GeeksHubs
academy _

Kafka

Clusters Kafka y Replicación

Introducción

- Herramientas básicas para clusters
- Replicación
- Gestión de fallos
- Políticas de confirmación de mensajes
- Herramientas de Replicación

Herramientas básicas para clusters

Herramientas básicas para clusters

Cuando trabajamos con un sistema de colas implementado a través de Kafka, muchas veces nos encontramos con la necesidad de hacer una réplica de datos o simplemente hacer un traspaso de información de un cluster a otro.

Para hacerlo sin complicarnos mucho y que el resultado sea eficaz y eficiente podemos usar herramientas que existen.

Kafka, en sí, nos provee de una herramienta bastante útil llamada Kafka Mirror Maker para hacer replicado de datos sin que esto suponga un gran esfuerzo más que el coste de la configuración inicial, pero también hay otras herramientas como CMAK de la cual una de sus funciones es la gestión multi-cluster.

Herramientas básicas para clusters

Kafka Mirror Maker

Está diseñada para transferir datos de uno o más tópicos entre un cluster origen y otro destino. Se encarga de levantar un proceso que hace las veces de consumidor y productor a la vez, recolectando los mensajes de los tópicos indicados en el cluster origen y llevándolos al cluster destino.

Tiene la responsabilidad de desempeñar tanto de consumidor como de productor, por este motivo es imprescindible indicar la configuración de cada una de estas piezas para llevar a cabo la ejecución del mismo.

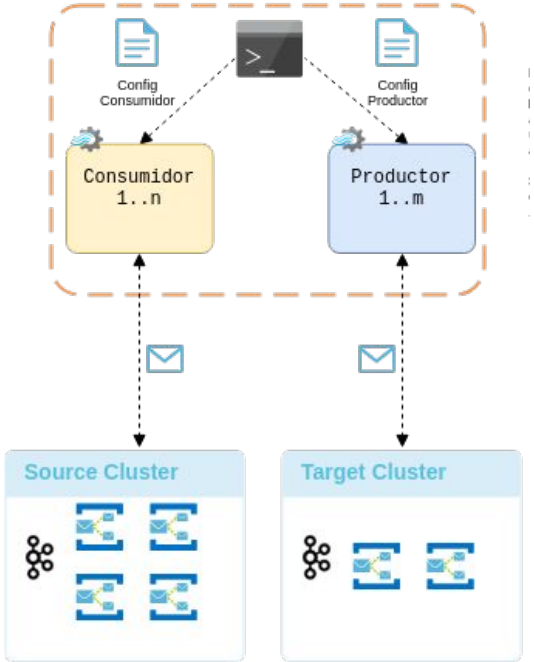
Herramientas básicas para clusters

Kafka Mirror Maker

A la derecha una muestra de arquitectura básica.

Es recomendable disponer de más de una instancia del Kafka Mirror Maker, preferiblemente alojadas en diferentes ubicaciones para garantizar la alta disponibilidad del servicio.

Si el agente replicador reside en el cluster destino, se reduce la latencia.



Herramientas básicas para clusters

CMAK

Antes llamada kafka-manager, es una herramienta para gestionar clusters, sus funciones básicas son las siguientes:

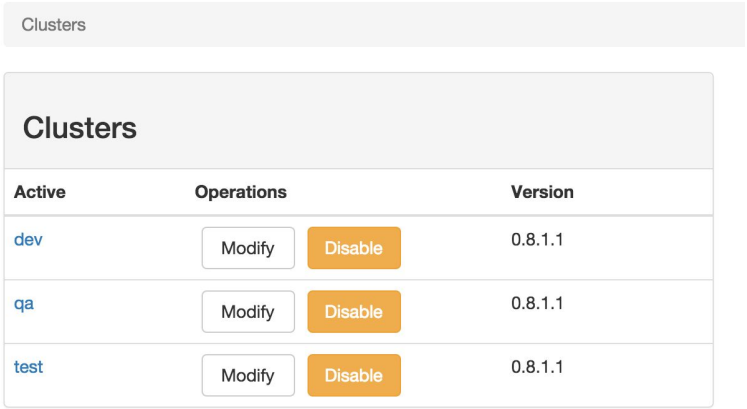
- Gestión multi-cluster.
- Monitoreo transitorio del cluster (topics, consumidores, compensaciones, brokers, distribución de réplicas, distribución de particiones, etc.)
- Elección de réplica prioritaria.
- Topic, partición, gestión de copias (creación de topics, ajuste de parámetros, asignación de copia, etc.)
- Monitoreo de métricas de corredor y nivel de tema.
- Gestión de la información básica del consumidor.

Herramientas básicas para clusters

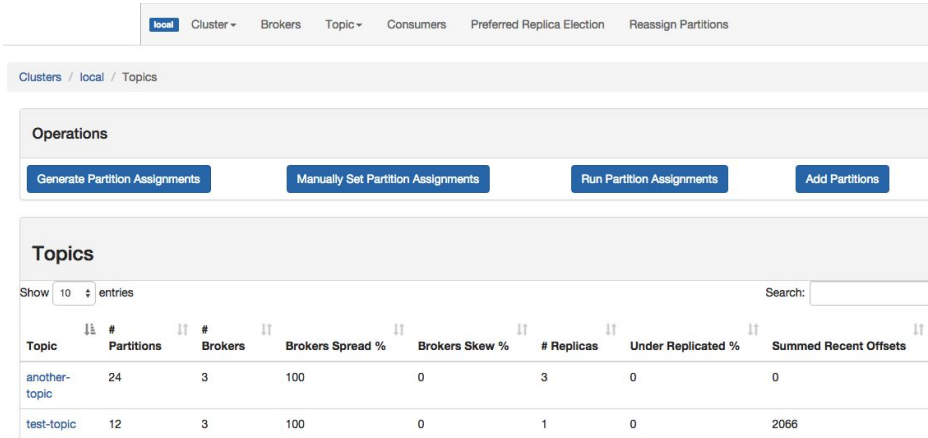
CMAK

Ejemplos de interfaz

Gestión de clusters



Lista de topics



Replicación

Replicación

Una réplica en Kafka consiste en realizar una copia de una partición disponible en otro broker. E**ste mecanismo de replicación permite a Kafka ser tolerante a fallos** y asegura que no hay pérdida de datos, de modo que los mensajes siguen estando disponibles en presencia de fallos.

Kafka está pensado para ser utilizado con **replicación por defecto**, de hecho implementamos temas no replicados como temas replicados donde el factor de replicación es uno.

Cuando existen varias réplicas disponibles, **una de ellas es elegida como líder, y el resto como seguidores.** Las réplicas que siguen al líder y están sincronizadas se marcan como **ISR** (In Sync Replica).

Para que un topic se encuentre en un estado sano, el valor de ISR debe ser igual al factor de replicación.

Replicación

En condiciones de no fallo, cada partición tiene un único líder y cero o más seguidores. El número total de réplicas, incluido el líder, constituye el factor de replicación. Todas las lecturas y escrituras van al líder de la partición. Normalmente, hay muchas más particiones que brokers y los líderes están distribuidos uniformemente entre los brokers.

Los registros de los seguidores son idénticos a los del líder: todos tienen los mismos desplazamientos y los mensajes en el mismo orden, aunque, por supuesto, en un momento dado el líder puede tener algunos mensajes aún no replicados al final de su registro.

Los seguidores consumen mensajes del líder como lo haría un consumidor normal de Kafka y los aplican a su propio registro. El hecho de que los seguidores tiren del líder tiene la agradable propiedad de permitir al seguidor agrupar de forma natural las entradas de registro que están aplicando a su registro.

Gestión de fallos

Gestión de fallos

Como en la mayoría de los sistemas distribuidos, el manejo automático de los fallos requiere una definición precisa de lo que significa que un nodo esté "vivo".

En el caso de Kafka, la vitalidad de un nodo tiene dos condiciones

- Un nodo debe ser capaz de mantener su sesión con ZooKeeper (a través del mecanismo heartbeat de ZooKeeper).

- Si es un seguidor, debe replicar las escrituras que se producen en el líder y no quedarse "demasiado" atrás.

Gestión de fallos

Nos referimos a los nodos que satisfacen estas dos condiciones como "sincronizados" para evitar la vaguedad de "vivos" o "fallidos". El líder lleva la cuenta del conjunto de nodos "sincronizados". Si un seguidor muere, se atasca o se retrasa, el líder lo eliminará de la lista de réplicas sincronizadas.

La determinación de las réplicas atascadas y retrasadas se controla mediante la configuración `replica.lag.time.max.ms`.

En la terminología de los sistemas distribuidos sólo se intenta manejar un modelo de fallos "fallo/recuperación" en el que los nodos dejan de funcionar repentinamente y se recuperan más tarde (quizás sin saber que han muerto).

Kafka no maneja los llamados fallos "bizantinos" en los que los nodos producen respuestas arbitrarias o maliciosas (quizás debido a bugs o juego sucio).

Políticas de confirmación de mensajes

Políticas de confirmación de mensajes

Ahora podemos definir con mayor precisión que **un mensaje se considera comprometido cuando todas las réplicas sincronizadas para esa partición lo han aplicado a su registro**.

Sólo los mensajes comprometidos se entregan al consumidor. Esto significa que el consumidor no tiene que preocuparse por la posibilidad de ver un mensaje que podría perderse si el líder falla.

Los productores, por otro lado, tienen la opción de esperar a que el mensaje se comprometa o no, dependiendo de su preferencia por el equilibrio entre la latencia y la durabilidad. Esta preferencia está controlada por la configuración de Acks que utiliza el productor.

Hay que tener en cuenta que los temas tienen una configuración para el "número mínimo" de réplicas en sincronización que se comprueba cuando el productor solicita el reconocimiento de que un mensaje se ha escrito en el conjunto completo de réplicas en sincronización.

Políticas de confirmación de mensajes

Si el productor solicita un acuse de recibo menos estricto, el mensaje puede ser consignado, y consumido, incluso si el número de réplicas in-sync es inferior al mínimo (por ejemplo, puede ser tan bajo como el líder).

La garantía que ofrece Kafka es que un mensaje comprometido no se perderá, siempre que haya al menos una réplica en sincronía viva, en todo momento.

Kafka permanecerá disponible en presencia de fallos de nodos después de un breve período de conmutación por error, pero puede no permanecer disponible en presencia de particiones de la red.

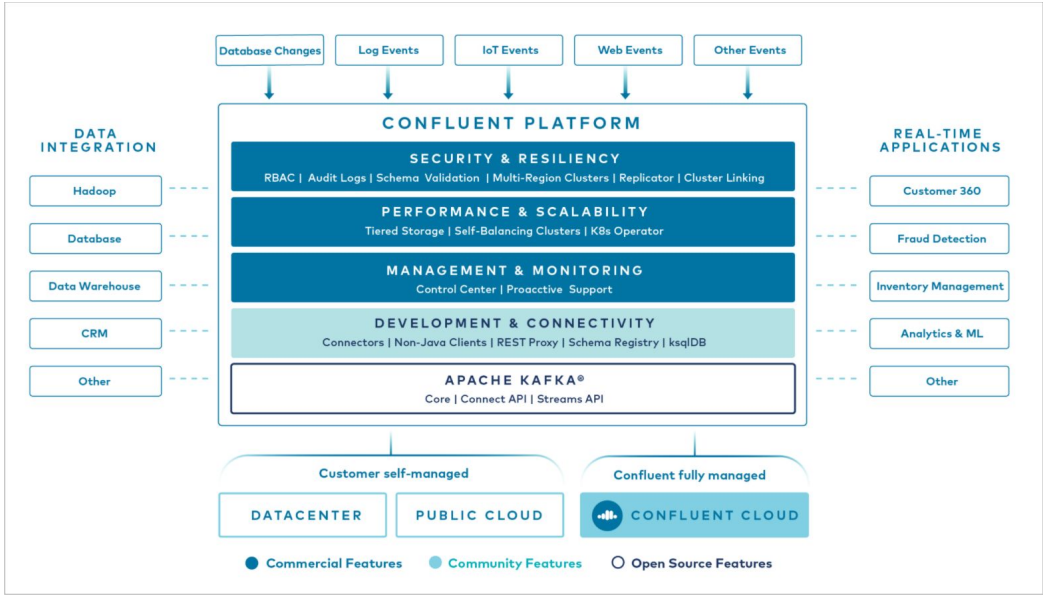
Herramientas de replicación

Herramientas de replicación

Confluent Platform for Kafka

Es una plataforma de flujo de datos a gran escala que le permite acceder, almacenar y gestionar fácilmente los datos como flujos continuos en tiempo real.

Creada por los creadores originales de Apache Kafka, Confluent amplía las ventajas de Kafka con funciones de nivel empresarial al tiempo que elimina la carga de la gestión o la supervisión de Kafka.



Herramientas de replicación

Confluent Platform for Kafka

Confluent Replicator facilita más que nunca el mantenimiento de varios clusters de Kafka en varios centros de datos. La gestión de la replicación de datos y la configuración de topics entre centros de datos permite casos de uso como:

- Despliegues geo-localizados activo-activo: Permite a los usuarios acceder a un centro de datos cercano para optimizar su arquitectura de baja latencia y alto rendimiento.
- Análisis centralizados: Agrega los datos de varios clústeres de Kafka en una sola ubicación para la analítica de toda la organización.
- Migración a la nube: Utiliza Kafka para sincronizar los datos entre las aplicaciones on-prem y las implementaciones en la nube.

Puede utilizar Replicator para configurar y gestionar la replicación para todos estos escenarios desde el Centro de Control de Confluent o desde las herramientas de línea de comandos.