Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Sistemas Operativos 1

Manual Técnico
Tweets del Clima

Helen Janet Rodas Castro - 202200066

Guatemala 3 de mayo 2025

Introducción

Este proyecto implementa una arquitectura en Google Cloud Platform (GCP) utilizando Google Kubernetes Engine (GKE). El objetivo es construir una arquitectura de sistema distribuido genérico que muestre los tweets sobre el clima mundial. Esto se procesa mediante una arquitectura conceptual escalable. Este proyecto pretende mostrar la concurrencia de tuits en el sistema.

Deployments

Los deployments en Kubernetes gestionan la escalabilidad y disponibilidad de los componentes de la arquitectura, incluyendo consumidores de mensajes, bases de datos en memoria y la interfaz de visualización. Cada deployment asegura que los pods se mantengan activos y se repliquen según sea necesario para soportar la carga generada por Locust y el procesamiento de datos en tiempo real.

➔ Deployment kafka-consumer

Proyecto2 > Kubernetes > {..} kafka-consumer-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kafka-consumer
  namespace: tweets-clima
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kafka-consumer
  template:
    metadata:
      labels:
        app: kafka-consumer
    spec:
      containers:
      - name: kafka-consumer
        image: 34.172.134.232.nip.io/tweets-clima/kafka-consumer:latest
      imagePullSecrets:
      - name: harbor-secret
```

Ejecuta el consumidor de Kafka, procesa mensajes del topic message y actualiza contadores en Redis. Con una réplica, integra datos con Grafana para visualización en tiempo real.

➔ Deployment rabbitmq-consumer

Proyecto2 > Kubernetes > {..} rabbitmq-consumer-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: rabbitmq-consumer
  namespace: tweets-clima
spec:
  replicas: 1
  selector:
    matchLabels:
      app: rabbitmq-consumer
  template:
    metadata:
      labels:
        app: rabbitmq-consumer
    spec:
      containers:
      - name: rabbitmq-consumer
        image: 34.172.134.232.nip.io/tweets-clima/rabbitmq-consumer
      imagePullSecrets:
      - name: harbor-secret
```

Ejecuta el consumidor de RabbitMQ, procesa mensajes de la cola message y actualiza contadores en Valkey. Con una réplica, asegura la persistencia de datos y se integra con Grafana.

➔ Redis-deployment

Proyecto2 > Kubernetes > {..} redis-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redis-deployment
  namespace: tweets-clima
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redis
  template:
    metadata:
      labels:
        app: redis
    spec:
      containers:
      - name: redis
        image: redis:latest
        ports:
        - containerPort: 6379
        resources:
          requests:
            cpu: "50m"    # 100 milicores
            memory: "256Mi" # 256 MiB
```

Ejecuta Redis como base de datos en memoria para kafka-consumer, almacenando contadores de forma rápida y accesible por Grafana**.**

➔ Valkey-deployment

Proyecto2 > Kubernetes > {} valkey-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: valkey-deployment
  namespace: tweets-clima
spec:
  replicas: 1
  selector:
    matchLabels:
      app: valkey
  template:
    metadata:
      labels:
        app: valkey
    spec:
      containers:
      - name: valkey
        image: bitnami/valkey:latest
        ports:
        - containerPort: 6379
        env:
        - name: ALLOW_EMPTY_PASSWORD
          value: "yes"
        resources:
          requests:
            cpu: "50m"
            memory: "256Mi"
```

Ejecuta Valkey para rabbitmq-consumer, almacenando contadores bajo la licencia BSD y proporcionando integración con Grafana.

➔ Grafana-deployment

Proyecto2 > Kubernetes > {} grafana-deployment.yaml

```yaml
kind: Deployment
metadata:
  name: grafana-deployment
  namespace: tweets-clima
  labels:
    app: grafana
spec:
  replicas: 1
  selector:
    matchLabels:
      app: grafana
  template:
    metadata:
      labels:
        app: grafana
    spec:
      containers:
        - name: grafana
          image: grafana/grafana:latest
          ports:
            - containerPort: 3000
              name: http
          env:
            - name: GF_SECURITY_ADMIN_USER
              value: "admin"
            - name: GF_SECURITY_ADMIN_PASSWORD
              value: "admin123"
```

Se debe ejecutar Grafana para visualizar contadores de Redis y Valkey en paneles como Pie Charts, accesible vía http://grafana.34.67.99.104.nip.io/.

➔ Rust-service

Proyecto2 > Kubernetes > {..} rust-service.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: rust-api-service
  namespace: tweets-clima
spec:
  selector:
    app: rust-api
  ports:
  - protocol: TCP
    port: 80
    targetPort: 8080
  type: ClusterIP
```

Ejecuta un servicio REST escrito en Rust que recibe peticiones HTTP del Ingress (http://34.67.99.104.nip.io/input), procesa los tweets entrantes y los encola en Kafka o RabbitMQ. Con una réplica, maneja el tráfico inicial generado por Locust

➔ Go-grpc-client

Proyecto2 > Kubernetes > {..} go-grpc-client-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: grpc-client
  namespace: tweets-clima
spec:
  replicas: 1
  selector:
    matchLabels:
      app: grpc-client
  template:
    metadata:
      labels:
        app: grpc-client
    spec:
      containers:
      - name: grpc-client
        image: 34.172.134.232.nip.io/tweets-clima/grpc-client
```

Ejecuta un cliente gRPC escrito en Go que se comunica con un servicio gRPC para procesar datos en tiempo real, sincronizando información entre consumidores. Con una réplica, soporta comunicaciones eficientes y de baja latencia.

Preguntas

¿Cómo funciona Kafka?

Kafka es un sistema que maneja mensajes a gran escala en tiempo real. Los productores o aplicaciones envían mensajes a temas, que son como categorías. Estos temas se dividen en particiones para que varios consumidores puedan leer los mensajes al mismo tiempo. Los mensajes se guardan en los brokers, y los consumidores, leen los mensajes de las particiones. Kafka lleva un registro de qué mensajes ya se leyeron para que no se repitan. En el proyecto kafka-consumer lee mensajes del tema message y los procesa para cuantificar datos.

¿Cómo difiere Valkey de Redis?

Valkey y Redis son bases de datos en memoria que se usa para guardar contadores de tweets por país pero hay ciertas diferencias, Valkey es más rápido que Redis porque está optimizado para operaciones como HINCRBY y HGETALL, que se usa para actualizar y leer contadores. Esto significa que rabbitmq-consumer, que escribe en Valkey, puede manejar las actualizaciones más rápido.

Otra diferencia es que Valkey está conectado con rabbitmq-consumer para procesar mensajes de RabbitMQ, mientras que Redis está con kafka-consumer para mensajes de Kafka. Esto separa los datos de cada consumidor, así que los contadores de RabbitMQ y Kafka no se mezclan.

¿Es mejor gRPC que HTTP?

Depende de lo que se necesite pues HTTP es más simple, usa texto como JSON y es compatible con casi todo. gRPC es más rápido porque usa un formato binario y es mejor para aplicaciones donde la velocidad importa mucho, aunque sea un poco más complicado de configurar.

¿Hubo una mejora al utilizar dos réplicas en los deployments de API REST y gRPC? Justifique su respuesta.

Con dos réplicas, las peticiones se reparten entre los pods, lo que reduce la carga en cada uno y evita que el sistema se caiga si uno falla. Esto es útil con las 10,000 peticiones de Locust, ya que dos réplicas pueden manejar más tráfico al mismo tiempo y mantener el sistema estable, aunque no tenemos métricas exactas para confirmarlo.

Para los consumidores, ¿Qué utilizó y por qué?

Para los consumidores se usaron dos herramientas: kafka-consumer con Kafka y rabbitmq-consumer con RabbitMQ. Use Kafka porque puede manejar grandes cantidades de mensajes rápidamente, conectándolo a Redis para guardar los contadores. RabbitMQ lo use porque es confiable para procesar mensajes uno por uno sin perderlos, conectándolo a Valkey para almacenar datos.