# Les charges de travail Apache Kafka avec le stockage NetApp NFS

NetApp Solutions

NetApp
April 24, 2024

This PDF was generated from https://docs.netapp.com/fr-fr/netapp-solutions/data-analytics/kafka-nfs-introduction.html on April 24, 2024. Always check docs.netapp.com for the latest.

# Sommaire

Les charges de travail Apache Kafka avec le stockage NetApp NFS . . . 1
Tr-4947 : charge de travail Apache Kafka avec stockage NetApp NFS - validation fonctionnelle et performances . . . 1
Solution NetApp pour renommer Silly les charges de travail NFS en Kafka . . . 2
Validation fonctionnelle - changement de nom Silly . . . 3
Pourquoi choisir NetApp NFS pour les charges de travail Kafka ? . . . 8
Présentation des performances et validation dans AWS . . . 20
Présentation des performances et validation dans AWS FSX pour NetApp ONTAP . . . 33
Présentation des performances et validation avec AFF A900 sur site . . . 41
Conclusion . . . 48
Où trouver des informations complémentaires . . . 48

# Les charges de travail Apache Kafka avec le stockage NetApp NFS

## Tr-4947 : charge de travail Apache Kafka avec stockage NetApp NFS - validation fonctionnelle et performances

Shantanu Chakole, Karthikeyan Nagalingam et Joe Scott, NetApp

Kafka est un système de messagerie distribué à abonnement et publication doté d'une file d'attente robuste capable d'accepter de grandes quantités de données de messages. Avec Kafka, les applications peuvent écrire et lire des données sur des sujets de manière très rapide. En raison de sa tolérance aux pannes et de son évolutivité, Kafka est souvent utilisé dans l'espace Big Data comme un moyen fiable d'ingérer et de déplacer de nombreux flux de données très rapidement. Les utilisations incluent le traitement des flux, le suivi des activités sur le site Web, la collecte et la surveillance de metrics, l'agrégation de journaux, l'analytique en temps réel, etc.

Bien que les opérations Kafka normales sur NFS fonctionnent bien, le système "renommer silly" Ce problème bloque l'application lors du redimensionnement ou de la repartitionnement d'un cluster Kafka qui s'exécute sur NFS. Ce problème est important, car un cluster Kafka doit être redimensionné ou repartitionné à des fins d'équilibrage de la charge ou de maintenance. Vous trouverez des détails supplémentaires "ici".

Ce document aborde les sujets suivants :

- Le problème du changement de nom et la validation de la solution
- Réduction de l'utilisation du CPU pour réduire le temps d'attente d'E/S.
- Délai de restauration du courtier Kafka plus rapide
- Performances dans le cloud et sur site

### Pourquoi utiliser le stockage NFS pour les charges de travail Kafka ?

Les charges de travail Kafka des applications de production peuvent transférer d'importantes quantités de données entre les applications. Ces données sont conservées et stockées dans les nœuds du courtier Kafka du cluster Kafka. Kafka est également connu pour sa disponibilité et son parallélisme, ce qu'il obtient en brisant les rubriques en partitions, puis en répliquant ces partitions dans le cluster. Cela signifie finalement que la quantité considérable de données qui transitent par un cluster Kafka est généralement multipliée par sa taille. NFS accélère et simplifie le rééquilibrage des données à mesure que le nombre de courtiers change. Dans les environnements de grande taille, il est très long de rééquilibrer les données sur DAS lorsque le nombre de courtiers change. Dans la plupart des environnements Kafka, le nombre de courtiers change fréquemment.

Voici d'autres avantages :

- **Maturité.** NFS est un protocole mature, ce qui signifie que la plupart des aspects de sa mise en œuvre, de sa sécurisation et de son utilisation sont bien compris.
- **Open.** NFS est un protocole ouvert, et son développement continu est documenté dans les spécifications Internet comme un protocole de réseau libre et ouvert.

- **Économique.** NFS est une solution économique pour le partage de fichiers en réseau, facile à configurer parce qu'il utilise l'infrastructure réseau existante.
- **Géré de manière centralisée.** la gestion centralisée de NFS réduit le besoin de logiciels supplémentaires et d'espace disque sur les systèmes utilisateur individuels.
- **Distributed.** NFS peut être utilisé comme système de fichiers distribué, réduisant ainsi le besoin de périphériques de stockage amovibles.

### Pourquoi choisir NetApp pour les charges de travail Kafka ?

L'implémentation NFS de NetApp est considérée comme une norme Gold pour le protocole et elle est utilisée dans d'innombrables environnements NAS d'entreprise. Outre la crédibilité de NetApp, il offre également les avantages suivants :

- Fiabilité et efficacité
- Évolutivité et performances
- Haute disponibilité (partenaire HA dans un cluster NetApp ONTAP)
- Protection des données
  - **Reprise sur incident (NetApp SnapMirror).** votre site s'arrête ou vous voulez effectuer des démarrages rapides sur un autre site et continuer là où vous vous êtes arrêté.
  - Facilité de gestion de votre système de stockage (administration et gestion grâce à NetApp OnCommand).
  - **Équilibrage de la charge.** le cluster vous permet d'accéder à différents volumes à partir des LIFs de données hébergées sur différents nœuds.
  - **Continuité de l'activité.** les LIF ou les déplacements de volumes sont transparents pour les clients NFS.

## Solution NetApp pour renommer Silly les charges de travail NFS en Kafka

Kafka est construit en supposant que le système de fichiers sous-jacent est conforme aux exigences POSIX, par exemple, XFS ou Ext4. Le rééquilibrage des ressources par Kafka supprime les fichiers tant que l'application les utilise toujours. Un système de fichiers compatible POSIX permet de procéder à l'annulation de la liaison. Cependant, il supprime le fichier uniquement après avoir supprimé toutes les références au fichier. Si le système de fichiers sous-jacent est connecté au réseau, le client NFS intercepte les appels unlink et gère le flux de travail. Comme des ouvertures sont en attente pour le fichier en cours de dissociation, le client NFS envoie une demande de changement de nom au serveur NFS et, à la dernière fermeture du fichier non lié, émet une opération de suppression sur le fichier renommé. Ce comportement, communément appelé renommage NFS, est orchestré par le client NFS.

Tout courtier Kafka qui utilise le stockage d'un serveur NFSv3 peut rencontrent des problèmes à cause de ce comportement. Toutefois, le protocole NFSv4.x dispose de fonctionnalités permettant de résoudre ce problème en autorisant le serveur à prendre la responsabilité des fichiers ouverts et non liés. Les serveurs NFS prenant en charge cette fonctionnalité facultative communiquent la fonctionnalité de propriété au client NFS au moment de l'ouverture du fichier. Le client NFS cesse ensuite la gestion de liaison lorsque des ouvertures sont en

attente et permet au serveur de gérer le flux. Bien que la spécification NFSv4 fournit des instructions d'implémentation, jusqu'à présent, aucune implémentation de serveurs NFS connue ne soutenait cette fonctionnalité facultative.

Pour résoudre le problème de changement de nom de fichier, le serveur NFS et le client NFS doivent être modifiés comme suit :

- **Modifications apportées au client NFS (Linux).** au moment de l'ouverture du fichier, le serveur NFS répond avec un indicateur indiquant la capacité de gérer le déchaînage des fichiers ouverts. Les modifications côté client NFS permettent au serveur NFS de gérer le déchaînage en présence de l'indicateur. NetApp a mis à jour le client Linux NFS open source avec ces modifications. Le client NFS mis à jour est désormais disponible dans les versions RHEL8.7 et RHEL9.1.
- **Modifications apportées au serveur NFS.** le serveur NFS suit les ouvertures. La dissociation d'un fichier ouvert existant est désormais gérée par le serveur pour correspondre à la sémantique POSIX. Lorsque la dernière ouverture est fermée, le serveur NFS lance alors la suppression réelle du fichier et évite ainsi le processus de renommage. Le serveur NFS ONTAP a mis en œuvre cette fonctionnalité dans sa dernière version, ONTAP 9.12.1.

Grâce aux modifications ci-dessus apportées au client et au serveur NFS, Kafka peut, en toute sécurité, bénéficier de tous les avantages du stockage NFS connecté au réseau.

# Validation fonctionnelle - changement de nom Silly

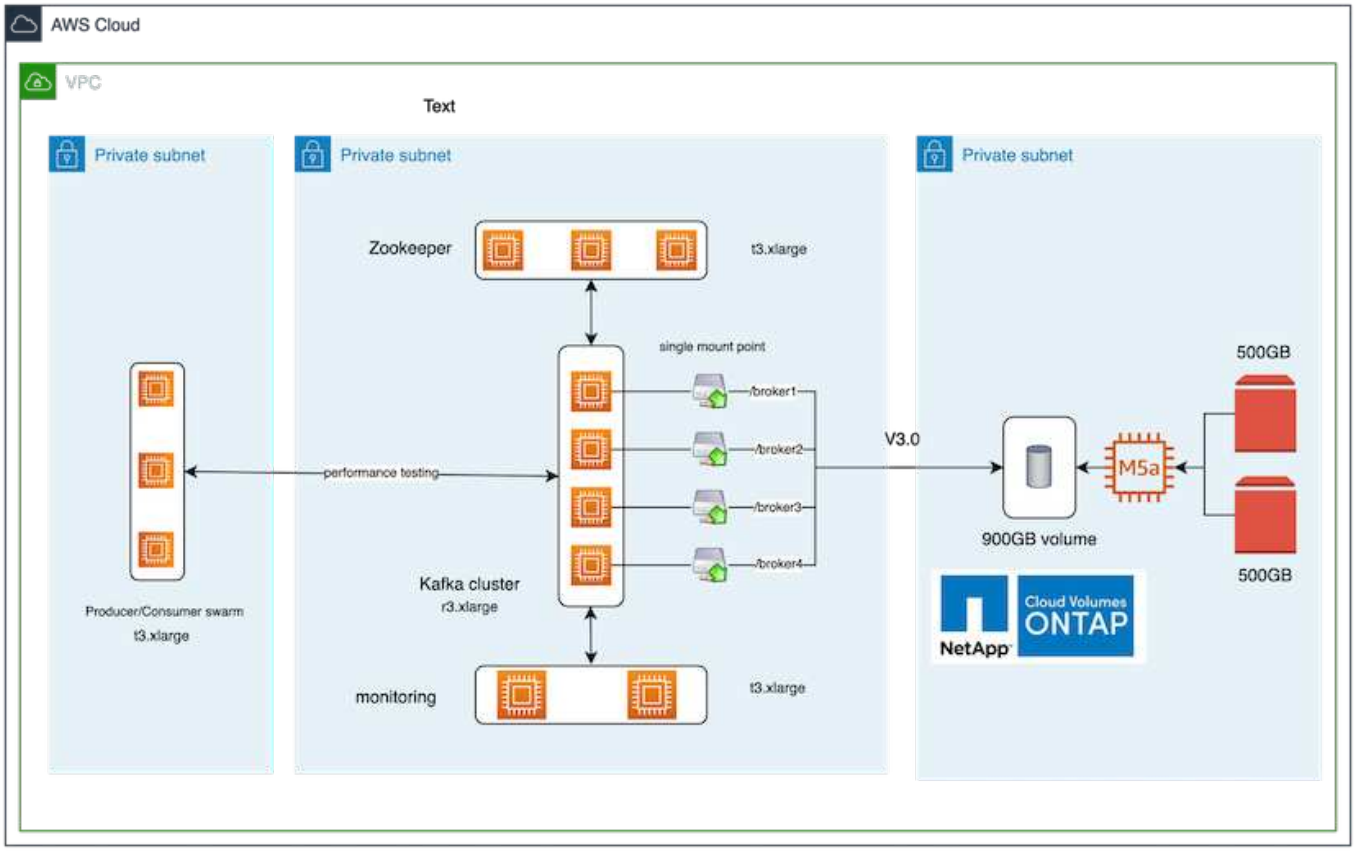
Pour la validation fonctionnelle, nous avons démontré qu'un cluster Kafka avec un montage NFSv3 pour le stockage ne parvient pas à effectuer les opérations Kafka comme la redistribution des partitions, tandis qu'un autre cluster monté sur NFSv4 avec la réparation peut effectuer les mêmes opérations sans aucune perturbation.

## Configuration de la validation

La configuration s'exécute sur AWS. Le tableau suivant présente les différents composants de la plate-forme et la configuration environnementale utilisés pour la validation.

| Composant de plate-forme | Configuration de l'environnement |
|---|---|
| Plate-forme Confluent version 7.2.1 | • 3 x zookeepers – t3.xlarge<br>• 4 serveurs de courtage – r3.xlarge<br>• 1 x Grafana – t3.Xlarge<br>• 1 x centre de contrôle – t3.xlarge<br>• 3 x Producteur/consommateur |
| Système d'exploitation sur tous les nœuds | RHEL8.7ou version ultérieure |
| Instance NetApp Cloud Volumes ONTAP | Instance à un seul nœud – M5.2xLarge |

La figure suivante présente la configuration architecturale de cette solution.




## Flux architectural

- **Compute.** nous avons utilisé un cluster Kafka à quatre nœuds avec un ensemble de zoogardien à trois nœuds fonctionnant sur des serveurs dédiés.
- **Contrôle.** nous avons utilisé deux nœuds pour une combinaison Prometheus-Grafana.
- **Charge de travail.** pour la génération des charges de travail, nous avons utilisé un cluster à trois nœuds distinct qui peut produire et consommer à partir de ce cluster Kafka.
- **Stockage** nous avons utilisé une instance NetApp Cloud Volumes ONTAP à un seul nœud avec deux volumes GP2 AWS EBS de 500 Go rattachés à l'instance. Ces volumes ont ensuite été exposés au cluster Kafka en tant que volume NFSv4.1 unique via une LIF.

Les propriétés par défaut de Kafka ont été choisies pour tous les serveurs. La même chose a été faite pour le bras du zoogardien.

## Méthodologie de test

1. Mise à jour `-is-preserve-unlink-enabled true` pour le volume kafka, comme suit :

```
aws-shantanclastrecall-aws::*> volume create -vserver kafka_svm -volume
kafka_fg_vol01 -aggregate kafka_aggr -size 3500GB -state online -policy
kafka_policy -security-style unix -unix-permissions 0777 -junction-path
/kafka_fg_vol01 -type RW -is-preserve-unlink-enabled true
[Job 32] Job succeeded: Successful
```

2. Deux clusters Kafka similaires ont été créés à la différence suivante :
   - **Cluster 1.** le serveur backend NFS v4.1 exécutant ONTAP version 9.12.1 prêt pour la production était hébergé par une instance NetApp CVO. RHEL 8.7/RHEL 9.1 ont été installés sur les courtiers.
   - **Cluster 2.** le serveur NFS dorsal était un serveur Linux NFSv3 générique créé manuellement.
3. Une rubrique de démonstration a été créée sur les deux clusters Kafka.

   Cluster 1 :

```
[root@ip-172-30-0-160 demo]# kafka-topics --bootstrap-server=172.30.0.160:9092,172.30.0.172:9092,172.30.0.188:90
92,172.30.0.123:9092 --describe --topic __a_demo_topic
Topic: __a_demo_topic   TopicId: 2ty29xfhQLq65HKsUQv-pg PartitionCount: 4       ReplicationFactor: 2    Configs:
 min.insync.replicas=1,segment.bytes=1073741824
        Topic: __a_demo_topic   Partition: 0    Leader: 4       Replicas: 4,1   Isr: 4,1        Offline:
        Topic: __a_demo_topic   Partition: 1    Leader: 2       Replicas: 2,4   Isr: 2,4        Offline:
        Topic: __a_demo_topic   Partition: 2    Leader: 3       Replicas: 3,2   Isr: 3,2        Offline:
        Topic: __a_demo_topic   Partition: 3    Leader: 1       Replicas: 1,3   Isr: 1,3        Offline:
```

   Cluster 2 :

```
[root@ip-172-30-0-198 demo]# kafka-topics --bootstrap-server=172.30.0.198:9092,172.30.0.163:9092,172.30.0.221:90
92,172.30.0.204:9092 --describe --topic __a_demo_topic
Topic: __a_demo_topic   TopicId: AwQpsZTQShyeMIhaquCG3Q PartitionCount: 4       ReplicationFactor: 2    Configs:
 min.insync.replicas=1,segment.bytes=1073741824
        Topic: __a_demo_topic   Partition: 0    Leader: 2       Replicas: 2,3   Isr: 2,3        Offline:
        Topic: __a_demo_topic   Partition: 1    Leader: 3       Replicas: 3,1   Isr: 3,1        Offline:
        Topic: __a_demo_topic   Partition: 2    Leader: 1       Replicas: 1,4   Isr: 1,4        Offline:
        Topic: __a_demo_topic   Partition: 3    Leader: 4       Replicas: 4,2   Isr: 4,2        Offline:
```

4. Les données ont été chargées dans ces rubriques nouvellement créées pour les deux clusters. Cette opération a été effectuée à l'aide du kit d'outils producteur-perf-test fourni avec le package Kafka par défaut :

```
./kafka-producer-perf-test.sh --topic __a_demo_topic --throughput -1
--num-records 3000000 --record-size 1024 --producer-props acks=all
bootstrap.servers=172.30.0.160:9092,172.30.0.172:9092,172.30.0.188:9092,
172.30.0.123:9092
```

5. Une vérification de l'état du broker-1 a été effectuée pour chaque cluster à l'aide de telnet :
   - telnet `172.30.0.160 9092`
   - telnet `172.30.0.198 9092`

     La capture d'écran suivante présente une vérification de l'état des courtiers sur les deux clusters :

```
shantanu@shantanc-mac-0 ~ % telnet 172.30.0.160 9092
Trying 172.30.0.160...
Connected to 172.30.0.160.
Escape character is '^]'.
^[

Connection closed by foreign host.
shantanu@shantanc-mac-0 ~ % telnet 172.30.0.198 9092
Trying 172.30.0.198...
Connected to 172.30.0.198.
Escape character is '^]'.
^[
```

6. Pour déclencher la condition de défaillance à l'origine de la panne des clusters Kafka qui utilisent des volumes de stockage NFSv3, nous avons lancé le processus de réaffectation des partitions sur les deux clusters. La réaffectation de partition a été effectuée à l'aide de `kafka-reassign-partitions.sh`. Le processus détaillé est le suivant :
   a. Pour réaffecter les partitions pour une rubrique dans un cluster Kafka, nous avons généré le fichier JSON de configuration de réaffectation proposé (ceci a été effectué pour les deux clusters).

      ```
      kafka-reassign-partitions --bootstrap
      -server=172.30.0.160:9092,172.30.0.172:9092,172.30.0.188:9092,172.30.
      0.123:9092 --broker-list "1,2,3,4" --topics-to-move-json-file
      /tmp/topics.json --generate
      ```

   b. Le fichier JSON de réaffectation généré a ensuite été enregistré dans `/tmp/reassignment-file.json`.
   c. Le processus réel de réaffectation de partition a été déclenché par la commande suivante :

      ```
      kafka-reassign-partitions --bootstrap
      -server=172.30.0.198:9092,172.30.0.163:9092,172.30.0.221:9092,172.30.
      0.204:9092 --reassignment-json-file /tmp/reassignment-file.json
      -execute
      ```

7. Après quelques minutes, une autre vérification de l'état des courtiers a révélé que le cluster utilisant des volumes de stockage NFSv3 présentait un problème de changement de nom et s'était écrasé, tandis que le cluster 1 utilisant des volumes de stockage NetApp ONTAP NFSv4.1 avec la correction continuait ses opérations sans aucune perturbation.

```
shantanu@shantanc-mac-0 ~ % telnet 172.30.0.160 9092
Trying 172.30.0.160...
Connected to 172.30.0.160.
Escape character is '^]'.
^[

Connection closed by foreign host.
shantanu@shantanc-mac-0 ~ % telnet 172.30.0.198 9092
Trying 172.30.0.198...
telnet: connect to address 172.30.0.198: Connection refused
telnet: Unable to connect to remote host
```

- Cluster1-Broker-1 est actif.
- Le cluster2-broker-1 est en panne.

8. Lors de la vérification des répertoires des journaux Kafka, il était évident que le cluster 1 utilisant des volumes de stockage NetApp ONTAP NFSv4.1 avec la correction avait une affectation de partition propre, tandis que le cluster 2 utilisant un stockage NFSv3 générique n'était pas dû à des problèmes de renommage, ce qui a entraîné la panne. L'image suivante montre le rééquilibrage des partitions du cluster 2, ce qui a provoqué un problème de renommage sur le stockage NFSv3.

```
/demo/broker_demo_1/__a_demo_topic-1.b31a8dd60fd443b283ffda2ecca9c2b9-delete:
total 40
drwxr-xr-x.   2 nobody nobody  4096 Sep 19 10:37 .
drwxr-xr-x. 246 nobody nobody 32768 Sep 19 10:36 ..
-rw-r--r--.   1 nobody nobody     5 Sep 19 10:22 .nfs0000000025f9008400000045
-rw-r--r--.   1 nobody nobody     0 Sep 19 10:25 .nfs0000000025f91d6800000048

/demo/broker_demo_1/__a_demo_topic-2:
total 832592
drwxr-xr-x.   2 nobody nobody      4096 Sep 19 10:26 .
drwxr-xr-x. 246 nobody nobody     32768 Sep 19 10:36 ..
-rw-r--r--.   1 nobody nobody         5 Sep 19 10:22 .nfs0000000025f91d5500000046
-rw-r--r--.   1 nobody nobody         0 Sep 19 10:25 .nfs0000000025f91fce00000047
-rw-r--r--.   1 nobody nobody  10485760 Sep 19 10:24 00000000000000000000.index
-rw-r--r--.   1 nobody nobody 848113134 Sep 19 10:24 00000000000000000000.log
-rw-r--r--.   1 nobody nobody  10485756 Sep 19 10:24 00000000000000000000.timeindex
-rw-r--r--.   1 nobody nobody         0 Sep 19 10:16 leader-epoch-checkpoint
-rw-r--r--.   1 nobody nobody        43 Sep 19 10:16 partition.metadata
```

L'image suivante montre un rééquilibrage des partitions propres du cluster 1 avec un système de stockage NetApp NFSv4.1.

```
/demo/broker_demo_1/__a_demo_topic-0:
total 710932
drwxr-xr-x.  2 nobody nobody      4096 Sep 19 10:26 .
drwxr-xr-x. 85 nobody nobody      8192 Sep 19 10:37 ..
-rw-r--r--.  1 nobody nobody  10485760 Sep 19 10:25 00000000000000000000.index
-rw-r--r--.  1 nobody nobody 724167522 Sep 19 10:25 00000000000000000000.log
-rw-r--r--.  1 nobody nobody  10485756 Sep 19 10:25 00000000000000000000.timeindex
-rw-r--r--.  1 nobody nobody         0 Sep 19 10:15 leader-epoch-checkpoint
-rw-r--r--.  1 nobody nobody        43 Sep 19 10:15 partition.metadata

/demo/broker_demo_1/__a_demo_topic-2:
total 780016
drwxr-xr-x.  2 nobody nobody      4096 Sep 19 10:35 .
drwxr-xr-x. 85 nobody nobody      8192 Sep 19 10:37 ..
-rw-r--r--.  1 nobody nobody  10485760 Sep 19 10:36 00000000000000000000.index
-rw-r--r--.  1 nobody nobody 794575786 Sep 19 10:36 00000000000000000000.log
-rw-r--r--.  1 nobody nobody  10485756 Sep 19 10:36 00000000000000000000.timeindex
-rw-r--r--.  1 nobody nobody         0 Sep 19 10:35 leader-epoch-checkpoint
-rw-r--r--.  1 nobody nobody        43 Sep 19 10:35 partition.metadata
```

# Pourquoi choisir NetApp NFS pour les charges de travail Kafka ?

Maintenant qu'il existe une solution au problème du renommage dans le stockage NFS avec Kafka, vous pouvez créer des déploiements fiables qui exploitent le stockage NetApp ONTAP pour votre charge de travail Kafka. Non seulement cette configuration réduit considérablement la surcharge opérationnelle, mais elle offre également les avantages suivants à vos clusters Kafka :

- **Utilisation réduite du CPU sur les courtiers Kafka** l'utilisation du stockage NetApp ONTAP désagrégée sépare les opérations d'E/S du disque du courtier et réduit ainsi l'empreinte du processeur.
- **Temps de restauration du courtier plus rapide.** comme le stockage ONTAP NetApp désagrégée est partagé entre les nœuds du courtier Kafka, une nouvelle instance de calcul peut remplacer un mauvais courtier à tout moment et en beaucoup moins de temps qu'avec les déploiements Kafka classiques, sans reconstruire les données.
- **Efficacité du stockage.** comme la couche de stockage de l'application est maintenant provisionnée via NetApp ONTAP, les clients peuvent bénéficier de tous les avantages de l'efficacité du stockage fournie avec ONTAP, tels que la compression, la déduplication et la compaction des données à la volée.

Ces avantages ont été testés et validés dans des scénarios de test que nous aborderons en détail dans cette section.

## Réduction de l'utilisation du processeur sur le courtier Kafka

Nous avons découvert que l'utilisation globale du processeur était inférieure à celle de son homologue DAS lorsque nous exécutions des charges de travail similaires sur deux clusters Kafka à spermatozoïdes identiques dans leurs spécifications techniques, mais dont les technologies de stockage différaient. Non seulement l'utilisation globale du processeur est inférieure lorsque le cluster Kafka utilise le stockage ONTAP, mais l'augmentation de l'utilisation du CPU s'est avérée plus modérée que dans un cluster Kafka basé sur DAS.

### Installation architecturale

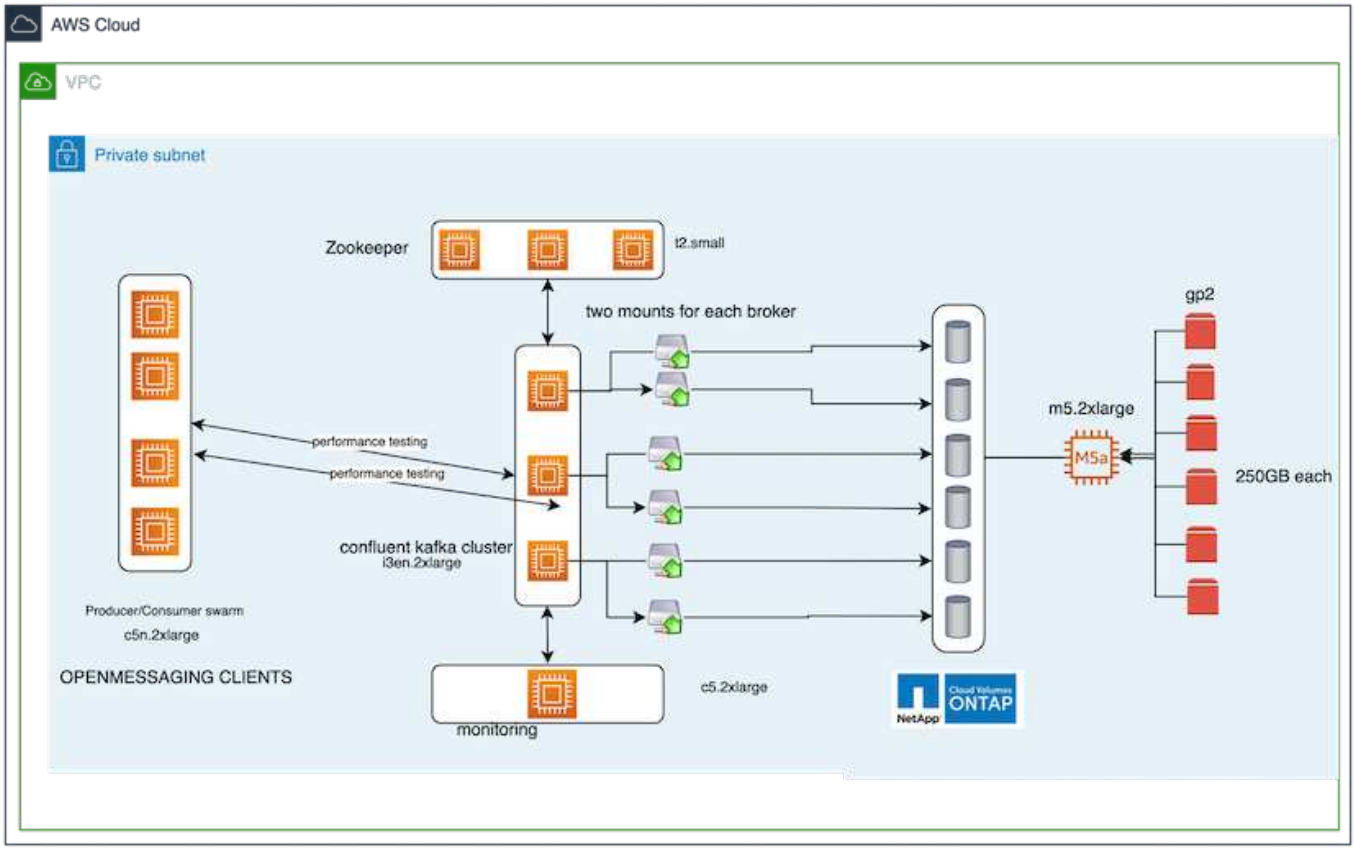
Le tableau suivant présente la configuration environnementale utilisée pour démontrer la réduction de

l'utilisation du processeur.

| Composant de plate-forme | Configuration de l'environnement |
|---|---|
| Outil de banc d'essai Kafka 3.2.3 : OpenMessaging | • 3 x zookeepers – t2.small<br>• 3 serveurs de broker – i3en.2xlarge<br>• 1 x Grafana – c5n.2xlarge<br>• 4 x Producteur/consommateur — c5n.2xlarge |
| Système d'exploitation sur tous les nœuds | RHEL 8.7 ou version ultérieure |
| Instance NetApp Cloud Volumes ONTAP | Instance à un seul nœud – M5.2xLarge |

### Outil d'évaluation

L'outil d'analyse comparative utilisé dans ce cas d'essai est le "OpenMessaging" structure. OpenMessaging est indépendant du langage et du fournisseur ; il fournit des directives sectorielles pour la finance, le commerce électronique, l'IoT et le Big Data ; il aide également à développer des applications de messagerie et de diffusion en continu sur des systèmes et plates-formes hétérogènes. La figure suivante illustre l'interaction des clients OpenMessaging avec un cluster Kafka.




- **Compute.** nous avons utilisé un cluster Kafka à trois nœuds avec un ensemble de zoogardien à trois nœuds fonctionnant sur des serveurs dédiés. Chaque courtier disposait de deux points de montage NFSv4.1 sur un seul volume de l'instance NetApp CVO via une LIF dédiée.
- **Contrôle.** nous avons utilisé deux nœuds pour une combinaison Prometheus-Grafana. Pour la génération des charges de travail, nous disposons d'un cluster séparé à trois nœuds qui peut produire et consommer

à partir de ce cluster Kafka.

- **Stockage** nous avons utilisé une instance NetApp Cloud Volumes ONTAP à un seul nœud avec six volumes GP2 AWS EBS de 250 Go montés sur l'instance. Ces volumes ont ensuite été exposés au cluster Kafka en tant que six volumes NFSv4.1 via des LIF dédiées.
- **Configuration.** les deux éléments configurables dans ce cas de test étaient les courtiers Kafka et les charges de travail OpenMessaging.
  - **Broker config.** les spécifications suivantes ont été sélectionnées pour les courtiers Kafka. Nous avons utilisé un facteur de réplication de 3 pour toutes les mesures, comme indiqué ci-dessous.

```
broker.id=1
advertised.listeners=PLAINTEXT://172.30.0.185:9092
log.dirs=/mnt/data-1
zookeeper.connect=172.30.0.13:2181,172.30.0.108:2181,172.30.0.253:2181
num.replica.fetchers=8
message.max.bytes=10485760
replica.fetch.max.bytes=10485760
num.network.threads=8
default.replication.factor=3
replica.lag.time.max.ms=100000000
replica.fetch.max.bytes=1048576
replica.fetch.wait.max.ms=500
num.replica.fetchers=1
replica.high.watermark.checkpoint.interval.ms=5000
fetch.purgatory.purge.interval.requests=1000
producer.purgatory.purge.interval.requests=1000
replica.socket.timeout.ms=30000
replica.socket.receive.buffer.bytes=65536
```

- **OpenMessaging benchmark (OMB) configuration de la charge de travail.** les spécifications suivantes ont été fournies. Nous avons spécifié un taux d'apporteur cible, mis en surbrillance ci-dessous.

```
name: 4 producer / 4 consumers on 1 topic
topics: 1
partitionsPerTopic: 100
messageSize: 1024
payloadFile: "payload/payload-1Kb.data"
subscriptionsPerTopic: 1
consumerPerSubscription: 4
producersPerTopic: 4
producerRate: 40000
consumerBacklogSizeGB: 0
testDurationMinutes: 5
```

#### Méthodologie de test

1. Deux grappes similaires ont été créées, chacune ayant son propre ensemble de essaims de grappes de

benchmarking.

- **Cluster 1.** cluster Kafka basé sur NFS.
- **Cluster 2.** cluster Kafka à base de DAS.

2. À l'aide d'une commande OpenMessaging, des charges de travail similaires ont été déclenchées sur chaque cluster.

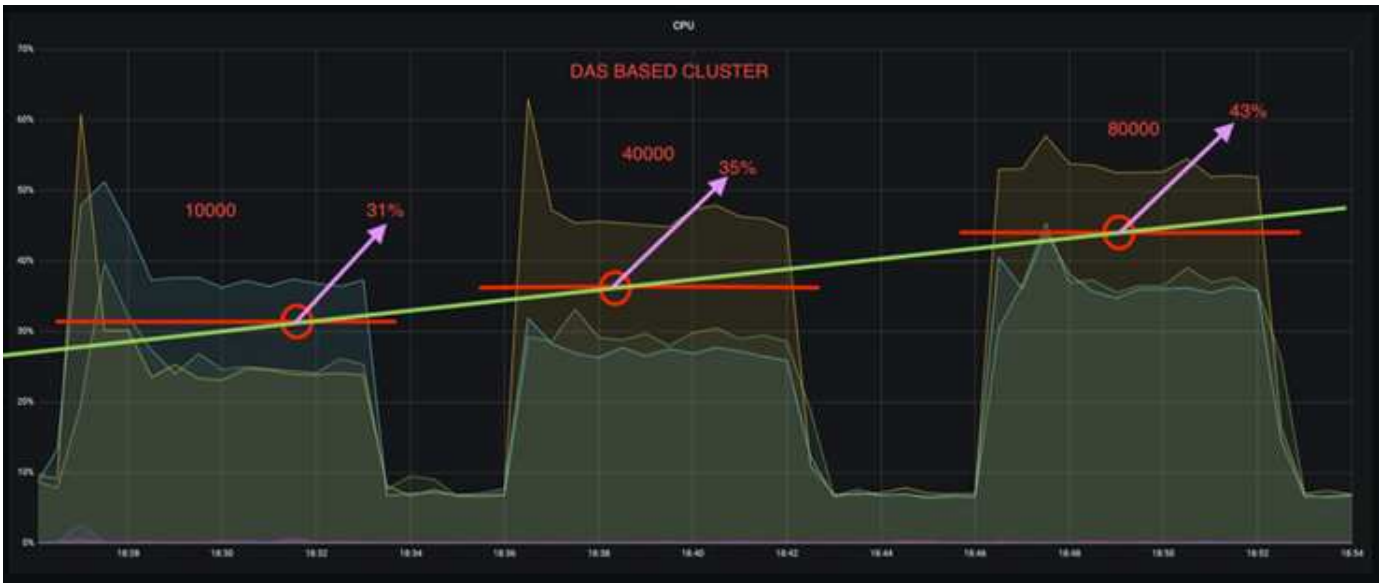
```
sudo bin/benchmark --drivers driver-kafka/kafka-group-all.yaml
workloads/1-topic-100-partitions-1kb.yaml
```

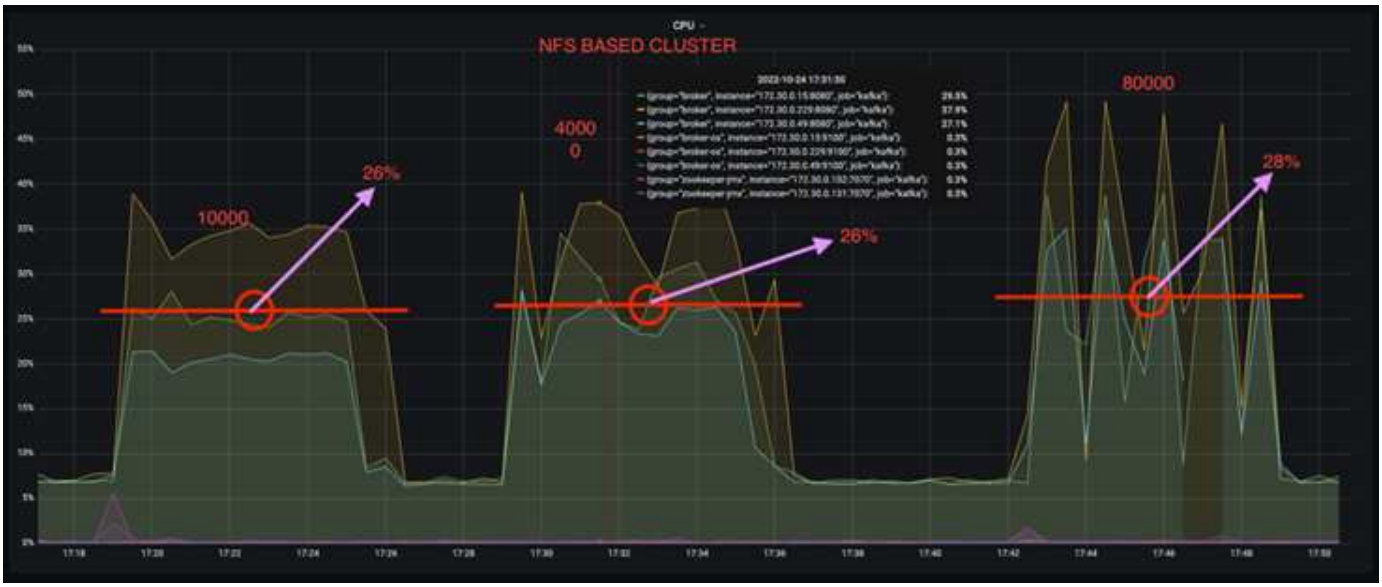
3. La configuration du débit de production a été augmentée en quatre itérations et l'utilisation du processeur a été enregistrée avec Grafana. Le taux de production a été défini sur les niveaux suivants :
   - 10,000
   - 40,000
   - 80,000
   - 100,000

**Observation**

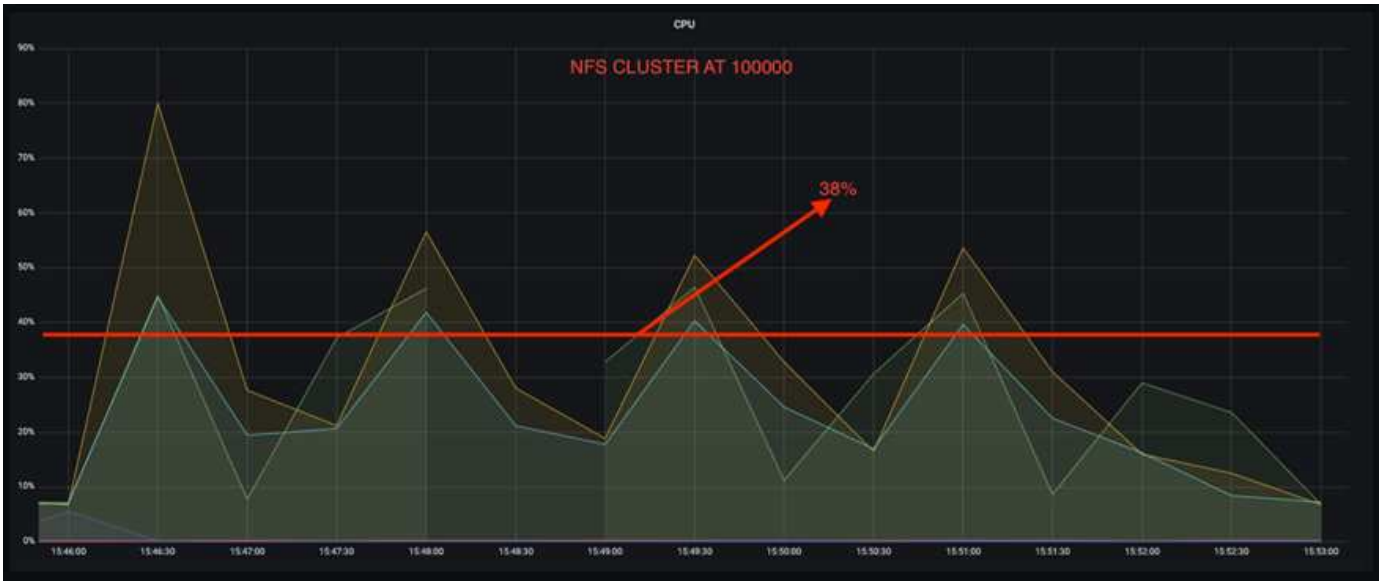
L'utilisation du stockage NFS NetApp avec Kafka présente deux avantages principaux :

- **Vous pouvez réduire l'utilisation du processeur de près d'un tiers.** l'utilisation globale du processeur sous des charges de travail similaires a été plus faible pour NFS que pour les SSD DAS ; les économies vont de 5 % pour des taux de production inférieurs à 32 % pour des taux de production plus élevés.
- **Une diminution de trois fois de la dérive de l'utilisation du CPU à des taux de production plus élevés.** comme prévu, il y a eu une dérive à la hausse de l'augmentation de l'utilisation du CPU au fur et à mesure que les taux de production ont été augmentés. Cependant, le taux d'utilisation du CPU sur les courtiers Kafka qui utilisent le DAS est passé de 31 % pour le taux de production inférieur à 70 % pour le taux de production supérieur, soit une augmentation de 39 %. Cependant, avec un système de stockage NFS back-end, l'utilisation du processeur est passée de 26 à 38 %, soit une augmentation de 12 %.






De plus, avec 100,000 messages, le DAS affiche une plus grande utilisation du CPU qu'un cluster NFS.



## Une restauration plus rapide des courtiers

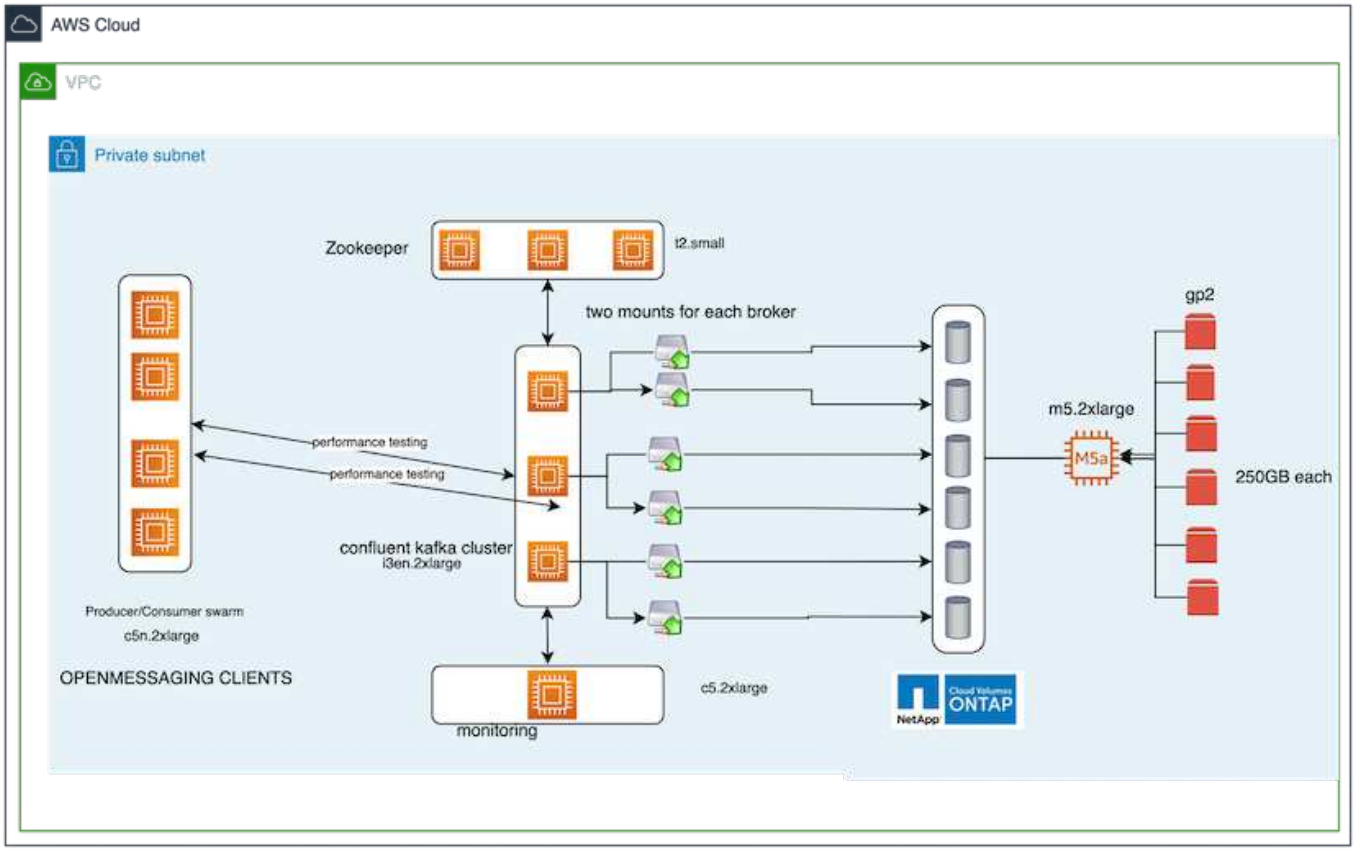
Nous avons découvert que les courtiers Kafka accélèrent la restauration lorsqu'ils utilisent un stockage NetApp NFS partagé. Lorsqu'un courtier tombe en panne dans un cluster Kafka, ce courtier peut être remplacé par un courtier sain avec le même ID de courtier. Lors de l'exécution de ce test, nous avons constaté que, dans le cas d'un cluster Kafka basé sur DAS, le cluster reconstruit les données sur un nouveau courtier en état de fonctionnement, ce qui prend du temps. Dans le cas d'un cluster Kafka basé sur NetApp NFS, le courtier qui remplace le système continue à lire les données à partir du précédent répertoire de journaux et restaure beaucoup plus rapidement.

### Installation architecturale

Le tableau suivant présente la configuration environnementale d'un cluster Kafka utilisant un NAS.

| Composant de plate-forme | Configuration de l'environnement |
|---|---|
| Kafka 3.2.3 | • 3 x zookeepers – t2.small<br>• 3 serveurs de broker – i3en.2xlarge<br>• 1 x Grafana – c5n.2xlarge<br>• 4 x producteur/consommateur — c5n.2xlarge<br>• 1 nœud Kafka de sauvegarde – i3en.2xlarge |
| Système d'exploitation sur tous les nœuds | RHEL8.7 ou version ultérieure |
| Instance NetApp Cloud Volumes ONTAP | Instance à un seul nœud – M5.2xLarge |

La figure suivante illustre l'architecture d'un cluster Kafka basé sur NAS.




- **Compute.** un cluster Kafka à trois nœuds avec un ensemble de zoogardien à trois nœuds fonctionnant sur des serveurs dédiés. Chaque courtier dispose de deux points de montage NFS sur un seul volume de l'instance NetApp CVO via une LIF dédiée.
- **Contrôle.** deux nœuds pour une combinaison Prometheus-Grafana. Pour la génération des charges de travail, nous utilisons un cluster séparé à trois nœuds qui peut produire et consommer sur ce cluster Kafka.
- **Stockage** instance NetApp Cloud Volumes ONTAP à un seul nœud avec six volumes GP2 AWS EBS de 250 Go montés sur l'instance. Ces volumes sont ensuite exposés au cluster Kafka en tant que six volumes NFS via des LIF dédiées.
- **Configuration Broker.** dans ce cas de test, un élément configurable est un courtier Kafka. Les spécifications suivantes ont été sélectionnées pour les courtiers Kafka. Le `replica.lag.time.mx.ms` Est défini sur une valeur élevée car cela détermine la vitesse à laquelle un nœud particulier est extrait de la liste ISR. Lorsque vous basculez entre les nœuds défectueux et les nœuds sains, vous ne voulez pas que cet ID de courtier soit exclu de la liste ISR.

```
broker.id=1
advertised.listeners=PLAINTEXT://172.30.0.185:9092
log.dirs=/mnt/data-1
zookeeper.connect=172.30.0.13:2181,172.30.0.108:2181,172.30.0.253:2181
num.replica.fetchers=8
message.max.bytes=10485760
replica.fetch.max.bytes=10485760
num.network.threads=8
default.replication.factor=3
replica.lag.time.max.ms=100000000
replica.fetch.max.bytes=1048576
replica.fetch.wait.max.ms=500
num.replica.fetchers=1
replica.high.watermark.checkpoint.interval.ms=5000
fetch.purgatory.purge.interval.requests=1000
producer.purgatory.purge.interval.requests=1000
replica.socket.timeout.ms=30000
replica.socket.receive.buffer.bytes=65536
```

### Méthodologie de test

1. Deux clusters similaires ont été créés :
   - Cluster courant basé sur EC2.
   - Cluster NetApp NFS confluent.
2. Un nœud Kafka de secours a été créé avec une configuration identique à celle des nœuds du cluster Kafka d'origine.
3. Sur chacun des clusters, un sujet d'exemple a été créé et environ 110 Go de données ont été remplis sur chacun des courtiers.
   - **Cluster basé sur EC2.** Un répertoire de données de courtier Kafka est mappé sur `/mnt/data-2` (Dans la figure suivante, Broker-1 du cluster 1 [terminal gauche]).
   - **Cluster NetApp NFS** Un répertoire de données Kafka Broker est monté sur un point NFS `/mnt/data` (Dans la figure suivante, Broker-1 du cluster2 [terminal droit]).

4. Dans chacun des clusters, Broker-1 a été arrêté pour déclencher un processus de restauration de courtier en échec.
5. Après la fin du courtier, l'adresse IP du courtier a été attribuée comme adresse IP secondaire au courtier en attente. Cette opération était nécessaire car un courtier d'un cluster Kafka est identifié par ce qui suit :
   - **Adresse IP.** attribuée en réaffectant l'adresse IP du courtier en échec au courtier en attente.
   - **ID du courtier.** il a été configuré dans le courtier en attente `server.properties`.
6. Lors de l'attribution de l'adresse IP, le service Kafka a été démarré sur le courtier en veille.
7. Au bout d'un moment, les journaux du serveur ont été extraits pour vérifier le temps nécessaire à la création des données sur le nœud de remplacement du cluster.

### Observation

La restauration du courtier Kafka était presque neuf fois plus rapide. Le temps nécessaire à la restauration d'un nœud de courtier en échec s'est avéré considérablement plus rapide lors de l'utilisation du stockage partagé NetApp NFS que lors de l'utilisation de disques SSD DAS dans un cluster Kafka. Pour 1 To de données topic, le temps de restauration d'un cluster DAS était de 48 minutes, contre moins de 5 minutes pour un cluster Kafka basé sur NetApp-NFS.

Nous avons constaté que la reconstruction des 110 Go de données sur le nouveau nœud intermédiaire du cluster basé sur EC2 a pris 10 minutes, alors que la restauration s'est effectuée en 3 minutes. Nous avons également observé dans les journaux que les décalages consommateur pour les partitions pour EC2 étaient 0, tandis que, sur le cluster NFS, les décalages consommateur étaient récupérés auprès du précédent courtier.

```
[2022-10-31 09:39:17,747] INFO [LogLoader partition=test-topic-51R3EWs-
0000-55, dir=/mnt/kafka-data/broker2] Reloading from producer snapshot and
rebuilding producer state from offset 583999 (kafka.log.UnifiedLog$)
[2022-10-31 08:55:55,170] INFO [LogLoader partition=test-topic-qbVsEZg-
0000-8, dir=/mnt/data-1] Loading producer state till offset 0 with message
format version 2 (kafka.log.UnifiedLog$)
```

#### Cluster basé sur DAS

1. Le nœud de sauvegarde a démarré à 08:55:53,730.

```
2  [2022-10-31 08:55:53,661] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotia
3  [2022-10-31 08:55:53,727] INFO Registered signal handlers for TERM, INT, HUP (or
4  [2022-10-31 08:55:53,730] INFO starting (kafka.server.KafkaServer)
5  [2022-10-31 08:55:53,730] INFO Connecting to zookeeper on 172.30.0.17:2181,172.3
6  [2022-10-31 08:55:53,755] INFO [ZooKeeperClient Kafka server] Initializing a new
```

2. Le processus de reconstruction des données s'est terminé à 09:05:24,860. Le traitement de 110 Go de données a nécessité environ 10 minutes.

```
[2022-10-31 09:05:24,860] INFO [ReplicaFetcherManager on broker 1] Removed fetcher for
partitions HashSet(test-topic-qbVsEZg-0000-95, test-topic-qbVsEZg-0000-5,
test-topic-qbVsEZg-0000-41, test-topic-qbVsEZg-0000-23, test-topic-qbVsEZg-0000-11,
test-topic-qbVsEZg-0000-47, test-topic-qbVsEZg-0000-83, test-topic-qbVsEZg-0000-35,
test-topic-qbVsEZg-0000-89, test-topic-qbVsEZg-0000-71, test-topic-qbVsEZg-0000-53,
test-topic-qbVsEZg-0000-29, test-topic-qbVsEZg-0000-59, test-topic-qbVsEZg-0000-77,
test-topic-qbVsEZg-0000-65, test-topic-qbVsEZg-0000-17)
(kafka.server.ReplicaFetcherManager)
```

#### Cluster basé sur NFS

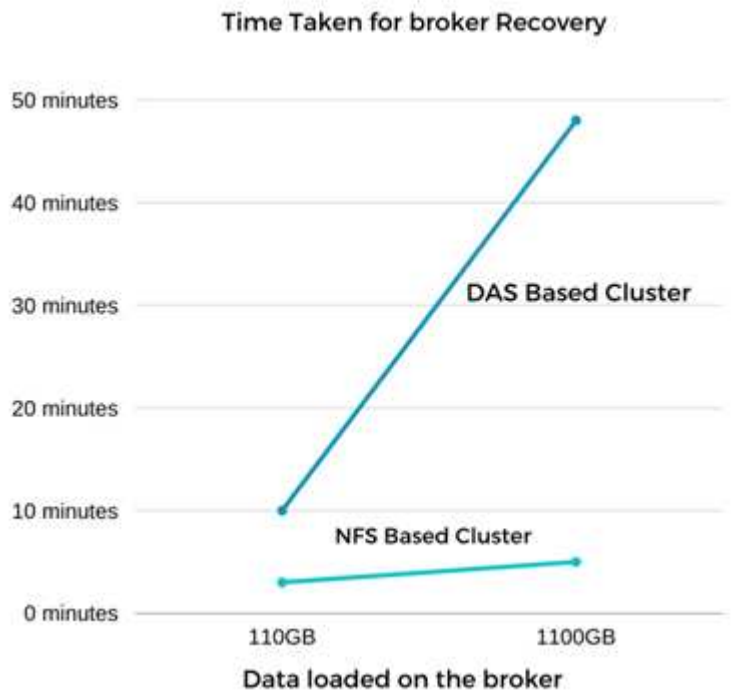
1. Le nœud de sauvegarde a été démarré à 09:39:17,213. L'entrée du journal de démarrage est mise en surbrillance ci-dessous.

```
[2022-10-31 09:39:17,142] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiati
[2022-10-31 09:39:17,211] INFO Registered signal handlers for TERM, INT, HUP (org.
[2022-10-31 09:39:17,213] INFO starting (kafka.server.KafkaServer)
[2022-10-31 09:39:17,214] INFO Connecting to zookeeper on 172.30.0.22:2181,172.30.
[2022-10-31 09:39:17,238] INFO [ZooKeeperClient Kafka server] Initializing a new s
[2022-10-31 09:39:17,244] INFO Client environment:zookeeper.version=3.6.3--6401e4a
[2022-10-31 09:39:17,244] INFO Client environment:host.name=ip-172-30-0-110.ec2.in
[2022-10-31 09:39:17,244] INFO Client environment:java.version=11.0.17 (org.apache
```

2. Le processus de reconstruction des données s'est terminé à 09:42:29,115. Le traitement de 110 Go de données a nécessité environ 3 minutes.

```
[2022-10-31 09:42:29,115] INFO [GroupMetadataManager brokerId=1] Finished loading offsets
and group metadata from __consumer_offsets-20 in 28478 milliseconds for epoch 3, of which
28478 milliseconds was spent in the scheduler.
(kafka.coordinator.group.GroupMetadataManager)
```

Le test a été répété pour les courtiers contenant environ 1 To de données, ce qui a pris environ 48 minutes pour le DAS et 3 minutes pour le NFS. Les résultats sont présentés dans le graphique suivant.




## Efficacité du stockage

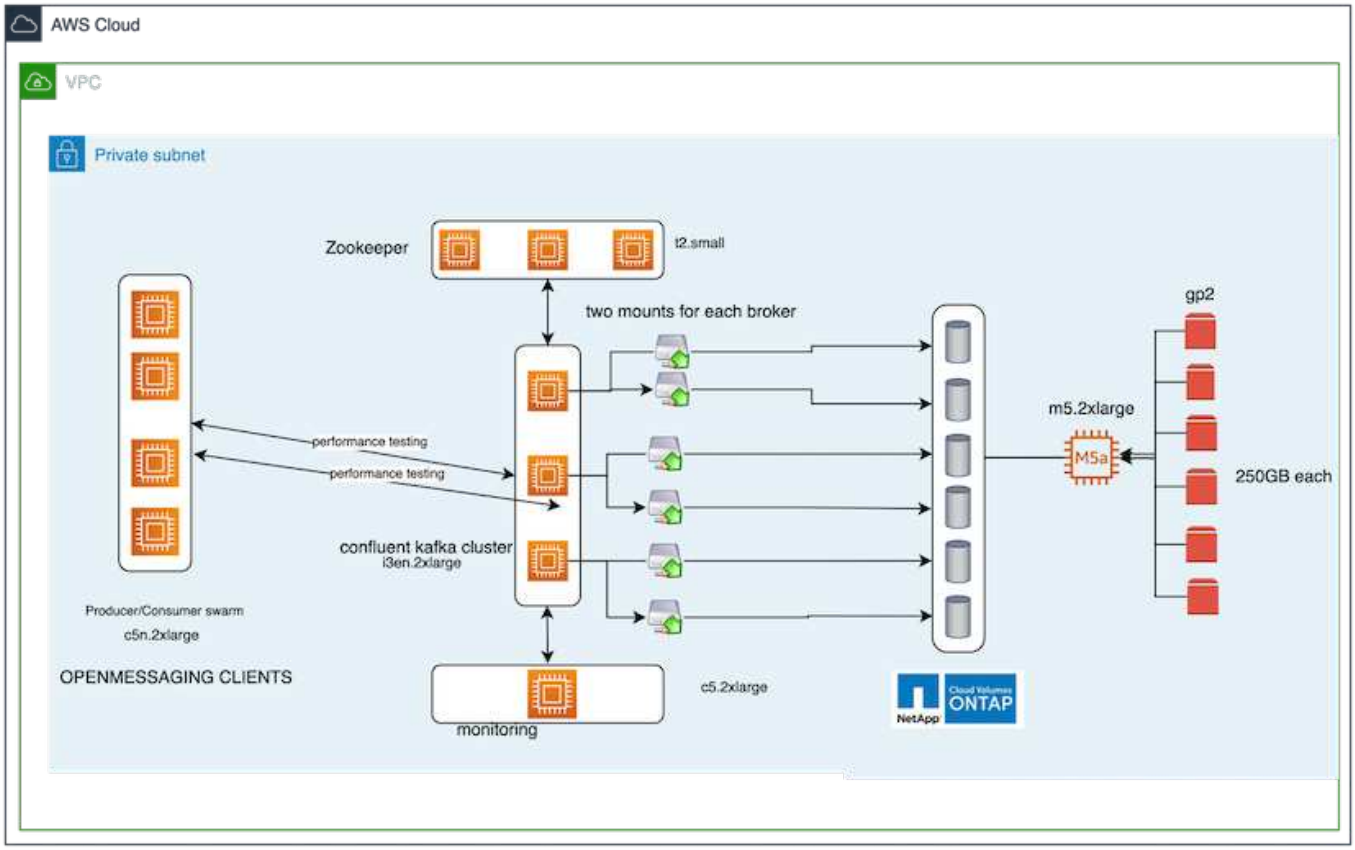
Comme la couche de stockage du cluster Kafka a été provisionnée via NetApp ONTAP, nous avons toutes les capacités d'efficacité du stockage de ONTAP. Ce test a été effectué en générant une quantité importante de données sur un cluster Kafka avec stockage NFS provisionné sur Cloud Volumes ONTAP. Nous avons pu constater qu'il y avait une réduction d'espace importante grâce aux fonctionnalités ONTAP.

### Installation architecturale

Le tableau suivant présente la configuration environnementale d'un cluster Kafka utilisant un NAS.

| Composant de plate-forme | Configuration de l'environnement |
|---|---|
| Kafka 3.2.3 | • 3 x zookeepers – t2.small<br>• 3 serveurs de broker – i3en.2xlarge<br>• 1 x Grafana – c5n.2xlarge<br>• 4 x producteur/consommateur — c5n.2xlarge * |
| Système d'exploitation sur tous les nœuds | RHEL8.7 ou version ultérieure |
| Instance NetApp Cloud Volumes ONTAP | Instance à un seul nœud – M5.2xLarge |

La figure suivante illustre l'architecture d'un cluster Kafka basé sur NAS.




- **Compute.** nous avons utilisé un cluster Kafka à trois nœuds avec un ensemble de zoogardien à trois nœuds fonctionnant sur des serveurs dédiés. Chaque courtier disposait de deux points de montage NFS sur un seul volume de l'instance NetApp CVO via une LIF dédiée.
- **Contrôle.** nous avons utilisé deux nœuds pour une combinaison Prometheus-Grafana. Pour la génération des charges de travail, nous avons utilisé un cluster séparé à trois nœuds qui était capable de produire et de consommer sur ce cluster Kafka.
- **Stockage** nous avons utilisé une instance NetApp Cloud Volumes ONTAP à un seul nœud avec six volumes GP2 AWS EBS de 250 Go montés sur l'instance. Ces volumes ont ensuite été exposés au cluster Kafka en tant que six volumes NFS via des LIF dédiées.
- **Configuration.** les éléments configurables dans ce cas de test étaient les courtiers Kafka.

La compression a été désactivée à l'extrémité du producteur, permettant ainsi aux producteurs de générer un

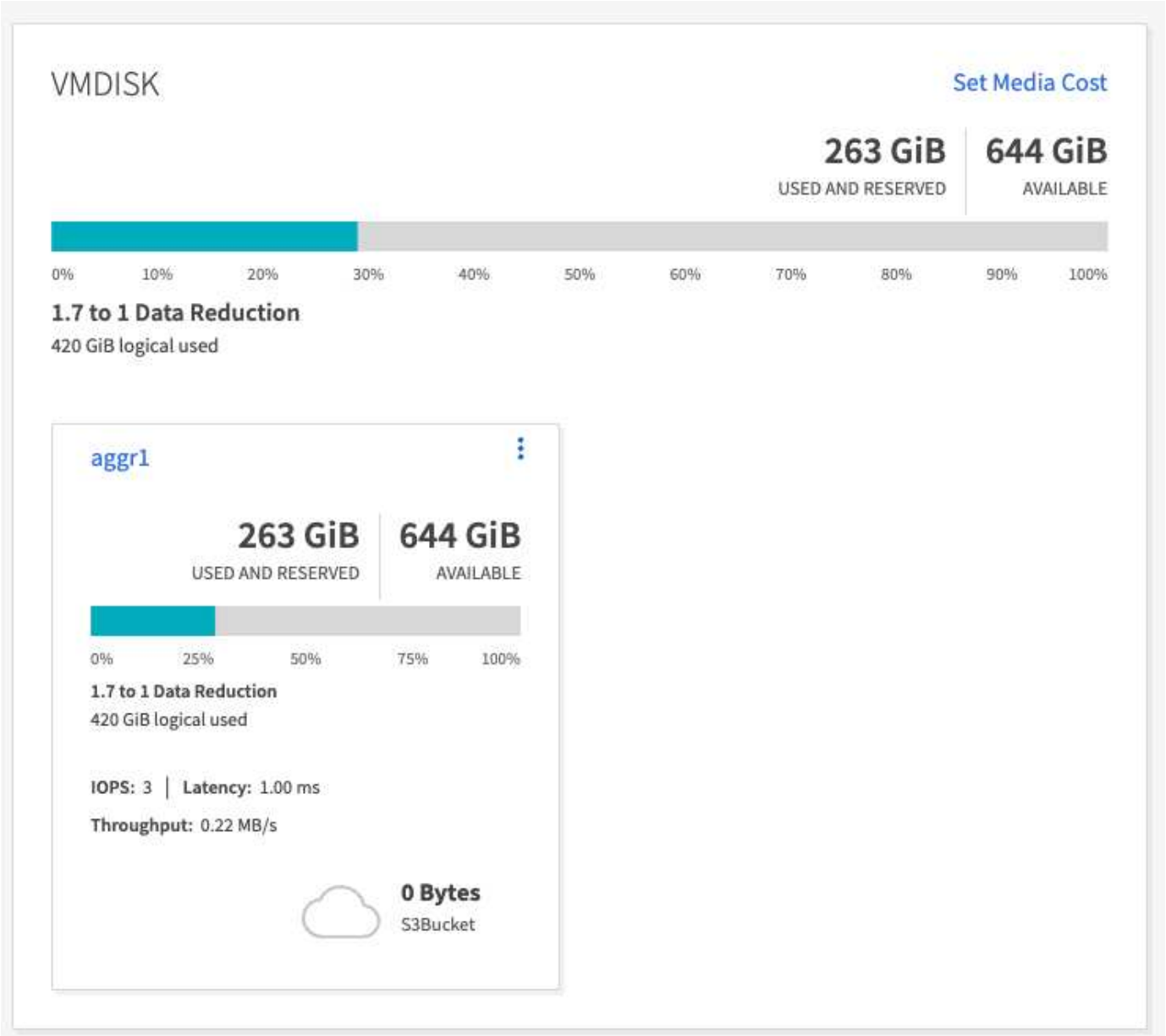
rendement élevé. À la place, l'efficacité du stockage était gérée par la couche de calcul.

### Méthodologie de test

1. Un cluster Kafka a été provisionné avec les spécifications mentionnées ci-dessus.
2. Sur le cluster, environ 350 Go de données ont été produites à l'aide de l'outil d'analyse comparative OpenMessaging.
3. Une fois la charge de travail terminée, les statistiques d'efficacité du stockage ont été collectées à l'aide de ONTAP System Manager et de l'interface de ligne de commandes.

### Observation

Concernant les données générées à l'aide de l'outil OMB, nous avons constaté des économies d'espace d'environ 33 % avec un ratio d'efficacité du stockage de 1.70:1. Comme le montrent les figures suivantes, l'espace logique utilisé par les données produites était de 420,3 Go et l'espace physique utilisé pour les données était de 281,7 Go.



```
shantanuCVOinstancenew::> df -h -S

Warning: The "-S" parameter is deprecated and may be removed in a future release. To show the efficiency ratio use "aggr show-efficiency"
         command.
Filesystem               used       total-saved    %total-saved    deduplicated    %deduplicated    compressed    %compressed  Vserver
/vol/vol0/             7319MB                0B              0%              0B               0%            0B             0%  shantanuCVOinstancenew-01
/vol/kafka_vol/         281GB             138GB             33%           138GB              33%            0B             0%  svm_shantanuCVOinstancenew
/vol/svm_shantanuCVOinstancenew_root/
                        660KB                0B              0%              0B               0%            0B             0%  svm_shantanuCVOinstancenew
3 entries were displayed.
```

```
                        Name of the Aggregate: aggr1
                 Node where Aggregate Resides: shantanuCVOinstancenew-01
                Total Storage Efficiency Ratio: 1.70:1
Total Data Reduction Efficiency Ratio Without Snapshots: 1.70:1
Total Data Reduction Efficiency Ratio without snapshots and flexclones: 1.70:1
           Logical Space Used for All Volumes: 420.3GB
          Physical Space Used for All Volumes: 281.7GB
```

# Présentation des performances et validation dans AWS

Les performances dans le cloud AWS ont été évaluées sur un cluster Kafka avec la couche de stockage montée sur NetApp NFS. Les exemples de benchmarking sont décrits dans les sections suivantes.

## Kafka dans le cloud AWS avec NetApp Cloud Volumes ONTAP (paire haute disponibilité et nœud unique)

Un cluster Kafka avec NetApp Cloud Volumes ONTAP (paire haute disponibilité) a été testé sur banc d'essai pour la performance dans le cloud AWS. Cette analyse comparative est décrite dans les sections suivantes.

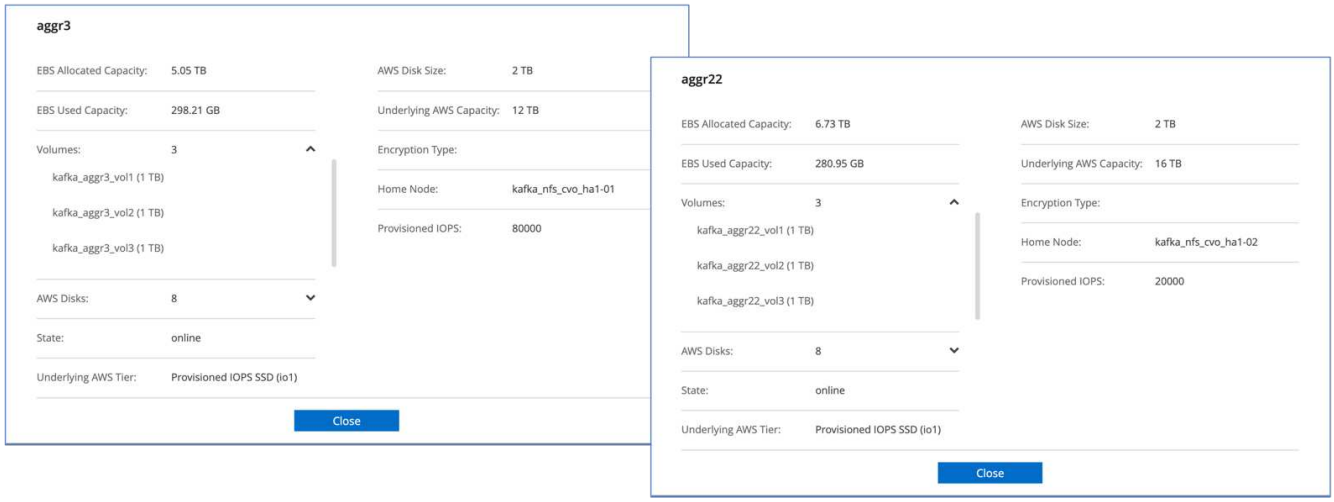
### Installation architecturale

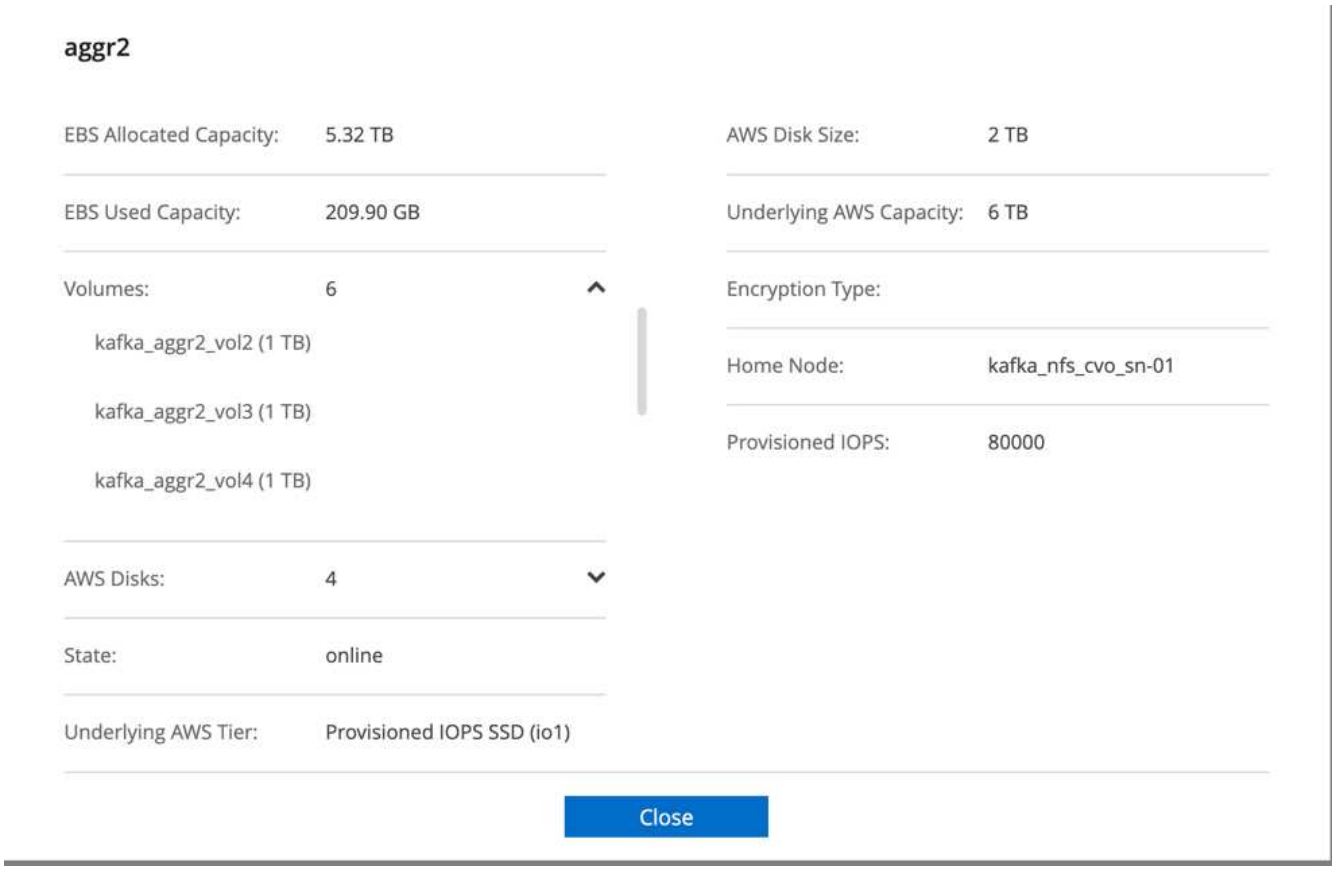
Le tableau suivant présente la configuration environnementale d'un cluster Kafka utilisant un NAS.

| Composant de plate-forme | Configuration de l'environnement |
|---|---|
| Kafka 3.2.3 | • 3 x zookeepers – t2.small<br>• 3 serveurs de broker – i3en.2xlarge<br>• 1 x Grafana – c5n.2xlarge<br>• 4 x producteur/consommateur — c5n.2xlarge * |
| Système d'exploitation sur tous les nœuds | RHEL8.6 |
| Instance NetApp Cloud Volumes ONTAP | Instance de paire HAUTE DISPONIBILITÉ – m5dn.12xLarge x 2 nœuds instance de nœud unique - m5dn.12xLarge x 1 nœud |

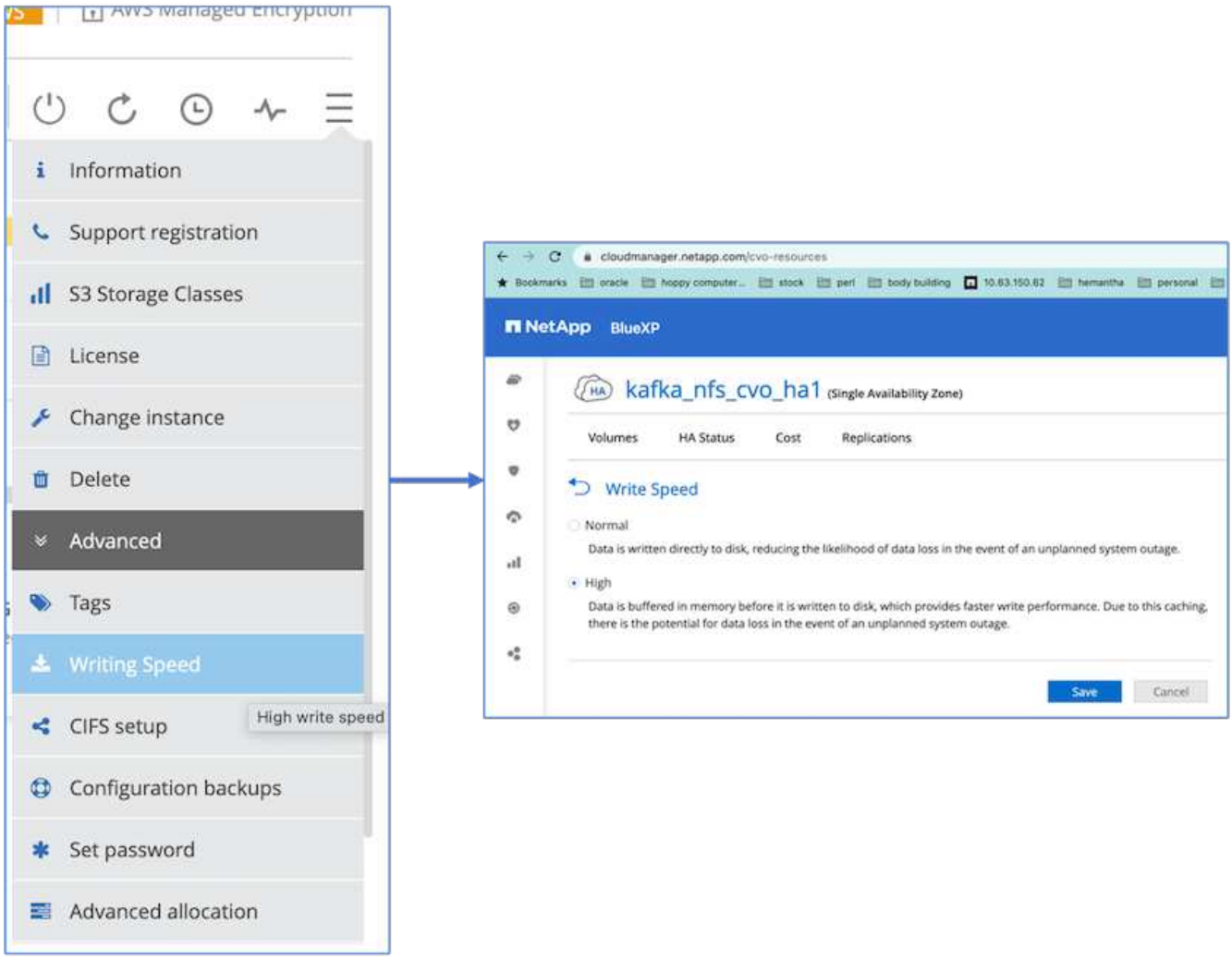
### Configuration de NetApp Cluster volume ONTAP

1. Pour la paire haute disponibilité Cloud Volumes ONTAP, nous avons créé deux agrégats avec trois volumes sur chaque agrégat de chaque contrôleur de stockage. Pour le seul nœud Cloud Volumes ONTAP, nous créons six volumes dans un agrégat.



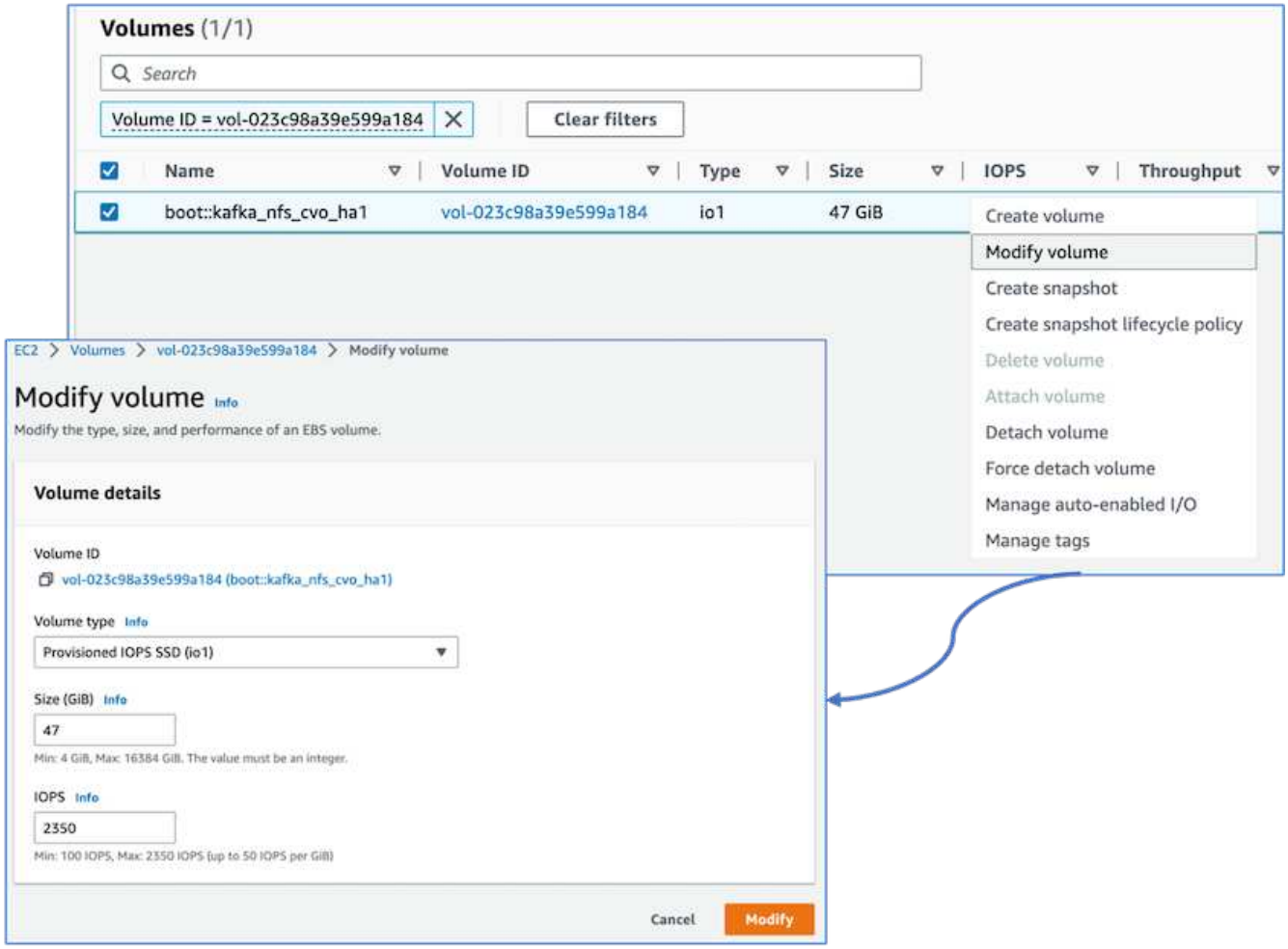




2. Afin d'améliorer les performances réseau, nous avons activé une mise en réseau haut débit pour la paire haute disponibilité et le nœud unique.



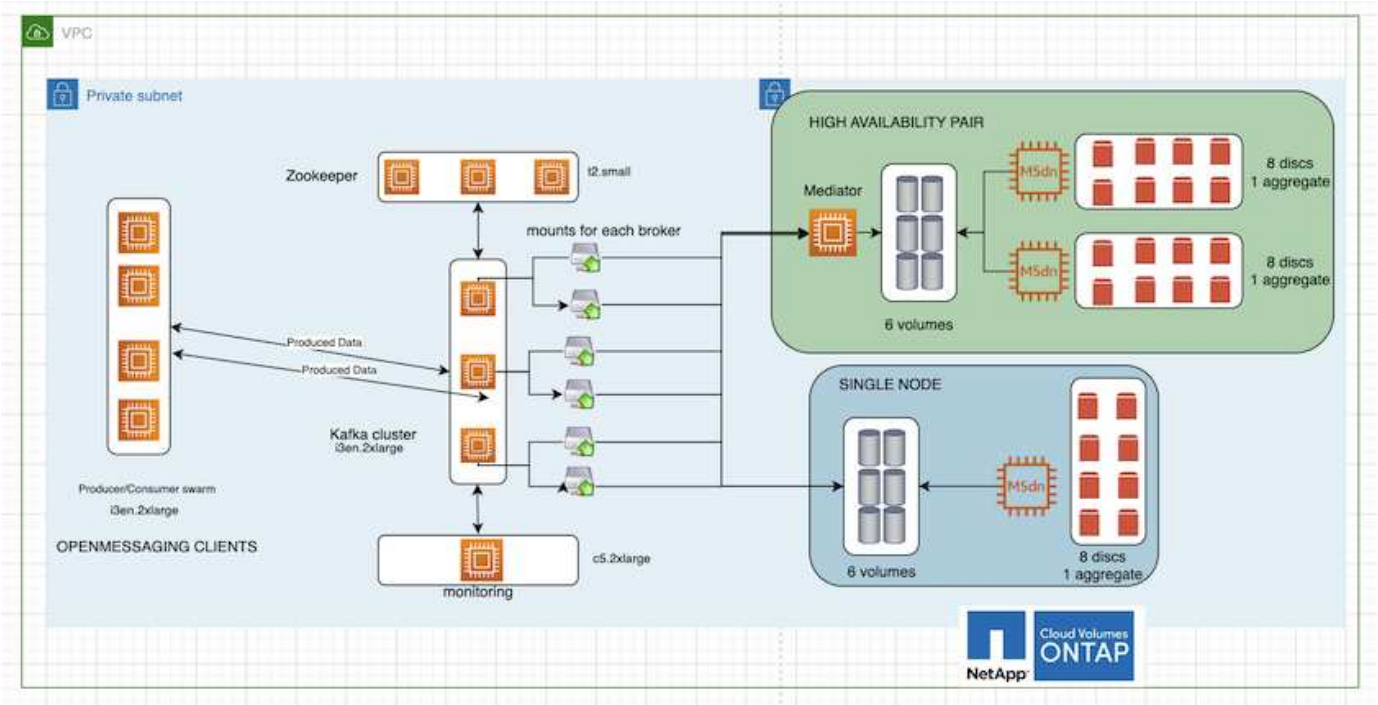

3. Nous avons constaté que la mémoire NVRAM de ONTAP offrait plus d'IOPS. Nous avons donc remplacé le nombre d'IOPS par un nombre de 2350 pour le volume racine Cloud Volumes ONTAP. La taille du disque de volume racine dans Cloud Volumes ONTAP était de 47 Go. La commande ONTAP suivante s'applique à la paire HA, et la même étape s'applique à un seul nœud.

```
statistics start -object vnvram -instance vnvram -counter
backing_store_iops -sample-id sample_555
kafka_nfs_cvo_ha1::*> statistics show -sample-id sample_555
Object: vnvram
Instance: vnvram
Start-time: 1/18/2023 18:03:11
End-time: 1/18/2023 18:03:13
Elapsed-time: 2s
Scope: kafka_nfs_cvo_ha1-01
    Counter                                                     Value
    -------------------------------- --------------------------------
    backing_store_iops                                           1479
Object: vnvram
Instance: vnvram
Start-time: 1/18/2023 18:03:11
End-time: 1/18/2023 18:03:13
Elapsed-time: 2s
Scope: kafka_nfs_cvo_ha1-02
    Counter                                                     Value
    -------------------------------- --------------------------------
    backing_store_iops                                           1210
2 entries were displayed.
kafka_nfs_cvo_ha1::*>
```




La figure suivante illustre l'architecture d'un cluster Kafka basé sur NAS.

- **Compute.** nous avons utilisé un cluster Kafka à trois nœuds avec un ensemble de zoogardien à trois nœuds fonctionnant sur des serveurs dédiés. Chaque courtier disposait de deux points de montage NFS sur un seul volume de l'instance Cloud Volumes ONTAP via une LIF dédiée.
- **Contrôle.** nous avons utilisé deux nœuds pour une combinaison Prometheus-Grafana. Pour la génération des charges de travail, nous avons utilisé un cluster séparé à trois nœuds qui était capable de produire et de consommer sur ce cluster Kafka.
- **Stockage** nous avons utilisé une instance Cloud Volumes ONTAP à paire haute disponibilité avec un volume GP3 AWS-EBS de 6 To monté sur l'instance. Le volume a ensuite été exporté vers le courtier Kafka avec un montage NFS.




### Configurations de test OpenMessage

1. Pour améliorer les performances NFS, nous avons besoin d'un plus grand nombre de connexions réseau entre le serveur NFS et le client NFS, qui peuvent être créées à l'aide de nconnect. Montez les volumes NFS sur les nœuds de courtier avec l'option nconnect en exécutant la commande suivante :

```
[root@ip-172-30-0-121 ~]# cat /etc/fstab
UUID=eaa1f38e-de0f-4ed5-a5b5-2fa9db43bb38/xfsdefaults00
/dev/nvme1n1 /mnt/data-1 xfs defaults,noatime,nodiscard 0 0
/dev/nvme2n1 /mnt/data-2 xfs defaults,noatime,nodiscard 0 0
172.30.0.233:/kafka_aggr3_vol1 /kafka_aggr3_vol1 nfs
defaults,nconnect=16 0 0
172.30.0.233:/kafka_aggr3_vol2 /kafka_aggr3_vol2 nfs
defaults,nconnect=16 0 0
172.30.0.233:/kafka_aggr3_vol3 /kafka_aggr3_vol3 nfs
defaults,nconnect=16 0 0
172.30.0.242:/kafka_aggr22_vol1 /kafka_aggr22_vol1 nfs
defaults,nconnect=16 0 0
172.30.0.242:/kafka_aggr22_vol2 /kafka_aggr22_vol2 nfs
defaults,nconnect=16 0 0
172.30.0.242:/kafka_aggr22_vol3 /kafka_aggr22_vol3 nfs
defaults,nconnect=16 0 0
[root@ip-172-30-0-121 ~]# mount -a
[root@ip-172-30-0-121 ~]# df -h
Filesystem                      Size  Used Avail Use% Mounted on
devtmpfs                         31G     0   31G   0% /dev
tmpfs                            31G  249M   31G   1% /run
tmpfs                            31G     0   31G   0% /sys/fs/cgroup
/dev/nvme0n1p2                   10G  2.8G  7.2G  28% /
/dev/nvme1n1                    2.3T  248G  2.1T  11% /mnt/data-1
/dev/nvme2n1                    2.3T  245G  2.1T  11% /mnt/data-2
172.30.0.233:/kafka_aggr3_vol1   1.0T   12G 1013G   2% /kafka_aggr3_vol1
172.30.0.233:/kafka_aggr3_vol2   1.0T  5.5G 1019G   1% /kafka_aggr3_vol2
172.30.0.233:/kafka_aggr3_vol3   1.0T  8.9G 1016G   1% /kafka_aggr3_vol3
172.30.0.242:/kafka_aggr22_vol1  1.0T  7.3G 1017G   1%
/kafka_aggr22_vol1
172.30.0.242:/kafka_aggr22_vol2  1.0T  6.9G 1018G   1%
/kafka_aggr22_vol2
172.30.0.242:/kafka_aggr22_vol3  1.0T  5.9G 1019G   1%
/kafka_aggr22_vol3
tmpfs                           6.2G     0  6.2G   0% /run/user/1000
[root@ip-172-30-0-121 ~]#
```

2. Vérifiez les connexions réseau dans Cloud Volumes ONTAP. La commande ONTAP suivante est utilisée à partir du nœud Cloud Volumes ONTAP unique. La même étape s'applique à la paire haute disponibilité Cloud Volumes ONTAP.

```
Last login time: 1/20/2023 00:16:29
kafka_nfs_cvo_sn::> network connections active show -service nfs*
-fields remote-host
node                cid        vserver              remote-host
```

```
------------------- ---------- -------------------- ------------
kafka_nfs_cvo_sn-01 2315762628 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762629 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762630 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762631 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762632 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762633 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762634 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762635 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762636 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762637 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762639 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762640 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762641 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762642 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762643 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762644 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762645 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762646 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762647 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762648 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762649 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762650 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762651 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762652 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762653 svm_kafka_nfs_cvo_sn 172.30.0.121
kafka_nfs_cvo_sn-01 2315762656 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762657 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762658 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762659 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762660 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762661 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762662 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762663 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762664 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762665 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762666 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762667 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762668 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762669 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762670 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762671 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762672 svm_kafka_nfs_cvo_sn 172.30.0.72
kafka_nfs_cvo_sn-01 2315762673 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762674 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762676 svm_kafka_nfs_cvo_sn 172.30.0.121
```

```
kafka_nfs_cvo_sn-01 2315762677 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762678 svm_kafka_nfs_cvo_sn 172.30.0.223
kafka_nfs_cvo_sn-01 2315762679 svm_kafka_nfs_cvo_sn 172.30.0.223
48 entries were displayed.

kafka_nfs_cvo_sn::>
```

3. Nous utilisons Kafka suivant `server.properties` Dans tous les courtiers Kafka de la paire HA Cloud Volumes ONTAP. Le `log.dirs` la propriété est différente pour chaque courtier, et les autres propriétés sont communes aux courtiers. Pour broker1, le `log.dirs` la valeur est la suivante :

```
[root@ip-172-30-0-121 ~]# cat /opt/kafka/config/server.properties
broker.id=0
advertised.listeners=PLAINTEXT://172.30.0.121:9092
#log.dirs=/mnt/data-1/d1,/mnt/data-1/d2,/mnt/data-1/d3,/mnt/data-
2/d1,/mnt/data-2/d2,/mnt/data-2/d3
log.dirs=/kafka_aggr3_vol1/broker1,/kafka_aggr3_vol2/broker1,/kafka_aggr
3_vol3/broker1,/kafka_aggr22_vol1/broker1,/kafka_aggr22_vol2/broker1,/ka
fka_aggr22_vol3/broker1
zookeeper.connect=172.30.0.12:2181,172.30.0.30:2181,172.30.0.178:2181
num.network.threads=64
num.io.threads=64
socket.send.buffer.bytes=102400
socket.receive.buffer.bytes=102400
socket.request.max.bytes=104857600
num.partitions=1
num.recovery.threads.per.data.dir=1
offsets.topic.replication.factor=1
transaction.state.log.replication.factor=1
transaction.state.log.min.isr=1
replica.fetch.max.bytes=524288000
background.threads=20
num.replica.alter.log.dirs.threads=40
num.replica.fetchers=20
[root@ip-172-30-0-121 ~]#
```

   - Pour broker2, le `log.dirs` la valeur de la propriété est la suivante :

```
log.dirs=/kafka_aggr3_vol1/broker2,/kafka_aggr3_vol2/broker2,/kafka_a
ggr3_vol3/broker2,/kafka_aggr22_vol1/broker2,/kafka_aggr22_vol2/broke
r2,/kafka_aggr22_vol3/broker2
```

   - Pour broker3, le `log.dirs` la valeur de la propriété est la suivante :

```
log.dirs=/kafka_aggr3_vol1/broker3,/kafka_aggr3_vol2/broker3,/kafka_a
ggr3_vol3/broker3,/kafka_aggr22_vol1/broker3,/kafka_aggr22_vol2/broke
r3,/kafka_aggr22_vol3/broker3
```

4. Pour le seul nœud Cloud Volumes ONTAP, Kafka `servers.properties` Est identique à celui de la paire haute disponibilité Cloud Volumes ONTAP, à l'exception du `log.dirs` propriété.
   - Pour broker1, le `log.dirs` la valeur est la suivante :

     ```
     log.dirs=/kafka_aggr2_vol1/broker1,/kafka_aggr2_vol2/broker1,/kafka_a
     ggr2_vol3/broker1,/kafka_aggr2_vol4/broker1,/kafka_aggr2_vol5/broker1
     ,/kafka_aggr2_vol6/broker1
     ```

   - Pour broker2, le `log.dirs` la valeur est la suivante :

     ```
     log.dirs=/kafka_aggr2_vol1/broker2,/kafka_aggr2_vol2/broker2,/kafka_a
     ggr2_vol3/broker2,/kafka_aggr2_vol4/broker2,/kafka_aggr2_vol5/broker2
     ,/kafka_aggr2_vol6/broker2
     ```

   - Pour broker3, le `log.dirs` la valeur de la propriété est la suivante :

     ```
     log.dirs=/kafka_aggr2_vol1/broker3,/kafka_aggr2_vol2/broker3,/kafka_a
     ggr2_vol3/broker3,/kafka_aggr2_vol4/broker3,/kafka_aggr2_vol5/broker3
     ,/kafka_aggr2_vol6/broker3
     ```

5. La charge de travail dans l'OMB est configurée avec les propriétés suivantes : `(/opt/benchmark/workloads/1-topic-100-partitions-1kb.yaml)`.

   ```
   topics: 4
   partitionsPerTopic: 100
   messageSize: 32768
   useRandomizedPayloads: true
   randomBytesRatio: 0.5
   randomizedPayloadPoolSize: 100
   subscriptionsPerTopic: 1
   consumerPerSubscription: 80
   producersPerTopic: 40
   producerRate: 1000000
   consumerBacklogSizeGB: 0
   testDurationMinutes: 5
   ```

   Le `messageSize` peuvent varier selon les utilisations. Lors de notre test de performance, nous avons

utilisé 3 Ko.

Nous avons utilisé deux pilotes différents, Sync ou Throughput, d'OMB pour générer la charge de travail sur le cluster Kafka.

- Le fichier yaml utilisé pour les propriétés du pilote Sync est le suivant `(/opt/benchmark/driver-kafka/kafka-sync.yaml)`:

```
name: Kafka
driverClass:
io.openmessaging.benchmark.driver.kafka.KafkaBenchmarkDriver
# Kafka client-specific configuration
replicationFactor: 3
topicConfig: |
  min.insync.replicas=2
  flush.messages=1
  flush.ms=0
commonConfig: |

bootstrap.servers=172.30.0.121:9092,172.30.0.72:9092,172.30.0.223:909
2
producerConfig: |
  acks=all
  linger.ms=1
  batch.size=1048576
consumerConfig: |
  auto.offset.reset=earliest
  enable.auto.commit=false
  max.partition.fetch.bytes=10485760
```

- Le fichier yaml utilisé pour les propriétés du pilote Throughput est le suivant `(/opt/benchmark/driver- kafka/kafka-throughput.yaml)`:

```
name: Kafka
driverClass:
io.openmessaging.benchmark.driver.kafka.KafkaBenchmarkDriver
# Kafka client-specific configuration
replicationFactor: 3
topicConfig: |
  min.insync.replicas=2
commonConfig: |

bootstrap.servers=172.30.0.121:9092,172.30.0.72:9092,172.30.0.223:909
2
  default.api.timeout.ms=1200000
  request.timeout.ms=1200000
producerConfig: |
  acks=all
  linger.ms=1
  batch.size=1048576
consumerConfig: |
  auto.offset.reset=earliest
  enable.auto.commit=false
  max.partition.fetch.bytes=10485760
```

## Méthodologie de test

1. Un cluster Kafka a été provisionné selon la spécification décrite ci-dessus à l'aide de Terraform et Ansible. Terraform est utilisé pour créer l'infrastructure à l'aide d'instances AWS pour le cluster Kafka et Ansible y intègre le cluster Kafka.
2. Une charge de travail OMB a été déclenchée avec la configuration de la charge de travail décrite ci-dessus et le pilote Sync.

   ```
   Sudo bin/benchmark –drivers driver-kafka/kafka- sync.yaml workloads/1-
   topic-100-partitions-1kb.yaml
   ```

3. Une autre charge de travail a été déclenchée avec le pilote de débit avec la même configuration de charge de travail.

   ```
   sudo bin/benchmark –drivers driver-kafka/kafka-throughput.yaml
   workloads/1-topic-100-partitions-1kb.yaml
   ```

## Observation

Deux types de pilotes différents ont été utilisés pour générer des charges de travail afin de tester les performances d'une instance Kafka fonctionnant sur NFS. La différence entre les pilotes est la propriété log

flush.

Pour une paire Cloud Volumes ONTAP HA :

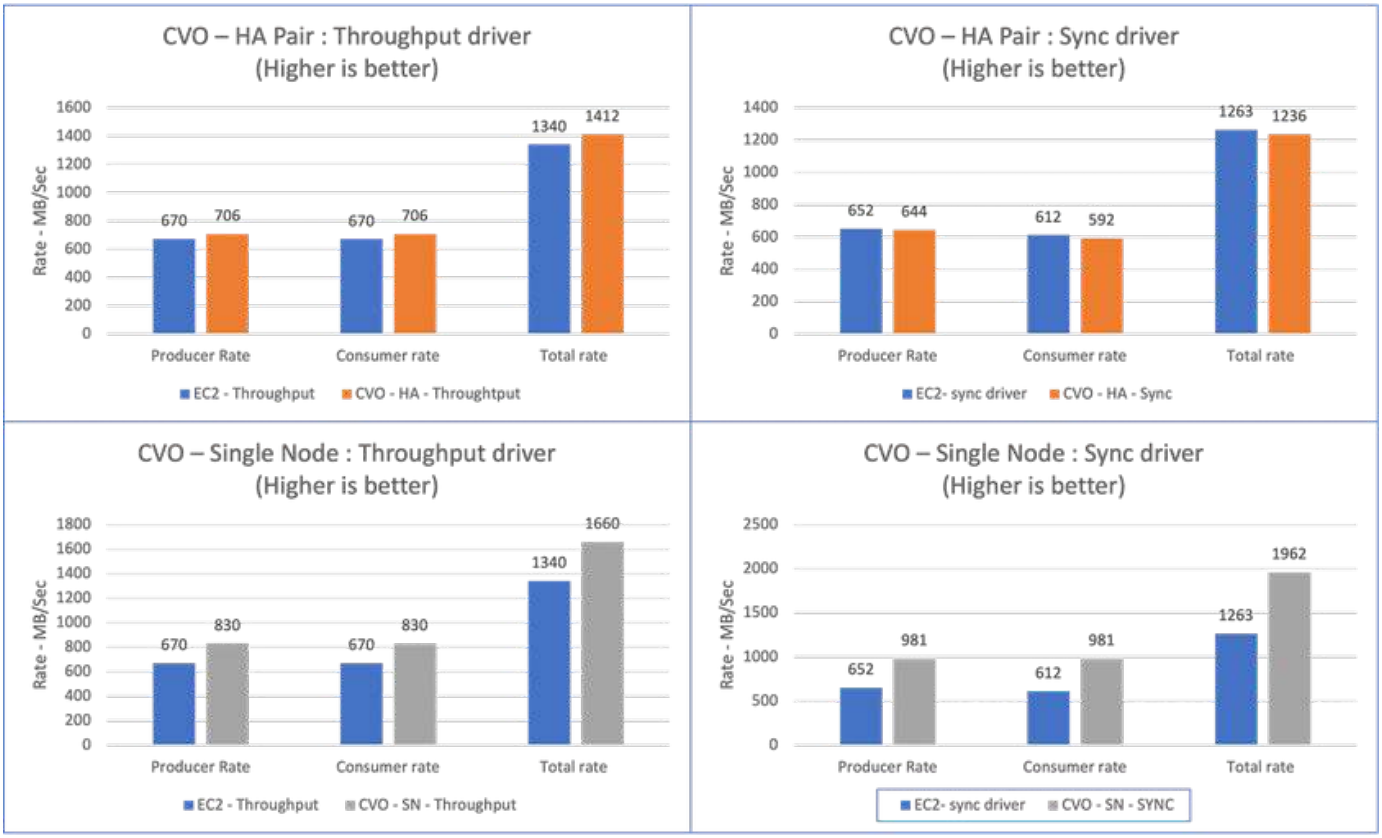
- Débit total généré de manière cohérente par le pilote de synchronisation : environ 1236 Mbit/s.
- Débit total généré pour le pilote de débit : pic de ~1412 Mbit/s.

Pour un seul nœud Cloud Volumes ONTAP :

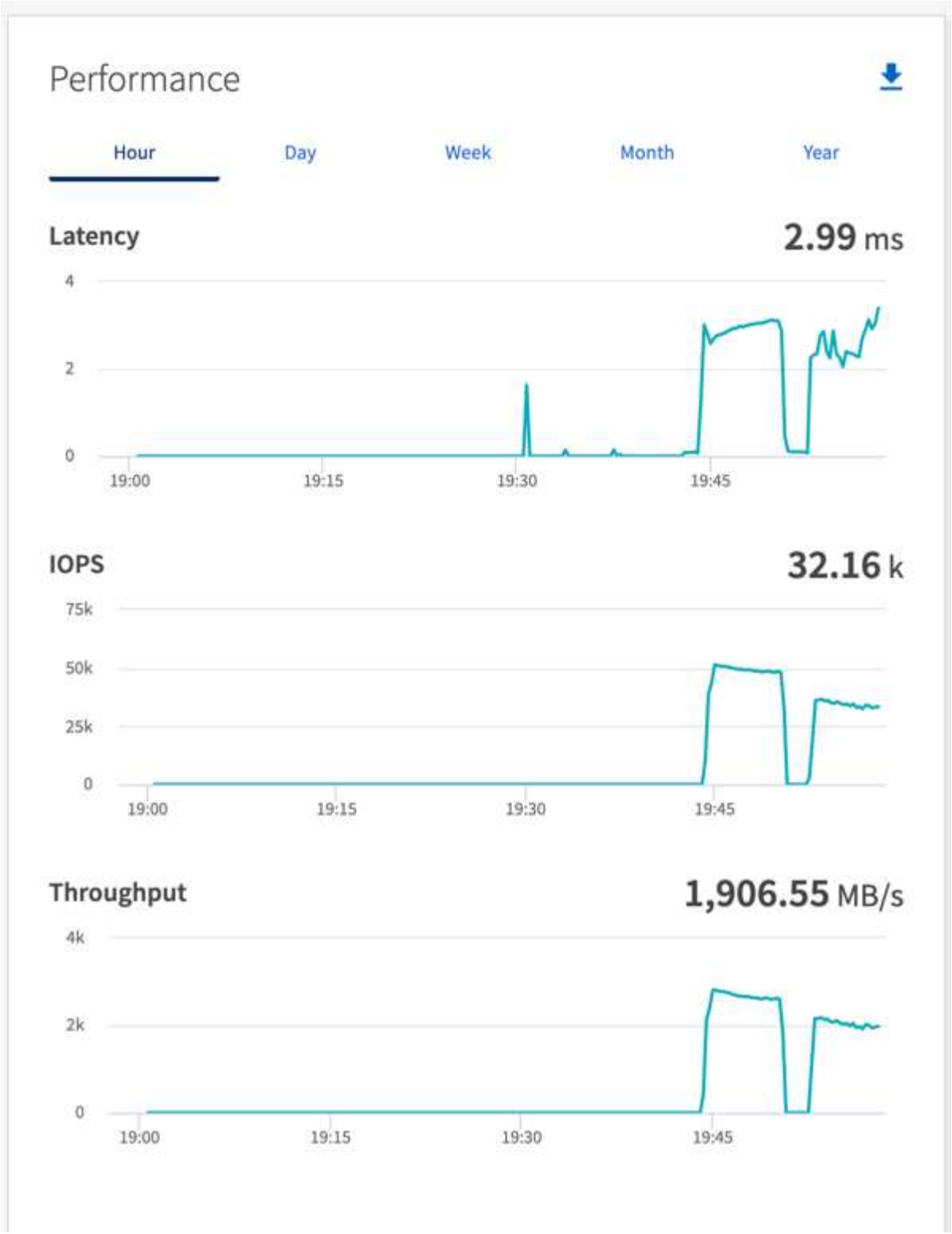
- Débit total généré de manière cohérente par le pilote Sync : ~ 1962 Mbit/s.
- Débit total généré par le pilote de débit : pic d'environ 1660 Mbit/s.

Le pilote de synchronisation peut générer un débit constant lorsque les journaux sont immédiatement transmis au disque, tandis que le pilote de débit génère des pics de débit lorsque les journaux sont validés sur le disque en bloc.

Ces valeurs de débit sont générées pour la configuration AWS appropriée. Pour des besoins de performances plus élevés, il est possible de renforcer l'évolutivité des types d'instances et de les ajuster davantage pour obtenir un meilleur débit. Le débit total ou le taux total est la combinaison du taux de production et du taux de consommation.




Vérifiez le débit de stockage lorsque vous effectuez une évaluation du débit ou du pilote de synchronisation.




## Présentation des performances et validation dans AWS FSX pour NetApp ONTAP

Un cluster Kafka avec la couche de stockage montée sur NetApp NFS a été testé sur les performances dans AWS FSX pour NetApp ONTAP. Les exemples de benchmarking sont décrits dans les sections suivantes.

## Apache Kafka dans AWS FSX pour NetApp ONTAP

NFS (Network File System) est un système de fichiers réseau largement utilisé pour stocker de grandes quantités de données. Dans la plupart des entreprises, les données sont de plus en plus générées par des applications de streaming telles qu'Apache Kafka. Ces charges de travail nécessitent une évolutivité, une faible latence et une architecture d'ingestion robuste des données dotée de fonctionnalités de stockage modernes. Pour activer l'analytique en temps réel et fournir des informations exploitables, une infrastructure bien conçue et haute performance est nécessaire.

Par conception, Kafka fonctionne avec un système de fichiers conforme POSIX et s'appuie sur le système de fichiers pour traiter les opérations de fichiers. Cependant, lors du stockage des données sur un système de fichiers NFSv3, le client NFS du courtier Kafka peut interpréter les opérations de fichiers différemment d'un système de fichiers local tel que XFS ou Ext4. Le renommage NFS entraîne une défaillance des courtiers Kafka lors de l'extension des clusters et de la réaffectation des partitions, ce qui en est un exemple courant. Pour faire face à ce défi, NetApp a mis à jour le client Linux NFS open-source avec des modifications désormais généralement disponibles dans RHEL8.7, RHEL9.1, et prises en charge à partir de la version actuelle de FSX for NetApp ONTAP, ONTAP 9.12.1.

Amazon FSX pour NetApp ONTAP fournit un système de fichiers NFS entièrement géré, évolutif et haute performance dans le cloud. Les données Kafka sur FSX pour NetApp ONTAP peuvent être évolutives afin de traiter d'importants volumes de données et d'assurer la tolérance aux pannes. NFS assure la gestion centralisée du stockage et la protection des données pour les datasets stratégiques et sensibles.

Ces améliorations permettent aux clients AWS d'exploiter FSX for NetApp ONTAP lors de l'exécution des workloads Kafka sur les services de calcul AWS. Ces avantages sont les suivants :
* Réduction de l'utilisation du CPU pour réduire le temps d'attente d'E/S.
* Délai de récupération du courtier Kafka plus rapide.
* Fiabilité et efficacité.
* Évolutivité et performances.
* Disponibilité de la zone de disponibilité multiple.
* Protection des données.

### Présentation des performances et validation dans AWS FSX pour NetApp ONTAP

Les performances dans le cloud AWS ont été évaluées sur un cluster Kafka avec la couche de stockage montée sur NetApp NFS. Les exemples de benchmarking sont décrits dans les sections suivantes.

#### Kafka dans AWS FSX pour NetApp ONTAP

Les performances dans le cloud AWS ont été évaluées à l'aide d'un cluster Kafka avec AWS FSX pour NetApp ONTAP. Cette analyse comparative est décrite dans les sections suivantes.

#### Installation architecturale

Le tableau suivant présente la configuration environnementale d'un cluster Kafka à l'aide d'AWS FSX pour NetApp ONTAP.

| Composant de plate-forme | Configuration de l'environnement |
|---|---|
| Kafka 3.2.3 | • 3 x zookeepers – t2.small<br>• 3 serveurs de broker – i3en.2xlarge<br>• 1 x Grafana – c5n.2xlarge<br>• 4 x producteur/consommateur — c5n.2xlarge * |
| Système d'exploitation sur tous les nœuds | RHEL8.6 |
| AWS FSX pour NetApp ONTAP | Plusieurs zones de disponibilité avec un débit de 4 Go/s et 160000 000 IOPS |

#### Configuration de NetApp FSX pour NetApp ONTAP

1. Pour nos premiers tests, nous avons créé un système de fichiers FSX pour NetApp ONTAP d'une capacité de 2 To et de 40000 000 IOPS pour un débit de 2 Go/s.

   ```
   [root@ip-172-31-33-69 ~]# aws fsx create-file-system --region us-east-2
   --storage-capacity 2048 --subnet-ids <desired subnet 1> subnet-<desired
   subnet 2> --file-system-type ONTAP --ontap-configuration
   DeploymentType=MULTI_AZ_HA_1,ThroughputCapacity=2048,PreferredSubnetId=<
   desired primary subnet>,FsxAdminPassword=<new
   password>,DiskIopsConfiguration="{Mode=USER_PROVISIONED,Iops=40000"}
   ```

   Dans notre exemple, nous déployons FSX pour NetApp ONTAP via l'interface de ligne de commande AWS. Vous devrez personnaliser davantage la commande dans votre environnement, si nécessaire. FSX pour NetApp ONTAP peut également être déployé et géré via la console AWS pour une expérience de déploiement plus simple et rationalisée avec moins d'entrées de ligne de commande.

   Documentation dans FSX for NetApp ONTAP, le nombre maximal d'IOPS pouvant être atteint pour un système de fichiers à débit de 2 Go/s dans notre région de test (US-East-1) est de 80,000 000 iops. les iops maximales totales pour un système de fichiers FSX for NetApp ONTAP sont de 160,000 000 iops, ce qui requiert un déploiement à un débit de 4 Go/s, comme nous le montrerons plus loin dans ce document.

   Pour plus d'informations sur les spécifications de performances de FSX for NetApp ONTAP, n'hésitez pas à consulter la documentation d'AWS FSX for NetApp ONTAP ici : https://docs.aws.amazon.com/fsx/latest/ONTAPGuide/performance.html .

   Vous trouverez la syntaxe détaillée de la ligne de commande de FSX « create-file-system » ici : https://docs.aws.amazon.com/cli/latest/reference/fsx/create-file-system.html

   Par exemple, vous pouvez spécifier une clé KMS spécifique, par opposition à la clé principale AWS FSX par défaut utilisée lorsqu'aucune clé KMS n'est spécifiée.

2. Lors de la création du système de fichiers FSX pour NetApp ONTAP, attendez que le statut « cycle de vie » passe à « DISPONIBLE » dans votre retour JSON après avoir décrit votre système de fichiers comme suit :

```
[root@ip-172-31-33-69 ~]# aws fsx describe-file-systems  --region us-
east-1 --file-system-ids fs-02ff04bab5ce01c7c
```

3. Validez les informations d'identification en vous connectant à FSX for NetApp ONTAP SSH avec l'utilisateur fsxadmin :
   Fsxadmin est le compte admin par défaut pour les systèmes de fichiers FSX for NetApp ONTAP lors de leur création. Le mot de passe pour fsxadmin est le mot de passe qui a été configuré lors de la création du système de fichiers dans la console AWS ou via l'interface de ligne de commande AWS, comme nous l'avons fait à l'étape 1.

```
[root@ip-172-31-33-69 ~]# ssh fsxadmin@198.19.250.244
The authenticity of host '198.19.250.244 (198.19.250.244)' can't be
established.
ED25519 key fingerprint is
SHA256:mgCyRXJfWRc2d/jOjFbMBsUcYOWjxoIky0ltHvVDL/Y.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '198.19.250.244' (ED25519) to the list of
known hosts.
(fsxadmin@198.19.250.244) Password:

This is your first recorded login.
```

4. Une fois vos identifiants validés, créez l'ordinateur virtuel de stockage sur le système de fichiers FSX pour NetApp ONTAP

```
[root@ip-172-31-33-69 ~]# aws fsx --region us-east-1 create-storage-
virtual-machine --name svmkafkatest --file-system-id fs-
02ff04bab5ce01c7c
```

   Un SVM (Storage Virtual machine) est un serveur de fichiers isolé avec ses propres identifiants d'administration et terminaux pour l'administration et l'accès aux données dans FSX pour les volumes NetApp ONTAP. Il fournit également la colocation FSX pour NetApp ONTAP.

5. Une fois que vous avez configuré votre machine virtuelle de stockage primaire, SSH dans le nouveau système de fichiers FSX pour NetApp ONTAP et créez des volumes dans la machine virtuelle de stockage à l'aide de l'exemple de commande ci-dessous. De la même manière, nous créons 6 volumes pour cette validation. En fonction de notre validation, vous devez conserver le composant par défaut (8) ou moins, ce qui améliore les performances de kafka.

```
FsxId02ff04bab5ce01c7c::*> volume create -volume kafkafsxN1 -state
online -policy default -unix-permissions ---rwxr-xr-x -junction-active
true -type RW -snapshot-policy none  -junction-path /kafkafsxN1 -aggr
-list aggr1
```

6. Nous aurons besoin de capacité supplémentaire dans nos volumes pour nos tests. Étendez la taille du volume à 2 To et montez sur le chemin de jonction.

```
FsxId02ff04bab5ce01c7c::*> volume size -volume kafkafsxN1 -new-size +2TB
vol size: Volume "svmkafkatest:kafkafsxN1" size set to 2.10t.

FsxId02ff04bab5ce01c7c::*> volume size -volume kafkafsxN2 -new-size +2TB
vol size: Volume "svmkafkatest:kafkafsxN2" size set to 2.10t.

FsxId02ff04bab5ce01c7c::*> volume size -volume kafkafsxN3 -new-size +2TB
vol size: Volume "svmkafkatest:kafkafsxN3" size set to 2.10t.

FsxId02ff04bab5ce01c7c::*> volume size -volume kafkafsxN4 -new-size +2TB
vol size: Volume "svmkafkatest:kafkafsxN4" size set to 2.10t.

FsxId02ff04bab5ce01c7c::*> volume size -volume kafkafsxN5 -new-size +2TB
vol size: Volume "svmkafkatest:kafkafsxN5" size set to 2.10t.

FsxId02ff04bab5ce01c7c::*> volume size -volume kafkafsxN6 -new-size +2TB
vol size: Volume "svmkafkatest:kafkafsxN6" size set to 2.10t.

FsxId02ff04bab5ce01c7c::*> volume show -vserver svmkafkatest -volume *
Vserver   Volume       Aggregate    State      Type       Size
Available Used%
--------- ------------ ------------ ---------- ---- ----------
---------- -----
svmkafkatest
          kafkafsxN1   -            online     RW       2.10TB
1.99TB    0%
svmkafkatest
          kafkafsxN2   -            online     RW       2.10TB
1.99TB    0%
svmkafkatest
          kafkafsxN3   -            online     RW       2.10TB
1.99TB    0%
svmkafkatest
          kafkafsxN4   -            online     RW       2.10TB
1.99TB    0%
svmkafkatest
          kafkafsxN5   -            online     RW       2.10TB
1.99TB    0%
svmkafkatest
          kafkafsxN6   -            online     RW       2.10TB
1.99TB    0%
svmkafkatest
          svmkafkatest_root
```

```
                        aggr1        online     RW          1GB
968.1MB    0%
7 entries were displayed.

FsxId02ff04bab5ce01c7c::*> volume mount -volume kafkafsxN1 -junction
-path /kafkafsxN1

FsxId02ff04bab5ce01c7c::*> volume mount -volume kafkafsxN2 -junction
-path /kafkafsxN2

FsxId02ff04bab5ce01c7c::*> volume mount -volume kafkafsxN3 -junction
-path /kafkafsxN3

FsxId02ff04bab5ce01c7c::*> volume mount -volume kafkafsxN4 -junction
-path /kafkafsxN4

FsxId02ff04bab5ce01c7c::*> volume mount -volume kafkafsxN5 -junction
-path /kafkafsxN5

FsxId02ff04bab5ce01c7c::*> volume mount -volume kafkafsxN6 -junction
-path /kafkafsxN6
```

Dans FSX for NetApp ONTAP, les volumes peuvent être à provisionnement fin. Dans notre exemple, la capacité totale du volume étendu dépasse la capacité totale du système de fichiers. Nous devrons donc étendre la capacité totale du système de fichiers afin de déverrouiller la capacité supplémentaire du volume provisionné, comme nous le démontrerons à l'étape suivante.

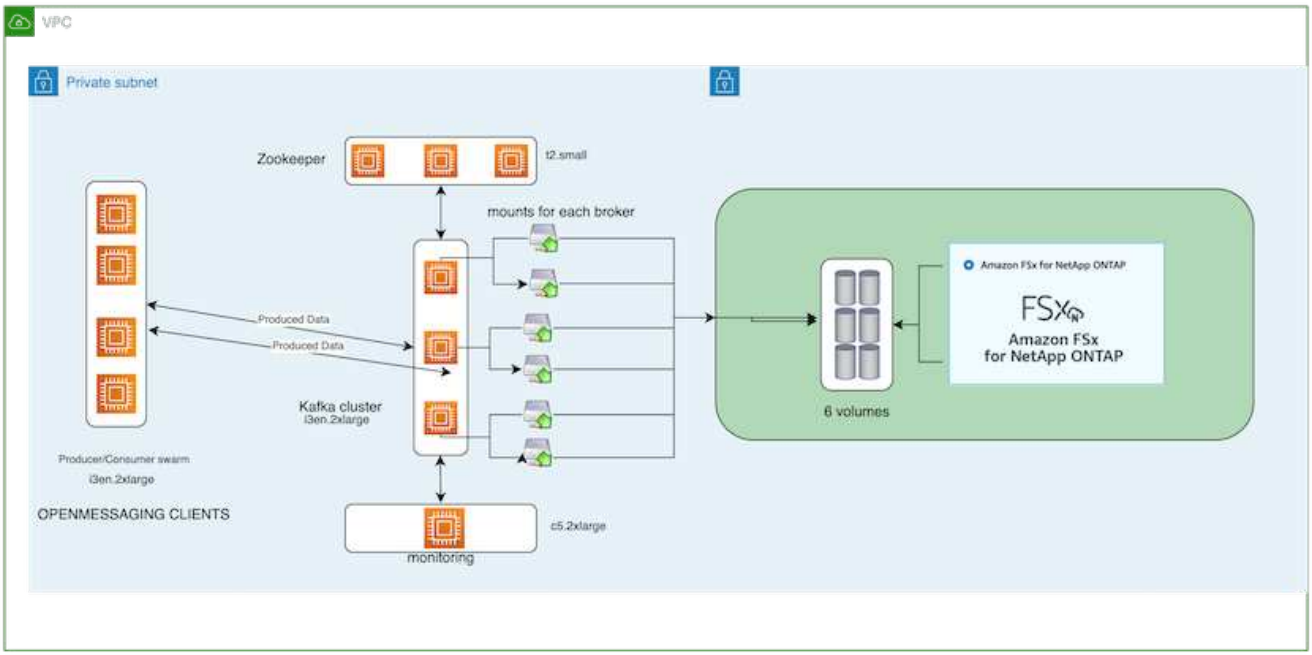
7. Ensuite, pour des performances et une capacité supplémentaires, nous étendons la capacité de débit de FSX for NetApp ONTAP de 2 Go/s à 4 Go/s et d'IOPS à 160000, et la capacité à 5 To

```
[root@ip-172-31-33-69 ~]# aws fsx update-file-system --region us-east-1
--storage-capacity 5120 --ontap-configuration
'ThroughputCapacity=4096,DiskIopsConfiguration={Mode=USER_PROVISIONED,Io
ps=160000}' --file-system-id fs-02ff04bab5ce01c7c
```

Vous trouverez la syntaxe détaillée de la ligne de commande du système de fichiers « update-file-system » de FSX ici :
https://docs.aws.amazon.com/cli/latest/reference/fsx/update-file-system.html

8. Les volumes FSX pour NetApp ONTAP sont montés avec nconnect et les ions par défaut dans les courtiers Kafka

La figure suivante présente l'architecture finale d'un cluster Kafka à base de FSX pour NetApp ONTAP :




- Calcul. Nous avons utilisé un cluster Kafka à trois nœuds avec un ensemble de zoocontrôle à trois nœuds qui s'exécute sur des serveurs dédiés. Chaque courtier disposait de six points de montage NFS pour six volumes sur l'instance FSX pour NetApp ONTAP.
- Contrôle. Nous avons utilisé deux nœuds pour une combinaison Prometheus-Grafana. Pour la génération des charges de travail, nous avons utilisé un cluster séparé à trois nœuds qui était capable de produire et de consommer sur ce cluster Kafka.
- Stockage. Nous avons utilisé FSX pour NetApp ONTAP avec six volumes de 2 To montés. Le volume a ensuite été exporté vers le courtier Kafka avec un montage NFS. Les volumes FSX pour NetApp ONTAP sont montés avec 16 sessions nconnect et les options par défaut dans les courtiers Kafka.

**Configurations de test OpenMessage.**

Nous avons utilisé la même configuration que celle utilisée pour le NetApp Cloud Volumes ONTAP et leurs détails sont ici - https://docs.netapp.com/us-en/netapp-solutions/data-analytics/kafka-nfs-performance-overview-and-validation-in-aws.html#architectural-setup

**Méthodologie de test**

1. Un cluster Kafka a été provisionné selon la spécification décrite ci-dessus à l'aide d'un système téraforme et ansible. Terraform est utilisé pour créer l'infrastructure à l'aide d'instances AWS pour le cluster Kafka et ansible pour y construire le cluster Kafka.
2. Une charge de travail OMB a été déclenchée avec la configuration de la charge de travail décrite ci-dessus et le pilote Sync.

```
sudo bin/benchmark -drivers driver-kafka/kafka-sync.yaml workloads/1-
topic-100-partitions-1kb.yaml
```

3. Une autre charge de travail a été déclenchée avec le pilote de débit avec la même configuration de charge de travail.

```
sudo bin/benchmark -drivers driver-kafka/kafka-throughput.yaml
workloads/1-topic-100-partitions-1kb.yaml
```

**Observation**

Deux types de pilotes différents ont été utilisés pour générer des charges de travail afin de tester les performances d'une instance Kafka fonctionnant sur NFS. La différence entre les pilotes est la propriété log flush.

Pour un facteur de réplication Kafka 1 et FSX pour NetApp ONTAP :

- Débit total généré de manière cohérente par le pilote de synchronisation : environ 3218 Mbit/s et performances maximales d'environ 3652 Mbit/s.
- Débit total généré de manière cohérente par le pilote de débit : environ 3679 Mbit/s et performances de pointe d'environ 3908 Mbit/s.

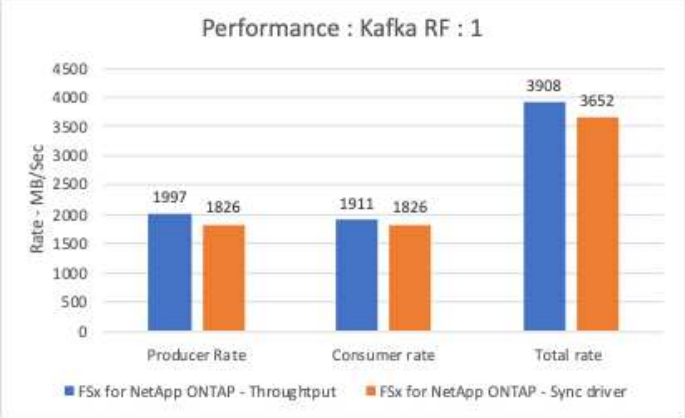
Pour Kafka avec facteur de réplication 3 et FSX pour NetApp ONTAP :

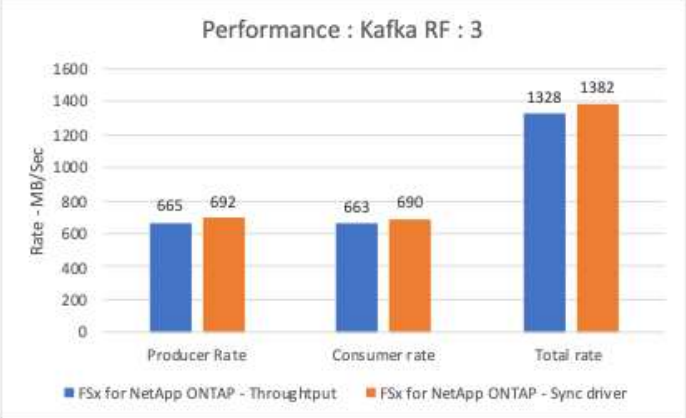
- Débit total généré de manière cohérente par le pilote de synchronisation : environ 1252 Mbit/s et performances maximales d'environ 1382 Mbit/s.
- Débit total généré de manière cohérente par le pilote de débit : environ 1218 Mbit/s et performances de pointe d'environ 1328 Mbit/s.

Dans le facteur de réplication Kafka 3, l'opération de lecture et d'écriture s'est produite trois fois sur FSX pour NetApp ONTAP. Dans le facteur de réplication Kafka 1, l'opération de lecture et d'écriture est une fois sur FSX pour NetApp ONTAP, donc dans les deux cas de validation, Nous avons pu atteindre un débit maximal de 4 Go/s.

Le pilote de synchronisation peut générer un débit constant lorsque les journaux sont immédiatement transmis au disque, tandis que le pilote de débit génère des pics de débit lorsque les journaux sont validés sur le disque en bloc.

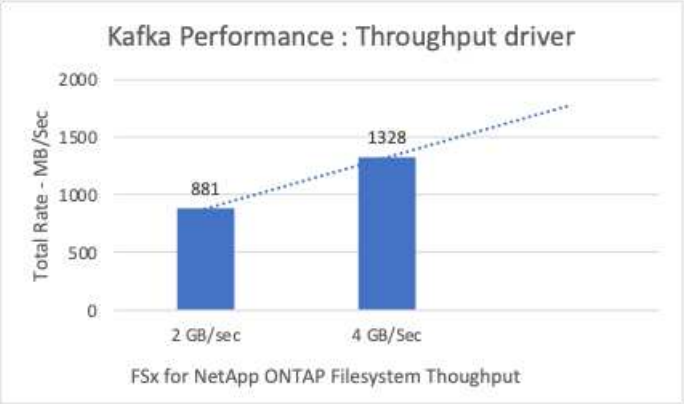
Ces valeurs de débit sont générées pour la configuration AWS appropriée. Pour des besoins de performances plus élevés, il est possible de renforcer l'évolutivité des types d'instances et de les ajuster davantage pour obtenir un meilleur débit. Le débit total ou le taux total est la combinaison du taux de production et du taux de consommation.

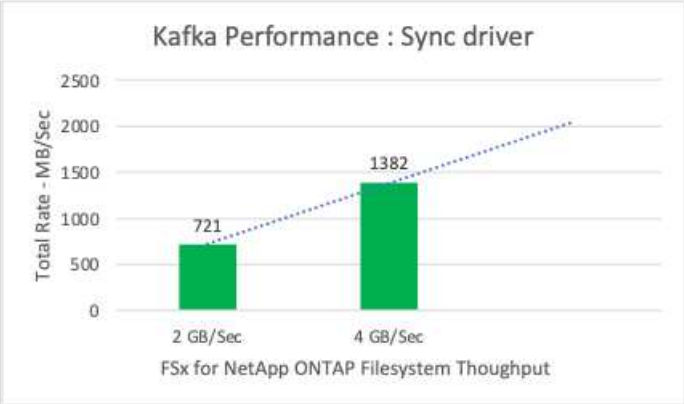


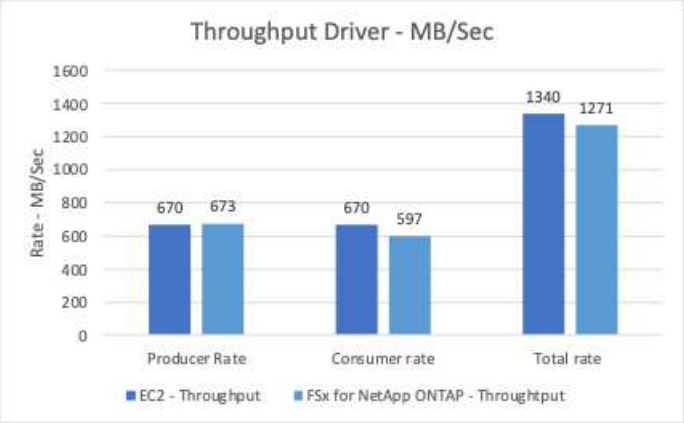

Le tableau ci-dessous présente la FSX 2 Go/s pour NetApp ONTAP et la performance de 4 Go/s pour le

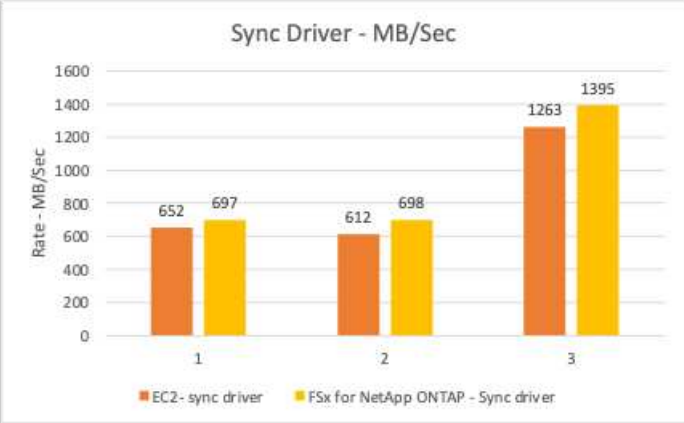
facteur de réplication Kafka 3. Le facteur de réplication 3 effectue l'opération de lecture et d'écriture trois fois sur FSX pour le stockage NetApp ONTAP. Le débit total du pilote de débit est de 881 Mo/s, ce qui permet de lire et d'écrire le fonctionnement Kafka à environ 2.64 Go/s sur le système de fichiers FSX for NetApp ONTAP de 2 Go/s. Le débit total du pilote de débit est de 1328 Mo/s, ce qui permet de lire et d'écrire le fonctionnement kafka à environ 3.98 Go/s. Les performances de Kafka sont linéaires et évolutives en fonction du débit de FSX pour NetApp ONTAP.




Le tableau ci-dessous présente les performances entre une instance EC2 et FSX pour NetApp ONTAP (facteur de réplication Kafka : 3)

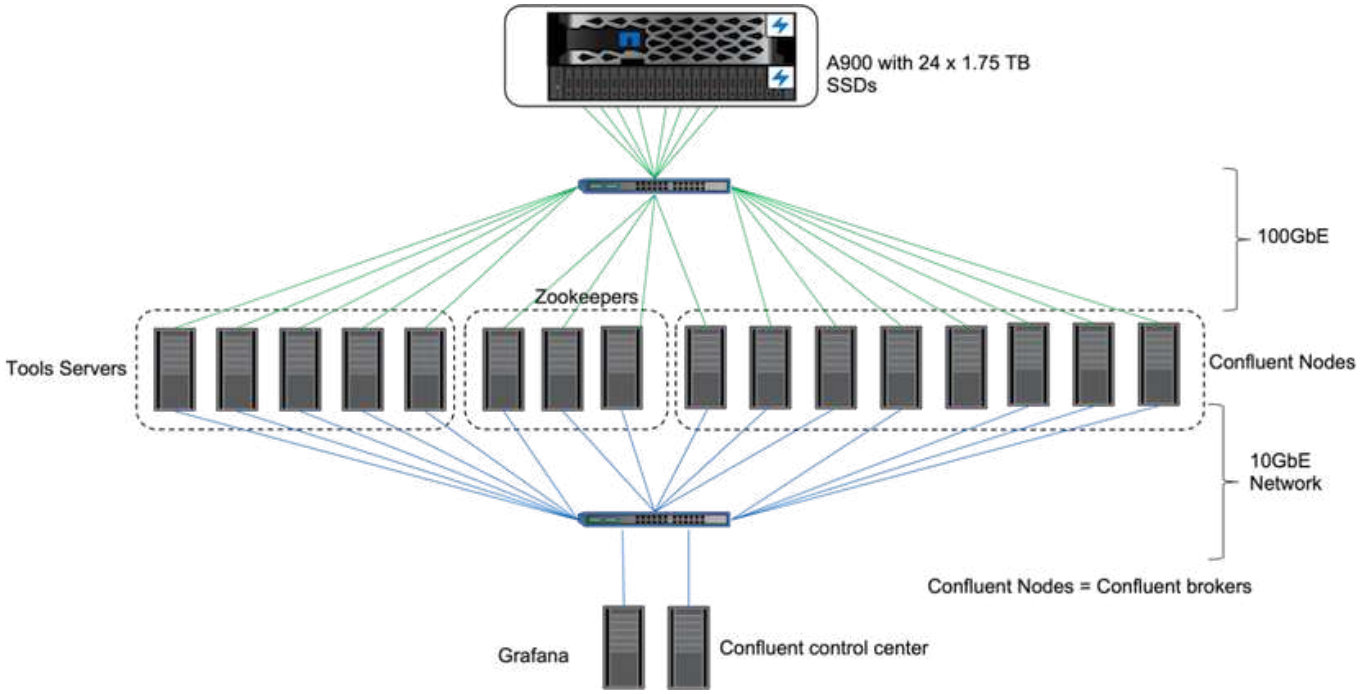



# Présentation des performances et validation avec AFF A900 sur site

Sur site, nous avons utilisé le contrôleur de stockage NetApp AFF A900 avec ONTAP 9.12.1RC1 pour valider les performances et l'évolutivité d'un cluster Kafka. Avec ONTAP et AFF, nous avons utilisé le même banc d'essai que nos précédentes meilleures pratiques en matière de stockage à plusieurs niveaux.

Nous avons utilisé Kafka 6.2.0 confluent pour évaluer le système AFF A900. Le cluster comporte huit nœuds de broker et trois nœuds de zookeeper. Pour les tests de performances, nous avons utilisé cinq nœuds workers OMB.




## Configuration de stockage sous-jacente

Nous avons utilisé les instances NetApp FlexGroups pour fournir un namespace unique pour les répertoires de journaux, ce qui simplifie la restauration et la configuration. Nous avons utilisé NFSv4.1 et pNFS pour fournir un accès direct aux données des segments de journal.

## Réglage du client

Chaque client a monté l'instance FlexGroup avec la commande suivante.

```
mount -t nfs -o vers=4.1,nconnect=16 172.30.0.121:/kafka_vol01
/data/kafka_vol01
```

En outre, nous avons augmenté le `max_session_slots` par défaut `64` à `180`. Ceci correspond à la limite d'emplacement de session par défaut dans ONTAP.

## Réglage du courtier Kafka

Pour optimiser le débit du système testé, nous avons considérablement augmenté les paramètres par défaut de certains pools de threads clés. Nous recommandons de suivre les bonnes pratiques relatives à Kafka plus fluides pour la plupart des configurations. Ce réglage a été utilisé pour optimiser la simultanéité des E/S en attente au stockage. Ces paramètres peuvent être ajustés pour correspondre aux ressources de calcul et aux attributs de stockage de votre courtier.

```
num.io.threads=96
num.network.threads=96
background.threads=20
num.replica.alter.log.dirs.threads=40
num.replica.fetchers=20
queued.max.requests=2000
```

## Méthodologie de test du générateur de charges de travail

Nous avons utilisé les mêmes configurations OMB que pour les tests de Cloud pour le pilote de débit et la configuration de rubrique.

1. Une instance FlexGroup a été provisionnée à l'aide d'Ansible sur un cluster AFF.

```
---
- name: Set up kafka broker processes
  hosts: localhost
  vars:
    ntap_hostname: ‘hostname’
    ntap_username: 'user'
    ntap_password: 'password'
    size: 10
    size_unit: tb
    vserver: vs1
    state: present
    https: true
    export_policy: default
    volumes:
      - name: kafka_fg_vol01
        aggr: ["aggr1_a", "aggr2_a", “aggr1_b”, “aggr2_b”]
        path: /kafka_fg_vol01
  tasks:
    - name: Edit volumes
      netapp.ontap.na_ontap_volume:
        state: "{{ state }}"
        name: "{{ item.name }}"
        aggr_list: "{{ item.aggr }}"
        aggr_list_multiplier: 8
        size: "{{ size }}"
        size_unit: "{{ size_unit }}"
        vserver: "{{ vserver }}"
        snapshot_policy: none
        export_policy: default
        junction_path: "{{ item.path }}"
        qos_policy_group: none
        wait_for_completion: True
        hostname: "{{ ntap_hostname }}"
        username: "{{ ntap_username }}"
        password: "{{ ntap_password }}"
        https: "{{ https }}"
        validate_certs: false
      connection: local
      with_items: "{{ volumes }}"
```

2. PNFS a été activé sur le SVM ONTAP.

```
vserver modify -vserver vs1 -v4.1-pnfs enabled -tcp-max-xfer-size 262144
```

3. La charge de travail a été déclenchée avec le pilote de débit utilisant la même configuration de charge de travail que pour Cloud Volumes ONTAP. Voir la section «Performances à l'état stable» ci-dessous. La charge de travail utilisait un facteur de réplication de 3, ce qui signifie que trois copies des segments de journal ont été conservées dans NFS.

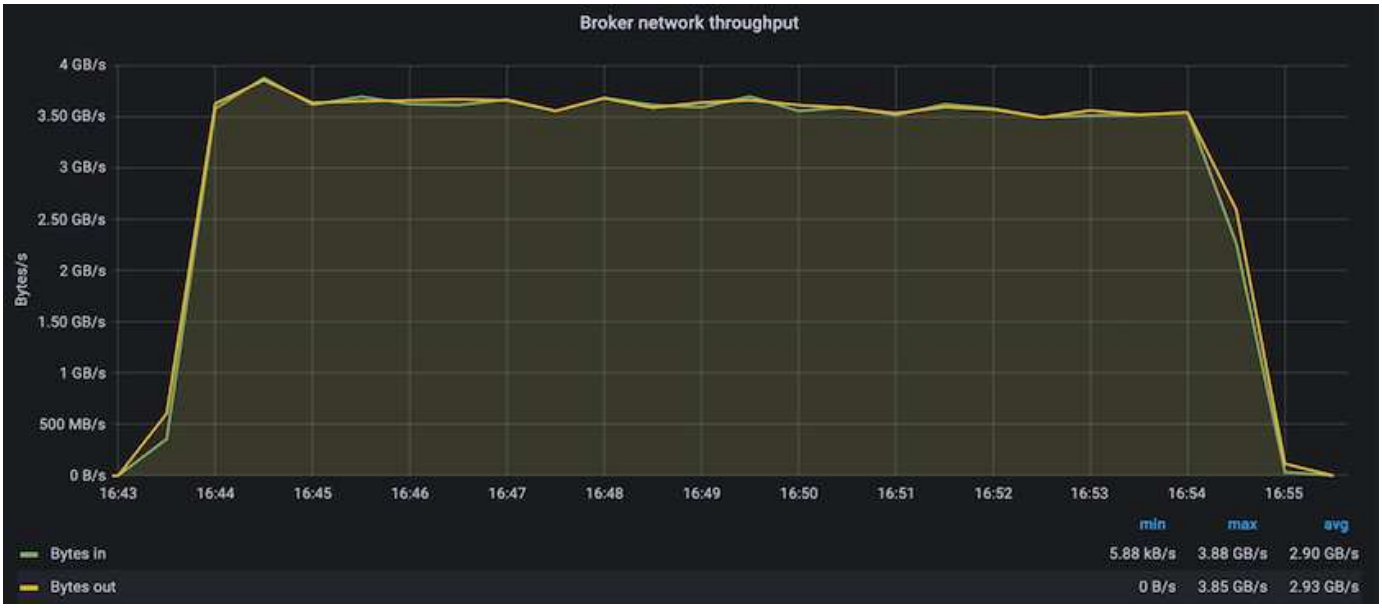
```
sudo bin/benchmark --drivers driver-kafka/kafka-throughput.yaml
workloads/1-topic-100-partitions-1kb.yaml
```

4. Enfin, nous avons réalisé des mesures en utilisant un carnet de commandes pour mesurer la capacité des consommateurs à rattraper les derniers messages. L'OMB crée un arriéré en interrompant les consommateurs au début d'une mesure. Cela produit trois phases distinctes : la création de carnet de commandes (trafic uniquement pour les producteurs), la purge des arriérés (une phase lourde pour les consommateurs dans laquelle les consommateurs rattrapent les événements manqués dans un sujet) et l'état stable. Voir la section «exploration des limites de stockage" pour plus d'informations.

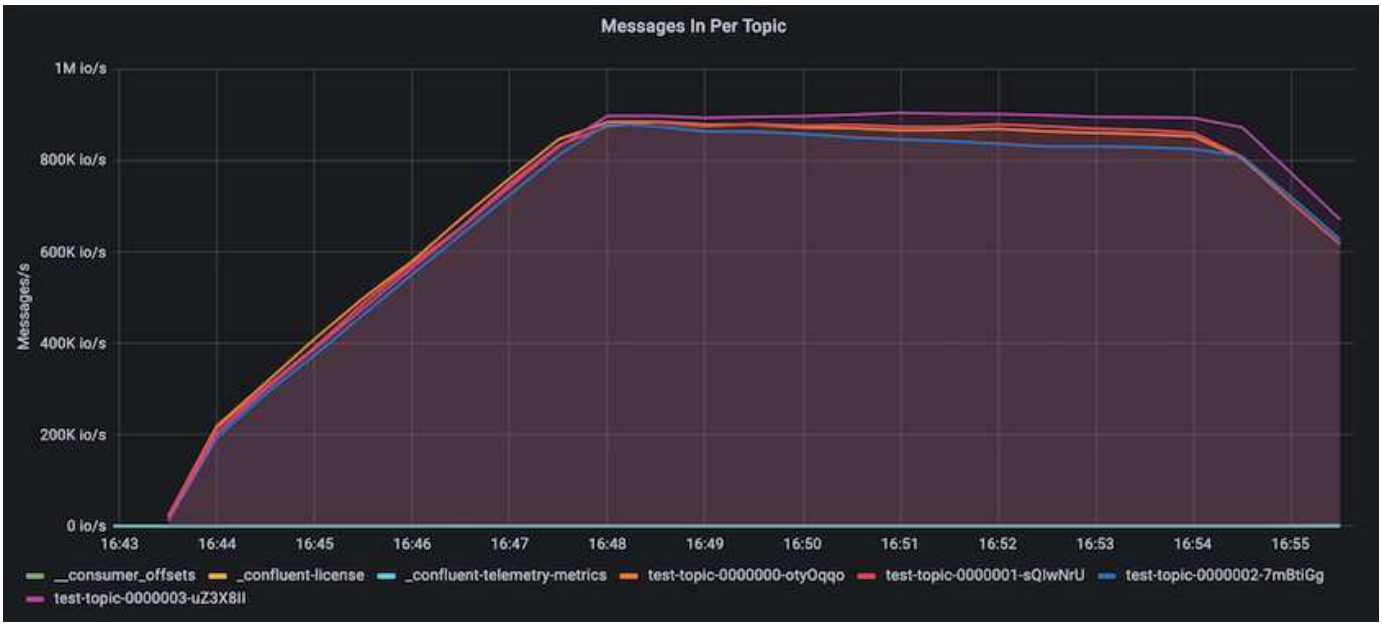
**Performances à l'état stable**

Nous avons évalué le système AFF A900 à l'aide du banc d'essai OpenMessaging afin de fournir une comparaison similaire à celle de Cloud Volumes ONTAP dans AWS et du DAS dans AWS. Toutes les valeurs de performance correspondent au débit du cluster Kafka au niveau de la production et de l'utilisateur.

Grâce à Kafka et au AFF A900, les performances stables ont atteint un débit moyen supérieur à 3,4 Gbit/s pour les producteurs et les consommateurs. Il s'agit de plus de 3.4 millions de messages sur le cluster Kafka. En visualisant le débit continu en octets par seconde pour BrokerTopicMetrics, nous constatons l'excellente performance à l'état stable et le trafic pris en charge par le système AFF A900.




Cela s'aligne parfaitement sur la vue des messages transmis par sujet. Le graphique suivant présente une répartition par thème. Dans la configuration testée, nous avons vu près de 900 000 messages par sujet sur quatre sujets.

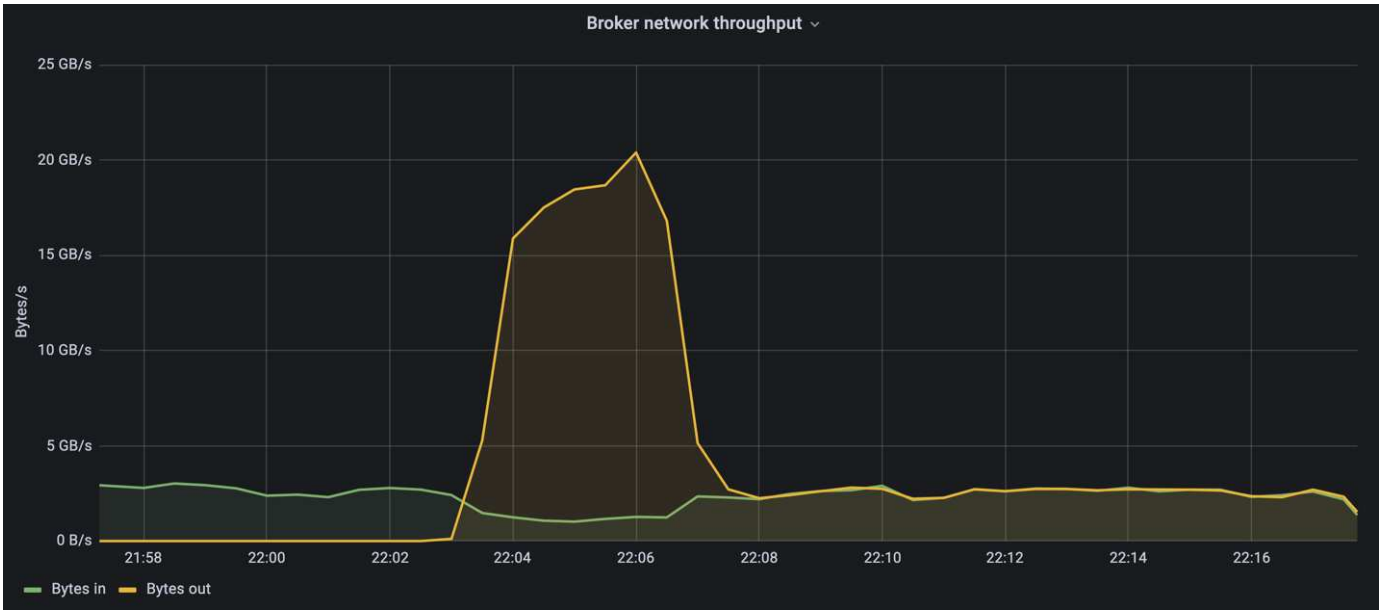



## Des performances extrêmes et l'exploration des limites de stockage

Pour AFF, nous avons également testé OMB en utilisant la fonction backlog. La fonction backlog met en pause les abonnements des consommateurs alors qu'un backlog d'événements est créé dans le cluster Kafka. Au cours de cette phase, seul le trafic producteur se produit, ce qui génère des événements qui sont validés dans les journaux. Cela émule le plus étroitement les flux de travail de traitement par lots ou d'analyse hors ligne. Dans ces flux de travail, les abonnements client sont démarrés et doivent lire les données historiques qui ont déjà été supprimées du cache du courtier.

Pour comprendre les limites de stockage du débit consommateur dans cette configuration, nous avons mesuré la phase réservée à la production afin de déterminer la quantité de trafic d'écriture que le système A900 pourrait absorber. Voir la section suivante «Conseils de dimensionnementpour comprendre comment exploiter ces données.

Lors de la partie réservée aux producteurs de cette mesure, nous avons constaté un débit de pointe élevé qui a repoussé les limites des performances d'A900 (lorsque les autres ressources des courtiers n'étaient pas saturées pour desservir le trafic des producteurs et du consommateur).



La taille des messages a été portée à 16 000 pour cette mesure, afin de limiter les frais par message et d'optimiser le débit de stockage aux points de montage NFS.

```
messageSize: 16384
consumerBacklogSizeGB: 4096
```

Le cluster Kafka confluent a atteint un débit producteur maximal de 4,03 Gbit/s.

```
18:12:23.833 [main] INFO WorkloadGenerator - Pub rate 257759.2 msg/s /
4027.5 MB/s | Pub err     0.0 err/s …
```

Une fois que l'OMB a terminé de remplir le carnet de commandes d'événements, le trafic des consommateurs a été redémarré. Au cours des mesures avec une vidange de l'arriéré, nous avons observé un débit de consommation maximal de plus de 20 Gbit/s sur tous les sujets. Le débit combiné du volume NFS qui stocke les données du journal OMB est d'environ 30 Gbit/s.

## Conseils de dimensionnement

Amazon Web Services propose une "guide de dimensionnement" Pour le dimensionnement et l'évolutivité du cluster Kafka.

Ce dimensionnement constitue une formule utile pour déterminer les besoins en débit du stockage pour votre cluster Kafka :

Pour un débit agrégé produit dans le cluster de tcluster avec un facteur de réplication de r, le débit reçu par le stockage du courtier est le suivant :

```
t[storage] = t[cluster]/#brokers + t[cluster]/#brokers * (r-1)
          = t[cluster]/#brokers * r
```

Cela peut encore être simplifié :

```
max(t[cluster]) <= max(t[storage]) * #brokers/r
```

Cette formule vous permet de sélectionner la plateforme ONTAP adaptée à vos besoins en matière de Tier actif Kafka.

Le tableau suivant explique le débit producteur anticipé pour le système A900 avec différents facteurs de réplication :

| Facteur de réplication | Débit producteur (GPP) |
|---|---|
| 3 (mesuré) | 3.4 |
| 2 | 5.1 |
| 1 | 10.2 |

# Conclusion

La solution NetApp pour résoudre le problème du changement de nom soyeux offre une forme de stockage simple, économique et gérée de manière centralisée pour les charges de travail qui étaient auparavant incompatibles avec NFS.

Ce nouveau modèle permet aux clients de créer des clusters Kafka plus faciles à migrer et à mettre en miroir à des fins de reprise après incident et de protection des données.
Nous avons également constaté que NFS offre des avantages supplémentaires, tels que la réduction de l'utilisation du processeur et une restauration plus rapide, une efficacité du stockage considérablement améliorée et des performances supérieures grâce à NetApp ONTAP.

# 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 qu'Apache Kafka ?

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

- Qu'est-ce qu'un changement de nom ?

  "https://linux-nfs.org/wiki/index.php/Server-side_silly_rename"

- ONAT est lu pour les applications de streaming.

  "https://www.netapp.com/blog/ontap-ready-for-streaming-applications/"

- Problème de changement de nom avec Kafka

  "https://sbg.technology/2018/07/10/kafka-nfs/"

- Documentation des produits NetApp

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

- Qu'est-ce que NFS ?

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

- Qu'est-ce que la réaffectation de la partition Kafka ?

  "https://docs.cloudera.com/runtime/7.2.10/kafka-managing/topics/kafka-manage-cli-reassign-overview.html"

- Qu'est-ce que le banc d'essai OpenMessaging ?

  "https://openmessaging.cloud/"

- Comment migrer un courtier Kafka ?

  "https://medium.com/@sanchitbansal26/how-to-migrate-kafka-cluster-with-no-downtime-58c216129058"

- Comment surveiller le courtier Kafka avec Prometheus ?

  https://www.confluent.io/blog/monitor-kafka-clusters-with-prometheus-grafana-and-confluent/

- Plateforme gérée pour Apache Kafka

  https://www.instaclustr.com/platform/managed-apache-kafka/

- Prise en charge d'Apache Kafka

  https://www.instaclustr.com/support-solutions/kafka-support/

- Services de conseil pour Apache Kafka

  https://www.instaclustr.com/services/consulting/

**Informations sur le copyright**

Copyright © 2024 NetApp, Inc. Tous droits réservés. Imprimé aux États-Unis. Aucune partie de ce document protégé par copyright ne peut être reproduite sous quelque forme que ce soit ou selon quelque méthode que ce soit (graphique, électronique ou mécanique, notamment par photocopie, enregistrement ou stockage dans un système de récupération électronique) sans l'autorisation écrite préalable du détenteur du droit de copyright.

Les logiciels dérivés des éléments NetApp protégés par copyright sont soumis à la licence et à l'avis de non-responsabilité suivants :

CE LOGICIEL EST FOURNI PAR NETAPP « EN L'ÉTAT » ET SANS GARANTIES EXPRESSES OU TACITES, Y COMPRIS LES GARANTIES TACITES DE QUALITÉ MARCHANDE ET D'ADÉQUATION À UN USAGE PARTICULIER, QUI SONT EXCLUES PAR LES PRÉSENTES. EN AUCUN CAS NETAPP NE SERA TENU POUR RESPONSABLE DE DOMMAGES DIRECTS, INDIRECTS, ACCESSOIRES, PARTICULIERS OU EXEMPLAIRES (Y COMPRIS L'ACHAT DE BIENS ET DE SERVICES DE SUBSTITUTION, LA PERTE DE JOUISSANCE, DE DONNÉES OU DE PROFITS, OU L'INTERRUPTION D'ACTIVITÉ), QUELLES QU'EN SOIENT LA CAUSE ET LA DOCTRINE DE RESPONSABILITÉ, QU'IL S'AGISSE DE RESPONSABILITÉ CONTRACTUELLE, STRICTE OU DÉLICTUELLE (Y COMPRIS LA NÉGLIGENCE OU AUTRE) DÉCOULANT DE L'UTILISATION DE CE LOGICIEL, MÊME SI LA SOCIÉTÉ A ÉTÉ INFORMÉE DE LA POSSIBILITÉ DE TELS DOMMAGES.

NetApp se réserve le droit de modifier les produits décrits dans le présent document à tout moment et sans préavis. NetApp décline toute responsabilité découlant de l'utilisation des produits décrits dans le présent document, sauf accord explicite écrit de NetApp. L'utilisation ou l'achat de ce produit ne concède pas de licence dans le cadre de droits de brevet, de droits de marque commerciale ou de tout autre droit de propriété intellectuelle de NetApp.

Le produit décrit dans ce manuel peut être protégé par un ou plusieurs brevets américains, étrangers ou par une demande en attente.

LÉGENDE DE RESTRICTION DES DROITS : L'utilisation, la duplication ou la divulgation par le gouvernement sont sujettes aux restrictions énoncées dans le sous-paragraphe (b)(3) de la clause Rights in Technical Data-Noncommercial Items du DFARS 252.227-7013 (février 2014) et du FAR 52.227-19 (décembre 2007).

Les données contenues dans les présentes se rapportent à un produit et/ou service commercial (tel que défini par la clause FAR 2.101). Il s'agit de données propriétaires de NetApp, Inc. Toutes les données techniques et tous les logiciels fournis par NetApp en vertu du présent Accord sont à caractère commercial et ont été exclusivement développés à l'aide de fonds privés. Le gouvernement des États-Unis dispose d'une licence limitée irrévocable, non exclusive, non cessible, non transférable et mondiale. Cette licence lui permet d'utiliser uniquement les données relatives au contrat du gouvernement des États-Unis d'après lequel les données lui ont été fournies ou celles qui sont nécessaires à son exécution. Sauf dispositions contraires énoncées dans les présentes, l'utilisation, la divulgation, la reproduction, la modification, l'exécution, l'affichage des données sont interdits sans avoir obtenu le consentement écrit préalable de NetApp, Inc. Les droits de licences du Département de la Défense du gouvernement des États-Unis se limitent aux droits identifiés par la clause 252.227-7015(b) du DFARS (février 2014).

**Informations sur les marques commerciales**

NETAPP, le logo NETAPP et les marques citées sur le site http://www.netapp.com/TM sont des marques déposées ou des marques commerciales de NetApp, Inc. Les autres noms de marques et de produits sont des marques commerciales de leurs propriétaires respectifs.