

Monitorización y logging en Microservicios

Contenidos

- **Monitorización y Logging** en Microservicios C/C++
- Monitorización del rendimiento en tiempo real:
 - Uso de **Prometheus** y **Grafana** para la recolección y visualización de métricas.
 - Integración con **cAdvisor** para monitorizar contenedores.
- Implementación de tracing distribuido:
 - Uso de **Jaeger** o **OpenTelemetry** para tracing distribuido.
 - Análisis de la latencia y tiempos de respuesta en microservicios C++.
- Gestión de logs en entornos distribuidos:
 - Uso de **Fluentd** y **Elasticsearch** para la centralización de logs.
 - Búsqueda y análisis de logs en sistemas distribuidos."

Monitorización del rendimiento en tiempo real

Herramientas / Métricas

- Uso de Prometheus y Grafana para la recolección y visualización de métricas

Métrica clave	¿Por qué importa?
Tiempo de respuesta	Detecta lentitud o cuellos de botella
Errores HTTP	Identifica fallos en endpoints
Uso de CPU/RAM	Previene saturación del sistema
Disponibilidad	Asegura que el servicio esté online
Latencia de base de datos	Detecta consultas lentas

Prometheus

Introducción

- **Prometheus** es una herramienta de **monitorización y alertas** muy potente, ampliamente usada en entornos DevOps, microservicios y sistemas distribuidos.
- **Es gratuita y de código abierto** bajo licencia Apache 2.0.
- Prometheus es un sistema que **recolecta, almacena y consulta de métricas en formato de series temporales**. Esto significa que guarda datos como “uso de CPU”, “latencia de red” o “número de peticiones” junto con una marca de tiempo, lo que permite analizar el comportamiento de tus servicios a lo largo del tiempo.

¿Para que sirve?

- **Prometheus** permite:
- **Monitorear servicios web, APIs, bases de datos, servidores, contenedores, etc.**
- **Configurar alertas automáticas** cuando algo falla o se sale de rango.
- **Visualizar métricas en tiempo real** (especialmente junto a Grafana).
- **Diagnosticar problemas de rendimiento** antes de que afecten al usuario.
- **Integrarse con exporters** para MySQL, Redis, Node.js, Docker, etc.

Instalación

- En **Windows**

- Descarga el binario desde <https://prometheus.io/download/>
- Extrae el archivo ZIP
- Ejecuta prometheus.exe con el archivo de configuración prometheus.yml

- En **Docker:**

```
docker run -d --name prometheus \
  -p 9090:9090 \
  -v /path/to/prometheus.yml:/etc/prometheus/prometheus.yml \
  prom/prometheus
```

- <http://localhost:9090>

Uso de prometheus

- Tu aplicación expone métricas en un endpoint tipo **/metrics**.
- Prometheus consulta ese endpoint cada X segundos.
- Guarda los datos en su base de **series temporales**.
- Puedes hacer consultas con **PromQL** (su lenguaje de consulta).
- Puedes visualizarlo con Grafana o configurar alertas con **Alertmanager**.

Prometheus

- Prometheus necesita un fichero de configuración: `prometheus.yml`
- Normalmente se aloja en la misma carpeta que el fichero `Docker-compose.yml`.
- Para probarlo: una vez arrancado el contenedor, se puede probar prometheus consigo mismo.
 - Expone métricas en: <http://localhost:9090/metrics>

prometheus.yml

- Para probarlo consigo mismo:

global:

scrape_interval: 15s *# intervalo de recolección de métricas*

scrape_configs:

- job_name: 'prometheus'

static_configs:

- targets: ['localhost:9090']

- Con esto veremos métricas de uso de CPU, memoria, y actividad interna de prometheus.

Limitar el espacio







- `--storage.tsdb.retention.time=7d`
- `--storage.tsdb.retention.size=2GB`
- Esto limita la retención a 7 días o 2 GB, lo que ayuda a mantener el uso bajo control.

Grafana

Introducción

- **Grafana**, una de las herramientas más potentes y populares para la **visualización de métricas y monitoreo en tiempo real**, especialmente cuando se combina con Prometheus.
- **Grafana** es una plataforma de **visualización de datos de series temporales**. Te permite crear **dashboards interactivos** para monitorear servidores, servicios web, bases de datos, contenedores, sensores IoT, y mucho más.
 - Compatible con múltiples fuentes de datos (Prometheus, InfluxDB, MySQL, PostgreSQL, etc.)
 - Ofrece gráficos, tablas, mapas, alertas y paneles personalizables
 - Permite configurar alertas visuales y notificaciones automáticas
 - Se integra fácilmente con Docker, Kubernetes, y herramientas DevOps

¿Para que sirve?

Función clave	¿Qué te permite hacer?
 Visualizar métricas	CPU, RAM, tráfico, errores HTTP, latencia, etc.
 Diagnóstico en tiempo real	Detectar cuellos de botella y anomalías
 Alertas automáticas	Enviar notificaciones por email, Slack, Telegram, etc.
 Seguimiento histórico	Analizar el comportamiento de tus servicios a lo largo del tiempo
 Integración con Prometheus	Mostrar métricas recolectadas por Prometheus
 Modo kiosko/TV	Mostrar dashboards en pantallas de monitoreo

Instalación

- En **Windows**

- Descarga el instalador desde: <https://grafana.com/download>
- Ejecuta el .exe y sigue los pasos
- Accede a <http://localhost:3000> tras iniciar el servicio
- **login / pass → admin**
- Cambiar contraseña: **admin / antonio**

- En **Docker:**

- `docker run -d --name=grafana -p 3000:3000 grafana/grafana`

Fuentes compatibles con Grafana

- Grafana puede conectarse a:
 - **Prometheus**
 - **InfluxDB**
 - **Graphite**
 - **MySQL / PostgreSQL**
 - **ElasticSearch**
 - **Loki** (para logs)
 - **Tempo** (para trazas)

Ejemplo de uso

- Tu servicio web (por ejemplo, implementado con **Crow** o **gRPC**) expone métricas en **/metrics**.
- **Prometheus** recolecta esas métricas cada X segundos.
- **Grafana** se conecta a Prometheus como fuente de datos.
- Crear **dashboards** con gráficos, alertas y visualizaciones.
- Puedes compartirlos, exportarlos o mostrarlos en modo kiosko.

Visualizar datos de Prometheus

- Desde el panel principal de Grafana:
 - Nuevo datasource
 - Seleccionar en el **datasource** prometheus.
 - Ojo la URL: <http://prometheus:9090> (no es localhost)
 - Botón **save & test**

Grafana

- **Crear Dashboards personalizados**

- Grafana te permite crear paneles visuales con gráficos, tablas, y alertas. Lo típico es:
- **Monitorear servicios:** peticiones HTTP, errores, tiempos de respuesta.
- **Monitorear infraestructura:** CPU, RAM, disco, red (usando Node Exporter).
- **Monitorear bases de datos:** conexiones, latencia, uso de disco.
- **Monitorear contenedores:** con cAdvisor o Prometheus + Docker metrics.

Grafana

- **Explorar métricas**

- En el menú lateral:
- Ve a **Explore**.
- Selecciona la fuente de datos (Prometheus).
- Escribe una consulta como `up, rate(http_requests_total[1m])`, o `node_cpu_seconds_total`.
- Esto te permite probar métricas antes de agregarlas a un dashboard.

Grafana

- Usar paneles predefinidos
- Grafana tiene una galería de dashboards listos para usar:
 - Ve a Grafana Dashboards
 - Busca por tecnología: Node Exporter, Docker, PostgreSQL, etc.
 - Copia el ID del dashboard y en Grafana ve a + → Import para cargarlo.

Grafana

- **Configurar alertas**

- Puedes crear alertas que se disparen cuando una métrica supera un umbral:
- En un panel, haz clic en "**Edit**" → "**Alert**".
- Define la condición (ej. CPU > 80%).
- Configura la notificación (Slack, email, etc.).

Grafana

- **Gestionar usuarios y permisos**
- Si estás en equipo:
 - Crea usuarios con roles (Viewer, Editor, Admin).
 - Asigna permisos por dashboard o carpeta.

cAdvisor

cAdvisor

- **cAdvisor** (short for *Container Advisor*) es una herramienta desarrollada por Google que se utiliza para **monitorear el rendimiento de contenedores Docker** en tiempo real.
 - Recolecta métricas de uso de CPU, memoria, disco y red por contenedor.
 - Expone esas métricas en una interfaz web (<http://localhost:8080>) y en formato Prometheus ([/metrics](#)).
 - Permite visualizar qué contenedores están corriendo y cómo están consumiendo recursos.

cAdvisor

- Se puede utilizar para usarlo con Prometheus y Grafana:
 - Prometheus puede recolectar las métricas que cAdvisor expone.
 - Grafana puede visualizarlas en dashboards.
- Es ideal para monitorear entornos Docker sin necesidad de instalar agentes en cada contenedor.

Implementación de tracing distribuido

Herramientas

- **Jaeger:**
 - Herramienta de **tracing distribuido** desarrollada originalmente por Uber.
 - Sirve para rastrear cómo fluyen las peticiones a través de los distintos servicios de tu sistema.
- OpenTelemetry

Jaeger

Jaeger

- **¿Para qué se usa?**

- Ver cuánto tarda cada servicio en responder.
- Detectar cuellos de botella o errores en la cadena de llamadas.
- Visualizar el recorrido completo de una petición (por ejemplo, desde el frontend hasta la base de datos).

- **¿Cómo funciona?**

- Cada servicio genera "spans" (fragmentos de trazas).
- Jaeger recolecta esos spans y los muestra en una interfaz web.
- Puedes ver gráficos de tiempo, dependencias entre servicios, y más.

OpenTelemetry

OpenTelemetry

- **OpenTelemetry** es un **framework abierto y estandarizado** para recolectar datos de observabilidad:
 - **traces,**
 - **metrics**
 - **y logs.**
- **¿Qué hace?**
 - Proporciona SDKs para múltiples lenguajes (Java, Python, Go, etc.).
 - Te permite instrumentar tu código para generar métricas y trazas.
 - Envía esos datos a herramientas como Jaeger, Prometheus, Grafana, Zipkin, etc.

OpenTelemetry

- **¿Cómo se relacionan Jaeger y OpenTelemetry?**
 - **OpenTelemetry genera los datos** (trazas, métricas).
 - **Jaeger los visualiza** (trazas).
 - Puedes usar OpenTelemetry para enviar trazas a Jaeger, métricas a Prometheus, y logs a otros sistemas.

Gestión de logs en entornos distribuidos

Herramientas

- **Fluentd** es un colector de logs de código abierto que:
 - Recolecta logs desde múltiples fuentes (archivos, syslog, stdout, etc.)
 - Los transforma y filtra (formato, etiquetas, niveles)
 - Los reenvía a destinos como Elasticsearch, Kafka, S3, etc.
 - Es parte de la **Cloud Native Computing Foundation (CNCF)** y se usa en producción por empresas como Amazon, Microsoft y Google.
- **Elasticsearch**
 - Es un motor de búsqueda y análisis de texto distribuido que:
 - Indexa logs en tiempo real
 - Permite búsquedas complejas y agregaciones
 - Se integra con **Kibana** para visualización

Integración Fluentd y Elasticsearch

- Fluentd recolecta los logs del servicio C++ (por ejemplo, desde archivos generados por spdlog, Boost.Log, etc.)
- Los transforma (añade etiquetas, convierte a JSON, etc.)
- Los envía a Elasticsearch, donde se almacenan e indexan
- Kibana (opcional) los visualiza en dashboards interactivos

Instalación en Docker

- **Fluentd**

```
docker run -d -p 24224:24224 -p 24224:24224/udp \
-v /path/to/fluent.conf:/fluentd/etc/fluent.conf \
fluent/fluentd
```

- **Elasticsearch**

```
docker run -d -p 9200:9200 -p 9300:9300 \
-e "discovery.type=single-node" \
elasticsearch:8.12.0
```

Se puede auditar

- Accesos a endpoints
- Errores HTTP
- IPs de origen
- Usuarios autenticados
- Cambios en datos sensibles