README.md

Observability

In this architecture, **each microservice plays a dual role** in enhancing observability by sending both traces and metrics to the monitoring stack. Each microservice is configured to generate and send distributed traces to Grafana Tempo, a purpose-built tool for tracing within distributed systems. Tempo collects these traces to provide an end-to-end view of how requests traverse the entire system, highlighting latencies, bottlenecks, and errors across various microservices.

In addition to collecting traces, Tempo is configured to process this trace data and derive key insights, which it writes as **custom metrics to Prometheus**. These metrics enable quantitative tracking of critical parameters such as request durations, error rates, and throughput. By writing derived metrics into Prometheus, Tempo bridges the gap between tracing and system-wide monitoring.

Simultaneously, each microservice exposes its own **Prometheus metrics endpoint**, providing granular visibility into its internal performance. This includes metrics like request counts, response times. Prometheus scrapes these metrics directly from the microservices at regular intervals, ensuring real-time visibility into their health and performance.

Metrics

Numerical measurements that represent the performance or behavior of a system over time. They are lightweight, structured, and aggregated data points that help monitor the health and efficiency of application. They are stored as time-series data (values measured at specific intervals) in **Prometheus** container

Traces

Capture the flow of requests through a distributed system, showing how different components (microservices) interact with one another to fulfill a user request. A trace represents the end-to-end execution of a single transaction, request, or workflow. Provides context and relationships between services, also allowing to identify bottlenecks and latencies.

Logs

Semi-structured (`json` based) records that describe events within a system and provide granular insights into what happened at specific points in time. Currently captured are logs from `App_Proxy` only.

Relationship Between Metrics, Traces, and Logs

- **Metrics** give a high-level view of system performance.
- **Traces** provide insights into the flow of requests and service interactions when an issue arises.
- **Logs** give contextual information about specific events or errors.