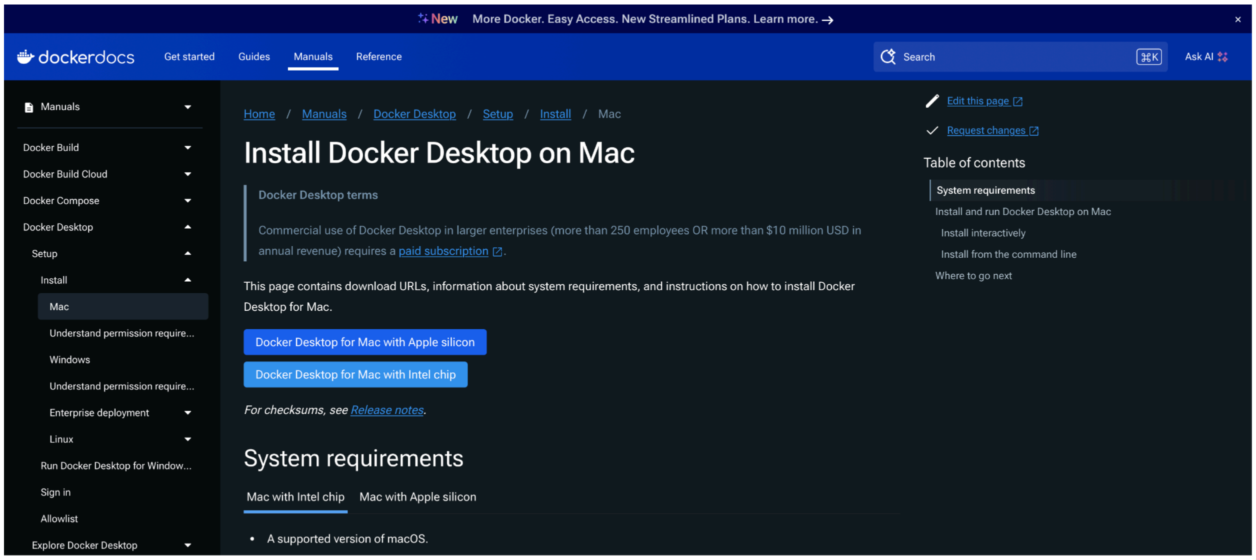
Rashmi Purandare 20177

Week 11 Homework 2: Project: GenAI - Develop your containerized app

Step 1: GenAI - Containerize your app

1. First, install the latest version of Docker Desktop.


New More Docker. Easy Access. New Streamlined Plans. Learn more. →
dockerdocs
Get started
Guides
Manuals
Reference
Search
Ask AI
Manuals
Docker Build
Docker Build Cloud
Docker Compose
Docker Desktop
Setup
Install
Mac
Understand permission require...
Windows
Understand permission require...
Enterprise deployment
Linux
Run Docker Desktop for Window...
Sign in
Allowlist
Explore Docker Desktop
Home / Manuals / Docker Desktop / Setup / Install / Mac
Edit this page
Request changes
Table of contents
System requirements
Install and run Docker Desktop on Mac
Install interactively
Install from the command line
Where to go next
Docker Desktop for Mac with Apple silicon
Docker Desktop for Mac with Intel chip
Mac with Intel chip
Mac with Apple silicon

Install Docker Desktop on Mac

Docker Desktop terms

Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than $10 million USD in annual revenue) requires a paid subscription.

This page contains download URLs, information about system requirements, and instructions on how to install Docker Desktop for Mac.

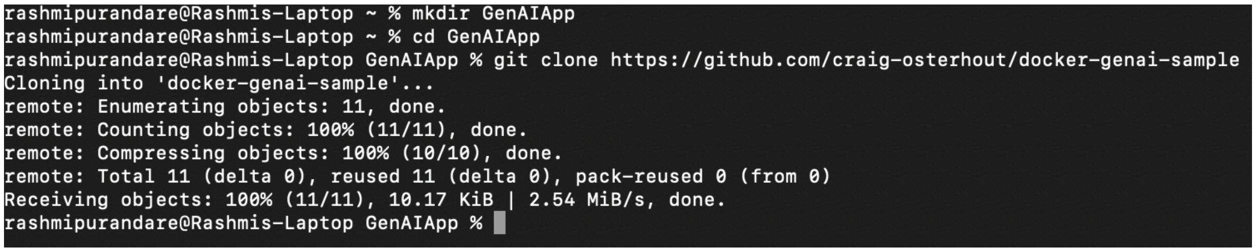
For checksums, see Release notes.

System requirements

- A supported version of macOS.

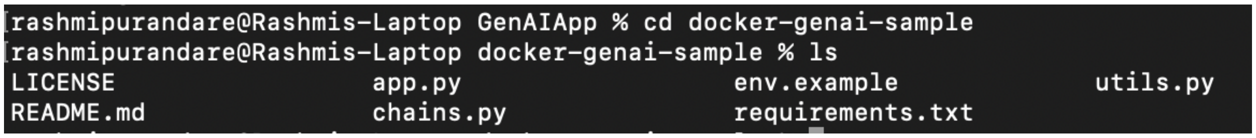
2. Go to the terminal and navigate to working directory.

3. Clone the sample application. We run the following command to clone the repository:
git clone https://github.com/craig-osterhout/docker-genai-sample



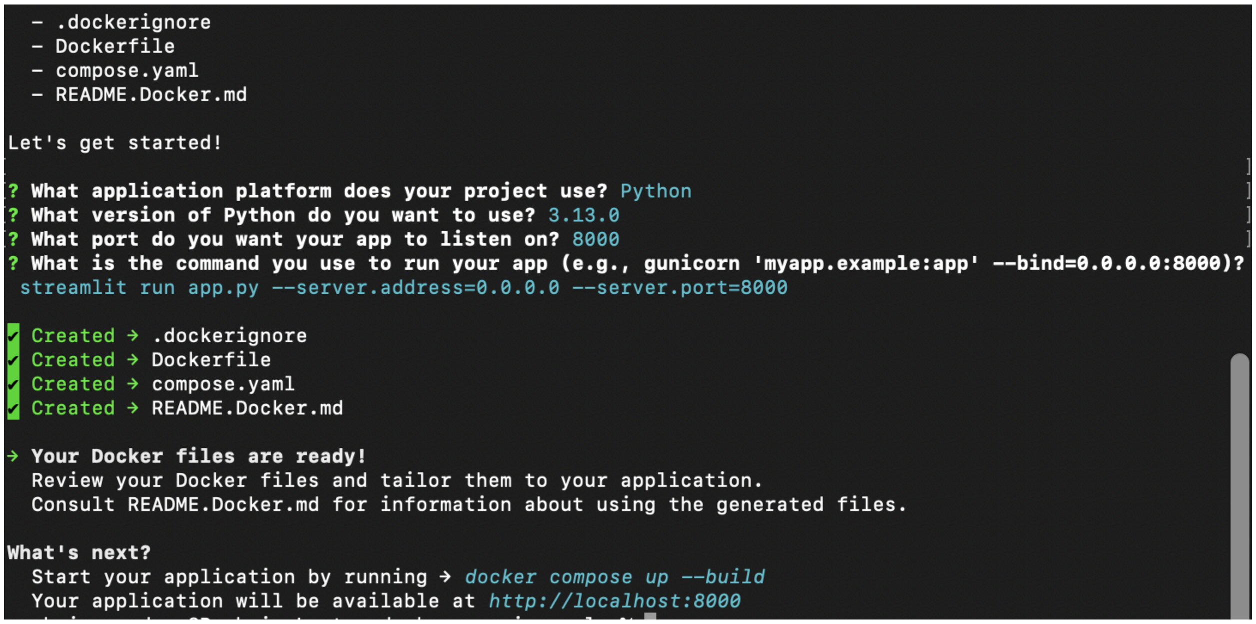
```
rashmipurandare@Rashmis-Laptop ~ % mkdir GenAIApp
rashmipurandare@Rashmis-Laptop ~ % cd GenAIApp
rashmipurandare@Rashmis-Laptop GenAIApp % git clone https://github.com/craig-osterhout/docker-genai-sample
Cloning into 'docker-genai-sample'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 11 (delta 0), reused 11 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (11/11), 10.17 KiB | 2.54 MiB/s, done.
rashmipurandare@Rashmis-Laptop GenAIApp %
```

• You should now have the following files in your docker-genai-sample directory.



```
[rashmipurandare@Rashmis-Laptop GenAIApp % cd docker-genai-sample
[rashmipurandare@Rashmis-Laptop docker-genai-sample % ls
LICENSE                 app.py                  env.example             utils.py
README.md               chains.py               requirements.txt
```

4. Now that we have an application, we can use docker init to create the necessary Docker assets to containerize our application. Inside the docker-genai-sample directory, run the docker init command.



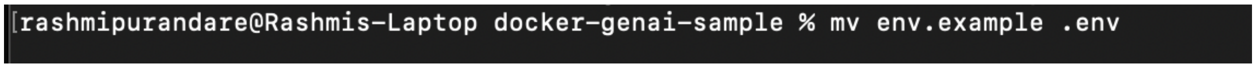
```
  - .dockerignore
  - Dockerfile
  - compose.yaml
  - README.Docker.md

Let's get started!

? What application platform does your project use? Python
? What version of Python do you want to use? 3.13.0
? 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

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

→ Your Docker files are ready!
  Review your Docker files and tailor them to your application.
  Consult README.Docker.md for information about using the generated files.

What's next?
  Start your application by running → docker compose up --build
  Your application will be available at http://localhost:8000
```

Step 2: GenAI - Develop your app

Adding a Local Database

Here we will update the compose.yaml file to define a database service, and we will specify an environment variables file to load the database connection information rather than manually entering the information every time. To run the database service:

1. In the cloned repository's directory, rename env.example file to .env. This file contains the environment variables that the containers will use.



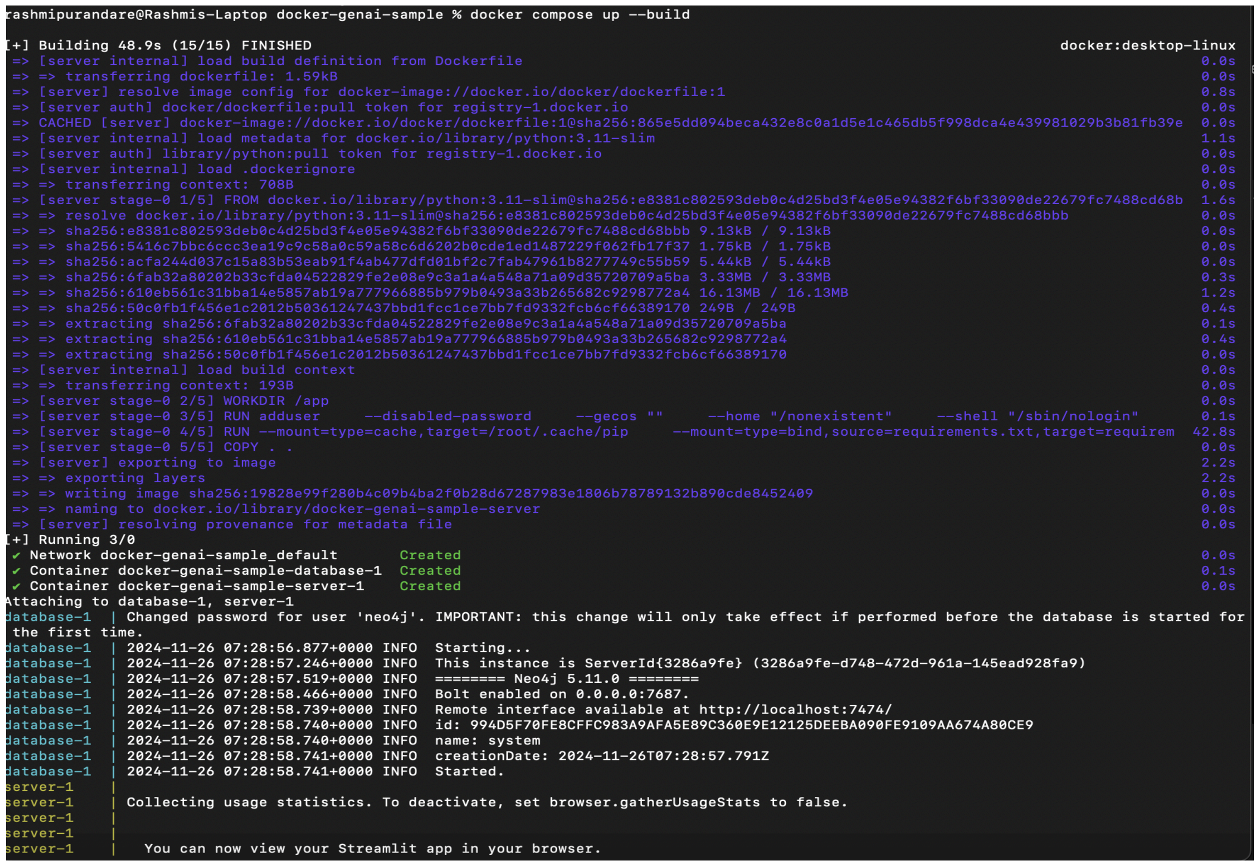
```
rashmipurandare@Rashmis-Laptop docker-genai-sample % mv env.example .env
```

2. Then open the compose.yaml file in an IDE or text editor.
$ nano compose.yaml

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

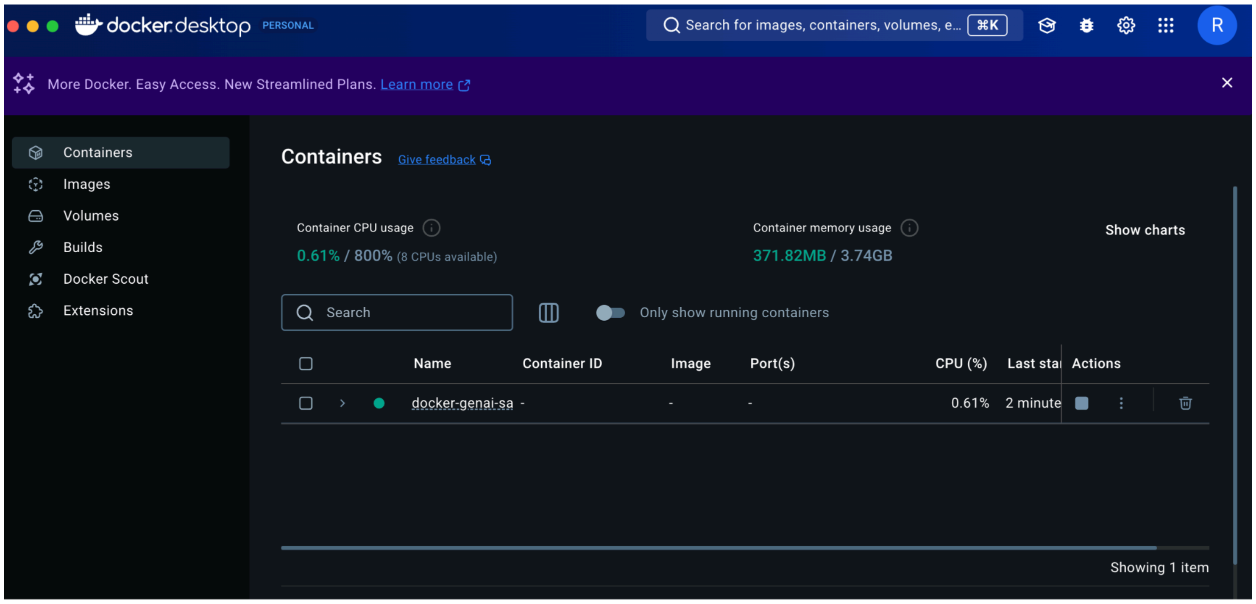
```
services:
  server:
    build:
      context: .
    ports:
      - 8000:8000
    env_file:
      - .env
    depends_on:
      database:
        condition: service_healthy

  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
```

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



```
rashmipurandare@Rashmis-Laptop docker-genai-sample % docker compose up --build

[+] Building 48.9s (15/15) FINISHED                                                                                  docker:desktop-linux
 => [server internal] load build definition from Dockerfile                                                                          0.0s
 => => transferring dockerfile: 1.59kB                                                                                               0.0s
 => [server] resolve image config for docker-image://docker.io/docker/dockerfile:1                                                   0.8s
 => [server auth] docker/dockerfile:pull token for registry-1.docker.io                                                              0.0s
 => CACHED [server] docker-image://docker.io/docker/dockerfile:1@sha256:865e5dd094beca432e8c0a1d5e1c465db5f998dca4e439981029b3b81fb39e  0.0s
 => [server internal] load metadata for docker.io/library/python:3.11-slim                                                           1.1s
 => [server auth] library/python:pull token for registry-1.docker.io                                                                 0.0s
 => [server internal] load .dockerignore                                                                                             0.0s
 => => transferring context: 708B                                                                                                    0.0s
 => [server stage-0 1/5] FROM docker.io/library/python:3.11-slim@sha256:e8381c802593deb0c4d25bd3f4e05e94382f6bf33090de22679fc7488cd68b  1.6s
 => => resolve docker.io/library/python:3.11-slim@sha256:e8381c802593deb0c4d25bd3f4e05e94382f6bf33090de22679fc7488cd68bbb             0.0s
 => => sha256:e8381c802593deb0c4d25bd3f4e05e94382f6bf33090de22679fc7488cd68bbb 9.13kB / 9.13kB                                       0.0s
 => => sha256:5416c7bbc6ccc3ea19c9c58a0c59a58c6d6202b0cde1ed1487229f062fb17f37 1.75kB / 1.75kB                                       0.0s
 => => sha256:acfa244d037c15a83b53eab91f4ab477dfd01bf2c7fab47961b8277749c55b59 5.44kB / 5.44kB                                       0.0s
 => => sha256:6fab32a80202b33cfda04522829fe2e08e9c3a1a4a548a71a09d35720709a5ba 3.33MB / 3.33MB                                       0.3s
 => => sha256:610eb561c31bba14e5857ab19a777966885b979b0493a33b265682c9298772a4 16.13MB / 16.13MB                                     1.2s
 => => sha256:50c0fb1f456e1c2012b50361247437bbd1fcc1ce7bb7fd9332fcb6cf66389170 249B / 249B                                           0.4s
 => => extracting sha256:6fab32a80202b33cfda04522829fe2e08e9c3a1a4a548a71a09d35720709a5ba                                            0.1s
 => => extracting sha256:610eb561c31bba14e5857ab19a777966885b979b0493a33b265682c9298772a4                                            0.4s
 => => extracting sha256:50c0fb1f456e1c2012b50361247437bbd1fcc1ce7bb7fd9332fcb6cf66389170                                            0.0s
 => [server internal] load build context                                                                                             0.0s
 => => transferring context: 193B                                                                                                    0.0s
 => [server stage-0 2/5] WORKDIR /app                                                                                                0.0s
 => [server stage-0 3/5] RUN adduser     --disabled-password     --gecos ""     --home "/nonexistent"     --shell "/sbin/nologin"     0.1s
 => [server stage-0 4/5] RUN --mount=type=cache,target=/root/.cache/pip     --mount=type=bind,source=requirements.txt,target=requirem  42.8s
 => [server stage-0 5/5] COPY . .                                                                                                    0.0s
 => [server] exporting to image                                                                                                      2.2s
 => => exporting layers                                                                                                              2.2s
 => => writing image sha256:19828e99f280b4c09b4ba2f0b28d67287983e1806b78789132b890cde8452409                                         0.0s
 => => naming to docker.io/library/docker-genai-sample-server                                                                        0.0s
 => [server] resolving provenance for metadata file                                                                                  0.0s
[+] Running 3/0
 ✔ Network docker-genai-sample_default       Created                                                                                 0.0s
 ✔ Container docker-genai-sample-database-1  Created                                                                                 0.1s
 ✔ Container docker-genai-sample-server-1    Created                                                                                 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.
database-1  | 2024-11-26 07:28:56.877+0000 INFO  Starting...
database-1  | 2024-11-26 07:28:57.246+0000 INFO  This instance is ServerId{3286a9fe} (3286a9fe-d748-472d-961a-145ead928fa9)
database-1  | 2024-11-26 07:28:57.519+0000 INFO  ======== Neo4j 5.11.0 ========
database-1  | 2024-11-26 07:28:58.466+0000 INFO  Bolt enabled on 0.0.0.0:7687.
database-1  | 2024-11-26 07:28:58.739+0000 INFO  Remote interface available at http://localhost:7474/
database-1  | 2024-11-26 07:28:58.740+0000 INFO  id: 994D5F70FE8CFFC983A9AFA5E89C360E9E12125DEEBA090FE9109AA674A80CE9
database-1  | 2024-11-26 07:28:58.740+0000 INFO  name: system
database-1  | 2024-11-26 07:28:58.741+0000 INFO  creationDate: 2024-11-26T07:28:57.791Z
database-1  | 2024-11-26 07:28:58.741+0000 INFO  Started.
server-1    |
server-1    | Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.
server-1    |
server-1    |
server-1    |   You can now view your Streamlit app in your browser.
```

• We can also see the progress from Docker Desktop.


docker desktop PERSONAL
Search for images, containers, volumes, e... ⌘K
More Docker. Easy Access. New Streamlined Plans. Learn more
Containers
Images
Volumes
Builds
Docker Scout
Extensions
Give feedback
Show charts
Search
Only show running containers
Showing 1 item

Containers

Container CPU usage 0.61% / 800% (8 CPUs available)

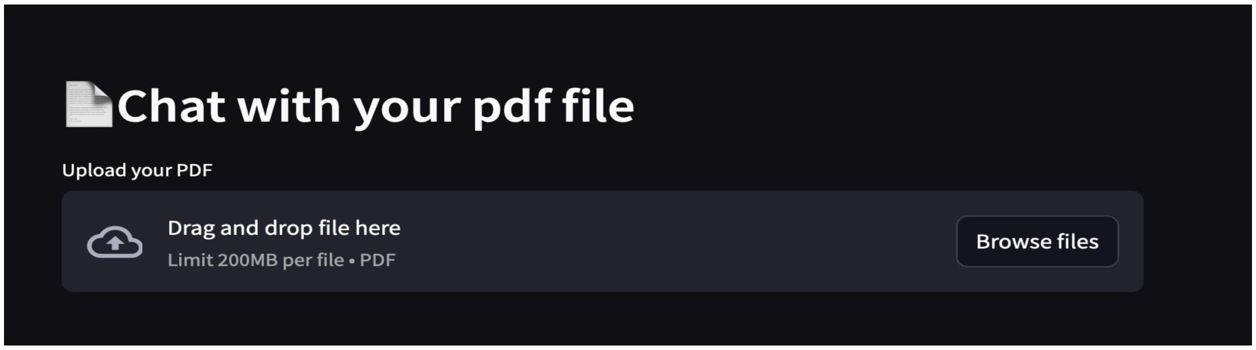
Container memory usage 371.82MB / 3.74GB

	Name	Container ID	Image	Port(s)	CPU (%)	Last sta	Actions
●	docker-genai-sa	-	-	-	0.61%	2 minute	

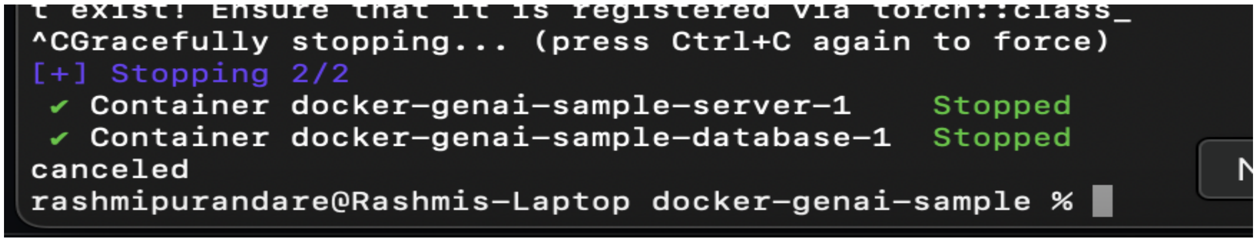
5. Access the application. Open a browser and view the application at http://localhost:8000.

You should see a simple Streamlit application.

• Note that asking questions to a PDF will cause the application to fail because the LLM service specified in the .env file isn't running yet.



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



```
t exist! Ensure that it is registered via torch::class_
^CGracefully stopping... (press Ctrl+C again to force)
[+] Stopping 2/2
 ✔ Container docker-genai-sample-server-1     Stopped
 ✔ Container docker-genai-sample-database-1   Stopped
canceled
rashmipurandare@Rashmis-Laptop docker-genai-sample % 
```

Rashmi Purandare 20177

Week 11 Homework 2: Project: GenAI - Develop your containerized app

Adding a Local or Remote LLM Service

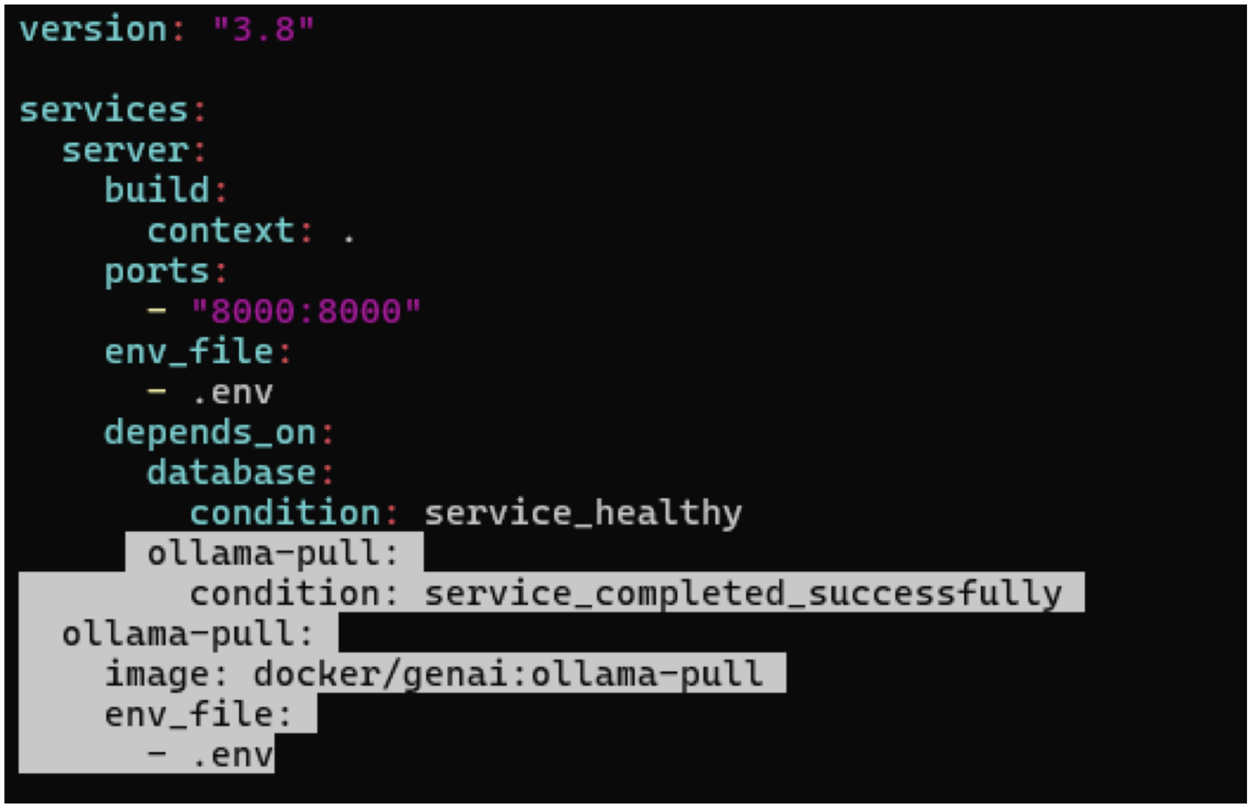
1. Install the prerequisites.
• For Docker Engine on Linux, install the NVIDIA Container Toolkit.
• For Docker Desktop on Windows 10/11, install the latest NVIDIA driver and make sure you are using the WSL2 backend

2. Add the Ollama service and a volume in your compose.yaml. The following is the updated 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"
      - "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

  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:
```

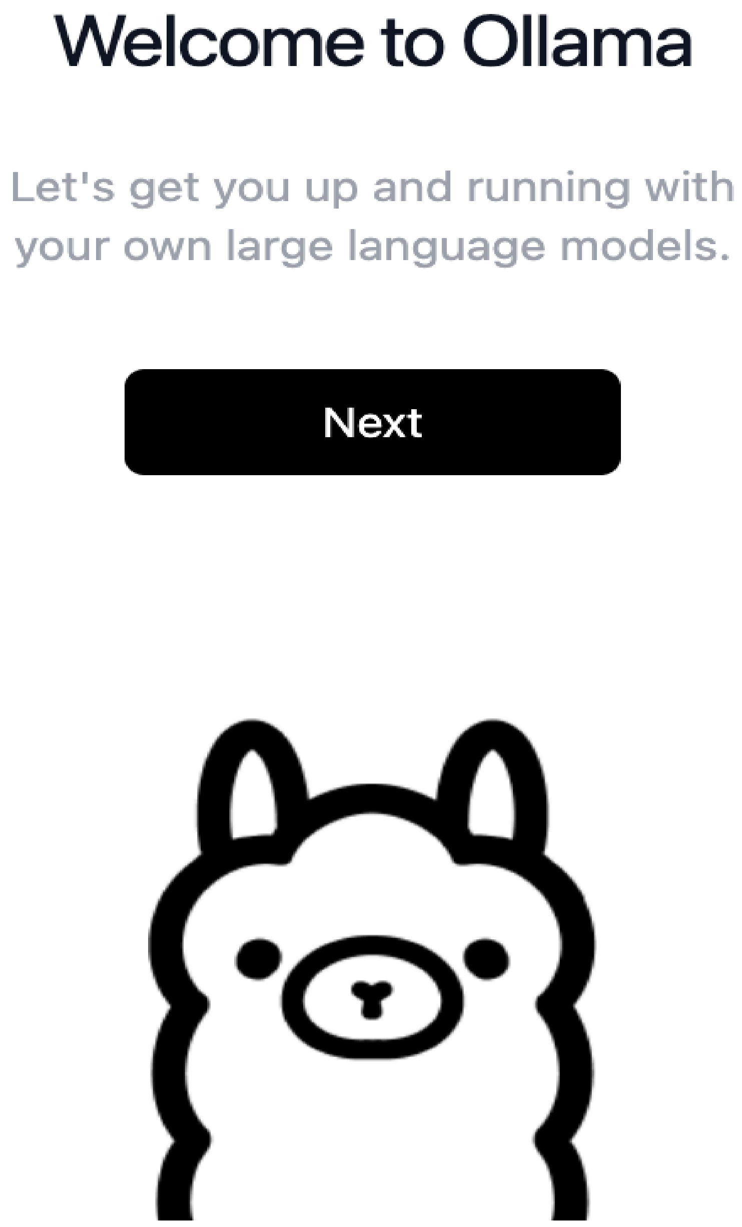
3. Add the ollama-pull service to your compose.yaml file. This service uses the docker/genai:ollama-pull image, based on the GenAI Stack's pull_model.Dockerfile and will automatically pull the model for your Ollama container. The following is the updated section of the compose.yaml file:



```
version: "3.8"

services:
  server:
    build:
      context: .
    ports:
      - "8000:8000"
    env_file:
      - .env
    depends_on:
      database:
        condition: service_healthy
      ollama-pull:
        condition: service_completed_successfully
  ollama-pull:
    image: docker/genai:ollama-pull
    env_file:
      - .env
```

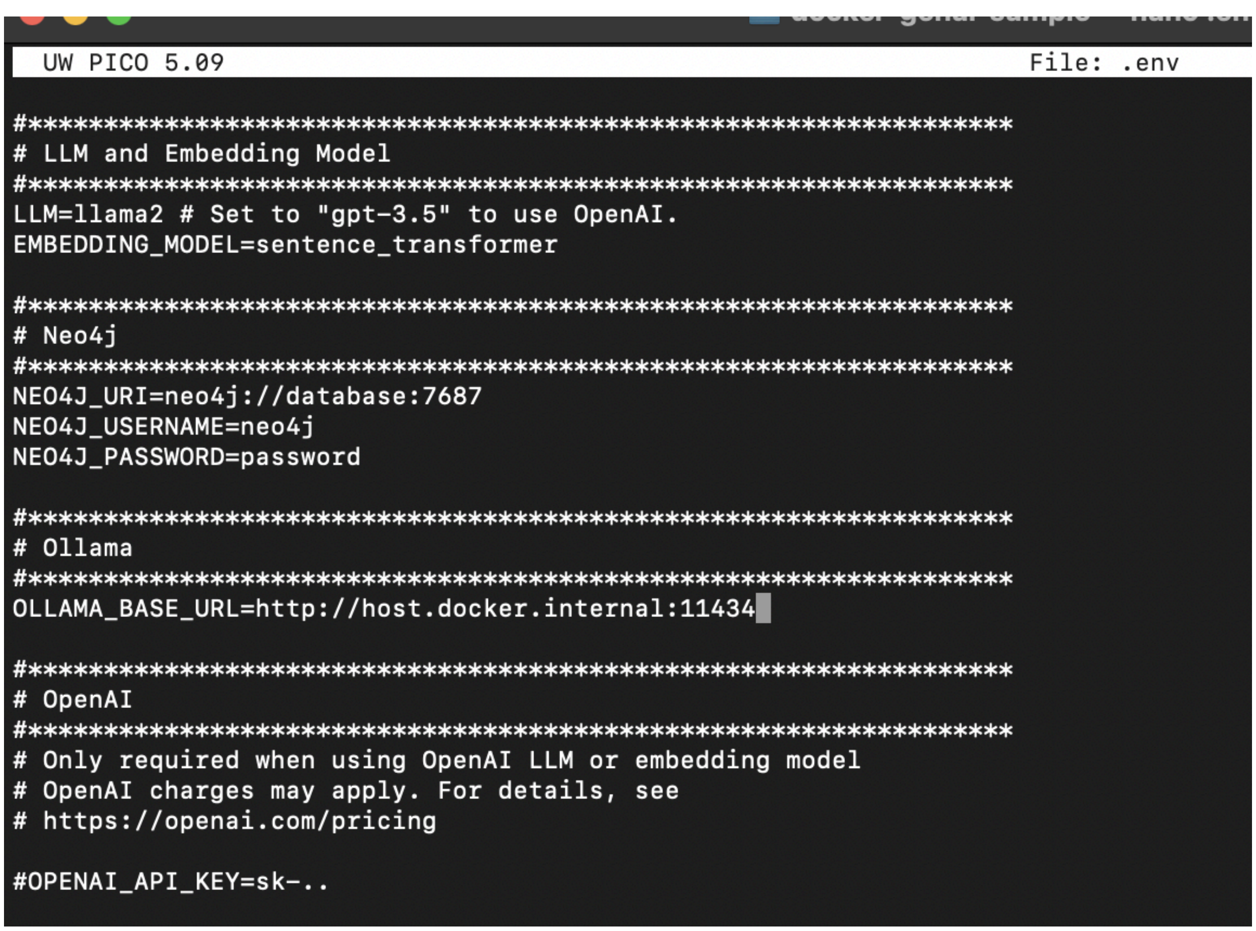
Run Ollama in a Container

1. Install and run Ollama on your host machine.


Welcome to Ollama

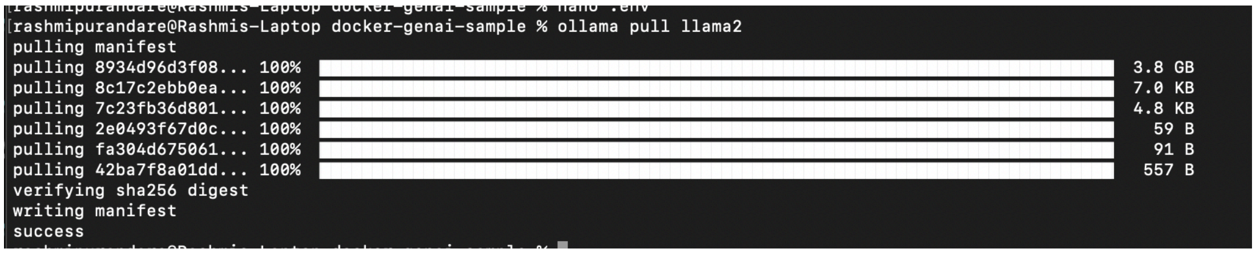
Let's get you up and running with your own large language models.

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



```
  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://host.docker.internal: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-..
```

3. Pull the model to Ollama using the following command.
$ ollama pull llama2

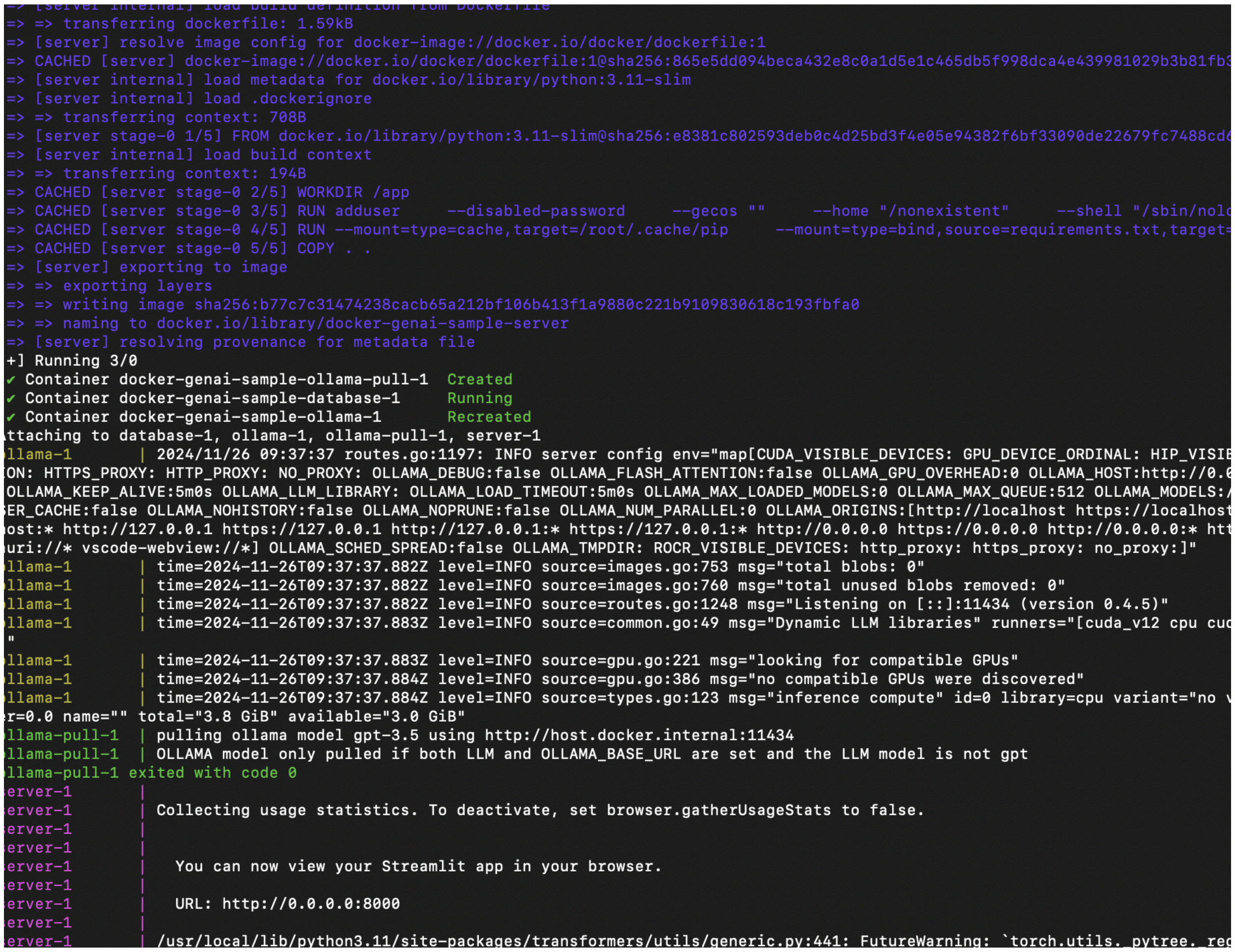


```
rashmipurandare@Rashmis-Laptop docker-genai-sample % nano .env
rashmipurandare@Rashmis-Laptop docker-genai-sample % ollama pull llama2
pulling manifest
pulling 8934d96d3f08... 100%                                      3.8 GB
pulling 8c17c2ebb0ea... 100%                                      7.0 KB
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
success
```

Note: In case you are using OpenAI you can do the following steps instead.
i. ii. Update the LLM value in your .env file to gpt-3.5.
Uncomment and update the OPENAI_API_KEY value in your .env file to your OpenAI API key

Run Your GenAI Application

1. To run all the services, run the following command.
$ docker compose up --build

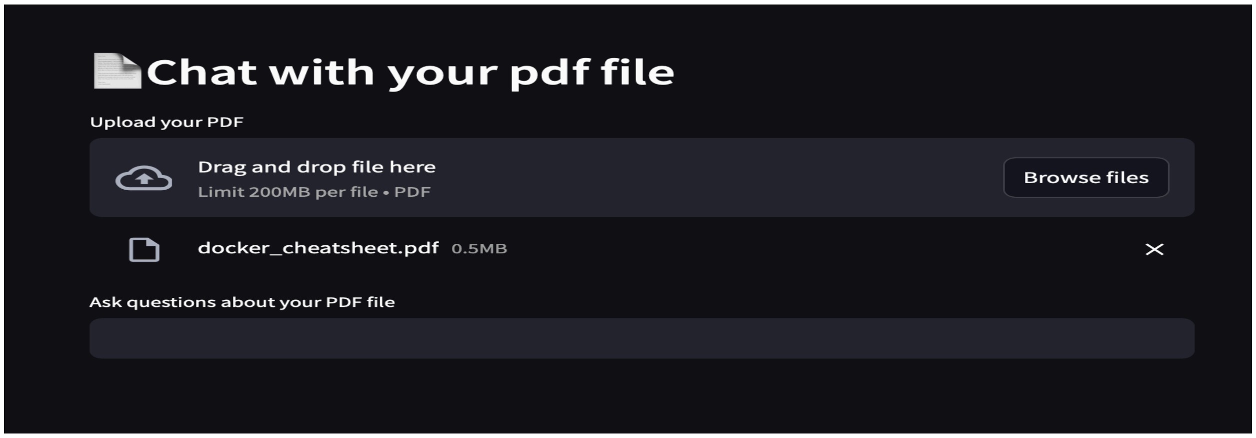


```
=> => transferring dockerfile: 1.59kB
=> [server] resolve image config for docker-image://docker.io/docker/dockerfile:1
=> CACHED [server] docker-image://docker.io/docker/dockerfile:1@sha256:865e5dd094beca432e8c0a1d5e1c465db5f998dca4e439981029b3b81fb3
=> [server internal] load metadata for docker.io/library/python:3.11-slim
=> [server internal] load .dockerignore
=> => transferring context: 708B
=> [server stage-0 1/5] FROM docker.io/library/python:3.11-slim@sha256:e8381c802593deb0c4d25bd3f4e05e94382f6bf33090de22679fc7488cd6
=> [server internal] load build context
=> => transferring context: 194B
=> CACHED [server stage-0 2/5] WORKDIR /app
=> CACHED [server stage-0 3/5] RUN adduser     --disabled-password     --gecos ""     --home "/nonexistent"     --shell "/sbin/nolo
=> CACHED [server stage-0 4/5] RUN --mount=type=cache,target=/root/.cache/pip     --mount=type=bind,source=requirements.txt,target=
=> CACHED [server stage-0 5/5] COPY . .
=> [server] exporting to image
=> => exporting layers
=> => writing image sha256:b77c7c31474238cacb65a212bf106b413f1a9880c221b9109830618c193fbfa0
=> => naming to docker.io/library/docker-genai-sample-server
=> [server] resolving provenance for metadata file
+] Running 3/0
✔ Container docker-genai-sample-ollama-pull-1  Created
✔ Container docker-genai-sample-database-1     Running
✔ Container docker-genai-sample-ollama-1       Recreated
ttaching to database-1, ollama-1, ollama-pull-1, server-1
llama-1        | 2024/11/26 09:37:37 routes.go:1197: INFO server config env="map[CUDA_VISIBLE_DEVICES: GPU_DEVICE_ORDINAL: HIP_VISIB
ON: HTTPS_PROXY: HTTP_PROXY: NO_PROXY: OLLAMA_DEBUG:false OLLAMA_FLASH_ATTENTION:false OLLAMA_GPU_OVERHEAD:0 OLLAMA_HOST:http://0.0
OLLAMA_KEEP_ALIVE:5m0s OLLAMA_LLM_LIBRARY: OLLAMA_LOAD_TIMEOUT:5m0s OLLAMA_MAX_LOADED_MODELS:0 OLLAMA_MAX_QUEUE:512 OLLAMA_MODELS:/
ER_CACHE:false OLLAMA_NOHISTORY:false OLLAMA_NOPRUNE:false OLLAMA_NUM_PARALLEL:0 OLLAMA_ORIGINS:[http://localhost https://localhost
ost:* http://127.0.0.1 https://127.0.0.1 http://127.0.0.1:* https://127.0.0.1:* http://0.0.0.0 https://0.0.0.0 http://0.0.0.0:* htt
uri://* vscode-webview://*] OLLAMA_SCHED_SPREAD:false OLLAMA_TMPDIR: ROCR_VISIBLE_DEVICES: http_proxy: https_proxy: no_proxy:]"
llama-1        | time=2024-11-26T09:37:37.882Z level=INFO source=images.go:753 msg="total blobs: 0"
llama-1        | time=2024-11-26T09:37:37.882Z level=INFO source=images.go:760 msg="total unused blobs removed: 0"
llama-1        | time=2024-11-26T09:37:37.882Z level=INFO source=routes.go:1248 msg="Listening on [::]:11434 (version 0.4.5)"
llama-1        | time=2024-11-26T09:37:37.883Z level=INFO source=common.go:49 msg="Dynamic LLM libraries" runners="[cuda_v12 cpu cuc
"
llama-1        | time=2024-11-26T09:37:37.883Z level=INFO source=gpu.go:221 msg="looking for compatible GPUs"
llama-1        | time=2024-11-26T09:37:37.884Z level=INFO source=gpu.go:386 msg="no compatible GPUs were discovered"
llama-1        | time=2024-11-26T09:37:37.884Z level=INFO source=types.go:123 msg="inference compute" id=0 library=cpu variant="no v
r=0.0 name="" total="3.8 GiB" available="3.0 GiB"
llama-pull-1   | pulling ollama model gpt-3.5 using http://host.docker.internal:11434
llama-pull-1   | OLLAMA model only pulled if both LLM and OLLAMA_BASE_URL are set and the LLM model is not gpt
llama-pull-1 exited with code 0
erver-1        |
erver-1        | Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.
erver-1        |
erver-1        |
erver-1        |   You can now view your Streamlit app in your browser.
erver-1        |
erver-1        |   URL: http://0.0.0.0:8000
erver-1        |
erver-1        | /usr/local/lib/python3.11/site-packages/transformers/utils/generic.py:441: FutureWarning: `torch.utils._pytree._reg
```

- Wait until everything is built and service is started.

2. Once the application is running, open a browser and access the application at http://localhost:8000.

3. Then we can upload a PDF file, for example the Docker CLI Cheat Sheet, and ask a question about the PDF.



Chat with your pdf file

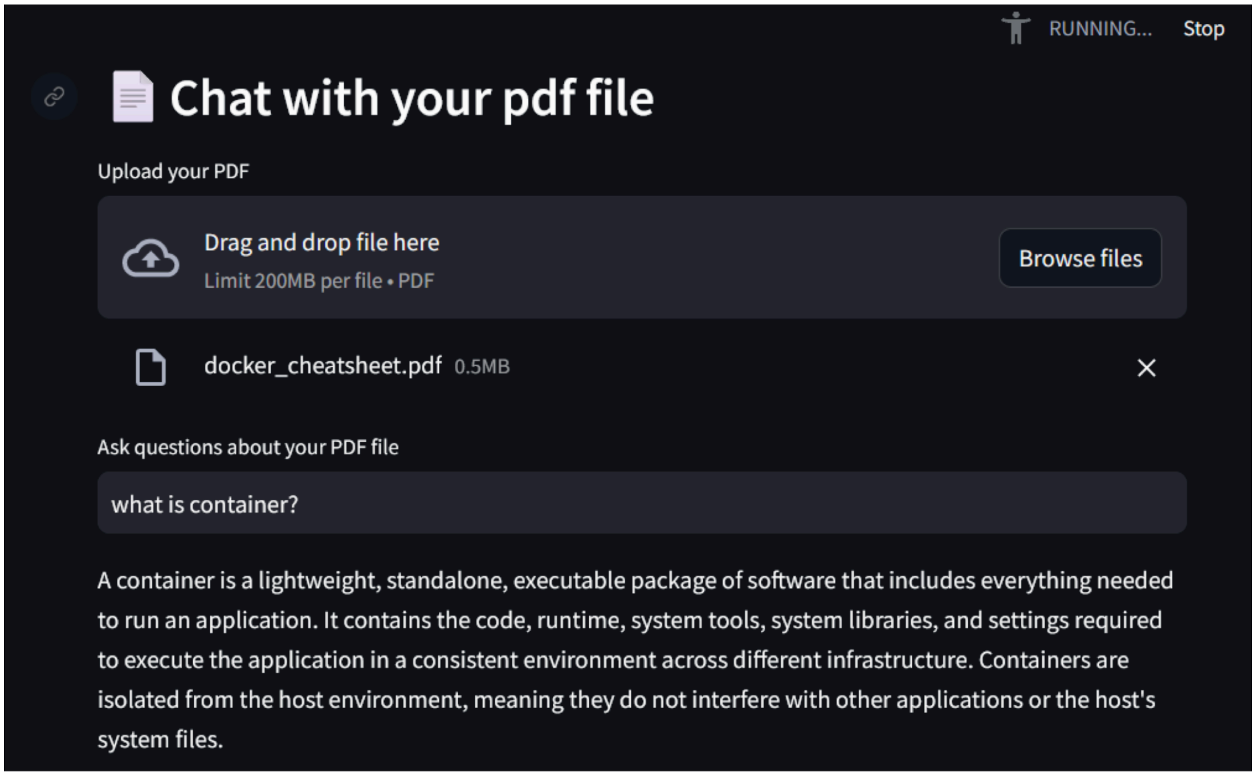
Upload your PDF

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

docker_cheatsheet.pdf 0.5MB

Ask questions about your PDF file

Week 11 Homework 2: Project: GenAI - Develop your containerized app



Chat with your pdf file

Upload your PDF

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.

Through this we have set up a development environment that provides access to all the services that our GenAI application needs.

===

Step 3: Link to GitHub