LLM Deployment:
Tips, Tricks & Techniques

June 2025

DOUBLEWORD

What will we learn?

- The real problem with API-based AI
- What self-hosting actually means
- When it’s the right choice (and when it’s not)
- What it really takes to make it work

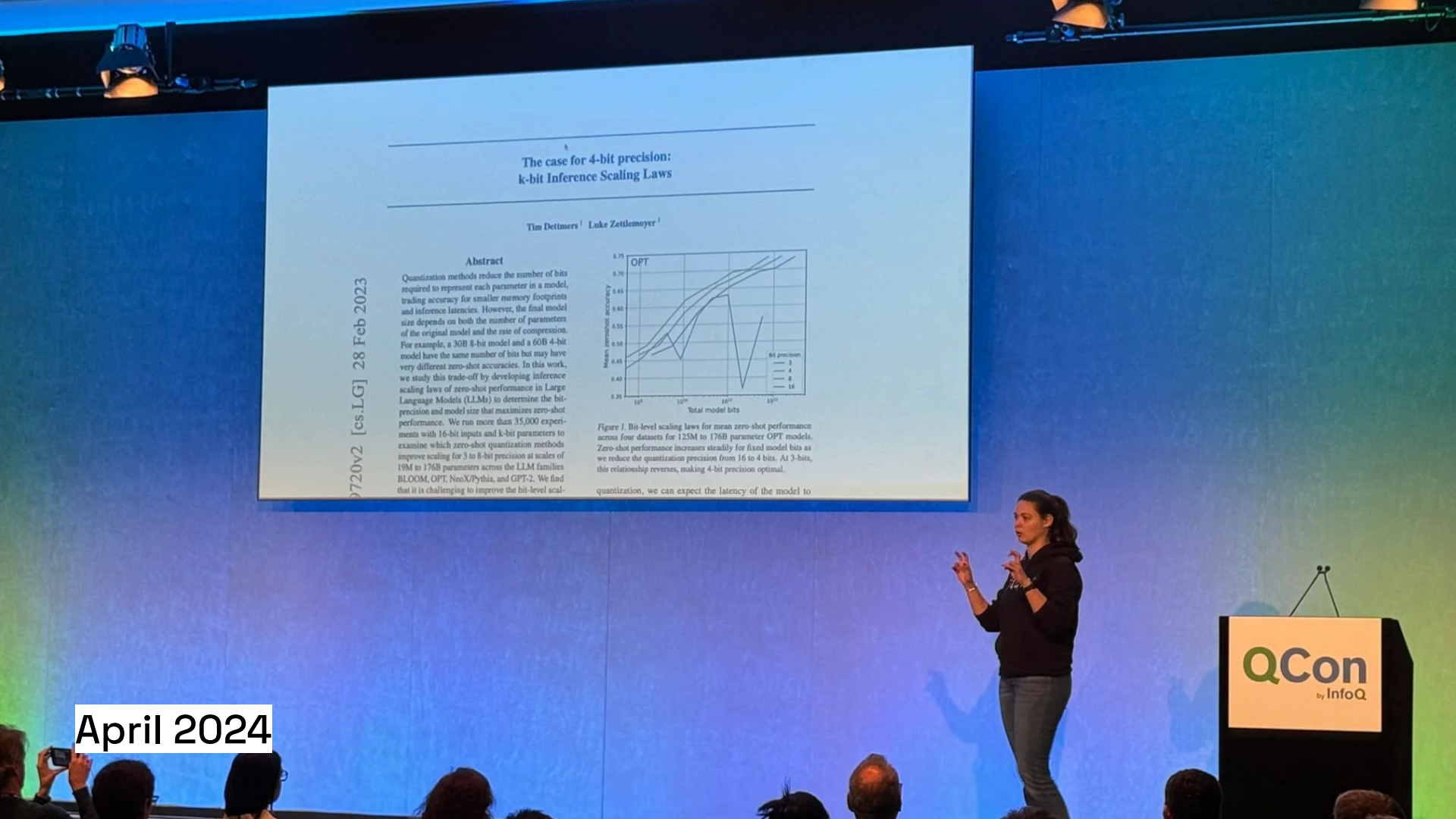
But first... Hi!

- Meryem Arik - CEO & Co-founder @Doubleword
- About Doubleword:
  - We’re a self-hosted inference platform purpose-built for enterprises
  - Specializing in on-premise and VPC AI Deployments
  - Deep expertise in inference and AI model deployments
  - Our whole founding team used to be physicists once upon a time!
- (Doubleword used to be TitanML)

Interested in self-hosting or AI inference at scale? Let’s chat after!

Chapter 1

A brief history of this talk


The case for 4-bit precision:
k-bit Inference Scaling Laws
Tim Dettmers [1] Luke Zettlemoyer [1]
Abstract
Quantization methods reduce the number of bits required to represent each parameter in a model, trading accuracy for smaller memory footprints and inference latencies. However, the final model size depends on both the number of parameters of the original model and the rate of compression. For example, a 30B 8-bit model and a 60B 4-bit model have the same number of bits but may have very different zero-shot accuracies. In this work, we study this trade-off by developing inference scaling laws of zero-shot performance in Large Language Models (LLMs) to determine the bit-precision and model size that maximizes zero-shot performance. We run more than 35,000 experiments with 16-bit inputs and k-bit parameters to examine which zero-shot quantization methods improve scaling for 3 to 8-bit precision at scales of 19M to 176B parameters across the LLM families BLOOM, OPT, NeoX/Pythia, and GPT-2. We find that it is challenging to improve the bit-level scal-
9720v2 [cs.LG] 28 Feb 2023
OPT
Mean zerosshot accuracy
Total model bits
Bit precision
3
4
8
16
Figure 1. Bit-level scaling laws for mean zero-shot performance across four datasets for 125M to 176B parameter OPT models. Zero-shot performance increases steadily for fixed model bits as we reduce the quantization precision from 16 to 4 bits. At 3-bits, this relationship reverses, making 4-bit precision optimal.
quantization, we can expect the latency of the model to
QCon
by InfoQ

April 2024

November 2024

MLOPS LDN - APRIL TALKS

Navigating LLM Deployment: Tips, Tricks, and Techniques

Meryem Arik
CEO & Co-founder, TitanML

April 2025

vents &
webinars

June 2025

London • June 16 & 17, 2025

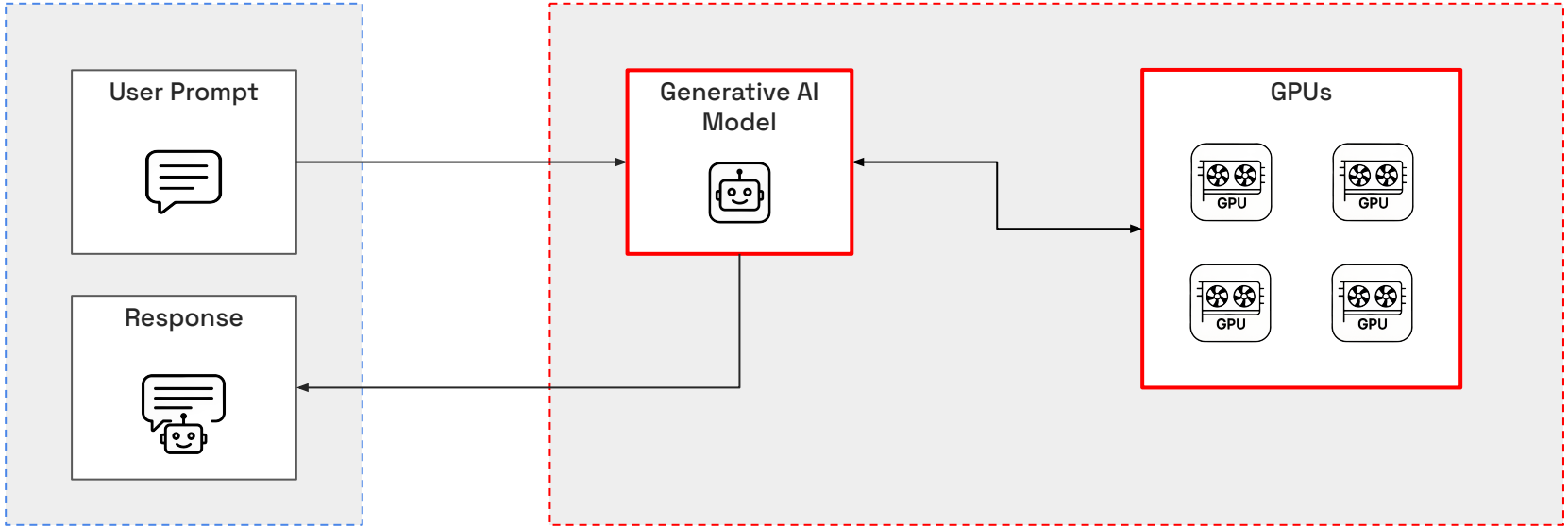
The Festival of Engineering Leadership

by LeadDev

What is self-hosting?

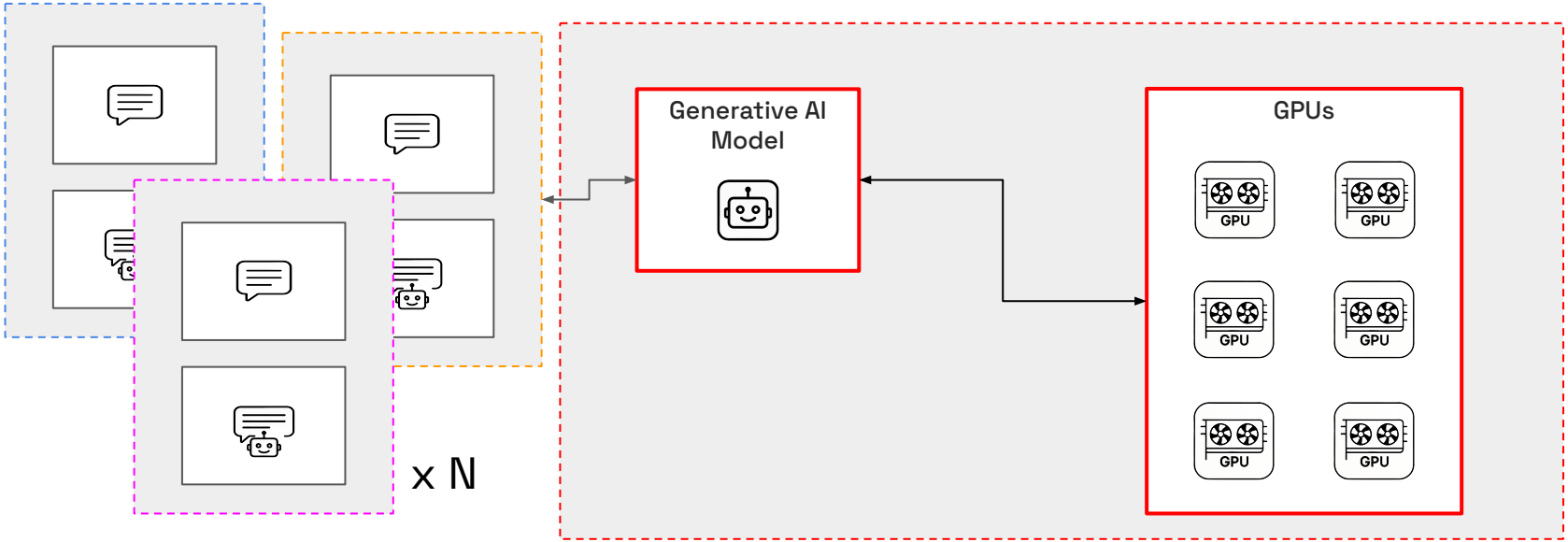
Hosted AI APIs vs Self-Hosting

AI APIs: Provider-Owned Models and Infrastructure



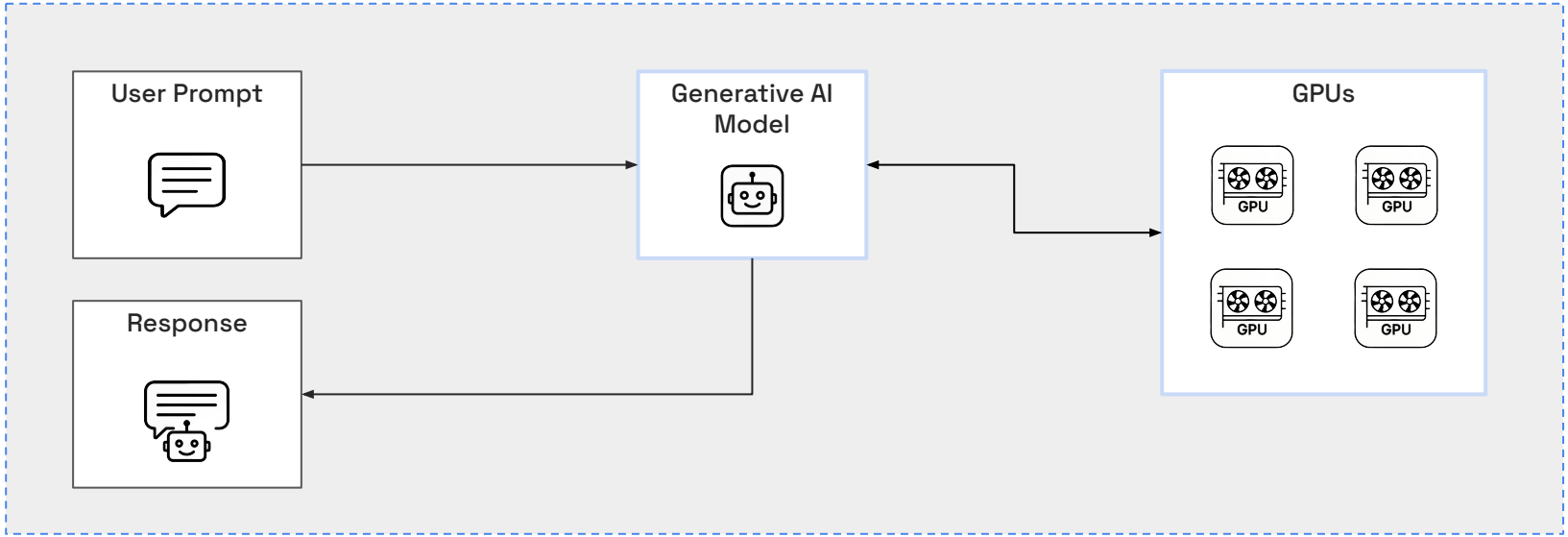
OpenAI ANTHROP\C Amazon Bedrock vertex.ai Grok Fireworks AI cohere

AI APIs: One Model, Many Customers



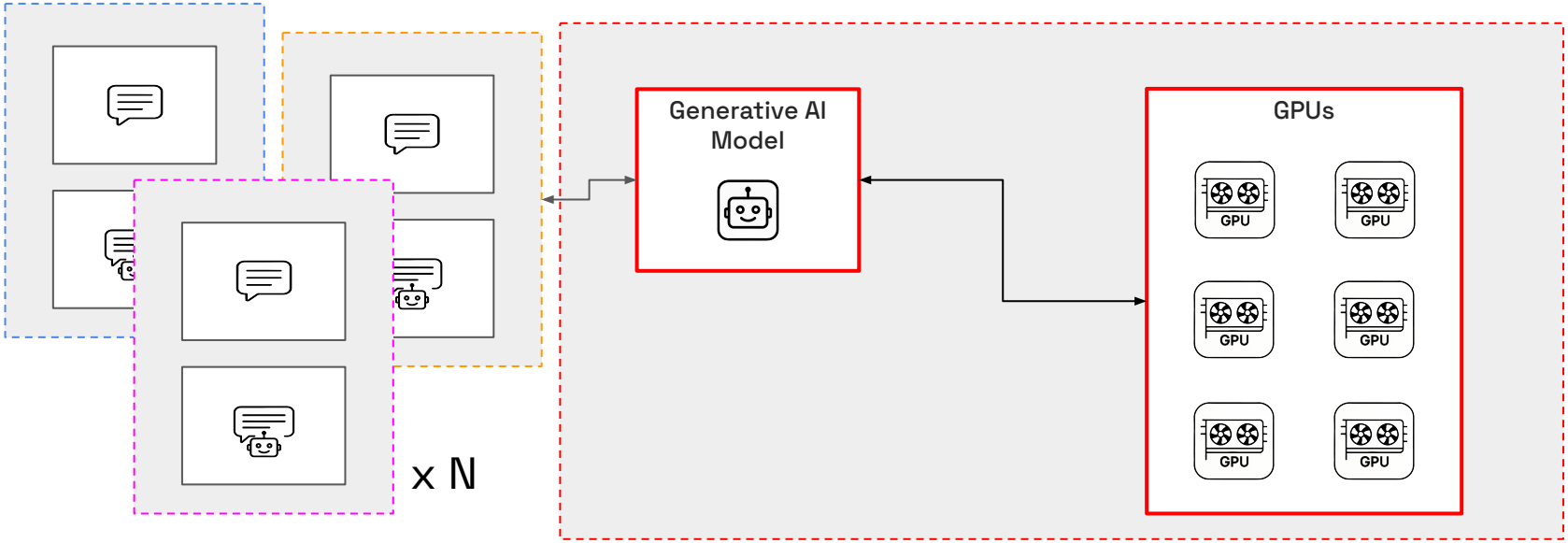
OpenAI ANTHROP\C Amazon Bedrock vertex.ai Grok Fireworks AI cohere

Self-Hosting: Your Models, Your Infra, Your Control



When would I want to self-host?

AI APIs: One Model, Many Customers



Hosted AI APIs: The Impact

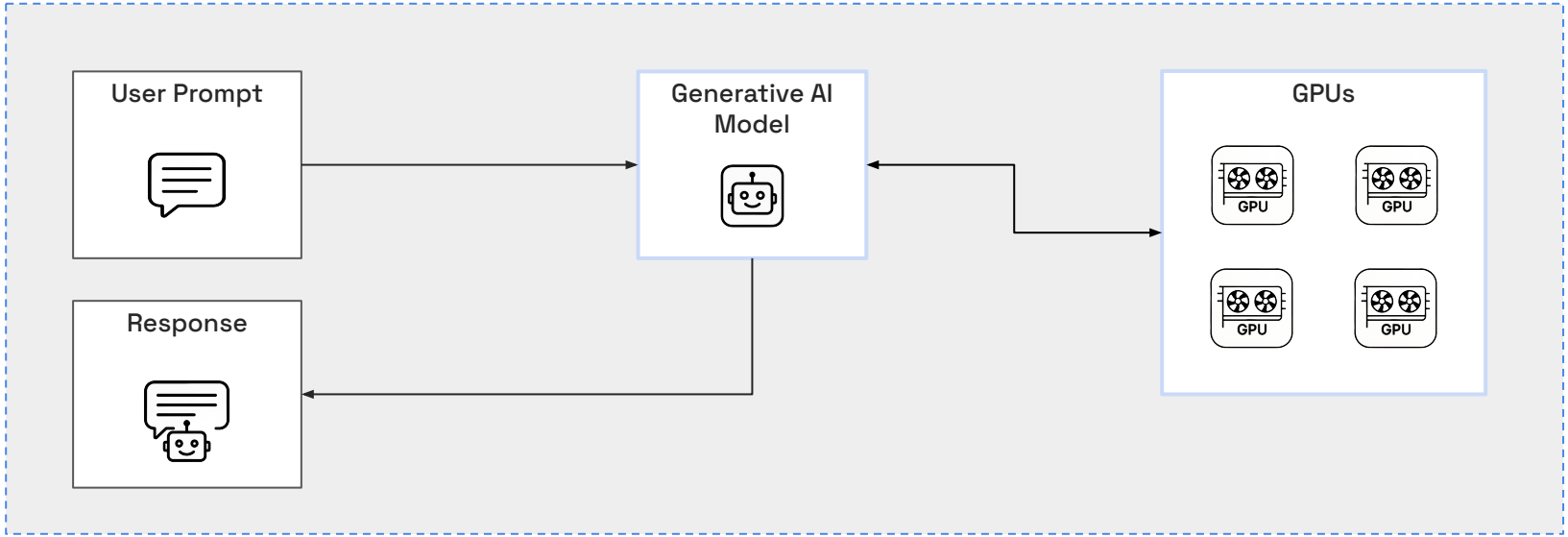
Pros:

- Easy to set up - someone else manages the infrastructure
- Cheaper at small scale

Cons:

- Sending data and IP to a 3rd party environment
- Models are not yours
- Scaling issues: Rate limits, high latency, poor throughput, expensive at scale
- Only access to generic one-size fits all models

Self-Hosting: Your Models, Your Infra, Your Control



Self-Hosted AI: The Impact

Pros:

- All data and IP stays in your secure compute environment
- You own the full application including the models
- No rate limits, opportunity for lower latency, higher throughput, cheaper at scale
- Access and deploy any AI model including specialised ones

Cons:

- Harder to set up
- You need to manage the infrastructure

Hosted AI APIs vs Self-Hosting

- Early and easy experimentation with public or non-sensitive data
- Application is not very latency / throughput sensitive

- Sensitive data or mission critical application
- Applications deployed at scale
- Domain specific application
- Multi-cloud / On-premise set up

What infrastructure do I need to think about when self-hosting?

DOUBLEWORD

Everything Needed to Self-Host your AI Inference

Core Components

- Hardware
- Models
- Model repo
- Inference engine

Orchestration

- Self-healing
- CI/CD
- LLM-aware Autoscaler
- LLM-aware Scheduler

Optimizations

- LLM-Aware Load Balancer
- Multi-Level Caching
- Quantization
- Inference Engine Configuration
- GPU Sharing and Sharding
- Serverless PEFT / LoRA

API Access & Security

- API Gateways
- Authentication
- Authorization
- Role based access control

Management

- Front-end / UI
- Multi-cluster & multi-cloud management
- Model approval process
- Deployment & model management
- 3rd party integrations
- Chargeback system

Broad Model Support

- Multi-modal support
- Fine-tuned model support
- Wide architecture support
- Auxiliary model support

Observability

- Logging & monitoring
- Dashboards & Analytics
- Alerts
- Multi-deployment aggregation

Core Application Level Functionality

- Vector DBs
- Agents
- Document Intelligence
- Model evaluation
- Guardrails
- Multi-stage pipelines

Testing and Resilience

- Load Testing
- Scaling Behaviour

DOUBLEWORD

Everything Needed to Self-Host your AI Inference

Core Components

- Hardware
- Models
- Model repo
- Inference engine

Orchestration

- Self-healing
- CI/CD
- LLM-aware Autoscaler
- LLM-aware Scheduler

Optimizations

- LLM-Aware Load Balancer
- Multi-Level Caching
- Quantization
- Inference Engine Configuration
- GPU Sharing and Sharding
- Serverless PEFT / LoRA

API Access & Security

- API Gateways
- Authentication
- Authorization
- Role based access control

Management

- Front-end / UI
- Multi-cluster & multi-cloud management
- Model approval process
- Deployment & model management
- 3rd party integrations
- Chargeback system

Broad Model Support

- Multi-modal support
- Fine-tuned model support
- Wide architecture support
- Auxiliary model support

Observability

- Logging & monitoring
- Dashboards & Analytics
- Alerts
- Multi-deployment aggregation

Core Application Level Functionality

- Vector DBs
- Agents
- Document Intelligence
- Model evaluation
- Guardrails
- Multi-stage pipelines

Testing and Resilience

- Load Testing
- Scaling Behaviour

DOUBLEWORD

Everything Needed to Self-Host your AI Inference

Core Components

- Hardware
- Models
- Model repo
- Inference engine

Orchestration

- Self-healing
- CI/CD
- LLM-aware Autoscaler
- LLM-aware Scheduler

Optimizations

- LLM-Aware Load Balancer
- Multi-Level Caching
- Quantization
- Inference Engine Configuration
- GPU Sharing and Sharding
- Serverless PEFT / LoRA

API Access & Security

- API Gateways
- Authentication
- Authorization
- Role based access control

Management

- Front-end / UI
- Multi-cluster & multi-cloud management
- Model approval process
- Deployment & model management
- 3rd party integrations
- Chargeback system

Broad Model Support

- Multi-modal support
- Fine-tuned model support
- Wide architecture support
- Auxiliary model support

Observability

- Logging & monitoring
- Dashboards & Analytics
- Alerts
- Multi-deployment aggregation

Core Application Level Functionality

- Vector DBs
- Agents
- Document Intelligence
- Model evaluation
- Guardrails
- Multi-stage pipelines

Testing and Resilience

- Load Testing
- Scaling Behaviour

DOUBLEWORD

Everything Needed to Self-Host your AI Inference

Core Components

- Hardware
- Models
- Model repo
- Inference engine

Orchestration

- Self-healing
- CI/CD
- LLM-aware Autoscaler
- LLM-aware Scheduler

Optimizations

- LLM-Aware Load Balancer
- Multi-Level Caching
- Quantization
- Inference Engine Configuration
- GPU Sharing and Sharding
- Serverless PEFT / LoRA

API Access & Security

- API Gateways
- Authentication
- Authorization
- Role based access control

Management

- Front-end / UI
- Multi-cluster & multi-cloud management
- Model approval process
- Deployment & model management
- 3rd party integrations
- Chargeback system

Broad Model Support

- Multi-modal support
- Fine-tuned model support
- Wide architecture support
- Auxiliary model support

Observability

- Logging & monitoring
- Dashboards & Analytics
- Alerts
- Multi-deployment aggregation

Core Application Level Functionality

- Vector DBs
- Agents
- Document Intelligence
- Model evaluation
- Guardrails
- Multi-stage pipelines

Testing and Resilience

- Load Testing
- Scaling Behaviour

DOUBLEWORD

Everything Needed to Self-Host your AI Inference

Core Components

- Hardware
- Models
- Model repo
- Inference engine

Orchestration

- Self-healing
- CI/CD
- LLM-aware Autoscaler
- LLM-aware Scheduler

Optimizations

- LLM-Aware Load Balancer
- Multi-Level Caching
- Quantization
- Inference Engine Configuration
- GPU Sharing and Sharding
- Serverless PEFT / LoRA

API Access & Security

- API Gateways
- Authentication
- Authorization
- Role based access control

Management

- Front-end / UI
- Multi-cluster & multi-cloud management
- Model approval process
- Deployment & model management
- 3rd party integrations
- Chargeback system

Broad Model Support

- Multi-modal support
- Fine-tuned model support
- Wide architecture support
- Auxiliary model support

Observability

- Logging & monitoring
- Dashboards & Analytics
- Alerts
- Multi-deployment aggregation

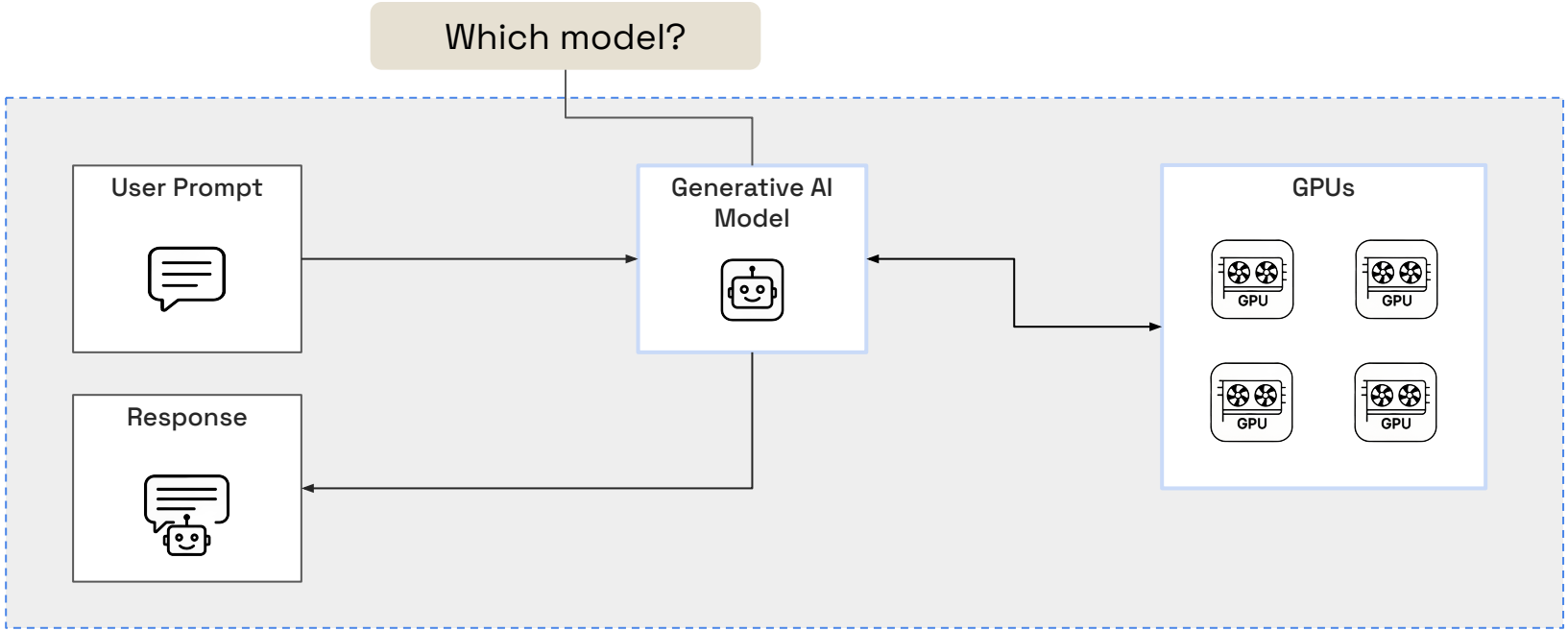
Core Application Level Functionality

- Vector DBs
- Agents
- Document Intelligence
- Model evaluation
- Guardrails
- Multi-stage pipelines

Testing and Resilience

- Load Testing
- Scaling Behaviour

Starting with the basics...



Which model should I pick?

1. Open-source model base [Llama, Mistral, Gemma]

   Why? So you can deploy it and own the deployment.

2. Which one?

   What size? ~~ What hardware do I have available?

   Don't forget quantized versions!

3. Is it widely supported by common infrastructure?
4. Create a shortlist & systematically test against your use case (With RAG & Prompt tuning)

Self-hosting Infrastructure

Overview | Text | WebDev | Vision | Text-to-Image | Search | Copilot

Task

Text Arena

View rankings across various LLMs on their versatility, linguistic precision, and cultural context across text.

Last Updated: Jun 16, 2025

Total Votes: 3,077,387

Total M: 250

Overall

Search by model name...

Open Source

Rank (UB)	Model	Score	95% CI (±)	Votes	Organization	License
149	tulu-2-dpo-70b	1169	+7/-8	6,658		mpAC...
11	qwen3-235b-a22b-no-thinking	1394	+6/-7	7,369	Alibaba	Apache 2.0
20	qwen3-235b-a22b	1367	+7/-6	11,240		
35	qwq-32b	1335	+5/-5	16,280	Alibaba	
37	qwen3-32b	1328	+8/-10	3,960	Alibaba	
69	mistral-small-3.1-24b-instruct-2503	1295	+10/-9	3,058	Mistral	
74	qwen3-30b-a3b	1295	+7/-12	4,159	Alibaba	Apache 2.0

Trustworthy org?

Model size

Model memory calcula

What if that doesn't work... Fine-tuning

When to Fine-Tune

You need **domain-specific tone**, structure, or language

RAG isn't enough (e.g. classification, function-calling)

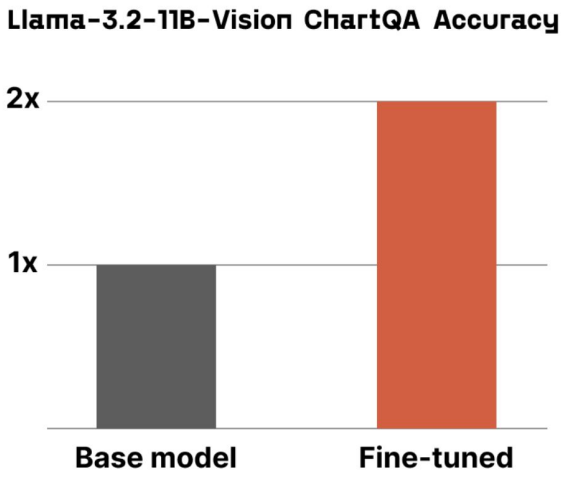
Model needs better performance on edge cases

Types of Fine-Tuning

Full fine-tuning – large gains, costly, rarely needed

LoRA / QLoRA – lightweight, efficient, production-ready

Instruction tuning – aligns behavior with task prompts



Source: predibase

Best Practices

Use high-quality, structured data, Start small (1–5k examples) before scaling, Track overfitting – test on unseen prompts, Always compare to RAG baseline first

How do I actually get this model running though?

An Inference Engine!!!

DOUBLEWORD

Everything Needed to Self-Host your AI Inference

Core Components

- Hardware
- Models
- Model repo
- Inference engine

Orchestration

- Self-healing
- CI/CD
- LLM-aware Autoscaler
- LLM-aware Scheduler

Optimizations

- LLM-Aware Load Balancer
- Multi-Level Caching
- Quantization
- Inference Engine Configuration
- GPU Sharing and Sharding
- Serverless PEFT / LoRA

API Access & Security

- API Gateways
- Authentication
- Authorization
- Role based access control

Management

- Front-end / UI
- Multi-cluster & multi-cloud management
- Model approval process
- Deployment & model management
- 3rd party integrations
- Chargeback system

Broad Model Support

- Multi-modal support
- Fine-tuned model support
- Wide architecture support
- Auxiliary model support

Observability

- Logging & monitoring
- Dashboards & Analytics
- Alerts
- Multi-deployment aggregation

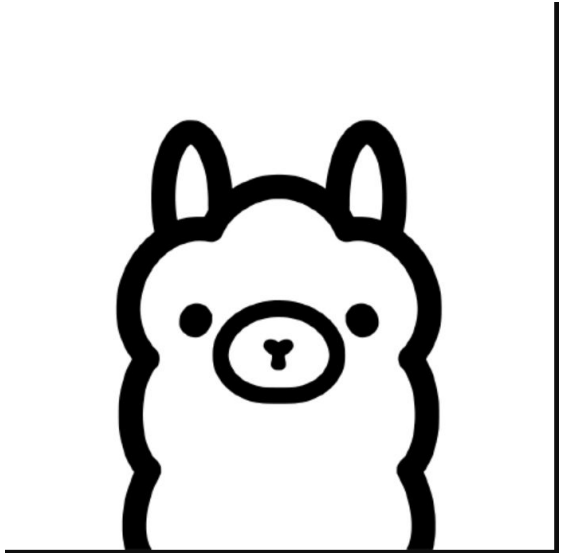
Core Application Level Functionality

- Vector DBs
- Agents
- Document Intelligence
- Model evaluation
- Guardrails
- Multi-stage pipelines

Testing and Resilience

- Load Testing
- Scaling Behaviour

How do I actually get this model running though?



Ollama - easy to use, non-production, experimentation, consumer hardware

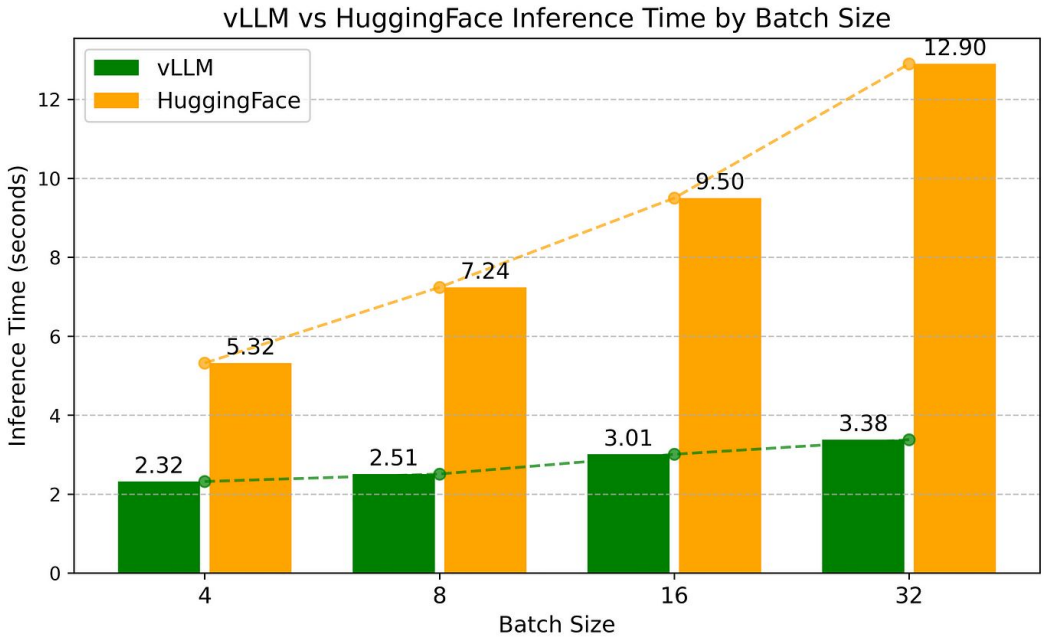
vLLM

SGL

DOUBLEWORD

Production ready, high performance inference engines

Inference Engine choice really matters

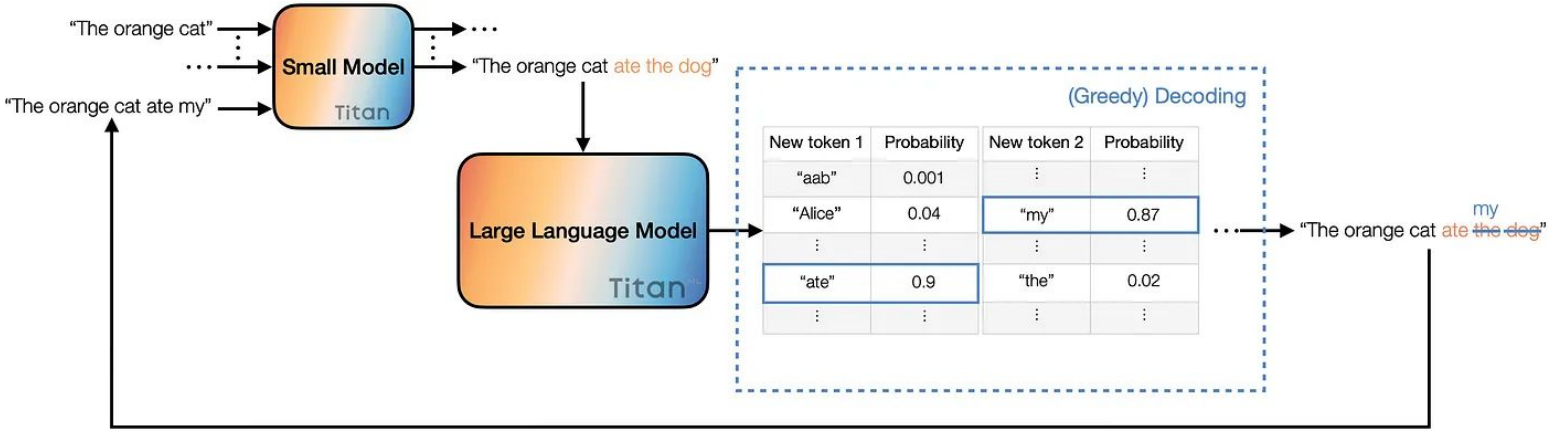


Source: https://medium.com/@alishafique3/vllm-vs-hugging-face-for-high-performance-offline-llm-inference-2d953b4fb3b4

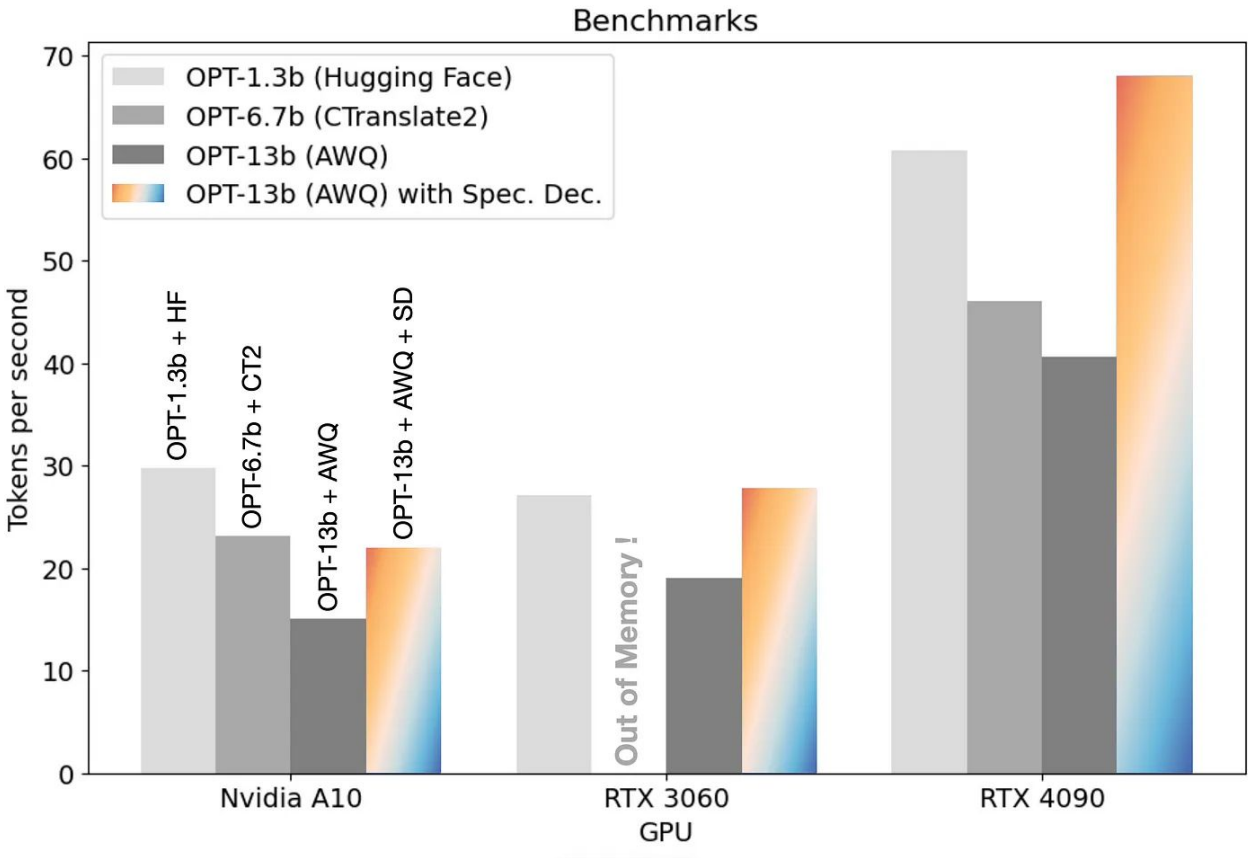
Inference Optimisation in Practice: Speculative Decoding

https://medium.com/@TitanML/in-the-fast-lane-speculative-decoding-10x-larger-model-no-extra-cost-f33ea39d065a

Inference Optimisation in Practice: Speculative Decoding



https://medium.com/@TitanML/in-the-fast-lane-speculative-decoding-10x-larger-model-no-extra-cost-f33ea39d065a


Benchmarks
OPT-1.3b (Hugging Face)
OPT-6.7b (CTranslate2)
OPT-13b (AWQ)
OPT-13b (AWQ) with Spec. Dec.
Tokens per second
0
10
20
30
40
50
60
70
OPT-1.3b + HF
OPT-6.7b + CT2
OPT-13b + AWQ
OPT-13b + AWQ + SD
Out of Memory !
Nvidia A10
RTX 3060
RTX 4090
GPU

Build the App, Not the Infra!

Work with state-of-the-art inference engines & trusted partners to optimize the inference for you.

DOUBLEWORD

Everything Needed to Self-Host your AI Inference

Core Components

- Hardware
- Models
- Model repo
- Inference engine

Orchestration

- Self-healing
- CI/CD
- LLM-aware Autoscaler
- LLM-aware Scheduler

Optimizations

- LLM-Aware Load Balancer
- Multi-Level Caching
- Quantization
- Inference Engine Configuration
- GPU Sharing and Sharding
- Serverless PEFT / LoRA

API Access & Security

- API Gateways
- Authentication
- Authorization
- Role based access control

Management

- Front-end / UI
- Multi-cluster & multi-cloud management
- Model approval process
- Deployment & model management
- 3rd party integrations
- Chargeback system

Broad Model Support

- Multi-modal support
- Fine-tuned model support
- Wide architecture support
- Auxiliary model support

Observability

- Logging & monitoring
- Dashboards & Analytics
- Alerts
- Multi-deployment aggregation

Core Application Level Functionality

- Vector DBs
- Agents
- Document Intelligence
- Model evaluation
- Guardrails
- Multi-stage pipelines

Testing and Resilience

- Load Testing
- Scaling Behaviour

What about everything else?

The devil is in the detail… here's what often gets missed:

- Logging, monitoring & alerting
  - Log: latency, token count, failures, context length
  - Monitor: GPU utilization, queue depth, cache hit rate
  - Alerting: latency spikes, OOMs, token overloads
  - Dashboards: per-model and per-tenant views
- Model approval process
  - Approved model list by provider
  - Internal model risk assessment
  - Update & rollback policies
- Chargeback system
  - Prompt + output tokens (and model used) per request
  - GPU time or compute cost per inference
  - Cache hit/miss rates (impacts actual cost)
  - User/team attribution
- Etc [Quota management, Model Caching, API rate limiting, Reliability SLAs]

DOUBLEWORD

Everything Needed to Self-Host your AI Inference

Core Components

- Hardware
- Models
- Model repo
- Inference engine

Orchestration

- Self-healing
- CI/CD
- LLM-aware Autoscaler
- LLM-aware Scheduler

Optimizations

- LLM-Aware Load Balancer
- Multi-Level Caching
- Quantization
- Inference Engine Configuration
- GPU Sharing and Sharding
- Serverless PEFT / LoRA

API Access & Security

- API Gateways
- Authentication
- Authorization
- Role based access control

Management

- Front-end / UI
- Multi-cluster & multi-cloud management
- Model approval process
- Deployment & model management
- 3rd party integrations
- Chargeback system

Broad Model Support

- Multi-modal support
- Fine-tuned model support
- Wide architecture support
- Auxiliary model support

Observability

- Logging & monitoring
- Dashboards & Analytics
- Alerts
- Multi-deployment aggregation

Core Application Level Functionality

- Vector DBs
- Agents
- Document Intelligence
- Model evaluation
- Guardrails
- Multi-stage pipelines

Testing and Resilience

- Load Testing
- Scaling Behaviour

What does that look like?

Meet Clara: CTO of MedSure Health

About her:

- Regulated environment
- Building internal AI assistant
- Initially wanted to use OpenAI, but cyber killed it and struggled with rate limits
- Moved to self-hosting

What she did right (and wrong)

- ❌ Took time to swap different models
- ✅ Picked Llama3 with quantization
- ✅ Used fine-tuning for performance
- ❌ Didn't use caching = high latency
- ✅ Fixed it with a proper inference engine + metrics

Takeaways

- ✅ Self-hosting it’s often necessary and preferable
- ✅ You need more than a model: infra, security, governance matter
- ✅ Don’t build everything from scratch—use the right tools and partners
- ✅ Start small, but design for scale from day one

What should you do next?

✅ **Decide if self-hosting is in your future**
Audit your AI roadmap: Will scale, latency, cost, or compliance force you off hosted APIs in the next 12–24 months?

✅ **Map your gaps**
What infra, team skills, or governance tooling do you not have yet? Don't wait until deployment day to find out.

✅ **Pick a first project**
Choose a use case that's internal, valuable, and not customer-facing. Perfect for testing infra, model configs, and workflows.

✅ **Don't go it alone**
Use existing inference engines, deployment frameworks, and vendor support. Build what's unique to you—buy the rest.

Let’s chat!

Want to self-host AI? Let’s talk.

meryem@doubleword.ai OR catch me in person after!