UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
SISTEMAS OPERATIVOS 1
SEGUNDO SEMESTRE 2022
ING. SERGIO ARNALDO MENDEZ AGUILAR
AUX. CARLOS DAVID RAMIREZ ALTAN

PRÁCTICA #2

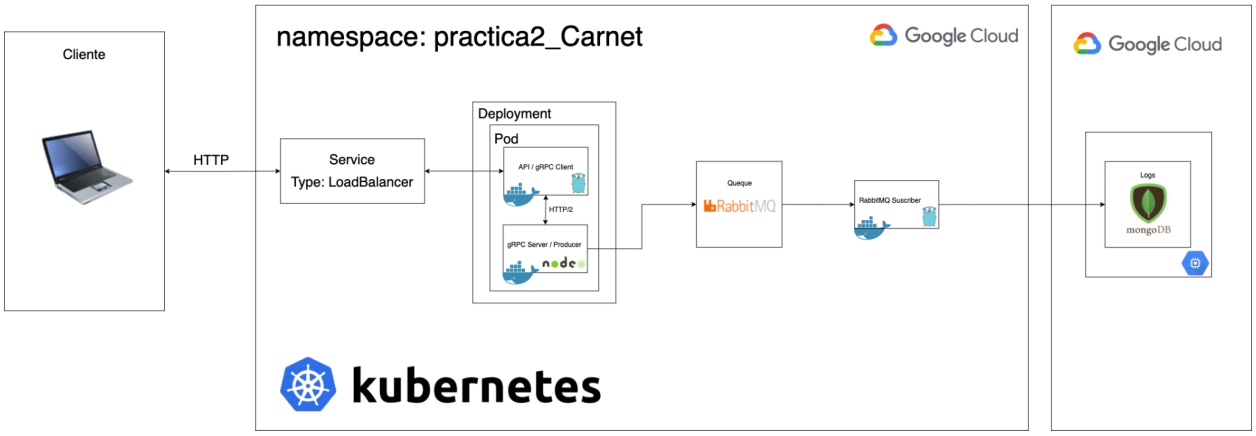
Se solicita construir un sistema genérico de arquitectura distribuida que muestre estadísticas en tiempo real utilizando Kubernetes y service mesh como Linkerd y otras tecnologías Cloud Native. En la última parte se utilizará una service mesh para dividir el tráfico. Adicionalmente, se añadirá Chaos Mesh para implementar Chaos Engineering.

Este proyecto se aplicará a la visualización de los resultados de juegos implementados por los estudiantes.

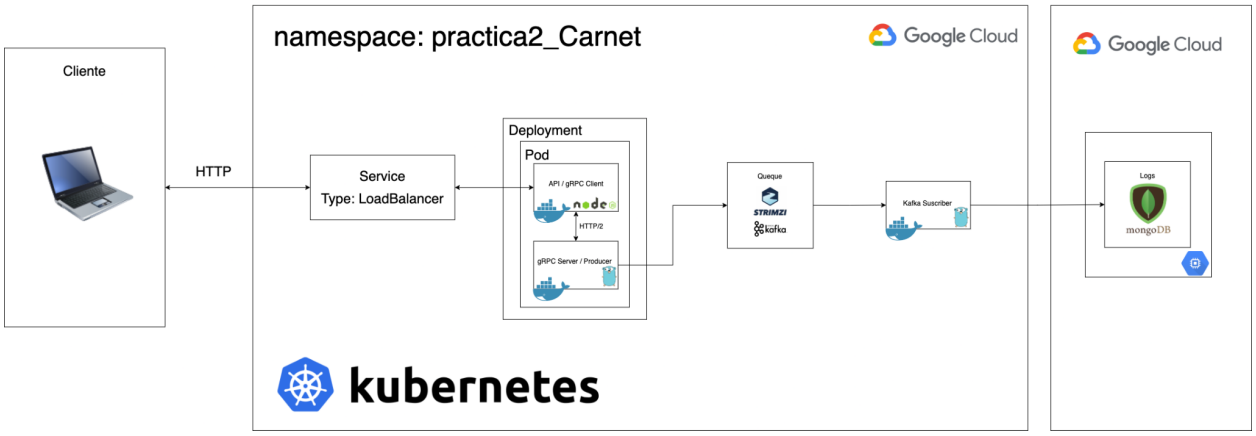
OBJETIVOS

- Experimentar y probar tecnologías Cloud Native que ayudan a desarrollar sistemas distribuidos modernos.
- Diseñar estrategias de sistemas distribuidos para mejorar la respuesta de alta concurrencia.
- Implementar contenedores y orquestadores en sistemas distribuidos.
- Medir la confiabilidad y el rendimiento en un sistema de alta disponibilidad.

ARQUITECTURA - CARNET TERMINACIÓN PAR



ARQUITECTURA - CARNET TERMINACIÓN IMPAR



Explicación:

Se solicita al estudiante desplegar en un cluster de kubernetes las rutas definidas en la fase 3 del proyecto de manera que los estudiantes con carnet par deberán desplegar la ruta de RabbitMQ y los estudiantes con carnet impar deberán desplegar la ruta de Kafka.

Juegos:

Los juegos serán algoritmos sencillos implementados por los estudiantes estos algoritmos deben estar implementados en el servidor de gRCP de manera que al recibir los datos del juego en este servidor se seleccionara al azar uno de

los juegos implementados por el estudiante de manera que al finalizar el juego y tener un ganador se procede a mandar los resultados del juego a la cola (Queue) correspondiente.

Por ejemplo, un juego tendrá las siguientes reglas:

Genera 10 números

Elija al azar un número como ganador y el otro perderá el juego.

Nota: **Se deberán implementar como mínimo 5 juegos diferentes** sin embargo si el estudiante desea implementar más de 5 está bien.

PRIMERA PARTE (REGISTRY)

Registry:

Deberá utilizar un registry para la publicación de las imágenes creadas, deberá utilizar el siguiente formato para las imágenes creadas <USER_DOCKERHUB>/<IMAGE_NAME>_<CARNET>

Ejemplo:

racarlosdavid/api_client_grpc_201213132

racarlosdavid/server_grpc_201213132

SEGUNDA PARTE (DOCKER, KUBERNETES Y BALANCEADORES DE CARGA)

PRIMERA RUTA (RabbitMQ):

- Generador de Trafico
- Load Balancer
- API / gRPC Client (Go)
- grPC Server (NodeJS) / Producer RabbitMQ Queue
- RabbitMQ
- RabbitMQ Subscriber
- Escribir en las bases de datos NoSQL (Redis,Tidis y MongoDB)

SEGUNDA RUTA (kafka):

- Generador de Trafico
- Load balancer
- API / gRPC Client (NodeJS)
- grPC Server (Go) / Producer Kafka Queue
- Kafka
- Kafka Subscriber
- Escribir en las bases de datos NoSQL (Redis,Tidis y MongoDB)

TERCERA PARTE (RPC AND BROKERS)

La principal 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, Consulte el diagrama de arquitectura.

gRPC: Es un framework de RPC de alto desempeño que puede ser ejecutado en cualquier ambiente, usado principalmente para conectar servicios de backend.

Kafka: Es un sistema de colas de alta disponibilidad para la transmisión de datos en aplicación en tiempo real.

RabbitMQ: Es un modo de sistema de cola de la vieja escuela para funcionar como intermediario o procesamiento de tareas.

Nota: **gRCP debe ser implementado en 2 lenguajes de programación diferente go y nodejs.**

CUARTA PARTE (NOSQL DATABASES)

• MongoDB: es una base de datos NoSQL documental que almacena la información utilizando el formato de datos JSON. Un ejemplo de log seria:

```
{
    "request_number":30001, //Que numero de ejecucion es por ejemplo antes de este juego se jugaron 30000
    "game":1, //Id del juego
    "gamename":"Game1", //Nombre del juego
    "winner":"001", //Jugador que gano el juego
    "players":20 //Cantidad de jugadores que participaron en el juego
    "worker":"RabbitMQ" //El worker hizo la insercion del log en MongoDB
}
```

OBSERVACIONES

- Todo debe implementarse utilizando el lenguaje o la herramienta especificada.
- La práctica debe realizarse de forma individual.
- Deben escribir un manual técnico explicando todos los componentes del proyecto.
- Deben escribir un manual de usuario y un manual técnico.
- Las copias detectadas tendrán una nota de cero puntos y serán reportadas a la Escuela de Ciencias y Sistemas.
- No se aceptarán entregas después de la fecha y hora especificada.
- Se deberá agregar al usuario **racarlosdavid** al repositorio de la práctica.

ENTREGABLES

- Link del repositorio, debe incluir todo el código fuente con los archivos de manifiesto o cualquier archivo adicional de configuración.
- Manuales.

FECHA DE ENTREGA

29 abril antes de las 23:59 a través de UEDi. No hay prórrogas.