Bonnes pratiques pour Confluent Kafka

NetApp artificial intelligence solutions

NetApp
February 12, 2026

Sommaire

Bonnes pratiques pour Confluent Kafka

TR-4912 : Recommandations de bonnes pratiques pour le stockage hiérarchisé Confluent Kafka avec NetApp

Karthikeyan Nagalingam, Joseph Kandatilparambil, NetApp Rankesh Kumar, Confluent

Apache Kafka est une plateforme de streaming d'événements distribuée par la communauté, capable de gérer des milliards d'événements par jour. Initialement conçu comme une file d'attente de messagerie, Kafka est basé sur une abstraction d'un journal de validation distribué. Depuis sa création et sa publication en open source par LinkedIn en 2011, Kafka est passé d'une simple file d'attente de messages à une plateforme de streaming d'événements à part entière. Confluent fournit la distribution d'Apache Kafka avec la plateforme Confluent. La plateforme Confluent complète Kafka avec des fonctionnalités communautaires et commerciales supplémentaires conçues pour améliorer l'expérience de streaming des opérateurs et des développeurs en production à grande échelle.

Ce document décrit les meilleures pratiques pour l'utilisation du stockage hiérarchisé Confluent sur une offre de stockage d'objets NetApp en fournissant le contenu suivant :

- Vérification confluente avec le stockage d'objets NetApp – NetApp StorageGRID
- Tests de performances de stockage hiérarchisé
- Lignes directrices sur les meilleures pratiques pour Confluent sur les systèmes de stockage NetApp

Pourquoi choisir le stockage hiérarchisé Confluent ?

Confluent est devenu la plateforme de streaming en temps réel par défaut pour de nombreuses applications, en particulier pour les charges de travail de Big Data, d'analyse et de streaming. Le stockage hiérarchisé permet aux utilisateurs de séparer le calcul du stockage sur la plateforme Confluent. Il rend le stockage des données plus rentable, vous permet de stocker des quantités pratiquement infinies de données et d'augmenter (ou de réduire) les charges de travail à la demande, et facilite les tâches administratives telles que le rééquilibrage des données et des locataires. Les systèmes de stockage compatibles S3 peuvent tirer parti de toutes ces fonctionnalités pour démocratiser les données avec tous les événements en un seul endroit, éliminant ainsi le besoin d'une ingénierie de données complexe. Pour plus d'informations sur les raisons pour lesquelles vous devriez utiliser le stockage hiérarchisé pour Kafka, consultez"cet article de Confluent" .

NetApp instaclustr prend également en charge Kafka avec stockage hiérarchisé à partir de la version 3.8.1. Veuillez consulter plus de détails ici "instaclust utilisant le stockage hiérarchisé de Kafka"

Pourquoi NetApp StorageGRID pour le stockage hiérarchisé ?

StorageGRID est une plate-forme de stockage d'objets leader du secteur de NetApp. StorageGRID est une solution de stockage basée sur des objets définie par logiciel qui prend en charge les API d'objets standard du secteur, notamment l'API Amazon Simple Storage Service (S3). StorageGRID stocke et gère les données non structurées à grande échelle pour fournir un stockage d'objets sécurisé et durable. Le contenu est placé au bon endroit, au bon moment et sur le bon niveau de stockage, optimisant ainsi les flux de travail et réduisant les coûts des médias riches distribués à l'échelle mondiale.

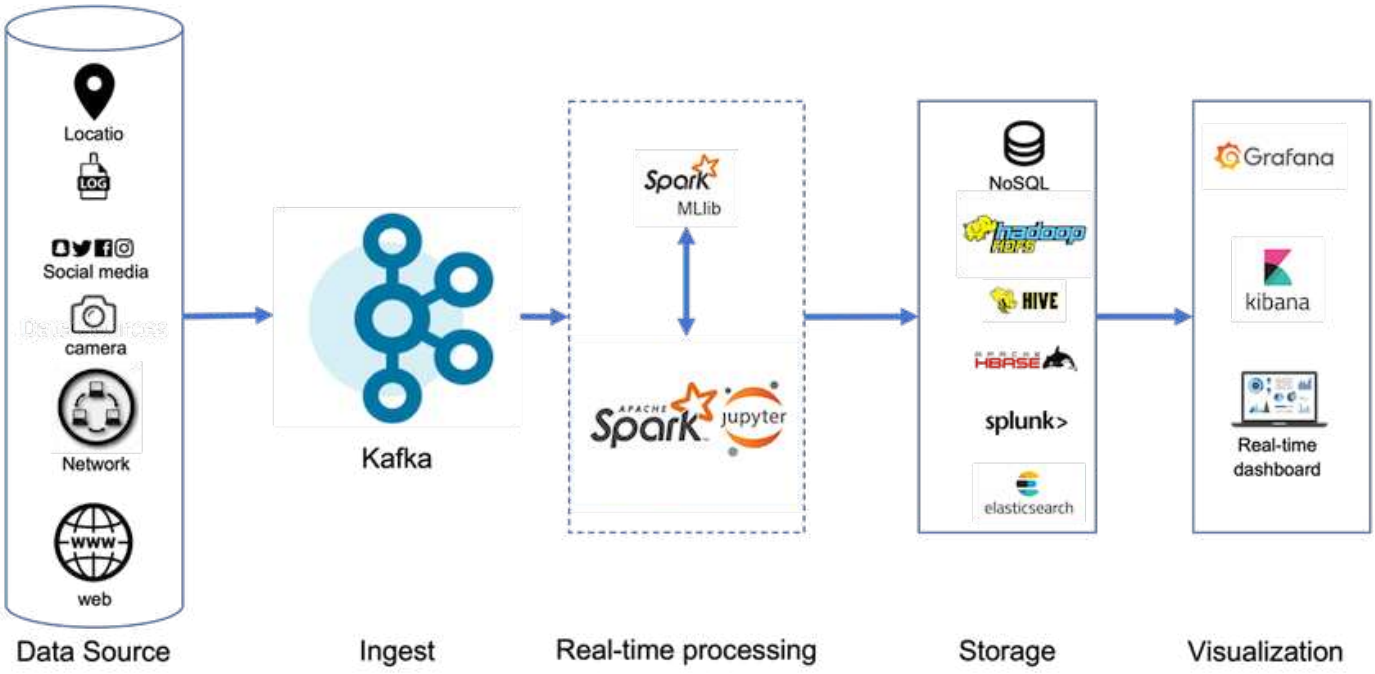
Le principal facteur de différenciation de StorageGRID est son moteur de politique de gestion du cycle de vie des informations (ILM) qui permet une gestion du cycle de vie des données basée sur des politiques. Le moteur de politique peut utiliser les métadonnées pour gérer la manière dont les données sont stockées tout au long de leur durée de vie afin d'optimiser initialement les performances et d'optimiser automatiquement les coûts et la durabilité à mesure que les données vieillissent.

Activation du stockage hiérarchisé Confluent

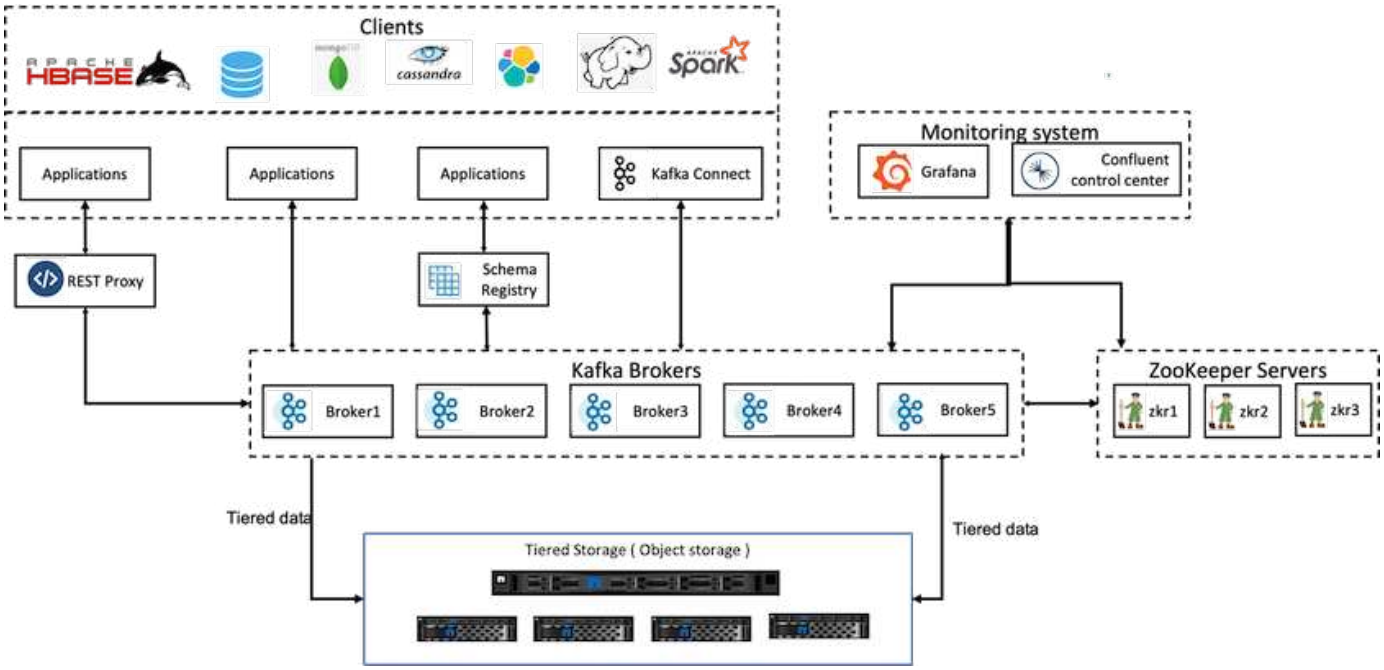
L'idée de base du stockage hiérarchisé est de séparer les tâches de stockage des données du traitement des données. Grâce à cette séparation, il devient beaucoup plus facile pour le niveau de stockage des données et le niveau de traitement des données de s'adapter indépendamment.

Une solution de stockage à plusieurs niveaux pour Confluent doit tenir compte de deux facteurs. Tout d'abord, il doit contourner ou éviter les propriétés courantes de cohérence et de disponibilité du magasin d'objets, telles que les incohérences dans les opérations LIST et l'indisponibilité occasionnelle des objets. Deuxièmement, il doit gérer correctement l'interaction entre le stockage hiérarchisé et le modèle de réplication et de tolérance aux pannes de Kafka, y compris la possibilité que les leaders zombies continuent à hiérarchiser les plages de décalage. Le stockage d'objets NetApp offre à la fois une disponibilité d'objet cohérente et un modèle HA qui rend le stockage fatigué disponible pour les plages de décalage de niveaux. Le stockage d'objets NetApp offre une disponibilité d'objet cohérente et un modèle HA pour rendre le stockage fatigué disponible pour les plages de décalage de niveaux.

Avec le stockage hiérarchisé, vous pouvez utiliser des plates-formes hautes performances pour les lectures et écritures à faible latence près de la fin de vos données en streaming, et vous pouvez également utiliser des magasins d'objets évolutifs et moins chers comme NetApp StorageGRID pour les lectures historiques à haut débit. Nous avons également une solution technique pour Spark avec le contrôleur de stockage NetApp et les détails sont ici. La figure suivante montre comment Kafka s'intègre dans un pipeline d'analyse en temps réel.



La figure suivante illustre comment NetApp StorageGRID s'intègre en tant que niveau de stockage d'objets de Confluent Kafka.



Détails de l’architecture de la solution

Cette section couvre le matériel et les logiciels utilisés pour la vérification Confluent. Ces informations s’appliquent au déploiement de la plateforme Confluent avec le stockage NetApp . Le tableau suivant couvre l’architecture de la solution testée et les composants de base.

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

Aperçu de la technologie

Cette section décrit la technologie utilisée dans cette solution.

NetApp StorageGRID

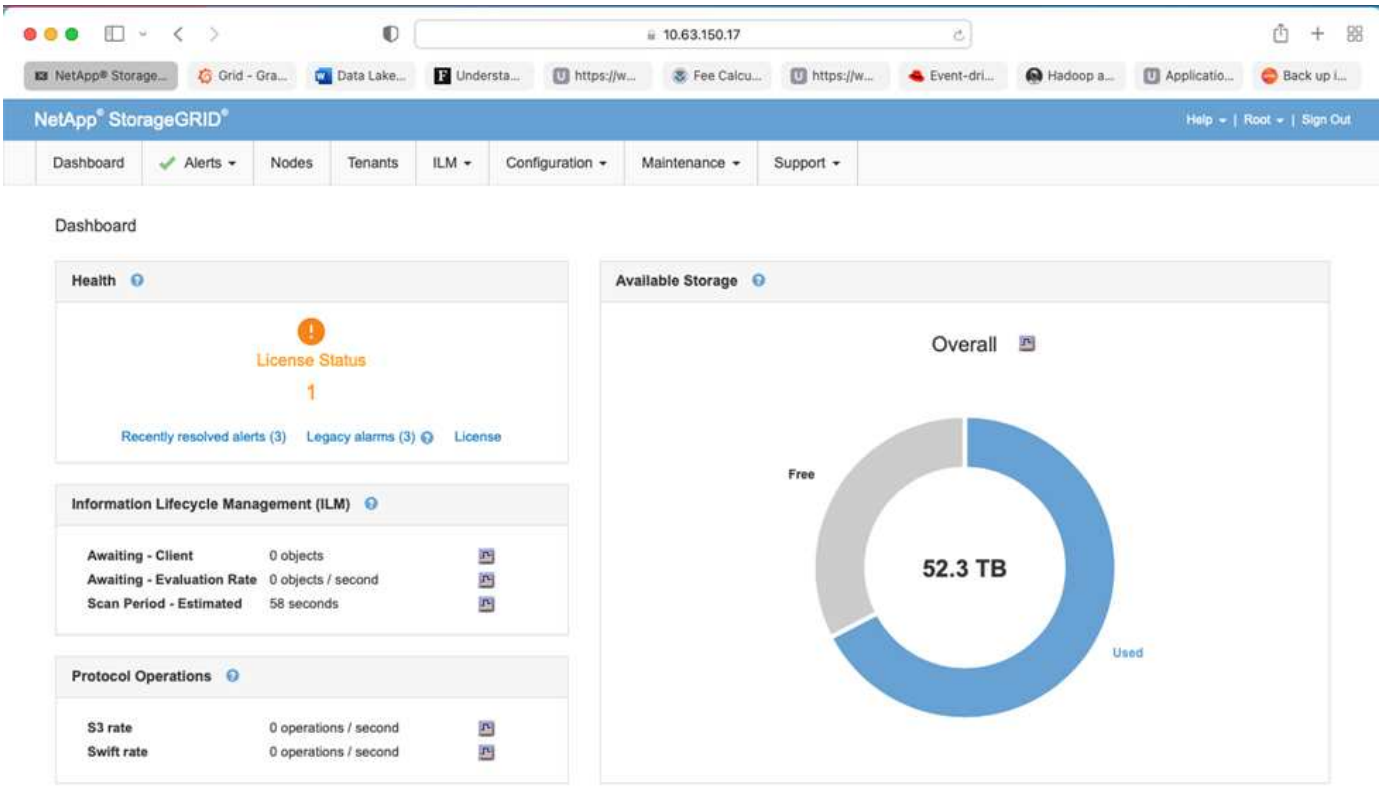
NetApp StorageGRID est une plate-forme de stockage d'objets hautes performances et rentable. En utilisant le stockage hiérarchisé, la plupart des données sur Confluent Kafka, qui sont stockées dans le stockage local ou le stockage SAN du courtier, sont déchargées vers le magasin d'objets distant. Cette configuration entraîne des améliorations opérationnelles significatives en réduisant le temps et le coût nécessaires pour rééquilibrer, étendre ou réduire les clusters ou remplacer un courtier défaillant. Le stockage d'objets joue un rôle important dans la gestion des données qui résident sur le niveau du magasin d'objets, c'est pourquoi il est important de choisir le bon stockage d'objets.

StorageGRID offre une gestion intelligente des données mondiales basée sur des politiques à l'aide d'une architecture de grille distribuée basée sur des nœuds. Il simplifie la gestion de pétaoctets de données non structurées et de milliards d'objets grâce à son espace de noms d'objets global omniprésent combiné à des fonctionnalités de gestion de données sophistiquées. L'accès aux objets par appel unique s'étend sur plusieurs sites et simplifie les architectures à haute disponibilité tout en garantissant un accès continu aux objets, quelles que soient les pannes du site ou de l'infrastructure.

La multilocation permet à plusieurs applications de données cloud et d'entreprise non structurées d'être gérées en toute sécurité au sein de la même grille, augmentant ainsi le retour sur investissement et les cas d'utilisation de NetApp StorageGRID. Vous pouvez créer plusieurs niveaux de service avec des politiques de cycle de vie d'objet basées sur les métadonnées, optimisant la durabilité, la protection, les performances et la localité dans plusieurs zones géographiques. Les utilisateurs peuvent ajuster les politiques de gestion des données et surveiller et appliquer des limites de trafic pour se réaligner sur le paysage des données de manière non perturbatrice à mesure que leurs besoins évoluent dans des environnements informatiques en constante évolution.

Gestion simple avec Grid Manager

StorageGRID Grid Manager est une interface graphique basée sur un navigateur qui vous permet de configurer, de gérer et de surveiller votre système StorageGRID sur des emplacements distribués à l'échelle mondiale dans une seule fenêtre.



Vous pouvez effectuer les tâches suivantes avec l'interface StorageGRID Grid Manager :

- Gérez des référentiels d'objets distribués à l'échelle mondiale et à l'échelle du pétaoctet, tels que des images, des vidéos et des enregistrements.
- Surveillez les nœuds et les services de la grille pour garantir 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, comment elles sont protégées contre la perte, où les données de l'objet sont stockées et pendant combien de temps.
- Surveiller les transactions, les performances et les opérations au sein du système.

Politiques de gestion du cycle de vie de l'information

StorageGRID dispose de politiques de gestion des données flexibles qui incluent la conservation de copies de réplique de vos objets et l'utilisation de schémas EC (codage d'effacement) tels que 2+1 et 4+2 (entre autres) pour stocker vos objets, en fonction des exigences spécifiques en matière de performances et de protection des données. À mesure que les charges de travail et les exigences évoluent au fil du temps, il est courant que les politiques ILM doivent également évoluer au fil du temps. La modification des politiques ILM est une fonctionnalité essentielle, permettant aux clients de StorageGRID de s'adapter rapidement et facilement à leur environnement en constante évolution.

Performances

StorageGRID augmente les performances en ajoutant davantage de nœuds de stockage, qui peuvent être des machines virtuelles, du matériel nu ou des appareils spécialement conçus comme le"SG5712, SG5760, SG6060 ou SGF6024" . Lors de nos tests, nous avons dépassé les exigences de performances clés d'Apache Kafka avec une grille 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 les performances et la capacité.

Configuration de l’équilibreur de charge et du point de terminaison

Les nœuds d’administration de StorageGRID fournissent l’interface utilisateur de Grid Manager et le point de terminaison de l’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 point de terminaison S3 hautement disponible pour le stockage hiérarchisé Confluent Kafka, nous avons implémenté l’équilibreur de charge StorageGRID , qui s’exécute en tant que service sur les nœuds d’administration et les nœuds de passerelle. De plus, l’équilibreur de charge gère également le trafic local et communique avec le GSLB (Global Server Load Balancing) pour faciliter la reprise après sinistre.

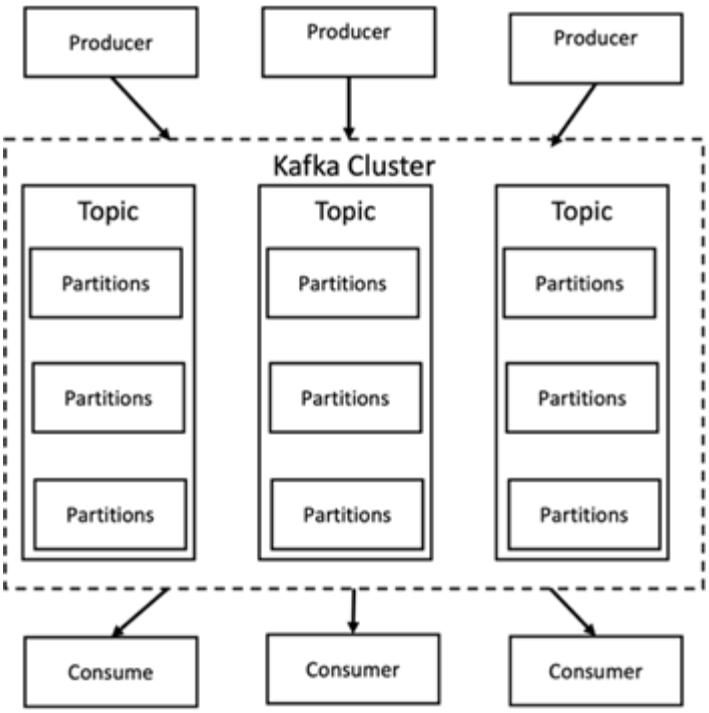
Pour améliorer davantage la configuration des points de terminaison, StorageGRID fournit des stratégies de classification du trafic intégrées au nœud d’administration, vous permet de surveiller le trafic de votre charge de travail et applique diverses limites de qualité de service (QoS) à vos charges de travail. Les stratégies de classification du trafic sont appliquées aux points de terminaison sur le service StorageGRID Load Balancer pour les nœuds de passerelle et les nœuds d’administration. Ces politiques peuvent aider à réguler et à surveiller le trafic.

Classification du trafic dans StorageGRID

StorageGRID dispose d’une fonctionnalité QoS intégrée. Les politiques de classification du trafic peuvent aider à surveiller différents types de trafic S3 provenant d’une application cliente. Vous pouvez ensuite créer et appliquer des politiques pour limiter ce trafic en fonction de la bande passante entrante/sortante, du nombre de requêtes simultanées en lecture/écriture ou du taux de requêtes en lecture/écriture.

Apache Kafka

Apache Kafka est une implémentation framework d’un bus logiciel utilisant le traitement de flux écrit en Java et Scala. Son objectif est de fournir une plate-forme unifiée, à haut débit et à faible latence pour gérer les 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 et fournit des flux Kafka, une bibliothèque de traitement de flux Java. Kafka utilise un protocole binaire basé sur TCP, optimisé pour l’efficacité et s’appuyant sur une abstraction « d’ensemble de messages » qui regroupe naturellement les messages pour réduire la surcharge de l’aller-retour réseau. Cela permet des opérations de disque séquentielles plus importantes, des paquets réseau plus volumineux et des blocs de mémoire contigus, permettant ainsi à Kafka de transformer un flux rafaleux d’écritures de messages aléatoires en écritures linéaires. La figure suivante illustre le flux de données de base d’Apache Kafka.



Kafka stocke les messages clé-valeur provenant d'un nombre arbitraire de processus appelés producteurs. Les données peuvent être partitionnées en différentes partitions au sein de différents sujets. Au sein d'une partition, les messages sont strictement ordonnés par leurs décalages (la position d'un message au sein d'une partition) et indexés et stockés avec un horodatage. D'autres processus appelés consommateurs peuvent lire les messages des partitions. Pour le traitement des flux, Kafka propose l'API Streams qui permet d'écrire des applications Java qui consomment des données de Kafka et réécrivent les résultats dans Kafka. Apache Kafka fonctionne également avec des systèmes de traitement de flux externes tels qu'Apache Apex, Apache Flink, Apache Spark, Apache Storm et Apache NiFi.

Kafka s'exécute sur un cluster d'un ou plusieurs serveurs (appelés courtiers), et les partitions de tous les sujets sont réparties sur les nœuds du cluster. De plus, les partitions sont répliquées sur plusieurs courtiers. Cette architecture permet à Kafka de diffuser des flux massifs de messages de manière tolérante aux pannes et lui a permis de remplacer certains des systèmes de messagerie conventionnels tels que Java Message Service (JMS), Advanced Message Queuing Protocol (AMQP), etc. Depuis la version 0.11.0.0, Kafka propose des écritures transactionnelles, qui fournissent un traitement de flux une seule fois à l'aide de l'API Streams.

Kafka prend en charge deux types de sujets : réguliers et compactés. Les sujets réguliers peuvent être configurés avec un temps de rétention ou une limite d'espace. S'il existe des enregistrements plus anciens que la durée de conservation spécifiée ou si l'espace limité est dépassé pour une partition, Kafka est autorisé à supprimer les anciennes données pour libérer de l'espace de stockage. 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 ultérieurs comme des mises à jour d'un message plus ancien avec la même clé et garantit de ne jamais supprimer le dernier message par clé. Les utilisateurs peuvent supprimer entièrement les messages en écrivant un message dit « tombstone » avec la valeur nulle pour une clé spécifique.

Il existe cinq API principales dans Kafka :

- **API du producteur.** Permet à une application de publier des flux d'enregistrements.
- **API consommateur.** Permet à une application de s'abonner à des sujets et de traiter des flux d'enregistrements.

- **API du connecteur.** Exécute les API de production et de consommation réutilisables qui peuvent lier les sujets aux applications existantes.
- **API de flux.** Cette API convertit les flux d'entrée en sortie et produit le résultat.
- **API d'administration.** Utilisé pour gérer les sujets Kafka, les courtiers et autres objets Kafka.

Les API consommateur et producteur s'appuient sur le protocole de messagerie Kafka et offrent 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 consommateurs ou producteurs dans n'importe quel langage de programmation. Cela 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 de l'activité du site Web, les métriques, l'agrégation de journaux, le traitement de flux, l'approvisionnement d'événements et la journalisation des validations.

- Kafka a amélioré le débit, le partitionnement intégré, la réplication et la tolérance aux pannes, ce qui en fait une bonne solution pour les applications de traitement de messages à grande échelle.
- Kafka peut reconstruire les activités d'un utilisateur (pages vues, recherches) dans un pipeline de suivi sous la forme d'un ensemble de flux de publication-abonnement en temps réel.
- Kafka est souvent utilisé pour les données de surveillance opérationnelle. Il s'agit d'agréger des statistiques provenant d'applications distribuées pour produire des flux centralisés de données opérationnelles.
- De nombreuses personnes utilisent Kafka comme solution de remplacement pour une solution d'agrégation de journaux. L'agrégation de journaux collecte généralement les fichiers journaux physiques des serveurs et les place dans un emplacement central (par exemple, un serveur de fichiers ou HDFS) pour traitement. Kafka résume les détails des fichiers et fournit une abstraction plus propre des données de journal ou d'événement sous forme de flux de messages. Cela permet un traitement à faible latence et une prise en charge plus facile de plusieurs sources de données et d'une consommation de données distribuée.
- De nombreux utilisateurs de Kafka traitent les données dans des pipelines de traitement composés de plusieurs étapes, dans lesquels les données d'entrée brutes sont consommées à partir des rubriques Kafka, puis agrégées, enrichies ou transformées d'une autre manière en de nouvelles rubriques pour une consommation ultérieure ou un traitement de suivi. Par exemple, un pipeline de traitement pour recommander des articles d'actualité peut explorer le contenu des articles à partir de flux RSS et le publier dans une rubrique « articles ». Un traitement ultérieur pourrait normaliser ou dédupliquer ce contenu et publier le contenu de l'article nettoyé dans une nouvelle rubrique, et une étape de traitement finale pourrait tenter de recommander ce contenu aux utilisateurs. Ces pipelines de traitement créent des graphiques de flux de données en temps réel en fonction des sujets individuels.
- L'approvisionnement d'événements est un style de conception d'application pour lequel les changements d'état sont enregistrés sous la forme d'une séquence d'enregistrements ordonnée dans le temps. La prise en charge par Kafka de données de journaux stockées très volumineuses en fait un excellent backend pour une application construite dans ce style.
- Kafka peut servir de sorte de journal de validation externe pour un système distribué. Le journal permet de répliquer les données entre les nœuds et agit comme un mécanisme de resynchronisation pour les nœuds défaillants afin de restaurer leurs données. La fonctionnalité de compactage des journaux dans Kafka permet de prendre en charge ce cas d'utilisation.

Confluent

Confluent Platform est une plateforme prête pour l'entreprise qui complète Kafka avec des fonctionnalités avancées conçues pour aider à accélérer le développement et la connectivité des applications, permettre les transformations grâce au traitement des flux, simplifier les opérations d'entreprise à grande échelle et répondre aux exigences architecturales strictes. Conçu par les créateurs originaux d'Apache Kafka, Confluent étend les avantages de Kafka avec des fonctionnalités de niveau entreprise tout en supprimant le fardeau de la gestion ou de la surveillance de Kafka. Aujourd'hui, plus de 80 % des entreprises du Fortune 100 utilisent la technologie de streaming de données, et la plupart d'entre elles utilisent Confluent.

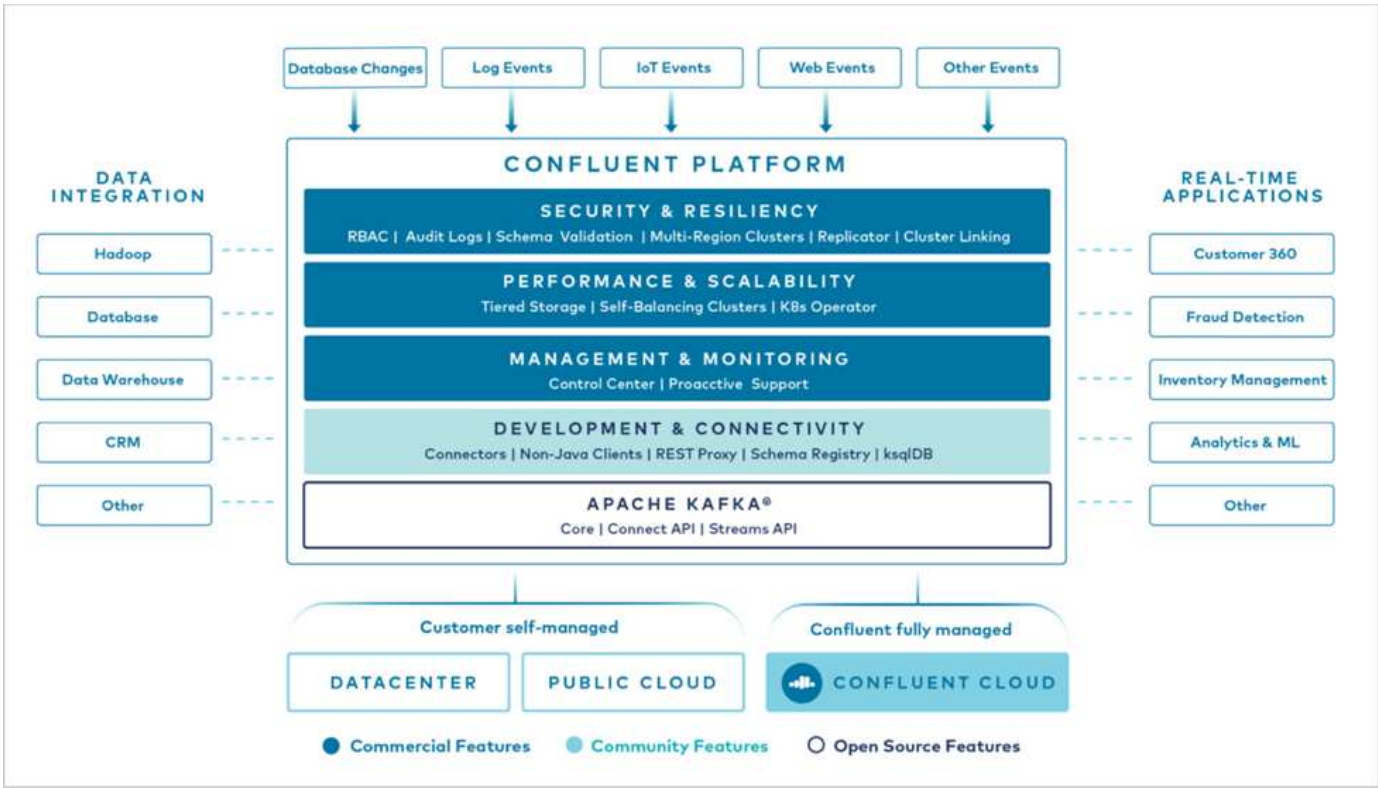
Pourquoi Confluent ?

En intégrant des données historiques et en temps réel dans une source unique et centrale de vérité, Confluent facilite la création d'une toute nouvelle catégorie d'applications modernes axées sur les événements, l'obtention d'un pipeline de données universel et le déblocage de nouveaux cas d'utilisation puissants avec une évolutivité, des performances et une fiabilité complètes.

À quoi sert Confluent ?

Confluent Platform vous permet de vous concentrer sur la manière de tirer profit de vos données plutôt que de vous soucier des mécanismes sous-jacents, tels que la manière dont les données sont transportées ou intégrées entre des systèmes disparates. Plus précisément, Confluent Platform simplifie la connexion des sources de données à Kafka, la création d'applications de streaming, ainsi que la sécurisation, la surveillance et la gestion de votre infrastructure Kafka. Aujourd'hui, Confluent Platform est utilisé pour un large éventail de cas d'utilisation dans de nombreux secteurs, des services financiers, de la vente au détail omnicanal et des voitures autonomes, à la détection de fraude, aux microservices et à l'IoT.

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



Présentation de la technologie de streaming d'événements de Confluent

Au cœur de la plateforme Confluent se trouve "Apache Kafka" , la plateforme de streaming distribuée open source la plus populaire. Les principales fonctionnalités de Kafka sont les suivantes :

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

Prêt à l'emploi, Confluent Platform inclut également Schema Registry, REST Proxy, un total de plus de 100 connecteurs Kafka prédéfinis et ksqlDB.

Aperçu des fonctionnalités d'entreprise de la plateforme Confluent

- **Centre de contrôle Confluent.** Un système basé sur une interface graphique pour la gestion et la surveillance de Kafka. Il vous permet de gérer facilement Kafka Connect et de créer, modifier et gérer des connexions à d'autres systèmes.
- **Confluent pour Kubernetes.** Confluent pour Kubernetes est un opérateur Kubernetes. Les opérateurs Kubernetes étendent les capacités d'orchestration de Kubernetes en fournissant les fonctionnalités et les exigences uniques pour une application de plate-forme spécifique. Pour Confluent Platform, cela inclut la simplification considérable du processus de déploiement de Kafka sur Kubernetes et l'automatisation des tâches typiques du cycle de vie de l'infrastructure.
- **Connecteurs confluents vers Kafka.** Les connecteurs utilisent l'API Kafka Connect pour connecter Kafka à d'autres systèmes tels que des bases de données, des magasins de valeurs clés, des index de recherche et des systèmes de fichiers. Confluent Hub propose des connecteurs téléchargeables pour les sources et récepteurs de données les plus populaires, y compris des versions entièrement testées et prises en charge de ces connecteurs avec Confluent Platform. Plus de détails peuvent être trouvés "ici" .
- **Clusters auto-équilibrés.** Fournit un équilibrage de charge automatisé, une détection des pannes et une auto-réparation. Il fournit un support pour l'ajout ou la désactivation de courtiers selon les besoins, sans réglage manuel.
- **Liaison de cluster confluent.** Connecte directement les clusters entre eux et reflète les sujets d'un cluster à un autre via un pont de liaison. La liaison de cluster simplifie la configuration des déploiements multi-centres de données, multi-clusters et cloud hybride.
- **Équilibreur automatique de données Confluent.** Surveille votre cluster pour le nombre de courtiers, la taille des partitions, le nombre de partitions et le nombre de leaders au sein du cluster. Il vous permet de déplacer les données pour créer une charge de travail uniforme sur votre cluster, tout en limitant le trafic de rééquilibrage pour minimiser l'effet sur les charges de travail de production lors du rééquilibrage.
- **Réplicateur confluent.** Il est plus facile que jamais de maintenir plusieurs clusters Kafka dans plusieurs centres de données.
- **Stockage à plusieurs niveaux.** Fournit des options pour stocker de grands volumes de données Kafka à l'aide de votre fournisseur de cloud préféré, réduisant ainsi la charge et les coûts opérationnels. Avec le stockage hiérarchisé, vous pouvez conserver les données sur un stockage d'objets rentable et faire évoluer les courtiers uniquement lorsque vous avez besoin de davantage de ressources de calcul.
- **Client JMS confluent.** Confluent Platform inclut un client compatible JMS pour Kafka. Ce client Kafka implémente l'API standard JMS 1.1, en utilisant les courtiers Kafka comme backend. 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 d'appareils et de passerelles MQTT sans avoir besoin d'un courtier MQTT au milieu.

- **Plugins de sécurité Confluent.** Les plugins de sécurité Confluent sont utilisés pour ajouter des fonctionnalités de sécurité à divers outils et produits de la plateforme Confluent. Actuellement, il existe un plugin disponible pour le proxy REST Confluent qui permet d'authentifier les requêtes entrantes et de propager le principal authentifié aux requêtes vers Kafka. Cela permet aux clients proxy Confluent REST d'utiliser les fonctionnalités de sécurité multilocataire du courtier Kafka.

Vérification confluente

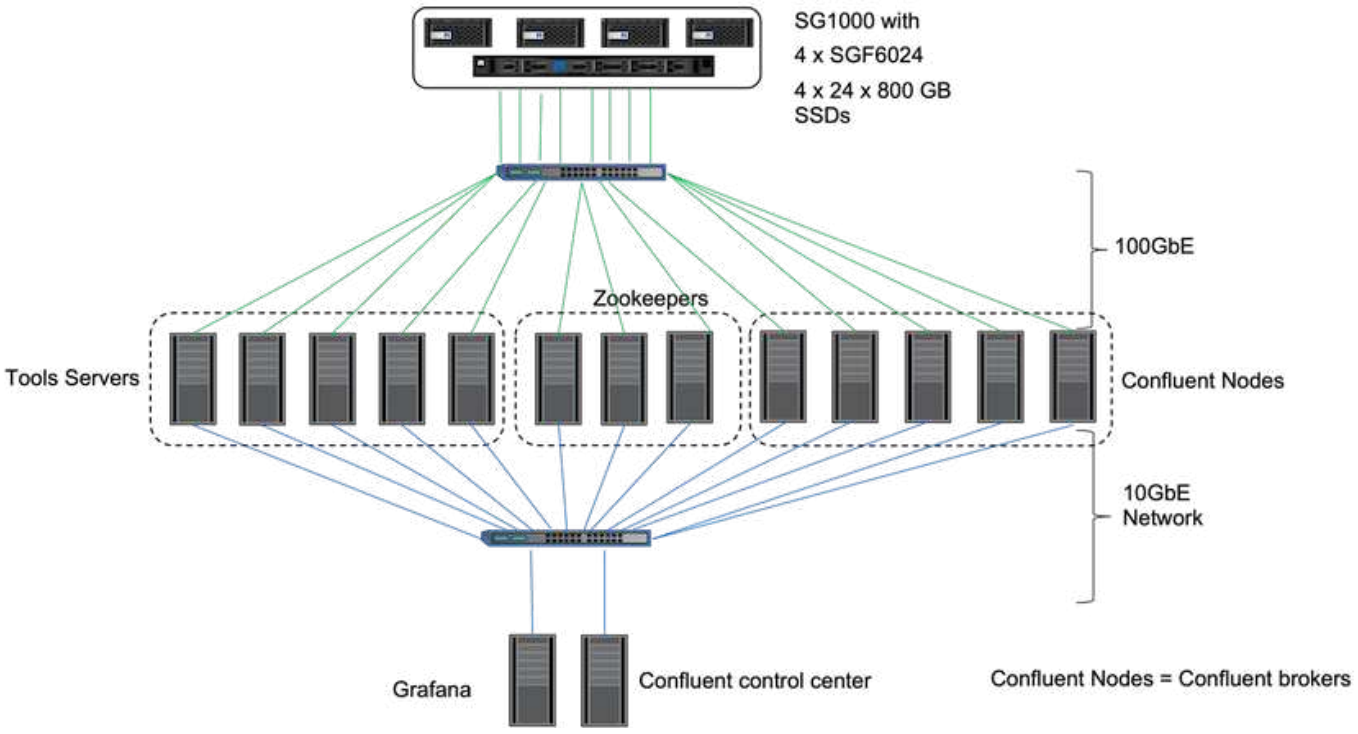
Nous avons effectué une vérification avec Confluent Platform 6.2 Tiered Storage dans NetApp StorageGRID. Les équipes NetApp et Confluent ont travaillé ensemble sur cette vérification et ont exécuté les cas de test requis pour la vérification.

Configuration de la plateforme Confluent

Nous avons utilisé la configuration suivante pour la vérification.

Pour la vérification, nous avons utilisé trois gardiens de zoo, 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 avec quatre SGF6024. Le stockage et les courtiers étaient connectés via des connexions 100 GbE.

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



Les serveurs d'outils agissent comme des clients d'application qui envoient des requêtes aux nœuds Confluent.

Configuration de stockage hiérarchisé Confluent

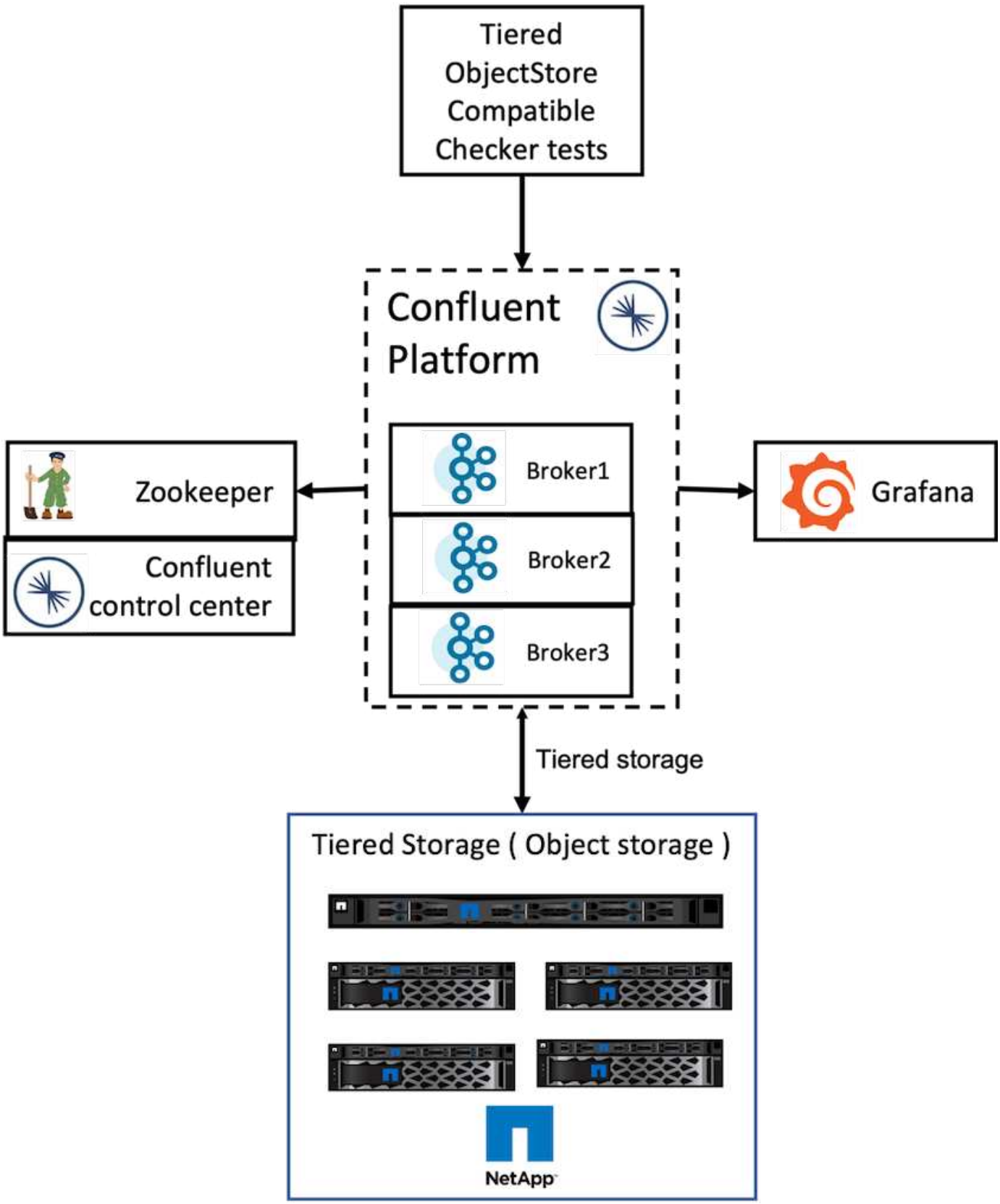
La configuration du stockage hiérarchisé nécessite les paramètres suivants dans Kafka :

```
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
```

Pour la 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 d'objets NetApp - StorageGRID

Nous avons configuré une configuration à site unique dans StorageGRID pour vérification.



Tests de vérification

Nous avons réalisé les cinq cas de test suivants pour la vérification. Ces tests sont exécutés sur le framework Trogdor. Les deux premiers étaient des tests de fonctionnalité et les trois autres étaient des tests de performance.

Test d’exactitude du magasin d’objets

Ce test détermine si toutes les opérations de base (par exemple, obtenir/mettre/supprimer) sur l’API du magasin d’objets fonctionnent bien en fonction des besoins du stockage hiérarchisé. Il s’agit d’un test de base que chaque service de magasin d’objets doit s’attendre à réussir avant les tests suivants. Il s’agit d’un test assertif qui réussit ou échoue.

Test d’exactitude des fonctionnalités de hiérarchisation

Ce test détermine si la fonctionnalité de stockage hiérarchisé de bout en bout fonctionne bien avec un test assertif qui réussit ou échoue. Le test crée une rubrique de test qui, par défaut, est configurée avec la hiérarchisation activée et une taille de hotset fortement réduite. Il produit un flux d’événements vers la rubrique de test nouvellement créée, il attend que les courtiers archivent les segments dans le magasin d’objets, puis il consomme le flux d’événements et valide que le flux consommé correspond au flux produit. Le nombre de messages produits dans le flux d’événements est configurable, ce qui permet à l’utilisateur de générer une charge de travail suffisamment importante en fonction des besoins des tests. La taille réduite du hotset garantit que les récupérations du consommateur en dehors du segment actif sont servies uniquement 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 fautes dans le magasin d’objets. Nous avons simulé une défaillance de nœud en arrêtant le service du gestionnaire de services dans l’un des nœuds de StorageGRID et en validant que la fonctionnalité de bout en bout fonctionne avec le stockage d’objets.

Benchmark de récupération de niveaux

Ce test a validé les performances de lecture du stockage d’objets hiérarchisé et a vérifié la plage de requêtes de lecture d’extraction sous une charge importante à partir des segments générés par le benchmark. Dans cette référence, Confluent a développé des clients personnalisés pour répondre aux demandes de récupération de niveau.

Benchmark de la charge de travail de production et de consommation

Ce test a généré indirectement une charge de travail d’écriture sur le magasin d’objets via l’archivage des segments. La charge de travail de lecture (segments lus) a été générée à partir du stockage d’objets lorsque les groupes de consommateurs ont récupéré les segments. Cette charge de travail a été générée par le script de test. Ce test a vérifié les performances de lecture et d’écriture sur le stockage d’objets dans des threads parallèles. Nous avons testé avec et sans injection de pannes de magasin d’objets comme nous l’avons fait pour le test d’exactitude de la fonctionnalité de hiérarchisation.

Benchmark de la charge de travail de rétention

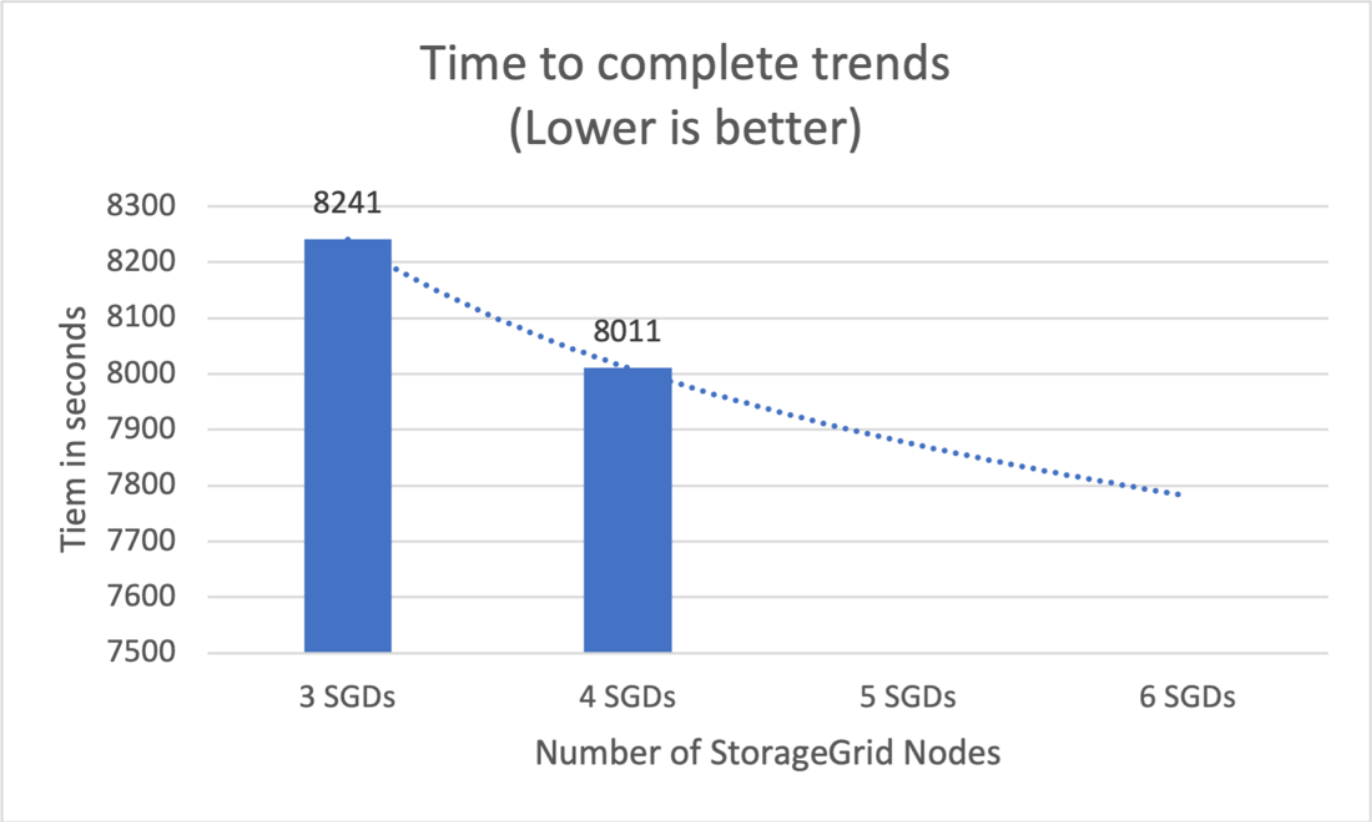
Ce test a vérifié les performances de suppression d’un magasin d’objets sous une charge de travail de rétention de sujets importante. La charge de travail de rétention a été générée à l’aide d’un script de test qui produit de nombreux messages en parallèle avec une rubrique de test. Le sujet de test consistait à configurer un paramètre de rétention agressif basé sur la taille et le temps, ce qui entraînait la purge continue du flux d’événements du magasin d’objets. Les segments ont ensuite été archivés. Cela a conduit à un grand nombre de suppressions dans le stockage d’objets par le courtier et à la collecte des performances des opérations de suppression du magasin d’objets.

Tests de performance avec évolutivité

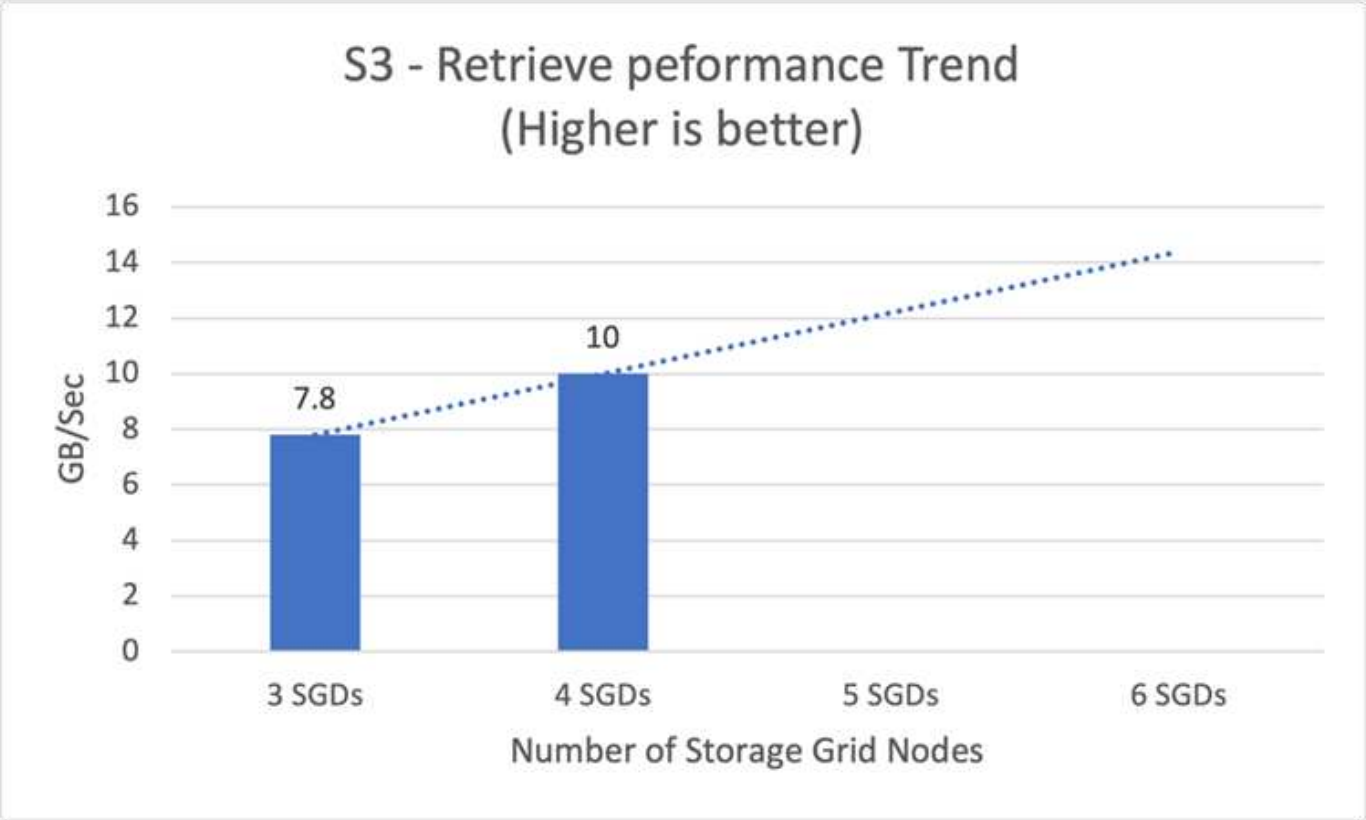
Nous avons effectué les tests de stockage hiérarchisé avec trois à quatre nœuds pour les charges de travail des producteurs et des consommateurs avec la configuration NetApp StorageGRID . Selon nos tests, le temps d’exécution et les résultats de performance

étaient directement proportionnels au nombre de nœuds StorageGRID . La configuration de StorageGRID nécessitait un minimum de trois nœuds.

- Le temps nécessaire pour terminer l'opération de production et de consommation a diminué linéairement lorsque le nombre de nœuds de stockage a augmenté.



- Les performances de l'opération de récupération s3 ont augmenté linéairement en fonction du nombre de nœuds StorageGRID . StorageGRID prend en charge jusqu'à 200 nœuds StorgeGRID.



Connecteur Confluent s3

Le connecteur Amazon S3 Sink exporte les données des rubriques Apache Kafka vers des objets S3 aux formats Avro, JSON ou Bytes. Le connecteur de récepteur Amazon S3 interroge périodiquement les données de Kafka et les télécharge à son tour vers S3. Un partitionneur est utilisé pour diviser les données de chaque partition Kafka en morceaux. Chaque bloc de données est représenté sous la forme d'un objet S3. Le nom de la clé code le sujet, la partition Kafka et le décalage de début de ce bloc de données.

Dans cette configuration, nous vous montrons comment lire et écrire des rubriques dans le stockage d'objets à partir de Kafka directement à l'aide du connecteur de récepteur Kafka s3. Pour ce test, nous avons utilisé un cluster Confluent autonome, mais cette configuration est applicable à un cluster distribué.

1. Téléchargez Confluent Kafka depuis le site Web de Confluent.
2. Décompressez le package dans un dossier sur 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 Confluent Kafka autonome, le cluster crée un dossier racine temporaire dans `/tmp` Il crée également Zookeeper, Kafka, un registre de schémas, connect, un serveur ksql et des dossiers de 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 Zookeeper. Vous n'avez rien à modifier 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 gardiens de zoo pour la sélection du chef Kafka.

6. Nous avons créé un fichier myid dans `/tmp/confluent.406980/zookeeper/data` avec un identifiant unique :

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

Nous avons utilisé le dernier numéro d'adresses IP pour le fichier myid. Nous avons utilisé des valeurs par défaut pour les configurations Kafka, connect, control-center, Kafka, Kafka-rest, ksql-server et schema-registry.

7. Démarrez 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, les services prennent plus de temps à démarrer. Assurez-vous que tous les services sont opérationnels.

8. Installer Kafka Connect en utilisant `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. Mettre 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 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` déposer.

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

13. Vérifiez que le bucket 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-sink pour la configuration s3 et 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. Importez quelques enregistrements dans le 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. Chargez le connecteur 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. Vérifiez l’état du 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. Vérifiez le journal pour vous assurer que s3-sink est prêt à accepter des sujets.

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

19. Consultez 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. Vérifiez les objets dans le 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. Pour vérifier le contenu, copiez chaque fichier de S3 vers votre système de fichiers local en exécutant 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:~#
```

Connecteurs Instaclustr Kafka Connect

Instaclustr prend en charge les connecteurs Kafka Connect et leurs détails - "Plus de détails". Instaclustr fournit des connecteurs supplémentaires "leurs détails"

Clusters auto-équilibrés confluents

Si vous avez déjà géré un cluster Kafka, vous connaissez probablement les défis liés à la réaffectation manuelle de partitions à différents courtiers pour garantir que la charge de travail est équilibrée sur l'ensemble du cluster. Pour les organisations disposant de déploiements Kafka importants, la réorganisation de grandes quantités de données peut être intimidante, fastidieuse et risquée, en particulier si des applications critiques sont construites sur le cluster. Cependant, même pour les plus petits cas d'utilisation de Kafka, le processus prend du temps et est sujet à des erreurs humaines.

Dans notre laboratoire, nous avons testé la fonctionnalité de clusters auto-équilibrés de Confluent, qui automatise le rééquilibrage en fonction des changements de topologie du cluster ou d'une charge inégale. Le test de rééquilibrage Confluent permet de mesurer le temps nécessaire pour ajouter un nouveau courtier lorsque la défaillance du nœud ou le nœud de mise à l'échelle nécessite un rééquilibrage des données entre les courtiers. Dans les configurations Kafka classiques, la quantité de données à rééquilibrer augmente à mesure que le cluster se développe, mais, dans le stockage hiérarchisé, le rééquilibrage est limité à une petite quantité de données. D'après notre validation, le rééquilibrage du stockage hiérarchisé prend quelques secondes ou minutes dans une architecture Kafka classique et augmente de manière linéaire à mesure que le cluster se développe.

Dans les clusters à équilibrage automatique, les rééquilibrages de partition sont entièrement automatisés pour optimiser le débit de Kafka, accélérer la mise à l'échelle du courtier et réduire la charge opérationnelle liée à l'exécution d'un grand cluster. À l'état stable, les clusters auto-équilibrés surveillent l'asymétrie des données entre les courtiers et réaffectent en permanence les partitions pour optimiser les performances du cluster. Lors de la mise à l'échelle de la plate-forme vers le haut ou vers le bas, les clusters auto-équilibrés reconnaissent automatiquement la présence de nouveaux courtiers ou la suppression d'anciens courtiers et déclenchent une réaffectation de partition ultérieure. Cela vous permet d'ajouter et de désactiver facilement des courtiers, rendant vos clusters Kafka fondamentalement plus élastiques. Ces avantages sont offerts sans aucune intervention manuelle, sans calculs complexes ni risque d'erreur humaine que les réaffectations de partitions impliquent généralement. Par conséquent, les rééquilibrages de données sont effectués en beaucoup moins de temps et vous êtes libre de vous concentrer sur des projets de diffusion d'événements à plus forte valeur ajoutée plutôt que de devoir superviser en permanence vos clusters.

Instaclustr prend également en charge les fonctions d'auto-rééquilibrage et a été mis en œuvre chez plusieurs clients.

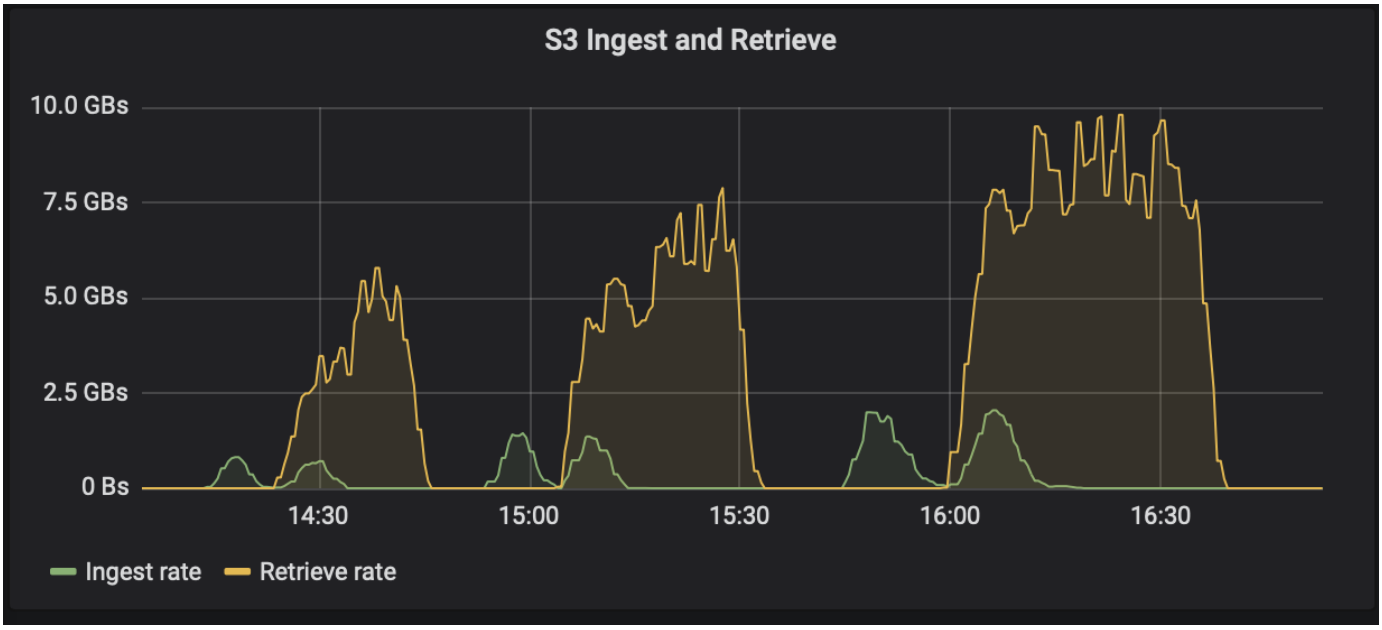
Lignes directrices sur les meilleures pratiques

Cette section présente les enseignements tirés de cette certification.

- Sur la base de notre validation, le stockage d'objets S3 est le meilleur moyen pour Confluent de conserver les données.
- Nous pouvons utiliser un SAN à haut débit (en particulier FC) pour conserver les données chaudes du courtier ou le disque local, car, dans la configuration de stockage hiérarchisé Confluent, la taille des données conservées dans le répertoire de données des courtiers est basée sur la taille du segment et le temps de rétention lorsque les données sont déplacées vers le stockage d'objets.

- Les magasins d'objets offrent de meilleures performances lorsque segment.bytes 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 dans le sujet est contrôlée par le `length.key.value` paramètre. Pour StorageGRID, les performances d'ingestion et de récupération d'objets S3 ont été augmentées à des valeurs plus élevées. Par exemple, 512 octets ont fourni une récupération de 5,8 Gbit/s, 1 024 octets ont fourni une récupération S3 de 7,5 Gbit/s et 2 048 octets ont fourni près de 10 Gbit/s.

La figure suivante présente l'objet S3 ingéré et récupéré en fonction de `length.key.value` .



- **Réglage Kafka.** Pour améliorer les performances du stockage hiérarchisé, vous pouvez augmenter TierFetcherNumThreads et TierArchiverNumThreads. En règle générale, vous souhaitez augmenter TierFetcherNumThreads pour qu'il corresponde au nombre de cœurs de processeur physiques et augmenter TierArchiverNumThreads à la moitié du nombre de cœurs de processeur. Par exemple, dans les propriétés du serveur, si vous disposez d'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 sujets.** Lorsqu'une rubrique est supprimée, la suppression des fichiers de segment de journal dans le stockage d'objets ne commence pas immédiatement. Il existe plutôt un intervalle de temps avec une valeur par défaut de 3 heures avant que la suppression de ces fichiers n'ait lieu. 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 dans le compartiment correspondant.
- **ACL sur les sujets internes du stockage hiérarchisé.** Une bonne pratique recommandée pour les déploiements sur site consiste à activer un autorisateur ACL sur les rubriques internes utilisées pour le stockage hiérarchisé. Définissez des règles ACL pour limiter l'accès à ces données à l'utilisateur du courtier uniquement. Cela sécurise les sujets internes et empêche l'accès non autorisé aux données et 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 courtier principal réel dans votre déploiement.

Par exemple, la commande `confluent-tier-state` définit les ACL sur le sujet interne pour le stockage hiérarchisé. Actuellement, il n'existe qu'un seul sujet interne lié au stockage hiérarchisé. L'exemple crée une ACL qui fournit l'autorisation principale Kafka pour toutes les opérations sur la rubrique interne.

Dimensionnement

Le dimensionnement de Kafka peut être effectué avec quatre modes de configuration : simple, granulaire, inversé et partitions.

Simple

Le mode simple convient aux nouveaux utilisateurs d'Apache Kafka ou aux premiers cas d'utilisation. Pour ce mode, vous fournissez des exigences telles que le débit en Mo/s, la diffusion en lecture, la rétention et le pourcentage d'utilisation des ressources (60 % par défaut). Vous entrez également dans l'environnement, tel que sur site (bare-metal, VMware, Kubernetes ou OpenStack) ou dans le cloud. Sur la base de ces informations, le dimensionnement d'un cluster Kafka fournit le nombre de serveurs requis pour le courtier, le zookeeper, les travailleurs de connexion Apache Kafka, le registre de schémas, un proxy REST, ksqlDB et le centre de contrôle Confluent.

Pour le stockage hiérarchisé, envisagez le mode de configuration granulaire pour dimensionner un cluster Kafka. Le mode granulaire convient aux utilisateurs expérimentés d'Apache Kafka 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.** Étincelle.
- **Type de producteur.** Application ou service, proxy (REST, MQTT, autre) et base de données existante (RDBMS, 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 compressé (max 1 Mo ; 1000 par exemple).
- **Format du message.** Les options incluent Avro, JSON, les tampons de protocole, le binaire, le 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 un temps infini. Le calculateur

suppose une durée 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 un stockage infini ? »
- Lorsque le stockage hiérarchisé est activé, les champs de rétention contrôlent l'ensemble de données chaudes stockées localement sur le courtier. Les champs de conservation des archives contrôlent la durée pendant laquelle les données sont stockées dans le stockage d'objets d'archives.
- **Conservation des archives.** Un an (par exemple). Combien de temps souhaitez-vous que vos données soient stockées dans un stockage d'archives ? Entrez -1 avec n'importe quelle unité pour une durée infinie. Le calculateur suppose une conservation de 10 ans pour une conservation infinie.
- **Multiplicateur de croissance.** 1 (par exemple). Si la valeur de ce paramètre est basée sur le débit actuel, définissez-la sur 1. Pour dimensionner en fonction de la croissance supplémentaire, définissez ce paramètre sur un multiplicateur de croissance.
- **Nombre d'instances de producteurs.** 10 (par exemple). Combien d'instances de producteurs seront exécutées ? Cette entrée est nécessaire pour intégrer la charge du processeur dans le calcul de dimensionnement. Une valeur vide indique que la charge du processeur 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 Go/s. Débit maximal en octets non compressés : 4 Go/s. Débit moyen en octets compressés : 400 Mo/s. Débit maximal en octets compressés : 1,6 Go/s. Ceci est basé sur un taux de compression par défaut de 60 % (vous pouvez modifier cette valeur).
  - Stockage total de hotset sur courtier requis : 31 104 To, y compris la réplication, compressée. Stockage d'archives hors courtier total requis : 378 432 To, compressé. Utiliser"https://fusion.netapp.com" pour le dimensionnement de StorageGRID .

Les processeurs de flux doivent décrire leurs applications ou services qui consomment des données d'Apache Kafka et les reproduisent dans Apache Kafka. Dans la plupart des cas, ils sont intégrés à ksqlDB ou à Kafka Streams.

- **Nom.** Streamer Spark.
- **Temps de traitement.** Combien de temps ce processeur prend-il pour traiter un seul message ?
  - 1 ms (transformation simple et sans état) [exemple], 10 ms (opération avec état en mémoire).
  - 100 ms (opération réseau ou disque avec état), 1 000 ms (appel REST tiers).
  - J'ai évalué ce paramètre et je sais exactement combien de temps cela prend.
- **Rétention de sortie.** 1 jour (exemple). Un processeur de flux renvoie sa sortie à Apache Kafka. Combien de temps souhaitez-vous que ces données de sortie 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 un stockage infini ? »
- **Conservation des archives.** 1 an (par exemple). Combien de temps souhaitez-vous que vos données soient stockées dans un stockage d'archives ? Entrez -1 avec n'importe quelle unité pour une durée infinie. Le calculateur suppose une conservation de 10 ans pour une conservation infinie.
- **Pourcentage de transmission de sortie.** 100 (par exemple). Un processeur de flux renvoie sa sortie à Apache Kafka. Quel pourcentage du débit entrant sera renvoyé vers Apache Kafka ? Par exemple, si le débit entrant est de 20 Mo/s et que cette valeur est de 10, le débit de sortie sera de 2 Mo/s.
- À partir de quelles applications cela est-il lu ? Sélectionnez « Spark », le nom utilisé dans le dimensionnement basé sur le type de producteur. Sur la base des données ci-dessus, vous pouvez vous attendre aux effets suivants du dimensionnement sur les instances de processeur de flux et les estimations

de partition de sujet :

- Cette application de traitement de flux nécessite le nombre d'instances suivant. Les sujets entrants nécessitent probablement également autant de partitions. Contactez Confluent pour confirmer ce paramètre.
  - 1 000 pour un débit moyen sans multiplicateur de croissance
  - 4 000 pour un débit maximal sans multiplicateur de croissance
  - 1 000 pour un 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 des données d'Apache Kafka et ne les réinjectent pas dans Apache Kafka ; par exemple, un client natif ou un connecteur Kafka.

- **Nom.** Consommateur Spark.
- **Temps de traitement.** Combien de temps faut-il à ce consommateur pour traiter un seul message ?
  - 1 ms (par exemple, une tâche simple et sans état comme la journalisation)
  - 10 ms (écritures rapides dans une banque de données)
  - 100 ms (écritures lentes dans une banque de données)
  - 1000 ms (appel REST tiers)
  - Un autre processus de référence de durée connue.
- **Type de consommateur.** Application, proxy ou récepteur vers un magasin de données existant (SGBDR, NoSQL, autre).
- À partir de quelles applications cela est-il lu ? Connectez ce paramètre au dimensionnement du producteur et du flux déterminé précédemment.

Sur la base des données ci-dessus, vous devez déterminer le dimensionnement des instances de consommateur et les estimations de partition de sujet. Une application consommateur nécessite le nombre d'instances suivant.

- 2 000 pour un débit moyen, sans multiplicateur de croissance
- 8 000 pour un débit maximal, sans multiplicateur de croissance
- 2 000 pour un débit moyen, multiplicateur de croissance inclus
- 8 000 pour un débit maximal, multiplicateur de croissance inclus

Les sujets entrants ont probablement également besoin de ce nombre de partitions. Contactez Confluent pour confirmer.

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

- **Il est temps de reconstruire.** Par exemple, 4 heures. Si un hôte de courtier Apache Kafka tombe en panne, ses données sont perdues et un nouvel hôte est provisionné pour remplacer l'hôte défaillant, à quelle vitesse ce nouvel hôte doit-il se reconstruire ? Laissez ce paramètre vide si la valeur est inconnue.
- **Objectif d'utilisation des ressources (pourcentage).** Par exemple, 60. Dans quelle mesure souhaitez-vous que vos hôtes soient utilisés pendant le débit moyen ? Confluent recommande une utilisation de 60 %, sauf si vous utilisez des clusters auto-équilibrés Confluent, auquel cas l'utilisation peut être plus élevée.

Décrivez votre environnement

- **Dans quel environnement votre cluster fonctionnera-t-il ?** 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 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/volume de stockage (en Go).** 1000 (par exemple). Quelle quantité de stockage un seul volume peut-il stocker en gigaoctets ? Confluent recommande des disques de 1 To.
- **Configuration de stockage.** Comment les volumes de stockage sont-ils configurés ? Confluent recommande RAID10 pour profiter de toutes les fonctionnalités de Confluent. JBOD, SAN, RAID 1, RAID 0, RAID 5 et d'autres types sont également pris en charge.
- **Débit d'un volume unique (Mbit/s).** 125 (par exemple). À quelle vitesse un seul volume de stockage peut-il lire ou écrire en mégaoctets par seconde ? Confluent recommande des disques durs standard, qui ont généralement un débit de 125 Mo/s.
- **Capacité de mémoire (Go).** 64 (par exemple).

Après avoir déterminé vos variables environnementales, sélectionnez Dimensionner mon cluster. Sur la base des paramètres d'exemple indiqués ci-dessus, nous avons déterminé le dimensionnement suivant pour Confluent Kafka :

- **Apache Kafka.** Nombre de courtiers : 22. Votre cluster est lié au stockage. Envisagez d'activer le stockage hiérarchisé pour réduire le nombre d'hôtes et permettre un stockage infini.
- **Apache ZooKeeper.** Nombre : 5 ; Apache Kafka Connect Workers : Nombre : 2 ; Registre de schémas : Nombre : 2 ; Proxy REST : Nombre : 2 ; ksqlDB : Nombre : 2 ; Centre de contrôle Confluent : Nombre : 1.

Utilisez le mode inversé pour les équipes de plateforme sans cas d'utilisation en tête. Utilisez le mode partitions pour calculer le nombre de partitions requises par une seule rubrique. Voir https://eventsizer.io pour le dimensionnement basé sur les modes inverse et partitions.

Conclusion

Ce document fournit des directives sur les meilleures pratiques pour l'utilisation de Confluent Tiered Storage avec le stockage NetApp , y compris les tests de vérification, les résultats des performances du stockage hiérarchisé, le réglage, les connecteurs Confluent S3 et la fonction d'auto-équilibrage. Compte tenu des politiques ILM, des performances de Confluent avec plusieurs tests de performances pour la vérification et des API S3 standard du secteur, le stockage d'objets NetApp StorageGRID est un choix optimal pour le stockage hiérarchisé de Confluent.

Où trouver des informations supplémentaires

Pour en savoir plus sur les informations décrites dans ce document, consultez les documents et/ou sites Web suivants :

- Qu'est-ce qu'Apache Kafka

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

- Documentation produit NetApp

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

- Détails des paramètres du puits 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 sur la plateforme Confluent

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

- Stockage hiérarchisé Confluent : bonnes pratiques et dimensionnement

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

- Connecteur de récepteur Amazon S3 pour la plateforme Confluent

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

- Dimensionnement Kafka

  "https://eventsizer.io"

- Dimensionnement de StorageGRID

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

- Cas d'utilisation de Kafka

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

- Clusters Kafka auto-équilibrés sur la plateforme 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/"

- Exemples de clients Instaclustr et détails de leurs cas d'utilisation

  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/

Informations sur le copyright

Informations sur les marques commerciales