CS571:Week 11 Homework 3: Optional ==> GenAI - Develop your containerized app
Name: Feven Belay Araya
ID: 20027

labnet.sfbu.edu/~henry/sfbu/course/cloud_computing/genai/slide/exercise_kubernetes.html

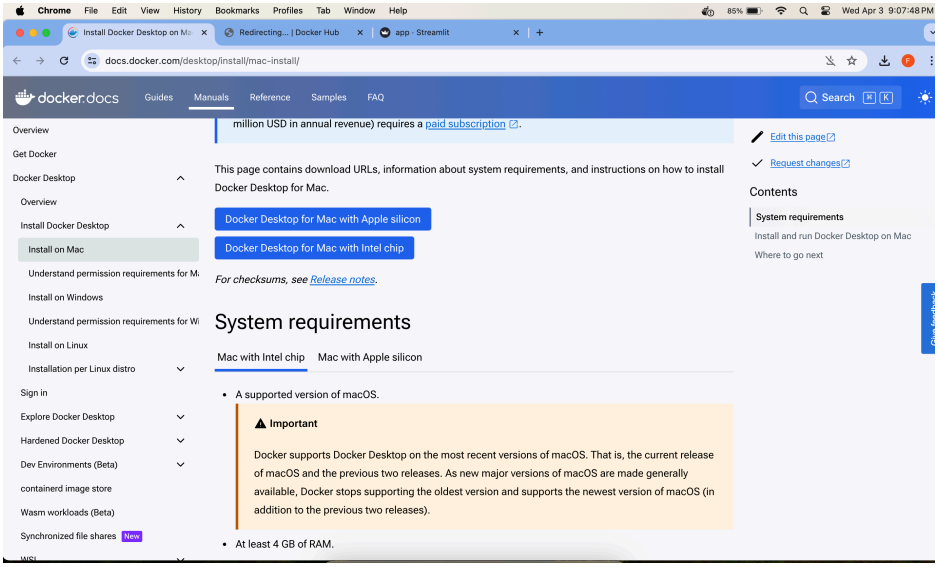
Q5 ===> GenAI - Develop your containerized app

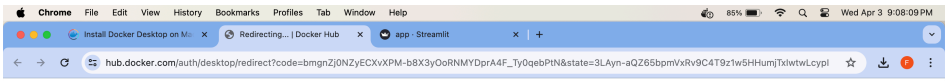
1. GenAI - Develop your containerized app
    - Step 1: GenAI - Containerize your app
    - Step 2: Develop your app
    - Step 3: Update your portfolio and publish your doucments on GitHub using this structure
        -
        - Cloud Computing
        - Kubernetes
        - Generative AI

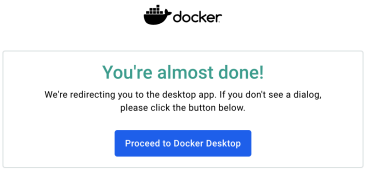
    - References
        - docker / genai-stack - GitHub

Step 1: GenAI - Containerize your app

1. First download the Docker Desktop from the following link https://docs.docker.com/desktop/install/mac-install/







1. Open Terminal, run the following command to clone the repository:
   git clone https://github.com/craig-osterhout/docker-genai-sample

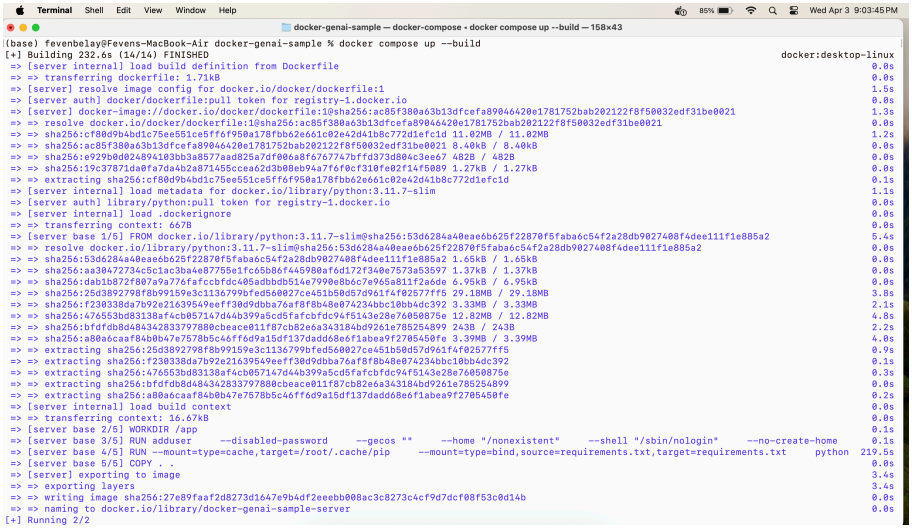
```
(base) fevenbelay@Fevens-MacBook-Air ~ % git clone https://github.com/craig-osterhout/docker-genai-sample
Cloning into 'docker-genai-sample'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 11 (delta 0), reused 11 (delta 0), pack-reused 0
Receiving objects: 100% (11/11), 10.17 KiB | 5.08 MiB/s, done.
```

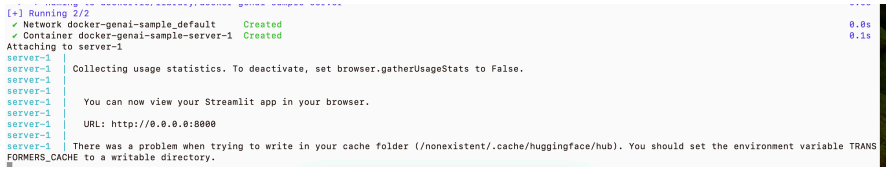
2. Go to the docker-genai-sample directory.

```
(base) fevenbelay@Fevens-MacBook-Air ~ % cd docker-genai-sample
```

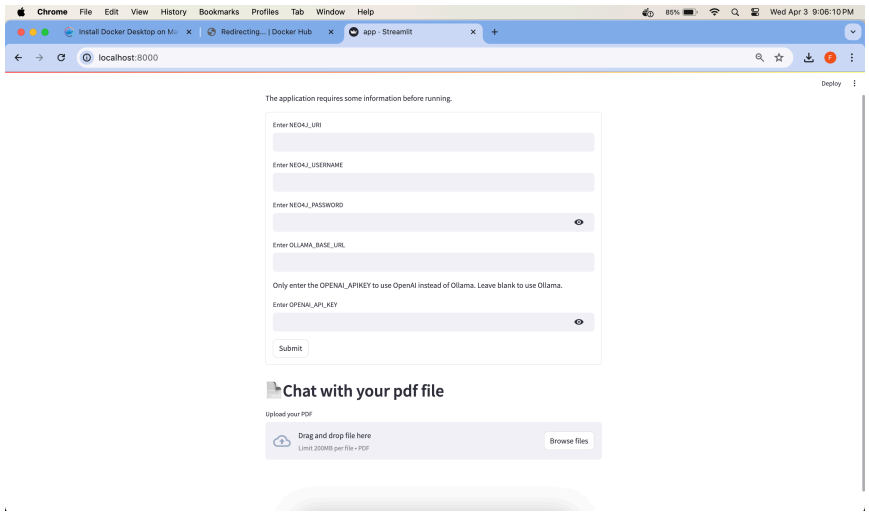
3. Then run ***docker init*** to create the necessary Docker assets to containerize your application.

4. Run the following command, **docker compose up –build** to builds and runs your application





5. Open a browser and view the application at. **http://localhost:8000.**
   - The streamlit application is then shown as follows.



6. In the terminal, press **ctrl+c** to stop the application

```
^CGracefully stopping... (press Ctrl+C again to force)
[+] Stopping 1/1
 ✔ Container docker-genai-sample-server-1  Stopped
canceled
```

Step 2: Develop your app

1. In the cloned repository's directory(docker-genai-sample), rename env.example file to .env. This file contains the environment variables that the containers will use.

```
(base) fevenbelay@Fevens-MacBook-Air ~ % cd docker-genai-sample
(base) fevenbelay@Fevens-MacBook-Air docker-genai-sample % ls
Dockerfile              app.py                  requirements.txt
LICENSE                 chains.py               utils.py
README.Docker.md        compose.yaml
README.md               env.example
(base) fevenbelay@Fevens-MacBook-Air docker-genai-sample % mv env.example .env

(base) fevenbelay@Fevens-MacBook-Air docker-genai-sample %
```

2. In the cloned repository's directory, open the compose.yaml file in an IDE or text editor.

```
(base) fevenbelay@Fevens-MacBook-Air docker-genai-sample % nano compose.yaml
```

3. In the compose.yaml file, add the following:
    i. Add instructions to run a Neo4j database
    ii. Specify the environment file under the server service in order to pass in the environment variables for the connection

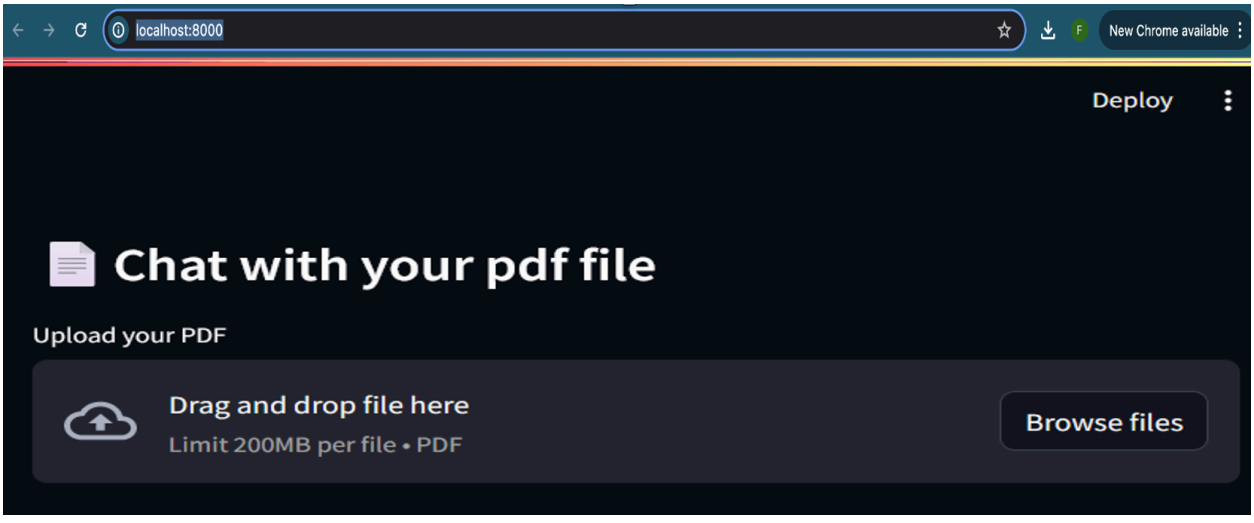
compose.yaml

```
  UW PICO 5.09                                File: compose.yaml

services:
  server:
    build:
      context: .
    ports:
      - 8000:8000
    env_file:
      - .env
    depends_on:
      database:
        condition: service_healthy

  database:
    image: neo4j:5.11
    ports:
      - "7474:7474"  # HTTP
      - "7687:7687"  # Bolt
    environment:
      - NEO4J_AUTH=${NEO4J_USERNAME}/${NEO4J_PASSWORD}
    healthcheck:
      test: ["CMD-SHELL", "wget --no-verbose --tries=1 --spider http://localhost:7474 || exit 1"]
      interval: 5s
      timeout: 3s
      retries: 5
```

4. Run the application. Inside the docker-genai-sample directory, run the following command in a terminal.
    - $ docker compose up --build

```
(base) fevenbelay@Fevens-MacBook-Air docker-genai-sample % docker compose up --build
[+] Running 6/6
 ✔ database 5 layers [⣿⣿⣿⣿⣿]      0B/0B      Pulled                                                                                    47.0s
   ✔ f96ab1515704 Pull complete                                                                                                       9.2s
   ✔ 0067a4775f01 Pull complete                                                                                                      42.5s
   ✔ 1dfb4d919908 Pull complete                                                                                                       1.0s
   ✔ 3ffb6fc03615 Pull complete                                                                                                       2.1s
   ✔ cb9cae1349bc Pull complete                                                                                                      35.3s
[+] Building 8.3s (14/14) FINISHED                                                                                 docker:desktop-linux
 => [server internal] load build definition from Dockerfile                                                                          0.1s
 => => transferring dockerfile: 1.71kB                                                                                               0.0s
 => [server] resolve image config for docker.io/docker/dockerfile:1                                                                  4.7s
 => [server auth] docker/dockerfile:pull token for registry-1.docker.io                                                              0.0s
 => [server] docker-image://docker.io/docker/dockerfile:1@sha256:dbbd5e059e8a07ff7ea6233b213b36aa516b4c53c645f1817a4dd18b83cbea56    2.4s
 => => resolve docker.io/docker/dockerfile:1@sha256:dbbd5e059e8a07ff7ea6233b213b36aa516b4c53c645f1817a4dd18b83cbea56                  0.0s
 => => sha256:7938f7c0984b428949317499eff7773a7294fdc03c085c649e24d1e3a452fba7 11.23MB / 11.23MB                                     2.2s
 => => sha256:dbbd5e059e8a07ff7ea6233b213b36aa516b4c53c645f1817a4dd18b83cbea56 8.40kB / 8.40kB                                       0.0s
 => => sha256:ab4d84ec8cad8a7b0e8b6ad1fb49536f000b86e30993c74a251bb16f3e6e3f5b 482B / 482B                                           0.0s
 => => sha256:6bb827dc58ef706c3d261e31d3a2ebd9965fa640ec3bc1cef4c98d8068b55685 1.26kB / 1.26kB                                       0.0s
 => => extracting sha256:7938f7c0984b428949317499eff7773a7294fdc03c085c649e24d1e3a452fba7                                            0.1s
 => [server internal] load metadata for docker.io/library/python:3.11.7-slim                                                         0.9s
 => [server auth] library/python:pull token for registry-1.docker.io                                                                 0.0s
 => [server internal] load .dockerignore                                                                                             0.0s
 => => transferring context: 667B                                                                                                    0.0s
 => [server base 1/5] FROM docker.io/library/python:3.11.7-slim@sha256:53d6284a40eae6b625f22870f5faba6c54f2a28db9027408f4dee111f1e885a2  0.0s
 => [server internal] load build context                                                                                             0.0s
 => => transferring context: 252B                                                                                                    0.0s
 => CACHED [server base 2/5] WORKDIR /app                                                                                            0.0s
 => CACHED [server base 3/5] RUN adduser     --disabled-password     --gecos ""     --home "/nonexistent"     --shell "/sbin/nologin"     -  0.0s
 => CACHED [server base 4/5] RUN --mount=type=cache,target=/root/.cache/pip     --mount=type=bind,source=requirements.txt,target=requiremen  0.0s
 => [server base 5/5] COPY . .                                                                                                       0.0s
 => [server] exporting to image                                                                                                      0.0s
 => => exporting layers                                                                                                              0.0s
 => => writing image sha256:bced9b8384df9bc0579ac9f6febcdde6065a4d1fa97ba9c2e3a1644c788ee9b5                                         0.0s
 => => naming to docker.io/library/docker-genai-sample-server                                                                        0.0s
[+] Running 2/1
 ✔ Container docker-genai-sample-database-1  Created                                                                                 0.0s
 ✔ Container docker-genai-sample-server-1    Recreated                                                                               0.0s
Attaching to database-1, server-1
database-1  | Changed password for user 'neo4j'. IMPORTANT: this change will only take effect if performed before the database is started for the
first time.
```

5. Access the application. Open a browser and view the application at http://localhost:8000 to see a simple Streamlit application.



6. Stop the application. In the terminal, press ctrl+c to stop the application.

```
^CGracefully stopping... (press Ctrl+C again to force)
[+] Stopping 0/1
 ⠋ Container docker-genai-sample-server-1  Stopping                                                    0.2s
[+] Killing 2/2
[+] Stopping 2/1ker-genai-sample-database-1  Killed                                                   0.2s
 ✔ Container docker-genai-sample-server-1    Stopped                                                  0.4s
 ✔ Container docker-genai-sample-database-1  Stopped                                                  0.0s
canceled
```

7. Add the Ollama service and a volume in your compose.yaml

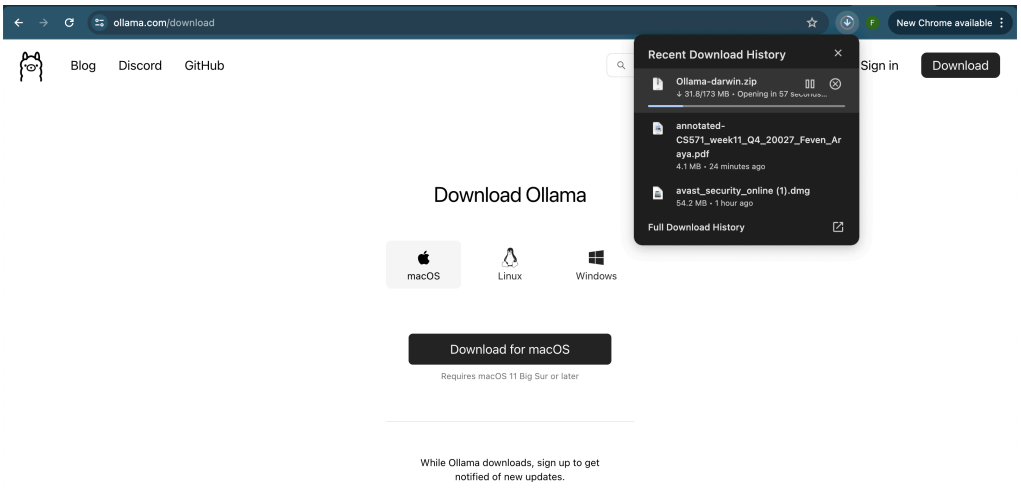
```
services:
  server:
    build:
      context: .
    ports:
      - 8000:8000
    env_file:
      - .env
    depends_on:
      database:
        condition: service_healthy

  database:
    image: neo4j:5.11
    ports:
      - "7474:7474"  # HTTP
      - "7687:7687"  # Bolt
    environment:
      - NEO4J_AUTH=${NEO4J_USERNAME}/${NEO4J_PASSWORD}
    healthcheck:
      test: ["CMD-SHELL", "wget --no-verbose --tries=1 --spider http://localhost:7474 || exit 1"]
      interval: 5s
      timeout: 3s
      retries: 5
  ollama:
    image: ollama/ollama:latest
    ports:
      - "11434:11434"
    volumes:
      - ollama_volume:/root/.ollama
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: all
              capabilities: [gpu]
```

```
  UW PICO 5.09                                                File: compose.yaml

    healthcheck:
      test: ["CMD-SHELL", "wget --no-verbose --tries=1 --spider http://localhost:7474 || exit 1"]
      interval: 5s
      timeout: 3s
      retries: 5
  ollama:
    image: ollama/ollama:latest
    ports:
      - "11434:11434"
    volumes:
      - ollama_volume:/root/.ollama
    deploy:
      resources:
        reservations:
          devices:
            - driver: nvidia
              count: all
              capabilities: [gpu]
volumes:
  ollama_volume:
```

8. Add the ollama-pull service to your compose.yaml file

```
  ollama-pull:
    image: docker/genai:ollama-pull
    env_file:
      - .env

volumes:
  ollama_volume: {}
```

9. To run Ollama in a container and provide GPU access- Install and run Ollama on your host machine.



10. Update the OLLAMA_BASE_URL value in your .env file to http://host.docker.internal:11434.

compose.yaml

```
  UW PICO 5.09                                                File: .env

#*********************************************************************
# LLM and Embedding Model
#*********************************************************************
LLM=llama2 # Set to "gpt-3.5" to use OpenAI.
EMBEDDING_MODEL=sentence_transformer

#*********************************************************************
# Neo4j
#*********************************************************************
NEO4J_URI=neo4j://database:7687
NEO4J_USERNAME=neo4j
NEO4J_PASSWORD=password

#*********************************************************************
# Ollama
#*********************************************************************
OLLAMA_BASE_URL=http://ollama:11434

#*********************************************************************
# OpenAI
#*********************************************************************
# Only required when using OpenAI LLM or embedding model
# OpenAI charges may apply. For details, see
# https://openai.com/pricing

#OPENAI_API_KEY=sk-..
```

11. Pull the model to Ollama using the following command.

```
(base) fevenbelay@Fevens-MacBook-Air docker-genai-sample % ollama pull llama2

pulling manifest
pulling 8934d96d3f08... 100% ▕████████████████████████████████████████▏ 3.8 GB
pulling 8c17c2ebb0ea... 100% ▕████████████████████████████████████████▏ 7.0 KB
pulling 7c23fb36d801... 100% ▕████████████████████████████████████████▏ 4.8 KB
pulling 2e0493f67d0c... 100% ▕████████████████████████████████████████▏   59 B
pulling fa304d675061... 100% ▕████████████████████████████████████████▏   91 B
pulling 42ba7f8a01dd... 100% ▕████████████████████████████████████████▏  557 B
verifying sha256 digest
writing manifest
removing any unused layers
success
```

12. Update the LLM value in your .env file to gpt-3.5.

```
#*****************************************************************
# LLM and Embedding Model
#*****************************************************************
LLM=gpt-3.5
EMBEDDING_MODEL=sentence_transformer

#*****************************************************************
# Neo4j
#*****************************************************************
NEO4J_URI=neo4j://database:7687
NEO4J_USERNAME=neo4j
NEO4J_PASSWORD=password

#*****************************************************************
# Ollama
#*****************************************************************
OLLAMA_BASE_URL=http://ollama:11434

#*****************************************************************
# OpenAI
#*****************************************************************
# Only required when using OpenAI LLM or embedding model
# OpenAI charges may apply. For details, see
# https://openai.com/pricing

#OPENAI_API_KEY=sk-..
```

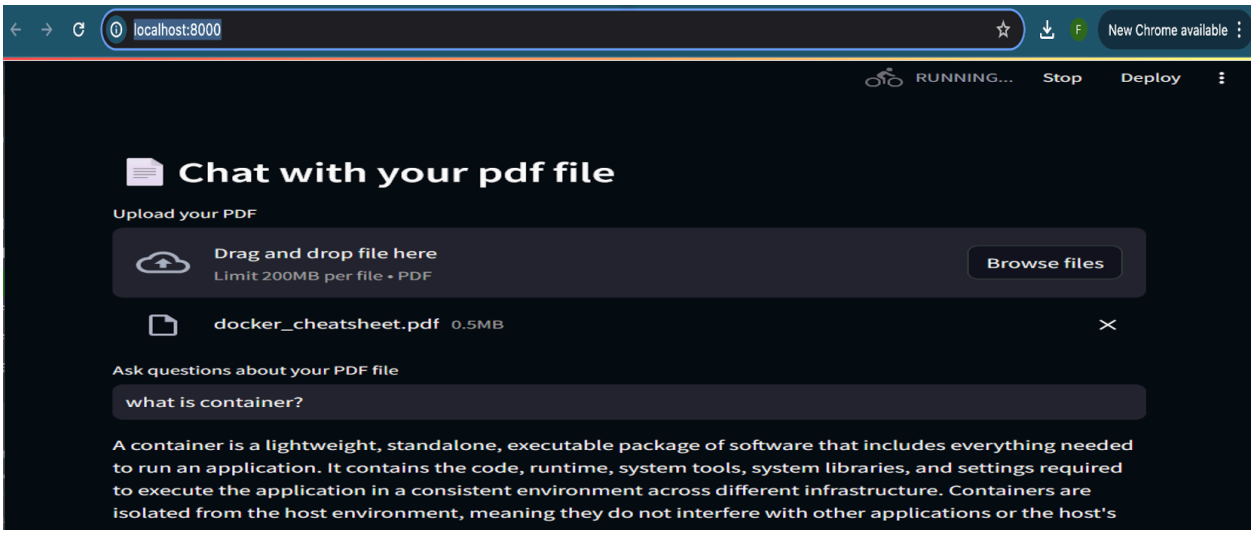
13. Uncomment and update the OPENAI_API_KEY value in your .env file to your OpenAI API key

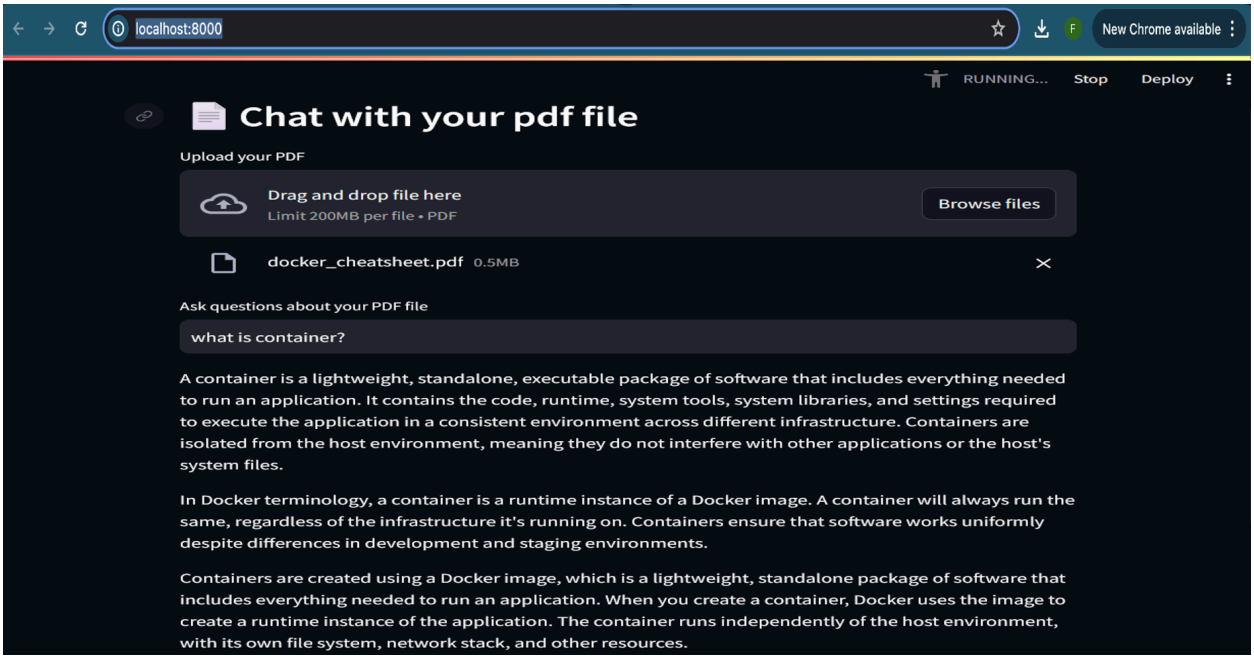
```
#*****************************************************************
# LLM and Embedding Model
#*****************************************************************
LLM=gpt-3.5
EMBEDDING_MODEL=sentence_transformer

#*****************************************************************
# Neo4j
#*****************************************************************
NEO4J_URI=neo4j://database:7687
NEO4J_USERNAME=neo4j
NEO4J_PASSWORD=password

#*****************************************************************
# Ollama
#*****************************************************************
OLLAMA_BASE_URL=http://ollama:11434

#*****************************************************************
# OpenAI
#*****************************************************************
# Only required when using OpenAI LLM or embedding model
# OpenAI charges may apply. For details, see
# https://openai.com/pricing

#OPENAI_API_KEY=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
```

14. To run all the services, run the following command in your docker-genai-sample directory:

```
(base) fevenbelay@Fevens-MacBook-Air docker-genai-sample % docker compose up --build
[+] Building 2.4s (12/12) FINISHED                                                                                  docker:desktop-linux
 => [server internal] load build definition from Dockerfile                                                                     0.0s
 => => transferring dockerfile: 1.71kB                                                                                          0.0s
 => [server] resolve image config for docker.io/docker/dockerfile:1                                                             1.7s
 => CACHED [server] docker-image://docker.io/docker/dockerfile:1@sha256:dbbd5e059e8a07ff7ea6233b213b36aa516b4c53c645f1817a4dd18b83cbea56     0.0s
 => [server internal] load metadata for docker.io/library/python:3.11.7-slim                                                    0.5s
 => [server internal] load .dockerignore                                                                                        0.0s
 => => transferring context: 667B                                                                                               0.0s
 => [server base 1/5] FROM docker.io/library/python:3.11.7-slim@sha256:53d6284a40eae6b625f22870f5faba6c54f2a28db9027408f4dee111f1e885a2     0.0s
 => [server internal] load build context                                                                                        0.0s
 => => transferring context: 252B                                                                                               0.0s
 => CACHED [server base 2/5] WORKDIR /app                                                                                       0.0s
 => CACHED [server base 3/5] RUN adduser     --disabled-password     --gecos ""     --home "/nonexistent"     --shell "/sbin/nologin"     -  0.0s
 => CACHED [server base 4/5] RUN --mount=type=cache,target=/root/.cache/pip     --mount=type=bind,source=requirements.txt,target=requiremen  0.0s
 => CACHED [server base 5/5] COPY . .                                                                                           0.0s
 => [server] exporting to image                                                                                                 0.0s
 => => exporting layers                                                                                                         0.0s
 => => writing image sha256:bced9b8384df9bc0579ac9f6febcdde6065a4d1fa97ba9c2e3a1644c788ee9b5                                    0.0s
 => => naming to docker.io/library/docker-genai-sample-server                                                                   0.0s
[+] Running 2/0
 ✓ Container docker-genai-sample-database-1     Created                                                                         0.0s
 ✓ Container docker-genai-sample-ollama-pull-1  Created                                                                         0.0s
```

- Once the application is running, open a browser and access the application using *http://localost:8000*
- Upload a PDF file, for example the Docker CLI Cheat Sheet, and ask a question about the PDF.


localhost:8000
New Chrome available
RUNNING...
Stop
Deploy
Chat with your pdf file
Upload your PDF
Drag and drop file here
Limit 200MB per file • PDF
Browse files
docker_cheatsheet.pdf 0.5MB
Ask questions about your PDF file
what is container?
A container is a lightweight, standalone, executable package of software that includes everything needed to run an application. It contains the code, runtime, system tools, system libraries, and settings required to execute the application in a consistent environment across different infrastructure. Containers are isolated from the host environment, meaning they do not interfere with other applications or the host's


localhost:8000
New Chrome available
RUNNING...
Stop
Deploy
Chat with your pdf file
Upload your PDF
Drag and drop file here
Limit 200MB per file • PDF
Browse files
docker_cheatsheet.pdf 0.5MB
Ask questions about your PDF file
what is container?
A container is a lightweight, standalone, executable package of software that includes everything needed to run an application. It contains the code, runtime, system tools, system libraries, and settings required to execute the application in a consistent environment across different infrastructure. Containers are isolated from the host environment, meaning they do not interfere with other applications or the host's system files.
In Docker terminology, a container is a runtime instance of a Docker image. A container will always run the same, regardless of the infrastructure it's running on. Containers ensure that software works uniformly despite differences in development and staging environments.
Containers are created using a Docker image, which is a lightweight, standalone package of software that includes everything needed to run an application. When you create a container, Docker uses the image to create a runtime instance of the application. The container runs independently of the host environment, with its own file system, network stack, and other resources.