HOW TO BUILD A CHATBOT

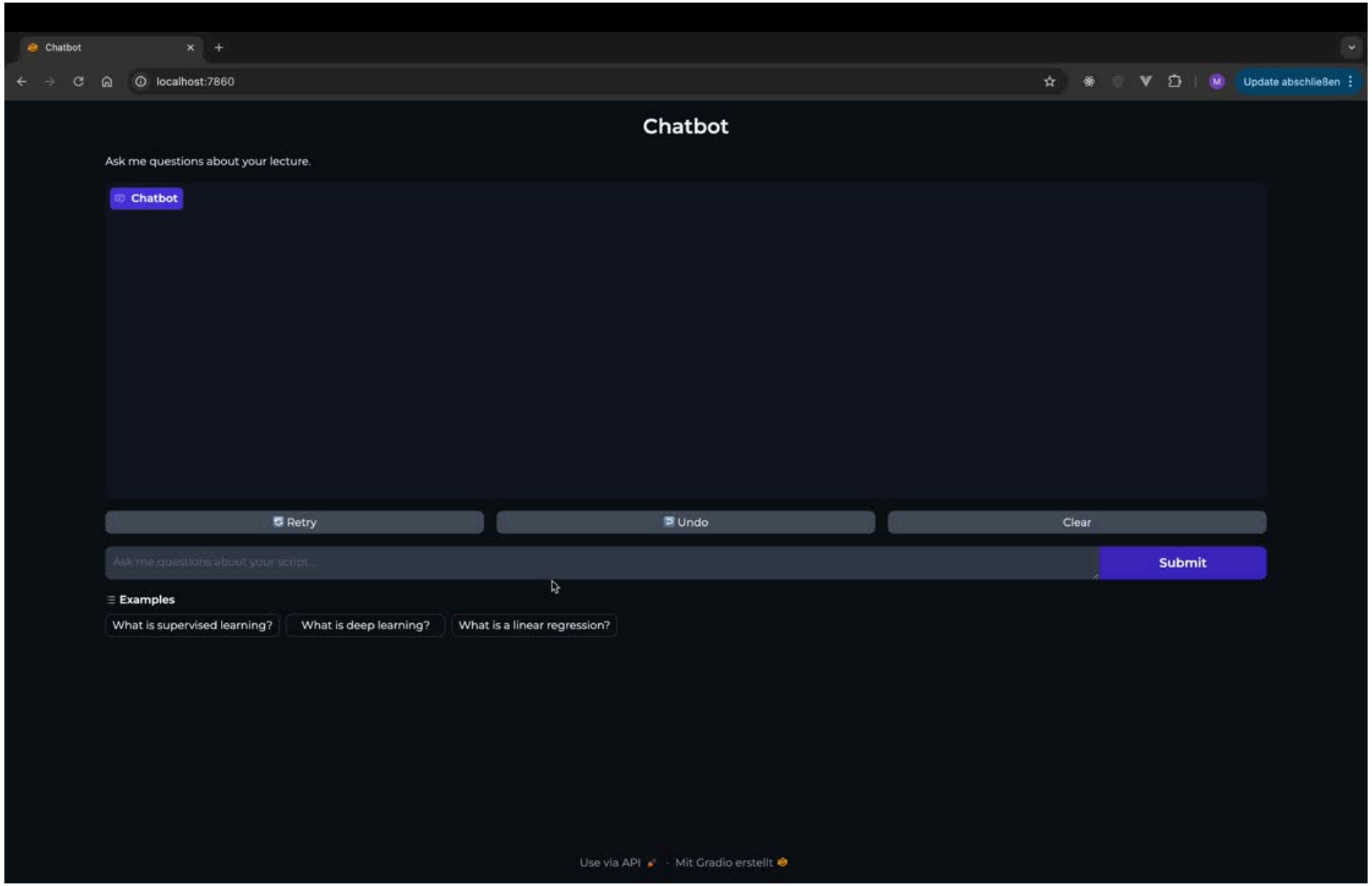
Session 5 - Building a Chatbot

SESSION 5 AGENDA

1. Demo of Target Solution
2. Target Architecture
3. Building Blocks

DEMO OF TARGET SOLUTION

Chatbot app in action.



Chatbot
localhost:7860
Update abschließen
Chatbot

Ask me questions about your lecture.

Chatbot
Retry
Undo
Clear
Ask me questions about your script...
Submit
Examples
What is supervised learning?
What is deep learning?
What is a linear regression?
Use via API · Mit Gradio erstellt

TARGET ARCHITECTURE

Frontend:

- **Web app built with Gradio, accessible via browser.**

Backend:

- **Python-based with FastAPI and LangChain.**

LLM Serving:

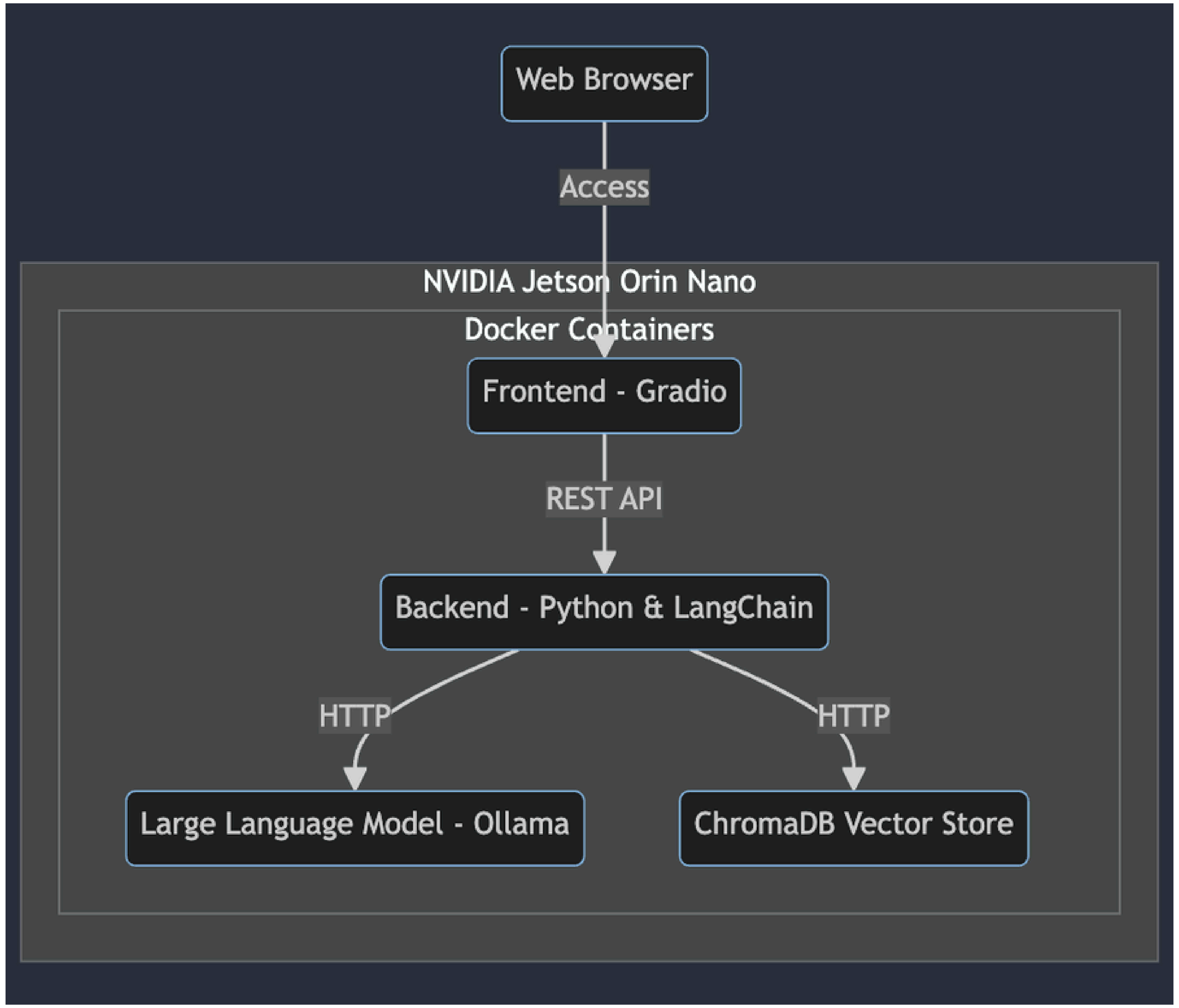
- **Ollama for managing large language models.**

Knowledge Storage:

- **Vector database for knowledge management.**

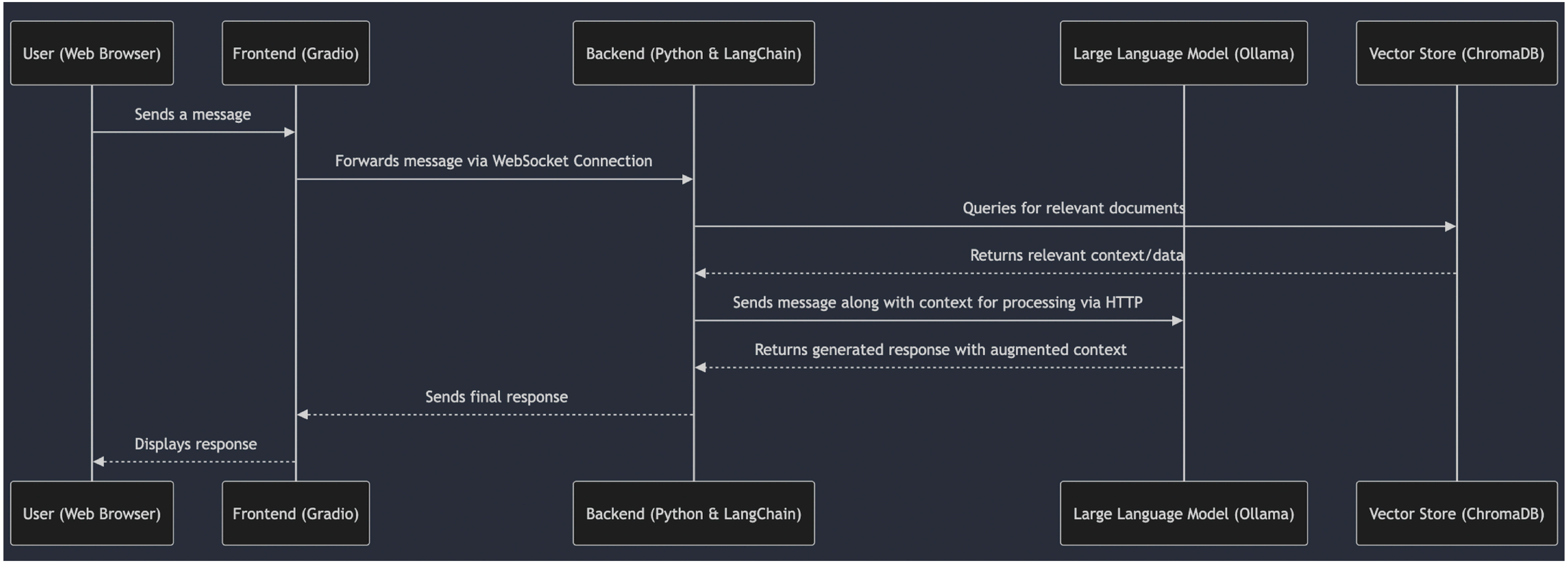
Deployment:

- **Docker containers for application deployment.**



TARGET ARCHITECTURE

User interaction workflow.

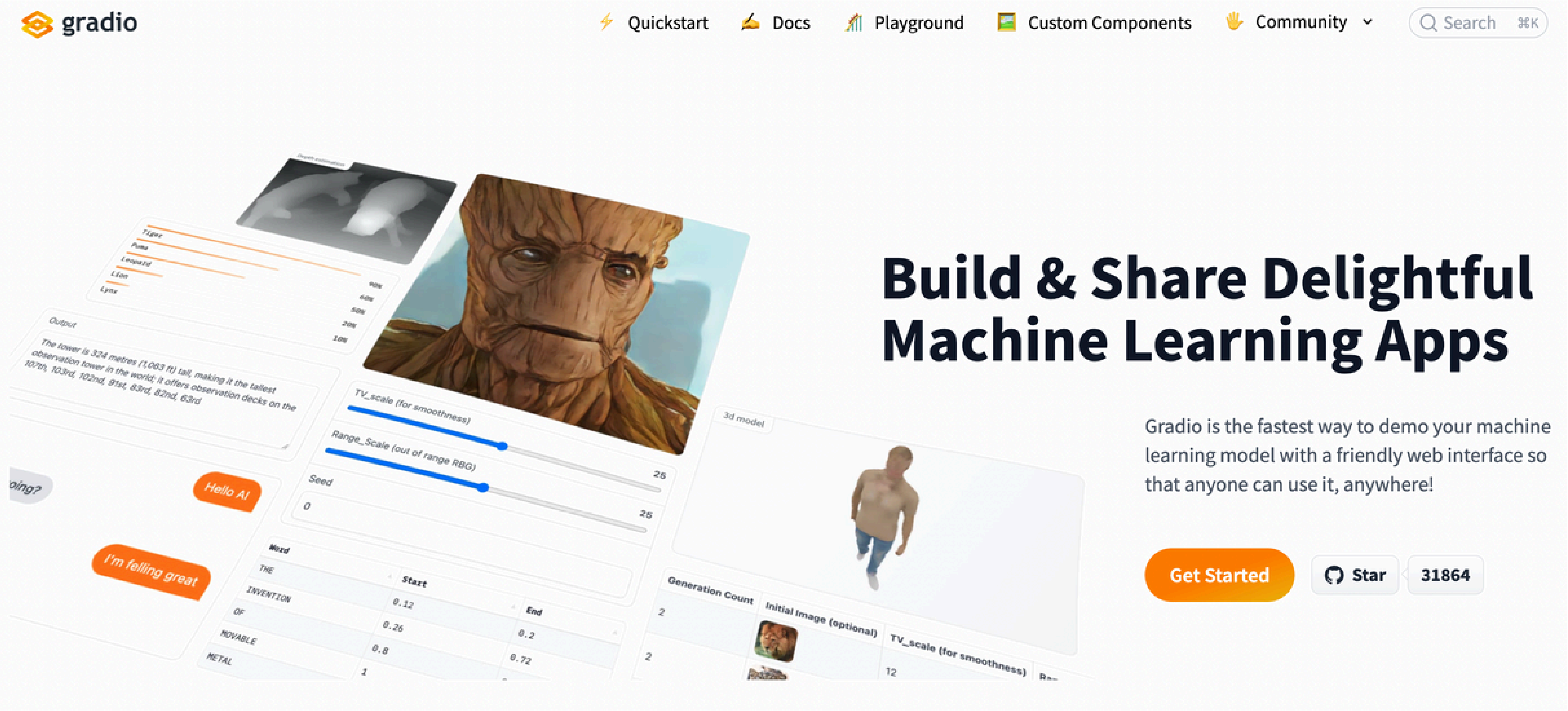

User (Web Browser)
Frontend (Gradio)
Backend (Python & LangChain)
Large Language Model (Ollama)
Vector Store (ChromaDB)
Sends a message
Forwards message via WebSocket Connection
Queries for relevant documents
Returns relevant context/data
Sends message along with context for processing via HTTP
Returns generated response with augmented context
Sends final response
Displays response
User (Web Browser)
Frontend (Gradio)
Backend (Python & LangChain)
Large Language Model (Ollama)
Vector Store (ChromaDB)

BUILDING BLOCKS

Gradio Frontend (webapp)

- Open-source Python library
- Build interactive ML interfaces without frontend code.
- Pre-built components for quick testing of ML models.
- Generates public links for real-time model interaction.
- Supports ML frameworks likeTensorFlow, PyTorch, Hugging Face, and more.

gradio

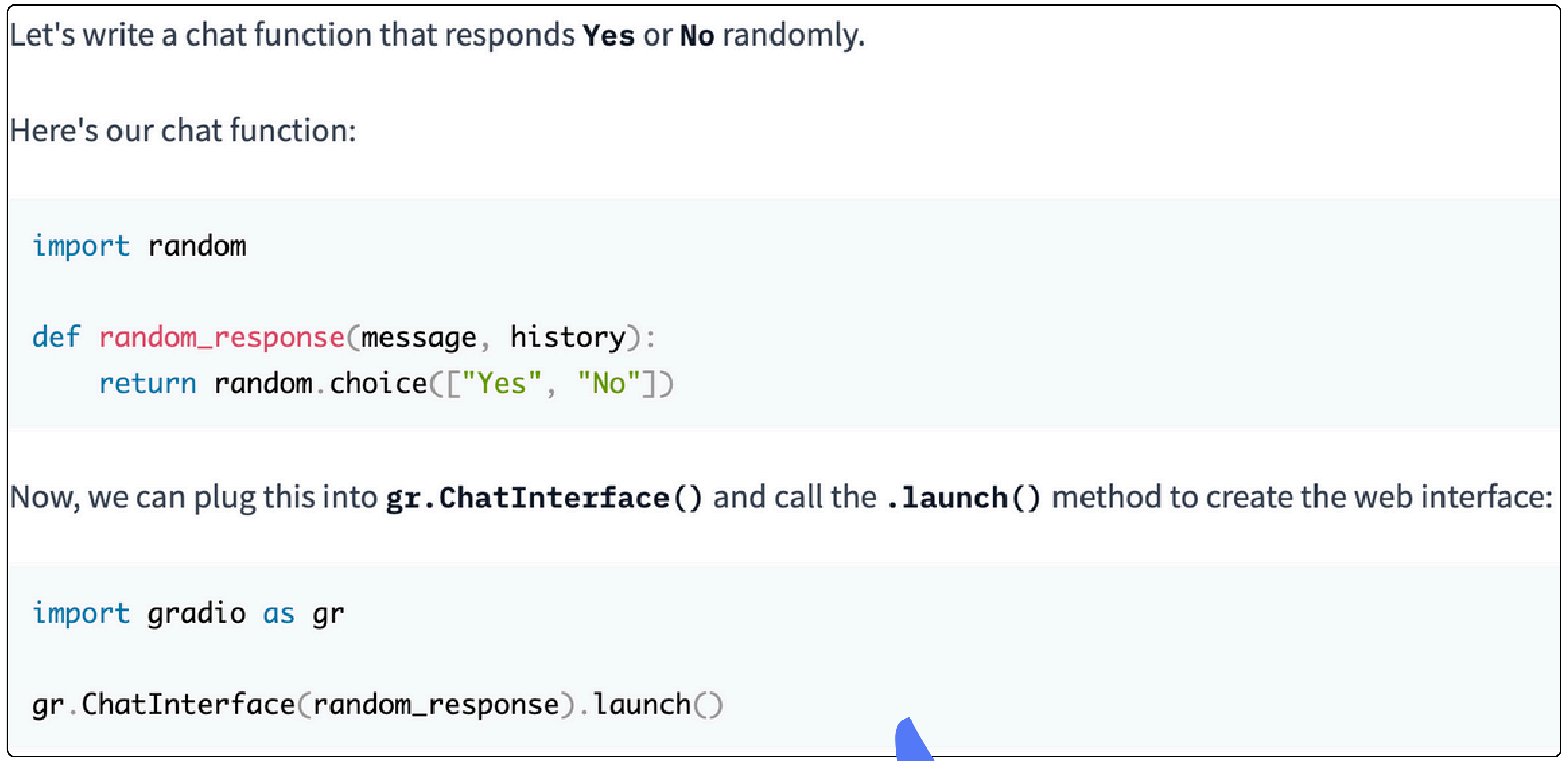


Build & Share Delightful Machine Learning Apps

Gradio is the fastest way to demo your machine learning model with a friendly web interface so that anyone can use it, anywhere!

BUILDING BLOCKS

Gradio - build fast ML webapps.



Let's write a chat function that responds **Yes** or **No** randomly.

Here's our chat function:

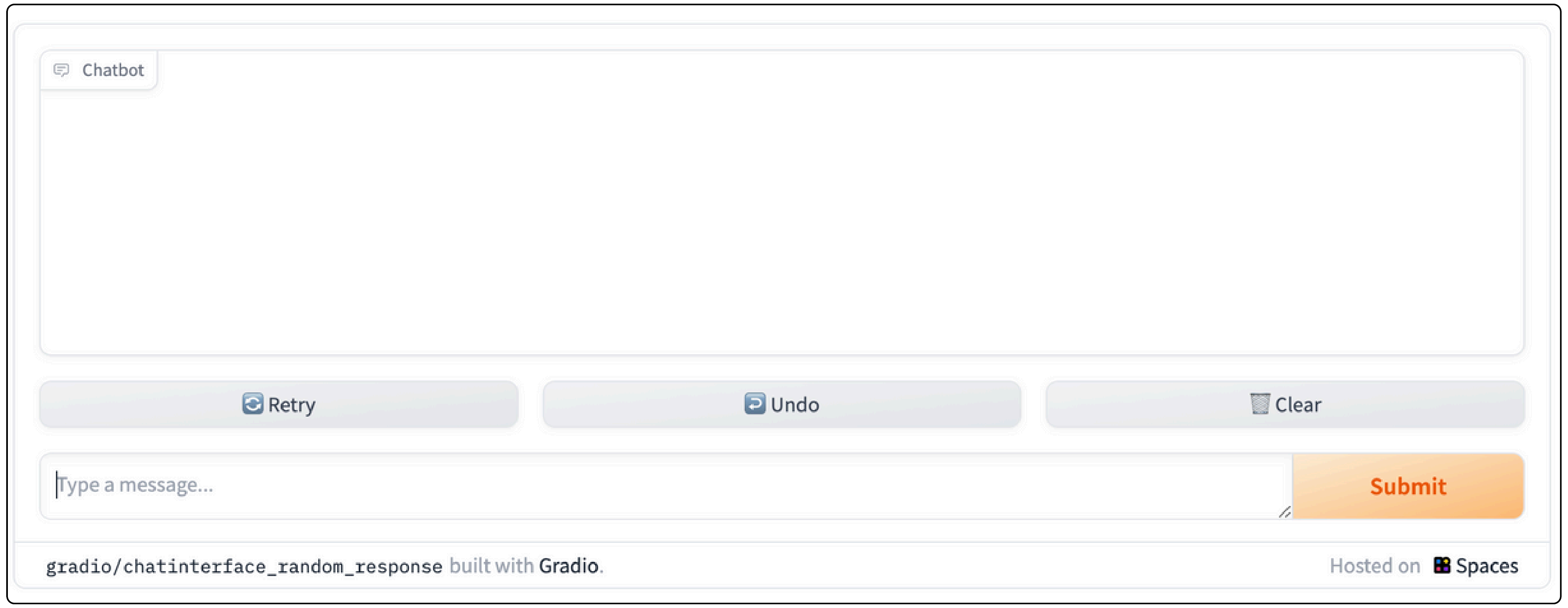
```python
import random

def random_response(message, history):
    return random.choice(["Yes", "No"])
```

Now, we can plug this into **`gr.ChatInterface()`** and call the **`.launch()`** method to create the web interface:

```python
import gradio as gr

gr.ChatInterface(random_response).launch()
```



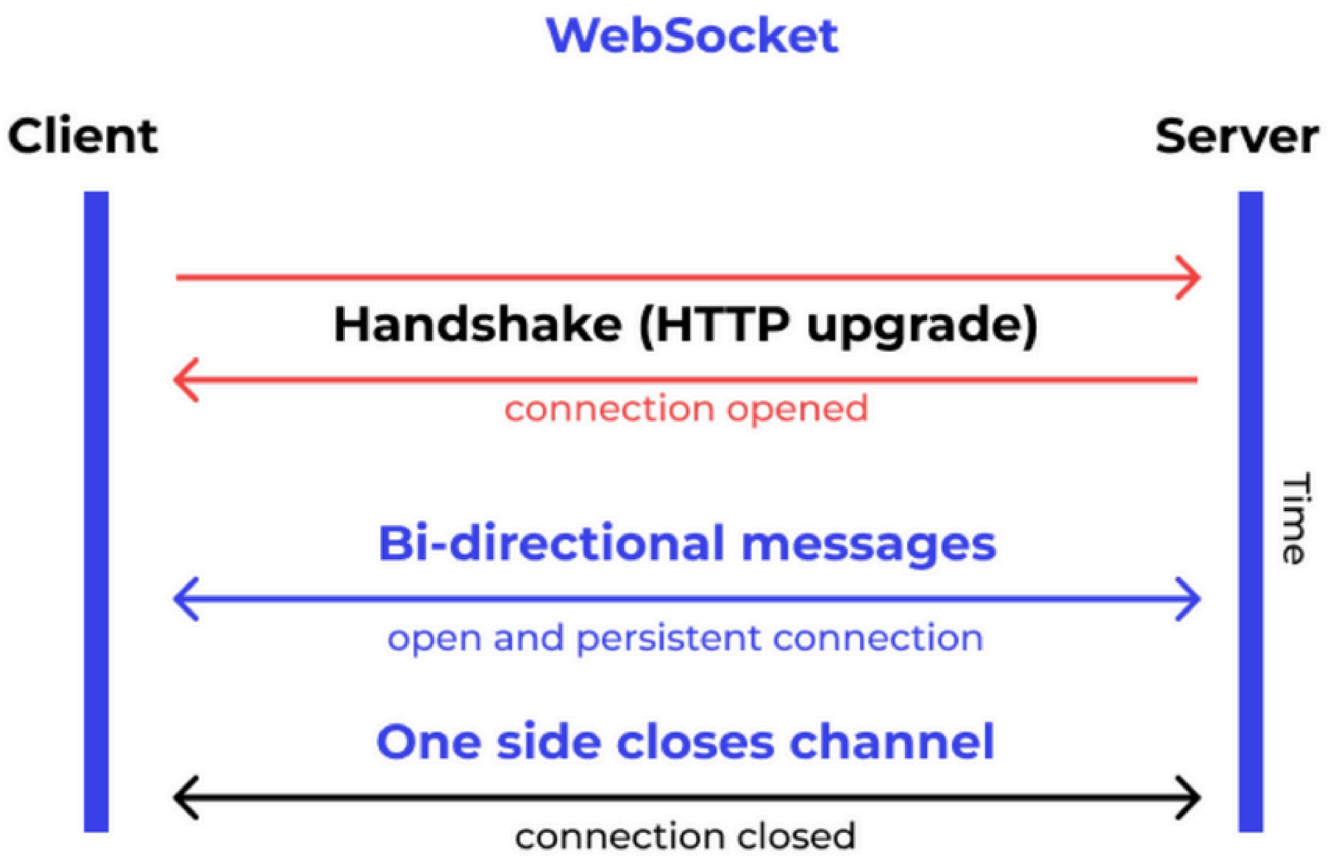
BUILDING BLOCKS

FastAPI Backend

- Asynchronous web framework optimized for building fast APIs.
- Simple syntax, leveraging Python type hints for automatic validation.
- Generates OpenAPI and Swagger documentation automatically.
- Supports async programming, WebSockets, and background tasks.

-> we will build Websocket API

FastAPI

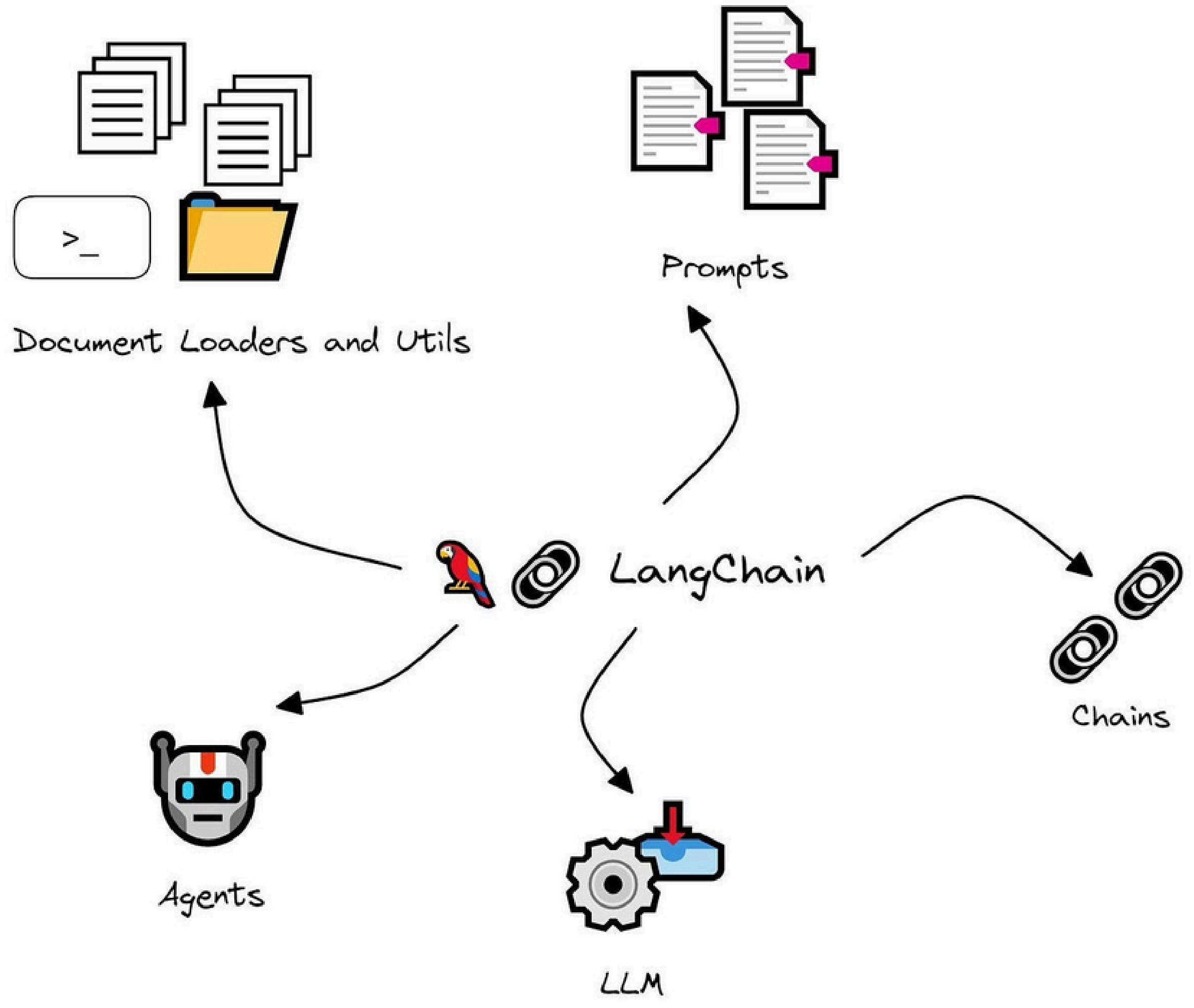


HTTPS://WWW.WALLARM.COM/WHAT/A-SIMPLE-EXPLANATION-OF-WHAT-A-WEBSOCKET-IS

BUILDING BLOCKS

RAG Chatbot with LangChain

- Build LLM based apps
- Build complex pipelines by linking LLMs and tools.
- Supports APIs, databases, and custom logic for flexible workflows.
- Enables context persistence across multiple interactions.

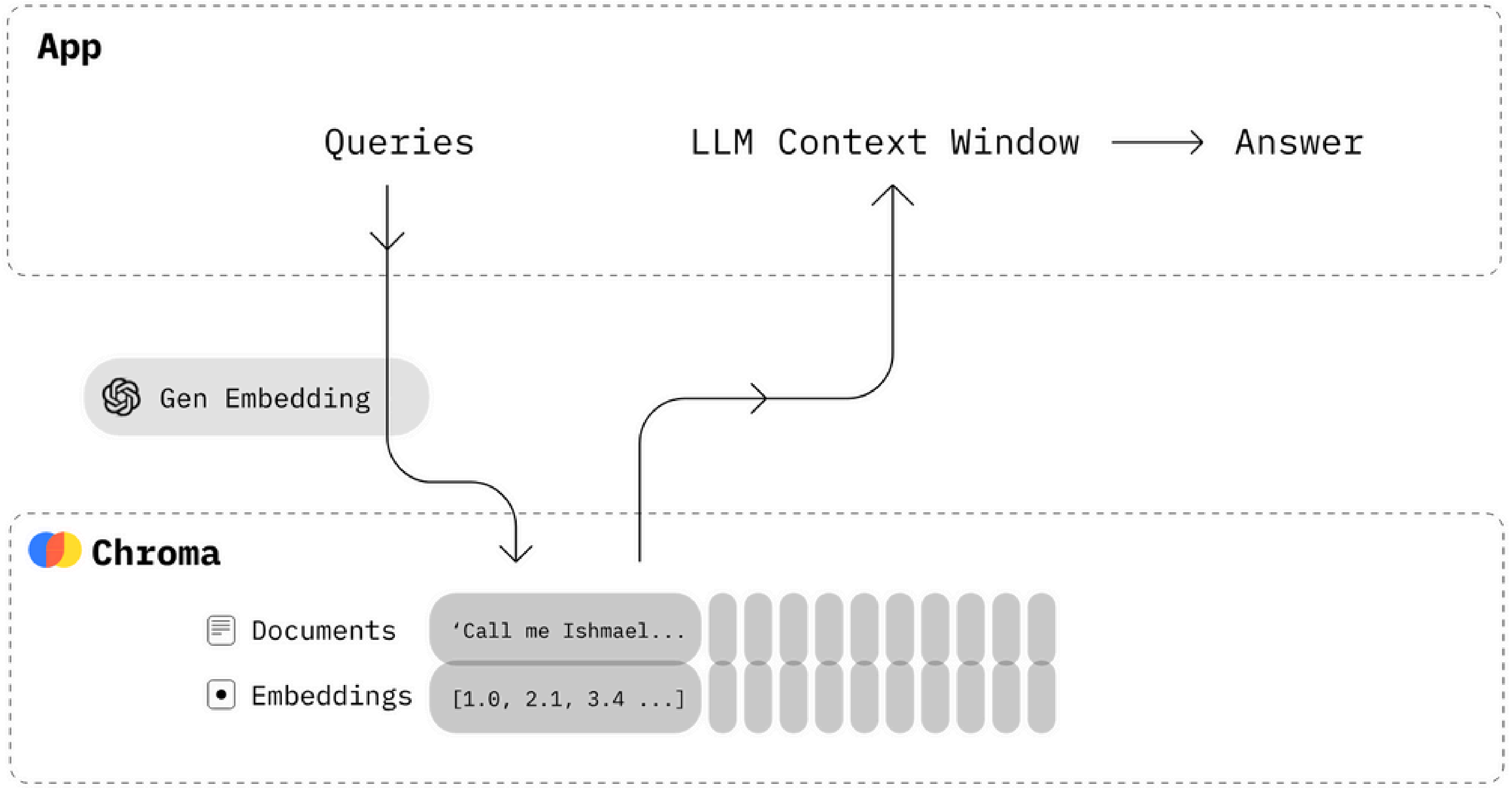


HTTPS://BLOG.STACKADEMIC.COM/WHAT-IS-LANGCHAIN-AND-HOW-TO-USE-IT-C9A656B80CEA

BUILDING BLOCKS

Chroma as Vector Database

- Specialized for storing and querying high-dimensional embeddings.
- Designed to handle large-scale data efficiently.
- Works with popular ML frameworks like LangChain.
- Enables fast similarity searches for embeddings-based applications.

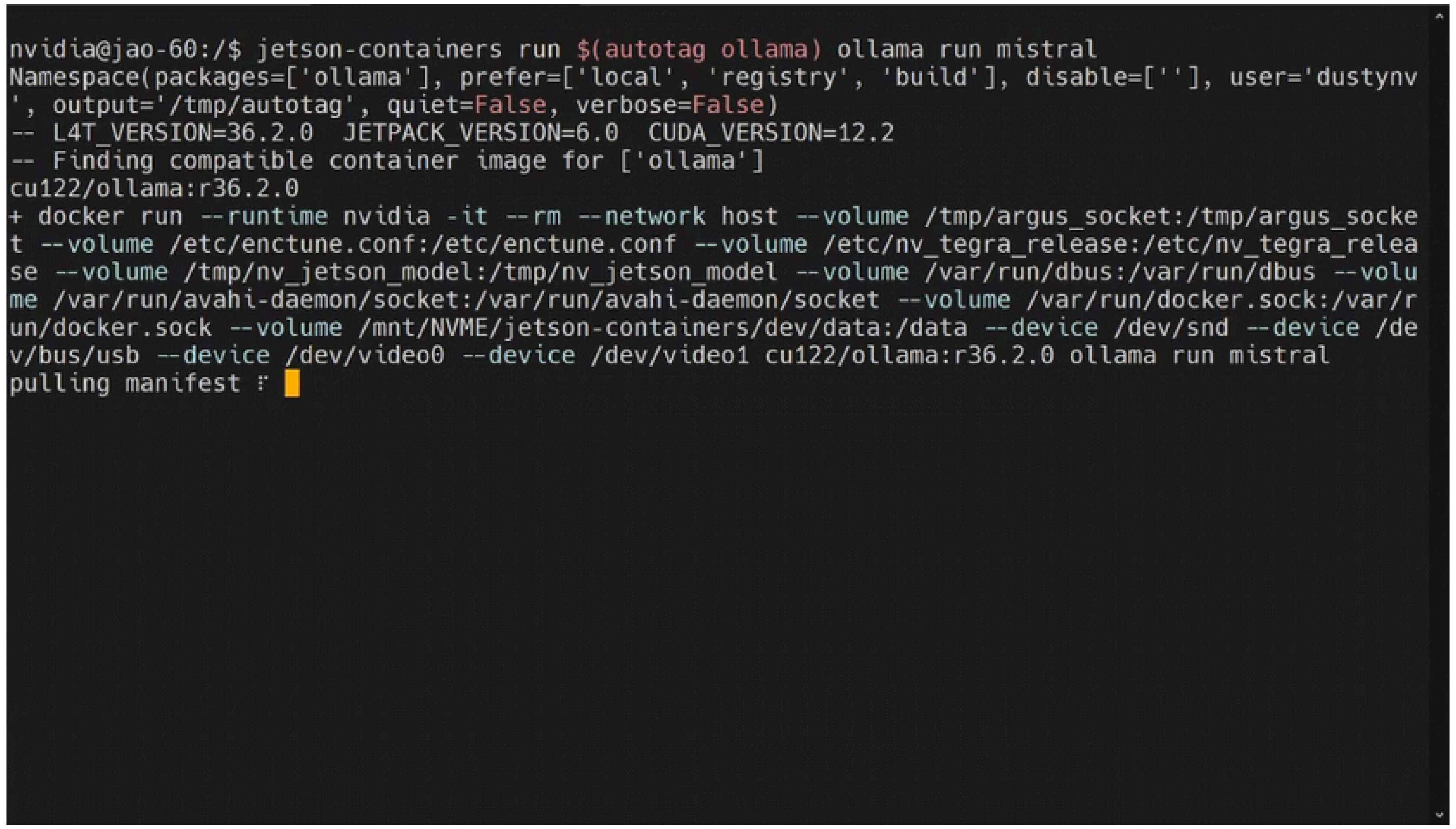


HTTPS://DOCS.TRYCHROMA.COM/

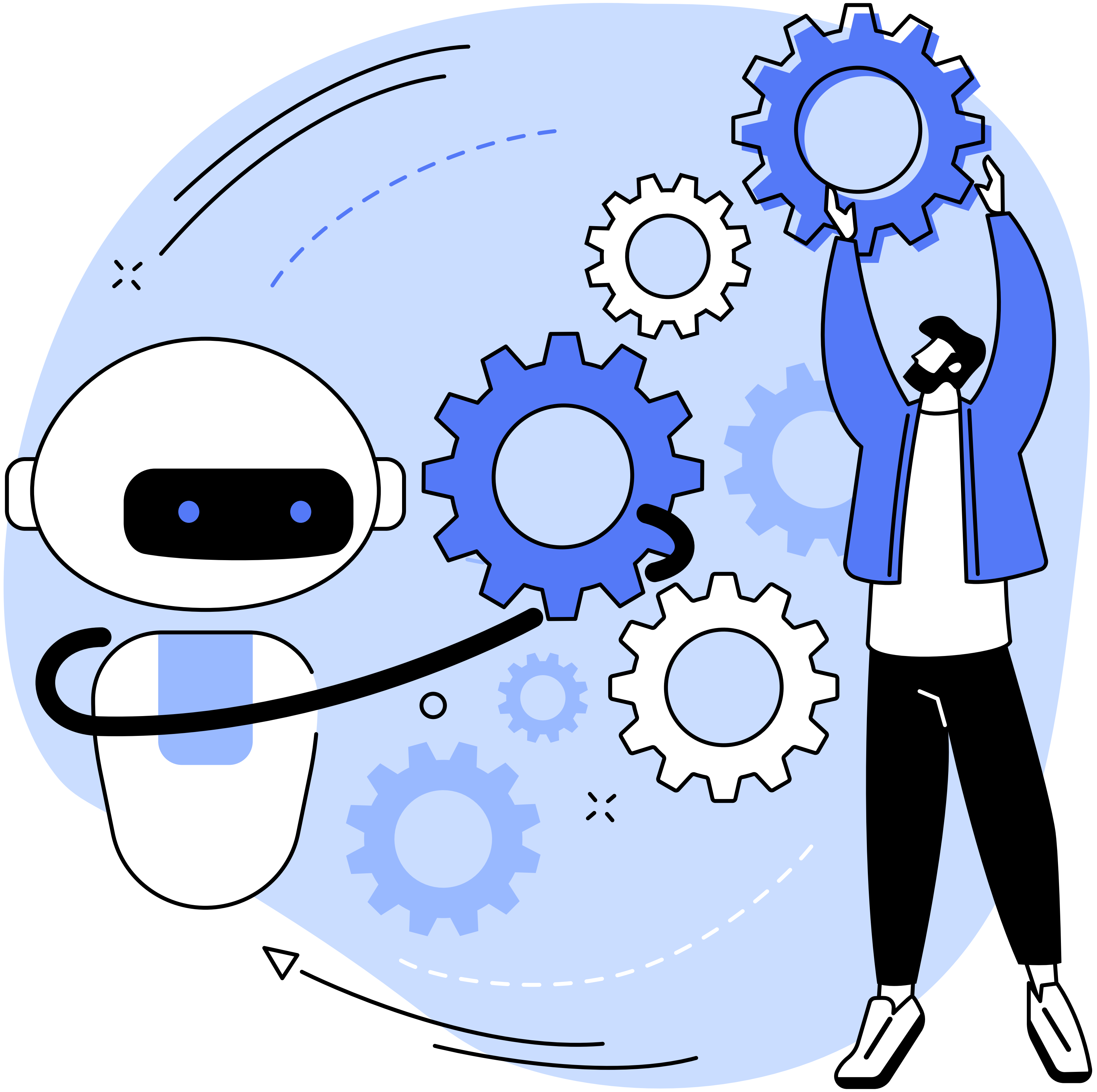
BUILDING BLOCKS

Ollama as LLM Runtime

- Run large language models on local machines efficiently.
- Keeps data local, ensuring better control over sensitive information.
- Designed for high-speed inference with minimal resource usage.
- Simple setup for running and experimenting with LLMs on your device.



```
nvidia@jao-60:/$ jetson-containers run $(autotag ollama) ollama run mistral
Namespace(packages=['ollama'], prefer=['local', 'registry', 'build'], disable=[''], user='dustynv
', output='/tmp/autotag', quiet=False, verbose=False)
-- L4T_VERSION=36.2.0  JETPACK_VERSION=6.0  CUDA_VERSION=12.2
-- Finding compatible container image for ['ollama']
cu122/ollama:r36.2.0
+ docker run --runtime nvidia -it --rm --network host --volume /tmp/argus_socket:/tmp/argus_socke
t --volume /etc/enctune.conf:/etc/enctune.conf --volume /etc/nv_tegra_release:/etc/nv_tegra_relea
se --volume /tmp/nv_jetson_model:/tmp/nv_jetson_model --volume /var/run/dbus:/var/run/dbus --volu
me /var/run/avahi-daemon/socket:/var/run/avahi-daemon/socket --volume /var/run/docker.sock:/var/r
un/docker.sock --volume /mnt/NVME/jetson-containers/dev/data:/data --device /dev/snd --device /de
v/bus/usb --device /dev/video0 --device /dev/video1 cu122/ollama:r36.2.0 ollama run mistral
pulling manifest ⠏
```



IT’S YOUR TURN