CSC8110/CSC8634 - Cloud Computing

Assessment 1

Assessment Overview

This assessment contributes 40% towards the total mark for this module. It is an individual exercise: no group work is permitted for the assessment. You are advised to watch all the instructional videos, provided in-line with the individual task specifications, as well as read all tutorial resources before you start answering these questions. Marks shown are indicative only.

The coursework report must be submitted on NESS by 22:00 pm on 13th December, 2024. After completing Tasks 1-4, you will need to provide in-depth details on how you implemented the solutions (e.g., code, commands) to solve the aforementioned tasks in a final Report (Task 5). Additionally, you will be required to demonstrate successful executions of Tasks 1 - 4. Before the report submission deadline, you will be provided a 15 mins slot to conduct live demonstrations.

Final marks for this assessment will be decided by your performance in the live demonstrations and the details provided in the final report. While the final report needs to be submitted to NESS by 22:00 pm on 13th December, 2024.

You are required to complete Tasks 1-4 using the command line tool ([https://kubernetes.io/docs/reference/kubectl/**]).**

Objectives

This coursework helps you to learn and understand the fundamentals of programming and deploying Kubernetes-based (an emerging cloud virtualisation technology) application hosting environment. By successfully completing the coursework, you will be able to gain hands-on experience in the following interrelated aspects including:

- Configuring a Kubernetes-based application hosting environment;
- Building, pushing and pulling images from the Docker Hub (global repository of software components' images maintained by the developers);
- Creating and deploying a complex web application stack consisting of multiple software components (e.g., web server, monitoring, etc.);

- Monitoring the ongoing performance of your application stack using the inherent features provided by the Kubernetes environment.

Pre-Requisites

Before attempting this assessment, you are advised to go through the instructions given at [https://github.com/ncl-iot-team/csc8110] which provides details on:

- how to run your experiments on Azure Labs
- how to download and install VMware on your laptop/PC and
- how to run a virtual machine image, which is packaged with all necessary programming environments required for this coursework, in your laptop/PC.

Specification of Tasks

The coursework consists of 5 tasks. **Tasks 1-4 need to be done by the command line tool (kubectl).**

Task 1: Deploy and access the Kubernetes Dashboard and a Web Application Component (10 Marks)

Task Objective : Understand and learn basic concepts of Kubernetes, and how to deploy applications to a Kubernetes cluster.

Hints : Please refer to [https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/]

A sample *Java web application component* image, named "nclcloudcomputing/javabenchmarkapp", has been uploaded to the Docker Hub. The image contains a ready-to-use implementation of a web application deployed in a Tomcat server (an open-source web server). In terms of computational logic, the web application implements a prime number check on a large number. By doing so, the application can generate high CPU and memory load.

1. Deploy 'Kubernetes Dashboard' on the provided VM with CLI and access/login the Dashboard.
2. Deploy an instance of the Docker image "nclcloudcomputing/javabenchmarkapp" via CLI.
3. Deploy a NodePort service so that the web app is accessible via http://localhost:30000/primecheck. The container uses port 8080 internally.
4. You cannot use the dashboard addon from microk8s.

Task 2: Deploy the monitoring stack of Kubernetes **(30 Marks)**

Task Objective : Understand and learn how to deploy a monitoring stack of Kubernetes consisting of Prometheus, metrics server, Grafana.

1. Grafana must reach Prometheus as a data source
2. Grafana must be reachable from the host
3. Prometheus must reach each instance of the metric server to collect metrics
4. The metric server must be deployed as a DaemonSet

Task 3: Load Generator **(20 Marks)**

Task Objective : Understand the logic of the load generator of benchmarking web applications and the process of deploying your own application of the cluster. Additionally, understand how to build and push a Docker image from scratch.

Hints : To push to a private registry, you can do `docker push [address]:[port]/[image_name]`
A load generator creates load on the web application by calling its URL multiple times. This could be used to benchmark the system's performance.

1. Write a load generator with the following specifications
   (a) Accepts two configurable values either via a config file or environment variables. `target` (The address for the load generation) and `frequency` (Request per second)
   (b) Generate web request to the target at the specified frequency
   (c) Collect 2 types of metrics. Average response time and accumulated number of failures
   (d) Request should timeout if it takes more than 10 seconds. Counted as failures
   (e) Test results need to be printed to the console
   (f) There are no requirements in programming language
2. After programming, pack the program as a standalone Docker image and push it to the local registry at port 32000. Name the image as *load-generator*.

Task 4: Monitor benchmarking results **(10 Marks)**

Task Objective : To learn and understand how to monitor container metrics.

Hints : In the Grafana panel, you can specify metric: `container_cpu_usage_seconds_total/ container_memory_usage_bytes`

1. Deploy *load-generator* service created in Task 3.
2. During the benchmarking, create a new dashboard on Grafana and add 2 new panels which should contain queries of CPU/memory usage of the web application
3. Screenshot the two panels

Task 5: Report **(30 Marks)**

Prepare the Final Report in plain English. The report should consist of:

1. Detailed response to each task and related sub-tasks;
2. Screenshots of running services in Kubernetes;
3. Plots of Benchmarking results.
4. Discussion of the results and related conclusions.