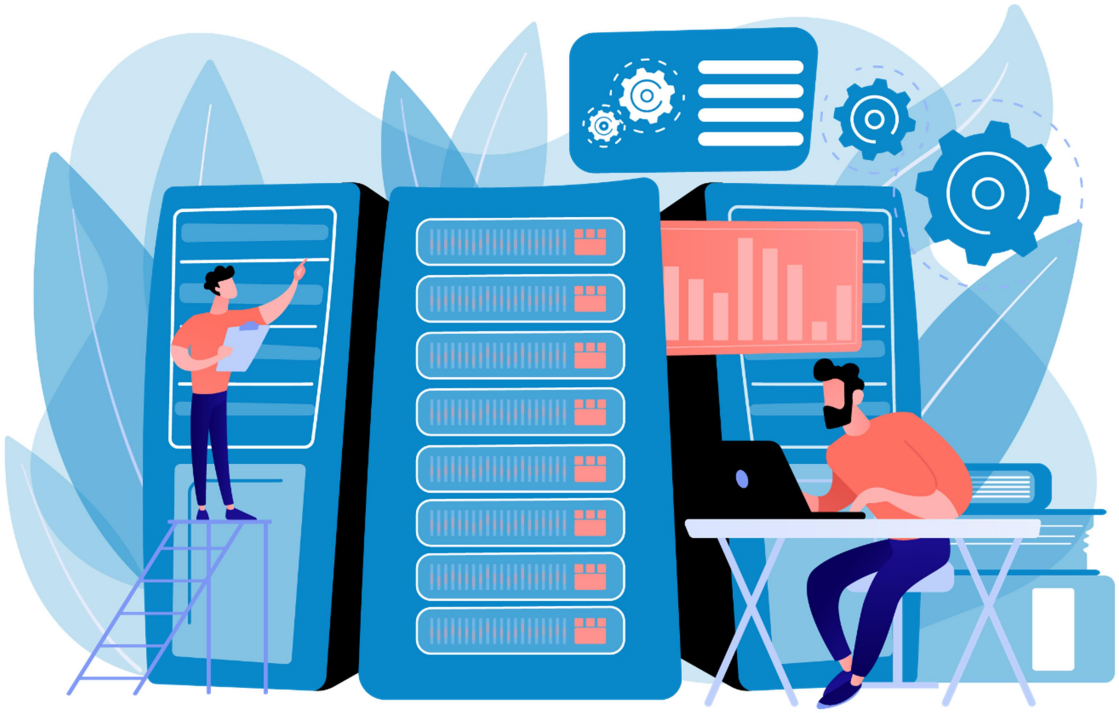
IES MONTE NARANCO

APUNTES DE TEORÍA

DEVOPS Y CLOUD COMPUTING



UT - 7

MONITORIZACIÓN Y OBSERVABILIDAD EN DEVOPS

Índice

1. Importancia de la monitorización en entornos productivos ____ *4*

2. Métricas clave y visualización de datos ____ *5*

2.1 Métricas más habituales: ____ **5**

2.2 Visualización ____ **5**

2.3 Buenas prácticas de visualización ____ **6**

3. Logging centralizado y detección de errores ____ *7*

3.1 ¿Por qué centralizar logs? ____ **7**

3.2 Tecnologías habituales ____ **7**

3.3 Detección de errores ____ **7**

3.4 Buenas prácticas ____ **8**

4. Configuración de alertas y respuestas automatizadas ____ *9*

4.1 Alertas ____ **9**

4.2 Respuestas automatizadas ____ **10**

1. Importancia de la monitorización en entornos productivos

En un entorno de producción, las aplicaciones y servicios deben funcionar de manera estable, segura y eficiente. La *monitorización* es el conjunto de técnicas y herramientas que permiten observar el comportamiento del sistema en tiempo real y anticiparse a problemas antes de que afecten al usuario final.

Razones por las que la monitorización es esencial:

- **Detección temprana de fallos**: Permite identificar comportamientos anómalos (latencias elevadas, incremento brusco de uso de CPU, errores repetidos…) antes de que provoquen caídas del servicio.
- **Mejora del rendimiento**: Analizando métricas a lo largo del tiempo, se pueden detectar cuellos de botella, optimizar consultas a bases de datos o ajustar recursos.
- **Garantía de disponibilidad**: La monitorización ayuda a cumplir los SLA (Acuerdos de Nivel de Servicio), asegurando que el servicio esté disponible el mayor tiempo posible.
- **Toma de decisiones basada en datos**: La información recogida permite planificar escalados, actualizaciones, y tomar decisiones objetivas basadas en el uso real.
- **Seguridad**: Actividades sospechosas, accesos no autorizados o patrones anómalos pueden detectarse mediante monitorización adecuada.

En definitiva, la monitorización es un pilar fundamental en DevOps, SRE (Site Reliability Engineering) y en cualquier equipo que gestione software en producción.

2. Métricas clave y visualización de datos

Las métricas son datos recopilados de un sistema, aplicación o infraestructura. Elegir las adecuadas y representarlas correctamente es esencial para entender el estado real de los servicios.

2.1 Métricas más habituales:

- **Métricas de infraestructura**:
  - **CPU**: Porcentaje de uso, carga media.
  - **Memoria RAM**: Uso, memoria libre, SWAP utilizada.
  - **Almacenamiento**: Espacio disponible, I/O del disco.
  - **Red**: Tráfico de entrada/salida, latencia, paquetes perdidos.
- **Métricas de aplicación**:
  - **Tiempo de respuesta**: Latencia media, picos.
  - **Número de peticiones por segundo (RPS)**.
  - **Errores**: Tasa de errores 4xx, 5xx.
  - **Hilos activos, conexiones abiertas, colas de tareas**.
- **Métricas de negocio**:
  - Carritos creados, ventas por minuto, registros de usuarios, etc.
  - Son útiles para evaluar impacto real en usuarios.

2.2 Visualización

Las herramientas de visualización permiten convertir métricas en gráficos intuitivos y paneles (dashboards).

Herramientas más usadas:

- **Grafana**: Estándar de la industria para crear paneles personalizables.
- **Kibana**: Muy usado con Elasticsearch para explorar logs y métricas.
- **Datadog, New Relic, Prometheus + Grafana**: Soluciones completas para monitorización de aplicaciones y sistemas.

2.3 Buenas prácticas de visualización

- No sobrecargar los paneles: mostrar solo lo esencial.
- Usar colores consistentes (verde = OK, amarillo = aviso, rojo = crítico).
- Agrupar métricas por contexto (servidores, bases de datos, API…).
- Incluir gráficos históricos para analizar tendencias.

3. Logging centralizado y detección de errores

Además de las métricas, los *logs* son otra fuente crítica de información. Recogen eventos, errores y trazas generadas por los servicios.

3.1 ¿Por qué centralizar logs?

En entornos distribuidos (microservicios, contenedores, clusters…), cada componente genera sus propios logs. Tenerlos en un solo sitio permite:

- Realizar búsquedas globales.
- Correlacionar eventos entre varios servicios.
- Analizar errores y patrones complejos.
- Mejorar la capacidad de auditoría y trazabilidad.

3.2 Tecnologías habituales

- **ELK Stack**: Elasticsearch, Logstash, Kibana.
- **EFK Stack**: Elasticsearch, Fluentd, Kibana.
- **Graylog, Splunk, Datadog Logs, Papertrail**.

3.3 Detección de errores

La detección se basa en:

- **Patrones en logs**: palabras clave, excepciones, códigos de error.
- **Anomalías en métricas**: cambios bruscos en el comportamiento habitual.

- **Alertas basadas en logs**: cuando aparecen errores repetitivos o eventos críticos.

3.4 Buenas prácticas

- Normalizar logs en formato JSON.
- Añadir metadatos: timestamps, servicio, nivel (INFO, WARN, ERROR), ID de correlación.
- No almacenar información sensible.

4. Configuración de alertas y respuestas automatizadas

4.1 Alertas

Las alertas notifican cuando algo requiere la atención del equipo.

Tipos:

- **Alertas basadas en métricas**: CPU > 80%, latencia demasiado alta, errores 5xx.
- **Alertas basadas en logs**: más de X errores en Y tiempo.
- **Alertas predictivas**: análisis de tendencias (machine learning básico).

Canales habituales:

- Email
- Slack / Teams
- SMS
- Herramientas de incident management como PagerDuty u Opsgenie

Diseño de alertas efectivas

- Evitar alertas ruidosas: solo avisar de lo realmente importante.
- Definir umbrales adecuados.
- Incluir información contextual (qué servicio falla, posibles causas).
- Probar las alertas periódicamente.

4.2 Respuestas automatizadas

La automatización permite actuar sin intervención humana cuando se producen ciertas condiciones.

Ejemplos:

- Reiniciar automáticamente un servicio que se queda colgado.
- Escalar horizontalmente si aumenta la carga (autoescalado en Kubernetes o AWS).
- Bloquear IPs sospechosas si se detecta actividad maliciosa.

Ventajas:

- Reducción del tiempo de recuperación (MTTR).
- Menos intervención manual.
- Mayor resiliencia del sistema.