docker

Choosing the Smallest LLM That Won’t Completely Fail You

ContainerDays London ‘26

$> *what are we going to see today?*

01. SLMs

02. Run local models (with Open Source!)

03. Benchmark models (with Open Source!)

$> tell me everything you know about me

Manuel de la Peña

Staff Software Engineer

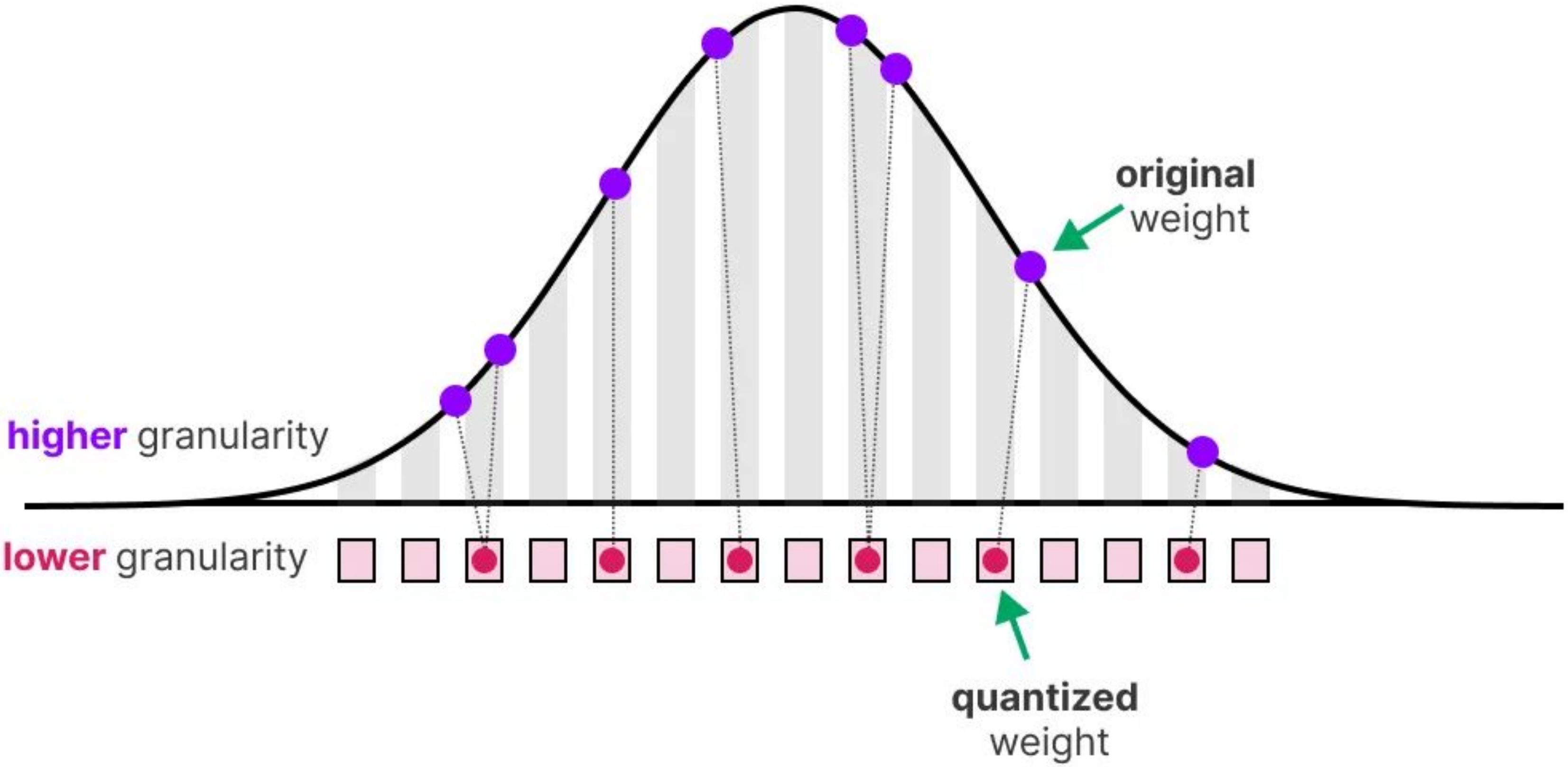
Docker

- 2022 → AtomicJar, acquired by Docker
- 2019 → Elastic: Eng. Productivity (Observability)
- 2011 → Liferay: Core Engineer > QA Tech Lead (Cloud)
- 2008 → Consultant
- 2005 → Public Admin in Spain
- Hitting keyboards since 1994

$> what are Small Language Models (SLMs)?

- ➔ Mostly Open Source models
- ➔ They can fit in your host's GPUs
    - ◆ 2-8 Gb of disk (and GPUs)
- ➔ Not producing as rich responses as the Big Players' models
- ➔ Ideal for experimentation, or for specialised tasks
- ➔ Multiple flavours (by **quantization**)
    - ◆ Reducing how the weights are stored in floating point numbers
        - FP32: 100% of the size
        - FP16: 50% of the size

$> draw a graph representing quantization



$> what are the trade-offs for using quantization?

Quantization comes with trade-offs:

- Slightly worse reasoning
- More hallucinations at higher temperatures
- Sensitive tasks (math, code) degrade sooner

$> Reality is ...

You may still use the LARGE models

- Way RICHER responses and outputs

Would you use the production models to test every change while building the app?

- Local development
- CI

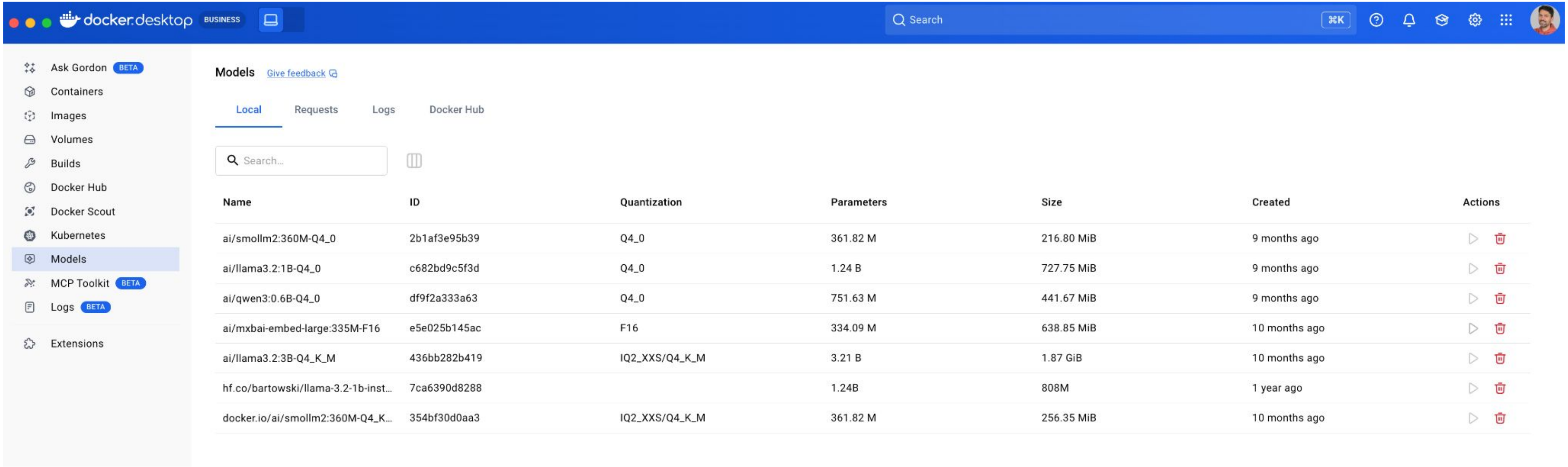
I DON'T ALWAYS TEST MY CODE
BUT WHEN I DO, I DO IT IN PRODUCTION
quickmeme.com

$> /stats

THISISELLIZ

$> how can I run models locally with Open Source?

- ➔ Llama.cpp, [ardanlabs/kronk](#) (Go library), [Ollama](#)...
- ➔ [Docker Model Runner](#) (Linux, MacOS and Windows)



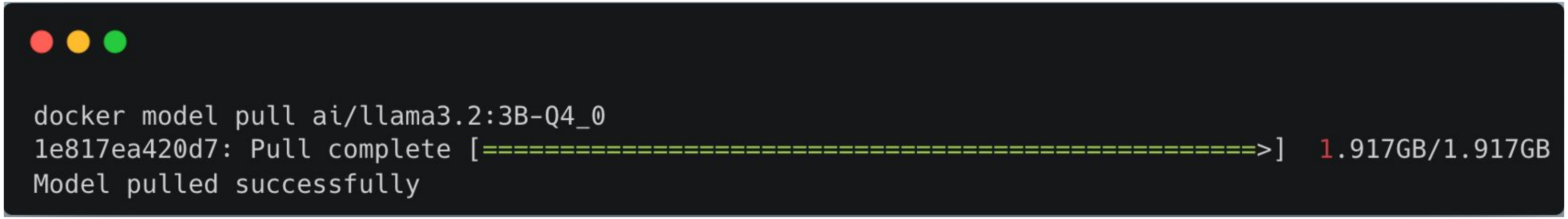
Name	ID	Quantization	Parameters	Size	Created	Actions
ai/smollm2:360M-Q4_0	2b1af3e95b39	Q4_0	361.82 M	216.80 MiB	9 months ago	▷ 🗑
ai/llama3.2:1B-Q4_0	c682bd9c5f3d	Q4_0	1.24 B	727.75 MiB	9 months ago	▷ 🗑
ai/qwen3:0.6B-Q4_0	df9f2a333a63	Q4_0	751.63 M	441.67 MiB	9 months ago	▷ 🗑
ai/mxbai-embed-large:335M-F16	e5e025b145ac	F16	334.09 M	638.85 MiB	10 months ago	▷ 🗑
ai/llama3.2:3B-Q4_K_M	436bb282b419	IQ2_XXS/Q4_K_M	3.21 B	1.87 GiB	10 months ago	▷ 🗑
hf.co/bartowski/llama-3.2-1b-inst...	7ca6390d8288		1.24B	808M	1 year ago	▷ 🗑
docker.io/ai/smollm2:360M-Q4_K...	354bf30d0aa3	IQ2_XXS/Q4_K_M	361.82 M	256.35 MiB	10 months ago	▷ 🗑

$> how and where can I get the models?

➔ Docker Hub: AI Models

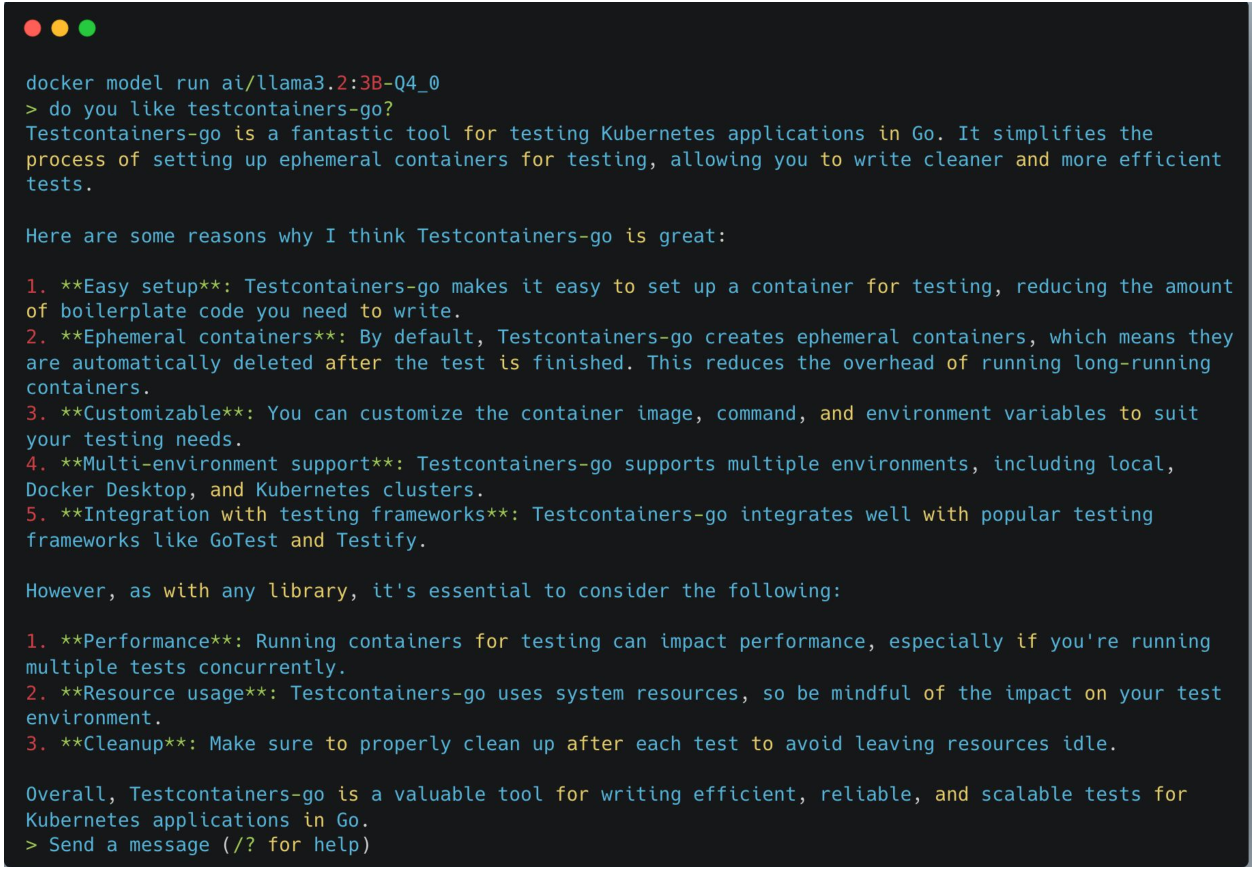
- ◆ Models as OCI artifacts → **docker model pull ai/llama3.2:3B-Q4_0**
- ◆ TIP: to download the recommended model → **docker model pull gpt-oss**

➔ Hugging Face



```
docker model pull ai/llama3.2:3B-Q4_0
1e817ea420d7: Pull complete [==================================================>]  1.917GB/1.917GB
Model pulled successfully
```

$> can I run models with Docker Model Runner?



```
docker model run ai/llama3.2:3B-Q4_0
> do you like testcontainers-go?
Testcontainers-go is a fantastic tool for testing Kubernetes applications in Go. It simplifies the
process of setting up ephemeral containers for testing, allowing you to write cleaner and more efficient
tests.

Here are some reasons why I think Testcontainers-go is great:

1. **Easy setup**: Testcontainers-go makes it easy to set up a container for testing, reducing the amount
of boilerplate code you need to write.
2. **Ephemeral containers**: By default, Testcontainers-go creates ephemeral containers, which means they
are automatically deleted after the test is finished. This reduces the overhead of running long-running
containers.
3. **Customizable**: You can customize the container image, command, and environment variables to suit
your testing needs.
4. **Multi-environment support**: Testcontainers-go supports multiple environments, including local,
Docker Desktop, and Kubernetes clusters.
5. **Integration with testing frameworks**: Testcontainers-go integrates well with popular testing
frameworks like GoTest and Testify.

However, as with any library, it's essential to consider the following:

1. **Performance**: Running containers for testing can impact performance, especially if you're running
multiple tests concurrently.
2. **Resource usage**: Testcontainers-go uses system resources, so be mindful of the impact on your test
environment.
3. **Cleanup**: Make sure to properly clean up after each test to avoid leaving resources idle.

Overall, Testcontainers-go is a valuable tool for writing efficient, reliable, and scalable tests for
Kubernetes applications in Go.
> Send a message (/? for help)
```

$> explain Docker Hub models names

Models in Docker Hub use the recommended quantization to not lose accuracy and reduce size.

I.e. llama3.2:3B-Q4_K_M

- ➔ Number of **parameters** (3B/3b for 3 billions)
- ➔ **Quantization** (Q4/q4 for 4-bit quantization → using 4 bits instead of 32)
- ➔ **K-quantization** (from llama.cpp) uses groups quantization for better accuracy
- ➔ **Medium variant of K**, balanced choice between speed, memory and accuracy

- ➔ So "**q4_k_m**" is usually a sweet spot for local and edge use.

Benchmark models with Open Source

$> what pieces do you find in a regular AI software?

When building AI software, we usually need to define the following:

- ➔ System prompts for the models to define initial behaviors
- ➔ Model parameters (Temperature, Top_P, Top_K...)
- ➔ The model itself (Reasoning capabilities, Using Tools...)
- ➔ Response formats (Natural language, JSON, ...)

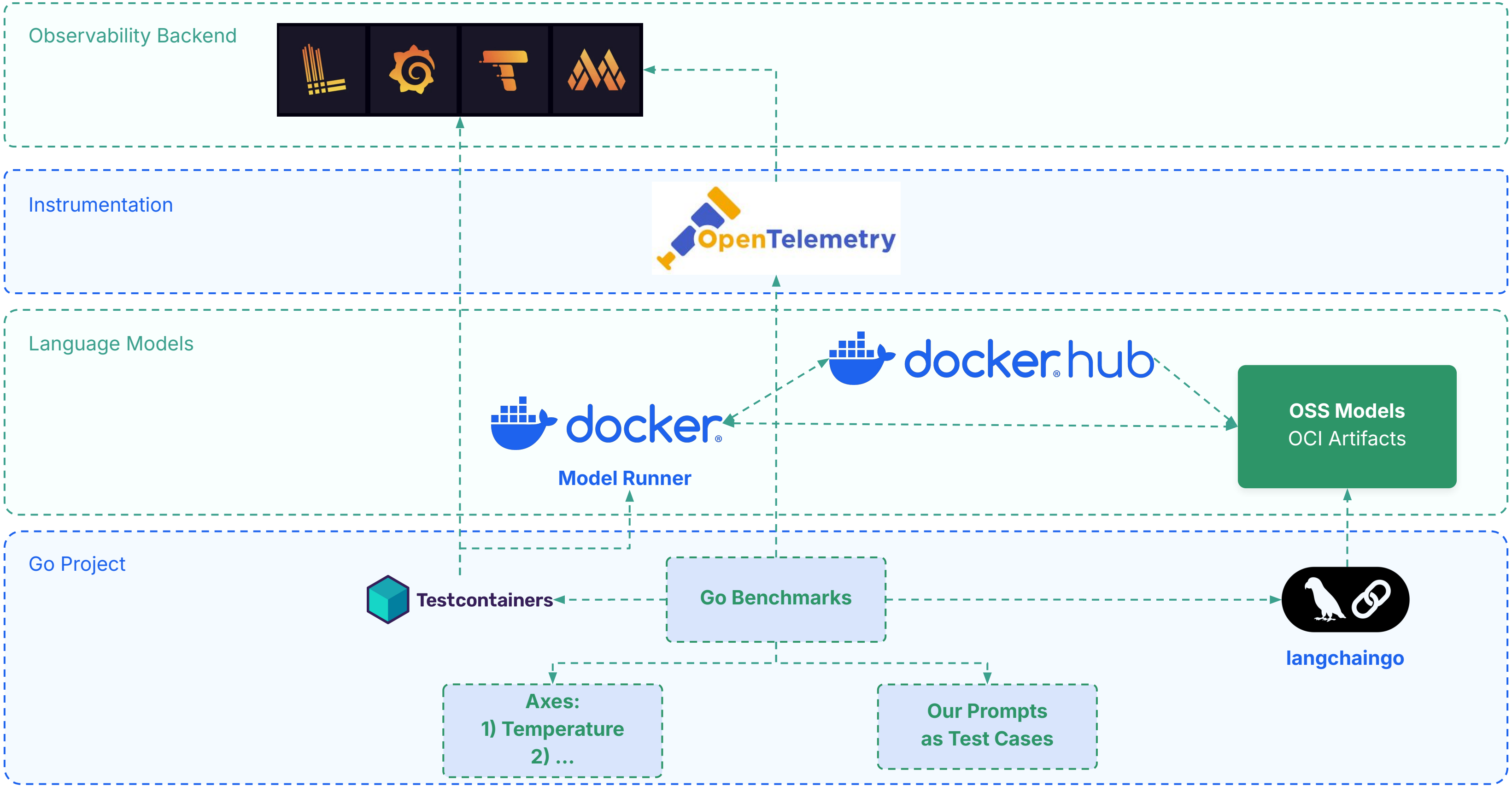
How do you know that your choices are the best for your use case?

- ◆ MMLU benchmark (Massive Multitask Language Understanding)
- ◆ Benchmarks by third-parties

$> list OSS-only tools I use to benchmark models

- Golang Benchmarks from the testing package
- Docker Model Runner (DMR) to load the OSS models
    - gpt-5.1
    - ai/llama3.2:1B-Q4_0
    - ai/llama3.2:3B-Q4_K_M
    - ai/qwen3:0.6B-Q4_0
    - hf.co/bartowski/Llama-3.2-1B-Instruct-GGUF
- OpenTelemetry (OTEL) for contributing metrics, traces and logs
- Grafana LGTM as observability backend
- Langchain Go for talking to the models in Go
- Testcontainers Go for starting containers (using predefined modules!)
    - Grafana, Python (for code execution)
    - Socat to talk to DMR through model-runner.docker.internal:80

$> draw an architectural diagram



Observability Backend

Instrumentation

Language Models

OSS Models
OCI Artifacts

Model Runner

Go Project

Testcontainers

Go Benchmarks

langchaingo

Axes:
1) Temperature
2) …

Our Prompts
as Test Cases

$> design the benchmarks (8×5×5=200 scenarios)

- Axe-0: 8 task types
  - **Code Explanation** of a function implementing Fibonacci.
  - **Mathematical operations** (what is the sum of all numbers between 1 and 100, both inclusive?=)
  - **Creative Writing**: Write a hilarious joke about the Fibonacci sequence.
  - **Factual Question**: What was the significance of Toledo, Spain during the medieval period, particularly regarding the translation movement?
  - **Code Generation**: Write a Go function that calculates the Fibonacci sequence using recursion.
  - **Calculator Reasoning**: Break down complex arithmetic into multiple calculator calls (
  - **Code Validation**: Generate Python code for Fibonacci, and execute it via a code executor tool
  - **Api Data Retrieval**: Use Fetch tool to get GitHub repository data and summarizes the key details
- Axe-1: 5 Temperatures → 0.1, 0.3, 0.5, 0.7, 0.9
- Axe-2: 5 OSS Models

$> what metrics could we collect from a model?

- Metrics:
  - Latency Percentiles: overall response time
  - Latency Histogram: response time distribution
  - Prompt Evaluation Time: Time to first token (TTFT)
  - Tokens per Operation: Token usage and verbosity
  - Tokens per Second: Generation throughput
  - Success Rate: Model reliability
  - GPU Utilization: Hardware efficiency
  - GPU Memory: Consumption
  - Score per Operation: Quality metrics provided by an Evaluator agent

docker

Code Demo

docker

GitHub repository

$> summarise what we have learn today

01. OSS is cool

02. SLM models and quantization

03. Using OSS tools to benchmark models

04. Plot the benchmarks into an OTEL backend

$> *You are absolutely cool!*

Thanks

ContainerDays London ‘26

Annexes

Massively Large models
Vs
Open Source models

(Massively) Large Language Models

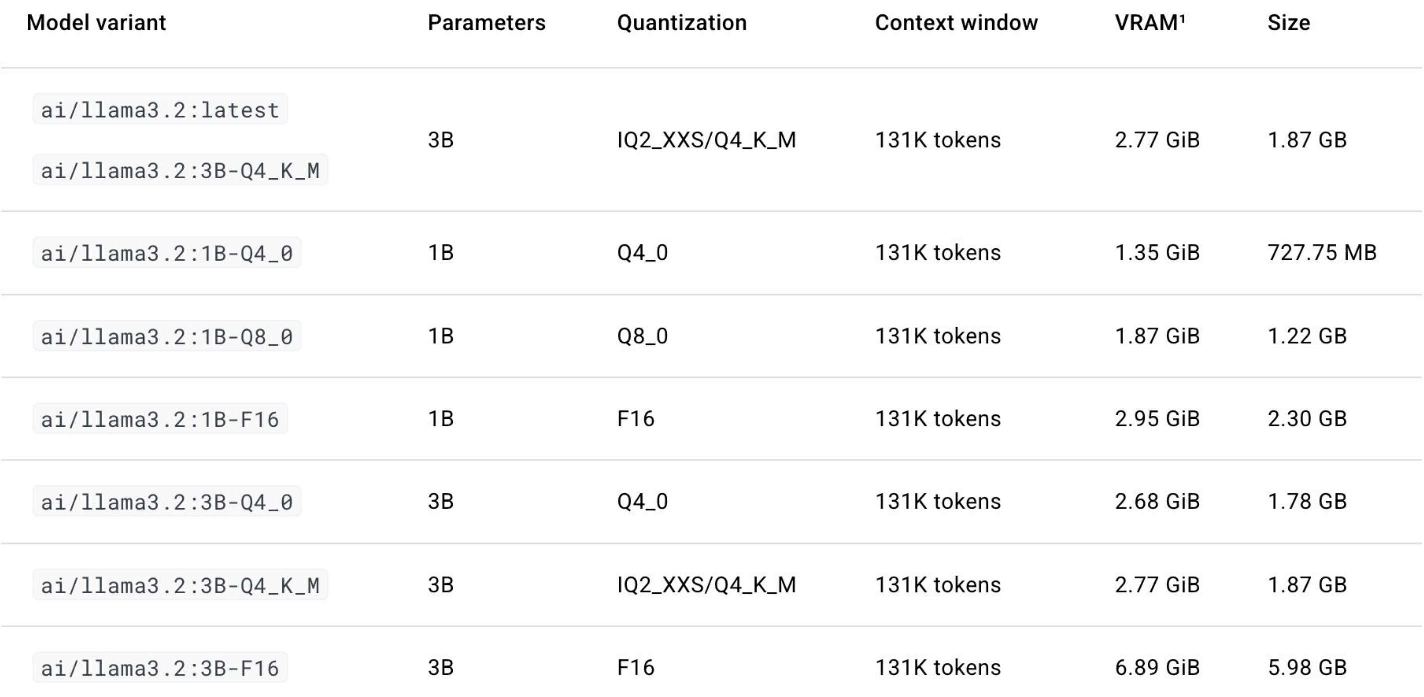
Big players have huge models with a humongous number of parameters.

Real numbers are not disclosed

- ➔ OpenAI
  - GPT-5.2 → trillions?
  - GPT 4o → 100 billions?
- ➔ Anthropic:
  - Claude 4 Sonnet → 300-500 billions?
  - Opus → 1-2 trillions?
- ➔ Google:
  - Gemini 3 → 1-3 trillions?

Open Source Language Models

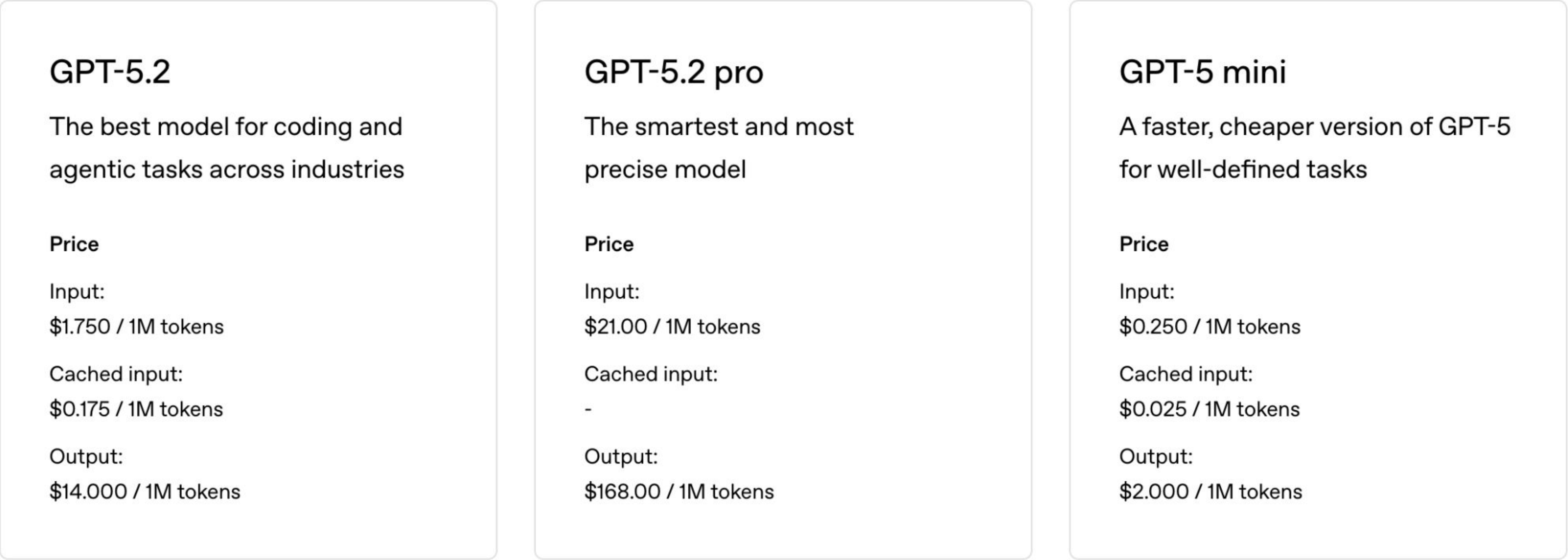
Ideal to run locally or in your own infrastructure, ideally on GPUs



Model variant	Parameters	Quantization	Context window	VRAM¹	Size
`ai/llama3.2:latest` `ai/llama3.2:3B-Q4_K_M`	3B	IQ2_XXS/Q4_K_M	131K tokens	2.77 GiB	1.87 GB
`ai/llama3.2:1B-Q4_0`	1B	Q4_0	131K tokens	1.35 GiB	727.75 MB
`ai/llama3.2:1B-Q8_0`	1B	Q8_0	131K tokens	1.87 GiB	1.22 GB
`ai/llama3.2:1B-F16`	1B	F16	131K tokens	2.95 GiB	2.30 GB
`ai/llama3.2:3B-Q4_0`	3B	Q4_0	131K tokens	2.68 GiB	1.78 GB
`ai/llama3.2:3B-Q4_K_M`	3B	IQ2_XXS/Q4_K_M	131K tokens	2.77 GiB	1.87 GB
`ai/llama3.2:3B-F16`	3B	F16	131K tokens	6.89 GiB	5.98 GB

… AND size matters!

The bigger the model, the more expensive using it



GPT-5.2

The best model for coding and agentic tasks across industries

Price

Input:
$1.750 / 1M tokens

Cached input:
$0.175 / 1M tokens

Output:
$14.000 / 1M tokens

GPT-5.2 pro

The smartest and most precise model

Price

Input:
$21.00 / 1M tokens

Cached input:
-

Output:
$168.00 / 1M tokens

GPT-5 mini

A faster, cheaper version of GPT-5 for well-defined tasks

Price

Input:
$0.250 / 1M tokens

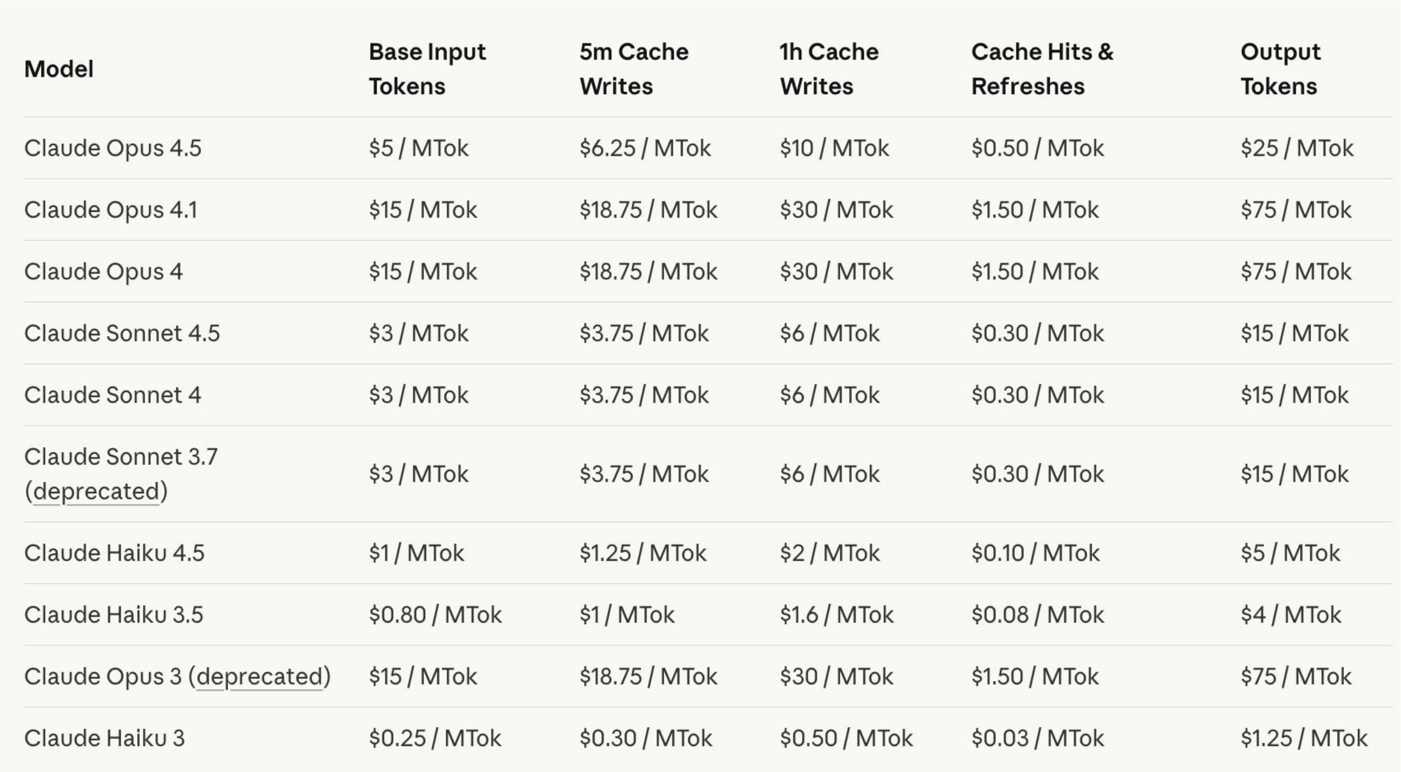
Cached input:
$0.025 / 1M tokens

Output:
$2.000 / 1M tokens

Source: https://openai.com/api/pricing/

… AND size matters!

The bigger the model, the more expensive using it

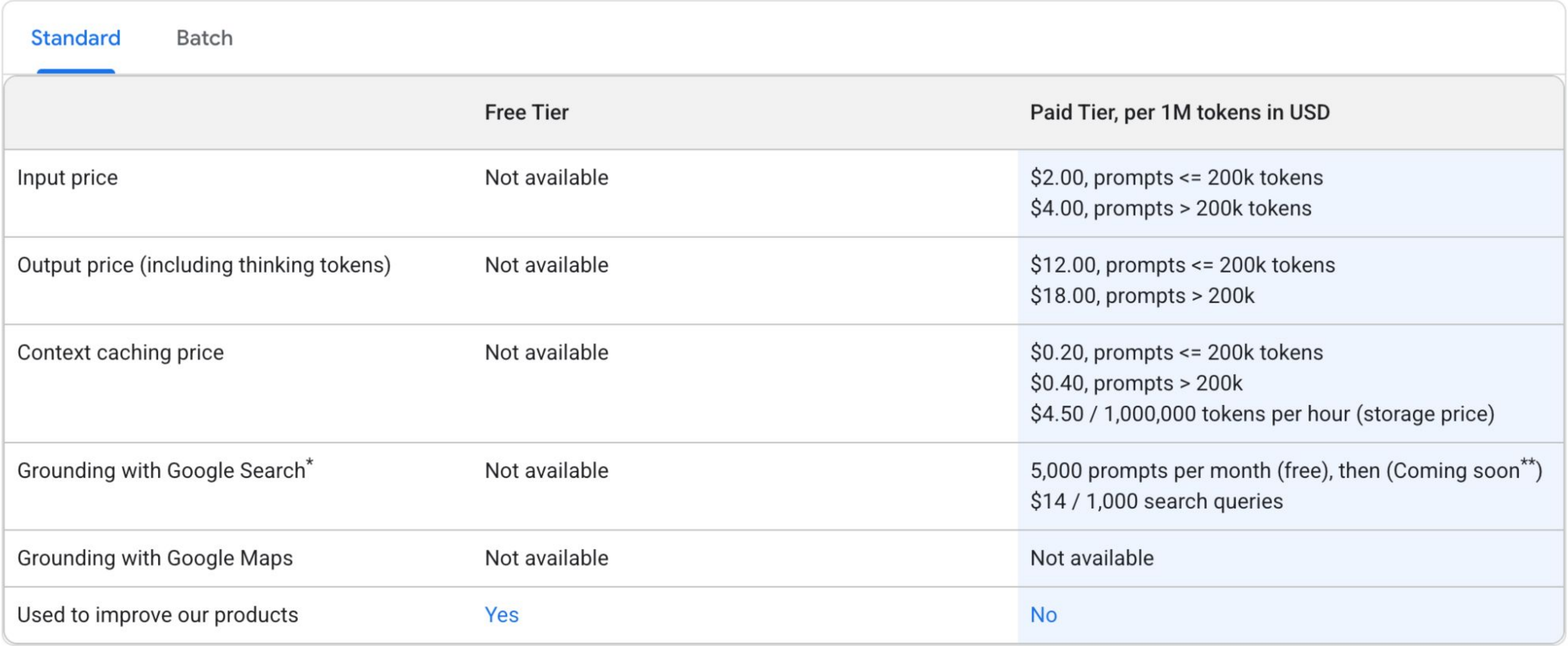


Model	Base Input Tokens	5m Cache Writes	1h Cache Writes	Cache Hits & Refreshes	Output Tokens
Claude Opus 4.5	$5 / MTok	$6.25 / MTok	$10 / MTok	$0.50 / MTok	$25 / MTok
Claude Opus 4.1	$15 / MTok	$18.75 / MTok	$30 / MTok	$1.50 / MTok	$75 / MTok
Claude Opus 4	$15 / MTok	$18.75 / MTok	$30 / MTok	$1.50 / MTok	$75 / MTok
Claude Sonnet 4.5	$3 / MTok	$3.75 / MTok	$6 / MTok	$0.30 / MTok	$15 / MTok
Claude Sonnet 4	$3 / MTok	$3.75 / MTok	$6 / MTok	$0.30 / MTok	$15 / MTok
Claude Sonnet 3.7 (deprecated)	$3 / MTok	$3.75 / MTok	$6 / MTok	$0.30 / MTok	$15 / MTok
Claude Haiku 4.5	$1 / MTok	$1.25 / MTok	$2 / MTok	$0.10 / MTok	$5 / MTok
Claude Haiku 3.5	$0.80 / MTok	$1 / MTok	$1.6 / MTok	$0.08 / MTok	$4 / MTok
Claude Opus 3 (deprecated)	$15 / MTok	$18.75 / MTok	$30 / MTok	$1.50 / MTok	$75 / MTok
Claude Haiku 3	$0.25 / MTok	$0.30 / MTok	$0.50 / MTok	$0.03 / MTok	$1.25 / MTok

Source: https://platform.claude.com/docs/en/about-claude/pricing

… AND size matters!

The bigger the model, the more expensive using it



	Free Tier	Paid Tier, per 1M tokens in USD
Input price	Not available	$2.00, prompts <= 200k tokens $4.00, prompts > 200k tokens
Output price (including thinking tokens)	Not available	$12.00, prompts <= 200k tokens $18.00, prompts > 200k
Context caching price	Not available	$0.20, prompts <= 200k tokens $0.40, prompts > 200k $4.50 / 1,000,000 tokens per hour (storage price)
Grounding with Google Search[*]	Not available	5,000 prompts per month (free), then (Coming soon[**]) $14 / 1,000 search queries
Grounding with Google Maps	Not available	Not available
Used to improve our products	Yes	No

Source: https://ai.google.dev/gemini-api/docs/pricing