Mejores prácticas para Confluent Kafka

NetApp artificial intelligence solutions

NetApp
February 12, 2026

Tabla de contenidos

Mejores prácticas para Confluent Kafka

TR-4912: Pautas recomendadas para el almacenamiento en niveles de Confluent Kafka con NetApp

Karthikeyan Nagalingam, Joseph Kandatilparambil, NetApp Rankesh Kumar, Confluent

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 mensajes, Kafka se basa en una abstracción de un registro de confirmación distribuido. Desde que LinkedIn lo creó y lo puso en código abierto en 2011, Kafka ha evolucionado desde una cola de mensajes a una plataforma completa de transmisión de eventos. Confluent ofrece la distribución de Apache Kafka con la plataforma Confluent. La plataforma Confluent complementa Kafka con funciones comerciales y comunitarias adicionales diseñadas para mejorar la experiencia de transmisión tanto de operadores como de desarrolladores en producción a gran escala.

Este documento describe las pautas recomendadas para usar Confluent Tiered Storage en una oferta de almacenamiento de objetos de NetApp proporcionando el siguiente contenido:

- Verificación confluente con almacenamiento de objetos de NetApp – NetApp StorageGRID
- Pruebas de rendimiento de almacenamiento por niveles
- Pautas de mejores prácticas para Confluent en sistemas de almacenamiento NetApp

¿Por qué el almacenamiento en niveles de Confluent?

Confluent se ha convertido en la plataforma de transmisión en tiempo real predeterminada para muchas aplicaciones, especialmente para big data, análisis y cargas de trabajo de transmisión. El almacenamiento en 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 prácticamente infinitas de datos y escalar cargas de trabajo hacia arriba (o hacia abajo) según demanda, y facilita las tareas administrativas como el reequilibrio de datos e inquilinos. Los sistemas de almacenamiento compatibles con S3 pueden aprovechar todas estas capacidades para democratizar los datos con todos los eventos en un solo lugar, eliminando la necesidad de una ingeniería de datos compleja. Para obtener más información sobre por qué debería utilizar almacenamiento por niveles para Kafka, consulte"Este artículo de Confluent" .

NetApp instaclustr también es compatible con Kafka con almacenamiento en niveles desde 3.8.1. Por favor, consulte más detalles aquí "Instaclust con almacenamiento por niveles de Kafka"

¿Por qué NetApp StorageGRID para el almacenamiento en niveles?

StorageGRID es una plataforma de almacenamiento de objetos líder en la industria de NetApp. StorageGRID es una solución de almacenamiento basada en objetos y definida por software que admite API de objetos estándar de la industria, incluida la API de Amazon Simple Storage Service (S3). StorageGRID almacena y administra datos no estructurados a escala para proporcionar un almacenamiento de objetos seguro y duradero. El contenido se coloca en el lugar correcto, en el momento correcto y en el nivel de almacenamiento correcto, lo que optimiza los flujos de trabajo y reduce los costos de los medios enriquecidos distribuidos globalmente.

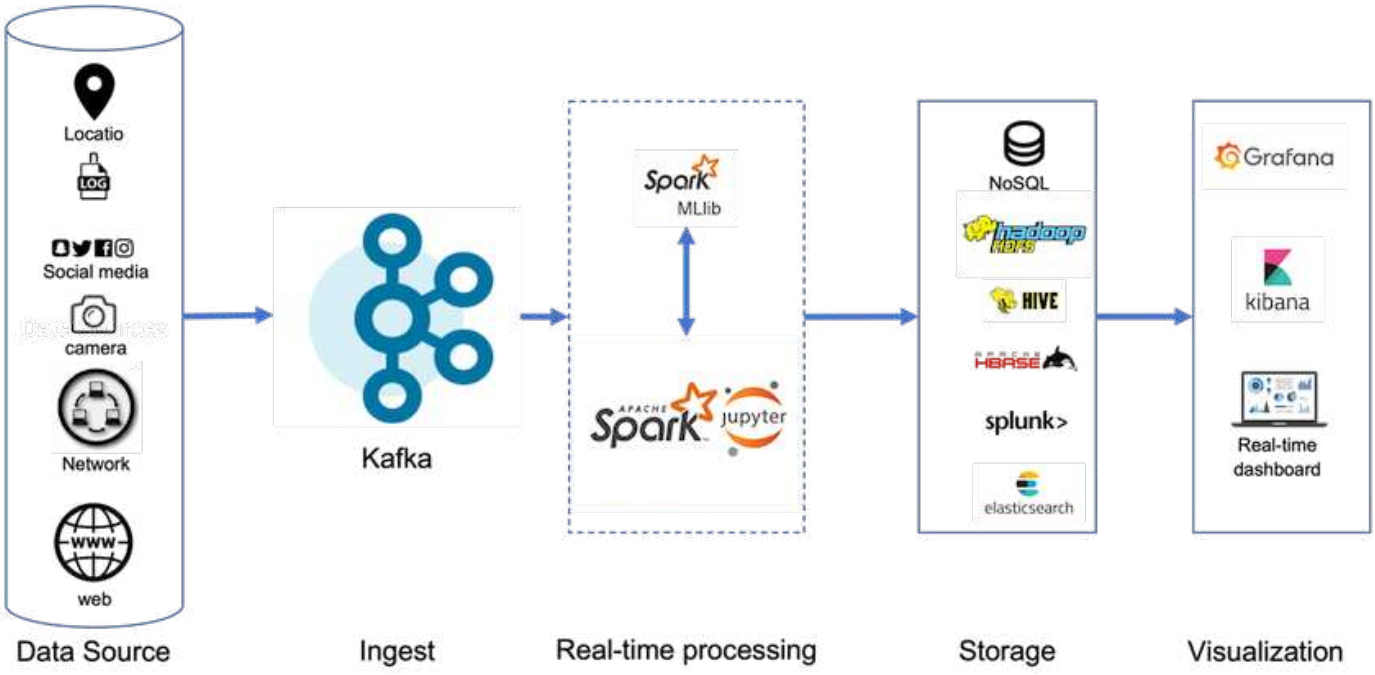
El mayor diferenciador de StorageGRID es su motor de políticas de gestión del ciclo de vida de la información (ILM) que permite la gestión del ciclo de vida de los datos basada en políticas. El motor de políticas puede usar metadatos para administrar cómo se almacenan los datos a lo largo de su vida útil para optimizar inicialmente el rendimiento y optimizar automáticamente el costo y la durabilidad a medida que los datos envejecen.

Habilitación del almacenamiento en niveles confluent

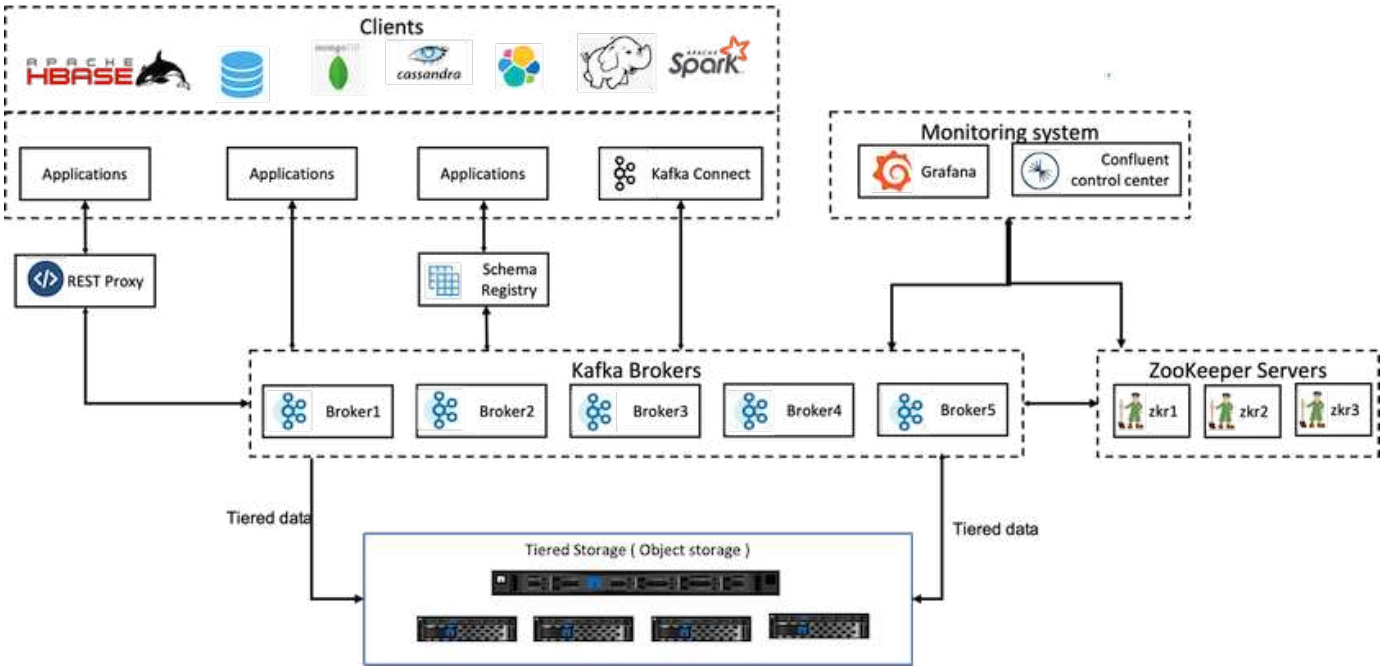
La idea básica del almacenamiento por niveles es separar las tareas de almacenamiento de datos del procesamiento de datos. Con esta separación, resulta mucho más fácil que el nivel de almacenamiento de datos y el nivel de procesamiento de datos escalen de forma independiente.

Una solución de almacenamiento por niveles para Confluent debe tener en cuenta dos factores. En primer lugar, debe solucionar o evitar las propiedades comunes de consistencia y disponibilidad del almacén de objetos, como las inconsistencias en las operaciones LIST y la falta de disponibilidad ocasional de objetos. En segundo lugar, debe gestionar correctamente la interacción entre el almacenamiento en niveles y el modelo de replicación y tolerancia a fallas de Kafka, incluida la posibilidad de que los líderes zombis sigan compensando rangos en niveles. El almacenamiento de objetos de NetApp brinda disponibilidad de objetos constante y el modelo HA hace que el almacenamiento agotado esté disponible para rangos de compensación de niveles. El almacenamiento de objetos de NetApp brinda disponibilidad de objetos constante y un modelo de alta disponibilidad (HA) para que el almacenamiento agotado esté disponible para rangos de compensación de niveles.

Con el almacenamiento en niveles, puede usar plataformas de alto rendimiento para lecturas y escrituras de baja latencia cerca del final de sus datos de transmisión, y también puede usar almacenes de objetos más económicos y escalables como NetApp StorageGRID para lecturas históricas de alto rendimiento. También tenemos una solución técnica para Spark con controlador de almacenamiento NetApp y los detalles están aquí. La siguiente figura muestra cómo Kafka encaja en un flujo de trabajo de análisis en tiempo real.



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



Detalles de la arquitectura de la solución

Esta sección cubre el hardware y el software utilizados para la verificación de Confluent. Esta información se aplica a la implementación de la plataforma Confluent con almacenamiento NetApp . La siguiente tabla cubre la arquitectura de la solución probada y los componentes base.

Componentes de la solución	Detalles
Confluent Kafka versión 6.2	• Tres cuidadores del zoológico • Cinco servidores de corredores • Cinco servidores de herramientas • Una Grafana • Un centro de control
Linux (ubuntu 18.04)	Todos los servidores
NetApp StorageGRID para almacenamiento en niveles	• Software StorageGRID • 1 x SG1000 (balanceador de carga) • 4 x SGF6024 • 4 x 24 x 800 SSD • Protocolo S3 • 4 x 100 GbE (conectividad de red entre el agente y las instancias de StorageGRID)
15 servidores Fujitsu PRIMERGY RX2540	Cada uno equipado con: * 2 CPU, 16 núcleos físicos en total * Intel Xeon * 256 GB de memoria física * Puerto dual de 100 GbE

Descripción general de la tecnología

Esta sección describe la tecnología utilizada en esta solución.

StorageGRID en NetApp

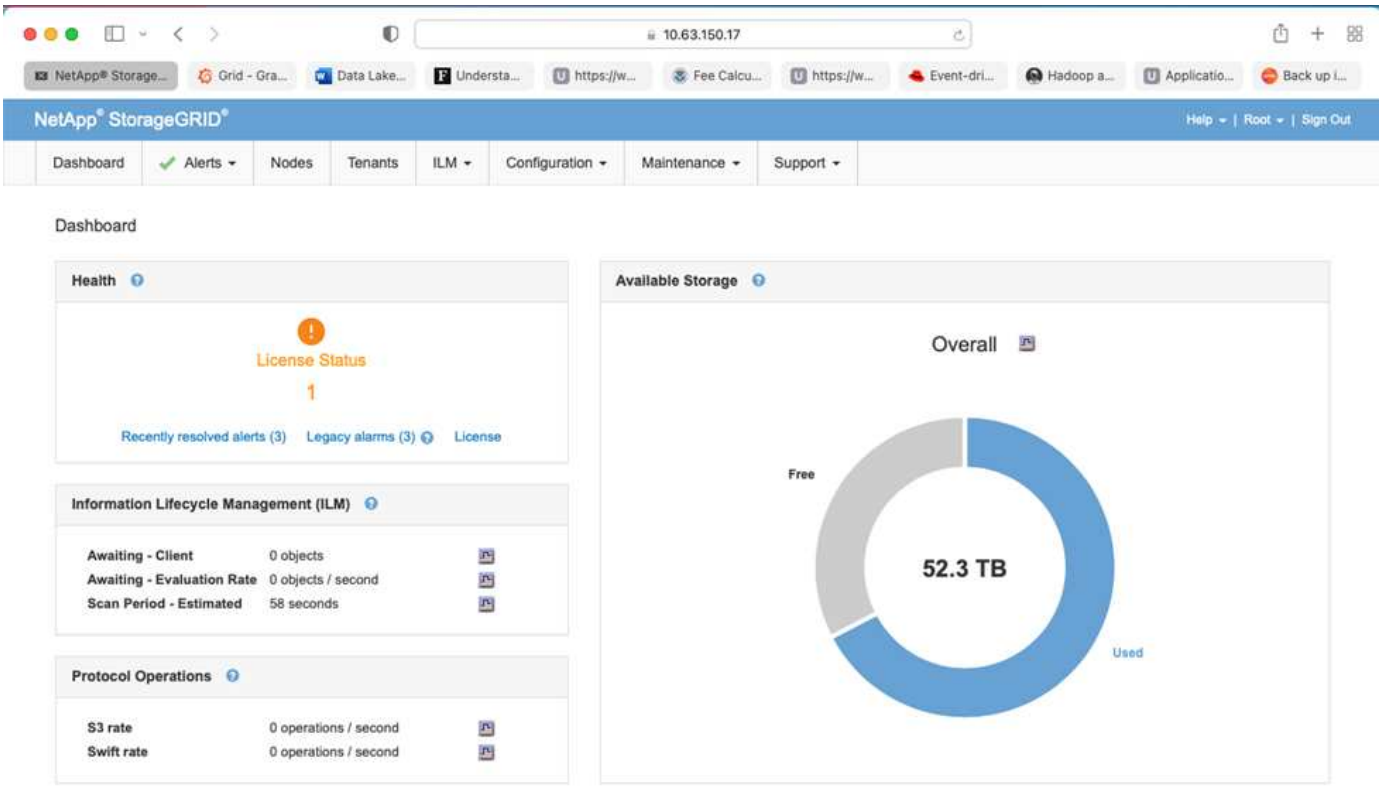
NetApp StorageGRID es una plataforma de almacenamiento de objetos rentable y de alto rendimiento. Al utilizar almacenamiento en niveles, la mayoría de los datos en Confluent Kafka, que se almacenan en el almacenamiento local o en el almacenamiento SAN del broker, se descargan al almacén de objetos remoto. Esta configuración genera mejoras operativas significativas al reducir el tiempo y el costo de reequilibrar, expandir o reducir clústeres o reemplazar un agente fallido. El almacenamiento de objetos juega un papel importante en la administración de datos que residen en el nivel de almacenamiento de objetos, por eso es importante elegir el almacenamiento de objetos correcto.

StorageGRID ofrece una gestión de datos global inteligente basada en políticas mediante una arquitectura de cuadrícula distribuida basada en nodos. Simplifica la gestión de petabytes de datos no estructurados y miles de millones de objetos a través de su omnipresente espacio de nombres de objetos globales combinado con sofisticadas funciones de gestión de datos. El acceso a objetos mediante una sola llamada se extiende a través de los sitios y simplifica las arquitecturas de alta disponibilidad al tiempo que garantiza el acceso continuo a los objetos, independientemente de las interrupciones del sitio o la infraestructura.

La multitenencia permite que múltiples aplicaciones de datos empresariales y en la nube no estructurados se atiendan de forma segura dentro de la misma red, lo que aumenta el retorno de la inversión y los casos de uso de NetApp StorageGRID. Puede crear múltiples niveles de servicio con políticas de ciclo de vida de objetos basadas en metadatos, optimizando la durabilidad, la protección, el rendimiento y la localidad en múltiples geografías. Los usuarios pueden ajustar las políticas de gestión de datos y monitorear y aplicar límites de tráfico para realinearlos al panorama de datos de manera no disruptiva a medida que sus requisitos cambian en entornos de TI en constante cambio.

Gestión sencilla con Grid Manager

StorageGRID Grid Manager es una interfaz gráfica basada en navegador que le permite configurar, administrar y monitorear su sistema StorageGRID en ubicaciones distribuidas globalmente en un solo panel.



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

- Administre repositorios de objetos, como imágenes, vídeos y registros, distribuidos globalmente y a escala de petabytes.
- Supervisar los nodos y servicios de la red para garantizar la disponibilidad de los objetos.
- Gestione la ubicación de los datos de objetos a lo largo del tiempo utilizando reglas de gestión del ciclo de vida de la información (ILM). Estas reglas rigen lo que sucede con los datos de un objeto después de su ingesta, cómo se protegen contra pérdidas, dónde se almacenan los datos del objeto y durante cuánto tiempo.
- Supervisar transacciones, rendimiento y operaciones dentro del sistema.

Políticas de gestión del ciclo de vida de la información

StorageGRID tiene políticas de administración de datos flexibles que incluyen mantener copias de réplica de sus objetos y usar esquemas EC (codificación de borrado) como 2+1 y 4+2 (entre otros) para almacenar sus objetos, dependiendo de los 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 deban cambiar con el tiempo. La modificación de las políticas de ILM es una característica fundamental que permite a los clientes de StorageGRID adaptarse a su entorno en constante cambio de forma rápida y sencilla.

Actuación

StorageGRID escala el rendimiento agregando más nodos de almacenamiento, que pueden ser máquinas virtuales, hardware o dispositivos diseñados específicamente como el"SG5712, SG5760, SG6060 o SGF6024" . En nuestras pruebas, superamos los requisitos clave de rendimiento de Apache Kafka con una cuadrícula de tres nodos de tamaño mínimo utilizando el dispositivo SGF6024. A medida que los clientes escalan su clúster de Kafka con agentes adicionales, pueden agregar más nodos de almacenamiento para aumentar el rendimiento y la capacidad.

Configuración del balanceador de carga y del punto final

Los nodos de administración en StorageGRID proporcionan la interfaz de usuario (IU) de Grid Manager y el punto final de API REST para ver, configurar y administrar su sistema StorageGRID , así como registros de auditoría para rastrear la actividad del sistema. Para proporcionar un punto final S3 de alta disponibilidad para el almacenamiento en niveles de Confluent Kafka, implementamos el balanceador de carga StorageGRID , que se ejecuta como un servicio en los nodos de administración y los nodos de puerta de enlace. Además, el balanceador de carga también administra el tráfico local y se comunica con GSLB (Global Server Load Balancing) para ayudar con la recuperación ante desastres.

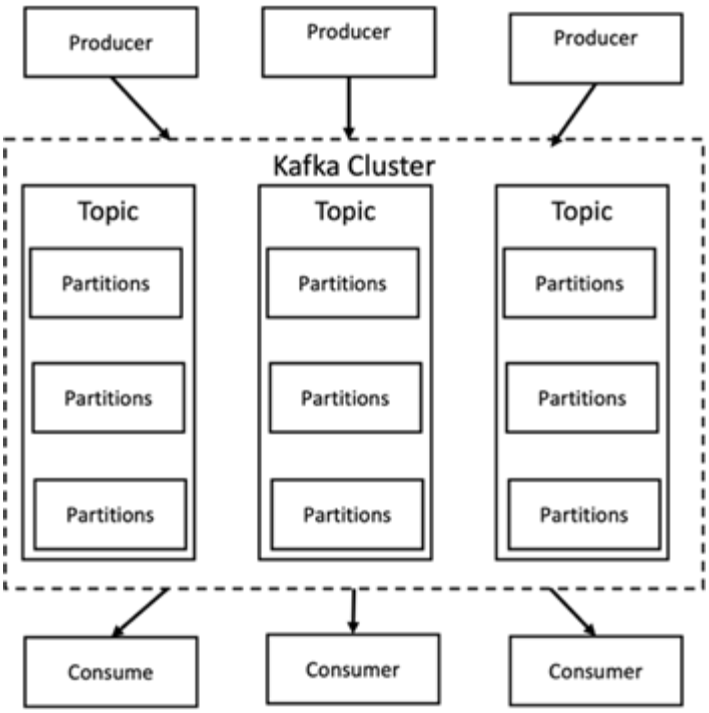
Para mejorar aún más la configuración de los puntos finales, StorageGRID proporciona políticas de clasificación de tráfico integradas en el nodo de administración, le permite monitorear el tráfico de su carga de trabajo y aplica varios límites de calidad de servicio (QoS) a sus cargas de trabajo. Las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

Clasificación del tráfico en StorageGRID

StorageGRID tiene funcionalidad QoS incorporada. Las políticas de clasificación de tráfico pueden ayudar a monitorear diferentes tipos de tráfico S3 provenientes de una aplicación cliente. Luego, puede crear y aplicar políticas para poner límites a este tráfico en función del ancho de banda de entrada y salida, la cantidad de solicitudes de lectura y escritura simultáneas o la tasa de solicitudes de lectura y escritura.

Apache Kafka

Apache Kafka es una implementación de marco de un bus de software que utiliza procesamiento de flujo escrito en Java y Scala. Su objetivo es proporcionar una plataforma unificada, de alto rendimiento y baja latencia para gestionar transmisiones de datos en tiempo real. Kafka puede conectarse a un sistema externo para exportar e importar datos a través de Kafka Connect y proporciona Kafka Streams, una biblioteca de procesamiento de flujos de Java. Kafka utiliza un protocolo binario basado en TCP que está optimizado para la eficiencia y se apoya en una abstracción de "conjunto de mensajes" que agrupa naturalmente los mensajes para reducir la sobrecarga del viaje de ida y vuelta de la red. Esto permite operaciones de disco secuenciales más grandes, paquetes de red más grandes y bloques de memoria contiguos, lo que permite a Kafka convertir un flujo ráfaga de escrituras de mensajes aleatorios en escrituras lineales. La siguiente figura representa el flujo de datos básico de Apache Kafka.



Kafka almacena mensajes clave-valor que provienen de una cantidad arbitraria de procesos llamados productores. Los datos se pueden dividir 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 tiempo. Otros procesos llamados consumidores pueden leer mensajes de las particiones. Para el procesamiento de flujos, Kafka ofrece la API Streams que permite escribir aplicaciones Java que consumen datos de Kafka y escriben los resultados en Kafka. Apache Kafka también funciona con sistemas de procesamiento de flujo externos como Apache Apex, Apache Flink, Apache Spark, Apache Storm y Apache NiFi.

Kafka se ejecuta en un clúster de uno o más servidores (llamados intermediarios) y las particiones de todos los temas se distribuyen entre los nodos del clúster. Además, las particiones se replican en múltiples intermediarios. Esta arquitectura permite a Kafka entregar flujos masivos de mensajes de manera tolerante a fallos y le ha permitido reemplazar algunos de los sistemas de mensajería convencionales como Java Message Service (JMS), Advanced Message Queuing Protocol (AMQP), etc. Desde el lanzamiento de la versión 0.11.0.0, Kafka ofrece escrituras transaccionales, que proporcionan exactamente un procesamiento de flujo mediante la API Streams.

Kafka admite dos tipos de temas: regulares y compactados. Los temas regulares se pueden configurar con un tiempo de retención o un límite de espacio. Si hay registros que son más antiguos que el tiempo de retención especificado o si se excede el límite de espacio para una partición, Kafka puede eliminar datos antiguos para liberar espacio de almacenamiento. De forma predeterminada, los temas están configurados con un tiempo de retención de 7 días, pero también es posible almacenar datos indefinidamente. Para los temas compactados, los registros no caducan según límites de tiempo o espacio. En cambio, Kafka trata los mensajes posteriores como actualizaciones de mensajes más antiguos con la misma clave y garantiza nunca eliminar el mensaje más reciente por clave. Los usuarios pueden eliminar mensajes por completo escribiendo un mensaje denominado "tombstone" con un valor nulo para una clave específica.

Hay cinco API principales en Kafka:

- **API de productor.** Permite que una aplicación publique flujos de registros.
- **API del consumidor.** Permite que una aplicación se suscriba a temas y procese flujos de registros.
- **API de conector.** Ejecuta las API de productor y consumidor reutilizables que pueden vincular los temas a

las aplicaciones existentes.

- **API de transmisiones.** Esta API convierte los flujos de entrada en salida y produce el resultado.
- **API de administración.** Se utiliza para administrar temas de Kafka, intermediarios y otros objetos de Kafka.

Las API de consumidor y productor se basan en el protocolo de mensajería de Kafka y ofrecen una implementación de referencia para los clientes consumidores y productores de Kafka en Java. El protocolo de mensajería subyacente es un protocolo binario que los desarrolladores pueden usar para escribir sus propios clientes consumidores o productores en cualquier lenguaje de programación. Esto desbloquea Kafka del ecosistema de la máquina virtual Java (JVM). En la wiki de Apache Kafka se mantiene una lista de clientes que no son Java disponibles.

Casos de uso de Apache Kafka

Apache Kafka es más popular para mensajería, seguimiento de actividad del sitio web, métricas, agregación de registros, procesamiento de transmisiones, abastecimiento de eventos y registro de confirmaciones.

- Kafka ha mejorado el rendimiento, la partición integrada, la replicación y la tolerancia a fallas, 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 (visitas de página, búsquedas) en un canal de seguimiento como un conjunto de feeds de publicación y suscripción en tiempo real.
- Kafka se utiliza a menudo para datos de seguimiento operativo. Esto implica agregar estadísticas de aplicaciones distribuidas para producir fuentes centralizadas de datos operativos.
- Muchas personas utilizan Kafka como reemplazo de una solución de agregación de registros. La agregación de registros generalmente recopila archivos de registro físicos 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 eventos como un flujo de mensajes. Esto permite un procesamiento de menor latencia y un soporte más sencillo para múltiples fuentes de datos y un consumo de datos distribuido.
- Muchos usuarios de Kafka procesan datos en canales de procesamiento que constan de varias etapas, en las que los datos de entrada sin procesar se consumen de los 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, un canal de procesamiento para recomendar artículos de noticias podría rastrear el contenido de los artículos desde fuentes RSS y publicarlo en un tema de "artículos". Un procesamiento posterior podría normalizar o desduplicar este contenido y publicar el contenido del artículo limpio en un nuevo tema, y una etapa de procesamiento final podría intentar recomendar este contenido a los usuarios. Estos canales de procesamiento crean gráficos de flujos de datos en tiempo real basados en temas individuales.
- El almacenamiento en caché de eventos es un estilo de diseño de aplicaciones para el cual los cambios de estado se registran como una secuencia de registros ordenada en el tiempo. El soporte de Kafka para datos de registros almacenados de gran tamaño lo convierte en un excelente backend para una aplicación creada 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 un mecanismo de resincronización para que los nodos fallidos restauren sus datos. La función de compactación de registros en Kafka ayuda a respaldar este caso de uso.

Confluente

Confluent Platform es una plataforma preparada para la empresa que completa Kafka con capacidades avanzadas diseñadas para ayudar a acelerar el desarrollo y la conectividad de las aplicaciones, permitir

transformaciones a través del procesamiento de flujo, simplificar las operaciones empresariales a escala y cumplir con estrictos requisitos arquitectónicos. Desarrollado por los creadores originales de Apache Kafka, Confluent amplía los beneficios de Kafka con funciones de nivel empresarial y al mismo tiempo elimina la carga de la administración o el monitoreo de Kafka. Hoy en día, más del 80% de las empresas Fortune 100 utilizan tecnología de transmisión de datos y la mayoría de ellas utilizan Confluent.

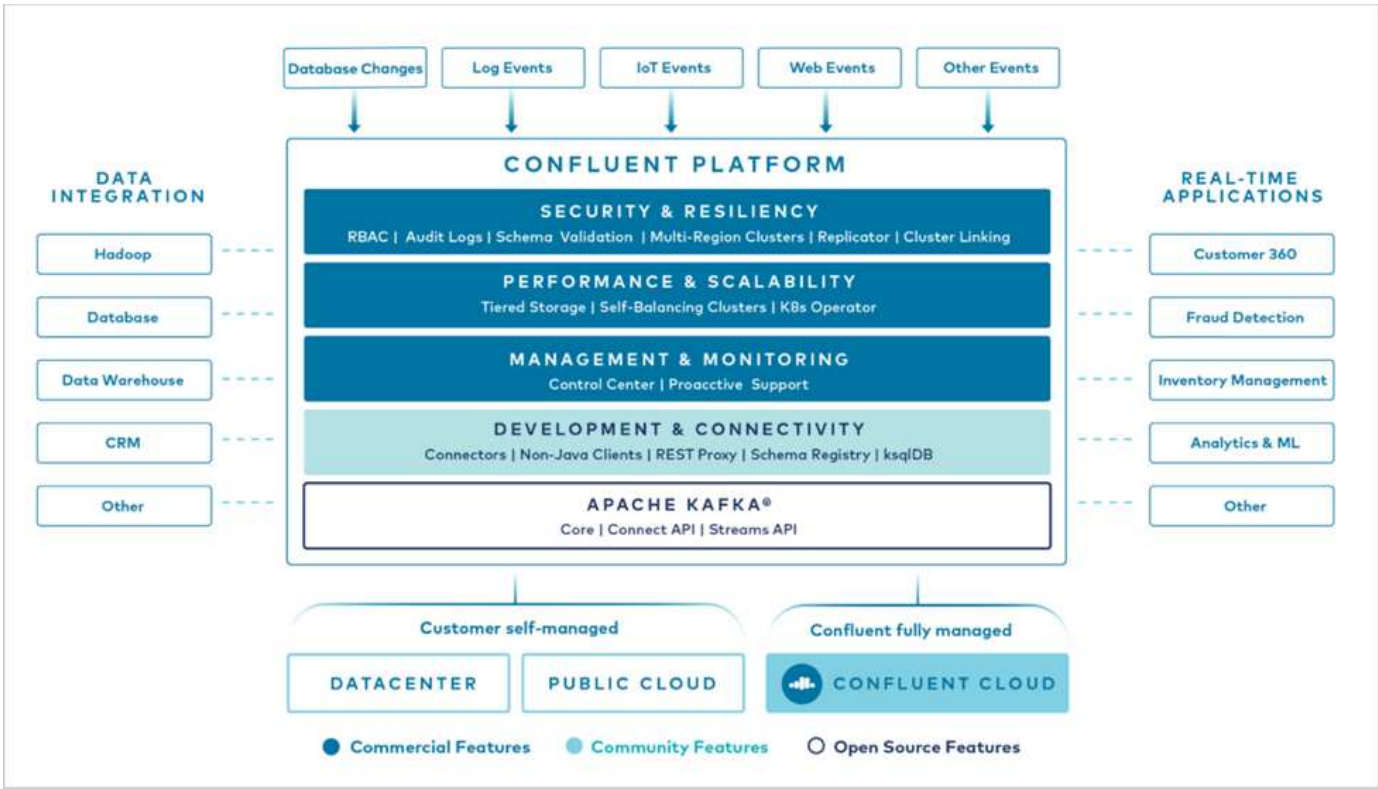
¿Por qué Confluent?

Al integrar datos históricos y en tiempo real en una única fuente central de verdad, Confluent facilita la creación de una categoría totalmente nueva de aplicaciones modernas basadas en eventos, obtiene una canalización de datos universal y desbloquea nuevos casos de uso poderosos con total escalabilidad, rendimiento y confiabilidad.

¿Para qué se utiliza Confluent?

Confluent Platform le permite centrarse en cómo obtener valor comercial de sus datos en lugar de preocuparse por la mecánica subyacente, como la forma en que se transportan o integran los datos entre sistemas dispares. En concreto, Confluent Platform simplifica la conexión de fuentes de datos a Kafka, la creación de aplicaciones de transmisión, así como la protección, la supervisión y la gestión de su infraestructura de Kafka. Hoy en día, Confluent Platform se utiliza para una amplia gama de casos de uso en numerosas industrias, desde servicios financieros, venta minorista omnicanal y automóviles autónomos hasta detección de fraudes, microservicios e IoT.

La siguiente figura muestra los componentes de la plataforma Confluent Kafka.



Descripción general de la tecnología de transmisión de eventos de Confluent

En el núcleo de la Plataforma Confluent se encuentra "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 manera tolerante a fallos.
- Procesar flujos de registros.

De fábrica, Confluent Platform también incluye Schema Registry, REST Proxy, un total de más de 100 conectores Kafka prediseñados y ksqlDB.

Descripción general de las funciones empresariales de la plataforma Confluent

- **Centro de Control de Confluentes.** Un sistema basado en GUI para administrar y supervisar Kafka. Le permite administrar fácilmente Kafka Connect y crear, editar y administrar conexiones a otros sistemas.
- **Confluent para Kubernetes.** Confluent for Kubernetes es un operador de Kubernetes. Los operadores de Kubernetes amplían las capacidades de orquestación de Kubernetes al proporcionar características y requisitos únicos para una aplicación de plataforma específica. Para Confluent Platform, esto incluye simplificar enormemente el proceso de implementación de Kafka en Kubernetes y automatizar las tareas típicas del ciclo de vida de la infraestructura.
- **Conectores confluentes a Kafka.** Los conectores utilizan la API de 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 y receptores de datos más populares, incluidas versiones totalmente probadas y compatibles de estos conectores con Confluent Platform. Se pueden encontrar más detalles "aquí" .
- **Clústeres autoequilibrados.** Proporciona equilibrio de carga automatizado, detección de fallas y autorreparación. Proporciona soporte para agregar o desmantelar corredores según sea necesario, sin necesidad de realizar ajustes manuales.
- **Enlace de clústeres confluentes.** Conecta directamente los clústeres entre sí y refleja temas de un clúster a otro a través de un puente de enlace. La vinculación de clústeres simplifica la configuración de implementaciones de múltiples centros de datos, múltiples clústeres y nubes híbridas.
- **Balanceador automático de datos Confluent.** Supervisa su clúster para conocer la cantidad de intermediarios, el tamaño de las particiones, la cantidad de particiones y la cantidad de líderes dentro del clúster. Le permite cambiar datos para crear una carga de trabajo uniforme en todo el clúster, al mismo tiempo que limita el tráfico de reequilibrio para minimizar el efecto en las cargas de trabajo de producción durante el reequilibrio.
- **Replicador confluente.** Hace que sea más fácil que nunca mantener múltiples clústeres de Kafka en múltiples centros de datos.
- **Almacenamiento por niveles.** Proporciona opciones para almacenar grandes volúmenes de datos de Kafka utilizando su proveedor de nube favorito, reduciendo así la carga operativa y los costos. Con el almacenamiento por niveles, puede mantener los datos en un almacenamiento de objetos rentable y escalar intermediarios solo cuando necesite más recursos computacionales.
- **Cliente JMS Confluent.** Confluent Platform incluye un cliente compatible con JMS para Kafka. Este cliente de Kafka implementa la API estándar JMS 1.1, utilizando intermediarios de Kafka como backend. Esto es útil si tiene aplicaciones heredadas que usan JMS y desea reemplazar el agente de mensajes JMS existente con Kafka.
- **Proxy MQTT confluente.** Proporciona una manera de publicar datos directamente en Kafka desde dispositivos y puertas de enlace MQTT sin la necesidad de un agente MQTT en el medio.
- **Complementos de seguridad de Confluent.** Los complementos de seguridad de Confluent se utilizan para agregar capacidades de seguridad a varias herramientas y productos de la plataforma Confluent. Actualmente, hay un complemento disponible para el proxy REST de Confluent que ayuda a autenticar las solicitudes entrantes y propagar el principal autenticado a las solicitudes a Kafka. Esto permite que los clientes proxy REST de Confluent utilicen las funciones de seguridad multiinquilino del bróker Kafka.

Verificación confluente

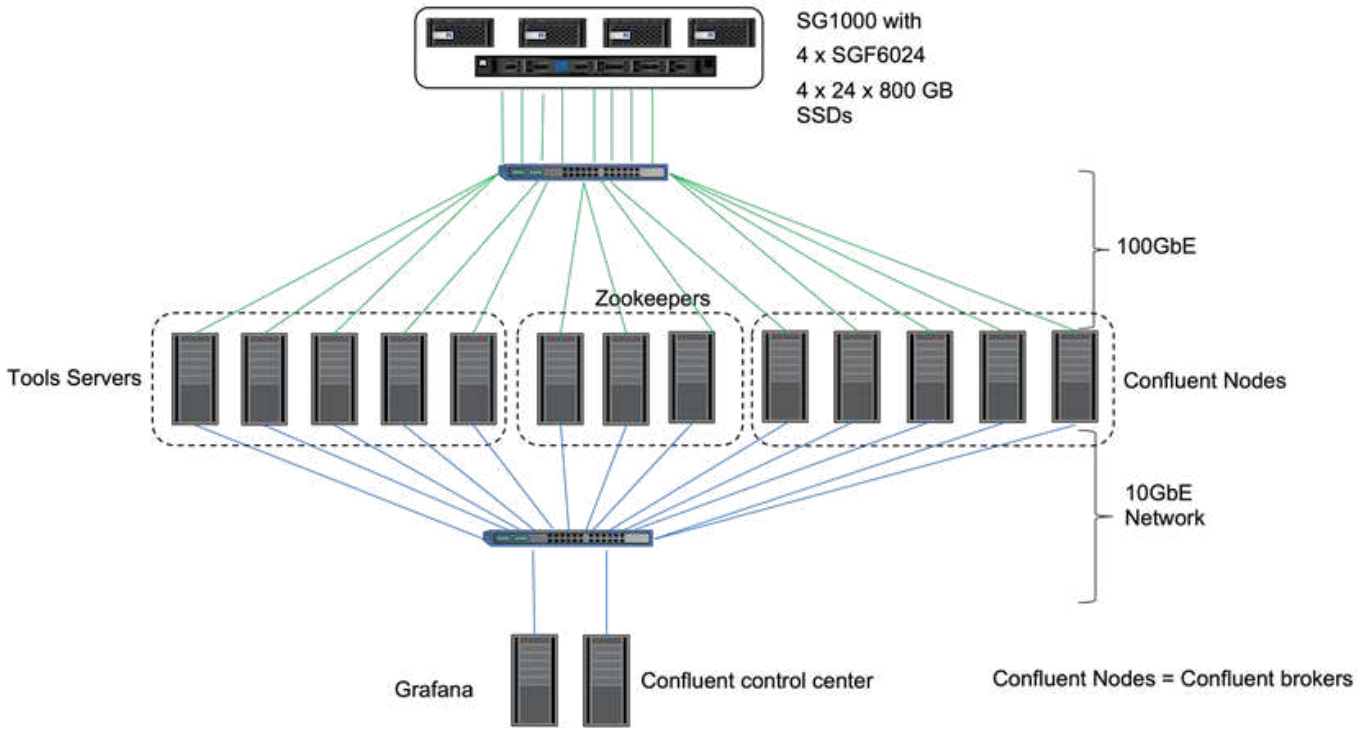
Realizamos la verificación con Confluent Platform 6.2 Tiered Storage en NetApp StorageGRID. Los equipos de NetApp y Confluent trabajaron juntos en esta verificación y ejecutaron los casos de prueba necesarios para la verificación.

Configuración de la plataforma Confluent

Utilizamos la siguiente configuración para la verificación.

Para la verificación, utilizamos tres guardianes del zoológico, cinco corredores, cinco servidores de ejecución de scripts de prueba, servidores de herramientas con nombre con 256 GB de RAM y 16 CPU. Para el almacenamiento de NetApp , utilizamos StorageGRID con un balanceador de carga SG1000 con cuatro SGF6024. El almacenamiento y los intermediarios se conectaron a través de conexiones de 100 GbE.

La siguiente figura muestra la topología de red de la configuración utilizada para la verificación de Confluent.



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

Configuración de almacenamiento en niveles de Confluent

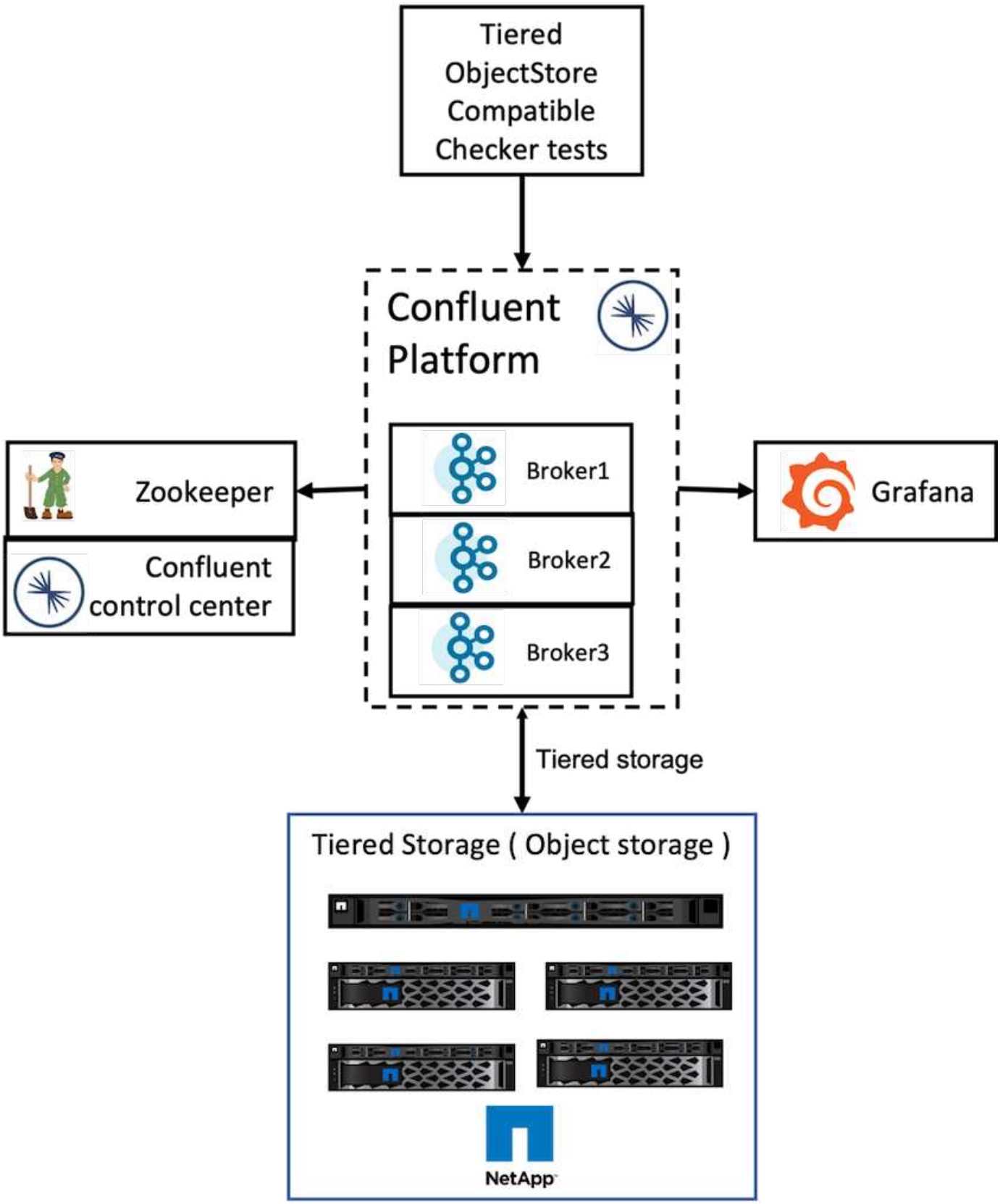
La configuración de almacenamiento por niveles requiere los siguientes parámetros en Kafka:

```
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 proporcionado en el `confluent.tier.s3.cred.file.path` parámetro.

Almacenamiento de objetos de NetApp - StorageGRID

Configuramos la configuración de sitio único en StorageGRID para la verificación.



Pruebas de verificación

Completamos los siguientes cinco casos de prueba para la verificación. Estas pruebas se ejecutan en el marco 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, obtener/colocar/eliminar) en la API del almacén de objetos funcionan bien de acuerdo con las necesidades del almacenamiento en niveles. Es una prueba básica que todo servicio de almacenamiento de objetos debe esperar pasar antes de las siguientes pruebas. Es una prueba asertiva que o se aprueba o se suspende.

Prueba de corrección de la funcionalidad de niveles

Esta prueba determina si la funcionalidad de almacenamiento en niveles de extremo a extremo funciona bien con una prueba asertiva que pasa o falla. La prueba crea un tema de prueba que, de manera predeterminada, está configurado con niveles habilitados y un tamaño de conjunto activo muy reducido. Produce un flujo de eventos para el tema de prueba recién creado, espera a que los intermediarios archiven los segmentos en el almacén de objetos y luego consume el flujo de eventos y valida que el flujo consumido coincida con el flujo producido. La cantidad de mensajes producidos en el flujo de eventos es configurable, lo que permite al usuario generar una carga de trabajo suficientemente grande según las necesidades de las pruebas. El tamaño reducido del conjunto activo garantiza que las búsquedas del consumidor fuera del segmento activo se atiendan solo desde el almacén de objetos; esto ayuda a probar la exactitud del almacén de objetos para las lecturas. Hemos realizado esta prueba con y sin inyección de falla en el almacén de objetos. Simulamos una falla de nodo deteniendo el servicio del administrador de servicios en uno de los nodos en StorageGRID y validando que la funcionalidad de extremo a extremo funciona con el almacenamiento de objetos.

Punto de referencia de búsqueda de niveles

Esta prueba validó el rendimiento de lectura del almacenamiento de objetos en niveles y verificó las solicitudes de lectura de obtención de rango bajo una carga pesada de los segmentos generados por el punto de referencia. En este punto de referencia, Confluent desarrolló clientes personalizados para atender las solicitudes de búsqueda de niveles.

Referencia de carga de trabajo de producción y consumo

Esta prueba generó indirectamente una carga de trabajo de escritura en el almacén de objetos a través del 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. Esta prueba verificó el rendimiento de lectura y escritura en el almacenamiento de objetos en subprocesos paralelos. Realizamos pruebas con y sin inyección de fallas en el almacén de objetos tal como lo hicimos para la prueba de corrección de la funcionalidad de niveles.

Punto de referencia de la carga de trabajo de retención

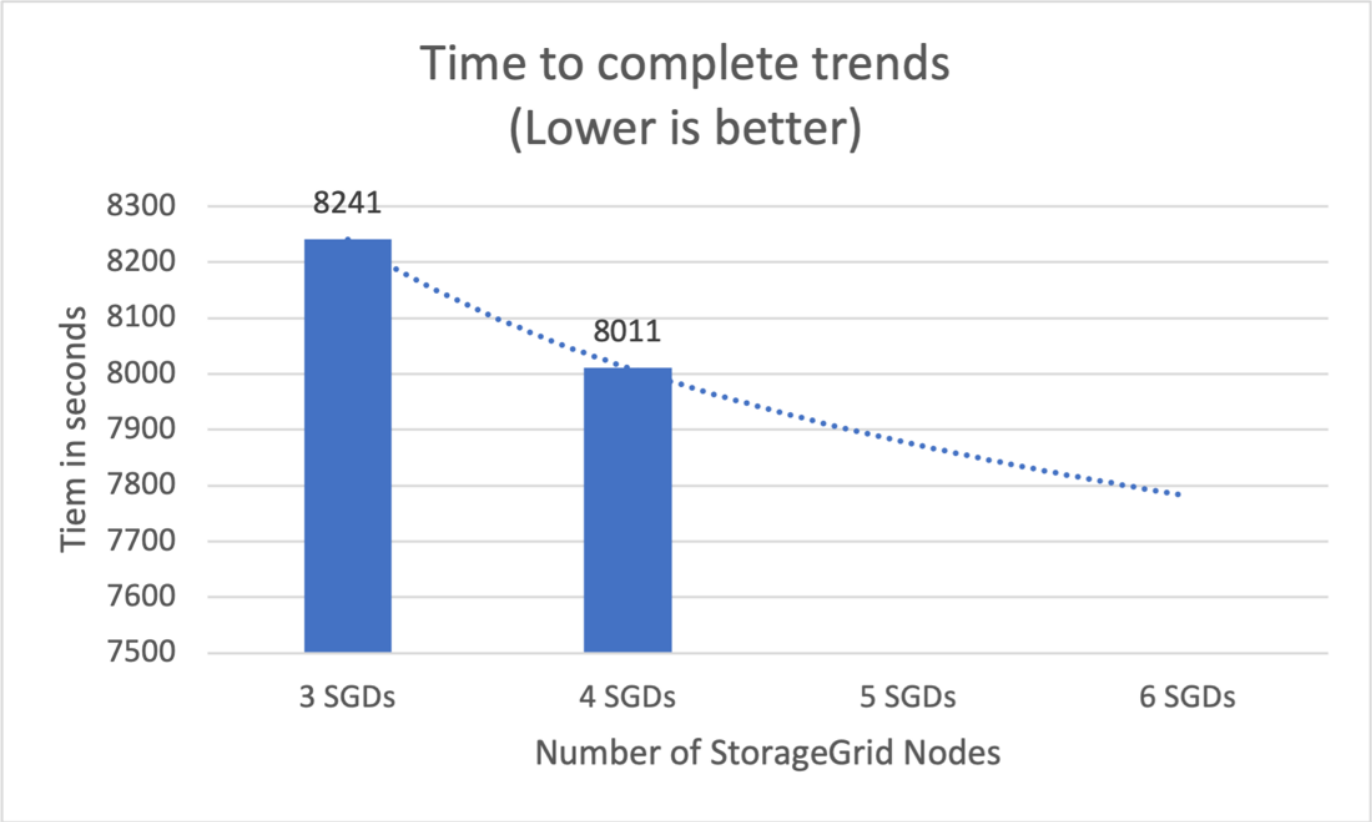
Esta prueba verificó el rendimiento de eliminación de un almacén de objetos bajo una carga de trabajo pesada de retención de temas. La carga de trabajo de retención se generó utilizando un script de prueba que produce muchos mensajes en paralelo a un tema de prueba. El tema de prueba fue configurar una configuración de retención agresiva basada en el tamaño y el tiempo que provocó que el flujo de eventos se purgara continuamente del almacén de objetos. Los segmentos fueron luego archivados. Esto provocó una gran cantidad de eliminaciones en el almacenamiento de objetos por parte del agente y la recopilación del rendimiento de las operaciones de eliminación del almacén de objetos.

Pruebas de rendimiento con escalabilidad

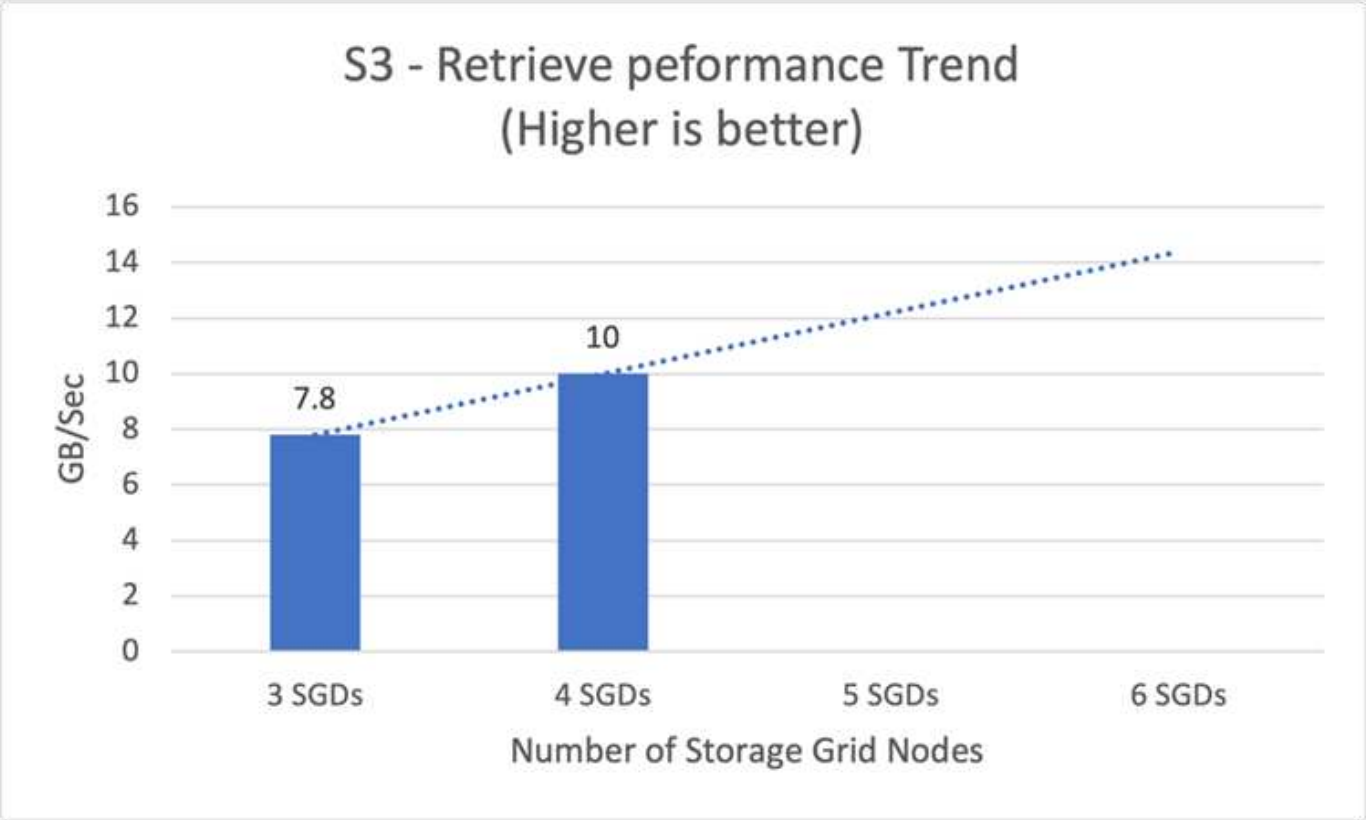
Realizamos pruebas de almacenamiento en niveles con tres o cuatro nodos para cargas de trabajo de productores y consumidores con la configuración NetApp StorageGRID . Según nuestras pruebas, el tiempo de finalización y los resultados de rendimiento fueron

directamente proporcionales a la cantidad de nodos StorageGRID . La configuración de StorageGRID requirió un mínimo de tres nodos.

- El tiempo para completar la operación de producción y consumo disminuyó linealmente cuando aumentó el número de nodos de almacenamiento.



- El rendimiento de la operación de recuperación s3 aumentó linealmente según la cantidad de nodos StorageGRID . StorageGRID admite hasta 200 nodos StorgeGRID.



Conector s3 confluente

El conector Amazon S3 Sink exporta datos de temas de Apache Kafka a objetos S3 en formatos Avro, JSON o Bytes. El conector de sumidero 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 la clave codifica el tema, la partición de Kafka y el desplazamiento de inicio de este fragmento de datos.

En esta configuración, le mostramos cómo leer y escribir temas en el almacenamiento de objetos desde Kafka directamente usando el conector de receptor Kafka s3. Para esta prueba, utilizamos un clúster Confluent independiente, pero esta configuración es aplicable a un clúster distribuido.

1. Descargue Confluent Kafka desde el sitio web de Confluent.
2. Descomprima el paquete en una carpeta en su 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 de Confluent Kafka, el clúster crea una carpeta raíz temporal en `/tmp` También crea carpetas de Zookeeper, Kafka, un registro de esquema, connect, un ksql-server y un centro de control y copia sus respectivos archivos de configuración desde `$CONFLUENT_HOME` . Vea 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. Configurar 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 el `server. xxx` propiedad. De forma predeterminada, se necesitan tres guardianes del zoológico para la selección del líder de Kafka.

6. Creamos 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. Utilizamos valores predeterminados para las configuraciones de Kafka, connect, control-center, Kafka, Kafka-rest, ksql-server y schema-registry.

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, que ayuda a solucionar problemas. En algunos casos los servicios tardan más tiempo en iniciarse. Asegúrese de que todos los servicios estén en funcionamiento.

8. Instalar Kafka connect usando `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 puedes instalar una versión específica usando `confluent-hub install confluentinc/kafka-connect-s3:10.0.3`.

9. Por defecto, `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 del complemento 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 de 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. Configurar el ID de acceso y la clave secreta en el `/root/.aws/credentials` archivo.

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

13. Verifique que el depósito sea accesible.

```
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-sink para la configuración de s3 y del bucket.

```
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. Importar algunos registros al bucket 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 s3-sink.

```
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. Verifique 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. Verifique el registro para asegurarse de que s3-sink esté listo para aceptar temas.

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

19. Consulte 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. Verifique los objetos en el bucket 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 de S3 a su 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 el "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:~#
```

Conectores de Kafka Connect de Instaclustr

Instaclustr admite los conectores de Kafka Connect y sus detalles: "Más detalles". Instaclustr proporciona conectores adicionales "sus detalles"

Clústeres autoequilibrados confluentes

Si ha administrado un clúster de Kafka anteriormente, probablemente esté familiarizado con los desafíos que implica reasignar manualmente particiones a diferentes agentes para asegurarse de que la carga de trabajo esté equilibrada en todo el clúster. Para las organizaciones con grandes implementaciones de Kafka, reorganizar grandes cantidades de datos puede ser una tarea abrumadora, tediosa y riesgosa, especialmente si las aplicaciones de misión crítica se crean sobre 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 autoequilibrio de Confluent, que automatiza el reequilibrio en función de los cambios en la topología del clúster o de la carga desigual. La prueba de reequilibrio de Confluent ayuda a medir el tiempo necesario para agregar un nuevo agente cuando falla un nodo o el nodo de escalamiento requiere reequilibrar los datos entre los agentes. En las configuraciones clásicas de Kafka, la cantidad de datos a reequilibrar crece a medida que crece el clúster, pero, en el almacenamiento en niveles, el reequilibrio está restringido a una pequeña cantidad de datos. Según nuestra validación, el reequilibrio en el almacenamiento en niveles toma segundos o minutos en una arquitectura clásica de Kafka y crece linealmente a medida que crece el clúster.

En los clústeres con autoequilibrio, los reequilibrios de particiones están completamente automatizados para optimizar el rendimiento de Kafka, acelerar el escalamiento del agente y reducir la carga operativa de ejecutar un clúster grande. En estado estable, los clústeres autoequilibrados monitorean la desviación de los datos entre los intermediarios y reasignan particiones continuamente para optimizar el rendimiento del clúster. Al escalar la plataforma hacia arriba o hacia abajo, los clústeres de autoequilibrio reconocen automáticamente la presencia de nuevos intermediarios o la eliminación de intermediarios antiguos y activan una reasignación de partición posterior. Esto le permite agregar y desmantelar corredores fácilmente, lo que hace que sus clústeres de Kafka sean fundamentalmente más elásticos. Estos beneficios se obtienen sin necesidad de intervención manual, cálculos matemáticos complejos o el riesgo de error humano que normalmente conllevan las reasignaciones de particiones. Como resultado, los reequilibrios de datos se completan en mucho menos tiempo y usted puede concentrarse en proyectos de transmisión de eventos de mayor valor en lugar de tener que supervisar constantemente sus clústeres.

Instaclustr también admite funciones de reequilibrio automático y se ha implementado para múltiples clientes.

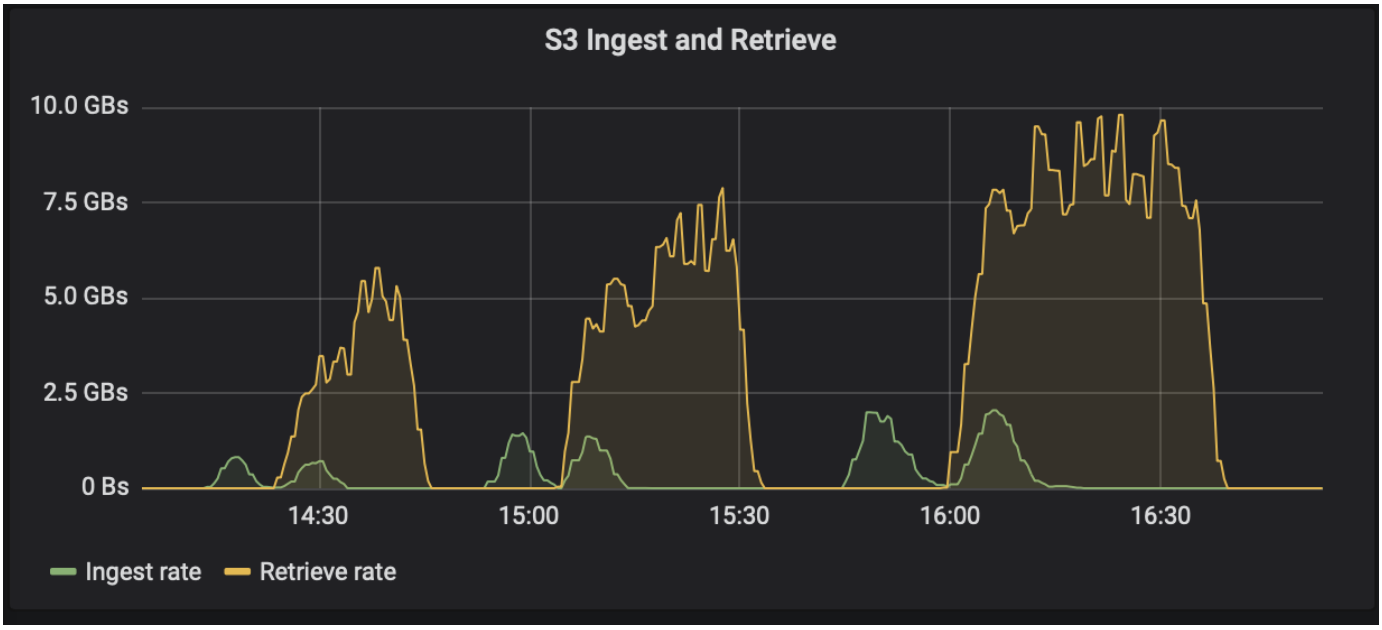
Pautas de mejores prácticas

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

- Según nuestra validación, el almacenamiento de objetos S3 es la mejor opción para que Confluent conserve datos.
- Podemos usar SAN de alto rendimiento (específicamente FC) para mantener los datos activos del broker o el disco local, porque, en la configuración de almacenamiento en niveles de Confluent, el tamaño de los datos almacenados en el directorio de datos del broker se basa en el tamaño del segmento y el tiempo de retención cuando los datos se mueven al almacenamiento de objetos.

- Los almacenes de objetos proporcionan un mejor rendimiento cuando segment.bytes es mayor; probamos 512 MB.
- En Kafka, la longitud de la clave o el valor (en bytes) para cada registro producido para el tema está controlada por el `length.key.value` parámetro. Para StorageGRID, el rendimiento de ingesta y recuperación de objetos S3 aumentó a valores más altos. Por ejemplo, 512 bytes proporcionaron una recuperación de 5,8 GBps, 1024 bytes proporcionaron una recuperación s3 de 7,5 GBps y 2048 bytes proporcionaron cerca de 10 GBps.

La siguiente figura presenta la ingesta y recuperación de objetos S3 en función de `length.key.value` .



- **Afinación de Kafka.** Para mejorar el rendimiento del almacenamiento en niveles, puede aumentar TierFetcherNumThreads y TierArchiverNumThreads. Como regla general, se recomienda aumentar TierFetcherNumThreads para que coincida con la cantidad de núcleos de CPU físicos y aumentar TierArchiverNumThreads a la mitad de la cantidad de núcleos de CPU. Por ejemplo, en las propiedades del servidor, si tiene una máquina con ocho núcleos físicos, configure confluent.tier.fetcher.num.threads = 8 y confluent.tier.archiver.num.threads = 4.
- **Intervalo de tiempo para eliminar temas.** Cuando se elimina un tema, la eliminación de los archivos de segmentos de registro en el almacenamiento de objetos no comienza de inmediato. Más bien, hay un intervalo de tiempo con un valor predeterminado de 3 horas antes de que se eliminen esos archivos. Puede modificar la configuración, confluent.tier.topic.delete.check.interval.ms, para cambiar el valor de este intervalo. Si elimina un tema o un clúster, también puede eliminar manualmente los objetos en el depósito correspondiente.
- **ACL sobre temas internos de almacenamiento en niveles.** Una práctica recomendada para las implementaciones locales es habilitar un autorizador de ACL en los temas internos utilizados para el almacenamiento en niveles. Establezca reglas de ACL para limitar el acceso a estos datos únicamente al usuario del corredor. Esto protege los temas internos y evita el acceso no autorizado a los metadatos y datos de almacenamiento en niveles.

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

Reemplazar al usuario `<kafka>` con el principal del broker real en su implementación.

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

Apresto

El dimensionamiento de Kafka se puede realizar con cuatro modos de configuración: simple, granular, inverso y particiones.

Simple

El modo simple es apropiado para quienes utilizan Apache Kafka por primera vez o para casos de uso en etapas iniciales. Para este modo, debe proporcionar requisitos como el rendimiento en MBps, la distribución de lectura, la retención y el porcentaje de utilización de recursos (el 60 % es el valor predeterminado). También ingresa al entorno, como local (bare-metal, VMware, Kubernetes u OpenStack) o en la nube. En función de esta información, el dimensionamiento de un clúster de Kafka proporciona la cantidad de servidores necesarios para el broker, el zookeeper, los trabajadores de conexión de Apache Kafka, el registro de esquema, un proxy REST, ksqlDB y el centro de control de Confluent.

Para el almacenamiento en niveles, considere el modo de configuración granular para dimensionar un clúster de Kafka. El modo granular es apropiado para usuarios experimentados de Apache Kafka o casos de uso bien definidos. Esta sección describe el dimensionamiento para productores, procesadores de flujo y consumidores.

Productores

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

- **Nombre.** Chispa.
- **Tipo de productor.** Aplicación o servicio, proxy (REST, MQTT, otros) y base de datos existente (RDBMS, NOSQL, otros). También puedes seleccionar "No sé".
- **Rendimiento promedio.** En eventos por segundo (1.000.000 por ejemplo).
- **Rendimiento máximo.** En eventos por segundo (4.000.000 por ejemplo).
- **Tamaño promedio del mensaje.** En bytes, sin comprimir (máximo 1 MB; 1000 por ejemplo).
- **Formato del mensaje.** Las opciones incluyen Avro, JSON, buffers de protocolo, binario, texto, "No sé" y otros.
- **Factor de replicación.** Las opciones son 1, 2, 3 (recomendación Confluent), 4, 5 o 6.

- **Tiempo de retención.** Un día (por ejemplo). ¿Cuánto tiempo desea que sus datos se almacenen en Apache Kafka? Introduzca -1 con cualquier unidad por tiempo infinito. La calculadora asume un tiempo de retención de 10 años para una retención infinita.
- Seleccione la casilla de verificación "¿Habilitar almacenamiento en niveles para disminuir la cantidad de agentes y permitir almacenamiento infinito?"
- Cuando el almacenamiento en niveles está habilitado, los campos de retención controlan el conjunto activo de datos que se almacenan localmente en el agente. Los campos de retención de archivo controlan durante cuánto tiempo se almacenan los datos en el almacenamiento de objetos de archivo.
- **Retención de almacenamiento de archivo.** Un año (por ejemplo). ¿Durante cuánto tiempo desea que sus datos se mantengan almacenados en el almacenamiento de archivo? Introduce -1 con cualquier unidad durante una duración infinita. La calculadora asume una retención de 10 años para una 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, configure este parámetro en un multiplicador de crecimiento.
- **Número de instancias de productor.** 10 (por ejemplo). ¿Cuántas instancias de productor se ejecutarán? 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 incorporada en el cálculo.

Con base en este ejemplo de entrada, el dimensionamiento tiene el siguiente efecto sobre los productores:

- Rendimiento promedio en bytes sin comprimir: 1 GBps. Rendimiento máximo en bytes sin comprimir: 4 GBps. Rendimiento promedio en bytes comprimidos: 400 MBps. Rendimiento máximo en bytes comprimidos: 1,6 GBps. Esto se basa en una tasa de compresión predeterminada del 60 % (puede cambiar este valor).
    - Almacenamiento total en el broker requerido: 31 104 TB, incluida la replicación, comprimido. Almacenamiento de archivo total fuera del bróker requerido: 378 432 TB, comprimido. Usar"https://fusion.netapp.com" para dimensionar StorageGRID .

Los procesadores de flujo deben describir sus aplicaciones o servicios que consumen datos de Apache Kafka y los producen nuevamente en Apache Kafka. En la mayoría de los casos, estos se construyen en ksqlDB o Kafka Streams.

- **Nombre.** Serpentina de chispas.
- **Tiempo de procesamiento.** ¿Cuánto tiempo tarda este procesador en procesar un solo mensaje?
    - 1 ms (transformación simple, sin estado) [ejemplo], 10 ms (operación en memoria con estado).
    - 100 ms (operación 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 flujo produce su salida en Apache Kafka. ¿Cuánto tiempo desea que se almacenen estos datos de salida en Apache Kafka? Introduce -1 con cualquier unidad durante una duración infinita.
- Seleccione la casilla de verificación "¿Habilitar almacenamiento en niveles para disminuir el número de agentes y permitir almacenamiento infinito?"
- **Retención de almacenamiento de archivo.** 1 año (por ejemplo). ¿Durante cuánto tiempo desea que sus datos se mantengan almacenados en el almacenamiento de archivo? Introduce -1 con cualquier unidad durante una duración infinita. La calculadora asume una retención de 10 años para una retención infinita.
- **Porcentaje de paso de salida.** 100 (por ejemplo). Un procesador de flujo produce su salida en Apache Kafka. ¿Qué porcentaje del rendimiento entrante se devolverá a Apache Kafka? Por ejemplo, si el

rendimiento de entrada es de 20 MBps y este valor es 10, el rendimiento de salida será de 2 MBps.

- ¿Desde qué aplicaciones se lee esto? Seleccione “Spark”, el nombre utilizado en el dimensionamiento basado en el tipo de productor. Con base en la información anterior, puede esperar los siguientes efectos del tamaño en las instancias del procesador de flujo y las estimaciones de partición de temas:
- Esta aplicación de procesador de flujo requiere la siguiente cantidad de instancias. Es probable que los temas entrantes también requieran esta cantidad de particiones. Póngase en contacto con Confluent para confirmar este parámetro.
  - 1.000 para un rendimiento promedio sin multiplicador de crecimiento
  - 4.000 para un rendimiento máximo sin multiplicador de crecimiento
  - 1.000 para un rendimiento promedio con un multiplicador de crecimiento
  - 4.000 para un rendimiento máximo con un multiplicador de crecimiento

Consumidores

Describa sus aplicaciones o servicios que consumen datos de Apache Kafka y no los producen nuevamente en Apache Kafka; por ejemplo, un cliente nativo o un conector de Kafka.

- **Nombre.** Consumidor de Spark.
- **Tiempo de procesamiento.** ¿Cuánto tiempo 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 REST de terceros)
  - Algún otro proceso referencial de duración conocida.
- **Tipo de consumidor.** Aplicación, proxy o receptor de un almacén de datos existente (RDBMS, NoSQL, otros).
- ¿Desde qué aplicaciones se lee esto? Conecte este parámetro con el productor y el tamaño del flujo determinados previamente.

Con base en la información anterior, debe determinar el tamaño de las instancias de consumidor y las estimaciones de partición de temas. Una aplicación de consumidor requiere la siguiente cantidad de instancias.

- 2.000 para un rendimiento promedio, sin multiplicador de crecimiento
- 8.000 para un rendimiento máximo, sin multiplicador de crecimiento
- 2.000 para un rendimiento promedio, incluido el multiplicador de crecimiento
- 8.000 para un rendimiento máximo, incluido el multiplicador de crecimiento

Es probable que los temas entrantes también necesiten esta cantidad de particiones. Comuníquese con Confluent para confirmar.

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

- **Tiempo de reconstrucción.** Por ejemplo, 4 horas. Si un host de agente Apache Kafka falla, se pierden sus datos y se aprovisiona un nuevo host para reemplazar al host fallado, ¿con qué rapidez debe reconstruirse este nuevo host? Deje este parámetro en blanco si se desconoce el valor.

- **Objetivo de utilización de recursos (porcentaje).** Por ejemplo, 60. ¿Qué tan utilizados desea que estén sus hosts durante el rendimiento promedio? Confluent recomienda una utilización del 60 % a menos que utilice clústeres de autoequilibrio de Confluent, en cuyo caso la utilización puede ser mayor.

Describe tu entorno

- ¿En qué entorno se ejecutará su clúster? ¿Amazon Web Services, Microsoft Azure, plataforma en la nube de Google, hardware local, VMware local, OpenStack local o Kubernates local?
- **Detalles del anfitrión.** Número de núcleos: 48 (por ejemplo), tipo de tarjeta de red (10GbE, 40GbE, 16GbE, 1GbE u otro tipo).
- **Volúmenes de almacenamiento.** Anfitrión: 12 (por ejemplo). ¿Cuántos discos duros o SSD se admiten por host? Confluent recomienda 12 discos duros por host.
- **Capacidad/volumen de almacenamiento (en GB).** 1000 (por ejemplo). ¿Cuánto almacenamiento puede almacenar un solo volumen en gigabytes? Confluent recomienda discos de 1TB.
- **Configuración de almacenamiento.** ¿Cómo se configuran los volúmenes de almacenamiento? Confluent recomienda RAID10 para aprovechar todas las funciones de Confluent. También se admiten JBOD, SAN, RAID 1, RAID 0, RAID 5 y otros tipos.
- **Rendimiento de volumen único (MBps).** 125 (por ejemplo). ¿Qué tan rápido puede leer o escribir un solo volumen de almacenamiento en megabytes por segundo? Confluent recomienda discos duros estándar, que normalmente tienen un rendimiento de 125 MBps.
- **Capacidad de memoria (GB).** 64 (por ejemplo).

Después de haber determinado sus variables ambientales, seleccione Dimensionar mi clúster. Basándonos en los parámetros de ejemplo indicados anteriormente, determinamos el siguiente tamaño para Confluent Kafka:

- **Apache Kafka.** Número de corredores: 22. Su clúster está limitado al almacenamiento. Considere habilitar el almacenamiento por niveles para disminuir la cantidad de hosts y permitir un almacenamiento infinito.
- **Apache Guardián del Zoológico.** Recuento: 5; Trabajadores de Apache Kafka Connect: Recuento: 2; Registro de esquema: Recuento: 2; Proxy REST: Recuento: 2; ksqlDB: Recuento: 2; Centro de control de Confluent: Recuento: 1.

Utilice el modo inverso para equipos de plataforma que no tengan un caso de uso en mente. Utilice el modo de particiones para calcular cuántas particiones requiere un solo tema. Ver https://eventsizer.io para dimensionar en función de los modos inverso y de particiones.

Conclusión

Este documento proporciona pautas recomendadas para usar Confluent Tiered Storage con almacenamiento NetApp , incluidas pruebas de verificación, resultados de rendimiento del almacenamiento en niveles, ajustes, conectores Confluent S3 y la función de autoequilibrio. Teniendo en cuenta las políticas de ILM, el rendimiento de Confluent con múltiples pruebas de rendimiento para verificación y las API S3 estándares de la industria, el almacenamiento de objetos NetApp StorageGRID es una opción óptima para el almacenamiento en niveles de Confluent.

Dónde encontrar información adicional

Para obtener más información sobre la información que se describe en este documento, revise los siguientes documentos y/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 del sumidero S3

  "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 la plataforma Confluent

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

- Almacenamiento en niveles de Confluent: mejores prácticas y dimensionamiento

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

- Conector de receptor de Amazon S3 para la plataforma Confluent

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

- El tamaño de Kafka

  "https://eventsizer.io"

- Dimensionamiento de StorageGRID

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

- Casos de uso de Kafka

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

- Clústeres de Kafka autoequilibrados en la plataforma Confluent 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/"

- Ejemplos de clientes de Instaclustr y detalles de sus casos de uso

  https://www.instaclustr.com/blog/netapp-and-pegasystems-open-source-support-package/, https://www.instaclustr.com/wp-content/uploads/Insta_Case_Study_Pegasystems_1_21sep25.pdf

  https://www.instaclustr.com/resources/customer-case-study-pubnub/

  https://www.instaclustr.com/resources/customer-case-study-tesouro/