HOW TO BUILD A CHATBOT

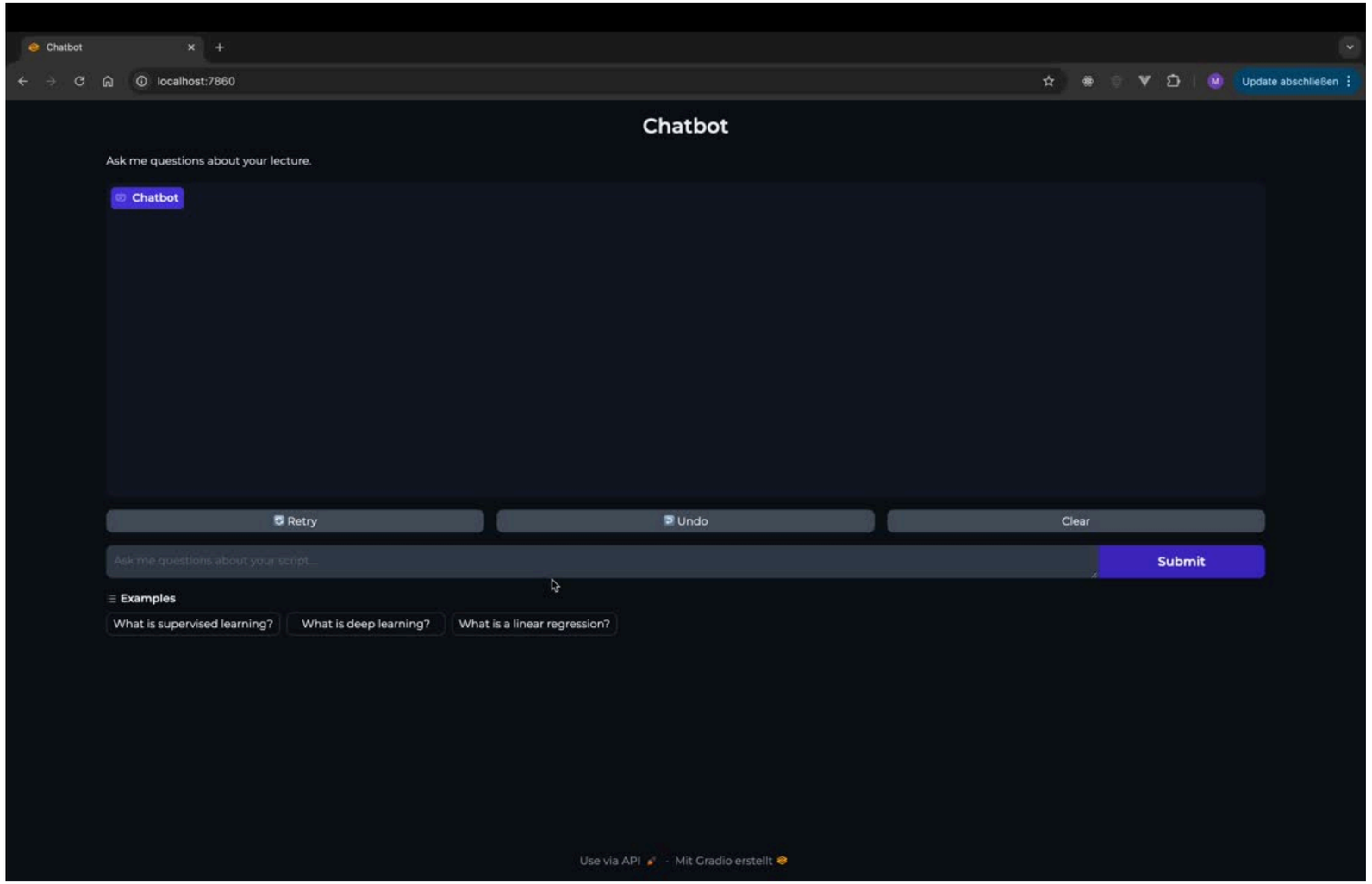
Session 5 - Building a Chatbot

SESSION 5 AGENDA

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

DEMO OF TARGET SOLUTION

Chatbot App in Action



TARGET ARCHITECTURE

Frontend:

- Gradio Webapp, accessible via browser.

Backend:

- Python-based with FastAPI and LangChain.

LLM Serving:

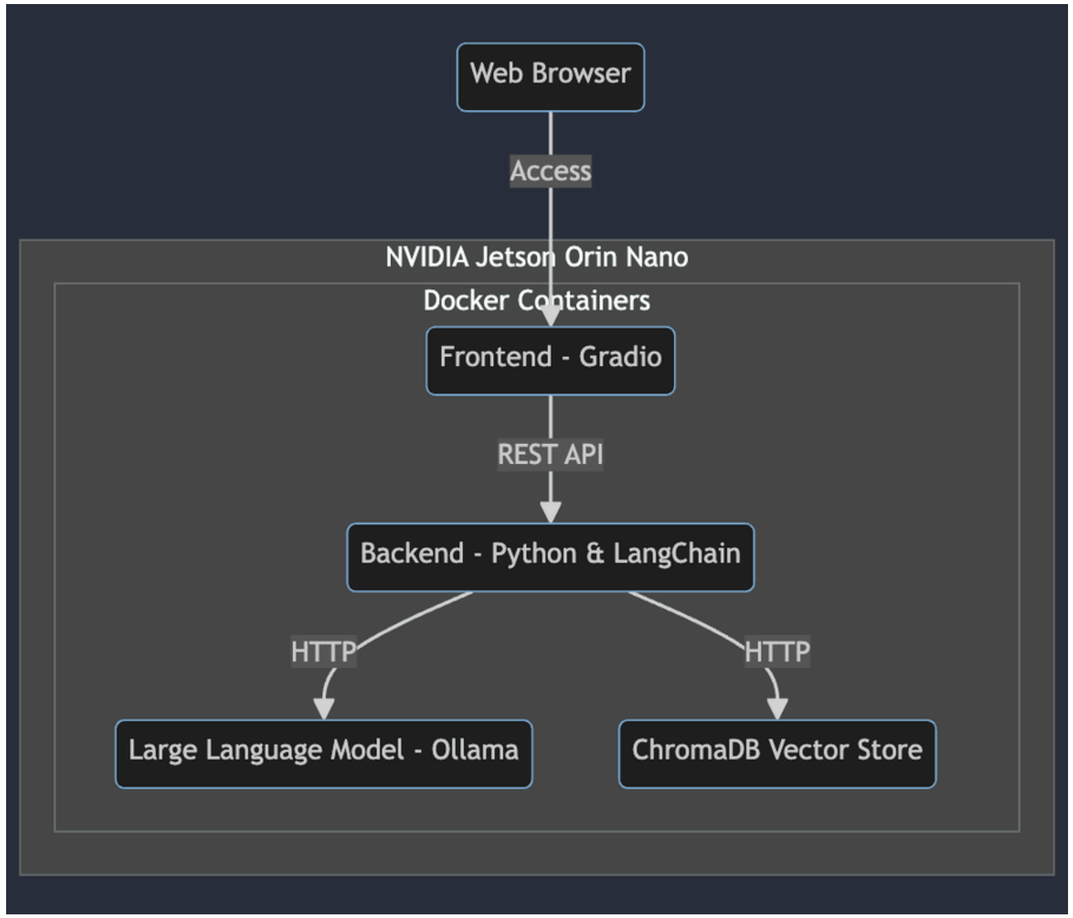
- Ollama for managing large language models.

Knowledge Storage:

- Vector database for knowledge management.

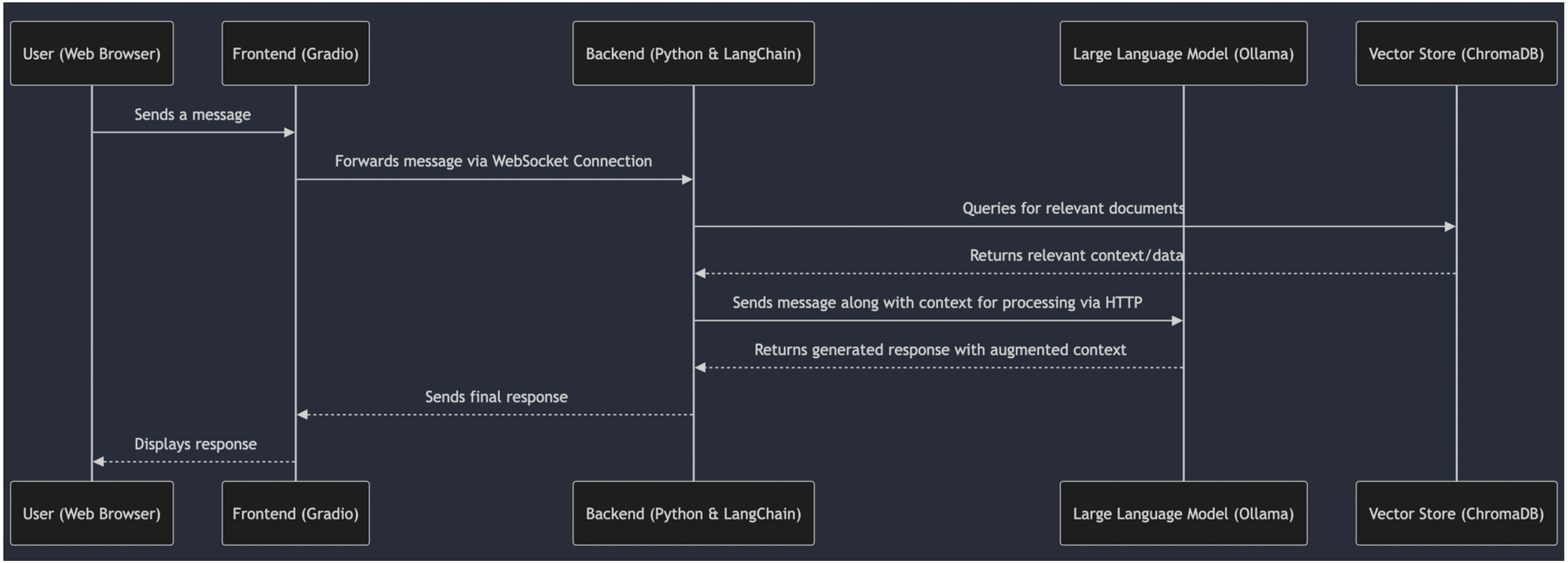
Deployment:

- Docker containers for application deployment.



TARGET ARCHITECTURE

User Interaction Workflow

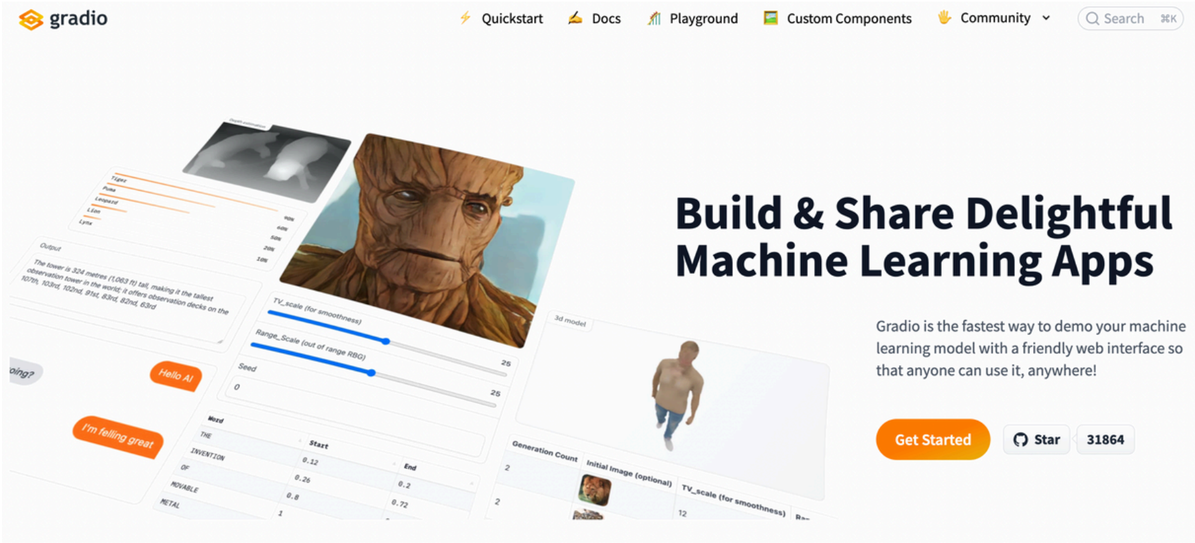


BUILDING BLOCKS

Frontend - Gradio Webapp

- Open-source Python library
- Build interactive ML interfaces
- Pre-built components for quick testing of ML models.
- Supports ML frameworks likeTensorFlow, PyTorch, Hugging Face, and more.

gradio



[1]

BUILDING BLOCKS

Gradio - build fast ML webapps.

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

Here's our chat function:

```
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:

```
import gradio as gr

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

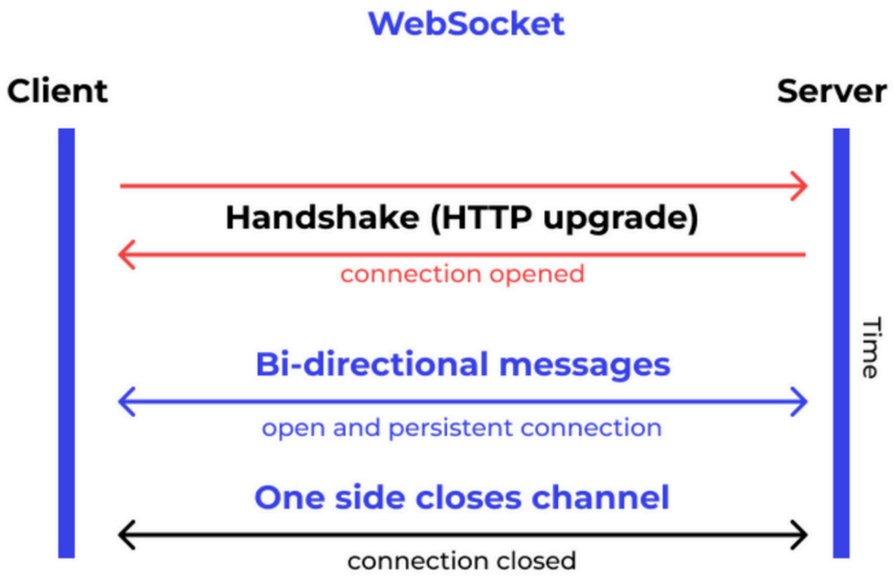
BUILDING BLOCKS

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.

-> Chatbot is using Websocket Protocol

FastAPI

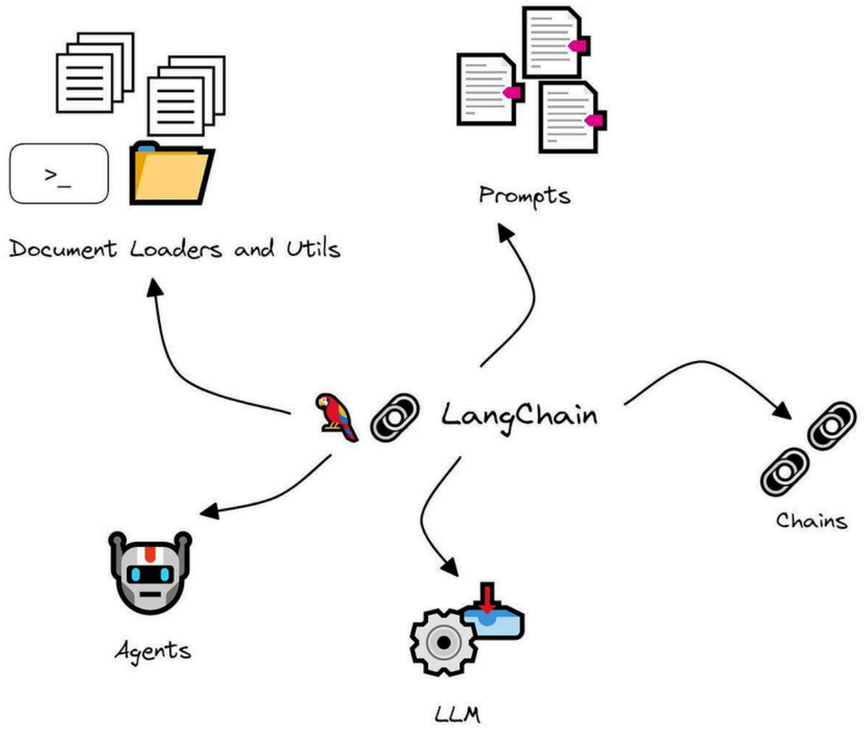


[2]

BUILDING BLOCKS

RAG Chatbot with LangChain

- Build LLM based apps
- Supports APIs, databases, and custom logic for flexible workflows.
- Enables context persistence across multiple interactions.

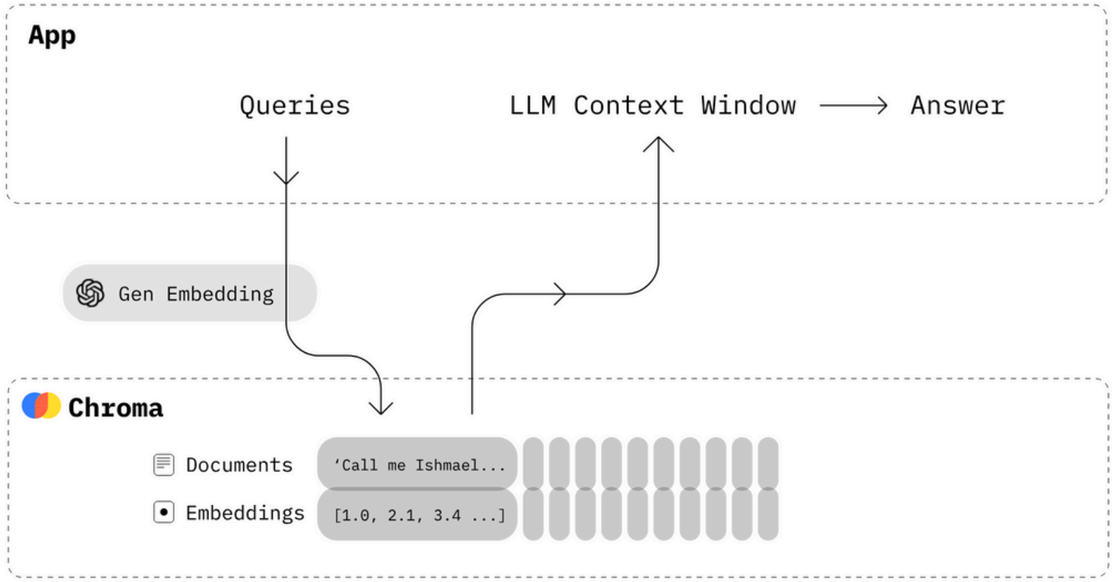


[3]

BUILDING BLOCKS

Chroma as Vector Database

- Specialized for storing and querying high-dimensional embeddings.
- Works with popular ML frameworks like LangChain.
- Enables fast similarity searches for embeddings-based applications.

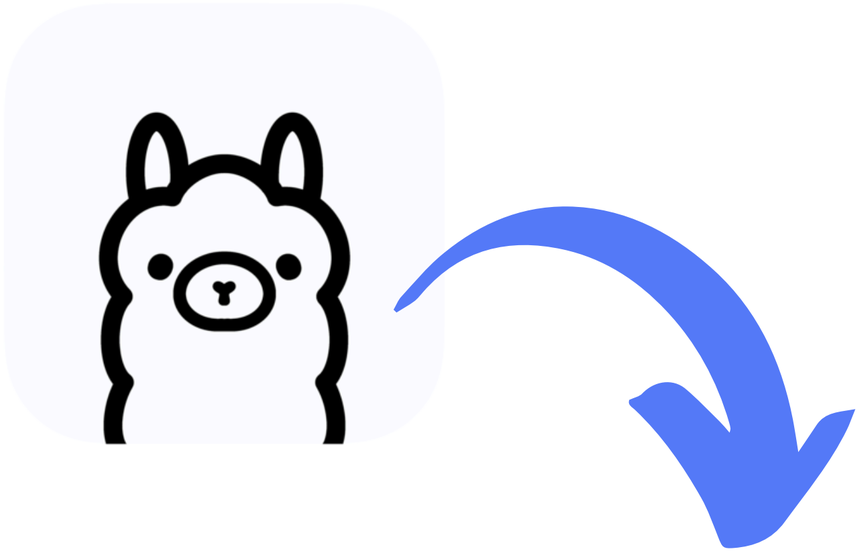


[3]

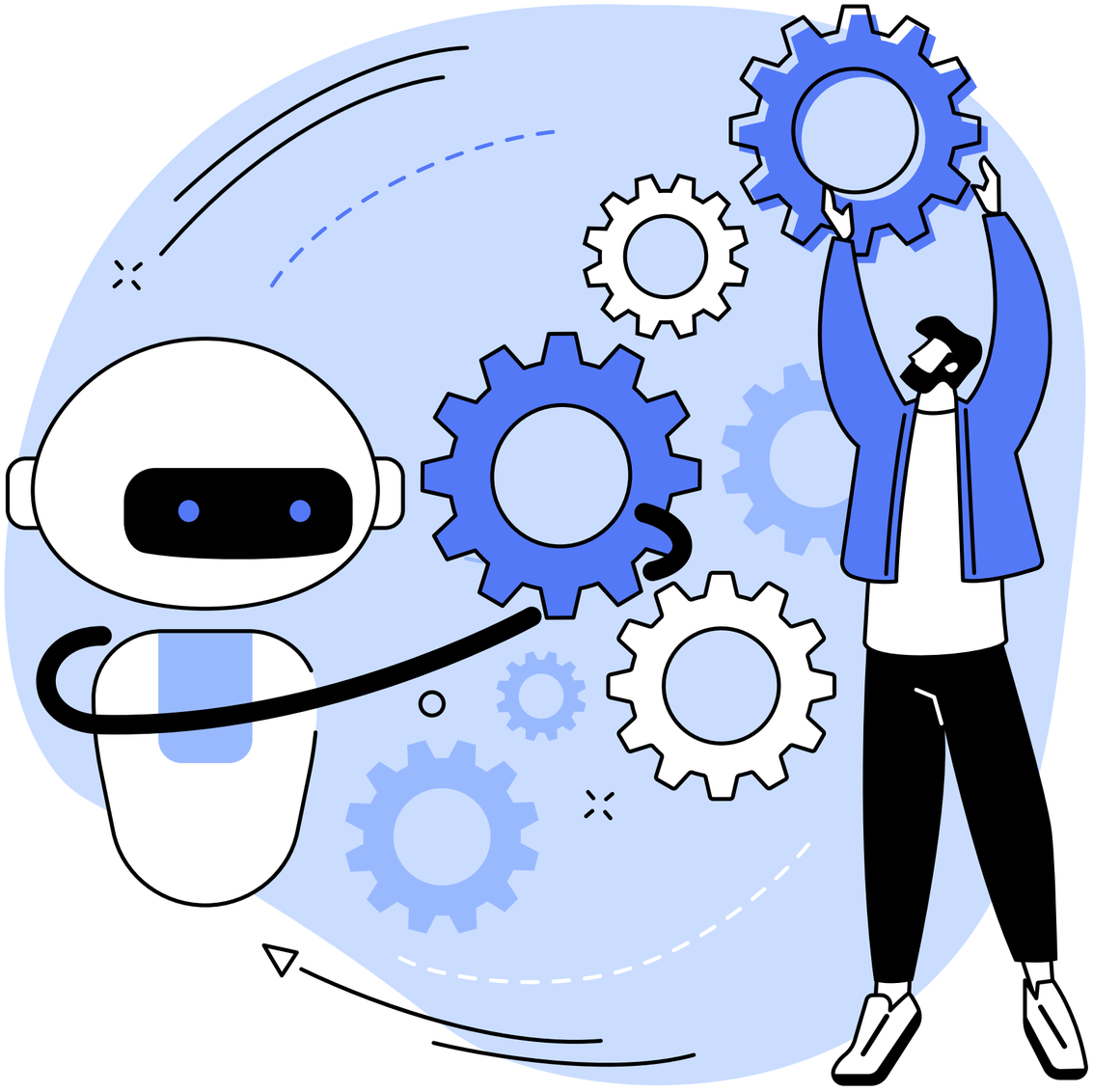
BUILDING BLOCKS

Ollama as LLM Runtime

- Run large language models on local machines efficiently.
- 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

Sources:

[1]: https://qdrant.tech/articles/what-is-rag-in-ai/

[2]: https://www.wallarm.com/what/a-simple-explanation-of-what-a-websocket-is

[3]: https://blog.stackademic.com/what-is-langchain-and-how-to-use-it-c9a656b80cea

[4]: https://docs.trychroma.com/