LLM Observability: Unlocking Transparency and Control in Large Language Models

Siva Bagavathi, Co-Founder

GuhaTek

LLM Trivia: Mind-Boggling Numbers

Delve into some fascinating statistics that highlight the sheer scale and impact of Large Language Models.

~$ 200M

Cost to Train Model

Training a frontier model today (100–500B parameters) can cost in compute alone using 30K GPUs.

60%

Retrieval-Augmented LLMs cut hallucinations by

RAG can introduce **new** hallucinations if retrieval is bad (“grounding drift”).

$ 200B+

Global LLM Market by 2030

OpenAI, Anthropic, Google, Meta, and xAI dominate **60–70%** of global LLM value creation.

20 MW

Power consumption to run AI labs with GPUs

Same as a small town.

LLM Architecture and Internals

Core Components Inside an LLM

Embedding Layers

Convert discrete tokens into dense vector representations that capture semantic meaning and relationships between words.

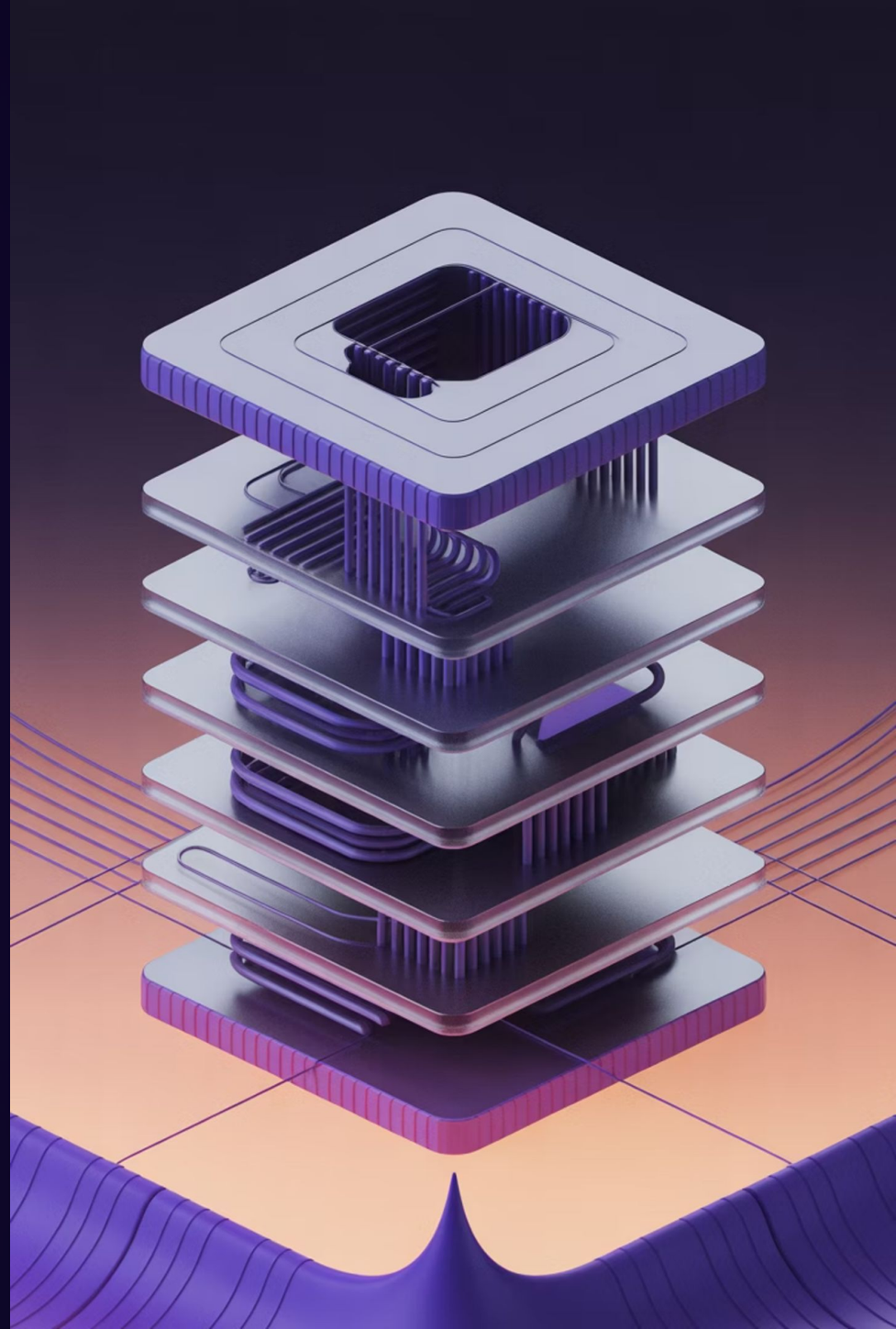
Transformer Blocks

Multiple layers of self-attention mechanisms that capture context and dependencies across the entire input sequence.

Output Layers

Generate probability distributions over the vocabulary for next token prediction, enabling text generation.

Non-deterministic outputs arise from sampling strategies and temperature parameters, making each generation potentially unique even with identical inputs.



CPU vs. GPU: The Powerhouses of LLM Operations

Understanding the distinct architectures and capabilities of Central Processing Units (CPUs) and Graphics Processing Units (GPUs) is fundamental to optimizing and observing LLMs, as their roles dictate efficiency and performance.

Central Processing Unit (CPU)

The CPU is the general-purpose processor, excelling in sequential task execution and control flow. Its strengths lie in:

- Executing diverse program instructions
- Managing system resources and I/O operations
- Handling non-parallelizable computational tasks

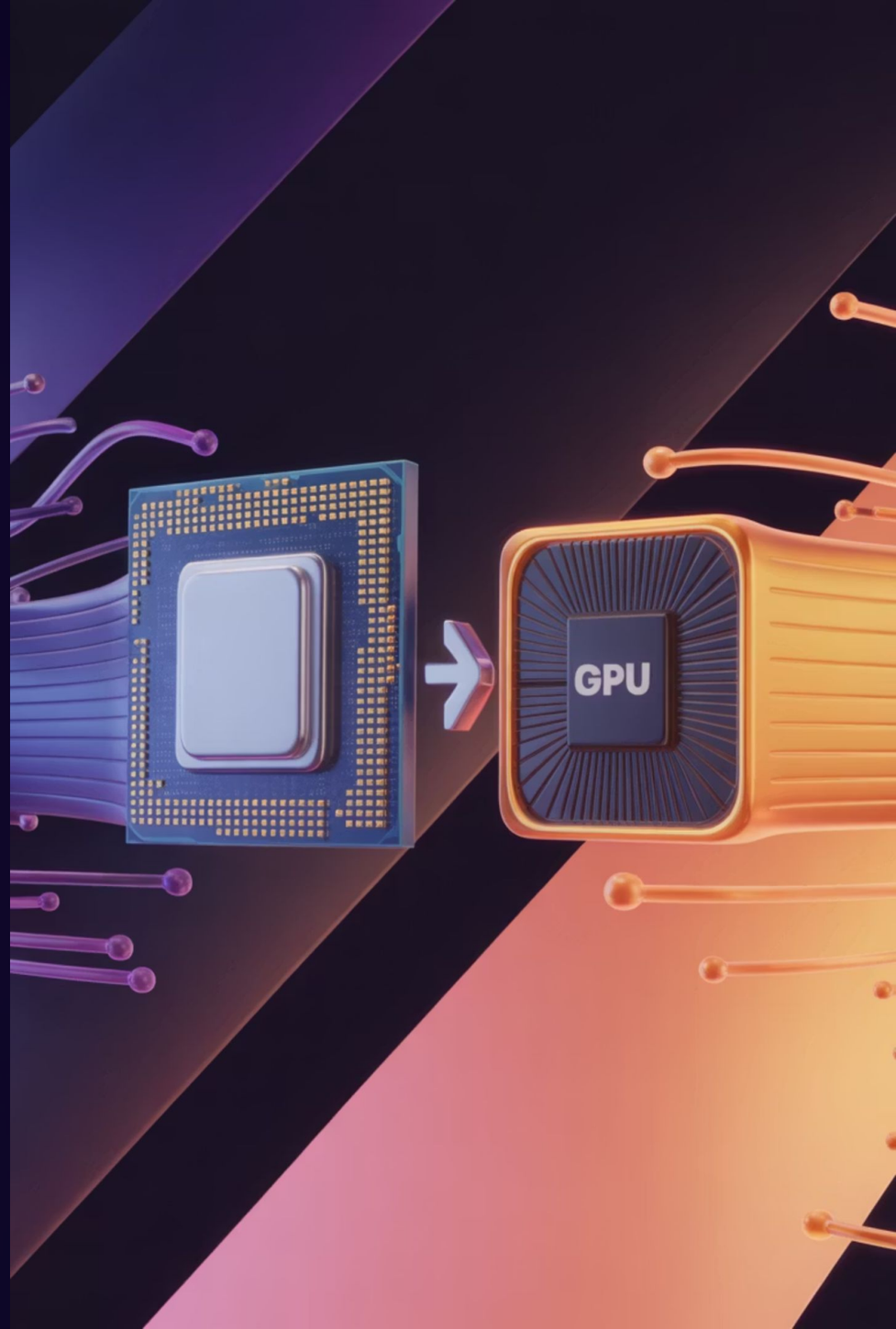
While essential for system orchestration, CPUs are less efficient for the massive parallel computations inherent in LLM operations.

Graphics Processing Unit (GPU)

GPUs are specialized accelerators designed for highly parallel computations, making them indispensable for LLMs due to:

- Thousands of processing cores for concurrent operations
- Optimized architecture for matrix multiplications (neural networks)
- Rapid acceleration of deep learning training and inference
- High memory bandwidth to manage large model parameters

They are the primary compute backbone for deploying and training modern LLMs.



GPU Monitoring

https://github.com/utkuozdemir/nvidia_gpu_exporter

Amazon CloudWatch

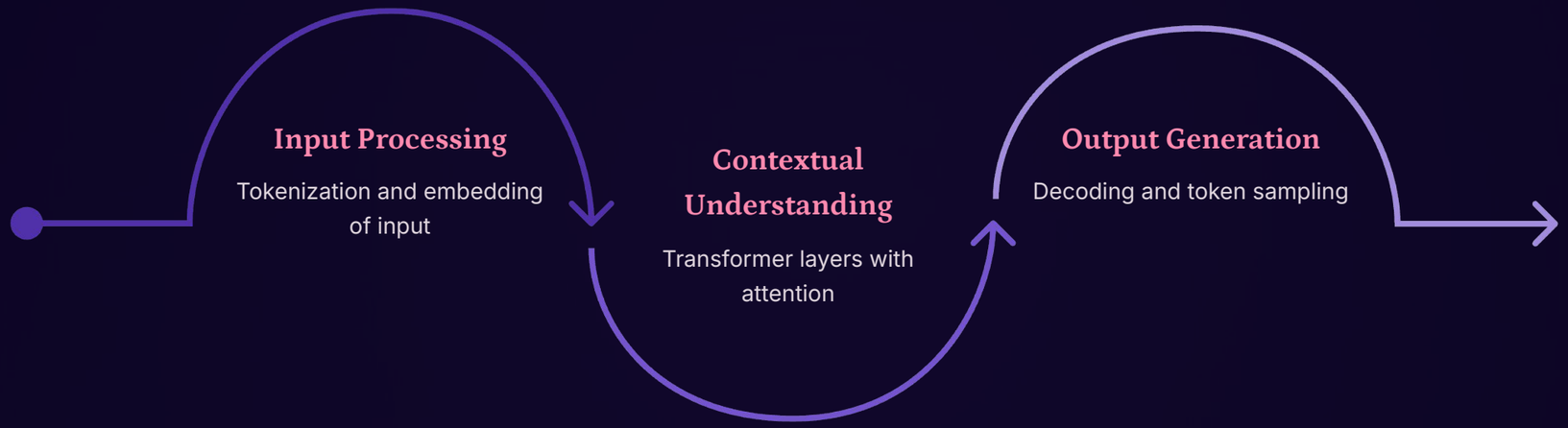
Azure Monitor

Cloud Monitoring

- nvidia-smi
- nvtop / nvitop
- gpustat
- intel_gpu_top
- radeontop

- Btop
- Glances

LLM Internals: Data Flow Architecture



The process begins by converting raw text into numerical embeddings, which are then fed into multiple transformer layers. These layers iteratively refine the contextual understanding of the input, enabling the model to learn complex patterns and relationships before finally generating a coherent and relevant output.

Example: What is machine learning?

User Query

The user inputs the natural language question: **"What is machine learning?"**

Tokenization & Embedding

The question is broken into tokens: `["What", "is", "machine", "learning", "?"]`. Each token is then converted into a numerical vector (embedding) representing its semantic meaning.

Transformer Processing

Embedded tokens pass through transformer layers. Attention mechanisms analyze relationships, e.g., "machine" and "learning" are closely related, forming a concept. The model builds a deep contextual understanding.

Output Generation

Based on the learned context, the model predicts the next most probable tokens. This iterative process generates a coherent answer, such as: **"Machine learning is a subset of artificial intelligence..."**

Tokens are not words

Tokens are the fundamental units of text that Large Language Models process. They are crucial for breaking down human language into a numerical format that models can understand and operate on.

What are Tokens?

Tokens are segments of text, often generated through a process called tokenization. Tokens might be:

- a whole word
- part of a word
- punctuation
- weird character fragments

Example

hello - 1 to 3 tokens

internationalization - 5 to 8 tokens

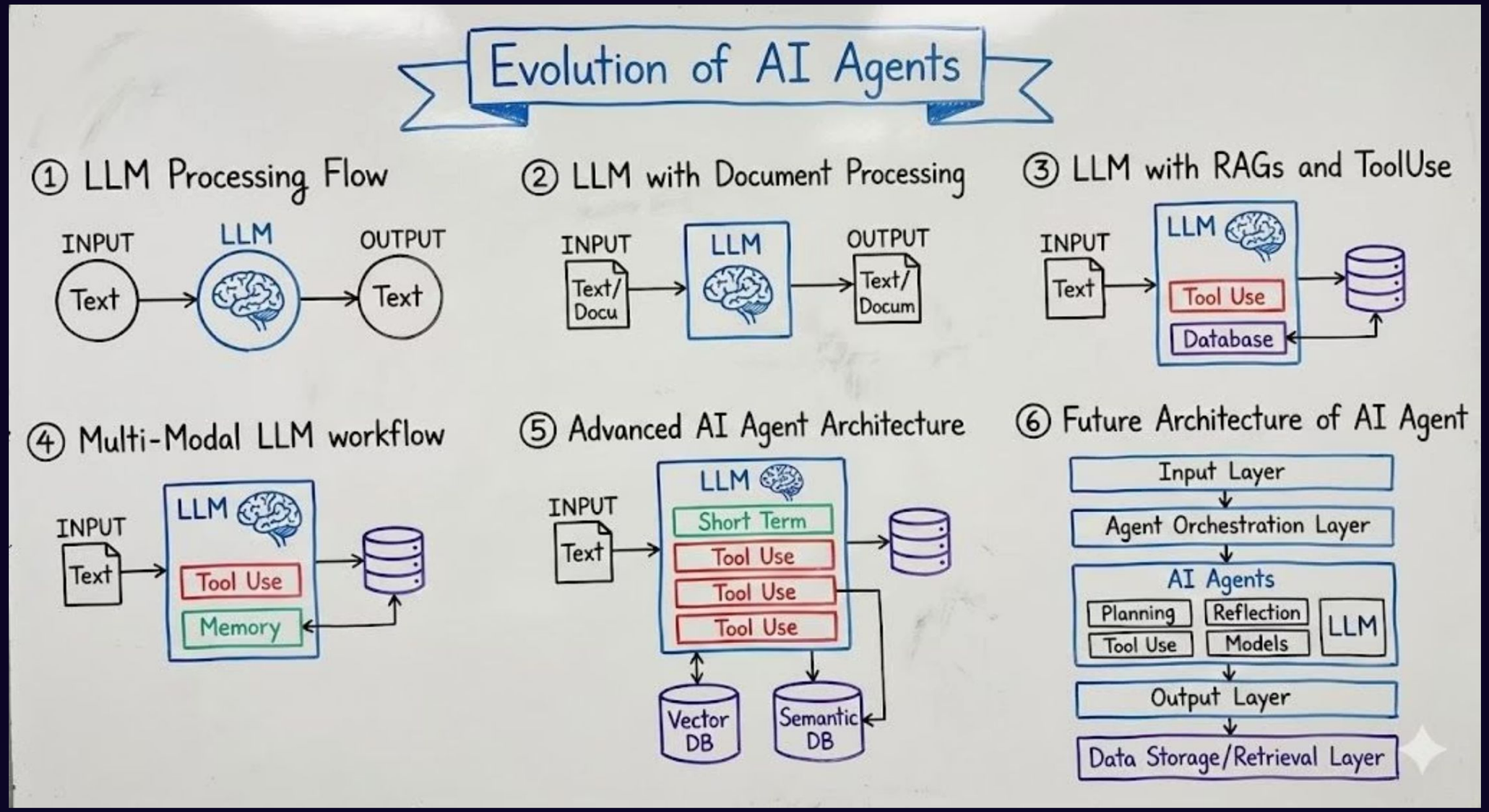
I like pizza. - 4 to 6 tokens

How to observe?

```
from transformers import AutoTokenizertokenizer = AutoTokenizer.from_pretrained("meta-llama/Llama-3-8B")tokens = tokenizer("I love running local models!", return_tensors="pt")print(tokens["input_ids"][0])
```

GPU has to store all tokens, and the more tokens, the more VRAM you burn.

LLM Evolution



Inside the LLM: Architecture and Parameters

Understanding the fundamental components that make up a Large Language Model is key to comprehending its capabilities and how to effectively observe its behavior.

Architecture

The blueprint of the model, defining how layers, connections, and computational blocks are arranged to process information.

Weights & Biases

Billions of adjustable parameters learned during training, representing the model's knowledge and enabling it to generate intelligent responses.

Tokenizer

A crucial pre-processing component that breaks raw text into numerical "tokens" for the model, and converts output tokens back into human-readable language.

Transformer Blocks

The "real engine" – these layers leverage self-attention mechanisms to efficiently process sequential data, capturing complex relationships within text and across long distances.

Quantization: Saving VRAM Without Breaking Quality

Reduced Precision

Converts high-precision model weights (e.g., 32-bit floats) to lower-precision formats (e.g., 8-bit integers or 4-bit). This drastically cuts down the data stored per parameter.

VRAM Efficiency

Significantly shrinks the model's memory footprint, allowing larger, more complex models to fit into the available GPU VRAM, making them accessible on consumer-grade hardware.

Minimal Quality Loss

Through advanced algorithms, quantization often results in negligible degradation of model accuracy or output quality, particularly for inference tasks, maintaining a good balance.

Enables Greater Scale

Unlocks the ability to run larger LLMs or process longer context windows, enhancing capabilities and expanding the reach of advanced AI applications.

Visibility is Control

Challenges Unique to LLMs

Silent Failures

Models generate confident but incorrect or hallucinated answers without any error signals, making detection difficult without human review or validation systems.

Performance Drift

Model quality degrades over time as data distributions shift, fine-tuning changes behavior, or prompt engineering becomes less effective.

Unbounded Costs

Token usage can spiral unpredictably through retry logic, verbose outputs, or inefficient prompting strategies, leading to budget overruns.

Opaque Reasoning

Complex multi-hop workflows and chain-of-thought processes make it challenging to understand how the model arrived at specific conclusions.

Compliance Risks

Without comprehensive traceability and audit logs, organizations face governance challenges and regulatory compliance issues.

Core LLM Observability Metrics

Quality Metrics

Track factual accuracy, hallucination rates, grounding in source material, and automated scores like BLEU (Bilingual Evaluation Understudy)and ROUGE (Recall-Oriented Understudy for Gisting Evaluation) to ensure output quality meets standards.

Reliability Metrics

Monitor latency, error rates, API rate limits, and uptime (targeting ~99.95%) to maintain consistent performance and availability.

Safety Metrics

Detect jailbreak attempts, measure toxicity levels, and identify PII leakage to protect users and maintain ethical AI deployment.

Cost Metrics

Count tokens consumed, track retry attempts, monitor API usage patterns, and ensure adherence to budget constraints.

Governance Metrics

Maintain comprehensive traceability, audit logs, and request lineage for compliance, debugging, and continuous improvement.

Stages of LLM Observability

Open Source Frameworks and Tools

OpenTelemetry

Metrics

Counters: Track total LLM requests, token counts, and error occurrences

Gauges: Monitor active connections and concurrent requests

Histograms: Capture latency distributions, token usage patterns, and response times

Traces

Spans: Represent individual operations like prompt preprocessing, embedding generation, model inference, and post-processing

Context: Parent-child relationships reveal the complete request journey across microservices and LLM providers

Prometheus

Grafana

OpenLLMetry

OpenLLMetry extends OpenTelemetry's capabilities, providing specialized instrumentation to capture granular data from Large Language Model interactions. It ensures deeper insights into prompt engineering, model performance, and AI-driven decision-making processes.

Semantic Conventions

Standardized definitions for LLM operations like prompt input/output, token counts, and model responses, ensuring consistent data capture.

Deep Trace Context

Capture end-to-end traces of LLM requests, from initial user input through multiple model calls, RAG pipelines, and agent reasoning steps.

Performance Metrics

Collect vital metrics on latency, token usage, cost, and error rates, enabling optimization and cost management for LLM applications.

Ecosystem Compatibility

Leverage existing OpenTelemetry collectors, processors, and observability backends, simplifying integration into current monitoring infrastructures.

Langfuse

LLM Application Observability

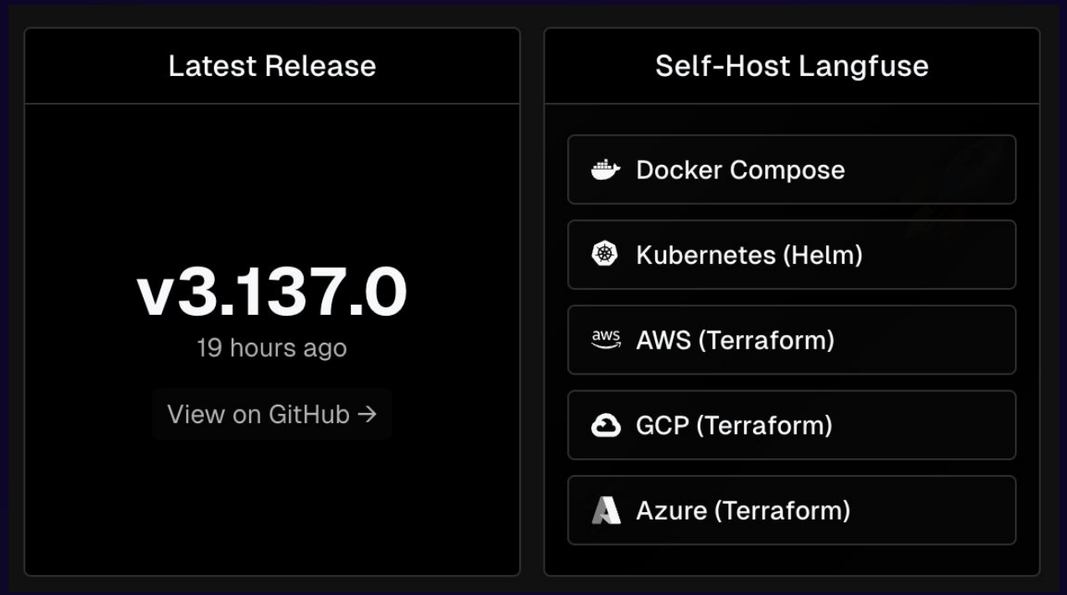
Instrument to get metrics and traces tracking LLM calls and other relevant logic in the app such as retrieval, embedding, or agent actions

Prompt Management

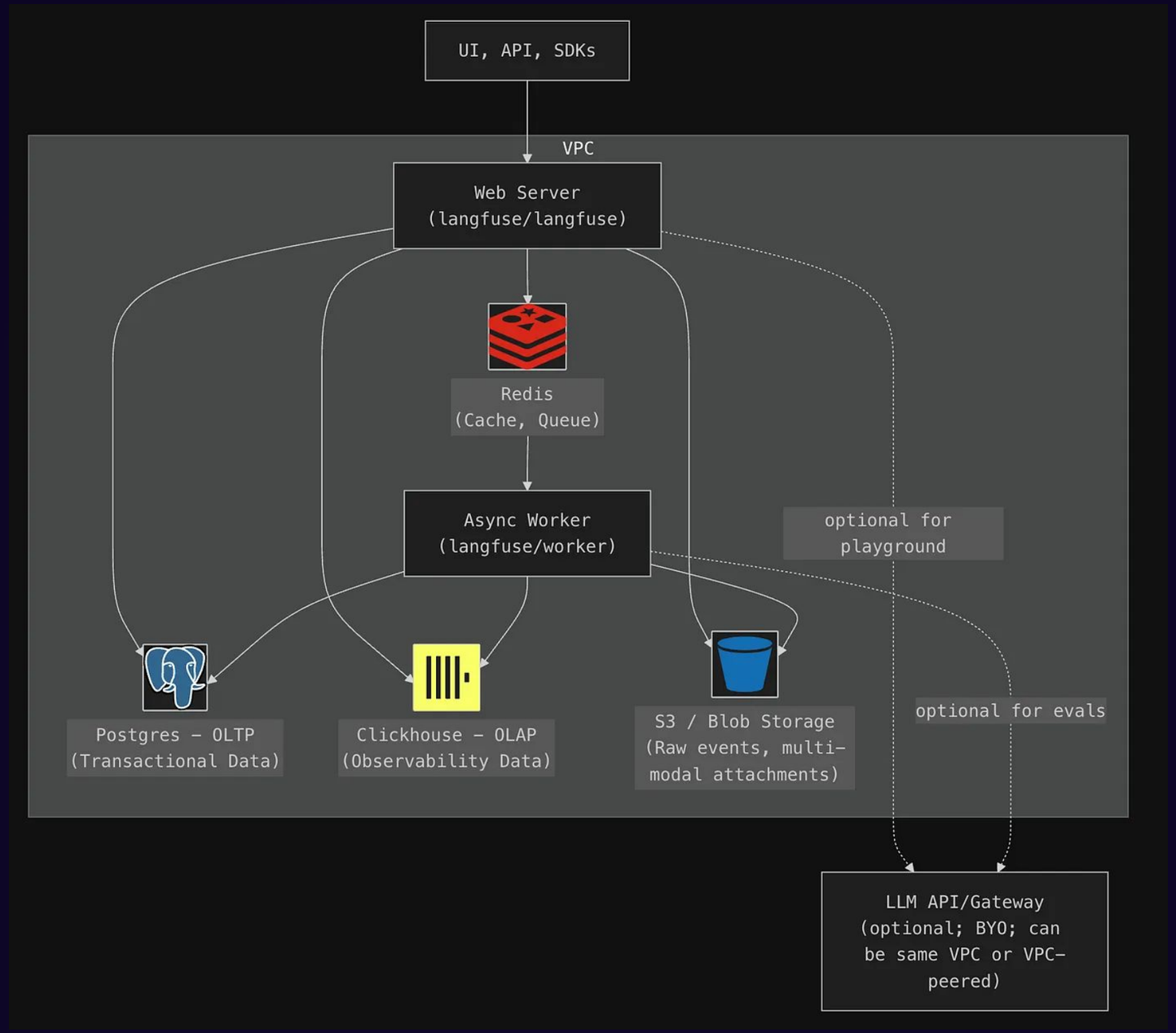
Centrally manage, version control, and collaboratively iterate on prompts with caching on server and client side, iterate on prompts without adding latency to the application.

Evaluations

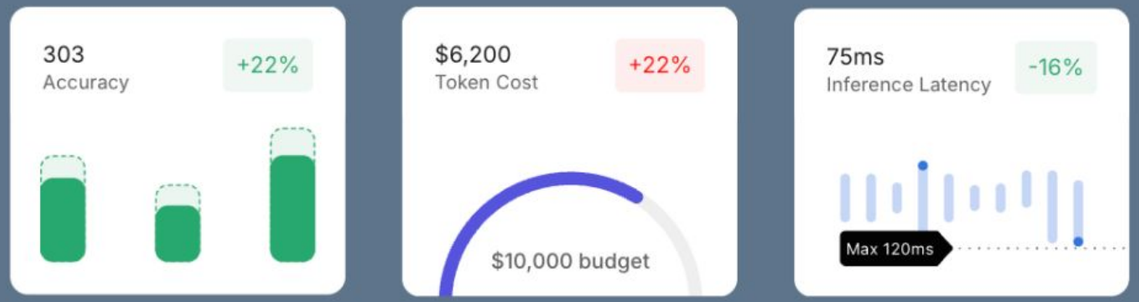
LLM-as-a-judge, user feedback collection, manual labeling, and custom evaluation pipelines via APIs/SDKs.



Langfuse


UI, API, SDKs
VPC
Web Server
(langfuse/langfuse)
Redis
(Cache, Queue)
Async Worker
(langfuse/worker)
optional for
playground
Postgres – OLTP
(Transactional Data)
Clickhouse – OLAP
(Observability Data)
S3 / Blob Storage
(Raw events, multi-
modal attachments)
optional for evals
LLM API/Gateway
(optional; BYO; can
be same VPC or VPC-
peered)

Langtrace



Track Vital Metrics

Dashboards to track token usage, cost, latency and evaluated accuracies.

Explore API Requests

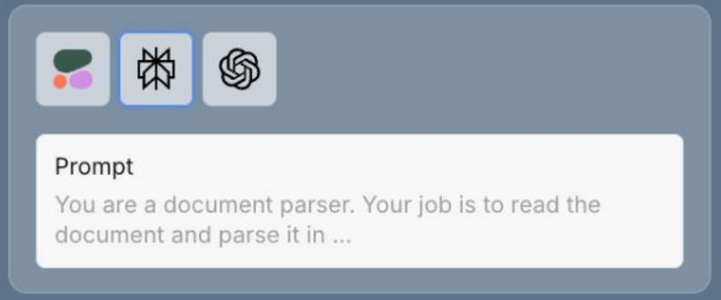
Automatically trace your GenAI stack and surface relevant metadata.

Evaluations

Measure baseline performance, curate datasets for automated evaluations and finetuning

Prompt Version Control

Store and version control your prompts. Deploy new prompts or roll back with just a few clicks.



Playground

Compare the performance of your prompts across different models.

Opik by Comet is an open-source platform designed specifically for evaluating, testing, and monitoring LLM applications. It provides end-to-end observability with built-in evaluation metrics, experiment tracking, and production monitoring capabilities.

Trace & Debug LLM Calls

Automatically capture and visualize every LLM interaction including prompts, completions, token usage, and latency. Debug complex chains and agent workflows with detailed execution traces.

Evaluate with Built-in Metrics

Access pre-built evaluation metrics for hallucination detection, answer relevance, context precision, and more. Create custom evaluators tailored to your specific use cases via Heuristics and LLM-as-a-Judge metrics.

Experiment Tracking

Compare different prompts, models, and parameters side-by-side. Track experiments across development iterations to identify the best-performing configurations.

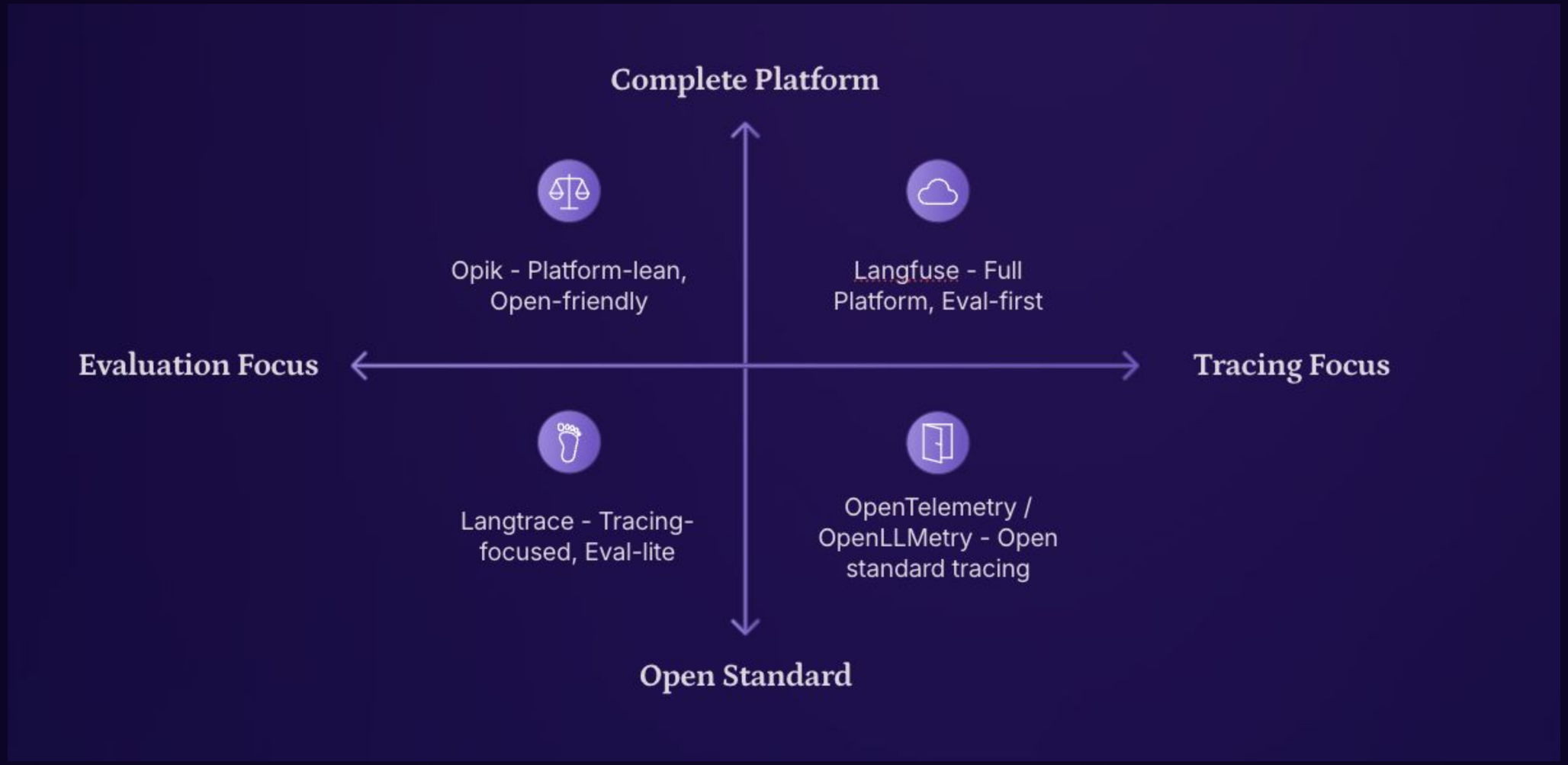
Production Monitoring

Monitor LLM applications in production with real-time dashboards, cost tracking, and performance analytics. Set up alerts for anomalies and quality degradation.

Comparison

Tool	Primary Focus	LLM Tracing (Prompts/Responses)	Token & Cost Tracking	Evaluation & QA	Production Monitoring	OTEL Compatible	UI Included
OpenLLMetry	LLM telemetry via OpenTelemetry	✅ (as OTEL spans)	✅	❌	✅ (via Grafana/Jaeger etc.)	✅ Native	❌ (uses external OTEL UI)
Langfuse	Full LLM app observability	✅	✅	✅ (built-in evals, A/B)	✅	⚠️ Partial	✅
Langtrace	OTEL-native LLM tracing	✅	✅	✅	✅	✅ Native	✅
OTEL + Prom/Grafana/Jaeger	Infra + app observability	⚠️ Manual instrumentation	⚠️ Manual	❌	✅ (very mature)	✅ Native	✅
Opik	LLM observability + evaluation + safety	✅	✅	✅ (LLM-as-judge, testing)	✅	❌	✅

Comparison


Complete Platform
Opik - Platform-lean, Open-friendly
Langfuse - Full Platform, Eval-first
Evaluation Focus
Tracing Focus
Langtrace - Tracing-focused, Eval-lite
OpenTelemetry / OpenLLMetry - Open standard tracing
Open Standard

Demo

Live Walkthrough

Watch as we trace an agent's complete interaction lifecycle, from initial prompt through tool invocations, inter-agent communications, and final response generation.

1. **Agent Receives Task**
   Initial prompt and context setup
2. **Tool Invocations (Optional)**
   Internal transformation and output generation
3. **MCP Messages (Optional)**
   Communication with other agents
4. **Response Generation**
   Final output and metrics collection

24,700ms
Average Latency
End-to-end response time

$0
Cost Per Request
Token usage and API calls

100%
Safety Checks
All guardrails passed

4.2
Output Quality
Automated evaluation score

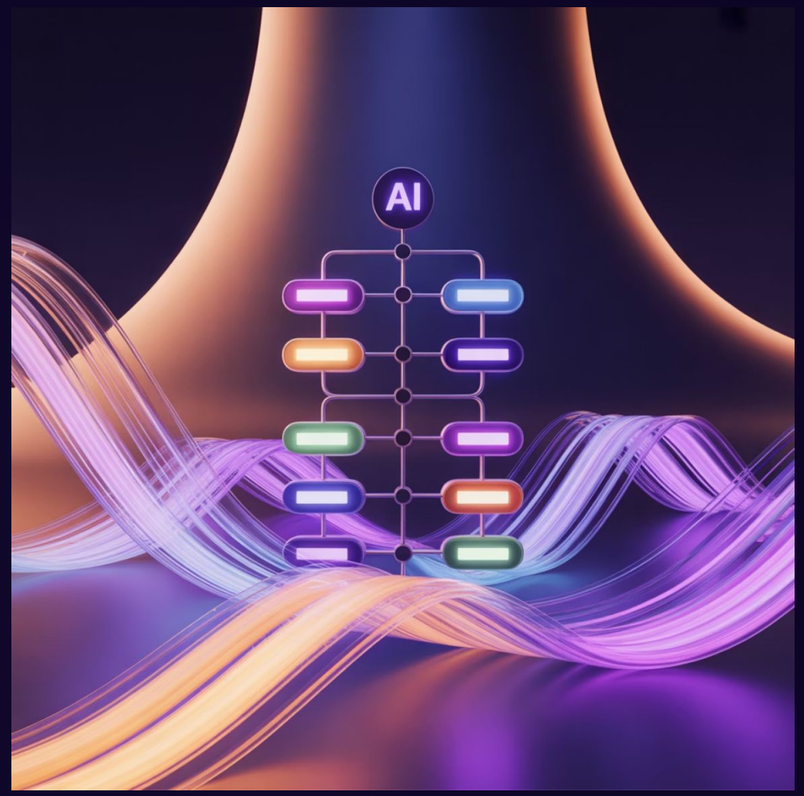
Observability of AI Agents

Beyond Single Models

Observing AI Agents in Action

Agent Complexity

AI agents orchestrate multiple LLM calls, integrate various tools and external APIs, and make autonomous decisions based on complex reasoning chains. This creates new observability challenges.



Decision Path Tracking

Monitor the complete sequence of agent decisions, including reasoning steps, tool selections, and branching logic to understand behavior.

Tool Invocation Success

Track success rates, latencies, and fallback behaviors for each tool and API call to identify bottlenecks and failure points.

Silent Failure Detection

Identify reasoning errors and incorrect assumptions across multi-step agent workflows before they cascade into larger issues.

Observing Agent-to-Agent Communication

The MCP enables agents to communicate securely, delegate tasks effectively, and share context across distributed AI systems, creating sophisticated multi-agent workflows.

Agent Communication

Standardized message formats and protocols

Task Delegation

Intelligent work distribution across agents

Observability

Message flow tracking and protocol compliance

Context Sharing

Secure information exchange and state management

MCP observability provides unprecedented transparency into message flows, latency between agents, protocol compliance, and coordination patterns in complex multi-agent AI systems.

Layers of AI Agent Observability

1

Application Layer

User-facing frontend/backend app with end-user feedback (thumbs up/down, text feedback)

2

Orchestration Layer

Advanced metrics & analysis (guardrails, chain performance, prompt caching, model usage)

3

Agentic Layer

Agent to Agent communication, protocol, command execution, tool usage, handover

4

Model / LLM Layer

Model token usage/cost, stability, latency, invocation errors, resource utilization

5

RAG / Vector DB Layer

RAG pipeline: data retrieval performance, accuracy, LLM input and outputs, semantic analysis

6

Infrastructure Layer

Compute, GPU, network, resource monitoring

Best Practices for Production

1

Start with Core Metrics

Begin with latency, cost, and error rates before expanding to quality and safety metrics.

2

Automate Evaluation

Implement automated testing and quality checks to catch issues before they reach users.

3

Build Feedback Loops

Use observability data to continuously improve prompts, models, and system architecture.

4

Prioritize Traceability

Maintain comprehensive logs and traces for debugging, compliance, and continuous learning.

llm-observability - GitHub Page

https://github.com/guhatek/llm-observability

Session Feedback

Post your valuable feedback in comments section

THANKS!

Stop Scrolling! Start rolling to make a positive impact!

Follow-Up questions or feedback?

Email

hello@guhatek.com

Visit us

www.guhatek.com

GuhaTek

You Build it, We Perfect It