Les meilleures pratiques pour Kafka fluide

NetApp Solutions

NetApp
April 25, 2024

Sommaire

Les meilleures pratiques pour Kafka fluide

Tr-4912 : recommandations sur les meilleures pratiques pour le stockage hiérarchisé Kafka fluide avec NetApp

Karthikeyan Nagalingam, Joseph Kandatilparambil, NetApp Rankesh Kumar, confluent

Apache Kafka est une plateforme de streaming aux événements distribuée par la communauté qui prend en charge des milliards d'événements par jour. Initialement conçu comme une file d'attente de messagerie, Kafka repose sur l'abstraction d'un journal de validation distribué. Depuis sa création et l'open source par LinkedIn en 2011, Kafka a évolué depuis la file d'attente des messages vers une plateforme complète de streaming d'événements. Confluent assure la distribution d'Apache Kafka avec la plateforme confluent. La plateforme Confluent complète Kafka avec des fonctions communautaires et commerciales supplémentaires conçues pour améliorer l'expérience de streaming tant des opérateurs que des développeurs en production à grande échelle.

Ce document présente les meilleures pratiques pour l'utilisation du stockage hiérarchisé de niveau confluent sur une offre de stockage objet NetApp en fournissant le contenu suivant :

- Vérification couramment assurée avec le stockage objet NetApp : NetApp StorageGRID
- Tests des performances du stockage à plusieurs niveaux
- Instructions sur les meilleures pratiques pour parler couramment les systèmes de stockage NetApp

Pourquoi le stockage à plusieurs niveaux confluent ?

Confluent est devenu la plateforme de streaming en temps réel par défaut pour de nombreuses applications, en particulier pour le Big Data, l'analytique et les charges de travail de streaming. Le stockage à plusieurs niveaux permet aux utilisateurs de séparer les ressources de calcul du stockage dans la plateforme confluent. Cette solution rend le stockage des données plus économique, vous permet de stocker des volumes presque infinis de données et de faire évoluer les charges de travail à la demande (ou en réduisant). Elle simplifie également les tâches administratives telles que le rééquilibrage des données et des locataires. Les systèmes de stockage compatibles S3 peuvent tirer parti de toutes ces capacités pour démocratiser les données avec tous les événements. L'ingénierie des données est ainsi inutile. Pour plus d'informations sur la raison pour laquelle vous devez utiliser le stockage à plusieurs niveaux pour Kafka, vérifiez "Cet article par confluent".

Pourquoi choisir NetApp StorageGRID pour le stockage hiérarchisé ?

StorageGRID est une plateforme de stockage objet leader du marché, StorageGRID est une solution de stockage objet Software-defined qui prend en charge les API objet standard telles qu'Amazon simple Storage Service (S3). StorageGRID stocke et gère des volumes massifs de données non structurées pour un stockage objet sécurisé et durable. Vous placez vos contenus au bon endroit, au bon moment, dans le Tier de stockage adéquat, afin d'optimiser les workflows et de réduire les coûts du contenu enrichi distribué à l'échelle mondiale.

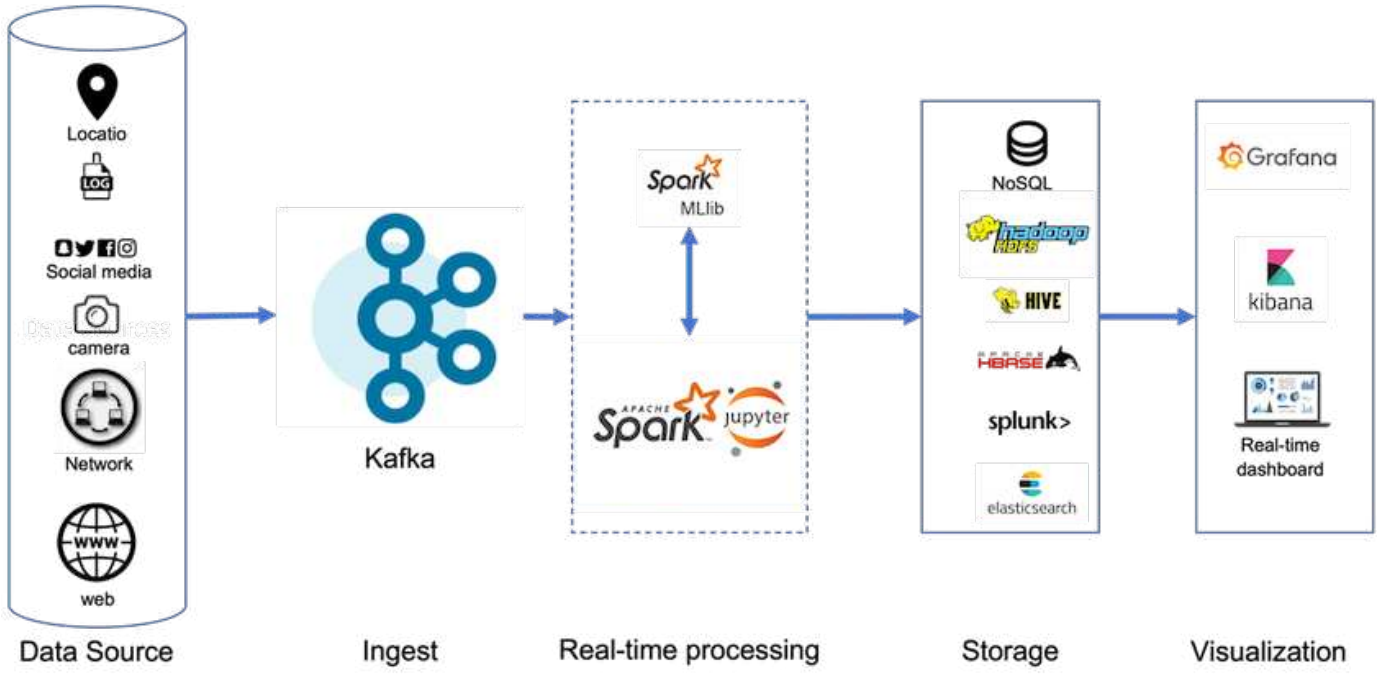
Le plus grand atout concurrentiel de StorageGRID est son moteur de règles de gestion du cycle de vie de l'information (ILM) qui permet une gestion du cycle de vie des données pilotée par les règles (policy). Le moteur de règles peut utiliser les métadonnées pour gérer la façon dont les données sont stockées tout au long de leur durée de vie. Il optimise alors la performance et optimise automatiquement les coûts et la durabilité à mesure que les données vieillissent.

Activation du stockage à plusieurs niveaux confluent

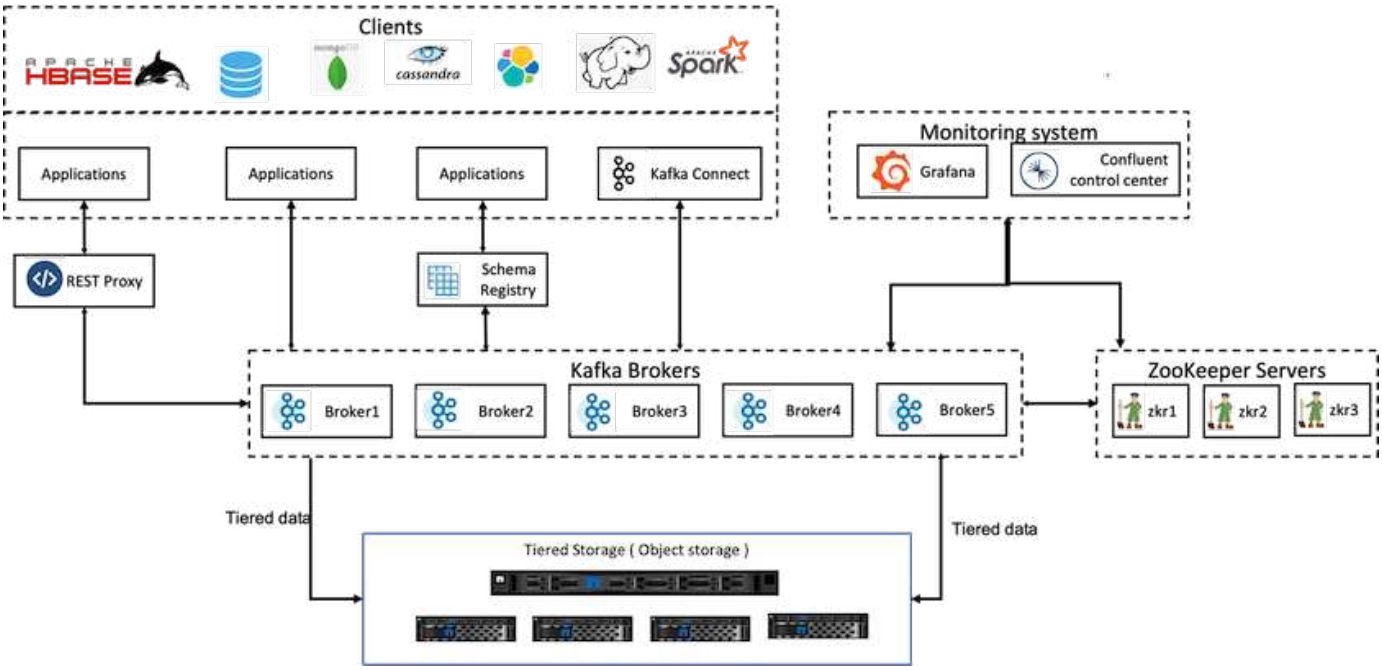
L'idée de base d'un stockage hiérarchisé est de séparer les tâches du stockage des données du traitement des données. Cette séparation facilite l'évolutivité indépendante du niveau de stockage et du Tier de traitement des données.

Une solution de stockage à plusieurs niveaux pour confluent doit contenir deux facteurs. Tout d'abord, ils doivent contourner ou éviter les propriétés communes de disponibilité et de cohérence du magasin d'objets, comme les incohérences dans les opérations DE LISTE et l'indisponibilité occasionnelle d'objets. Deuxièmement, il doit traiter correctement l'interaction entre le stockage à plusieurs niveaux et le modèle de réplication et de tolérance aux pannes de Kafka, y compris la possibilité pour les dirigeants zombies de continuer à classer les plages de décalage. Le stockage objet NetApp fournit à la fois une disponibilité cohérente des objets et des modèles de haute disponibilité qui rendent le stockage fatigué disponible dans les plages de décalage de Tier. Le stockage objet NetApp procure une disponibilité cohérente des objets et un modèle de haute disponibilité pour que le stockage en fatigue soit disponible dans les plages de décalage de Tier.

Le stockage à plusieurs niveaux vous permet d'utiliser des plateformes haute performance pour les lectures et les écritures à faible latence à proximité de l'arrière de vos données de streaming. Vous pouvez également utiliser des magasins d'objets plus économiques et évolutifs comme NetApp StorageGRID pour les lectures historiques à haut débit. Nous disposons également d'une solution technique pour Spark avec le contrôleur de stockage netapp et découvrez comment dans ce document les détails. La figure suivante montre comment Kafka s'intègre dans un pipeline analytique en temps réel.



La figure suivante décrit le positionnement de NetApp StorageGRID comme le Tier de stockage objet couramment utilisé par Kafka.



Détails de l'architecture de la solution

Cette section couvre le matériel et les logiciels utilisés pour la vérification de confluent. Ces informations s'appliquent pour couramment le déploiement des plateformes avec le stockage NetApp. Le tableau suivant couvre l'architecture de la solution testée et les composants de base.

Composants de la solution	Détails
Kafka confluent version 6.2	• Trois zoogardiens • Cinq serveurs de courtage • Cinq serveurs d'outils • Un seul Grafana • Un centre de contrôle
Linux (ubuntu 18.04)	Tous les serveurs
NetApp StorageGRID pour le stockage à plusieurs niveaux	• Logiciel StorageGRID • 1 x SG1000 (équilibreur de charge) • 4 x SGF6024 • 4 SSD 24 x 800 • Protocole S3 • 4 x 100 GbE (connectivité réseau entre le courtier et les instances StorageGRID)
15 serveurs Fujitsu PRIMERGY RX2540	Chacun équipé de : * 2 processeurs, 16 cœurs physiques au total * mémoire physique Intel Xeon * 256 Go * double port 100 GbE

Présentation de la technologie

Cette section décrit la technologie utilisée dans cette solution.

NetApp StorageGRID

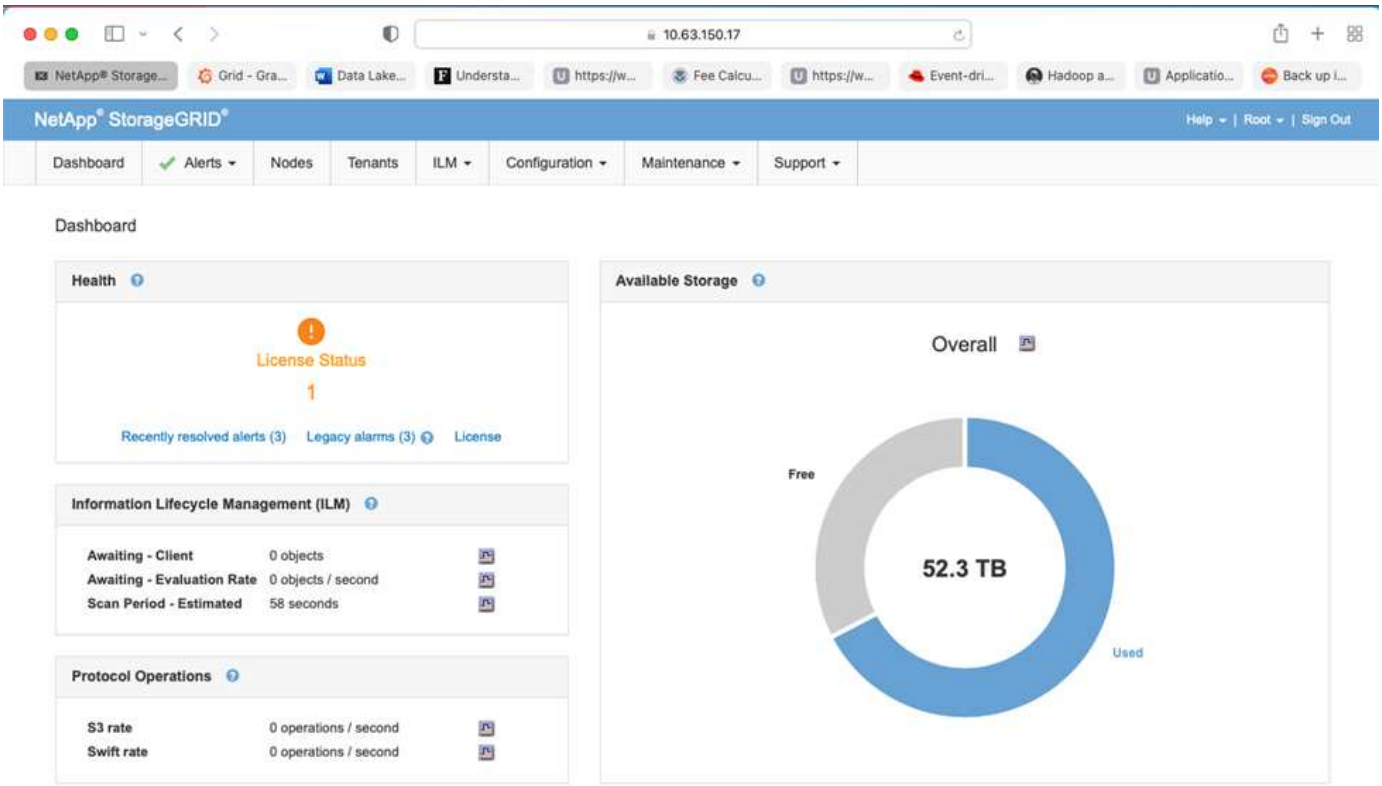
NetApp StorageGRID est une plateforme de stockage objet haute performance et économique. Avec le stockage à plusieurs niveaux, la plupart des données que fournit Kafka, qui sont stockées sur le stockage local ou le stockage SAN du courtier, sont déchargées sur le magasin d'objets distant. Cette configuration apporte d'importantes améliorations opérationnelles en réduisant le temps et les coûts nécessaires au rééquilibrage, à l'extension ou à la réduction des clusters ou au remplacement d'un courtier en panne. Le stockage objet joue un rôle important dans la gestion des données qui résident sur le Tier de stockage objet. Il est donc important de choisir le bon stockage objet.

StorageGRID propose une gestion intelligente et globale des données pilotée par des règles sur une architecture de grid distribuée basée sur des nœuds. Elle simplifie la gestion de pétaoctets de données non structurées et de milliards d'objets grâce à son espace de noms d'objet global universel unique et à des fonctionnalités avancées de gestion des données. Un accès aux objets unique s'étend sur tous les sites et simplifie les architectures haute disponibilité tout en assurant un accès continu aux objets, en cas de panne au niveau du site ou de l'infrastructure.

La colocation permet de prendre en charge plusieurs applications de cloud et de données d'entreprise non structurées dans un même grid, ce qui améliore le ROI et les utilisations de NetApp StorageGRID. Elle offre la possibilité de créer plusieurs niveaux de services avec des règles de cycle de vie des objets basées sur des métadonnées pour optimiser la durabilité, la protection, la performance et la localisation sur plusieurs sites. Les utilisateurs peuvent adapter les règles de gestion des données, surveiller et appliquer des limites de trafic pour s'adapter sans interruption à l'environnement de données, lorsque leurs exigences changent dans des environnements IT en constante évolution.

Gestion simple avec Grid Manager

L'interface graphique de StorageGRID Grid Manager vous permet de configurer, de gérer et de surveiller votre système StorageGRID sur l'ensemble des sites dispersés à travers le monde, dans une seule fenêtre.



L'interface StorageGRID Grid Manager permet d'effectuer les tâches suivantes :

- Gérez des référentiels d'objets répartis à travers le monde de plusieurs pétaoctets, tels que des images, des vidéos et des dossiers.
- Surveiller les nœuds et les services du grid pour assurer la disponibilité des objets.
- Gérez le placement des données d'objet au fil du temps à l'aide de règles de gestion du cycle de vie des informations (ILM). Ces règles régissent ce qui arrive aux données d'un objet après son ingestion, mais aussi leur protection contre la perte, l'emplacement de stockage des données d'objet et leur durée.
- Surveillance des transactions, des performances et des opérations dans le système

Stratégies de gestion du cycle de vie des informations

StorageGRID propose des règles de gestion des données flexibles qui incluent la conservation des copies de réplica de vos objets et l'utilisation de schémas EC (codage d'effacement) comme 2+1 et 4+2 (entre autres) pour stocker vos objets, selon des exigences de performance et de protection des données spécifiques. Les exigences et les charges de travail évoluent au fil du temps. Les règles ILM doivent également évoluer au fil du temps. La modification des règles ILM est une fonction centrale, qui permet aux clients StorageGRID de s'adapter rapidement et facilement à l'évolution permanente de leur environnement. Veuillez vérifier le "Politique ILM" et "Règles ILM" configuration dans StorageGRID.

Performance

StorageGRID permet d'améliorer les performances en ajoutant des nœuds de stockage, qui peuvent être des machines virtuelles, des serveurs bare Metal ou des appliances dédiées telles que la "SG5712, SG5760, SG6060 OU SGF6024". Lors de nos tests, nous avons dépassé les exigences clés de performance Apache Kafka avec un grid à trois nœuds de taille minimale utilisant l'appliance SGF6024. À mesure que les clients font évoluer leur cluster Kafka avec des courtiers supplémentaires, ils peuvent ajouter davantage de nœuds de stockage pour augmenter la performance et la capacité.

Équilibreur de charge et configuration de point final

Les nœuds d'administration d'StorageGRID fournissent l'interface utilisateur Grid Manager (interface utilisateur) et le terminal d'API REST pour afficher, configurer et gérer votre système StorageGRID, ainsi que des journaux d'audit pour suivre l'activité du système. Pour fournir un terminal S3 hautement disponible pour le stockage hiérarchisé Kafka, nous avons implémenté le équilibreur de charge StorageGRID qui s'exécute comme un service sur les nœuds d'administration et les nœuds de passerelle. En outre, l'équilibreur de charge gère également le trafic local et communique avec le GSLB (Global Server Load Balancing) pour faciliter la reprise après incident.

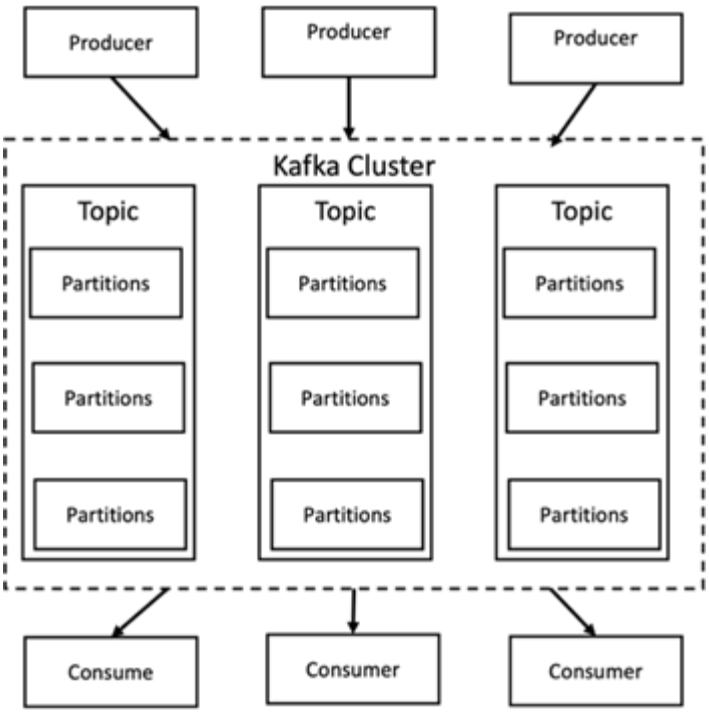
Pour améliorer encore la configuration des terminaux, StorageGRID fournit des règles de classification du trafic intégrées au nœud d'administration, vous permet de surveiller le trafic des workloads et d'appliquer diverses limites de qualité de service à vos charges de travail. Les règles de classification du trafic sont appliquées aux terminaux du service StorageGRID Load Balancer pour les nœuds de passerelle et les nœuds d'administration. Ces politiques peuvent faciliter la mise en forme et la surveillance du trafic.

Classification du trafic à StorageGRID

StorageGRID offre des fonctionnalités de QoS intégrées. Les règles de classification du trafic peuvent aider à surveiller différents types de trafic S3 provenant d'une application client. Vous pouvez ensuite créer et appliquer des stratégies pour mettre des limites sur ce trafic en fonction de la bande passante entrée/sortie, du nombre de demandes simultanées de lecture/écriture ou du taux de demande de lecture/écriture.

Apache Kafka

Apache Kafka est un framework conçu par un bus logiciel qui utilise le traitement en flux écrit dans Java et Scala. Elle vise à fournir une plate-forme unifiée haut débit à faible latence pour la gestion des flux de données en temps réel. Kafka peut se connecter à un système externe pour l'exportation et l'importation de données via Kafka Connect. Ce système fournit les flux Kafka, une bibliothèque de traitement de flux Java. Kafka utilise un protocole TCP binaire optimisé pour son efficacité et s'appuie sur une abstraction « jeu de messages » qui regroupe naturellement les messages ensemble pour réduire la surcharge liée au réseau. Cela permet d'effectuer des opérations sur disque séquentielles plus volumineuses, des paquets réseau plus volumineux et des blocs de mémoire contigus. Kafka peut ainsi transformer un flux d'écritures de messages aléatoires en rafales en écritures linéaires. La figure suivante illustre le flux de données de base d'Apache Kafka.



Kafka stocke les messages clés provenant d'un nombre arbitraire de processus appelés producteurs. Les données peuvent être partitionnées en différentes partitions dans différentes rubriques. Dans une partition, les messages sont strictement ordonnés par leur décalage (la position d'un message dans une partition) et indexés et stockés avec un horodatage. D'autres processus appelés consommateurs peuvent lire des messages à partir de partitions. Pour le traitement par flux, Kafka propose l'API stream qui permet d'écrire les applications Java qui utilisent les données depuis Kafka et écrivent les résultats sur Kafka. Apache Kafka fonctionne également avec les systèmes de traitement de flux externes comme Apache Apex, Apache Flink, Apache Spark, Apache Storm et Apache NiFi.

Kafka s'exécute sur un cluster composé d'un ou de plusieurs serveurs (appelés courtiers), et les partitions de tous les sujets sont distribuées sur les nœuds du cluster. En outre, les partitions sont répliquées sur plusieurs courtiers. Cette architecture permet à Kafka de fournir un flux de messages volumineux tolérant aux pannes et lui a permis de remplacer certains des systèmes de messagerie traditionnels comme JMS (Java message Service), AMQP (Advanced message Queuing Protocol), etc. Depuis la version 0.11.0.0, Kafka propose les écritures transactionnelles qui fournissent un traitement exact du flux à l'aide de l'API stream.

Kafka prend en charge deux types de sujets : classiques et compactés. Les rubriques régulières peuvent être configurées avec une durée de conservation ou une limite d'espace. Si certains enregistrements sont plus anciens que le temps de rétention spécifié ou si la limite d'espace est dépassée pour une partition, Kafka est autorisée à supprimer les anciennes données dans l'espace de stockage disponible. Par défaut, les sujets sont configurés avec une durée de conservation de 7 jours, mais il est également possible de stocker des données indéfiniment. Pour les sujets compactés, les enregistrements n'expirent pas en fonction des limites de temps ou d'espace. Au lieu de cela, Kafka traite les messages plus récents comme des mises à jour de messages plus anciens avec la même clé et garantit de ne jamais supprimer le message le plus récent par clé. Les utilisateurs peuvent entièrement supprimer des messages en écrivant un message appelé tombstone avec la valeur NULL pour une clé spécifique.

Fournit cinq API principales à Kafka :

- **Producer API.** permet à une application de publier des flux d'enregistrements.
- **Consumer API.** permet à une application de s'abonner aux rubriques et de traiter les flux d'enregistrements.

- **API de connecteur.** exécute les API de producteur et de consommateur réutilisables qui peuvent lier les rubriques aux applications existantes.
- **API de flux** cette API convertit les flux d'entrée en sortie et produit le résultat.
- **Admin API.** utilisé pour gérer les sujets Kafka, les courtiers et les autres objets Kafka.

Les API grand public et producteur s'appuient sur le protocole de messagerie Kafka et proposent une implémentation de référence pour les clients consommateurs et producteurs Kafka en Java. Le protocole de messagerie sous-jacent est un protocole binaire que les développeurs peuvent utiliser pour écrire leurs propres clients client ou producteurs dans n'importe quel langage de programmation. Ceci déverrouille Kafka de l'écosystème Java Virtual machine (JVM). Une liste des clients non Java disponibles est conservée dans le wiki Apache Kafka.

Cas d'utilisation d'Apache Kafka

Apache Kafka est le plus populaire pour la messagerie, le suivi des activités du site Web, les metrics, l'agrégation de journaux, le traitement du flux, approvisionnement des événements et consignation des enregistrements.

- Kafka a amélioré le débit, le partitionnement intégré, la réplication et la tolérance aux pannes, ce qui en fait une solution idéale pour les applications de traitement de messages à grande échelle.
- Kafka peut reconstruire les activités d'un utilisateur (vues de pages, recherches) dans un pipeline de suivi comme un ensemble de flux de publication-abonnement en temps réel.
- Kafka est souvent utilisé pour les données de surveillance opérationnelle. Cela implique d'agréger des statistiques à partir d'applications distribuées pour produire des flux centralisés de données opérationnelles.
- Beaucoup de gens utilisent Kafka comme solution de remplacement d'agrégation de journaux. L'agrégation de journaux collecte généralement les fichiers journaux physiques hors des serveurs et les place dans un emplacement central (par exemple, un serveur de fichiers ou HDFS) pour le traitement. Kafka extrait les détails des fichiers et assure un abstraction plus fluide des données du journal ou d'événements sous forme de flux de messages. Cela permet un traitement à faible latence et une prise en charge simplifiée de plusieurs sources de données et de la consommation des données distribuées.
- De nombreux utilisateurs du traitement des données Kafka traitent les données de pipelines de traitement comme plusieurs étapes. Ces données brutes sont consommées à partir de sujets Kafka, puis sont agrégées, enrichies ou transformées en nouveaux sujets afin de favoriser la consommation ou le traitement du suivi. Par exemple, un pipeline de traitement pour recommander des articles de nouvelles peut ramper le contenu de l'article à partir des flux RSS et le publier dans un thème "articles". Un traitement plus poussé peut normaliser ou dédupliquer ce contenu et publier le contenu de l'article nettoyé vers un nouveau sujet, et une étape de traitement finale peut tenter de recommander ce contenu aux utilisateurs. Ces pipelines de traitement créent des graphiques de flux de données en temps réel sur la base de sujets individuels.
- Le sourd d'événement est un style de conception d'application pour lequel les changements d'état sont consignés sous forme d'une séquence d'enregistrements ordonnée à l'heure. La prise en charge de Kafka pour les journaux stockés les plus volumineux en fait un excellent back-end pour une application intégrée dans ce style.
- Kafka peut servir de journal externe destiné à un système distribué. Ce journal aide à la réplication des données entre les nœuds et agit comme un mécanisme de resynchronisation pour les nœuds défaillants afin de restaurer leurs données. La fonctionnalité de compaction des journaux dans Kafka vous aide à prendre en charge ce cas d'utilisation.

Confluent

La plateforme Confluent est une plateforme prête pour l'entreprise qui complète Kafka avec les capacités avancées conçues pour accélérer le développement et la connectivité des applications, permettre les transformations par le traitement du flux, simplifier les opérations à grande échelle et répondre aux exigences architecturales strictes. Conçu par les créateurs d'Apache Kafka à l'origine, ce logiciel étend les avantages de Kafka avec des fonctionnalités haute performance tout en éliminant les tâches de gestion et de surveillance Kafka. Aujourd'hui, plus de 80 % des entreprises classées au Fortune 100 sont équipées de technologies de streaming de données, et la plupart d'entre elles utilisent la technique de confluent.

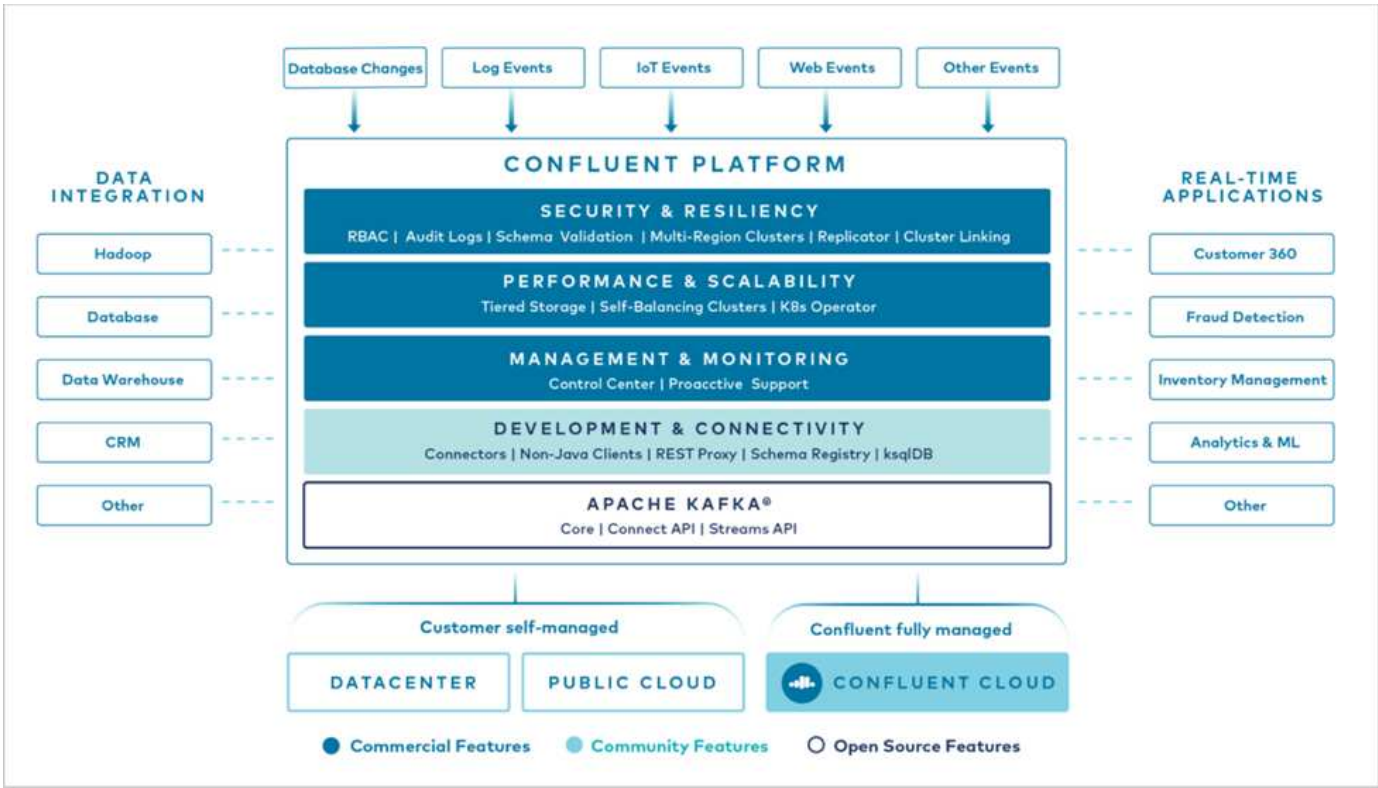
Pourquoi confluent ?

En intégrant des données historiques et en temps réel dans une seule source centrale de vérité, Confluent facilite la création d'une toute nouvelle catégorie d'applications modernes orientées événements, en bénéficiant d'un pipeline de données universel et en permettant d'exploiter de nouveaux cas d'utilisation avec évolutivité, performances et fiabilité.

À quoi sert le confluent ?

Confluent Platform vous permet de vous concentrer sur la manière de tirer de la valeur commerciale de vos données plutôt que de vous soucier des mécanismes sous-jacents, tels que le mode de transport ou d'intégration des données entre des systèmes disparates. La plateforme Confluent simplifie la connexion des sources de données à Kafka, créant des applications de streaming, ainsi que la sécurisation, le contrôle et la gestion de votre infrastructure Kafka. Aujourd'hui, la plateforme parler couramment utilisée pour de nombreux cas d'utilisation dans de nombreux secteurs, qu'il s'agisse des services financiers, de la vente en canaux multiples, des voitures autonomes, de la détection des fraudes, Les microservices et l'IoT.

La figure suivante montre les composants confluent de la plateforme Kafka.



Présentation de la technologie de diffusion d'événements de Confluent

Au cœur de la plate-forme de confluent est "Apache Kafka", la plate-forme de streaming distribuée open-source la plus populaire. Les capacités clés de Kafka sont les suivantes :

- Publiez et abonnez-vous à des flux d'enregistrements.
- Stockez les flux d'enregistrements de manière tolérante aux pannes.
- Traiter les flux d'enregistrements.

La plate-forme confluent prête à l'emploi comprend également le registre de schéma, le proxy REST, un total de plus de 100 connecteurs prédéfinis Kafka et ksqlDB.

Présentation des fonctionnalités d'entreprise de la plate-forme confluent

- **Confluent Control Center.** Un système à interface graphique pour la gestion et le contrôle de Kafka. Il vous permet de gérer facilement Kafka Connect et de créer, modifier et gérer les connexions avec d'autres systèmes.
- **Confluent pour Kubernetes.** Confluent pour Kubernetes est un opérateur Kubernetes. Les opérateurs Kubernetes étendent les fonctionnalités d'orchestration de Kubernetes en fournissant des fonctionnalités et des exigences uniques pour une application de plateforme spécifique. Pour la plateforme Confluent, cela inclut de simplifier considérablement le processus de déploiement de Kafka sur Kubernetes et d'automatiser les tâches du cycle de vie de l'infrastructure classiques.
- **Connecteurs confluent à Kafka.** les connecteurs utilisent l'API Kafka Connect pour connecter Kafka à d'autres systèmes tels que les bases de données, les magasins de valeur clé, les index de recherche et les systèmes de fichiers. Confluent Hub dispose de connecteurs téléchargeables pour les sources de données et les éviers les plus populaires, y compris les versions entièrement testées et prises en charge de ces connecteurs avec plate-forme confluent. Plus de détails sont disponibles "ici".
- **Clusters à auto-équilibrage.** offre un équilibrage de charge automatisé, une détection des pannes et une auto-rétablissement. Il permet d'ajouter ou de désaffecter des courtiers en fonction des besoins, sans réglage manuel.
- * Liaison cluster de confluent.* connecte directement les clusters et met en miroir les sujets d'un cluster à un autre via un pont de liaison. La liaison entre clusters simplifie la configuration des déploiements de clouds hybrides, multiclouds et multiclouds.
- **BALANCER de données de confluent.** surveille le nombre de courtiers, la taille des partitions, le nombre de partitions et le nombre de lignes d'attache au sein du cluster. Il vous permet de déplacer des données pour créer une charge de travail homogène dans le cluster, tout en limitant le trafic pour limiter l'impact sur les workloads de production tout en procédant à un rééquilibrage.
- **Le réplicateur confluent.** facilite plus que jamais la maintenance de plusieurs clusters Kafka dans de multiples centres de données.
- **Stockage à plusieurs niveaux.** fournit des options pour stocker des volumes importants de données Kafka à l'aide de votre fournisseur de cloud favori, ce qui réduit la charge opérationnelle et le coût. Le stockage hiérarchisé permet de conserver les données sur un stockage objet économique et de les faire évoluer uniquement lorsque vous avez besoin de ressources de calcul supplémentaires.
- * Client JMS confluent.* plate-forme confluent comprend un client compatible JMS pour Kafka. Ce client Kafka met en œuvre l'API standard JMS 1.1, en utilisant les courtiers Kafka comme back-end. Ceci est utile si vous avez des applications héritées utilisant JMS et que vous souhaitez remplacer le courtier de messages JMS existant par Kafka.
- **Proxy MQTT confluent.** fournit un moyen de publier des données directement sur Kafka à partir de périphériques et passerelles MQTT sans avoir besoin d'un courtier MQTT au milieu.

- * Plugins de sécurité confluent.* des plugins de sécurité confluent sont utilisés pour ajouter des capacités de sécurité à divers outils et produits de plate-forme confluent. Actuellement, un plug-in est disponible pour le proxy REST confluent qui permet d'authentifier les demandes entrantes et de propager le principal authentifié aux demandes vers Kafka. Les clients proxy REST prolixes utilisent ainsi les fonctionnalités de sécurité multilocataires du courtier Kafka.

Vérification confluent

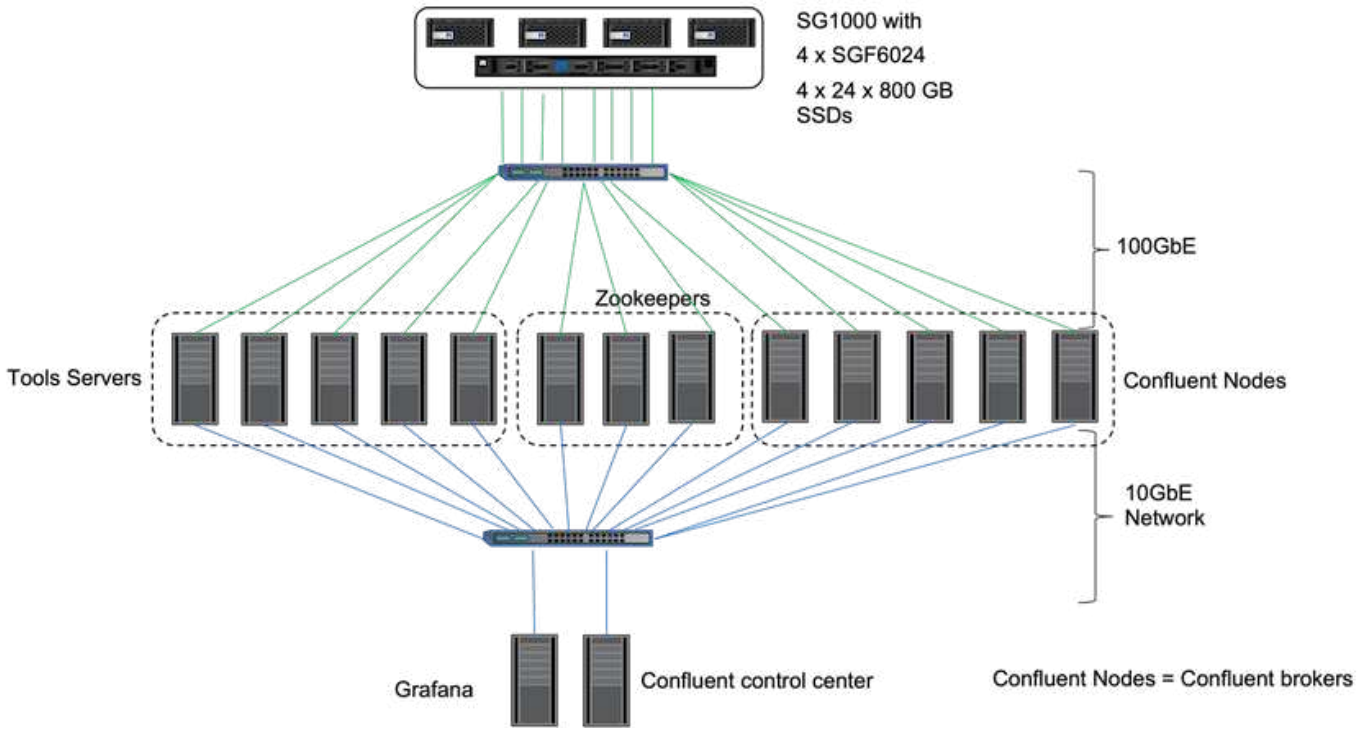
Nous avons effectué une vérification avec le stockage hiérarchisé Confluent Platform 6.2 dans NetApp StorageGRID. Les équipes NetApp et confluent ont collaboré à cette vérification et ont exécuté les cas de test requis pour la vérification.

Configuration de la plate-forme Confluent

Nous avons utilisé la configuration suivante pour la vérification.

À des fins de vérification, nous avons utilisé trois zoopers, cinq courtiers, cinq serveurs d'exécution de scripts de test, des serveurs d'outils nommés avec 256 Go de RAM et 16 processeurs. Pour le stockage NetApp, nous avons utilisé StorageGRID avec un équilibreur de charge SG1000 et avec quatre SGF6024s. Le stockage et les courtiers étaient connectés via des connexions 100 GbE.

La figure suivante montre la topologie réseau de la configuration utilisée pour la vérification de confluent.



Les serveurs d'outils agissent comme des clients d'application qui envoient des demandes aux nœuds de confluent.

Configuration du stockage multi-niveaux fluide

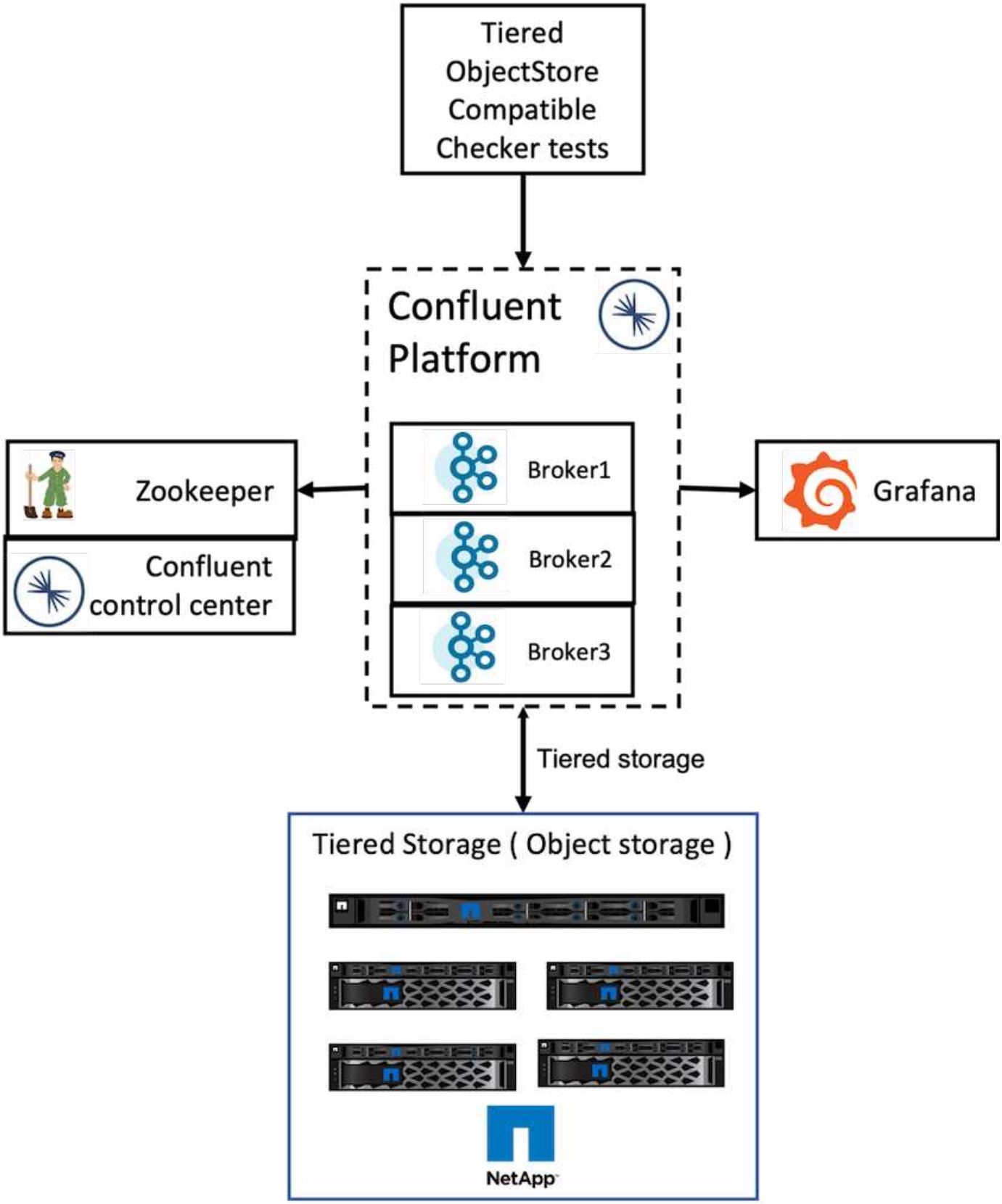
La configuration de stockage à plusieurs niveaux nécessite les paramètres suivants dans Kafka :

```
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
```

À des fins de vérification, nous avons utilisé StorageGRID avec le protocole HTTP, mais HTTPS fonctionne également. La clé d'accès et la clé secrète sont stockées dans le nom de fichier fourni dans le `confluent.tier.s3.cred.file.path` paramètre.

Stockage objet NetApp - StorageGRID

Nous avons configuré la configuration du site unique dans StorageGRID pour la vérification.



Tests de vérification

Nous avons complété les cinq tests suivants pour la vérification. Ces tests sont exécutés sur le cadre de Trogdor. Les deux premiers étaient les tests de fonctionnalité et les trois autres étaient les tests de performance.

Test d’exactitude du magasin d’objets

Ce test détermine si toutes les opérations de base (par exemple GET/PUT/delete) de l’API du magasin d’objets fonctionnent bien selon les besoins du stockage hiérarchisé. C’est un test de base que chaque service de magasin d’objets devrait s’attendre à passer avant les tests suivants. C’est un test d’assurance qui réussit ou échoue.

Test d’exactitude des fonctionnalités de hiérarchisation

Ce test détermine si la fonctionnalité de stockage à plusieurs niveaux fonctionne correctement avec un test assertif qui réussit ou échoue. Le test crée un sujet de test qui est configuré par défaut avec le Tiering activé et une taille de groupe de signets très réduite. Il produit un flux d’événements vers le nouveau sujet de test créé, il attend que les courtiers archivent les segments dans le magasin d’objets, puis il utilise le flux d’événements et valide que le flux consommé correspond au flux produit. Le nombre de messages produits au flux d’événements est configurable, ce qui permet à l’utilisateur de générer une charge de travail suffisamment importante en fonction des besoins du test. La taille réduite du hot set garantit que les fetches du consommateur en dehors du segment actif ne sont servies qu’à partir du magasin d’objets ; cela permet de tester l’exactitude du magasin d’objets pour les lectures. Nous avons effectué ce test avec et sans injection de défaut dans le magasin d’objets. Nous avons simulé une panne des nœuds en arrêtant le service Service Manager dans l’un des nœuds de StorageGRID et en validant que la fonctionnalité de bout en bout fonctionne avec le stockage objet.

Banc d’essai de récupération de Tier

Ce test a validé les performances de lecture du stockage d’objets hiérarchisés et vérifié les demandes de lecture de plage en charge lourde à partir des segments générés par le banc d’essai. Dans ce banc d’essai, confluent a développé des clients personnalisés pour traiter les demandes d’extraction de niveau.

Banc d’essai des charges de travail « production »

Ce test a généré indirectement le workload d’écriture sur le magasin d’objets via l’archivage de segments. Le workload de lecture (segments lus) a été généré à partir du stockage objet lorsque les groupes de consommateurs ont extrait les segments. Ce workload a été généré par le script de test. Ce test a vérifié les performances de lecture et d’écriture sur le stockage objet dans les threads parallèles. Nous avons testé avec et sans injection de panne dans le magasin d’objets, comme nous l’avons fait pour le test d’exactitude de la fonctionnalité de Tiering.

Banc d’essai des workloads de conservation

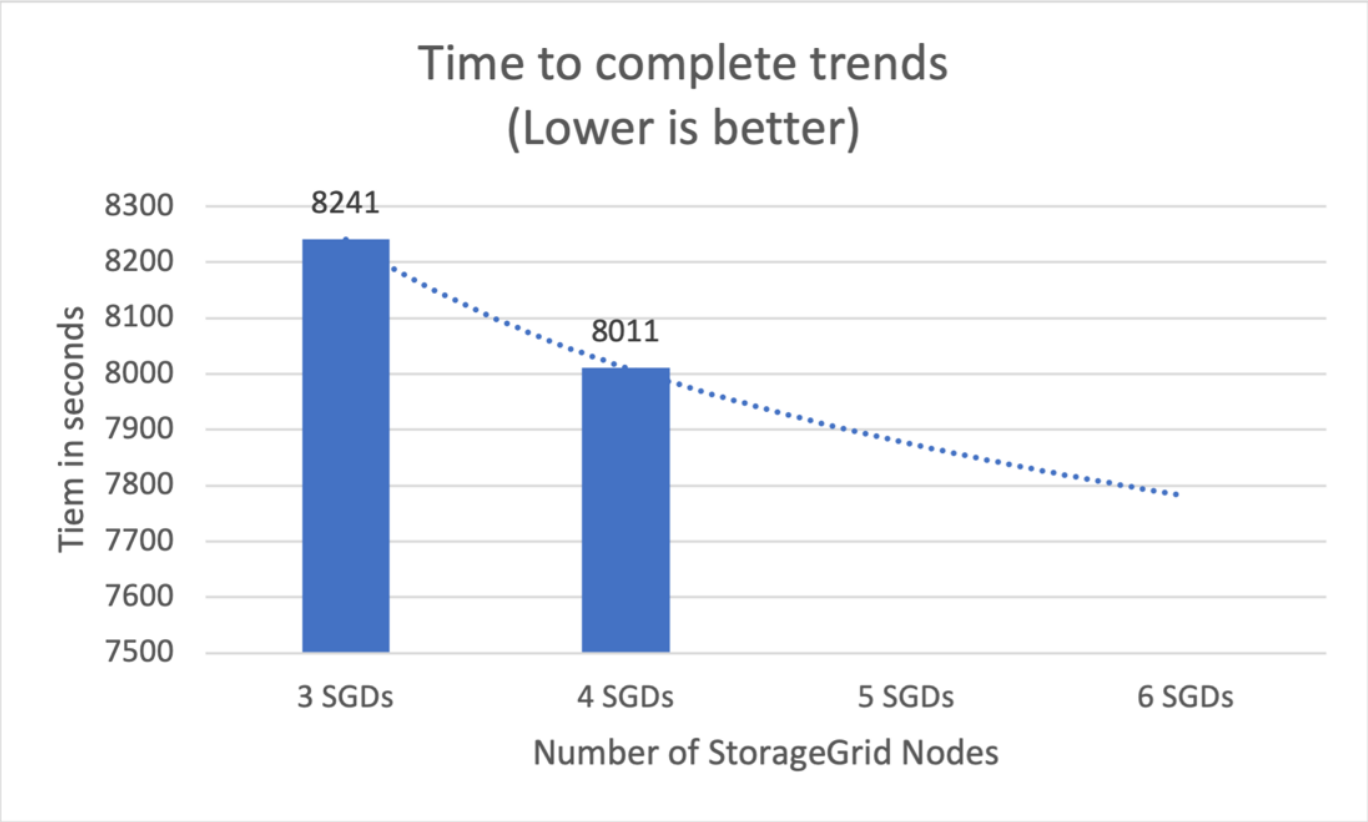
Ce test a permis de vérifier les performances de suppression d’un magasin d’objets sous un workload de conservation des rubriques lourd. La charge de travail de rétention a été générée à l’aide d’un script de test qui produit de nombreux messages en parallèle à un sujet de test. La rubrique de test était configurée avec un paramètre de conservation basé sur la taille et le temps agressif qui a provoqué la purge continue du flux d’événements du magasin d’objets. Les segments ont ensuite été archivés. Cela a entraîné de nombreuses suppressions dans le stockage objet par le courtier et la collecte des performances des opérations de suppression du magasin d’objets.

Tests de performances avec évolutivité

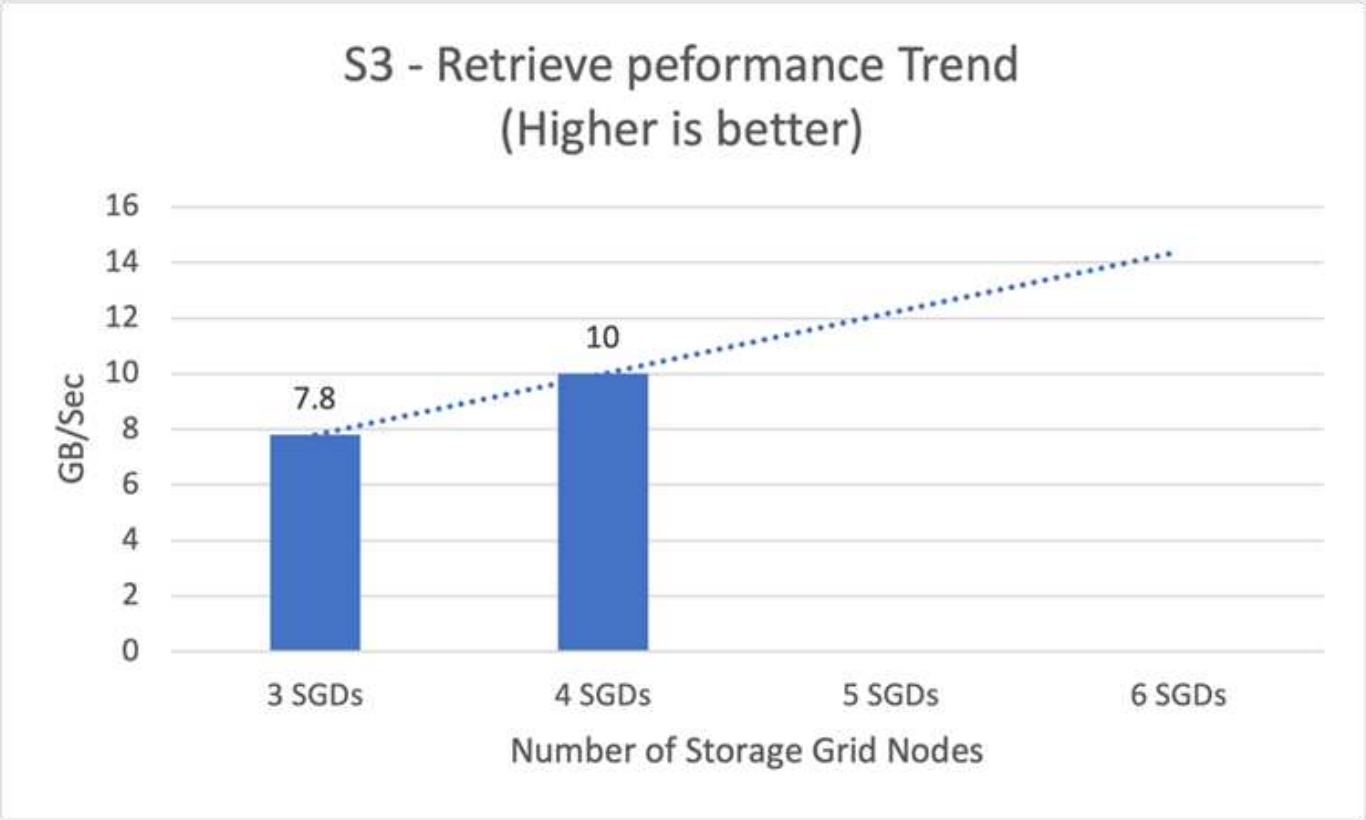
Nous avons réalisé le test du stockage sur plusieurs niveaux avec trois à quatre nœuds pour les charges de travail des producteurs et des consommateurs, grâce à la configuration NetApp StorageGRID. Selon nos tests, le temps d’exécution et les résultats en termes de performances étaient directement proportionnels au nombre de nœuds

StorageGRID. Le setup StorageGRID a nécessité au moins trois nœuds.

- La durée des opérations de production et de production a diminué de façon linéaire lorsque le nombre de nœuds de stockage augmente.



- Les performances de l'opération de récupération s3 augmentent de façon linéaire en fonction du nombre de nœuds StorageGRID. StorageGRID prend en charge jusqu'à 200 nœuds StorageGRID.



Connecteur s3 confluent

Le connecteur d'évier Amazon S3 exporte les données des sujets Apache Kafka vers des objets S3 au format Avro, JSON ou octets. Le connecteur d'évier Amazon S3 interroge régulièrement les données depuis Kafka et les télécharge à son tour sur S3. Un partitionneur est utilisé pour diviser les données de chaque partition Kafka en segments. Chaque bloc de données est représenté en tant qu'objet S3. Le nom de clé encode le sujet, la partition Kafka et le décalage de début de ce segment de données.

Dans ce configuration, nous vous montrons comment lire et écrire des sujets dans le stockage objet depuis Kafka directement à l'aide du connecteur lavabo Kafka s3. Pour ce test, nous avons utilisé un cluster Confluent autonome, mais cette configuration s'applique à un cluster distribué.

1. Téléchargez le livre confluent Kafka depuis le site Web confluent.
2. Déballez le paquet dans un dossier de votre serveur.
3. Exporter deux variables.

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

4. Pour une configuration autonome que Kafka confluent, le cluster crée un dossier racine temporaire dans `/tmp`. Cette solution crée également Zookeeper, Kafka, un registre de schéma, Connect, un serveur ksql, et les dossiers du centre de contrôle et copie leurs fichiers de configuration respectifs à partir de `$CONFLUENT_HOME`. Voir l'exemple suivant :

```
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. Configurer le Zookeeper. Vous n'avez rien à changer si vous utilisez les paramètres par défaut.

```
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:~#
```

Dans la configuration ci-dessus, nous avons mis à jour le `server. xxx` propriété. Par défaut, vous avez besoin de trois zoopers pour la sélection du leader Kafka.

6. Nous avons créé un fichier myID dans `/tmp/confluent.406980/zookeeper/data` Avec un ID unique :

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

Nous avons utilisé le dernier nombre d'adresses IP pour le fichier myID. Nous avons utilisé des valeurs par défaut pour Kafka, Connect, control-Center, Kafka, Kafka-REST, configurations de serveur ksql et de registre de schéma.

7. Démarrer les services 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#
```

Il existe un dossier journal pour chaque configuration, ce qui permet de résoudre les problèmes. Dans certains cas, le démarrage des services prend plus de temps. Assurez-vous que tous les services sont opérationnels.

8. Installez Kafka Connect à l'aide de `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#
```

Vous pouvez également installer une version spécifique en utilisant `confluent-hub install confluentinc/kafka-connect-s3:10.0.3`.

9. Par défaut, `confluentinc-kafka-connect-s3` est installé dans `/data/confluent/confluent-6.2.0/share/confluent-hub-components/confluentinc-kafka-connect-s3`.
10. Mettez à jour le chemin du plug-in avec le nouveau `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. Arrêtez les services de confluent et redémarrez-les.

```
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. Configurez l'ID d'accès et la clé secrète dans le `/root/.aws/credentials` fichier.

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

13. Vérifier que le godet est accessible.

```
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. Configurez le fichier de propriétés s3-lavabo pour s3 et la configuration de compartiment.

```
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. Importez quelques enregistrements dans le compartiment 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. Chargez le connecteur de l’évier s3.

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

17. Vérifiez l’état de l’évier 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. Vérifiez le journal pour vous assurer que s3-lavabo est prêt à accepter les rubriques.

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

19. Vérifiez les sujets dans 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. Vérification des objets dans le compartiment 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. Pour vérifier le contenu, copiez chaque fichier depuis S3 vers votre système de fichiers local à l'aide de la commande suivante :

```
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. Pour imprimer les enregistrements, utilisez avro-tools-1.11.0.1.jar (disponible dans le "Archives 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:~#
```

Clusters d’auto-équilibrage fluides

Si vous avez déjà géré un cluster Kafka, vous connaissez probablement les défis liés à la réaffectation manuelle des partitions vers différents courtiers afin de vous assurer que la charge de travail est équilibrée sur le cluster. Pour les entreprises dotées de déploiements Kafka volumineux, la nécessité de déstocker d’importants volumes de données peut s’avérer fastidieux, fastidieuse et risquée, en particulier si les applications stratégiques sont intégrées au cluster. Toutefois, même dans les cas d’utilisation de Kafka les plus petits, le processus prend du temps et est sujet aux erreurs humaines.

Nous avons testé la fonctionnalité de clusters d’auto-équilibrage courants qui automatise le rééquilibrage en fonction des modifications de la topologie du cluster ou des charges irrégulières. Le test de rééquilibrage courant permet de mesurer le temps nécessaire à l’ajout d’un nouveau courtier en cas de défaillance du nœud ou d’évolution nécessitant un rééquilibrage des données dans les différents courtiers. Dans les configurations Kafka classiques, le volume de données à rééquilibrer augmente avec la croissance du cluster, mais dans le stockage à plusieurs niveaux, le rééquilibrage est limité à une petite quantité de données. Basé sur notre validation, le rééquilibrage du stockage à plusieurs niveaux ne prend que quelques secondes ou minutes dans une architecture Kafka classique, ce qui augmente de façon linéaire à mesure que le cluster augmente.

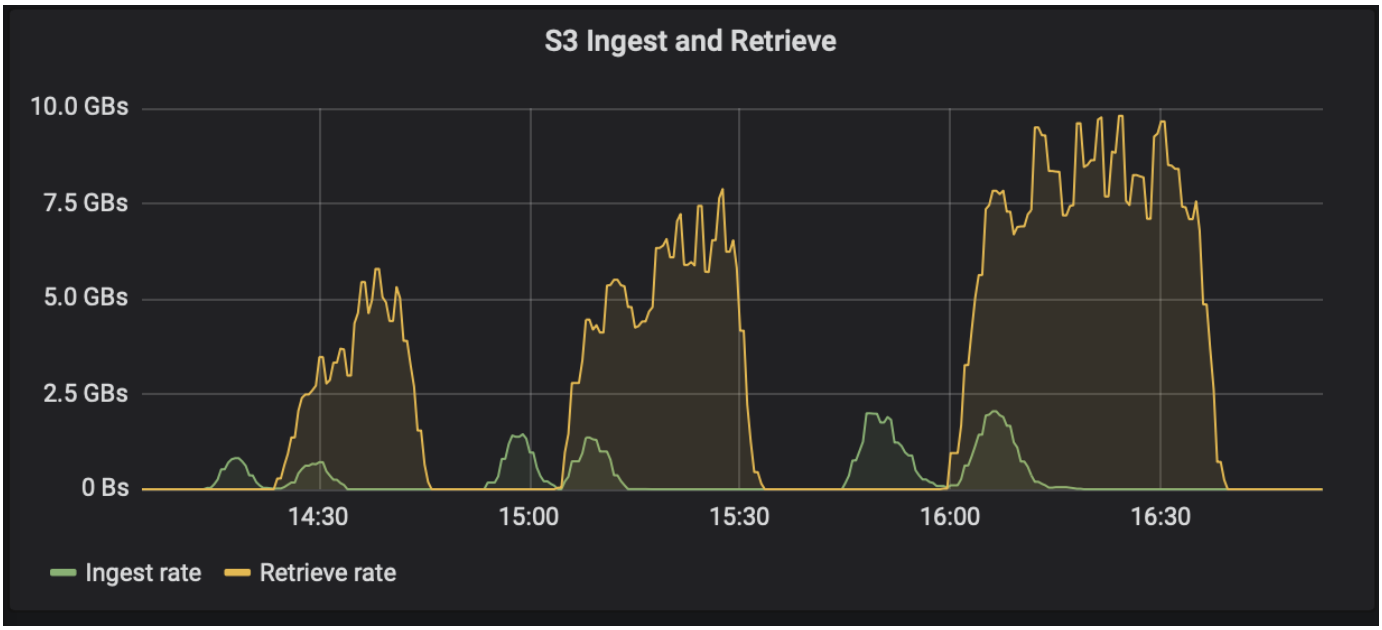
Dans les clusters à auto-équilibrage, le rééquilibrage des partitions est entièrement automatisé afin d’optimiser le débit de Kafka, d’accélérer l’évolutivité des courtiers et de réduire la charge opérationnelle liée à l’exécution d’un grand cluster. Dans un état stable, les clusters à auto-équilibrage surveillent l’inclinaison des données dans les « courtiers » et réaffirment en permanence les partitions afin d’optimiser les performances du cluster. Lorsque la plateforme peut évoluer verticalement ou horizontalement, les clusters à équilibrage automatique reconnaissent automatiquement la présence de nouveaux courtiers ou le retrait d’anciens courtiers et déclenchent une réaffectation ultérieure des partitions. Vous pouvez ainsi ajouter et désaffecter facilement des courtiers et, ce qui rend vos clusters Kafka fondamentalement plus élastiques. Ces avantages sont sans intervention manuelle, mathématiques complexes ou risque d’erreur humaine que les réaffectations de partition entraînent généralement. Le rééquilibrage des données se fait donc en beaucoup moins de temps. Il reste à votre disposition des projets de streaming à plus forte valeur ajoutée, plutôt que de devoir superviser constamment vos clusters.

Recommandations sur les bonnes pratiques

Cette section présente les leçons tirées de cette certification.

- Sur la base de notre validation, le stockage objet S3 convient parfaitement au maintien fluide des données.
- Nous pouvons utiliser un SAN haut débit (notamment FC) pour conserver les données actives du courtier ou le disque local, car, en termes de configuration du stockage multi-niveaux, la taille des données stockées dans le répertoire des courtiers dépend de la taille du segment et de la durée de conservation lorsque les données sont déplacées vers le stockage objet.
- Les magasins d’objets offrent de meilleures performances lorsque segment.octets est plus élevé ; nous avons testé 512 Mo.
- Dans Kafka, la longueur de la clé ou de la valeur (en octets) pour chaque enregistrement produit sur le sujet est contrôlée par le `length.key.value` paramètre. Pour StorageGRID, les valeurs d’ingestion et de récupération d’objets S3 sont supérieures à la valeur supérieure. Par exemple, 512 octets fournis pour une récupération de 5,8 Gbit/s, 1024 octets fournis pour 7,5Gbit/s S3 en récupération et 2048 octets fournis à proximité de 10 Gbit/s.

La figure suivante présente l’ingestion et la récupération d’objet S3 basées sur `length.key.value`.



- **Kafka Tuning.** pour améliorer les performances du stockage à plusieurs niveaux, vous pouvez augmenter TierFetcherNumThreads et TierArchiverNumThreads. En règle générale, vous souhaitez augmenter TierFetcherNumThreads afin qu'ils correspondent au nombre de cœurs de CPU physiques et qu'ils augmentent TierArchiverNumThreads à la moitié du nombre de cœurs de CPU. Par exemple, dans les propriétés du serveur, si vous avez une machine avec huit cœurs physiques, définissez confluent.Tier.fetcher.num.threads = 8 et confluent.Tier.archiver.num.threads = 4.
- **Intervalle de temps pour les suppressions de rubrique.** lorsqu'une rubrique est supprimée, la suppression des fichiers de segment de journal dans le stockage d'objet ne commence pas immédiatement. Il y a plutôt un intervalle de temps avec une valeur par défaut de 3 heures avant la suppression de ces fichiers. Vous pouvez modifier la configuration, confluent.tier.topic.delete.check.interval.ms, pour modifier la valeur de cet intervalle. Si vous supprimez une rubrique ou un cluster, vous pouvez également supprimer manuellement les objets du compartiment correspondant.
- **Listes de contrôle d'accès sur les sujets internes de stockage à plusieurs niveaux.** Une meilleure pratique recommandée pour les déploiements sur site consiste à activer un approbateur ACL sur les sujets internes utilisés pour le stockage à plusieurs niveaux. Définissez les règles ACL pour limiter l'accès à ces données à l'utilisateur du courtier uniquement. Cela sécurise les sujets internes et empêche les accès non autorisés aux données et aux métadonnées de stockage hiérarchisées.

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

Remplacer l'utilisateur `<kafka>` avec le véritable courtier principal dans votre déploiement.

Par exemple, la commande `confluent-tier-state` Définit les listes de contrôle d'accès sur la rubrique interne pour le stockage à plusieurs niveaux. Actuellement, une seule rubrique interne est consacrée au stockage à plusieurs niveaux. L'exemple crée une liste de contrôle d'accès qui fournit l'autorisation Kafka principale pour toutes les opérations sur le sujet interne.

Dimensionnement

Le dimensionnement Kafka peut être effectué avec quatre modes de configuration : simples, granulaires, inverses et partitions.

Simplicité

Le mode simple est adapté aux utilisateurs Apache Kafka pour la première fois ou aux cas d'utilisation précoces. Dans ce mode, vous indiquez des exigences telles que le débit Mbit/s, la lecture de l'extraction, la conservation et le pourcentage d'utilisation des ressources (60 % par défaut). Vous entrez également dans cet environnement, comme sur site (bare-Metal, VMware, Kubernetes ou OpenStack) ou dans le cloud. Sur la base de ces informations, le dimensionnement d'un cluster Kafka indique le nombre de serveurs requis pour le courtier, le gardien de domaine, les employés Apache Kafka Connect, le registre de schéma, un proxy REST, ksqlDB et le centre de contrôle confluent.

Pour le stockage à plusieurs niveaux, tenez compte du mode de configuration granulaire pour le dimensionnement d'un cluster Kafka. Le mode granulaire est adapté aux utilisateurs Apache Kafka expérimentés ou aux cas d'utilisation bien définis. Cette section décrit le dimensionnement des producteurs, des processeurs de flux et des consommateurs.

Producteurs

Pour décrire les producteurs d'Apache Kafka (par exemple un client natif, un proxy REST ou un connecteur Kafka), fournissez les informations suivantes :

- **Nom.** Spark.
- **Type Producteur.** application ou service, proxy (REST, MQTT, autre) et base de données existante (SGBDR, NOSQL, autre). Vous pouvez également sélectionner « Je ne sais pas ».
- **Débit moyen.** en événements par seconde (1,000,000 par exemple).
- **Débit maximal.** en événements par seconde (4,000,000 par exemple).
- **Taille moyenne des messages.** en octets non compressés (max 1MB; 1000 par exemple).
- **Format de message.** les options incluent Avro, JSON, tampons de protocole, binaire, texte, « Je ne sais pas » et autres.
- **Facteur de réplication.** les options sont 1, 2, 3 (recommandation confluent), 4, 5, ou 6.
- **Temps de rétention.** un jour (par exemple). Combien de temps souhaitez-vous que vos données soient stockées dans Apache Kafka ? Entrez -1 avec n'importe quelle unité pour une durée infinie. La calculatrice suppose un temps de rétention de 10 ans pour une rétention infinie.
- Cochez la case « Activer le stockage à plusieurs niveaux pour réduire le nombre de courtiers et autoriser le stockage Infinite Storage ? ».
- Lorsque le stockage à plusieurs niveaux est activé, les champs de conservation contrôlent le jeu actif des données stockées localement sur le courtier. Les champs de conservation d'archivage contrôlent la durée de stockage des données dans le stockage objet d'archivage.
- * Conservation du stockage d'archives.* un an (par exemple). Combien de temps souhaitez-vous que vos données soient stockées dans vos archives ? Entrez -1 avec n'importe quelle unité pour une durée infinie. La calculatrice suppose une rétention de 10 ans pour une rétention infinie.
- **Multiplicateur de croissance.** 1 (par exemple). Si la valeur de ce paramètre est basée sur le débit actuel, réglez-la sur 1. Pour une taille basée sur une croissance supplémentaire, définissez ce paramètre sur un multiplicateur de croissance.

- **Nombre d'instances d'apporteurs.** 10 (par exemple). Combien d'instances d'apporteurs s'exécutent ? Cette entrée est nécessaire pour incorporer la charge CPU dans le calcul du dimensionnement. Une valeur vide indique que la charge CPU n'est pas intégrée au calcul.

Sur la base de cet exemple d'entrée, le dimensionnement a l'effet suivant sur les producteurs :

- Débit moyen en octets non compressés : 1 Gbits/s Débit maximal en octets non compressés : 4 Gbit/s Débit moyen en octets compressés : 400 Mbit/s. Débit maximal en octets compressés : 1,6 Gbit/s Ceci est basé sur un taux de compression par défaut de 60 % (vous pouvez modifier cette valeur).
  - Stockage total des jeux d'accès on-broker requis : 31 104 To, incluant la réplication, compressé. Stockage d'archivage hors courtier total requis : 378 432 To, compressé. Utiliser "https://fusion.netapp.com" Pour le dimensionnement des StorageGRID.

Les processeurs stream doivent décrire leurs applications ou services qui consomment les données d'Apache Kafka et les produisent dans Apache Kafka. Dans la plupart des cas, ces systèmes sont basés sur des flux ksqlDB ou Kafka.

- **Nom.** Spark streamer.
- **Temps de traitement.** combien de temps ce processeur prend-il pour traiter un seul message?
  - 1 ms (transformation simple sans état) [exemple], 10 ms (fonctionnement avec état en mémoire).
  - 100 ms (fonctionnement avec état du réseau ou du disque), 1 000 ms (appels REST tiers).
  - J'ai évalué ce paramètre et je sais exactement combien de temps il prend.
- **Conservation de la sortie.** 1 jour (exemple). Un processeur de flux reproduit son débit vers Apache Kafka. Combien de temps souhaitez-vous que ces données soient stockées dans Apache Kafka ? Entrez -1 avec n'importe quelle unité pour une durée infinie.
- Cochez la case « Activer le stockage à plusieurs niveaux pour réduire le nombre de courtiers et autoriser le stockage Infinite Storage ? ».
- * Conservation du stockage d'archives.* 1 an (par exemple). Combien de temps souhaitez-vous que vos données soient stockées dans vos archives ? Entrez -1 avec n'importe quelle unité pour une durée infinie. La calculatrice suppose une rétention de 10 ans pour une rétention infinie.
- **Pourcentage de Passthrough de sortie.** 100 (par exemple). Un processeur de flux reproduit son débit vers Apache Kafka. Quel pourcentage du débit entrant sera reputé dans Apache Kafka ? Par exemple, si le débit entrant est de 20 Mbit/s et que cette valeur est de 10, le débit de sortie est de 2 Mbit/s.
- À partir de quelles applications est-ce lu ? Sélectionnez « Spark », le nom utilisé dans le dimensionnement basé sur le type d'apporteur. En vous basant sur les données ci-dessus, vous pouvez vous attendre à ce que les effets suivants de dimensionnement sur les instances de processus de flux et les estimations de partition de rubrique :
- Cette application de processeur de flux nécessite le nombre d'instances suivant. Les sujets entrants requièrent probablement aussi ce grand nombre de partitions. Contactez confluent pour confirmer ce paramètre.
  - 1,000 pour le débit moyen sans multiplicateur de croissance
  - 4,000 pour un débit maximal sans multiplicateur de croissance
  - 1,000 pour le débit moyen avec un multiplicateur de croissance
  - 4,000 pour un débit maximal avec un multiplicateur de croissance

Consommateurs

Décrivez vos applications ou services qui consomment les données d'Apache Kafka et qui ne produisent pas de retour dans Apache Kafka, par exemple un client natif ou un connecteur Kafka.

- **Nom.** Spark consumer.
- **Temps de traitement.** combien de temps ce consommateur prend-il pour traiter un seul message?
  - 1 ms (par exemple, une tâche simple avec état, comme la journalisation)
  - 10 ms (écritures rapides vers un datastore)
  - 100 ms (écritures lentes dans un datastore)
  - 1 000 ms (appel REST tiers)
  - Un autre processus de test de durée connue.
- **Type de client.** application, proxy ou évier à un datastore existant (RDBMS, NoSQL, autre).
- À partir de quelles applications est-ce lu ? Connectez ce paramètre avec le dimensionnement du producteur et du flux déterminé précédemment.

En vous basant sur les données ci-dessus, vous devez déterminer le dimensionnement des instances grand public et des estimations de partition de rubrique. Une application client nécessite le nombre d'instances suivant.

- 2,000 pour le débit moyen, pas de multiplicateur de croissance
- 8,000 pour le débit maximal, pas de multiplicateur de croissance
- 2,000 pour le débit moyen, y compris le multiplicateur de croissance
- 8,000 pour le débit maximal, y compris le multiplicateur de croissance

Les rubriques entrantes ont probablement également besoin de ce nombre de partitions. Contactez le confluent pour confirmer.

En plus des exigences des producteurs, des transformateurs de flux et des consommateurs, vous devez fournir les exigences supplémentaires suivantes :

- **Temps de reconstruction.** par exemple, 4 heures. Si un hôte de courtier Apache Kafka échoue, ses données sont perdues et un nouvel hôte est provisionné pour remplacer l'hôte défaillant, à quel rythme ce nouvel hôte doit-il se reconstruire lui-même ? Laissez ce paramètre vide si la valeur est inconnue.
- **Objectif d'utilisation des ressources (pourcentage).** par exemple, 60. De quelle manière souhaitez-vous que vos hôtes soient en débit moyen ? Confluent recommande une utilisation de 60 %, à moins d'utiliser des clusters d'auto-équilibrage fluides, dans lesquels le taux d'utilisation peut être plus élevé.

Décrivez votre environnement

- **Quel environnement votre cluster sera-t-il exécuté ?** Amazon Web Services, Microsoft Azure, plateforme cloud Google, bare-Metal sur site, VMware sur site, OpenStack sur site ou Kubernates sur site ?
- **Détails de l'hôte.** nombre de cœurs : 48 (par exemple), type de carte réseau (10 GbE, 40 GbE, 16 GbE, 1 GbE ou un autre type).
- **Volumes de stockage.** hôte : 12 (par exemple). Combien de disques durs ou SSD sont pris en charge par hôte ? Confluent recommande 12 disques durs par hôte.
- **Capacité de stockage/volume (en Go).** 1000 (par exemple). Quelle quantité de stockage un seul volume

peut-il stocker en gigaoctets ? Le confluent recommande des disques de 1 To.

- **Configuration du stockage.** Comment les volumes de stockage sont-ils configurés ? Confluent recommande RAID10 pour tirer profit de toutes les caractéristiques confluentes. JBOD, SAN, RAID 1, RAID 0, RAID 5, et d'autres types sont également pris en charge.
- **Débit volumique unique (Mbit/s).** 125 (par exemple). Quelle est la vitesse à laquelle un volume de stockage peut-il lire ou écrire en mégaoctets par seconde ? Confluent recommande des disques durs standard dont le débit est généralement de 125 Mbit/s.
- **Capacité de mémoire (Go).** 64 (par exemple).

Une fois les variables d'environnement déterminées, sélectionnez Size My Cluster (taille du cluster). Sur la base des exemples de paramètres indiqués ci-dessus, nous avons déterminé le dimensionnement suivant pour Kafka confluent :

- **Apache Kafka.** Courtier nombre: 22. Votre cluster est lié au stockage. Envisagez d'activer un stockage à plusieurs niveaux afin de réduire le nombre d'hôtes et d'autoriser une capacité de stockage infinie.
- **Apache ZooKeeper.** nombre: 5; Apache Kafka Connect Employés: Count: 2; Schéma Registry: Count: 2; proxy REST: Count: 2; ksqlDB: Count: 2; Confluent Control Center: Count: 1.

Utilisez le mode inverse pour les équipes chargées des plateformes en toute sérénité. Utilisez le mode partitions pour calculer le nombre de partitions requises par une seule rubrique. Voir https://eventsizer.io pour le dimensionnement en fonction des modes inverse et partitions.

Conclusion

Ce document fournit des recommandations sur les meilleures pratiques pour l'utilisation en traitant le stockage à plusieurs niveaux avec un stockage NetApp, notamment des tests de vérification, des résultats de performances de stockage à plusieurs niveaux, un réglage, des connecteurs S3 confluent et la fonctionnalité d'équilibrage automatique. Compte tenu des règles ILM, des performances fluides avec de multiples tests de performances pour la vérification et des API S3 standard, le stockage objet NetApp StorageGRID est la solution idéale pour parler du stockage hiérarchisé.

Où trouver des informations complémentaires

Pour en savoir plus sur les informations données dans ce livre blanc, consultez ces documents et/ou sites web :

- Qu'est-ce que Apache Kafka

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

- Documentation produit NetApp

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

- Détails des paramètres de l'évier 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"

- Stockage infini dans une plateforme fluide

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

- Stockage à plusieurs niveaux confluent : meilleures pratiques et dimensionnement

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

- Connecteur de dissipateur Amazon S3 pour plate-forme de raccordement

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

- Dimensionnement de Kafka

  "https://eventsizer.io"

- Dimensionnement de StorageGRID

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

- Cas d'utilisation de Kafka

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

- Clusters Kafka autoéquilibrant dans la plateforme 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/"