Melhores práticas para Confluent Kafka

NetApp artificial intelligence solutions

NetApp
February 12, 2026

This PDF was generated from https://docs.netapp.com/pt-br/netapp-solutions-ai/data-analytics/confluent-kafka-introduction.html on February 12, 2026. Always check docs.netapp.com for the latest.

Índice

Melhores práticas para Confluent Kafka

TR-4912: Diretrizes de práticas recomendadas para armazenamento em camadas do Confluent Kafka com NetApp

Karthikeyan Nagalingam, Joseph Kandatilparambil, NetApp Rankesh Kumar, Confluente

O Apache Kafka é uma plataforma de streaming de eventos distribuída pela comunidade, capaz de lidar com trilhões de eventos por dia. Inicialmente concebido como uma fila de mensagens, o Kafka é baseado em uma abstração de um log de confirmação distribuído. Desde que foi criado e disponibilizado de código aberto pelo LinkedIn em 2011, o Kafka evoluiu de uma fila de mensagens para uma plataforma completa de transmissão de eventos. A Confluent fornece a distribuição do Apache Kafka com a Confluent Platform. A plataforma Confluent complementa o Kafka com recursos comunitários e comerciais adicionais projetados para melhorar a experiência de streaming de operadores e desenvolvedores em produção em grande escala.

Este documento descreve as diretrizes de práticas recomendadas para usar o Confluent Tiered Storage em uma oferta de armazenamento de objetos da NetApp, fornecendo o seguinte conteúdo:

- Verificação confluente com armazenamento de objetos NetApp – NetApp StorageGRID
- Testes de desempenho de armazenamento em camadas
- Diretrizes de práticas recomendadas para Confluent em sistemas de armazenamento NetApp

Por que usar o armazenamento em camadas da Confluent?

O Confluent se tornou a plataforma de streaming em tempo real padrão para muitas aplicações, especialmente para big data, análise e cargas de trabalho de streaming. O armazenamento em camadas permite que os usuários separem a computação do armazenamento na plataforma Confluent. Ele torna o armazenamento de dados mais econômico, permite que você armazene quantidades virtualmente infinitas de dados e aumente (ou diminua) as cargas de trabalho sob demanda, além de facilitar tarefas administrativas como rebalanceamento de dados e locatários. Os sistemas de armazenamento compatíveis com S3 podem aproveitar todos esses recursos para democratizar dados com todos os eventos em um só lugar, eliminando a necessidade de engenharia de dados complexa. Para obter mais informações sobre por que você deve usar armazenamento em camadas para Kafka, verifique"este artigo da Confluent" .

O NetApp instaclustr também oferece suporte ao Kafka com armazenamento em camadas a partir da versão 3.8.1. Confira mais detalhes aqui. "Instaclust usando armazenamento em camadas Kafka"

Por que usar o NetApp StorageGRID para armazenamento em camadas?

StorageGRID é uma plataforma de armazenamento de objetos líder do setor da NetApp. O StorageGRID é uma solução de armazenamento baseada em objetos e definida por software que oferece suporte a APIs de objetos padrão do setor, incluindo a API do Amazon Simple Storage Service (S3). O StorageGRID armazena e gerencia dados não estruturados em escala para fornecer armazenamento de objetos seguro e durável. O conteúdo é colocado no local certo, na hora certa e no nível de armazenamento certo, otimizando fluxos de trabalho e reduzindo custos para mídia avançada distribuída globalmente.

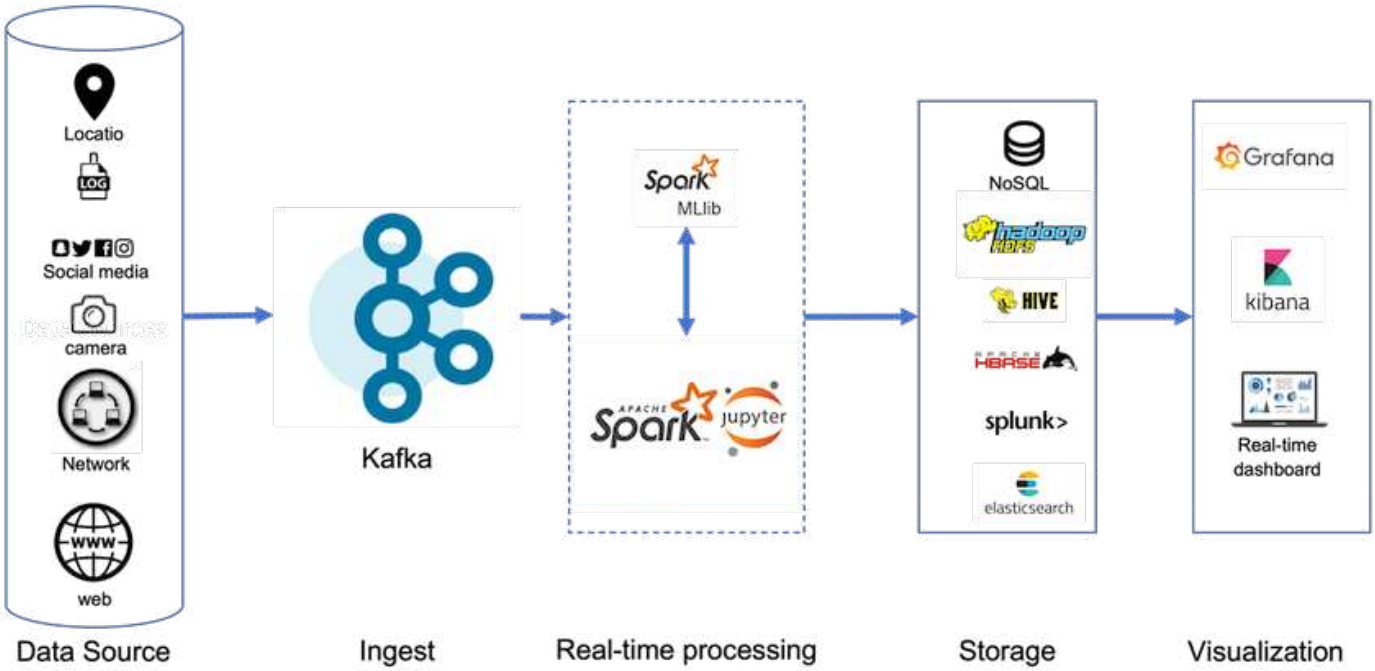
O maior diferencial do StorageGRID é seu mecanismo de política de gerenciamento do ciclo de vida das informações (ILM), que permite o gerenciamento do ciclo de vida dos dados orientado por políticas. O mecanismo de política pode usar metadados para gerenciar como os dados são armazenados ao longo de sua vida útil para otimizar inicialmente o desempenho e otimizar automaticamente o custo e a durabilidade à medida que os dados envelhecem.

Habilitando o armazenamento em camadas do Confluent

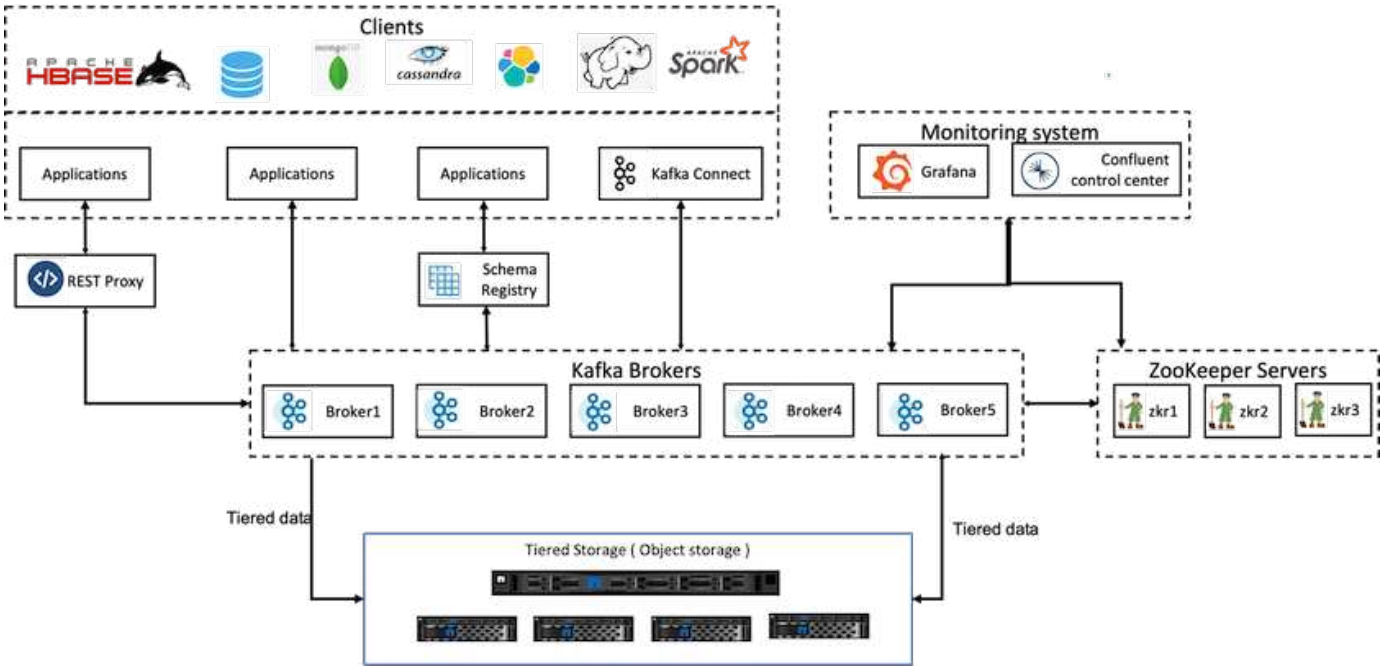
A ideia básica do armazenamento em camadas é separar as tarefas de armazenamento de dados do processamento de dados. Com essa separação, fica muito mais fácil para a camada de armazenamento de dados e a camada de processamento de dados escalarem de forma independente.

Uma solução de armazenamento em camadas para o Confluent deve atender a dois fatores. Primeiro, ele deve contornar ou evitar propriedades comuns de consistência e disponibilidade de armazenamento de objetos, como inconsistências em operações LIST e indisponibilidade ocasional de objetos. Em segundo lugar, ele deve lidar corretamente com a interação entre o armazenamento em camadas e o modelo de replicação e tolerância a falhas do Kafka, incluindo a possibilidade de líderes zumbis continuarem a estratificar intervalos de deslocamento. O armazenamento de objetos da NetApp fornece disponibilidade consistente de objetos e o modelo de alta disponibilidade torna o armazenamento desgastado disponível para intervalos de deslocamento de camadas. O armazenamento de objetos da NetApp fornece disponibilidade consistente de objetos e um modelo de alta disponibilidade para disponibilizar o armazenamento desgastado para intervalos de deslocamento de camadas.

Com o armazenamento em camadas, você pode usar plataformas de alto desempenho para leituras e gravações de baixa latência perto do final dos seus dados de streaming e também pode usar armazenamentos de objetos mais baratos e escaláveis, como o NetApp StorageGRID , para leituras históricas de alto rendimento. Também temos uma solução técnica para Spark com controlador de armazenamento netapp e os detalhes estão aqui. A figura a seguir mostra como o Kafka se encaixa em um pipeline de análise em tempo real.



A figura a seguir mostra como o NetApp StorageGRID se encaixa na camada de armazenamento de objetos do Confluent Kafka.



Detalhes da arquitetura da solução

Esta seção aborda o hardware e o software usados para verificação do Confluent. Estas informações são aplicáveis à implantação da Confluent Platform com armazenamento NetApp . A tabela a seguir abrange a arquitetura da solução testada e os componentes básicos.

Componentes da solução	Detalhes
Confluent Kafka versão 6.2	• Três tratadores de zoológico • Cinco servidores de corretores • Cinco ferramentas de servidores • Uma Grafana • Um centro de controle
Linux (ubuntu 18.04)	Todos os servidores
NetApp StorageGRID para armazenamento em camadas	• Software StorageGRID • 1 x SG1000 (balanceador de carga) • 4 x SGF6024 • 4 x 24 x 800 SSDs • Protocolo S3 • 4 x 100GbE (conectividade de rede entre o broker e as instâncias do StorageGRID)
15 servidores Fujitsu PRIMERGY RX2540	Cada um equipado com: * 2 CPUs, 16 núcleos físicos no total * Intel Xeon * 256 GB de memória física * Porta dupla de 100 GbE

Visão geral da tecnologia

Esta seção descreve a tecnologia usada nesta solução.

NetApp StorageGRID

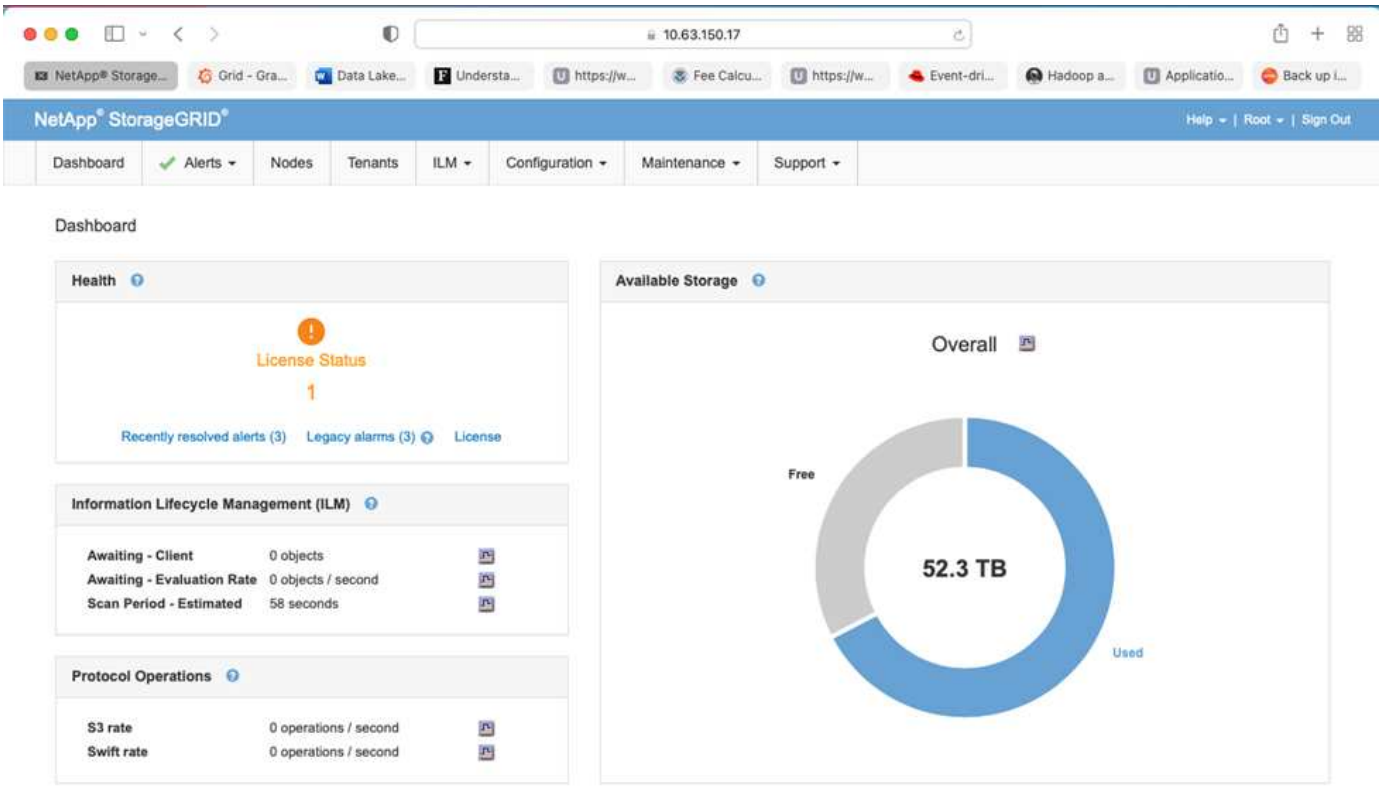
O NetApp StorageGRID é uma plataforma de armazenamento de objetos de alto desempenho e econômica. Ao usar o armazenamento em camadas, a maioria dos dados no Confluent Kafka, que são armazenados no armazenamento local ou no armazenamento SAN do broker, são descarregados para o armazenamento de objetos remoto. Essa configuração resulta em melhorias operacionais significativas ao reduzir o tempo e o custo de rebalanceamento, expansão ou redução de clusters ou substituição de um broker com falha. O armazenamento de objetos desempenha um papel importante no gerenciamento de dados que residem na camada de armazenamento de objetos, e é por isso que escolher o armazenamento de objetos correto é importante.

O StorageGRID oferece gerenciamento de dados globais inteligente e orientado por políticas usando uma arquitetura de grade distribuída baseada em nós. Ele simplifica o gerenciamento de petabytes de dados não estruturados e bilhões de objetos por meio de seu onipresente namespace de objetos globais combinado com recursos sofisticados de gerenciamento de dados. O acesso a objetos de chamada única se estende por todos os sites e simplifica arquiteturas de alta disponibilidade, ao mesmo tempo em que garante acesso contínuo a objetos, independentemente de interrupções no site ou na infraestrutura.

A multilocação permite que vários aplicativos de dados corporativos e de nuvem não estruturados sejam atendidos com segurança na mesma grade, aumentando o ROI e os casos de uso do NetApp StorageGRID. Você pode criar vários níveis de serviço com políticas de ciclo de vida de objetos orientadas por metadados, otimizando durabilidade, proteção, desempenho e localidade em várias regiões geográficas. Os usuários podem ajustar as políticas de gerenciamento de dados, além de monitorar e aplicar limites de tráfego para se realinhar com o cenário de dados sem interrupções, conforme seus requisitos mudam em ambientes de TI em constante mudança.

Gerenciamento simples com Grid Manager

O StorageGRID Grid Manager é uma interface gráfica baseada em navegador que permite configurar, gerenciar e monitorar seu sistema StorageGRID em locais distribuídos globalmente em um único painel.



Você pode executar as seguintes tarefas com a interface do StorageGRID Grid Manager:

- Gerencie repositórios de objetos, como imagens, vídeos e registros, distribuídos globalmente e em escala de petabytes.
- Monitore nós e serviços de grade para garantir a disponibilidade dos objetos.
- Gerencie o posicionamento de dados de objetos ao longo do tempo usando regras de gerenciamento do ciclo de vida das informações (ILM). Essas regras controlam o que acontece com os dados de um objeto depois que eles são ingeridos, como eles são protegidos contra perdas, onde os dados do objeto são armazenados e por quanto tempo.
- Monitore transações, desempenho e operações dentro do sistema.

Políticas de Gestão do Ciclo de Vida da Informação

O StorageGRID tem políticas flexíveis de gerenciamento de dados que incluem manter cópias de réplicas dos seus objetos e usar esquemas de EC (codificação de eliminação) como 2+1 e 4+2 (entre outros) para armazenar seus objetos, dependendo de requisitos específicos de desempenho e proteção de dados. Como as cargas de trabalho e os requisitos mudam ao longo do tempo, é comum que as políticas de ILM também mudem. Modificar políticas de ILM é um recurso essencial, permitindo que os clientes do StorageGRID se adaptem ao seu ambiente em constante mudança de forma rápida e fácil.

Desempenho

O StorageGRID dimensiona o desempenho adicionando mais nós de armazenamento, que podem ser VMs, bare metal ou dispositivos desenvolvidos para esse fim, como o"SG5712, SG5760, SG6060 ou SGF6024" . Em nossos testes, superamos os principais requisitos de desempenho do Apache Kafka com uma grade de três nós de tamanho mínimo usando o dispositivo SGF6024. À medida que os clientes escalam seu cluster Kafka com corretores adicionais, eles podem adicionar mais nós de armazenamento para aumentar o desempenho e a capacidade.

Configuração do balanceador de carga e do endpoint

Os nós de administração no StorageGRID fornecem a interface de usuário (UI) do Grid Manager e o endpoint da API REST para visualizar, configurar e gerenciar seu sistema StorageGRID , bem como logs de auditoria para rastrear a atividade do sistema. Para fornecer um ponto de extremidade S3 de alta disponibilidade para o armazenamento em camadas do Confluent Kafka, implementamos o balanceador de carga StorageGRID , que é executado como um serviço em nós de administração e nós de gateway. Além disso, o balanceador de carga também gerencia o tráfego local e se comunica com o GSLB (Global Server Load Balancing) para ajudar na recuperação de desastres.

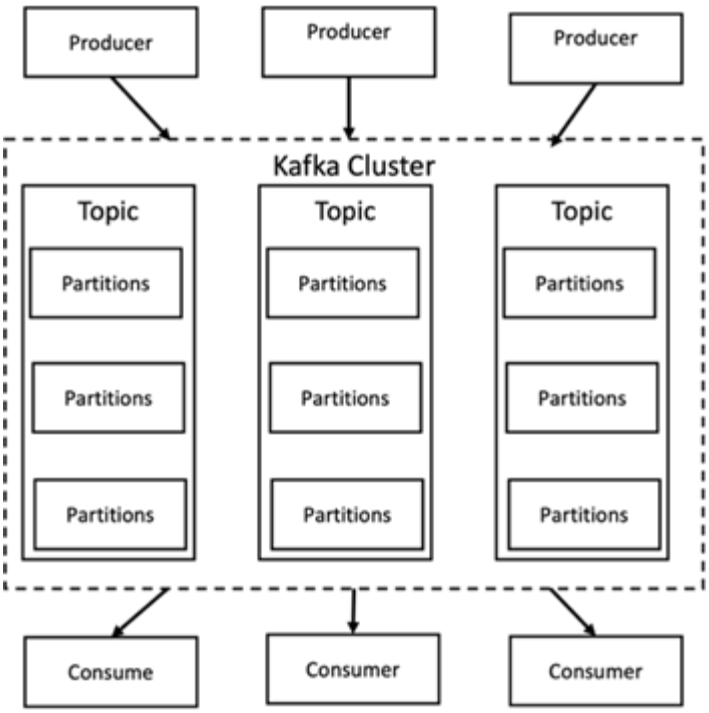
Para aprimorar ainda mais a configuração do endpoint, o StorageGRID fornece políticas de classificação de tráfego incorporadas ao nó de administração, permite monitorar o tráfego da carga de trabalho e aplica vários limites de qualidade de serviço (QoS) às suas cargas de trabalho. As políticas de classificação de tráfego são aplicadas aos endpoints no serviço StorageGRID Load Balancer para nós de gateway e nós de administração. Essas políticas podem ajudar na modelagem e monitoramento do tráfego.

Classificação de tráfego no StorageGRID

O StorageGRID tem funcionalidade de QoS integrada. As políticas de classificação de tráfego podem ajudar a monitorar diferentes tipos de tráfego S3 provenientes de um aplicativo cliente. Você pode então criar e aplicar políticas para colocar limites nesse tráfego com base na largura de banda de entrada/saída, no número de solicitações simultâneas de leitura/gravação ou na taxa de solicitações de leitura/gravação.

Apache Kafka

Apache Kafka é uma implementação de framework de um barramento de software que utiliza processamento de fluxo escrito em Java e Scala. O objetivo é fornecer uma plataforma unificada, de alto rendimento e baixa latência para lidar com feeds de dados em tempo real. O Kafka pode se conectar a um sistema externo para exportação e importação de dados por meio do Kafka Connect e fornece fluxos do Kafka, uma biblioteca de processamento de fluxo Java. O Kafka usa um protocolo binário baseado em TCP que é otimizado para eficiência e depende de uma abstração de "conjunto de mensagens" que agrupa mensagens naturalmente para reduzir a sobrecarga da viagem de ida e volta da rede. Isso permite operações de disco sequenciais maiores, pacotes de rede maiores e blocos de memória contíguos, permitindo assim que o Kafka transforme um fluxo contínuo de gravações de mensagens aleatórias em gravações lineares. A figura a seguir descreve o fluxo de dados básico do Apache Kafka.



O Kafka armazena mensagens de valor-chave que vêm de um número arbitrário de processos chamados produtores. Os dados podem ser particionados em diferentes partições dentro de diferentes tópicos. Dentro de uma partição, as mensagens são estritamente ordenadas por seus deslocamentos (a posição de uma mensagem dentro de uma partição) e indexadas e armazenadas junto com um registro de data e hora. Outros processos chamados consumidores podem ler mensagens de partições. Para processamento de fluxo, o Kafka oferece a API Streams, que permite escrever aplicativos Java que consomem dados do Kafka e gravam os resultados de volta no Kafka. O Apache Kafka também funciona com sistemas de processamento de fluxo externo, como Apache Apex, Apache Flink, Apache Spark, Apache Storm e Apache NiFi.

O Kafka é executado em um cluster de um ou mais servidores (chamados brokers), e as partições de todos os tópicos são distribuídas entre os nós do cluster. Além disso, as partições são replicadas para vários corretores. Essa arquitetura permite que o Kafka entregue grandes fluxos de mensagens de forma tolerante a falhas e permitiu que ele substituísse alguns dos sistemas de mensagens convencionais, como Java Message Service (JMS), Advanced Message Queuing Protocol (AMQP) e assim por diante. Desde a versão 0.11.0.0, o Kafka oferece gravações transacionais, que fornecem exatamente um processamento de fluxo usando a API Streams.

O Kafka suporta dois tipos de tópicos: regulares e compactados. Tópicos regulares podem ser configurados com um tempo de retenção ou um limite de espaço. Se houver registros mais antigos que o tempo de retenção especificado ou se o limite de espaço for excedido para uma partição, o Kafka poderá excluir dados antigos para liberar espaço de armazenamento. Por padrão, os tópicos são configurados com um tempo de retenção de 7 dias, mas também é possível armazenar dados indefinidamente. Para tópicos compactados, os registros não expiram com base em limites de tempo ou espaço. Em vez disso, o Kafka trata mensagens posteriores como atualizações de mensagens mais antigas com a mesma chave e garante nunca excluir a mensagem mais recente por chave. Os usuários podem excluir mensagens completamente escrevendo uma mensagem de exclusão com o valor nulo para uma chave específica.

Existem cinco APIs principais no Kafka:

- **API do produtor.** Permite que um aplicativo publique fluxos de registros.
- **API do consumidor.** Permite que um aplicativo assine tópicos e processe fluxos de registros.
- **API do conector.** Executa as APIs reutilizáveis de produtor e consumidor que podem vincular os tópicos

aos aplicativos existentes.

- **API de fluxos.** Esta API converte os fluxos de entrada em saída e produz o resultado.
- **API de administração.** Usado para gerenciar tópicos, corretores e outros objetos do Kafka.

As APIs de consumidor e produtor são baseadas no protocolo de mensagens Kafka e oferecem uma implementação de referência para clientes consumidores e produtores Kafka em Java. O protocolo de mensagens subjacente é um protocolo binário que os desenvolvedores podem usar para escrever seus próprios clientes consumidores ou produtores em qualquer linguagem de programação. Isso desbloqueia o Kafka do ecossistema da Máquina Virtual Java (JVM). Uma lista de clientes não Java disponíveis é mantida no wiki do Apache Kafka.

Casos de uso do Apache Kafka

O Apache Kafka é mais popular para mensagens, rastreamento de atividades de sites, métricas, agregação de logs, processamento de fluxo, fornecimento de eventos e registro de confirmações.

- O Kafka tem melhor produtividade, particionamento integrado, replicação e tolerância a falhas, o que o torna uma boa solução para aplicativos de processamento de mensagens em larga escala.
- O Kafka pode reconstruir as atividades de um usuário (visualizações de páginas, pesquisas) em um pipeline de rastreamento como um conjunto de feeds de publicação e assinatura em tempo real.
- O Kafka é frequentemente usado para dados de monitoramento operacional. Isso envolve agregar estatísticas de aplicativos distribuídos para produzir feeds centralizados de dados operacionais.
- Muitas pessoas usam o Kafka como um substituto para uma solução de agregação de logs. A agregação de log normalmente coleta arquivos de log físicos de servidores e os coloca em um local central (por exemplo, um servidor de arquivos ou HDFS) para processamento. O Kafka abstrai detalhes de arquivos e fornece uma abstração mais limpa de dados de log ou evento como um fluxo de mensagens. Isso permite um processamento de menor latência e suporte mais fácil para múltiplas fontes de dados e consumo de dados distribuídos.
- Muitos usuários do Kafka processam dados em pipelines de processamento que consistem em vários estágios, nos quais dados de entrada brutos são consumidos de tópicos do Kafka e então agregados, enriquecidos ou transformados em novos tópicos para consumo posterior ou processamento de acompanhamento. Por exemplo, um pipeline de processamento para recomendar artigos de notícias pode rastrear o conteúdo do artigo de feeds RSS e publicá-lo em um tópico "artigos". O processamento posterior pode normalizar ou desduplicar esse conteúdo e publicar o conteúdo do artigo limpo em um novo tópico, e um estágio de processamento final pode tentar recomendar esse conteúdo aos usuários. Esses pipelines de processamento criam gráficos de fluxos de dados em tempo real com base em tópicos individuais.
- O sourcing de eventos é um estilo de design de aplicativo no qual as alterações de estado são registradas como uma sequência de registros ordenada por tempo. O suporte do Kafka para grandes volumes de dados de log armazenados o torna um excelente backend para um aplicativo criado nesse estilo.
- O Kafka pode servir como um tipo de log de confirmação externo para um sistema distribuído. O log ajuda a replicar dados entre nós e atua como um mecanismo de ressincronização para nós com falha restaurarem seus dados. O recurso de compactação de log no Kafka ajuda a dar suporte a esse caso de uso.

Confluente

A Confluent Platform é uma plataforma pronta para empresas que complementa o Kafka com recursos avançados projetados para ajudar a acelerar o desenvolvimento e a conectividade de aplicativos, permitir transformações por meio do processamento de fluxo, simplificar as operações empresariais em escala e atender a requisitos arquitetônicos rigorosos. Desenvolvido pelos criadores originais do Apache Kafka, o

Confluent expande os benefícios do Kafka com recursos de nível empresarial, ao mesmo tempo em que elimina o fardo do gerenciamento ou monitoramento do Kafka. Hoje, mais de 80% das empresas da Fortune 100 são alimentadas por tecnologia de streaming de dados, e a maioria delas usa Confluent.

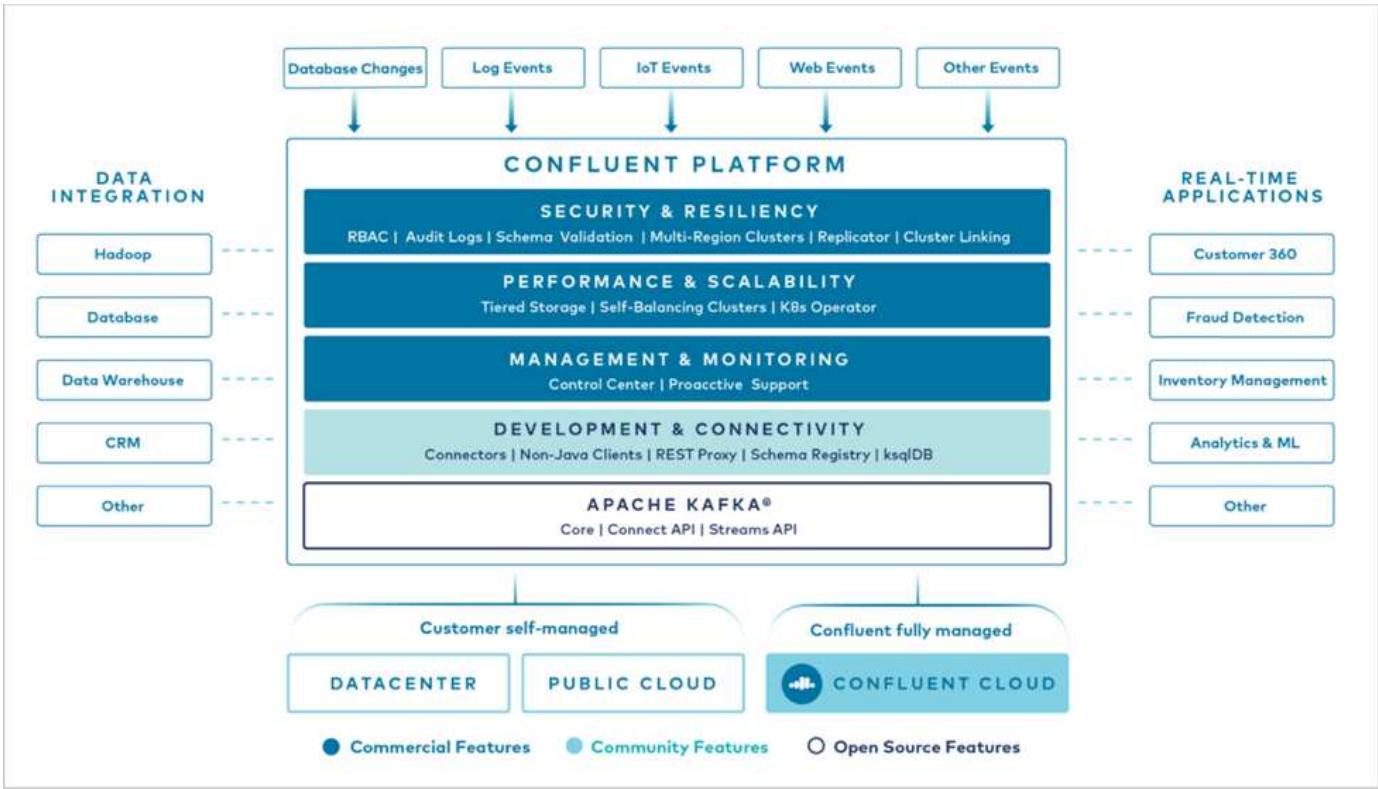
Por que Confluent?

Ao integrar dados históricos e em tempo real em uma única fonte central de verdade, a Confluent facilita a criação de uma categoria inteiramente nova de aplicativos modernos e orientados a eventos, obtém um pipeline de dados universal e desbloqueia novos e poderosos casos de uso com total escalabilidade, desempenho e confiabilidade.

Para que é usado o Confluent?

A Confluent Platform permite que você se concentre em como extrair valor comercial dos seus dados em vez de se preocupar com a mecânica subjacente, como a forma como os dados estão sendo transportados ou integrados entre sistemas distintos. Especificamente, a Confluent Platform simplifica a conexão de fontes de dados ao Kafka, a criação de aplicativos de streaming, bem como a proteção, o monitoramento e o gerenciamento da sua infraestrutura Kafka. Hoje, a Confluent Platform é usada para uma ampla gama de casos de uso em vários setores, desde serviços financeiros, varejo omnicanal e carros autônomos até detecção de fraudes, microsserviços e IoT.

A figura a seguir mostra os componentes da plataforma Confluent Kafka.



Visão geral da tecnologia de streaming de eventos da Confluent

No centro da Plataforma Confluent está "Apache Kafka" , a plataforma de streaming distribuída de código aberto mais popular. Os principais recursos do Kafka são os seguintes:

- Publique e assine fluxos de registros.
- Armazene fluxos de registros de forma tolerante a falhas.

- Processar fluxos de registros.

Pronto para uso, o Confluent Platform também inclui Schema Registry, REST Proxy, mais de 100 conectores Kafka pré-criados e ksqlDB.

Visão geral dos recursos empresariais da plataforma Confluent

- **Centro de Controle Confluente.** Um sistema baseado em GUI para gerenciar e monitorar o Kafka. Ele permite que você gerencie facilmente o Kafka Connect e crie, edite e gerencie conexões com outros sistemas.
- **Confluent para Kubernetes.** Confluent for Kubernetes é um operador do Kubernetes. Os operadores do Kubernetes estendem os recursos de orquestração do Kubernetes, fornecendo recursos e requisitos exclusivos para um aplicativo de plataforma específico. Para a Confluent Platform, isso inclui simplificar bastante o processo de implantação do Kafka no Kubernetes e automatizar tarefas típicas do ciclo de vida da infraestrutura.
- **Conectores confluentes para Kafka.** Os conectores usam a API do Kafka Connect para conectar o Kafka a outros sistemas, como bancos de dados, armazenamentos de chave-valor, índices de pesquisa e sistemas de arquivos. O Confluent Hub tem conectores para download para as fontes e coletores de dados mais populares, incluindo versões totalmente testadas e suportadas desses conectores com a Confluent Platform. Mais detalhes podem ser encontrados "aqui" .
- **Aglomerados autobalanceados.** Fornece balanceamento de carga automatizado, detecção de falhas e autocorreção. Ele fornece suporte para adicionar ou desativar corretores conforme necessário, sem ajuste manual.
- **Ligação de cluster confluente.** Conecta clusters diretamente e espelha tópicos de um cluster para outro por meio de uma ponte de link. A vinculação de clusters simplifica a configuração de implantações de vários datacenters, vários clusters e nuvens híbridas.
- **Balanceador automático de dados Confluent.** Monitora seu cluster quanto ao número de corretores, ao tamanho das partições, ao número de partições e ao número de líderes dentro do cluster. Ele permite que você transfira dados para criar uma carga de trabalho uniforme em seu cluster, ao mesmo tempo em que reequilibra o tráfego para minimizar o efeito nas cargas de trabalho de produção durante o rebalanceamento.
- **Replicador confluente.** Torna mais fácil do que nunca manter vários clusters Kafka em vários data centers.
- **Armazenamento em camadas.** Oferece opções para armazenar grandes volumes de dados do Kafka usando seu provedor de nuvem favorito, reduzindo assim a carga operacional e os custos. Com o armazenamento em camadas, você pode manter dados em armazenamento de objetos econômico e escalar corretores somente quando precisar de mais recursos de computação.
- **Cliente JMS Confluent.** A Confluent Platform inclui um cliente compatível com JMS para Kafka. Este cliente Kafka implementa a API padrão do JMS 1.1, usando corretores Kafka como backend. Isso é útil se você tiver aplicativos legados usando JMS e quiser substituir o broker de mensagens JMS existente pelo Kafka.
- **Proxy MQTT Confluent.** Fornece uma maneira de publicar dados diretamente no Kafka a partir de dispositivos e gateways MQTT sem a necessidade de um broker MQTT no meio.
- **Plugins de segurança Confluent.** Os plugins de segurança Confluent são usados para adicionar recursos de segurança a várias ferramentas e produtos da plataforma Confluent. Atualmente, há um plugin disponível para o proxy REST do Confluent que ajuda a autenticar as solicitações recebidas e a propagar o principal autenticado para as solicitações ao Kafka. Isso permite que os clientes proxy REST da Confluent utilizem os recursos de segurança multilocatários do broker Kafka.

Verificação confluente

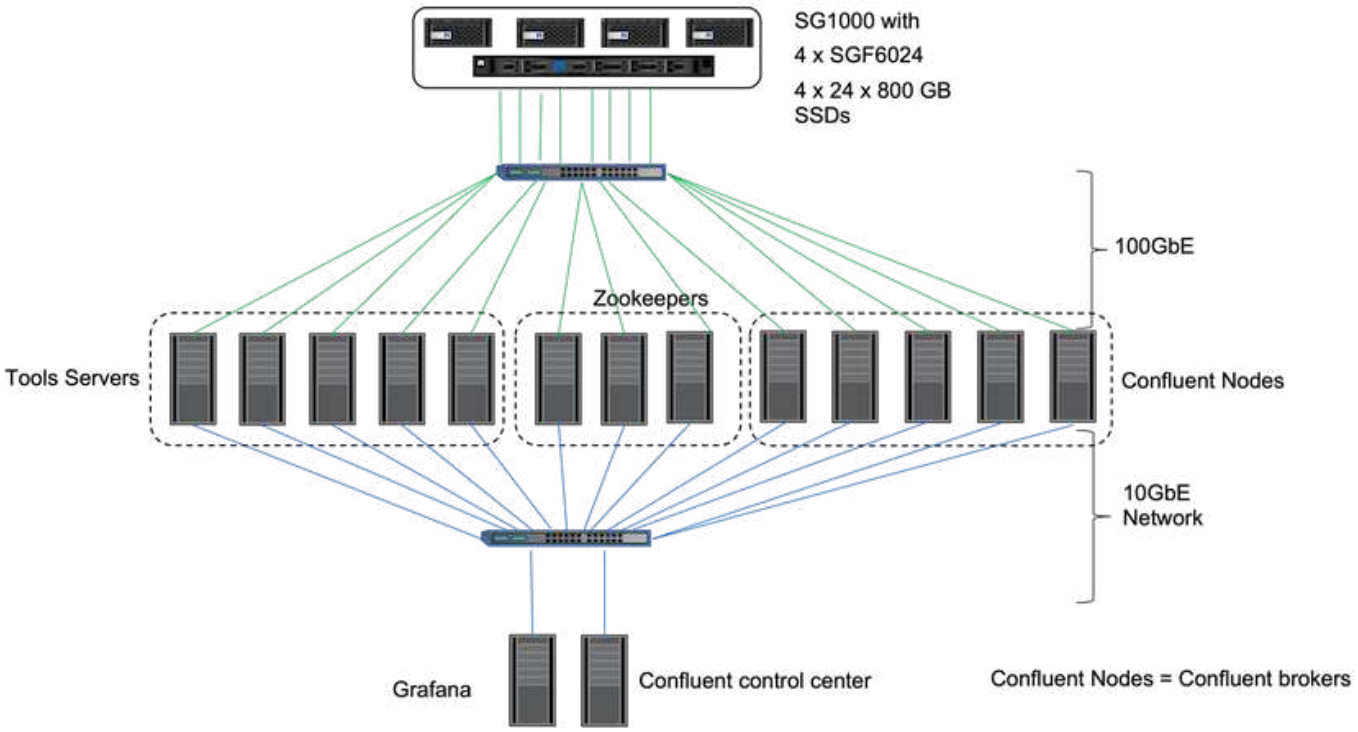
Realizamos a verificação com o Confluent Platform 6.2 Tiered Storage no NetApp StorageGRID. As equipes da NetApp e da Confluent trabalharam juntas nessa verificação e executaram os casos de teste necessários para a verificação.

Configuração da plataforma Confluent

Usamos a seguinte configuração para verificação.

Para verificação, usamos três tratadores, cinco corretores, cinco servidores de execução de scripts de teste, servidores de ferramentas nomeadas com 256 GB de RAM e 16 CPUs. Para armazenamento NetApp , usamos o StorageGRID com um balanceador de carga SG1000 com quatro SGF6024s. O armazenamento e os corretores foram conectados por meio de conexões de 100 GbE.

A figura a seguir mostra a topologia de rede da configuração usada para verificação do Confluent.



Os servidores de ferramentas atuam como clientes de aplicativos que enviam solicitações aos nós do Confluent.

Configuração de armazenamento em camadas confluente

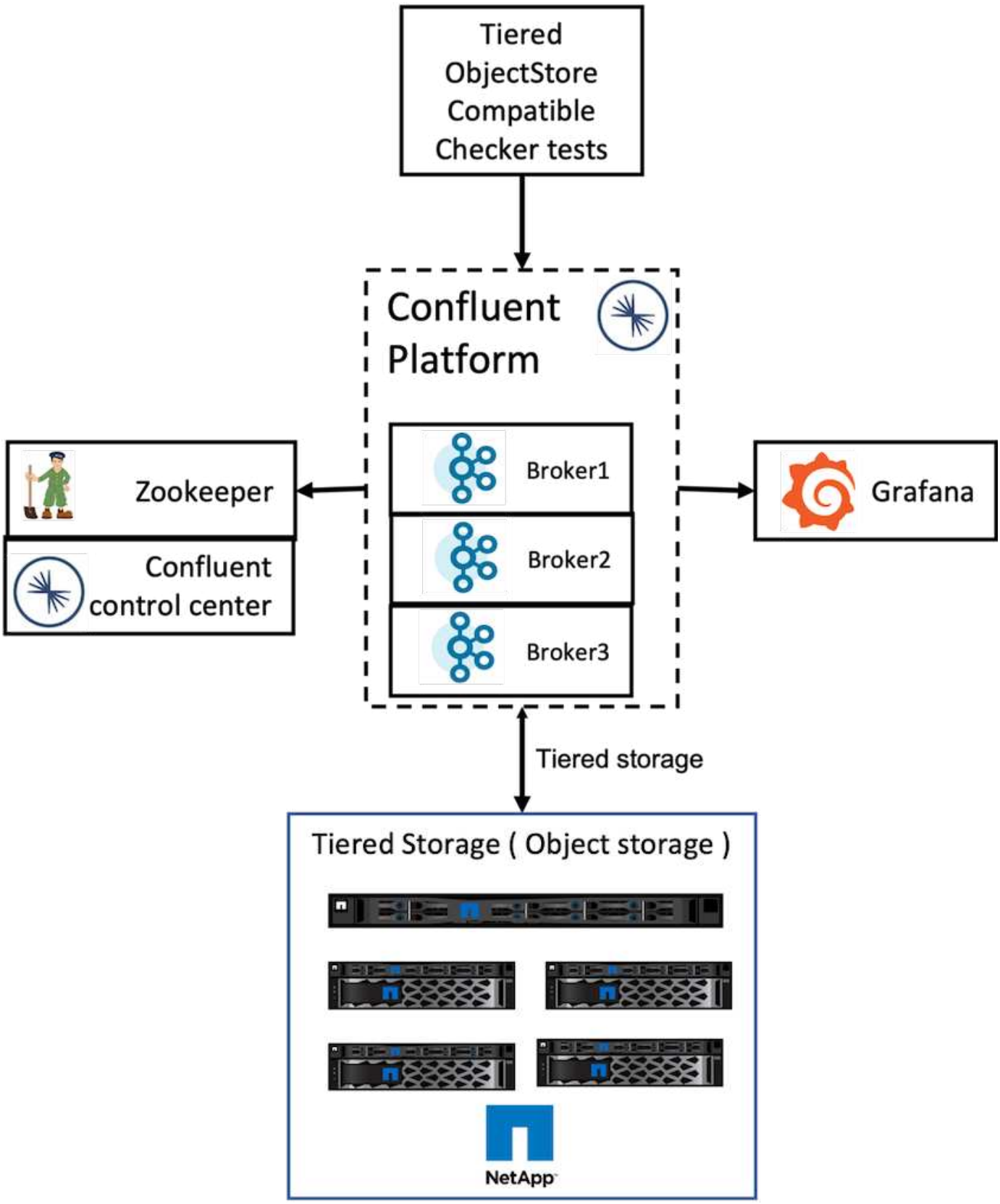
A configuração de armazenamento em camadas requer os seguintes parâmetros no Kafka:

```
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 verificação, usamos StorageGRID com o protocolo HTTP, mas HTTPS também funciona. A chave de acesso e a chave secreta são armazenadas no nome do arquivo fornecido no `confluent.tier.s3.cred.file.path` parâmetro.

Armazenamento de objetos NetApp - StorageGRID

Configuramos a configuração de site único no StorageGRID para verificação.



Testes de verificação

Concluímos os cinco casos de teste a seguir para verificação. Esses testes são executados no framework Trogdor. Os dois primeiros foram testes de funcionalidade e os três restantes foram testes de desempenho.

Teste de correção de armazenamento de objetos

Este teste determina se todas as operações básicas (por exemplo, obter/colocar/excluir) na API de armazenamento de objetos funcionam bem de acordo com as necessidades do armazenamento em camadas. É um teste básico que todo serviço de armazenamento de objetos deve esperar passar antes dos testes seguintes. É um teste assertivo que pode ser aprovado ou reprovado.

Teste de correção da funcionalidade de hierarquização

Este teste determina se a funcionalidade de armazenamento em camadas de ponta a ponta funciona bem com um teste assertivo que passa ou falha. O teste cria um tópico de teste que, por padrão, é configurado com camadas habilitadas e um tamanho de hotset altamente reduzido. Ele produz um fluxo de eventos para o tópico de teste recém-criado, aguarda que os corretores arquivem os segmentos no armazenamento de objetos e, então, consome o fluxo de eventos e valida se o fluxo consumido corresponde ao fluxo produzido. O número de mensagens produzidas no fluxo de eventos é configurável, o que permite ao usuário gerar uma carga de trabalho suficientemente grande de acordo com as necessidades de teste. O tamanho reduzido do hotset garante que as buscas do consumidor fora do segmento ativo sejam atendidas somente pelo armazenamento de objetos; isso ajuda a testar a exatidão do armazenamento de objetos para leituras. Realizamos este teste com e sem uma injeção de falha de armazenamento de objeto. Simulamos uma falha de nó interrompendo o serviço do gerenciador de serviços em um dos nós no StorageGRID e validando se a funcionalidade de ponta a ponta funciona com o armazenamento de objetos.

Benchmark de busca de nível

Este teste validou o desempenho de leitura do armazenamento de objetos em camadas e verificou o intervalo de solicitações de leitura de busca sob carga pesada de segmentos gerados pelo benchmark. Neste benchmark, a Confluent desenvolveu clientes personalizados para atender às solicitações de busca de camadas.

Benchmark de carga de trabalho de produção e consumo

Este teste gerou indiretamente carga de trabalho de gravação no armazenamento de objetos por meio do arquivamento de segmentos. A carga de trabalho de leitura (segmentos lidos) foi gerada a partir do armazenamento de objetos quando grupos de consumidores buscavam os segmentos. Esta carga de trabalho foi gerada pelo script de teste. Este teste verificou o desempenho de leitura e gravação no armazenamento de objetos em threads paralelos. Testamos com e sem injeção de falha de armazenamento de objetos, assim como fizemos para o teste de correção da funcionalidade de camadas.

Referência de carga de trabalho de retenção

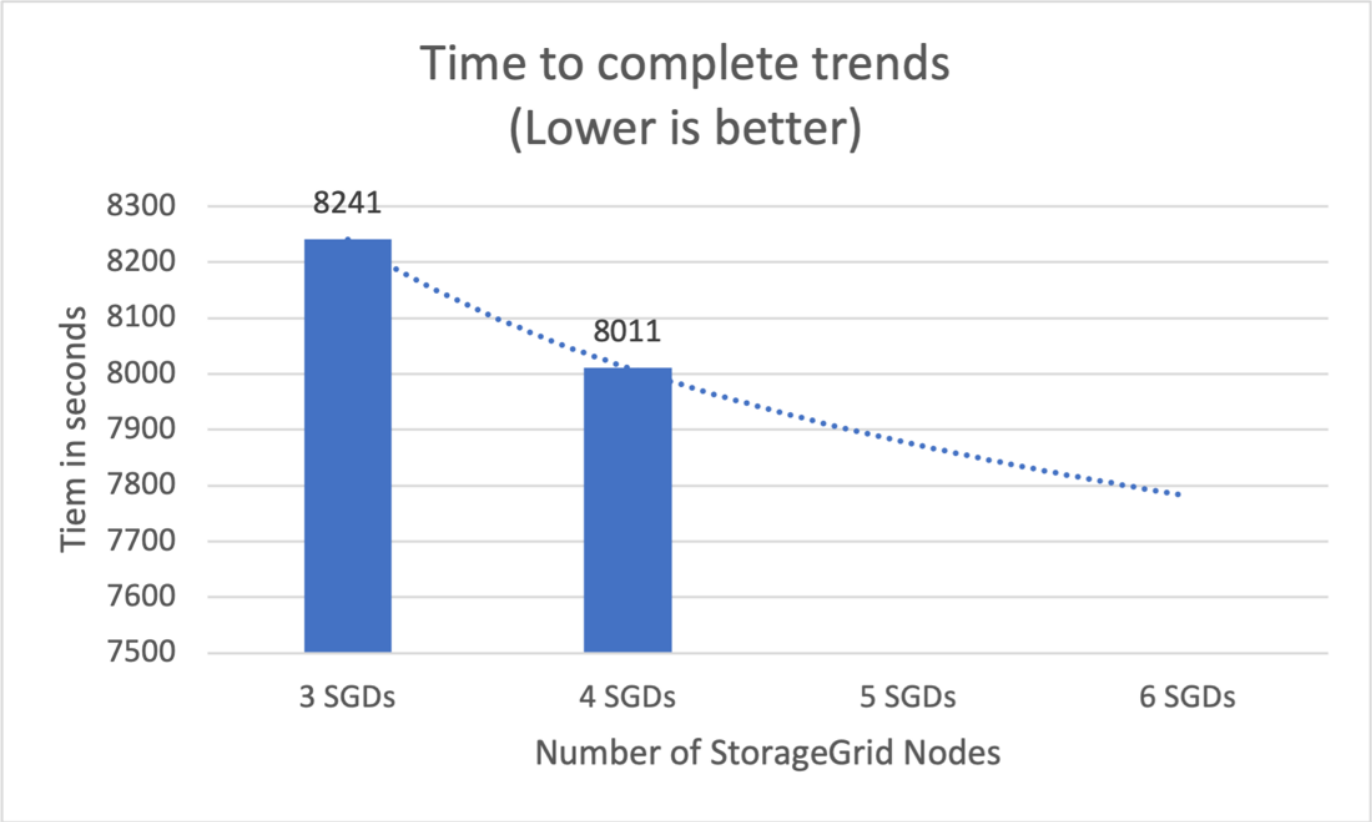
Este teste verificou o desempenho de exclusão de um armazenamento de objetos sob uma carga de trabalho pesada de retenção de tópicos. A carga de trabalho de retenção foi gerada usando um script de teste que produz muitas mensagens em paralelo a um tópico de teste. O tópico de teste foi a configuração com uma configuração agressiva de retenção baseada em tamanho e tempo que fazia com que o fluxo de eventos fosse continuamente eliminado do armazenamento de objetos. Os segmentos foram então arquivados. Isso levou a um grande número de exclusões no armazenamento de objetos pelo broker e à cobrança do desempenho das operações de exclusão do armazenamento de objetos.

Testes de desempenho com escalabilidade

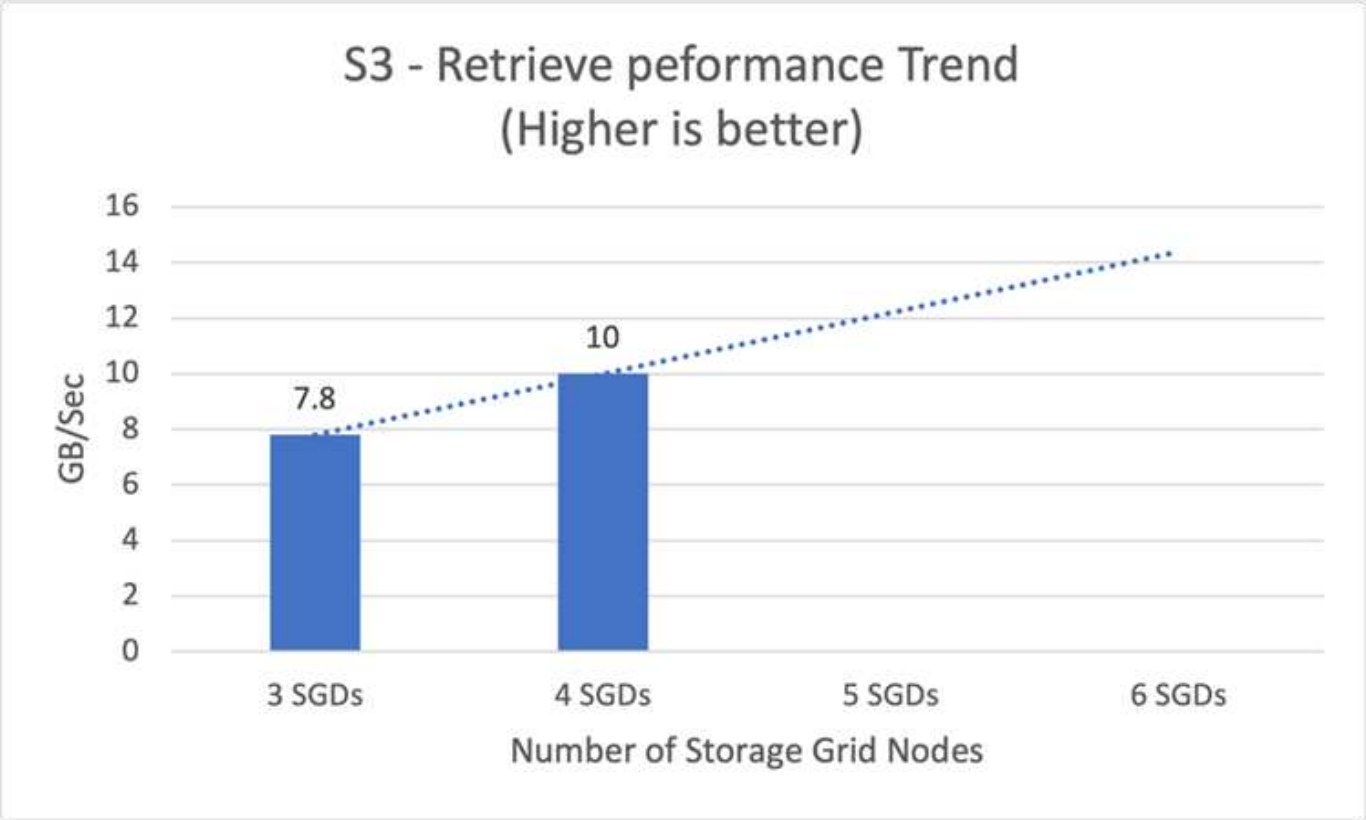
Realizamos testes de armazenamento em camadas com três a quatro nós para cargas de trabalho de produtor e consumidor com a configuração NetApp StorageGRID . De acordo com nossos testes, o tempo de conclusão e os resultados de desempenho foram

diretamente proporcionais ao número de nós StorageGRID . A configuração do StorageGRID exigiu no mínimo três nós.

- O tempo para concluir a operação de produção e consumo diminuiu linearmente quando o número de nós de armazenamento aumentou.



- O desempenho da operação de recuperação s3 aumentou linearmente com base no número de nós StorageGRID . O StorageGRID suporta até 200 nós StorgeGRID.



Conector Confluent s3

O conector Amazon S3 Sink exporta dados de tópicos do Apache Kafka para objetos S3 nos formatos Avro, JSON ou Bytes. O conector de coletor do Amazon S3 pesquisa periodicamente dados do Kafka e, por sua vez, os carrega no S3. Um particionador é usado para dividir os dados de cada partição do Kafka em pedaços. Cada bloco de dados é representado como um objeto S3. O nome da chave codifica o tópico, a partição do Kafka e o deslocamento inicial deste bloco de dados.

Nesta configuração, mostramos como ler e escrever tópicos no armazenamento de objetos do Kafka diretamente usando o conector de coletor S3 do Kafka. Para este teste, usamos um cluster Confluent autônomo, mas esta configuração é aplicável a um cluster distribuído.

1. Baixe o Confluent Kafka do site da Confluent.
2. Descompacte o pacote em uma pasta no seu servidor.
3. Exporte duas variáveis.

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

4. Para uma configuração autônoma do Confluent Kafka, o cluster cria uma pasta raiz temporária em `/tmp` . Ele também cria Zookeeper, Kafka, um registro de esquema, connect, um ksql-server e pastas do centro de controle e copia seus respectivos arquivos de configuração de `$CONFLUENT_HOME` . Veja o exemplo a seguir:

```
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 o Zookeeper. Você não precisa alterar nada se usar os parâmetros padrões.

```
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:~#
```

Na configuração acima, atualizamos o `server. xxx` propriedade. Por padrão, você precisa de três tratadores para a seleção do líder Kafka.

6. Criamos um arquivo myid em `/tmp/confluent.406980/zookeeper/data` com um ID único:

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

Usamos o último número de endereços IP para o arquivo myid. Usamos valores padrão para as configurações Kafka, connect, control-center, Kafka, Kafka-rest, ksql-server e schema-registry.

7. Inicie os serviços do 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#
```

Há uma pasta de log para cada configuração, o que ajuda a solucionar problemas. Em alguns casos, os serviços demoram mais para iniciar. Certifique-se de que todos os serviços estejam funcionando.

8. Instalar o 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#
```

Você também pode instalar uma versão específica usando `confluent-hub install confluentinc/kafka-connect-s3:10.0.3`.

9. Por padrão, `confluentinc-kafka-connect-s3` está instalado em `/data/confluent/confluent-6.2.0/share/confluent-hub-components/confluentinc-kafka-connect-s3`.
10. Atualize o caminho do plug-in com o novo `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. Pare os serviços do Confluent e reinicie-os.

```
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 o ID de acesso e a chave secreta no `/root/.aws/credentials` arquivo.

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

13. Verifique se o balde está acessível.

```
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 o arquivo de propriedades s3-sink para configuração do s3 e do 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. Importe alguns registros para o 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. Carregue o 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 o status do s3-sink.

```
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 o log para ter certeza de que o s3-sink está pronto para aceitar tópicos.

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

19. Confira os tópicos em 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 os objetos no 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 o conteúdo, copie cada arquivo do S3 para o seu sistema de arquivos local executando o seguinte 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 os registros, use avro-tools-1.11.0.1.jar (disponível no "Arquivos 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 Instaclustr Kafka Connect

A Instaclustr oferece suporte a conectores Kafka Connect e seus detalhes. "Mais detalhes". A Instaclustr fornece conectores adicionais. "seus detalhes"

Aglomerados autobalanceados confluentes

Se você já gerenciou um cluster Kafka antes, provavelmente está familiarizado com os desafios que surgem ao reatribuir manualmente partições a diferentes brokers para garantir que a carga de trabalho seja balanceada no cluster. Para organizações com grandes implantações do Kafka, reorganizar grandes quantidades de dados pode ser assustador, tedioso e arriscado, especialmente se aplicativos de missão crítica forem criados no cluster. Entretanto, mesmo para os menores casos de uso do Kafka, o processo é demorado e propenso a erros humanos.

Em nosso laboratório, testamos o recurso de clusters de autobalanceamento do Confluent, que automatiza o rebalanceamento com base em alterações na topologia do cluster ou carga irregular. O teste de rebalanceamento do Confluent ajuda a medir o tempo para adicionar um novo broker quando ocorre uma falha no nó ou o nó de dimensionamento exige o rebalanceamento de dados entre os brokers. Em configurações clássicas do Kafka, a quantidade de dados a serem rebalanceados aumenta à medida que o cluster cresce, mas, no armazenamento em camadas, o rebalanceamento é restrito a uma pequena quantidade de dados. Com base em nossa validação, o rebalanceamento no armazenamento em camadas leva segundos ou minutos em uma arquitetura Kafka clássica e cresce linearmente conforme o cluster cresce.

Em clusters de autobalanceamento, os rebalanceamentos de partições são totalmente automatizados para otimizar a taxa de transferência do Kafka, acelerar o dimensionamento do broker e reduzir a carga operacional de execução de um cluster grande. Em estado estável, os clusters autobalanceados monitoram a distorção de dados entre os corretores e reatribuem partições continuamente para otimizar o desempenho do cluster. Ao dimensionar a plataforma para cima ou para baixo, os clusters de autobalanceamento reconhecem automaticamente a presença de novos corretores ou a remoção de corretores antigos e acionam uma reatribuição de partição subsequente. Isso permite que você adicione e desative corretores facilmente, tornando seus clusters Kafka fundamentalmente mais elásticos. Esses benefícios não exigem intervenção manual, cálculos matemáticos complexos ou o risco de erro humano que as reatribuições de partições normalmente envolvem. Como resultado, os rebalanceamentos de dados são concluídos em muito menos tempo, e você fica livre para se concentrar em projetos de streaming de eventos de maior valor, em vez de precisar supervisionar constantemente seus clusters.

A Instaclustr também oferece recursos de rebalanceamento automático e já foi implementada para diversos clientes.

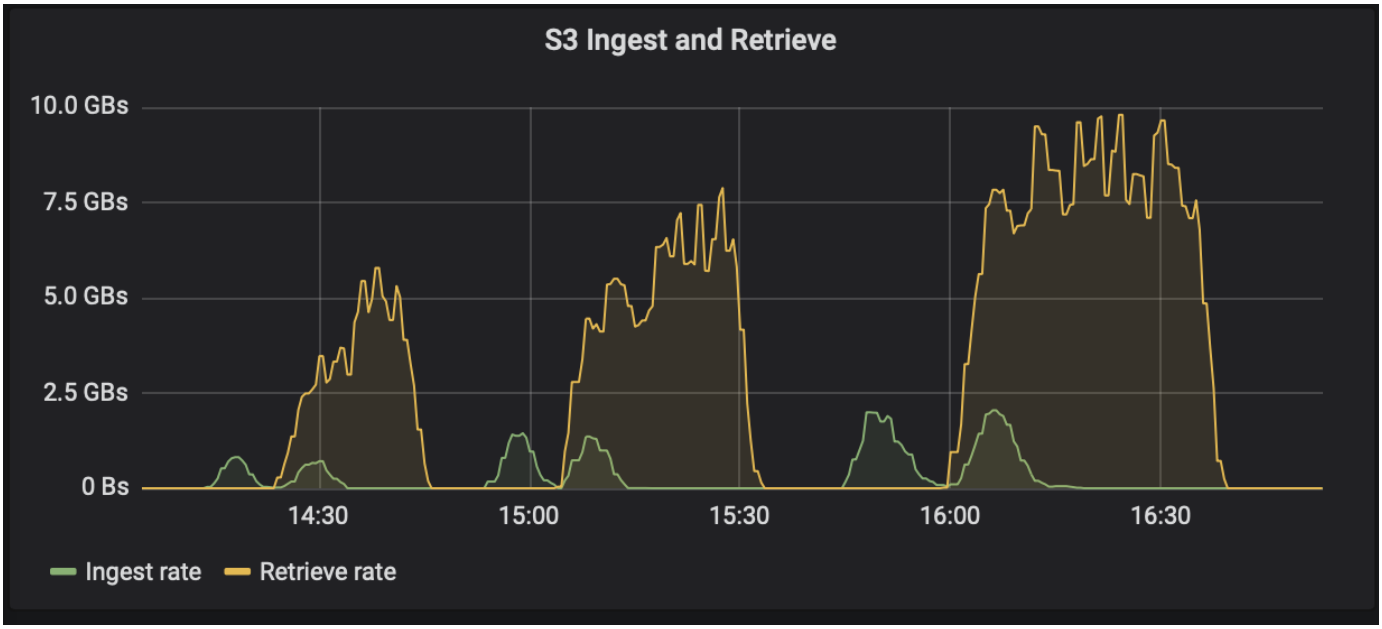
Diretrizes de melhores práticas

Esta seção apresenta lições aprendidas com esta certificação.

- Com base em nossa validação, o armazenamento de objetos S3 é melhor para o Confluent manter dados.
- Podemos usar SAN de alto rendimento (especificamente FC) para manter os dados ativos do broker ou o disco local, porque, na configuração de armazenamento em camadas do Confluent, o tamanho dos dados mantidos no diretório de dados do broker é baseado no tamanho do segmento e no tempo de retenção quando os dados são movidos para o armazenamento de objetos.
- Os armazenamentos de objetos fornecem melhor desempenho quando segment.bytes é maior; testamos 512 MB.

- No Kafka, o comprimento da chave ou valor (em bytes) para cada registro produzido no tópico é controlado pelo `length.key.value` parâmetro. Para StorageGRID, o desempenho de ingestão e recuperação de objetos S3 aumentou para valores mais altos. Por exemplo, 512 bytes forneceram uma recuperação de 5,8 GBps, 1.024 bytes forneceram uma recuperação s3 de 7,5 GBps e 2.048 bytes forneceram perto de 10 GBps.

A figura a seguir apresenta a ingestão e recuperação de objetos S3 com base em `length.key.value` .



- **Afinação de Kafka.** Para melhorar o desempenho do armazenamento em camadas, você pode aumentar TierFetcherNumThreads e TierArchiverNumThreads. Como orientação geral, você deve aumentar TierFetcherNumThreads para corresponder ao número de núcleos físicos da CPU e aumentar TierArchiverNumThreads para metade do número de núcleos da CPU. Por exemplo, nas propriedades do servidor, se você tiver uma máquina com oito núcleos físicos, defina confluent.tier.fetcher.num.threads = 8 e confluent.tier.archiver.num.threads = 4.
- **Intervalo de tempo para exclusão de tópicos.** Quando um tópico é excluído, a exclusão dos arquivos de segmento de log no armazenamento de objetos não começa imediatamente. Em vez disso, há um intervalo de tempo com um valor padrão de 3 horas antes que a exclusão desses arquivos ocorra. Você pode modificar a configuração, confluent.tier.topic.delete.check.interval.ms, para alterar o valor deste intervalo. Se você excluir um tópico ou cluster, também poderá excluir manualmente os objetos no respectivo bucket.
- **ACLs sobre tópicos internos de armazenamento em camadas.** Uma prática recomendada para implantações locais é habilitar um autorizador de ACL nos tópicos internos usados para armazenamento em camadas. Defina regras de ACL para limitar o acesso a esses dados somente ao usuário do broker. Isso protege os tópicos internos e impede o acesso não autorizado a dados de armazenamento em camadas e metadados.

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

Substituir o usuário `<kafka>` com o principal corretor atual em sua implantação.

Por exemplo, o comando `confluent-tier-state` define ACLs no tópico interno para armazenamento em camadas. Atualmente, há apenas um único tópico interno relacionado ao armazenamento em camadas. O exemplo cria uma ACL que fornece a permissão principal do Kafka para todas as operações no tópico interno.

Dimensionamento

O dimensionamento do Kafka pode ser executado com quatro modos de configuração: simples, granular, reverso e partições.

Simples

O modo simples é apropriado para usuários iniciantes do Apache Kafka ou casos de uso em estágio inicial. Para este modo, você fornece requisitos como taxa de transferência em MBps, distribuição de leitura, retenção e porcentagem de utilização de recursos (60% é o padrão). Você também entra no ambiente, como no local (bare-metal, VMware, Kubernetes ou OpenStack) ou na nuvem. Com base nessas informações, o dimensionamento de um cluster Kafka fornece o número de servidores necessários para o broker, o zookeeper, os trabalhadores de conexão do Apache Kafka, o registro de esquema, um proxy REST, o ksqlDB e o centro de controle do Confluent.

Para armazenamento em camadas, considere o modo de configuração granular para dimensionar um cluster Kafka. O modo granular é apropriado para usuários experientes do Apache Kafka ou casos de uso bem definidos. Esta seção descreve o dimensionamento para produtores, processadores de fluxo e consumidores.

Produtores

Para descrever os produtores do Apache Kafka (por exemplo, um cliente nativo, proxy REST ou conector Kafka), forneça as seguintes informações:

- **Nome.** Fagulha.
- **Tipo de produtor.** Aplicação ou serviço, proxy (REST, MQTT, outro) e banco de dados existente (RDBMS, NOSQL, outro). Você também pode selecionar "Não sei".
- **Rendimento médio.** Em eventos por segundo (1.000.000 por exemplo).
- **Pico de rendimento.** Em eventos por segundo (4.000.000 por exemplo).
- **Tamanho médio da mensagem.** Em bytes, não compactado (máx. 1 MB; 1000 por exemplo).
- **Formato da mensagem.** As opções incluem Avro, JSON, buffers de protocolo, binário, texto, "Não sei" e outros.
- **Fator de replicação.** As opções são 1, 2, 3 (recomendação da Confluent), 4, 5 ou 6.
- **Tempo de retenção.** Um dia (por exemplo). Por quanto tempo você deseja que seus dados sejam armazenados no Apache Kafka? Digite -1 com qualquer unidade por um tempo infinito. A calculadora assume um tempo de retenção de 10 anos para retenção infinita.
- Marque a caixa de seleção "Habilitar armazenamento em camadas para diminuir a contagem de corretores e permitir armazenamento infinito?"
- Quando o armazenamento em camadas está habilitado, os campos de retenção controlam o conjunto ativo de dados armazenados localmente no broker. Os campos de retenção de arquivamento controlam por quanto tempo os dados são armazenados no armazenamento de objetos de arquivamento.

- **Retenção de armazenamento de arquivo.** Um ano (por exemplo). Por quanto tempo você deseja que seus dados sejam armazenados em armazenamento de arquivo? Digite -1 com qualquer unidade por uma duração infinita. A calculadora assume uma retenção de 10 anos para retenção infinita.
- **Multiplicador de crescimento.** 1 (por exemplo). Se o valor deste parâmetro for baseado na taxa de transferência atual, defina-o como 1. Para dimensionar com base no crescimento adicional, defina este parâmetro como um multiplicador de crescimento.
- **Número de instâncias do produtor.** 10 (por exemplo). Quantas instâncias de produtor estarão em execução? Esta entrada é necessária para incorporar a carga da CPU no cálculo de dimensionamento. Um valor em branco indica que a carga da CPU não está incorporada no cálculo.

Com base neste exemplo de entrada, o dimensionamento tem o seguinte efeito sobre os produtores:

- Taxa de transferência média em bytes não compactados: 1 GBps. Taxa de transferência máxima em bytes não compactados: 4 GBps. Taxa de transferência média em bytes compactados: 400 MBps. Taxa de transferência máxima em bytes compactados: 1,6 GBps. Isso se baseia em uma taxa de compressão padrão de 60% (você pode alterar esse valor).
    - Armazenamento total de hotset no broker necessário: 31.104 TB, incluindo replicação, compactado. Total de armazenamento de arquivo off-broker necessário: 378.432 TB, compactado. Usar"https://fusion.netapp.com" para dimensionamento do StorageGRID .

Os processadores de fluxo devem descrever seus aplicativos ou serviços que consomem dados do Apache Kafka e os produzem de volta no Apache Kafka. Na maioria dos casos, eles são criados no ksqlDB ou no Kafka Streams.

- **Nome.** Serpentina de faíscas.
- **Tempo de processamento.** Quanto tempo esse processador leva para processar uma única mensagem?
    - 1 ms (transformação simples e sem estado) [exemplo], 10 ms (operação com estado na memória).
    - 100 ms (operação de rede ou disco com estado), 1000 ms (chamada REST de terceiros).
    - Eu comparei esse parâmetro e sei exatamente quanto tempo leva.
- **Retenção de saída.** 1 dia (exemplo). Um processador de fluxo produz sua saída de volta para o Apache Kafka. Por quanto tempo você deseja que esses dados de saída sejam armazenados no Apache Kafka? Digite -1 com qualquer unidade por uma duração infinita.
- Marque a caixa de seleção "Habilitar armazenamento em camadas para diminuir a contagem de corretores e permitir armazenamento infinito?"
- **Retenção de armazenamento de arquivo.** 1 ano (por exemplo). Por quanto tempo você deseja que seus dados sejam armazenados em armazenamento de arquivo? Digite -1 com qualquer unidade por uma duração infinita. A calculadora assume uma retenção de 10 anos para retenção infinita.
- **Porcentagem de passagem de saída.** 100 (por exemplo). Um processador de fluxo produz sua saída de volta para o Apache Kafka. Qual porcentagem da taxa de transferência de entrada será retornada ao Apache Kafka? Por exemplo, se a taxa de transferência de entrada for 20 MBps e esse valor for 10, a taxa de transferência de saída será 2 MBps.
- De quais aplicativos isso é lido? Selecione "Spark", o nome usado no dimensionamento baseado no tipo de produtor. Com base na entrada acima, você pode esperar os seguintes efeitos de dimensionamento em instâncias do processador de fluxo e estimativas de partição de tópicos:
- Este aplicativo de processador de fluxo requer o seguinte número de instâncias. Os tópicos recebidos provavelmente também exigirão essa quantidade de partições. Entre em contato com a Confluent para confirmar este parâmetro.
    - 1.000 para rendimento médio sem multiplicador de crescimento

- 4.000 para pico de rendimento sem multiplicador de crescimento
- 1.000 para rendimento médio com um multiplicador de crescimento
- 4.000 para pico de rendimento com um multiplicador de crescimento

Consumidores

Descreva seus aplicativos ou serviços que consomem dados do Apache Kafka e não os produzem de volta no Apache Kafka; por exemplo, um cliente nativo ou um conector Kafka.

- **Nome.** Consumidor Spark.
- **Tempo de processamento.** Quanto tempo esse consumidor leva para processar uma única mensagem?
  - 1 ms (por exemplo, uma tarefa simples e sem estado, como registro)
  - 10 ms (gravações rápidas em um armazenamento de dados)
  - 100 ms (gravações lentas em um armazenamento de dados)
  - 1000 ms (chamada REST de terceiros)
  - Algum outro processo de referência de duração conhecida.
- **Tipo de consumidor.** Aplicação, proxy ou coletor para um armazenamento de dados existente (RDBMS, NoSQL, outro).
- De quais aplicativos isso é lido? Conecte este parâmetro ao produtor e ao dimensionamento do fluxo determinados anteriormente.

Com base na entrada acima, você deve determinar o dimensionamento para instâncias do consumidor e estimativas de partição de tópicos. Um aplicativo de consumidor requer o seguinte número de instâncias.

- 2.000 para rendimento médio, sem multiplicador de crescimento
- 8.000 para pico de rendimento, sem multiplicador de crescimento
- 2.000 para rendimento médio, incluindo multiplicador de crescimento
- 8.000 para pico de rendimento, incluindo multiplicador de crescimento

Os tópicos recebidos provavelmente também precisam desse número de partições. Entre em contato com a Confluent para confirmar.

Além dos requisitos para produtores, processadores de fluxo e consumidores, você deve fornecer os seguintes requisitos adicionais:

- **Hora da reconstrução.** Por exemplo, 4 horas. Se um host do broker do Apache Kafka falhar, seus dados serão perdidos e um novo host for provisionado para substituir o host com falha. Com que rapidez esse novo host deve se reconstruir? Deixe este parâmetro em branco se o valor for desconhecido.
- **Meta de utilização de recursos (porcentagem).** Por exemplo, 60. Quão utilizados você quer que seus hosts sejam durante a taxa de transferência média? A Confluent recomenda uma utilização de 60%, a menos que você esteja usando clusters de autobalanceamento Confluent, caso em que a utilização pode ser maior.

Descreva seu ambiente

- **Em qual ambiente seu cluster será executado?** Amazon Web Services, Microsoft Azure, plataforma de nuvem do Google, bare-metal local, VMware local, OpenStack local ou Kubernates local?

- **Detalhes do anfitrião.** Número de núcleos: 48 (por exemplo), tipo de placa de rede (10GbE, 40GbE, 16GbE, 1GbE ou outro tipo).
- **Volumes de armazenamento.** Anfitrião: 12 (por exemplo). Quantos discos rígidos ou SSDs são suportados por host? A Confluent recomenda 12 discos rígidos por host.
- **Capacidade/volume de armazenamento (em GB).** 1000 (por exemplo). Quanto armazenamento um único volume pode armazenar em gigabytes? A Confluent recomenda discos de 1 TB.
- **Configuração de armazenamento.** Como os volumes de armazenamento são configurados? A Confluent recomenda RAID10 para aproveitar todos os recursos da Confluent. JBOD, SAN, RAID 1, RAID 0, RAID 5 e outros tipos também são suportados.
- **Taxa de transferência de volume único (MBps).** 125 (por exemplo). Qual a velocidade com que um único volume de armazenamento pode ler ou gravar em megabytes por segundo? A Confluent recomenda discos rígidos padrão, que normalmente têm taxa de transferência de 125 MBps.
- **Capacidade de memória (GB).** 64 (por exemplo).

Depois de determinar suas variáveis ambientais, selecione Dimensionar meu cluster. Com base nos parâmetros de exemplo indicados acima, determinamos o seguinte dimensionamento para o Confluent Kafka:

- **Apache Kafka.** Contagem de corretores: 22. Seu cluster está vinculado ao armazenamento. Considere habilitar o armazenamento em camadas para diminuir sua contagem de hosts e permitir armazenamento infinito.
- **Apache ZooKeeper.** Contagem: 5; Trabalhadores de conexão do Apache Kafka: Contagem: 2; Registro de esquema: Contagem: 2; Proxy REST: Contagem: 2; ksqlDB: Contagem: 2; Centro de controle Confluent: Contagem: 1.

Use o modo reverso para equipes de plataforma sem um caso de uso em mente. Use o modo de partições para calcular quantas partições um único tópico requer. Ver https://eventsizer.io para dimensionamento com base nos modos reverso e de partições.

Conclusão

Este documento fornece diretrizes de práticas recomendadas para usar o Confluent Tiered Storage com o armazenamento NetApp , incluindo testes de verificação, resultados de desempenho de armazenamento em camadas, ajuste, conectores Confluent S3 e o recurso de autobalanceamento. Considerando as políticas de ILM, o desempenho do Confluent com vários testes de desempenho para verificação e as APIs S3 padrão do setor, o armazenamento de objetos NetApp StorageGRID é uma escolha ideal para o armazenamento em camadas do Confluent.

Onde encontrar informações adicionais

Para saber mais sobre as informações descritas neste documento, revise os seguintes documentos e/ou sites:

- O que é Apache Kafka

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

- Documentação do produto NetApp

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

- Detalhes do parâmetro S3-sink

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

- Apache Kafka

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

- Armazenamento infinito na plataforma Confluent

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

- Armazenamento em camadas Confluent - Melhores práticas e dimensionamento

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

- Conector de coletor Amazon S3 para plataforma Confluent

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

- Dimensionamento de Kafka

  "https://eventsizer.io"

- Dimensionamento do StorageGRID

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

- Casos de uso do Kafka

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

- Clusters Kafka autobalanceados na plataforma confluente 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/"

- Exemplos de clientes da Instaclustr e detalhes de seus 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/

Informações sobre direitos autorais

Copyright © 2026 NetApp, Inc. Todos os direitos reservados. Impresso nos EUA. Nenhuma parte deste documento protegida por direitos autorais pode ser reproduzida de qualquer forma ou por qualquer meio — gráfico, eletrônico ou mecânico, incluindo fotocópia, gravação, gravação em fita ou storage em um sistema de recuperação eletrônica — sem permissão prévia, por escrito, do proprietário dos direitos autorais.

O software derivado do material da NetApp protegido por direitos autorais está sujeito à seguinte licença e isenção de responsabilidade:

ESTE SOFTWARE É FORNECIDO PELA NETAPP "NO PRESENTE ESTADO" E SEM QUAISQUER GARANTIAS EXPRESSAS OU IMPLÍCITAS, INCLUINDO, SEM LIMITAÇÕES, GARANTIAS IMPLÍCITAS DE COMERCIALIZAÇÃO E ADEQUAÇÃO A UM DETERMINADO PROPÓSITO, CONFORME A ISENÇÃO DE RESPONSABILIDADE DESTE DOCUMENTO. EM HIPÓTESE ALGUMA A NETAPP SERÁ RESPONSÁVEL POR QUALQUER DANO DIRETO, INDIRETO, INCIDENTAL, ESPECIAL, EXEMPLAR OU CONSEQUENCIAL (INCLUINDO, SEM LIMITAÇÕES, AQUISIÇÃO DE PRODUTOS OU SERVIÇOS SOBRESSALENTES; PERDA DE USO, DADOS OU LUCROS; OU INTERRUPÇÃO DOS NEGÓCIOS), INDEPENDENTEMENTE DA CAUSA E DO PRINCÍPIO DE RESPONSABILIDADE, SEJA EM CONTRATO, POR RESPONSABILIDADE OBJETIVA OU PREJUÍZO (INCLUINDO NEGLIGÊNCIA OU DE OUTRO MODO), RESULTANTE DO USO DESTE SOFTWARE, MESMO SE ADVERTIDA DA RESPONSABILIDADE DE TAL DANO.

A NetApp reserva-se o direito de alterar quaisquer produtos descritos neste documento, a qualquer momento e sem aviso. A NetApp não assume nenhuma responsabilidade nem obrigação decorrentes do uso dos produtos descritos neste documento, exceto conforme expressamente acordado por escrito pela NetApp. O uso ou a compra deste produto não representam uma licença sob quaisquer direitos de patente, direitos de marca comercial ou quaisquer outros direitos de propriedade intelectual da NetApp.

O produto descrito neste manual pode estar protegido por uma ou mais patentes dos EUA, patentes estrangeiras ou pedidos pendentes.

LEGENDA DE DIREITOS LIMITADOS: o uso, a duplicação ou a divulgação pelo governo estão sujeitos a restrições conforme estabelecido no subparágrafo (b)(3) dos Direitos em Dados Técnicos - Itens Não Comerciais no DFARS 252.227-7013 (fevereiro de 2014) e no FAR 52.227- 19 (dezembro de 2007).

Os dados aqui contidos pertencem a um produto comercial e/ou serviço comercial (conforme definido no FAR 2.101) e são de propriedade da NetApp, Inc. Todos os dados técnicos e software de computador da NetApp fornecidos sob este Contrato são de natureza comercial e desenvolvidos exclusivamente com despesas privadas. O Governo dos EUA tem uma licença mundial limitada, irrevogável, não exclusiva, intransferível e não sublicenciável para usar os Dados que estão relacionados apenas com o suporte e para cumprir os contratos governamentais desse país que determinam o fornecimento de tais Dados. Salvo disposição em contrário no presente documento, não é permitido usar, divulgar, reproduzir, modificar, executar ou exibir os dados sem a aprovação prévia por escrito da NetApp, Inc. Os direitos de licença pertencentes ao governo dos Estados Unidos para o Departamento de Defesa estão limitados aos direitos identificados na cláusula 252.227-7015(b) (fevereiro de 2014) do DFARS.

Informações sobre marcas comerciais

NETAPP, o logotipo NETAPP e as marcas listadas em http://www.netapp.com/TM são marcas comerciais da NetApp, Inc. Outros nomes de produtos e empresas podem ser marcas comerciais de seus respectivos proprietários.