Prácticas recomendadas para Confluent Kafka

NetApp Solutions

NetApp
April 23, 2024

This PDF was generated from https://docs.netapp.com/es-es/netapp-solutions/data-analytics/confluent-kafka-introduction.html on April 23, 2024. Always check docs.netapp.com for the latest.

Tabla de contenidos

Prácticas recomendadas para Confluent Kafka . . . 1
TR-4912: Directrices de mejores prácticas para el almacenamiento por niveles de Confluent Kafka con NetApp . . . 1
Detalles de la arquitectura de la solución . . . 3
Información general de la tecnología . . . 4
Verificación confluente . . . 11
Pruebas de rendimiento con escalabilidad . . . 14
Conector confluent s3 . . . 16
Clústeres de equilibrio automático confluente . . . 25
Directrices de prácticas recomendadas . . . 25
Ajuste de tamaño . . . 27
Conclusión . . . 30

Prácticas recomendadas para Confluent Kafka

TR-4912: Directrices de mejores prácticas para el almacenamiento por niveles de Confluent Kafka con NetApp

Karthikeyan Nagalingam, Joseph Kandatillparambil, NetApp Rankesh Kumar, Confluente

Apache Kafka es una plataforma de transmisión de eventos distribuida por la comunidad capaz de gestionar billones de eventos al día. Inicialmente concebido como una cola de mensajería, Kafka se basa en la abstracción de un registro de confirmación distribuido. Desde que LinkedIn creó y obtuvo el código abierto en 2011, Kafka ha pasado de ser una cola de mensajes a una plataforma de transmisión de eventos completa. Con fluidez, entrega la distribución de Apache Kafka con la plataforma Confluent. La plataforma Confluent complementa a Kafka con otras funciones comerciales y comunitarias diseñadas para mejorar la experiencia en streaming de operadores y desarrolladores en producción a gran escala.

Este documento describe las directrices de mejores prácticas sobre el uso de Confluent Tiered Storage en una oferta de almacenamiento de objetos de NetApp mediante el siguiente contenido:

- Verificación fluida con el almacenamiento de objetos de NetApp – StorageGRID de NetApp
- Pruebas de rendimiento del almacenamiento por niveles
- Directrices de prácticas recomendadas para hablar con fluidez de los sistemas de almacenamiento de NetApp

¿Por qué Confluent Tiered Storage?

Confluent se ha convertido en la plataforma de streaming en tiempo real predeterminada para muchas aplicaciones, especialmente para cargas de trabajo de Big Data, análisis y streaming. El almacenamiento por niveles permite a los usuarios separar el procesamiento del almacenamiento en la plataforma Confluent. Hace que el almacenamiento de datos sea más rentable, le permite almacenar cantidades de datos virtualmente infinitas y escalar las cargas de trabajo cuanto antes (o incluso reduciendo) bajo demanda, además facilita las tareas administrativas como los datos y el reequilibrio de inquilinos. Los sistemas de almacenamiento compatibles con S3 pueden aprovechar todas estas funcionalidades para democratizar los datos con todos los eventos en un mismo lugar, eliminando la necesidad de realizar una ingeniería de datos compleja. Para obtener más información sobre por qué se debe usar el almacenamiento por niveles para Kafka, consulte "Este artículo de Confluent".

¿Por qué utilizar StorageGRID de NetApp para el almacenamiento por niveles?

StorageGRID es una plataforma de almacenamiento de objetos líder en el sector por parte de NetApp. StorageGRID es una solución de almacenamiento basada en cloud definida por software compatible con las API de objetos estándares del sector, incluida la API de Amazon simple Storage Service (S3). StorageGRID almacena y gestiona datos no estructurados a escala para proporcionar un almacenamiento de objetos seguro y duradero. El contenido se sitúa en el lugar, en el momento y el nivel de almacenamiento adecuados, optimizando los flujos de trabajo y reduciendo los costes de la distribución global de medios enriquecidos.

El mayor diferenciador de StorageGRID es su motor de políticas de gestión del ciclo de vida de la información

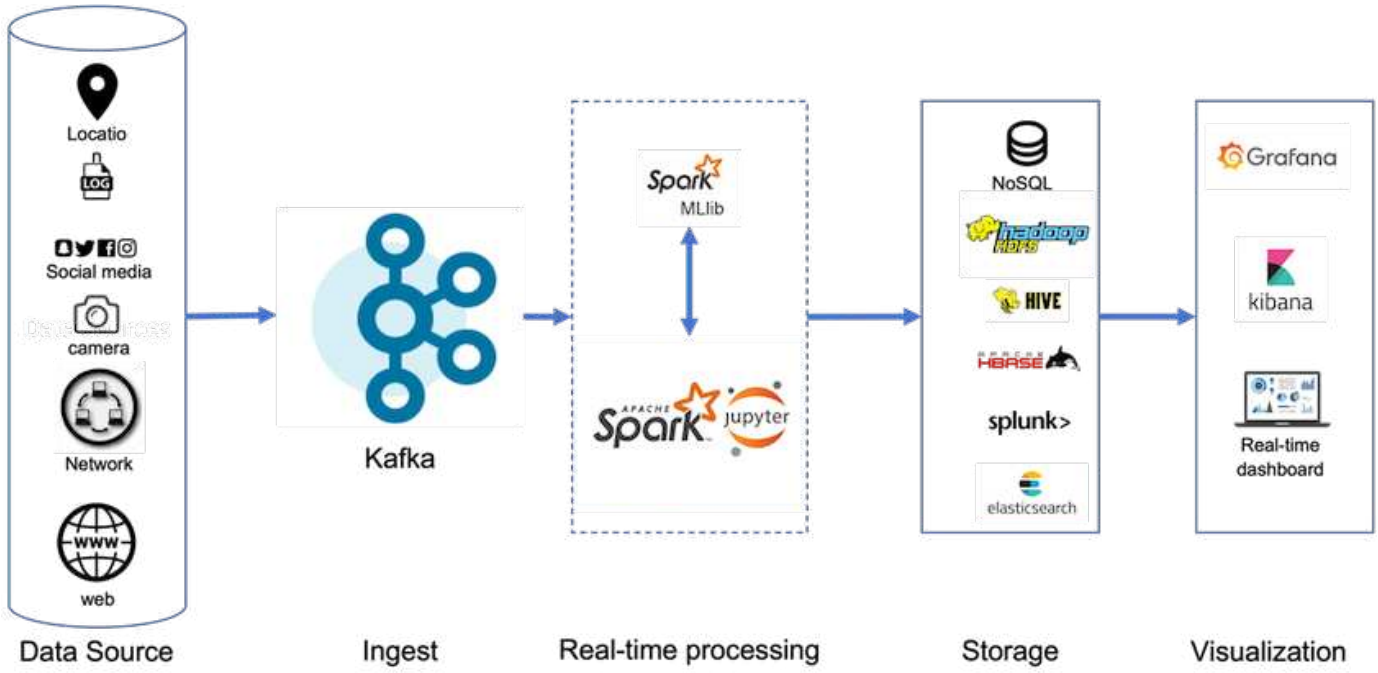
(ILM) que posibilita la gestión del ciclo de vida de los datos condicionada por políticas. El motor de políticas puede utilizar metadatos para gestionar cómo se almacenan los datos a lo largo de su vida útil para optimizar inicialmente el rendimiento y optimizar automáticamente los costes y la durabilidad a medida que los datos envejecen.

Habilitar el almacenamiento multinivel fluido

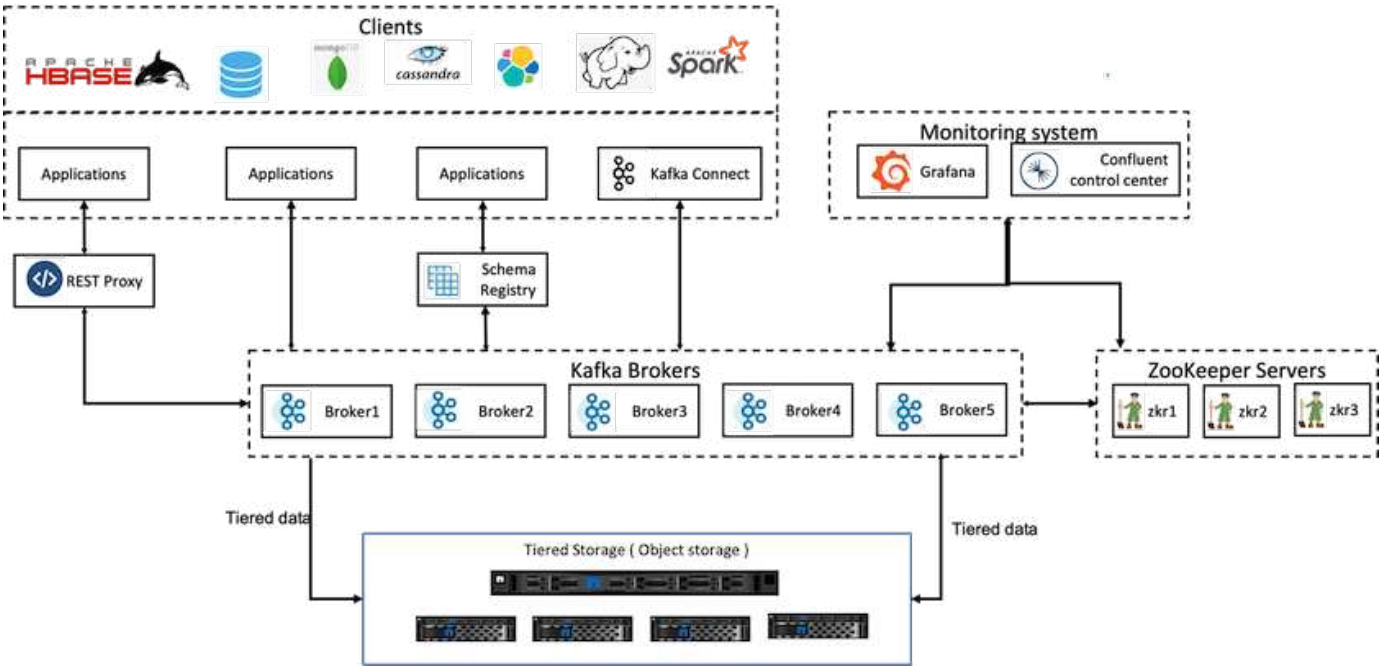
La idea básica del almacenamiento por niveles es separar las tareas del almacenamiento de datos del procesamiento de los datos. Con esta separación, resulta mucho más fácil que el nivel de almacenamiento de datos y el nivel de procesamiento de datos se escalen de forma independiente.

Una solución de almacenamiento por niveles para Confluent debe enfrentarse a dos factores. En primer lugar, debe trabajar o evitar las propiedades comunes de consistencia y disponibilidad de los almacenes de objetos, como incoherencias en las operaciones de LISTA y falta de disponibilidad de objetos ocasionales. En segundo lugar, debe manejar correctamente la interacción entre el almacenamiento organizado por niveles y el modelo de tolerancia a fallos y replicación de Kafka, incluida la posibilidad de que los líderes zombis sigan manteniendo rangos de desviación de nivel. El almacenamiento de objetos de NetApp proporciona tanto la disponibilidad de objetos consistente como el modelo de alta disponibilidad hacen que el almacenamiento cansado esté disponible para gamas de compensación por nivel. El almacenamiento de objetos de NetApp proporciona disponibilidad de objetos coherente y un modelo de alta disponibilidad para que el almacenamiento cansado esté disponible para equidistancias de nivel.

Con el almacenamiento por niveles, puede utilizar plataformas de alto rendimiento para lecturas y escrituras de baja latencia cerca del final de los datos en streaming. Además, puede utilizar almacenes de objetos más económicos y escalables como StorageGRID de NetApp para lecturas históricas de alto rendimiento. También disponemos de una solución técnica para Spark con la controladora de almacenamiento de netapp y nuestros detalles están aquí. La siguiente figura muestra cómo Kafka se adapta a una canalización de análisis en tiempo real.



La figura siguiente muestra cómo StorageGRID de NetApp se integra como nivel de almacenamiento de objetos de Confluent Kafka.



Detalles de la arquitectura de la solución

Esta sección trata sobre el hardware y el software utilizados para la verificación de Confluent. Esta información se aplica a la puesta en marcha de plataformas Confluent con el almacenamiento de NetApp. La siguiente tabla recoge los componentes básicos y la arquitectura probada de la solución.

Componentes de la solución	Detalles
Confluente Kafka versión 6.2	• Tres zookeepers • Cinco servidores de broker • Cinco servidores de herramientas • Un Grafana • Un centro de control
Linux (ubuntu 18.04)	Todos los servidores
StorageGRID de NetApp para el almacenamiento por niveles	• Software StorageGRID • 1 x SG1000 (equilibrador de carga) • 4 x SGF6024 • 4 x 24 x 800 SSD • Protocolo S3 • 4 x 100 GbE (conectividad de red entre instancias de intermediarios y StorageGRID)
15 servidores PRIMERGY RX2540 de Fujitsu	Cada una equipada con: * 2 CPU, 16 núcleos físicos en total * Intel Xeon * 256 GB de memoria física * 100 GbE de doble puerto

Información general de la tecnología

En este apartado se describe la tecnología utilizada en esta solución.

StorageGRID de NetApp

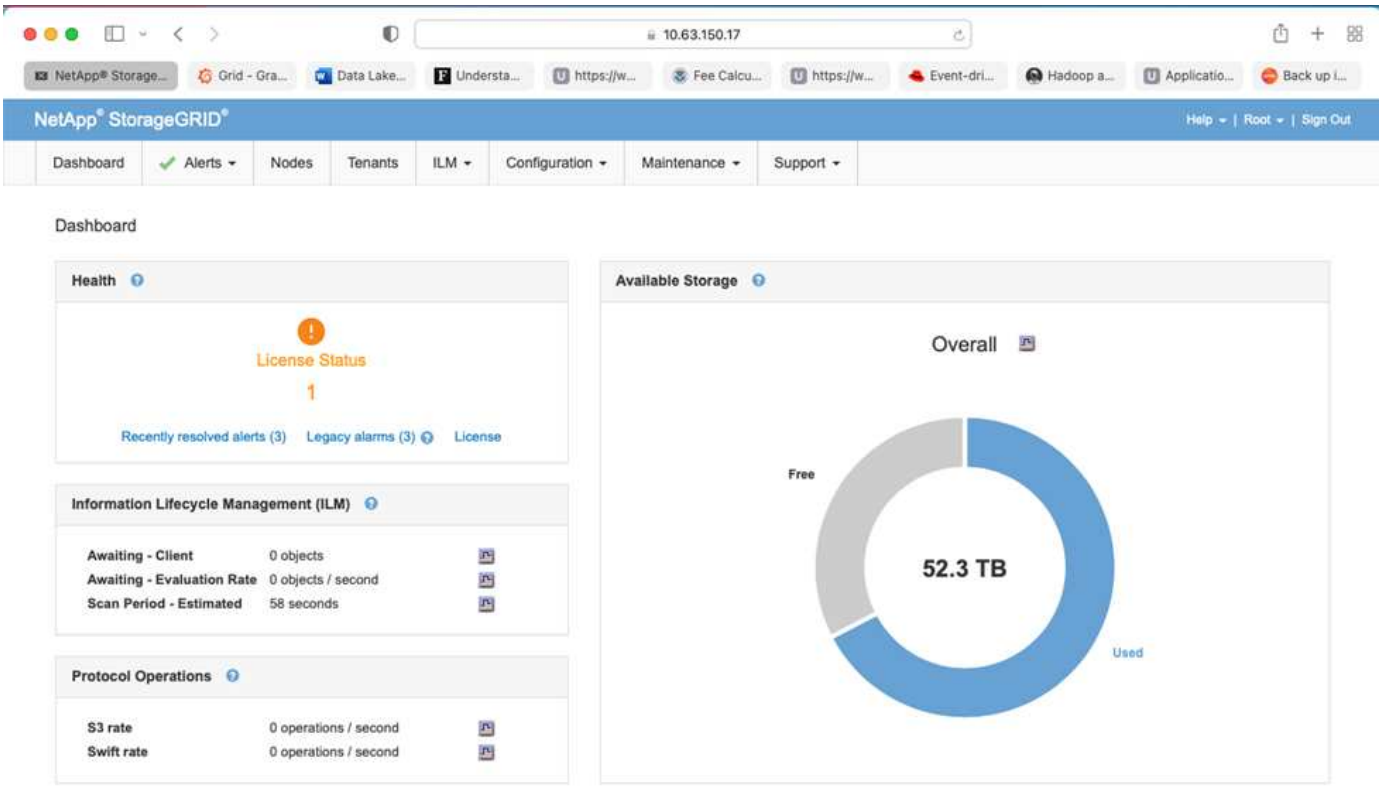
StorageGRID de NetApp es una plataforma de almacenamiento de objetos rentable y de alto rendimiento. Mediante el almacenamiento por niveles, la mayoría de los datos de Confluent Kafka, que se almacenan en el almacenamiento local o el almacenamiento SAN del intermediario, se descargan al almacén de objetos remoto. Esta configuración da lugar a importantes mejoras operativas al reducir el tiempo y el coste necesarios para reequilibrar, ampliar o reducir clústeres, o sustituir un intermediario fallido. El almacenamiento de objetos desempeña un papel importante en la gestión de datos que residen en el nivel de almacén de objetos; por este motivo, es importante elegir el almacenamiento de objetos adecuado.

StorageGRID ofrece gestión de datos inteligente y global condicionada por políticas mediante una arquitectura de grid distribuida basada en nodos. Simplifica la gestión de petabytes de datos no estructurados y de miles de millones de objetos mediante su espacio de nombre de objetos global omnipresente combinado con funciones sofisticadas de gestión de datos. El acceso en una sola llamada a objetos abarca varios sitios y simplifica las arquitecturas de alta disponibilidad al tiempo que garantiza un acceso continuo a los objetos, independientemente de las interrupciones del servicio del sitio o de la infraestructura.

El multi-tenancy permite que varias aplicaciones de datos empresariales y cloud no estructurados se puedan mantener de forma segura dentro del mismo grid, lo que aumenta el retorno de la inversión y los casos de uso de StorageGRID de NetApp. Puede crear múltiples niveles de servicio con políticas de ciclo de vida de objetos condicionados por metadatos, y así optimizar la durabilidad, la protección, el rendimiento y la localidad en múltiples geografías. Los usuarios pueden ajustar las políticas de gestión de datos y supervisar y aplicar los límites de tráfico para realinear el entorno de datos sin interrupciones a medida que cambien sus requisitos en los entornos TECNOLÓGICOS en constante cambio.

Gestión sencilla con Grid Manager

Grid Manager de StorageGRID es una interfaz gráfica basada en navegador que permite configurar, gestionar y supervisar el sistema StorageGRID en ubicaciones distribuidas por todo el mundo en una misma consola.



Puede realizar las siguientes tareas con la interfaz de Grid Manager de StorageGRID:

- Gestione repositorios de objetos distribuidos de varios petabytes como imágenes, vídeo y registros.
- Supervise nodos de grid y servicios para garantizar la disponibilidad de los objetos.
- Gestione la ubicación de datos de objetos a lo largo del tiempo mediante las reglas de gestión de ciclo de vida de la información (ILM). Estas reglas rigen lo que ocurre con los datos de un objeto después de ingerirlo, cómo se protege de la pérdida, dónde se almacenan los datos de objetos y durante cuánto tiempo.
- Supervisar las transacciones, el rendimiento y las operaciones dentro del sistema.

Políticas de gestión del ciclo de vida de la información

StorageGRID cuenta con políticas de gestión de datos flexibles, que incluyen mantener copias de réplicas de los objetos y usar esquemas de EC (codificación de borrado) como 2+1 y 4+2 (entre otros) para almacenar los objetos, en función de requisitos específicos de rendimiento y protección de datos. A medida que las cargas de trabajo y los requisitos cambian con el tiempo, es común que las políticas de ILM también cambien con el tiempo. La modificación de las políticas de ILM es una función fundamental, por lo que permite a los clientes de StorageGRID adaptarse a su entorno en constante cambio de forma rápida y sencilla. Compruebe la "Política de ILM" y.. "Reglas de ILM" configure en StorageGRID.

Rendimiento

StorageGRID escala el rendimiento añadiendo más nodos de almacenamiento, que pueden ser máquinas virtuales, dispositivos físicos o específicos como el "SG5712, SG5760, SG6060 O SGF6024". En nuestras pruebas, superamos los requisitos de rendimiento clave de Apache Kafka con un grid de tres nodos de tamaño mínimo con el dispositivo SGF6024. A medida que los clientes escalan su clúster Kafka con agentes adicionales, pueden añadir más nodos de almacenamiento para aumentar el rendimiento y la capacidad.

Equilibrador de carga y configuración de punto final

Los nodos de administración en StorageGRID proporcionan la interfaz de usuario (interfaz de usuario) de Grid Manager y el extremo de la API DE REST para ver, configurar y gestionar el sistema StorageGRID, así como registros de auditoría para realizar un seguimiento de la actividad del sistema. Para proporcionar un extremo de S3 de alta disponibilidad para el almacenamiento por niveles de Confluent Kafka, implementamos el equilibrador de carga StorageGRID, que se ejecuta como servicio en nodos de administración y nodos de puerta de enlace. Además, el equilibrador de carga también gestiona el tráfico local y habla con el GSLB (equilibrio de carga de servidor global) para ayudar en la recuperación ante desastres.

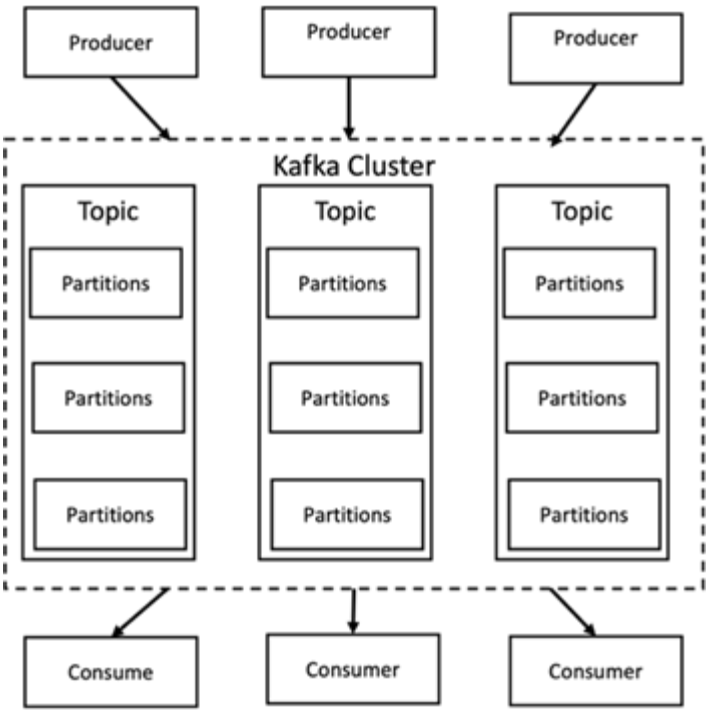
Para mejorar aún más la configuración de los extremos, StorageGRID proporciona políticas de clasificación del tráfico integradas en el nodo de administración, le permite supervisar el tráfico de la carga de trabajo y aplica diversos límites de calidad de servicio a sus cargas de trabajo. Las políticas de clasificación del tráfico se aplican a los extremos en el servicio StorageGRID Load Balancer para los nodos de puerta de enlace y los nodos de administración. Estas políticas pueden ayudar a configurar y supervisar el tráfico.

Clasificación del tráfico en StorageGRID

StorageGRID cuenta con la funcionalidad de calidad de servicio incorporada. Las políticas de clasificación del tráfico pueden ayudar a supervisar diferentes tipos de tráfico S3 que provienen de una aplicación cliente. A continuación, puede crear y aplicar políticas para limitar este tráfico en función del ancho de banda de entrada/salida, el número de solicitudes simultáneas de lectura/escritura o la tasa de solicitud de lectura/escritura.

Apache Kafka

Apache Kafka es una implementación de marco de un bus de software que utiliza procesamiento de secuencias escrito en Java y Scala. Su objetivo es proporcionar una plataforma unificada, de alto rendimiento y baja latencia para manejar fuentes de datos en tiempo real. Kafka se puede conectar a un sistema externo para exportar e importar datos a través de Kafka Connect y proporciona flujos Kafka, una biblioteca de procesamiento de secuencias Java. Kafka utiliza un protocolo binario basado en TCP optimizado para la eficiencia y se basa en una abstracción de «conjunto de mensajes» que agrupa naturalmente los mensajes juntos para reducir la sobrecarga de la red ida y vuelta. De este modo, permite realizar operaciones de disco secuenciales de mayor tamaño, paquetes de red de mayor tamaño y bloques de memoria contiguos, con lo que Kafka puede convertir un flujo de ráfagas de escrituras de mensajes aleatorios en escrituras lineales. La figura siguiente muestra el flujo de datos básicos de Apache Kafka.



Kafka almacena mensajes de valor clave que provienen de un número arbitrario de procesos llamados productores. Los datos pueden particionarse en diferentes particiones dentro de diferentes temas. Dentro de una partición, los mensajes se ordenan estrictamente por sus desplazamientos (la posición de un mensaje dentro de una partición) y se indexan y almacenan junto con una Marca de hora. Otros procesos llamados consumidores pueden leer mensajes de las particiones. Para el procesamiento en streaming, Kafka ofrece la API Streams, que permite escribir aplicaciones Java que consumen datos de Kafka y escribir resultados de nuevo en Kafka. Apache Kafka también funciona con sistemas de procesamiento de secuencias externas como Apache Apex, Apache Flink, Apache Spark, Apache Storm y Apache NIFI.

Kafka se ejecuta en un clúster de uno o más servidores (denominados intermediarios), y las particiones de todos los temas se distribuyen entre los nodos del clúster. Además, las particiones se replican en varios agentes. Esta arquitectura permite a Kafka distribuir flujos masivos de mensajes de forma tolerante a fallos y le ha permitido reemplazar algunos de los sistemas de mensajería convencionales como Java Message Service (JMS), Advanced Message Queue Server Protocol (AMQP), etc. Desde la versión 0.11.0.0, Kafka ofrece escrituras transaccionales, que proporcionan un procesamiento de secuencias exactamente una vez con la API Streams.

Kafka apoya dos tipos de temas: Regular y compactado. Los temas regulares se pueden configurar con un tiempo de retención o con un enlace al espacio. Si hay registros antiguos que superan el tiempo de retención especificado o si se supera el límite de espacio para una partición, se permite a Kafka eliminar los datos antiguos para liberar espacio de almacenamiento. Los temas se configuran por defecto con un tiempo de retención de 7 días, pero también es posible almacenar los datos de forma indefinida. Para temas compactados, los registros no caducan en función de los límites de tiempo o espacio. En su lugar, Kafka trata los mensajes posteriores como actualizaciones del mensaje más antiguo con la misma clave y garantiza que nunca se elimine el mensaje más reciente por clave. Los usuarios pueden eliminar mensajes por completo escribiendo un mensaje de desecho llamado con el valor nulo para una clave específica.

Hay cinco API principales en Kafka:

- **API de productores.** permite que una aplicación publique flujos de registros.
- **Consumer API.** permite a una aplicación suscribirse a temas y procesos flujos de registros.
- **API de conector.** ejecuta las API reutilizables de productores y consumidores que pueden vincular los

temas a las aplicaciones existentes.

- **Streams API.** esta API convierte los flujos de entrada a salida y produce el resultado.
- **Admin API.** se utiliza para administrar temas de Kafka, corredores y otros objetos de Kafka.

Las API de consumidores y productores se basan en el protocolo de mensajería Kafka y ofrecen una implementación de referencia para clientes consumidores y productores de Kafka en Java. El protocolo de mensajería subyacente es un protocolo binario que los desarrolladores pueden utilizar para escribir sus propios clientes consumidores o productores en cualquier lenguaje de programación. Esto libera a Kafka del ecosistema Java Virtual Machine (JVM). En el wiki de Apache Kafka se mantiene una lista de clientes no Java disponibles.

Casos de uso de Apache Kafka

Apache Kafka es más popular en mensajería, seguimiento de la actividad de sitios web, métricas, agregación de registros, procesamiento de flujos, origen de eventos y registro de confirmación.

- Kafka ha mejorado el rendimiento, las particiones integradas, la replicación y la tolerancia a fallos, lo que lo convierte en una buena solución para aplicaciones de procesamiento de mensajes a gran escala.
- Kafka puede reconstruir las actividades de un usuario (vistas de página, búsquedas) en una canalización de seguimiento como un conjunto de fuentes de suscripción-publicación en tiempo real.
- Kafka se utiliza a menudo para datos de supervisión operativa. Esto implica agregar estadísticas de aplicaciones distribuidas para producir fuentes centralizadas de datos operativos.
- Muchas personas usan Kafka como reemplazo de una solución de agregación de registros. La agregación de registros normalmente recopila archivos de registro físicos fuera de los servidores y los coloca en un lugar central (por ejemplo, un servidor de archivos o HDFS) para su procesamiento. Kafka abstrae los detalles de los archivos y proporciona una abstracción más limpia de los datos de registro o evento como una secuencia de mensajes. De este modo, se consigue un procesamiento de menor latencia y una compatibilidad más sencilla con múltiples fuentes de datos y consumo de datos distribuido.
- Muchos usuarios de los datos de procesos de Kafka en las canalizaciones de procesamiento, que consisten en múltiples etapas, en las que los datos de entrada sin procesar se consumen a partir de temas de Kafka y luego se agregan, enriquecen o transforman de otro modo en nuevos temas para un mayor consumo o procesamiento de seguimiento. Por ejemplo, una canalización de procesamiento para recomendar artículos de noticias podría rastrear el contenido del artículo de fuentes RSS y publicarlo en un tema de "artículos". Un procesamiento posterior puede normalizar o deduplicar este contenido, publicar el contenido del artículo limpio en un nuevo tema, y una fase final de procesamiento puede intentar recomendar este contenido a los usuarios. Estas canalizaciones de procesamiento crean gráficos de flujos de datos en tiempo real basados en temas individuales.
- La externalización de eventos es un estilo de diseño de aplicación para el que los cambios de estado se registran como una secuencia de registros ordenada por tiempo. La compatibilidad de Kafka con datos de registro almacenados muy grandes lo convierte en un entorno de administración excelente para una aplicación integrada en este estilo.
- Kafka puede servir como una especie de registro de confirmación externo para un sistema distribuido. El registro ayuda a replicar datos entre nodos y actúa como mecanismo de repetición de la sincronización de los nodos con errores para restaurar sus datos. La función de compactación de registros de Kafka ayuda a dar soporte a este caso de uso.

Confluente

Confluent Platform es una plataforma lista para las empresas que completa Kafka con funcionalidades avanzadas diseñadas para ayudar a acelerar el desarrollo y la conectividad de las aplicaciones, permitir transformaciones a través del procesamiento de secuencias, simplificar las operaciones empresariales a

escala y cumplir los estrictos requisitos de arquitectura. Diseñado 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 supervisión de Kafka. Hoy en día, más del 80 % de las empresas Fortune 100 cuentan con tecnología de transmisión de datos; la mayoría de ellas utilizan Confluent.

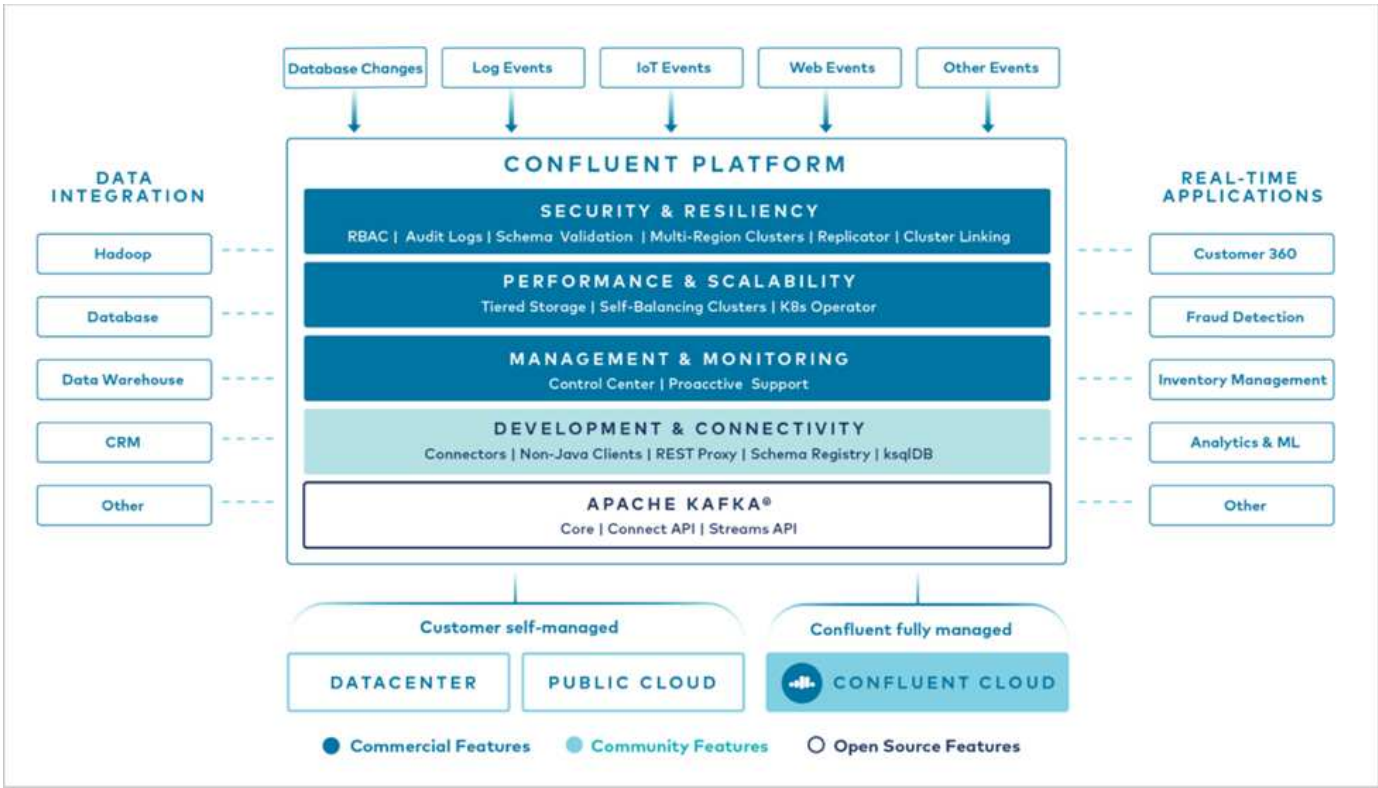
¿Por qué confluente?

Mediante la integración de datos históricos y en tiempo real en una única fuente central de verdad, Confluent facilita la creación de una categoría completamente nueva de aplicaciones modernas condicionadas por eventos, obtener una canalización de datos universal y desbloquear nuevos casos de uso potentes con total escalabilidad, rendimiento y fiabilidad.

¿Para qué se utiliza Confluent?

Gracias a la plataforma Confluent podrá centrarse en cómo obtener valor empresarial de sus datos en lugar de preocuparse por los mecanismos subyacentes, como por ejemplo, cómo se transportan datos o se integran entre sistemas dispares. En concreto, Confluent Platform simplifica la conexión de fuentes de datos a Kafka, la creación de aplicaciones de streaming y la protección, supervisión y gestión de la infraestructura de Kafka. En la actualidad, Confluent Platform se utiliza para una amplia variedad de casos de uso en numerosos sectores, desde servicios financieros, ventas al por menor de canal integral y coches autónomos, hasta detección de fraude, Microservicios y el Internet de las cosas.

En la siguiente figura, se muestran los componentes de la plataforma Confluent Kafka.



Descripción general de la tecnología de transmisión de eventos de Confluent

En el centro de la plataforma Confluente lo es "Apache Kafka", la plataforma de transmisión distribuida de código abierto más popular. Las capacidades clave de Kafka son las siguientes:

- Publicar y suscribirse a flujos de registros.

- Almacene flujos de registros de forma tolerante a fallos.
- Procesar flujos de registros.

Lista para usar, Confluent Platform también incluye registro de esquemas, proxy REST, un total de más de 100 conectores Kafka predefinidos y ksqlDB.

Descripción general de las características empresariales de la plataforma Confluent

- * Confluent Control Center.* un sistema basado en GUI para la gestión y monitorización de Kafka. Le permite gestionar fácilmente Kafka Connect y crear, editar y gestionar conexiones a otros sistemas.
- * Confluent for Kubernetes.* Confluent for Kubernetes es un operador de Kubernetes. Los operadores de Kubernetes amplían las funcionalidades de orquestación de Kubernetes, al proporcionar las funciones y requisitos únicos para una aplicación de plataforma específica. En el caso de la plataforma con fluidez, esto incluye simplificar en gran medida el proceso de puesta en marcha de Kafka en Kubernetes y automatizar las tareas habituales del ciclo de vida de la infraestructura.
- * Conectores Confluent a Kafka.* los conectores usan la API Kafka Connect para conectar Kafka a otros sistemas como bases de datos, almacenes de clave-valor, índices de búsqueda y sistemas de archivos. Confluent Hub tiene conectores descargables para las fuentes de datos y los sumideros más populares, incluidas versiones totalmente probadas y compatibles de estos conectores con Confluent Platform. Encontrará más información "aquí".
- **Clústeres de equilibrio automático.** proporciona equilibrio de carga automatizado, detección de fallos y autorreparación. Proporciona soporte para agregar o decomisionar intermediarios según sea necesario, sin realizar ajustes manuales.
- **Confluent cluster linkando.** conecta directamente los clusters y refleja temas de un cluster a otro a través de un puente de enlace. La vinculación de clústeres simplifica la configuración de implementaciones en varios centros de datos, varios clústeres y nube híbrida.
- * Confluent auto data equilibrador.* supervisa su clúster para el número de corredores, el tamaño de particiones, el número de particiones y el número de líderes dentro del clúster. Permite mover datos para crear una carga de trabajo uniforme en su clúster, a la vez que se desregula el tráfico del reequilibrio para minimizar el efecto en las cargas de trabajo de producción al mismo tiempo que se reequilibran.
- * Confluent replicator.* hace más fácil que nunca mantener múltiples clústeres de Kafka en varios centros de datos.
- **Almacenamiento Tiered.** ofrece opciones para el almacenamiento de grandes volúmenes de datos Kafka con su proveedor de cloud preferido, reduciendo así la carga y los costes operativos. Con un almacenamiento por niveles, puede mantener los datos en un almacenamiento de objetos rentable y a los agentes de escalado solo cuando necesite más recursos informáticos.
- * Confluent JMS Client.* Confluent Platform incluye un cliente compatible con JMS para Kafka. Este cliente Kafka implementa la API estándar JMS 1.1, utilizando los agentes Kafka como back-end. Esto resulta útil si tiene aplicaciones heredadas con JMS y desea reemplazar el agente de mensajes JMS existente con Kafka.
- * Proxy de Confluent MQTT.* proporciona una forma de publicar datos directamente a Kafka desde dispositivos MQTT y puertas de enlace sin necesidad de un intermediario de MQTT en el medio.
- * Plugins de seguridad Confluent.* los plugins de seguridad Confluent se utilizan para agregar capacidades de seguridad a varias herramientas y productos de Confluent Platform. Actualmente, hay un plugin disponible para el proxy de REST de Confluent que ayuda a autenticar las solicitudes entrantes y propagar el principal autenticado a solicitudes a Kafka. Esto permite a los clientes proxy DE Confluent REST utilizar las funciones de seguridad multitenant del agente Kafka.

Verificación confluente

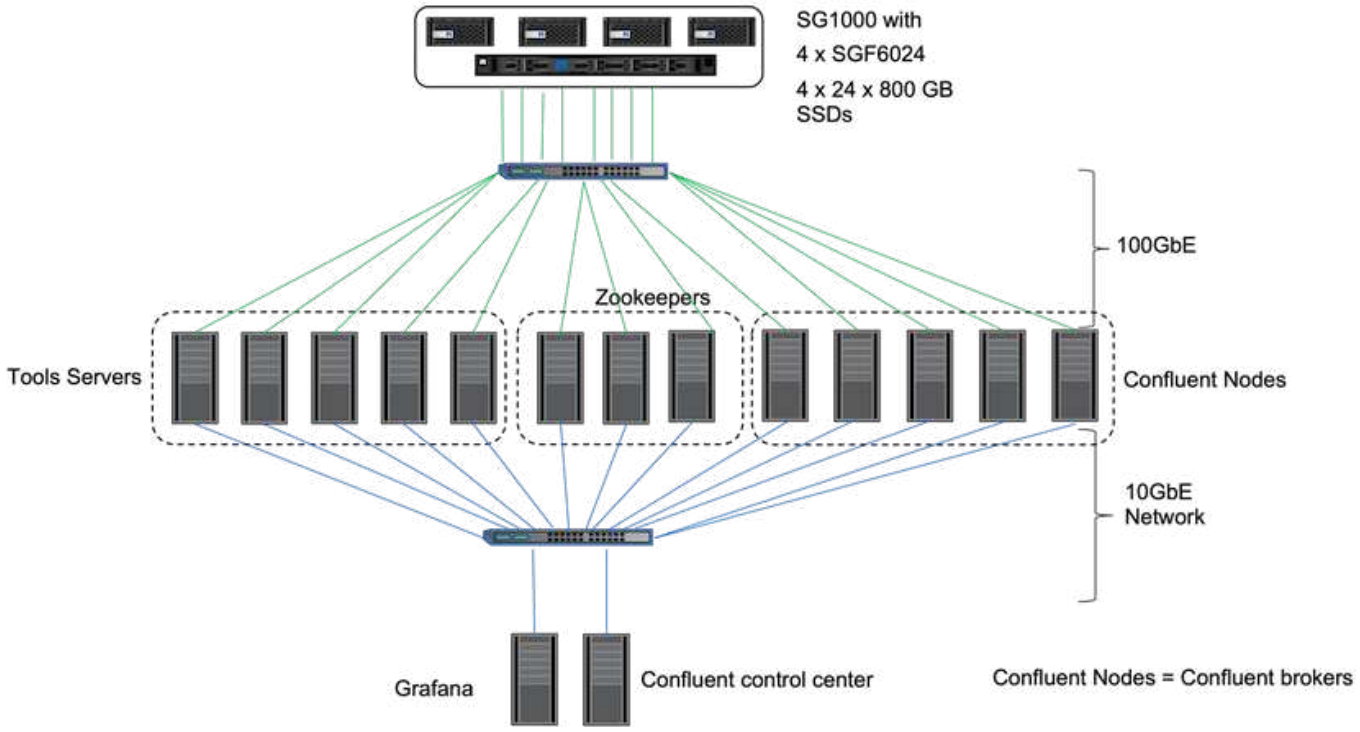
Hemos realizado la verificación con el almacenamiento en niveles de 6.2 de la plataforma Confluent en StorageGRID de NetApp. Los equipos de NetApp y Confluent trabajaron juntos en esta verificación y ejecutaron los casos de prueba necesarios para la verificación.

Configuración de la plataforma Confluent

Utilizamos la siguiente configuración para la verificación.

Para la verificación, utilizamos tres zoeepers, cinco brokers, cinco servidores de ejecución de script de prueba, servidores de herramientas con nombres con 256 GB de RAM y 16 CPU. En el caso del almacenamiento de NetApp, utilizamos StorageGRID con un equilibrador de carga SG1000 con cuatro SGF6024s. El almacenamiento y los agentes se conectaron a través de conexiones de 100 GbE.

En la siguiente figura, se muestra la topología de red de la configuración utilizada para la verificación fluida.



Los servidores de herramientas actúan como clientes de aplicación que envían solicitudes a nodos Confluent.

Configuración de almacenamiento por niveles confluente

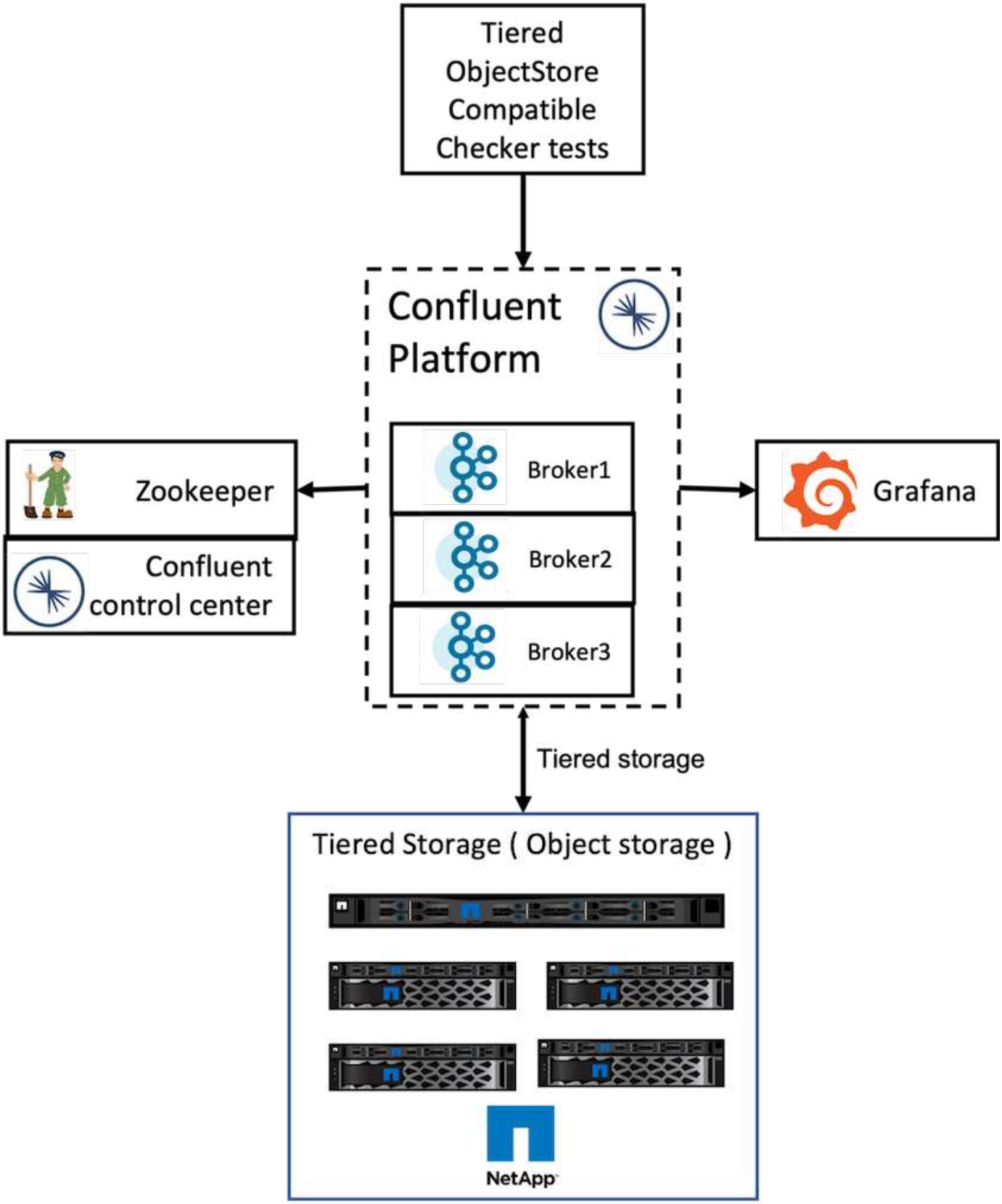
La configuración de almacenamiento por niveles requiere los siguientes parámetros en Kafka:

```
Confluent.tier.archiver.num.threads=16
confluent.tier.fetcher.num.threads=32
confluent.tier.enable=true
confluent.tier.feature=true
confluent.tier.backend=S3
confluent.tier.s3.bucket=kafkasgdbucket1-2
confluent.tier.s3.region=us-west-2
confluent.tier.s3.cred.file.path=/data/kafka/.ssh/credentials
confluent.tier.s3.aws.endpoint.override=http://kafkasgd.rtpppe.netapp.com:
10444/
confluent.tier.s3.force.path.style.access=true
```

Para la verificación, utilizamos StorageGRID con el protocolo HTTP, pero HTTPS también funciona. La clave de acceso y la clave secreta se almacenan en el nombre de archivo que se proporciona en el `confluent.tier.s3.cred.file.path` parámetro.

Almacenamiento de objetos de NetApp: StorageGRID

Hemos configurado la configuración de un único sitio en StorageGRID para la verificación.



Pruebas de verificación

Completamos los cinco casos de prueba siguientes para la verificación. Estas pruebas se ejecutan en el marco de Trogdor. Las dos primeras fueron pruebas de funcionalidad y las tres restantes fueron pruebas de rendimiento.

Prueba de corrección del almacén de objetos

Esta prueba determina si todas las operaciones básicas (por ejemplo, Get/put/delete) en la API de almacén de objetos funcionan bien de acuerdo con las necesidades de almacenamiento por niveles. Es una prueba básica que cada servicio de almacén de objetos debe esperar pasar por delante de las siguientes pruebas. Es una prueba asertiva que pasa o falla.

Prueba de corrección de la funcionalidad de organización en niveles

Esta prueba determina si la funcionalidad de almacenamiento por niveles completo funciona bien con una prueba asertiva que pasa o falla. La prueba crea un tema de prueba que, de forma predeterminada, se configura con la clasificación por niveles habilitada y un tamaño de conjunto de datos activo muy reducido. Produce una secuencia de eventos para el tema de prueba recién creado, espera a que los agentes archiven los segmentos en el almacén de objetos y luego consume la secuencia de eventos y valida que la secuencia consumida coincide con la secuencia producida. El número de mensajes producidos en el flujo de eventos es configurable, lo que permite al usuario generar una carga de trabajo lo suficientemente grande en función de las necesidades de las pruebas. El tamaño reducido del hotset garantiza que las búsquedas del consumidor fuera del segmento activo se sirvan sólo desde el almacén de objetos; esto ayuda a probar la corrección del almacén de objetos para las lecturas. Hemos realizado este test con y sin inyección de fallo del almacén de objetos. Simulamos el fallo del nodo deteniendo el servicio de gestor de servicios en uno de los nodos en StorageGRID y validando que la funcionalidad integral funciona con almacenamiento de objetos.

Punto de referencia de obtención de nivel

En esta prueba se validó el rendimiento de lectura del almacenamiento de objetos por niveles y se comprueban las solicitudes de lectura de recuperación de rango bajo carga pesada de los segmentos generados por la prueba de rendimiento. En esta prueba, Confluent desarrolló clientes personalizados para atender las solicitudes de obtención de nivel.

Producir-consumir prueba de rendimiento de cargas de trabajo

Esta prueba generó indirectamente la carga de trabajo de escritura en el almacén de objetos mediante el archivado de segmentos. La carga de trabajo de lectura (segmentos leídos) se generó desde el almacenamiento de objetos cuando los grupos de consumidores obtuvieron los segmentos. Esta carga de trabajo fue generada por el script de prueba. En esta prueba se verificó el rendimiento de lectura y escritura en el almacenamiento de objetos en subprocesos paralelos. Hemos probado con y sin inyección de fallo en el almacén de objetos como lo hicimos con la prueba de corrección de la funcionalidad de organización en niveles.

Prueba de rendimiento de cargas de trabajo de retención

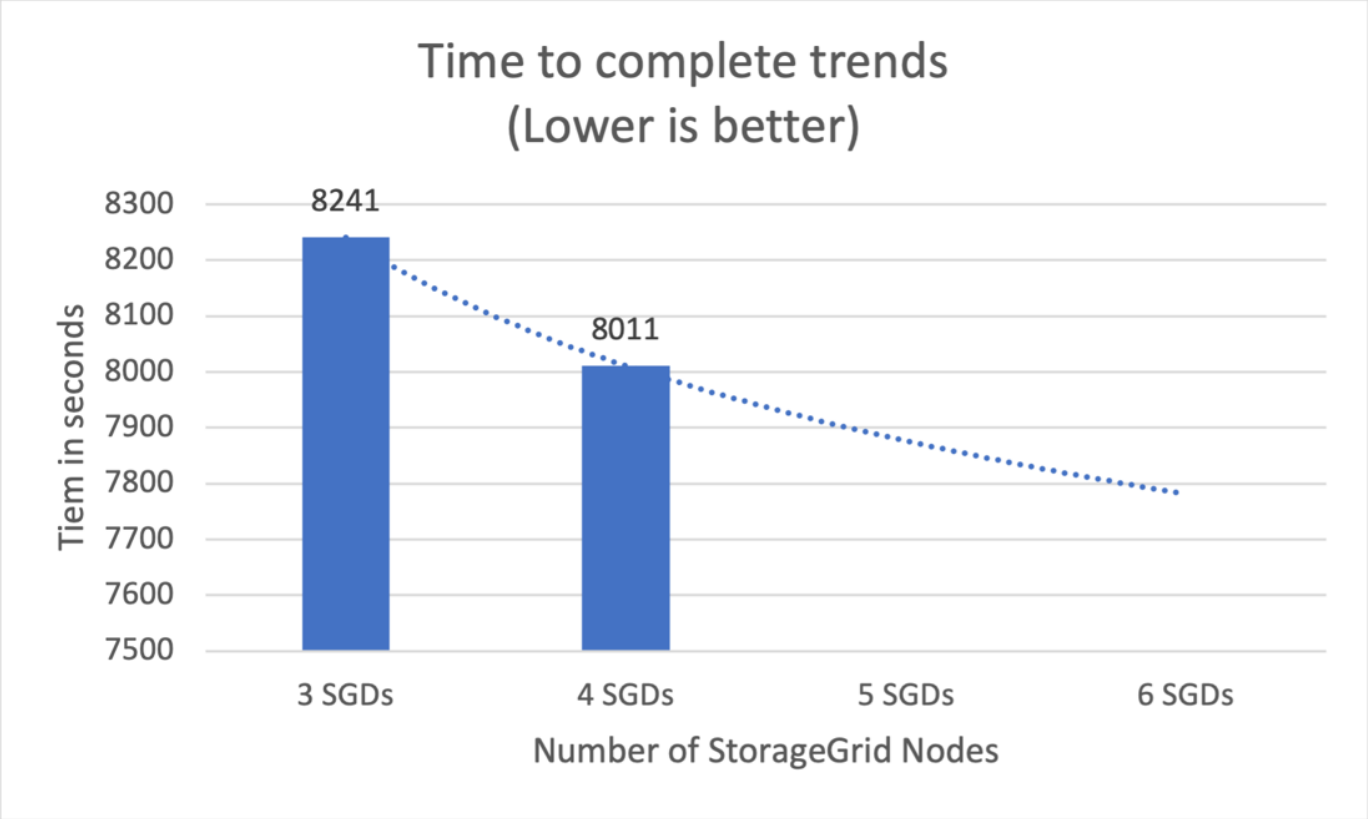
En esta prueba se comprobó el rendimiento de eliminación de un almacén de objetos en una gran carga de trabajo de retención de temas. La carga de trabajo de retención se generó mediante un script de prueba que produce muchos mensajes en paralelo a un tema de prueba. El tema de prueba se configuraba con una configuración de retención agresiva basada en tamaño y en tiempo que provocaba que la secuencia de eventos se purgaran continuamente del almacén de objetos. A continuación, se archivaron los segmentos. Esto llevó a un gran número de eliminaciones en el almacenamiento de objetos por parte del agente y la colección del rendimiento de las operaciones de eliminación del almacén de objetos.

Pruebas de rendimiento con escalabilidad

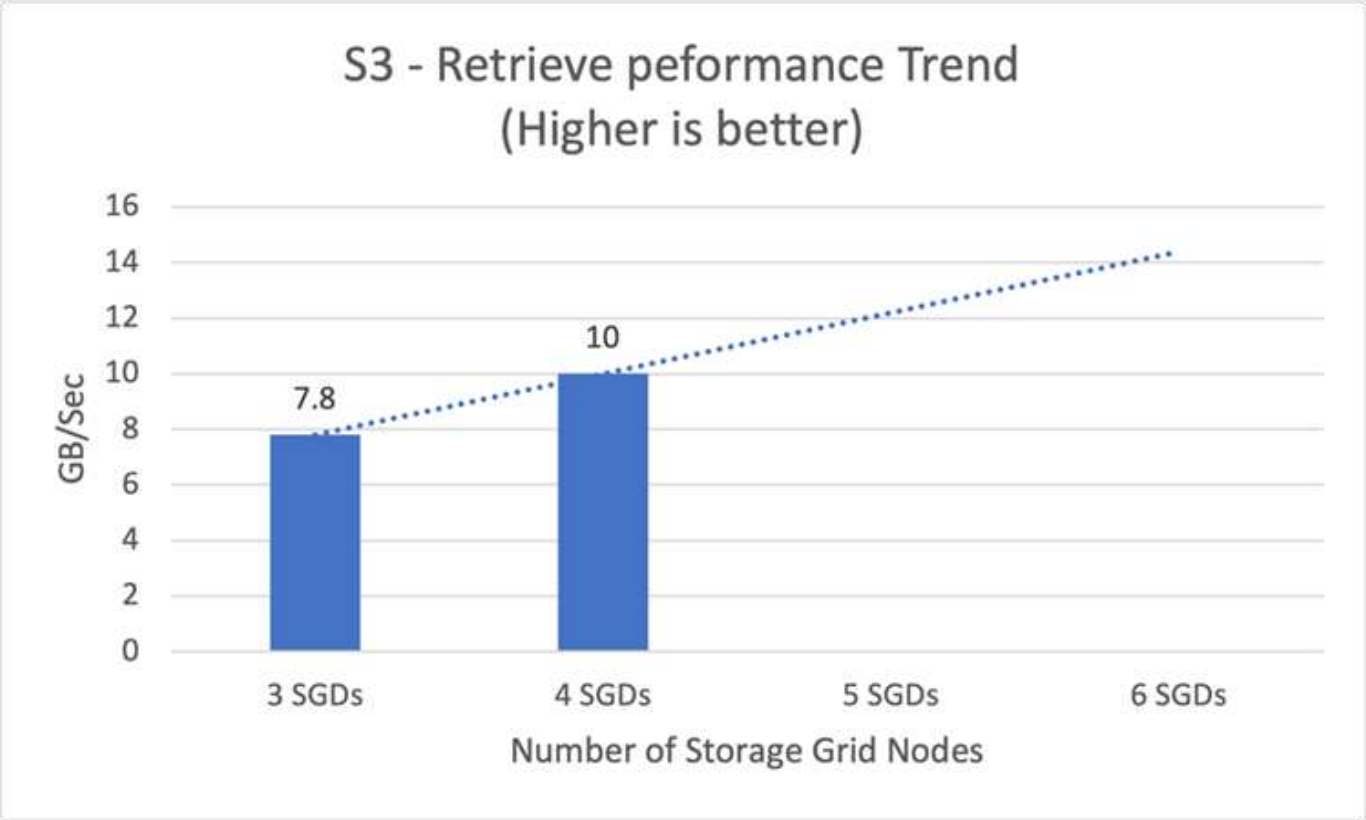
Realizamos las pruebas de almacenamiento por niveles con de tres a cuatro nodos para cargas de trabajo de producción y consumo con la configuración de StorageGRID de

NetApp. Según nuestras pruebas, el tiempo hasta el final y los resultados del rendimiento eran directamente proporcionales al número de nodos StorageGRID. La configuración de StorageGRID requería un mínimo de tres nodos.

- El tiempo para completar el funcionamiento del producto y del consumidor disminuyó linealmente cuando aumentó el número de nodos de almacenamiento.



- El rendimiento de la operación de recuperación de s3 aumentó de forma lineal en función del número de nodos StorageGRID. StorageGRID admite hasta 200 nodos StorageGRID.



Conector confluent s3

El conector Amazon S3 Sink exporta datos de temas de Apache Kafka a objetos de S3 en formato Avro, JSON o bytes. El conector hunk de Amazon S3 sondea periódicamente datos de Kafka y, a su vez, los carga en S3. Se utiliza un particionador para dividir los datos de cada partición de Kafka en fragmentos. Cada fragmento de datos se representa como un objeto S3. El nombre de clave codifica el tema, la partición Kafka y el desplazamiento inicial de este fragmento de datos.

En esta configuración, le mostramos cómo leer y escribir temas en almacenamiento de objetos de Kafka directamente con el conector Kafka s3 sink. Para esta prueba, utilizamos un clúster Confluent independiente, pero esta configuración se aplica a un clúster distribuido.

1. Descargar Confluent Kafka desde el sitio web de Confluent.
2. Desembale el paquete en una carpeta del servidor.
3. Exportar dos variables.

```
Export CONFLUENT_HOME=/data/confluent/confluent-6.2.0
export PATH=$PATH:/data/confluent/confluent-6.2.0/bin
```

4. Para una configuración independiente Confluent Kafka, el clúster crea una carpeta raíz temporal en `/tmp`. También crea Zookeeper, Kafka, un registro de esquemas, Connect, un ksql-Server, y las carpetas del centro de control y copia sus archivos de configuración respectivos desde `$CONFLUENT_HOME`. Consulte el siguiente ejemplo:

```
root@stlrx2540m1-108:~# ls -ltr /tmp/confluent.406980/
total 28
drwxr-xr-x 4 root root 4096 Oct 29 19:01 zookeeper
drwxr-xr-x 4 root root 4096 Oct 29 19:37 kafka
drwxr-xr-x 4 root root 4096 Oct 29 19:40 schema-registry
drwxr-xr-x 4 root root 4096 Oct 29 19:45 kafka-rest
drwxr-xr-x 4 root root 4096 Oct 29 19:47 connect
drwxr-xr-x 4 root root 4096 Oct 29 19:48 ksql-server
drwxr-xr-x 4 root root 4096 Oct 29 19:53 control-center
root@stlrx2540m1-108:~#
```

5. Configure Zookeeper. No es necesario cambiar nada si utiliza los parámetros predeterminados.

```
root@stlrx2540m1-108:~# cat
/tmp/confluent.406980/zookeeper/zookeeper.properties  | grep -iv ^#
dataDir=/tmp/confluent.406980/zookeeper/data
clientPort=2181
maxClientCnxns=0
admin.enableServer=false
tickTime=2000
initLimit=5
syncLimit=2
server.179=controlcenter:2888:3888
root@stlrx2540m1-108:~#
```

En la configuración anterior, actualizamos la `server. xxx` propiedad. De forma predeterminada, se necesitan tres Zookeepers para la selección de líder de Kafka.

6. Hemos creado un archivo myid en `/tmp/confluent.406980/zookeeper/data` Con un ID único:

```
root@stlrx2540m1-108:~# cat /tmp/confluent.406980/zookeeper/data/myid
179
root@stlrx2540m1-108:~#
```

Utilizamos el último número de direcciones IP para el archivo myid. Hemos utilizado los valores predeterminados para Kafka, Connect, control-centre, Kafka, Kafka-Rest, ksql-server y configuraciones de registro de esquema.

7. Inicie los servicios de Kafka.

```
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin# confluent
local services  start
The local commands are intended for a single-node development
environment only,
NOT for production usage.

Using CONFLUENT_CURRENT: /tmp/confluent.406980
ZooKeeper is [UP]
Kafka is [UP]
Schema Registry is [UP]
Kafka REST is [UP]
Connect is [UP]
ksqlDB Server is [UP]
Control Center is [UP]
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin#
```

Hay una carpeta de registro para cada configuración, lo que ayuda a solucionar problemas. En algunos casos, los servicios tardan más tiempo en iniciarse. Asegúrese de que todos los servicios estén activos y en funcionamiento.

8. Instale Kafka Connect con `confluent-hub`.

```
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin# ./confluent-
hub install confluentinc/kafka-connect-s3:latest
The component can be installed in any of the following Confluent
Platform installations:
  1. /data/confluent/confluent-6.2.0 (based on $CONFLUENT_HOME)
  2. /data/confluent/confluent-6.2.0 (where this tool is installed)
Choose one of these to continue the installation (1-2): 1
Do you want to install this into /data/confluent/confluent-
6.2.0/share/confluent-hub-components? (yN) y

Component's license:
Confluent Community License
http://www.confluent.io/confluent-community-license
I agree to the software license agreement (yN) y
Downloading component Kafka Connect S3 10.0.3, provided by Confluent,
Inc. from Confluent Hub and installing into /data/confluent/confluent-
6.2.0/share/confluent-hub-components
Do you want to uninstall existing version 10.0.3? (yN) y
Detected Worker's configs:
  1. Standard: /data/confluent/confluent-6.2.0/etc/kafka/connect-
distributed.properties
  2. Standard: /data/confluent/confluent-6.2.0/etc/kafka/connect-
standalone.properties
```

```
  3. Standard: /data/confluent/confluent-6.2.0/etc/schema-
registry/connect-avro-distributed.properties
  4. Standard: /data/confluent/confluent-6.2.0/etc/schema-
registry/connect-avro-standalone.properties
  5. Based on CONFLUENT_CURRENT:
/tmp/confluent.406980/connect/connect.properties
  6. Used by Connect process with PID 15904:
/tmp/confluent.406980/connect/connect.properties
Do you want to update all detected configs? (yN) y
Adding installation directory to plugin path in the following files:
  /data/confluent/confluent-6.2.0/etc/kafka/connect-
distributed.properties
  /data/confluent/confluent-6.2.0/etc/kafka/connect-
standalone.properties
  /data/confluent/confluent-6.2.0/etc/schema-registry/connect-avro-
distributed.properties
  /data/confluent/confluent-6.2.0/etc/schema-registry/connect-avro-
standalone.properties
  /tmp/confluent.406980/connect/connect.properties
  /tmp/confluent.406980/connect/connect.properties

Completed
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin#
```

También puede instalar una versión específica mediante `confluent-hub install confluentinc/kafka-connect-s3:10.0.3`.

9. De forma predeterminada, `confluentinc-kafka-connect-s3` está instalado en `/data/confluent/confluent-6.2.0/share/confluent-hub-components/confluentinc-kafka-connect-s3`.
10. Actualice la ruta de plugins con el nuevo `confluentinc-kafka-connect-s3`.

```
root@stlrx2540m1-108:~# cat /data/confluent/confluent-
6.2.0/etc/kafka/connect-distributed.properties | grep plugin.path
#
plugin.path=/usr/local/share/java,/usr/local/share/kafka/plugins,/opt/co
nnectors,
plugin.path=/usr/share/java,/data/zookeeper/confluent/confluent-
6.2.0/share/confluent-hub-components,/data/confluent/confluent-
6.2.0/share/confluent-hub-components,/data/confluent/confluent-
6.2.0/share/confluent-hub-components/confluentinc-kafka-connect-s3
root@stlrx2540m1-108:~#
```

11. Detenga los servicios Confluent y reinícielos.

```
confluent local services  stop
confluent local services  start
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin# confluent
local services  status
The local commands are intended for a single-node development
environment only,
NOT for production usage.

Using CONFLUENT_CURRENT: /tmp/confluent.406980
Connect is [UP]
Control Center is [UP]
Kafka is [UP]
Kafka REST is [UP]
ksqlDB Server is [UP]
Schema Registry is [UP]
ZooKeeper is [UP]
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin#
```

12. Configure el ID de acceso y la clave secreta en la `/root/.aws/credentials` archivo.

```
root@stlrx2540m1-108:~# cat /root/.aws/credentials
[default]
aws_access_key_id = xxxxxxxxxxxx
aws_secret_access_key = xxxxxxxxxxxxxxxxxxxxxxxxx
root@stlrx2540m1-108:~#
```

13. Verifique que se pueda acceder al cucharón.

```
root@stlrx2540m4-01:~# aws s3 –endpoint-url
http://kafkasgd.rtpppe.netapp.com:10444 ls kafkasgdbucket1-2
2021-10-29 21:04:18       1388 1
2021-10-29 21:04:20       1388 2
2021-10-29 21:04:22       1388 3
root@stlrx2540m4-01:~#
```

14. Configure el archivo de propiedades s3-hunden para la configuración de bloques y s3.

```
root@stlrx2540m1-108:~# cat /data/confluent/confluent-
6.2.0/share/confluent-hub-components/confluentinc-kafka-connect-
s3/etc/quickstart-s3.properties | grep -v ^#
name=s3-sink
connector.class=io.confluent.connect.s3.S3SinkConnector
tasks.max=1
topics=s3_testtopic
s3.region=us-west-2
s3.bucket.name=kafkasgdbucket1-2
store.url=http://kafkasgd.rtpppe.netapp.com:10444/
s3.part.size=5242880
flush.size=3
storage.class=io.confluent.connect.s3.storage.S3Storage
format.class=io.confluent.connect.s3.format.avro.AvroFormat
partitioner.class=io.confluent.connect.storage.partitioner.DefaultPartit
ioner
schema.compatibility=NONE
root@stlrx2540m1-108:~#
```

15. Importe unos pocos registros en el bloque de s3.

```
kafka-avro-console-producer --broker-list localhost:9092 --topic
s3_topic \
--property
value.schema='{"type":"record","name":"myrecord","fields":[{"name":"f1",
"type":"string"}]}'
{"f1": "value1"}
{"f1": "value2"}
{"f1": "value3"}
{"f1": "value4"}
{"f1": "value5"}
{"f1": "value6"}
{"f1": "value7"}
{"f1": "value8"}
{"f1": "value9"}
```

16. Cargue el conector del receptor s3.

```
root@stlrx2540m1-108:~# confluent local services connect connector load
s3-sink  --config /data/confluent/confluent-6.2.0/share/confluent-hub-
components/confluentinc-kafka-connect-s3/etc/quickstart-s3.properties
The local commands are intended for a single-node development
environment only,
NOT for production usage.
https://docs.confluent.io/current/cli/index.html
{
  "name": "s3-sink",
  "config": {
    "connector.class": "io.confluent.connect.s3.S3SinkConnector",
    "flush.size": "3",
    "format.class": "io.confluent.connect.s3.format.avro.AvroFormat",
    "partitioner.class":
"io.confluent.connect.storage.partitioner.DefaultPartitioner",
    "s3.bucket.name": "kafkasgdbucket1-2",
    "s3.part.size": "5242880",
    "s3.region": "us-west-2",
    "schema.compatibility": "NONE",
    "storage.class": "io.confluent.connect.s3.storage.S3Storage",
    "store.url": "http://kafkasgd.rtpppe.netapp.com:10444/",
    "tasks.max": "1",
    "topics": "s3_testtopic",
    "name": "s3-sink"
  },
  "tasks": [],
  "type": "sink"
}
root@stlrx2540m1-108:~#
```

17. Compruebe el estado del receptor s3.

```
root@stlrx2540m1-108:~# confluent local services connect connector
status s3-sink
The local commands are intended for a single-node development
environment only,
NOT for production usage.
https://docs.confluent.io/current/cli/index.html
{
  "name": "s3-sink",
  "connector": {
    "state": "RUNNING",
    "worker_id": "10.63.150.185:8083"
  },
  "tasks": [
    {
      "id": 0,
      "state": "RUNNING",
      "worker_id": "10.63.150.185:8083"
    }
  ],
  "type": "sink"
}
root@stlrx2540m1-108:~#
```

18. Revise el registro para asegurarse de que el receptor s3 esté listo para aceptar temas.

```
root@stlrx2540m1-108:~# confluent local services connect log
```

19. Consulta los temas en Kafka.

```
kafka-topics --list --bootstrap-server localhost:9092
…
connect-configs
connect-offsets
connect-statuses
default_ksql_processing_log
s3_testtopic
s3_topic
s3_topic_new
root@stlrx2540m1-108:~#
```

20. Compruebe los objetos en el bloque de s3.

```
root@stlrx2540m1-108:~# aws s3 --endpoint-url
http://kafkasgd.rtpppe.netapp.com:10444 ls --recursive kafkasgdbucket1-
2/topics/
2021-10-29 21:24:00        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000000.avro
2021-10-29 21:24:00        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000003.avro
2021-10-29 21:24:00        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000006.avro
2021-10-29 21:24:08        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000009.avro
2021-10-29 21:24:08        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000012.avro
2021-10-29 21:24:09        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000015.avro
root@stlrx2540m1-108:~#
```

21. Para verificar el contenido, copie cada archivo desde S3 en el sistema de archivos local ejecutando el siguiente comando:

```
root@stlrx2540m1-108:~# aws s3 --endpoint-url
http://kafkasgd.rtpppe.netapp.com:10444 cp s3://kafkasgdbucket1-
2/topics/s3_testtopic/partition=0/s3_testtopic+0+0000000000.avro
tes.avro
download: s3://kafkasgdbucket1-
2/topics/s3_testtopic/partition=0/s3_testtopic+0+0000000000.avro to
./tes.avro
root@stlrx2540m1-108:~#
```

22. Para imprimir los registros, utilice avro-tools-1.11.0.1.jar (disponible en "Archivos Apache").

```
root@stlrx2540m1-108:~# java -jar /usr/src/avro-tools-1.11.0.1.jar
tojson tes.avro
21/10/30 00:20:24 WARN util.NativeCodeLoader: Unable to load native-
hadoop library for your platform... using builtin-java classes where
applicable
{"f1":"value1"}
{"f1":"value2"}
{"f1":"value3"}
root@stlrx2540m1-108:~#
```

Clústeres de equilibrio automático confluente

Si ha gestionado antes un clúster Kafka, probablemente esté familiarizado con los retos que surgen con la reasignación manual de particiones a distintos agentes para garantizar que la carga de trabajo se equilibre a través del clúster. Para las organizaciones con grandes puestas en marcha de Kafka, cambiar grandes cantidades de datos puede ser abrumador, tedioso y arriesgado, especialmente si las aplicaciones vitales para el negocio se basan en el clúster. Sin embargo, incluso para los casos de uso más pequeños de Kafka, el proceso consume mucho tiempo y es propenso a errores humanos.

En nuestro laboratorio, probamos la función de clústeres de equilibrio automático con fluidez, que automatiza el reequilibrado en función de cambios de topología de clúster o carga irregular. La prueba de reequilibrado fluido ayuda a medir el tiempo para agregar un nuevo intermediario cuando el fallo del nodo o el nodo de escalado requiere reequilibrar los datos entre los distintos intermediarios. En las configuraciones clásicas de Kafka, la cantidad de datos que se deben reequilibrar crece a medida que crece el clúster, pero, en almacenamiento por niveles, el reequilibrio se limita a una pequeña cantidad de datos. Según nuestra validación, el reequilibrio en el almacenamiento por niveles se demora segundos o minutos en una arquitectura Kafka clásica y crece de forma lineal a medida que crece el clúster.

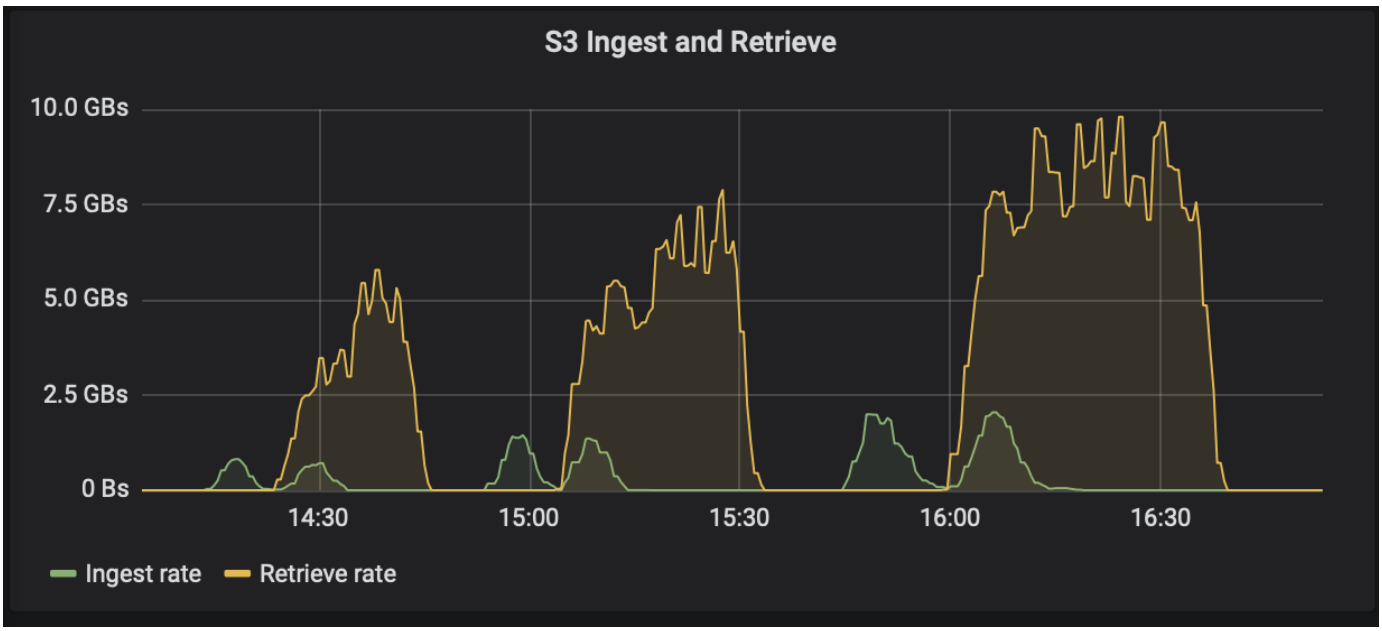
En los clústeres de equilibrio automático, los reequilibrios de las particiones están totalmente automatizados para optimizar el rendimiento de Kafka, acelerar el escalado de los intermediarios y reducir la carga operativa de ejecutar un gran clúster. En clústeres de estado constante, los clústeres de equilibrio automático supervisan el desfase de datos a través de los intermediarios y reasignan continuamente particiones para optimizar el rendimiento del clúster. Al escalar la plataforma de forma horizontal o vertical, los clústeres de equilibrio automático reconocen automáticamente la presencia de nuevos agentes o la eliminación de agentes antiguos y activan la reasignación posterior de particiones. Esto le permite añadir y retirar agentes fácilmente, lo que hace que sus clústeres Kafka sean mucho más elásticos. Estas ventajas se obtienen sin necesidad de intervención manual, matemáticas complejas o el riesgo de error humano que suelen implicar la reasignación de particiones. Como resultado, los reequilibrios de los datos se completan en mucho menos tiempo y puede centrarse en proyectos de streaming de eventos de mayor valor, en lugar de tener que supervisar constantemente los clústeres.

Directrices de prácticas recomendadas

En esta sección se presentan las lecciones aprendidas en esta certificación.

- Basado en nuestra validación, el almacenamiento de objetos S3 es la mejor opción para Confluent mantener los datos.
- Podemos usar SAN de alto rendimiento (específicamente FC) para mantener el intermediario de datos activos o el disco local, ya que, en la configuración de almacenamiento por niveles fluido, el tamaño de los datos contenidos en el directorio de datos del brokers se basa en el tamaño del segmento y el tiempo de retención en el que se mueven los datos al almacenamiento de objetos.
- Los almacenes de objetos proporcionan un mejor rendimiento cuando el segmento.bytes es mayor; probamos 512 MB.
- En Kafka, la longitud de la clave o el valor (en bytes) para cada registro producido en el tema se controla mediante el `length.key.value` parámetro. Para StorageGRID, la ingesta de objetos S3 y la recuperación de rendimiento aumentaron a valores más altos. Por ejemplo, 512 bytes proporcionaron una recuperación de 5,8 Gbps, 1024 bytes proporcionaban una recuperación de 7,5 Gbps s3 y 2048 bytes proporcionados cerca de 10 Gbps.

En la figura siguiente, se presenta la ingesta y la recuperación de objetos de S3 según `length.key.value`.



- **Ajuste Kafka.** para mejorar el rendimiento del almacenamiento por niveles, puede aumentar TierFetcherNumThreads y TierArchiverNumThreads. Como guía general, desea aumentar TierFetcherNumThreads para que coincida con el número de núcleos de CPU físicos y aumentar TierArchiverNumThreads a la mitad del número de núcleos de CPU. Por ejemplo, en las propiedades del servidor, si tiene una máquina con ocho núcleos físicos, establezca confluent.tier.fetcher.num.threads = 8 y confluent.tier.archivver.num.threads = 4.
- **Intervalo de tiempo para las eliminaciones de temas.** cuando se elimina un tema, la eliminación de los archivos del segmento de registro en el almacenamiento de objetos no comienza inmediatamente. En su lugar, hay un intervalo de tiempo con un valor predeterminado de 3 horas antes de que se elimine dichos archivos. Puede modificar la configuración, confluent.tier.topic.delete.check.interval.ms, para cambiar el valor de este intervalo. Si elimina un tema o clúster, también puede eliminar manualmente los objetos en el bloque correspondiente.
- **ACL en temas internos de almacenamiento por niveles.** una práctica recomendada para implementaciones en las instalaciones es habilitar un autorizador de ACL en los temas internos utilizados para el almacenamiento por niveles. Defina las reglas de ACL para limitar el acceso a estos datos sólo al usuario de broker. De este modo, se protegen los temas internos y se evita el acceso no autorizado a metadatos y datos de almacenamiento por niveles.

```
kafka-acls --bootstrap-server localhost:9092 --command-config adminclient-
configs.conf \
--add --allow-principal User:<kafka> --operation All --topic "_confluent-
tier-state"
```

Sustituya al usuario `<kafka>` con el agente principal en la puesta en marcha.

Por ejemplo, el comando `confluent-tier-state` Define las ACL en el tema interno para el almacenamiento por niveles. Actualmente, solo existe un solo tema interno relacionado con el almacenamiento por niveles. El ejemplo crea una ACL que proporciona el permiso Kafka principal para todas las operaciones del tema interno.

Ajuste de tamaño

El ajuste de tamaño de Kafka se puede realizar con cuatro modos de configuración: Sencillo, granular, inverso y particiones.

Sencillo

El modo simple es apropiado por primera vez para los usuarios de Apache Kafka o los casos de uso de estado anteriores. Para este modo, se proporcionan requisitos como el rendimiento en Mbps, el ventilador de lectura, la retención y el porcentaje de utilización de recursos (el 60% es el predeterminado). También puede entrar al entorno, como en las instalaciones (con configuración básica, VMware, Kubernetes u OpenStack) o en el cloud. Basándose en esta información, el tamaño de un clúster Kafka proporciona el número de servidores necesarios para el corredor, el zookeeper, los trabajadores de Apache Kafka connect, el registro de esquemas, un proxy REST, ksqlDB y el centro de control de Confluent.

En el caso del almacenamiento por niveles, tenga en cuenta el modo de configuración granular para ajustar el tamaño de un clúster Kafka. El modo granular es adecuado para usuarios con experiencia de Apache Kafka o casos de uso bien definidos. En esta sección se describe el dimensionamiento para los productores, los procesadores de flujo y los consumidores.

Productores

Para describir a los productores de Apache Kafka (por ejemplo, un cliente nativo, proxy REST o conector Kafka), proporcione la siguiente información:

- **Nombre.** Spark.
- **Tipo de productor.** aplicación o servicio, proxy (REST, MQTT, otros) y base de datos existente (RDBMS, NOSQL, otros). También puede seleccionar “no lo sé”.
- **Rendimiento medio.** en eventos por segundo (1,000,000 por ejemplo).
- **Rendimiento máximo.** en eventos por segundo (4,000,000 por ejemplo).
- **Tamaño medio del mensaje.** en bytes, sin comprimir (máx. 1 MB; 1000 por ejemplo).
- **Formato de mensaje.** las opciones incluyen Avro, JSON, búferes de protocolo, binario, texto, “No lo sé” y otros.
- * Factor de replicación.* las opciones son 1, 2, 3 (recomendación de fluidez), 4, 5, o 6.
- **Tiempo de retención.** un día (por ejemplo). ¿Cuánto tiempo quiere que sus datos se almacenen en Apache Kafka? Introduzca -1 con cualquier unidad durante un tiempo infinito. La calculadora asume un tiempo de retención de 10 años para retención infinita.
- Active la casilla de comprobación para "Habilitar el almacenamiento por niveles para reducir el número de agentes y permitir el almacenamiento por Infinite Volume?"
- Cuando se habilita el almacenamiento por niveles, los campos de retención controlan el conjunto de datos activos que se almacenan de forma local en el agente. Los campos de retención de archivado controlan el tiempo que se almacenan los datos en el almacenamiento de objetos de archivado.
- **Retención de almacenamiento de ficheros.** un año (por ejemplo). ¿Cuánto tiempo desea que los datos se almacenen en el almacenamiento de archivado? Introduzca -1 con cualquier unidad para una duración infinita. La calculadora asume una retención de 10 años para retención infinita.
- **Multiplicador de crecimiento.** 1 (por ejemplo). Si el valor de este parámetro se basa en el rendimiento actual, configúrelo en 1. Para ajustar el tamaño en función del crecimiento adicional, establezca este parámetro en un multiplicador de crecimiento.

- **Número de instancias de productores.** 10 (por ejemplo). ¿Cuántas instancias de productor se estarán ejecutando? Esta entrada es necesaria para incorporar la carga de la CPU en el cálculo de tamaño. Un valor en blanco indica que la carga de la CPU no está integrada en el cálculo.

Basándose en este ejemplo, el dimensionamiento tiene el siguiente efecto sobre los productores:

- Promedio de rendimiento en bytes sin comprimir: 1 Gbps. Rendimiento máximo en bytes sin comprimir: 4Gbps. Rendimiento promedio en bytes comprimidos: 400 Mbps. Máximo rendimiento en bytes comprimidos: 1,6 Gbps. Esto se basa en una tasa de compresión predeterminada del 60 % (puede cambiar este valor).
  - Total de almacenamiento con hotset de On-broker requerido: 31,104 TB, incluyendo replicación y compresión. Almacenamiento total de archivado fuera de agentes necesario: 378,432 TB, comprimidos. Uso "https://fusion.netapp.com" Para ajustar el tamaño de StorageGRID.

Stream Processor debe describir sus aplicaciones o servicios que consumen datos de Apache Kafka y los vuelven a producir en Apache Kafka. En la mayoría de los casos se construyen en arroyos ksqlDB o Kafka.

- **Nombre.** Spark streamer.
- **Tiempo de procesamiento.** ¿Cuánto tarda este procesador en procesar un solo mensaje?
  - 1 ms (simple, transformación sin estado) [ejemplo], 10 ms (funcionamiento con estado en memoria).
  - 100 ms (funcionamiento de disco o red con estado), 1000 ms (llamada REST de terceros).
  - He evaluado este parámetro y sé exactamente cuánto tiempo lleva.
- **Retención de salida.** 1 día (ejemplo). Un procesador de secuencias devuelve su salida a Apache Kafka. ¿Cuánto tiempo desea que estos datos de salida se almacenen en Apache Kafka? Introduzca -1 con cualquier unidad para una duración infinita.
- Marque la casilla de comprobación "Habilitar el almacenamiento por niveles para reducir el número de agentes y permitir Infinite Storage?"
- **Retención de almacenamiento de ficheros.** 1 año (por ejemplo). ¿Cuánto tiempo desea que los datos se almacenen en el almacenamiento de archivado? Introduzca -1 con cualquier unidad para una duración infinita. La calculadora asume una retención de 10 años para retención infinita.
- **Porcentaje de paso a través de salida.** 100 (por ejemplo). Un procesador de secuencias devuelve su salida a Apache Kafka. ¿Qué porcentaje de rendimiento de entrada se propondrá en Apache Kafka? Por ejemplo, si el rendimiento de entrada es de 20 Mbps y este valor es de 10, el rendimiento de salida será de 2 Mbps.
- ¿De qué aplicaciones se lee? Seleccione “Spark” (Spark), el nombre utilizado en el ajuste de tamaño basado en el tipo de productor. En función de la entrada anterior, puede esperar los siguientes efectos del ajuste de tamaño en las instancias del procesador de secuencias y las estimaciones de las particiones de temas:
- Esta aplicación de procesador de secuencias requiere el siguiente número de instancias. Es probable que los temas entrantes también requieran muchas particiones. Contacte con Confluent para confirmar este parámetro.
  - 1,000 para un rendimiento medio sin multiplicador de crecimiento
  - 4,000 para un rendimiento máximo sin multiplicador de crecimiento
  - 1,000 para un rendimiento medio con un multiplicador de crecimiento
  - 4,000 para obtener el máximo rendimiento con un multiplicador de crecimiento

Consumidores

Describa sus aplicaciones o servicios que consumen datos de Apache Kafka y no vuelven a producir en Apache Kafka; por ejemplo, un cliente nativo o un conector Kafka.

- **Nombre.** Cliente de Spark.
- **Tiempo de procesamiento.** ¿Cuánto tarda este consumidor en procesar un solo mensaje?
  - 1 ms (por ejemplo, una tarea simple y sin estado, como el registro)
  - 10 ms (escrituras rápidas en un almacén de datos)
  - 100 ms (escrituras lentas en un almacén de datos)
  - 1000 ms (llamada DE REST de terceros)
  - Algún otro proceso de referencia de duración conocida.
- **Tipo de consumidor.** aplicación, proxy o receptor a un almacén de datos existente (RDBMS, NoSQL, otros).
- ¿De qué aplicaciones se lee? Conecte este parámetro con el tamaño del productor y del flujo determinado anteriormente.

Basándose en la información anterior, debe determinar la configuración de las instancias de cliente y las estimaciones de las particiones del tema. Una aplicación de cliente requiere el siguiente número de instancias.

- 2,000 para un rendimiento medio, sin multiplicador de crecimiento
- 8,000 para un rendimiento máximo, sin multiplicador de crecimiento
- 2,000 para un rendimiento medio, incluido el multiplicador de crecimiento
- 8,000 para un rendimiento máximo, incluido el multiplicador de crecimiento

Es probable que los temas entrantes también necesiten este número de particiones. Póngase en contacto con Confluent para confirmar.

Además de los requisitos para los productores, los procesadores de flujo y los consumidores, debe proporcionar los siguientes requisitos adicionales:

- **Tiempo de regeneración.** por ejemplo, 4 horas. Si un host de Apache Kafka Broker falla, sus datos se pierden y se aprovisiona un nuevo host para sustituir el host fallido, ¿con qué rapidez debe reconstruir este nuevo host? Deje este parámetro en blanco si el valor es desconocido.
- **Objetivo de utilización de recursos (porcentaje).** por ejemplo, 60. ¿Cómo se hace uso de los hosts durante el rendimiento medio? Confluent recomienda un aprovechamiento del 60% a menos que se utilicen clústeres de equilibrio automático Confluent, en cuyo caso el uso puede ser mayor.

Describa su entorno

- **¿En qué entorno se estará ejecutando su clúster?** Amazon Web Services, Microsoft Azure, plataforma en nube de Google, configuración básica en las instalaciones, VMware en las instalaciones, ¿OpenStack en las instalaciones o Kubas en el entorno local?
- **Detalles del host.** número de núcleos: 48 (por ejemplo), tipo de tarjeta de red (10 GbE, 40 GbE, 16 GbE, 1 GbE u otro tipo).
- **Volúmenes de almacenamiento.** Host: 12 (por ejemplo). ¿Cuántos discos duros o SSD son compatibles por host? Confluent recomienda 12 unidades de disco duro por host.
- **Capacidad de almacenamiento/volumen (en GB).** 1000 (por ejemplo). ¿Cuánto almacenamiento puede

almacenar un volumen único en gigabytes? Confluent recomienda discos de 1 TB.

- **Configuración de almacenamiento.** ¿Cómo se configuran los volúmenes de almacenamiento? Confluent recomienda RAID10 para aprovechar todas las características de Confluent. JBOD, SAN, RAID 1, RAID 0, RAID 5, y también se admiten otros tipos.
- **Rendimiento de volumen único (Mbps).** 125 (por ejemplo). ¿Con qué rapidez puede leer o escribir un único volumen de almacenamiento en megabytes por segundo? Confluent recomienda unidades de disco duro estándar, que normalmente tienen un rendimiento de 125 MBps.
- **Capacidad de memoria (GB).** 64 (por ejemplo).

Una vez que haya determinado sus variables de entorno, seleccione Size my Cluster. Basándonos en los parámetros de ejemplo indicados anteriormente, hemos determinado el tamaño siguiente para Confluent Kafka:

- * Apache Kafka.* número de broker: 22. El clúster está vinculado al almacenamiento. Considere la posibilidad de habilitar el almacenamiento por niveles para reducir el número de hosts y permitir el almacenamiento infinito.
- * Apache ZooKeeper.* Conde: 5; Apache Kafka Connect Workers: Count: 2; Registro de esquema: Cuenta: 2; Proxy REST: Cuenta: 2; ksqlDB: Cuenta: 2; Centro de Control de Confluente: Cuenta: 1.

Utilice el modo inverso para los equipos de plataformas sin tener en cuenta un caso de uso. Utilice el modo Partitions para calcular cuántas particiones necesita un solo tema. Consulte https://eventsizer.io para ajustar el tamaño en función de los modos inverso y de particiones.

Conclusión

Este documento proporciona directrices de prácticas recomendadas para el uso del almacenamiento por niveles con fluidez de NetApp, como pruebas de verificación, resultados de rendimiento del almacenamiento por niveles, ajuste, conectores Confluent S3 y la función de equilibrio automático. Teniendo en cuenta las políticas de ILM, el rendimiento fluido con varias pruebas de rendimiento para la verificación y las API S3 estándares del sector, el almacenamiento de objetos StorageGRID de NetApp es una opción óptima para el almacenamiento por niveles fluido.

Dónde encontrar información adicional

Si quiere más información sobre el contenido de este documento, consulte los siguientes documentos o sitios web:

- Qué es Apache Kafka

  "https://www.confluent.io/what-is-apache-kafka/"

- Documentación de productos de NetApp

  "https://www.netapp.com/support-and-training/documentation/"

- Detalles de los parámetros S3-hunk

  "https://docs.confluent.io/kafka-connect-s3-sink/current/configuration_options.html#s3-configuration-options"

- Apache Kafka

  "https://en.wikipedia.org/wiki/Apache_Kafka"

- Almacenamiento infinito en plataforma confluente

  "https://www.confluent.io/blog/infinite-kafka-storage-in-confluent-platform/"

- Confluent Tiered Storage: Mejores prácticas y ajuste de tamaño

  "https://docs.confluent.io/platform/current/kafka/tiered-storage.html#best-practices-and-recommendations"

- Conector de receptor Amazon S3 para la plataforma Confluent

  "https://docs.confluent.io/kafka-connect-s3-sink/current/overview.html"

- Tamaño de Kafka

  "https://eventsizer.io"

- Ajuste de tamaño de StorageGRID

  "https://fusion.netapp.com/"

- Casos de uso de Kafka

  "https://kafka.apache.org/uses"

- Clústeres de Kafka con equilibrio automático en una plataforma con fluidez 6.0

  "https://www.confluent.io/blog/self-balancing-kafka-clusters-in-confluent-platform-6-0/"

  "https://www.confluent.io/blog/confluent-platform-6-0-delivers-the-most-powerful-event-streaming-platform-to-date/"

Información de copyright

Copyright © 2024 NetApp, Inc. Todos los derechos reservados. Imprimido en EE. UU. No se puede reproducir este documento protegido por copyright ni parte del mismo de ninguna forma ni por ningún medio (gráfico, electrónico o mecánico, incluidas fotocopias, grabaciones o almacenamiento en un sistema de recuperación electrónico) sin la autorización previa y por escrito del propietario del copyright.

El software derivado del material de NetApp con copyright está sujeto a la siguiente licencia y exención de responsabilidad:

ESTE SOFTWARE LO PROPORCIONA NETAPP «TAL CUAL» Y SIN NINGUNA GARANTÍA EXPRESA O IMPLÍCITA, INCLUYENDO, SIN LIMITAR, LAS GARANTÍAS IMPLÍCITAS DE COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN CONCRETO, CUYA RESPONSABILIDAD QUEDA EXIMIDA POR EL PRESENTE DOCUMENTO. EN NINGÚN CASO NETAPP SERÁ RESPONSABLE DE NINGÚN DAÑO DIRECTO, INDIRECTO, ESPECIAL, EJEMPLAR O RESULTANTE (INCLUYENDO, ENTRE OTROS, LA OBTENCIÓN DE BIENES O SERVICIOS SUSTITUTIVOS, PÉRDIDA DE USO, DE DATOS O DE BENEFICIOS, O INTERRUPCIÓN DE LA ACTIVIDAD EMPRESARIAL) CUALQUIERA SEA EL MODO EN EL QUE SE PRODUJERON Y LA TEORÍA DE RESPONSABILIDAD QUE SE APLIQUE, YA SEA EN CONTRATO, RESPONSABILIDAD OBJETIVA O AGRAVIO (INCLUIDA LA NEGLIGENCIA U OTRO TIPO), QUE SURJAN DE ALGÚN MODO DEL USO DE ESTE SOFTWARE, INCLUSO SI HUBIEREN SIDO ADVERTIDOS DE LA POSIBILIDAD DE TALES DAÑOS.

NetApp se reserva el derecho de modificar cualquiera de los productos aquí descritos en cualquier momento y sin aviso previo. NetApp no asume ningún tipo de responsabilidad que surja del uso de los productos aquí descritos, excepto aquello expresamente acordado por escrito por parte de NetApp. El uso o adquisición de este producto no lleva implícita ninguna licencia con derechos de patente, de marcas comerciales o cualquier otro derecho de propiedad intelectual de NetApp.

Es posible que el producto que se describe en este manual esté protegido por una o más patentes de EE. UU., patentes extranjeras o solicitudes pendientes.

LEYENDA DE DERECHOS LIMITADOS: el uso, la copia o la divulgación por parte del gobierno están sujetos a las restricciones establecidas en el subpárrafo (b)(3) de los derechos de datos técnicos y productos no comerciales de DFARS 252.227-7013 (FEB de 2014) y FAR 52.227-19 (DIC de 2007).

Los datos aquí contenidos pertenecen a un producto comercial o servicio comercial (como se define en FAR 2.101) y son propiedad de NetApp, Inc. Todos los datos técnicos y el software informático de NetApp que se proporcionan en este Acuerdo tienen una naturaleza comercial y se han desarrollado exclusivamente con fondos privados. El Gobierno de EE. UU. tiene una licencia limitada, irrevocable, no exclusiva, no transferible, no sublicenciable y de alcance mundial para utilizar los Datos en relación con el contrato del Gobierno de los Estados Unidos bajo el cual se proporcionaron los Datos. Excepto que aquí se disponga lo contrario, los Datos no se pueden utilizar, desvelar, reproducir, modificar, interpretar o mostrar sin la previa aprobación por escrito de NetApp, Inc. Los derechos de licencia del Gobierno de los Estados Unidos de América y su Departamento de Defensa se limitan a los derechos identificados en la cláusula 252.227-7015(b) de la sección DFARS (FEB de 2014).

Información de la marca comercial

NETAPP, el logotipo de NETAPP y las marcas que constan en http://www.netapp.com/TM son marcas comerciales de NetApp, Inc. El resto de nombres de empresa y de producto pueden ser marcas comerciales de sus respectivos propietarios.