LLMs in AWS : Observability Maturity from Foundations to AIOps

Indika Wimalasuriya

Large Language Models (LLMs) - 2024

“From geeks to everyone: AI is now mainstream with LLM-based GenAI apps everywhere! “

“From Dev Success to Ops Woes: You Know How It's Going to End“

Agenda

Quick Intro about myself

- Resides in Colombo, Sri Lanka, with my beautiful daughter and wife.
- Reliability Engineering Advocate, Solution Architect (specializing in SRE, Observability, AIOps, & GenAI).
- Employed at Virtusa, overseeing technical delivery and capability development.
- Passionate Technical Trainer.
- Energetic Technical Blogger.
- AWS Community Builder - Cloud Operations.
- Ambassador at DevOps Institute (PeopleCert).

Amazon Bedrock

The easiest way to build and scale generative AI applications with foundation models

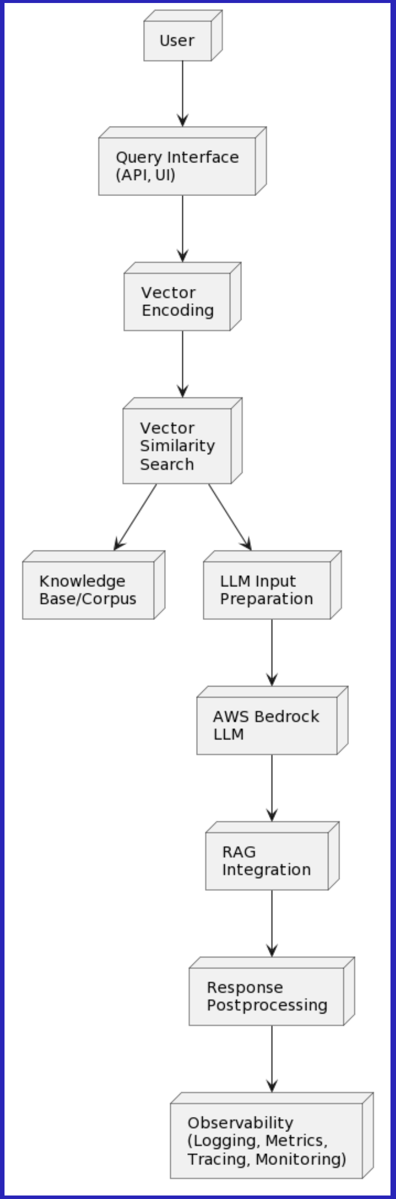
- **Amazon Bedrock Overview:**
  - Fully managed service
  - Offers a choice of high-performing foundation models (FMs) from leading AI companies
  - Provides a single API for accessing models
  - Includes a broad set of capabilities for building generative AI applications

- **Foundation Models (FMs) Providers:**
  - AI21 Labs
  - Anthropic
  - Cohere
  - Meta
  - Mistral AI
  - Stability AI
  - Amazon

- **Key Features:**
  - Experimentation and evaluation of top FMs for specific use cases
  - Private customization with own data using techniques like fine-tuning and Retrieval Augmented Generation (RAG)
  - Building agents executing tasks using enterprise systems and data sources

- **Serverless Infrastructure:**
  - No need to manage any infrastructure
  - Secure integration and deployment of generative AI capabilities into applications
  - Utilizes familiar AWS services

Overview of GenAI Apps

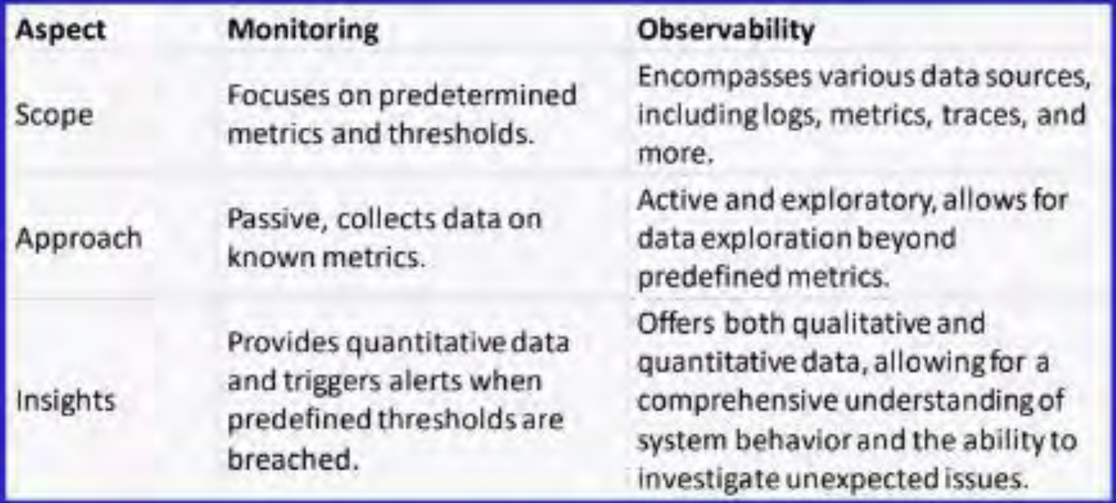


- User Query Processing: Preprocessing and encoding the user's query.
- Vector Similarity Search: Finding relevant documents based on query similarity in the vector database.
- Document Retrieval: Retrieving the top-k most relevant documents from the database.
- LLM Input Preparation: Combining the user's query with retrieved documents for LLM input.
- LLM Inference: Sending the input to the LLM for response generation.
- RAG Integration: Combining the LLM output with retrieved information using RAG.
- Response Postprocessing: Formatting, filtering, or applying additional logic to the final response.
- Response Delivery: Delivering the final response to the user interface or API.

What is Observability?

Observability - Ability to gain insight into the internal workings and behavior of a system or application through the analysis of its outputs, often without direct access to its internal state

Observability Pilers – Logs, Metrics & Tracers



Aspect	Monitoring	Observability
Scope	Focuses on predetermined metrics and thresholds.	Encompasses various data sources, including logs, metrics, traces, and more.
Approach	Passive, collects data on known metrics.	Active and exploratory, allows for data exploration beyond predefined metrics.
Insights	Provides quantitative data and triggers alerts when predefined thresholds are breached.	Offers both qualitative and quantitative data, allowing for a comprehensive understanding of system behavior and the ability to investigate unexpected issues.

- What is happening in my system right now?
- How is it performing, and are there any issues or anomalies?
- How are different components of my system interacting with each other?
- What caused a particular issue or incident, and how can it be resolved?

Observability in LLMs: What Exactly Is It?

LLM Observability

- Direct LLM Observability or Observability of LLM Itself
  - Monitor, evaluate, and troubleshoot the LLM directly
  - Understand behavior and efficiency
- Indirect LLM Observability or Observability of Application/System Using the LLM
  - Monitor the application/system utilizing the LLM
  - Ensure robustness and reliability in real-world deployments

Purpose

- Ensure robustness and reliability in real-world deployments

Techniques Involved

- Logging model outputs
- Monitoring internal states
- Detecting anomalies
- Tracking performance metrics
- Implementing human-in-the-loop methodologies
- Employing model explainability strategies

Insight into the Core: Direct LLM Observability

- Integration of observability capabilities during training and deployment
- Focus on monitoring internal states and operations

Key Areas of Focus:

- Logging activations, attention weights, and other internal states during inference
- Implementing probes or instrumentation within the model architecture
- Tracking performance metrics such as latency and memory usage
- Enabling explainability techniques like attention visualization

Objective:

- Gain insights into LLM's functioning
- Detect anomalies or performance issues
- Understand decision-making processes

Beyond the Surface: Indirect LLM Observability

- Integration of observability capabilities into the application rather than the LLM itself
- Focus on understanding the internal state of the application while indirectly monitoring the LLM

Key Areas of Focus:

- Logging inputs and outputs to/from the LLM
- Monitoring application performance metrics
- Implementing anomaly detection on LLM outputs
- Enabling human feedback loops to assess LLM output quality
- Tracking application-level metrics like error rates and latency

Objective:

- Gain insight into LLM usage within the application
- Observe the application's behavior and performance when interacting with the LLM
- Ensure reliability of the application built around the LLM

LLM Observability : Observability Maturity Model for AWS Bedrock Integrated Applications

- **Indirect LLM Observability**
- **Maturity model for observing applications with AWS Bedrock integration**
- **Integration of observability practices into application architecture**
- **Comprehensive monitoring of application's internal state**
- **Indirect oversight of LLM functionalities**
- **Professional and reliable framework for assessing performance and reliability**
- **Ensures robustness of applications utilizing AWS Bedrock**

Why Observability matters for LLMs

Monitoring of LLM Performance	Detecting Anomalies and Biases	Monitor Model Drift
Data Privacy and Security	Optimizing Resource Utilization	Debug and Troubleshooting
Continues Improvement	Coloration and knowledge sharing	Continuous Improvements

Pillars shaping (indirect) LLM Observability

LLM Specific Metrics

Metric	Details
LLM Inference Latency	• Track the latency of LLM inference requests within the Bedrock application. • Monitor the latency at different stages of the request lifecycle, such as API Gateway, Lambda functions, and the LLM service itself. • Identify potential bottlenecks and optimize the LLM integration for better performance.
LLM Inference Success Rate	• Monitor the success rate of LLM inference requests within the Bedrock application. • Track the percentage of successful inferences compared to failed or errored requests. • Identify and troubleshoot any issues related to the LLM integration or the input data.
LLM Output Quality	• Implement automated evaluation metrics to assess the quality of the LLM's output within the Bedrock application. • Use metrics like BLEU or ROUGE scores for text generation tasks, or task-specific metrics for other use cases. • Monitor changes in output quality over time and correlate them with updates to the LLM model or integration code.
LLM Prompt Effectiveness	• Track the effectiveness of the prompts used to query the LLM within the Bedrock application. • Monitor the quality of the LLM's output for different prompts and identify prompts that lead to suboptimal or undesirable outputs. • Continuously refine and improve the prompts based on the observed effectiveness.

LLM Specific Metrics (Cont.)

Metric	Details
LLM Model Drift	• Monitor the distribution of the LLM's output within the Bedrock application over time. • Identify significant changes in the output distribution that may indicate model drift. • Track the performance of the LLM on a held-out evaluation set or benchmark tasks specific to the Bedrock application.
LLM Cost Optimization	• Monitor the cost associated with LLM inference requests within the Bedrock application. • Track the cost per inference and the total cost over time. • Identify opportunities for cost optimization, such as caching or batching inference requests.
LLM Integration Errors	• Monitor and log any errors or exceptions that occur during the integration of the LLM with the Bedrock application. • Track the frequency and severity of integration errors. • Identify and troubleshoot issues related to the integration code or the communication between the Bedrock application and the LLM service.
LLM Ethical Considerations	• Monitor the LLM's output within the Bedrock application for potential ethical risks or violations. • Track instances of harmful, illegal, or discriminatory content generated by the LLM. • Ensure that the LLM's output aligns with established ethical principles and guidelines for responsible AI development and deployment.

Prompt Engineering Properties

Properties *	Details
Temperature	• Controls randomness in model output. Higher temperatures yield more diverse responses; lower temperatures, more focused.
Top-p Sampling	• Controls output diversity by considering only most probable tokens.
Top-k Sampling	• Considers only k most probable tokens for generating next token.
Max Token Length	• Sets maximum length of generated text.
Stop Tokens	• Signals model to stop generating text when encountered.
Repetition Penalty	• Penalizes model for repeating text, encouraging diversity.
Presence Penalty	• Penalizes model for generating already generated tokens.
Batch Size	• Determines number of input sequences processed simultaneously.

* Log data to CloudWatch Logs, publish it as custom metrics, set up alarms, and visualize it using CloudWatch Dashboards or AWS Managed Grafan

Properties **	Details
Inference Latency	• Time taken for model to generate output given input.
Model Accuracy & Metrics	• Task-specific metrics like accuracy, perplexity, or BLEU score.

** Instrument the application with AWS X-Ray, publish as custom metrics to CloudWatch, set up alarms, and visualize the data in CloudWatch Dashboards or AWS Managed Grafana.

Performance Metrics, Logging, and Tracing for RAG Models

Telemetry	Details
Metrics	• **Query latency**: Track the time it takes for the RAG model to process a query and generate a response. This can help identify performance bottlenecks. • **Success rate**: Monitor the percentage of successful queries versus failed queries. This can indicate issues with the model or the underlying infrastructure. • **Resource utilization**: Monitor the CPU, memory, and network usage of the RAG model and the associated services. This can help with capacity planning and identifying resource constraints. • **Cache hit rate**: If you're using a cache for retrieved documents, monitor the cache hit rate to understand its effectiveness.
Logs	• **Query logs**: Log the input queries, retrieved documents, and generated responses. This can aid in debugging and understanding the model's behavior. • **Error logs**: Log any errors or exceptions that occur during query processing or document retrieval. This can help identify and troubleshoot issues. • Audit logs: Log user interactions, authentication events, and any sensitive operations for security and compliance purposes.
Tracers	• **End-to-end tracing**: Implement distributed tracing to track the flow of a query through the various components of the RAG model, including document retrieval, encoding, and generation.

Tracing

Distributed Tracing:	**Details**
AWS X-Ray:	• Implement distributed tracing using AWS X-Ray. • Gain end-to-end visibility into your LLM application's request flow. • Identify performance bottlenecks. • Troubleshoot issues across distributed components.
Integration with AWS Services:	• Integrate AWS X-Ray with other AWS services • Enable X-Ray tracing in the respective service configurations. • Capture traces across your application's architecture.

Visualization Tools

Distributed Tracing:	Details
AWS CloudWatch Dashboard	• Create custom dashboards in CloudWatch. • Visualize metrics, logs, and traces related to your LLM application. • Add widgets for various data sources, such as metric graphs, log insights, and X-Ray service maps.
Integration with AWS Services:	• Consider using third-party visualization tools like Grafana or Kibana. • Integrate these tools with AWS services. • Utilize services like Amazon Managed Service for Prometheus or Amazon OpenSearch Service for integration.

Alerting and Incident Management:

Distributed Tracing:	Details
CloudWatch Alarms:	• Set up CloudWatch Alarms to receive notifications. • Configure alarms to trigger automated actions based on predefined thresholds for critical metrics or log patterns. • Choose notification methods such as email, SMS, or integration with other AWS services like Amazon SNS or AWS Lambda.
AWS Systems Manager	• Integrate with AWS Systems Manager for incident response and remediation workflows. • Utilize Systems Manager to automate the execution of runbooks, scripts, or other actions in response to incidents or alerts.

Security and Compliance:

Distributed Tracing:	Details
AWS CloudTrail:	• Leverage AWS CloudTrail to audit and monitor API calls made to AWS services by your LLM application. • Ensure compliance with security and regulatory requirements. • Integrate CloudTrail logs with CloudWatch Logs for centralized log management and analysis.
AWS Config Rules	• Implement AWS Config Rules to continuously monitor and assess the configuration of your AWS resources. • Ensure compliance with best practices and compliance standards. • Set up Config Rules to monitor specific resource types and configurations. • Trigger remediation actions or notifications when violations are detected.

Cost Optimization:

Distributed Tracing:	**Details**
AWS Cost Explorer	• Use AWS Cost Explorer to monitor and analyze the costs associated with your LLM application's observability infrastructure. • Access detailed reports and visualizations provided by Cost Explorer. • Identify cost drivers and optimize spending based on the insights gained.
AWS Budgets:	• Set up AWS Budgets to receive alerts. • Receive notifications when your observability costs exceed predefined thresholds. • Utilize Budgets to proactively manage and control your observability costs.

AIOps Capabilities

Distributed Tracing:	Details
Metric Anomaly Detection	• AWS CloudWatch provides anomaly detection capabilities for metrics. • Monitor LLM performance using anomaly detection. • Set up anomaly detection to identify unusual patterns or deviations in LLM model metrics. • Focus on metrics such as inference latency or resource utilization.
Log Anomaly Detection:	• AWS CloudWatch Logs Insights can be used to detect anomalies in your application's log data. • Utilize Logs Insights to analyze the LLM model's outputs and application log data. • Create custom queries and patterns to identify potential biases, inappropriate content, or other anomalies in the generated text.
AWS DevOps Guru:	• AWS DevOps Guru is a machine learning-powered service. • It helps detect and resolve operational issues in your application. • DevOps Guru analyzes various data sources, including metrics, logs, and events. • It identifies potential problems and provides recommendations for remediation.

AIOps Capabilities (Cont.)

Distributed Tracing:	**Details**
AWS CodeGuru	• AWS CodeGuru can analyze your application's code. • It identifies potential issues such as performance bottlenecks, security vulnerabilities, or code quality problems. • CodeGuru provides recommendations for improving your code. • It helps optimize your application's performance..
AWS Forecasting	• Integrate AWS Forecasting for predictive analytics • Utilize machine learning models to forecast resource utilization and application performance trends • Enhance proactive decision-making and capacity planning for LLM-powered applications

Why you need Maturity Model?

Maturity Framework

Level 3 Advanced LLM Observability with AIOps

- Integrate AWS DevOps Guru for automated insights and remediation.
- Utilize AWS CodeGuru for code reviews and optimization.
- Explore and integrate third-party observability tools.
- Implement advanced AIOps practices for self-healing and auto-remediation.

Level 2 Proactive LLM Observability

- Capture and analyze advanced LLM metrics.
- Log and monitor advanced prompt properties.
- Enhance alert and incident management processes.
- Ensure security compliance and advanced cost optimization.
- Leverage AWS forecasting services.
- Set up metric and log anomaly detection.

Level 1 Foundation LLM Observability

- Capture basic LLM metrics.
- Log and monitor basic prompt properties.
- Implement basic logging and distributed tracing.
- Set up basic visualizations and dashboards.

Level 1: Foundation LLM Observability

Telemetry Data	Service	Description
LLM Metrics	AWS CloudWatch Metrics	Capture and log basic LLM metrics (e.g., inference time, model size, prompt length)
Prompt Properties	AWS CloudWatch Logs	Log and monitor basic prompt properties (e.g., prompt content, prompt source)
Logs	AWS CloudWatch Logs	Implement basic logging for LLM-related events and outputs
Distributed Tracing	AWS X-Ray	Implement distributed tracing to understand the flow of requests through your LLM-powered application
Visualizations	AWS CloudWatch Dashboards	Set up basic visualizations and dashboards for LLM metrics and logs
Alert and Incident Management	AWS CloudWatch Alarms	Establish basic alert and incident management processes for LLM-related issues
Security Compliance	AWS Cost Explorer	Implement basic monitoring for LLM usage and cost controllers
Cost Optimization	AWS Cost Explorer	Implement basic cost optimization strategies for LLM inference

Level 1: Foundation LLM Observability

Telemetry Data	Service	Description
LLM Metrics	AWS CloudWatch Metrics	Capture and analyze advanced LLM metrics (e.g., model performance, output quality)
Prompt Properties	AWS CloudWatch Logs	Log and monitor advanced prompt properties (e.g., prompt performance, prompt versioning)
Alert and Incident Management	AWS Systems Manager	Enhance alert and incident management processes for LLM-related issues
Security Compliance	AWS Security Hub	Ensure LLM security compliance and monitoring
Cost Optimization	AWS Cost Explorer	Implement advanced cost optimization strategies for LLM inference
AWS Forecasting	AWS Cost Explorer	Leverage AWS forecasting services to predict future LLM usage and costs
Metric Anomaly Detection	AWS CloudWatch Anomaly Detection	Set up metric anomaly detection for LLM metrics
Log Anomaly Detection	AWS CloudWatch Logs Insights	Implement log anomaly detection to identify unusual patterns in LLM logs

Level 1: Foundation LLM Observability

Telemetry Data	Service	Description
AWS DevOps Guru	AWS DevOps Guru	Integrate AWS DevOps Guru for automated operational insights and remediation
AWS CodeGuru	AWS CodeGuru	Utilize AWS CodeGuru for automated code reviews and optimization recommendations
Third-Party Tool Integration	Amazon Managed Service for Prometheus (AMP)	Explore and integrate third-party observability tools specific to LLMs
Third-Party Tool Integration	Amazon OpenSearch Service	Explore and integrate third-party observability tools specific to LLMs
AIOps Practices	AWS Systems Manager Automation	Implement advanced AIOps practices for self-healing and auto-remediation
AIOps Practices	AWS Lambda	Implement advanced AIOps practices for self-healing and auto-remediation

LLM : Measure Progress with Business Outcomes

- **LLM Output Quality:** *Track coherence, relevance, factual accuracy, and potential biases or toxicity in the LLM's output to ensure high-quality results.*
- **LLM Prompt Engineering:** *Assess prompt quality and diversity to optimize the effectiveness of queries and improve output quality over time.*
- **LLM Model Drift**: *Monitor output distribution and performance degradation over time to identify and address model drift issues promptly.*
- **LLM Ethical Considerations:** *Identify and mitigate potential ethical risks in the LLM's output, ensuring alignment with ethical principles and guidelines.*
- **LLM Interpretability and Explainability:** *Monitor rationale, reasoning, uncertainty, and confidence in the LLM's output to enhance interpretability and user trust in the system.*

General : Measure Progress with Business Outcomes

- Define clear goals (e.g., reducing downtime, enhancing satisfaction).
- Track observability's impact over time.
- Focus on metrics like cost reduction, faster issue resolution.
- Set improvement targets for each maturity stage.
- Use data to quantify customer experience benefits.
- Relate maturity stages to enhanced customer service.
- Showcase how maturity accelerates innovation.
- Align observability with strategic customer-focused objectives.
- Secure executive buy-in by highlighting customer-centric results.

Best practices

- Adopt structured logging practices with tools like AWS Lambda Powertools.
- Instrument code for comprehensive observability with metrics, logs, and traces.
- Leverage AWS service integrations (e.g., CloudWatch, X-Ray) for streamlined setup.
- Define meaningful metrics and set up alerts for proactive monitoring.
- Implement distributed tracing with AWS X-Ray for end-to-end visibility.
- Automate observability infrastructure setup using IaC tools.
- Continuously analyze data for improvement and foster a culture of iterative observability enhancement.

Pitfalls to Avoid

- Ensure observability strategy aligns with security and compliance requirements.
- Monitor and optimize costs associated with observability infrastructure.
- Clearly define roles and responsibilities for observability management.
- Implement appropriate data retention policies for observability data.
- Explore third-party integrations for advanced analytics and visualization capabilities.

Thank you.