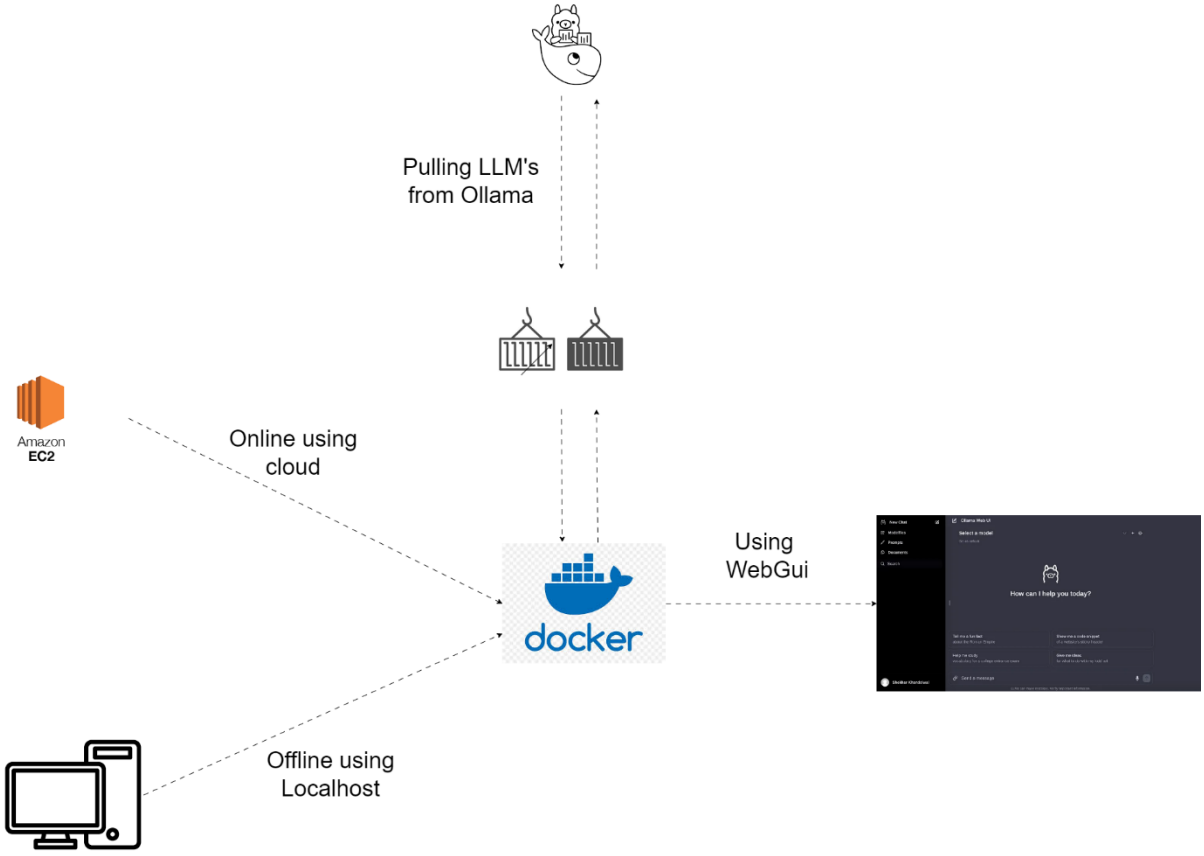
Introduction to Deploying Ollama with Docker and WebGUI



When it comes to deploying machine learning models, the process can often feel like a juggling act between setting up infrastructure, managing dependencies, and ensuring that everything runs smoothly across different environments.

This is where **Ollama** comes in, a platform designed to simplify the deployment and management of AI models, making it easier for developers to focus on building and refining their models rather than worrying about the underlying infrastructure.

Ollama allows you to load and manage pre-trained models efficiently, providing the flexibility to deploy them across a variety of environments, whether it's on a local machine, a cloud platform, or even hybrid solutions. It brings a streamlined approach to AI model deployment, significantly reducing the complexity for both developers and researchers.

One of the key features that makes Ollama accessible and user-friendly is its **WebGUI** (Web Graphical User Interface). A WebGUI offers an intuitive, browser-based interface to interact with the deployed models. Instead of navigating the command line or dealing with complex scripts, users can manage models, track performance metrics, and scale deployments right from their web browser.

This ease of access makes WebGUI an excellent tool, especially for those who might not be deeply technical but still need to manage and interact with machine learning models.

In this project, we'll walk through how to deploy Ollama using Docker and a WebGUI, making it possible to containerize your model deployments and manage them with ease. Docker, a containerization platform, allows us to bundle Ollama and its dependencies into isolated environments. Meanwhile, Kubernetes (K8s) offers a powerful way to scale and manage these containerized applications across distributed systems, perfect for handling the demands of production environments.

Let's get started on how you can leverage Docker to deploy Ollama with a WebGUI interface and scale it with Kubernetes for robust, production-grade deployments.

Step-1

Update and Upgrade the packages and install docker using

apt install docker.io

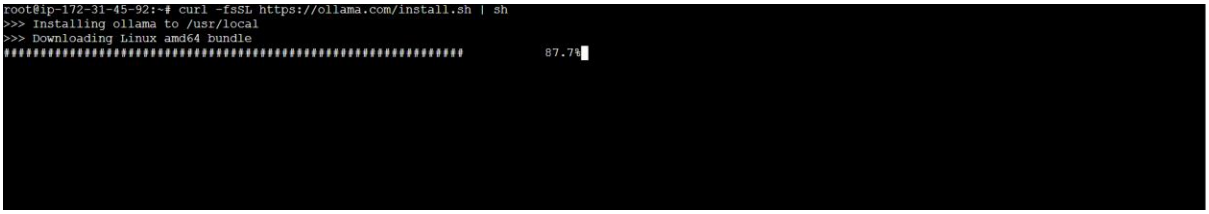
and Start the docker service

Systemctl enable docker --now

Step-2

Copy this command to download the Ollama on local machine

curl -fsSL https://ollama.com/install.sh | sh



Step-3

Ollama by-default is CLI and does not support GUI to use GUI on ollama we need to install additional API call WebGUI

Installaing WebGUI copy paste the following command

docker run -d -p 3000:8080 --add-host=host.docker.internal:host-gateway -v open-webui:/app/backend/data --name open-webui --restart always ghcr.io/open-webui/open-webui:main

```
root@ip-172-31-45-92:~# docker run -d -p 3000:8080 --add-host=host.docker.internal:host-gateway -v open-webui:/app/backend/data --name open-webui --restart always
 ghcr.io/open-webui/open-webui:main
Unable to find image 'ghcr.io/open-webui/open-webui:main' locally
main: Pulling from open-webui/open-webui
a2318d6c47ec: Pull complete
40d734479f14: Pull complete
0b61b7b259eb: Pull complete
081a3493c0e7: Pull complete
e8027c27b65b: Pull complete
4f4fb700ef54: Pull complete
a211859e0845: Pull complete
3b28612a0571: Pull complete
24211b5d3443: Pull complete
5d40084defd1: Pull complete
28cf92cf999e: Pull complete
87697932c3dc: Pull complete
e693573ffe23: Pull complete
ca52356c6c96: Pull complete
69b9baca91b3: Pull complete
Digest: sha256:7090199fcd76bbc7add0c53d64072a47b25df82373bb94c9435d8d53651f144a
Status: Downloaded newer image for ghcr.io/open-webui/open-webui:main
a5bccee8e6bf8c11215866c2cb2f88b4fd8864661adb2f670abeec43323f1362
root@ip-172-31-45-92:~#
```

Step-4

We can use **docker ps –a** command to check if webGui is Up and Running

```
root@ip-172-31-45-92:~# docker ps -a
CONTAINER ID   IMAGE                                COMMAND           CREATED          STATUS                    PORTS                                       NAMES
a5bccee8e6bf   ghcr.io/open-webui/open-webui:main   "bash start.sh"   56 seconds ago   Up 51 seconds (healthy)   0.0.0.0:3000->8080/tcp, :::3000->8080/tcp   open-
webui
root@ip-172-31-45-92:~#
```

Step-6

Go to instance Security Group and add port 3000 and 8080 for TCP protocol because the WebGui container uses this port.

Inbound rules Info

Security group rule ID	Type Info	Protocol Info	Port range Info	Source Info		Description - optional Info	
sgr-09769f87d5dc6544c	HTTP	TCP	80	Custom	0.0.0.0/0		Delete
sgr-07f62f354f1015a9f	SSH	TCP	22	Custom	0.0.0.0/0		Delete
sgr-0143da39d17ef1129	Custom TCP	TCP	8080	Custom	0.0.0.0/0		Delete
sgr-0fc6fb66dd64f88a2	Custom TCP	TCP	3000	Custom	0.0.0.0/0		Delete

Add rule

Step 7-

Pulling the LLM model using ollama

Ollama provides various models such as

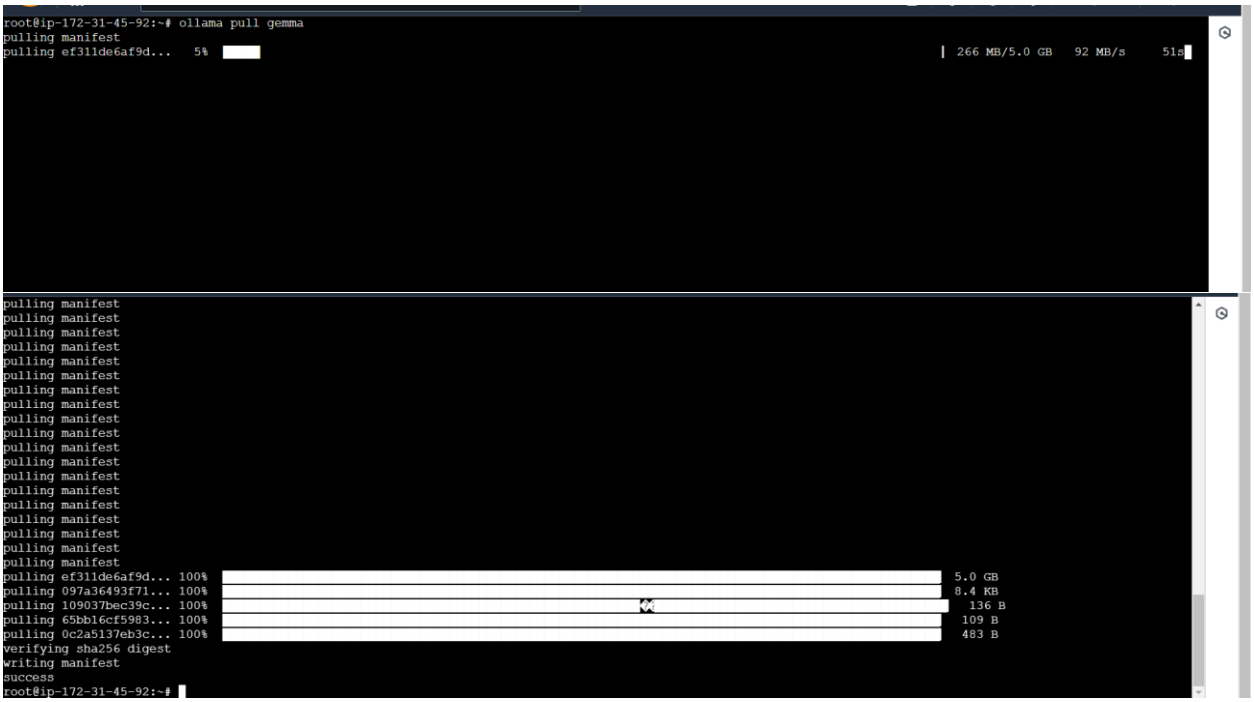
1) Mistral

2)Gemma 2

3)Llama 3

To pull model we need to use command

ollama pull Model_name(gemma,mistral)



Step 8-

Once the model is pulled we can directly start using the model

Ollama run model_name (which we installed earlier)

The model is running and we can ask any questions that we want like we ask to various AI models such as ChatGpt , Gemini , Llama

This command will run the model in CLI mode where we can ask questions through CLI mode.

NOTE- You have to use your Model_Name here to run Model

Step 9-

We can access the WebGui using

http://localhost:3000 or http://localhost:8080 from your web Browser

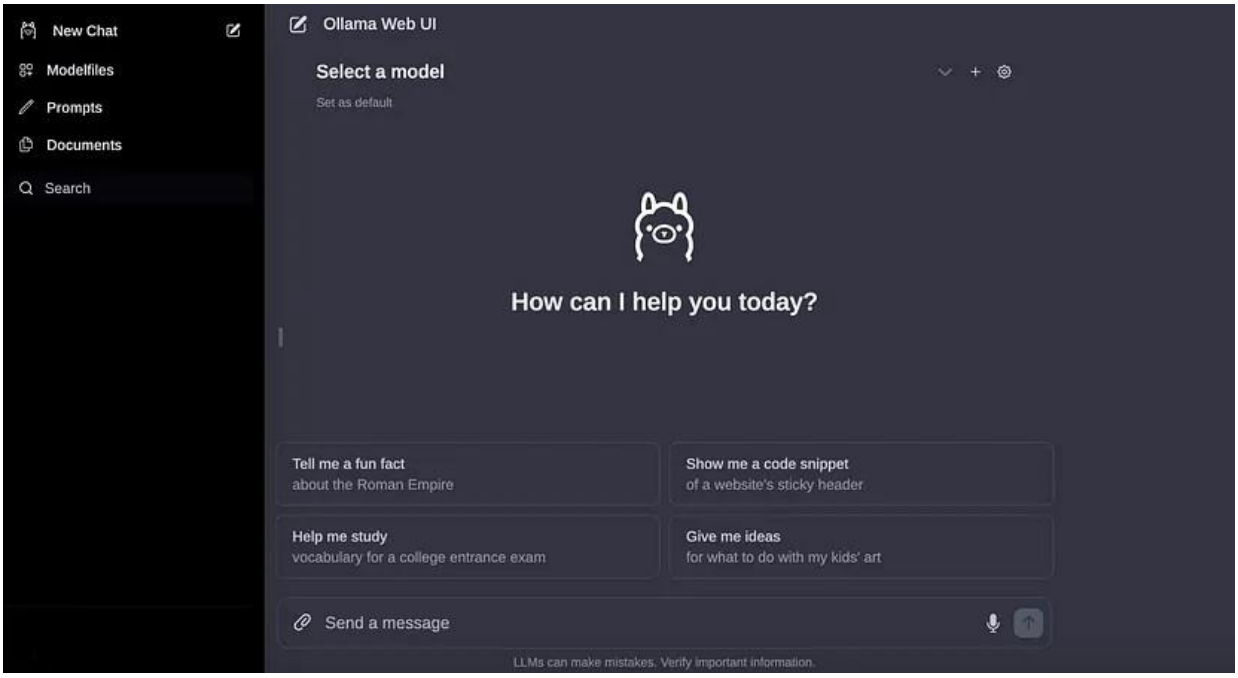
Once Accessed it ask user to create account we will do that and will login into WebGui

A) Go seetting's on left side

B) Click on Admin Panel/Account

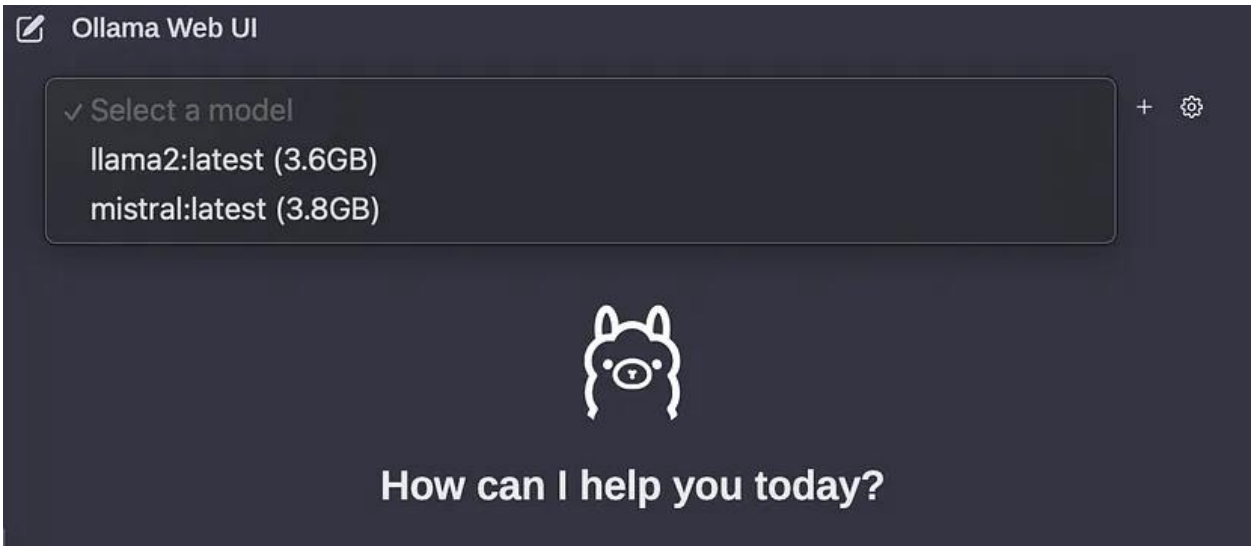
C) Select Model / Download Model

D) Select the downloaded Model and Done we can ask all sort's of question using GUI



Step 10 –

We can also pull and select multiple Model to Work with.



Thank You

Deploying machine learning models doesn't have to be a complicated process. By leveraging Ollama alongside Docker and a WebGUI, you can simplify model management and deployment while providing an accessible interface for non-technical users. The combination of containerization and intuitive interfaces can drastically reduce the time and effort required to manage AI models, making it a practical solution for developers and researchers alike.

Thank you for taking the time to read this Project Documentation. I hope it provided valuable insights into deploying Ollama with Docker and WebGUI. If you have any questions or feedback, feel free to share them in the comments section. Happy deploying, and best of luck in your machine learning journey!