Observability with OpenTelemetry

A comprehensive guide to understanding observability principles and implementing OpenTelemetry in modern distributed systems.

By **Naga Murali Krishna Koneru**, Technical Architect

Agenda

Understanding Observability

Foundations and key concepts

Observability In-Depth

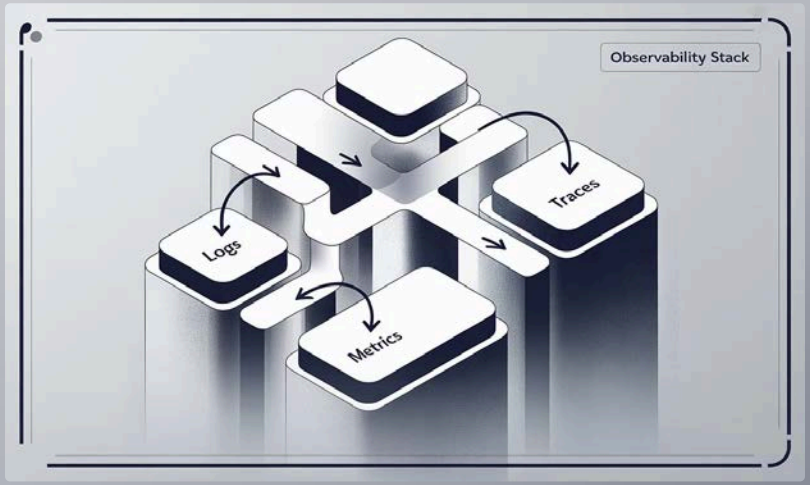
Telemetry data types and collection methods

OpenTelemetry Essentials

Architecture, components, and implementation

Data Sources & Integration

Connecting OpenTelemetry with your ecosystem



Foundations of Observability

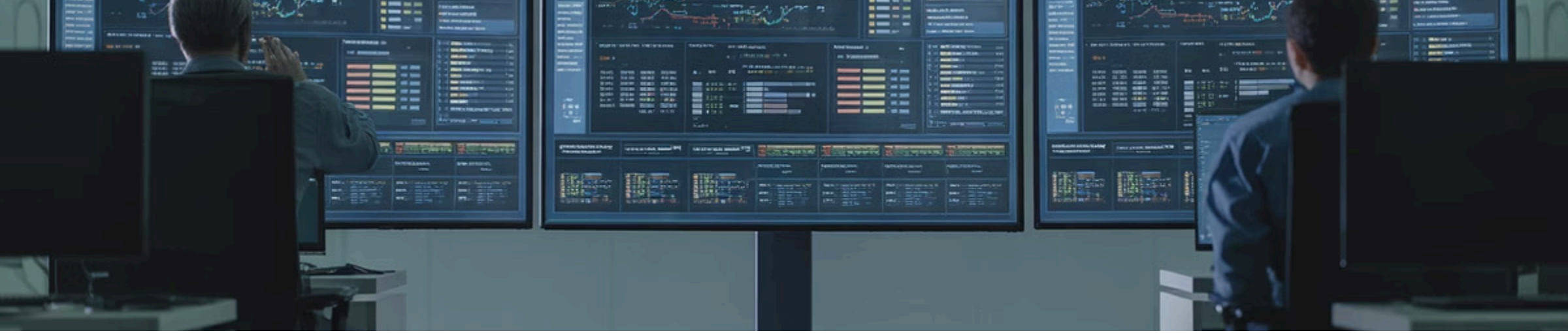
Telemetry Data

- The automated collection and transmission of measurements, logs, and events from remote or distributed systems to a central monitoring system for analysis
- The collected monitoring data used to locate potential problems in your systems.

Key DevOps Metrics

- Mean Time to Detection (MTTD)
- Mean Time to Resolve (MTTR)

These metrics measure how quickly teams identify and resolve issues.



Monitoring Methods

RED Method

- Rate (requests per second)
- Errors (failed requests)
- Duration (response time)

USE Method

- Utilization (resource usage)
- Saturation (queue length)
- Errors (hardware/software)

Four Golden Signals

- Latency (processing time)
- Traffic (request volume)
- Errors (failure rate)
- Saturation (system load)

Core Web Vitals

LCP

Largest Contentful Paint measures loading performance.

FID

First Input Delay measures responsiveness to user interactions.

CLS

Cumulative Layout Shift measures visual stability during loading.

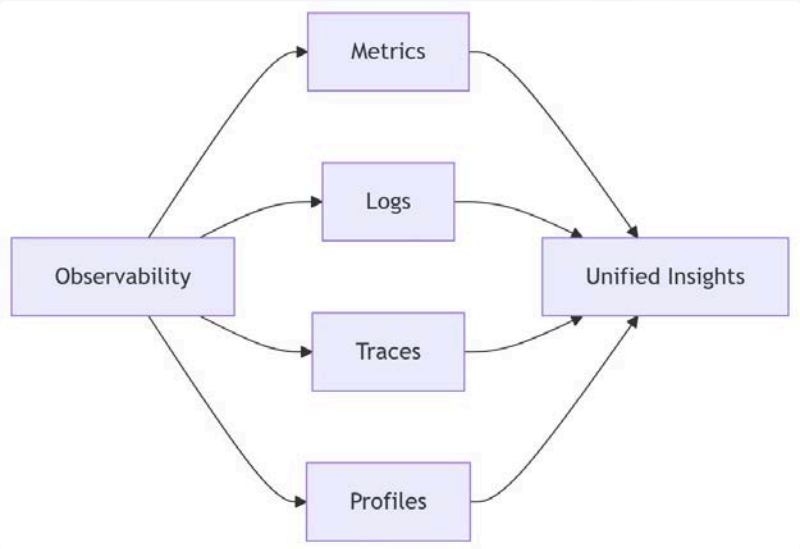
Monitoring vs. Observability

Monitoring

- Reactive approach
- Tracks known issues
- Best for monolithic systems
- Shows when issues occur

Observability

- Proactive approach
- Uncovers "unknown unknowns"
- Ideal for complex systems
- Shows where, when, and why



Telemetry Data Types

Metrics

Quantitative measurements that answer "What is happening?"

Logs

Time-stamped event records that answer "What happened?"

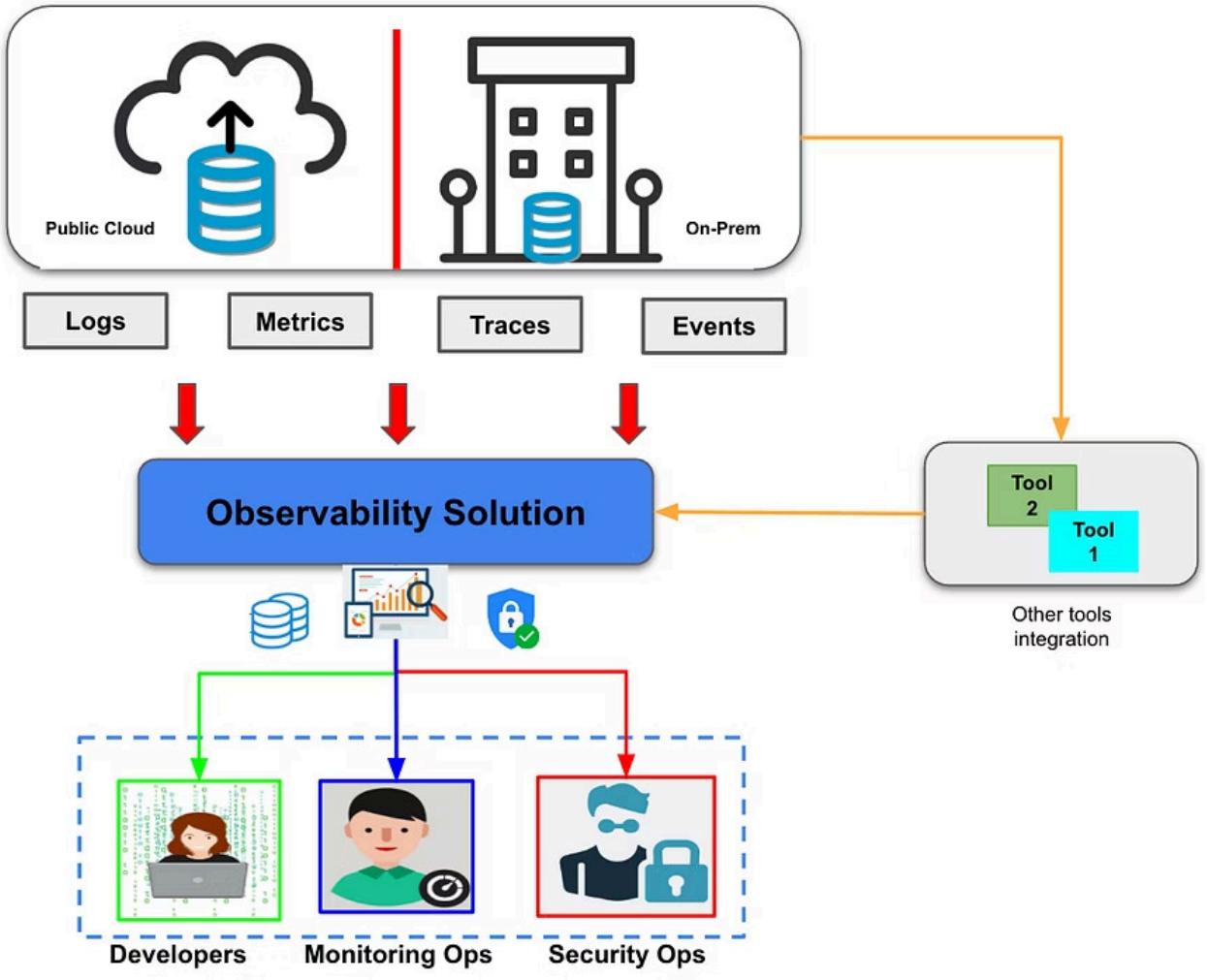
Traces

End-to-end request journeys that answer "Where is the bottleneck?"

Profiles

Resource-level diagnostics that answer "Why is it slow?"

Observability in Modern Systems

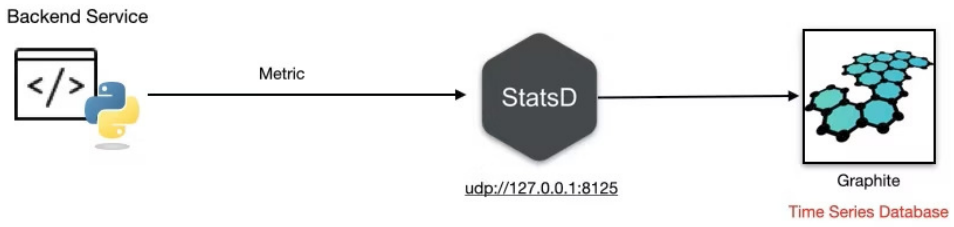


Metric Collection Methods

Push Method

Applications actively send metrics to collection endpoints via TCP/UDP.

Example: Application pushes metrics to StatsD, which forwards to Graphite.



Scrape Method

Applications expose metrics for time-series databases to collect.

Example: Prometheus scrapes metrics from application endpoints.

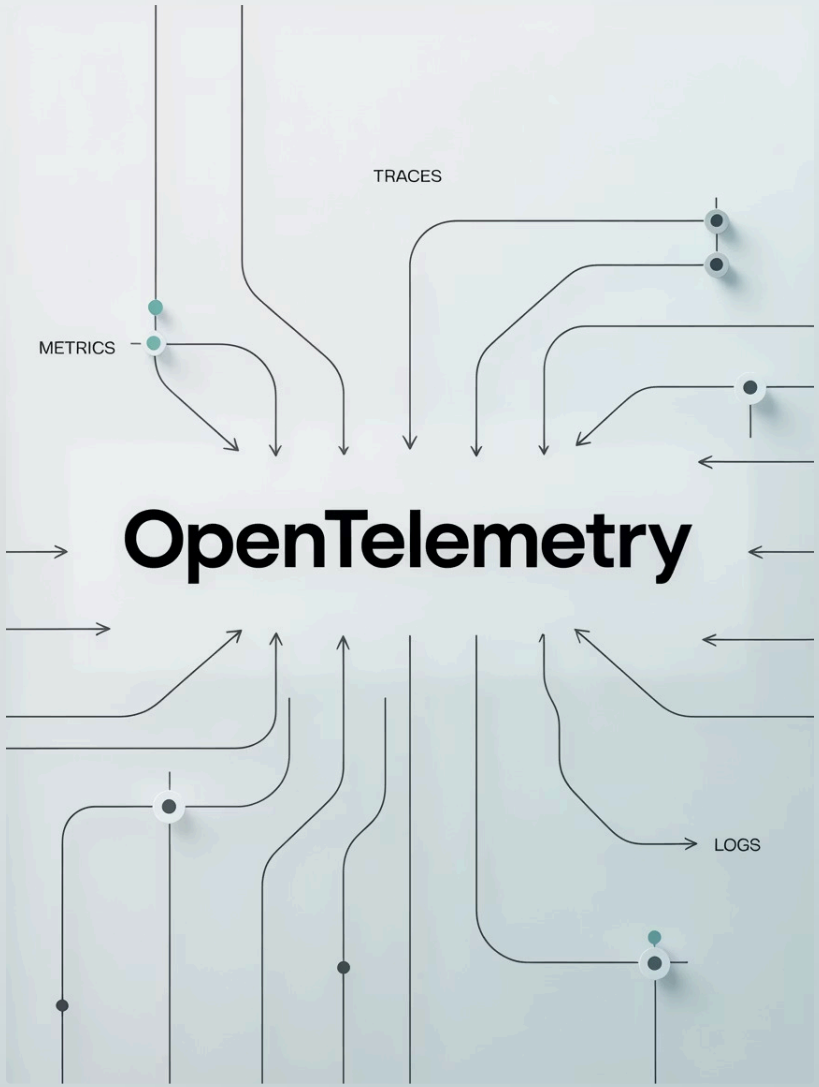
Choosing Your Collection Method

Choose Scrape When

- Using Kubernetes or dynamic environments
- Preferring centralized control
- Apps can expose HTTP endpoints

Choose Push When

- Apps are short-lived
- Real-time metrics are needed
- Apps run in restricted networks



Introduction to OpenTelemetry

Open-Source Framework

Standardizes generation, collection, and management of telemetry data.

CNCF Project

Incubated under the Cloud Native Computing Foundation.

Key Benefits

Vendor-neutral, cross-language support, standardization.

Comprehensive Coverage

Supports logs, metrics, and traces in one framework.

OpenTelemetry Architecture

Instrumentation Libraries

Enhance applications to generate telemetry data.

Collectors

Receive, process, and export telemetry data.

OTLP Protocol

Standardized protocol for transmitting telemetry data.

Receivers

Collect telemetry data from various sources.

Processors

Manipulate and transform data before exporting.

Exporters

Send processed data to observability backends.

OpenTelemetry Collector

Central executable

Receives, processes, and exports telemetry data to multiple targets.

Protocol support

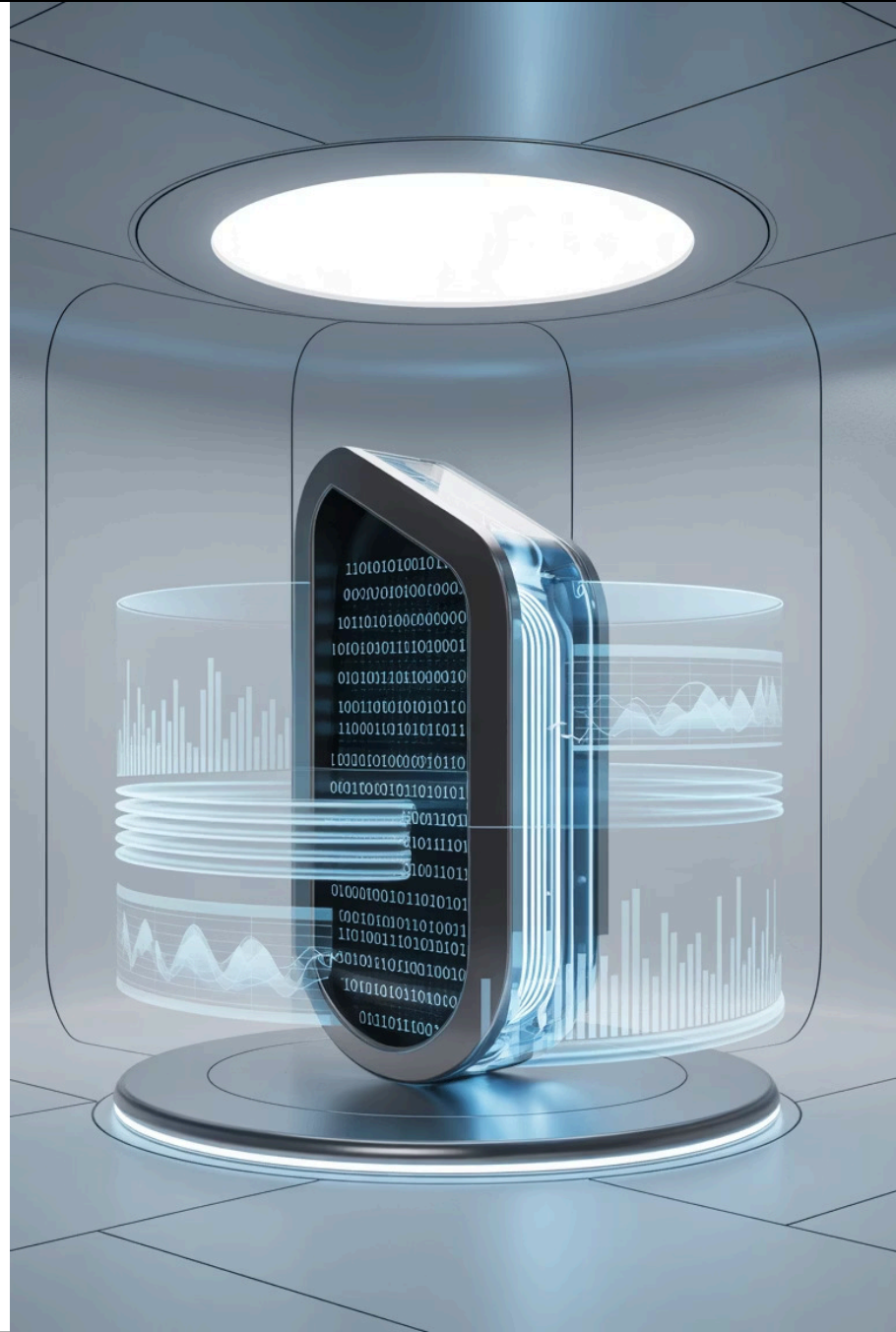
Works with popular open-source telemetry protocols.

Intermediary role

Acts between applications and backend analysis tools.

Key advantages

Reduces resource consumption, centralizes configuration, improves security.



Collector Components

Receivers

Entry points for telemetry data, accepting various formats.

Processors

Manipulate data before export: filtering, enriching, sampling.

Exporters

Send processed data to backend systems for analysis.

Advanced Collector Components

Connectors

- Facilitate communication between pipelines
- Transform between signal types
- Enable multi-stage processing flows

Extensions

- Enrich component capabilities
- Add performance analysis
- Provide authentication services

Collector Pipelines

Receivers

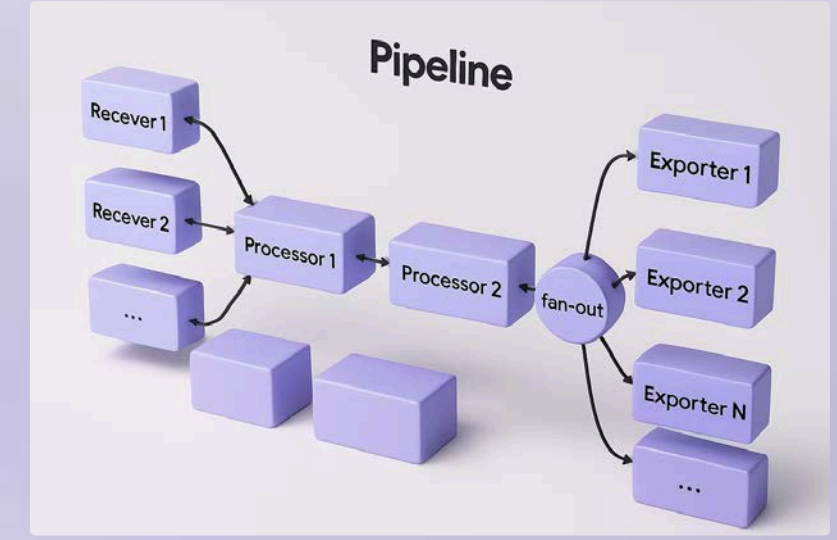
Listen on network ports to collect incoming data.

Processors

Modify, filter, enrich, aggregate data to optimize storage, reduce costs, and enhance observability.

Exporters

Send processed data to various backend systems.



OpenTelemetry Data Sources

Application Instrumentation

Apps using OpenTelemetry SDKs to send telemetry data.

Service Mesh

Traffic metrics and traces from Istio or Linkerd.

Node-level Metrics

Data from Kubelet about nodes and running pods.

Kubernetes Events

Cluster events providing insights into system activities.

Additional OpenTelemetry Sources

Logging Daemons

Fluentd and Fluent Bit collect and forward logs.

Cloud Provider Metrics

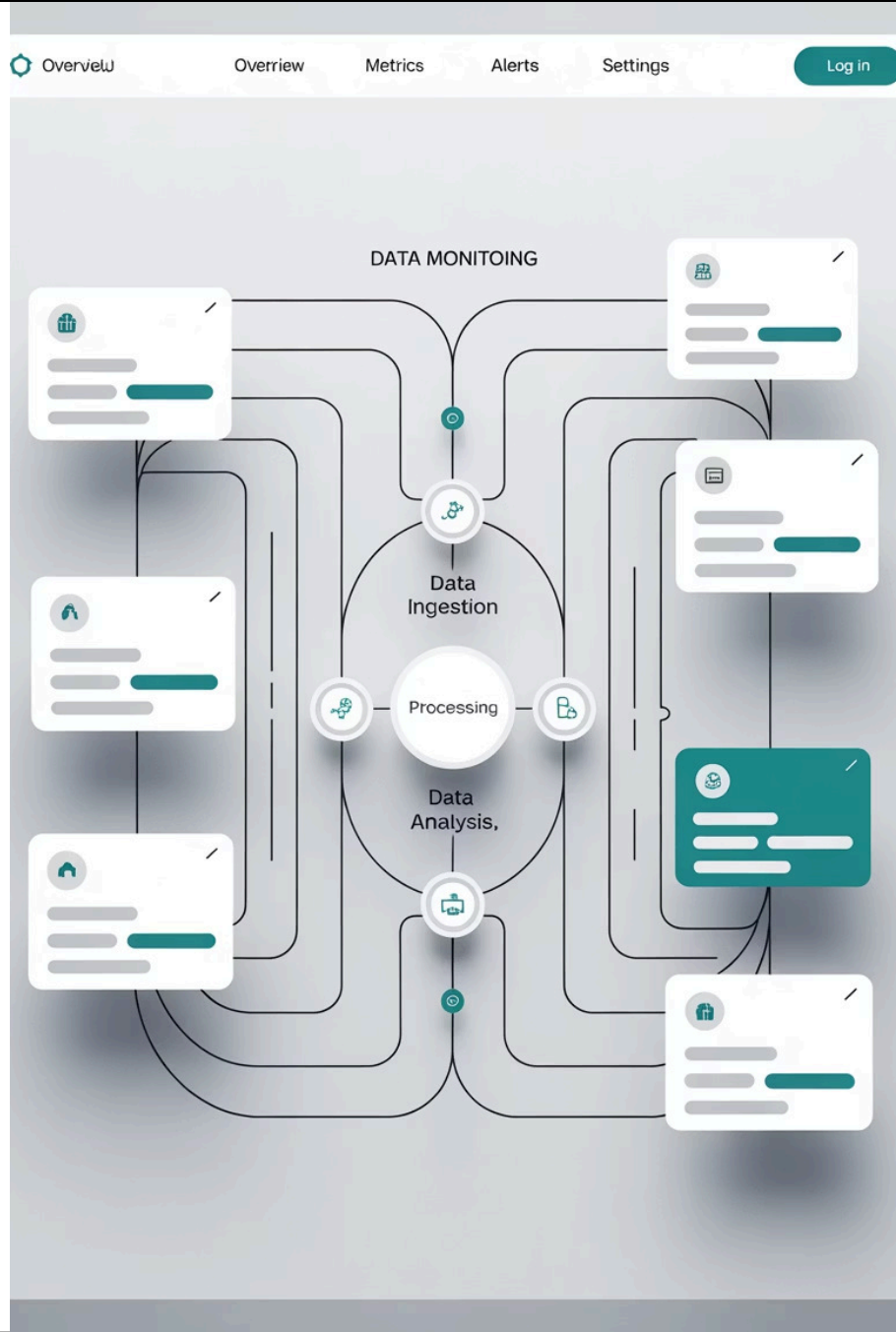
Data from AWS, GCP, or Azure services.

Probes and Health Checks

Liveness and readiness status data.

Container Runtime

Metrics about container states and resource usage.



Implementation Approaches

Auto Instrumentation

- Immediate visibility
- Broad coverage
- Minimal development effort
- Great starting point

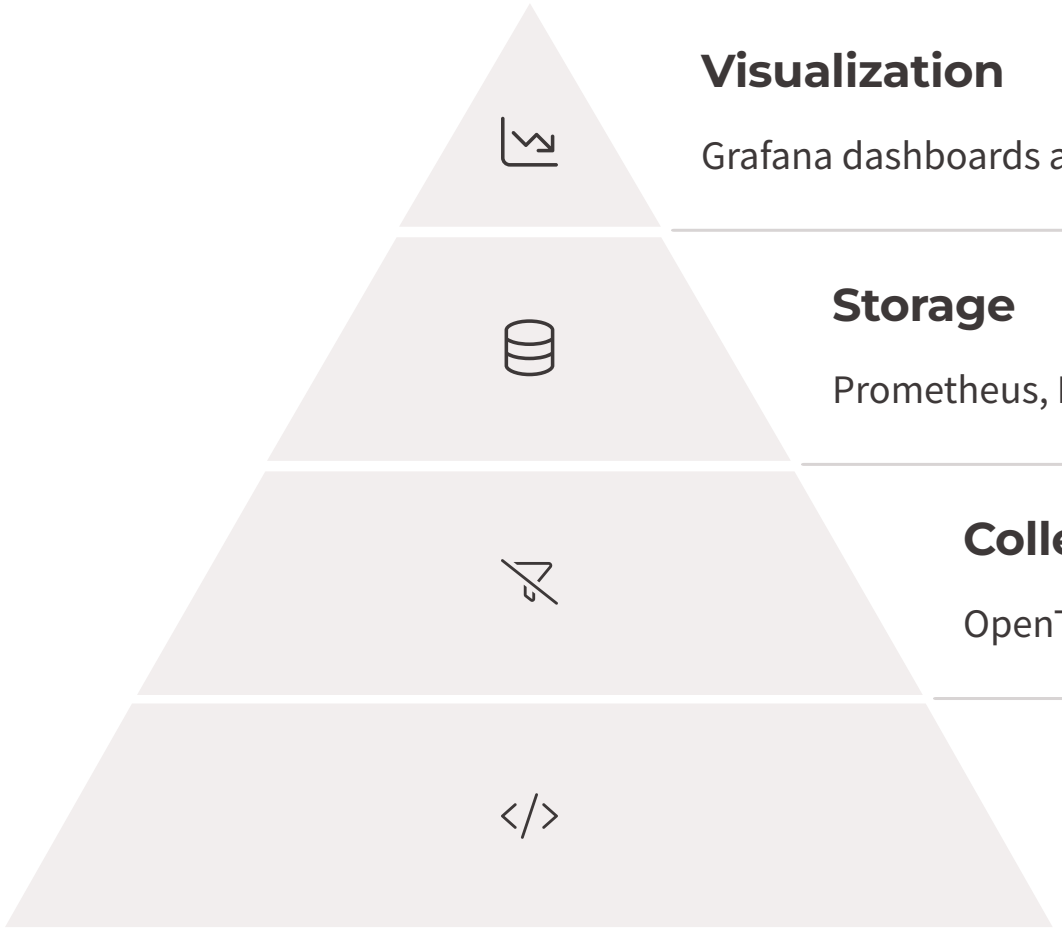
Automatically instruments common libraries without extensive code changes.

Manual Instrumentation

- Custom metrics
- Precise control
- Business-specific insights
- Greater flexibility

Adding specific code to capture exactly what you need, when you need it.

Full Observability Stack



Visualization

Grafana dashboards and alerts

Storage

Prometheus, Loki, Tempo, Mimir

Collection

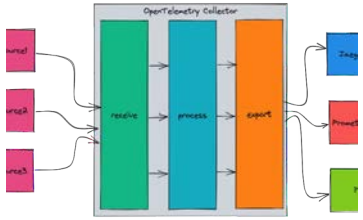
OpenTelemetry, Fluent Bit

Instrumentation

Application code with OTel SDKs

Observability Stack Components

Data flows through an integrated telemetry pipeline



Collector

Collects and forwards telemetry data from various sources.

Tempo

Distributed tracing system for request journey analysis.

Loki

Log aggregation platform for centralized log management.

Prometheus

Time-series database for metrics collection and storage.

Grafana

Web UI for visualization, dashboards, and alerting.

Each component handles a specific telemetry type, creating a comprehensive observability solution.

Integrated Observability Ecosystem

4

Key Components

Grafana, Loki, Mimir, and Fluent Bit form the core stack.

3

Signal Types

Metrics, logs, and traces provide complete visibility.

1

Unified Platform

OpenTelemetry standardizes all telemetry collection.