Agentic AI for Autonomous Infrastructure Observability and Control

Approach

As data centers scale, legacy infrastructure management becomes reactive, fragmented, and labor-intensive, leading to inefficiency, human error, and missed optimizations. Our approach was to build a modern, agentic, AI-powered platform that combines continuous observability with autonomous, intelligent control.

- Hardware telemetry is ingested and analyzed for anomalies, faults, and inefficiencies
- An LLM-powered agent (with LangChain + Gemini) interprets, reasons, and acts. It's able to trigger remediation and optimizations with minimal human intervention
- Observability is not just for monitoring, but enables proactive, autonomous action

Architecture

The platform uses a modular design enabling real-time monitoring, data analysis, and automated control.

1. Redfish API (Mock)

- Simulates Baseboard Management Controller (BMC) hardware telemetry (fan speed, power, voltages, etc.) and supports control commands
- Fully developed from scratch for this project

2. FastAPI Backend

- Periodically fetches telemetry from the Redfish mock API
- Implements signal classification and data storage (MongoDB)
- Streams real-time logs (via Server-Sent Events or polling) to the frontend
- Exposes /metrics endpoints for Prometheus scraping

3. Prometheus & Grafana

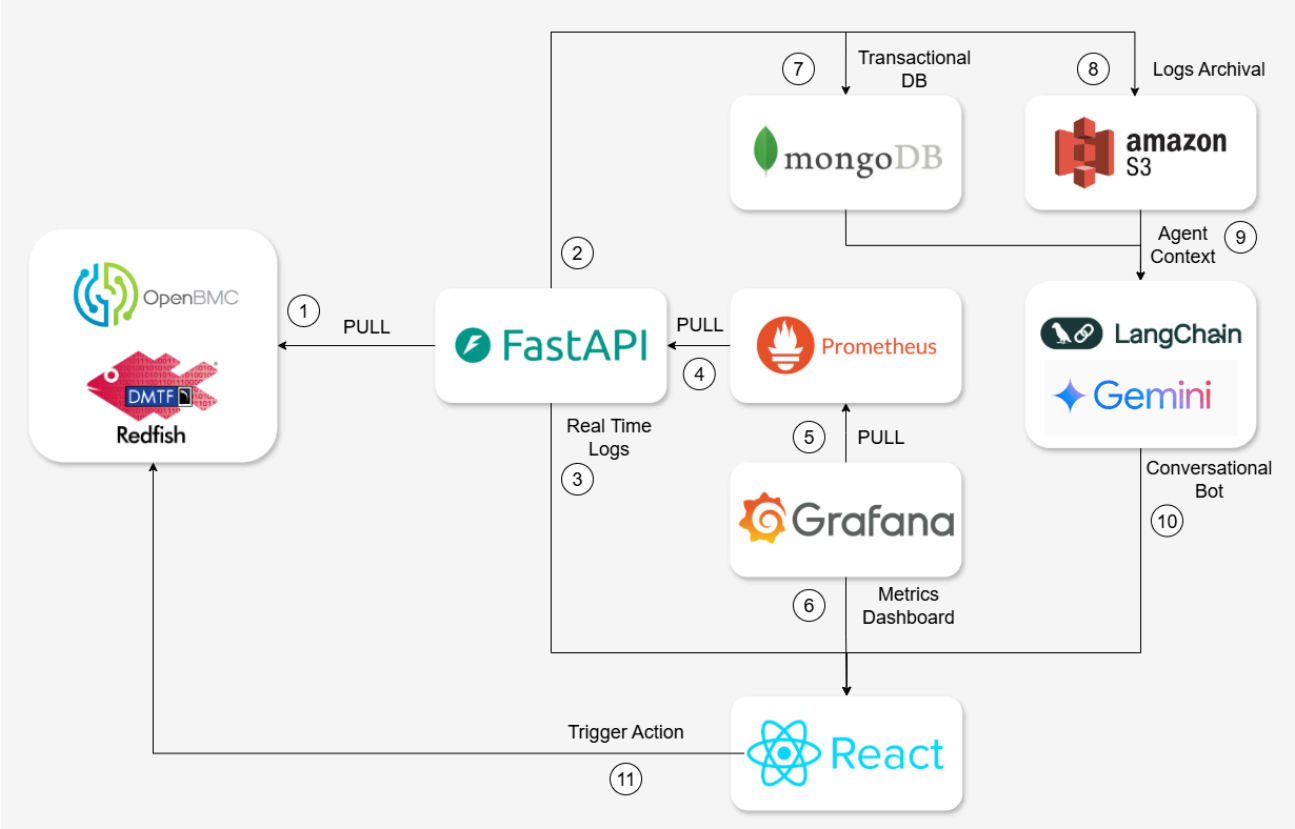
- Prometheus scrapes FastAPI's metrics endpoints every 5 seconds
- Grafana visualizes this data, with dashboards embedded into the React frontend

4. MongoDB & S3

- MongoDB stores active & historical telemetry for fast and archival queries

- S3 is used for storing large log files and telemetry snapshots

5. LangChain + Gemini (AI Layer)
   - Handles two query types: INFERENCE (e.g., “How many faults last hour?”) and ACTION (e.g., “Set PSU 1 threshold to 500W”)
   - Agent can access MongoDB and S3, providing conversational insights or executing control logic
6. React Frontend
   - Chat panel for natural language inference queries and control commands
   - Real-time logs and embedded Grafana dashboards provide unified observability



Control Flow

1. User sends a One-Shot Natural Language query via React UI
2. FastAPI processes and performs context aware routing (INFERENCE or ACTION) to the respective Agents
3. Gemini LLM interprets and triggers data queries or hardware actions
4. UI reflects updates in real time via logs and dashboards

Credit Assignment

Aditya Dawadikar

- **Architecture:** Designed the Architecture and handled Team coordination
- **Agentic AI:** Designed and Implemented Langchain based AI agents for Context Aware Routing, Inference and RedFish API Triggering logic
- **Visualization:** Setup Prometheus-Grafana dashboards by scraping RedFish APIs
- **Logging:** Implemented RealTime Log streaming and Event Sourcing solution for Agent events

Udayan Atreya

- **Backend:** Developed FastAPI backend for conversational chat integration.
- **Anomaly Detection:** Analysis of raw telemetry data and classification with reasoning for summarization on MongoDB.
- **Data Management:** MongoDB, S3 setup and integration for Data Persistence and Scalability.
- **Archival:** Developed end-to-end pipeline for Archived logs Querying

Sohan Vallapureddy

- **Chatbot Integration and UI:** Integrated secure, context-aware LLM conversations with both real-time and historical telemetry data
- **Backend and API endpoints:** Enhanced the FastAPI backend by building robust endpoints for Redfish API queries and data aggregation
- **Authentication and Security:** Developed and enforced the authentication framework supporting secure admin login and protected AI endpoints
- **Documentation:** Created and organized project documentation and report, ensuring all deliverables were clear, well-structured, and professionally formatted

Harshavardhan Valmiki

- **Data Ingestion:** Generated event summaries using Gemini LLM from raw Real Time data and ingested to S3 and MongoDB
- **Context Engineering:** Generated query response using condensed knowledge from historical data from MongoDB and S3
- **Prompt Engineering**- Engineered prompts so that agent can dynamically choose between MongoDB for summaries and S3 for detailed telemetry based on query intent.

Collaboration highlights: All team members participated in integration, API standardization, code reviews, documentation, and demo preparation.