CS 203: Software Tools & Techniques for AI
IIT Gandhinagar
Sem-II - 2024-25

LAB 08

Total marks: 10
Submission deadline:
Submission guidelines:

1. Code should be added to a GitHub repository, and the repository details should be shared.
2. Late submissions will be penalized 20% per day.
3. Google form submission link:

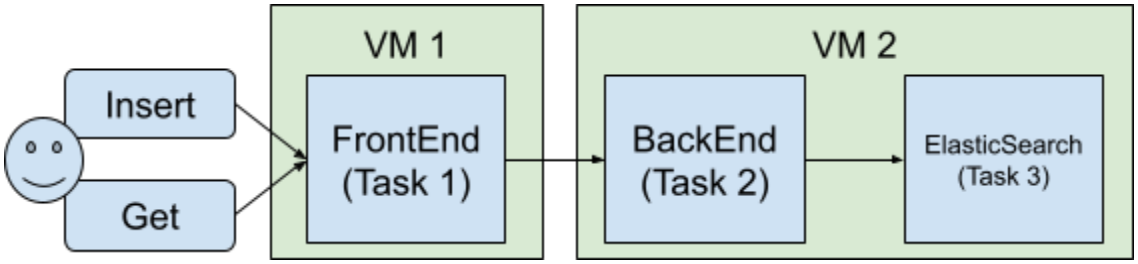
Note: By submitting this assignment solution you confirm to follow the IITGN's honor code. We shall strictly penalize the submissions containing plagiarized text/code. **Copying/Pasting using any AI Tool like ChatGPT is strictly prohibited for this assignment.**

It is mandatory to perform the assignment on Google Cloud or Kutrim (Ola Cloud).

In this lab, we will focus on **Containerization of applications using docker.** We will be building our own docker images and running them. Docker containers need to communicate with each other properly to make the whole application work.
The goal is to understand docker build, docker network, and docker volumes.

This lab is to be completed in teams of size 2 students.

For the whole assignment, it is mandatory to use CLI, except for testing the frontend web page.



Task 1: FrontEnd

Create FrontEnd using FastAPI (Similar to Flask)

Note: Flask is not used in the production environment hence we are using FastAPI.

The front end should be simple no need to have any style or decorations. It should be isolated from other VM.

- FrontEnd should run on port number “9567” and should be public. (i.e. Directly accessible from the base OS in this scenario)
- From BaseOS https://0.0.0.0:9567 link should work.
- The HTML page shown should have an input text box, two buttons (get, insert), and some areas where output is shown.

  Note: No need to beautify the HTML page keep it simple, only functionality will be checked.

  The get button should retrieve the document with the best score.

  The insert button should insert a large document inside the elastic search.

- This will be running inside the docker container.

Task 2: Backend

- A FastAPI code will be running as a backend on port “9567”.
- The backend code’s job is to get the input from the frontend container and query the elasticsearch container.
- HostOS should not be able to connect with the Backend directly.
- This will be running inside the docker container.

Task 3: Elasticsearch

Elasticsearch is a well-known application used in AI for RAG (Retrieval Augmented & Generation)

- There will be an elastic search container running which will take queries from backend and give the search result to it.

- An Elasticsearch will be running as a backend on port “9567”.
- Create an index in the elastic search having two fields id, text as type keyword, and text, respectively.
- Insert 4 documents in the index having the first 4 paragraphs from “https://en.wikipedia.org/wiki/India” page.
- HostOS should not be able to connect with the elastic search directly.
- The user should be able to insert a query to Elasticsearch via the frontend only. For search operations, implement match search.
- Even after deleting the container , thedata inserted by the user should not be deleted.

Evaluation Criteria

Environment Setup (70%)

- Performing All the 3 Task.
- Uploading image created for all the 3 tasks in the docker hub (It should be public). Name the images as fastapi:1, fastapi:2, elastic search for task 1, task 2,and task 3, respectively.

Documentation (20%)

- Documentation of the work done in PDF.
- Provide a Link for the the uploaded images on docker hub.
- GoogleDrive public link for “dockerfile” used to create images for all three tasks.
- **Screenshot** when performing all the tasks. (Dockerfile code, building command and its output, running container, and its output)
- Screenshot of what has been done to make FastAPI and ElasticSearch run on specific ports.
- Screenshot of “netstat -antp | grep LISTEN” on Base Linux OS, where the container is running.
- Screenshot of “docker images”
- Screenshot when testing both get and insert query. Then stop and delete all three 3 containers and run it again, showing that the inserted query still exists. (It must been shown clearly before that the get query output is different compared to when it has been inserted later)
- Screenshot of “docker inspect” for all the 3 tasks, both container and images. (Give the whole output in the screenshot it is essential to verify whether the task was done properly or not)
- VM creation steps are taken, and screenshots are on Google Cloud or Kutrim.
- Give a detailed architectural diagram.

Optimization (10%)

- Images should take the least possible space.
- The container should start as quickly as possible.

Note: No additional instructions should be given manually inside the container once it is started.