Monitoreo centralizado de microservicios

Solución centralizada para logs:

Opción recomendada: Grafana Loki con Grafana Dashboard.

- **Por qué Loki y Grafana:**
  Loki es un sistema de almacenamiento y consulta de logs diseñado específicamente para ser eficiente en términos de costo y escalabilidad. Es "log-oriented" y compatible con Prometheus, lo que facilita su integración con métricas y alertas.

- **Alternativa: Elasticsearch con Kibana (ELK Stack).**
  - Elasticsearch permite consultas complejas y análisis avanzado de los logs.
  - Kibana es una herramienta robusta para visualización y paneles interactivos.

Configuración básica para Loki:

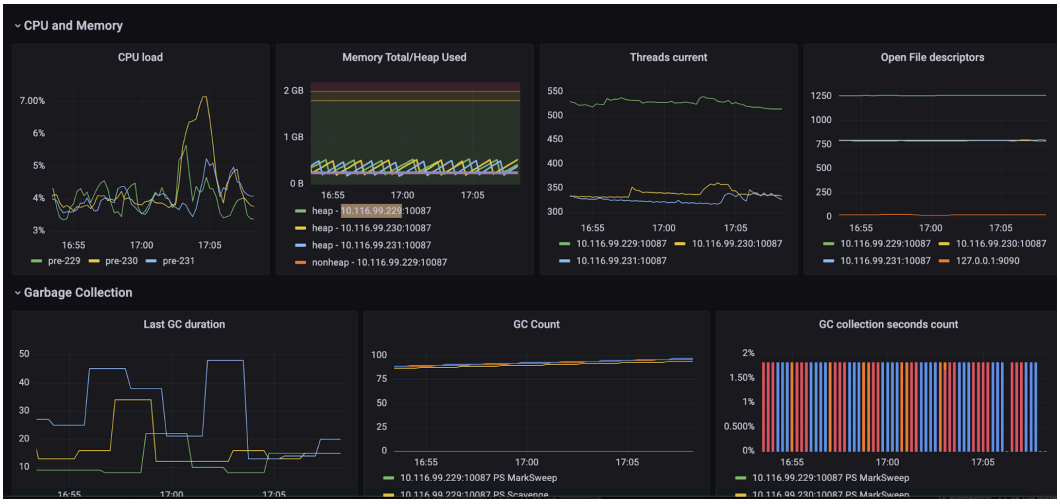
1. **Instalación:**
   - usar **Helm Charts** para implementar Loki en los clústeres EKS.
   - usar **Promtail** o **Fluent Bit** como agentes recolectores de logs en cada nodo del cluster.
2. **Integración con Grafana:**
   - Enlazar Grafana con Loki como fuente de datos

     docker run -d --name=loki -p 3100:3100 grafana/loki:latest

     - Para enviar logs a loki hay que configurar el promtail-config.yaml y correr el comando

       docker run -d --name=promtail -v /var/log:/var/log -v $(pwd)/config:/etc/promtail grafana/promtail:latest -config.file=/etc/promtail/promtail-config.yaml

- Crear dashboards de visualización para logs con filtros personalizables por servicio, nivel de error, timestamp, etc.



Costo y eficiencia:

- **Loki:**
  - Es más ligero y consume menos recursos comparado con Elasticsearch, ya que indexa metadatos, no el contenido completo de los logs.
  - Perfecto para equipos que necesitan monitoreo sin un aumento drástico de costos.

- **ELK Stack:**
  - Ofrece capacidades avanzadas, pero con mayores costos de almacenamiento y procesamiento.

2. Proceso

a. Recolección de logs desde microservicios:

1. **Instalación de agentes:**
   - Deployar agentes de logs (Promtail, Fluent Bit o Fluentd) en los nodos del clúster.

    - Configurar los agentes para leer los logs generados por contenedores en el runtime de Kubernetes (generalmente almacenados en `/var/log/containers`).
    - Enviar los logs a la solución centralizada (Loki o Elasticsearch).
2. **Configuración de etiquetas:**
    - Añadir etiquetas automáticas a los logs, como:
        - `service_name`: Nombre del microservicio.
        - `namespace`: Espacio de nombres del clúster.
        - `log_level`: Nivel de severidad (info, warning, error).
    - Esto facilita la búsqueda y el filtrado en los dashboards.
3. **Políticas de retención:**
    - Configurar políticas para retener logs críticos más tiempo (e.g., errores) y purgar logs menos relevantes (e.g., debug) según el presupuesto disponible.

b. Consulta de fallos en tiempo real:

1. **Uso de dashboards:**
    - Los desarrolladores pueden utilizar Grafana o Kibana para filtrar logs en tiempo real por:
        - Nombre del microservicio.
        - Tipo de error o palabras clave.
        - Intervalos de tiempo específicos.
2. **Alertas automatizadas:**
    - Configurar alertas basadas en patrones de logs:
        - Ejemplo: Detectar errores HTTP 500 o excepciones frecuentes.
    - Las alertas se pueden enviar a canales de comunicación como Slack, Teams o email.
3. **Acceso seguro:**
    - Usar **IAM roles** y **autenticación mediante SSO** para garantizar que solo los desarrolladores autorizados accedan a los logs.

---

3. Métricas clave para medir la salud de los microservicios (opcional)

1. **Uso de recursos:**
    - CPU y memoria por contenedor.
    - Tasa de uso de disco (especialmente en servicios que generan muchos logs).
2. **Tiempos de respuesta:**
    - Latencia promedio y máxima por endpoint.
    - Distribución de los tiempos de respuesta.
3. **Errores:**
    - Tasa de errores (e.g., porcentaje de errores HTTP 5xx).
    - Tasa de logs de nivel `error` o `critical`.

4. **Disponibilidad:**
    - Número de instancias activas por microservicio.
    - Tiempos de reinicio de contenedores (indica posibles fallos).

En Resumen

realice este esquema basandome en herramientas ligeras y fácilmente integrables como Loki y Grafana, ideales para startups o equipos con presupuestos limitados. Si el equipo tiene mayores necesidades de análisis y escalabilidad, Elasticsearch y Kibana son opciones robustas, aunque más costosas. El enfoque modular asegura quepodamos escalar la solución fácilmente mientras vaya aumentando la carga o la cantidad de microservicios.