

**1. Chức năng của từng thành phần**

**Applications**

* **Chức năng**: Đây là các ứng dụng hoặc dịch vụ (microservices, monolithic apps, v.v.) cần được giám sát. Chúng có thể được viết bằng nhiều ngôn ngữ (Java, Python, Go, v.v.) và chạy trên các nền tảng khác nhau (Kubernetes, VMs, bare-metal).
* **Vai trò**: Tạo ra dữ liệu telemetry (metrics, traces, logs) thông qua các công cụ như OpenTelemetry SDKs hoặc auto-instrumentation.

**OpenTelemetry SDKs**

* **Chức năng**: Cung cấp API và SDK để instrument ứng dụng, thu thập dữ liệu telemetry (metrics, traces, logs). Auto-instrumentation hoặc manual instrumentation có thể được sử dụng để giảm thiểu thay đổi mã nguồn.
* **Vai trò**: Gắn mã vào ứng dụng để ghi lại thông tin về hiệu suất (e.g., thời gian xử lý request, lỗi) và gửi dữ liệu này đến OpenTelemetry Collector.
* **Ví dụ**: Một ứng dụng Java sử dụng OpenTelemetry Java SDK để ghi lại thời gian phản hồi của API và gửi dưới dạng OTLP (OpenTelemetry Protocol).

**OpenTelemetry Collector**

* **Chức năng**: Một dịch vụ trung gian thu thập, xử lý, và xuất dữ liệu telemetry từ nhiều nguồn (ứng dụng, cơ sở hạ tầng) đến các backend như Prometheus, Loki, hoặc Tempo.
* **Vai trò**:
  + **Thu thập**: Nhận dữ liệu từ OpenTelemetry SDKs qua giao thức OTLP (gRPC/HTTP).
  + **Xử lý**: Lọc, nhóm, hoặc làm giàu dữ liệu (e.g., thêm metadata như môi trường hoặc vùng).
  + **Xuất**: Gửi dữ liệu đến các hệ thống lưu trữ phù hợp (Prometheus cho metrics, Loki cho logs, Tempo cho traces).
* **Ví dụ**: Collector nhận metrics từ ứng dụng, nhóm chúng theo lô (batch) để tối ưu, và gửi đến Prometheus.

**Zabbix Agents**

* **Chức năng**: Các tác nhân (agents) được cài đặt trên máy chủ hoặc thiết bị để thu thập metrics cơ sở hạ tầng (CPU, RAM, disk, network) và gửi về Zabbix Server.
* **Vai trò**: Giám sát các thiết bị vật lý, máy ảo, hoặc container mà không cần instrumentation ứng dụng.
* **Ví dụ**: Zabbix Agent trên một server Linux thu thập CPU usage và gửi đến Zabbix Server qua giao thức Zabbix.

**Zabbix**

* **Chức năng**: Hệ thống giám sát tập trung, lưu trữ và phân tích dữ liệu từ Zabbix Agents hoặc các nguồn khác (SNMP, IPMI, HTTP).
* **Vai trò**:
  + Lưu trữ dữ liệu vào cơ sở dữ liệu (e.g., PostgreSQL, MySQL).
  + Cung cấp API để truy vấn dữ liệu và tạo cảnh báo dựa trên ngưỡng (thresholds).
  + Hỗ trợ giám sát cơ sở hạ tầng truyền thống (servers, network devices).
* **Ví dụ**: Zabbix phát hiện một server có CPU usage > 90% và gửi cảnh báo qua email.

**Prometheus/Loki/Tempo**

* **Chức năng**:
  + **Prometheus**: Lưu trữ và truy vấn metrics thời gian thực (time-series data) từ OpenTelemetry hoặc các nguồn khác.
  + **Loki**: Lưu trữ và truy vấn logs hiệu quả, được tối ưu cho việc tìm kiếm và lọc.
  + **Tempo**: Lưu trữ và truy vấn traces để theo dõi luồng xử lý trong hệ thống phân tán.
* **Vai trò**: Là các backend lưu trữ dữ liệu telemetry từ OpenTelemetry Collector, được tối ưu cho từng loại dữ liệu (metrics, logs, traces).
* **Ví dụ**:
  + Prometheus lưu trữ metrics như số lượng request mỗi giây.
  + Loki lưu trữ logs ứng dụng như lỗi HTTP 500.
  + Tempo lưu trữ traces cho thấy thời gian xử lý của một request qua nhiều microservices.

**Grafana**

* **Chức năng**: Nền tảng trực quan hóa dữ liệu, tích hợp với nhiều nguồn dữ liệu (Zabbix, Prometheus, Loki, Tempo) để tạo dashboard và cảnh báo.
* **Vai trò**:
  + Kéo dữ liệu từ Zabbix (qua API hoặc database) và từ Prometheus/Loki/Tempo.
  + Tạo dashboard hiển thị metrics, logs, và traces trong một giao diện thống nhất.
  + Hỗ trợ phân tích và liên kết dữ liệu (e.g., từ metrics đến traces).
* **Ví dụ**: Một dashboard Grafana hiển thị CPU usage từ Zabbix, số lượng request từ Prometheus, và traces từ Tempo.

**Zabbix API/Database**

* **Chức năng**:
  + **API**: Cung cấp giao diện để Grafana truy vấn dữ liệu từ Zabbix.
  + **Database**: Lưu trữ dữ liệu lịch sử và trends từ Zabbix.
* **Vai trò**: Cho phép Grafana truy xuất dữ liệu Zabbix để hiển thị trên dashboard.
* **Ví dụ**: Grafana sử dụng Zabbix API để lấy dữ liệu về disk usage trong 24 giờ.

**Dashboards (Metrics, Traces, Logs)**

* **Chức năng**: Giao diện trực quan hiển thị dữ liệu từ nhiều nguồn, hỗ trợ phân tích và ra quyết định.
* **Vai trò**: Tổng hợp thông tin từ Zabbix và OpenTelemetry để cung cấp cái nhìn toàn diện về hệ thống.
* **Ví dụ**: Một dashboard hiển thị CPU usage (Zabbix), latency của API (Prometheus), và lỗi trong logs (Loki).

**2. Luồng dữ liệu**

Dữ liệu di chuyển qua các thành phần theo các bước sau:

1. **Từ Applications đến OpenTelemetry SDKs**:
   * Ứng dụng được instrument bằng OpenTelemetry SDKs hoặc auto-instrumentation (e.g., Grafana Beyla).
   * SDKs thu thập:
     + **Metrics**: Số liệu như request rate, error rate, latency.
     + **Traces**: Thông tin về luồng xử lý request qua các dịch vụ (e.g., API call từ service A đến service B).
     + **Logs**: Nhật ký ứng dụng như thông báo lỗi hoặc sự kiện.
   * Dữ liệu được gửi đến OpenTelemetry Collector qua giao thức OTLP (gRPC hoặc HTTP).
2. **Từ OpenTelemetry SDKs đến OpenTelemetry Collector**:
   * Collector nhận dữ liệu OTLP từ nhiều ứng dụng.
   * Collector xử lý dữ liệu:
     + Lọc bỏ dữ liệu không cần thiết (e.g., traces không quan trọng).
     + Nhóm dữ liệu thành lô (batch) để tối ưu truyền tải.
     + Thêm metadata như tên dịch vụ, môi trường (dev, prod).
   * Collector xuất dữ liệu đến các backend:
     + Metrics → Prometheus.
     + Logs → Loki.
     + Traces → Tempo.
3. **Từ Zabbix Agents đến Zabbix**:
   * Zabbix Agents chạy trên các server hoặc thiết bị, thu thập metrics cơ sở hạ tầng (CPU, memory, disk, network).
   * Dữ liệu được gửi đến Zabbix Server qua giao thức Zabbix (thường là TCP port 10051).
   * Zabbix Server lưu trữ dữ liệu vào cơ sở dữ liệu (e.g., PostgreSQL) và cung cấp qua API.
4. **Từ OpenTelemetry Collector đến Prometheus/Loki/Tempo**:
   * Collector gửi dữ liệu đã xử lý đến các backend tương ứng:
     + **Prometheus**: Nhận metrics qua Prometheus Remote Write.
     + **Loki**: Nhận logs qua HTTP API.
     + **Tempo**: Nhận traces qua OTLP.
   * Mỗi backend lưu trữ dữ liệu theo cách tối ưu cho loại dữ liệu của nó (time-series, log indexing, trace indexing).
5. **Từ Zabbix/Prometheus/Loki/Tempo đến Grafana**:
   * Grafana kết nối với các nguồn dữ liệu:
     + **Zabbix**: Qua Zabbix API (sử dụng plugin Grafana-Zabbix) hoặc truy vấn trực tiếp database (PostgreSQL/MySQL).
     + **Prometheus**: Qua HTTP API để lấy metrics.
     + **Loki**: Qua HTTP API để lấy logs.
     + **Tempo**: Qua gRPC/HTTP để lấy traces.
   * Grafana tổng hợp dữ liệu từ các nguồn này và hiển thị trên dashboard.
6. **Từ Grafana đến Dashboards**:
   * Grafana tạo các dashboard hiển thị:
     + Metrics: Biểu đồ CPU usage (Zabbix), request latency (Prometheus).
     + Logs: Danh sách lỗi từ Loki.
     + Traces: Luồng xử lý request qua các dịch vụ từ Tempo.
   * Người dùng tương tác với dashboard để phân tích, liên kết dữ liệu (e.g., click vào metric để xem trace liên quan), và nhận cảnh báo.