Project Documentation: GenAI - Containerize your app

Name: MD ROKIBUL HASAN

ID: 19900

Introduction:

This project involves containerizing a Python-based Generative AI (GenAI) application using Docker. The primary goal is to demonstrate the process of deploying an AI application within a Docker container, emphasizing portability and efficiency across different environments.

Overview:

Objective: The aim is to containerize a Python-based GenAI application using Docker. Significance: The project seeks to merge advanced AI technology with modern deployment methods, aiming to enhance the accessibility and ease of deployment of AI applications.

Key Features: Focus areas include Docker containerization for consistent deployment, handling a Python-based AI application, and establishing a local development environment.

The below steps will cover the process, challenges encountered, and solutions found, providing a detailed guide for Dockerizing AI applications.

Prerequisites

- Ensure GPU acceleration is available if using Docker Desktop on Windows with WSL2 backend, or Docker Engine on Linux for GPU support.
- Docker Desktop or Docker Engine installed, with the latest version preferred.
- A git client, preferably command-line based, for cloning repositories.

Project Steps

The application is a modified version of the PDF Reader from the GenAI Stack demo applications. It's a full-stack Python application allowing queries about PDF files. Technologies used include LangChain for orchestration, Streamlit for UI, Ollama for running the LLM, and Neo4j for storing vectors.

Clone the repository: git clone https://github.com/craig-osterhout/docker-genai-sample

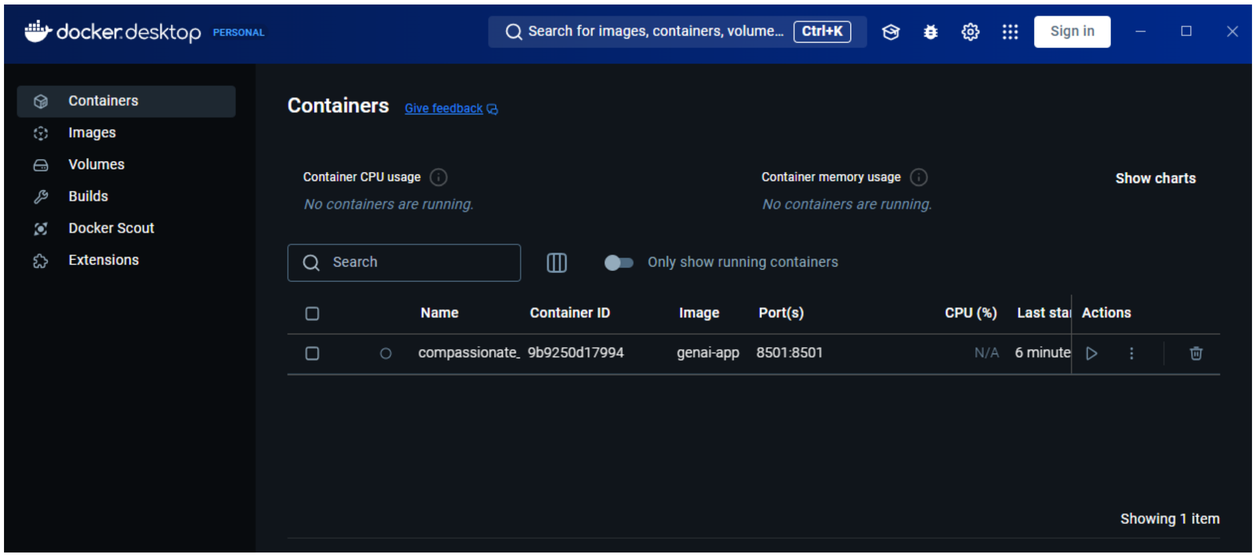
After cloning, the docker-genai-sample directory will include various files like .gitignore, app.py, chains.py, requirements.txt, util.py, LICENSE, and README.md.

Here’s a step-by-step guide to containerizing a Python-based Generative AI (GenAI) application on **Windows** using Docker. This guide assumes you have basic familiarity with command-line operations.

Step 1: Install Docker on Windows

1.1 Install Docker Desktop for Windows:

- **Download Docker Desktop**: Go to the official Docker website to download Docker Desktop for Windows:
  Docker Desktop Download
- **Install Docker**: Run the installer and follow the prompts to complete the installation. During installation, ensure that WSL2 (Windows Subsystem for Linux) is selected, as this will provide the best performance for Docker on Windows.
- **Enable WSL2 (if needed)**: If you don't have WSL2 enabled, Docker will prompt you to do so. Follow the instructions provided by Docker Desktop to enable WSL2.
- **Start Docker Desktop**: Once installed, start Docker Desktop. You may need to restart your computer for changes to take effect.
- **Verify Installation**: Open a command prompt or PowerShell and run the following command to verify Docker is working:



Container CPU usage
No containers are running.

Container memory usage
No containers are running.

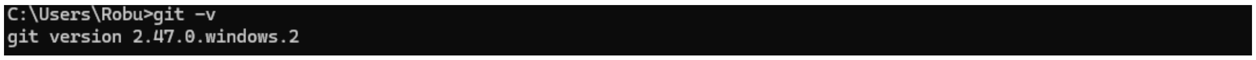
	Name	Container ID	Image	Port(s)	CPU (%)	Last sta	Actions
	compassionate_	9b9250d17994	genai-app	8501:8501	N/A	6 minute	

Step 2: Install Git on Windows

2.1 Install Git:

- **Download Git**: Go to the Git website and download the Git installer for Windows:
  [Git Download]
- **Verify Git Installation**: After installation, open the command prompt or PowerShell and verify Git is installed:

git –version



```
C:\Users\Robu>git -v
git version 2.47.0.windows.2
```

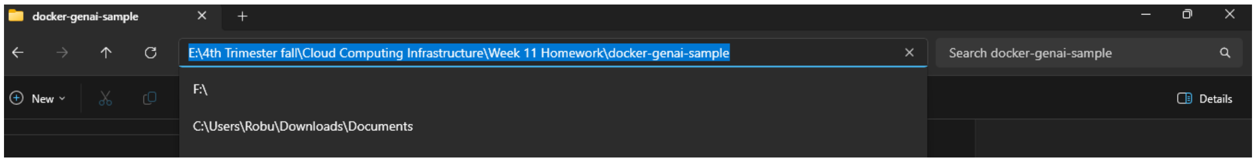
Step 3: Clone the GenAI Repository

3.1 Clone the Repository:

- Open **Git Bash** (or a command prompt) and navigate to the directory where you want to store the project.
- Clone the GenAI repository with this command:

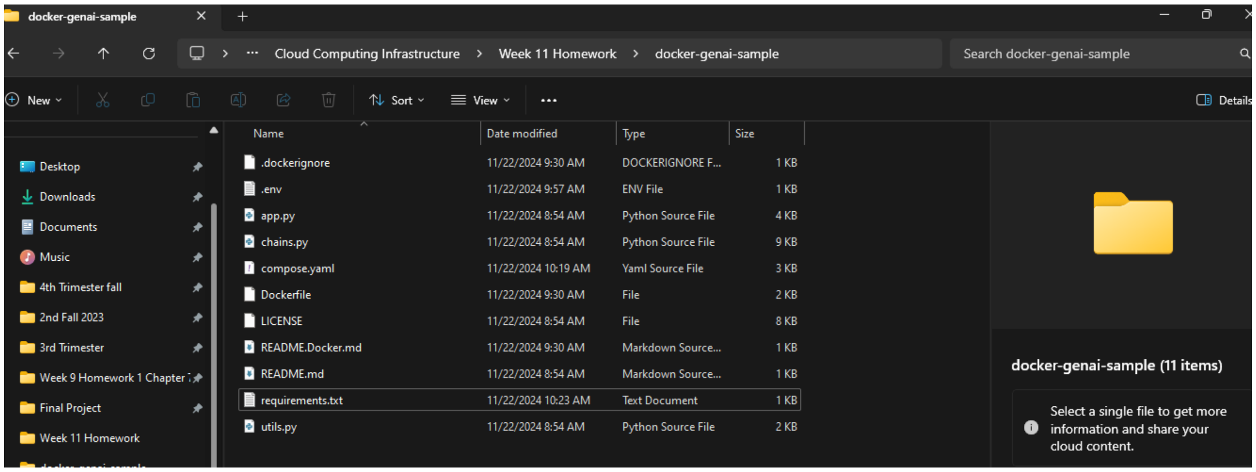
git clone https://github.com/craig-osterhout/docker-genai-sample

- Change into the docker-genai-sample directory:



```
E:\4th Trimester fall\Cloud Computing Infrastructure\Week 11 Homework\docker-genai-sample
F:\
C:\Users\Robu\Downloads\Documents
```

Path: E:\4th Trimester fall\Cloud Computing Infrastructure\Week 11 Homework\docker-genai-sample



Name	Date modified	Type	Size
.dockerignore	11/22/2024 9:30 AM	DOCKERIGNORE F...	1 KB
.env	11/22/2024 9:57 AM	ENV File	1 KB
app.py	11/22/2024 8:54 AM	Python Source File	4 KB
chains.py	11/22/2024 8:54 AM	Python Source File	9 KB
compose.yaml	11/22/2024 10:19 AM	Yaml Source File	3 KB
Dockerfile	11/22/2024 9:30 AM	File	2 KB
LICENSE	11/22/2024 8:54 AM	File	8 KB
README.Docker.md	11/22/2024 9:30 AM	Markdown Source...	1 KB
README.md	11/22/2024 8:54 AM	Markdown Source...	1 KB
requirements.txt	11/22/2024 10:23 AM	Text Document	1 KB
utils.py	11/22/2024 8:54 AM	Python Source File	2 KB

docker-genai-sample (11 items)

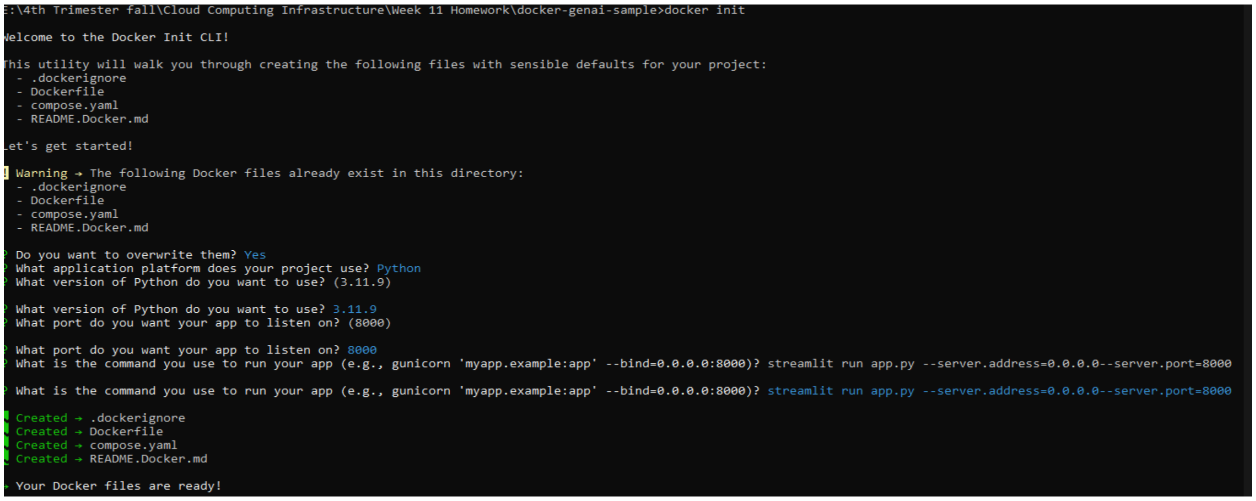
Select a single file to get more information and share your cloud content.

You should now have the project files on your local machine.

Step 4: Create the Dockerfile

In the root of the docker-genai-sample project run this command: **docker init**

Then ans the following question one by one:



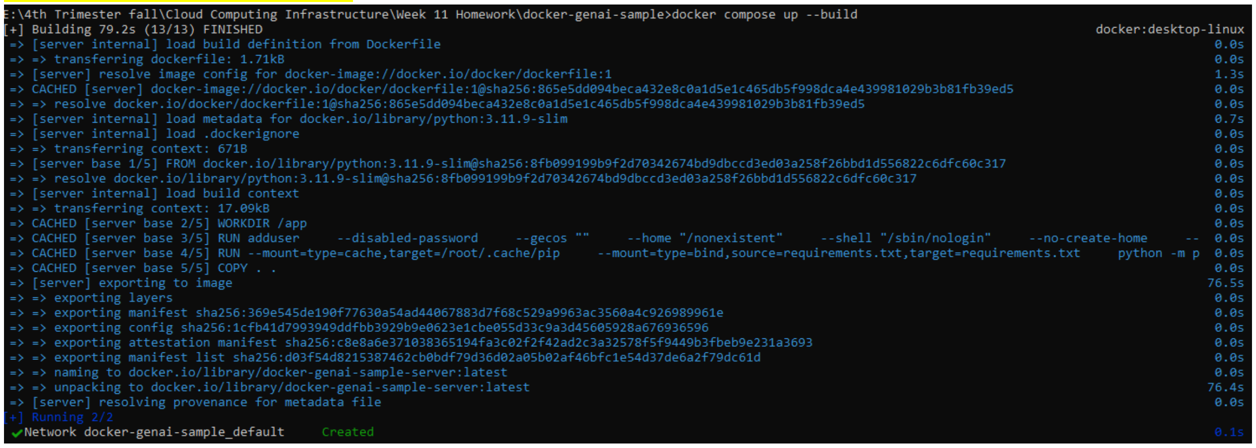
```
E:\4th Trimester fall\Cloud Computing Infrastructure\Week 11 Homework\docker-genai-sample>docker init

Welcome to the Docker Init CLI!

This utility will walk you through creating the following files with sensible defaults for your project:
  - .dockerignore
  - Dockerfile
  - compose.yaml
  - README.Docker.md

Let's get started!

! Warning → The following Docker files already exist in this directory:
  - .dockerignore
  - Dockerfile
  - compose.yaml
  - README.Docker.md

? Do you want to overwrite them? Yes
? What application platform does your project use? Python
? What version of Python do you want to use? (3.11.9)

? What version of Python do you want to use? 3.11.9
? What port do you want your app to listen on? (8000)

? What port do you want your app to listen on? 8000
? What is the command you use to run your app (e.g., gunicorn 'myapp.example:app' --bind=0.0.0.0:8000)? streamlit run app.py --server.address=0.0.0.0--server.port=8000

? What is the command you use to run your app (e.g., gunicorn 'myapp.example:app' --bind=0.0.0.0:8000)? streamlit run app.py --server.address=0.0.0.0--server.port=8000

✔ Created → .dockerignore
✔ Created → Dockerfile
✔ Created → compose.yaml
✔ Created → README.Docker.md

→ Your Docker files are ready!
```

5. Run the Application:

Step 1: Ensure Docker Desktop is running on your machine. You can start it manually if it's not already running. On a Mac, you can open Docker Desktop from the Applications folder, Launchpad, or Spotlight search.

Step 2: To run the application, execute the below command from the root directory of your project:

docker compose up --build



```
E:\4th Trimester fall\Cloud Computing Infrastructure\Week 11 Homework\docker-genai-sample>docker compose up --build
[+] Building 79.2s (13/13) 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-image://docker.io/docker/dockerfile:1                                           1.3s
 => CACHED [server] docker-image://docker.io/docker/dockerfile:1@sha256:865e5dd094beca432e8c0a1d5e1c465db5f998dca4e439981029b3b81fb39ed5   0.0s
 => => resolve docker.io/docker/dockerfile:1@sha256:865e5dd094beca432e8c0a1d5e1c465db5f998dca4e439981029b3b81fb39ed5         0.0s
 => [server internal] load metadata for docker.io/library/python:3.11.9-slim                                                 0.7s
 => [server internal] load .dockerignore                                                                                     0.0s
 => => transferring context: 671B                                                                                            0.0s
 => [server base 1/5] FROM docker.io/library/python:3.11.9-slim@sha256:8fb099199b9f2d70342674bd9dbccd3ed03a258f26bbd1d556822c6dfc60c317   0.0s
 => => resolve docker.io/library/python:3.11.9-slim@sha256:8fb099199b9f2d70342674bd9dbccd3ed03a258f26bbd1d556822c6dfc60c317 0.0s
 => [server internal] load build context                                                                                     0.0s
 => => transferring context: 17.09kB                                                                                         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"     --no-create-home     --  0.0s
 => CACHED [server base 4/5] RUN --mount=type=cache,target=/root/.cache/pip     --mount=type=bind,source=requirements.txt,target=requirements.txt     python -m p  0.0s
 => CACHED [server base 5/5] COPY . .                                                                                        0.0s
 => [server] exporting to image                                                                                             76.5s
 => => exporting layers                                                                                                      0.0s
 => => exporting manifest sha256:369e545de190f77630a54ad44067883d7f68c529a9963ac3560a4c926989961e                            0.0s
 => => exporting config sha256:1cfb41d7993949ddfbb3929b9e0623e1cbe055d33c9a3d45605928a676936596                              0.0s
 => => exporting attestation manifest sha256:c8e8a6e371038365194fa3c02f2f42ad2c3a32578f5f9449b3fbeb9e231a3693                0.0s
 => => exporting manifest list sha256:d03f54d8215387462cb0bdf79d36d02a05b02af46bfc1e54d37de6a2f79dc61d                       0.0s
 => => naming to docker.io/library/docker-genai-sample-server:latest                                                         0.0s
 => => unpacking to docker.io/library/docker-genai-sample-server:latest                                                     76.4s
 => [server] resolving provenance for metadata file                                                                          0.0s
[+] Running 2/2
 ✔Network docker-genai-sample_default     Created                                                                            0.1s
```

Step 3: After building, which may take a few minutes, access the application at http://localhost:8501

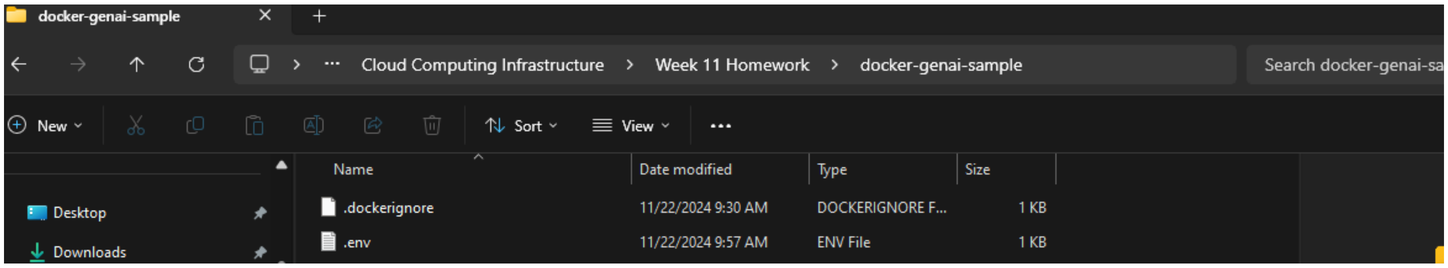
Next Steps: Setting Up Neo4J Database and LLM Service

The next step is to set up a development environment to access all the services that the generative AI (GenAI) application needs. This includes:

- Adding a local database
- Adding a local or remote LLM service

1. Add a Local Database

- Rename env.example to .env in the cloned repository's directory.



docker-genai-sample
Cloud Computing Infrastructure > Week 11 Homework > docker-genai-sample
Search docker-genai-sa
New
Sort
View
Desktop
Downloads

Name	Date modified	Type	Size
.dockerignore	11/22/2024 9:30 AM	DOCKERIGNORE F...	1 KB
.env	11/22/2024 9:57 AM	ENV File	1 KB

- Update the **compose.yaml** file to include a Neo4j database service (the section in yellow)

  and specify environment variables for database connection.
  services:
  server:
  build:
  context: .

```
ports:
- 80501:8501
env_file:
- .env
depends_on:
database:
condition: service_healthy
database:
image: neo4j:5.11
ports:
- "7474:7474"
- "7687:7687"
environment:
- NEO4J_AUTH=${NEO4J_USERNAME}/${NEO4J_PASSWORD}
healthcheck:
test: ["CMD-SHELL", "wget --no-verbose --tries=1 --spider localhost:7474 || exit 1"]
interval: 5s
timeout: 3s
retries: 5
```

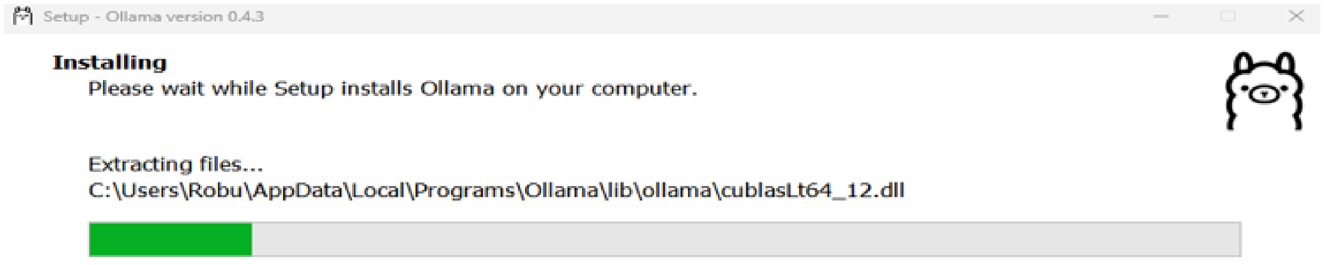
2. Add a Local or Remote LLM Service

For the GenAI application, a Large Language Model (LLM) service is essential for sophisticated natural language processing. On an Apple Silicon Mac, it is recommended to run the LLM service, specifically Ollama, outside of a container. This approach is advised due to the current limitations in containerized environments on Apple Silicon regarding optimal performance and GPU support.

Running Ollama outside of a container enables the application to leverage the full capabilities of the Mac's hardware for processing complex language tasks, such as text analysis, query responses, and language generation. This setup ensures that the GenAI application maintains high efficiency, scalability, and accesses the latest developments in language AI technology, all while being aligned with the specific hardware capabilities of Apple Silicon Macs.

To run Ollama outside of a container: Install Ollama from the link below and run it on your host machine.

https://github.com/ollama/ollama?tab=readme-ov-file


Setup - Ollama version 0.4.3

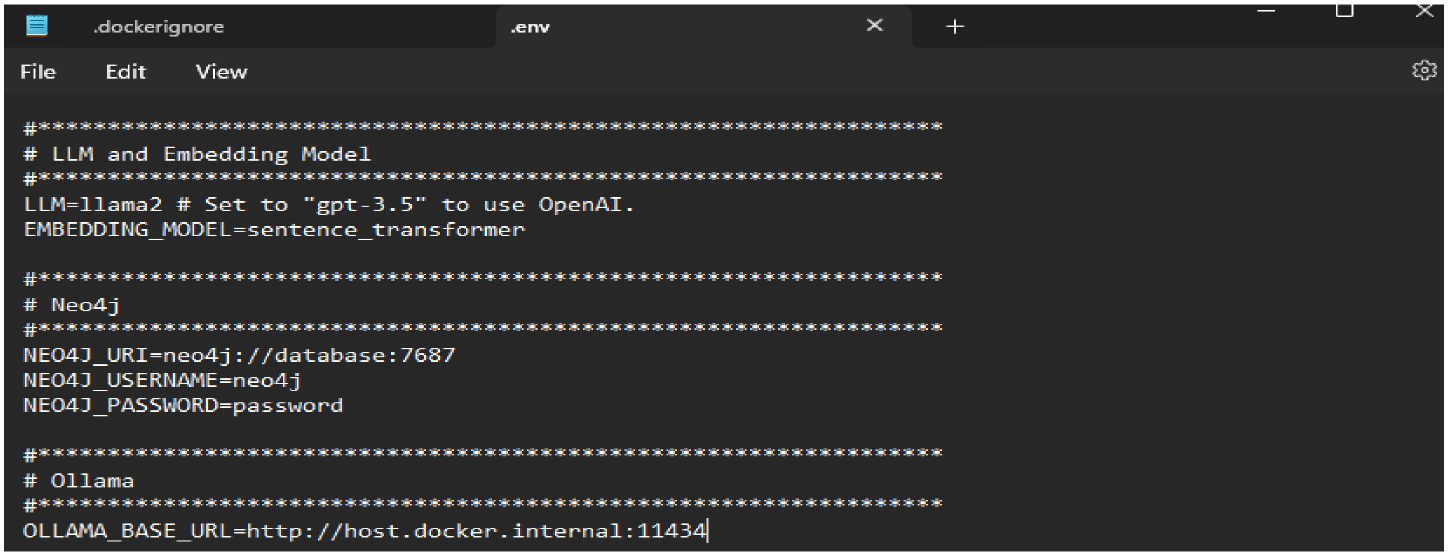
Installing

Please wait while Setup installs Ollama on your computer.

Extracting files...

C:\Users\Robu\AppData\Local\Programs\Ollama\lib\ollama\cublasLt64_12.dll

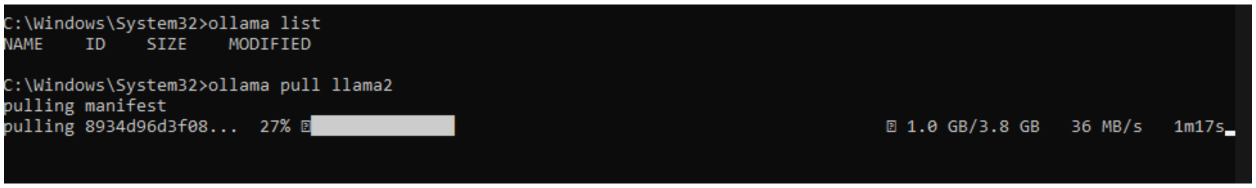
Update the OLLAMA_BASE_URL value in your .env file to http://host.docker.internal:11434.



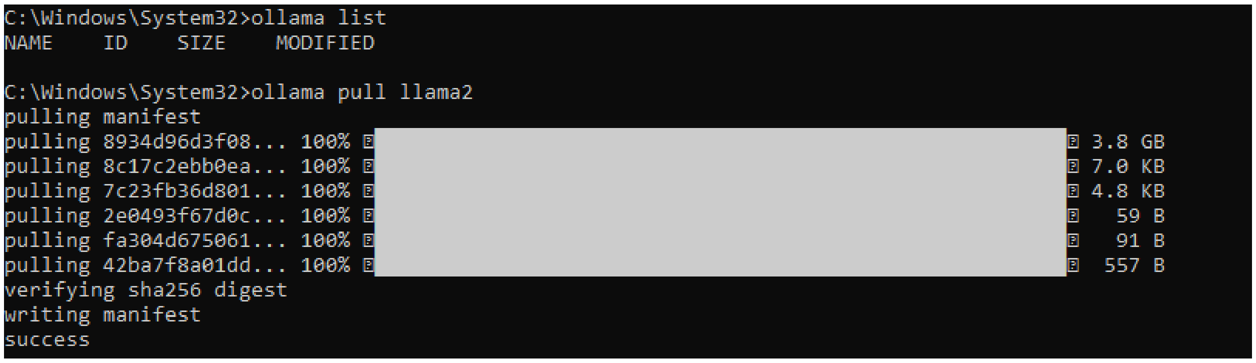
```
#*****************************************************************
# LLM and Embedding Model
#*****************************************************************
LLM=llama2 # Set to "gpt-3.5" to use OpenAI.
EMBEDDING_MODEL=sentence_transformer

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

#*****************************************************************
# Ollama
#*****************************************************************
OLLAMA_BASE_URL=http://host.docker.internal:11434
```

Pull the model to Ollama using the following command.

ollama pull llama2



```
C:\Windows\System32>ollama list
NAME    ID    SIZE    MODIFIED

C:\Windows\System32>ollama pull llama2
pulling manifest
pulling 8934d96d3f08...  27%                                       1.0 GB/3.8 GB   36 MB/s   1m17s
```



```
C:\Windows\System32>ollama list
NAME     ID     SIZE     MODIFIED

C:\Windows\System32>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
success
```

- Now, you have the following services in your Compose file:
  - Server service for your main GenAI application
  - Database service to store vectors in a Neo4j database
  - Ollama service to run the LLM

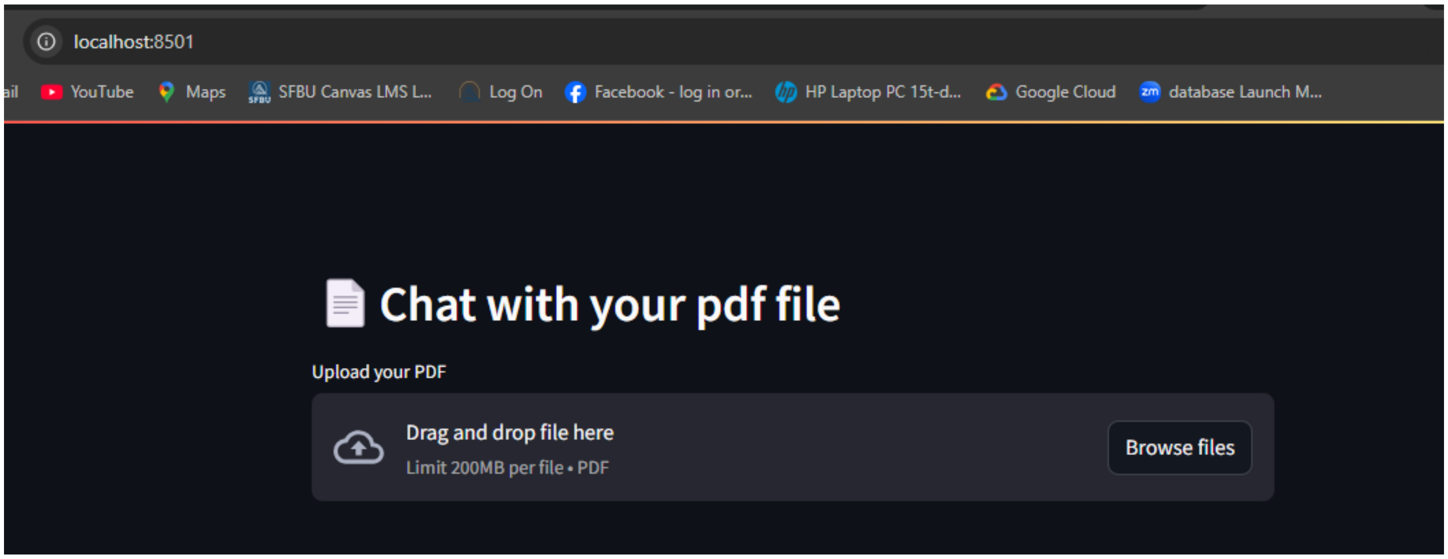
Keep in mind that there are alternatives to running Ollama locally. For example you may use OpenAI API or run Ollama in a container if you are on Linux.

3. Run the GenAI Application

To run the application, execute the below command from the root directory of your project

docker compose up --build

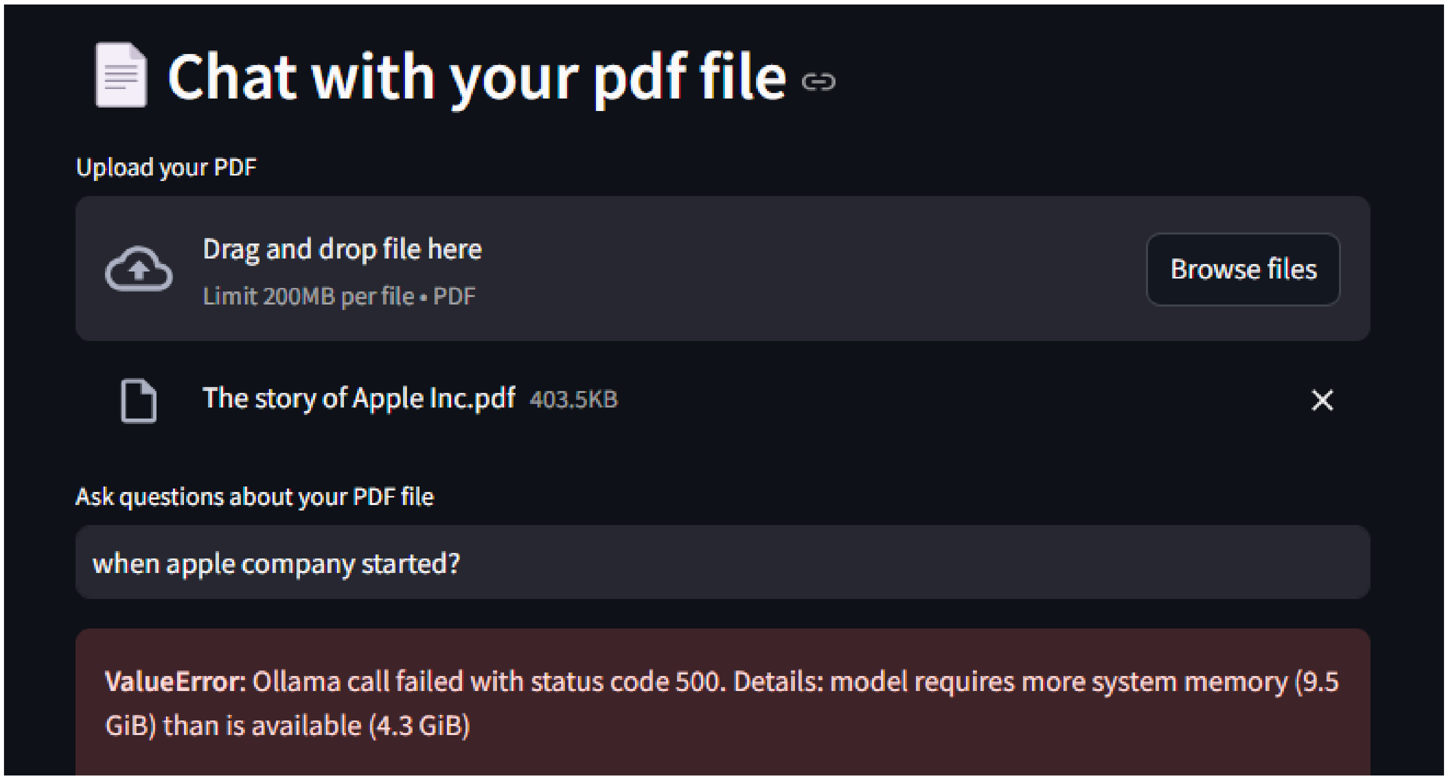
Access the application at **http://localhost:8501** after all services are up and **running.**


localhost:8501
ail YouTube Maps SFBU Canvas LMS L... Log On Facebook - log in or... HP Laptop PC 15t-d... Google Cloud database Launch M...

Chat with your pdf file

Upload your PDF

Drag and drop file here
Limit 200MB per file • PDF



Chat with your pdf file

Upload your PDF

Drag and drop file here
Limit 200MB per file • PDF

The story of Apple Inc.pdf 403.5KB

Ask questions about your PDF file

when apple company started?

ValueError: Ollama call failed with status code 500. Details: model requires more system memory (9.5 GiB) than is available (4.3 GiB)

Note: The app is running but need more system memory to run perfectly. In my machine, only 4.3 GB is available while requirement is 9.5 GB.

Thanks!