Cloud Basics Exercise

General

Submission date: 3.12.25 | Submission mode: Single | Exercise weight: 10%

In this exercise, you will take the Docker container you developed in EX5 (last semester) and expose it for general use via a cloud provider of your choice.

Depending on the cloud provider you select, you will need to learn and explore how to deploy and run your Docker container on the cloud, exposing it to the end user.

An automation tool will test your code by accessing a URL you supply and invoking HTTP requests, comparing them to the expected responses from the container.

The Exercise

In this exercise, you are not required to develop new code. All logic relies heavily on the *server exercise (ex3)*. The same logic is preserved, and all you need to do is enable it via a cloud provider of your choice, as was discussed and demonstrated in class.

You should explore and investigate how to operate with the cloud provider you choose to work with.

The cloud providers recommended for use are:

- GCP - as was demonstrated in class
- Azure - self-explanatory. Free student account. No credit card needed!
- AWS - you all gain MTA's student license

Note: The above three cloud providers are the most common ones used in the industry, so it's preferred to gain some experience working with them and become familiar with them.
Having said that, you can choose to use any other cloud provider you are familiar with (e.g., Firebase, Oracle), but ensure you achieve the preferred outcome (e.g., 24/7 availability).

Unlike earlier exercises, this time you have full control over the port you work with.
Your result will be a full URL**, on which the automation will append the relevant resources it needs to invoke (e.g.,**/calculator/health**), and will expect certain results to be returned (again, using the very same logic and structure as you already developed during the last exercises).

The automation also assumes that your server is bootstrapped from scratch: clean and empty of any history or past invocations information, without any remains that might exist from your own testing of its functionality.

It is recommended to define only one instance for the service to preserve minimum cold-start times

** make sure your URL starts with either **http://** or **https://** prefix.
The automation takes it **AS IS** and expects it to be a valid, complete URL to execute.

What to Submit

You should submit a simple json file (**DO NOT** zip it !).
The json file must be in a valid json format. (**PLEASE, I BEG you** to use a json validator for that)

IT'S A MUST.
NO APPEAL WILL BE ACCEPTED FOR THIS REASON.
BY NOW, YOU ARE ALL BIG BOYS & GIRLS….

The json will hold the following data:

```
{
        “name”: string <your name>,
        “id”: string <your ID>,
        “url”: string “<full url exposed 24\7 for external usage>”,
}
```

Example:

```
{
        “name”: “Menashe Hateymani”,
        “id”: “98989898”,
        “url”: "https://docker-example-ilov-egat-uc.a.run.app"
}
```

The automation will take the URL and append the endpoint resource as described in EX3.
e.g.

https://docker-example-ilov-egat-uc.a.run.app/calculator/health

Note:

- The JSON file must be a valid json file. Note the use of (“) wherever you need.
  In order to be sure it's a valid file, you can test yourself using an online json validator:
  https://jsonlint.com/
- The keys are **case-sensitive** and must appear **exactly** as they appear in the above example.
- Please take special care to use the correct URL. The automation will take it **AS IS** and **WILL NOT** manipulate it in any way or manner. (e.g., adding https protocol prefix..)
- For convenience, you can download and use this example file as a template for your file
  To make it clear: you still need to check it is valid
  (It is preferred and recommended to use this file and not copy-paste the json above due to the corrupted (“) sign in Google Docs…)