sidenote: The result in majority of the cases is going to be infinite, and that's fine. We just want to generate some artificial CPU load.) (1pt)

D) Implement a load balancing access node application. One that distributes client requests across several instances (layer 7, simple HTTP redirect is fine) of your HTTP server. It needs to be able to distribute load to at least 3 server instances in a round-robin fashion. Do NOT use existing load balancers (HAProxy, nginx, etc.) Feel free to use your creativity on how to implement this from scratch. (1-2 pts)

Stage 2

A) Write a service definition around your HTTP server and Redis (docker-compose.yml). (1 pt)

B) Deploy and scale your service up to 3 instances of your HTTP server and one instance of Redis. (1 pt)

Benchmarking

A) Benchmark the final deployment of both Stage 1 and 2 in the following manner:

1. Limit CPU usage of your HTTP server containers to 5% (--cpus=".05") (don't limit your load balancer or redis).
2. Scale your HTTP server to 5 containers and run cec_benchmark.py against your web app.
3. Repeat the previous, but scale server instances to 3 and 1.
4. Store the results. (1 pt)

B) Plot the results. You should get two plots: One for each stage (1&2), and three lines in each plot (server scaled to 1, 3 and 5 containers). Use mean response time on the Y-axis, and the number of simultaneous threads (HTTP GET spammers) on the X-axis. (1 pt)

C) Explain the results. (1 pt)

OTHER REMARKS

External libraries

Feel free to use the language and library of your choice for implementing the HTTP server. After the project deadline model answers will be given in Node.JS using the built-in http API. Using other languages (Python, Ruby, etc.) and helper libraries for HTTP (e.g. Flask, django, RRails, etc.) is accepted.

Kubernetes is not necessary for completing this project. It will probably only make your task more complicated. However, we won't punish, nor reward, students who want to use Kubernetes.

Deliverables

All files you wish to return should be stored under /home/cecuser/Project/! Your deliverable is going to be that folder compressed into one single ".tar.gz" archive, which you may then upload to Moodle. Name the file with your university username. The exact command to do this inside your VM is:

```
tar –cvzf [your username].tar.gz /home/cecuser/Project/
```

This should create you an archive in your current working directory (e.g. jsmith.tar.gz, in case your name is John Smith). Double check that all necessary files are in it, and then upload it through Moodle.

Inside the folder `~/Project/` there are pre-made subfolders for all parts of the project. They contain also bash script placeholder files, where each command necessary for running each part of the task should be explicitly written in an executable format.

All files from the benchmarking stage should go under `~/Project/benchmark`. This includes the results in a CSV-file format, PDF/PS or other style of graphic representation of the plots, and a PDF/text file with explanation for step C.

Documentation

We expect `execute_stage1.sh`, `execute_stage2.sh`, and your load balancer implementation to be self-sufficiently documented with comments. Keep in mind that comments can help you in case there is something unclear with your code or commands.

Deadline

Project **deadline is on March 1st at 23:00**. Submit on time! Late submissions are not accepted.