LLM Containerization & API Deployment Report

Title: Deployment of a Local LLM Chatbot via Docker and FastAPI
Author: Anbhi Thakur
Date: 16 June 2025
Groq API Key Used: `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`

Objective

The objective of this task is to create and deploy a Local Language Model (LLM)-based chatbot using Groq API. The model is to be containerized using Docker and exposed as an API endpoint. This enables other applications or users to send prompts and receive responses programmatically via HTTP requests.

System Overview

The system consists of a chatbot backend built using FastAPI. This backend accepts prompts from users through a REST API and forwards them to the Groq LLM using the provided API key. The chatbot runs inside a Docker container and is exposed locally. Optionally, it can also be exposed over the internet using tools like Ngrok.

Key Components

1. **FastAPI Backend**
   Used to create the `/chat` endpoint for sending and receiving messages.
2. **Groq LLM API**
   Provides powerful responses to user prompts using large-scale language models hosted by Groq.
3. **Docker**
   Used for containerizing the FastAPI application for easy and consistent deployment.
4. **Ngrok (Optional)**
   Provides a secure URL to access the chatbot API publicly from the internet.

Technology Stack

- Python 3.10+
- FastAPI framework
- Requests library for API calls
- Docker for containerization

- Groq’s hosted LLM API
- Ngrok for public exposure (optional)

Implementation Summary

- A FastAPI application was developed to handle chat prompts via POST requests.
- The user’s prompt is forwarded to Groq’s API, and the response is returned via the endpoint.
- The application is containerized using Docker.
- The service is tested locally and optionally exposed via Ngrok for public access.

Deployment Steps

1. Developed the chatbot application using FastAPI.
2. Integrated Groq's LLM using the provided API key.
3. Created a Dockerfile and built a Docker image.
4. Deployed the chatbot in a Docker container locally.
5. Started the server and verified its accessibility at `localhost:8000/chat`.
6. Used Ngrok (optional) to expose the API for remote access.
7. Verified functionality by sending prompts and receiving AI-generated replies.

Output

After successful deployment and testing, the following outputs were observed:

- The FastAPI server ran successfully and was accessible at `http://localhost:8000/chat`.
- When prompted with user input such as:
  `"What is artificial intelligence?"`
  The API returned a response like:
  ```
  "Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. These processes include learning, reasoning, and self-correction."
  ```
- Public API URL generated via Ngrok (example):
  `https://abc1234.ngrok.io/chat`
- Sample interaction:

  Input Prompt:
  `"Tell me a programming joke"`

  Response Returned by API:
  `"Why do programmers prefer dark mode? Because light attracts bugs."`

This confirmed that the chatbot was successfully receiving prompts and returning contextually correct and natural language responses via Groq's LLM.

Conclusion

This task demonstrates how to:

- Build an API-based chatbot using FastAPI
- Integrate a hosted LLM using Groq’s API
- Containerize the application with Docker for portability
- Expose and test the chatbot both locally and publicly

The deployed solution is robust, lightweight, and serves as a foundation for scaling or enhancing LLM-based applications.

Project Folder Structure

```
llm-chatbot/
│
├── Dockerfile
├── main.py
├── requirements.txt
└── README.md
```