CSC8426 - Emerging Technologies

Course work

Objective

This coursework helps you to understand the basics of Docker (an emerging cloud computing services' virtualisation technology) and gives hands on experience in setting up a Docker host, configuring and hosting an application stack on a Docker environment. It gives a basic understanding of benchmarking and monitoring Dockerized applications.

Pre-Requisites

Before attempting this course work, complete the practise sessions mentioned in https://github.com/nclcloudcomputing/csc8426. In these practise sessions, we provide tutorials including how to download and install VMware in your laptop/PC and how to import given virtual machine image which contains all necessary programming environments for this coursework. Besides, we also provide Docker relevant references including Docker swarm, cAdvisor, Docker hub, etc.

Deploying Web application using Docker and performing benchmarking and monitoring

In this exercise, a web application along with several other services have to be deployed in a containerized environment.

Task 1: Deploy a web application in Docker (2.5 Marks)

Task Objective: Understand how to pull and run Docker image from Docker Hub in command line interface.

Tips: The video tutorial (Task1 EmergingTechnologies Coursework) is provided in https://github.com/nclcloudcomputing/csc8426.

A sample Java web application is provided in the image "nclcloudcomputing/javabenchmarkapp" from Docker Hub. The image contains an application deployed on a Tomcat application server. The application performs a prime number check on a large number, placing a load on the CPU. It listens to port 8080. URL: http://<hostname>:8080/primecheck.

1. Pull and run the Docker image "nclcloudcomputing/javabenchmarkapp" from Docker Hub. Perform these tasks using the command line interface (CLI).

2. Verify whether the application is working by pointing your web browser to http://localhost:8080/primecheck. The page should display the time taken to perform a random prime number checking calculation.

Task 2: Deploy a service stack in Docker (10 Marks)

Task Objective: Understand how to create a complex web application stack consisting of multiple Docker containers (web server, load generator, database server, monitoring engine, visualizer) using Docker compose and Docker swarm techniques.

Tips: The video tutorial (Introduction to Docker swarm final) is provided in https://github.com/nclcloudcomputing/csc8426.

1. Pull a load generator Docker image (locustio/locust), a MongoDB Docker image (mongo), a cAdvisor monitoring Docker image (google/cadvisor) and A Docker Swarm Visualizer (dockersamples/visualizer) from Docker Hub.

2. Define a Docker compose file which contains all necessary services (web server, locust, MongoDB, cAdvisor, Docker Swarm Visualizer)

3. Create a Docker service on a single Docker swarm node with the defined docker compose configuration, the architecture of this service is shown in Figure 1.

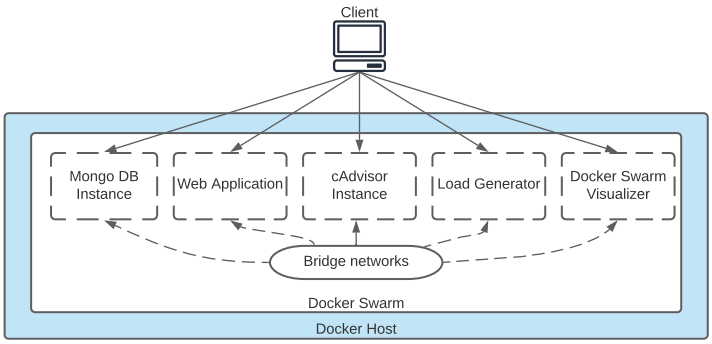

Figure 1: Deployment Architecture

Task 3: Build a Docker image and push it to Docker Hub (2.5 Marks)

Task Objective: Understand how to build a Docker image from source code and how to push it to Docker Hub.

Tips: The video tutorial (Docker Build Tutorial) is provided in https://github.com/nclcloudcomputing/csc8426.

1. Download the source code of the provided program https://github.com/ncl-iot-team/cadvisor-scraper. The program extracts the monitoring information from API, for each container instance of the web application and insert those records into the MongoDB database, whilst running the load generator. Edit several fields (containerId, mongodbAddress, endpoint) in the provided Java code.

2. Build the program as a Docker image, then push it to Docker Hub.

Task 4: Execute application and insert benchmark results into database (5 Marks)

Task Objective: Execute all services in the stack, make sure the load generator is sending load request to the web server; using cAdvisor to monitor the status of the web server; using provided program to extract monitoring records from cAdvisor then insert into database.

Tips: The video tutorial (cadvisor scraper) is provided in https://github.com/nclcloudcomputing/csc8426.

1. Pull cadvisor scraper image back to host and add this service in the Docker compose file.
2. Run the new Docker compose file to execute the benchmarking experiment.
3. After executing benchmarking experiment, make sure cadvisor status records are stored in the MongoDB.

Task 5: Report (15 Marks)

Produce a detailed report for all the tasks performed in not less then 1500 words. The result must includes the following components

1. Explanation of how to complete each tasks.
2. Screenshots of each running services in Task 4.
3. Benchmarking results plotted in the form of graphs using any tool (e.g. R, matlab, excelsheet).
4. Discussion of the results and detailed inference.

Marking

Results of Task 1 - Task 4 are to be presented to a marker in a live demo session of 10-15 minutes. While Task 1 - Task 4 can be demonstrated on student's own laptop or on the desktop provided by the University.

Report (Task 5) along with code should be submitted in NESS.