Observabilidad

¿Qué es la observabilidad?
-Detectar cambios, para ver si el estado de un servicio cambia.
-Observar el flujo de datos de información dentro del sistema.
-Monitorización del sistema.
-Logs y métricas.
-Capacidad de observar los estados mediante métricas.

Apuntes de Aitor, a partir de ahora en negrita:

- **Cambios**
- **Estado servicio**
- **Flujo de datos**
- **Monitorización**
- **Logs/Métricas**

-SLA
-Rendimiento y escalabilidad -> esto repercute en los costes de cara al sistema y de cara a la facturación a los clientes (los precios).
-Costes (sistema)
-Facturación (precios a los clientes)
-Seguridad: para que a la empresa no le roben datos y no tengan que pagar cantidades millonarias.

Todo repercute principalmente en costes y precios

Si queremos organizar los datos de acuerdo a esta información debemos organizarlos de forma lógica de acuerdo a:

- **Datos de negocio:** para saber los precios, la cantidad de ventas, etc. , pueden ser por ejemplo:
    - Log de peticiones a las APIs
    - Cantidad de nuevas cuentas y cantidad de cancelaciones.
    - Con estos datos podemos hacer análisis de marketing (con todos los datos que generas, son muchos)
- **Datos de operación:** pueden ser por ejemplo:
    - Peticiones
    - Input/Output de disco
    - Número de servicios funcionando
    - ¿Cómo están la CPU y la memoria de los clusters de servicios?
    - ¿Cuántas peticiones se tiran, o vienen de distintos atacantes?

---Diagrama de servicio del sistema

¿Cómo obtenemos estos datos?
-Logs de acceso: del balanceador y de kong sacamos los datos de este cliente, desde este ip, ha tardado tanto, esto es lo que le he devuelto.
-Logs de máquina: Los servidores están pensados para enviarlo al log de máquina (en linux /var/log).
-Métricas de las máquinas: carga de la CPU
-Logs de acceso a base de datos y métricas de base de datos: concesiones, input, output
-Logs de nuestros propios servicios:
-Logs que genera AWS: tipo he levantado esta máquina y nos da métricas de cómo funciona la máquina
-Logs del servicio de KONG:

Tenemos que intentar homogeneizarlos para que más o menos se puedan procesar de la misma forma.

--Se diferencia:

- Quién lo va a ver.
- Volumen de los datos: una treintena de máquinas hace más o menos 7-8 gigas al día.
- Tiempo: Por ejemplo que la CPU estaba al 80% hace dos años no nos interesa, pero que el cliente X ha hecho una petición es necesario almacenarlo.

Arquitecturas genéricas

--Diagrama servicio máquina y agentes

Agente recolector: coge los logs y los envía a un destino (una base de datos). Las bases de datos tradicionales no están pensadas para guardar estos datos. Recurrir a bases de datos NoSQL, de estilos documentales MongoDB, ElasticSearch (trae su propia interfaz para buscar los datos, es open source). También existen bases de datos NoSQL que están pensadas para series temporales, algunas métricas son series temporales (Ejemplo: Prometheus, que no admite Kibana, pero puedes utilizar Grafana)

Agente de logs: filebeat
Agente de metricas: metricbeat
Agente para la administración de logs: Logstash
Herramienta para hacer Dashboard: Kibana, Grafana
ELK: ElasticSearch, Logstash y Kibana
ETL: Extract, Transform and Load

No se puede recurrir nunca a poner en postgres columna1, columna2, …, columnaN, y luego en una tabla excel describir esas columnas.

Tener alertas de lo que está ocurriendo es importante: mucho cuidado con ello, es muy importante saber cuales son las alertas, las prioridades de las alertas, etc.

Arquitectura basada en peticiones síncronas: es la que está anteriormente
Arquitecturas basadas en publicación subscripción: