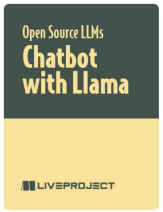
Portable LLM apps in Docker

Michael Yuan, PhD
Docker Captain, Gaia Network

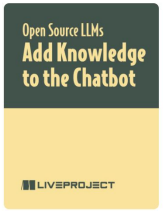
x: juntao github: juntao

My latest book



Project 1 Chatbot with Llama

In this liveProject, you'll take on the role of a full-stack developer working for a school district. Your district wants to further student learning by developing its own custom large language model (LLM) to assist students, parents, and teachers. Your goal is to develop an MVP of this by creating a chatbot that can answer chemistry questions and provide follow-up answers and conversations. You'll utilize the open source Llama LLM from Meta AI to do this. Your model will need to run with very low resources—it needs to work on the school's outdated computers—and be equipped with an intuitive user interface. Let's get started!

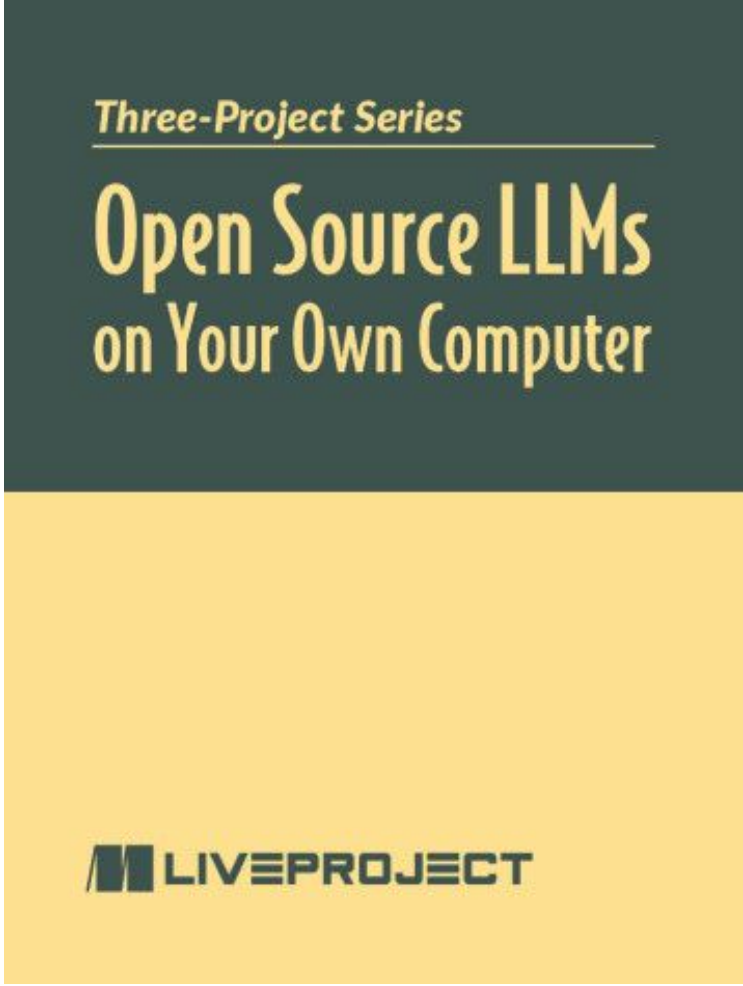


Project 2 Add Knowledge to the Chatbot

Fine-tuning an LLM is time-consuming and expensive! That's why your local school district has tasked you with using RAG (Retrieval Augmented Generation) to help improve the capabilities of a chemistry chatbot based on Meta AI's Laama. RAG allows an LLM to search a database to help answer questions, avoiding any unfortunate hallucinations. You'll create a vector database for external knowledge for your chatbot's RAG, establish an RAG API server for it to use, and then deploy your new bot to both the web and Discord.

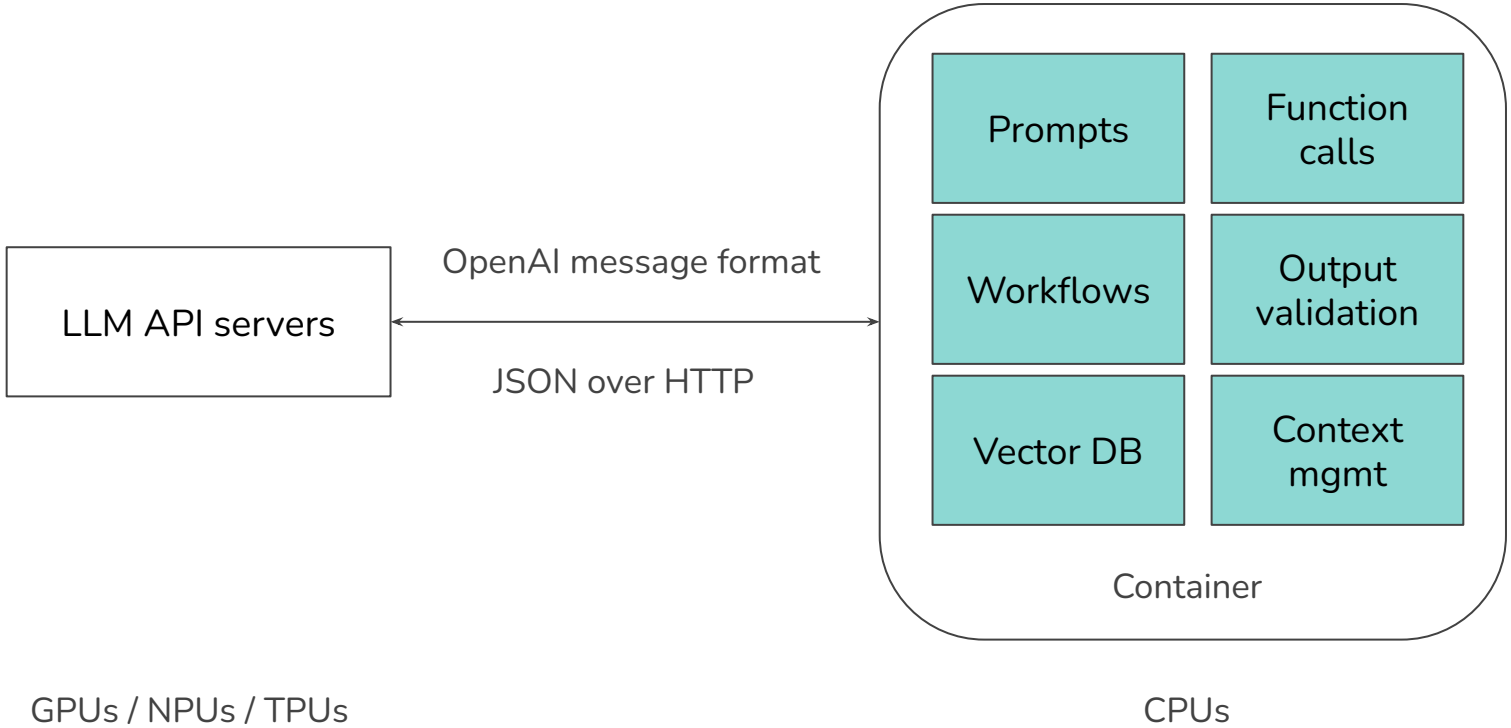
Project 3 Fine-Tune the Llama Model

Your local school district has a basic LLM chatbot based on Llama—now it needs fine-tuning! That's where you come in. In this liveProject, you'll utilize the Supervised Fine Tuning (SFT) approach to customize a chemistry chatbot for teaching children. SFT uses question-and-answer pairs to train a model to answer questions with a given answer. To achieve this, you will need to prepare your training data, utilizellama.cpp tools for LORA fine-tuning, and CLI tools to test inference on a fine-tuned model.



https://www.manning.com/liveprojectseries/open-source-llms-on-your-own-computer

The “LLM sidecar” pattern




Elon Musk @elonmusk · Nov 14
Btw, I'd like to apologize for Twitter being super slow in many countries.
App is doing >1000 poorly batched RPCs just to render a home timeline!
24.3K
14.7K
173.8K


Elon Musk @elonmusk · Nov 14
Replying to @sampullara
I was told ~1200 RPCs independently by several engineers at Twitter, which matches # of microservices. The ex-employee is wrong.
Same app in US takes ~2 secs to refresh (too long), but ~20 secs in India, due to bad batching/verbose comms. Actually useful data transferred is low.
604
861
8,229


Elon Musk @elonmusk · Nov 14
Replying to @elonmusk and @sampullara
Part of today will be turning off the "microservices" bloatware. Less than 20% are actually needed for Twitter to work!
1,694
4,887
9,635

Problems of loosely coupled LLM servers

- Model versions must be matched to runtime versions
  - When the runtime is upgraded to support a new version of a popular model, many old (custom finetuned) models would stop working
- Different models require different prompts
  - Each model has its unique prompt template
  - Some models can follow instructions in system prompts better
  - Some models can follow complex prompts such as CoT while others cannot
  - Different models have different context sizes
- Different models require different preparations of the RAG knowledge base
  - Chunking strategies
  - Vectorization / search strategies
  - Embedding vector sizes
  - Embedding context lengths
- Different models require different post-processing of responses
  - Some models are more reliable at generating JSON or code
  - Some models can handle tool call prompts better

LLM apps are fundamentally tightly coupled

We need to embed the LLM into the app and deliver it as an entire package


IT WORKS ON MY MACHINE
THEN WE'LL SHIP YOUR MACHINE
AND THAT IS HOW DOCKER WAS BORN

What’s inside the container

Not Python

Extreme bloat

The PyTorch Docker image size is between 5GB to 8GB. Adding large model files, it is very difficult to run a Python setup on edge devices or servers.

Complexity

It is very hard to install all required Python packages for Whisper, LLM, TTS and other models, as well as HTTP servers on the same edge server. Supply chain security is also a major issue for complex Python packages.

Performance

Python is 10,000x slower than native code. Even as Python may only be used for workflow glue code or for lightweight computation (e.g., tokenization), it is still too slow for resource constrained edge devices.

Greg Brockman
@gdb

Much of modern ML engineering is making Python not be your bottleneck.

6:55 AM · 7/6/23 from Earth · **244K** Views

Greg Brockman
@gdb

current status: installing dependencies

10:34 PM · 2/12/24 From Earth · **33K** Views

Bojan Tunguz @tunguz · Apr 21
AGI will be built with Python.

Let that sink in.

519 382 5,084 3.8M

Elon Musk
@elonmusk

Rust

4:40 AM · Apr 22, 2023 · **3.7M** Views

682 Retweets **333** Quotes **10.4K** Likes **334** Bookmarks

Not Ollama

Model selection

Ollama only supports LLMs and recently VLMs.

But agent apps need to incorporate many models, including traditional vision, audio, and OCR models based on Torch.

Operational weight

Ollama is a large GO app built on top of llama.cpp. It incorporate multiple platform-specific binaries for portability.

It requires a sudo daemon and access to its proprietary model hub.

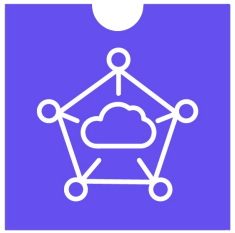
Not llama.cpp, whisper.cpp, flux.cpp

Developer challenges

Difficult to interface and integrate with several different C/C++ based libraries.

Safety

Prone to memory errors and crashes. This is especially a problem with GPU drivers for the container runtime..



WasmEdgeRuntime

https://github.com/WasmEdge/WasmEdge

https://github.com/LlamaEdge/LlamaEdge

- Lightweight
- Portable across OSes / CPUs / GPUs / NPUs
- Model agnostic
  - GenAI: LLMs, VLMs, voice-to-text, text-to-voice, text-to-image etc.
  - Also supports Torch, ONNX models (YoLo, MediaPipe, OCR etc)
- Standardized Rust and JS API
- Easily embeddable into applications

gaia

Lightweight and portable containers for “personal knowledge agents”

- Any open-source / fine-tuned LLMs + Multimodal AI models
- LlamaEdge + WasmEdge runtimes
- Qdrant vector database
- Personal / private knowledge as vector embeddings
- Tailored prompts and search strategies for the LLM and knowledge embeddings
- OpenAI compatible API
- Built-in HTTPS tunneling
- Built-in access and payment gates

https://github.com/GaiaNet-AI/gaianet-node

Try it!

Pull and run the Docker image for the model, runtime, and knowledge base

```
docker run --name gaianet \
  -p 8080:8080 \
  -v $(pwd)/qdrant_storage:/root/gaianet/qdrant/storage:z \
  gaianet/qwen2-0.5b-instruct_rustlang:latest
```

Chat!

```
http://localhost:8080/
```

https://hub.docker.com/repository/docker/gaianet/qwen2-0.5b-instruct_rustlang/general

Convert a string to i32 integer

Certainly! In Rust, you can convert a string to an i32 integer using the parse method. Here's how you can do it:

rust Copy code

```rust
fn main() {
    let s = "123";

    // Attempt to parse the string as an i32
    match s.parse::<i32>() {
        Ok(num) => println!("The number is: {}", num),
        Err(e) => println!("Error parsing integer: {}", e),
    }
}
```

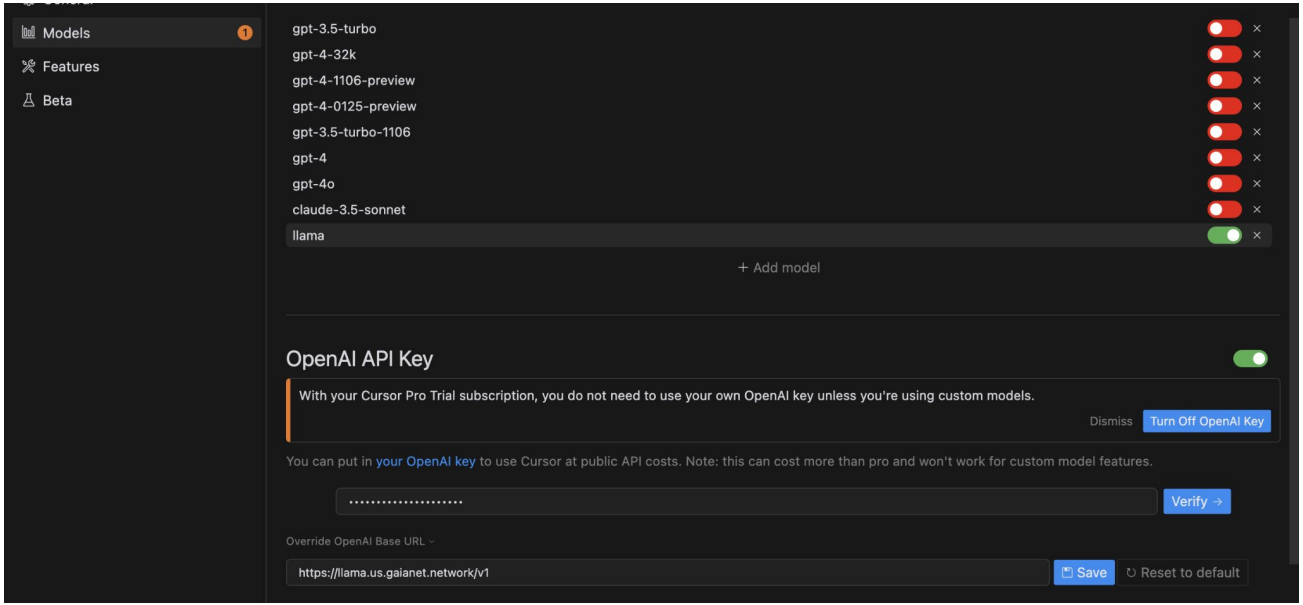
```
> The number is: 123
```

In this example, s.parse::<i32>() tries to convert the string s into an i32. The match statement is used to handle two possible outcomes:

1. Ok(num): This means that the conversion was successful, and num will contain the parsed integer.
2. Err(e): This means that there was an error during parsing (e.g., if the string was not a valid number), and e will be an instance of std::num::ParseIntError.

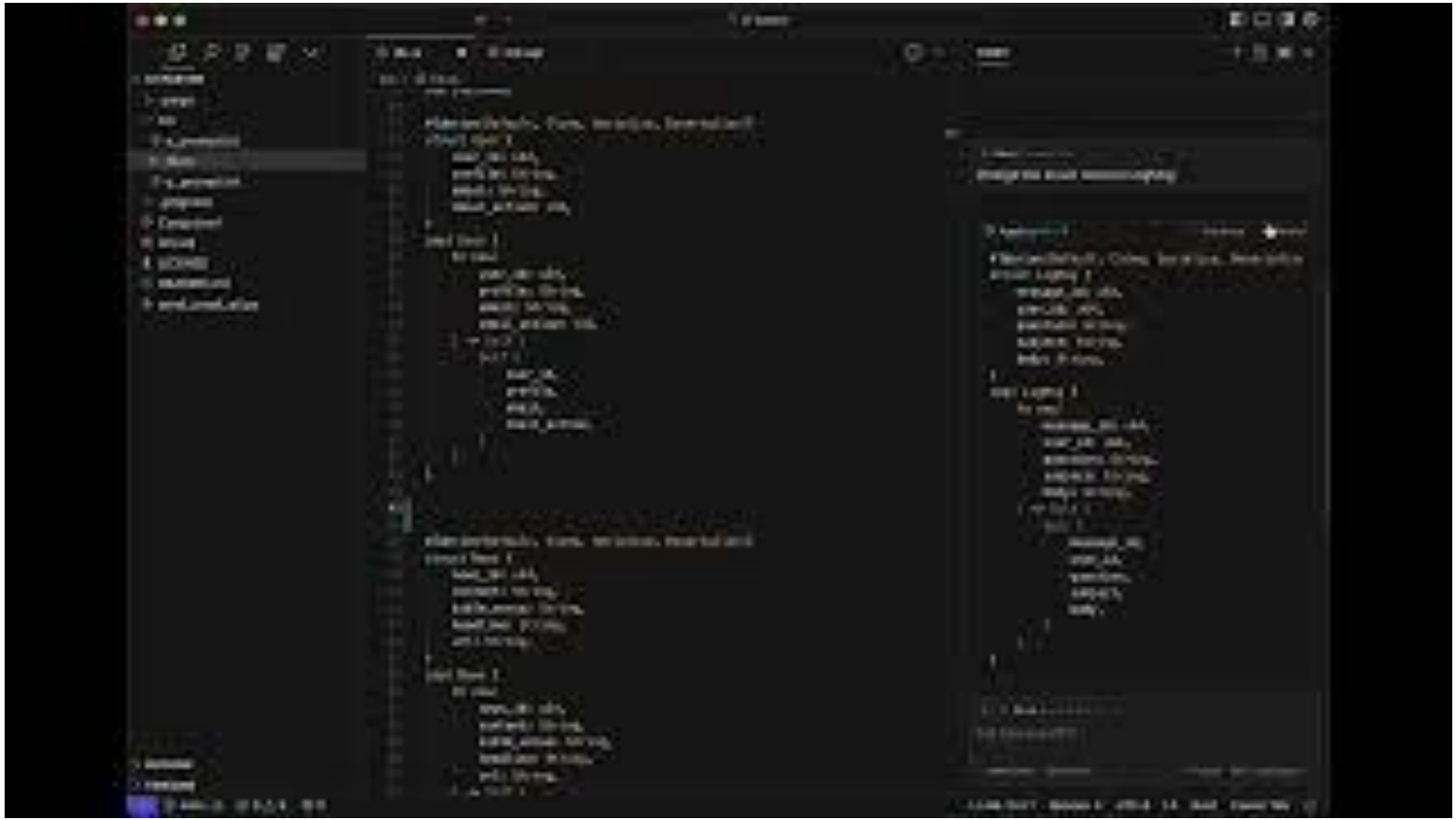
This is a safe way to handle potential errors when converting strings to integers.

Demo #1: A Rust coding agent for Cursor

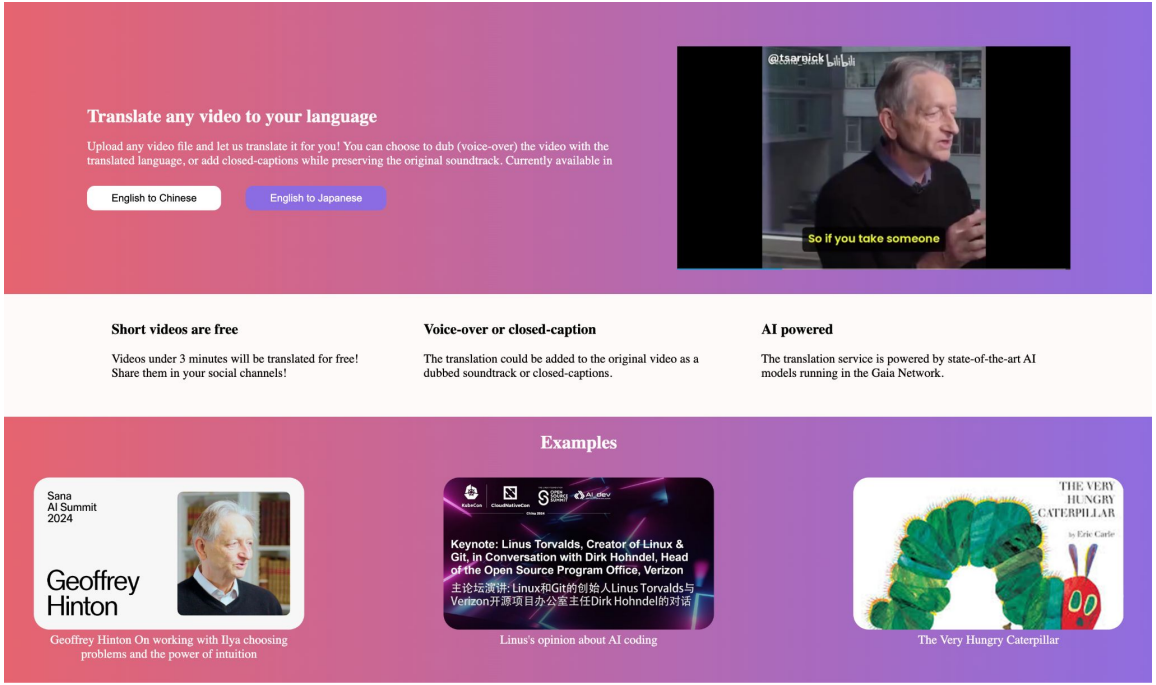


https://docs.gaianet.ai/user-guide/apps/cursor

Use an Open source LLM and Gaia to create a Rust coding assistant on Cursor



Demo #2: A multi-agent app for video translation



https://VideoLangua.com/

But, Docker images are not really portable

You need different images for each CPU and GPU combo

cuda12

Last pushed **2 months ago** by juntaoyuan

Digest	OS/ARCH
786241716 5f0	linux/amd64
fffdb640759f	linux/arm64

TAG

cuda11

Last pushed **2 months ago** by juntaoyuan

Digest	OS/ARCH
148ef2a8e6ed	linux/amd64
03346462c520	linux/arm64

TAG

latest

Last pushed **2 months ago** by juntaoyuan

Digest	OS/ARCH
caafe95c22c7	linux/amd64
5a46957593ca	linux/arm64

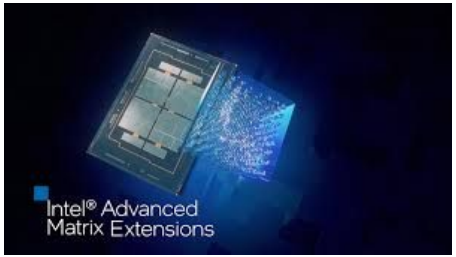
NVIDIA CUDA

AMD ROCm

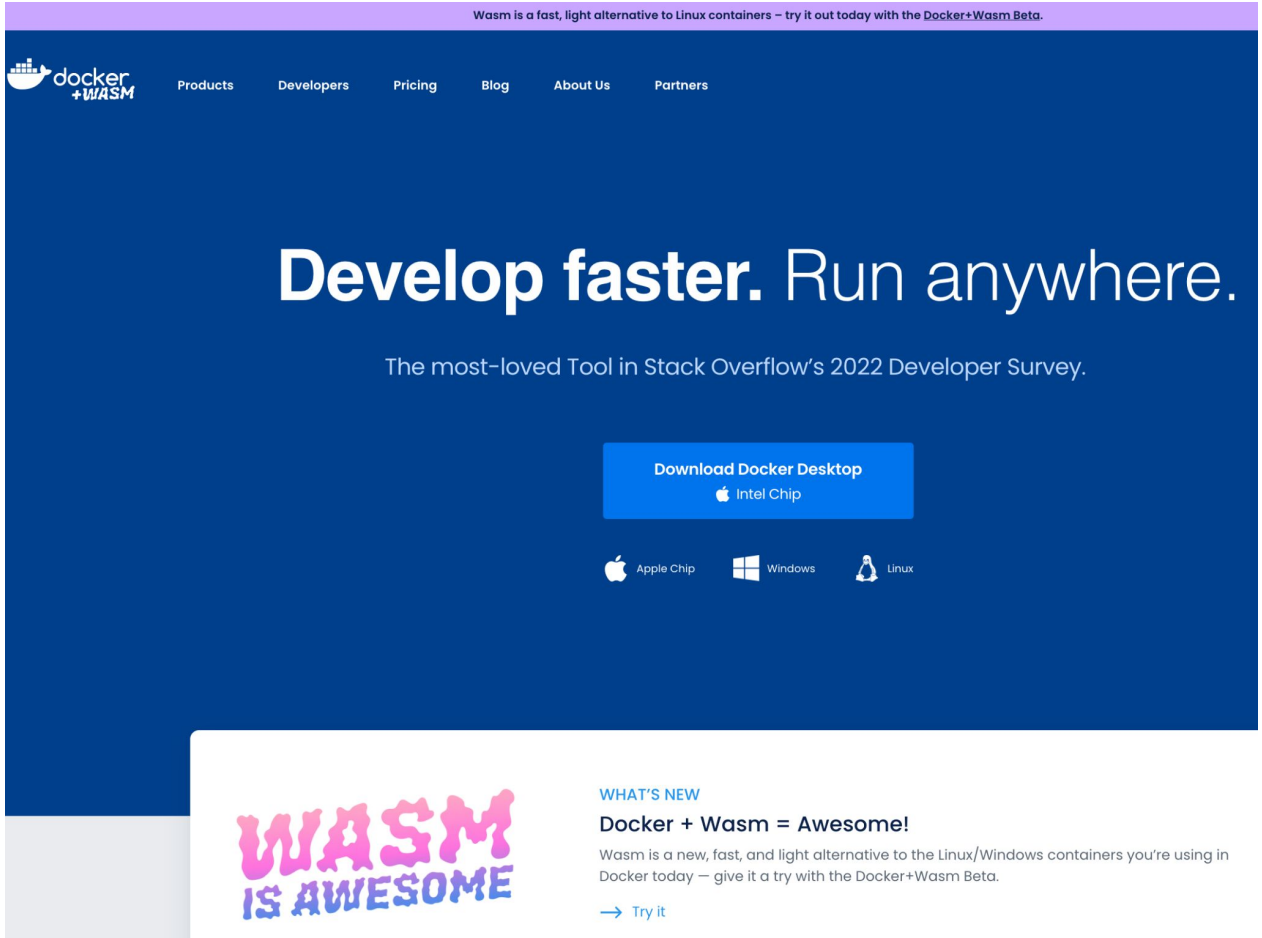
M4

intel Gaudi

groq®



What about Docker + Wasm?



Wasm as a container

- Benefits
  - Portable across CPUs, GPUs, NPUs etc.
  - Supported in Docker (runwasi), Podman (crun), OpenShift, Kubernetes etc.
  - Use OCI containers to hold compiled Wasm apps together with model files and config files
- Disadvantages
  - The entire app must be compiled to Wasm
  - Not an OS environment

https://gist.github.com/hydai/cd9ac23283c29ba86b4ab2d1e26cf6d2

One more thing: WebGPU

- Already a W3C standard
- An abstraction that proven works
  - Already available in most browsers across all consumer GPUs
  - Already have an ecosystem of apps and devs for browser apps
- Not just for the browser
  - Has standard C definitions to use outside of the browser!
  - In fact, in the browser, it is also common to compile C-based WebGPU app to Wasm
- High performance modern stack
  - Replaces WebGL

Docker is supporting WebGPU

Make WebGPU API available from inside the containers

```
FROM scratch

COPY tiny_en.cfg /tiny_en.cfg
COPY tiny_en.mpk /tiny_en.mpk
COPY tokenizer.json /tokenizer.json
COPY whisper-api-server.wasm /app.wasm

ENTRYPOINT [ "/app.wasm" ]
```

Docker Desktop Preview with WebGPU example

https://github.com/LlamaEdge/LlamaEdge/blob/main/docker/webgpu.md

Thank you

Learn more:
https://github.com/WasmEdge/WasmEdge