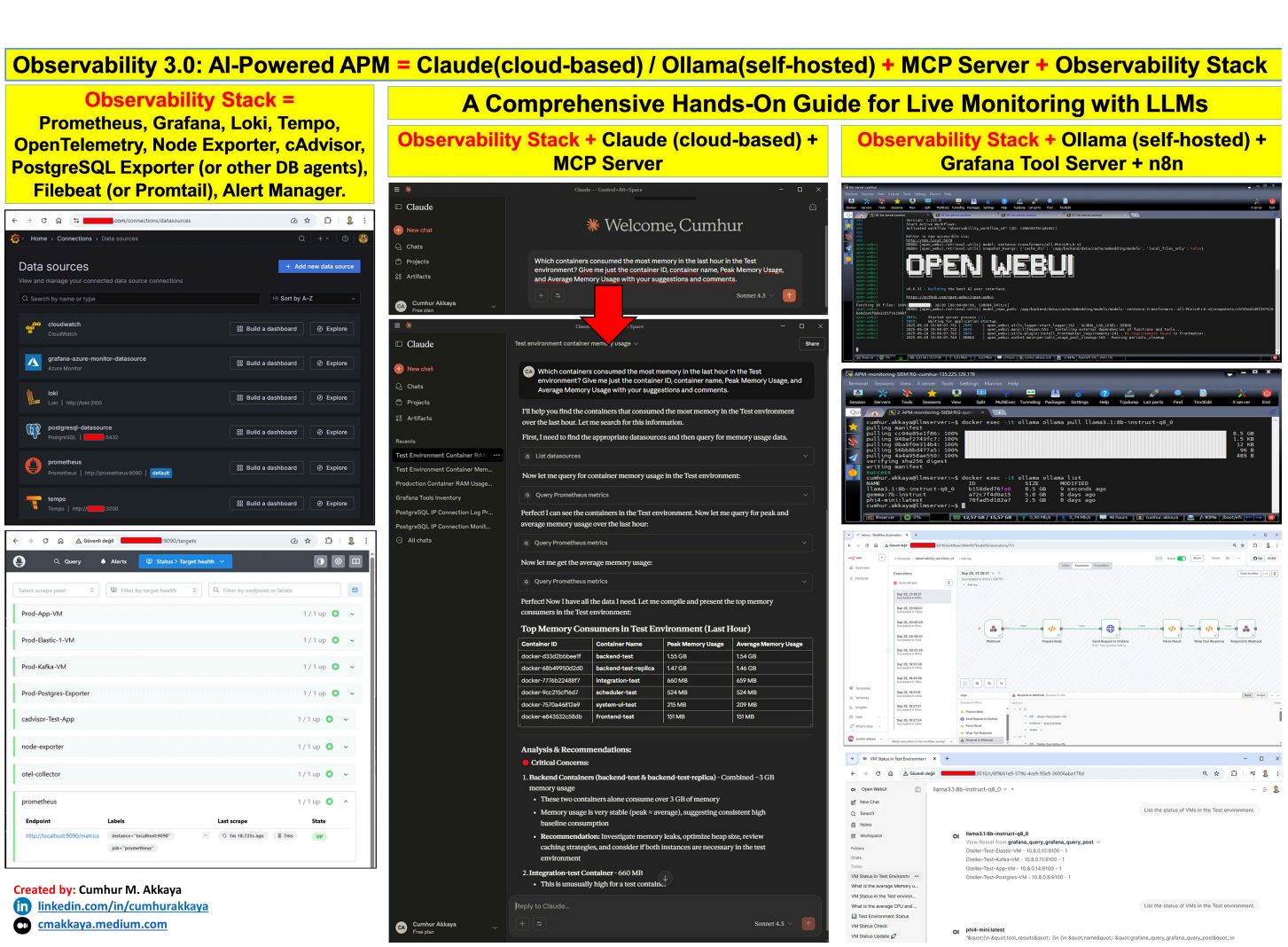
Member-only story

Observability 3.0 AI-Powered APM = Claude (cloud-based) / Ollama (self-hosted) + MCP Server + n8n + Prometheus, Grafana, Loki, Tempo, OpenTelemetry, PostgreSQL Exporter, Node Exporter, cAdvisor, Filebeat, Alert Manager — A Comprehensive Hands-On Guide for Live Monitoring with LLMs (Large Language Models)

In today's cloud-based world, observability is no longer just about collecting logs, metrics, and traces. We can take observability to the next level with AI-powered APM (Application Performance Monitoring). Tools like Claude (cloud-based AI) and Ollama (self-hosted AI), combined with Prometheus, Grafana, Loki, Tempo, and OpenTelemetry, enable us to create an intelligent observability ecosystem. In this hands-on guide, I'll show you step-by-step how to integrate AI with observability platforms. We'll run with various LLM models and compare their results. The integration of AI and observability systems will enable real-time anomaly detection and faster root cause analysis for your systems. I will share with you a comprehensive overview of my experiences with integrating AI and observability systems.

11 min read · Oct 17, 2025

Cumhur M. Akkaya



Topics we will cover:

If you liked the article, I would be happy if you clicked on the clap 👏 button and the Medium Following button to encourage me to write and not miss future

articles.

Your clapping 👏, following, or subscribing helps my articles reach a broader audience.

💙 **Sharing this article with your network, so it reaches more people like you. Thank you in advance for your support.**

For more info and questions, don't hesitate to get in touch with me on **LinkedIn** or **Medium**.

. . .

1. Introduction

We all use AI in our work nowadays. Sometimes, while automatically generating YAML files in CI/CD pipelines, writing Kubernetes manifests, preparing Terraform scripts, or solving a complex error message.
Sometimes, while doing more specific tasks, such as analyzing code and detecting security vulnerabilities (e.g., SQL injection, XSS), writing unit/integration/load test scenarios, finding patterns and anomaly detection (e.g., in logs or user behavior) in databases, generating API documentation and user manuals from existing code, and making AI-based chatbots that reduce customer support burden. We can add many more examples to these.

But AI has now begun to go much further than these. And we are entering a new Age in which those who cannot adapt to this new trend will be subjected to natural selection and will become extinct like the dinosaurs.
“AI Won’t Replace Humans — But Humans With AI Will Replace Humans Without AI” — Harvard Business Review
Of course, this applies not only to people but also to companies.

1.1. Business world and AI

Artificial intelligence is bringing in a new revolution in the business world. Companies that fail to adapt their departments, tools, and services to AI will face a downfall even faster than Nokia, which was the telecommunications giant of the 1990s (at one point, one out of every two phones sold worldwide was the Nokia brand). Nokia missed a huge opportunity by insisting on Symbian instead of adapting to the rise of Google’s Android operating system and by failing to recognize the Touchscreen Revolution in time. In the end, in 2013, it was forced to sell its mobile phone division to Microsoft. Nokia sold its mobile division to Microsoft for $7.2 billion. It had once been worth over $250 billion (1). This tragic story remains a striking example of how even a market leader in the technology industry can become fragile when it fails to adapt to change.

. . .

1.2. AI-Powered Observability

Of course, one of my most important goals in every project is to create an ecosystem that is fully compatible with AI and to utilize its full power.

Most recently, I did this by integrating the observability processes of a project I was working on with AI.

We all know that **monitoring**, **APM** (Application Performance Monitoring), or more broadly, **observability**, is critical for accurately assessing system health and performance in modern software and infrastructure systems. It is our greatest helper in finding answers to many questions about our running system (VMs, Nodes, Containers, Applications, Databases, and others).

It is our greatest helper in finding answers to many questions about our running system. Picture source: (2)

I know colleagues at big companies who focus solely on this task. However, searching for values across dozens of dashboards, monitoring real-time metrics for hundreds of VMs, nodes, or containers, and analyzing log and trace data can create significant challenges and complexity. As the system grows, obtaining accurate and timely data using traditional methods becomes nearly impossible, and understanding the causes of errors and resolving performance issues can take hours.

1.3. Open-source Observability Stack

Open-source Observability Stack that we'll use throughout the hands-on exercises.

I typically have a comprehensive open-source observability system built with **Prometheus**, **Grafana**, **Loki**, **Tempo**, **OpenTelemetry**, **PostgreSQL Exporter** (or other **DB agents**), **Node Exporter**, **cAdvisor**, **Filebeat**(or **Promtail**), and **Alert Manager**. I monitor logs, traces, and metrics for all resources in my system (VMs, nodes, containers, databases, application backends, etc.) on the dashboards I've set up. I can send alerts to email and Slack channels when thresholds are exceeded using alert rules. I've also implemented the necessary authentication and authorization for users who need to access this Observability system. However, although excellent, dealing with the challenges inherent in traditional observability systems, as I mentioned above, is often annoying and time-consuming.

AI Stack that we'll use throughout the hands-on exercises.

However, when I integrated these systems with AI, I saw that all these activities began to become dramatically easier. AI detects and prioritizes critical issues in advance, meaningfully summarizing the performance of hundreds of VMs, containers, and nodes in a single console by automatically analyzing real-time metrics and logs. Instead of getting lost in complex dashboards, AI provided me with clear and meaningful insights, quickly uncovering the root causes of errors and recommending proactive actions. This made monitoring, analysis, and optimization processes both faster and more reliable.

It provided surprisingly accurate answers to my questions. It not only answered them, but also listed, explained the reasons, assessed whether there were any anomalies, compared them with other services/containers, and so on, all without error.

Which container consumed the most RAM in the last hour in the production environment? Which applications have you detected anomalies in CPU and RAM usage in the production environment in the last month?

Claude AI-powered observability system's answer to my question.

It can provide you with not only observability but also any system-related information:
When you asked, "*Which VM is my cAdvisor application running on?*", it provided detailed information about both the VM and the cAdvisor container. It also runs as a system inventory.
It provides surprising recommendations on how to improve your system and its shortcomings.

This transformation not only increased operational efficiency but also made decision-making processes more informed and strategic. Without AI integration, these processes are not only inefficient but also prone to human error, reactive, and costly. For example, missing critical alerts that appear before the problem, being overwhelmed by false alarms, or making the wrong optimization decisions are inevitable.

2. Content of Hands-On and Articles

"*Observability 3.0: AI-Powered APM = Claude (cloud-based) / Ollama (self-hosted) + MCP Server* + Grafana Tool Server + *Prometheus, Grafana, Loki, Tempo, OpenTelemetry, PostgreSQL Exporter, Node Exporter, cAdvisor, Filebeat, Alert Manager — A Comprehensive Hands-On Guide for Live Monitoring with LLMs (Large Language Models)*"

All of them are open source and free (Claude's free use will be sufficient for us). You can run this stack either in the cloud (using Claude for cloud-based AI) or a self-hosted LLM server (on the Ollama runtime environment with models such as Llama 3, Mistral, Gemma, GPT-OSS, or Phi-4).

I will publish it in 3 parts so that the articles are not too long.

Introduction Observability 3.0: AI-Powered APM = AI Stack + Observability Stack (Part-1)

In the first article of this series, I'll discuss the AI and Observability tools and system architecture we'll use.

The AI and Observability tools that we'll use throughout the hands-on exercises.

First, I will briefly introduce **Prometheus**, **Grafana**, **Loki**, **Tempo**, **OpenTelemetry**, **PostgreSQL Exporter (or other DB agents)**, **Node Exporter**, **cAdvisor**, **Filebeat**(or **Promtail**), and **Alert Manager**, which we'll use as observability tools throughout the article series.

Then, I will briefly introduce **Ollama**, **Claude AI**, **Open WebUI**, **MCP**, **n8n**, **Grafana Tool Server**, which we'll use as Artificial Intelligence(**AI**) tools throughout the article series, and **LLAMA**, **GPT OSS**, **Gemma**, **Mistral AI**, and **Phi-4**, which we'll use as Large Language Models (**LLMs**) throughout the article series.

Then, we will prepare the test environment that we will use on the Ubuntu server. We'll deploy and test our sample application, consisting of nine microservices, which we'll use throughout the hands-on exercises. We'll also explore our GitHub repo for this hands-on.

Finally, we will run **Observability Stack** and check whether it monitors the microservices application and VMs.

We will check the metric of running **containers** via cAdvisor on the Grafana dashboard, as seen in the figure below.

cAdvisor metrics

We will check the metrics of running **VMs** via Node Exporter on the Grafana dashboard, as seen in the figure below.

Node Exporter metrics

We will check the metrics, traces, and logs of running **applications** via Loki, Tempo, and OpenTelemetry on the Grafana dashboard, as seen in the figure below.

Loki logs, Tempo traces, and OpenTelemetry metrics.

We will check the metrics of running **databases** via PostgreSQL Exporter on the Grafana dashboard, as seen in the figure below. Postgres Exporter is an agent that allows us to transfer status and performance information from the PostgreSQL

database to Prometheus. This allows us to monitor most performance-related data within the database (3).

PostgreSQL Exporter metrics

We have seen that the applications and observability systems are currently working successfully. We can now observe the VMs, applications, and Databases.
In the following sections, we will integrate this system with Artificial Intelligence in various ways.

. . .

Observability 3.0: AI-Powered APM = Ollama (self-hosted) + GrafanaToolServer + Observability Stack — A Comprehensive Hands-On Guide for Live Monitoring with LLMs (Large Language Models) (Part-2)

In the second article of the series, we'll create a VM (self-hosted LLM server) and run Olloma as a runtime environment within it. We'll install Olloma on Ubuntu, both manually and with Docker (5). We'll examine the differences and limitations between running in a Docker container and running a manual installation on Ubuntu.

Then, we'll install Open Web UI to interact with Ollama via a graphical user interface. Open WebUI makes it easy to connect and manage your Ollama instance. We'll access it at localhost:3000. You can also use Open Web UI to access all AI models, including ChatGPT, Claude, and Gemini, from one self-hosted interface. (4)

We'll download and connect models such as llama3, mistral, gpt-oss, phi4, and gpt-oss on this self-hosted LLM server we created.

Next, we'll run the models on Olloma via Open Web UI.

Then, we'll connect Grafana and Open Web UI using the Grafana Tool Server. After testing this connection using various methods, we will make the necessary settings on Open Web UI and the Grafana Tool Server to ensure stable system operation. Then, we'll implement best practices regarding system security.

Self-hosted LLM+Ollama AI-powered observability system's answer to my question.

Finally, we'll run various queries on the models we downloaded in Ollama to examine the AI system's responses. We'll compare the results by testing various models.
Finally, we'll clean up the system and make a general assessment.

. . .

Observability 3.0: AI-Powered APM = Claude (cloud-based) + MCP Server + Observability Stack — A Comprehensive Hands-On Guide for Live Monitoring with LLMs (Large Language Models) (Part-3)

Claude AI-powered observability system's answer to my question.

In the third article of the series, we will first install Claude on our local computer.

Before running Claude Desktop, we will make the necessary adjustments in Claude Desktop's configuration files to connect to our MCP server and Observability system.

Next, we will connect the model we're running to our Observability system via the MCP server. After testing this connection using various methods, we will make the necessary adjustments to Claude Desktop and the MCP server to ensure stable system operation.

We will then implement best practices regarding system security.

Claude AI-powered observability system's answer to my question.

Finally, we will run various queries and tests through Claude Desktop to examine the responses given by the AI system. We will compare the results of the Claude Opus 4.1 and Claude Sonnet 4 models.
Finally, we will clean up the system and make a general assessment.

All of them are open source and free (Claude's free use will be sufficient for us). You can run this stack either in the cloud (using Claude for cloud-based AI) or a self-hosted LLM server (on the Ollama runtime environment with models such as Llama 3, Mistral, Gemma, GPT-OSS, or Phi-4).

We will do these practically, step by step, in the next articles. Stay tuned!

Happy Clouding…

I hope you enjoyed reading this article. Don't forget to follow **my LinkedIn** or **Medium** account to be informed about new articles.

Your clapping 👏, following, or subscribing helps my articles reach a broader audience.

💙 **Sharing this article with your network, so it reaches more people like you. Thank you in advance for your support.**

. . .

🔗3. References

(1) https://medium.com/@celestineriza/how-a-300b-giant-blew-it-all-9e3dd915f487
(2) https://grafana.com/docs/tempo/latest/introduction/telemetry/
(3) https://github.com/prometheus-community/postgres_exporter
(4) https://docs.openwebui.com/
(5) https://ollama.com/

. . .

Learn how to build AI-powered observability with Claude, Ollama, Prometheus, Grafana, Loki, Tempo, and OpenTelemetry in a hands-on guide.

#Observability #Observability3 #AIMonitoring #ClaudeAI #Ollama #SelfHostedAI #MCPServer #Prometheus #Grafana #Loki #Tempo #OpenTelemetry #PostgreSQL #PostgreSQLExporter #NodeExporter #cAdvisor #Filebeat #AlertManager #DevOps #AIDevOps #LLM #LLMMonitoring #CloudObservability #AIObservability #RootCauseAnalysis

Ai Powered Monitoring | Self Hosted Ai | Ollama | Claude Ai | Mcp Server

Follow

Written by Cumhur M. Akkaya

2.3K followers · 23 following

✦Multi-Cloud & DevOps Engineer, ✦Technical Writer, ✦AWS Community Builder ✦LinkedInCloudTop Voice
✦Believes in learning by doing ✦ linkedin.com/in/cumhurakkaya/

No responses yet

Bgerby

What are your thoughts?

More from Cumhur M. Akkaya

Cumhur M. Akkaya

Mastering Keycloak: A Step-by-Step Comprehensive Tutorial to Modern Identity and Access Management

Secure identity and Access Management is the foundation of digital transformation. Open-source Keycloak simplifies IAM process. Modern IAM.

Jul 1 105

Cumhur M. Akkaya

Developing a Scalable AI Chatbot Using Azure OpenAI (LLM provider), Java Microservices, and MySQL...

Step-by-step guide to integrating an AI chatbot into a Java-based microservices app with MySQL using Azure OpenAI as the LLM provider.

Sep 4 77 2

Cumhur M. Akkaya

Real-Time Change Data Capture (CDC) Tutorial: How to Stream PostgreSQL Changes to a Kafka Cluster...

Learn ow to set up real-time Change Data Capture (CDC) with PostgreSQL, Apache Kafka, and Debezium. Stream database changes to Kafka...

Jul 28 55 1

Cumhur M. Akkaya

GitLab CI/CD - 1 : Building a Java Project using Maven and Docker within the GitLab CI pipeline.

In this article, we will create a GitLab CI (continuous integration) pipeline, using Maven and Docker. We'll use the Java application as...

Jun 24, 2023 104 4

See all from Cumhur M. Akkaya

Recommended from Medium

In AWS Tip by Osman ALP

Mastering Kubernetes Observability: Deploying Grafana with Loki, Tempo, Mimir, and Alloy

The Modern Observability Trinity: Metrics, Logs, Traces, and Telemetry Collection

Oct 13

Abhijeet Yadav

Stop Updating YAMLs Manually: How I Automated Kubernetes Deployments with Argo CD Image Updater

Modern DevOps teams strive for speed, reliability, and automation in their deployment pipelines. In Kubernetes environments, Argo CD is the...

6d ago 2

In AWS in Plain English by Aman Pathak | DevOps | AWS | K8s | Terraform

AWS Outage Root Cause Revealed: How a DNS Mismatch Affected DynamoDB and Core Services

Discover what caused the October 2025 AWS us-east-1 outage—a DNS mismatch that broke DynamoDB and triggered cascading failures across...

4d ago 31 1

In Data, AI and Beyond by Julius Nyerere Nyambok

Generate AWS architectural diagrams using this simple method

A combination of LLMs and MCP Servers.

Oct 14 52 1

Gokul Srinivas

Top 7 Kubernetes v1.34 Features You Should Actually Care About

As Lead Kubernetes Engineer's Take on What Matters (and What Doesn't)

Oct 15 4

In Stackademic by Mohab AbdelKarim

10 Kubernetes Tools That Finally End YAML Hell in 2025

YAML Fatigue Is Real—Here Are 10 Tools DevOps Engineers Swear By in 2025

Oct 16 40

See more recommendations