Prerequisites

1. **Ensure GPU Acceleration**:
   - Verify GPU acceleration is available for Docker Desktop on Windows with WSL2 backend. This is necessary for applications that leverage GPU for enhanced performance. Done before running the app on the bottom of the file.
2. **Install Docker Desktop**:
   - Download and install Docker Desktop from Docker's official site. Ensure you have the latest version to utilize all the latest features and bug fixes.

```
PS C:\Users\aaron> docker --version
Docker version 27.0.3, build 7d4bcd8
```

3. **Install Git**:
   - Download and install Git from Git's official site. Git is required to clone the application repository.

Step-by-Step Guide to Install Git on Windows

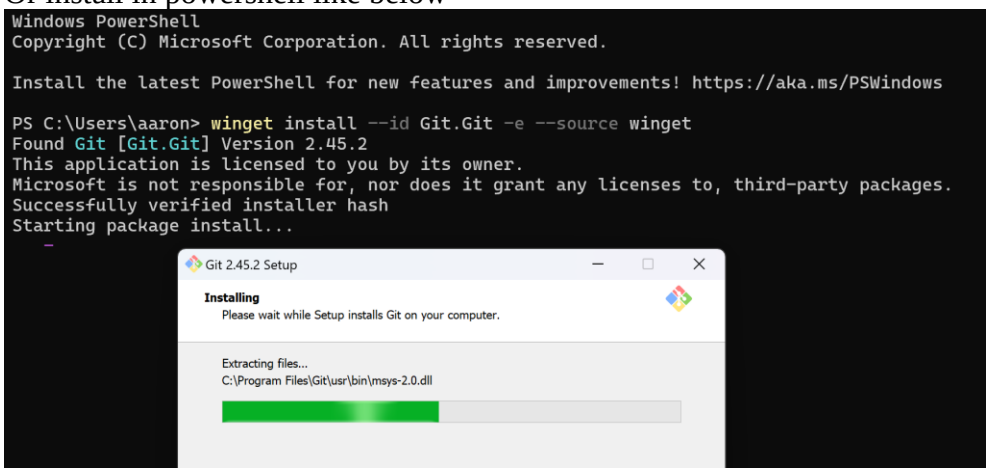
4. **Download Git**:
   - Go to the Git official website.
   - Click on "Windows" to download the Git installer for Windows.
5. **Run the Installer**:
   - Open the downloaded `.exe` file.
   - Follow the installation instructions. You can usually accept the default options unless you have specific preferences.
6. **Verify the Installation**:
   - Open Command Prompt or PowerShell.
   - Run the following command to verify that Git is installed:

     ```
     git -version
     ```

     ```
     PS C:\Users\aaron> git --version
     git version 2.45.2.windows.1
     ```

   - You should see the Git version information, indicating that Git is successfully installed.

- Or install in powershell like below

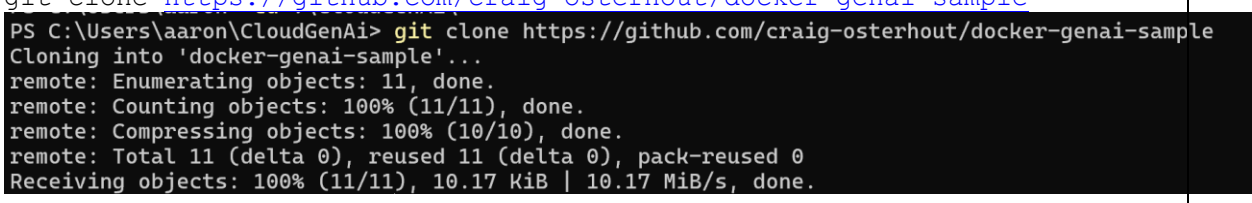


Step-by-Step Instructions

1. Get the Sample Application

1. **Open Command Prompt or PowerShell**:
   - This will be your interface for running commands.
2. **Clone the repository**:
   - Run:

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



   - This command downloads the sample application to your local machine.
3. **Navigate to the cloned directory**:
   - Run:

```
cd docker-genai-sample
```

   - Change your working directory to the location of the cloned repository.

2. Initialize Docker Assets

1. **Run the initialization command**:
   - Execute:

```
docker init
```

- This initializes Docker configuration files for your application.

```
PS C:\Users\aaron\CloudGenAi\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!

? What application platform does your project use?  [Use arrows to move, type to filter]
> Python - (detected) suitable for a Python server application
  Go - suitable for a Go server application
  Node - suitable for a Node server application
  Rust - suitable for a Rust server application
  ASP.NET Core - suitable for an ASP.NET Core application
  PHP with Apache - suitable for a PHP web application
  Java - suitable for a Java application that uses Maven and packages as an uber jar
  Other - general purpose starting point for containerizing your application
  Don't see something you need? Let us know!
  Quit
```

2. **Provide the following inputs when prompted**:
   - **Application platform**: Python
   - **Python version**: 3.11.4
   - **Application port**: 8000
   - **Command to run the app**: `streamlit run app.py --server.address=0.0.0.0 --server.port=8000`
   - These inputs help Docker create appropriate configuration files (`.dockerignore`, `Dockerfile`, `compose.yaml`, and `README.Docker.md`).

```
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!

? What application platform does your project use? Python
? What version of Python do you want to use? (3.11.9) 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) 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 '? 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
```

3. Run the Application

1. **Ensure Docker Desktop is running**:

- Make sure Docker Desktop is active on your machine.

2. **Execute the command to build and run the application**:
   - Run:

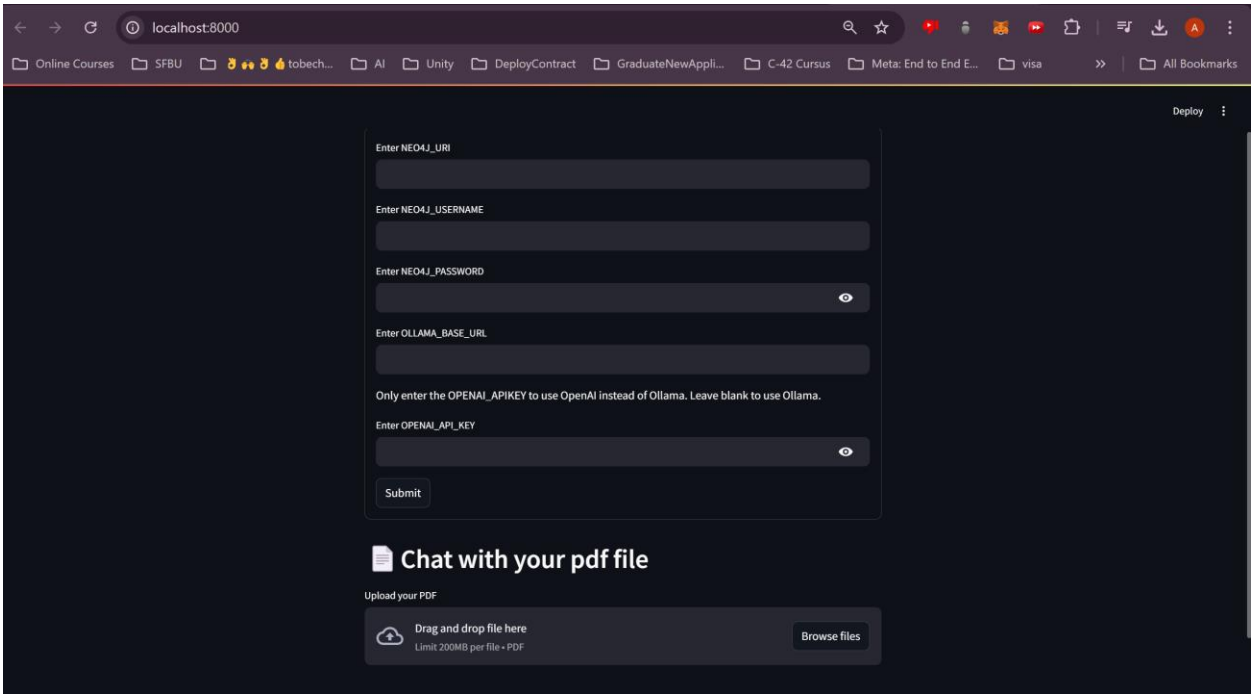
```
docker compose up -build
```

```
PS C:\Users\aaron\CloudGenAi\docker-genai-sample> docker compose up --build
[+] Building 552.1s (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-image://docker.io/docker/dockerfile:1                                         1.9s
 => [server] docker-image://docker.io/docker/dockerfile:1@sha256:fe40cf4e92cd0c467be2cfc30657a680ae2398318afd50b0c80585784c6  1.9s
 => => resolve docker.io/docker/dockerfile:1@sha256:fe40cf4e92cd0c467be2cfc30657a680ae2398318afd50b0c80585784c604f28          0.0s
 => => sha256:fe40cf4e92cd0c467be2cfc30657a680ae2398318afd50b0c80585784c604f28 8.40kB / 8.40kB                             0.0s
 => => sha256:dc9e236567481e0aca4c1f52351af213b9a176622f10e3f4a86e5cc48919fa01 482B / 482B                                 0.0s
 => => sha256:fd020648a727ee1aa6fe2924bf9c498d19385fa2491ddeecb9da9a499c43e35a 1.26kB / 1.26kB                             0.0s
 => => sha256:2ba8a93af1b3f8d1c5354117c15aa2eaa674a24a81b6622506a8a524ba8d3fc9 12.46MB / 12.46MB                           1.7s
 => => extracting sha256:2ba8a93af1b3f8d1c5354117c15aa2eaa674a24a81b6622506a8a524ba8d3fc9                                  0.1s
 => [server internal] load metadata for docker.io/library/python:3.11.9-slim                                              2.7s
 => [server internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 671B                                                                                          0.0s
 => [server base 1/5] FROM docker.io/library/python:3.11.9-slim@sha256:3f3c35617e79276c5f6a2e6a13cdbabdd10257332df963c90c986  7.8s
 => => resolve docker.io/library/python:3.11.9-slim@sha256:3f3c35617e79276c5f6a2e6a13cdbabdd10257332df963c90c986858b26fad5e   0.0s
 => => sha256:13d803497f988e563809a4c2664c4b9d23883a3a7ba3fbdad6d729dcaaf43e56 6.89kB / 6.89kB                             0.0s
 => => sha256:efc2b5ad9eec05befa54239d53feeae3569ccbef689aa5e5dbfc25da6c4df559 29.13MB / 29.13MB                           5.5s
 => => sha256:60462faabbc27679d4e1a907afda153c5f2294df5f752f0e706937779faa6d22 3.51MB / 3.51MB                             1.7s
 => => sha256:11f0c4afa075fefa38e75f0bc033f3c60251827dfbecce64bf57bc67fc3718f0 12.87MB / 12.87MB                           5.4s
 => => sha256:3f3c35617e79276c5f6a2e6a13cdbabdd10257332df963c90c986858b26fad5e 9.12kB / 9.12kB                             0.0s
 => => sha256:2bbbd2863f3a2cf5a60794c8cad4530d6be3c9e72f23730239146e61a345009f 1.94kB / 1.94kB                             0.0s
 => => sha256:d8393bf961f1ad054e82d4740c6557c77315c37cb25c19036e329d6194617c13 230B / 230B                                 1.8s
 => => sha256:e1558965ee47a72ec3c70592a93ae13e931a2d0f5719ab6c643c4a531c103236 3.21MB / 3.21MB                             3.2s
 => => extracting sha256:efc2b5ad9eec05befa54239d53feeae3569ccbef689aa5e5dbfc25da6c4df559                                  1.1s
 => => extracting sha256:60462faabbc27679d4e1a907afda153c5f2294df5f752f0e706937779faa6d22                                  0.1s
 => => extracting sha256:11f0c4afa075fefa38e75f0bc033f3c60251827dfbecce64bf57bc67fc3718f0                                  0.4s
 => => extracting sha256:d8393bf961f1ad054e82d4740c6557c77315c37cb25c19036e329d6194617c13                                  0.0s
 => => extracting sha256:e1558965ee47a72ec3c70592a93ae13e931a2d0f5719ab6c643c4a531c103236                                  0.2s
 => [server internal] load build context                                                                                   0.0s
 => => transferring context: 17.09kB                                                                                       0.0s
 => [server base 2/5] WORKDIR /app                                                                                         0.1s
 => [server base 3/5] RUN adduser     --disabled-password     --gecos ""     --home "/nonexistent"     --shell "/sbin/nologi  0.4s
 => [server base 4/5] RUN --mount=type=cache,target=/root/.cache/pip     --mount=type=bind,source=requirements.txt,target=  522.6s
 => [server base 5/5] COPY . .                                                                                             0.1s
 => [server] exporting to image                                                                                           13.9s
 => => exporting layers                                                                                                   13.8s
 => => writing image sha256:209783c421bf79b531dafbd06b2d52ba8bcb7123ffe4d4a4d25968c66bc9ea70                               0.0s
 => => naming to docker.io/library/docker-genai-sample-server                                                              0.0s
time="2024-07-31T02:33:57-07:00" level=warning msg="current commit information was not captured by the build" error="failed to read
 current commit information with git rev-parse --is-inside-work-tree"
[+] Running 2/2
 ✔Network docker-genai-sample_default     Created                                                                          0.1s
 ✔Container docker-genai-sample-server-1  Created                                                                          0.2s
Attaching to server-1
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.
server-1  |
server-1  |   URL: http://0.0.0.0:8000
server-1  |


v View in Docker Desktop   o View Config   w Enable Watch
```

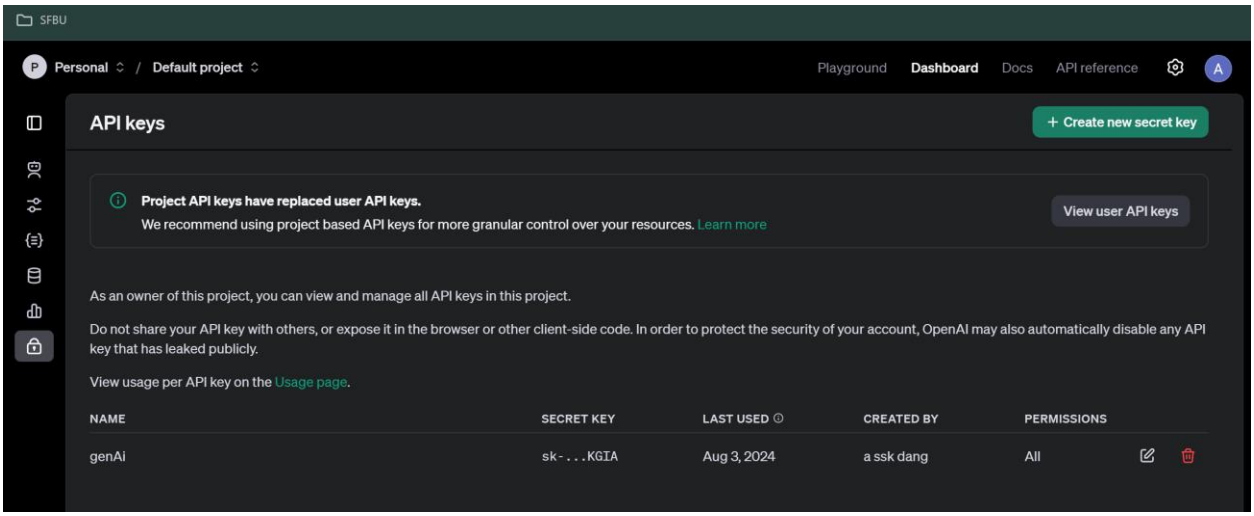
   - This command builds the Docker image and starts the application.

3. **Access the application**:
   - Open http://localhost:8000 in your web browser to view the running application.



4. Set Up Neo4j Database and LLM Service

1. To make this work we need to use gpt3.5 and we should get API from the open-ai website. https://platform.openai.com/api-keys



2. **Rename `env.example` to `.env`:**
   - Run:

```
mv env.example .env
```

File Edit Selection View Go ···

compose.yaml .env

C: > Users > aaron > CloudGenAi > docker-genai-sample >

```
# 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 (if needed)
OPENAI_API_KEY=sk-proj-zasuR9ZsnHrOvoC2
```

platform.openai.com/api-keys

SFBU

Personal / Default project

Playground Dashboard Docs API reference

DASHBOARD

Assistants
Fine-tuning
Batches
Storage
Usage
API keys

API keys

+ Create new secret key

Project API keys have replaced user API keys.
We recommend using project based API keys for more granular control over your resources. Learn more

View user API keys

As an owner of this project, you can view and manage all API keys in this project.

Do not share your API key with others, or expose it in the browser or other client-side code. In order to protect the security of your account, OpenAI may also automatically disable any API key that has leaked publicly.

View usage per API key on the Usage page.

NAME	SECRET KEY	LAST USED	CREATED BY	PERMISSIONS
genAi	sk-...KGIA	Never	a ssk dang	All

- Include the temporary neo4j password and use the API-key we generated from the open-AI.
- Update the OLLAMA_BASE_URL value in your .env file

```
            to http://host.docker.internal:11434.
canceled
PS C:\Users\aaron\CloudGenAi\docker-genai-sample> mv env.example .env
```

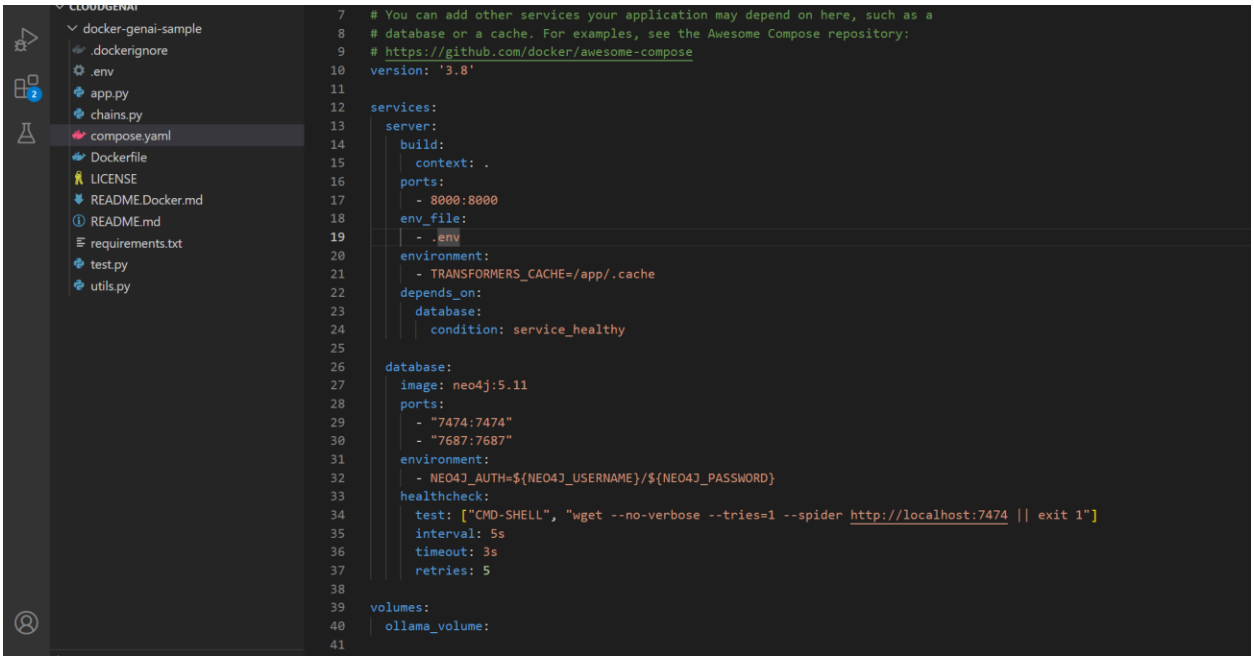
- This creates an environment configuration file from the example provided.

3. **Update `compose.yaml` to include a Neo4j database service**:
   - Add the following configuration:

```
version: '3.8'
services:
  server:
    build:
      context: .
    ports:
      - 8000:8000
    env_file:
      - .env
    environment:
      - TRANSFORMERS_CACHE=/app/.cache
    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
http://localhost:7474 || exit 1"]
      interval: 5s
      timeout: 3s
      retries: 5

volumes:
  ollama_volume:  ollama_volume:
```

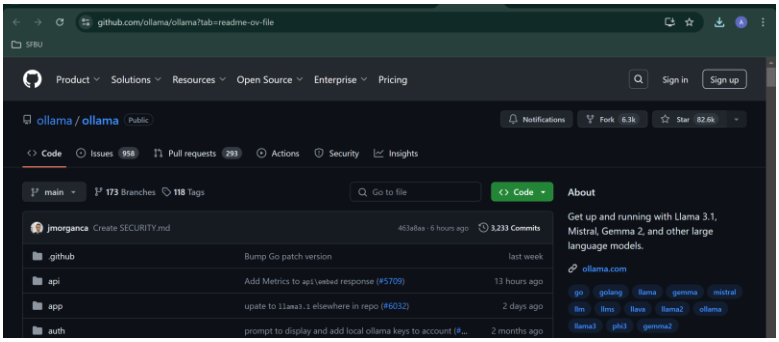


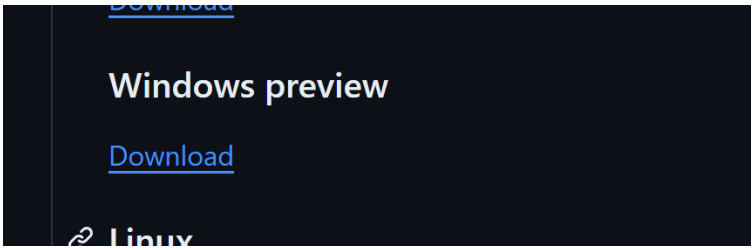
- This configuration sets up a Neo4j service necessary for the application.

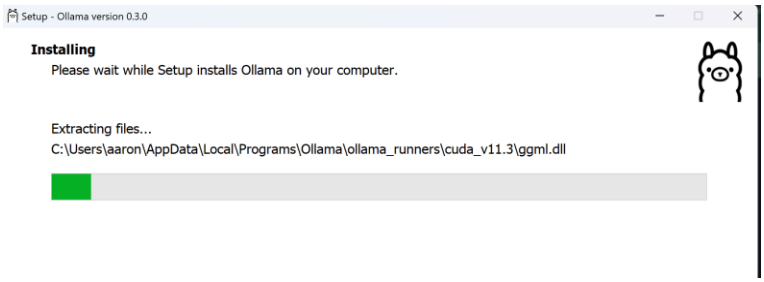
5. Add a Local or Remote LLM Service

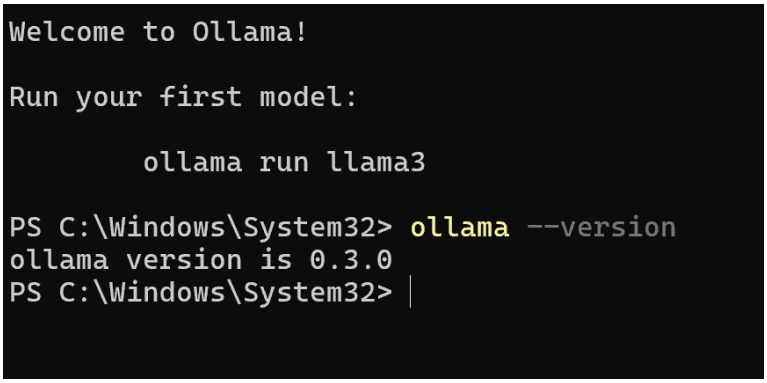
Step 1: Download and Install Ollama

1. **Download Ollama:**
   - Visit the Ollama GitHub repository.
   - Follow the installation instructions provided in the repository. Typically, this involves downloading an installer or cloning the repository and running an install script.









Step 2: Add Ollama to System PATH (if not done automatically)

2. **Locate Ollama Installation Directory:**
   - Find the directory where Ollama was installed. This might be a directory like `C:\Program Files\Ollama` or similar.
3. **Add Directory to PATH:**
   - Open PowerShell as Administrator and run the following commands to add Ollama to your PATH:

     ```
     [Environment]::SetEnvironmentVariable("Path", $env:Path +
     ";C:\Path\To\Ollama", [EnvironmentVariableTarget]::Machine)
     ```

   - Replace `C:\Path\To\Ollama` with the actual path to the Ollama executable.

4. **Pull the model to Ollama**: (make sure to restart terminal after installing Ollama)
   - Run:

     ```
     ollama pull llama2
     ```

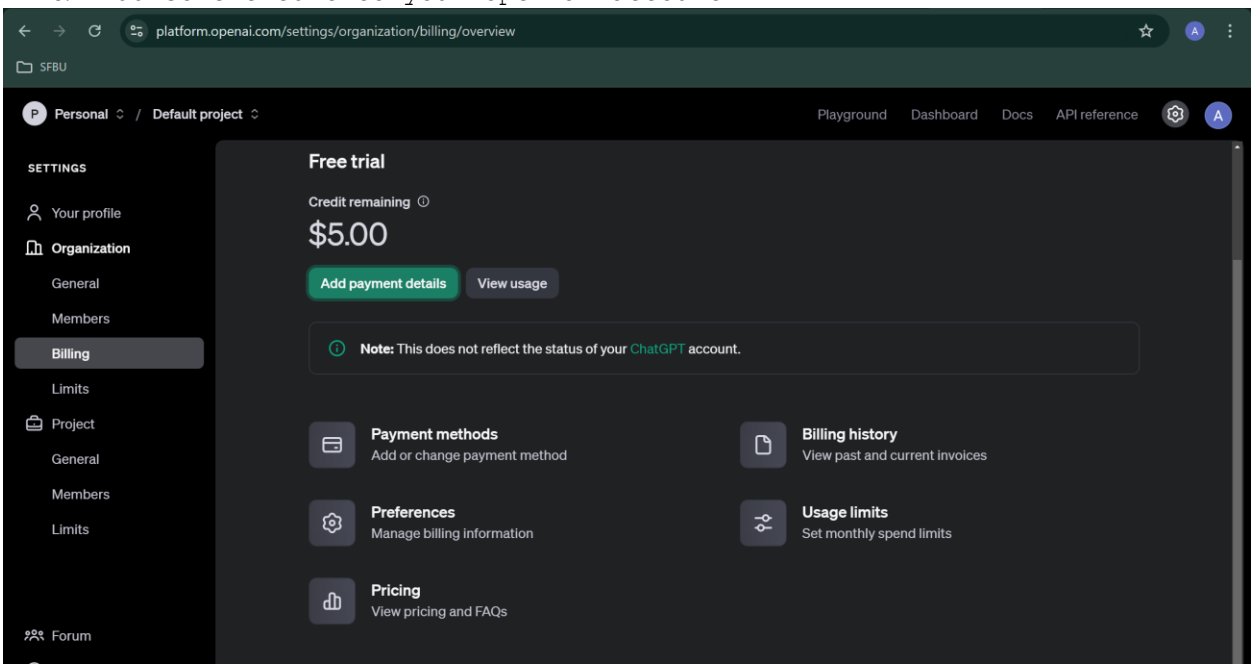
```
PS C:\Users\aaron> cd .\CloudGenAi\
PS C:\Users\aaron\CloudGenAi> cd .\docker-genai-sample\
PS C:\Users\aaron\CloudGenAi\docker-genai-sample> ollama pull llama2
pulling manifest
pulling 8934d96d3f08... 100%  3.8 GB
pulling 8c17c2ebb0ea... 100%  7.0 KB
pulling 7c23fb36d801... 100%  4.8 KB
pulling 2e0493f67d0c... 100%   59 B
pulling fa304d675061... 100%   91 B
pulling 42ba7f8a01dd... 100%  557 B
verifying sha256 digest
writing manifest
removing any unused layers
success
PS C:\Users\aaron\CloudGenAi\docker-genai-sample>
```

- This command pulls the required model for the LLM service.

5. Install WSL

```
PS C:\Users\aaron\CloudGenAi\docker-genai-sample> wsl --install
Installing: Ubuntu
[                              0.0%                              ]
```

6. Add some credit to your open-ai account



7. Run the GenAI Application

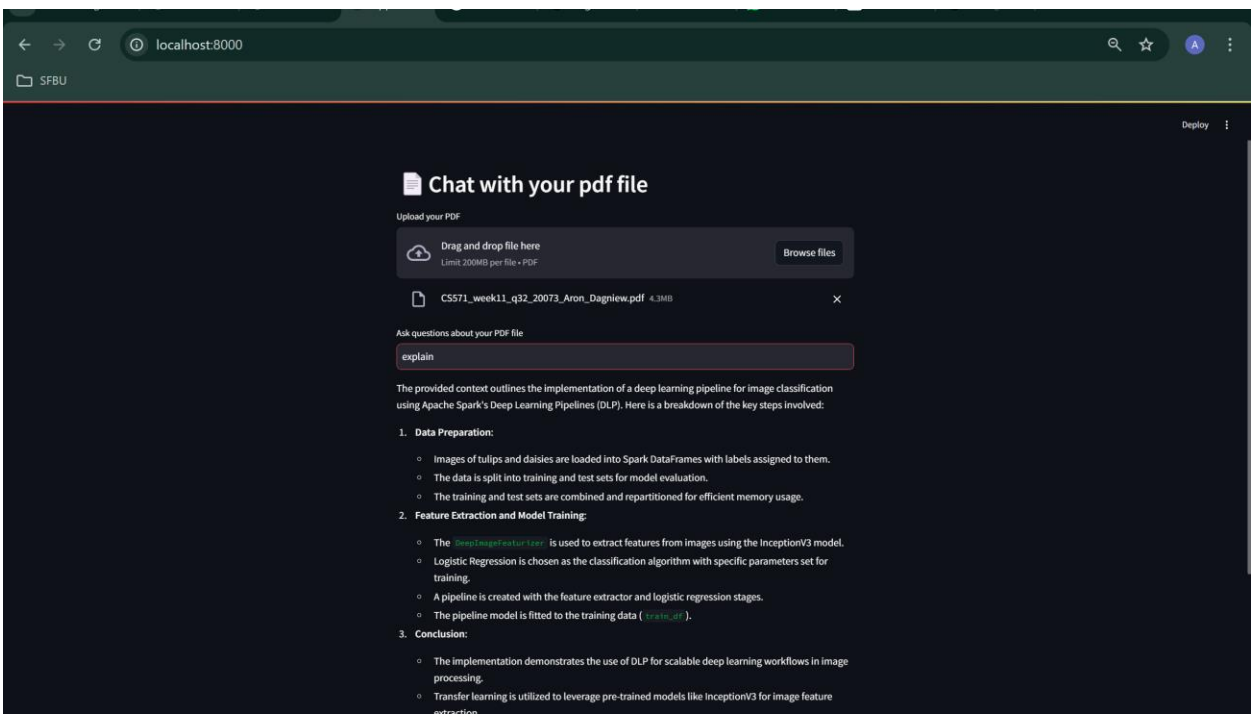
1. **Execute the command to build and run the application**:
   - From the root directory of your project (`docker-genai-sample`), run:

```
docker compose up -build
```

```
 => [server] exporting to image                                                                    0.1s
 => => exporting layers                                                                            0.0s
 => => writing image sha256:ab33be77cd822fd489aa51e96ac7c94dda24860ab239f72faf2b2538d4f76f43       0.0s
 => => naming to docker.io/library/docker-genai-sample-server                                      0.0s
time="2024-08-03T07:40:09-07:00" level=warning msg="current commit information was not captured by the build" error="failed to read
 current commit information with git rev-parse --is-inside-work-tree"
time="2024-08-03T07:40:11-07:00" level=warning msg="Found orphan containers ([docker-genai-sample-ollama-1 docker-genai-sample-olla
ma-pull-1]) for this project. If you removed or renamed this service in your compose file, you can run this command with the --remo
ve-orphans flag to clean it up."
[+] Running 3/2
 ✔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 s
tarted for the first time.
database-1  | 2024-08-03 14:40:16.953+0000 INFO  Starting...
database-1  | 2024-08-03 14:40:17.707+0000 INFO  This instance is ServerId{5ecec3bc} (5ecec3bc-c88a-4cbe-bb86-bbff1980a41b)
database-1  | 2024-08-03 14:40:18.305+0000 INFO  ======== Neo4j 5.11.0 ========
database-1  | 2024-08-03 14:40:20.187+0000 INFO  Bolt enabled on 0.0.0.0:7687.
database-1  | 2024-08-03 14:40:20.764+0000 INFO  Remote interface available at http://localhost:7474/
database-1  | 2024-08-03 14:40:20.768+0000 INFO  id: A3A58A2D65F4D779721D0BFD7EC9075FBD3B189FD7D29F867172EDCA5AD6E649
database-1  | 2024-08-03 14:40:20.768+0000 INFO  name: system
database-1  | 2024-08-03 14:40:20.769+0000 INFO  creationDate: 2024-08-03T14:40:18.837Z
database-1  | 2024-08-03 14:40:20.769+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.
server-1    |
server-1    |   URL: http://0.0.0.0:8000
```

- This command builds the Docker image and starts the application along with all required services.

2. **Access the application**:
   - Open http://localhost:8000 in your web browser once all services are up and running.



As you can see it above it is successfully working.