Universidad San de Guatemala

Facultad de ingeniería

Sistemas Operativos 1

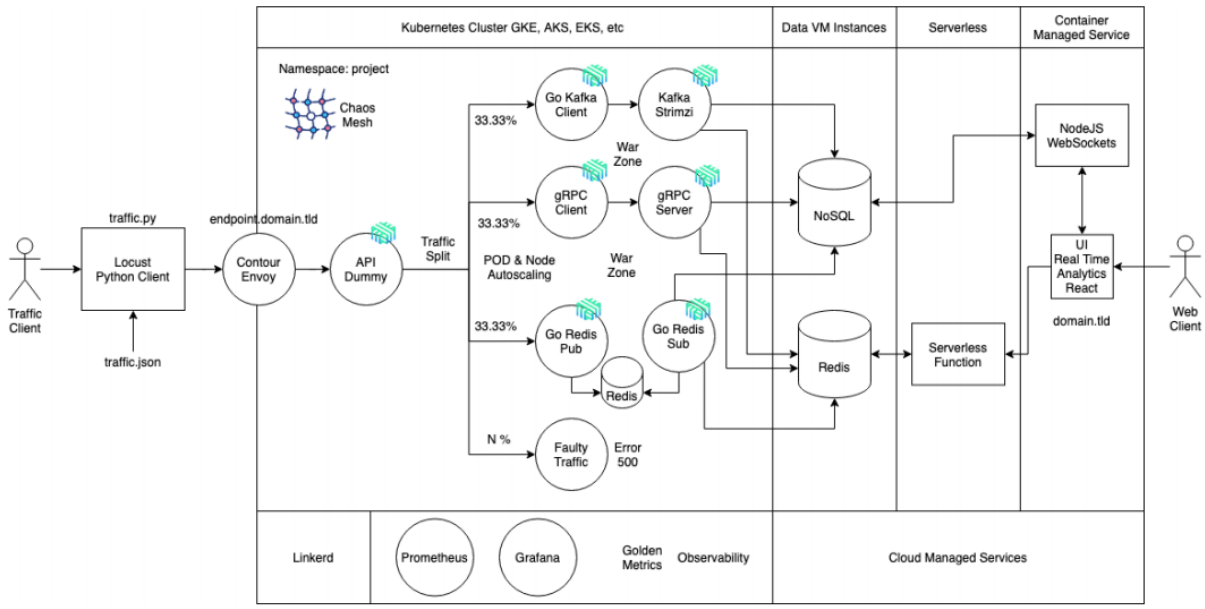
Grupo 15

Manual de Tecnico

Objetivos

- Comparar el tiempo de respuesta y el desempeño de las distintas rutas.

Arquitectura



Descripción

El trafico será generado utilizando Locust y el lenguaje de programación Python. El tráfico será recibido por un balanceador de carga público (k8s ingress) con el dominio: endpoint.domain.tld; este dominio será expuesto utilizando un ingress controller

Luego de haber generado el trafico este llegara a la parte de git y kubernetes para esta parte de debe configurar un balanceador de carga de capa 7 (como Kubernetes Ingress) en el clúster de Kubernetes utilizando Helm o Kubectl.

Se implemento un traffic splitter utilizando Linkerd. dividiendo el tráfico en las tres rutas en un tercio de tráfico para cada ruta. Para implementarlo Linkerd utiliza un dummy service que puede ser la copia de un servicio de cualquiera de las rutas y debe utilizarse nginx para esta funcionalidad.

La idea es crear una manera de escribir datos en bases de datos NoSQL con alto desempeño utilizando la comunicación por RPC, message brokers y las colas de mensajería. La meta es comparar el desempeño de las rutas. Estas rutas son:

- gRPC: Es un framework de RPC de alto desempeño que puede ser ejecutado en cualquier ambiente, usado principalmente para conectar servicios de backend.
- Redis Pub/Sub: Es una manera de crear un sistema de colas de mensajería para aplicaciones nativas de cloud y arquitecturas de microservicios.
- Kafka: Es un sistema de colas de alta disponibilidad para la transmisión de datos en aplicación en tiempo real.

Como ultimo segmento todos lo datos llegan a diferentes bases de datos NoSQL, entre ellas están:

- MongoDB: es una base de datos NoSQL documental que almacena la información utilizando el formato de datos JSON.
- Redis: es una base de datos NoSQL de clave-valor que implementa distintos tipos de estructuras de datos como listas, conjuntos, conjuntos ordenados, etc.

Luego de ser almacenados los datos estos serán desplegados en un sitio web en react mostrando las siguientes consultas:

1. Datos almacenados en la base de datos, en MongoDB.
2. Los diez países con más vacunados, en Redis.
3. Personas vacunadas por cada país, en Redis.
4. Gráfica de pie de los géneros de los vacunados por país, en MongoDB.
5. Los últimos cinco vacunados almacenados por país, en MongoDB.
6. Gráfica de barras del rango de edades (de diez en diez) por cada país, en Redis.
7. Mapa interactivo que muestre estos reportes.

Herramientas

Git: Será la herramienta para gestionar las versiones del código del proyecto..

Docker: Se utilizará para empaquetar la aplicación en contenedores, donde se sugiere utilizar técnicas distroless para crear imágenes pequeñas si es posible.

Kubernetes: Kubernetes será el encargado de la orquestación de los contenedores. Antes de que el cliente genere el tráfico, el proyecto implementará un clúster de Kubernetes que se utilizará para desplegar distintos objetos: Ingress controllers, deployments y services y por ultimo pods.

Preguntas

1. ¿Cómo se pueden interpretar las cuatro pruebas de faulty traffic?

Son pruebas que se utilización para determinar los posibles casos de error en los pods previamente creados

2. ¿Qué patrones de conducta fueron descubiertos?

Se descubrió que al simular un error con chaos mesh dentro de un pod este trafico se toma como trafico fallido y el flujo en las demás ramas sigue siendo el mismo.

3. ¿Qué sistema de mensajería es más rápido? ¿Por qué?

Por medio de las peticiones aceptadas por segundo de cada mensajería se determino que Kafka es el sistema de mensajería mas rápido.

4. ¿Cuántos recursos utiliza cada sistema de mensajería?

Kafka producer: 0.00048

Kafka consumer: 0.014

Grpc cliente: 0.00056

Grpc server: 0.00071

Pubsub client: 0.00068

Pubsub server: 0.27

5. ¿Cuáles son las ventajas y desventajas de cada servicio de mensajería?

Kafka:

- Ventajas
    - Baja latencia
- Desventajas
    - No soporta mensajes de gran tamaño (1MB)
    - Se debe tener un servidor conectado a consumer con producer
    - No tiene muchas opciones de monitorización potentes

Grpc

- Ventajas
    - Fácil de implementar

    - Flexible
- Desventajas
    - Al conectar ordenadores a través de largas distancias, gRPC está mucho más expuesto que un sistema local

Pubsub

- Ventajas
    - La velocidad que nos proporciona redis en almacenamiento
- Desventajas
    - Puede aumentar costos por la implementación de redis como cache

6. ¿Cuál es el mejor sistema de mensajería?

Tomando en cuenta la cantidad de recursos y la velocidad en la peticiones, se determino que Grpc es el mejor método de mensajeria.

7. ¿Cuál de las dos bases de datos se desempeña mejor y por qué?

Se determino que MongoDB es mejor base de datos ya que soporta mayores cantidades que redis ya que redis esta enfocado en el almacenamiento cache. Redis seria mas útil para sistemas de mensajería como pubsub y no como base de datos principal.

8. ¿Cómo se refleja en los dashboards de Linkerd los experimentos de Chaos Mesh?

Antes de aplicar chaos mesh se observa que tiene un buen flujo y al aplicar pod kill se muestra que tiene un bajon en la grafica ya que esta la elimina.

9. ¿En qué se diferencia cada uno de los experimentos realizados?

En que cada manifiesto afecta de manera diferente a una parte en especifico del proyecto.

10. ¿Cuál de todos los experimentos es el más dañino?

Considerando las pruebas pod failure es el mas dañino porque al momento de estar recibiendo trafico y este se aplica se puede perder información valiosa.