



Guide du développeur

Amazon Keyspaces (pour Apache Cassandra)



Amazon Keyspaces (pour Apache Cassandra): Guide du développeur

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques et la présentation commerciale d'Amazon ne peuvent être utilisées en relation avec un produit ou un service qui n'est pas d'Amazon, d'une manière susceptible de créer une confusion parmi les clients, ou d'une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d'Amazon appartiennent à leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Qu'est-ce qu'Amazon Keyspaces ?	1
Comment ça marche	2
High-level architecture	2
Modèle de données Cassandra	5
Accès à Amazon Keyspaces	6
Cas d'utilisation	7
Qu'est-ce que le CQL ?	8
Comparez Amazon Keyspaces avec Cassandra	10
Différences fonctionnelles avec Apache Cassandra	11
API, opérations et types de données Apache Cassandra	12
Création et suppression asynchrones d'espaces clés et de tables	12
Authentification et autorisation	12
Par lots	12
Capture des données de modification (CDC)	13
Configuration du cluster	13
Connexions	13
INmot clé	14
FROZENcollections	15
Transactions légères	15
Équilibrage de charge	15
Pagination	16
Cloisons	16
Déclarations préparées	16
Supprimer une plage	17
Tables système	17
Horodatages	18
User-defined types (UDT)	18
API, opérations, fonctions et types de données Cassandra pris en charge	18
Support de l'API Cassandra	19
Support de l'API du plan de contrôle Cassandra	20
Support de l'API du plan de données Cassandra	21
Support de la fonction Cassandra	21
Support des types de données Cassandra	22
Niveaux de cohérence Cassandra pris en charge	24

Niveaux de cohérence en écriture	24
Lire les niveaux de cohérence	25
Niveaux de cohérence non pris en charge	26
Migrer vers Amazon Keyspaces	27
Migration depuis Cassandra	28
Compatibilité	29
Estimer les prix	30
Stratégie de migration	39
Migration en ligne	40
Migration hors ligne	53
Migration hybride	56
Outils de migration	60
Chargement de données à l'aide de cqlsh	62
Chargement de données à l'aide de DSBulk	75
Accès à Amazon Keyspaces	94
Inscrivez-vous pour un Compte AWS	94
Configuration d'Amazon Keyspaces	94
Utilisation de la console	95
CloudShell intégration à la console	95
En utilisant AWS CloudShell	97
AWS CloudShell options d'intégration	98
Conditions préalables	98
Commencer avec AWS CloudShell	98
Obtention des autorisations IAM pour AWS CloudShell	99
Installation du client cqlsh	100
Configuration des certificats TLS	100
Connexion à Amazon Keyspaces	100
Utilisation de CQLSH avec Amazon Keyspaces	101
Données persistantes dans AWS CloudShell	101
Bonnes pratiques	101
Résolution des problèmes	101
Exemple : Comment interagir avec Amazon Keyspaces en utilisant AWS CloudShell	102
Création d'identifiants d'accès programmatiques	103
Créez des informations d'identification spécifiques au service	105
Création d'informations d'identification IAM pour l' AWS authentication	107
Points de terminaison de service	116

Ports et protocoles	116
Points de terminaison globaux	117
AWS GovCloud (US) Region Points de terminaison FIPS	123
Points de terminaison pour les régions de Chine	124
Points de terminaison des diffusions	124
Connexion à des points de terminaison à double pile	129
Prise en charge de l'IPv6 dans Amazon Keyspaces	130
Résolution DNS dans Amazon Keyspaces	130
Utilisation de <code>cqlsh</code>	131
Utilisation de <code>cqlsh-expansion</code>	132
Comment configurer manuellement les <code>cqlsh</code> connexions pour le protocole TLS	137
À l'aide du AWS CLI	139
Téléchargement et configuration du AWS CLI	139
Utilisation du AWS CLI avec Amazon Keyspaces	139
Utilisation de l'API	144
Utilisation d'un pilote client Cassandra	144
Utilisation d'un pilote client Java Cassandra	145
Utilisation d'un pilote client Cassandra Python	163
Utilisation d'un pilote Node.js client Cassandra	167
Utilisation d'un pilote client Cassandra .NET Core	172
Utilisation d'un pilote client Cassandra Go	174
Utilisation d'un pilote client Cassandra Perl	180
Configuration de l'accès entre comptes	181
Configuration de l'accès entre comptes dans un VPC partagé	182
Configuration de l'accès entre comptes sans VPC partagé	186
Prise en main	191
Conditions préalables	192
Création d'un espace clé	193
Vérifier l'état de création de l'espace clé	196
Création d'une table	197
Vérifier l'état de création de la table	205
Opérations CRUD	206
Créer	207
Lecture	210
Mettre à jour	215
Suppression	216

Supprimer une table	218
Supprimer un espace clé	221
Tutoriels et solutions	224
Connexion aux points de terminaison VPC	224
Conditions préalables	225
Étape 1 : lancer une instance Amazon EC2	225
Étape 2 : configurer votre instance Amazon EC2	227
Étape 3 : créer un point de terminaison VPC pour Amazon Keyspaces	230
Étape 4 : configurer les autorisations pour la connexion du point de terminaison VPC	236
Étape 5 : Configuration de la surveillance	240
Étape 6 : (Facultatif) Meilleures pratiques pour les connexions	241
Étape 7 : (Facultatif) Nettoyer	244
Intégration à Apache Spark	245
Conditions préalables	246
Étape 1 : Configuration d'Amazon Keyspaces	247
Étape 2 : Configuration du connecteur Apache Cassandra Spark	248
Étape 3 : Création du fichier de configuration de l'application	250
Étape 4 : Préparation des données source et de la table cible	253
Étape 5 : Écrire et lire les données Amazon Keyspaces	255
Résolution des problèmes	258
Connexion depuis Amazon EKS	260
Conditions préalables	261
Étape 1 : Configuration du cluster Amazon EKS	263
Étape 2 : Configuration de l'application	268
Étape 3 : Création de l'image de l'application	271
Étape 4 : déploiement de l'application sur Amazon EKS	273
Étape 5 : Nettoyage (facultatif)	278
Exportation de données vers Amazon S3	281
Conditions préalables	281
Étape 1 : infrastructure et AWS Glue emplois Bootstrap	283
Étape 2 : Exécuter la tâche d'exportation	286
Étape 3 : (Facultatif) Planifiez la tâche d'exportation	288
Étape 4 : Nettoyage (facultatif)	290
Gestion des ressources sans serveur	293
Estimer la taille des lignes	294
Estimer la taille codée des colonnes	295

Estimer la taille codée des valeurs de données en fonction du type de données	296
Tenez compte de l'impact des fonctionnalités d'Amazon Keyspaces sur la taille des lignes ..	298
Choisissez la bonne formule pour calculer la taille codée d'une ligne	299
Exemple de calcul de la taille des lignes	299
Estimation de la consommation de capacité	301
Estimer la consommation de capacité des requêtes de plage	302
Estimer la consommation de capacité de lecture des requêtes limitées	303
Estimez la consommation de capacité de lecture des scans de tables	304
Estimation de la consommation de capacité du LWT	305
Estimation de la consommation de capacité des colonnes statiques	306
Estimation de la capacité d'une table multirégionale	310
Estimez la consommation de capacité avec CloudWatch	312
Configuration des modes read/write de capacité	312
Configuration du mode capacité à la demande	313
Configuration du mode de capacité provisionnée	315
Afficher le mode de capacité d'une table	317
Changer le mode de capacité	319
Configurer le préchauffage	322
Gérez la capacité de débit avec la mise à l'échelle automatique	337
Comment fonctionne la mise à l'échelle automatique d'Amazon Keyspaces	337
Fonctionnement de la mise à l'échelle automatique pour les tables multirégions	339
Notes d'utilisation	340
Configuration et mise à jour des politiques de dimensionnement automatique	341
Utiliser la capacité de rafale	356
Utilisation des fonctionnalités d'Amazon Keyspaces	357
Keyspaces du système	358
system	359
system_schema	361
system_schema_mcs	362
system_multiregion_info	365
User-defined types (UDT)	368
Configurer les autorisations	369
Création d'un UDT	372
Afficher les UDT	376
Supprimer un UDT	380
Utilisation des requêtes CQL	381

Utiliser IN SELECT	381
Utiliser des relevés par lots	386
Résultats de la commande	393
Paginer les résultats	394
Travailler avec les flux CDC	395
Comment fonctionnent les flux CDC	396
Utiliser les flux CDC	401
Utilisation de partitionneurs	437
Changer le partitionneur	438
Client-side horodatages	439
Intégration avec les AWS services	441
Création d'un tableau avec des horodatages côté client	441
Configuration des horodatages côté client	445
Utiliser les horodatages côté client dans les requêtes	448
Multi-Region réplication	449
Avantages	449
Modes de capacité et tarification	450
Comment ça marche	451
Notes d'utilisation	456
Configuration de la réplication multi-régions	458
Backup et restauration avec restauration instantanée	488
Comment ça marche	489
Utilisez la restauration instantanée	494
Expirer les données avec Time to Live	508
Intégration avec les AWS services	509
Créer une table avec une valeur TTL par défaut	510
Mettre à jour la valeur TTL par défaut de la table	514
Créer un tableau avec un TTL personnalisé	517
Mettre à jour le TTL personnalisé du tableau	519
INSERTÀ utiliser pour définir un TTL personnalisé pour les nouvelles lignes	522
UPDATEÀ utiliser pour définir un TTL personnalisé pour les lignes et les colonnes	523
Utilisation des AWS SDK	524
Utilisation des tags	525
Restrictions de balisage	526
Étiquetez les espaces clés, les tables et les flux	527
Création de rapports de répartition des coûts	546

Créez des CloudFormation ressources	547
Amazon Keyspaces et modèles CloudFormation	547
En savoir plus sur CloudFormation	548
NoSQL Workbench	548
Download	549
Prise en main	549
Visualisez un modèle de données	551
Création d'un modèle de données	555
Modifier un modèle de données	558
Commencer un modèle de données	560
Exemples de modèle de données	571
Historique de versions	572
Exemples de code	574
Principes de base	575
Bonjour Amazon Keyspaces	575
Principes de base	580
Actions	643
Bibliothèques et outils	697
Bibliothèques et exemples	697
Boîte à outils pour développeurs Amazon Keyspaces (pour Apache Cassandra)	697
Exemples d'Amazon Keyspaces (pour Apache Cassandra)	697
AWS Plug-ins d'authentification Signature Version 4 (SigV4)	698
Exemples et dépôts d'outils de développement mis en évidence	698
Tampons de protocole Amazon Keyspaces	698
CloudFormation modèle pour créer un CloudWatch tableau de bord Amazon pour les	
métriques Amazon Keyspaces (pour Apache Cassandra)	698
Utilisation d'Amazon Keyspaces (pour Apache Cassandra) avec AWS Lambda	699
Utilisation d'Amazon Keyspaces (pour Apache Cassandra) avec Spring	699
Utilisation d'Amazon Keyspaces (pour Apache Cassandra) avec Scala	699
Utilisation d'Amazon Keyspaces (pour Apache Cassandra) avec AWS Glue	699
Amazon Keyspaces (pour Apache Cassandra) Langage de requête Cassandra (CQL) vers	
convertisseur CloudFormation	700
Amazon Keyspaces (pour Apache Cassandra), aides pour le pilote Apache Cassandra pour	
Java	700
Démon de compression rapide d'Amazon Keyspaces (pour Apache Cassandra)	700
Démon d'Amazon Keyspaces (pour Apache Cassandra) et du codec Amazon S3	700

Bonnes pratiques	701
Conception NoSQL	702
NoSQL comparé au SGBDR	703
Deux concepts essentiels	703
Approche générale	704
Connexions	705
Comment ils fonctionnent	706
Comment configurer les connexions	707
Comment configurer les politiques de nouvelle tentative	709
Connexions aux points de terminaison VPC	709
Comment surveiller les connexions	711
Comment gérer les erreurs de connexion	712
Modélisation des données	712
Création de clés de partition	713
Optimisation des coûts	716
Évaluer les coûts au niveau de la table	716
Évaluez le mode capacité de votre table	718
Évaluez les paramètres Application Auto Scaling de votre table	723
Identification de vos ressources inutilisées	731
Évaluer les modèles d'utilisation d'une table	737
Évaluer la capacité allouée pour un dimensionnement approprié	738
Résolution des problèmes	748
Erreurs générales	749
Erreurs générales	749
Erreurs de connexion	751
Erreurs de connexion à un point de terminaison Amazon Keyspaces	751
Erreurs liées à la gestion des capacités	763
Erreurs de capacité sans serveur	764
Erreurs de langage de définition des données	769
Erreurs de langage de définition des données	769
Surveillance d'Amazon Keyspaces	775
Surveillance avec CloudWatch	776
Utilisation de métriques	777
Dimensions et métriques	779
Affichage des métriques	802
Création d'alarmes	803

Se connecter avec CloudTrail	804
Configuration des entrées du fichier journal dans CloudTrail	804
Informations DDL dans CloudTrail	806
Informations DML dans CloudTrail	807
Présentation des entrées des fichiers journaux	808
Sécurité	819
Protection des données	820
Chiffrement au repos	821
Chiffrement en transit	843
Confidentialité du trafic inter-réseau	843
Gestion des identités et des accès AWS	844
Public ciblé	845
Authentification par des identités	845
Gestion de l'accès à l'aide de politiques	847
Comment Amazon Keyspaces fonctionne avec IAM	849
Identity-based exemples de politiques	854
AWS politiques gérées	863
Résolution des problèmes	875
Utilisation des rôles liés à un service	879
Validation de conformité	889
Résilience	889
Sécurité de l'infrastructure	890
Utilisation des points de terminaison de VPC d'interface	892
Utilisation de points de terminaison VPC d'interface avec des flux	899
Analyse de configuration et de vulnérabilité pour Amazon Keyspaces	903
Bonnes pratiques de sécurité	903
Bonnes pratiques de sécurité préventive	903
Bonnes pratiques de sécurité de détection	905
Référence du langage CQL	908
Éléments de langage	909
Identifiants	909
Constantes	909
Termes	910
Types de données	910
Encodage JSON des types de données Amazon Keyspaces	914
Instructions DDL	918

Keyspaces	919
Tables	923
Types	937
Instructions DML	940
SELECT	940
INSERT	943
UPDATE	945
DELETE	946
Built-in fonctions	947
Fonctions scalaires	947
Quotas	950
Quotas de service Amazon Keyspaces	950
Augmentation ou diminution du débit (pour les tables allouées)	957
Augmentation du débit alloué	957
Diminution du débit alloué	957
Le chiffrement d'Amazon Keyspaces est au repos	958
Quotas et valeurs par défaut pour les types définis par l'utilisateur (UDT) dans Amazon Keyspaces	958
Quotas UDT et valeurs par défaut d'Amazon Keyspaces	958
Historique de la documentation	960
.....	cmlxxvi

Qu'est-ce qu'Amazon Keyspaces (pour Apache Cassandra) ?

Amazon Keyspaces (pour Apache Cassandra) est un service de base de données évolutif, hautement disponible et géré compatible avec Apache Cassandra. Avec Amazon Keyspaces, vous n'avez pas besoin de provisionner, de corriger ou de gérer des serveurs, et vous n'avez pas besoin d'installer, de maintenir ou d'exploiter des logiciels.

Amazon Keyspaces fonctionne sans serveur. Vous ne payez donc que pour les ressources que vous utilisez, et le service adapte automatiquement les tableaux à la hausse et à la baisse en fonction du trafic des applications. Vous pouvez créer des applications au service de milliers de demandes par seconde avec un débit et un stockage pratiquement illimités.

Note

Apache Cassandra est un magasin de données open source, à colonne large, conçue pour gérer de grandes quantités de données. Pour de plus amples informations, veuillez consulter [Apache Cassandra](#).

Amazon Keyspaces facilite la migration, l'exécution et la mise à l'échelle des charges de travail Cassandra dans le. AWS Cloud En quelques clics sur la console de AWS gestion ou en quelques lignes de code, vous pouvez créer des espaces clés et des tables dans Amazon Keyspaces, sans déployer d'infrastructure ni installer de logiciel.

Avec Amazon Keyspaces, vous pouvez exécuter vos charges de travail Cassandra existantes AWS en utilisant le même code d'application Cassandra et les mêmes outils de développement que ceux que vous utilisez aujourd'hui.

Grâce au calculateur de [prix pour Amazon Keyspaces \(pour Apache Cassandra\)](#) disponible sur Github, vous pouvez estimer vos coûts mensuels pour Amazon Keyspaces en fonction de votre charge de travail Apache Cassandra existante. Entrez les mesures issues de la sortie d'état de votre outil de nœud Cassandra et de la configuration sans serveur prévue pour Amazon Keyspaces afin de comparer les coûts directs entre les deux solutions. Notez que ce calculateur se concentre uniquement sur les coûts opérationnels d'Amazon Keyspaces par rapport à votre déploiement Cassandra existant. Il n'inclut pas les facteurs liés au coût total de possession (TCO) tels que la maintenance de l'infrastructure, les frais d'exploitation ou les coûts de support pour Cassandra.

Pour obtenir la liste des points de terminaison disponibles Régions AWS , consultez la section Points de terminaison [de service pour Amazon Keyspaces](#).

Nous vous recommandons de commencer par lire les sections suivantes :

Rubriques

- [Amazon Keyspaces : comment ça marche](#)
- [Cas d'utilisation d'Amazon Keyspaces](#)
- [Qu'est-ce que le langage de requête Cassandra \(CQL\) ?](#)

Amazon Keyspaces : comment ça marche

Amazon Keyspaces supprime les frais administratifs liés à la gestion de Cassandra. Pour comprendre pourquoi, il est utile de commencer par l'architecture de Cassandra, puis de la comparer à Amazon Keyspaces.

Rubriques

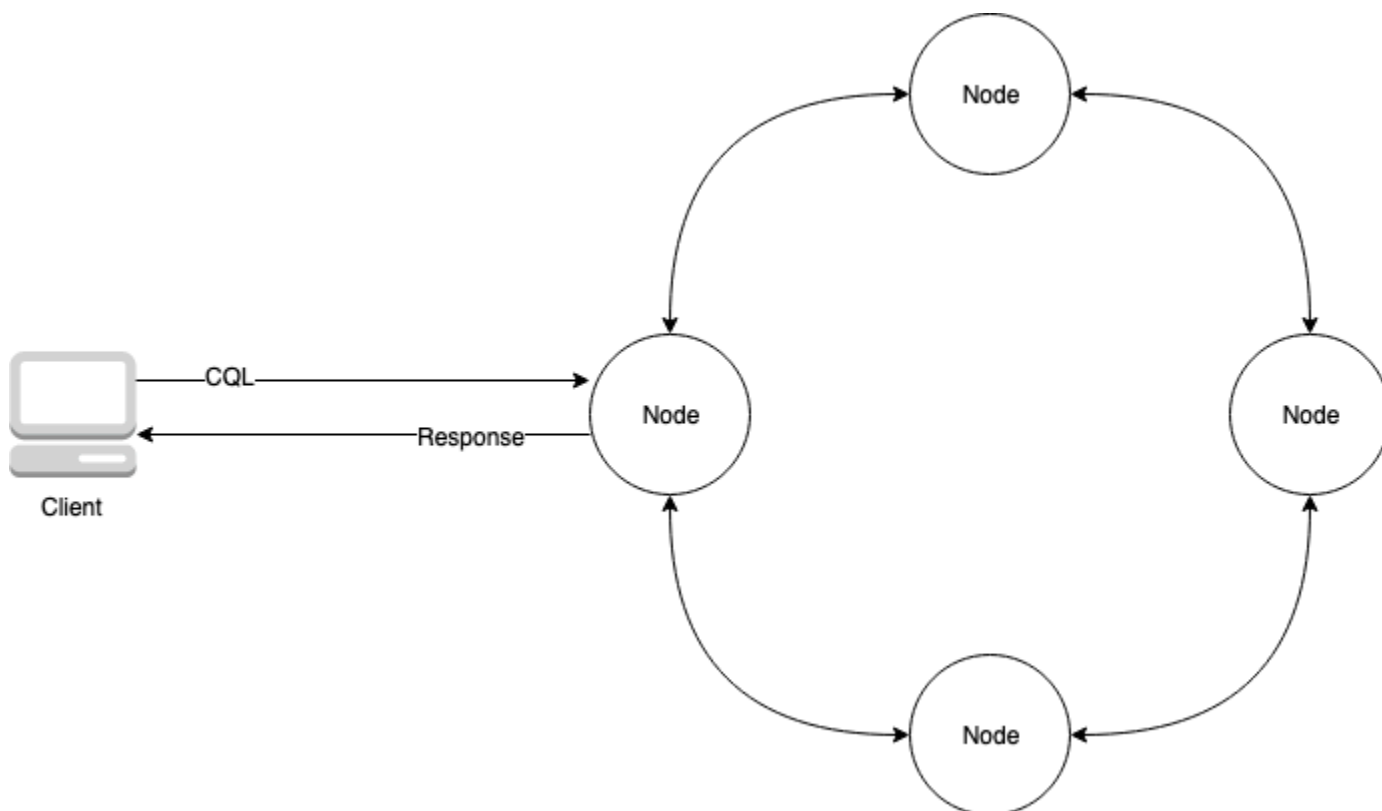
- [High-level architecture : Apache Cassandra contre Amazon Keyspaces](#)
- [Modèle de données Cassandra](#)
- [Accès à Amazon Keyspaces depuis une application](#)

High-level architecture : Apache Cassandra contre Amazon Keyspaces

Apache Cassandra traditionnelle est déployée dans un cluster composé d'un ou plusieurs nœuds. Vous êtes responsable de la gestion de chaque nœud et de l'ajout et de la suppression de nœuds à mesure que votre cluster évolue.

Un programme client accède à Cassandra en se connectant à l'un des nœuds et en émettant des instructions Cassandra Query Language (CQL). CQL est similaire à SQL, le langage populaire utilisé dans les bases de données relationnelles. Même si Cassandra n'est pas une base de données relationnelle, CQL fournit une interface familière pour interroger et manipuler des données dans Cassandra.

Le diagramme suivant montre un cluster Apache Cassandra simple, composé de quatre nœuds.



Un déploiement Cassandra de production peut être composé de centaines de nœuds, s'exécutant sur des centaines d'ordinateurs physiques dans un ou plusieurs centres de données physiques. Cela peut entraîner un fardeau opérationnel pour les développeurs d'applications qui doivent allouer, corriger et gérer des serveurs en plus de l'installation, de la maintenance et de l'exploitation des logiciels.

Avec Amazon Keyspaces (pour Apache Cassandra), vous n'avez pas besoin de provisionner, de patcher ou de gérer des serveurs. Vous pouvez donc vous concentrer sur la création de meilleures applications. Amazon Keyspaces propose deux modes de capacité de débit pour les lectures et les écritures : à la demande et provisionné. Vous pouvez choisir le mode de capacité de débit de votre table pour optimiser le prix des lectures et des écritures en fonction de la prévisibilité et de la variabilité de votre charge de travail.

Avec le mode à la demande, vous ne payez que les lectures et les écritures que votre application effectue. Il n'est pas nécessaire de spécifier la capacité de débit de votre table à l'avance. Amazon Keyspaces gère le trafic de vos applications presque instantanément, à mesure qu'il augmente ou diminue, ce qui en fait une bonne option pour les applications dont le trafic est imprévisible.

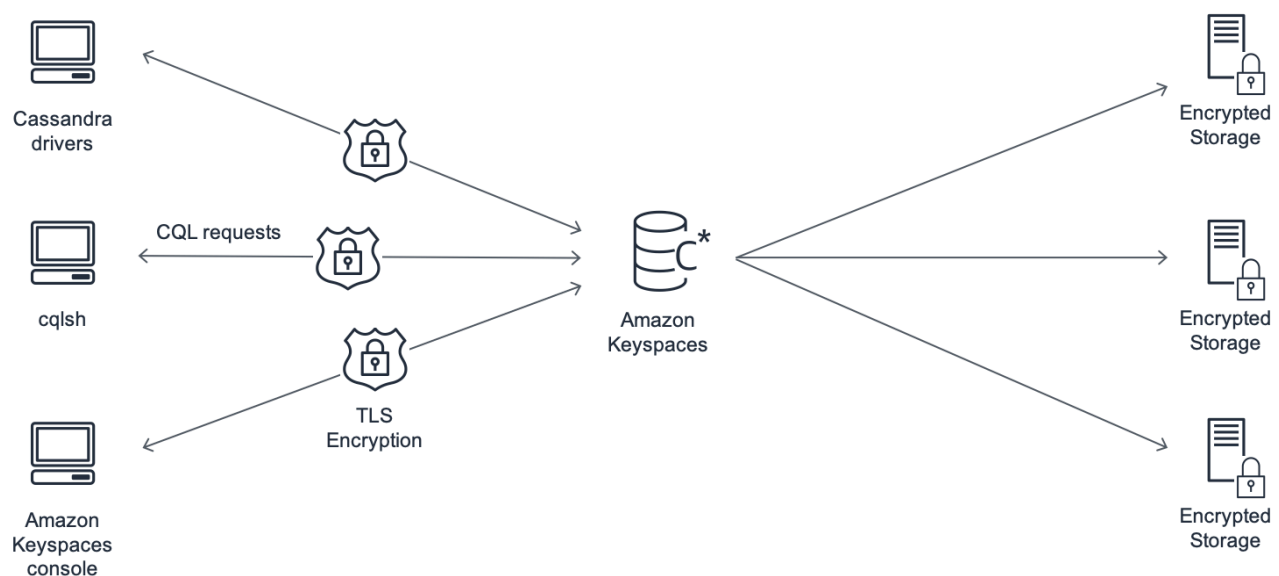
Le mode de capacité allouée permet d'optimiser le débit si vous disposez d'un trafic d'application prévisible et pouvez prévoir les besoins en capacité de votre table. Avec le mode de capacité

allouée, vous spécifiez le nombre de lectures et d'écritures par seconde dont vous pensez que votre application aura besoin. Vous pouvez augmenter et diminuer automatiquement la capacité allouée de votre table en activant la mise à [l'échelle automatique](#).

Vous pouvez modifier le mode de capacité de votre table une fois par jour à mesure que vous en apprendrez plus sur les schémas de trafic de votre charge de travail, ou si vous prévoyez avoir une forte explosion de trafic, par exemple en raison d'un événement majeur qui, selon vous, entraînera beaucoup de trafic de table. Pour de plus amples informations sur l'allocation de capacité en lecture et en écriture, reportez-vous à la section [the section called “Configuration des modes read/write de capacité”](#).

Amazon Keyspaces (pour Apache Cassandra) stocke trois copies de vos données dans plusieurs zones de [disponibilité](#) pour des raisons de durabilité et de haute disponibilité. En outre, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des organisations les plus pointilleuses en termes de sécurité. Le chiffrement au repos est automatiquement activé lorsque vous créez une nouvelle table Amazon Keyspaces et que toutes les connexions client nécessitent le protocole TLS (Transport Layer Security). Les fonctionnalités AWS de sécurité supplémentaires incluent [la surveillance](#) et [Gestion des identités et des accès AWS](#) les points de terminaison [du cloud privé](#) virtuel (VPC). Pour obtenir une vue d'ensemble de toutes les fonctionnalités de sécurité disponibles, reportez-vous à la section [Sécurité](#).

Le schéma suivant montre l'architecture d'Amazon Keyspaces.



Un programme client accède à Amazon Keyspaces en se connectant à un point de terminaison prédéterminé (nom d'hôte et numéro de port) et en émettant des instructions CQL. Pour obtenir la liste des points de terminaison disponibles, reportez-vous à la section [the section called “Points de terminaison de service”](#).

Modèle de données Cassandra

La manière dont vous modélisez vos données pour votre analyse de rentabilisation est essentielle pour optimiser les performances d'Amazon Keyspaces. Un modèle de données de médiocre qualité peut considérablement dégrader les performances.

Même si CQL ressemble à SQL, les backends de Cassandra et des bases de données relationnelles sont très différents et doivent être abordés différemment. Voici quelques-unes des questions les plus importantes à considérer :

Stockage

Vous pouvez visualiser vos données Cassandra dans des tables, chaque ligne représentant un enregistrement et chaque colonne un champ dans cet enregistrement.

Conception de la table : recherchez d'abord

Il n'y a pas d'instruction JOIN dans CQL. Par conséquent, vous devez concevoir vos tables en fonction de la forme de vos données et de la manière dont vous devez y accéder pour vos cas d'utilisation professionnels. Cela peut entraîner une dénormalisation avec des données dupliquées. Vous devez concevoir chacune de vos tables spécifiquement pour un modèle d'accès particulier.

Partitions

Vos données sont stockées dans des partitions sur le disque. Le nombre de partitions dans lesquelles vos données sont stockées et la façon dont elles sont distribuées entre les partitions sont déterminées par votre clé de partition. La façon dont vous définissez votre clé de partition peut avoir un impact significatif sur les performances de vos requêtes. Pour connaître les bonnes pratiques, consultez [the section called “Création de clés de partition”](#).

Clé primaire

Dans Cassandra, les données sont stockées sous la forme d'une paire clé-valeur. Chaque table Cassandra doit avoir une clé primaire, qui est la clé unique de chaque ligne de la table. La clé primaire est le composite d'une clé de partition requise et de colonnes de clustering facultatives.

Les données qui composent la clé primaire doivent être uniques pour tous les enregistrements d'une table.

- **Clé de partition** : la partie clé de partition de la clé primaire est requise et détermine dans quelle partition de votre cluster les données sont stockées. La clé de partition peut être une seule colonne ; il peut aussi s'agir d'une valeur composée de deux colonnes ou plus. Vous utiliseriez une clé de partition composée si une clé de partition à colonne unique devait aboutir à une seule partition ou à un très petit nombre de partitions contenant la plupart des données et supportant ainsi la majorité des I/O opérations sur le disque.
- **Colonne de clustering** : la partie optionnelle de la colonne de clustering de votre clé primaire détermine la manière dont les données sont regroupées et triées au sein de chaque partition. Si vous incluez une colonne de clustering dans votre clé primaire, la colonne de clustering peut comporter une ou plusieurs colonnes. S'il y a plusieurs colonnes dans la colonne de regroupement, l'ordre de tri est déterminé par l'ordre dans lequel les colonnes sont répertoriées dans la colonne de regroupement, de gauche à droite.

Pour plus d'informations sur la conception NoSQL et Amazon Keyspaces, consultez [the section called “Conception NoSQL”](#). Pour plus d'informations sur Amazon Keyspaces et la modélisation des données, consultez [the section called “Modélisation des données”](#).

Accès à Amazon Keyspaces depuis une application

Amazon Keyspaces (pour Apache Cassandra) implémente l'API Apache Cassandra Query Language (CQL), afin que vous puissiez utiliser les pilotes CQL et Cassandra que vous utilisez déjà. Mettre à jour votre application est aussi simple que de mettre à jour votre pilote ou votre `cqlsh` configuration Cassandra pour qu'il pointe vers le point de terminaison du service Amazon Keyspaces. Pour plus d'informations sur les informations d'identification requises, consultez [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#).

Note

Pour vous aider à démarrer, vous pouvez trouver des exemples de code de bout en bout relatifs à la connexion à Amazon Keyspaces à l'aide de différents pilotes clients Cassandra dans le référentiel d'exemples de code Amazon Keyspaces sur [GitHub](#)

Considérez le programme Python suivant, qui se connecte à un cluster Cassandra et interroge une table.

```
from cassandra.cluster import Cluster
#TLS/SSL configuration goes here

ksp = 'MyKeyspace'
tbl = 'WeatherData'

cluster = Cluster(['NNN.NNN.NNN.NNN'], port=NNNN)
session = cluster.connect(ksp)

session.execute('USE ' + ksp)

rows = session.execute('SELECT * FROM ' + tbl)
for row in rows:
    print(row)
```

Pour exécuter le même programme sur Amazon Keyspaces, vous devez :

- Ajouter le point de terminaison et le port du cluster : Par exemple, l'hôte peut être remplacé par un point de terminaison de service, tel que `cassandra.us-east-1.amazonaws.com`, et le numéro de port par : 9142.
- Ajouter la TLS/SSL configuration : pour plus d'informations sur l'ajout de la TLS/SSL configuration permettant de se connecter à Amazon Keyspaces à l'aide d'un pilote Python client Cassandra, consultez. [Utilisation d'un pilote client Cassandra Python pour accéder à Amazon Keyspaces par programmation](#)

Cas d'utilisation d'Amazon Keyspaces

Voici quelques-unes des manières dont vous pouvez utiliser Amazon Keyspaces :

- Créez des applications nécessitant une faible latence : traitez les données à haute vitesse pour les applications nécessitant une latence d'un chiffre en millisecondes, telles que la maintenance des équipements industriels, la surveillance commerciale, la gestion de flotte et l'optimisation des itinéraires.
- Créez des applications à l'aide de technologies open source : créez des applications à AWS l'aide d'API et de pilotes Cassandra open source disponibles pour un large éventail de langages de programmation, tels que Java, Python, Ruby, Microsoft .NET, PHP Node.js, C++, Perl et Go. Pour des exemples de code, consultez [Bibliothèques et outils](#).

- Transférez vos charges de travail Cassandra vers le cloud — Gérer vous-même les tables Cassandra prend du temps et coûte cher. Avec Amazon Keyspaces, vous pouvez configurer, sécuriser et faire évoluer les tables Cassandra AWS Cloud sans gérer l'infrastructure. Pour de plus amples informations, veuillez consulter [Gestion des ressources sans serveur](#).

Qu'est-ce que le langage de requête Cassandra (CQL) ?

Le Cassandra Query Language (CQL) est le langage principal pour communiquer avec Apache Cassandra. Amazon Keyspaces (pour Apache Cassandra) est compatible avec l'API CQL 3.x (rétrocompatible avec la version 2.x).

Dans CQL, les données sont stockées dans des tableaux, des colonnes et des lignes. En ce sens, le CQL est similaire au langage SQL (Structured Query Language). Ce sont les concepts clés de la CQL.

- Éléments CQL — Les éléments fondamentaux du CQL sont les identificateurs, les constantes, les termes et les types de données.
- Langage de définition des données (DDL) : les instructions DDL sont utilisées pour gérer des structures de données telles que les espaces clés et les tableaux, qui sont des AWS ressources dans Amazon Keyspaces. Les instructions DDL sont des opérations sur le plan de contrôle dans AWS.
- Langage de manipulation des données (DML) : les instructions DML sont utilisées pour gérer les données dans les tableaux. Les instructions DML sont utilisées pour sélectionner, insérer, mettre à jour et supprimer des données. Il s'agit d'opérations sur des plans de données dans AWS.
- Built-in fonctions : Amazon Keyspaces prend en charge diverses fonctions scalaires intégrées que vous pouvez utiliser dans les instructions CQL.

Pour de plus amples informations sur CQL, veuillez consulter [Référence du langage CQL pour Amazon Keyspaces \(pour Apache Cassandra\)](#). Pour les différences fonctionnelles avec Apache Cassandra, voir [the section called “Différences fonctionnelles avec Apache Cassandra”](#).

Pour exécuter les requêtes CQL, vous pouvez effectuer l'une des opérations suivantes :

- Utilisez l'éditeur CQL dans le Console de gestion AWS.
- Utilisation AWS CloudShell et extension [cq1sh](#).
- Utilisez un `cq1sh` client.

- Utilisez un pilote client Cassandra sous licence Apache 2.0.

Outre le CQL, vous pouvez effectuer des opérations DDL (Data Definition Language) dans Amazon Keyspaces à l'aide des kits de développement logiciel (AWS SDK) et du. AWS Command Line Interface

Pour plus d'informations sur l'utilisation de ces méthodes pour accéder à Amazon Keyspaces, consultez [Accès à Amazon Keyspaces \(pour Apache Cassandra\)](#).

Quelle est la différence entre Amazon Keyspaces (pour Apache Cassandra) et Apache Cassandra ?

[Pour établir une connexion à Amazon Keyspaces, vous pouvez utiliser un point de terminaison de AWS service public ou un point de terminaison privé à l'aide des points de terminaison Interface VPC \(AWS PrivateLink\) dans Amazon Virtual Private Cloud.](#) Selon le point de terminaison utilisé, Amazon Keyspaces peut apparaître au client de l'une des manières suivantes.

AWS connexion aux terminaux de service

Il s'agit d'une connexion établie sur n'importe quel point de [terminaison public](#). Dans ce cas, Amazon Keyspaces apparaît sous la forme d'un cluster Apache Cassandra 3.11.2 à neuf nœuds pour le client.

Interface : connexion au point de terminaison VPC

Il s'agit d'une connexion privée établie à l'aide d'un point de [terminaison VPC d'interface](#). Dans ce cas, Amazon Keyspaces apparaît sous la forme d'un cluster Apache Cassandra 3.11.2 à trois nœuds pour le client.

Indépendamment du type de connexion et du nombre de nœuds visibles par le client, Amazon Keyspaces fournit un débit et un stockage pratiquement illimités. Pour ce faire, Amazon Keyspaces mappe les nœuds aux équilibres de charge qui acheminent vos requêtes vers l'une des nombreuses partitions de stockage sous-jacentes. Pour plus d'informations sur les connexions, consultez [the section called "Comment ils fonctionnent"](#).

Amazon Keyspaces stocke les données dans des partitions. Une partition est une allocation de stockage pour une table, soutenue par des disques SSD (Solid State Drive). Amazon Keyspaces réplique automatiquement vos données sur plusieurs [zones de disponibilité](#) dans un souci de durabilité et de Région AWS haute disponibilité. À mesure que vos besoins en débit ou en stockage augmentent, Amazon Keyspaces gère la gestion des partitions pour vous et provisionne automatiquement les partitions supplémentaires requises.

Amazon Keyspaces prend en charge toutes les opérations courantes du plan de données Cassandra, telles que la création d'espaces de touches et de tables, la lecture de données et l'écriture de données. Amazon Keyspaces fonctionne [sans serveur](#), vous n'avez donc pas besoin de provisionner, de patcher ou de gérer des serveurs. Vous n'avez pas non plus besoin d'installer,

de maintenir ou d'utiliser un logiciel. Par conséquent, dans Amazon Keyspaces, vous n'avez pas besoin d'utiliser les opérations de l'API du plan de contrôle Cassandra pour gérer les paramètres des clusters et des nœuds.

Amazon Keyspaces configure automatiquement les paramètres tels que le facteur de réplication et le niveau de cohérence pour vous garantir une haute disponibilité, une durabilité et des performances à un chiffre en millisecondes. [Pour encore plus de résilience et des lectures locales à faible latence, Amazon Keyspaces propose une réplication multirégionale.](#)

Rubriques

- [Différences fonctionnelles : Amazon Keyspaces et Apache Cassandra](#)
- [API, opérations, fonctions et types de données Cassandra pris en charge](#)
- [Niveaux de cohérence en lecture et écriture pris en charge par Apache Cassandra et coûts associés](#)

Différences fonctionnelles : Amazon Keyspaces et Apache Cassandra

Voici les différences fonctionnelles entre Amazon Keyspaces et Apache Cassandra.

Rubriques

- [API, opérations et types de données Apache Cassandra](#)
- [Création et suppression asynchrones d'espaces clés et de tables](#)
- [Authentification et autorisation](#)
- [Par lots](#)
- [Capture des données de modification \(CDC\)](#)
- [Configuration du cluster](#)
- [Connexions](#)
- [INmot clé](#)
- [FROZENcollections](#)
- [Transactions légères](#)
- [Équilibrage de charge](#)
- [Pagination](#)

- [Cloisons](#)
- [Déclarations préparées](#)
- [Supprimer une plage](#)
- [Tables système](#)
- [Horodatages](#)
- [User-defined types \(UDT\)](#)

API, opérations et types de données Apache Cassandra

Amazon Keyspaces prend en charge toutes les opérations courantes du plan de données Cassandra, telles que la création d'espaces de touches et de tables, la lecture et l'écriture de données. Pour voir ce qui est actuellement pris en charge, veuillez consulter [API, opérations, fonctions et types de données Cassandra pris en charge](#).

Création et suppression asynchrones d'espaces clés et de tables

Amazon Keyspaces exécute des opérations de langage de définition de données (DDL), telles que la création et la suppression d'espaces clés, de tables et de types de manière asynchrone. Pour savoir comment surveiller l'état de création des ressources, consultez [the section called “Vérifier l'état de création de l'espace clé”](#) et [the section called “Vérifier l'état de création de la table”](#). Pour obtenir la liste des instructions DDL dans la référence du langage CQL, consultez. [the section called “Instructions DDL”](#)

Authentification et autorisation

Amazon Keyspaces (pour Apache Cassandra) utilise Gestion des identités et des accès AWS (IAM) pour l'authentification et l'autorisation des utilisateurs, et prend en charge les politiques d'autorisation équivalentes à celles d'Apache Cassandra. Amazon Keyspaces ne prend donc pas en charge les commandes de configuration de sécurité d'Apache Cassandra.

Par lots

Amazon Keyspaces prend en charge les commandes par lots suivantes :

- Lot enregistré : prend en charge jusqu'à 100 commandes par lot.
- Lot non enregistré : prend en charge jusqu'à 30 commandes par lot.

Seules INSERT UPDATE les DELETE commandes ou sont autorisées dans un lot.

Capture des données de modification (CDC)

Dans Apache Cassandra, la capture des données de modification (CDC) copie les segments du journal de validation qui capturent les mises à jour des tables dans un répertoire spécifié sur le nœud. Chaque enregistrement de modification dans le journal du CDC peut inclure des mises à jour sur plusieurs espaces clés et tables. Lorsqu'une ligne est mise à jour ou supprimée, le journal du CDC d'Apache Cassandra affiche uniquement les colonnes modifiées. Le journal n'inclut pas l'état antérieur des données avant la modification. Pour comprendre tous les détails des modifications des données, vous devez désérialiser ces journaux à l'aide d'outils spécifiques à Cassandra.

Avec les flux CDC d'Amazon Keyspaces, vous pouvez sélectionner le type d'informations que le flux CDC collecte pour chaque ligne. Par défaut, les flux CDC d'Amazon Keyspaces capturent la version de la ligne avant et après la modification. Amazon Keyspaces CDC fournit des enregistrements de modifications dédoublés et ordonnés au niveau de la table.

[Dans Amazon Keyspaces, chaque flux CDC est une AWS ressource associée à un Amazon Resource Name \(ARN\) et votre application peut consommer des flux Amazon Keyspaces CDC à l'aide de l'API Amazon Keyspaces CDC via l'un des points de terminaison disponibles.](#)

Pour plus d'informations sur Amazon Keyspaces CDC, consultez. [the section called “Travailler avec les flux CDC”](#)

Configuration du cluster

Amazon Keyspaces fonctionne sans serveur, il n'y a donc pas de clusters, d'hôtes ou de machines virtuelles Java (JVM) à configurer. Les paramètres de Cassandra relatifs au compactage, à la compression, à la mise en cache, à la collecte des déchets et au filtrage des floras ne sont pas applicables à Amazon Keyspaces et sont ignorés s'ils sont spécifiés.

Connexions

Vous pouvez utiliser les pilotes Cassandra existants pour communiquer avec Amazon Keyspaces, mais vous devez les configurer différemment. Amazon Keyspaces prend en charge jusqu'à 3 000 requêtes CQL par connexion TCP et par seconde, mais le nombre de connexions qu'un pilote peut établir est illimité.

La plupart des pilotes Cassandra open source établissent un pool de connexions avec Cassandra et équilibrent la charge des requêtes sur ce pool de connexions. Amazon Keyspaces expose 9

adresses IP homologues aux pilotes, et le comportement par défaut de la plupart des conducteurs est d'établir une connexion unique avec chaque adresse IP homologue. Par conséquent, le débit de requêtes CQL maximal d'un pilote utilisant les paramètres par défaut est de 27 000 requêtes CQL par seconde.

Pour augmenter ce nombre, nous vous recommandons d'augmenter le nombre de connexions par adresse IP que votre pilote maintient dans son groupe de connexions. Par exemple, si vous définissez le nombre maximum de connexions par adresse IP sur 2, le débit maximal de votre pilote est doublé pour atteindre 54 000 requêtes CQL par seconde.

Il est recommandé de configurer les pilotes pour qu'ils utilisent 500 requêtes CQL par seconde et par connexion afin de réduire les coûts et d'améliorer la distribution. Dans ce scénario, la planification de 18 000 requêtes CQL par seconde nécessite 36 connexions. La configuration du pilote pour 4 connexions sur 9 points de terminaison permet d'obtenir 36 connexions effectuant 500 demandes par seconde. Pour plus d'informations sur les meilleures pratiques en matière de connexions, consultez [the section called “Connexions”](#).

Lorsque vous vous connectez à des points de terminaison VPC, il se peut qu'il y ait moins de points de terminaison disponibles. Cela signifie que vous devez augmenter le nombre de connexions dans la configuration du pilote. Pour plus d'informations sur les meilleures pratiques relatives aux connexions VPC, consultez [the section called “Connexions aux points de terminaison VPC”](#)

INmot clé

Amazon Keyspaces prend en charge le IN mot clé dans la SELECT déclaration. IN n'est pas compatible avec UPDATE et DELETE. Lorsque vous utilisez le IN mot clé dans l'SELECT instruction, les résultats de la requête sont renvoyés dans l'ordre dans lequel les clés sont présentées dans l'SELECT instruction. Dans Cassandra, les résultats sont classés lexicographiquement.

Lors de l'utilisation ORDER BY, la réorganisation complète avec pagination désactivée n'est pas prise en charge et les résultats sont classés au sein d'une page. Les requêtes Slice ne sont pas prises en charge avec le IN mot clé. TOKENS ne sont pas pris en charge par le IN mot clé. Amazon Keyspaces traite les requêtes avec le IN mot clé en créant des sous-requêtes. Chaque sous-requête est considérée comme une connexion dans le cadre de la limite de 3 000 requêtes CQL par connexion TCP et par seconde. Pour de plus amples informations, veuillez consulter [the section called “Utiliser IN SELECT”](#).

FROZENcollections

Le FROZEN mot clé de Cassandra sérialise plusieurs composants d'un type de données de collection en une seule valeur immuable traitée comme un. BLOB INSERT et les UPDATE instructions remplacent l'ensemble de la collection.

Amazon Keyspaces prend en charge jusqu'à 8 niveaux d'imbrication pour les collections figées par défaut. Pour de plus amples informations, veuillez consulter [the section called “Quotas de service Amazon Keyspaces”](#).

Amazon Keyspaces ne prend pas en charge les comparaisons d'inégalités qui utilisent l'intégralité de la collection figée dans un conditionnel UPDATE ou SELECT une instruction. Le comportement des collections et des collections gelées est le même dans Amazon Keyspaces.

Lorsque vous utilisez des collections figées avec des horodatages côté client, dans le cas où l'horodatage d'une opération d'écriture est le même que celui d'une colonne existante qui n'est ni expirée ni gravée, Amazon Keyspaces n'effectue aucune comparaison. Au lieu de cela, il permet au serveur de déterminer le dernier rédacteur, et c'est le dernier rédacteur qui gagne.

Pour plus d'informations sur les collections congelées, consultez [the section called “Types de collections”](#).

Transactions légères

Amazon Keyspaces (pour Apache Cassandra) prend entièrement en charge les fonctionnalités de comparaison et de définition sur INSERT, et DELETE les commandes UPDATE, connues sous le nom de transactions légères (LWT) dans Apache Cassandra. En tant qu'offre sans serveur, Amazon Keyspaces (pour Apache Cassandra) fournit des performances constantes à toutes les échelles, y compris pour les transactions légères. Avec Amazon Keyspaces, l'utilisation de transactions légères n'entraîne aucune baisse de performance.

Équilibrage de charge

Les entrées du `system.peers` tableau correspondent aux équilibrateurs de charge Amazon Keyspaces. Pour de meilleurs résultats, nous vous recommandons d'utiliser une politique d'équilibrage de charge circulaire et d'ajuster le nombre de connexions par IP en fonction des besoins de votre application.

Pagination

Amazon Keyspaces pagine les résultats en fonction du nombre de lignes lues pour traiter une demande, et non du nombre de lignes renvoyées dans le jeu de résultats. Par conséquent, certaines pages peuvent contenir moins de lignes que ce que vous spécifiez dans `TAILLE DE PAGE` pour les requêtes filtrées. En outre, Amazon Keyspaces pagine automatiquement les résultats après avoir lu 1 Mo de données afin de fournir aux clients des performances de lecture cohérentes à un chiffre en millisecondes. Pour de plus amples informations, veuillez consulter [the section called “Pagner les résultats”](#).

Dans les tables comportant des colonnes statiques, Apache Cassandra et Amazon Keyspaces définissent la valeur de colonne statique de la partition au début de chaque page dans une requête multipage. Lorsqu'une table comporte de grandes lignes de données, en raison du comportement de pagination d'Amazon Keyspaces, il est plus probable que le résultat d'une opération de lecture par page renvoie plus de pages pour Amazon Keyspaces que pour Apache Cassandra. Par conséquent, il est plus probable dans Amazon Keyspaces que des mises à jour simultanées de la colonne statique entraînent une différence de valeur de colonne statique entre les différentes pages du jeu de résultats de lecture de la page.

Cloisons

Le partitionneur par défaut dans Amazon Keyspaces est le `Cassandra-compatible Murmur3Partitioner`. En outre, vous avez le choix d'utiliser soit les Amazon Keyspaces, `DefaultPartitioner` soit le `Cassandra-compatible RandomPartitioner`.

Avec Amazon Keyspaces, vous pouvez modifier le partitionneur de votre compte en toute sécurité sans avoir à recharger vos données Amazon Keyspaces. Une fois la modification de configuration terminée, qui prend environ 10 minutes, les clients verront le nouveau partitionneur se configurer automatiquement lors de leur prochaine connexion. Pour de plus amples informations, veuillez consulter [the section called “Utilisation de partitionneurs”](#).

Déclarations préparées

Amazon Keyspaces prend en charge l'utilisation d'instructions préparées pour les opérations en langage de manipulation de données (DML), telles que la lecture et l'écriture de données. Amazon Keyspaces ne prend actuellement pas en charge l'utilisation d'instructions préparées pour les opérations en langage de définition de données (DDL), telles que la création de tables et d'espaces de touches. Les opérations DDL doivent être exécutées en dehors des instructions préparées.

Supprimer une plage

Important

À compter du 1er juillet 2026, les opérations de suppression de plages passeront du traitement synchrone au traitement asynchrone. Après cette modification, une réponse positive confirme que votre demande de suppression a été acceptée, mais la suppression est peut-être toujours en cours. Pour vérifier que la suppression est terminée, recherchez des éléments dans la plage spécifiée. Lorsque la requête ne renvoie aucun résultat, la suppression est terminée.

Amazon Keyspaces prend en charge la suppression de lignes dans une plage. Une plage est un ensemble contigu de lignes au sein d'une partition. Vous spécifiez une plage dans une opération DELETE à l'aide d'une clause WHERE. Vous pouvez spécifier que la plage correspond à une partition complète.

En outre, vous pouvez définir une plage comme un sous-ensemble de lignes contiguës au sein d'une partition en utilisant des opérateurs relationnels (par exemple, « > », « < ») ou en incluant la clé de partition et en omettant une ou plusieurs colonnes de clustering. Avec Amazon Keyspaces, vous pouvez supprimer jusqu'à 1 000 lignes dans une plage en une seule opération.

Les suppressions de plages ne sont pas isolées. Les suppressions de lignes individuelles sont visibles pour les autres opérations lorsqu'une suppression de plage est en cours.

Tables système

Amazon Keyspaces remplit les tables système requises par les pilotes open source Cassandra d'Apache 2.0. Les tables système visibles par un client contiennent des informations propres à l'utilisateur authentifié. Les tables système sont entièrement contrôlées par Amazon Keyspaces et sont en lecture seule. Pour de plus amples informations, veuillez consulter [the section called “Keyspaces du système”](#).

Read-only l'accès aux tables système est requis, et vous pouvez le contrôler à l'aide des politiques d'accès IAM. Pour de plus amples informations, veuillez consulter [the section called “Gestion de l'accès à l'aide de politiques”](#). Vous devez définir des politiques de contrôle d'accès basées sur des balises pour les tables système différemment selon que vous utilisez le AWS SDK ou des appels d'API Cassandra Query Language (CQL) via les pilotes Cassandra et les outils de développement. Pour en savoir plus sur le contrôle d'accès basé sur des balises pour les tables

système, consultez [the section called “Accès aux ressources Amazon Keyspaces basé sur des balises”](#).

Si vous accédez à Amazon Keyspaces via des [points de terminaison Amazon VPC](#), des entrées apparaissent dans le tableau pour `system.peers` chaque point de terminaison Amazon VPC qu'Amazon Keyspaces est autorisé à consulter. Par conséquent, votre pilote Cassandra peut émettre un [message d'avertissement](#) concernant le nœud de contrôle lui-même dans le `system.peers` tableau. Vous pouvez ignorer cet avertissement en toute sécurité.

Horodatages

Dans Amazon Keyspaces, les horodatages au niveau des cellules compatibles avec les horodatages par défaut d'Apache Cassandra sont une fonctionnalité optionnelle.

La `USING TIMESTAMP` clause et la `WRITETIME` fonction ne sont disponibles que lorsque les horodatages côté client sont activés pour une table. Pour en savoir plus sur les horodatages côté client dans Amazon Keyspaces, consultez [the section called “Client-side horodatages”](#)

User-defined types (UDT)

L'opérateur d'inégalité n'est pas pris en charge pour les UDT dans Amazon Keyspaces.

Pour savoir comment utiliser les UDT dans Amazon Keyspaces, consultez [the section called “User-defined types \(UDT\)”](#)

Pour connaître le nombre d'UDT pris en charge par espace de touches, les niveaux d'imbrication pris en charge et les autres valeurs et quotas par défaut liés aux UDT, voir [the section called “Quotas et valeurs par défaut pour les types définis par l'utilisateur \(UDT\) dans Amazon Keyspaces”](#)

API, opérations, fonctions et types de données Cassandra pris en charge

Amazon Keyspaces (pour Apache Cassandra) est compatible avec l'API Cassandra Query Language (CQL) 3.11 (rétrocompatible avec la version 2.x).

Amazon Keyspaces prend en charge toutes les opérations courantes du plan de données Cassandra, telles que la création d'espaces de touches et de tables, la lecture de données et l'écriture de données.

Les sections suivantes répertorient les fonctionnalités prises en charge.

Rubriques

- [Support de l'API Cassandra](#)
- [Support de l'API du plan de contrôle Cassandra](#)
- [Support de l'API du plan de données Cassandra](#)
- [Support de la fonction Cassandra](#)
- [Support des types de données Cassandra](#)

Support de l'API Cassandra

Opération API	Pris en charge
CREATE KEYSPACE	Oui
ALTER KEYSPACE	Oui
DROP KEYSPACE	Oui
CREATE TABLE	Oui
ALTER TABLE	Oui
DROP TABLE	Oui
CREATE INDEX	Non
DROP INDEX	Non
UNLOGGED BATCH	Oui
LOGGED BATCH	Oui
SELECT	Oui
INSERT	Oui
DELETE	Oui
UPDATE	Oui

Opération API	Pris en charge
USE	Oui
CREATE TYPE	Oui
ALTER TYPE	Non
DROP TYPE	Oui
CREATE TRIGGER	Non
DROP TRIGGER	Non
CREATE FUNCTION	Non
DROP FUNCTION	Non
CREATE AGGREGATE	Non
DROP AGGREGATE	Non
CREATE MATERIALIZED VIEW	Non
ALTER MATERIALIZED VIEW	Non
DROP MATERIALIZED VIEW	Non
TRUNCATE	Non

Support de l'API du plan de contrôle Cassandra

Amazon Keyspaces étant géré, les opérations de l'API du plan de contrôle Cassandra pour gérer les paramètres des clusters et des nœuds ne sont pas requises. Par conséquent, les fonctionnalités Cassandra suivantes ne sont pas applicables.

Fonctionnalité	Raison
Basculement des écritures durables	Toutes les écritures sont durables

Fonctionnalité	Raison
Lire les paramètres de réparation	Non applicable
Secondes de grâce GC	Non applicable
Paramètres du filtre de Bloom	Non applicable
Paramètres de compactage	Non applicable
Compression settings (Paramètres de compression)	Non applicable
Paramètres de mise en cache	Non applicable
Réglages de sécurité	Remplacé par IAM

Support de l'API du plan de données Cassandra

Fonctionnalité	Pris en charge
Support JSON pour les instructions SELECT et INSERT	Oui
Colonnes statiques	Oui
Durée de vie (TTL)	Oui

Support de la fonction Cassandra

Pour plus d'informations sur les fonctions prises en charge, consultez [the section called “Built-in functions”](#).

Fonction	Pris en charge
Fonctions Aggregate	Non
Blobconversion	Oui

Fonction	Pris en charge
Cast	Oui
Fonctions Datetime	Oui
Fonctions de conversion temporelle	Oui
Fonctions TimeUuid	Oui
Token	Oui
User defined functions (UDF)	Non
Uuid	Oui

Support des types de données Cassandra

Le tableau suivant répertorie les types de données Apache Cassandra pris en charge dans Amazon Keyspaces. Pour plus d'informations sur les types de données dans Amazon Keyspaces, consultez [the section called “Types de données”](#)

Type de données	Pris en charge	
ascii	Oui	
bigint	Oui	
blob	Oui	
boolean	Oui	
counter	Oui	
date	Oui	
decimal	Oui	
double	Oui	

Type de données	Pris en charge	
float	Oui	
frozen	Oui	
inet	Oui	
int	Oui	
list	Oui	
map	Oui	
set	Oui	
smallint	Oui	
text	Oui	
time	Oui	
timestamp	Oui	
timeuuid	Oui	
tinyint	Oui	
tuple	Oui	
user-defined types (UDTs)	Oui	
uuid	Oui	
varchar	Oui	
varint	Oui	

Niveaux de cohérence en lecture et écriture pris en charge par Apache Cassandra et coûts associés

Les rubriques de cette section décrivent les niveaux de cohérence d'Apache Cassandra pris en charge pour les opérations de lecture et d'écriture dans Amazon Keyspaces (pour Apache Cassandra).

Rubriques

- [Niveaux de cohérence en écriture](#)
- [Lire les niveaux de cohérence](#)
- [Niveaux de cohérence non pris en charge](#)

Niveaux de cohérence en écriture

Amazon Keyspaces reproduit trois fois toutes les opérations d'écriture sur plusieurs zones de disponibilité pour garantir durabilité et haute disponibilité. Les écritures sont stockées durablement avant d'être reconnues à l'aide du niveau de cohérence `LOCAL_QUORUM`. Pour chaque écriture de 1 Ko, vous êtes facturé 1 unité de capacité d'écriture (WCU) pour les tables utilisant le mode de capacité allouée ou 1 unité de demande d'écriture (WRU) pour les tables utilisant le mode à la demande.

Vous pouvez `cqlsh` utiliser le code suivant pour définir la cohérence de toutes les requêtes de la session `LOCAL_QUORUM` en cours.

```
CONSISTENCY LOCAL_QUORUM;
```

Pour configurer le niveau de cohérence par programmation, vous pouvez définir la cohérence avec les pilotes clients Cassandra appropriés. Par exemple, les pilotes Java de la version 4.x vous permettent de définir le niveau de cohérence dans le `app.config` fichier comme indiqué ci-dessous.

```
basic.request.consistency = LOCAL_QUORUM
```

Si vous utilisez un pilote Java Cassandra version 3.x, vous pouvez spécifier le niveau de cohérence de la session en ajoutant `.withQueryOptions(new QueryOptions().setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM))` comme indiqué dans l'exemple de code suivant.


```
Session session = Cluster.builder()
    .addContactPoint(endPoint)
    .withPort(portNumber)
    .withAuthProvider(new SigV4AuthProvider("us-east-1"))
    .withSSL()
    .withQueryOptions(new
QueryOptions().setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
    .build())
    .connect();
```

Pour configurer le niveau de cohérence pour des opérations d'écriture spécifiques, vous pouvez définir la cohérence lorsque vous appelez `QueryBuilder.insertInto` avec un `setConsistencyLevel` argument lorsque vous utilisez le pilote Java.

Lire les niveaux de cohérence

Amazon Keyspaces prend en charge trois niveaux de cohérence de lecture : `ONELOCAL_ONE`, et `LOCAL_QUORUM`. Au cours d'une `LOCAL_QUORUM` lecture, Amazon Keyspaces renvoie une réponse reflétant les mises à jour les plus récentes issues de toutes les opérations d'écriture réussies. L'utilisation du niveau de cohérence `ONE` ou `LOCAL_ONE` peut améliorer les performances et la disponibilité de vos demandes de lecture, mais la réponse peut ne pas refléter les résultats d'une écriture récemment terminée.

Pour chaque lecture de 4 Ko à l'aide de la cohérence `ONE` ou `LOCAL_ONE`, vous êtes facturé 0,5 unités de capacité de lecture (RCU) pour les tables utilisant le mode de capacité allouée ou 0,5 unités de demande de lecture (RRU) pour les tables utilisant le mode à la demande. Pour chaque lecture de 4 Ko avec la cohérence `LOCAL_QUORUM`, vous êtes facturé 1 unité de capacité de lecture (RCU) pour les tables utilisant le mode de capacité allouée ou 1 unité de demande de lecture (RRU) pour les tables utilisant le mode à la demande.

Facturation basée sur la cohérence de lecture et le mode de débit de capacité de lecture par table pour chaque 4 Ko de lecture

Niveau de cohérence	Alloué	On-demand
ONE	0,5 RCU (Read Capacity Unit, unité de capacité de lecture)	0,5 RRU
LOCAL_ONE	0,5 RCU (Read Capacity Unit, unité de capacité de lecture)	0,5 RRU

Niveau de cohérence	Alloué	On-demand
LOCAL_QUORUM	1 RCU	1 RRU

Pour spécifier une cohérence différente pour les opérations de lecture, appelez `QueryBuilder.select` avec un `setConsistencyLevel` argument lorsque vous utilisez le pilote Java.

Niveaux de cohérence non pris en charge

Les niveaux de cohérence suivants ne sont pas pris en charge par Amazon Keyspaces et entraîneront des exceptions.

Niveaux de cohérence non pris en charge

Apache Cassandra	Amazon Keyspaces
EACH_QUORUM	Non pris en charge
QUORUM	Non pris en charge
ALL	Non pris en charge
TWO	Non pris en charge
THREE	Non pris en charge
ANY	Non pris en charge
SERIAL	Non pris en charge
LOCAL_SERIAL	Non pris en charge

Migration vers Amazon Keyspaces (pour Apache Cassandra)

La migration vers Amazon Keyspaces (pour Apache Cassandra) présente de nombreux avantages intéressants pour les entreprises et les organisations. Voici quelques avantages clés qui font d'Amazon Keyspaces un choix intéressant pour la migration.

- **Évolutivité** — Amazon Keyspaces est conçu pour gérer des charges de travail massives et évoluer de manière fluide afin de s'adapter à l'augmentation des volumes de données et du trafic. Avec le Cassandra traditionnel, le dimensionnement n'est pas effectué à la demande et nécessite de planifier les pics futurs. Avec Amazon Keyspaces, vous pouvez facilement augmenter ou diminuer vos tableaux en fonction de la demande, afin que vos applications puissent gérer des pics de trafic soudains sans compromettre les performances.
- **Performances** — Amazon Keyspaces offre un accès aux données à faible latence, permettant aux applications de récupérer et de traiter les données à une vitesse exceptionnelle. Son architecture distribuée garantit que les opérations de lecture et d'écriture sont réparties sur plusieurs nœuds, et offre des temps de réponse constants de l'ordre de la milliseconde, même à des taux de demandes élevés.
- **Entièrement géré** — Amazon Keyspaces est un service entièrement géré fourni par AWS. Cela signifie qu'il AWS gère les aspects opérationnels de la gestion des bases de données, notamment le provisionnement, la configuration, les correctifs, les sauvegardes et le dimensionnement. Cela vous permet de vous concentrer davantage sur le développement de vos applications et moins sur les tâches d'administration de base de données.
- **Architecture sans serveur** — Amazon Keyspaces fonctionne sans serveur. Vous ne payez que pour la capacité consommée, sans qu'aucun provisionnement de capacité initial ne soit requis. Vous n'avez pas de serveurs à gérer ni d'instances à choisir. Ce modèle de paiement à la demande permet de réduire les coûts et de minimiser les frais opérationnels, car vous ne payez que pour les ressources que vous utilisez sans avoir à provisionner et à surveiller les capacités.
- **Flexibilité NoSQL avec schéma** — Amazon Keyspaces suit un modèle de données NoSQL, offrant ainsi de la flexibilité dans la conception des schémas. Avec Amazon Keyspaces, vous pouvez stocker des données structurées, semi-structurées et non structurées, ce qui les rend parfaitement adaptées à la gestion de types de données divers et évolutifs. En outre, Amazon Keyspaces valide le schéma en écriture, ce qui permet une évolution centralisée du modèle de données. Cette flexibilité permet d'accélérer les cycles de développement et de s'adapter plus facilement à l'évolution des besoins de l'entreprise.

- Haute disponibilité et durabilité : Amazon Keyspaces réplique les données dans plusieurs [zones de disponibilité](#) au sein d'un même Région AWS, garantissant ainsi une haute disponibilité et une durabilité des données. Il gère automatiquement la réplication, le basculement et la restauration, minimisant ainsi le risque de perte de données ou d'interruption de service. Amazon Keyspaces fournit un SLA de disponibilité allant jusqu'à 99,999 %. [Pour encore plus de résilience et des lectures locales à faible latence, Amazon Keyspaces propose une réplication multirégionale.](#)
- Sécurité et conformité : Amazon Keyspaces s'intègre Gestion des identités et des accès AWS pour un contrôle d'accès précis. Il fournit un chiffrement au repos et en transit, ce qui contribue à améliorer la sécurité de vos données. Amazon Keyspaces a été évalué par des auditeurs tiers en termes de sécurité et de conformité à des programmes spécifiques, notamment HIPAA, PCI DSS et SOC, vous permettant de répondre aux exigences réglementaires. Pour de plus amples informations, veuillez consulter [the section called "Validation de conformité"](#).
- Intégration à AWS l'écosystème — Dans le cadre de l' AWS écosystème, Amazon Keyspaces s'intègre parfaitement à d'autres Services AWS, par exemple AWS CloudFormation Amazon CloudWatch et. AWS CloudTrail Cette intégration vous permet de créer des architectures sans serveur, de tirer parti de l'infrastructure sous forme de code et de créer des applications pilotées par les données en temps réel. Pour de plus amples informations, veuillez consulter [Surveillance d'Amazon Keyspaces](#).

Rubriques

- [Créez un plan de migration pour migrer d'Apache Cassandra vers Amazon Keyspaces](#)
- [Comment sélectionner le bon outil pour le téléchargement groupé ou la migration de données vers Amazon Keyspaces](#)

Créez un plan de migration pour migrer d'Apache Cassandra vers Amazon Keyspaces

Pour une migration réussie d'Apache Cassandra vers Amazon Keyspaces, nous vous recommandons de passer en revue les concepts de migration applicables et les meilleures pratiques, ainsi que de comparer les options disponibles.

Cette rubrique décrit le fonctionnement du processus de migration en présentant plusieurs concepts clés ainsi que les outils et techniques à votre disposition. Vous pouvez évaluer les différentes stratégies de migration afin de sélectionner celle qui répond le mieux à vos besoins.

Rubriques

- [Compatibilité fonctionnelle](#)
- [Estimez le prix d'Amazon Keyspaces](#)
- [Choisissez une stratégie de migration](#)
- [Migration en ligne vers Amazon Keyspaces : stratégies et bonnes pratiques](#)
- [Processus de migration hors ligne : Apache Cassandra vers Amazon Keyspaces](#)
- [Utilisation d'une solution de migration hybride : Apache Cassandra vers Amazon Keyspaces](#)

Compatibilité fonctionnelle

Examinez attentivement les différences fonctionnelles entre Apache Cassandra et Amazon Keyspaces avant la migration. Amazon Keyspaces prend en charge toutes les opérations de plan de données Cassandra couramment utilisées, telles que la création d'espaces clés et de tableaux, la lecture de données et l'écriture de données.

Cependant, certaines API Cassandra ne sont pas prises en charge par Amazon Keyspaces. Pour plus d'informations sur les API prises en charge, consultez [the section called “API, opérations, fonctions et types de données Cassandra pris en charge”](#). Pour une vue d'ensemble de toutes les différences fonctionnelles entre Amazon Keyspaces et Apache Cassandra, consultez [the section called “Différences fonctionnelles avec Apache Cassandra”](#)

Pour comparer les API et le schéma Cassandra que vous utilisez avec les fonctionnalités prises en charge dans Amazon Keyspaces, vous pouvez exécuter un script de compatibilité disponible dans la boîte à outils Amazon Keyspaces sur [GitHub](#)

Comment utiliser le script de compatibilité

1. Téléchargez le script Python de compatibilité depuis [GitHub](#) et déplacez-le vers un emplacement ayant accès à votre cluster Apache Cassandra existant.
2. Le script de compatibilité utilise des paramètres similaires à ceux utilisés CQLSH. Pour `--host` et `--port` entrez l'adresse IP et le port que vous utilisez pour vous connecter et exécuter des requêtes sur l'un des nœuds Cassandra de votre cluster.

Si votre cluster Cassandra utilise l'authentification, vous devez également fournir `-username` et `-password`. Pour exécuter le script de compatibilité, vous pouvez utiliser la commande suivante.

```
python toolkit-compat-tool.py --host hostname or IP -u "username" -p "password" --  
port native transport port
```

Estimez le prix d'Amazon Keyspaces

Cette section fournit une vue d'ensemble des informations que vous devez collecter à partir de vos tables Apache Cassandra pour calculer le coût estimé d'Amazon Keyspaces. Chacune de vos tables nécessite différents types de données, doit prendre en charge différentes requêtes CQL et gère un read/write trafic distinct.

La définition de vos besoins sur la base de tableaux s'aligne sur les modes d'isolation des ressources au niveau des tables et [read/write de capacité de débit d'Amazon Keyspace](#). Avec Amazon Keyspaces, vous pouvez définir des politiques de read/write capacité et de dimensionnement [automatique](#) pour les tables de manière indépendante.

Comprendre les exigences relatives aux tables vous permet de hiérarchiser les tables à migrer en fonction des fonctionnalités, des coûts et de l'effort de migration.

Collectez les métriques suivantes de la table Cassandra avant une migration. Ces informations permettent d'estimer le coût de votre charge de travail sur Amazon Keyspaces.

- Nom de la table : nom de l'espace clé complet et nom de la table.
- Description : description de la table, par exemple comment elle est utilisée ou quel type de données y est stocké.
- Nombre moyen de lectures par seconde : nombre moyen de lectures au niveau des coordonnées par rapport à la table sur un intervalle de temps important.
- Nombre moyen d'écritures par seconde : nombre moyen d'écritures au niveau des coordonnées par rapport à la table sur un intervalle de temps important.
- Taille moyenne des lignes en octets : taille moyenne des lignes en octets.
- Taille de stockage en Go : taille de stockage brute d'une table.
- Répartition de la cohérence des lectures : pourcentage de lectures utilisant une cohérence finale (LOCAL_ONE ou ONE) par rapport à une cohérence forte (LOCAL_QUORUM).

Ce tableau présente un exemple des informations relatives à vos tableaux que vous devez rassembler lors de la planification d'une migration.

Nom de la table	Description	Nombre moyen de lectures par seconde	Nombre moyen d'écritures par seconde	Taille moyenne des lignes en octets	Taille de stockage en Go	Lire la répartition de la cohérence
mykeyspace.mytable	Utilisé pour stocker l'historique du panier	10 000	5 000	2 200	2 000	100 % LOCAL_ONE
mykeyspace.mytable2	Utilisé pour stocker les dernières informations de profil	20 000	1 000	850	1 000	25 % LOCAL_QUORUM 75 % LOCAL_ONE

Comment collecter les métriques des tableaux

Cette section fournit des instructions étape par étape sur la manière de collecter les métriques de table nécessaires à partir de votre cluster Cassandra existant. Ces mesures incluent la taille des lignes, la taille des tables et les read/write demandes par seconde (RPS). Ils vous permettent d'évaluer les exigences en matière de capacité de débit pour un tableau Amazon Keyspaces et d'estimer les prix.

Comment collecter les métriques de la table dans la table source de Cassandra

1. Déterminer la taille des lignes

La taille des lignes est importante pour déterminer la capacité de lecture et l'utilisation de la capacité d'écriture dans Amazon Keyspaces. Le schéma suivant montre la distribution typique des données sur une plage de jetons Cassandra.



Vous pouvez utiliser un script d'échantillonnage de taille de ligne disponible sur [GitHub](#) pour collecter des mesures de taille de ligne pour chaque table de votre cluster Cassandra.

Le script exporte les données des tables depuis Apache Cassandra en utilisant `cqlsh` et `awk` pour calculer l'écart minimum, maximum, moyen et type de la taille des lignes sur un ensemble d'échantillons configurable de données de table. L'échantillonneur de taille de ligne transmet les arguments à `cqlsh`, de sorte que les mêmes paramètres peuvent être utilisés pour vous connecter et lire à partir de votre cluster Cassandra.

La déclaration suivante en est un exemple.

```
./row-size-sampler.sh 10.22.33.44 9142 \\  
-u "username" -p "password" --ssl
```

Pour plus d'informations sur la façon dont la taille des lignes est calculée dans Amazon Keyspaces, consultez [the section called "Estimer la taille des lignes"](#).

2. Déterminer la taille de la table

Avec Amazon Keyspaces, vous n'avez pas besoin de provisionner l'espace de stockage à l'avance. Amazon Keyspaces surveille en permanence la taille facturable de vos tables afin de déterminer vos frais de stockage. Le stockage est facturé au. GB-month La taille du tableau Amazon Keyspaces est basée sur la taille brute (non compressée) d'une seule réplique.

Pour contrôler la taille du tableau dans Amazon Keyspaces, vous pouvez utiliser la métrique `BillableTableSizeInBytes`, qui est affichée pour chaque tableau dans le Console de gestion AWS.

Pour estimer la taille facturable de votre tableau Amazon Keyspaces, vous pouvez utiliser l'une des deux méthodes suivantes :

- Utilisez la taille moyenne des lignes et multipliez par le nombre de lignes.

Vous pouvez estimer la taille de la table Amazon Keyspaces en multipliant la taille moyenne des lignes par le nombre de lignes de votre table source Cassandra. Utilisez l'exemple de script de taille de ligne de la section précédente pour capturer la taille moyenne des lignes. Pour capturer le nombre de lignes, vous pouvez utiliser des outils tels que `dsbulk count` pour déterminer le nombre total de lignes de votre table source.

- Utilisez le `nodetool` pour collecter les métadonnées du tableau.

`Nodetool` est un outil d'administration fourni dans la distribution Apache Cassandra qui fournit un aperçu de l'état du processus Cassandra et renvoie les métadonnées des tables. Vous pouvez les utiliser `nodetool` pour échantillonner des métadonnées relatives à la taille de la table et ainsi extrapoler la taille de la table dans Amazon Keyspaces.

La commande à utiliser est `nodetool tablestats`. `Tablestats` renvoie la taille et le taux de compression du tableau. La taille de la table est enregistrée comme `tablelivespace` pour la table et vous pouvez la diviser par `compression ratio`. Multipliez ensuite cette valeur de taille par le nombre de nœuds. Enfin, divisez par le facteur de réplication (généralement trois).

Il s'agit de la formule complète pour le calcul que vous pouvez utiliser pour évaluer la taille de la table.

```
((tablelivespace / compression ratio) * (total number of nodes)) / (replication factor)
```

Supposons que votre cluster Cassandra possède 12 nœuds. L'exécution de la `nodetool tablestats` commande renvoie une valeur `tablelivespace` de 200 Go et une valeur `compression ratio` de 0,5. Le keyspace possède un facteur de réplication de trois.

Voici à quoi ressemble le calcul pour cet exemple.

```
(200 GB / 0.5) * (12 nodes) / (replication factor of 3)
= 4,800 GB / 3
```

```
Keyspaces = 1,600 GB is the table size estimate for Amazon
```

3. Capturez le nombre de lectures et d'écritures

Pour déterminer les exigences de capacité et d'évolutivité de vos tables Amazon Keyspaces, capturez le taux de demandes de lecture et d'écriture de vos tables Cassandra avant la migration.

Amazon Keyspaces fonctionne sans serveur et vous ne payez que pour ce que vous utilisez. En général, le prix du read/write débit dans Amazon Keyspaces est basé sur le nombre et la taille des requêtes.

Amazon Keyspaces propose deux modes de capacité :

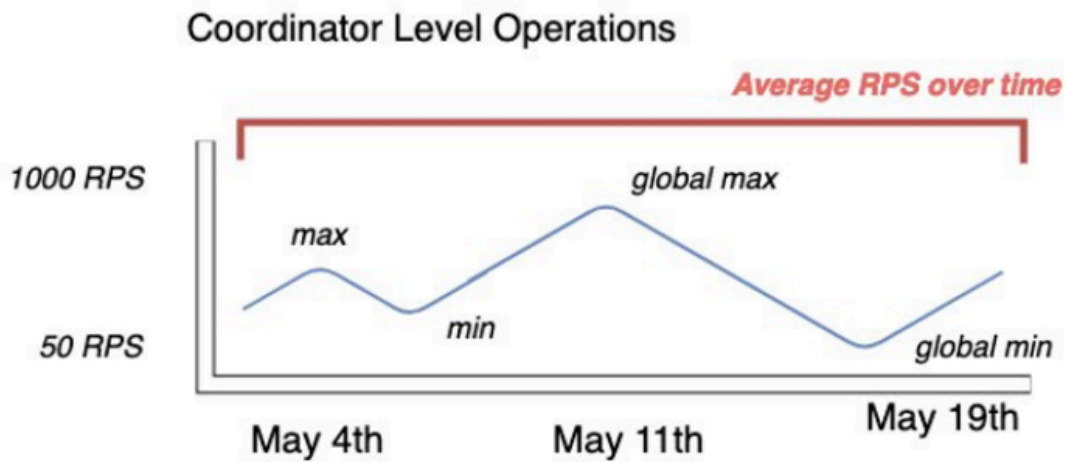
- [On-demand](#) — Il s'agit d'une option de facturation flexible capable de traiter des milliers de demandes par seconde sans qu'il soit nécessaire de planifier les capacités. Il propose une tarification à la demande pour les demandes de lecture et d'écriture afin que vous ne payiez que pour ce que vous utilisez.
- [Provisionné](#) : si vous choisissez le mode de capacité de débit provisionnée, vous spécifiez le nombre de lectures et d'écritures par seconde requis pour votre application. Cela vous permet de gérer votre utilisation d'Amazon Keyspaces pour qu'elle reste inférieure ou égale à un taux de demandes défini afin de préserver la prévisibilité.

Le mode provisionné propose une mise [à l'échelle](#) automatique pour ajuster automatiquement votre débit provisionné pour augmenter ou diminuer afin d'améliorer l'efficacité opérationnelle. Pour plus d'informations sur la gestion des ressources sans serveur, consultez [Gestion des ressources sans serveur](#).

Étant donné que vous provisionnez séparément la capacité de débit de lecture et d'écriture dans Amazon Keyspaces, vous devez mesurer indépendamment le taux de demandes pour les lectures et les écritures dans vos tables existantes.

Pour recueillir les mesures d'utilisation les plus précises de votre cluster Cassandra existant, capturez la moyenne des demandes par seconde (RPS) pour les opérations de lecture et d'écriture au niveau du coordinateur sur une période prolongée pour une table agrégée sur tous les nœuds d'un seul centre de données.

La capture du RPS moyen sur une période d'au moins plusieurs semaines permet de saisir les pics et les creux de vos modèles de trafic, comme le montre le schéma suivant.



Deux options s'offrent à vous pour déterminer le taux de demandes de lecture et d'écriture de votre table Cassandra.

- Utiliser la surveillance Cassandra existante

Vous pouvez utiliser les mesures présentées dans le tableau suivant pour observer les demandes de lecture et d'écriture. Notez que les noms des métriques peuvent changer en fonction de l'outil de surveillance que vous utilisez.

Dimension	Métrique Cassandra JMX
Écritures	<code>org.apache.cassandra.metrics:type=ClientRequest,scope=Write,name=Latency#Count</code>
Lectures	<code>org.apache.cassandra.metrics:type=ClientRequest,scope=Read,name=Latency#Count</code>

- Utilisation de la `nodetool`

Utilisez `nodetool tablestats` et `nodetool info` pour capturer les opérations de lecture et d'écriture moyennes à partir de la table. `tablestats` renvoie le nombre total de lectures et d'écritures depuis le moment où le nœud a été initié. `nodetool info` fournit le temps de disponibilité d'un nœud en secondes.

Pour obtenir la moyenne des lectures et des écritures par seconde, divisez le nombre de lectures et d'écritures par le temps de disponibilité du nœud en secondes. Ensuite, pour les lectures, vous divisez par le niveau de cohérence et pour les écritures, vous divisez par le facteur de réplication. Ces calculs sont exprimés dans les formules suivantes.

Formule pour le nombre moyen de lectures par seconde :

```
((number of reads * number of nodes in cluster) / read consistency quorum
(2)) / uptime
```

Formule pour le nombre moyen d'écritures par seconde :

```
((number of writes * number of nodes in cluster) / replication factor of 3) /
uptime
```

Supposons que nous ayons un cluster de 12 nœuds en activité depuis 4 semaines. `nodetool info` renvoie 2 419 200 secondes de disponibilité et `nodetool tablestats` renvoie 1 milliard d'écritures et 2 milliards de lectures. Cet exemple aboutirait au calcul suivant.

```
((2 billion reads * 12 in cluster) / read consistency quorum (2)) / 2,419,200
seconds
= 12 billion reads / 2,419,200 seconds
= 4,960 read request per second
((1 billion writes * 12 in cluster) / replication
factor of 3) / 2,419,200 seconds
= 4 billion writes / 2,419,200 seconds
= 1,653 write request per second
```

4. Déterminez l'utilisation de la capacité de la table

Pour estimer l'utilisation moyenne de la capacité, commencez par les taux de demande moyens et la taille moyenne des lignes de votre table source Cassandra.

Amazon Keyspaces utilise des unités de capacité de lecture (RCU) et des unités de capacité d'écriture (WCU) pour mesurer la capacité de débit allouée pour les lectures et les écritures des tables. Pour cette estimation, nous utilisons ces unités pour calculer les besoins en capacité de lecture et d'écriture de la nouvelle table Amazon Keyspaces après la migration.

Plus loin dans cette rubrique, nous verrons comment le choix entre le mode de capacité provisionné et le mode de capacité à la demande affecte la facturation. Mais pour l'estimation de l'utilisation de la capacité dans cet exemple, nous supposons que la table est en mode provisionné.

- **Lectures** : une RCU représente une LOCAL_QUORUM ou deux demandes de LOCAL_ONE lecture pour une ligne d'une taille maximale de 4 Ko. Si vous devez lire une ligne de plus de 4 Ko, l'opération de lecture utilise des RCU supplémentaires. Le nombre total de RCU requis dépend de la taille de la ligne et de la question de savoir si vous souhaitez utiliser LOCAL_QUORUM ou LOCAL_ONE lire la cohérence.

Par exemple, la lecture d'une ligne de 8 Ko nécessite 2 UCR utilisant la cohérence de LOCAL_QUORUM lecture, et 1 RCU si vous choisissez la cohérence de LOCAL_ONE lecture.

- **Écritures** : une WCU représente une écriture pour une ligne d'une taille maximale de 1 Ko. Toutes les écritures utilisent la cohérence LOCAL_QUORUM et il n'y a pas de frais supplémentaires pour l'utilisation de transactions légères (LWT).

Le nombre total de WCU requis dépend de la taille de la ligne. Si vous devez écrire une ligne supérieure à 1 Ko, l'opération d'écriture utilise des WCU supplémentaires. Par exemple, si la taille de votre ligne est de 2 Ko, vous avez besoin de 2 WCU pour effectuer une demande d'écriture.

La formule suivante peut être utilisée pour estimer les UCR et les WCU requises.

- La capacité de lecture des UCR peut être déterminée en multipliant le nombre de lectures par seconde par le nombre de lignes lues par lecture multiplié par la taille moyenne des lignes divisée par 4 Ko et arrondie au nombre entier le plus proche.
- La capacité d'écriture dans les WCU peut être déterminée en multipliant le nombre de demandes par la taille moyenne des lignes divisée par 1 Ko et arrondie au nombre entier le plus proche.

Ceci est exprimé dans les formules suivantes.

$$\text{Read requests per second} * \text{ROUNDUP}((\text{Average Row Size})/4096 \text{ per unit}) = \text{RCUs per second}$$

$$\text{Write requests per second} * \text{ROUNDUP}(\text{Average Row Size}/1024 \text{ per unit}) = \text{WCUs per second}$$

Par exemple, si vous effectuez 4 960 demandes de lecture avec une taille de ligne de 2,5 Ko sur votre table Cassandra, vous avez besoin de 4 960 RCU dans Amazon Keyspaces. Si vous effectuez actuellement 1 653 demandes d'écriture par seconde avec une taille de ligne de 2,5 Ko sur votre table Cassandra, vous avez besoin de 4 959 WCU par seconde dans Amazon Keyspaces.

Cet exemple est exprimé dans les formules suivantes.

$$\begin{aligned} &4,960 \text{ read requests per second} * \text{ROUNDUP}(2.5\text{KB} / 4\text{KB bytes per unit}) \\ &= 4,960 \text{ read requests per second} * 1 \text{ RCU} \\ &= 4,960 \text{ RCUs} \end{aligned}$$

$$\begin{aligned} &1,653 \text{ write requests per second} * \text{ROUNDUP}(2.5\text{KB}/1\text{KB per unit}) \\ &= 1,653 \text{ requests per second} * 3 \text{ WCUs} \\ &= 4,959 \text{ WCUs} \end{aligned}$$

L'utilisation vous eventual consistency permet d'économiser jusqu'à la moitié de la capacité de débit sur chaque demande de lecture. Chaque lecture finalement cohérente peut consommer jusqu'à 8 Ko. Vous pouvez calculer d'éventuelles lectures cohérentes en multipliant le calcul précédent par 0,5, comme indiqué dans la formule suivante.

$$\begin{aligned} &4,960 \text{ read requests per second} * \text{ROUNDUP}(2.5\text{KB} / 4\text{KB per unit}) * .5 \\ &= 2,480 \text{ read request per second} * 1 \text{ RCU} \\ &= 2,480 \text{ RCUs} \end{aligned}$$

5. Calculez l'estimation de prix mensuelle pour Amazon Keyspaces

Pour estimer la facturation mensuelle de la table en fonction du débit de read/write capacité, vous pouvez calculer la tarification pour le mode à la demande et pour le mode provisionné à l'aide de différentes formules et comparer les options de votre tableau.

Mode provisionné : la consommation de capacité de lecture et d'écriture est facturée selon un taux horaire basé sur les unités de capacité par seconde. Tout d'abord, divisez ce taux par 0,7 pour représenter l'utilisation cible d'autoscaling par défaut de 70 %. Multipliez ensuite par 30 jours civils, 24 heures sur 24, et le tarif régional.

Ce calcul est résumé dans les formules suivantes.

```
(read capacity per second / .7) * 24 hours * 30 days * regional rate
(write capacity per second / .7) * 24 hours * 30 days * regional
rate
```

On-demand mode — Les capacités de lecture et d'écriture sont facturées au tarif par demande. Commencez par multiplier le taux de demandes par 30 jours civils et 24 heures par jour. Divisez ensuite par un million d'unités de requête. Enfin, multipliez par le taux régional.

Ce calcul est résumé dans les formules suivantes.

```
((read capacity per second * 30 * 24 * 60 * 60) / 1 Million read request units) *
regional rate
((write capacity per second * 30 * 24 * 60 * 60) / 1 Million write
request units) * regional rate
```

Choisissez une stratégie de migration

Vous pouvez choisir entre les stratégies de migration suivantes lors de la migration d'Apache Cassandra vers Amazon Keyspaces :

- **En ligne** : il s'agit d'une migration dynamique utilisant deux écritures pour commencer à écrire de nouvelles données simultanément sur Amazon Keyspaces et le cluster Cassandra. Ce type de migration est recommandé pour les applications qui ne nécessitent aucun temps d'arrêt pendant la migration et qui garantissent une cohérence entre lecture et écriture.

Pour plus d'informations sur la planification et la mise en œuvre d'une stratégie de migration en ligne, consultez [the section called “Migration en ligne”](#).

- **Hors ligne** : cette technique de migration consiste à copier un ensemble de données de Cassandra vers Amazon Keyspaces pendant une période d'indisponibilité. La migration hors ligne peut

simplifier le processus de migration, car elle ne nécessite pas de modifications de votre application ni de résolution de conflits entre les données historiques et les nouvelles écritures.

Pour plus d'informations sur la planification d'une migration hors ligne, consultez [the section called “Migration hors ligne”](#).

- **Hybride** : cette technique de migration permet de répliquer les modifications dans Amazon Keyspaces quasiment en temps réel, mais sans cohérence entre lecture et écriture.

Pour plus d'informations sur la planification d'une migration hybride, consultez [the section called “Migration hybride”](#).

Après avoir passé en revue les techniques de migration et les meilleures pratiques abordées dans cette rubrique, vous pouvez placer les options disponibles dans un arbre de décision afin de concevoir une stratégie de migration en fonction de vos besoins et des ressources disponibles.

Migration en ligne vers Amazon Keyspaces : stratégies et bonnes pratiques

Si vous devez maintenir la disponibilité des applications lors d'une migration d'Apache Cassandra vers Amazon Keyspaces, vous pouvez préparer une stratégie de migration en ligne personnalisée en mettant en œuvre les principaux composants abordés dans cette rubrique. En suivant ces bonnes pratiques pour les migrations en ligne, vous pouvez vous assurer que la disponibilité des applications et la cohérence entre lecture et écriture sont maintenues tout au long du processus de migration, minimisant ainsi l'impact sur vos utilisateurs.

Lorsque vous concevez une stratégie de migration en ligne d'Apache Cassandra vers Amazon Keyspaces, vous devez prendre en compte les étapes clés suivantes.

1. Écrire de nouvelles données

- **Proxy à double écriture ZDM pour la migration d'Amazon Keyspaces** : utilisez le proxy à double écriture ZDM disponible sur [Github](#) pour effectuer une migration sans interruption d'Apache Cassandra vers Amazon Keyspaces. Le proxy ZDM effectue deux écritures sans qu'il soit nécessaire de refactoriser les applications existantes et effectue deux lectures pour la validation des requêtes.
- **Écritures doubles d'applications** : vous pouvez implémenter des écritures doubles dans votre application à l'aide des bibliothèques clientes et des pilotes Cassandra existants. Désignez une base de données comme leader et l'autre comme suiveuse. Les échecs d'écriture dans la base de données des abonnés sont enregistrés dans une file d'attente de lettres [mortes \(DLQ\) à des fins d'analyse](#).

- Écritures doubles au niveau de la messagerie : vous pouvez également configurer votre plateforme de messagerie existante pour envoyer des écritures à la fois à Cassandra et à Amazon Keyspaces en utilisant un client supplémentaire. Cela permet de créer des vues cohérentes dans les deux bases de données.

2. Migration des données historiques

- Copier les données historiques : vous pouvez migrer les données historiques de Cassandra vers Amazon Keyspaces à l'aide de AWS Glue scripts personnalisés d'extraction, de transformation et de chargement (ETL). Gérez la résolution des conflits entre les écritures doubles et les chargements en masse à l'aide de techniques telles que les transactions légères ou l'horodatage.
- Utilisation Time-To-Live (TTL) : pour des périodes de conservation des données plus courtes, vous pouvez utiliser le TTL à la fois dans Cassandra et Amazon Keyspaces afin d'éviter de télécharger des données historiques inutiles. Alors que les anciennes données expirent dans Cassandra et que de nouvelles données sont écrites par double écriture, Amazon Keyspaces finit par rattraper son retard.

3. Validation des données

- Lecture double : implémentez des lectures doubles à partir des bases de données Cassandra (principale) et Amazon Keyspaces (secondaire), en comparant les résultats de manière asynchrone. Les différences sont enregistrées ou envoyées à un DLQ.
- Exemples de lectures : utilisez les fonctions Λ pour échantillonner et comparer périodiquement les données des deux systèmes, en enregistrant toute anomalie dans un DLQ.

4. Migration de l'application

- Blue-green stratégie : modifiez votre application pour traiter Amazon Keyspaces comme le magasin de données principal et Cassandra comme le magasin de données secondaire en une seule étape. Surveillez les performances et revenez en arrière si des problèmes surviennent.
- Déploiement de Canary : commencez par déployer progressivement la migration vers un sous-ensemble d'utilisateurs, en augmentant progressivement le trafic vers Amazon Keyspaces en tant que site principal jusqu'à la migration complète.

5. Démantèlement de Cassandra

Une fois que votre application aura été entièrement migrée vers Amazon Keyspaces et que la cohérence des données aura été validée, vous pouvez planifier la mise hors service de votre cluster Cassandra en fonction des politiques de conservation des données.

En planifiant une stratégie de migration en ligne avec ces composants, vous pouvez effectuer une transition fluide vers le service Amazon Keyspaces entièrement géré avec un minimum de temps d'arrêt ou d'interruption. Les sections suivantes examinent chaque composant de manière plus détaillée.

Rubriques

- [Écrire de nouvelles données lors d'une migration en ligne](#)
- [Téléchargement de données historiques lors d'une migration en ligne](#)
- [Validation de la cohérence des données lors d'une migration en ligne](#)
- [Migration de l'application lors d'une migration en ligne](#)
- [Mise hors service de Cassandra après une migration en ligne](#)

Écrire de nouvelles données lors d'une migration en ligne

La première étape d'un plan de migration en ligne consiste à s'assurer que toutes les nouvelles données écrites par l'application sont stockées dans les deux bases de données, dans votre cluster Cassandra existant et dans Amazon Keyspaces. L'objectif est de fournir une vue cohérente sur les deux banques de données. Vous pouvez le faire en appliquant toutes les nouvelles écritures aux deux bases de données. Pour implémenter des écritures doubles, considérez l'une des trois options suivantes.

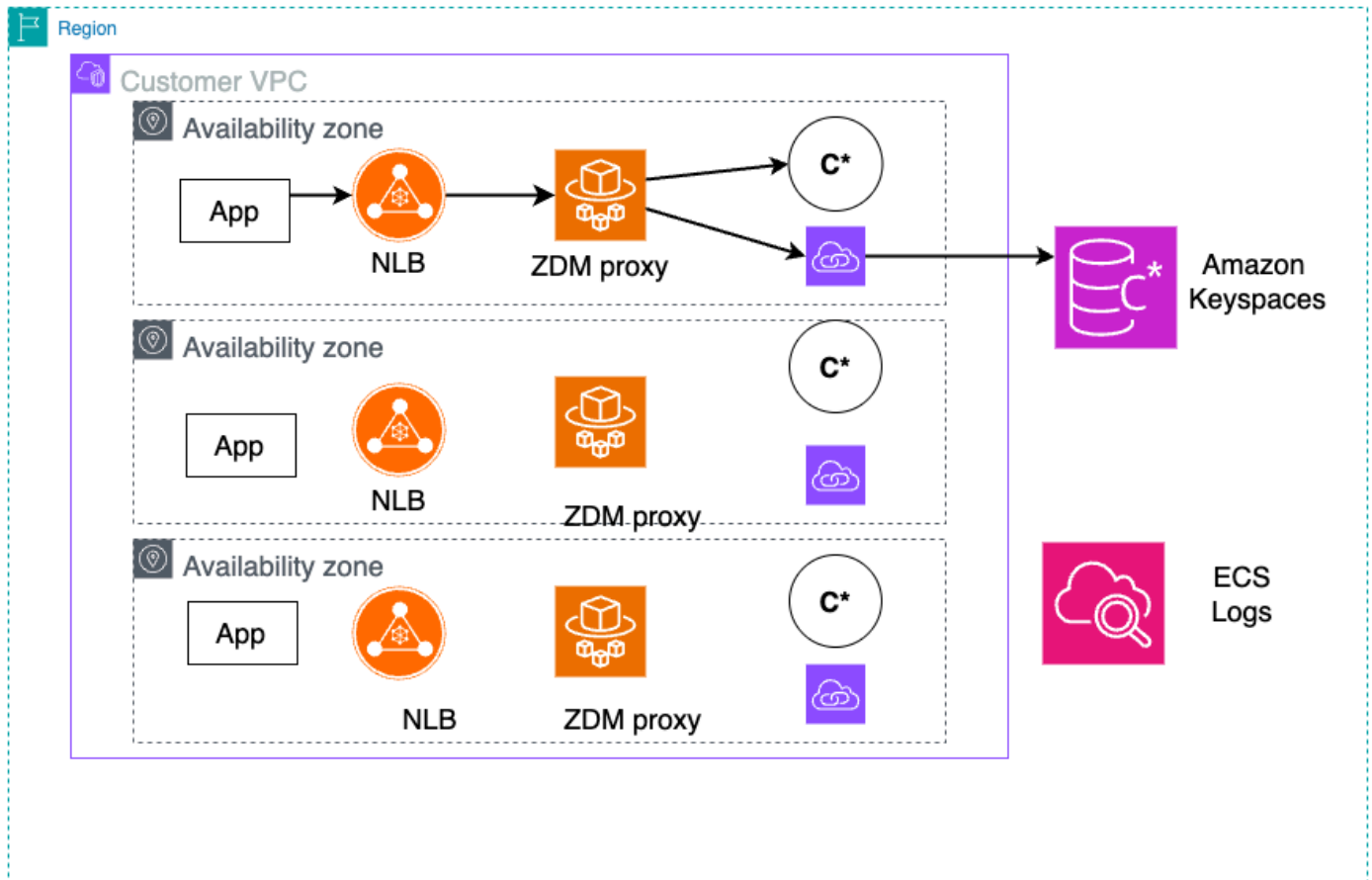
- Proxy à double écriture ZDM pour la migration d'Amazon Keyspaces — À l'aide du proxy ZDM pour Amazon Keyspaces disponible sur [Github](#), vous pouvez migrer vos charges de travail Apache Cassandra vers Amazon Keyspaces sans interruption des applications. Cette solution améliorée met en œuvre les AWS meilleures pratiques et étend les fonctionnalités officielles du proxy ZDM.
- Effectuez des migrations en ligne entre Apache Cassandra et Amazon Keyspaces.
- Écrivez des données simultanément dans les tables source et cible sans avoir à refactoriser les applications.
- Validez les requêtes par le biais d'opérations de double lecture.

La solution offre les améliorations suivantes en termes de compatibilité avec AWS Amazon Keyspaces.

- Déploiement de conteneurs : utilisez une image Docker préconfigurée provenant d'Amazon Elastic Container Registry (Amazon ECR) pour les VPC-accessible déploiements.

- Infrastructure en tant que code : déployez à l'aide AWS CloudFormation de modèles pour une configuration automatisée sur AWS Fargate.
- Compatibilité avec Amazon Keyspaces : accédez aux tables du système avec des adaptations personnalisées pour Amazon Keyspaces.

La solution fonctionne sur Amazon ECS avec Fargate, offrant une évolutivité sans serveur en fonction des exigences de votre charge de travail. Un équilibreur de charge réseau répartit le trafic applicatif entrant entre plusieurs tâches Amazon ECS pour une haute disponibilité.



- Écritures doubles d'applications : vous pouvez implémenter des écritures doubles en modifiant le moins possible le code de votre application en tirant parti des bibliothèques clientes et des pilotes Cassandra existants. Vous pouvez soit implémenter des écritures doubles dans votre application existante, soit créer une nouvelle couche dans l'architecture pour gérer les écritures doubles. Pour plus d'informations et une étude de cas client montrant comment la double écriture a été mise en œuvre dans une application existante, consultez l'étude [de cas sur la migration de](#) Cassandra.

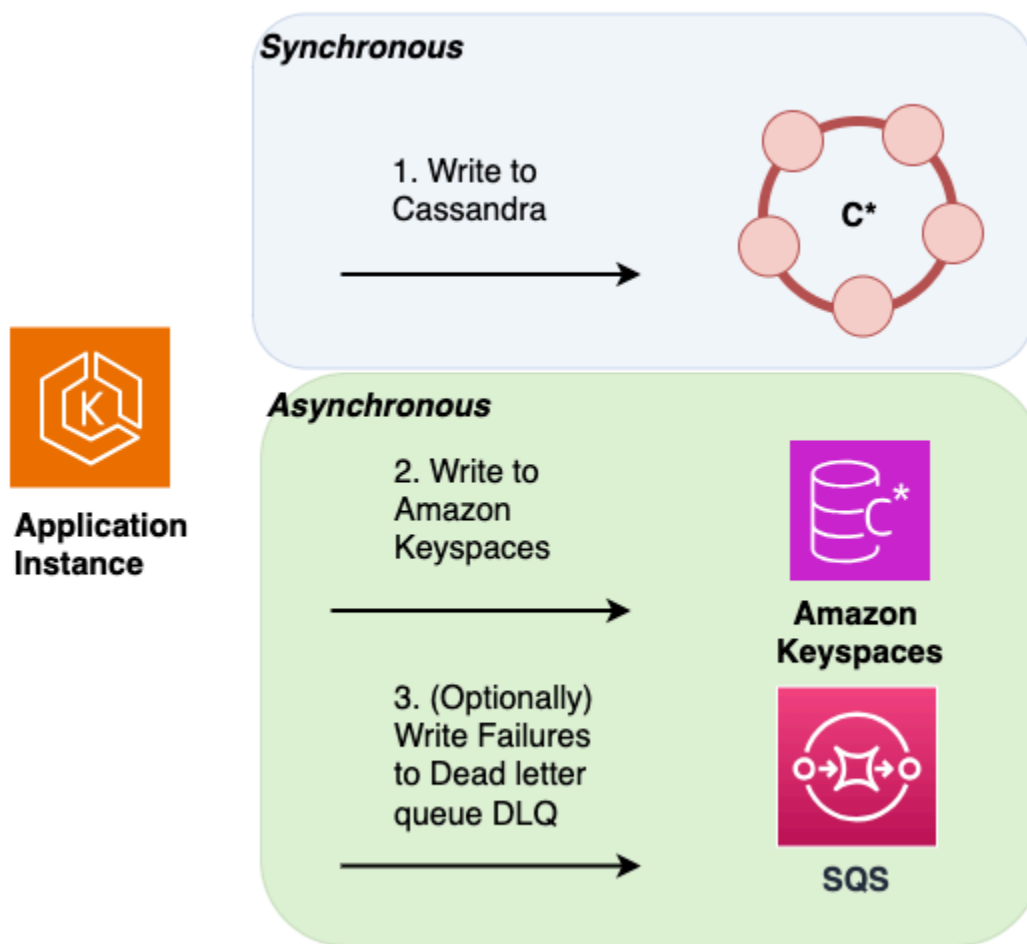
Lorsque vous implémentez des écritures doubles, vous pouvez désigner une base de données comme leader et l'autre comme suiveuse. Cela vous permet de continuer à écrire dans votre

base de données source ou de référence d'origine sans que les échecs d'écriture dans la base de données suiveuse ou de destination ne perturbent le chemin critique de votre application.

Au lieu de réessayer les écritures échouées dans le suiveur, vous pouvez utiliser Amazon Simple Queue Service pour enregistrer les écritures échouées dans une file d'attente de lettres [mortes \(DLQ\)](#). Le DLQ vous permet d'analyser les échecs d'écriture dans le suiveur et de déterminer pourquoi le traitement n'a pas abouti dans la base de données de destination.

Pour une implémentation plus sophistiquée de la double écriture, vous pouvez suivre les AWS meilleures pratiques pour concevoir une séquence de transactions locales à l'aide du modèle [saga](#). Un modèle de saga garantit qu'en cas d'échec d'une transaction, la saga exécute des transactions compensatoires pour annuler les modifications apportées à la base de données par les transactions précédentes.

Lorsque vous utilisez des écritures doubles pour une migration en ligne, vous pouvez configurer les écritures doubles en suivant le modèle de saga afin que chaque écriture soit une transaction locale afin de garantir des opérations atomiques sur des bases de données hétérogènes. Pour plus d'informations sur la conception d'applications distribuées à l'aide des modèles de conception recommandés pour le AWS Cloud, consultez [Modèles de conception, architectures et implémentations du cloud](#).



- Écritures doubles au niveau de la messagerie : au lieu d'implémenter des écritures doubles au niveau de la couche application, vous pouvez utiliser votre niveau de messagerie existant pour effectuer des écritures doubles sur Cassandra et Amazon Keyspaces.

Pour ce faire, vous pouvez configurer un utilisateur supplémentaire pour votre plateforme de messagerie afin qu'il envoie des écritures aux deux magasins de données. Cette approche fournit une stratégie simple à faible code utilisant le niveau de messagerie pour créer deux vues cohérentes dans les deux bases de données.

Téléchargement de données historiques lors d'une migration en ligne

Après avoir implémenté deux écritures pour garantir que les nouvelles données sont écrites dans les deux magasins de données en temps réel, l'étape suivante du plan de migration consiste à évaluer la quantité de données historiques que vous devez copier ou charger en masse depuis Cassandra vers Amazon Keyspaces. Cela garantit que les nouvelles données et les données historiques seront disponibles dans la nouvelle base de données Amazon Keyspaces avant la migration de l'application.

En fonction de vos exigences en matière de conservation des données, par exemple la quantité de données historiques que vous devez conserver en fonction des politiques de votre organisation, vous pouvez envisager l'une des deux options suivantes.

- **Téléchargement groupé de données historiques** — La migration des données historiques de votre déploiement Cassandra existant vers Amazon Keyspaces peut être réalisée à l'aide de différentes techniques, par exemple à l'aide AWS Glue de scripts personnalisés pour extraire, transformer et charger (ETL) les données. Pour plus d'informations sur l'utilisation AWS Glue pour charger des données historiques, consultez [the section called “Migration hors ligne”](#).

Lorsque vous planifiez le téléchargement groupé de données historiques, vous devez réfléchir à la manière de résoudre les conflits qui peuvent survenir lorsque de nouvelles écritures tentent de mettre à jour les mêmes données en cours de téléchargement. Le téléchargement en masse devrait être finalement cohérent, ce qui signifie que les données atteindront à terme tous les nœuds.

Si une mise à jour des mêmes données se produit en même temps en raison d'une nouvelle écriture, vous devez vous assurer qu'elles ne seront pas remplacées par le téléchargement des données historiques. Pour vous assurer de conserver les dernières mises à jour de vos données même pendant l'importation en masse, vous devez ajouter la résolution des conflits soit dans les scripts de téléchargement en bloc, soit dans la logique de l'application pour les écritures doubles.

Par exemple, vous pouvez utiliser [the section called “Transactions légères”](#) (LWT) pour comparer et définir des opérations. Pour ce faire, vous pouvez ajouter un champ supplémentaire à votre modèle de données qui représente l'heure de la modification ou l'état.

Amazon Keyspaces prend également en charge la fonction d'WRITETIMEhorodatage Cassandra. Vous pouvez utiliser les horodatages côté client d'Amazon Keyspaces pour préserver l'horodatage de la base de données source et implémenter la résolution des conflits du dernier auteur gagnant. Pour de plus amples informations, veuillez consulter [the section called “Client-side horodatages”](#).

- **Utilisation Time-to-Live (TTL)** : pour les périodes de conservation des données inférieures à 30, 60 ou 90 jours, vous pouvez utiliser le TTL dans Cassandra et Amazon Keyspaces pendant la migration afin d'éviter de télécharger des données historiques inutiles sur Amazon Keyspaces. TTL vous permet de définir une période au terme de laquelle les données sont automatiquement supprimées de la base de données.

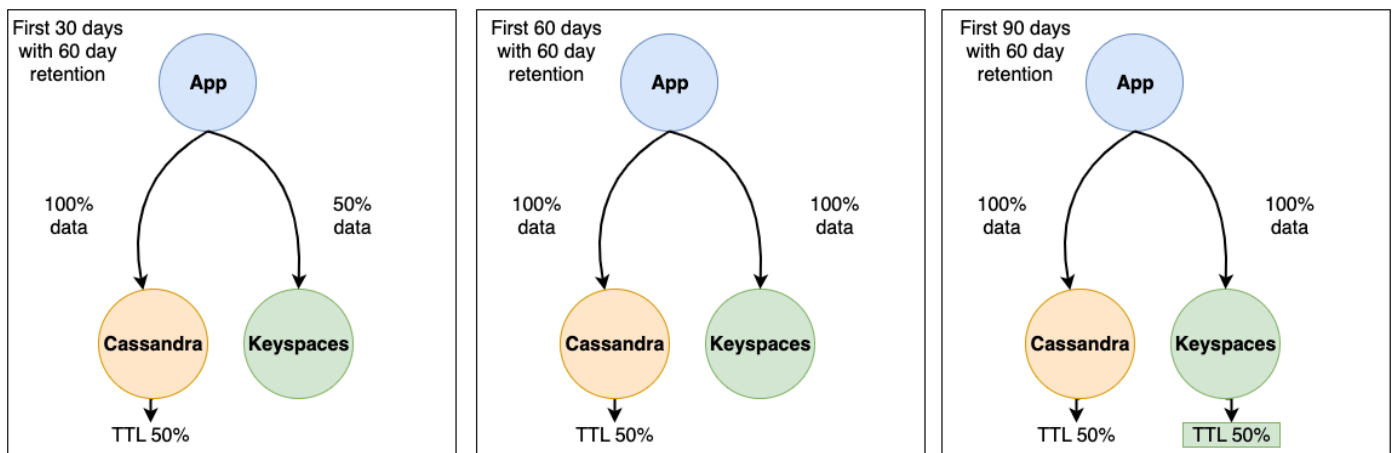
Au cours de la phase de migration, au lieu de copier les données historiques vers Amazon Keyspaces, vous pouvez configurer les paramètres TTL pour laisser les données historiques

expirer automatiquement dans l'ancien système (Cassandra) tout en appliquant uniquement les nouvelles écritures à Amazon Keyspaces à l'aide de la méthode de double écriture. Au fil du temps, les anciennes données expirant continuellement dans le cluster Cassandra et les nouvelles données étant écrites à l'aide de la méthode de double écriture, Amazon Keyspaces les rattrape automatiquement pour contenir les mêmes données que Cassandra.

Cette approche permet de réduire de manière significative la quantité de données à migrer, ce qui se traduit par un processus de migration plus efficace et rationalisé. Vous pouvez envisager cette approche lorsque vous traitez de grands ensembles de données dont les exigences de conservation des données varient. Pour plus d'informations sur TTL, consultez [the section called "Expirer les données avec Time to Live"](#).

Prenons l'exemple suivant d'une migration de Cassandra vers Amazon Keyspaces utilisant l'expiration des données TTL. Dans cet exemple, nous définissons le TTL pour les deux bases de données à 60 jours et montrons comment le processus de migration progresse sur une période de 90 jours. Les deux bases de données reçoivent les mêmes données nouvellement écrites au cours de cette période en utilisant la méthode des écritures doubles. Nous allons examiner trois phases différentes de la migration, chaque phase dure 30 jours.

Le fonctionnement du processus de migration pour chaque phase est illustré dans les images suivantes.



1. Après les 30 premiers jours, le cluster Cassandra et Amazon Keyspaces ont reçu de nouvelles écritures. Le cluster Cassandra contient également des données historiques dont la durée de conservation n'a pas encore atteint 60 jours, soit 50 % des données du cluster.

Les données datant de plus de 60 jours sont automatiquement supprimées dans le cluster Cassandra à l'aide du TTL. À ce stade, Amazon Keyspaces contient 50 % des données

stockées dans le cluster Cassandra, qui comprend les nouvelles écritures moins les données historiques.

2. Après 60 jours, le cluster Cassandra et Amazon Keyspaces contiennent les mêmes données écrites au cours des 60 derniers jours.
3. Dans les 90 jours, Cassandra et Amazon Keyspaces contiennent les mêmes données et expirent les données au même rythme.

Cet exemple montre comment éviter de télécharger des données historiques en utilisant le TTL dont la date d'expiration est fixée à 60 jours.

Validation de la cohérence des données lors d'une migration en ligne

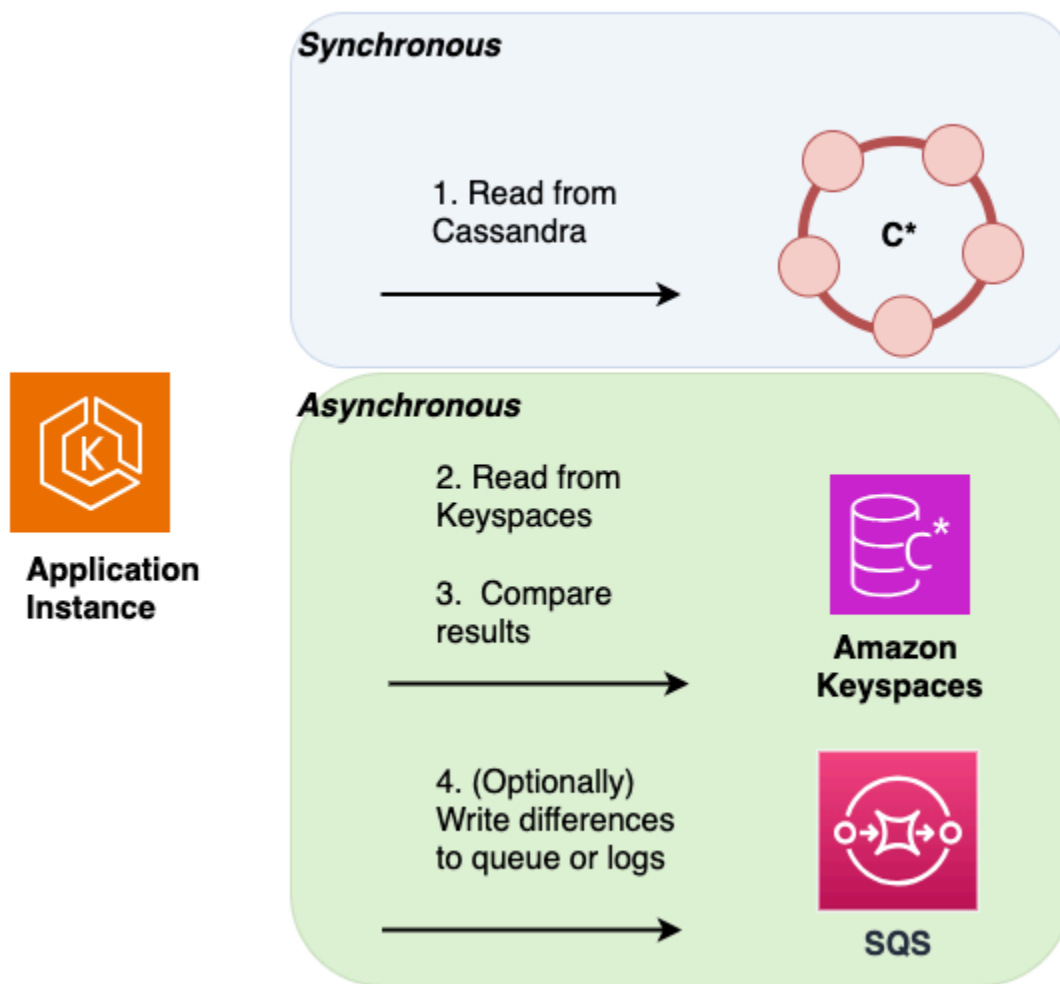
La prochaine étape du processus de migration en ligne est la validation des données. Les écritures doubles ajoutent de nouvelles données à votre base de données Amazon Keyspaces et vous avez terminé la migration des données historiques soit par téléchargement groupé, soit par expiration des données avec TTL.

Vous pouvez maintenant utiliser la phase de validation pour confirmer que les deux magasins de données contiennent les mêmes données et renvoient les mêmes résultats de lecture. Vous pouvez choisir l'une des deux options suivantes pour vérifier que vos deux bases de données contiennent des données identiques.

- **Lecture double** : pour vérifier que la base de données source et la base de données de destination contiennent le même ensemble de données récemment écrites et historiques, vous pouvez implémenter des lectures doubles. Pour ce faire, vous pouvez lire à la fois votre base de données Cassandra principale et votre base de données Amazon Keyspaces secondaire de la même manière que la méthode d'écriture double et vous comparez les résultats de manière asynchrone.

Les résultats de la base de données principale sont renvoyés au client et les résultats de la base de données secondaire sont utilisés pour valider par rapport à l'ensemble de résultats principal. Les différences détectées peuvent être enregistrées ou envoyées dans une file d'attente de lettres [mortes \(DLQ\)](#) pour un rapprochement ultérieur.

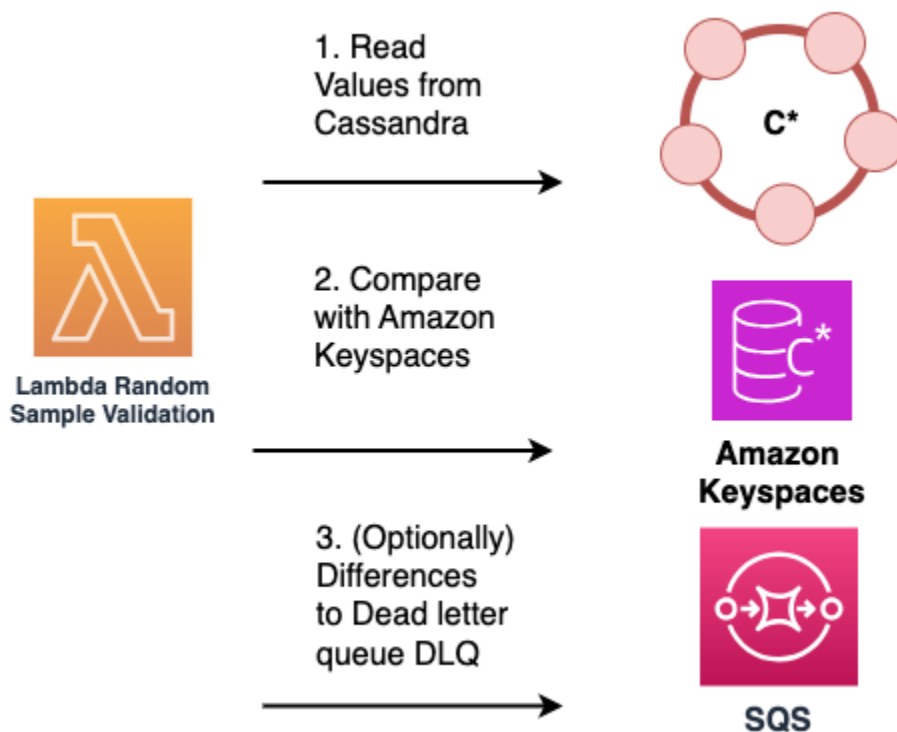
Dans le schéma suivant, l'application effectue une lecture synchrone depuis Cassandra, qui est le magasin de données principal) et une lecture asynchrone depuis Amazon Keyspaces, qui est le magasin de données secondaire.



- Exemples de lectures : une solution alternative ne nécessitant pas de modification du code de l'application consiste à utiliser des AWS Lambda fonctions pour échantillonner périodiquement et de manière aléatoire des données provenant à la fois du cluster Cassandra source et de la base de données Amazon Keyspaces de destination.

Ces fonctions Lambda peuvent être configurées pour être exécutées à intervalles réguliers. La fonction Lambda extrait un sous-ensemble aléatoire de données provenant des systèmes source et de destination, puis effectue une comparaison des données échantillonnées. Toute divergence ou incohérence entre les deux ensembles de données peut être enregistrée et envoyée à une file d'attente de lettres [mortes \(DLQ\)](#) dédiée pour un rapprochement ultérieur.

Ce processus est illustré dans le diagramme suivant.



Migration de l'application lors d'une migration en ligne

Au cours de la quatrième phase d'une migration en ligne, vous migrez votre application et vous passez à Amazon Keyspaces en tant que magasin de données principal. Cela signifie que vous passez votre application à la lecture et à l'écriture directement depuis et vers Amazon Keyspaces. Afin de perturber le moins possible vos utilisateurs, ce processus doit être bien planifié et coordonné.

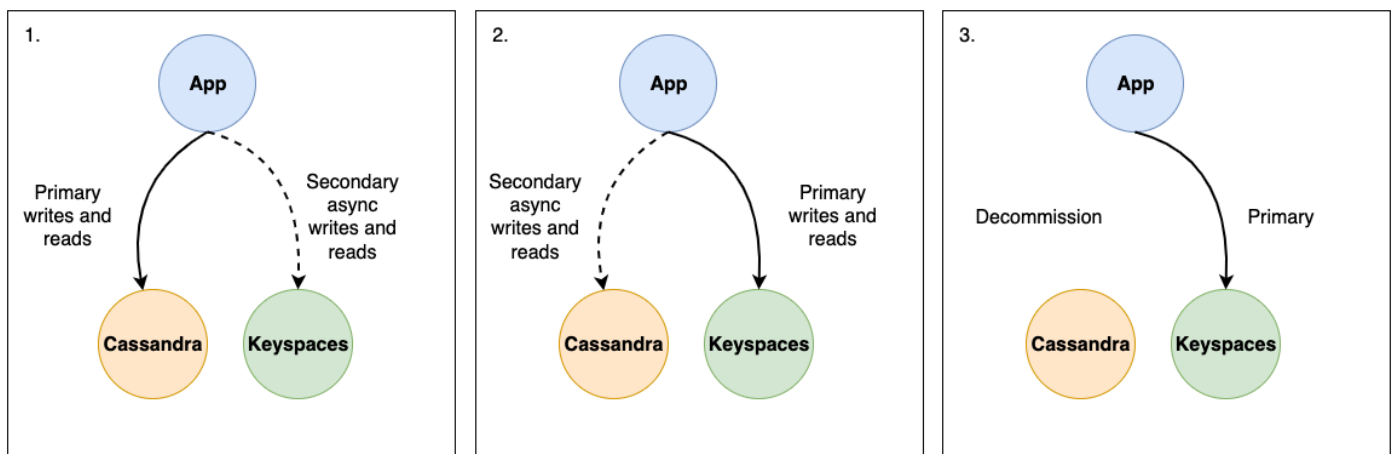
Deux solutions différentes recommandées pour la migration des applications sont disponibles, la stratégie de coupure bleu-vert et la stratégie de coupure canari. Les sections suivantes décrivent ces stratégies de manière plus détaillée.

- **Stratégie bleu-vert** — En utilisant cette approche, vous pouvez changer d'application pour traiter Amazon Keyspaces comme le magasin de données principal et Cassandra comme le magasin de données secondaire en une seule étape. Pour ce faire, utilisez un indicateur de AWS AppConfig fonctionnalité pour contrôler le choix des magasins de données principaux et secondaires sur l'instance de l'application. Pour plus d'informations sur les indicateurs de fonctionnalités, voir [Création d'un profil de configuration d'indicateur de fonctionnalité dans AWS AppConfig](#).

Après avoir fait d'Amazon Keyspaces le magasin de données principal, vous surveillez le comportement et les performances de l'application, afin de vous assurer qu'Amazon Keyspaces répond à vos exigences et que la migration est réussie.

Par exemple, si vous avez implémenté des lectures doubles pour votre application, pendant la phase de migration de l'application, vous transférez les lectures principales de Cassandra vers Amazon Keyspaces et les lectures secondaires d'Amazon Keyspaces vers Cassandra. Après la transition, vous continuez à surveiller et à comparer les résultats comme décrit dans la [section sur la validation](#) des données afin de garantir la cohérence entre les deux bases de données avant de mettre Cassandra hors service.

Si vous détectez des problèmes, vous pouvez rapidement revenir à l'état précédent en reprenant Cassandra comme magasin de données principal. Vous ne pouvez passer à la phase de mise hors service de la migration que si Amazon Keyspaces répond à tous vos besoins en tant que magasin de données principal.

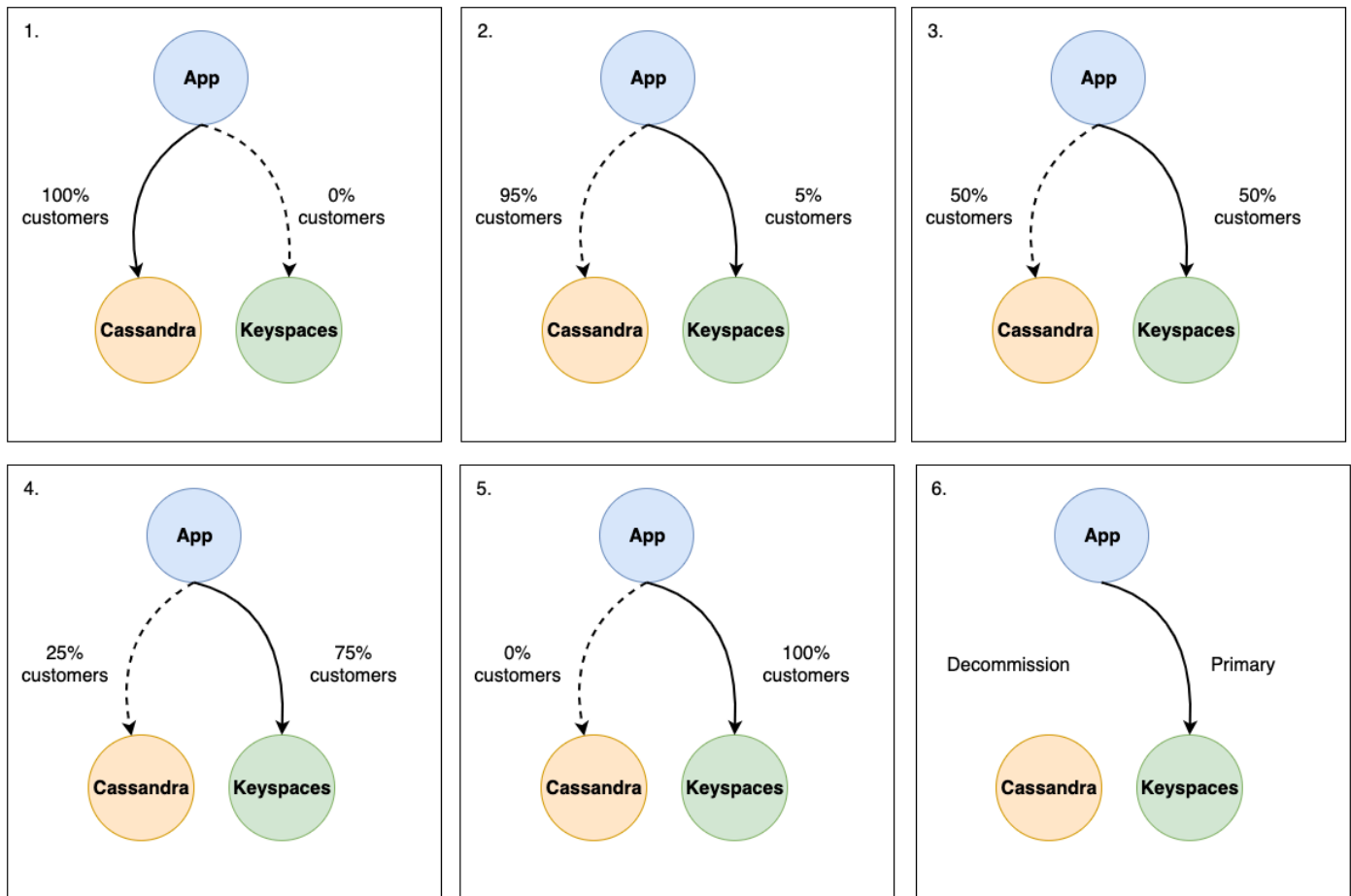


- **Stratégie Canary** — Dans cette approche, vous déployez progressivement la migration vers un sous-ensemble de vos utilisateurs ou de votre trafic. Au départ, un faible pourcentage du trafic de votre application, par exemple 5 % de l'ensemble du trafic, est acheminé vers la version utilisant Amazon Keyspaces comme magasin de données principal, tandis que le reste du trafic continue d'utiliser Cassandra comme magasin de données principal.

Cela vous permet de tester minutieusement la version migrée avec un trafic réel, de surveiller ses performances, sa stabilité et d'étudier les problèmes potentiels. Si vous ne détectez aucun problème, vous pouvez augmenter progressivement le pourcentage de trafic acheminé vers Amazon Keyspaces jusqu'à ce qu'il devienne le magasin de données principal pour tous les utilisateurs et le trafic.

Ce déploiement par étapes minimise le risque d'interruptions de service généralisées et permet un processus de migration plus contrôlé. Si des problèmes critiques surviennent lors du déploiement de Canary, vous pouvez rapidement revenir à la version précédente en utilisant Cassandra comme magasin de données principal pour le segment de trafic concerné. Vous ne pouvez passer à la phase de mise hors service de la migration qu'après avoir vérifié qu'Amazon Keyspaces traite 100 % de vos utilisateurs et de votre trafic comme prévu.

Le schéma suivant illustre les différentes étapes de la stratégie Canary.



Mise hors service de Cassandra après une migration en ligne

Une fois que la migration de l'application est terminée, que votre application est entièrement exécutée sur Amazon Keyspaces et que vous avez validé la cohérence des données sur une période donnée, vous pouvez planifier la mise hors service de votre cluster Cassandra. Au cours de cette phase, vous pouvez évaluer si les données restantes dans votre cluster Cassandra doivent être archivées ou

peuvent être supprimées. Cela dépend des politiques de votre organisation en matière de traitement et de conservation des données.

En suivant cette stratégie et en tenant compte des bonnes pratiques recommandées décrites dans cette rubrique lors de la planification de votre migration en ligne de Cassandra vers Amazon Keyspaces, vous pouvez garantir une transition fluide vers Amazon Keyspaces tout en préservant la cohérence lecture après écriture et la disponibilité de votre application.

La migration d'Apache Cassandra vers Amazon Keyspaces peut apporter de nombreux avantages, notamment une réduction des frais opérationnels, une mise à l'échelle automatique, une sécurité améliorée et un cadre qui vous aide à atteindre vos objectifs de conformité. En planifiant une stratégie de migration en ligne comprenant deux écritures, le téléchargement des données historiques, la validation des données et un déploiement progressif, vous pouvez garantir une transition fluide avec un minimum de perturbations pour votre application et ses utilisateurs.

La mise en œuvre de la stratégie de migration en ligne abordée dans cette rubrique vous permet de valider les résultats de la migration, d'identifier et de résoudre les problèmes éventuels, puis de mettre hors service votre déploiement Cassandra existant au profit du service Amazon Keyspaces entièrement géré.

Processus de migration hors ligne : Apache Cassandra vers Amazon Keyspaces

Les migrations hors ligne conviennent lorsque vous pouvez vous permettre une interruption de service pour effectuer la migration. Il est courant dans les entreprises de disposer de fenêtres de maintenance pour l'application de correctifs, les versions volumineuses ou les interruptions de service pour les mises à niveau matérielles ou majeures. La migration hors ligne peut utiliser cette fenêtre pour copier des données et transférer le trafic applicatif d'Apache Cassandra vers Amazon Keyspaces.

La migration hors ligne réduit les modifications apportées à l'application car elle ne nécessite pas de communication simultanée avec Cassandra et Amazon Keyspaces. En outre, lorsque le flux de données est suspendu, l'état exact peut être copié sans conserver les mutations.

Dans cet exemple, nous utilisons Amazon Simple Storage Service (Amazon S3) comme zone intermédiaire pour les données lors de la migration hors ligne afin de minimiser les temps d'arrêt. Vous pouvez importer automatiquement les données que vous avez stockées au format Parquet dans Amazon S3 dans un tableau Amazon Keyspaces à l'aide du connecteur Spark Cassandra et.

AWS Glue La section suivante va présenter une vue d'ensemble de haut niveau du processus. Vous pouvez trouver des exemples de code pour ce processus sur [Github](#).

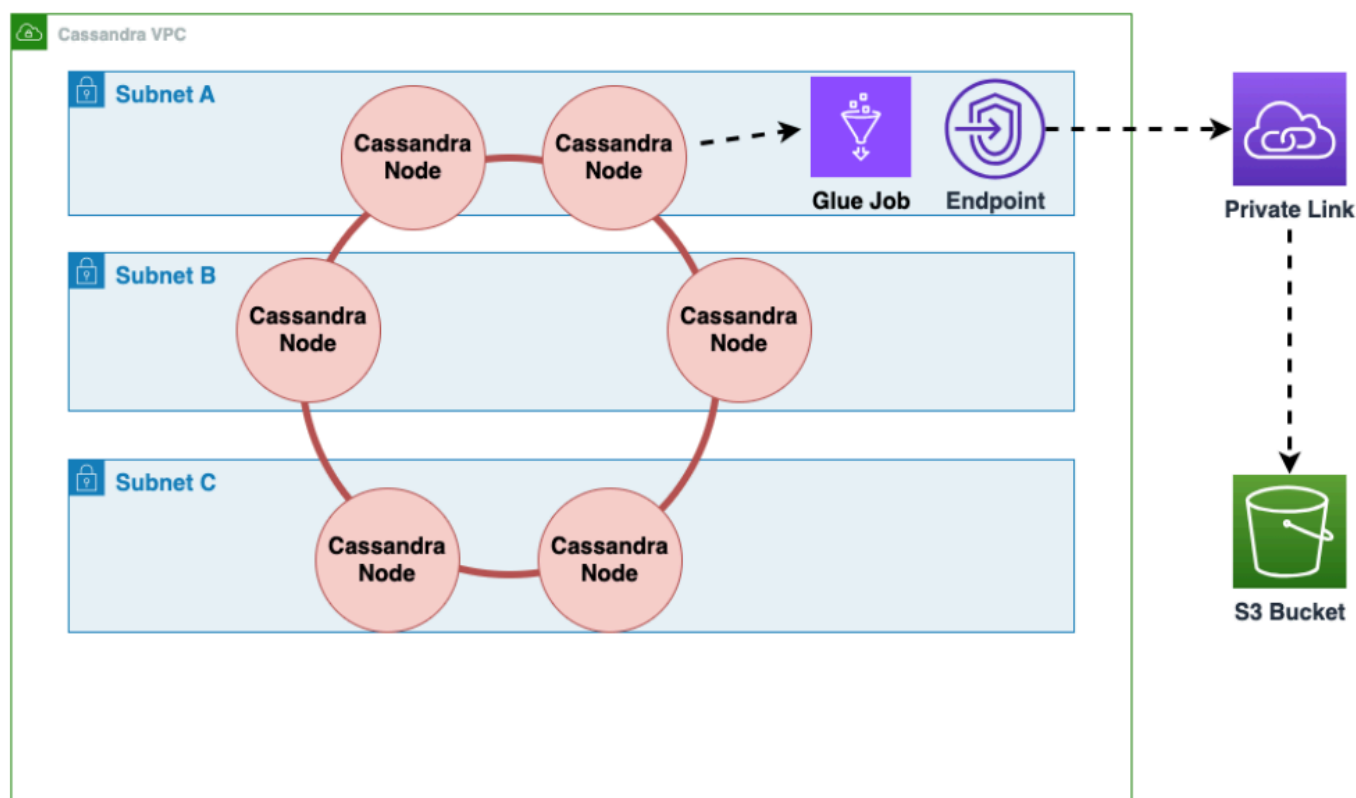
Le processus de migration hors ligne d'Apache Cassandra vers Amazon Keyspaces à l'aide d'Amazon S3 AWS Glue nécessite les tâches suivantes AWS Glue .

1. Une tâche ETL qui extrait et transforme les données CQL et les stocke dans un compartiment Amazon S3.
2. Une deuxième tâche qui importe les données du compartiment vers Amazon Keyspaces.
3. Une troisième tâche pour importer des données incrémentielles.

Comment effectuer une migration hors ligne vers Amazon Keyspaces depuis Cassandra exécutée sur Amazon EC2 dans un cloud privé virtuel Amazon

1. Vous pouvez d'abord AWS Glue exporter les données de table de Cassandra au format Parquet et les enregistrer dans un compartiment Amazon S3. Vous devez exécuter une AWS Glue tâche à l'aide d'un AWS Glue connecteur vers un VPC où réside l'instance Amazon EC2 exécutant Cassandra. Ensuite, à l'aide du point de terminaison privé Amazon S3, vous pouvez enregistrer des données dans le compartiment Amazon S3.

Le schéma suivant illustre ces étapes.



2. Mélangez les données du compartiment Amazon S3 pour améliorer la randomisation des données. Les données importées de manière uniforme permettent de répartir davantage le trafic dans la table cible.

Cette étape est requise lors de l'exportation de données depuis Cassandra avec de grandes partitions (partitions de plus de 1 000 lignes) afin d'éviter les modèles de touches de raccourci lors de l'insertion des données dans Amazon Keyspaces. Les problèmes liés aux touches de raccourci `WriteThrottleEvents` se produisent dans Amazon Keyspaces et entraînent une augmentation du temps de chargement.



3. Utilisez une autre AWS Glue tâche pour importer des données depuis le compartiment Amazon S3 vers Amazon Keyspaces. Les données mélangées du compartiment Amazon S3 sont stockées au format Parquet.



Pour plus d'informations sur le processus de migration hors ligne, consultez l'atelier [Amazon Keyspaces avec AWS Glue](#)

Utilisation d'une solution de migration hybride : Apache Cassandra vers Amazon Keyspaces

La solution de migration suivante peut être considérée comme un hybride entre la migration en ligne et hors ligne. Grâce à cette approche hybride, les données sont écrites dans la base de données de destination quasiment en temps réel sans assurer la cohérence entre lecture et écriture. Cela signifie que les données nouvellement écrites ne seront pas immédiatement disponibles et des retards sont à prévoir. Si vous avez besoin de cohérence lecture après écriture, consultez [the section called "Migration en ligne"](#).

Pour effectuer une migration en temps quasi réel d'Apache Cassandra vers Amazon Keyspaces, vous pouvez choisir entre deux méthodes disponibles.

- **CQLReplicator** — (Recommandé) CQLReplicator est un utilitaire open source disponible sur [Github](#) qui vous permet de migrer des données d'Apache Cassandra vers Amazon Keyspaces en temps quasi réel.

Pour déterminer les écritures et les mises à jour à propager vers la base de données de destination, CQLReplicator analyse la plage de jetons Apache Cassandra et utilise une AWS Glue tâche pour supprimer les événements dupliqués et appliquer les écritures et les mises à jour directement à Amazon Keyspaces.

- **Capture des données de modification (CDC)** — Si vous connaissez Cassandra CDC, la fonction CDC intégrée d'Apache Cassandra qui permet de capturer les modifications en copiant le journal de validation dans un répertoire CDC distinct est une autre option pour mettre en œuvre une migration hybride.

Vous pouvez le faire en répliquant les modifications de données dans Amazon Keyspaces, faisant du CDC une option alternative pour les scénarios de migration de données.

Si vous n'avez pas besoin de cohérence entre lecture et écriture, vous pouvez utiliser le CQLReplicator ou un pipeline CDC pour migrer les données d'Apache Cassandra vers Amazon Keyspaces en fonction de vos préférences et de votre familiarité avec les outils utilisés dans chaque solution. Services AWS L'utilisation de ces méthodes pour migrer des données en temps quasi réel peut être considérée comme une approche hybride de la migration offrant une alternative à la migration en ligne.

Cette stratégie est considérée comme une approche hybride, car outre les options décrites dans cette rubrique, vous devez mettre en œuvre certaines étapes de la progression de la migration en ligne, par exemple la copie des données historiques et les stratégies de migration des applications abordées dans la [rubrique sur la migration](#) en ligne.

Les sections suivantes examinent plus en détail les options de migration hybride.

Rubriques

- [Migrer des données à l'aide de CQLReplicator](#)
- [Migrer les données à l'aide de la capture des données de modification \(CDC\)](#)

Migrer des données à l'aide de CQLReplicator

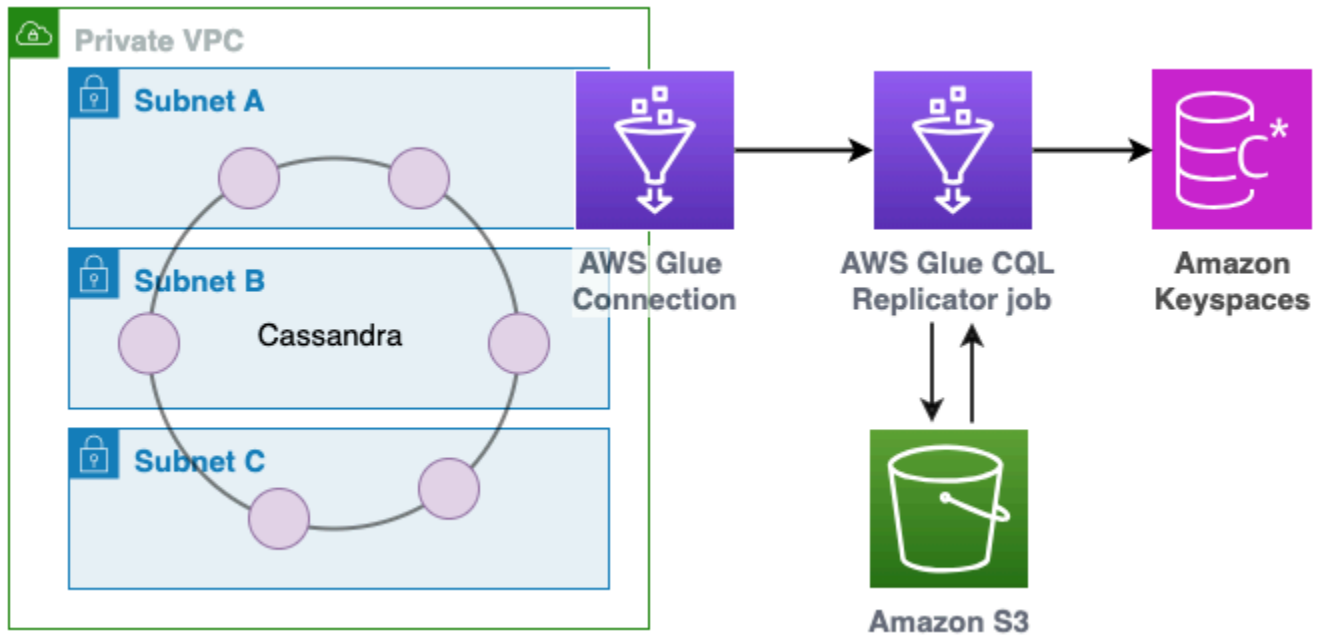
Avec [CQLReplicator](#), vous pouvez lire les données d'Apache Cassandra en temps quasi réel en scannant intelligemment l'anneau de jetons Cassandra à l'aide de requêtes CQL. CQLReplicator n'utilise pas Cassandra CDC et implémente à la place une stratégie de mise en cache pour réduire les pénalités de performances liées aux analyses complètes.

Pour réduire le nombre d'écritures vers la destination, CQLReplicator supprime automatiquement les événements de répliqués dupliqués. Avec CQLReplicator, vous pouvez ajuster la répliqués des modifications de la base de données source vers la base de données de destination, ce qui permet une migration en temps quasi réel des données d'Apache Cassandra vers Amazon Keyspaces.

Le schéma suivant montre l'architecture typique d'une tâche CQLReplicator utilisant. AWS Glue

1. Pour autoriser l'accès à Apache Cassandra s'exécutant dans un VPC privé, configurez une AWS Glue connexion avec le type de connexion Réseau.

2. Pour supprimer les doublons et activer la mise en cache des clés avec la tâche CQLReplicator, configurez Amazon Simple Storage Service (Amazon S3).
3. La base de données source vérifiée des flux de tâches CQLReplicator est modifiée directement vers Amazon Keyspaces.



Pour plus d'informations sur le processus de migration à l'aide de CQLReplicator, consultez l'article suivant sur le blog de AWS base de données [Migrer les charges de travail Cassandra vers Amazon Keyspaces à l'aide de CQLReplicator](#) et les instructions AWS prescriptives [Migrer les charges de travail Apache Cassandra vers Amazon Keyspaces en utilisant AWS Glue](#)

Migrer les données à l'aide de la capture des données de modification (CDC)

Si vous êtes déjà familiarisé avec la configuration d'un pipeline de capture des données de modification (CDC) avec [Debezium](#), vous pouvez utiliser cette option pour migrer des données vers Amazon Keyspaces au lieu d'utiliser CQLReplicator. Debezium est une plate-forme distribuée open source pour le CDC, conçue pour surveiller une base de données et capturer les modifications au niveau des lignes de manière fiable.

Le connecteur [Debezium pour Apache Cassandra](#) télécharge les modifications vers Amazon Managed Streaming pour Apache Kafka (Amazon MSK) afin qu'elles puissent être consommées et traitées par les consommateurs en aval qui, à leur tour, écrivent les données dans Amazon

Keyspaces. Pour plus d'informations, consultez les [conseils pour la migration continue des données d'Apache Cassandra vers Amazon Keyspaces](#).

Pour résoudre tout problème potentiel de cohérence des données, vous pouvez implémenter un processus avec Amazon MSK dans le cadre duquel un consommateur compare les clés ou les partitions de Cassandra à celles d'Amazon Keyspaces.

Pour mettre en œuvre cette solution avec succès, nous vous recommandons de prendre en compte les points suivants.

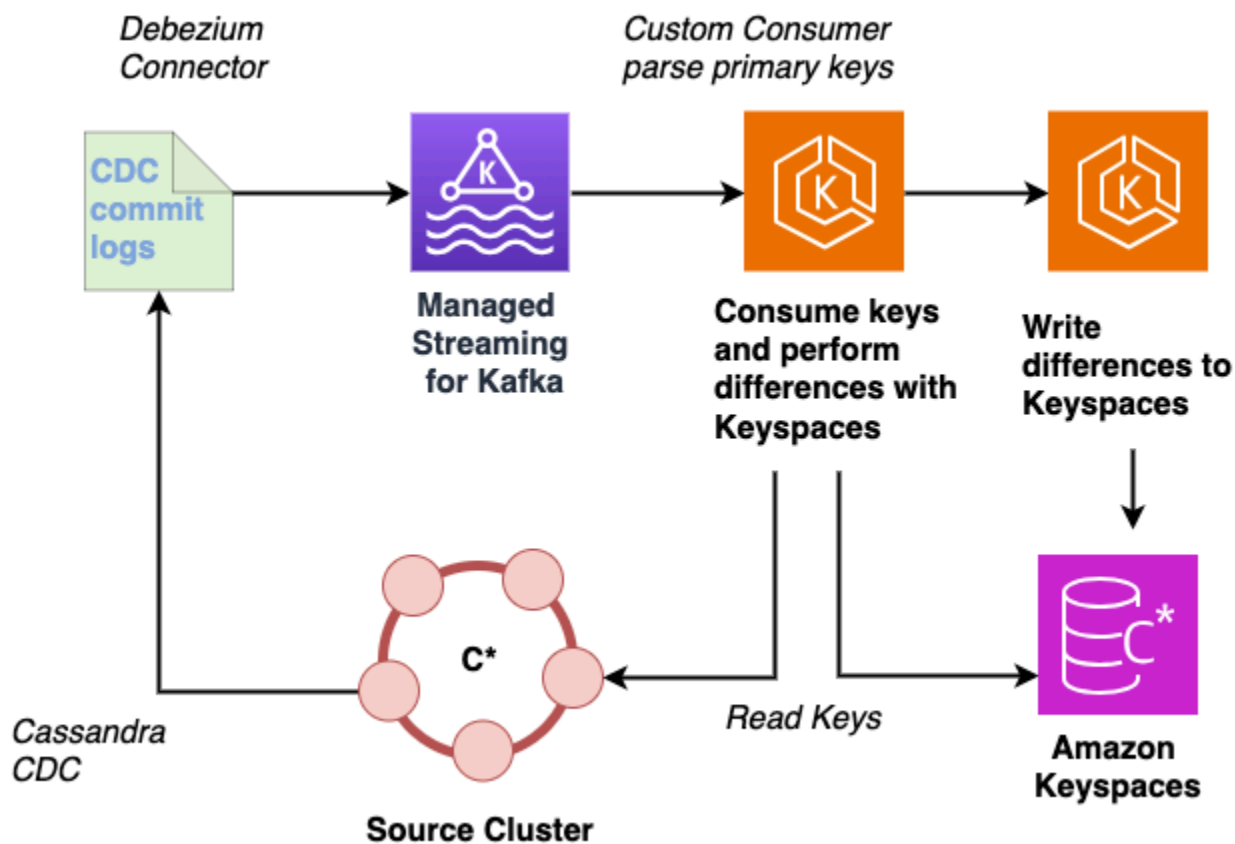
- Comment analyser le journal des validations du CDC, par exemple comment supprimer les événements dupliqués.
- Comment gérer le répertoire CDC, par exemple comment supprimer les anciens journaux.
- Comment gérer les échecs partiels dans Apache Cassandra, par exemple si une écriture ne réussit que dans une des trois répliques.
- Comment gérer l'allocation des ressources, par exemple en augmentant la taille de l'instance pour tenir compte des exigences supplémentaires en matière de processeur, de mémoire, de DISQUE et d'E/S pour le processus CDC qui se produit sur un nœud.

Ce modèle traite les modifications apportées par Cassandra comme un « indice » indiquant qu'une touche peut avoir changé par rapport à son état précédent. Pour déterminer si des modifications doivent être propagées à la base de données de destination, vous devez d'abord lire depuis le cluster Cassandra source en effectuant une LOCAL_QUORUM opération pour recevoir les derniers enregistrements, puis les écrire dans Amazon Keyspaces.

Dans le cas de suppressions ou de mises à jour de plages, vous devrez peut-être effectuer une comparaison avec l'ensemble de la partition pour déterminer quels événements d'écriture ou de mise à jour doivent être écrits dans votre base de données de destination.

Dans les cas où les écritures ne sont pas idempotentes, vous devez également comparer vos écritures avec ce qui se trouve déjà dans la base de données de destination avant d'écrire sur Amazon Keyspaces.

Le schéma suivant montre l'architecture typique d'un pipeline CDC utilisant Debezium et Amazon MSK.



Comment sélectionner le bon outil pour le téléchargement groupé ou la migration de données vers Amazon Keyspaces

Dans cette section, vous pouvez passer en revue les différents outils que vous pouvez utiliser pour télécharger ou migrer des données en masse vers Amazon Keyspaces, et apprendre à sélectionner l'outil approprié en fonction de vos besoins. En outre, cette section fournit une vue d'ensemble et des cas d'utilisation des didacticiels étape par étape disponibles qui montrent comment importer des données dans Amazon Keyspaces.

Pour passer en revue les stratégies disponibles pour migrer les charges de travail d'Apache Cassandra vers Amazon Keyspaces, consultez. [the section called “Migration depuis Cassandra”](#)

- Outils de migration
 - Grâce au [calculateur de prix pour Amazon Keyspaces \(pour Apache Cassandra\)](#) disponible sur Github, vous pouvez estimer vos coûts mensuels pour Amazon Keyspaces en fonction de votre charge de travail Apache Cassandra existante. Entrez les statistiques issues de la sortie de statut de votre outil de nœud Cassandra et de la configuration sans serveur prévue pour Amazon

Keyspaces afin de comparer les coûts directs entre les deux solutions. Notez que ce calculateur se concentre uniquement sur les coûts opérationnels d'Amazon Keyspaces par rapport à votre déploiement Cassandra existant. Il n'inclut pas les facteurs liés au coût total de possession (TCO) tels que la maintenance de l'infrastructure, les frais d'exploitation ou les coûts de support pour Cassandra.

- Proxy ZDM Dual Write pour la migration vers Amazon Keyspaces — Le proxy ZDM Dual Write disponible [sur](#) Github prend en charge la migration sans interruption d'Apache Cassandra vers Amazon Keyspaces.
- CQLReplicator — CQLReplicator est un utilitaire open source disponible sur [Github](#) qui vous aide à migrer des données d'Apache Cassandra vers Amazon Keyspaces en temps quasi réel.

Pour de plus amples informations, veuillez consulter [the section called “Réplicateur SQL”](#).

- Pour en savoir plus sur l'utilisation d'Amazon Managed Streaming pour Apache Kafka afin de mettre en œuvre un processus de [migration en ligne](#) avec des écritures doubles, consultez les [instructions relatives à la migration continue des données d'Apache Cassandra vers Amazon Keyspaces](#).
- Pour les migrations de grande envergure, pensez à utiliser un outil d'extraction, de transformation et de chargement (ETL). Vous pouvez l'utiliser AWS Glue pour effectuer rapidement et efficacement des migrations de transformation de données. Pour de plus amples informations, veuillez consulter [the section called “Migration hors ligne”](#).
- Pour savoir comment utiliser le connecteur Apache Cassandra Spark pour écrire des données sur Amazon Keyspaces, consultez [the section called “Intégration à Apache Spark”](#).
- Commencez rapidement à charger des données dans Amazon Keyspaces à l'aide de la COPY FROM commande cqlsh. cqlsh est inclus dans Apache Cassandra et convient parfaitement au chargement de petits ensembles de données ou de données de test. Pour obtenir des instructions détaillées, consultez [the section called “Chargement de données à l'aide de cqlsh”](#).
- Vous pouvez également utiliser le DataStax Bulk Loader pour Apache Cassandra pour charger des données dans Amazon Keyspaces à l'aide dsbulk de la commande. [DSBulk fournit des fonctionnalités d'importation plus robustes que cqlsh et est disponible depuis le référentiel. GitHub](#) Pour obtenir des instructions détaillées, consultez [the section called “Chargement de données à l'aide de DSBulk”](#).

Considérations générales relatives aux téléchargements de données vers Amazon Keyspaces

- Divisez le téléchargement des données en composants plus petits.

Tenez compte des unités de migration suivantes et de leur empreinte potentielle en termes de taille des données brutes. Le téléchargement de petites quantités de données en une ou plusieurs phases peut contribuer à simplifier votre migration.

- Par cluster : migrez toutes vos données Cassandra en une seule fois. Cette approche peut convenir aux petits clusters.
- Par espace de touches ou par table : divisez votre migration en groupes d'espaces de touches ou de tables. Cette approche peut vous aider à migrer les données par étapes en fonction de vos besoins pour chaque charge de travail.
- Par données — Envisagez de migrer les données pour un groupe spécifique d'utilisateurs ou de produits, afin de réduire encore davantage la taille des données.
- Priorisez les données à télécharger en premier en fonction de la simplicité.

Déterminez si vous avez des données qui pourraient d'abord être migrées plus facilement, par exemple des données qui ne changent pas à des moments précis, des données provenant de traitements par lots effectués la nuit, des données non utilisées pendant les heures hors ligne ou des données provenant d'applications internes.

Rubriques

- [Tutoriel : Chargement de données dans Amazon Keyspaces à l'aide de cqlsh](#)
- [Tutoriel : Chargement de données dans Amazon Keyspaces à l'aide de DSBulk](#)

Tutoriel : Chargement de données dans Amazon Keyspaces à l'aide de cqlsh

Ce didacticiel vous guide tout au long du processus de migration des données d'Apache Cassandra vers Amazon Keyspaces à l'aide de la commande `cqlsh COPY FROM`. La `cqlsh COPY FROM` commande est utile pour télécharger rapidement et facilement de petits ensembles de données sur Amazon Keyspaces à des fins académiques ou de test. Pour plus d'informations sur la migration des charges de travail de production, consultez [the section called “Migration hors ligne”](#). Dans ce didacticiel, vous allez effectuer les étapes suivantes :

Conditions préalables : configurez un AWS compte avec des informations d'identification, créez un fichier JKS Trust Store pour le certificat et configurez-le pour vous connecter `cqlsh` à Amazon Keyspaces.

1. Créer un fichier CSV source et une table cible : préparez un fichier CSV en tant que données source et créez le keyspace et le tableau cibles dans Amazon Keyspaces.
2. Préparation des données : répartissez les données du fichier CSV de manière aléatoire et analysez-les pour déterminer les tailles de ligne moyennes et maximales.
3. Définissez la capacité de débit : calculez les unités de capacité d'écriture (WCU) requises en fonction de la taille des données et du temps de chargement souhaité, puis configurez la capacité provisionnée de la table.
4. Configurer les paramètres `cqlsh` : déterminez les valeurs optimales pour des `cqlsh COPY FROM` paramètres tels que `INGESTRATE`, `NUMPROCESSES`, `MAXBATCHSIZE`, et pour répartir la charge `CHUNKSIZE` de travail de manière uniforme.
5. Exécuter la **`cqlsh COPY FROM`** commande : exécutez la `cqlsh COPY FROM` commande pour télécharger les données du fichier CSV vers le tableau Amazon Keyspaces et suivre la progression.

Dépannage — Résolvez les problèmes courants tels que les demandes non valides, les erreurs d'analyseur, les erreurs de capacité et les erreurs `cqlsh` lors du processus de téléchargement des données.

Rubriques

- [Conditions préalables : étapes à suivre avant de pouvoir télécharger des données à l'aide de `cqlsh COPY FROM`](#)
- [Étape 1 : Création du fichier CSV source et d'une table cible pour le téléchargement des données](#)
- [Étape 2 : Préparation des données sources pour un téléchargement de données réussi](#)
- [Étape 3 : définir la capacité de débit de la table](#)
- [Étape 4 : Configuration des paramètres `cqlsh COPY FROM`](#)
- [Étape 5 : Exécutez la commande `cqlsh COPY FROM` pour télécharger les données du fichier CSV vers la table cible](#)
- [Résolution des problèmes](#)

Conditions préalables : étapes à suivre avant de pouvoir télécharger des données à l'aide de **`cqlsh COPY FROM`**

Vous devez effectuer les tâches suivantes avant de pouvoir commencer ce didacticiel.

1. Si ce n'est pas déjà fait, inscrivez-vous Compte AWS en suivant les étapes indiquées sur [the section called “Configuration d'Amazon Keyspaces”](#).
2. Créez des informations d'identification spécifiques au service en suivant les étapes décrites dans [the section called “Créez des informations d'identification spécifiques au service”](#)
3. Configurez la connexion au shell Cassandra Query Language (cqlsh) et confirmez que vous pouvez vous connecter à Amazon Keyspaces en suivant les étapes indiquées sur [the section called “Utilisation de cqlsh”](#)

Étape 1 : Création du fichier CSV source et d'une table cible pour le téléchargement des données

Pour ce didacticiel, nous utilisons un fichier de valeurs séparées par des virgules (CSV) dont le `keyspaces_sample_table.csv` nom est le fichier source pour la migration des données. Le fichier d'exemple fourni contient quelques lignes de données pour une table portant le nom `book_awards`.

1. Créez le fichier source. Vous pouvez choisir l'une des options suivantes :
 - Téléchargez l'exemple de fichier CSV (`keyspaces_sample_table.csv`) contenu dans le fichier d'archive [samplermigration.zip](#) suivant. Décompressez l'archive et notez le chemin vers `keyspaces_sample_table.csv`.
 - Pour remplir un fichier CSV avec vos propres données stockées dans une base de données Apache Cassandra, vous pouvez remplir le fichier CSV source à l'aide de l'`cqlshCOPY` instruction illustrée dans l'exemple suivant.

```
cqlsh localhost 9042 -u "username" -p "password" --execute  
"COPY mykeyspace.mytable TO 'keyspaces_sample_table.csv' WITH HEADER=true";
```

Assurez-vous que le fichier CSV que vous créez répond aux exigences suivantes :

- La première ligne contient les noms des colonnes.
- Les noms des colonnes du fichier CSV source correspondent aux noms des colonnes de la table cible.
- Les données sont délimitées par une virgule.
- Toutes les valeurs de données sont des types de données Amazon Keyspaces valides. Consultez [the section called “Types de données”](#).

2. Créez le keyspace et le tableau cibles dans Amazon Keyspaces.

- a. Connectez-vous à Amazon Keyspaces en utilisant `cqlsh`, en remplaçant le point de terminaison du service, le nom d'utilisateur et le mot de passe dans l'exemple suivant par vos propres valeurs.

```
cqlsh cassandra.us-east-1.amazonaws.com 9142 -u "111122223333" -p
"wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY" --ssl
```

- b. Créez un nouvel espace de touches avec le `catalog` nom indiqué dans l'exemple suivant.

```
CREATE KEYSPACE catalog WITH REPLICATION = {'class': 'SingleRegionStrategy'};
```

- c. Lorsque le nouvel espace de touches est disponible, utilisez le code suivant pour créer la table `book_awards` cible.

```
CREATE TABLE "catalog.book_awards" (  
    year int,  
    award text,  
    rank int,  
    category text,  
    book_title text,  
    author text,  
    publisher text,  
    PRIMARY KEY ((year, award), category, rank)  
);
```

Si Apache Cassandra est votre source de données d'origine, un moyen simple de créer la table cible Amazon Keyspaces avec les en-têtes correspondants consiste à générer `CREATE TABLE` l'instruction à partir de la table source, comme indiqué dans l'instruction suivante.

```
cqlsh localhost 9042 -u "username" -p "password" --execute "DESCRIBE  
TABLE mykeyspace.mytable;"
```

Créez ensuite la table cible dans Amazon Keyspaces avec les noms des colonnes et les types de données correspondant à la description de la table source de Cassandra.

Étape 2 : Préparation des données sources pour un téléchargement de données réussi

La préparation des données sources pour un transfert efficace est un processus en deux étapes. Tout d'abord, vous répartissez les données de manière aléatoire. Au cours de la deuxième étape, vous analysez les données pour déterminer les valeurs de `cqlsh` paramètres appropriées et les paramètres de table requis afin de garantir le succès du téléchargement des données.

Randomiser les données

La `cqlsh COPY FROM` commande lit et écrit les données dans l'ordre dans lequel elles apparaissent dans le fichier CSV. Si vous utilisez la `cqlsh COPY TO` commande pour créer le fichier source, les données sont écrites dans un ordre trié par clé dans le fichier CSV. En interne, Amazon Keyspaces partitionne les données à l'aide de clés de partition. Bien qu'Amazon Keyspaces intègre une logique permettant d'équilibrer la charge des demandes pour la même clé de partition, le chargement des données est plus rapide et plus efficace si vous répartissez la commande de manière aléatoire. En effet, vous pouvez tirer parti de l'équilibrage de charge intégré qui se produit lorsque Amazon Keyspaces écrit sur différentes partitions.

Pour répartir uniformément les écritures sur les partitions, vous devez répartir les données de manière aléatoire dans le fichier source. Vous pouvez écrire une application pour cela ou utiliser un outil open source, tel que [Shuf](#). Shuf est disponible gratuitement sur les distributions Linux, sur macOS (en installant coreutils dans [homebrew](#)) et sur Windows (en utilisant le sous-système Windows pour Linux (WSL)). Une étape supplémentaire est nécessaire pour éviter que la ligne d'en-tête contenant les noms des colonnes ne soit modifiée au cours de cette étape.

Pour randomiser le fichier source tout en préservant l'en-tête, entrez le code suivant.

```
tail -n +2 keyspaces_sample_table.csv | shuf -o keyspace.table.csv && (head  
-1 keyspaces_sample_table.csv && cat keyspace.table.csv ) > keyspace.table.csv1 &&  
mv keyspace.table.csv1 keyspace.table.csv
```

Shuf réécrit les données dans un nouveau fichier CSV appelé `keyspace.table.csv`. Vous pouvez désormais supprimer le `keyspaces_sample_table.csv` fichier, vous n'en avez plus besoin.

Analyser les données

Déterminez la taille moyenne et maximale des lignes en analysant les données.

Vous le faites pour les raisons suivantes :

- La taille moyenne des lignes permet d'estimer la quantité totale de données à transférer.

- Vous avez besoin de la taille de ligne moyenne pour fournir la capacité d'écriture nécessaire au téléchargement des données.
- Vous pouvez vous assurer que la taille de chaque ligne est inférieure à 1 Mo, ce qui correspond à la taille de ligne maximale dans Amazon Keyspaces.

Note

Ce quota fait référence à la taille des lignes et non à la taille de la partition. Contrairement aux partitions Apache Cassandra, les partitions Amazon Keyspaces peuvent être de taille pratiquement indépendante. Les clés de partition et les colonnes de clustering nécessitent un espace de stockage supplémentaire pour les métadonnées, que vous devez ajouter à la taille brute des lignes. Pour de plus amples informations, veuillez consulter [the section called “Estimer la taille des lignes”](#).

Le code suivant utilise [AWK](#) pour analyser un fichier CSV et imprimer la taille de ligne moyenne et maximale.

```
awk -F, 'BEGIN {samp=10000;max=-1;}{if(NR>1){len=length($0);t+=len;avg=t/
NR;max=(len>max ? len : max)}}NR==samp{exit}END{printf("{lines: %d, average: %d bytes,
max: %d bytes}\n",NR,avg,max);}' keySPACE.table.csv
```

L'exécution de ce code génère le résultat suivant.

```
using 10,000 samples:
{lines: 10000, avg: 123 bytes, max: 225 bytes}
```

Vous utiliserez la taille de ligne moyenne à l'étape suivante de ce didacticiel pour configurer la capacité d'écriture de la table.

Étape 3 : définir la capacité de débit de la table

Ce didacticiel explique comment régler cqlsh pour charger des données dans un intervalle de temps défini. Comme vous savez à l'avance combien de lectures et d'écritures vous devez effectuer, utilisez le mode capacité provisionnée. Une fois le transfert de données terminé, vous devez définir le mode de capacité de la table en fonction des modèles de trafic de votre application. Pour en savoir plus sur la gestion des capacités, voir [Gestion des ressources sans serveur](#).

Avec le mode capacité provisionnée, vous spécifiez à l'avance la capacité de lecture et d'écriture que vous souhaitez allouer à votre table. La capacité d'écriture est facturée à l'heure et mesurée en unités de capacité d'écriture (WCU). Chaque WCU possède une capacité d'écriture suffisante pour prendre en charge l'écriture de 1 Ko de données par seconde. Lorsque vous chargez les données, le taux d'écriture doit être inférieur au nombre maximum de WCU (paramètre `:write_capacity_units`) défini sur la table cible.

Par défaut, vous pouvez allouer jusqu'à 40 000 WCU à une table et 80 000 WCU pour toutes les tables de votre compte. Si vous avez besoin de capacités supplémentaires, vous pouvez demander une augmentation de quota dans la console [Service Quotas](#). Pour plus d'informations sur les quotas, consultez [Quotas](#).

Calculez le nombre moyen de WCU nécessaires pour un insert

L'insertion de 1 Ko de données par seconde nécessite 1 WCU. Si votre fichier CSV comporte 360 000 lignes et que vous souhaitez charger toutes les données en 1 heure, vous devez écrire 100 lignes par seconde (360 000 lignes/60 minutes/60 secondes = 100 lignes par seconde). Si chaque ligne contient jusqu'à 1 Ko de données, pour insérer 100 lignes par seconde, vous devez allouer 100 WCU à votre table. Si chaque ligne contient 1,5 Ko de données, vous avez besoin de deux WCU pour insérer une ligne par seconde. Par conséquent, pour insérer 100 lignes par seconde, vous devez provisionner 200 WCU.

Pour déterminer le nombre de WCU dont vous avez besoin pour insérer une ligne par seconde, divisez la taille moyenne des lignes en octets par 1024 et arrondissez au nombre entier le plus proche.

Par exemple, si la taille moyenne des lignes est de 3 000 octets, vous avez besoin de trois WCU pour insérer une ligne par seconde.

$$\text{ROUNDUP}(3000 / 1024) = \text{ROUNDUP}(2.93) = 3 \text{ WCUs}$$

Calculer le temps et la capacité de chargement des données

Maintenant que vous connaissez la taille moyenne et le nombre de lignes de votre fichier CSV, vous pouvez calculer le nombre de WCU nécessaires pour charger les données dans un laps de temps donné, ainsi que le temps approximatif nécessaire pour charger toutes les données de votre fichier CSV en utilisant différents paramètres WCU.

Par exemple, si chaque ligne de votre fichier fait 1 Ko et que votre fichier CSV contient 1 000 000 lignes, pour charger les données en 1 heure, vous devez allouer au moins 278 WCU à votre table pendant cette heure.

```
1,000,000 rows * 1 KBs = 1,000,000 KBs
1,000,000 KBs / 3600 seconds = 277.8 KBs / second = 278 WCUs
```

Configuration des paramètres de capacité provisionnée

Vous pouvez définir les paramètres de capacité d'écriture d'une table lorsque vous créez la table ou à l'aide de la commande ALTER TABLE CQL. Voici la syntaxe permettant de modifier les paramètres de capacité provisionnée d'une table à l'aide de l'instruction ALTER TABLE CQL.

```
ALTER TABLE mykeyspace.mytable WITH custom_properties={'capacity_mode':
{'throughput_mode': 'PROVISIONED', 'read_capacity_units': 100,
'write_capacity_units': 278}} ;
```

Pour la référence linguistique complète, voir [the section called “ALTER TABLE”](#).

Étape 4 : Configuration des paramètres **cqlsh COPY FROM**

Cette section explique comment déterminer les valeurs des paramètres pour **cqlsh COPY FROM**. La **cqlsh COPY FROM** commande lit le fichier CSV que vous avez préparé précédemment et insère les données dans Amazon Keyspaces à l'aide de CQL. La commande divise les lignes et répartit les INSERT opérations entre un ensemble de travailleurs. Chaque collaborateur établit une connexion avec Amazon Keyspaces et envoie des INSERT demandes via ce canal.

La **cqlsh COPY** commande n'a pas de logique interne permettant de répartir le travail de manière égale entre ses employés. Cependant, vous pouvez le configurer manuellement pour vous assurer que le travail est réparti de manière uniforme. Commencez par passer en revue les principaux paramètres **cqlsh** suivants :

- **DÉLIMITEUR** — Si vous avez utilisé un délimiteur autre qu'une virgule, vous pouvez définir ce paramètre, dont la valeur par défaut est une virgule.
- **INGESTRATE** — Nombre cible de lignes à **cqlsh COPY FROM** traiter par seconde. S'il n'est pas défini, la valeur par défaut est de 100 000.
- **NUMPROCESSES** — Le nombre de processus enfants créés par **cqlsh** pour les tâches. **COPY FROM** Le maximum pour ce paramètre est de 16, la valeur par défaut `num_cores - 1` `num_cores` étant le nombre de cœurs de traitement sur l'hôte exécutant **cqlsh**.

- **MAXBATCHSIZE** — La taille du lot détermine le nombre maximal de lignes insérées dans la table de destination en un seul lot. S'il n'est pas défini, `cqlsh` utilise des lots de 20 lignes insérées.
- **CHUNKSIZE** — Taille de l'unité de travail transmise à l'enfant travailleur. Par défaut, il est défini sur 5 000.
- **MAXATTEMPTS** — Le nombre maximum de fois où il est possible de réessayer un worker chunk ayant échoué. Une fois le nombre maximal de tentatives atteint, les enregistrements ayant échoué sont écrits dans un nouveau fichier CSV que vous pourrez réexécuter ultérieurement après avoir examiné l'échec.

INGESTRATE Défini en fonction du nombre de WCU que vous avez provisionnés dans la table de destination cible. La `INGESTRATE cqlsh COPY FROM` commande n'est pas une limite, c'est une moyenne cible. Cela signifie qu'il peut (et c'est souvent le cas) dépasser le nombre que vous avez défini. Pour tenir compte des rafales et vous assurer que la capacité est suffisante pour traiter les demandes de chargement de données, définissez 90 % **INGESTRATE** de la capacité d'écriture de la table.

```
INGESTRATE = WCUs * .90
```

Définissez ensuite le **NUMPROCESSES** paramètre sur une valeur inférieure d'un au nombre de cœurs de votre système. Pour connaître le nombre de cœurs de votre système, vous pouvez exécuter le code suivant.

```
python -c "import multiprocessing; print(multiprocessing.cpu_count())"
```

Pour ce didacticiel, nous utilisons la valeur suivante.

```
NUMPROCESSES = 4
```

Chaque processus crée un travailleur, et chaque travailleur établit une connexion à Amazon Keyspaces. Amazon Keyspaces peut prendre en charge jusqu'à 3 000 demandes CQL par seconde à chaque connexion. Cela signifie que vous devez vous assurer que chaque travailleur traite moins de 3 000 demandes par seconde.

Comme c'est le cas **INGESTRATE**, les travailleurs dépassent souvent le nombre que vous avez défini et ne sont pas limités par les secondes d'horloge. Par conséquent, pour tenir compte des rafales, définissez vos paramètres `cqlsh` de manière à ce que chaque travailleur traite 2 500 demandes

par seconde. Pour calculer la quantité de travail distribuée à un travailleur, suivez les directives suivantes.

- Divisez `INGESTRATE` par `NUMPROCESSES`.
- Si $INGESTRATE / NUMPROCESSES > 2\,500$, abaissez le `INGESTRATE` pour que cette formule soit vraie.

```
INGESTRATE / NUMPROCESSES <= 2,500
```

Avant de configurer les paramètres pour optimiser le téléchargement de nos exemples de données, examinons les paramètres `cqlsh` par défaut et voyons comment leur utilisation influe sur le processus de téléchargement des données. Dans la `cqlsh COPY FROM` mesure où il est utilisé `CHUNKSIZE` pour créer des parties du travail (`INSERT` déclarations) à distribuer aux travailleurs, le travail n'est pas automatiquement distribué de manière uniforme. Certains travailleurs peuvent rester inactifs, selon le `INGESTRATE` réglage.

Pour répartir le travail de manière égale entre les travailleurs et maintenir chaque travailleur au taux optimal de 2 500 demandes par seconde, vous devez définir `CHUNKSIZE`, `MAXBATCHSIZE`, et `INGESTRATE` en modifiant les paramètres d'entrée. Pour optimiser l'utilisation du trafic réseau pendant le chargement des données, choisissez une valeur proche de la valeur maximale de 30. `MAXBATCHSIZE` En passant `CHUNKSIZE` à 100 et `MAXBATCHSIZE` à 25, les 10 000 rangées sont réparties uniformément entre les quatre travailleurs ($10\,000 / 2500 = 4$).

L'exemple de code suivant illustre cela.

```
INGESTRATE = 10,000
NUMPROCESSES = 4
CHUNKSIZE = 100
MAXBATCHSIZE. = 25
Work Distribution:
Connection 1 / Worker 1 : 2,500 Requests per second
Connection 2 / Worker 2 : 2,500 Requests per second
Connection 3 / Worker 3 : 2,500 Requests per second
Connection 4 / Worker 4 : 2,500 Requests per second
```

En résumé, utilisez les formules suivantes lors de la définition `cqlsh COPY FROM` des paramètres :

- `INGESTRATE = write_capacity_units * .90`
- `NUMPROCESSES = num_cores - 1` (par défaut)

- $\text{INGESTRATE/NUMPROCESSES} = 2\ 500$ (Cette déclaration doit être vraie.)
- $\text{MAXBATCHSIZE} = 30$ (20 par défaut). Amazon Keyspaces accepte des lots allant jusqu'à 30.)
- $\text{CHUNKSIZE} = (\text{INGESTRATE/ NUMPROCESSES})/\text{MAXBATCHSIZE}$

Maintenant que vous avez calculé $\text{NUMPROCESSESINGESTRATE}$, et CHUNKSIZE , vous êtes prêt à charger vos données.

Étape 5 : Exécutez la commande **cqlsh COPY FROM** pour télécharger les données du fichier CSV vers la table cible

Pour exécuter la `cqlsh COPY FROM` commande, procédez comme suit.

1. Connectez-vous à Amazon Keyspaces à l'aide de `cqlsh`.
2. Choisissez votre keyspace à l'aide du code suivant.

```
USE catalog;
```

3. Définissez la cohérence d'écriture sur `LOCAL_QUORUM`. Pour garantir la durabilité des données, Amazon Keyspaces n'autorise aucun autre paramètre de cohérence d'écriture. Consultez le code suivant.

```
CONSISTENCY LOCAL_QUORUM;
```

4. Préparez votre `cqlsh COPY FROM` syntaxe à l'aide de l'exemple de code suivant.

```
COPY book_awards FROM './keyspace.table.csv' WITH HEADER=true  
AND INGESTRATE=calculated ingestrate  
AND NUMPROCESSES=calculated numprocess  
AND MAXBATCHSIZE=20  
AND CHUNKSIZE=calculated chunksize;
```

5. Exécutez l'instruction préparée à l'étape précédente. `cqlsh` renvoie tous les paramètres que vous avez configurés.
 - a. Assurez-vous que les paramètres correspondent à ce que vous avez saisi. Consultez l'exemple suivant.

```
Reading options from the command line: {'chunksize': '120', 'header': 'true',  
'ingestrate': '36000', 'numprocesses': '15', 'maxbatchsize': '20'}
```



```
Using 15 child processes
```

- b. Vérifiez le nombre de lignes transférées et le taux moyen actuel, comme indiqué dans l'exemple suivant.

```
Processed: 57834 rows; Rate: 6561 rows/s; Avg. rate: 31751 rows/s
```

- c. Lorsque cqlsh a fini de télécharger les données, consultez le résumé des statistiques de chargement des données (nombre de fichiers lus, temps d'exécution et lignes ignorées) comme indiqué dans l'exemple suivant.

```
15556824 rows imported from 1 files in 8 minutes and 8.321 seconds (0 skipped).
```

Dans cette dernière étape du didacticiel, vous avez chargé les données sur Amazon Keyspaces.

Important

Maintenant que vous avez transféré vos données, ajustez les paramètres du mode capacité de votre table cible pour qu'ils correspondent aux modèles de trafic habituels de votre application. Vous êtes facturé au taux horaire pour votre capacité provisionnée jusqu'à ce que vous la modifiiez.

Résolution des problèmes

Une fois le téléchargement des données terminé, vérifiez si des lignes ont été ignorées. Pour ce faire, accédez au répertoire source du fichier CSV source et recherchez un fichier portant le nom suivant.

```
import_yourcsvfilename.err.timestamp.csv
```

cqlsh écrit toutes les lignes de données ignorées dans un fichier portant ce nom. Si le fichier existe dans votre répertoire source et contient des données, ces lignes n'ont pas été chargées sur Amazon Keyspaces. Pour réessayer ces lignes, vérifiez d'abord si des erreurs se sont produites lors du téléchargement et ajustez les données en conséquence. Pour réessayer ces lignes, vous pouvez relancer le processus.

Erreurs courantes

Les raisons les plus courantes pour lesquelles les lignes ne sont pas chargées sont les erreurs de capacité et les erreurs d'analyse.

Erreurs de demande non valides lors du chargement de données sur Amazon Keyspaces

Dans l'exemple suivant, la table source contient une colonne de compteur, qui génère des appels par lots enregistrés à partir de la commande `cqlshCOPY`. Les appels groupés enregistrés ne sont pas pris en charge par Amazon Keyspaces.

```
Failed to import 10 rows: InvalidRequest - Error from server: code=2200 [Invalid query]
message="Only UNLOGGED Batches are supported at this time.", will retry later,
attempt 22 of 25
```

Pour résoudre cette erreur, utilisez `DSBulk` pour migrer les données. Pour de plus amples informations, veuillez consulter [the section called “Chargement de données à l'aide de DSBulk”](#).

Erreurs de l'analyseur lors du chargement de données sur Amazon Keyspaces

L'exemple suivant montre une ligne ignorée en raison d'un `ParseException`.

```
Failed to import 1 rows: ParseException - Invalid ... -
```

Pour résoudre cette erreur, vous devez vous assurer que les données à importer correspondent au schéma de table dans Amazon Keyspaces. Vérifiez le fichier d'importation pour détecter les erreurs d'analyse. Vous pouvez essayer d'utiliser une seule ligne de données à l'aide d'une `INSERT` instruction pour isoler l'erreur.

Erreurs de capacité lors du chargement de données sur Amazon Keyspaces

```
Failed to import 1 rows: WriteTimeout - Error from server: code=1100 [Coordinator node
timed out waiting for replica nodes' responses]
message="Operation timed out - received only 0 responses." info={'received_responses':
0, 'required_responses': 2, 'write_type': 'SIMPLE', 'consistency':
'LOCAL_QUORUM'}, will retry later, attempt 1 of 100
```

Amazon Keyspaces utilise les `WriteTimeout` exceptions `ReadTimeout` et pour indiquer lorsqu'une demande d'écriture échoue en raison d'une capacité de débit insuffisante. Pour aider à diagnostiquer les exceptions liées à une capacité insuffisante, Amazon Keyspaces publie `WriteThrottleEvents` des `ReadThrottledEvents` statistiques sur Amazon CloudWatch. Pour de plus amples informations, veuillez consulter [the section called “Surveillance avec CloudWatch”](#).

erreurs cqlsh lors du chargement de données sur Amazon Keyspaces

Pour résoudre les erreurs cqlsh, réexécutez la commande défailante avec l'indicateur. `--debug`

Lorsque vous utilisez une version incompatible de cqlsh, l'erreur suivante s'affiche.

```
AttributeError: 'NoneType' object has no attribute 'is_up'  
Failed to import 3 rows: AttributeError - 'NoneType' object has no attribute 'is_up',  
given up after 1 attempts
```

Vérifiez que la version correcte de cqlsh est installée en exécutant la commande suivante.

```
cqlsh --version
```

Vous devriez voir quelque chose comme la sortie suivante.

```
cqlsh 5.0.1
```

Si vous utilisez Windows, remplacez toutes les instances de cqlsh par cqlsh.bat. Par exemple, pour vérifier la version de cqlsh sous Windows, exécutez la commande suivante.

```
cqlsh.bat --version
```

La connexion à Amazon Keyspaces échoue lorsque le client cqlsh a reçu trois erreurs consécutives de tout type de la part du serveur. Le client cqlsh échoue avec le message suivant.

```
Failed to import 1 rows: NoHostAvailable - , will retry later, attempt 3 of 100
```

Pour résoudre cette erreur, vous devez vous assurer que les données à importer correspondent au schéma de table dans Amazon Keyspaces. Vérifiez le fichier d'importation pour détecter les erreurs d'analyse. Vous pouvez essayer d'utiliser une seule ligne de données en utilisant une instruction INSERT pour isoler l'erreur.

Le client tente automatiquement de rétablir la connexion.

Tutoriel : Chargement de données dans Amazon Keyspaces à l'aide de DSBulk

Ce didacticiel étape par étape vous explique comment migrer des données d'Apache Cassandra vers Amazon Keyspaces à l'aide du DataStax Bulk Loader (DSBulk) disponible sur. [GitHub](#) L'utilisation

de DSBulk est utile pour télécharger des ensembles de données sur Amazon Keyspaces à des fins académiques ou de test. Pour plus d'informations sur la migration des charges de travail de production, consultez [the section called “Migration hors ligne”](#). Dans ce didacticiel, vous devez suivre les étapes suivantes.

Conditions préalables — Configurez un AWS compte avec des informations d'identification, créez un fichier JKS Trust Store pour le certificat, configurez `cqlsh`, téléchargez et installez DSBulk, puis configurez un fichier `application.conf`

1. Créer un fichier CSV source et une table cible : préparez un fichier CSV en tant que données source et créez le keyspace et le tableau cibles dans Amazon Keyspaces.
2. Préparation des données : répartissez les données du fichier CSV de manière aléatoire et analysez-les pour déterminer les tailles de ligne moyennes et maximales.
3. Définissez la capacité de débit : calculez les unités de capacité d'écriture (WCU) requises en fonction de la taille des données et du temps de chargement souhaité, puis configurez la capacité provisionnée de la table.
4. Configurer les paramètres DSBulk : créez un fichier de configuration DSBulk avec des paramètres tels que l'authentification SSL/TLS, le niveau de cohérence et la taille du pool de connexions.
5. Exécutez la commande DSBulk load : exécutez la commande DSBulk load pour télécharger les données du fichier CSV vers le tableau Amazon Keyspaces et suivre la progression.

Rubriques

- [Conditions préalables : étapes à suivre avant de pouvoir télécharger des données avec DSBulk](#)
- [Étape 1 : Création du fichier CSV source et d'une table cible pour le téléchargement des données à l'aide de DSBulk](#)
- [Étape 2 : préparer les données à télécharger à l'aide de DSBulk](#)
- [Étape 3 : définir la capacité de débit pour la table cible](#)
- [Étape 4 : configurer les paramètres DSBulk pour télécharger les données du fichier CSV vers la table cible](#)
- [Étape 5 : Exécutez la commande de chargement DSBulk pour télécharger les données du fichier CSV vers la table cible](#)

Conditions préalables : étapes à suivre avant de pouvoir télécharger des données avec DSBulk

Vous devez effectuer les tâches suivantes avant de pouvoir commencer ce didacticiel.

1. Si ce n'est pas déjà fait, créez un AWS compte en suivant les étapes indiquées sur [the section called “Configuration d'Amazon Keyspaces”](#).
2. Créez des informations d'identification en suivant les étapes décrites dans [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#).
3. Créez un fichier JKS Trust Store.
 - a. Téléchargez les certificats numériques suivants et enregistrez les fichiers localement ou dans votre répertoire personnel.
 1. AmazonRootCA1
 2. AmazonRootCA2
 3. AmazonRootCA3
 4. AmazonRootCA4
 5. Starfield Class 2 Root (en option, pour une rétrocompatibilité)

Pour télécharger les certificats, vous pouvez utiliser les commandes suivantes.

```
curl -O https://www.amazontrust.com/repository/AmazonRootCA1.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA2.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA3.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA4.pem
curl -O https://certs.secureserver.net/repository/sf-class2-root.crt
```

Note

Amazon Keyspaces utilisait auparavant des certificats TLS ancrés à l'autorité de certification Starfield Class 2. AWS migre le tout Régions AWS vers des certificats émis dans le cadre d'Amazon Trust Services (Amazon Root CA 1—4). Au cours de cette transition, configurez les clients de manière à ce qu'ils fassent confiance à la fois aux autorités de certification Amazon Root 1 à 4 et à la racine Starfield afin de garantir la compatibilité entre toutes les régions.

b. Convertissez les certificats numériques en fichiers TrustStore et ajoutez-les au keystore.

```
openssl x509 -outform der -in AmazonRootCA1.pem -out temp_file.der
keytool -import -alias amazon-root-ca-1 -keystore cassandra_truststore.jks -
file temp_file.der

openssl x509 -outform der -in AmazonRootCA2.pem -out temp_file.der
keytool -import -alias amazon-root-ca-2 -keystore cassandra_truststore.jks -
file temp_file.der

openssl x509 -outform der -in AmazonRootCA3.pem -out temp_file.der
keytool -import -alias amazon-root-ca-3 -keystore cassandra_truststore.jks -
file temp_file.der

openssl x509 -outform der -in AmazonRootCA4.pem -out temp_file.der
keytool -import -alias amazon-root-ca-4 -keystore cassandra_truststore.jks -
file temp_file.der

openssl x509 -outform der -in sf-class2-root.crt -out temp_file.der
keytool -import -alias cassandra -keystore cassandra_truststore.jks -file
temp_file.der
```

Dans la dernière étape, vous devez créer un mot de passe pour le keystore et faire confiance à chaque certificat. La commande interactive ressemble à ceci.

```
Enter keystore password:
Re-enter new password:
Owner: CN=Amazon Root CA 1, O=Amazon, C=US
Issuer: CN=Amazon Root CA 1, O=Amazon, C=US
Serial number: 66c9fcf99bf8c0a39e2f0788a43e696365bca
Valid from: Tue May 26 00:00:00 UTC 2015 until: Sun Jan 17 00:00:00 UTC 2038
Certificate fingerprints:
    SHA1: 8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
    SHA256:
    8E:CD:E6:88:4F:3D:87:B1:12:5B:A3:1A:C3:FC:B1:3D:70:16:DE:7F:57:CC:90:4F:E1:CB:97:C6:AE
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.19 Criticality=true
```

```

BasicConstraints:[
  CA:true
  PathLen:2147483647
]

#2: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
  DigitalSignature
  Key_CertSign
  Crl_Sign
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 84 18 CC 85 34 EC BC 0C   94 94 2E 08 59 9C C7 B2   ....4.....Y...
0010: 10 4E 0A 08                               .N..
]
]

Trust this certificate? [no]: yes
Certificate was added to keystore
Enter keystore password:
Owner: CN=Amazon Root CA 2, O=Amazon, C=US
Issuer: CN=Amazon Root CA 2, O=Amazon, C=US
Serial number: 66c9fd29635869f0a0fe58678f85b26bb8a37
Valid from: Tue May 26 00:00:00 UTC 2015 until: Sat May 26 00:00:00 UTC 2040
Certificate fingerprints:
    SHA1: 5A:8C:EF:45:D7:A6:98:59:76:7A:8C:8B:44:96:B5:78:CF:47:4B:1A
    SHA256:
    1B:A5:B2:AA:8C:65:40:1A:82:96:01:18:F8:0B:EC:4F:62:30:4D:83:CE:C4:71:3A:19:C3:9C:01:1E
Signature algorithm name: SHA384withRSA
Subject Public Key Algorithm: 4096-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

#2: ObjectId: 2.5.29.15 Criticality=true

```

```

KeyUsage [
  DigitalSignature
  Key_CertSign
  Crl_Sign
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
  KeyIdentifier [
    0000: B0 0C F0 4C 30 F4 05 58    02 48 FD 33 E5 52 AF 4B    ...L0..X.H.3.R.K
    0010: 84 E3 66 52                                ..fR
  ]
]

Trust this certificate? [no]: yes
Certificate was added to keystore
Enter keystore password:
Owner: CN=Amazon Root CA 3, O=Amazon, C=US
Issuer: CN=Amazon Root CA 3, O=Amazon, C=US
Serial number: 66c9fd5749736663f3b0b9ad9e89e7603f24a
Valid from: Tue May 26 00:00:00 UTC 2015 until: Sat May 26 00:00:00 UTC 2040
Certificate fingerprints:
    SHA1: 0D:44:DD:8C:3C:8C:1A:1A:58:75:64:81:E9:0F:2E:2A:FF:B3:D2:6E
    SHA256:
    18:CE:6C:FE:7B:F1:4E:60:B2:E3:47:B8:DF:E8:68:CB:31:D0:2E:BB:3A:DA:27:15:69:F5:03:43:B4
Signature algorithm name: SHA256withECDSA
Subject Public Key Algorithm: 256-bit EC (secp256r1) key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

#2: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
  DigitalSignature
  Key_CertSign
  Crl_Sign
]

```



```

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: AB B6 DB D7 06 9E 37 AC   30 86 07 91 70 C7 9C C4   .....7.0...p...
0010: 19 B1 78 C0                               ...x.
]
]

Trust this certificate? [no]: yes
Certificate was added to keystore
Enter keystore password:
Owner: CN=Amazon Root CA 4, O=Amazon, C=US
Issuer: CN=Amazon Root CA 4, O=Amazon, C=US
Serial number: 66c9fd7c1bb104c2943e5717b7b2cc81ac10e
Valid from: Tue May 26 00:00:00 UTC 2015 until: Sat May 26 00:00:00 UTC 2040
Certificate fingerprints:
    SHA1: F6:10:84:07:D6:F8:BB:67:98:0C:C2:E2:44:C2:EB:AE:1C:EF:63:BE
    SHA256:
    E3:5D:28:41:9E:D0:20:25:CF:A6:90:38:CD:62:39:62:45:8D:A5:C6:95:FB:DE:A3:C2:2B:0B:FB:25
Signature algorithm name: SHA384withECDSA
Subject Public Key Algorithm: 384-bit EC (secp384r1) key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
    CA:true
    PathLen:2147483647
]

#2: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
    DigitalSignature
    Key_CertSign
    Crl_Sign
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: D3 EC C7 3A 65 6E CC E1   DA 76 9A 56 FB 9C F3 86   ...:en...v.V....
0010: 6D 57 E5 81                               mW..
]

```

```

]

Trust this certificate? [no]:  yes
Certificate was added to keystore
Enter keystore password:
Owner: OU=Starfield Class 2 Certification Authority, O="Starfield Technologies,
  Inc.", C=US
Issuer: OU=Starfield Class 2 Certification Authority, O="Starfield
  Technologies, Inc.", C=US
Serial number: 0
Valid from: Tue Jun 29 17:39:16 UTC 2004 until: Thu Jun 29 17:39:16 UTC 2034
Certificate fingerprints:
    SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
    SHA256:
    14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB
Signature algorithm name: SHA1withRSA (weak)
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
  KeyIdentifier [
0000: BF 5F B7 D1 CE DD 1F 86   F4 5B 55 AC DC D7 10 C2   ._.....[U.....
0010: 0E A9 88 E7               ....
  ]
[OU=Starfield Class 2 Certification Authority, O="Starfield Technologies,
  Inc.", C=US]
SerialNumber: [    00]
]

#2: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
  KeyIdentifier [
0000: BF 5F B7 D1 CE DD 1F 86   F4 5B 55 AC DC D7 10 C2   ._.....[U.....
0010: 0E A9 88 E7               ....
  ]
]

```

```
]
```

Warning:

The input uses the SHA1withRSA signature algorithm which is considered a security risk. This algorithm will be disabled in a future update.

```
Trust this certificate? [no]: yes
Certificate was added to keystore
```

4. Configurez la connexion au shell Cassandra Query Language (cqlsh) et confirmez que vous pouvez vous connecter à Amazon Keyspaces en suivant les étapes indiquées sur [the section called “Utilisation de cqlsh”](#)
5. Téléchargez et installez DSBulk.

 Note

La version présentée dans ce didacticiel n'est peut-être pas la dernière version disponible. Avant de télécharger DSBulk, consultez la [page de téléchargement de DataStax Bulk Loader](#) pour obtenir la dernière version et mettez à jour le numéro de version dans les commandes suivantes en conséquence.

- a. Pour télécharger DSBulk, vous pouvez utiliser le code suivant.

```
curl -OL https://downloads.datastax.com/dsbulk/dsbulk-1.8.0.tar.gz
```

- b. Décompressez ensuite le fichier tar et ajoutez DSBulk à votre fichier, PATH comme indiqué dans l'exemple suivant.

```
tar -zxvf dsbulk-1.8.0.tar.gz
# add the DSBulk directory to the path
export PATH=$PATH:./dsbulk-1.8.0/bin
```

- c. Créez un `application.conf` fichier pour stocker les paramètres à utiliser par DSBulk. Vous pouvez enregistrer l'exemple suivant sous le nom `./dsbulk_keyspaces.conf`.
`localhost` Remplacez-le par le point de contact de votre cluster Cassandra local si vous n'êtes pas sur le nœud local, par exemple le nom DNS ou l'adresse IP. Prenez note du nom et du chemin du fichier, car vous devrez le spécifier ultérieurement dans la `dsbulk load` commande.

```
datastax-java-driver {  
  basic.contact-points = [ "localhost"  
  advanced.auth-provider {  
    class = software.aws.mcs.auth.SigV4AuthProvider  
    aws-region = us-east-1  
  }  
}
```

- d. Pour activer le support SigV4, téléchargez le jar fichier ombré depuis [GitHub](#) et placez-le dans le lib dossier DSBulk comme indiqué dans l'exemple suivant.

```
curl -O -L https://github.com/aws/aws-sigv4-auth-cassandra-java-driver-plugin/  
releases/download/4.0.6-shaded-v2/aws-sigv4-auth-cassandra-java-driver-  
plugin-4.0.6-shaded.jar
```

Étape 1 : Création du fichier CSV source et d'une table cible pour le téléchargement des données à l'aide de DSBulk

Pour ce didacticiel, nous utilisons un fichier de valeurs séparées par des virgules (CSV) dont le `keyspaces_sample_table.csv` nom est le fichier source pour la migration des données. Le fichier d'exemple fourni contient quelques lignes de données pour une table portant le nom `book_awards`.

1. Créez le fichier source. Vous pouvez choisir l'une des options suivantes :
 - Téléchargez l'exemple de fichier CSV (`keyspaces_sample_table.csv`) contenu dans le fichier d'archive [samplemigration.zip](#) suivant. Décompressez l'archive et notez le chemin vers `keyspaces_sample_table.csv`.
 - Pour remplir un fichier CSV avec vos propres données stockées dans une base de données Apache Cassandra, vous pouvez remplir le fichier CSV source en utilisant `dsbulk unload` comme indiqué dans l'exemple suivant.

```
dsbulk unload -k mykeyspace -t mytable -f ./my_application.conf  
> keyspace_sample_table.csv
```

Assurez-vous que le fichier CSV que vous créez répond aux exigences suivantes :

- La première ligne contient les noms des colonnes.

- Les noms des colonnes du fichier CSV source correspondent aux noms des colonnes de la table cible.
- Les données sont délimitées par une virgule.
- Toutes les valeurs de données sont des types de données Amazon Keyspaces valides. Consultez [the section called “Types de données”](#).

2. Créez le keyspace et le tableau cibles dans Amazon Keyspaces.

- a. Connectez-vous à Amazon Keyspaces en utilisant `cqlsh`, en remplaçant le point de terminaison du service, le nom d'utilisateur et le mot de passe dans l'exemple suivant par vos propres valeurs.

```
cqlsh cassandra.us-east-1.amazonaws.com 9142 -u "111122223333" -p  
"wJa1rXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY" --ssl
```

- b. Créez un nouvel espace de touches avec le `catalog` nom indiqué dans l'exemple suivant.

```
CREATE KEYSPACE catalog WITH REPLICATION = {'class': 'SingleRegionStrategy'};
```

- c. Une fois que le nouveau keyspace est devenu disponible, utilisez le code suivant pour créer la table `book_awards` cible. Pour en savoir plus sur la création de ressources asynchrones et sur la façon de vérifier si une ressource est disponible, consultez [the section called “Vérifier l'état de création de l'espace clé”](#)

```
CREATE TABLE catalog.book_awards (  
    year int,  
    award text,  
    rank int,  
    category text,  
    book_title text,  
    author text,  
    publisher text,  
    PRIMARY KEY ((year, award), category, rank)  
);
```

Si Apache Cassandra est votre source de données d'origine, un moyen simple de créer la table cible Amazon Keyspaces avec les en-têtes correspondants consiste à générer `CREATE TABLE` l'instruction à partir de la table source, comme indiqué dans l'instruction suivante.

```
cqlsh localhost 9042 -u "username" -p "password" --execute "DESCRIBE  
TABLE mykeyspace.mytable;"
```

Créez ensuite la table cible dans Amazon Keyspaces avec les noms des colonnes et les types de données correspondant à la description de la table source de Cassandra.

Étape 2 : préparer les données à télécharger à l'aide de DSBulk

La préparation des données sources pour un transfert efficace est un processus en deux étapes. Tout d'abord, vous répartissez les données de manière aléatoire. Dans la deuxième étape, vous analysez les données pour déterminer les valeurs de `dsbulk` paramètres appropriées et les paramètres de table requis.

Randomiser les données

La `dsbulk` commande lit et écrit les données dans l'ordre dans lequel elles apparaissent dans le fichier CSV. Si vous utilisez la `dsbulk` commande pour créer le fichier source, les données sont écrites dans un ordre trié par clé dans le fichier CSV. En interne, Amazon Keyspaces partitionne les données à l'aide de clés de partition. Bien qu'Amazon Keyspaces intègre une logique permettant d'équilibrer la charge des demandes pour la même clé de partition, le chargement des données est plus rapide et plus efficace si vous répartissez la commande de manière aléatoire. En effet, vous pouvez tirer parti de l'équilibrage de charge intégré qui se produit lorsque Amazon Keyspaces écrit sur différentes partitions.

Pour répartir uniformément les écritures sur les partitions, vous devez répartir les données de manière aléatoire dans le fichier source. Vous pouvez écrire une application pour cela ou utiliser un outil open source, tel que [Shuf](#). Shuf est disponible gratuitement sur les distributions Linux, sur macOS (en installant `coreutils` dans [homebrew](#)) et sur Windows (en utilisant le sous-système Windows pour Linux (WSL)). Une étape supplémentaire est nécessaire pour éviter que la ligne d'en-tête contenant les noms des colonnes ne soit modifiée au cours de cette étape.

Pour randomiser le fichier source tout en préservant l'en-tête, entrez le code suivant.

```
tail -n +2 keyspaces_sample_table.csv | shuf -o keyspace.table.csv && (head  
-1 keyspaces_sample_table.csv && cat keyspace.table.csv ) > keyspace.table.csv1 &&  
mv keyspace.table.csv1 keyspace.table.csv
```

Shuf réécrit les données dans un nouveau fichier CSV appelé `keyspace.table.csv`. Vous pouvez désormais supprimer le `keyspaces_sample_table.csv` fichier, vous n'en avez plus besoin.

Analyser les données

Déterminez la taille moyenne et maximale des lignes en analysant les données.

Vous le faites pour les raisons suivantes :

- La taille moyenne des lignes permet d'estimer la quantité totale de données à transférer.
- Vous avez besoin de la taille de ligne moyenne pour fournir la capacité d'écriture nécessaire au téléchargement des données.
- Vous pouvez vous assurer que la taille de chaque ligne est inférieure à 1 Mo, ce qui correspond à la taille de ligne maximale dans Amazon Keyspaces.

Note

Ce quota fait référence à la taille des lignes et non à la taille de la partition. Contrairement aux partitions Apache Cassandra, les partitions Amazon Keyspaces peuvent être de taille pratiquement indépendante. Les clés de partition et les colonnes de clustering nécessitent un espace de stockage supplémentaire pour les métadonnées, que vous devez ajouter à la taille brute des lignes. Pour de plus amples informations, veuillez consulter [the section called “Estimer la taille des lignes”](#).

Le code suivant utilise [AWK](#) pour analyser un fichier CSV et imprimer la taille de ligne moyenne et maximale.

```
awk -F, 'BEGIN {samp=10000;max=-1;}{if(NR>1){len=length($0);t+=len;avg=t/NR;max=(len>max ? len : max)}}NR==samp{exit}END{printf("{lines: %d, average: %d bytes, max: %d bytes}\n",NR,avg,max);}' keyspace.table.csv
```

L'exécution de ce code génère le résultat suivant.

```
using 10,000 samples:
{lines: 10000, avg: 123 bytes, max: 225 bytes}
```

Assurez-vous que la taille maximale de votre ligne ne dépasse pas 1 Mo. Si c'est le cas, vous devez diviser la ligne ou compresser les données pour ramener la taille de la ligne en dessous de 1 Mo.

Dans l'étape suivante de ce didacticiel, vous allez utiliser la taille de ligne moyenne pour allouer la capacité d'écriture de la table.

Étape 3 : définir la capacité de débit pour la table cible

Ce didacticiel explique comment régler DSBulk pour charger des données dans un intervalle de temps défini. Comme vous savez à l'avance combien de lectures et d'écritures vous devez effectuer, utilisez le mode capacité provisionnée. Une fois le transfert de données terminé, vous devez définir le mode de capacité de la table en fonction des modèles de trafic de votre application. Pour en savoir plus sur la gestion des capacités, voir [Gestion des ressources sans serveur](#).

Avec le mode capacité provisionnée, vous spécifiez à l'avance la capacité de lecture et d'écriture que vous souhaitez allouer à votre table. La capacité d'écriture est facturée à l'heure et mesurée en unités de capacité d'écriture (WCU). Chaque WCU possède une capacité d'écriture suffisante pour prendre en charge l'écriture de 1 Ko de données par seconde. Lorsque vous chargez les données, le taux d'écriture doit être inférieur au nombre maximum de WCU (paramètre `:write_capacity_units`) défini sur la table cible.

Par défaut, vous pouvez allouer jusqu'à 40 000 WCU à une table et 80 000 WCU pour toutes les tables de votre compte. Si vous avez besoin de capacités supplémentaires, vous pouvez demander une augmentation de quota dans la console [Service Quotas](#). Pour plus d'informations sur les quotas, consultez [Quotas](#).

Calculez le nombre moyen de WCU nécessaires pour un insert

L'insertion de 1 Ko de données par seconde nécessite 1 WCU. Si votre fichier CSV comporte 360 000 lignes et que vous souhaitez charger toutes les données en 1 heure, vous devez écrire 100 lignes par seconde ($360\,000 \text{ lignes} / 60 \text{ minutes} / 60 \text{ secondes} = 100 \text{ lignes par seconde}$). Si chaque ligne contient jusqu'à 1 Ko de données, pour insérer 100 lignes par seconde, vous devez allouer 100 WCU à votre table. Si chaque ligne contient 1,5 Ko de données, vous avez besoin de deux WCU pour insérer une ligne par seconde. Par conséquent, pour insérer 100 lignes par seconde, vous devez provisionner 200 WCU.

Pour déterminer le nombre de WCU dont vous avez besoin pour insérer une ligne par seconde, divisez la taille moyenne des lignes en octets par 1024 et arrondissez au nombre entier le plus proche.

Par exemple, si la taille moyenne des lignes est de 3 000 octets, vous avez besoin de trois WCU pour insérer une ligne par seconde.


```
ROUNDUP(3000 / 1024) = ROUNDUP(2.93) = 3 WCUs
```

Calculer le temps et la capacité de chargement des données

Maintenant que vous connaissez la taille moyenne et le nombre de lignes de votre fichier CSV, vous pouvez calculer le nombre de WCU nécessaires pour charger les données dans un laps de temps donné, ainsi que le temps approximatif nécessaire pour charger toutes les données de votre fichier CSV en utilisant différents paramètres WCU.

Par exemple, si chaque ligne de votre fichier fait 1 Ko et que votre fichier CSV contient 1 000 000 lignes, pour charger les données en 1 heure, vous devez allouer au moins 278 WCU à votre table pendant cette heure.

```
1,000,000 rows * 1 KBs = 1,000,000 KBs  
1,000,000 KBs / 3600 seconds = 277.8 KBs / second = 278 WCUs
```

Configuration des paramètres de capacité provisionnée

Vous pouvez définir les paramètres de capacité d'écriture d'une table lorsque vous créez la table ou à l'aide de la ALTER TABLE commande. Voici la syntaxe permettant de modifier les paramètres de capacité provisionnée d'une table à l'aide de la ALTER TABLE commande.

```
ALTER TABLE catalog.book_awards WITH custom_properties={'capacity_mode':  
{'throughput_mode': 'PROVISIONED', 'read_capacity_units': 100, 'write_capacity_units':  
278}} ;
```

Pour la référence linguistique complète, voir [the section called “CREATE TABLE”](#) et [the section called “ALTER TABLE”](#).

Étape 4 : configurer les paramètres **DSBulk** pour télécharger les données du fichier CSV vers la table cible

Cette section décrit les étapes requises pour configurer DSBulk pour le téléchargement de données vers Amazon Keyspaces. Vous configurez DSBulk à l'aide d'un fichier de configuration. Vous spécifiez le fichier de configuration directement depuis la ligne de commande.

1. Créez un fichier de configuration DSBulk pour la migration vers Amazon Keyspaces. Dans cet exemple, nous utilisons le nom du fichier. `dsbulk_keyspaces.conf` Spécifiez les paramètres suivants dans le fichier de configuration DSBulk.

- a. *PlainTextAuthProvider*— Créez le fournisseur d'authentification avec la *PlainTextAuthProvider* classe. *ServiceUserName* et *ServicePassword* doivent correspondre au nom d'utilisateur et au mot de passe que vous avez obtenus lorsque vous avez généré les informations d'identification spécifiques au service en suivant les étapes décrites dans [the section called “Création d'identifiants d'accès programmatiques”](#)
- b. *local-datacenter*— Définissez la valeur de Région AWS celle *local-datacenter* à laquelle vous vous connectez. Par exemple, si l'application se connecte à *cassandra.us-east-1.amazonaws.com*, définissez le centre de données local sur *us-east-1*. Pour toutes les options disponibles Régions AWS, voir [the section called “Points de terminaison de service”](#). Pour éviter les répliques, définissez *slow-replica-avoidance* sur *false*
- c. *SSLConnectionFactory*— Pour configurer SSL/TLS, initialisez le *SSLConnectionFactory* en ajoutant une section dans le fichier de configuration avec une seule ligne qui spécifie la classe avec *class = DefaultSslConnectionFactory*. Indiquez le chemin d'accès *cassandra_truststore.jks* et le mot de passe que vous avez créés précédemment.
- d. *consistency*— Réglez le niveau de cohérence sur *LOCAL_QUORUM*. Les autres niveaux de cohérence d'écriture ne sont pas pris en charge. Pour plus d'informations, consultez [the section called “Niveaux de cohérence Cassandra pris en charge”](#).
- e. Le nombre de connexions par pool est configurable dans le pilote Java. Pour cet exemple, définissez *advanced.connection.pool.local.size* sur 3.

L'exemple de fichier de configuration complet est le suivant.

```
datastax-java-driver {
  basic.contact-points = [ "cassandra.us-east-1.amazonaws.com:9142" ]
  advanced.auth-provider {
    class = PlainTextAuthProvider
    username = "ServiceUserName"
    password = "ServicePassword"
  }

  basic.load-balancing-policy {
    local-datacenter = "us-east-1"
    slow-replica-avoidance = false
  }

  basic.request {
    consistency = LOCAL_QUORUM
  }
}
```

```
default-idempotence = true
}
advanced.ssl-engine-factory {
  class = DefaultSslEngineFactory
  truststore-path = "./cassandra_truststore.jks"
  truststore-password = "my_password"
  hostname-validation = false
}
advanced.connection.pool.local.size = 3
}
```

2. Vérifiez les paramètres de la load commande DSBulk.

- a. *executor.maxPerSecond*— Le nombre maximum de lignes que la commande de chargement tente de traiter simultanément par seconde. S'il n'est pas défini, ce paramètre est désactivé avec -1.

executor.maxPerSecond Défini en fonction du nombre de WCU que vous avez provisionnés dans la table de destination cible. La *executor.maxPerSecond* load commande n'est pas une limite, c'est une moyenne cible. Cela signifie qu'il peut (et c'est souvent le cas) dépasser le nombre que vous avez défini. Pour tenir compte des rafales et vous assurer que la capacité est suffisante pour traiter les demandes de chargement de données, définissez 90 % *executor.maxPerSecond* de la capacité d'écriture de la table.

```
executor.maxPerSecond = WCUs * .90
```

Dans ce didacticiel, nous avons défini *executor.maxPerSecond* la valeur 5.

Note

Si vous utilisez DSBulk 1.6.0 ou une version ultérieure, vous pouvez utiliser *dsbulk.engine.maxConcurrentQueries* à la place.

- b. Configurez ces paramètres supplémentaires pour la load commande DSBulk.
 - *batch-mode*— Ce paramètre indique au système de regrouper les opérations par clé de partition. Nous vous recommandons de désactiver le mode batch, car cela peut entraîner des scénarios de raccourcis clavier et des causes *WriteThrottleEvents*.

- *driver.advanced.retry-policy-max-retries*— Cela détermine le nombre de tentatives à nouveau pour une requête qui a échoué. S'il n'est pas défini, la valeur par défaut est 10. Vous pouvez ajuster cette valeur selon vos besoins.
- *driver.basic.request.timeout*— Durée en minutes pendant laquelle le système attend le retour d'une requête. S'il n'est pas défini, la valeur par défaut est « 5 minutes ». Vous pouvez ajuster cette valeur selon vos besoins.

Étape 5 : Exécutez la commande de **chargement** DSBulk pour télécharger les données du fichier CSV vers la table cible

Dans la dernière étape de ce didacticiel, vous téléchargerez les données dans Amazon Keyspaces.

Pour exécuter la load commande DSBulk, procédez comme suit.

1. Exécutez le code suivant pour télécharger les données de votre fichier csv dans votre tableau Amazon Keyspaces. Assurez-vous de mettre à jour le chemin du fichier de configuration de l'application que vous avez créé précédemment.

```
dsbulk load -f ./dsbulk_keyspaces.conf --connector.csv.url keyspace.table.csv
--header true --batch.mode DISABLED --executor.maxPerSecond 5 --
driver.basic.request.timeout "5 minutes" --driver.advanced.retry-policy.max-
retries 10 -k catalog -t book_awards
```

2. La sortie inclut l'emplacement d'un fichier journal détaillant les opérations réussies et infructueuses. Le fichier est stocké dans le répertoire suivant.

```
Operation directory: /home/user_name/logs/UNLOAD_20210308-202317-801911
```

3. Les entrées du fichier journal incluront des métriques, comme dans l'exemple suivant. Vérifiez que le nombre de lignes correspond au nombre de lignes de votre fichier csv.

```
total | failed | rows/s | p50ms | p99ms | p999ms
200 | 0 | 200 | 21.63 | 21.89 | 21.89
```

Important

Maintenant que vous avez transféré vos données, ajustez les paramètres du mode capacité de votre table cible pour qu'ils correspondent aux modèles de trafic habituels de votre

application. Vous êtes facturé au taux horaire pour votre capacité provisionnée jusqu'à ce que vous la modifiiez. Pour de plus amples informations, veuillez consulter [the section called “Configuration des modes read/write de capacité”](#).

Accès à Amazon Keyspaces (pour Apache Cassandra)

Vous pouvez accéder à Amazon Keyspaces à l'aide de la console AWS CloudShell, par programmation en exécutant un `cqlsh` client, le AWS SDK, ou en utilisant un pilote Cassandra sous licence Apache 2.0. Amazon Keyspaces prend en charge les pilotes et les clients compatibles avec Apache Cassandra 3.11.2. Avant d'accéder à Amazon Keyspaces, vous devez terminer la configuration, Gestion des identités et des accès AWS puis accorder à une identité IAM les autorisations d'accès à Amazon Keyspaces.

Inscrivez-vous pour un Compte AWS

Pour commencer AWS, vous avez besoin d'un Compte AWS. Pour plus d'informations sur la création d'un Compte AWS, voir [Getting started with an Compte AWS](#) dans le Guide de Gestion de compte AWS référence.

Configuration d'Amazon Keyspaces

[L'accès aux ressources Amazon Keyspaces est géré à l'aide d'IAM.](#) À l'aide d'IAM, vous pouvez associer des politiques aux utilisateurs, aux rôles et aux identités fédérées IAM qui accordent des autorisations de lecture et d'écriture à des ressources spécifiques dans Amazon Keyspaces.

Pour commencer à accorder des autorisations à une identité IAM, vous pouvez utiliser l'une des politiques AWS gérées pour Amazon Keyspaces :

- [AmazonKeyspacesFullAccess](#)— cette politique accorde l'autorisation d'accéder à toutes les ressources d'Amazon Keyspaces avec un accès complet à toutes les fonctionnalités.
- [AmazonKeyspacesReadOnlyAccess_v2](#) : cette politique accorde des autorisations de lecture seule à Amazon Keyspaces.

Pour une explication détaillée des actions définies dans les politiques gérées, consultez [the section called “AWS politiques gérées”](#).

Pour limiter l'étendue des actions qu'une identité IAM peut effectuer ou limiter les ressources auxquelles l'identité peut accéder, vous pouvez créer une politique personnalisée qui utilise la stratégie `AmazonKeyspacesFullAccess` gérée comme modèle et supprimer toutes les autorisations dont vous n'avez pas besoin. Vous pouvez également limiter l'accès à des espaces clés ou à des tables spécifiques. Pour plus d'informations sur la façon de restreindre les actions ou de

limiter l'accès à des ressources spécifiques dans Amazon Keyspaces, consultez. [the section called “Comment Amazon Keyspaces fonctionne avec IAM”](#)

Pour accéder à Amazon Keyspaces après avoir créé Compte AWS et créé une politique qui accorde à une identité IAM l'accès à Amazon Keyspaces, passez à l'une des sections suivantes :

- [Utilisation de la console](#)
- [En utilisant AWS CloudShell](#)

Accès à Amazon Keyspaces à l'aide de la console

Vous pouvez accéder à la console d'Amazon Keyspaces à l'adresse. <https://console.aws.amazon.com/keyspaces/home> Pour plus d'informations sur Console de gestion AWS l'accès, consultez la section [Contrôle de l'accès des utilisateurs IAM au Console de gestion AWS dans le](#) guide de l'utilisateur IAM.

Vous pouvez utiliser la console pour effectuer les opérations suivantes dans Amazon Keyspaces :

- Créez, supprimez et gérez les espaces clés et les tables.
- Surveillez les statistiques importantes d'un tableau dans l'onglet Surveiller d'un tableau :
 - Taille du tableau facturable (octets)
 - Indicateurs de capacité
- Exécutez des requêtes à l'aide de l'éditeur CQL ou de l'[AWS CloudShell expérience intégrée](#), par exemple pour insérer, mettre à jour et supprimer des données.
- Connectez-vous directement à Amazon Keyspaces grâce au support AWS CloudShell CQLSH intégré.
- Modifiez la configuration du partitionneur du compte.
- Consultez les indicateurs de performance et d'erreur du compte sur le tableau de bord.

Connectez-vous à Amazon Keyspaces en utilisant AWS CloudShell depuis la console

AWS CloudShell fournit un moyen simplifié de se connecter à Amazon Keyspaces directement depuis la console. L' AWS CloudShell intégration initialise automatiquement le `cqlsh-expansion` pour vous et établit une connexion à un espace de touches spécifique.

Démarrage avec AWS CloudShell intégration

1. Accédez à l' AWS CloudShell intégration : depuis la console Amazon Keyspaces, accédez à l'espace de touches auquel vous souhaitez vous connecter et cliquez sur le bouton Connect to Keyspaces. AWS CloudShell
2. Pre-filled commande de configuration : lorsque vous cliquez AWS CloudShell sur le bouton, le système fournit une commande préremplie qui inclut les paramètres d'initialisation et de connexion nécessaires :

```
cqlsh-expansion.init ; cqlsh-expansion cassandra.region.amazonaws.com 9142 --ssl  
-my_keyspace
```

3. Cliquez sur Exécuter
4. Attendez qu'Amazon Keyspaces établisse la connexion : attendez que le processus de configuration soit terminé et que la connexion à l'espace de touches que vous avez sélectionné soit établie.
5. Exécuter des commandes — Une fois connecté, vous pouvez écrire et exécuter des commandes CQL directement sur l'espace de touches sélectionné.

Commandes CQLSH disponibles

L' AWS CloudShell intégration donne accès aux fonctionnalités CQLSH standard, y compris la prise en charge de commandes autres que le CQL de base.

- Décrire — Répertorier et afficher les descriptions détaillées des espaces clés et des tables.
- Copier : exporte le résultat de la requête vers un fichier pour un traitement ultérieur.
- Source — Exécute des instructions CQL à partir d'un fichier.
- Navigation dans l'historique des commandes : parcourez les commandes précédentes à l'aide des raccourcis clavier.
- Exécution des requêtes : exécutez des requêtes directement depuis l'interface du clavier.
- Copie de sortie : copiez les résultats sous forme de texte simple pour les utiliser dans d'autres applications.

Considérations relatives aux fonctionnalités

Lorsque vous utilisez l' AWS CloudShell intégration, tenez compte des compromis suivants.

Les fonctionnalités suivantes ne sont pas prises en charge :

- Suggestions de syntaxe CQL et fonctionnalité de saisie automatique.
- Des outils d'édition tels que l'affichage de la saisie semi-automatique, l'affichage de l'infobulle, la recherche, la recherche et le remplacement, le rétablissement, l'activation des blocs de commentaires, le pliage du code, l'inversion des commentaires sur les lignes et l'annulation.

Fonctionnalité modifiée :

- Téléchargements CSV : pour télécharger les résultats au format CSV, spécifiez l'>opérateur qui dirigera la sortie de la commande :

```
SELECT * FROM my_table > file_name.csv
```

- Affichage JSON - Vous pouvez afficher les éléments au format JSON sans nécessiter de conversion de données.
- Terminal-style interface - L'expérience fonctionne comme une interface de terminal plutôt que comme un éditeur de code complet.

L' AWS CloudShell intégration simplifie le processus de connexion tout en fournissant des fonctionnalités CQL essentielles pour gérer vos données Amazon Keyspaces directement depuis le. Console de gestion AWS

Pour savoir comment créer un espace de touches et une table Amazon Keyspaces et comment les configurer à l'aide d'exemples de données d'application, consultez. [Démarrez avec Amazon Keyspaces \(pour Apache Cassandra\)](#)

Utilisation AWS CloudShell pour accéder à Amazon Keyspaces

AWS CloudShell est un shell pré-authentifié basé sur un navigateur que vous pouvez lancer directement depuis le. Console de gestion AWS Vous pouvez exécuter AWS CLI des commandes sur AWS des services à l'aide de votre shell préféré (Bash PowerShell ou Z shell). AWS CloudShell est pré-authentifié avec les informations d'identification de votre console et inclut des outils courants pour travailler avec les AWS services, notamment AWS CLI Python et. Node.js

Vous [lancez AWS CloudShell à partir de Console de gestion AWS](#), et les AWS informations d'identification que vous avez utilisées pour vous connecter à la console sont automatiquement

disponibles dans une nouvelle session shell. Cette pré-authentification des AWS CloudShell utilisateurs vous permet d'ignorer la configuration des informations d'identification lorsque vous interagissez avec AWS des services tels qu'Amazon Keyspaces `cqlsh` en utilisant AWS CLI la version 2 (préinstallée sur l'environnement informatique du shell).

AWS CloudShell options d'intégration

Amazon Keyspaces propose deux méthodes d'utilisation : AWS CloudShell

1. AWS CloudShell Expérience intégrée : méthode de connexion rationalisée disponible directement depuis la console Amazon Keyspaces qui initialise `cqlsh-expansion` et se connecte automatiquement à un espace de touches spécifique. Pour obtenir des instructions complètes sur l'utilisation de cette expérience intégrée, consultez [the section called “CloudShell intégration à la console”](#).
2. AWS CloudShell Configuration manuelle : installez et configurez manuellement `cqlsh` dans AWS CloudShell. Les étapes sont décrites dans les sections suivantes et permettent de mieux contrôler le processus de configuration et d'activer des configurations personnalisées.

Conditions préalables

Avant d'utiliser Amazon Keyspaces avec AWS CloudShell, assurez-vous d'avoir :

- Un AWS compte avec accès à Amazon Keyspaces et AWS CloudShell
- Autorisations IAM appropriées pour utiliser les deux services
- Connaissance de base du langage de requête Cassandra (CQL)

Commencer avec AWS CloudShell

Pour accéder à AWS CloudShell :

1. Connectez-vous au Console de gestion AWS
2. Lancez en AWS CloudShell choisissant l' AWS CloudShell icône dans la barre de navigation en haut de la console ou en accédant à <https://console.aws.amazon.com/cloudshell/>
3. Attendez que l' AWS CloudShell environnement s'initialise

Obtention des autorisations IAM pour AWS CloudShell

À l'aide des ressources de gestion des accès fournies par Gestion des identités et des accès AWS, les administrateurs peuvent accorder des autorisations aux utilisateurs IAM afin qu'ils puissent accéder aux fonctionnalités de l'environnement AWS CloudShell et les utiliser.

Le moyen le plus rapide pour un administrateur d'accorder l'accès aux utilisateurs est d'utiliser une politique AWS gérée. Une [politique gérée par AWS](#) est une politique autonome qui est créée et gérée par AWS. La politique AWS gérée suivante pour CloudShell peut être attachée aux identités IAM :

- `AWSCloudShellFullAccess`: accorde l'autorisation d'utilisation AWS CloudShell avec un accès complet à toutes les fonctionnalités.

Si vous souhaitez limiter l'étendue des actions qu'un utilisateur IAM peut effectuer AWS CloudShell, vous pouvez créer une politique personnalisée qui utilise la stratégie `AWSCloudShellFullAccess` gérée comme modèle. Pour plus d'informations sur la limitation des actions disponibles pour les utilisateurs dans CloudShell, consultez la section [Gestion de l' AWS CloudShell accès et de l'utilisation avec les politiques IAM](#) dans le Guide de l'AWS CloudShell utilisateur.

Note

Votre identité IAM nécessite également une politique autorisant les appels vers Amazon Keyspaces.

Vous pouvez utiliser une politique AWS gérée pour donner à votre identité IAM l'accès à Amazon Keyspaces, ou commencer par utiliser la politique gérée comme modèle et supprimer les autorisations dont vous n'avez pas besoin. Vous pouvez également limiter l'accès à des espaces clés et à des tables spécifiques afin de créer une politique personnalisée. La politique gérée suivante pour Amazon Keyspaces peut être associée aux identités IAM :

- [AmazonKeyspacesFullAccess](#)— Cette politique autorise l'utilisation d'Amazon Keyspaces avec un accès complet à toutes les fonctionnalités.

Pour une explication détaillée des actions définies dans la politique gérée, consultez [the section called “AWS politiques gérées”](#).

Pour plus d'informations sur la façon de restreindre les actions ou de limiter l'accès à des ressources spécifiques dans Amazon Keyspaces, consultez [the section called “Comment Amazon Keyspaces fonctionne avec IAM”](#)

Installation du client cqlsh

AWS CloudShell n'est pas livré avec une `cqlsh` installation préinstallée. Vous devez l'installer pour interagir avec Amazon Keyspaces. Vous pouvez installer la Keyspaces-specific version Amazon `decqlsh`. Pour les instructions d'`cqlsh`-expansion installation, voir [the section called “Utilisation de l'`cqlsh`-expansion”](#).

Configuration des certificats TLS

Pour vous connecter à Amazon Keyspaces via une connexion TLS, vous devez télécharger les certificats numériques Amazon requis. Pour obtenir des instructions détaillées sur le téléchargement et la configuration du certificat SSL, consultez [the section called “Comment configurer manuellement les `cqlsh` connexions pour le protocole TLS”](#).

Connexion à Amazon Keyspaces

Vous pouvez vous connecter à Amazon Keyspaces à l'aide d'informations d'identification IAM (recommandées) ou d'informations d'identification spécifiques au service.

Connexion à l'aide des informations d'identification IAM (recommandé)

AWS CloudShell est déjà authentifié avec vos informations d'identification IAM. Pour vous connecter à l'aide de ces informations d'identification, vous pouvez utiliser le plugin d'authentification SigV4. Pour obtenir des instructions détaillées sur l'utilisation des informations d'identification IAM avec `cqlsh`, consultez [the section called “Utilisation de l'`cqlsh`-expansion”](#).

Connexion à l'aide d'informations d'identification spécifiques au service

Si vous avez créé des informations d'identification spécifiques au service pour Amazon Keyspaces, vous pouvez les utiliser pour vous connecter. Pour plus d'informations sur la création et l'utilisation d'informations d'identification spécifiques au service, consultez [the section called “Créez des informations d'identification spécifiques au service”](#)

Utilisation de CQLSH avec Amazon Keyspaces

Lorsque vous êtes connecté, vous pouvez utiliser les commandes CQL standard pour interagir avec vos espaces de touches et vos tables. Pour des informations détaillées sur les commandes et opérations CQL prises en charge, consultez [the section called “Utilisation de l’cqlsh-expansion”](#).

Données persistantes dans AWS CloudShell

AWS CloudShell fournit 1 Go de stockage persistant dans votre répertoire personnel. Cela signifie que les scripts, les certificats et les fichiers de configuration que vous créez restent disponibles d'une session à l'autre. Vous pouvez créer un fichier de configuration `cqlsh` pour simplifier les connexions.

Avec un fichier de configuration, vous pouvez exécuter `cqlsh` sans spécifier de paramètres supplémentaires. Pour plus d'informations sur la création et l'utilisation `cqlsh` de fichiers de configuration, consultez [the section called “Utilisation de l’cqlsh-expansion”](#).

Bonnes pratiques

Lorsque vous utilisez Amazon Keyspaces avec AWS CloudShell, nous vous recommandons de suivre les meilleures pratiques suivantes :

- Créez des scripts pour les tâches répétitives et stockez-les dans votre répertoire personnel permanent.
- Utilisez des variables d'environnement pour les informations sensibles plutôt que de les coder en dur.
- N'oubliez pas que les AWS CloudShell sessions peuvent être interrompues en cas d'inactivité.
- Tenez compte des quotas et des limites de AWS CloudShell service.
- Envisagez d'utiliser le AWS CLI pour automatiser les opérations avec Amazon Keyspaces.

Résolution des problèmes

Si vous rencontrez des problèmes lors de la connexion à Amazon Keyspaces depuis : AWS CloudShell

- Vérifiez que vos autorisations IAM incluent les actions Amazon Keyspaces nécessaires.
- Assurez-vous que vous utilisez le bon point de terminaison pour votre région.

- Vérifiez que les certificats SSL sont correctement téléchargés et référencés.
- Vérifiez que vos informations d'identification spécifiques au service sont correctes et qu'elles n'ont pas expiré.
- Si vous utilisez un fichier `cqlshrc`, vérifiez son formatage et ses autorisations.

Exemple : Comment interagir avec Amazon Keyspaces en utilisant AWS CloudShell

Après le lancement AWS CloudShell depuis le Console de gestion AWS, vous pouvez immédiatement commencer à interagir avec Amazon Keyspaces à l'aide `cqlsh` de l'interface de ligne de commande. Si vous ne l'avez pas encore installé `cqlsh-expansion`, consultez [the section called “Utilisation de l’cqlsh-expansion”](#) les étapes détaillées.

Note

Lorsque vous utilisez l'`cqlsh-expansion` entrée AWS CloudShell, vous n'avez pas besoin de configurer les informations d'identification avant de passer des appels, car vous êtes déjà authentifié dans le shell.

Connectez-vous à Amazon Keyspaces et créez un nouvel espace de touches. Lisez ensuite une table système pour confirmer que le keyspace a été créé à l'aide de AWS CloudShell

1. À partir de Console de gestion AWS, vous pouvez lancer CloudShell en choisissant les options suivantes disponibles dans la barre de navigation :
 - Choisissez l' CloudShell icône.
 - Commencez à taper `cloudshell` dans le champ de recherche, puis choisissez l' CloudShell option.
2. Vous pouvez établir une connexion à Amazon Keyspaces à l'aide de la commande suivante. Assurez-vous de le remplacer `cassandra.us-east-1.amazonaws.com` par le point de terminaison adapté à votre région.

```
cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 --ssl
```

Si la connexion est établie, vous devriez obtenir un résultat similaire à celui de l'exemple suivant.

```
Connected to Amazon Keyspaces at cassandra.us-east-1.amazonaws.com:9142
[cqlsh 6.1.0 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh current consistency level is ONE.
cqlsh>
```

3. Créez un nouvel espace de touches portant le nom `mykeyspace`. Pour ce faire, vous pouvez utiliser la commande suivante.

```
CREATE KEYSPACE mykeyspace WITH REPLICATION = {'class': 'SingleRegionStrategy'};
```

4. Pour confirmer que le keyspace a été créé, vous pouvez lire une table système à l'aide de la commande suivante.

```
SELECT * FROM system_schema_mcs.keyspaces WHERE keyspace_name = 'mykeyspace';
```

Si l'appel aboutit, la ligne de commande affiche une réponse du service similaire au résultat suivant :

```
keyspace_name | durable_writes | replication
-----+-----
+-----+-----+-----
mykeyspace    | True           | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}

(1 rows)
```

Créez des informations d'identification pour un accès programmatique à Amazon Keyspaces

Pour fournir aux utilisateurs et aux applications des informations d'identification leur permettant d'accéder par programmation aux ressources Amazon Keyspaces, vous pouvez effectuer l'une des opérations suivantes :

- Créez des informations d'identification spécifiques au service similaires au nom d'utilisateur et au mot de passe traditionnels que Cassandra utilise pour l'authentification et la gestion des accès. AWS les informations d'identification spécifiques au service sont associées à un utilisateur Gestion

des identités et des accès AWS (IAM) spécifique et ne peuvent être utilisées que pour le service pour lequel elles ont été créées. Pour plus d'informations, consultez la section [Utilisation d'IAM avec Amazon Keyspaces \(pour Apache Cassandra\)](#) dans le guide de l'utilisateur d'IAM.

 Warning

Les utilisateurs IAM disposent d'informations d'identification à long terme, ce qui présente un risque de sécurité. Pour atténuer ce risque, nous vous recommandons de ne fournir à ces utilisateurs que les autorisations dont ils ont besoin pour effectuer la tâche et de supprimer ces autorisations lorsqu'elles ne sont plus nécessaires.

- Pour renforcer la sécurité, nous vous recommandons de créer des identités IAM utilisées dans tous les AWS services et d'utiliser des informations d'identification temporaires. Le plugin d'authentification Amazon Keyspaces SigV4 pour les pilotes clients Cassandra vous permet d'authentifier les appels vers Amazon Keyspaces à l'aide de clés d'accès IAM plutôt que de votre nom d'utilisateur et de votre mot de passe. Pour en savoir plus sur la façon dont le plugin Amazon Keyspaces SigV4 permet aux [utilisateurs, aux rôles et aux identités fédérées d'IAM de s'authentifier](#) dans les demandes d'API Amazon Keyspaces, [AWS consultez](#) le processus Signature Version 4 (SigV4).

Vous pouvez télécharger les plug-ins SigV4 depuis les emplacements suivants.

- Java : <https://github.com/aws/aws-sigv4-auth-cassandra-java-driver-plugin>.
- Node.js: <https://github.com/aws/aws-sigv4-auth-cassandra-nodejs-driver-plugin>.
- Python: <https://github.com/aws/aws-sigv4-auth-cassandra-python-driver-plugin>.
- Allez : <https://github.com/aws/aws-sigv4-auth-cassandra-gocql-driver-plugin>.

Pour des exemples de code qui montrent comment établir des connexions à l'aide du plug-in d'authentification SigV4, voir [the section called “Utilisation d'un pilote client Cassandra”](#).

Rubriques

- [Créez des informations d'identification spécifiques au service pour un accès programmatique à Amazon Keyspaces](#)
- [Création et configuration des AWS informations d'identification pour Amazon Keyspaces](#)

Créez des informations d'identification spécifiques au service pour un accès programmatique à Amazon Keyspaces

Service-specific les informations d'identification sont similaires au nom d'utilisateur et au mot de passe traditionnels que Cassandra utilise pour l'authentification et la gestion des accès. Service-specific les informations d'identification permettent aux utilisateurs IAM d'accéder à un AWS service spécifique. Ces informations d'identification à long terme ne peuvent pas être utilisées pour accéder à d'autres AWS services. Ils sont associés à un utilisateur IAM spécifique et ne peuvent pas être utilisés par d'autres utilisateurs IAM.

Important

Service-specific les informations d'identification sont des informations d'identification à long terme associées à un utilisateur IAM spécifique et ne peuvent être utilisées que pour le service pour lequel elles ont été créées. Pour autoriser les rôles IAM ou les identités fédérées à accéder à toutes vos AWS ressources à l'aide d'informations d'identification temporaires, vous devez utiliser l'[AWS authentication avec le plugin d'authentification SigV4 pour Amazon Keyspaces](#).

Utilisez l'une des procédures suivantes pour générer des informations d'identification spécifiques au service.

Console

Créez des informations d'identification spécifiques au service à l'aide de la console

1. Connectez-vous à la Gestion des identités et des accès AWS console Console de gestion AWS et ouvrez-la à l'adresse <https://console.aws.amazon.com/iam/home>.
2. Dans le volet de navigation, choisissez Users, puis choisissez l'utilisateur que vous avez créé précédemment et qui dispose des autorisations Amazon Keyspaces (politique jointe).
3. Choisissez Informations d'identification de sécurité. Sous Credentials for Amazon Keyspaces, choisissez Generate credentials pour générer les identifiants spécifiques au service.

Vos informations d'identification spécifiques au service sont disponibles. C'est la seule fois que vous pouvez télécharger ou afficher le mot de passe. Vous ne pourrez pas la récupérer plus tard. Cependant, vous pouvez réinitialiser votre mot de passe à tout moment.

Enregistrez cet utilisateur et ce mot de passe dans un emplacement sécurisé, car vous en aurez besoin plus tard.

CLI

Créez des informations d'identification spécifiques au service à l'aide du AWS CLI

Avant de générer des informations d'identification spécifiques au service, vous devez télécharger, installer et configurer le AWS Command Line Interface (AWS CLI) :

1. Téléchargez le site AWS CLI à <http://aws.amazon.com/cli>.

Note

Il AWS CLI fonctionne sous Windows, macOS ou Linux.

2. Suivez les instructions d'[installation de la AWS CLI](#) et de [configuration de la AWS CLI](#) dans le guide de AWS Command Line Interface l'utilisateur.
3. À l'aide de AWS CLI, exécutez la commande suivante pour générer des informations d'identification spécifiques au service pour l'utilisateur `alice`, afin qu'elle puisse accéder à Amazon Keyspaces.

```
aws iam create-service-specific-credential \  
  --user-name alice \  
  --service-name cassandra.amazonaws.com
```

Le résultat se présente comme suit.

```
{  
  "ServiceSpecificCredential": {  
    "CreateDate": "2019-10-09T16:12:04Z",  
    "ServiceName": "cassandra.amazonaws.com",  
    "ServiceUserName": "alice-at-111122223333",  
    "ServicePassword": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",  
    "ServiceSpecificCredentialId": "ACCAYFI33SINPGJEBYESF",  
    "UserName": "alice",  
    "Status": "Active"  
  }  
}
```

Dans la sortie, notez les valeurs pour `ServiceUserName` et `ServicePassword`. Enregistrez ces valeurs dans un emplacement sécurisé, car vous en aurez besoin plus tard.

 Important

C'est la seule fois que le `ServicePassword` sera disponible pour vous.

Création et configuration des AWS informations d'identification pour Amazon Keyspaces

Pour accéder à Amazon Keyspaces par programmation à l'aide du AWS SDK ou à l'AWS CLI aide des pilotes clients Cassandra et du plugin SigV4, vous avez besoin d'un utilisateur IAM doté de clés d'accès. Lorsque vous utilisez Amazon Keyspaces par programmation, vous fournissez vos clés d'AWS accès afin de vérifier votre identité lors des AWS appels programmatiques. Vos clés d'accès se composent d'un identifiant de clé d'accès (par exemple, `AKIAIOSFODNN7EXAMPLE`) et d'une clé d'accès secrète (par exemple, `/b CYEXAMPLEKEY`). `wJalrXUtnFEMI/K7MDENG PxrFi` Cette rubrique décrit les étapes nécessaires à ce processus.

Les meilleures pratiques en matière de sécurité recommandent de créer des utilisateurs IAM dotés d'autorisations limitées et d'associer les rôles IAM aux autorisations nécessaires pour effectuer des tâches spécifiques. Les utilisateurs IAM peuvent ensuite assumer temporairement des rôles IAM pour effectuer les tâches requises. Par exemple, les utilisateurs IAM de votre compte utilisant la console Amazon Keyspaces peuvent passer à un rôle afin d'utiliser temporairement les autorisations du rôle dans la console. Les utilisateurs abandonnent leurs autorisations d'origine et acceptent les autorisations attribuées au rôle. Lorsque les utilisateurs quittent le rôle, leurs autorisations d'origine sont restaurées. Les informations d'identification utilisées par les utilisateurs pour assumer le rôle sont temporaires. Au contraire, les utilisateurs IAM disposent d'informations d'identification à long terme, ce qui présente un risque de sécurité si, au lieu d'assumer des rôles, des autorisations leur sont directement attribuées. Pour atténuer ce risque, nous vous recommandons de ne fournir à ces utilisateurs que les autorisations dont ils ont besoin pour effectuer la tâche et de supprimer ces autorisations lorsqu'elles ne sont plus nécessaires. Pour plus d'informations sur les rôles, consultez la section [Scénarios courants pour les rôles : utilisateurs, applications et services](#) dans le guide de l'utilisateur IAM.

Rubriques

- [Informations d'identification requises par AWS CLI le AWS SDK ou le plugin Amazon Keyspaces SigV4 pour les pilotes clients Cassandra](#)
- [Créez des informations d'identification temporaires pour vous connecter à Amazon Keyspaces à l'aide d'un rôle IAM et du plug-in SigV4](#)
- [Créez un utilisateur IAM pour un accès programmatique à Amazon Keyspaces dans votre compte AWS](#)
- [Création de nouvelles clés d'accès pour un utilisateur IAM](#)
- [Stocker les clés d'accès pour un accès programmatique](#)

Informations d'identification requises par AWS CLI le AWS SDK ou le plugin Amazon Keyspaces SigV4 pour les pilotes clients Cassandra

Les informations d'identification suivantes sont requises pour authentifier l'utilisateur ou le rôle IAM :

`AWS_ACCESS_KEY_ID`

Spécifie une clé AWS d'accès associée à un utilisateur ou à un rôle IAM.

La clé d'accès `aws_access_key_id` est requise pour se connecter à Amazon Keyspaces par programmation.

`AWS_SECRET_ACCESS_KEY`

Indique la clé secrète associée à la clé d'accès. Il s'agit du « mot de passe » de la clé d'accès.

`aws_secret_access_key` est nécessaire pour se connecter à Amazon Keyspaces par programmation.

`AWS_SESSION_TOKEN`— Facultatif

Spécifie la valeur du jeton de session requise si vous utilisez des informations d'identification de sécurité temporaires que vous avez extraites directement des opérations AWS Security Token Service . Pour de plus amples informations, veuillez consulter [the section called “Créez des informations d'identification temporaires pour vous connecter à Amazon Keyspaces”](#).

Si vous vous connectez avec un utilisateur IAM, ce `aws_session_token` n'est pas obligatoire.

Créez des informations d'identification temporaires pour vous connecter à Amazon Keyspaces à l'aide d'un rôle IAM et du plug-in SigV4

La méthode recommandée pour accéder à Amazon Keyspaces par programmation consiste à utiliser des informations d'[identification temporaires](#) pour s'authentifier auprès du plugin SigV4. Dans de nombreux cas, vous n'avez pas besoin d'une clé d'accès à long terme qui n'expire jamais (comme dans le cas d'un utilisateur IAM). Vous pouvez plutôt créer un rôle IAM et générer des informations d'identification de sécurité temporaires. Les informations d'identification de sécurité temporaires consistent en un ID de clé d'accès et une clé d'accès secrète, mais elles comprennent également un jeton de sécurité qui indique la date d'expiration des informations d'identification. Pour en savoir plus sur l'utilisation des rôles IAM plutôt que des clés d'accès à long terme, consultez la section [Passer à un rôle IAM \(AWS API\)](#).

Pour commencer à utiliser des informations d'identification temporaires, vous devez d'abord créer un rôle IAM.

Créez un rôle IAM qui accorde un accès en lecture seule à Amazon Keyspaces

1. Connectez-vous à la console IAM Console de gestion AWS et ouvrez-la à <https://console.aws.amazon.com/iam/> l'adresse.
2. Dans le volet de navigation, choisissez Rôles, puis Créer un rôle.
3. Sur la page Créer un rôle, sous Sélectionner le type d'entité de confiance, sélectionnez AWS service. Sous Choisir un cas d'utilisation, choisissez Amazon EC2, puis Next.
4. Sur la page Ajouter des autorisations, sous Politiques d'autorisations, choisissez Amazon Keyspaces Read Only Access dans la liste des politiques, puis choisissez Next.
5. Sur la page Nom, révision et création, entrez le nom du rôle et consultez les sections Sélectionner des entités fiables et Ajouter des autorisations. Vous pouvez également ajouter des balises facultatives pour le rôle sur cette page. Lorsque vous avez terminé, sélectionnez Créer un rôle. N'oubliez pas ce nom car vous en aurez besoin lorsque vous lancerez votre instance Amazon EC2.

Pour utiliser des informations d'identification de sécurité temporaires dans le code, vous devez appeler par programmation une AWS Security Token Service API similaire `AssumeRole` et extraire les informations d'identification et le jeton de session qui en résultent de votre rôle IAM que vous avez créé à l'étape précédente. Vous utilisez ensuite ces valeurs comme informations d'identification pour les appels suivants à AWS. L'exemple suivant montre un pseudocode expliquant comment utiliser les informations d'identification de sécurité temporaires :

```
assumeRoleResult = AssumeRole(role-arn);
tempCredentials = new SessionAWSCredentials(
    assumeRoleResult.AccessKeyId,
    assumeRoleResult.SecretAccessKey,
    assumeRoleResult.SessionToken);
cassandraRequest = CreateAmazoncassandraClient(tempCredentials);
```

Pour un exemple qui implémente des informations d'identification temporaires à l'aide du pilote Python pour accéder à Amazon Keyspaces, consultez. [???](#)

Pour plus d'informations sur la façon d'appeler `AssumeRole`, `GetFederationToken` et d'autres opérations d'API, consultez la [Référence sur l'API AWS Security Token Service](#). Pour plus d'informations sur l'obtention des informations d'identification de sécurité temporaires et du jeton de session à partir des résultats, consultez la documentation du kit SDK que vous utilisez. Vous trouverez la documentation de tous les AWS SDK sur la [page de AWS documentation](#) principale, dans la section SDK et boîtes à outils.

Créez un utilisateur IAM pour un accès programmatique à Amazon Keyspaces dans votre compte AWS

Pour obtenir les informations d'identification nécessaires à l'accès programmatique à Amazon Keyspaces avec AWS CLI le plugin, AWS le SDK ou le plugin SigV4, vous devez d'abord créer un utilisateur ou un rôle IAM. Le processus de création d'un utilisateur IAM et de configuration de cet utilisateur IAM pour qu'il dispose d'un accès programmatique à Amazon Keyspaces est illustré dans les étapes suivantes :

1. Créez l'utilisateur dans Console de gestion AWS les AWS CLI outils pour Windows PowerShell ou à l'aide d'une opération d' AWS API. Si vous créez l'utilisateur dans le Console de gestion AWS, les informations d'identification sont créées automatiquement.
2. Si vous créez l'utilisateur par programmation, vous devez créer une clé d'accès (ID de clé d'accès et clé d'accès secrète) pour cet utilisateur lors d'une étape supplémentaire.
3. Donnez à l'utilisateur l'autorisation d'accéder à Amazon Keyspaces.

Pour plus d'informations sur les autorisations dont vous avez besoin pour créer un utilisateur IAM, consultez la section [Autorisations requises pour accéder aux ressources IAM](#).

Console

Création d'un utilisateur IAM avec accès par programmation (console)

1. Connectez-vous à la console IAM Console de gestion AWS et ouvrez-la à <https://console.aws.amazon.com/iam/> l'adresse.
2. Dans le panneau de navigation, sélectionnez Users (Utilisateurs), puis Add users (Ajouter des utilisateurs).
3. Tapez le nom d'utilisateur du nouvel utilisateur. Il s'agit du nom de connexion pour AWS.

Note

Les noms d'utilisateur peuvent combiner jusqu'à 64 lettres, chiffres et caractères suivants : plus (+), égal (=), virgule (,), point (.), arobase (@), trait de soulignement (_) et tiret (-). Les noms doivent être uniques au sein d'un compte. Ils ne sont pas distingués au cas par cas. Par exemple, vous ne pouvez pas créer deux utilisateurs nommés TESTUSER et testuser.

4. Sélectionnez Clé d'accès - Accès par programmation pour créer une clé d'accès pour le nouvel utilisateur. Vous pouvez consulter ou télécharger la clé d'accès lorsque vous arrivez sur la page finale.

Choisissez Suivant : Autorisations.

5. Sur la page Définir les autorisations, choisissez Joindre directement les politiques existantes pour attribuer des autorisations au nouvel utilisateur.

Cette option affiche la liste des politiques AWS gérées et gérées par le client disponibles dans votre compte. Vous pouvez entrer dans le champ de recherche pour afficher uniquement les politiques relatives à Amazon Keyspaces.

Pour Amazon Keyspaces, les politiques gérées disponibles sont AmazonKeyspacesFullAccess et AmazonKeyspacesReadOnlyAccess. Pour plus d'informations sur chaque politique, consultez [the section called "AWS politiques gérées"](#).

À des fins de test et pour suivre les didacticiels de connexion, sélectionnez la AmazonKeyspacesReadOnlyAccess politique pour le nouvel utilisateur IAM. Remarque : En tant que bonne pratique, nous vous recommandons de suivre le principe du moindre privilège et de créer des politiques personnalisées qui limitent l'accès à des ressources

spécifiques et n'autorisent que les actions requises. Pour plus d'informations sur les politiques IAM et pour consulter des exemples de politiques pour Amazon Keyspaces, consultez. [the section called “Politiques basées sur l'identité d'Amazon Keyspaces”](#) Après avoir créé des politiques d'autorisation personnalisées, associez vos politiques aux rôles, puis laissez les utilisateurs assumer temporairement les rôles appropriés.

Choisissez Suivant : Balises.

6. Sur la page Ajouter des balises (facultatif), vous pouvez ajouter des balises pour l'utilisateur ou choisir Suivant : Réviser.
7. Sur la page Révision, vous pouvez voir tous les choix que vous avez faits jusqu'à présent. Lorsque vous êtes prêt à continuer, choisissez Create user.
8. Pour afficher les clés d'accès de l'utilisateur (ID de clé d'accès et clés d'accès secrètes), choisissez Afficher en regard du mot de passe et de la clé d'accès. Pour enregistrer les clés d'accès, choisissez télécharger .csv, puis enregistrez le fichier dans un emplacement sûr sur votre ordinateur.

 Important

Il s'agit de votre seule opportunité de visualiser ou de télécharger les clés d'accès secrètes, et vous avez besoin de ces informations pour qu'ils puissent utiliser le plugin SigV4. Enregistrez les nouveaux ID de clé d'accès et clé d'accès secrète de l'utilisateur dans un endroit sûr et sécurisé. Vous ne pourrez plus accéder aux clés d'accès secrètes après cette étape.

CLI

Création d'un utilisateur IAM avec accès par programmation ()AWS CLI

1. Créez un utilisateur avec le AWS CLI code suivant.
 - [aws iam create-user](#)
2. Donnez à l'utilisateur un accès programmatique. Cela nécessite des clés d'accès, qui peuvent être générées de la manière suivante.
 - AWS CLI: [aws iam create-access-key](#)
 - Outils pour Windows PowerShell : [New-IAMAccessKey](#)

- API IAM : [CreateAccessKey](#)

 Important

Il s'agit de votre seule opportunité de visualiser ou de télécharger les clés d'accès secrètes, et vous avez besoin de ces informations pour qu'ils puissent utiliser le plugin SigV4. Enregistrez les nouveaux ID de clé d'accès et clé d'accès secrète de l'utilisateur dans un endroit sûr et sécurisé. Vous ne pourrez plus accéder aux clés d'accès secrètes après cette étape.

3. Attachez à l'utilisateur la `AmazonKeyspacesReadOnlyAccess` politique qui définit les autorisations de l'utilisateur. Remarque : il est recommandé de gérer les autorisations des utilisateurs en ajoutant l'utilisateur à un groupe et en attachant une politique au groupe au lieu de l'associer directement à un utilisateur.

- AWS CLI: [aws iam attach-user-policy](#)

Création de nouvelles clés d'accès pour un utilisateur IAM

Si vous avez déjà un utilisateur IAM, vous pouvez créer de nouvelles clés d'accès à tout moment. Pour plus d'informations sur la gestion des clés, par exemple comment mettre à jour les clés d'accès, consultez [la section Gestion des clés d'accès pour les utilisateurs IAM](#).

Pour créer des clés d'accès pour un utilisateur IAM (console)

1. Connectez-vous à la console IAM Console de gestion AWS et ouvrez-la à <https://console.aws.amazon.com/iam/> l'adresse.
2. Dans le panneau de navigation, choisissez utilisateurs.
3. Choisissez le nom de l'utilisateur dont vous souhaitez créer les clés d'accès.
4. Sur la page Résumé de l'utilisateur, choisissez l'onglet Informations d'identification de sécurité.
5. Dans la section Clés d'accès, sous Meilleures pratiques et alternatives en matière de clés d'accès, sélectionnez le cas d'utilisation Autre. Cliquez sur Suivant, entrez les informations facultatives nécessaires, puis choisissez Créer une clé d'accès.

Pour afficher la nouvelle clé d'accès, choisissez Show (Afficher). Vos informations d'identification s'afficheront comme suit :

- ID de clé d'accès : AKIAIOSFODNN7EXAMPLE

- Clé d'accès secrète : wJalrXUtnFEMI/K7MDENG /b PxRfi CYEXAMPLEKEY

Note

Vous ne pourrez pas accéder à la clé d'accès secrète à nouveau une fois que cette boîte de dialogue se sera fermée.

Tenez compte des bonnes pratiques suivantes pour la paire de clés que vous avez créée.

- Ne stockez jamais votre clé d'accès en texte brut, dans un référentiel de code ou dans du code.
 - Désactivez ou supprimez les clés d'accès lorsqu'elles ne sont plus nécessaires.
 - Activez les autorisations du moindre privilège.
 - Faites régulièrement pivoter les touches d'accès.
6. Pour télécharger la paire de clés, choisissez Télécharger le fichier .csv. Stockez les clés dans un emplacement sûr.
 7. Après avoir téléchargé le fichier CSV, sélectionnez Close (Fermer).

Lorsque vous créez une clé d'accès, la paire de clés est activée par défaut et vous pouvez utiliser la paire immédiatement.

Stocker les clés d'accès pour un accès programmatique

Il est recommandé de ne pas intégrer les clés d'accès directement dans le code. Les AWS SDK et les outils de ligne de AWS commande vous permettent de placer les clés d'accès dans des emplacements connus afin de ne pas avoir à les conserver dans le code. Placez les clés d'accès à l'un des emplacements suivants :

- Variables d'environnement : sur un système mutualisé, choisissez les variables d'environnement utilisateur et non les variables d'environnement système.
- Fichier d'informations d'identification CLI — Le config fichier `credentials` est mis à jour lorsque vous exécutez la commande `aws configure`. Le `credentials` fichier se trouve sous `~/.aws/credentials` Linux, macOS ou Unix, ou `C:\Users\USERNAME\.aws\credentials` sous Windows. Ce fichier contient les informations d'identification du profil `default` et de tous les profils nommés.

- Fichier de configuration CLI — Le config fichier `credentials` est mis à jour lorsque vous exécutez la commande `aws configure`. Le config fichier se trouve sous `~/.aws/config` Linux, macOS ou Unix, ou `C:\Users\USERNAME\.aws\config` sous Windows. Ce fichier contient les paramètres de configuration du profil par défaut et des profils nommés.

Le stockage des clés d'accès en tant que variables d'environnement est une condition préalable à [la section called “Plugin d'authentification pour Java 4.x”](#). Notez que cela inclut la valeur par défaut Région AWS. Le client recherche les informations d'identification à l'aide de la chaîne de fournisseurs d'informations d'identification par défaut, et les clés d'accès stockées sous forme de variables d'environnement ont priorité sur tous les autres emplacements, par exemple les fichiers de configuration. Pour plus d'informations, consultez [la section Paramètres de configuration et priorité](#).

Les exemples suivants montrent comment vous pouvez configurer des variables d'environnement pour l'utilisateur par défaut.

Linux, macOS, or Unix

```
$ export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
$ export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of security token>
$ export AWS_DEFAULT_REGION=us-east-1
```

La définition de la variable d'environnement permet de modifier la valeur utilisée jusqu'à la fin de votre session shell, ou jusqu'à ce que vous définissiez la variable sur une autre valeur. Vous pouvez rendre les variables persistantes dans de futures sessions en les définissant dans votre script de démarrage de shell.

Windows Command Prompt

```
C:\> setx AWS_ACCESS_KEY_ID AKIAIOSFODNN7EXAMPLE
C:\> setx AWS_SECRET_ACCESS_KEY wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
C:\> setx AWS_SESSION_TOKEN AQoDYXdzEJr...<remainder of security token>
C:\> setx AWS_DEFAULT_REGION us-east-1
```

L'utilisation de [set](#) pour définir une variable d'environnement modifie la valeur utilisée jusqu'à la fin de la session d'invite de commande en cours, ou jusqu'à ce que vous définissiez la variable sur une autre valeur. L'utilisation de [setx](#) pour définir une variable d'environnement modifie la valeur utilisée dans la session d'invite de commande en cours et toutes les sessions d'invite de

commande que vous créez après l'exécution de la commande. Cela n'affecte pas les autres shells de commande qui sont déjà en cours d'exécution lorsque vous exécutez la commande.

PowerShell

```
PS C:\> $Env:AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
PS C:\> $Env:AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY"
PS C:\> $Env:AWS_SESSION_TOKEN="AQoDYXdzEJr...<remainder of security token>"
PS C:\> $Env:AWS_DEFAULT_REGION="us-east-1"
```

Si vous définissez une variable d'environnement à l'invite PowerShell, comme indiqué dans les exemples précédents, elle enregistre la valeur uniquement pendant la durée de la session en cours. Pour que le paramètre de variable d'environnement soit persistant dans toutes les sessions PowerShell et dans les sessions d'invite de commande, stockez-le à l'aide de l'application système du Panneau de configuration. Vous pouvez également définir la variable pour toutes les PowerShell sessions futures en l'ajoutant à votre PowerShell profil. Consultez la [PowerShell documentation](#) pour plus d'informations sur le stockage des variables d'environnement ou leur persistance d'une session à l'autre.

Points de terminaison de service pour Amazon Keyspaces

Rubriques

- [Ports et protocoles](#)
- [Points de terminaison globaux](#)
- [AWS GovCloud \(US\) Region Points de terminaison FIPS](#)
- [Points de terminaison pour les régions de Chine](#)
- [Points de terminaison des diffusions](#)
- [Connexion à des points de terminaison à double pile](#)

Ports et protocoles

Vous pouvez accéder à Amazon Keyspaces par programmation en exécutant un `cqlsh` client, avec un pilote Cassandra sous licence Apache 2.0, ou en utilisant le SDK et le SDK. AWS CLI AWS

Le tableau suivant indique les ports et les protocoles des différents mécanismes d'accès.

Accès programmatique	Port	Protocole
CQLSH	9142	TLS
Pilote Cassandra	9142	TLS
AWS CLI	443	HTTPS
AWS SDK	443	HTTPS

Pour les connexions TLS, Amazon Keyspaces utilise des certificats émis dans le cadre d'Amazon Trust Services (Amazon Root CA 1—4) pour s'authentifier auprès du serveur. Pour plus d'informations, consultez [the section called “Comment configurer manuellement les cqlsh connexions pour le protocole TLS”](#) la section « [Avant de commencer](#) » de votre pilote dans le [the section called “Utilisation d'un pilote client Cassandra”](#) chapitre.

Points de terminaison globaux

Amazon Keyspaces prend en charge les points de terminaison publics IPv4 et IPv6. Vous pouvez choisir entre des points de terminaison IPv4 et des points de terminaison à double pile. Les points de terminaison utilisent la convention de dénomination suivante, selon laquelle vous pouvez les *us-east-1* remplacer par une autre Région AWS disponible dans le tableau.

- Points de terminaison IPv4 — `cassandra.us-east-1.amazonaws.com`
- Dual-stack points de terminaison — `cassandra.us-east-1.api.aws`

Pour plus d'informations sur les points de terminaison à double pile et sur la façon de configurer les connexions, consultez [the section called “Connexion à des points de terminaison à double pile”](#)

Amazon Keyspaces est disponible dans les régions suivantes.

Nom de la région	Région	Point de terminaison	Protocole	
US East (Ohio)	us-east-2	cassandra.us-east-2.amazonaws.com cassandra.us-east-2.api.aws	HTTPS et TLS	

Nom de la région	Région	Point de terminaison	Protocole	
			HTTPS et TLS	
USA Est (Virginie du Nord)	us-east-1	cassandra.us-east-1.amazonaws.com	HTTPS et TLS	
		cassandra-fips.us-east-1.api.aws		
		cassandra-fips.us-east-1.amazonaws.com	HTTPS et TLS	
		cassandra.us-east-1.api.aws	HTTPS et TLS	
			HTTPS et TLS	
USA Ouest (Californie du Nord)	us-west-1	cassandra.us-west-1.amazonaws.com	HTTPS et TLS	
		cassandra.us-west-1.api.aws	HTTPS et TLS	
US West (Oregon)	us-west-2	cassandra.us-west-2.amazonaws.com	HTTPS et TLS	
		cassandra-fips.us-west-2.api.aws		
		cassandra-fips.us-west-2.amazonaws.com	HTTPS et TLS	
		cassandra.us-west-2.api.aws	HTTPS et TLS	
			HTTPS et TLS	

Nom de la région	Région	Point de terminaison	Protocole	
Afrique (Le Cap)	af-south-1	cassandra.af-south-1.amazonaws.com cassandra.af-south-1.api.aws	HTTPS et TLS HTTPS et TLS	
Asie-Pacifique (Hong Kong)	ap-east-1	cassandra.ap-east-1.amazonaws.com cassandra.ap-east-1.api.aws	HTTPS et TLS HTTPS et TLS	
Asie-Pacifique (Hyderabad)	ap-south-2	cassandra.ap-south-2.api.aws	HTTPS et TLS	
Asie-Pacifique (Jakarta)	ap-southeast-3	cassandra.ap-southeast-3.api.aws	HTTPS et TLS	
Asie-Pacifique (Malaisie)	ap-southeast-5	cassandra.ap-southeast-5.amazonaws.com cassandra.ap-southeast-5.api.aws	HTTPS et TLS HTTPS et TLS	
Asie-Pacifique (Melbourne)	ap-southeast-4	cassandra.ap-southeast-4.api.aws	HTTPS et TLS	

Nom de la région	Région	Point de terminaison	Protocole	
Asia Pacific (Mumbai)	ap-south-1	cassandra.ap-south-1.amazonaws.com cassandra.ap-south-1.api.aws	HTTPS et TLS HTTPS et TLS	
Asie-Pacifique (Nouvelle Zélande)	ap-southeast-6	cassandra.ap-southeast-6.api.aws	HTTPS et TLS	
Asie-Pacifique (Osaka)	ap-northeast-3	cassandra.ap-northeast-3.api.aws	HTTPS et TLS	
Asia Pacific (Seoul)	ap-northeast-2	cassandra.ap-northeast-2.amazonaws.com cassandra.ap-northeast-2.api.aws	HTTPS et TLS HTTPS et TLS	
Asie-Pacifique (Singapour)	ap-southeast-1	cassandra.ap-southeast-1.amazonaws.com cassandra.ap-southeast-1.api.aws	HTTPS et TLS HTTPS et TLS	
Asia Pacific (Sydney)	ap-southeast-2	cassandra.ap-southeast-2.amazonaws.com cassandra.ap-southeast-2.api.aws	HTTPS et TLS HTTPS et TLS	
Asie-Pacifique (Taipei)	ap-east-2	cassandra.ap-east-2.api.aws	HTTPS et TLS	

Nom de la région	Région	Point de terminaison	Protocole	
Asie-Pacifique (Thaïlande)	ap-southeast-7	cassandra.ap-southeast-7.amazonaws.com cassandra.ap-southeast-7.api.aws	HTTPS et TLS HTTPS et TLS	
Asie-Pacifique (Tokyo)	ap-northeast-1	cassandra.ap-northeast-1.amazonaws.com cassandra.ap-northeast-1.api.aws	HTTPS et TLS HTTPS et TLS	
Canada (Centre)	ca-central-1	cassandra.ca-central-1.amazonaws.com cassandra.ca-central-1.api.aws	HTTPS et TLS HTTPS et TLS	
Canada-Ouest (Calgary)	ca-west-1	cassandra.ca-west-1.amazonaws.com cassandra.ca-west-1.api.aws	HTTPS et TLS HTTPS et TLS	
Europe (Francfort)	eu-central-1	cassandra.eu-central-1.amazonaws.com cassandra.eu-central-1.api.aws	HTTPS et TLS HTTPS et TLS	
Europe (Irlande)	eu-west-1	cassandra.eu-west-1.amazonaws.com cassandra.eu-west-1.api.aws	HTTPS et TLS HTTPS et TLS	

Nom de la région	Région	Point de terminaison	Protocole	
Europe (Londres)	eu-west-2	cassandra.eu-west-2.amazonaws.com cassandra.eu-west-2.api.aws	HTTPS et TLS HTTPS et TLS	
Europe (Milan)	eu-south-1	cassandra.eu-south-1.api.aws	HTTPS et TLS	
Europe (Paris)	eu-west-3	cassandra.eu-west-3.amazonaws.com cassandra.eu-west-3.api.aws	HTTPS et TLS HTTPS et TLS	
Europe (Espagne)	eu-south-2	cassandra.eu-south-2.api.aws	HTTPS et TLS	
Europe (Stockholm)	eu-north-1	cassandra.eu-north-1.amazonaws.com cassandra.eu-north-1.api.aws	HTTPS et TLS HTTPS et TLS	
Europe (Zurich)	eu-central-2	cassandra.eu-central-2.api.aws	HTTPS et TLS	
Israël (Tel Aviv)	il-central-1	cassandra.il-central-1.api.aws	HTTPS et TLS	
Mexique (Centre)	mx-central-1	cassandra.mx-central-1.api.aws	HTTPS et TLS	

Nom de la région	Région	Point de terminaison	Protocole	
Moyen-Orient (Bahreïn)	me-south-1	cassandra.me-south-1.amazonaws.com cassandra.me-south-1.api.aws	HTTPS et TLS HTTPS et TLS	
Moyen-Orient (EAU)	me-central-1	cassandra.me-central-1.amazonaws.com cassandra.me-central-1.api.aws	HTTPS et TLS HTTPS et TLS	
Amérique du Sud (São Paulo)	sa-east-1	cassandra.sa-east-1.amazonaws.com cassandra.sa-east-1.api.aws	HTTPS et TLS HTTPS et TLS	
AWS GovCloud (US-East)	us-gov-east-1	cassandra.us-gov-east-1.amazonaws.com cassandra.us-gov-east-1.api.aws	HTTPS et TLS HTTPS et TLS	
AWS GovCloud (US-West)	us-gov-west-1	cassandra.us-gov-west-1.amazonaws.com cassandra.us-gov-west-1.api.aws	HTTPS et TLS HTTPS et TLS	

AWS GovCloud (US) Region Points de terminaison FIPS

Points de terminaison FIPS disponibles dans le. AWS GovCloud (US) Region Amazon Keyspaces prend en charge les points de terminaison FIPS IPv4 et IPv6. Vous pouvez choisir entre des points de terminaison IPv4 et des points de terminaison à double pile. Pour plus d'informations, consultez [Amazon Keyspaces dans le guide de l'AWS GovCloud \(US\) utilisateur](#).

Nom de la région	Région	Point de terminaison FIPS IPv4	Endpoint FIPS à double pile	Protocole
AWS GovCloud (US-East)	us-gov-east-1	cassandra.us-gov-east-1.amazonaws.com	cassandra.us-gov-east-1.api.aws	HTTPS et TLS
AWS GovCloud (US-West)	us-gov-west-1	cassandra.us-gov-west-1.amazonaws.com	cassandra.us-gov-west-1.api.aws	HTTPS et TLS

Points de terminaison pour les régions de Chine

Amazon Keyspaces prend en charge les points de terminaison IPv4 dans les régions de Chine. AWS

Pour accéder à ces points de terminaison, vous devez vous inscrire pour obtenir un ensemble distinct d'informations d'identification propres aux régions de Chine. Pour plus d'informations, voir [Inscription, comptes et informations d'identification en Chine](#).

Nom de la région	Région	Point de terminaison	Protocole
Chine (Beijing)	cn-north-1	cassandra.cn-north-1.amazonaws.com .cn	HTTPS et TLS
Chine (Ningxia)	cn-northwest-1	cassandra.cn-northwest-1.amazonaws.com .cn	HTTPS et TLS

Points de terminaison des diffusions

Amazon Keyspaces CDC streams est disponible dans les versions suivantes Régions AWS. Ce tableau indique le point de terminaison de service à double pile disponible pour chaque région. Pour

plus d'informations sur Amazon Keyspaces CDC streams, voir [the section called “Accédez aux points de terminaison des flux CDC”](#).

Nom de la région	Région	Point de terminaison	Protocole	
US East (Ohio)	us-east-2	cassandra-streams.us-east-2.api.aws	HTTPS	
USA Est (Virginie du Nord)	us-east-1	cassandra-streams.us-east-1.api.aws	HTTPS	
USA Ouest (Californie du Nord)	us-west-1	cassandra-streams.us-west-1.api.aws	HTTPS	
US West (Oregon)	us-west-2	cassandra-streams.us-west-2.api.aws	HTTPS	
Afrique (Le Cap)	af-south-1	cassandra-streams.af-south-1.api.aws	HTTPS	
Asie-Pacifique (Hong Kong)	ap-east-1	cassandra-streams.ap-east-1.api.aws	HTTPS	
Asie-Pacifique (Hyderabad)	ap-south-2	cassandra-streams.ap-south-2.api.aws	HTTPS	
Asie-Pacifique (Jakarta)	ap-southeast-3	cassandra-streams.ap-southeast-3.api.aws	HTTPS	

Nom de la région	Région	Point de terminaison	Protocole	
Asie-Pacifique (Malaisie)	ap-southeast-5	cassandra-streams.ap-southeast-5.api.aws	HTTPS	
Asie-Pacifique (Melbourne)	ap-southeast-4	cassandra-streams.ap-southeast-4.api.aws	HTTPS	
Asia Pacific (Mumbai)	ap-south-1	cassandra-streams.ap-south-1.api.aws	HTTPS	
Asie-Pacifique (Nouvelle Zélande)	ap-southeast-6	cassandra-streams.ap-southeast-6.api.aws	HTTPS	
Asie-Pacifique (Osaka)	ap-northeast-3	cassandra-streams.ap-northeast-3.api.aws	HTTPS	
Asia Pacific (Seoul)	ap-northeast-2	cassandra-streams.ap-northeast-2.api.aws	HTTPS	
Asie-Pacifique (Singapour)	ap-southeast-1	cassandra-streams.ap-southeast-1.api.aws	HTTPS	
Asia Pacific (Sydney)	ap-southeast-2	cassandra-streams.ap-southeast-2.api.aws	HTTPS	

Nom de la région	Région	Point de terminaison	Protocole	
Asie-Pacifique (Taïpei)	ap-east-2	cassandra-streams.ap-east-2.api.aws	HTTPS	
Asie-Pacifique (Thaïlande)	ap-southeast-7	cassandra-streams.ap-southeast-7.api.aws	HTTPS	
Asie-Pacifique (Tokyo)	ap-northeast-1	cassandra-streams.ap-northeast-1.api.aws	HTTPS	
Canada (Centre)	ca-central-1	cassandra-streams.ca-central-1.api.aws	HTTPS	
Canada-Ouest (Calgary)	ca-west-1	cassandra-streams.ca-west-1.api.aws	HTTPS	
Europe (Francfort)	eu-central-1	cassandra-streams.eu-central-1.api.aws	HTTPS	
Europe (Irlande)	eu-west-1	cassandra-streams.eu-west-1.api.aws	HTTPS	
Europe (Londres)	eu-west-2	cassandra-streams.eu-west-2.api.aws	HTTPS	
Europe (Milan)	eu-south-1	cassandra-streams.eu-south-1.api.aws	HTTPS	
Europe (Paris)	eu-west-3	cassandra-streams.eu-west-3.api.aws	HTTPS	

Nom de la région	Région	Point de terminaison	Protocole	
Europe (Espagne)	eu-south-2	cassandra-streams.eu-south-2.api.aws	HTTPS	
Europe (Stockholm)	eu-north-1	cassandra-streams.eu-north-1.api.aws	HTTPS	
Europe (Zurich)	eu-central-2	cassandra-streams.eu-central-2.api.aws	HTTPS	
Israël (Tel Aviv)	il-central-1	cassandra-streams.il-central-1.api.aws	HTTPS	
Mexique (Centre)	mx-central-1	cassandra-streams.mx-central-1.api.aws	HTTPS	
Moyen-Orient (Bahreïn)	me-south-1	cassandra-streams.me-south-1.api.aws	HTTPS	
Moyen-Orient (EAU)	me-central-1	cassandra-streams.me-central-1.api.aws	HTTPS	
Amérique du Sud (São Paulo)	sa-east-1	cassandra-streams.sa-east-1.api.aws	HTTPS	
AWS GovCloud (US-East)	us-gov-east-1	cassandra-streams.us-gov-east-1.api.aws	HTTPS	

Nom de la région	Région	Point de terminaison	Protocole	
AWS GovCloud (US-West)	us-gov-west-1	cassandra-streams.us-gov-west-1.api.aws	HTTPS	

Connexion à des points de terminaison à double pile

Les points de terminaison mondiaux Amazon Keyspaces sont des points de terminaison à double pile qui acceptent les requêtes IPv4 et IPv6.

Lorsque vous vous connectez à Amazon Keyspaces via IPv6, le service adapte automatiquement les réponses du tableau système en fonction de votre protocole de connexion. Cela garantit que vos applications reçoivent des informations d'adresse réseau cohérentes correspondant à leur type de connexion. Cela fournit des informations précises sur la topologie du réseau au client tout en maintenant la rétrocompatibilité avec les applications CQL existantes.

Amazon Keyspaces détecte automatiquement le protocole réseau (IPv4 ou IPv6) utilisé par votre connexion client et ajuste les réponses du tableau système en conséquence. Cette détection se produit de manière transparente lors de l'établissement de la connexion initiale, sans nécessiter de configuration supplémentaire de la part de l'application cliente.

Amazon Keyspaces renvoie les adresses IP en fonction de votre protocole de connexion. Par exemple, une demande provenant d'un réseau IPv4 renvoie la réponse suivante.

```
SELECT * FROM system.peers;  
-- Returns IPv4 addresses in peer column  
-- Example: 172.31.1.1, 172.31.1.2, etc.
```

Une connexion entre un réseau IPv6 et un point de terminaison à double pile, par exemple `cassandra.us-east-1.api.aws`, renvoie la réponse suivante.

```
SELECT * FROM system.peers;  
-- Returns IPv6 addresses in peer column  
-- Example: 2001:db8::1, 2001:db8::2, etc.
```

Pour plus d'informations sur la prise en charge d'IPv6 dans Amazon Keyspaces, consultez [the section called “Prise en charge de l'IPv6 dans Amazon Keyspaces”](#)

Prise en charge de l'IPv6 dans Amazon Keyspaces

La prise en charge d'IPv6 dans Amazon Keyspaces permet aux applications d'établir des connexions à l'aide du protocole Internet version 6, le protocole Internet de nouvelle génération qui fournit un espace d'adressage considérablement étendu par rapport à l'IPv4. La mise en œuvre utilise des points de terminaison à double pile qui prennent en charge simultanément IPv4 et IPv6, garantissant ainsi une rétrocompatibilité tout en permettant une connectivité prête pour l'avenir. Pour obtenir la liste des points de terminaison, consultez [the section called “Points de terminaison globaux”](#).

Amazon Keyspaces met en œuvre le support IPv6 via une architecture à double pile qui assure une rétrocompatibilité complète tout en permettant la connectivité IPv6.

Résolution DNS dans Amazon Keyspaces

Lorsque les applications se connectent à des points de terminaison à double pile, le processus de résolution DNS renvoie les deux types d'adresses :

Enregistrements A (IPv4)

Adresses IPv4 traditionnelles pour une rétrocompatibilité

Enregistrements AAAA (IPv6)

Nouvelles adresses IPv6 pour une connectivité moderne

Le système d'exploitation et la pile réseau du client sélectionnent automatiquement le protocole le plus approprié en fonction de la configuration locale, de la disponibilité du réseau et des préférences du système.

Le protocole Cassandra Query Language (CQL) prend parfaitement en charge la connectivité IPv6 sans qu'il soit nécessaire de modifier le code de l'application.

Sélection automatique du protocole

- Les applications spécifient le point de terminaison à double pile
- La pile réseau choisit IPv4 ou IPv6 en fonction de la disponibilité
- Aucune modification de code requise pour les applications CQL existantes

Compatibilité des pilotes

- Tous les principaux pilotes CQL prennent en charge IPv6 de manière transparente
- DataStax les pilotes gèrent les adresses IPv6 de manière native
- Open-source les pilotes fonctionnent sans modification

Cohérence des connexions

- Les tables système indiquent le protocole de connexion utilisé
- Les connexions IPv6 affichent les adresses IPv6 dans `system.peers`
- Les connexions IPv4 continuent d'afficher les adresses IPv4

Utilisation **cqlsh** pour se connecter à Amazon Keyspaces

Pour vous connecter à Amazon Keyspaces à l'aide de `cqlsh`, vous pouvez utiliser le `cqlsh-expansion`. Il s'agit d'une boîte à outils qui contient des outils Apache Cassandra courants `cqlsh` et des aides préconfigurés pour Amazon Keyspaces tout en maintenant une compatibilité totale avec Apache Cassandra. Le `cqlsh-expansion` intègre le plugin d'authentification SigV4 et vous permet de vous connecter à l'aide de clés d'accès IAM au lieu du nom d'utilisateur et du mot de passe. Vous devez uniquement installer les `cqlsh` scripts pour établir une connexion, et non la distribution complète d'Apache Cassandra, car Amazon Keyspaces fonctionne sans serveur. Ce package d'installation léger inclut les `cqlsh` scripts classiques `cqlsh-expansion` et les scripts que vous pouvez installer sur n'importe quelle plateforme supportant Python.

Note

`Murmur3Partitioner` est le partitionneur recommandé pour Amazon Keyspaces et le `cqlsh-expansion`. Le `cqlsh-expansion` ne prend pas en charge les `AmazonKeyspacesDefaultPartitioner`. Pour de plus amples informations, veuillez consulter [the section called “Utilisation de partitionneurs”](#).

Pour des informations générales sur `cqlsh`, voir [cqlsh: le shell CQL](#).

Rubriques

- [Utilisation du `cqlsh-expansion` pour se connecter à Amazon Keyspaces](#)
- [Comment configurer manuellement les `cqlsh` connexions pour le protocole TLS](#)

Utilisation du **cqlsh-expansion** pour se connecter à Amazon Keyspaces

Installation et configuration du **cqlsh-expansion**

1. Pour installer le package **cqlsh-expansion** Python, vous pouvez exécuter une **pip** commande. Cela installe les **cqlsh-expansion** scripts sur votre machine à l'aide d'une installation **pip** avec un fichier contenant une liste de dépendances. Il **--user** flag indique **pip** d'utiliser le répertoire d'installation utilisateur Python pour votre plateforme. Sur un système basé sur Unix, ce devrait être le **~/.local/** répertoire.

Vous avez besoin de Python 3 pour installer le **cqlsh-expansion**, pour connaître votre version de Python, utilisez **Python --version**. Pour l'installer, vous pouvez exécuter la commande suivante.

```
python3 -m pip install --user cqlsh-expansion
```

La sortie doit ressembler à ceci.

```
Collecting cqlsh-expansion
  Downloading cqlsh_expansion-0.9.6-py3-none-any.whl (153 kB)
##### 153.7/153.7 KB 3.3 MB/s eta 0:00:00
Collecting cassandra-driver
  Downloading cassandra_driver-3.28.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (19.1 MB)
##### 19.1/19.1 MB 44.5 MB/s eta 0:00:00
Requirement already satisfied: six>=1.12.0 in /usr/lib/python3/dist-packages (from cqlsh-expansion) (1.16.0)
Collecting boto3
  Downloading boto3-1.29.2-py3-none-any.whl (135 kB)
##### 135.8/135.8 KB 17.2 MB/s eta 0:00:00
Collecting cassandra-sigv4>=4.0.2
  Downloading cassandra_sigv4-4.0.2-py2.py3-none-any.whl (9.8 kB)
Collecting botocore<1.33.0,>=1.32.2
  Downloading botocore-1.32.2-py3-none-any.whl (11.4 MB)
##### 11.4/11.4 MB 60.9 MB/s eta 0:00:00
Collecting s3transfer<0.8.0,>=0.7.0
  Downloading s3transfer-0.7.0-py3-none-any.whl (79 kB)
##### 79.8/79.8 KB 13.1 MB/s eta 0:00:00
Collecting jmespath<2.0.0,>=0.7.1
  Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Collecting geomet<0.3,>=0.1
```

```

Downloading geomet-0.2.1.post1-py3-none-any.whl (18 kB)
Collecting python-dateutil<3.0.0,>=2.1
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
##### 247.7/247.7 KB 33.1 MB/s eta 0:00:00
Requirement already satisfied: urllib3<2.1,>=1.25.4 in /usr/lib/python3/dist-packages (from boto3<1.33.0,>=1.32.2->boto3->cqlsh-expansion) (1.26.5)
Requirement already satisfied: click in /usr/lib/python3/dist-packages (from geomet<0.3,>=0.1->cassandra-driver->cqlsh-expansion) (8.0.3)
Installing collected packages: python-dateutil, jmespath, geomet, cassandra-driver, boto3, s3transfer, boto3, cassandra-sigv4, cqlsh-expansion
  WARNING: The script geomet is installed in '/home/ubuntu/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
  WARNING: The scripts cqlsh, cqlsh-expansion and cqlsh-expansion.init are installed in '/home/ubuntu/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed boto3-1.29.2 botocore-1.32.2 cassandra-driver-3.28.0 cassandra-sigv4-4.0.2 cqlsh-expansion-0.9.6 geomet-0.2.1.post1 jmespath-1.0.1 python-dateutil-2.8.2 s3transfer-0.7.0

```

Si le répertoire d'installation ne se trouve pas dans le PATH, vous devez l'ajouter en suivant les instructions de votre système d'exploitation. Vous trouverez ci-dessous un exemple pour Ubuntu Linux.

```
export PATH="$PATH:/home/ubuntu/.local/bin"
```

Pour confirmer que le package est installé, vous pouvez exécuter la commande suivante.

```
cqlsh-expansion --version
```

La sortie doit ressembler à ceci.

```
cqlsh 6.1.0
```

2. Pour configurer le `cqlsh-expansion`, vous pouvez exécuter un script de post-installation pour effectuer automatiquement les étapes suivantes :

1. Créez le `.cassandra` répertoire dans le répertoire personnel de l'utilisateur s'il n'existe pas déjà.

2. Copiez un fichier de `cqlshrc` configuration préconfiguré dans le `.cassandra` répertoire.
3. Copiez le fichier de certificat combiné dans le `.cassandra` répertoire. Amazon Keyspaces utilise ce certificat pour configurer la connexion sécurisée avec le protocole TLS (Transport Layer Security). Le chiffrement en transit fournit une couche supplémentaire de protection des données en chiffrant vos données lorsqu'elles sont acheminées vers et depuis Amazon Keyspaces. Pour plus d'informations sur les certificats, consultez [the section called “Comment configurer manuellement les cqlsh connexions pour le protocole TLS”](#).

Pour consulter d'abord le script, vous pouvez y accéder dans le dépôt Github à l'adresse. [post_install.py](#)

Pour utiliser le script, vous pouvez exécuter la commande suivante.

```
cqlsh-expansion.init
```

 Note

Le répertoire et le fichier créés par le script de post-installation ne sont pas supprimés lorsque vous `cqlsh-expansion` désinstallez `pip uninstall` l'utilisateur et doivent être supprimés manuellement.

Connexion à Amazon Keyspaces à l'aide du **cqlsh-expansion**

1. Configurez votre variable d'environnement Région AWS et ajoutez-la en tant que variable d'environnement utilisateur.

Pour ajouter votre région par défaut en tant que variable d'environnement sur un système Unix, vous pouvez exécuter la commande suivante. Pour cet exemple, nous utilisons `us-east-1`.

```
export AWS_DEFAULT_REGION=us-east-1
```

Pour plus d'informations sur la définition des variables d'environnement, y compris pour d'autres plateformes, consultez [Comment définir des variables d'environnement](#).

2. Trouvez le point de terminaison de votre service.

Choisissez le point de terminaison de service approprié pour votre région. Pour consulter les points de terminaison disponibles pour Amazon Keyspaces, consultez [the section called “Points de terminaison de service”](#) Pour cet exemple, nous utilisons le point de terminaison `cassandra.us-east-1.amazonaws.com`.

3. Configurez la méthode d'authentification.

La connexion à l'aide de clés d'accès IAM (utilisateurs IAM, rôles et identités fédérées) est la méthode recommandée pour renforcer la sécurité.

Avant de pouvoir vous connecter à l'aide des clés d'accès IAM, vous devez suivre les étapes suivantes :

- a. Créez un utilisateur IAM ou suivez les meilleures pratiques et créez un rôle IAM que les utilisateurs IAM peuvent assumer. Pour plus d'informations sur la création de clés d'accès IAM, consultez [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#).
- b. Créez une politique IAM qui accorde au rôle (ou à l'utilisateur IAM) au moins un accès en lecture seule à Amazon Keyspaces. Pour plus d'informations sur les autorisations requises pour que l'utilisateur ou le rôle IAM se connecte à Amazon Keyspaces, consultez [the section called “Accès aux tables Amazon Keyspaces”](#)
- c. Ajoutez les clés d'accès de l'utilisateur IAM aux variables d'environnement de l'utilisateur, comme indiqué dans l'exemple suivant.

```
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

Pour plus d'informations sur la définition des variables d'environnement, y compris pour d'autres plateformes, consultez [Comment définir des variables d'environnement](#).

Note

Si vous vous connectez depuis une instance Amazon EC2, vous devez également configurer une règle sortante dans le groupe de sécurité qui autorise le trafic depuis l'instance vers Amazon Keyspaces. Pour plus d'informations sur la façon d'afficher et de modifier les règles de sortie EC2, consultez la section [Ajouter des règles à un groupe de sécurité dans le guide de l'utilisateur Amazon EC2](#).

4. Connectez-vous à Amazon Keyspaces à l'aide de l'authentification `cqlsh-expansion` et SigV4.

Pour vous connecter à Amazon Keyspaces avec `cqlsh-expansion`, vous pouvez utiliser la commande suivante. Assurez-vous de remplacer le point de terminaison du service par le point de terminaison adapté à votre région.

```
cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 --ssl
```

Si la connexion est établie, vous devriez obtenir un résultat similaire à celui de l'exemple suivant.

```
Connected to Amazon Keyspaces at cassandra.us-east-1.amazonaws.com:9142
[cqlsh 6.1.0 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh current consistency level is ONE.
cqlsh>
```

Si vous rencontrez une erreur de connexion, consultez [the section called “Erreurs de connexion Cqlsh”](#) les informations de dépannage.

- Connectez-vous à Amazon Keyspaces avec des informations d'identification spécifiques au service.

Pour vous connecter à la combinaison traditionnelle de nom d'utilisateur et de mot de passe utilisée par Cassandra pour l'authentification, vous devez d'abord créer des informations d'identification spécifiques au service pour Amazon Keyspaces, comme décrit dans [the section called “Créez des informations d'identification spécifiques au service”](#). Vous devez également autoriser cet utilisateur à accéder à Amazon Keyspaces. Pour plus d'informations, consultez [the section called “Accès aux tables Amazon Keyspaces”](#).

Après avoir créé des informations d'identification et des autorisations spécifiques au service pour l'utilisateur, vous devez mettre à jour le `cqlshrc` fichier, qui se trouve généralement dans le chemin du répertoire utilisateur. `~/.cassandra/` Dans le `cqlshrc` fichier, accédez à la `[authentication]` section Cassandra et commentez le module et la classe SigV4 `[auth_provider]` en utilisant le caractère « ; » comme indiqué dans l'exemple suivant.

```
[auth_provider]

; module = cassandra_sigv4.auth
```



```
; classname = SigV4AuthProvider
```

Après avoir mis à jour le `cqlshrc` fichier, vous pouvez vous connecter à Amazon Keyspaces avec des informations d'identification spécifiques au service à l'aide de la commande suivante.

```
cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 -u myUserName -  
p myPassword --ssl
```

Nettoyage

- Pour supprimer le `cqlsh-expansion` package, vous pouvez utiliser la `pip uninstall` commande.

```
pip3 uninstall cqlsh-expansion
```

La `pip3 uninstall` commande ne supprime pas le répertoire et les fichiers associés créés par le script de post-installation. Pour supprimer le dossier et les fichiers créés par le script de post-installation, vous pouvez supprimer le `.cassandra` répertoire.

Comment configurer manuellement les **cqlsh** connexions pour le protocole TLS

Amazon Keyspaces accepte uniquement les connexions sécurisées utilisant le protocole TLS (Transport Layer Security). Vous pouvez utiliser l'`cqlsh-expansion` utilitaire qui télécharge automatiquement les certificats pour vous et installe un fichier de `cqlshrc` configuration préconfiguré. Pour plus d'informations, consultez [the section called “Utilisation de l'`cqlsh-expansion`”](#) cette page.

Si vous souhaitez télécharger les certificats et configurer la connexion manuellement, procédez comme suit.

1. Téléchargez les certificats numériques suivants et enregistrez les fichiers localement ou dans votre répertoire personnel.
 - a. AmazonRootCA1
 - b. AmazonRootCA2

- c. AmazonRootCA3
- d. AmazonRootCA4
- e. Starfield Class 2 Root (en option, pour une rétrocompatibilité)

Pour télécharger les certificats, vous pouvez utiliser les commandes suivantes.

```
curl -O https://www.amazontrust.com/repository/AmazonRootCA1.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA2.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA3.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA4.pem
curl -O https://certs.secureserver.net/repository/sf-class2-root.crt
```

Note

Amazon Keyspaces utilisait auparavant des certificats TLS ancrés à l'autorité de certification Starfield Class 2. AWS migre le tout Régions AWS vers des certificats émis dans le cadre d'Amazon Trust Services (Amazon Root CA 1—4). Au cours de cette transition, configurez les clients de manière à ce qu'ils fassent confiance à la fois aux autorités de certification Amazon Root 1 à 4 et à la racine Starfield afin de garantir la compatibilité entre toutes les régions.

2. Combinez tous les certificats téléchargés dans un seul pem fichier portant le nom *keyspaces-bundle.pem* indiqué dans nos exemples. Pour ce faire, exécutez la commande suivante : Notez le chemin du fichier, vous en aurez besoin plus tard.

```
cat AmazonRootCA1.pem \
    AmazonRootCA2.pem \
    AmazonRootCA3.pem \
    AmazonRootCA4.pem \
    sf-class2-root.crt \
    > keyspaces-bundle.pem
```

3. Ouvrez le fichier `cqlshrc` de configuration dans le répertoire de base de Cassandra, par exemple, `${HOME}/.cassandra/cqlshrc` et ajoutez les lignes suivantes.

```
[connection]
port = 9142
factory = cqlshlib.ssl.ssl_transport_factory
```

```
[ssl]
validate = true
certfile = path_to_file/keysaces-bundle.pem
```

Utilisation du AWS CLI pour se connecter à Amazon Keyspaces

Vous pouvez utiliser le AWS Command Line Interface (AWS CLI) pour contrôler plusieurs AWS services depuis la ligne de commande et les automatiser par le biais de scripts. Avec Amazon Keyspaces, vous pouvez utiliser les opérations du langage AWS CLI de définition des données (DDL), telles que la création d'une table. En outre, vous pouvez utiliser des services et des outils d'infrastructure en tant que code (IaC) tels que AWS CloudFormation Terraform.

Avant de pouvoir l'utiliser AWS CLI avec Amazon Keyspaces, vous devez obtenir un identifiant de clé d'accès et une clé d'accès secrète. Pour de plus amples informations, veuillez consulter [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#).

Pour une liste complète de toutes les commandes disponibles pour Amazon Keyspaces dans le AWS CLI, consultez le [AWS CLI Command Reference](#).

Rubriques

- [Téléchargement et configuration du AWS CLI](#)
- [Utilisation du AWS CLI avec Amazon Keyspaces](#)

Téléchargement et configuration du AWS CLI

Le AWS CLI est disponible à l'adresse <https://aws.amazon.com/cli>. Elle s'exécute sous Windows, macOS ou Linux. Après avoir téléchargé le AWS CLI, procédez comme suit pour l'installer et le configurer :

1. Accédez au [guide de AWS Command Line Interface l'utilisateur](#)
2. Suivez les instructions d'[installation AWS CLI et de configuration du AWS CLI](#)

Utilisation du AWS CLI avec Amazon Keyspaces

Le format de ligne de commande consiste en un nom d'opération Amazon Keyspaces suivi des paramètres de cette opération. Il AWS CLI prend en charge une syntaxe abrégée pour les valeurs des paramètres, ainsi que du JSON. Les exemples Amazon Keyspaces suivants utilisent une syntaxe

AWS CLI abrégée. Pour plus d'informations, consultez la section [Utilisation de la syntaxe abrégée avec la CLI AWS](#).

La commande suivante crée un espace de touches avec le catalogue de noms.

```
aws keyspaces create-keyspace --keyspace-name 'catalog'
```

La commande renvoie la ressource Amazon Resource Name (ARN) dans la sortie.

```
{
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/catalog/"
}
```

Pour vérifier l'existence du catalogue Keyspace, vous pouvez utiliser la commande suivante.

```
aws keyspaces get-keyspace --keyspace-name 'catalog'
```

La sortie de la commande renvoie les valeurs suivantes.

```
{
  "keyspaceName": "catalog",
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/catalog/"
}
```

La commande suivante crée une table nommée `book_awards`. La clé de partition de la table comprend les colonnes `year` `award` et la clé de clustering les colonnes. Les deux colonnes `category` de `rank` clustering utilisent l'ordre de tri croissant. (Pour une lecture plus facile, les commandes longues dans cette section sont divisées en plusieurs lignes).

```
aws keyspaces create-table --keyspace-name 'catalog' --table-name 'book_awards'
                        --schema-definition 'allColumns=[{name=year,type=int},
{name=award,type=text},{name=rank,type=int},
                        {name=category,type=text}, {name=author,type=text},
{name=book_title,type=text},{name=publisher,type=text}],
                        partitionKeys=[{name=year},
{name=award}],clusteringKeys=[{name=category,orderBy=ASC},{name=rank,orderBy=ASC}]'
```

Cette commande produit le résultat suivant.

```
{
```

```
"resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/catalog/table/
book_awards"
}
```

Pour confirmer les métadonnées et les propriétés de la table, vous pouvez utiliser la commande suivante.

```
aws keyspaces get-table --keyspace-name 'catalog' --table-name 'book_awards'
```

Cette commande renvoie la sortie suivante.

```
{
  "keyspaceName": "catalog",
  "tableName": "book_awards",
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/catalog/table/
book_awards",
  "creationTimestamp": 1645564368.628,
  "status": "ACTIVE",
  "schemaDefinition": {
    "allColumns": [
      {
        "name": "year",
        "type": "int"
      },
      {
        "name": "award",
        "type": "text"
      },
      {
        "name": "category",
        "type": "text"
      },
      {
        "name": "rank",
        "type": "int"
      },
      {
        "name": "author",
        "type": "text"
      },
      {
        "name": "book_title",
        "type": "text"
      }
    ]
  }
}
```

```
    },
    {
      "name": "publisher",
      "type": "text"
    }
  ],
  "partitionKeys": [
    {
      "name": "year"
    },
    {
      "name": "award"
    }
  ],
  "clusteringKeys": [
    {
      "name": "category",
      "orderBy": "ASC"
    },
    {
      "name": "rank",
      "orderBy": "ASC"
    }
  ],
  "staticColumns": []
},
"capacitySpecification": {
  "throughputMode": "PAY_PER_REQUEST",
  "lastUpdateToPayPerRequestTimestamp": 1645564368.628
},
"encryptionSpecification": {
  "type": "AWS_OWNED_KMS_KEY"
},
"pointInTimeRecovery": {
  "status": "DISABLED"
},
"ttl": {
  "status": "ENABLED"
},
"defaultTimeToLive": 0,
"comment": {
  "message": ""
}
```

```
}
```

Lorsque vous créez des tables avec des schémas complexes, il peut être utile de charger la définition du schéma de la table à partir d'un fichier JSON. En voici un exemple. Téléchargez le fichier JSON d'exemple de définition de schéma depuis [schema_definition.zip](#) et `schema_definition.json` extrayez-le en prenant note du chemin d'accès au fichier. Dans cet exemple, le fichier JSON de définition du schéma se trouve dans le répertoire actuel. Pour connaître les différentes options de chemin de fichier, consultez [Comment charger des paramètres à partir d'un fichier](#).

```
aws keyspaces create-table --keyspace-name 'catalog'
                        --table-name 'book_awards' --schema-definition 'file://
schema_definition.json'
```

Les exemples suivants montrent comment créer une table simple nommée `MyTable` avec des options supplémentaires. Notez que les commandes sont divisées en lignes distinctes pour améliorer la lisibilité. Cette commande indique comment créer une table et :

- définir le mode de capacité de la table
- activer Point-in-time la restauration de la table
- définir la valeur TTL (Time to Live) par défaut de la table sur un an
- ajouter deux balises pour le tableau

```
aws keyspaces create-table --keyspace-name 'catalog' --table-name 'myTable'
                        --schema-definition 'allColumns=[{name=id,type=int},{name=name,type=text},
{name=date,type=timestamp}],partitionKeys=[{name=id}]'
                        --capacity-specification
                        'throughputMode=PROVISIONED,readCapacityUnits=5,writeCapacityUnits=5'
                        --point-in-time-recovery 'status=ENABLED'
                        --default-time-to-live '31536000'
                        --tags 'key=env,value=test' 'key=dpt,value=sec'
```

Cet exemple montre comment créer une nouvelle table qui utilise une clé gérée par le client pour le chiffrement et dont le protocole TTL est activé pour vous permettre de définir des dates d'expiration pour les colonnes et les lignes. Pour exécuter cet exemple, vous devez remplacer l'ARN de la ressource pour la AWS KMS clé gérée par le client par votre propre clé et vous assurer qu'Amazon Keyspaces y a accès.

```
aws keyspaces create-table --keyspace-name 'catalog' --table-name 'myTable'
```

```
--schema-definition 'allColumns=[{name=id,type=int},{name=name,type=text},
{name=date,type=timestamp}],partitionKeys=[{name=id}]'
--encryption-specification
'type=CUSTOMER_MANAGED_KMS_KEY,kmsKeyId=arn:aws:kms:us-
east-1:111122223333:key/11111111-2222-3333-4444-555555555555'
--ttl 'status=ENABLED'
```

Utilisation de l'API pour se connecter à Amazon Keyspaces

Vous pouvez utiliser le AWS SDK et le AWS Command Line Interface (AWS CLI) pour travailler de manière interactive avec Amazon Keyspaces. Vous pouvez utiliser l'API pour les opérations de définition du langage des données (DDL), telles que la création d'un espace de touches ou d'une table. En outre, vous pouvez utiliser des services et des outils d'infrastructure en tant que code (IaC) tels que AWS CloudFormation Terraform.

Avant de pouvoir l'utiliser AWS CLI avec Amazon Keyspaces, vous devez obtenir un identifiant de clé d'accès et une clé d'accès secrète. Pour de plus amples informations, veuillez consulter [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#).

Pour une liste complète de toutes les opérations disponibles pour Amazon Keyspaces dans l'API, consultez le manuel Amazon [Keyspaces](#) API Reference.

Utilisation d'un pilote client Cassandra pour accéder à Amazon Keyspaces par programmation

Vous pouvez utiliser de nombreux pilotes Cassandra tiers et open source pour vous connecter à Amazon Keyspaces. Amazon Keyspaces est compatible avec les pilotes Cassandra qui prennent en charge la version 3.11.2 d'Apache Cassandra. Voici les pilotes et les dernières versions que nous avons testés et que nous recommandons d'utiliser avec Amazon Keyspaces :

- Java v3.3
- Java v4.17
- Python Cassandra-driver 3.29.1
- Node.js cassandra driver -v 4.7.2
- G0 using GOCQL v1.6
- .NET CassandraCSharpDriver -v 3.20.1

Pour de plus amples informations sur les pilotes Cassandra, veuillez consulter [Pilotes client Apache Cassandra](#).

 Note

Pour vous aider à démarrer, vous pouvez consulter et télécharger des exemples de code de bout en bout qui établissent des connexions à Amazon Keyspaces à l'aide de pilotes courants. Consultez les exemples [d'Amazon Keyspaces](#) sur GitHub

Les didacticiels de ce chapitre incluent une simple requête CQL pour confirmer que la connexion à Amazon Keyspaces a été correctement établie. Pour savoir comment utiliser les espaces clés et les tableaux après vous être connecté à un point de terminaison Amazon Keyspaces, consultez. [Référence du langage CQL](#) Pour un didacticiel étape par étape qui montre comment se connecter à Amazon Keyspaces depuis un point de terminaison Amazon VPC, consultez. [the section called “Connexion aux points de terminaison VPC”](#)

Rubriques

- [Utilisation d'un pilote client Java Cassandra pour accéder à Amazon Keyspaces par programmation](#)
- [Utilisation d'un pilote client Cassandra Python pour accéder à Amazon Keyspaces par programmation](#)
- [Utilisation d'un pilote Node.js client Cassandra pour accéder à Amazon Keyspaces par programmation](#)
- [Utilisation d'un pilote client Cassandra .NET Core pour accéder à Amazon Keyspaces par programmation](#)
- [Utilisation d'un pilote client Cassandra Go pour accéder à Amazon Keyspaces par programmation](#)
- [Utilisation d'un pilote client Cassandra Perl pour accéder à Amazon Keyspaces par programmation](#)

Utilisation d'un pilote client Java Cassandra pour accéder à Amazon Keyspaces par programmation

Cette section explique comment vous connecter à Amazon Keyspaces à l'aide d'un pilote client Java.

 Note

Java 17 et le pilote DataStax Java 4.17 ne sont actuellement disponibles qu'en version bêta. Pour de plus amples informations, veuillez consulter https://docs.datastax.com/en/developer/java-driver/4.17/upgrade_guide/.

Pour fournir aux utilisateurs et aux applications des informations d'identification leur permettant d'accéder par programmation aux ressources Amazon Keyspaces, vous pouvez effectuer l'une des opérations suivantes :

- Créez des informations d'identification spécifiques au service associées à un utilisateur Gestion des identités et des accès AWS (IAM) spécifique.
- Pour renforcer la sécurité, nous vous recommandons de créer des clés d'accès IAM pour les identités IAM qui sont utilisées dans tous les AWS services. Le plug-in d'authentification Amazon Keyspaces SIGv4 pour les pilotes clients Cassandra vous permet d'authentifier les appels vers Amazon Keyspaces à l'aide de clés d'accès IAM au lieu d'un nom d'utilisateur et d'un mot de passe. Pour de plus amples informations, veuillez consulter [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#).

 Note

Pour un exemple d'utilisation d'Amazon Keyspaces avec Spring Boot, consultez <https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/java/datastax-v4/spring>.

Rubriques

- [Avant de commencer](#)
- [Step-by-step tutoriel pour se connecter à Amazon Keyspaces à l'aide du pilote DataStax Java pour Apache Cassandra à l'aide d'informations d'identification spécifiques au service](#)
- [Step-by-step tutoriel pour se connecter à Amazon Keyspaces à l'aide du pilote DataStax Java 4.x pour Apache Cassandra et du plug-in d'authentification Sigv4](#)
- [Connectez-vous à Amazon Keyspaces à l'aide du pilote DataStax Java 3.x pour Apache Cassandra et du plug-in d'authentification SIGv4](#)

Avant de commencer

Pour vous connecter à Amazon Keyspaces, vous devez effectuer les tâches suivantes avant de commencer.

1. Amazon Keyspaces nécessite l'utilisation du protocole TLS (Transport Layer Security) pour sécuriser les connexions avec les clients.
 - a. Téléchargez les certificats numériques suivants et enregistrez les fichiers localement ou dans votre répertoire personnel.
 1. AmazonRootCA1
 2. AmazonRootCA2
 3. AmazonRootCA3
 4. AmazonRootCA4
 5. Starfield Class 2 Root (en option, pour des raisons de rétrocompatibilité)

Pour télécharger les certificats, vous pouvez utiliser les commandes suivantes.

```
curl -O https://www.amazontrust.com/repository/AmazonRootCA1.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA2.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA3.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA4.pem
curl -O https://certs.secureserver.net/repository/sf-class2-root.crt
```

Note

Amazon Keyspaces utilisait auparavant des certificats TLS ancrés au Starfield Class 2 CA. AWS migre tous Régions AWS vers des certificats émis dans le cadre d'Amazon Trust Services (Amazon Root CA 1 à 4). Pendant cette transition, configurez les clients pour qu'ils fassent confiance à la fois aux autorités de certification Amazon Root 1 à 4 et à la racine Starfield afin de garantir la compatibilité entre toutes les régions.

- b. Convertissez les certificats numériques en fichiers TrustStore et ajoutez-les au keystore.

```
openssl x509 -outform der -in AmazonRootCA1.pem -out temp_file.der
```

```
keytool -import -alias amazon-root-ca-1 -keystore cassandra_truststore.jks -
file temp_file.der

openssl x509 -outform der -in AmazonRootCA2.pem -out temp_file.der
keytool -import -alias amazon-root-ca-2 -keystore cassandra_truststore.jks -
file temp_file.der

openssl x509 -outform der -in AmazonRootCA3.pem -out temp_file.der
keytool -import -alias amazon-root-ca-3 -keystore cassandra_truststore.jks -
file temp_file.der

openssl x509 -outform der -in AmazonRootCA4.pem -out temp_file.der
keytool -import -alias amazon-root-ca-4 -keystore cassandra_truststore.jks -
file temp_file.der

openssl x509 -outform der -in sf-class2-root.crt -out temp_file.der
keytool -import -alias cassandra -keystore cassandra_truststore.jks -file
temp_file.der
```

Dans la dernière étape, vous devez créer un mot de passe pour le keystore et faire confiance à chaque certificat. La commande interactive ressemble à ceci.

```
Enter keystore password:
Re-enter new password:
Owner: CN=Amazon Root CA 1, O=Amazon, C=US
Issuer: CN=Amazon Root CA 1, O=Amazon, C=US
Serial number: 66c9fcf99bf8c0a39e2f0788a43e696365bca
Valid from: Tue May 26 00:00:00 UTC 2015 until: Sun Jan 17 00:00:00 UTC 2038
Certificate fingerprints:
    SHA1: 8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
    SHA256:
    8E:CD:E6:88:4F:3D:87:B1:12:5B:A3:1A:C3:FC:B1:3D:70:16:DE:7F:57:CC:90:4F:E1:CB:97:C6:AE
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
    CA:true
    PathLen:2147483647
]
```

```
#2: ObjectId: 2.5.29.15 Criticality=true
```

```
KeyUsage [
```

```
    DigitalSignature
```

```
    Key_CertSign
```

```
    Crl_Sign
```

```
]
```

```
#3: ObjectId: 2.5.29.14 Criticality=false
```

```
SubjectKeyIdentifier [
```

```
KeyIdentifier [
```

```
0000: 84 18 CC 85 34 EC BC 0C    94 94 2E 08 59 9C C7 B2    ....4.....Y...
```

```
0010: 10 4E 0A 08                      .N..
```

```
]
```

```
]
```

```
Trust this certificate? [no]: yes
```

```
Certificate was added to keystore
```

```
Enter keystore password:
```

```
Owner: CN=Amazon Root CA 2, O=Amazon, C=US
```

```
Issuer: CN=Amazon Root CA 2, O=Amazon, C=US
```

```
Serial number: 66c9fd29635869f0a0fe58678f85b26bb8a37
```

```
Valid from: Tue May 26 00:00:00 UTC 2015 until: Sat May 26 00:00:00 UTC 2040
```

```
Certificate fingerprints:
```

```
    SHA1: 5A:8C:EF:45:D7:A6:98:59:76:7A:8C:8B:44:96:B5:78:CF:47:4B:1A
```

```
    SHA256:
```

```
    1B:A5:B2:AA:8C:65:40:1A:82:96:01:18:F8:0B:EC:4F:62:30:4D:83:CE:C4:71:3A:19:C3:9C:01:1E
```

```
Signature algorithm name: SHA384withRSA
```

```
Subject Public Key Algorithm: 4096-bit RSA key
```

```
Version: 3
```

```
Extensions:
```

```
#1: ObjectId: 2.5.29.19 Criticality=true
```

```
BasicConstraints:[
```

```
    CA:true
```

```
    PathLen:2147483647
```

```
]
```

```
#2: ObjectId: 2.5.29.15 Criticality=true
```

```
KeyUsage [
```

```
    DigitalSignature
```

```
    Key_CertSign
```

```
    Crl_Sign
```

```
]

```

```
#3: ObjectId: 2.5.29.14 Criticality=false

```

```
SubjectKeyIdentifier [

```

```
KeyIdentifier [

```

```
0000: B0 0C F0 4C 30 F4 05 58    02 48 FD 33 E5 52 AF 4B    ...L0..X.H.3.R.K

```

```
0010: 84 E3 66 52                                ..fR

```

```
]

```

```
]

```

```
Trust this certificate? [no]: yes

```

```
Certificate was added to keystore

```

```
Enter keystore password:

```

```
Owner: CN=Amazon Root CA 3, O=Amazon, C=US

```

```
Issuer: CN=Amazon Root CA 3, O=Amazon, C=US

```

```
Serial number: 66c9fd5749736663f3b0b9ad9e89e7603f24a

```

```
Valid from: Tue May 26 00:00:00 UTC 2015 until: Sat May 26 00:00:00 UTC 2040

```

```
Certificate fingerprints:

```

```
SHA1: 0D:44:DD:8C:3C:8C:1A:1A:58:75:64:81:E9:0F:2E:2A:FF:B3:D2:6E

```

```
SHA256:

```

```
18:CE:6C:FE:7B:F1:4E:60:B2:E3:47:B8:DF:E8:68:CB:31:D0:2E:BB:3A:DA:27:15:69:F5:03:43:B4

```

```
Signature algorithm name: SHA256withECDSA

```

```
Subject Public Key Algorithm: 256-bit EC (secp256r1) key

```

```
Version: 3

```

```
Extensions:

```

```
#1: ObjectId: 2.5.29.19 Criticality=true

```

```
BasicConstraints:[

```

```
CA:true

```

```
PathLen:2147483647

```

```
]

```

```
#2: ObjectId: 2.5.29.15 Criticality=true

```

```
KeyUsage [

```

```
DigitalSignature

```

```
Key_CertSign

```

```
Crl_Sign

```

```
]

```

```
#3: ObjectId: 2.5.29.14 Criticality=false

```

```
SubjectKeyIdentifier [

```

```
KeyIdentifier [

```

```
0000: AB B6 DB D7 06 9E 37 AC    30 86 07 91 70 C7 9C C4    .....7.0...p...

```

```

0010: 19 B1 78 C0                                     ...x.
]
]

Trust this certificate? [no]: yes
Certificate was added to keystore
Enter keystore password:
Owner: CN=Amazon Root CA 4, O=Amazon, C=US
Issuer: CN=Amazon Root CA 4, O=Amazon, C=US
Serial number: 66c9fd7c1bb104c2943e5717b7b2cc81ac10e
Valid from: Tue May 26 00:00:00 UTC 2015 until: Sat May 26 00:00:00 UTC 2040
Certificate fingerprints:
    SHA1: F6:10:84:07:D6:F8:BB:67:98:0C:C2:E2:44:C2:EB:AE:1C:EF:63:BE
    SHA256:
    E3:5D:28:41:9E:D0:20:25:CF:A6:90:38:CD:62:39:62:45:8D:A5:C6:95:FB:DE:A3:C2:2B:0B:FB:25
Signature algorithm name: SHA384withECDSA
Subject Public Key Algorithm: 384-bit EC (secp384r1) key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
    CA:true
    PathLen:2147483647
]

#2: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
    DigitalSignature
    Key_CertSign
    Crl_Sign
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: D3 EC C7 3A 65 6E CC E1    DA 76 9A 56 FB 9C F3 86    ...:en...v.V....
0010: 6D 57 E5 81                                     mW..
]
]

Trust this certificate? [no]: yes
Certificate was added to keystore

```

```

Enter keystore password:
Owner: OU=Starfield Class 2 Certification Authority, O="Starfield Technologies,
  Inc.", C=US
Issuer: OU=Starfield Class 2 Certification Authority, O="Starfield
  Technologies, Inc.", C=US
Serial number: 0
Valid from: Tue Jun 29 17:39:16 UTC 2004 until: Thu Jun 29 17:39:16 UTC 2034
Certificate fingerprints:
    SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
    SHA256:
    14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB
Signature algorithm name: SHA1withRSA (weak)
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
  KeyIdentifier [
0000: BF 5F B7 D1 CE DD 1F 86   F4 5B 55 AC DC D7 10 C2   ._.....[U.....
0010: 0E A9 88 E7                               ....
  ]
[OU=Starfield Class 2 Certification Authority, O="Starfield Technologies,
  Inc.", C=US]
SerialNumber: [    00]
]

#2: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
  KeyIdentifier [
0000: BF 5F B7 D1 CE DD 1F 86   F4 5B 55 AC DC D7 10 C2   ._.....[U.....
0010: 0E A9 88 E7                               ....
  ]
]

Warning:

```



```
The input uses the SHA1withRSA signature algorithm which is considered a
security risk. This algorithm will be disabled in a future update.
```

```
Trust this certificate? [no]: yes
Certificate was added to keystore
```

2. Joignez le fichier trustStore dans les arguments de la JVM :

```
-Djavax.net.ssl.trustStore=path_to_file/cassandra_truststore.jks
-Djavax.net.ssl.trustStorePassword=my_password
```

Step-by-step tutoriel pour se connecter à Amazon Keyspaces à l'aide du pilote DataStax Java pour Apache Cassandra à l'aide d'informations d'identification spécifiques au service

Le didacticiel étape par étape suivant vous explique comment vous connecter à Amazon Keyspaces à l'aide d'un pilote Java pour Cassandra à l'aide d'informations d'identification spécifiques au service. Plus précisément, vous utiliserez la version 4.0 du pilote DataStax Java pour Apache Cassandra.

Rubriques

- [Étape 1 : Prérequis](#)
- [Étape 2 : Configuration du pilote](#)
- [Étape 3 : Exécutez l'exemple d'application](#)

Étape 1 : Prérequis

Pour suivre ce didacticiel, vous devez générer des informations d'identification spécifiques au service et ajouter le pilote DataStax Java pour Apache Cassandra à votre projet Java.

- Générez des informations d'identification spécifiques au service pour votre utilisateur Amazon Keyspaces IAM en suivant les étapes décrites dans [the section called “Créez des informations d'identification spécifiques au service”](#). Si vous préférez utiliser des clés d'accès IAM pour l'authentification, consultez [the section called “Plugin d'authentification pour Java 4.x”](#).
- Ajoutez le pilote DataStax Java pour Apache Cassandra à votre projet Java. Assurez-vous que vous utilisez une version du pilote compatible avec Apache Cassandra 3.11.2. Pour plus d'informations, consultez la documentation [du pilote](#) DataStax Java pour Apache Cassandra.

Étape 2 : Configuration du pilote

Vous pouvez spécifier les paramètres du pilote DataStax Java Cassandra en créant un fichier de configuration pour votre application. Ce fichier de configuration remplace les paramètres par défaut et demande au pilote de se connecter au point de terminaison du service Amazon Keyspaces via le port 9142. Pour obtenir la liste des points de terminaison de service disponibles, consultez [the section called “Points de terminaison de service”](#).

Créez un fichier de configuration et enregistrez-le dans le dossier de ressources de l'application, par exemple. `src/main/resources/application.conf` Ouvrez `application.conf` et ajoutez les paramètres de configuration suivants.

1. Fournisseur d'authentification : créez le fournisseur d'authentification avec la `PlainTextAuthProvider` classe. `ServiceUserName` et `ServicePassword` doit correspondre au nom d'utilisateur et au mot de passe que vous avez obtenus lorsque vous avez généré les informations d'identification spécifiques au service en suivant les étapes décrites dans. [Créez des informations d'identification spécifiques au service pour un accès programmatique à Amazon Keyspaces](#)

Note

Vous pouvez utiliser des informations d'identification à court terme en utilisant le plug-in d'authentification du pilote DataStax Java pour Apache Cassandra au lieu de coder en dur les informations d'identification dans le fichier de configuration de votre pilote. Pour en savoir plus, suivez les instructions du [the section called “Plugin d'authentification pour Java 4.x”](#).

2. Centre de données local : définissez la valeur `local-datacenter` pour la région à laquelle vous vous connectez. Par exemple, si l'application se connecte à `cassandra.us-east-2.amazonaws.com`, configurez le centre de données local sur `us-east-2`. Pour tout ce qui est disponible Régions AWS, voir [???](#). Réglez `slow-replica-avoidance = false` pour équilibrer la charge par rapport à un plus petit nombre de nœuds.
3. SSL/TLS— Initialisez le `SSLConnectionFactory` en ajoutant une section dans le fichier de configuration avec une seule ligne qui spécifie la classe avec `class = DefaultSslConnectionFactory`. Indiquez le chemin d'accès au fichier `TrustStore` et le mot de passe que vous avez créé précédemment. Amazon Keyspaces ne prend pas en charge `hostname-validation` les pairs. Définissez cette option sur `false`.

```

datastax-java-driver {

    basic.contact-points = [ "cassandra.us-east-2.amazonaws.com:9142"]
    advanced.auth-provider{
        class = PlainTextAuthProvider
        username = "ServiceUserName"
        password = "ServicePassword"
    }
    basic.load-balancing-policy {
        local-datacenter = "us-east-2"
        slow-replica-avoidance = false
    }

    advanced.ssl-engine-factory {
        class = DefaultSslEngineFactory
        truststore-path = "./src/main/resources/cassandra_truststore.jks"
        truststore-password = "my_password"
        hostname-validation = false
    }
}

```

Note

Au lieu d'ajouter le chemin du TrustStore dans le fichier de configuration, vous pouvez également ajouter le chemin du TrustStore directement dans le code de l'application ou vous pouvez ajouter le chemin du TrustStore à vos arguments JVM.

Étape 3 : Exécutez l'exemple d'application

Cet exemple de code montre une application de ligne de commande simple qui crée un pool de connexions à Amazon Keyspaces à l'aide du fichier de configuration que nous avons créé précédemment. Il confirme que la connexion est établie en exécutant une requête simple.

```

package <your package>;
// add the following imports to your project
import com.datastax.oss.driver.api.core.CqlSession;
import com.datastax.oss.driver.api.core.config.DriverConfigLoader;
import com.datastax.oss.driver.api.core.cql.ResultSet;
import com.datastax.oss.driver.api.core.cql.Row;

```

```
public class App
{
    public static void main( String[] args )
    {
        //Use DriverConfigLoader to load your configuration file
        DriverConfigLoader loader =
        DriverConfigLoader.fromClasspath("application.conf");
        try (CqlSession session = CqlSession.builder()
            .withConfigLoader(loader)
            .build()) {

            ResultSet rs = session.execute("select * from system_schema.keyspaces");
            Row row = rs.one();
            System.out.println(row.getString("keyspace_name"));
        }
    }
}
```

Note

Utilisez un `try` bloc pour établir la connexion afin de vous assurer qu'elle est toujours fermée. Si vous n'utilisez pas de `try` blocage, pensez à fermer votre connexion pour éviter toute fuite de ressources.

Step-by-step tutoriel pour se connecter à Amazon Keyspaces à l'aide du pilote DataStax Java 4.x pour Apache Cassandra et du plug-in d'authentification Sigv4

La section suivante décrit comment utiliser le plug-in d'authentification Sigv4 pour le pilote DataStax Java 4.x open source pour Apache Cassandra afin d'accéder à Amazon Keyspaces (pour Apache Cassandra). Le plugin est disponible depuis le [GitHub référentiel](#).

Le plug-in d'authentification Sigv4 vous permet d'utiliser les informations d'identification IAM pour les utilisateurs ou les rôles lorsque vous vous connectez à Amazon Keyspaces. Au lieu de demander un nom d'utilisateur et un mot de passe, ce plugin signe les demandes d'API à l'aide de clés d'accès. Pour de plus amples informations, veuillez consulter [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#).

Étape 1 : Prérequis

Pour suivre ce didacticiel, vous devez effectuer les tâches suivantes.

- Si ce n'est pas déjà fait, créez des informations d'identification pour votre utilisateur ou votre rôle IAM en suivant les étapes décrites à [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#) la section. Ce didacticiel part du principe que les clés d'accès sont stockées en tant que variables d'environnement. Pour de plus amples informations, veuillez consulter [the section called “Gérer des clés d'accès”](#).
- Ajoutez le pilote DataStax Java pour Apache Cassandra à votre projet Java. Assurez-vous que vous utilisez une version du pilote compatible avec Apache Cassandra 3.11.2. Pour plus d'informations, consultez la documentation [du pilote](#) DataStax Java pour Apache Cassandra.
- Ajoutez le plug-in d'authentification à votre application. Le plug-in d'authentification prend en charge la version 4.x du pilote DataStax Java pour Apache Cassandra. Si vous utilisez Apache Maven, ou un système de génération qui peut utiliser les dépendances Maven, ajoutez les dépendances suivantes à votre fichier pom.xml.

Important

Remplacez la version du plugin par la dernière version, comme indiqué dans [GitHub le référentiel](#).

```
<dependency>
  <groupId>software.aws.mcs</groupId>
  <artifactId>aws-sigv4-auth-cassandra-java-driver-plugin</artifactId>
  <version>4.0.9</version>
</dependency>
```

Étape 2 : Configuration du pilote

Vous pouvez spécifier les paramètres du pilote DataStax Java Cassandra en créant un fichier de configuration pour votre application. Ce fichier de configuration remplace les paramètres par défaut et demande au pilote de se connecter au point de terminaison du service Amazon Keyspaces via le port 9142. Pour obtenir la liste des points de terminaison de service disponibles, consultez [the section called “Points de terminaison de service”](#).

Créez un fichier de configuration et enregistrez-le dans le dossier de ressources de l'application, par exemple `src/main/resources/application.conf`. Ouvrez `application.conf` et ajoutez les paramètres de configuration suivants.

1. Fournisseur d'authentification : définissez `advanced.auth-provider.class` le sur une nouvelle instance `software.aws.mcs.auth.SigV4AuthProvider`. `SigV4AuthProvider` Il s'agit du gestionnaire d'authentification fourni par le plugin pour effectuer l'authentification SIGv4.
2. Centre de données local : définissez la valeur `local-datacenter` pour la région à laquelle vous vous connectez. Par exemple, si l'application se connecte à `cassandra.us-east-2.amazonaws.com`, configurez le centre de données local sur `us-east-2`. Pour tout ce qui est disponible Régions AWS, voir [???](#). Réglez `slow-replica-avoidance` = `false` pour équilibrer la charge par rapport à tous les nœuds disponibles.
3. Idempotence — Définissez la valeur par défaut `idempotence` pour que l'application configure le pilote pour `true` qu'il réessaie toujours les demandes échouées ou `read/write` échouées. `prepare/execute` Il s'agit d'une bonne pratique pour les applications distribuées qui permet de gérer les défaillances transitoires en réessayant les demandes ayant échoué.
4. SSL/TLS— Initialisez le `SSLConnectionFactory` en ajoutant une section dans le fichier de configuration avec une seule ligne qui spécifie la classe avec `class = DefaultSslConnectionFactory`. Indiquez le chemin d'accès au fichier `TrustStore` et le mot de passe que vous avez créé précédemment. Amazon Keyspaces ne prend pas en charge `hostname-validation` les pairs. Définissez cette option sur `false`.
5. Connexions : créez au moins 3 connexions locales par point de terminaison en configurant `local.size` = 3. Il s'agit d'une bonne pratique qui aide votre application à gérer les frais généraux et les pics de trafic. Pour plus d'informations sur la façon de calculer le nombre de connexions locales par point de terminaison dont votre application a besoin en fonction des modèles de trafic attendus, consultez [the section called “Comment configurer les connexions”](#).
6. Politique de nouvelle tentative : mettez en œuvre la politique de nouvelle tentative d'Amazon Keyspaces `AmazonKeyspacesExponentialRetryPolicy` au lieu de `DefaultRetryPolicy` celle fournie avec le pilote Cassandra. Cela vous permet de configurer le nombre de nouvelles tentatives pour répondre `AmazonKeyspacesExponentialRetryPolicy` à vos besoins. Par défaut, le nombre de nouvelles tentatives pour le `AmazonKeyspacesExponentialRetryPolicy` est défini sur 3. Pour de plus amples informations, veuillez consulter [the section called “Comment configurer les politiques de nouvelle tentative”](#).

7. Instructions préparées : définissez sur `prepare-on-all-nodes` `False` pour optimiser l'utilisation du réseau.

```
datastax-java-driver {
  basic {
    contact-points = [ "cassandra.us-east-2.amazonaws.com:9142" ]
    request {
      timeout = 2 seconds
      consistency = LOCAL_QUORUM
      page-size = 1024
      default-idempotence = true
    }
    load-balancing-policy {
      local-datacenter = "us-east-2"
      class = DefaultLoadBalancingPolicy
      slow-replica-avoidance = false
    }
  }
  advanced {
    auth-provider {
      class = software.aws.mcs.auth.SigV4AuthProvider
      aws-region = us-east-2
    }
    ssl-engine-factory {
      class = DefaultSslEngineFactory
      truststore-path = "./src/main/resources/cassandra_truststore.jks"
      truststore-password = "my_password"
      hostname-validation = false
    }
    connection {
      connect-timeout = 5 seconds
      max-requests-per-connection = 512
      pool {
        local.size = 3
      }
    }
    retry-policy {
      class = com.aws.ssa.keyspaces.retry.AmazonKeyspacesExponentialRetryPolicy
    }
    max-attempts = 3
    min-wait = 10 mills
    max-wait = 100 mills
  }
}
```

```

    prepared-statements {
    prepare-on-all-nodes = false
    }
}
}

```

Note

Au lieu d'ajouter le chemin du TrustStore dans le fichier de configuration, vous pouvez également ajouter le chemin du TrustStore directement dans le code de l'application ou vous pouvez ajouter le chemin du TrustStore à vos arguments JVM.

Étape 3 : Exécutez l'application

Cet exemple de code montre une application de ligne de commande simple qui crée un pool de connexions à Amazon Keyspaces à l'aide du fichier de configuration que nous avons créé précédemment. Il confirme que la connexion est établie en exécutant une requête simple.

```

package <your package>;
// add the following imports to your project
import com.datastax.oss.driver.api.core.CqlSession;
import com.datastax.oss.driver.api.core.config.DriverConfigLoader;
import com.datastax.oss.driver.api.core.cql.ResultSet;
import com.datastax.oss.driver.api.core.cql.Row;

public class App
{
    public static void main( String[] args )
    {
        //Use DriverConfigLoader to load your configuration file
        DriverConfigLoader loader =
        DriverConfigLoader.fromClasspath("application.conf");
        try (CqlSession session = CqlSession.builder()
            .withConfigLoader(loader)
            .build()) {

            ResultSet rs = session.execute("select * from system_schema.keyspaces");
            Row row = rs.one();
            System.out.println(row.getString("keyspace_name"));

        }
    }
}

```



```
}  
}
```

Note

Utilisez un `try` bloc pour établir la connexion afin de vous assurer qu'elle est toujours fermée. Si vous n'utilisez pas de `try` blocage, pensez à fermer votre connexion pour éviter toute fuite de ressources.

Connectez-vous à Amazon Keyspaces à l'aide du pilote DataStax Java 3.x pour Apache Cassandra et du plug-in d'authentification SIGv4

La section suivante décrit comment utiliser le plug-in d'authentification Sigv4 pour le pilote DataStax Java open source 3.x pour Apache Cassandra afin d'accéder à Amazon Keyspaces. Le plugin est disponible depuis le [GitHub référentiel](#).

Le plug-in d'authentification Sigv4 vous permet d'utiliser les informations d'identification IAM pour les utilisateurs et les rôles lorsque vous vous connectez à Amazon Keyspaces. Au lieu de demander un nom d'utilisateur et un mot de passe, ce plugin signe les demandes d'API à l'aide de clés d'accès. Pour de plus amples informations, veuillez consulter [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#).

Étape 1 : Prérequis

Pour exécuter cet exemple de code, vous devez d'abord effectuer les tâches suivantes.

- Créez des informations d'identification pour votre utilisateur ou votre rôle IAM en suivant les étapes décrites à [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#). Ce didacticiel part du principe que les clés d'accès sont stockées en tant que variables d'environnement. Pour de plus amples informations, veuillez consulter [the section called “Gérer des clés d'accès”](#).
- Suivez les étapes décrites [the section called “Avant de commencer”](#) à l'adresse pour télécharger les certificats numériques, les convertir en fichiers TrustStore et joindre le keystore dans les arguments de la JVM à votre application.
- Ajoutez le pilote DataStax Java pour Apache Cassandra à votre projet Java. Assurez-vous que vous utilisez une version du pilote compatible avec Apache Cassandra 3.11.2. Pour plus d'informations, consultez la documentation [du pilote](#) DataStax Java pour Apache Cassandra.

- Ajoutez le plug-in d'authentification à votre application. Le plug-in d'authentification prend en charge la version 3.x du pilote DataStax Java pour Apache Cassandra. Si vous utilisez Apache Maven, ou un système de génération qui peut utiliser les dépendances Maven, ajoutez les dépendances suivantes à votre fichier `pom.xml`. Remplacez la version du plugin par la dernière version, comme indiqué dans [GitHub le référentiel](#).

```
<dependency>
  <groupId>software.aws.mcs</groupId>
  <artifactId>aws-sigv4-auth-cassandra-java-driver-plugin_3</artifactId>
  <version>3.0.3</version>
</dependency>
```

Étape 2 : Exécutez l'application

Cet exemple de code montre une application de ligne de commande simple qui crée un pool de connexions à Amazon Keyspaces. Il confirme que la connexion est établie en exécutant une requête simple.

```
package <your package>;
// add the following imports to your project

import software.aws.mcs.auth.SigV4AuthProvider;
import com.datastax.driver.core.Cluster;
import com.datastax.driver.core.ResultSet;
import com.datastax.driver.core.Row;
import com.datastax.driver.core.Session;

public class App
{
    public static void main( String[] args )
    {
        String endPoint = "cassandra.us-east-2.amazonaws.com";
        int portNumber = 9142;
        Session session = Cluster.builder()
                                .addContactPoint(endPoint)
                                .withPort(portNumber)
                                .withAuthProvider(new SigV4AuthProvider("us-east-2"))

                                .withSSL()
                                .build()
```

```
        .connect();

        ResultSet rs = session.execute("select * from system_schema.keyspaces");
        Row row = rs.one();
        System.out.println(row.getString("keyspace_name"));
    }
}
```

Remarques d'utilisation :

Pour obtenir la liste des points de terminaison disponibles, reportez-vous à la section [the section called “Points de terminaison de service”](#).

Consultez le référentiel suivant pour des politiques, des exemples et des meilleures pratiques utiles concernant les pilotes Java lors de l'utilisation du pilote Java avec Amazon Keyspaces : <https://github.com/aws-samples/amazon-keyspaces-java-driver-helpers>.

Utilisation d'un pilote client Cassandra Python pour accéder à Amazon Keyspaces par programmation

Dans cette section, nous vous montrons comment vous connecter à Amazon Keyspaces à l'aide d'un pilote client Python. Pour fournir aux utilisateurs et aux applications des informations d'identification leur permettant d'accéder par programmation aux ressources Amazon Keyspaces, vous pouvez effectuer l'une des opérations suivantes :

- Créez des informations d'identification spécifiques au service associées à un utilisateur Gestion des identités et des accès AWS (IAM) spécifique.
- Pour renforcer la sécurité, nous vous recommandons de créer des clés d'accès IAM pour les utilisateurs ou les rôles IAM utilisés dans tous les AWS services. Le plug-in d'authentification Amazon Keyspaces SIGv4 pour les pilotes clients Cassandra vous permet d'authentifier les appels vers Amazon Keyspaces à l'aide de clés d'accès IAM au lieu d'un nom d'utilisateur et d'un mot de passe. Pour de plus amples informations, veuillez consulter [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#).

Rubriques

- [Avant de commencer](#)
- [Connectez-vous à Amazon Keyspaces à l'aide du pilote Python pour Apache Cassandra et des informations d'identification spécifiques au service](#)

- [Connectez-vous à Amazon Keyspaces à l'aide du pilote DataStax Python pour Apache Cassandra et du plug-in d'authentification SIGv4](#)

Avant de commencer

Vous devez effectuer la tâche suivante avant de pouvoir commencer.

Amazon Keyspaces nécessite l'utilisation du protocole TLS (Transport Layer Security) pour sécuriser les connexions avec les clients. Pour vous connecter à Amazon Keyspaces via TLS, vous devez télécharger les certificats numériques Amazon et configurer le pilote Python pour utiliser TLS.

Téléchargez les certificats numériques suivants et enregistrez les fichiers localement ou dans votre répertoire personnel.

1. AmazonRootCA1
2. AmazonRootCA2
3. AmazonRootCA3
4. AmazonRootCA4
5. Starfield Class 2 Root (en option, pour des raisons de rétrocompatibilité)

Pour télécharger les certificats, vous pouvez utiliser les commandes suivantes.

```
curl -O https://www.amazontrust.com/repository/AmazonRootCA1.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA2.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA3.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA4.pem
curl -O https://certs.secyreserver.net/repository/sf-class2-root.crt
```

Note

Amazon Keyspaces utilisait auparavant des certificats TLS ancrés dans le Starfield Class 2 CA. AWS migre tous Régions AWS vers des certificats émis dans le cadre d'Amazon Trust Services (Amazon Root CA 1 à 4). Pendant cette transition, configurez les clients pour qu'ils fassent confiance à la fois aux autorités de certification Amazon Root 1 à 4 et à la racine Starfield afin de garantir la compatibilité entre toutes les régions.

Combinez tous les certificats téléchargés dans un seul pem fichier avec le nom *keyspaces-bundle.pem* indiqué dans nos exemples. Pour ce faire, exécutez la commande suivante : Prenez note du chemin d'accès au fichier, vous en aurez besoin plus tard.

```
cat AmazonRootCA1.pem \  
    AmazonRootCA2.pem \  
    AmazonRootCA3.pem \  
    AmazonRootCA4.pem \  
    sf-class2-root.crt \  
> keyspaces-bundle.pem
```

Connectez-vous à Amazon Keyspaces à l'aide du pilote Python pour Apache Cassandra et des informations d'identification spécifiques au service

L'exemple de code suivant vous montre comment vous connecter à Amazon Keyspaces à l'aide d'un pilote client Python et d'informations d'identification spécifiques au service.

```
from cassandra.cluster import Cluster  
from ssl import SSLContext, PROTOCOL_TLSv1_2 , CERT_REQUIRED  
from cassandra.auth import PlainTextAuthProvider  
  
ssl_context = SSLContext(PROTOCOL_TLSv1_2 )  
  
ssl_context.load_verify_locations('path_to_file/keyspaces-bundle.pem')  
  
ssl_context.verify_mode = CERT_REQUIRED  
auth_provider = PlainTextAuthProvider(username='ServiceUserName',  
    password='ServicePassword')  
cluster = Cluster(['cassandra.us-east-2.amazonaws.com'], ssl_context=ssl_context,  
    auth_provider=auth_provider, port=9142)  
session = cluster.connect()  
r = session.execute('select * from system_schema.keyspaces')  
print(r.current_rows)
```

Remarques d'utilisation :

1. Remplacez-le "*path_to_file/keyspaces-bundle.pem*" par le chemin d'accès au fichier de certificat combiné enregistré lors de la première étape.
2. Assurez-vous que le nom d'utilisateur *ServiceUserName* et le mot de passe que vous avez obtenus lors de la génération des informations d'identification spécifiques au service et qu'ils

ServicePassword correspondent à ceux que vous avez obtenus en suivant les étapes ci-dessous. [Créez des informations d'identification spécifiques au service pour un accès programmatique à Amazon Keyspaces](#)

3. Pour obtenir la liste des points de terminaison disponibles, reportez-vous à la section [the section called “Points de terminaison de service”](#).

Connectez-vous à Amazon Keyspaces à l'aide du pilote DataStax Python pour Apache Cassandra et du plug-in d'authentification SIGv4

La section suivante montre comment utiliser le plug-in d'authentification Sigv4 pour le pilote DataStax Python open source pour Apache Cassandra afin d'accéder à Amazon Keyspaces (pour Apache Cassandra).

Si ce n'est pas déjà fait, commencez par créer des informations d'identification pour votre rôle IAM en suivant les étapes décrites à [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#). Ce didacticiel utilise des informations d'identification temporaires, qui nécessitent un rôle IAM. Pour plus d'informations sur les informations d'identification temporaires, consultez [the section called “Créez des informations d'identification temporaires pour vous connecter à Amazon Keyspaces”](#).

Ajoutez ensuite le plug-in d'authentification Python SIGv4 à votre environnement depuis le [GitHub référentiel](#).

```
pip install cassandra-sigv4
```

L'exemple de code suivant montre comment se connecter à Amazon Keyspaces à l'aide du pilote DataStax Python open source pour Cassandra et du plug-in d'authentification SigV4. Le plugin dépend du AWS SDK pour Python (Boto3). Il permet `boto3.session` d'obtenir des informations d'identification temporaires.

```
from cassandra.cluster import Cluster
from ssl import SSLContext, PROTOCOL_TLSv1_2 , CERT_REQUIRED
from cassandra.auth import PlainTextAuthProvider
import boto3
from cassandra_sigv4.auth import SigV4AuthProvider

ssl_context = SSLContext(PROTOCOL_TLSv1_2)
ssl_context.load_verify_locations('path_to_file/keyspaces-bundle.pem')
```

```

ssl_context.verify_mode = CERT_REQUIRED

# use this if you want to use Boto to set the session parameters.
boto_session = boto3.Session(aws_access_key_id="AKIAIOSFODNN7EXAMPLE",
                             aws_secret_access_key="wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY",
                             aws_session_token="AQoDYXdzEJr...<remainder of token>",
                             region_name="us-east-2")
auth_provider = SigV4AuthProvider(boto_session)

# Use this instead of the above line if you want to use the Default Credentials and not
# bother with a session.
# auth_provider = SigV4AuthProvider()

cluster = Cluster(['cassandra.us-east-2.amazonaws.com'], ssl_context=ssl_context,
                  auth_provider=auth_provider,
                  port=9142)
session = cluster.connect()
r = session.execute('select * from system_schema.keyspaces')
print(r.current_rows)

```

Remarques d'utilisation :

1. "*path_to_file/keyspaces-bundle.pem*" Remplacez-le par le chemin d'accès au certificat enregistré lors de la première étape.
2. Assurez-vous que le *aws_access_key_id*, et le *aws_secret_access_key*, *aws_session_token* correspondent au Access Key Secret Access Key, et Session Token que vous avez obtenu en utilisant `boto3.session`. Pour de plus amples informations, veuillez consulter [Informations d'identification](#) dans le AWS SDK pour Python (Boto3).
3. Pour obtenir la liste des points de terminaison disponibles, reportez-vous à la section [the section called "Points de terminaison de service"](#).

Utilisation d'un pilote Node.js client Cassandra pour accéder à Amazon Keyspaces par programmation

Cette section explique comment vous connecter à Amazon Keyspaces à l'aide d'un pilote Node.js client. Pour fournir aux utilisateurs et aux applications des informations d'identification leur permettant d'accéder par programmation aux ressources Amazon Keyspaces, vous pouvez effectuer l'une des opérations suivantes :

- Créez des informations d'identification spécifiques au service associées à un utilisateur Gestion des identités et des accès AWS (IAM) spécifique.
- Pour renforcer la sécurité, nous vous recommandons de créer des clés d'accès IAM pour les utilisateurs ou les rôles IAM utilisés dans tous les AWS services. Le plug-in d'authentification Amazon Keyspaces SIGv4 pour les pilotes clients Cassandra vous permet d'authentifier les appels vers Amazon Keyspaces à l'aide de clés d'accès IAM au lieu d'un nom d'utilisateur et d'un mot de passe. Pour de plus amples informations, veuillez consulter [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#).

Rubriques

- [Avant de commencer](#)
- [Connectez-vous à Amazon Keyspaces à l'aide du Node.js DataStax pilote pour Apache Cassandra et des informations d'identification spécifiques au service](#)
- [Connectez-vous à Amazon Keyspaces à l'aide du DataStax Node.js pilote pour Apache Cassandra et du plug-in d'authentification SIGv4](#)

Avant de commencer

Vous devez effectuer la tâche suivante avant de pouvoir commencer.

Amazon Keyspaces nécessite l'utilisation du protocole TLS (Transport Layer Security) pour sécuriser les connexions avec les clients. Pour vous connecter à Amazon Keyspaces à l'aide du protocole TLS, vous devez télécharger un certificat numérique Amazon et configurer le pilote Python pour qu'il utilise le protocole TLS.

Téléchargez les certificats numériques suivants et enregistrez les fichiers localement ou dans votre répertoire personnel.

1. AmazonRootCA1
2. AmazonRootCA2
3. AmazonRootCA3
4. AmazonRootCA4
5. Starfield Class 2 Root (en option, pour des raisons de rétrocompatibilité)

Pour télécharger les certificats, vous pouvez utiliser les commandes suivantes.


```
curl -O https://www.amazontrust.com/repository/AmazonRootCA1.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA2.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA3.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA4.pem
curl -O https://certs.secureserver.net/repository/sf-class2-root.crt
```

Note

Amazon Keyspaces utilisait auparavant des certificats TLS ancrés dans le Starfield Class 2 CA. AWS migre tous Régions AWS vers des certificats émis dans le cadre d'Amazon Trust Services (Amazon Root CA 1 à 4). Pendant cette transition, configurez les clients pour qu'ils fassent confiance à la fois aux autorités de certification Amazon Root 1 à 4 et à la racine Starfield afin de garantir la compatibilité entre toutes les régions.

Combinez tous les certificats téléchargés dans un seul pem fichier avec le nom *keyspaces-bundle.pem* indiqué dans nos exemples. Pour ce faire, exécutez la commande suivante : Prenez note du chemin d'accès au fichier, vous en aurez besoin plus tard.

```
cat AmazonRootCA1.pem \
    AmazonRootCA2.pem \
    AmazonRootCA3.pem \
    AmazonRootCA4.pem \
    sf-class2-root.crt \
    > keyspaces-bundle.pem
```

Connectez-vous à Amazon Keyspaces à l'aide du Node.js DataStax pilote pour Apache Cassandra et des informations d'identification spécifiques au service

Configurez votre pilote pour utiliser le fichier de certificat combiné *keyspaces-bundle.pem* pour TLS et authentifiez-vous à l'aide d'informations d'identification spécifiques au service. Par exemple :

```
const cassandra = require('cassandra-driver');
const fs = require('fs');
const auth = new cassandra.auth.PlainTextAuthProvider('ServiceUserName',
    'ServicePassword');
const sslOptions1 = {
    ca: [
        fs.readFileSync('path_to_file/keyspaces-bundle.pem', 'utf-8')],
```

```
        host: 'cassandra.us-west-2.amazonaws.com',
        rejectUnauthorized: true
    };
    const client = new cassandra.Client({
        contactPoints: ['cassandra.us-west-2.amazonaws.com'],
        localDataCenter: 'us-west-2',
        authProvider: auth,
        sslOptions: sslOptions1,
        protocolOptions: { port: 9142 }
    });
    const query = 'SELECT * FROM system_schema.keyspaces';

    client.execute(query)
        .then( result => console.log('Row from Keyspaces %s',
            result.rows[0]))
        .catch( e=> console.log(`${e}`));
```

Remarques d'utilisation :

1. Remplacez-le "*path_to_file/keyspaces-bundle.pem*" par le chemin d'accès au fichier de certificat combiné enregistré lors de la première étape.
2. Assurez-vous que le nom d'utilisateur *ServiceUserName* et le mot de passe que vous avez obtenus lors de la génération des informations d'identification spécifiques au service et qu'ils *ServicePassword* correspondent à ceux que vous avez obtenus en suivant les étapes ci-dessous. [Créez des informations d'identification spécifiques au service pour un accès programmatique à Amazon Keyspaces](#)
3. Pour obtenir la liste des points de terminaison disponibles, reportez-vous à la section [the section called "Points de terminaison de service"](#).

Connectez-vous à Amazon Keyspaces à l'aide du DataStax Node.js pilote pour Apache Cassandra et du plug-in d'authentification SIGv4

La section suivante montre comment utiliser le plug-in d'authentification Sigv4 pour le DataStax Node.js pilote open source pour Apache Cassandra afin d'accéder à Amazon Keyspaces (pour Apache Cassandra).

Si ce n'est pas déjà fait, créez des informations d'identification pour votre utilisateur ou votre rôle IAM en suivant les étapes décrites à [the section called "Création d'informations d'identification IAM pour l'AWS authentication"](#) la section.

Ajoutez le plug-in d'authentification Node.js SigV4 à votre application depuis le [GitHub référentiel](#). Le plugin prend en charge la version 4.x du DataStax Node.js pilote pour Cassandra et dépend du AWS SDK pour Node.js. Il est utilisé `AWSCredentialsProvider` pour obtenir des informations d'identification.

```
$ npm install aws-sigv4-auth-cassandra-plugin --save
```

Cet exemple de code montre comment définir une Region-specific instance de `SigV4AuthProvider` comme fournisseur d'authentification.

```
const cassandra = require('cassandra-driver');
const fs = require('fs');
const sigV4 = require('aws-sigv4-auth-cassandra-plugin');

const auth = new sigV4.SigV4AuthProvider({
  region: 'us-west-2',
  accessKeyId: 'AKIAIOSFODNN7EXAMPLE',
  secretAccessKey: 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'});

const sslOptions1 = {
  ca: [
    fs.readFileSync('path_to_file/keyspaces-bundle.pem', 'utf-8')],
  host: 'cassandra.us-west-2.amazonaws.com',
  rejectUnauthorized: true
};

const client = new cassandra.Client({
  contactPoints: ['cassandra.us-west-2.amazonaws.com'],
  localDataCenter: 'us-west-2',
  authProvider: auth,
  sslOptions: sslOptions1,
  protocolOptions: { port: 9142 }
});

const query = 'SELECT * FROM system_schema.keyspaces';

client.execute(query).then(
  result => console.log('Row from Keyspaces %s', result.rows[0]))
  .catch( e=> console.log(`${e}`));
```

Remarques d'utilisation :

1. "*path_to_file/keyspaces-bundle.pem*" Remplacez-le par le chemin d'accès au certificat enregistré lors de la première étape.
2. Assurez-vous que la clé d'accès *accessKeyId secretAccessKey* et la clé d'accès secrète que vous avez obtenues à l'aide de `AWSCredentialsProvider`. Pour plus d'informations, consultez la section [Configuration des informations d'identification Node.js](#) JavaScript dans Node.js le AWS SDK.
3. Pour stocker les clés d'accès en dehors du code, consultez les meilleures pratiques à l'adresse [the section called "Gérer des clés d'accès"](#).
4. Pour obtenir la liste des points de terminaison disponibles, reportez-vous à la section [the section called "Points de terminaison de service"](#).

Utilisation d'un pilote client Cassandra .NET Core pour accéder à Amazon Keyspaces par programmation

Cette section explique comment vous connecter à Amazon Keyspaces à l'aide d'un pilote client .NET Core. Les étapes de configuration varient en fonction de votre environnement et de votre système d'exploitation. Vous devrez peut-être les modifier en conséquence. Amazon Keyspaces nécessite l'utilisation du protocole TLS (Transport Layer Security) pour sécuriser les connexions avec les clients. Pour vous connecter à Amazon Keyspaces à l'aide du protocole TLS, configurez votre pilote pour qu'il utilise le système Trust Store, qui inclut les autorités de certification Amazon Root 1 à 4.

1. Installez le Nuget `CassandraCSharpDriver` Through à l'aide de la console Nuget.

```
PM> Install-Package CassandraCSharpDriver
```

2. L'exemple suivant utilise un projet de console .NET Core C# pour se connecter à Amazon Keyspaces et exécuter une requête.

```
using Cassandra;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Security;
using System.Runtime.ConstrainedExecution;
using System.Security.Cryptography.X509Certificates;
using System.Text;
```

```
using System.Threading.Tasks;

namespace CSharpKeyspacesExample
{
    class Program
    {
        public Program(){}

        static void Main(string[] args)
        {
            var userName = "ServiceUserName";
            var pwd = "ServicePassword";
            certCollection.Add(amazoncert);

            var awsEndpoint = "cassandra.us-east-2.amazonaws.com" ;

            var cluster = Cluster.Builder()
                .AddContactPoints(awsEndpoint)
                .WithPort(9142)
                .WithAuthProvider(new PlainTextAuthProvider(userName, pwd))
                .WithSSL(new
SSLOptions().SetCertificateCollection(certCollection))
                .Build();

            var session = cluster.Connect();
            var rs = session.Execute("SELECT * FROM system_schema.tables;");
            foreach (var row in rs)
            {
                var name = row.GetValue<String>("keyspace_name");
                Console.WriteLine(name);
            }
        }
    }
}
```

Remarques d'utilisation :

1. Assurez-vous d'utiliser le système Trust Store par défaut, qui inclut les Amazon Root CA 1-4.
2. Assurez-vous que le nom d'utilisateur *ServiceUserName* et le mot de passe que vous avez obtenus lors de la génération des informations d'identification spécifiques au service et qu'ils *ServicePassword* correspondent à ceux que vous avez obtenus en suivant les étapes

ci-dessous. [Créez des informations d'identification spécifiques au service pour un accès programmatique à Amazon Keyspaces](#)

3. Pour obtenir la liste des points de terminaison disponibles, reportez-vous à la section [the section called “Points de terminaison de service”](#).

Utilisation d'un pilote client Cassandra Go pour accéder à Amazon Keyspaces par programmation

Cette section explique comment vous connecter à Amazon Keyspaces à l'aide d'un pilote client Go Cassandra. Pour fournir aux utilisateurs et aux applications des informations d'identification leur permettant d'accéder par programmation aux ressources Amazon Keyspaces, vous pouvez effectuer l'une des opérations suivantes :

- Créez des informations d'identification spécifiques au service associées à un utilisateur Gestion des identités et des accès AWS (IAM) spécifique.
- Pour renforcer la sécurité, nous vous recommandons de créer des clés d'accès IAM pour les principaux IAM qui sont utilisées dans tous les services. AWS Le plug-in d'authentification Amazon Keyspaces SIGv4 pour les pilotes clients Cassandra vous permet d'authentifier les appels vers Amazon Keyspaces à l'aide de clés d'accès IAM au lieu d'un nom d'utilisateur et d'un mot de passe. Pour de plus amples informations, veuillez consulter [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#).

Rubriques

- [Avant de commencer](#)
- [Connectez-vous à Amazon Keyspaces à l'aide du pilote Gocql pour Apache Cassandra et des informations d'identification spécifiques au service](#)
- [Connectez-vous à Amazon Keyspaces à l'aide du pilote Go pour Apache Cassandra et du plug-in d'authentification SIGv4](#)

Avant de commencer

Vous devez effectuer la tâche suivante avant de pouvoir commencer.

Amazon Keyspaces nécessite l'utilisation du protocole TLS (Transport Layer Security) pour sécuriser les connexions avec les clients. Pour vous connecter à Amazon Keyspaces via TLS, vous devez

télécharger un certificat numérique Amazon et configurer le pilote Go pour qu'il utilise le protocole TLS.

Téléchargez les certificats numériques suivants et enregistrez les fichiers localement ou dans votre répertoire personnel.

1. AmazonRootCA1
2. AmazonRootCA2
3. AmazonRootCA3
4. AmazonRootCA4
5. Starfield Class 2 Root (en option, pour des raisons de rétrocompatibilité)

Pour télécharger les certificats, vous pouvez utiliser les commandes suivantes.

```
curl -O https://www.amazontrust.com/repository/AmazonRootCA1.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA2.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA3.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA4.pem
curl -O https://certs.secureserver.net/repository/sf-class2-root.crt
```

Note

Amazon Keyspaces utilisait auparavant des certificats TLS ancrés dans le Starfield Class 2 CA. AWS migre tous Régions AWS vers des certificats émis dans le cadre d'Amazon Trust Services (Amazon Root CA 1 à 4). Pendant cette transition, configurez les clients pour qu'ils fassent confiance à la fois aux autorités de certification Amazon Root 1 à 4 et à la racine Starfield afin de garantir la compatibilité entre toutes les régions.

Combinez tous les certificats téléchargés dans un seul pem fichier avec le nom *keyspaces-bundle.pem* indiqué dans nos exemples. Pour ce faire, exécutez la commande suivante : Prenez note du chemin d'accès au fichier, vous en aurez besoin plus tard.

```
cat AmazonRootCA1.pem \
    AmazonRootCA2.pem \
    AmazonRootCA3.pem \
    AmazonRootCA4.pem \
    sf-class2-root.crt \
```

```
> keyspaces-bundle.pem
```

Connectez-vous à Amazon Keyspaces à l'aide du pilote Gocql pour Apache Cassandra et des informations d'identification spécifiques au service

1. Créez un nouveau répertoire pour votre application.

```
mkdir ./gocqlexample
```

2. Accédez au nouveau répertoire.

```
cd gocqlexample
```

3. Créez un dossier pour votre candidature.

```
touch cqlapp.go
```

4. Téléchargez le pilote Go.

```
go get github.com/gocql/gocql
```

5. Ajoutez l'exemple de code suivant au fichier cqlapp.go.

```
package main

import (
    "fmt"
    "github.com/gocql/gocql"
    "log"
)

func main() {

    // add the Amazon Keyspaces service endpoint
    cluster := gocql.NewCluster("cassandra.us-east-2.amazonaws.com")
    cluster.Port=9142
    // add your service specific credentials
    cluster.Authenticator = gocql.PasswordAuthenticator{
        Username: "ServiceUserName",
        Password: "ServicePassword"}

    // provide the path to the keyspaces-bundle.pem
```



```

cluster.SslOpts = &gocql.SslOptions{
    CaPath: "path_to_file/keyspaces-bundle.pem",
    EnableHostVerification: false,
}

// Override default Consistency to LocalQuorum
cluster.Consistency = gocql.LocalQuorum
cluster.DisableInitialHostLookup = false

session, err := cluster.CreateSession()
if err != nil {
    fmt.Println("err>", err)
}
defer session.Close()

// run a sample query from the system keyspace
var text string
iter := session.Query("SELECT keyspace_name FROM system_schema.tables;").Iter()
for iter.Scan(&text) {
    fmt.Println("keyspace_name:", text)
}
if err := iter.Close(); err != nil {
    log.Fatal(err)
}
session.Close()
}

```

Remarques d'utilisation :

- a. Remplacez-le "*path_to_file/keyspaces-bundle.pem*" par le chemin d'accès au fichier de certificat combiné enregistré lors de la première étape.
 - b. Assurez-vous que le nom d'utilisateur *ServiceUserName* et le mot de passe que vous avez obtenus lors de la génération des informations d'identification spécifiques au service et qu'ils *ServicePassword* correspondent à ceux que vous avez obtenus en suivant les étapes ci-dessous. [Créez des informations d'identification spécifiques au service pour un accès programmatique à Amazon Keyspaces](#)
 - c. Pour obtenir la liste des points de terminaison disponibles, reportez-vous à la section [the section called "Points de terminaison de service"](#).
6. Créez le programme.

```
go build cqldapp.go
```

7. Exécutez le programme.

```
./cqclapp
```

Connectez-vous à Amazon Keyspaces à l'aide du pilote Go pour Apache Cassandra et du plug-in d'authentification SIGv4

L'exemple de code suivant montre comment utiliser le plug-in d'authentification Sigv4 pour le pilote Go open source afin d'accéder à Amazon Keyspaces (pour Apache Cassandra).

Si ce n'est pas déjà fait, créez des informations d'identification pour votre principal IAM en suivant les étapes décrites à [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#). Si une application s'exécute sur Lambda ou sur une instance Amazon EC2, votre application utilise automatiquement les informations d'identification de l'instance. Pour exécuter ce didacticiel localement, vous pouvez stocker les informations d'identification sous forme de variables d'environnement locales.

Ajoutez le plug-in d'authentification Go SigV4 à votre application depuis le [GitHub référentiel](#). Le plugin prend en charge la version 1.2.x du pilote Go open source pour Cassandra et dépend du AWS SDK pour Go.

```
$ go mod init
$ go get github.com/aws/aws-sigv4-auth-cassandra-gocql-driver-plugin
```

Dans cet exemple de code, le point de terminaison Amazon Keyspaces est représenté par la `Cluster` classe. Il utilise la propriété `AwsAuthenticator` for the authenticator du cluster pour obtenir des informations d'identification.

```
package main

import (
    "fmt"
    "github.com/aws/aws-sigv4-auth-cassandra-gocql-driver-plugin/sigv4"
    "github.com/gocql/gocql"
    "log"
)

func main() {
    // configuring the cluster options
```

```

cluster := gocql.NewCluster("cassandra.us-west-2.amazonaws.com")
cluster.Port=9142

// the authenticator uses the default credential chain to find AWS credentials
cluster.Authenticator = sigv4.NewAwsAuthenticator()

cluster.SslOpts = &gocql.SslOptions{
    CaPath: "path_to_file/keyspaces-bundle.pem",
    EnableHostVerification: false,
}
cluster.Consistency = gocql.LocalQuorum
cluster.DisableInitialHostLookup = false

session, err := cluster.CreateSession()
if err != nil {
    fmt.Println("err>", err)
    return
}
defer session.Close()

// doing the query
var text string
iter := session.Query("SELECT keyspace_name FROM system_schema.tables;").Iter()
for iter.Scan(&text) {
    fmt.Println("keyspace_name:", text)
}
if err := iter.Close(); err != nil {
    log.Fatal(err)
}
}

```

Remarques d'utilisation :

1. Remplacez-le "*path_to_file/keyspaces-bundle.pem*" par le chemin d'accès au fichier de certificat enregistré lors de la première étape.
2. Pour que cet exemple soit exécuté localement, vous devez définir les variables suivantes en tant que variables d'environnement :
 - AWS_ACCESS_KEY_ID
 - AWS_SECRET_ACCESS_KEY
 - AWS_DEFAULT_REGION

3. Pour stocker les clés d'accès en dehors du code, consultez les meilleures pratiques à l'adresse [the section called “Gérer des clés d'accès”](#).
4. Pour obtenir la liste des points de terminaison disponibles, reportez-vous à la section [the section called “Points de terminaison de service”](#).

Utilisation d'un pilote client Cassandra Perl pour accéder à Amazon Keyspaces par programmation

Cette section explique comment vous connecter à Amazon Keyspaces à l'aide d'un pilote client Perl. Pour cet exemple de code, nous avons utilisé Perl 5. Amazon Keyspaces nécessite l'utilisation du protocole TLS (Transport Layer Security) pour sécuriser les connexions avec les clients.

Important

Les certificats Amazon Keyspaces sont en cours de transition vers la hiérarchie Amazon Trust Services (ATS). Assurez-vous que votre environnement fait confiance aux autorités de certification Amazon Root 1 à 4 pour éviter les erreurs de connexion lors de cette rotation. Le pilote Perl ne valide pas le certificat Amazon SSL du serveur, ce qui signifie que vous ne pouvez pas confirmer que vous vous connectez à Amazon Keyspaces. La deuxième étape, qui consiste à configurer le pilote pour qu'il utilise le protocole TLS lors de la connexion à Amazon Keyspaces, est toujours requise et garantit que les données transférées entre le client et le serveur sont cryptées.

1. Téléchargez le pilote Cassandra DBI depuis <https://metacpan.org/pod/DBD::Cassandra> et installez-le dans votre environnement Perl. Les étapes exactes dépendent de l'environnement. Voici un exemple courant.

```
cpanm DBD::Cassandra
```

2. Créez un dossier pour votre candidature.

```
touch cqlapp.pl
```

3. Ajoutez l'exemple de code suivant au fichier cqlapp.pl.

```
use DBI;  
my $user = "ServiceUserName";
```

```
my $password = "ServicePassword";
my $db = DBI->connect("dbi:Cassandra:host=cassandra.us-
east-2.amazonaws.com;port=9142;tls=1;",
$user, $password);

my $rows = $db->selectall_arrayref("select * from system_schema.keyspaces");
print "Found the following Keyspaces...\n";
for my $row (@$rows) {
    print join(" ",@$row['keyspace_name']),"\n";
}

$db->disconnect;
```

Important

Assurez-vous que le nom d'utilisateur *ServiceUserName* et le mot de passe que vous avez obtenus lors de la génération des informations d'identification spécifiques au service et qu'ils *ServicePassword* correspondent à ceux que vous avez obtenus en suivant les étapes ci-dessous. [Créez des informations d'identification spécifiques au service pour un accès programmatique à Amazon Keyspaces](#)

Note

Pour obtenir la liste des points de terminaison disponibles, reportez-vous à la section [the section called “Points de terminaison de service”](#).

4. Exécutez l'application.

```
perl cqlapp.pl
```

Configurer l'accès entre comptes à Amazon Keyspaces avec des points de terminaison VPC

Vous pouvez créer et utiliser séparément Comptes AWS pour isoler les ressources et pour les utiliser dans différents environnements, par exemple le développement et la production. Cette rubrique vous explique comment accéder à plusieurs comptes pour Amazon Keyspaces à l'aide de points

de terminaison VPC d'interface dans un. Amazon Virtual Private Cloud Pour plus d'informations sur la configuration de l'accès entre comptes IAM, voir [Exemple de scénario utilisant des comptes de développement et de production distincts](#) dans le guide de l'utilisateur IAM.

Pour plus d'informations sur Amazon Keyspaces et les points de terminaison VPC privés, consultez. [the section called “Utilisation des points de terminaison de VPC d'interface ”](#)

Rubriques

- [Configurer l'accès entre comptes à Amazon Keyspaces à l'aide de points de terminaison VPC dans un VPC partagé](#)
- [Configuration de l'accès entre comptes à Amazon Keyspaces sans VPC partagé](#)

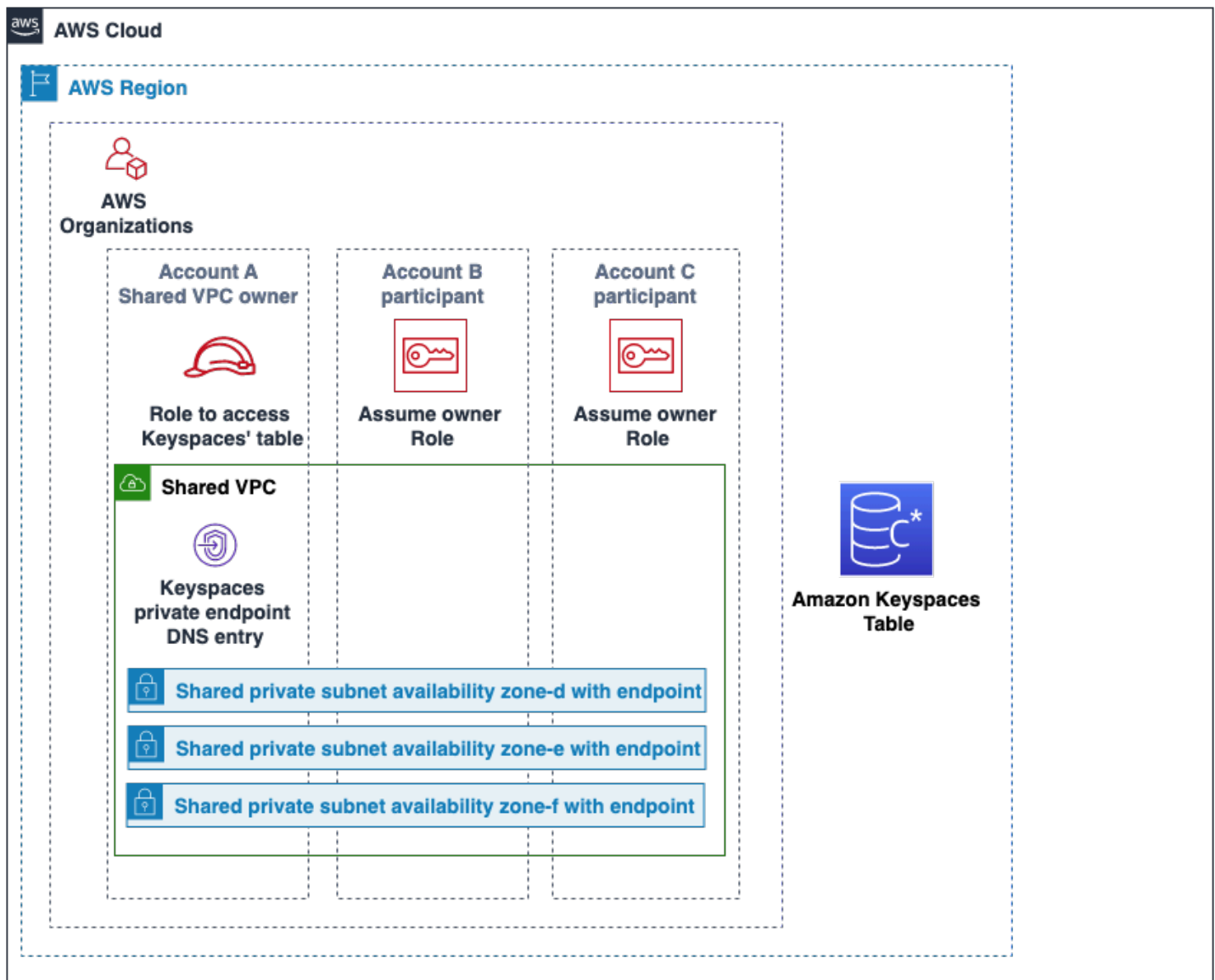
Configurer l'accès entre comptes à Amazon Keyspaces à l'aide de points de terminaison VPC dans un VPC partagé

Vous pouvez créer des ressources différentes Comptes AWS pour séparer les applications. Par exemple, vous pouvez créer un compte pour vos tables Amazon Keyspaces, un compte différent pour les applications dans un environnement de développement et un autre compte pour les applications dans un environnement de production. Cette rubrique décrit les étapes de configuration requises pour configurer l'accès entre comptes pour Amazon Keyspaces à l'aide de points de terminaison VPC d'interface dans un VPC partagé.

Pour connaître les étapes détaillées de configuration d'un point de terminaison VPC pour Amazon Keyspaces, consultez. [the section called “Étape 3 : créer un point de terminaison VPC pour Amazon Keyspaces”](#)

Dans cet exemple, nous utilisons les trois comptes suivants dans un VPC partagé :

- Account A:111111111111— Ce compte contient l'infrastructure, notamment les points de terminaison VPC, les sous-réseaux VPC et les tables Amazon Keyspaces.
- Account B:222222222222— Ce compte contient une application dans un environnement de développement qui doit se connecter à la table Amazon Keyspaces dans. Account A:111111111111
- Account C:333333333333— Ce compte contient une application dans un environnement de production qui doit se connecter à la table Amazon Keyspaces dans. Account A:111111111111



Account A:111111111111 est le compte qui contient les ressources (une table Amazon Keyspaces) auxquelles Account B:22222222222 et Account C:33333333333 dont vous avez besoin pour accéder, tout comme le compte de confiance. Account A:11111111111 Account B:22222222222 et Account C:33333333333 sont les comptes dont les principaux ont besoin d'accéder aux ressources (une table Amazon Keyspaces) Account A:11111111111, de même Account B:22222222222 que Account C:33333333333 les comptes de confiance. Le compte de confiance accorde les autorisations aux comptes approuvés en partageant un rôle IAM. La procédure suivante décrit les étapes de configuration requises dans Account A:11111111111.

Configuration pour **Compte A:111111111111**

1. AWS Resource Access Manager À utiliser pour créer un partage de ressources pour le sous-réseau et partager le sous-réseau privé avec Account B:222222222222 et Account C:333333333333

Account B:222222222222 et Account C:333333333333 peuvent désormais voir et créer des ressources dans le sous-réseau qui a été partagé avec eux.

2. Créez un point de terminaison VPC privé Amazon Keyspaces alimenté par AWS PrivateLink. Cela crée plusieurs points de terminaison sur des sous-réseaux partagés et des entrées DNS pour le point de terminaison du service Amazon Keyspaces.
3. Créez un espace de touches et un tableau Amazon Keyspaces.
4. Créez un rôle IAM doté d'un accès complet à la table Amazon Keyspaces, d'un accès en lecture aux tables du système Amazon Keyspaces et capable de décrire les ressources Amazon EC2 VPC, comme indiqué dans l'exemple de politique suivant. Account A:111111111111

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CrossAccountAccess",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcEndpoints",
        "cassandra:*"
      ],
      "Resource": "*"
    }
  ]
}
```

5. Configurez une politique de confiance pour le rôle IAM Account A:111111111111 afin que Account B:222222222222 et Account C:333333333333 puissiez assumer le rôle en tant que comptes approuvés. Voici un exemple :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
```



```

    "Effect": "Allow",
    "Principal": {
      "AWS": [
        "arn:aws:iam::222222222222:role/Cross-Account-Role-B",
        "arn:aws:iam::333333333333:role/Cross-Account-Role-C"
      ]
    },
    "Action": "sts:AssumeRole",
    "Condition": {}
  }
]
}

```

Pour plus d'informations sur les politiques IAM entre comptes, consultez les politiques du [Cross-account Guide](#) de l'utilisateur IAM.

Configuration dans **Compte B:222222222222** and **Compte C:333333333333**

1. Dans Account B:222222222222 et Account C:333333333333, créez de nouveaux rôles et associez la politique suivante qui permet au directeur d'assumer le rôle partagé créé dans Account A:111111111111.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

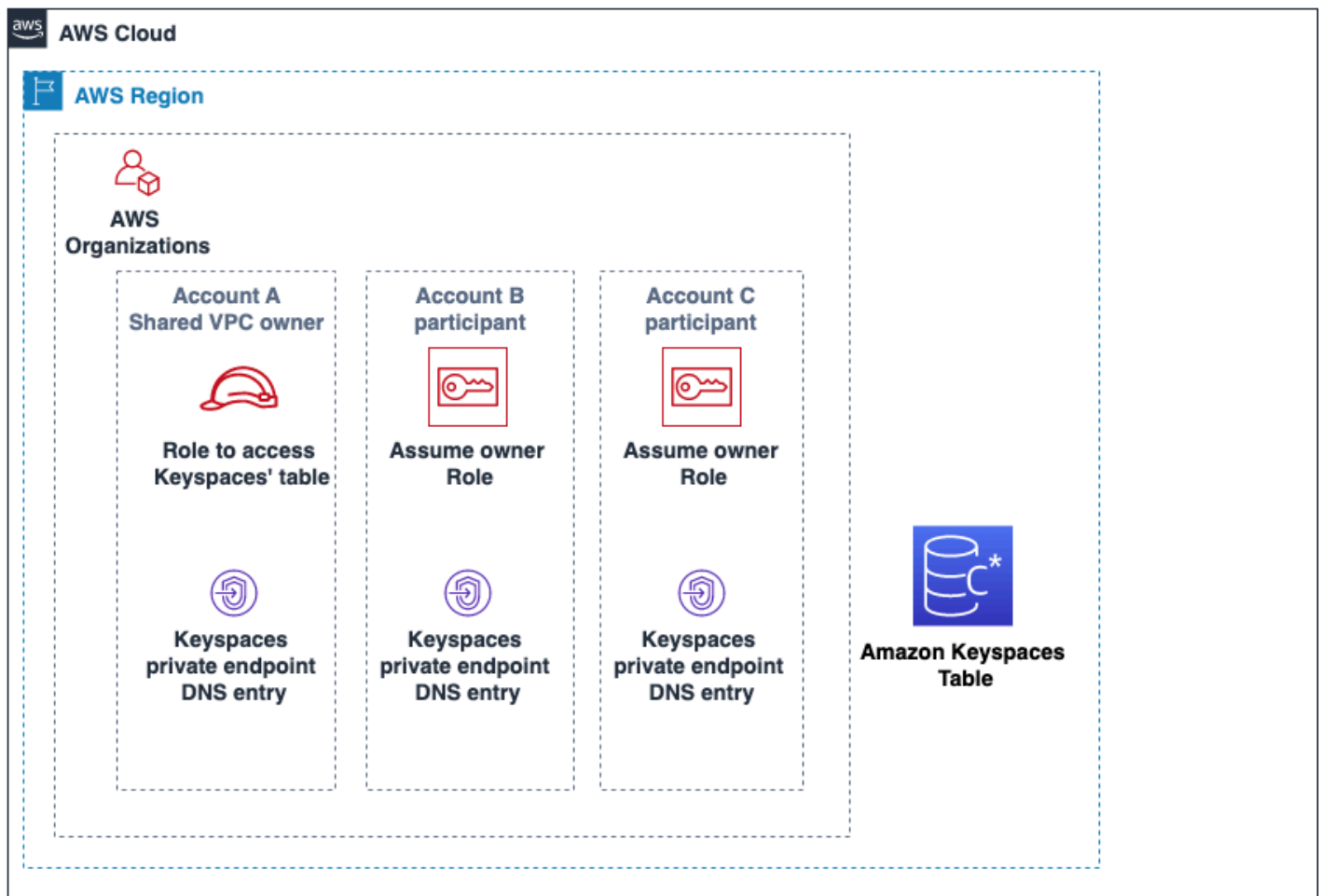
Permettre au principal d'assumer le rôle partagé est implémenté à l'aide de l'AssumeRoleAPI du AWS Security Token Service (AWS STS). Pour plus d'informations, consultez la section [Fournir un accès à un utilisateur IAM dans un autre utilisateur Compte AWS dont vous êtes le propriétaire](#) dans le Guide de l'utilisateur IAM.

2. Dans Account B:222222222222 et Account C:333333333333, vous pouvez créer des applications qui utilisent le plug-in d'authentification SIGV4, qui permet à une application d'assumer le rôle partagé pour se connecter à la table Amazon Keyspaces située via le point de terminaison VPC Account A:111111111111 dans le VPC partagé. Pour plus d'informations sur le plug-in d'authentification SIGV4, consultez [the section called “Création d'identifiants d'accès programmatiques”](#). Pour plus d'informations sur la configuration d'une application afin qu'elle assume un rôle dans un autre AWS compte, consultez la section [Authentification et accès](#) dans le Guide de référence AWS des SDK et des outils.

Configuration de l'accès entre comptes à Amazon Keyspaces sans VPC partagé

Si la table Amazon Keyspaces et le point de terminaison VPC privé appartiennent à des comptes différents mais ne partagent pas de VPC, les applications peuvent toujours se connecter entre comptes à l'aide de points de terminaison VPC. Parce que les comptes ne partagent pas les points de terminaison VPC, Account A:111111111111, Account B:222222222222, et Account C:333333333333 nécessitent leurs propres points de terminaison VPC. Pour le pilote du client Cassandra, Amazon Keyspaces apparaît comme un nœud unique au lieu d'un cluster à nœuds multiples. Lors de la connexion, le pilote client atteint le serveur DNS qui renvoie l'un des points de terminaison disponibles dans le VPC du compte.

Vous pouvez également accéder aux tables Amazon Keyspaces sur différents comptes sans point de terminaison VPC partagé en utilisant les points de terminaison publics ou en déployant un point de terminaison VPC privé dans chaque compte. Lorsque vous n'utilisez pas de VPC partagé, chaque compte a besoin de son propre point de terminaison VPC. Dans cet exemple Account A:111111111111, Account B:222222222222, et Account C:333333333333 ont besoin de leurs propres points de terminaison VPC pour accéder à la table dans Account A:111111111111. Lorsque vous utilisez des points de terminaison VPC dans cette configuration, Amazon Keyspaces apparaît comme un cluster à nœud unique pour le pilote client Cassandra au lieu d'un cluster à nœuds multiples. Lors de la connexion, le pilote client atteint le serveur DNS qui renvoie l'un des points de terminaison disponibles dans le VPC du compte. Mais le pilote client n'est pas en mesure d'accéder à la `system.peers` table pour découvrir des points de terminaison supplémentaires. Comme il y a moins d'hôtes disponibles, le pilote établit moins de connexions. Pour régler cela, multipliez par trois le paramètre du pool de connexions du pilote.



Account A:111111111111 est le compte qui contient les ressources (une table Amazon Keyspaces) auxquelles Account B:222222222222 et Account C:333333333333 dont vous avez besoin pour accéder, tout comme le compte de confiance. Account A:111111111111 Account B:222222222222 et Account C:333333333333 sont les comptes dont les principaux ont besoin d'accéder aux ressources (une table Amazon Keyspaces) Account A:111111111111, de même Account B:222222222222 que Account C:333333333333 les comptes de confiance. Le compte de confiance accorde les autorisations aux comptes approuvés en partageant un rôle IAM. La procédure suivante décrit les étapes de configuration requises dans Account A:111111111111.

Configuration pour **Compte A:111111111111**

1. Créez un espace de saisie Amazon Keyspaces et ajoutez-le. Account A:111111111111
2. Créez un rôle IAM offrant un accès complet à la table Amazon Keyspaces et un accès en lecture aux tables du système Amazon Keyspaces. Account A:111111111111

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cassandra:Select",
        "cassandra:Modify"
      ],
      "Resource": [
        "arn:aws:cassandra:us-east-1:111111111111:/keyspace/mykeyspace/table/mytable",
        "arn:aws:cassandra:us-east-1:111111111111:/keyspace/system*"
      ]
    }
  ]
}
```

3. Configurez une politique de confiance pour le rôle IAM dans Account A:111111111111 afin que les principaux Account C:333333333333 puissent assumer le rôle en Account B:222222222222 tant que comptes approuvés. Voici un exemple :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::222222222222:role/Cross-Account-Role-B",
          "arn:aws:iam::333333333333:role/Cross-Account-Role-C"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

Pour plus d'informations sur les politiques IAM entre comptes, consultez les politiques du [Cross-account Guide](#) de l'utilisateur IAM.

4. Configurez le point de terminaison VPC dans le point de terminaison Account A:111111111111 et attachez-lui des autorisations qui permettent aux rôles d'assumer le rôle d'utilisateur du point Account C:333333333333 de terminaison VPC Account B:222222222222 et d'assumer le rôle dans son Account A utilisation. Ces autorisations sont valides pour le point de terminaison VPC auquel elles sont attachées. Pour plus d'informations sur les politiques relatives aux points de terminaison VPC, consultez [the section called "Contrôle de l'accès aux points de terminaison VPC de l'interface pour Amazon Keyspaces"](#)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessfromSpecificIAMroles",
      "Effect": "Allow",
      "Action": "cassandra:*",
      "Resource": "*",
      "Principal": "*",
      "Condition": {
        "ArnEquals": {
          "aws:PrincipalArn": [
            "arn:aws:iam::222222222222:role/Cross-Account-Role-B",
            "arn:aws:iam::333333333333:role/Cross-Account-Role-C"
          ]
        }
      }
    }
  ]
}
```

Configuration dans **Compte B:222222222222** and **Compte C:333333333333**

1. Dans Account B:222222222222 et Account C:333333333333, créez de nouveaux rôles et associez la politique suivante qui permet au directeur d'assumer le rôle partagé créé dans Account A:111111111111.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
```

```
    "Resource": "arn:aws:iam::111111111111:role/keyspaces_access"  
  }  
}
```

Permettre au principal d'assumer le rôle partagé est implémenté à l'aide de l'AssumeRoleAPI du AWS Security Token Service (AWS STS). Pour plus d'informations, consultez la section [Fournir un accès à un utilisateur IAM dans un autre utilisateur Compte AWS dont vous êtes le propriétaire](#) dans le Guide de l'utilisateur IAM.

2. Dans Account B:222222222222 et Account C:333333333333, vous pouvez créer des applications qui utilisent le plugin d'authentification SIGV4, qui permet à une application d'assumer le rôle partagé pour se connecter à la table Amazon Keyspaces située dans Account A:111111111111. Pour plus d'informations sur le plug-in d'authentification SIGV4, consultez [the section called “Création d'identifiants d'accès programmatiques”](#). Pour plus d'informations sur la configuration d'une application afin qu'elle assume un rôle dans un autre AWS compte, consultez la section [Authentification et accès](#) dans le Guide de référence AWS des SDK et des outils.

Démarrez avec Amazon Keyspaces (pour Apache Cassandra)

Si vous utilisez Apache Cassandra et Amazon Keyspaces pour la première fois, ce didacticiel vous explique comment installer les programmes et outils nécessaires pour utiliser Amazon Keyspaces avec succès. Vous allez apprendre à créer un espace clé et une table en utilisant le langage de requête Cassandra (CQL) Console de gestion AWS, le ou le AWS Command Line Interface (CLI). Vous utilisez ensuite le Cassandra Query Language (CQL) pour effectuer des opérations de création, de lecture, de mise à jour et de suppression (CRUD) sur les données de votre table Amazon Keyspaces.

Ce didacticiel couvre les étapes suivantes.

- **Conditions préalables** : avant de commencer le didacticiel, suivez les instructions de AWS configuration pour vous inscrire AWS et créer un utilisateur IAM ayant accès à Amazon Keyspaces. Ensuite, vous configurez le `cqhsh-expansion` et AWS CloudShell. Vous pouvez également utiliser le AWS CLI pour créer des ressources dans Amazon Keyspaces.
- **Étape 1 : Création d'un espace clé et d'un tableau** — Dans cette section, vous allez créer un espace clé nommé « `catalog` » et un tableau nommé « `book_awards` » à l'intérieur de celui-ci. Vous allez spécifier les colonnes, les types de données, la clé de partition et la colonne de clustering de la table à l'aide du Console de gestion AWS, du CQL ou du AWS CLI
- **Étape 2 : Effectuez des opérations CRUD** — Ici, vous allez utiliser l'entrée `cqlsh-expansion` CloudShell pour insérer, lire, mettre à jour et supprimer des données dans le tableau « `book_awards` ». Vous apprendrez à utiliser différentes instructions CQL telles que `SELECT`, `INSERT`, `UPDATE` et `DELETE`, et vous vous entraînerez à filtrer et à modifier les données.
- **Étape 3 : Nettoyer les ressources** — Pour éviter des frais pour les ressources non utilisées, cette section vous explique comment supprimer la table « `book_awards` » et l'espace de touches « `catalog` » à l'aide de la console, de la CQL ou du AWS CLI

Pour des didacticiels sur la connexion par programmation à Amazon Keyspaces à l'aide de différents pilotes clients Apache Cassandra, consultez [the section called “Utilisation d'un pilote client Cassandra”](#) Pour des exemples de code utilisant différents AWS kits de développement logiciel, consultez la section Exemples de [code pour Amazon Keyspaces utilisant AWS des kits de développement logiciel \(SDK\)](#).

Rubriques

- [Prérequis et considérations relatifs au didacticiel](#)
- [Création d'un espace clé dans Amazon Keyspaces](#)
- [Vérifier l'état de création de l'espace clé dans Amazon Keyspaces](#)
- [Création d'un tableau dans Amazon Keyspaces](#)
- [Vérifier l'état de création d'une table dans Amazon Keyspaces](#)
- [Créez, lisez, mettez à jour et supprimez des données \(CRUD\) à l'aide de CQL dans Amazon Keyspaces](#)
- [Supprimer un tableau dans Amazon Keyspaces](#)
- [Supprimer un espace clé dans Amazon Keyspaces](#)

Prérequis et considérations relatifs au didacticiel

Avant de commencer à utiliser Amazon Keyspaces, suivez les instructions de AWS configuration figurant dans [Accès à Amazon Keyspaces \(pour Apache Cassandra\)](#). Ces étapes incluent l'inscription AWS et la création d'un utilisateur Gestion des identités et des accès AWS (IAM) ayant accès à Amazon Keyspaces.

Pour terminer toutes les étapes du didacticiel, vous devez installer `cqlsh`. Vous pouvez suivre les instructions de configuration à l'adresse [Utilisation cqlsh pour se connecter à Amazon Keyspaces](#).

Pour accéder à Amazon Keyspaces à l'aide du `cqlsh` ou du AWS CLI, nous vous recommandons d'utiliser AWS CloudShell. CloudShell est un shell pré-authentifié basé sur un navigateur que vous pouvez lancer directement depuis la console de gestion AWS. Vous pouvez exécuter des commandes AWS Command Line Interface (AWS CLI) sur Amazon Keyspaces à l'aide de votre shell préféré (Bash, PowerShell ou Z shell). Pour l'utiliser `cqlsh`, vous devez installer l'expansion `cqlsh`. Pour les instructions d'installation de l'expansion `cqlsh`, voir [la section appelée "Utilisation de l'expansion cqlsh"](#). Pour plus d'informations sur CloudShell, voir [la section appelée "En utilisant AWS CloudShell"](#).

Pour utiliser le AWS CLI pour créer, afficher et supprimer des ressources dans Amazon Keyspaces, suivez les instructions de configuration disponibles à [la section appelée "Téléchargement et configuration du AWS CLI"](#) l'adresse.

Après avoir effectué les étapes préalables, passez à [Création d'un espace clé dans Amazon Keyspaces](#).

Création d'un espace clé dans Amazon Keyspaces

Dans cette section, vous allez créer un espace clé à l'aide de la console `cqlsh`, ou du AWS CLI.

Note

Avant de commencer, assurez-vous d'avoir configuré tous les prérequis [du](#) didacticiel.

Un keyspace regroupe les tables associées qui sont pertinentes pour une ou plusieurs applications. Un keyspace contient une ou plusieurs tables et définit la stratégie de réplication pour toutes les tables qu'il contient. Pour de plus amples informations sur les keyspaces, veuillez consulter les rubriques suivantes :

- Instructions du langage de définition des données (DDL) dans la référence du langage CQL : [Keyspaces](#)
- [Quotas pour Amazon Keyspaces \(pour Apache Cassandra\)](#)

Dans ce didacticiel, nous créons un espace clé à région unique, et la stratégie de réplication de l'espace clé est la suivante : `SingleRegionStrategy`. À l'aide d'`AmazonKeyspacesSingleRegionStrategy`, il réplique les données de trois zones de [disponibilité](#) en une seule. Région AWS Pour savoir comment créer des espaces clés multi-régions, consultez [the section called “Création d'un espace de saisie multirégional”](#)

Utilisation de la console

Pour créer un keyspace à l'aide de la console

1. Connectez-vous au Console de gestion AWS, puis ouvrez la console Amazon Keyspaces à <https://console.aws.amazon.com/keyspaces/home> l'adresse.
2. Dans le panneau de navigation, choisissez Keyspaces.
3. Choisissez Create keyspace.
4. Dans la zone Nom du keyspace entrez **catalog** comme nom de votre keyspace.

Contraintes liées au nom :

- Le nom ne peut pas être vide.

- Caractères autorisés : caractères alphanumériques et soulignement (_).
 - La longueur maximale est de 48 caractères.
5. Dans la Région AWS section ci-dessous, confirmez que la Single-Region réplication est la stratégie de réplication pour l'espace clé.
 6. Pour créer le keyspace, choisissez Create keyspace (Créer un keyspace).
 7. Vérifiez que le keyspace catalog a été créé en procédant comme suit :
 - a. Dans le panneau de navigation, choisissez Keyspaces.
 - b. Localisez votre keyspace catalog dans la liste des keyspaces.

Utilisation de CQL

La procédure suivante crée un keyspace à l'aide de CQL.

Pour créer un keyspace avec CQL

1. Ouvrez Amazon Keyspaces AWS CloudShell et connectez-vous à celui-ci à l'aide de la commande suivante. Assurez-vous de mettre à jour *us-east-1* avec votre propre région.

```
cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 --ssl
```

Le résultat de cette commande devrait ressembler à ceci.

```
Connected to Amazon Keyspaces at cassandra.us-east-1.amazonaws.com:9142
[cqlsh 6.1.0 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh current consistency level is ONE.
```

2. Créez votre keyspace à l'aide de la commande CQL suivante.

```
CREATE KEYSPACE catalog WITH REPLICATION = {'class': 'SingleRegionStrategy'};
```

SingleRegionStrategy utilise un facteur de réplication de trois et réplique les données dans trois zones de AWS disponibilité de sa région.

Note

Amazon Keyspaces met par défaut toutes les entrées en minuscules, sauf si vous les placez entre guillemets.

3. Vérifiez que votre keyspace a été créé.

```
SELECT * from system_schema.keyspaces;
```

Le résultat de cette commande doit être similaire à celui-ci.

```
cqlsh> SELECT * from system_schema.keyspaces;
```

keyspace_name	durable_writes	replication
-----+-----		
system_schema	True	{'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
system_schema_mcs	True	{'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
system	True	{'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
system_multiregion_info	True	{'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
catalog	True	{'class': 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}

(5 rows)

Utilisation de AWS CLI

La procédure suivante crée un espace clé à l'aide du AWS CLI.

Pour créer un espace clé à l'aide de AWS CLI

1. Pour vérifier que votre environnement est configuré, vous pouvez exécuter la commande suivante dans CloudShell.

```
aws keyspaces help
```

2. Créez votre espace de touches à l'aide de l' AWS CLI instruction suivante.

```
aws keyspaces create-keyspace --keyspace-name 'catalog'
```

3. Vérifiez que votre espace clé a été créé à l'aide de l'instruction suivante AWS CLI

```
aws keyspaces get-keyspace --keyspace-name 'catalog'
```

La sortie de cette commande doit ressembler à celle de cet exemple.

```
{
  "keyspaceName": "catalog",
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/catalog/",
  "replicationStrategy": "SINGLE_REGION"
}
```

Vérifier l'état de création de l'espace clé dans Amazon Keyspaces

Amazon Keyspaces effectue des opérations en langage de définition des données (DDL), telles que la création et la suppression d'espaces clés, de manière asynchrone.

Vous pouvez surveiller l'état de création de nouveaux espaces clés dans le Console de gestion AWS, qui indique quand un espace clé est en attente ou actif. Vous pouvez également surveiller l'état de création d'un nouvel espace clé par programmation à l'aide de l'espace clé. `system_schema_mcs`
Un espace clé devient visible dans le `system_schema_mcs keyspaces` tableau lorsqu'il est prêt à être utilisé.

Le modèle de conception recommandé pour vérifier quand un nouvel espace clé est prêt à être utilisé consiste à interroger la `system_schema_mcs keyspaces` table Amazon Keyspaces (`system_schema_mcs.*`). Pour une liste des instructions DDL pour les espaces clés, consultez la [the section called “Keyspaces”](#) section de la référence du langage CQL.

La requête suivante indique si un espace clé a été créé avec succès.

```
SELECT * FROM system_schema_mcs.keyspaces WHERE keyspace_name = 'mykeyspace';
```

Pour un espace clé qui a été créé avec succès, le résultat de la requête ressemble à ce qui suit.

```

keyspace_name | durable_writes | replication
-----+-----+-----
mykeyspace | true | {...} 1 item

```

Création d'un tableau dans Amazon Keyspaces

Dans cette section, vous allez créer un tableau à l'aide de la console `cqlsh`, ou du AWS CLI.

Une table est l'endroit où vos données sont organisées et stockées. La clé primaire de votre table détermine la façon dont les données sont partitionnées dans votre table. La clé primaire est composée d'une clé de partition requise et d'une ou de plusieurs colonnes de clustering facultatives. Les valeurs combinées qui composent la clé primaire doivent être uniques pour toutes les données de la table. Pour de plus amples informations sur les balises, veuillez consulter les rubriques suivantes :

- Conception de la clé de partition : [the section called “Création de clés de partition”](#)
- Utilisation de tableaux : [the section called “Vérifier l'état de création de la table”](#)
- Les instructions DDL dans le langage CQL font référence : [Tables](#)
- Gestion des ressources des tables : [Gestion des ressources sans serveur](#)
- Utilisation des ressources de la table de surveillance : [the section called “Surveillance avec CloudWatch”](#)
- [Quotas pour Amazon Keyspaces \(pour Apache Cassandra\)](#)

Lorsque vous créez une table, vous spécifiez les éléments suivants :

- Nom de la table.
- Nom et type de données de chaque colonne de la table.
- Clé primaire de la table.
 - Clé de partition — Obligatoire
 - Colonnes de clustering — Facultatif

Suivez la procédure suivante pour créer une table contenant les colonnes, les types de données, les clés de partition et les colonnes de clustering spécifiés.

Utilisation de la console

La procédure suivante crée la table `book_awards` avec ces colonnes et types de données.

<code>year</code>	<code>int</code>
<code>award</code>	<code>text</code>
<code>rank</code>	<code>int</code>
<code>category</code>	<code>text</code>
<code>book_title</code>	<code>text</code>
<code>author</code>	<code>text</code>
<code>publisher</code>	<code>text</code>

Pour créer une table à l'aide de la console

1. Connectez-vous au Console de gestion AWS, puis ouvrez la console Amazon Keyspaces à <https://console.aws.amazon.com/keyspaces/home> l'adresse.
2. Dans le panneau de navigation, choisissez Keyspaces.
3. Choisissez `catalog` comme keyspace dans lequel vous souhaitez créer la table.
4. Choisissez Créer un tableau.
5. Dans la zone Nom de la table entrez **`book_awards`** comme nom pour votre table.

Contraintes liées au nom :

- Le nom ne peut pas être vide.
 - Caractères autorisés : caractères alphanumériques et soulignement (`_`).
 - La longueur maximale est de 48 caractères.
6. Dans la section Colonnes répétez les étapes suivantes pour chaque colonne à ajouter à cette table.

Ajoutez les colonnes et types de données suivants.

<code>year</code>	<code>int</code>
<code>award</code>	<code>text</code>
<code>rank</code>	<code>int</code>
<code>category</code>	<code>text</code>
<code>book_title</code>	<code>text</code>
<code>author</code>	<code>text</code>
<code>publisher</code>	<code>text</code>

- a. Nom — Entrez un nom pour la colonne.

Contraintes liées au nom :

- Le nom ne peut pas être vide.
- Caractères autorisés : caractères alphanumériques et soulignement (_).
- La longueur maximale est de 48 caractères.

- b. Type : dans la liste des types de données, choisissez le type de données pour cette colonne.
 - c. Pour ajouter une autre colonne, choisissez Ajouter une colonne.
7. Choisissez `award` et `year` comme clés de partition sous Clé de partition. Une clé de partition est requise pour chaque table. Une clé de partition peut être constituée d'une ou plusieurs colonnes.
 8. Ajoutez `category` et `rank` en tant que colonnes de clustering. Les colonnes de clustering sont facultatives et déterminent l'ordre de tri dans chaque partition.
 - a. Pour ajouter une colonne de clustering, choisissez Ajouter une colonne de clustering.
 - b. Dans la liste des colonnes, choisissez la catégorie. Dans la liste Ordre choisissez ASC pour trier par ordre croissant les valeurs de cette colonne. (Choisissez DESC pour l'ordre décroissant.)
 - c. Sélectionnez ensuite Ajouter une colonne de clustering et choisissez le classement.
 9. Dans la section Paramètres du tableau, choisissez Paramètres par défaut.
 10. Choisissez Créer un tableau.
 11. Vérifiez que votre table a été créée.
 - a. Dans le volet de navigation, choisissez Tables.
 - b. Confirmez que votre table figure dans la liste des tables.
 - c. Choisissez le nom de votre table.
 - d. Vérifiez que toutes vos colonnes et tous vos types de données sont corrects.

 Note

Les colonnes peuvent ne pas être répertoriées dans le même ordre que celui où vous les avez ajoutées à la table.

Utilisation de CQL

Cette procédure crée un tableau avec les colonnes et les types de données suivants à l'aide de CQL. Les award colonnes year et sont des clés de partition avec category et rank en tant que colonnes de clustering. Ensemble, elles constituent la clé primaire de la table.

year	int
award	text
rank	int
category	text
book_title	text
author	text
publisher	text

Pour créer une table avec CQL

1. Ouvrez Amazon Keyspaces AWS CloudShell et connectez-vous à celui-ci à l'aide de la commande suivante. Assurez-vous de mettre à jour *us-east-1* avec votre propre région.

```
cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 --ssl
```

Le résultat de cette commande devrait ressembler à ceci.

```
Connected to Amazon Keyspaces at cassandra.us-east-1.amazonaws.com:9142
[cqlsh 6.1.0 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh current consistency level is ONE.
```

2. À l'invite d'espace de touche (cqlsh:keyspace_name>), créez votre table en entrant le code suivant dans votre fenêtre de commande.

```
CREATE TABLE catalog.book_awards (  
  year int,  
  award text,  
  rank int,  
  category text,  
  book_title text,  
  author text,  
  publisher text,  
  PRIMARY KEY ((year, award), category, rank)
```



```
);
```

Note

ASC est l'ordre de mise en cluster par défaut. Vous pouvez également spécifier DESC pour l'ordre décroissant.

Notez que les colonnes `year` et `award` sont des clés de partition. Ensuite, `category` et `rank` les colonnes de regroupement sont classées par ordre croissant ()ASC. Ensemble, ces colonnes constituent la clé primaire de la table.

3. Vérifiez que votre table a été créée.

```
SELECT * from system_schema.tables WHERE keyspace_name='catalog.book_awards' ;
```

La sortie doit ressembler à celle-ci.

```
keyspace_name | table_name | bloom_filter_fp_chance | caching | cdc | comment
| compaction | compression | crc_check_chance | dclocal_read_repair_chance
| default_time_to_live | extensions | flags | gc_grace_seconds | id |
max_index_interval | memtable_flush_period_in_ms | min_index_interval |
read_repair_chance | speculative_retry
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

(0 rows)
```

4. Vérifiez la structure de votre table.

```
SELECT * FROM system_schema.columns WHERE keyspace_name = 'catalog' AND table_name
= 'book_awards';
```

Le résultat de cette instruction doit ressembler à cet exemple.

```
keyspace_name | table_name | column_name | clustering_order | column_name_bytes
| kind      | position | type
```

```

-----+-----+-----+-----
+-----+-----+-----+-----
      catalog | book_awards |      year |      none |
0x79656172 | partition_key |      0 | int
      catalog | book_awards |      award |      none |
0x6177617264 | partition_key |      1 | text
      catalog | book_awards |      category |      asc |
0x63617465676f7279 | clustering |      0 | text
      catalog | book_awards |      rank |      asc |
0x72616e6b | clustering |      1 | int
      catalog | book_awards |      author |      none |
0x617574686672 | regular |      -1 | text
      catalog | book_awards | book_title |      none |
0x626f66b5f7469746c65 | regular |      -1 | text
      catalog | book_awards | publisher |      none |
0x7075626c6973686572 | regular |      -1 | text

(7 rows)

```

Confirmez que toutes les colonnes et tous les types de données sont tels que vous l'escomptiez. L'ordre des colonnes peut être différent de celui de l'instruction CREATE.

Utilisation de AWS CLI

Cette procédure crée un tableau contenant les colonnes et les types de données suivants à l'aide du AWS CLI. Les award colonnes year et constituent la clé de partition avec category et en rank tant que colonnes de clustering.

```

year          int
award         text
rank          int
category      text
book_title    text
author        text
publisher     text

```

Pour créer un tableau à l'aide du AWS CLI

La commande suivante crée une table nommée `book_awards`. La clé de partition de la table se compose des colonnes `year` `award` et la clé de clustering comprend les colonnes. Les deux colonnes `category` de clustering utilisent l'ordre de tri croissant. `rank` (Pour faciliter la lisibilité,

la commande `schema-definition` de création de table de cette section est divisée en lignes distinctes.)

1. Vous pouvez créer le tableau à l'aide de l'instruction suivante.

```
aws keyspaces create-table --keyspace-name 'catalog' \  
                           --table-name 'book_awards' \  
                           --schema-definition 'allColumns=[{name=year,type=int},  
{name=award,type=text},{name=rank,type=int},  
                           {name=category,type=text}, {name=author,type=text},  
{name=book_title,type=text},{name=publisher,type=text}],  
                           partitionKeys=[{name=year},  
{name=award}],clusteringKeys=[{name=category,orderBy=ASC},{name=rank,orderBy=ASC}]'
```

Cette commande produit le résultat suivant.

```
{  
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/catalog/  
table/book_awards"  
}
```

2. Pour confirmer les métadonnées et les propriétés de la table, vous pouvez utiliser la commande suivante.

```
aws keyspaces get-table --keyspace-name 'catalog' --table-name 'book_awards'
```

Cette commande renvoie la sortie suivante.

```
{  
  "keyspaceName": "catalog",  
  "tableName": "book_awards",  
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/catalog/  
table/book_awards",  
  "creationTimestamp": "2024-07-11T15:12:55.571000+00:00",  
  "status": "ACTIVE",  
  "schemaDefinition": {  
    "allColumns": [  
      {  
        "name": "year",  
        "type": "int"  
      },  
      {  
        "name": "award",  
        "type": "text"  
      },  
      {  
        "name": "rank",  
        "type": "int"  
      },  
      {  
        "name": "category",  
        "type": "text"  
      },  
      {  
        "name": "author",  
        "type": "text"  
      },  
      {  
        "name": "book_title",  
        "type": "text"  
      },  
      {  
        "name": "publisher",  
        "type": "text"  
      }  
    ],  
    "partitionKeys": [  
      {  
        "name": "year",  
        "order": 1  
      },  
      {  
        "name": "award",  
        "order": 2  
      }  
    ],  
    "clusteringKeys": [  
      {  
        "name": "category",  
        "order": 1  
      },  
      {  
        "name": "rank",  
        "order": 2  
      }  
    ]  
  }  
}
```

```
{
  "name": "award",
  "type": "text"
},
{
  "name": "category",
  "type": "text"
},
{
  "name": "rank",
  "type": "int"
},
{
  "name": "author",
  "type": "text"
},
{
  "name": "book_title",
  "type": "text"
},
{
  "name": "publisher",
  "type": "text"
}
],
"partitionKeys": [
  {
    "name": "year"
  },
  {
    "name": "award"
  }
],
"clusteringKeys": [
  {
    "name": "category",
    "orderBy": "ASC"
  },
  {
    "name": "rank",
    "orderBy": "ASC"
  }
],
"staticColumns": []
```

```
{,
  "capacitySpecification": {
    "throughputMode": "PAY_PER_REQUEST",
    "lastUpdateToPayPerRequestTimestamp": "2024-07-11T15:12:55.571000+00:00"
  },
  "encryptionSpecification": {
    "type": "AWS_OWNED_KMS_KEY"
  },
  "pointInTimeRecovery": {
    "status": "DISABLED"
  },
  "defaultTimeToLive": 0,
  "comment": {
    "message": ""
  },
  "replicaSpecifications": []
}
```

Pour effectuer des opérations CRUD (création, lecture, mise à jour et suppression) sur les données de votre table, passez à [the section called “Opérations CRUD”](#).

Vérifier l'état de création d'une table dans Amazon Keyspaces

Amazon Keyspaces effectue des opérations en langage de définition des données (DDL), telles que la création et la suppression de tables, de manière asynchrone. Vous pouvez suivre l'état de création des nouvelles tables dans le Console de gestion AWS, qui indique quand une table est en attente ou active. Vous pouvez également surveiller l'état de création d'une nouvelle table par programmation à l'aide de la table de schéma du système.

Un tableau indique qu'il est actif dans le schéma du système lorsqu'il est prêt à être utilisé. Le modèle de conception recommandé pour vérifier quand une nouvelle table est prête à être utilisée consiste à interroger les tables de schéma du système Amazon Keyspaces (`system_schema_mcs.*`). Pour obtenir la liste des instructions DDL pour les tables, consultez la [the section called “Tables”](#) section de la référence du langage CQL.

La requête suivante indique l'état d'une table.

```
SELECT keyspace_name, table_name, status FROM system_schema_mcs.tables WHERE
  keyspace_name = 'mykeyspace' AND table_name = 'mytable';
```

Pour une table en cours de création et en attente, le résultat de la requête se présente comme suit.

```
keyspace_name | table_name | status
-----+-----+-----
mykeyspace   | mytable   | CREATING
```

Pour une table qui a été créée avec succès et qui est active, le résultat de la requête se présente comme suit.

```
keyspace_name | table_name | status
-----+-----+-----
mykeyspace   | mytable   | ACTIVE
```

Créez, lisez, mettez à jour et supprimez des données (CRUD) à l'aide de CQL dans Amazon Keyspaces

Dans cette étape du didacticiel, vous allez apprendre à insérer, lire, mettre à jour et supprimer des données dans un tableau Amazon Keyspaces à l'aide des instructions du langage de manipulation de données (DML) CQL. Dans Amazon Keyspaces, vous ne pouvez créer des instructions DML qu'en langage CQL. Dans ce didacticiel, vous allez vous entraîner à exécuter des instructions DML à l'aide du `cqlsh`-expansion signé [AWS CloudShell](#) with du Console de gestion AWS.

- **Insertion de données** — Cette section couvre l'insertion d'un ou de plusieurs enregistrements dans une table à l'aide de l'INSERT instruction. Vous allez apprendre à charger des données à partir d'un fichier CSV et à vérifier la réussite des insertions à l'aide de SELECT requêtes.
- **Lecture de données** — Ici, vous allez explorer différentes variantes de l'SELECT instruction pour récupérer les données d'un tableau. Les sujets incluent la sélection de toutes les données, la sélection de colonnes spécifiques, le filtrage des lignes en fonction des conditions à l'aide de la WHERE clause et la compréhension des conditions simples et composées.
- **Mise à jour des données** — Dans cette section, vous allez apprendre à modifier les données existantes d'un tableau à l'aide de l'UPDATE instruction. Vous vous entraînerez à mettre à jour une ou plusieurs colonnes tout en comprenant les restrictions liées à la mise à jour des colonnes de clé primaire.
- **Suppression de données** — La dernière section traite de la suppression de données d'un tableau à l'aide de l'DELETE instruction. Vous apprendrez comment supprimer des cellules spécifiques, des

lignes entières et les implications de la suppression de données par rapport à la suppression de la totalité du tableau ou de l'espace clé.

Tout au long du didacticiel, vous trouverez des exemples, des conseils et des opportunités pour vous entraîner à rédiger vos propres requêtes CQL pour différents scénarios.

Rubriques

- [Insertion et chargement de données dans un tableau Amazon Keyspaces](#)
- [Lire les données d'une table à l'aide de l'instruction CQL SELECT dans Amazon Keyspaces](#)
- [Mettre à jour les données d'une table Amazon Keyspaces à l'aide de CQL](#)
- [Supprimer les données d'une table à l'aide de l'instruction CQL DELETE](#)

Insertion et chargement de données dans un tableau Amazon Keyspaces

Pour créer des données dans votre table `book_awards`, utilisez l'instruction `INSERT` pour ajouter une seule ligne.

1. Ouvrez Amazon Keyspaces AWS CloudShell et connectez-vous à celui-ci à l'aide de la commande suivante. Assurez-vous de mettre à jour `us-east-1` avec votre propre région.

```
cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 --ssl
```

Le résultat de cette commande devrait ressembler à ceci.

```
Connected to Amazon Keyspaces at cassandra.us-east-1.amazonaws.com:9142
[cqlsh 6.1.0 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh current consistency level is ONE.
```

2. Avant de pouvoir écrire des données dans votre table Amazon Keyspaces à l'aide de `cqlsh`, vous devez définir la cohérence d'écriture pour la session `cqlsh` en cours sur `LOCAL_QUORUM`. Pour plus d'informations sur les niveaux de cohérence pris en charge, consultez [the section called "Niveaux de cohérence en écriture"](#). Notez que cette étape n'est pas requise si vous utilisez l'éditeur CQL dans le Console de gestion AWS.

```
CONSISTENCY LOCAL_QUORUM;
```

3. Pour insérer un seul enregistrement, exécutez la commande suivante dans l'éditeur CQL.

```
INSERT INTO catalog.book_awards (award, year, category, rank, author, book_title,
publisher)
VALUES ('Wolf', 2023, 'Fiction',3,'Shirley Rodriguez','Mountain', 'AnyPublisher') ;
```

4. Vérifiez que les données ont été correctement ajoutées à votre table en exécutant la commande suivante.

```
SELECT * FROM catalog.book_awards;
```

La sortie de l'instruction devrait ressembler à ceci.

```
year | award | category | rank | author          | book_title | publisher
-----+-----+-----+-----+-----+-----+-----
2023 | Wolf  | Fiction  | 3    | Shirley Rodriguez | Mountain   | AnyPublisher

(1 rows)
```

Pour insérer plusieurs enregistrements à partir d'un fichier avec `cqlsh`

1. Téléchargez l'exemple de fichier CSV (`keyspaces_sample_table.csv`) contenu dans le fichier d'archive [samplemigration.zip](#). Décompressez l'archive et notez le chemin vers `keyspaces_sample_table.csv`.

award	year	category	rank	author	book_title	publisher
Kwesi Manu Prize	2020	Fiction	1	Akua Mansa	Where did you go?	SomePublisher
Kwesi Manu Prize	2020	Fiction	2	John Stiles	Yesterday	Example Books
Kwesi Manu Prize	2020	Fiction	3	Nikki Wolf	Moving to the Chateau	AnyPublisher
Wolf	2020	Non-Fiction	1	Wang Xiulan	History of Ideas	Example Books
Wolf	2020	Non-Fiction	2	Ana Carolina Silva	Science Today	SomePublisher
Wolf	2020	Non-Fiction	3	Shirley Rodriguez	The Future of Sea Ice	AnyPublisher
Richard Roe	2020	Fiction	1	Alejandro Rosalez	Long Summer	SomePublisher
Richard Roe	2020	Fiction	2	Arnav Desai	The Key	Example Books
Richard Roe	2020	Fiction	3	Mateo Jackson	Inside the Whale	AnyPublisher

2. Ouvrez AWS CloudShell Console de gestion AWS et connectez-vous à Amazon Keyspaces à l'aide de la commande suivante. Assurez-vous de mettre à jour `us-east-1` avec votre propre région.

```
cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 --ssl
```


- À l'invite `cqlsh` (`cqlsh>`), spécifiez un keyspace.

```
USE catalog ;
```

- Définissez la cohérence d'écriture sur `LOCAL_QUORUM`. Pour plus d'informations sur les niveaux de cohérence pris en charge, consultez [the section called “Niveaux de cohérence en écriture”](#).

```
CONSISTENCY LOCAL_QUORUM;
```

- AWS CloudShell Choisissez **Actions** en haut à droite de l'écran, puis choisissez **Télécharger le fichier** pour télécharger le fichier csv téléchargé précédemment. Prenez note du chemin d'accès au fichier.
- À l'invite d'espace clavier (`cqlsh:catalog>`), exécutez l'instruction suivante.

```
COPY book_awards (award, year, category, rank, author, book_title, publisher) FROM
'/home/cloudshell-user/keyspaces_sample_table.csv' WITH header=TRUE ;
```

La sortie de l'instruction doit ressembler à ceci.

```
cqlsh:catalog> COPY book_awards (award, year, category, rank, author,
book_title, publisher) FROM '/home/cloudshell-user/
keyspaces_sample_table.csv' WITH delimiter=',' AND header=TRUE ;
cqlsh current consistency level is LOCAL_QUORUM.
Reading options from /home/cloudshell-user/.cassandra/cqlshrc:[copy]:
{'numprocesses': '16', 'maxattempts': '1000'}
Reading options from /home/cloudshell-user/.cassandra/cqlshrc:[copy-from]:
{'ingestrate': '1500', 'maxparseerrors': '1000', 'maxinserterrors': '-1',
'maxbatchsize': '10', 'minbatchsize': '1', 'chunksize': '30'}
Reading options from the command line: {'delimiter': ',', 'header': 'TRUE'}
Using 16 child processes

Starting copy of catalog.book_awards with columns [award, year, category, rank,
author, book_title, publisher].
OSError: handle is closed      0 rows/s; Avg. rate:      0 rows/s
Processed: 9 rows; Rate:      0 rows/s; Avg. rate:      0 rows/s
9 rows imported from 1 files in 0 day, 0 hour, 0 minute, and 26.706 seconds (0
skipped).
```

- Vérifiez que les données ont été correctement ajoutées à votre table en exécutant la requête suivante.

```
SELECT * FROM book_awards ;
```

Le résultat suivant doit s'afficher.

```

year | award          | category | rank | author          | book_title
-----+-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----+-----
2020 |      Wolf      | Non-Fiction |    1 | Wang Xiulan | History
of Ideas | Example Books
2020 |      Wolf      | Non-Fiction |    2 | Ana Carolina Silva |
Science Today | SomePublisher
2020 |      Wolf      | Non-Fiction |    3 | Shirley Rodriguez | The Future of
Sea Ice | AnyPublisher
2020 | Kwesi Manu Prize | Fiction |    1 | Akua Mansa | Where did
you go? | SomePublisher
2020 | Kwesi Manu Prize | Fiction |    2 | John Stiles |
Yesterday | Example Books
2020 | Kwesi Manu Prize | Fiction |    3 | Nikki Wolf | Moving to the
Chateau | AnyPublisher
2020 | Richard Roe | Fiction |    1 | Alejandro Rosalez | Long
Summer | SomePublisher
2020 | Richard Roe | Fiction |    2 | Arnav Desai |
The Key | Example Books
2020 | Richard Roe | Fiction |    3 | Mateo Jackson | Inside
the Whale | AnyPublisher

(9 rows)
```

Pour en savoir plus sur l'utilisation `cqlsh` COPY du chargement de données depuis des fichiers CSV vers un tableau Amazon Keyspaces, consultez [the section called “Chargement de données à l'aide de cqlsh”](#).

Lire les données d'une table à l'aide de l'instruction CQL SELECT dans Amazon Keyspaces

Dans la section [Insertion et chargement de données dans un tableau Amazon Keyspaces](#), vous avez utilisé l'instruction SELECT pour vérifier que vous avez correctement ajouté des données à

vosre table. Dans cette section, vous affinez votre utilisation de SELECT pour afficher des colonnes spécifiques, et uniquement des lignes qui répondent à des critères spécifiques.

La forme générale de l'instruction SELECT est la suivante.

```
SELECT column_list FROM table_name [WHERE condition [ALLOW FILTERING]] ;
```

Rubriques

- [Sélectionnez toutes les données de votre tableau](#)
- [Sélectionnez un sous-ensemble de colonnes](#)
- [Sélectionnez un sous-ensemble de lignes](#)

Sélectionnez toutes les données de votre tableau

La forme la plus simple de l'instruction SELECT renvoie toutes les données de votre table.

Important

Dans un environnement de production, il n'est généralement pas recommandé d'exécuter cette commande, car elle renvoie toutes les données de votre table.

Pour sélectionner toutes les données de votre table

1. Ouvrez Amazon Keyspaces AWS CloudShell et connectez-vous à celui-ci à l'aide de la commande suivante. Assurez-vous de mettre à jour *us-east-1* avec votre propre région.

```
cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 --ssl
```

2. Exécutez la requête suivante :

```
SELECT * FROM catalog.book_awards ;
```

Utilisation du caractère générique (*) pour que *column_list* sélectionne toutes les colonnes. La sortie de l'instruction ressemble à l'exemple suivant.

```
year | award          | category | rank | author          | book_title
    | publisher
```

```

-----+-----+-----+-----+-----
+-----+-----+-----+-----+-----
2020 |          Wolf | Non-Fiction |    1 |          Wang Xiulan |      History
of Ideas | AnyPublisher
2020 |          Wolf | Non-Fiction |    2 | Ana Carolina Silva |
Science Today | SomePublisher
2020 |          Wolf | Non-Fiction |    3 | Shirley Rodriguez | The Future of
Sea Ice | AnyPublisher
2020 | Kwesi Manu Prize | Fiction |    1 |          Akua Mansa |      Where did
you go? | SomePublisher
2020 | Kwesi Manu Prize | Fiction |    2 |          John Stiles |
Yesterday | Example Books
2020 | Kwesi Manu Prize | Fiction |    3 |          Nikki Wolf | Moving to the
Chateau | AnyPublisher
2020 | Richard Roe | Fiction |    1 | Alejandro Rosalez |          Long
Summer | SomePublisher
2020 | Richard Roe | Fiction |    2 |          Arnav Desai |
The Key | Example Books
2020 | Richard Roe | Fiction |    3 |          Mateo Jackson |      Inside
the Whale | AnyPublisher

```

Sélectionnez un sous-ensemble de colonnes

Pour interroger un sous-ensemble de colonnes

1. Ouvrez Amazon Keyspaces AWS CloudShell et connectez-vous à celui-ci à l'aide de la commande suivante. Assurez-vous de mettre à jour *us-east-1* avec votre propre région.

```
cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 --ssl
```

2. Pour récupérer uniquement les colonnes award, category et year, exécutez la requête suivante.

```
SELECT award, category, year FROM catalog.book_awards ;
```

La sortie contient uniquement les colonnes spécifiées dans l'ordre indiqué dans l'INSTRUCTION.

```

award          | category      | year
-----+-----+-----

```

```

        Wolf | Non-Fiction | 2020
        Wolf | Non-Fiction | 2020
        Wolf | Non-Fiction | 2020
Kwesi Manu Prize | Fiction | 2020
Kwesi Manu Prize | Fiction | 2020
Kwesi Manu Prize | Fiction | 2020
        Richard Roe | Fiction | 2020
        Richard Roe | Fiction | 2020
        Richard Roe | Fiction | 2020

```

Sélectionnez un sous-ensemble de lignes

Lors de l'interrogation d'un jeu de données volumineux, vous ne souhaitez peut-être que les enregistrements répondant à certains critères. Pour ce faire, vous pouvez ajouter une clause `WHERE` à la fin de notre instruction `SELECT`.

Pour interroger un sous-ensemble de lignes

1. Ouvrez Amazon Keyspaces AWS CloudShell et connectez-vous à celui-ci à l'aide de la commande suivante. Assurez-vous de mettre à jour `us-east-1` avec votre propre région.

```
cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 --ssl
```

2. Pour récupérer uniquement les enregistrements des récompenses d'une année donnée, exécutez la requête suivante.

```
SELECT * FROM catalog.book_awards WHERE year=2020 AND award='Wolf' ;
```

L'`SELECT` instruction précédente renvoie le résultat suivant.

```

year | award | category      | rank | author                | book_title          |
publisher
-----+-----+-----+-----+-----+-----+-----
+-----+
2020 | Wolf | Non-Fiction | 1 | Wang Xiulan | History of Ideas |
AnyPublisher
2020 | Wolf | Non-Fiction | 2 | Ana Carolina Silva | Science Today |
SomePublisher
2020 | Wolf | Non-Fiction | 3 | Shirley Rodriguez | The Future of Sea Ice |
AnyPublisher

```

Comprendre la **clause** WHERE

La clause WHERE est utilisée pour filtrer les données et renvoyer uniquement celles qui répondent aux critères spécifiés. Les critères spécifiés peuvent être une condition simple ou une condition composée.

Comment utiliser des conditions dans une **clause** WHERE

- Une condition simple : une seule colonne.

```
WHERE column_name=value
```

Vous pouvez utiliser une condition simple dans une clause WHERE si l'une des conditions suivantes est remplie :

- La colonne est la seule colonne clé de partition de la table.
- Vous ajoutez `ALLOW FILTERING` après la condition de la clause WHERE.

Sachez que l'utilisation d'`ALLOW FILTERING` peut entraîner des performances incohérentes, en particulier avec les tables volumineuses et à plusieurs partitions.

- Une condition composée — Plusieurs conditions simples reliées par `AND`.

```
WHERE column_name1=value1 AND column_name2=value2 AND column_name3=value3...
```

Vous pouvez utiliser des conditions composées dans une clause WHERE si l'une des conditions suivantes est remplie :

- Les colonnes que vous pouvez utiliser dans la WHERE clause doivent inclure la totalité ou un sous-ensemble des colonnes de la clé de partition de la table. Si vous souhaitez utiliser uniquement un sous-ensemble des colonnes de la WHERE clause, vous devez inclure un ensemble contigu de colonnes de clé de partition de gauche à droite, en commençant par la colonne de tête de la clé de partition. Par exemple, si les colonnes de la clé de partition sont `yearmonth,, award` vous pouvez utiliser les colonnes suivantes dans la WHERE clause :

- `year`
- `yearET month`
- `yearET month ET award`

- Vous ajoutez `ALLOW FILTERING` après la condition composée dans la clause WHERE, comme dans l'exemple suivant.

```
SELECT * FROM my_table WHERE col1=5 AND col2='Bob' ALLOW FILTERING ;
```

Sachez que l'utilisation d'ALLOW FILTERING peut entraîner des performances incohérentes, en particulier avec les tables volumineuses et à plusieurs partitions.

Essayez-le

Créez vos propres requêtes CQL pour trouver les éléments suivants dans votre table `book_awards` :

- Trouvez les gagnants des prix Wolf 2020 et affichez les titres des livres et les auteurs, classés par ordre de classement.
- Montrez les gagnants des premiers prix de tous les prix en 2020 et affichez les titres des livres et les noms des récompenses.

Mettre à jour les données d'une table Amazon Keyspaces à l'aide de CQL

Pour mettre à jour les données de votre table `book_awards`, utilisez l'instruction UPDATE.

La forme générale de l'instruction UPDATE est la suivante.

```
UPDATE table_name SET column_name=new_value WHERE primary_key=value ;
```

Tip

- Vous pouvez mettre à jour plusieurs colonnes à l'aide d'une liste de `column_names` et de valeurs séparées par des virgules, comme dans l'exemple suivant.

```
UPDATE my_table SET col1='new_value_1', col2='new_value2' WHERE col3='1' ;
```

- Si la clé primaire est composée de plusieurs colonnes, toutes les colonnes de clé primaire et leurs valeurs doivent être incluses dans la clause WHERE.
- Vous ne pouvez pas mettre à jour une colonne de la clé primaire car cela modifierait la clé primaire de l'enregistrement.

Pour mettre à jour une cellule unique

À l'aide de votre `book_awards` tableau, changez le nom d'un éditeur en tant que gagnant des Wolf Awards non-fictionnels en 2020.

```
UPDATE book_awards SET publisher='new Books' WHERE year = 2020 AND award='Wolf' AND
category='Non-Fiction' AND rank=1;
```

Vérifiez que l'éditeur l'est actuellement `new Books`.

```
SELECT * FROM book_awards WHERE year = 2020 AND award='Wolf' AND category='Non-Fiction'
AND rank=1;
```

L'instruction doit renvoyer le résultat suivant.

year	award	category	rank	author	book_title	publisher
2020	Wolf	Non-Fiction	1	Wang Xiulan	History of Ideas	new Books

Essayez-le

Avancé : Le gagnant du « Prix Kwezi Manu » de fiction 2020 a changé de nom. Mettez à jour cet enregistrement pour changer le nom en 'Akua Mansa-House'.

Supprimer les données d'une table à l'aide de l'instruction CQL **DELETE**

Pour supprimer les données de votre table `book_awards`, utilisez l'instruction `DELETE`.

Vous pouvez supprimer des données d'une ligne ou d'une partition. Soyez prudent lorsque vous supprimez des données, car les suppressions sont irréversibles.

La suppression d'une ou de toutes les lignes d'un tableau ne supprime pas le tableau. Ainsi, vous pouvez le remplir à nouveau avec des données. La suppression d'une table supprime la table et toutes les données qu'elle contient. Pour réutiliser la table, vous devez la recréer et y ajouter des données. La suppression d'un keyspace supprime le keyspace et toutes les tables qu'il contient. Pour utiliser le keyspace et les tables, vous devez les recréer, puis les remplir avec des données. Vous pouvez utiliser la restauration Amazon Keyspaces Point-in-time (PITR) pour restaurer les tables supprimées. Pour en savoir plus, consultez [the section called “Backup et restauration avec restauration instantanée”](#) Pour savoir comment restaurer une table supprimée avec PITR activé, consultez [the section called “Restaurer une table supprimée”](#).

Supprimer des cellules

La suppression d'une colonne d'une ligne supprime les données de la cellule spécifiée. Lorsque vous affichez cette colonne à l'aide d'une SELECT instruction, les données sont affichées sous forme *null*, bien qu'aucune valeur nulle ne soit stockée à cet emplacement.

La syntaxe générale permettant de supprimer une ou plusieurs colonnes spécifiques est la suivante.

```
DELETE column_name1[, column_name2...] FROM table_name WHERE condition ;
```

Dans votre book_awards tableau, vous pouvez voir que le titre du livre qui a remporté le premier prix du prix « Richard Roe » 2020 est « Long Summer ». Imaginez que ce titre ait été rappelé et que vous deviez supprimer les données de cette cellule.

Pour supprimer une cellule spécifique

1. Ouvrez Amazon Keyspaces AWS CloudShell et connectez-vous à celui-ci à l'aide de la commande suivante. Assurez-vous de mettre à jour *us-east-1* avec votre propre région.

```
cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 --ssl
```

2. Exécutez la requête DELETE suivante :

```
DELETE book_title FROM catalog.book_awards WHERE year=2020 AND award='Richard Roe'
AND category='Fiction' AND rank=1;
```

3. Vérifiez que la demande de suppression a été faite comme prévu.

```
SELECT * FROM catalog.book_awards WHERE year=2020 AND award='Richard Roe' AND
category='Fiction' AND rank=1;
```

Le résultat de cette instruction ressemble à ceci.

year	award	category	rank	author	book_title	publisher
2020	Richard Roe	Fiction	1	Alejandro Rosalez	null	SomePublisher

Supprimer les lignes

Il peut arriver que vous deviez supprimer une ligne entière, par exemple pour répondre à une demande de suppression de données. La syntaxe générale pour supprimer une ligne est la suivante.

```
DELETE FROM table_name WHERE condition ;
```

Pour supprimer une ligne

1. Ouvrez Amazon Keyspaces AWS CloudShell et connectez-vous à celui-ci à l'aide de la commande suivante. Assurez-vous de mettre à jour *us-east-1* avec votre propre région.

```
cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 --ssl
```

2. Exécutez la requête DELETE suivante :

```
DELETE FROM catalog.book_awards WHERE year=2020 AND award='Richard Roe' AND  
category='Fiction' AND rank=1;
```

3. Vérifiez que la suppression a été effectuée comme prévu.

```
SELECT * FROM catalog.book_awards WHERE year=2020 AND award='Richard Roe' AND  
category='Fiction' AND rank=1;
```

Le résultat de cette instruction ressemble à ceci une fois la ligne supprimée.

```
year | award | category | rank | author | book_title | publisher  
-----+-----+-----+-----+-----+-----+-----  
  
(0 rows)
```

Vous pouvez supprimer automatiquement les données expirées de votre tableau à l'aide d'Amazon Keyspaces Time to Live. Pour plus d'informations, consultez [the section called “Expirer les données avec Time to Live”](#).

Supprimer un tableau dans Amazon Keyspaces

Pour éviter d'être facturé pour des tableaux et des données dont vous n'avez pas besoin, supprimez tous les tableaux que vous n'utilisez pas. Lorsque vous supprimez une table, la table et ses données

sont supprimées et vous arrêtez d'accumuler des frais pour eux. Cependant, le keyspace demeure. Lorsque vous supprimez un keyspace, le keyspace et toutes ses tables sont supprimés et vous cessez d'accumuler des frais pour eux.

Vous pouvez supprimer une table à l'aide de la console, du CQL ou du AWS CLI. Lorsque vous supprimez une table, la table et toutes ses données sont supprimées.

Utilisation de la console

La procédure suivante supprime une table et toutes ses données avec Console de gestion AWS.

Pour supprimer une table à l'aide de la console

1. Connectez-vous au Console de gestion AWS, puis ouvrez la console Amazon Keyspaces à <https://console.aws.amazon.com/keyspaces/home> l'adresse.
2. Dans le volet de navigation, choisissez Tables.
3. Sélectionnez la case située à gauche du nom de chaque table à supprimer.
4. Sélectionnez Delete (Supprimer).
5. Dans l'écran Supprimer la table entrez **Delete** dans la zone. Puis, choisissez Delete Table (Supprimer la table).
6. Pour vérifier que la table a été supprimée, choisissez Tables dans le panneau de navigation et vérifiez que la table `book_awards` n'est plus répertoriée.

Utilisation de CQL

La procédure suivante supprime une table et toutes ses données à l'aide de CQL.

Pour supprimer une table avec CQL

1. Ouvrez Amazon Keyspaces AWS CloudShell et connectez-vous à celui-ci à l'aide de la commande suivante. Assurez-vous de mettre à jour `us-east-1` avec votre propre région.

```
cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 --ssl
```

2. Supprimez votre tableau en saisissant l'instruction suivante.

```
DROP TABLE IF EXISTS catalog.book_awards ;
```

3. Vérifiez que votre table a été supprimée.

```
SELECT * FROM system_schema.tables WHERE keyspace_name = 'catalog' ;
```

La sortie devrait ressembler à ceci. Notez que cela peut prendre un certain temps, donc réexécutez l'instruction au bout d'une minute si vous ne voyez pas ce résultat.

```
keyspace_name | table_name | bloom_filter_fp_chance | caching | cdc | comment
| compaction | compression | crc_check_chance | dclocal_read_repair_chance
| default_time_to_live | extensions | flags | gc_grace_seconds | id |
max_index_interval | memtable_flush_period_in_ms | min_index_interval |
read_repair_chance | speculative_retry
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

(0 rows)
```

Utilisation de AWS CLI

La procédure suivante supprime une table et toutes ses données avec AWS CLI.

Pour supprimer un tableau à l'aide du AWS CLI

1. Ouvrir CloudShell
2. Supprimez votre tableau à l'aide de l'instruction suivante.

```
aws keyspaces delete-table --keyspace-name 'catalog' --table-name 'book_awards'
```

3. Pour vérifier que votre table a été supprimée, vous pouvez répertorier toutes les tables dans un espace clé.

```
aws keyspaces list-tables --keyspace-name 'catalog'
```

Le résultat suivant doit s'afficher. Notez que cette opération asynchrone peut prendre un certain temps. Re-run la commande après un court instant pour confirmer que la table a été supprimée.

```
{
```

```
"tables": []  
}
```

Supprimer un espace clé dans Amazon Keyspaces

Pour éviter d'être facturé pour les espaces clés, supprimez tous les espaces clés que vous n'utilisez pas. Lorsque vous supprimez un keyspace, le keyspace et toutes ses tables sont supprimés et vous cessez d'accumuler des frais pour eux.

Vous pouvez supprimer un espace clé à l'aide de la console, du CQL ou du AWS CLI

Utilisation de la console

La procédure suivante supprime un espace clé et toutes ses tables et données à l'aide de la console.

Pour supprimer un keyspace à l'aide de la console

1. Connectez-vous au Console de gestion AWS, puis ouvrez la console Amazon Keyspaces à <https://console.aws.amazon.com/keyspaces/home> l'adresse.
2. Dans le panneau de navigation, choisissez Keyspaces.
3. Sélectionnez la case située à gauche du nom de chaque keyspace que vous souhaitez supprimer.
4. Sélectionnez Delete (Supprimer).
5. Dans l'écran Supprimer le keyspace entrez **Delete** dans la zone. Ensuite, choisissez Supprimer le keyspace.
6. Pour vérifier que le keyspace `catalog` a été supprimé, choisissez Keyspaces dans le panneau de navigation et vérifiez qu'il n'est plus répertorié. Étant donné que vous avez supprimé son keyspace, la table `book_awards` sous Tables ne doit pas non plus être répertoriée.

Utilisation de CQL

La procédure suivante supprime un keyspace et toutes ses tables et données à l'aide de CQL.

Pour supprimer un keyspace avec CQL

1. Ouvrez Amazon Keyspaces AWS CloudShell et connectez-vous à celui-ci à l'aide de la commande suivante. Assurez-vous de mettre à jour *us-east-1* avec votre propre région.

```
cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 --ssl
```

2. Supprimez votre espace clavier en saisissant l'instruction suivante.

```
DROP KEYSPACE IF EXISTS catalog ;
```

3. Vérifiez que votre keyspace a été supprimé.

```
SELECT * from system_schema.keyspaces ;
```

Votre keyspace ne doit pas être répertorié. Notez qu'étant donné qu'il s'agit d'une opération asynchrone, il peut y avoir un délai jusqu'à ce que l'espace clé soit supprimé. Une fois l'espace clavier supprimé, la sortie de l'instruction devrait ressembler à ceci.

```
keyspace_name          | durable_writes | replication
-----+-----
+-----+-----
          system_schema |             True | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
          system_schema_mcs |             True | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
              system |             True | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
system_multiregion_info |             True | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}

(4 rows)
```

Utilisation de AWS CLI

La procédure suivante supprime un keyspace ainsi que toutes ses tables et données avec AWS CLI.

Pour supprimer un espace clé à l'aide du AWS CLI

1. Ouvrir AWS CloudShell
2. Supprimez votre espace clavier en saisissant l'instruction suivante.

```
aws keyspaces delete-keyspace --keyspace-name 'catalog'
```

3. Vérifiez que votre keyspace a été supprimé.

```
aws keyspaces list-keyspaces
```

Le résultat de cette instruction doit ressembler à celui-ci et ne répertorier que les espaces clés du système. Notez qu'étant donné qu'il s'agit d'une opération asynchrone, il peut y avoir un délai jusqu'à ce que votre espace clé soit supprimé.

```
{
  "keyspaces": [
    {
      "keyspaceName": "system_schema",
      "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
system_schema/",
      "replicationStrategy": "SINGLE_REGION"
    },
    {
      "keyspaceName": "system_schema_mcs",
      "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
system_schema_mcs/",
      "replicationStrategy": "SINGLE_REGION"
    },
    {
      "keyspaceName": "system",
      "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
system/",
      "replicationStrategy": "SINGLE_REGION"
    },
    {
      "keyspaceName": "system_multiregion_info",
      "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
system_multiregion_info/",
      "replicationStrategy": "SINGLE_REGION"
    }
  ]
}
```

Tutoriels et solutions pour Amazon Keyspaces

Cette rubrique inclut différents didacticiels qui décrivent en détail les étapes requises pour se connecter à Amazon Keyspaces par programmation ou pour utiliser Amazon Keyspaces avec d'autres services afin de créer des solutions. AWS Les didacticiels disponibles couvrent différents scénarios d'accès multiservices courants et montrent comment intégrer Amazon Keyspaces à Amazon Virtual Private Cloud, Amazon Elastic Kubernetes Service, Amazon Simple AWS Glue Storage Service et/ou à des technologies open source telles qu'Apache Spark.

Pour des didacticiels détaillés expliquant comment se connecter à Amazon Keyspaces à l'aide des pilotes Apache Cassandra, consultez. [the section called “Utilisation d'un pilote client Cassandra”](#)

Rubriques

- [Tutoriel : Connectez-vous à Amazon Keyspaces à l'aide d'un point de terminaison VPC d'interface](#)
- [Tutoriel : Intégrer à Apache Spark pour importer ou exporter des données](#)
- [Tutoriel : Connectez-vous depuis une application conteneurisée hébergée sur Amazon Elastic Kubernetes Service](#)
- [Tutoriel : Exporter un tableau Amazon Keyspaces vers Amazon S3 à l'aide de AWS Glue](#)

Tutoriel : Connectez-vous à Amazon Keyspaces à l'aide d'un point de terminaison VPC d'interface

Ce didacticiel explique comment configurer et utiliser un point de terminaison VPC d'interface pour Amazon Keyspaces.

Les points de terminaison VPC d'interface permettent une communication privée entre votre cloud privé virtuel (VPC) exécuté dans Amazon VPC et Amazon Keyspaces. Les points de terminaison VPC d'interface sont alimentés par AWS PrivateLink un AWS service qui permet une communication privée entre les VPC et les services. AWS Pour de plus amples informations, veuillez consulter [the section called “Utilisation des points de terminaison de VPC d'interface”](#).

Rubriques

- [Conditions préalables et considérations relatives au didacticiel](#)
- [Étape 1 : lancer une instance Amazon EC2](#)
- [Étape 2 : configurer votre instance Amazon EC2](#)

- [Étape 3 : créer un point de terminaison VPC pour Amazon Keyspaces](#)
- [Étape 4 : configurer les autorisations pour la connexion du point de terminaison VPC](#)
- [Étape 5 : Configuration de la surveillance avec CloudWatch](#)
- [Étape 6 : \(Facultatif\) Meilleures pratiques pour configurer la taille du pool de connexions pour votre application](#)
- [Étape 7 : \(Facultatif\) Nettoyer](#)

Conditions préalables et considérations relatives au didacticiel

Avant de commencer ce didacticiel, suivez les instructions de AWS configuration indiquées dans [Accès à Amazon Keyspaces \(pour Apache Cassandra\)](#). Ces étapes incluent l'inscription AWS et la création d'un Gestion des identités et des accès AWS (IAM) principal ayant accès à Amazon Keyspaces. Prenez note du nom de l'utilisateur IAM et des clés d'accès, car vous en aurez besoin plus tard dans ce didacticiel.

Créez un espace de touches avec le nom myKeyspace et au moins une table pour tester la connexion à l'aide du point de terminaison VPC plus loin dans ce didacticiel. Vous trouverez des instructions détaillées dans [Prise en main](#).

Après avoir effectué les étapes préalables, passez à [Étape 1 : lancer une instance Amazon EC2](#).

Étape 1 : lancer une instance Amazon EC2

Au cours de cette étape, vous allez lancer une instance Amazon EC2 dans votre VPC Amazon par défaut. Vous pouvez ensuite créer et utiliser un point de terminaison VPC pour Amazon Keyspaces.

Pour lancer une instance Amazon EC2

1. Ouvrez la console Amazon EC2 à l'adresse <https://console.aws.amazon.com/ec2/>.
2. Choisissez Lancer une instance , puis procédez comme suit :

Dans le tableau de bord de la console EC2, dans la zone Launch instance, choisissez Launch instance, puis choisissez Launch instance parmi les options qui s'affichent.

Sous Nom et balises, pour Nom, entrez un nom descriptif pour votre instance.

Sous Images de l'application et du système d'exploitation (Amazon Machine Image) :

- Choisissez Quick Start, puis Ubuntu. Il s'agit du système d'exploitation (OS) de votre instance.

- Sous Amazon Machine Image (AMI), vous pouvez utiliser l'image par défaut marquée comme éligible au niveau gratuit. Une Amazon Machine Image (AMI) est une configuration de base qui sert de modèle à votre instance.

Sous Type d'instance :

- Dans la liste Type d'instance, choisissez le type d'instance t2.micro, qui est sélectionné par défaut.

Sous Paire de clés (connexion), pour Nom de la paire de clés, choisissez l'une des options suivantes pour ce didacticiel :

- Si vous n'avez pas de paire de clés Amazon EC2, choisissez Créer une paire de clés, puis suivez les instructions. Il vous sera demandé de télécharger un fichier de clé privée (fichier .pem). Vous aurez besoin de ce fichier ultérieurement lorsque vous vous connecterez à votre instance Amazon EC2. Prenez donc note du chemin du fichier.
- Si vous possédez déjà une paire de clés Amazon EC2, accédez à Select a key pair (Sélectionner une paire de clés), puis choisissez votre paire de clés dans la liste. Vous devez déjà posséder le fichier de clé privée (fichier .pem) pour pouvoir vous connecter à votre instance Amazon EC2.

Sous Paramètres réseau :

- Choisissez Modifier.
- Choisissez Select an existing security group.
- Dans la liste des groupes de sécurité, choisissez default. Il s'agit du groupe de sécurité par défaut pour votre VPC.

Passez au résumé.

- Consultez un résumé de la configuration de votre instance dans le panneau Résumé. Une fois que vous êtes prêt, choisissez Lancer l'instance.
3. Sur l'écran de finalisation de la nouvelle instance Amazon EC2, choisissez la vignette Connect to instance. L'écran suivant affiche les informations nécessaires et les étapes requises pour vous connecter à votre nouvelle instance. Prenez note des informations suivantes :

- Exemple de commande pour protéger le fichier clé
- La chaîne de connexion
- Le nom DNS IPv4 public

Après avoir pris note des informations de cette page, vous pouvez passer à l'étape suivante de ce didacticiel ([Étape 2 : configurer votre instance Amazon EC2](#)).

Note

Votre instance Amazon EC2 devient disponible en l'espace de quelques minutes. Avant de passer à l'étape suivante, assurez-vous que l'Instance State (État de l'instance) est `running` et que tous ses Status Checks (Contrôles d'état) ont réussi.

Étape 2 : configurer votre instance Amazon EC2

Lorsque votre instance Amazon EC2 est disponible, vous pouvez vous y connecter et la préparer pour la première utilisation.

Note

Les étapes suivantes supposent que vous vous connectez à votre instance Amazon EC2 depuis un ordinateur exécutant Linux. Pour d'autres méthodes de connexion, consultez [Connect to your Linux instance](#) dans le guide de l'utilisateur Amazon EC2.

Pour configurer votre instance Amazon EC2

1. Vous devez autoriser le trafic SSH entrant vers votre instance Amazon EC2. Pour ce faire, créez un nouveau groupe de sécurité EC2, puis attribuez-le à votre instance EC2.
 - a. Dans le panneau de navigation, choisissez Groupes de sécurité.
 - b. Sélectionnez Créer un groupe de sécurité. Dans la fenêtre Créer un groupe de sécurité, procédez comme suit :

- Nom du groupe de sécurité — Entrez le nom de votre groupe de sécurité. Par exemple : `my-ssh-access`
- Description — Entrez une brève description du groupe de sécurité.
- VPC — Choisissez votre VPC par défaut.
- Dans la section Règles de trafic entrant, choisissez Ajouter une règle et procédez comme suit :
 - Type — Choisissez SSH.
 - Source — Choisissez mon adresse IP.
 - Choisissez Ajouter une règle.

Au bas de la page, confirmez les paramètres de configuration et choisissez Create Security Group.

- Dans le panneau de navigation, choisissez Instances.
 - Choisissez l'instance Amazon EC2 que vous avez lancée dans [Étape 1 : lancer une instance Amazon EC2](#).
 - Choisissez Actions, sélectionnez Sécurité, puis sélectionnez Modifier les groupes de sécurité.
 - Dans Modifier les groupes de sécurité, accédez à Groupes de sécurité associés et entrez le groupe de sécurité que vous avez créé plus tôt dans cette procédure (par exemple, `my-ssh-access`). Le groupe de sécurité default existant doit également être sélectionné. Confirmez les paramètres de configuration et choisissez Enregistrer.
- Utilisez la commande suivante pour empêcher l'accès à votre fichier de clé privée. Si vous ignorez cette étape, la connexion échoue.

```
chmod 400 path_to_file/my-keypair.pem
```

- Utilisez la commande `ssh` pour vous connecter à votre instance Amazon EC2, comme dans l'exemple suivant.

```
ssh -i path_to_file/my-keypair.pem ubuntu@public-dns-name
```

Vous devez spécifier votre fichier de clé privée (fichier `.pem`) et le nom DNS public de votre instance. (Consultez [Étape 1 : lancer une instance Amazon EC2](#)).

L'ID de connexion est ubuntu. Aucun mot de passe n'est requis.

Pour plus d'informations sur l'autorisation des connexions à votre instance Amazon EC2 et pour AWS CLI obtenir des instructions, consultez [Autoriser le trafic entrant pour vos instances Linux](#) dans le guide de l'utilisateur Amazon EC2.

4. Téléchargez et installez la dernière version du AWS Command Line Interface.

a. Installer unzip.

```
sudo apt install unzip
```

b. Téléchargez le zip fichier avec le AWS CLI.

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o  
"awscliv2.zip"
```

c. Décompressez le fichier.

```
unzip awscliv2.zip
```

d. Installez le AWS CLI.

```
sudo ./aws/install
```

e. Vérifiez la version de l' AWS CLI installation.

```
aws --version
```

La sortie doit se présenter comme suit :

```
aws-cli/2.9.19 Python/3.9.11 Linux/5.15.0-1028-aws exe/x86_64.ubuntu.22 prompt/  
off
```

5. Configurez vos AWS informations d'identification, comme indiqué dans l'exemple suivant. Entrez l'ID de votre clé d' AWS accès, votre clé secrète et le nom de région par défaut lorsque vous y êtes invité.

```
aws configure
```

```
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
```

```
AWS Secret Access Key [None]: wJa1rXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-east-1
Default output format [None]:
```

6. Ajoutez des règles au groupe de sécurité de vos VPC qui autorisent l'accès HTTP, HTTPS et SSH entrant à partir d'adresses IPv6.
7. Pour vérifier que votre point de terminaison VPC a été correctement configuré, vous devez utiliser une `cqlsh` connexion à Amazon Keyspaces. Si vous utilisez votre environnement local ou l'éditeur CQL d'Amazon Keyspaces dans le Console de gestion AWS, la connexion passe automatiquement par le point de terminaison public plutôt que par votre point de terminaison VPC. À utiliser `cqlsh` pour tester la connexion de votre point de terminaison VPC dans ce didacticiel, suivez les instructions de configuration dans [Utilisation cqlsh pour se connecter à Amazon Keyspaces](#)

Vous êtes maintenant prêt à créer un point de terminaison VPC à double pile pour Amazon Keyspaces.

Étape 3 : créer un point de terminaison VPC pour Amazon Keyspaces

Au cours de cette étape, vous créez un point de terminaison VPC à double pile pour Amazon Keyspaces à l'aide du AWS CLI. Pour créer le point de terminaison VPC à l'aide de la console VPC, vous pouvez suivre les instructions de création [d'un point de terminaison VPC figurant](#) dans le guide AWS PrivateLink. Lorsque vous filtrez le nom du service, entrez **Cassandra**.

Pour créer un point de terminaison VPC à l'aide du AWS CLI

1. Avant de commencer, vérifiez que vous pouvez communiquer avec Amazon Keyspaces via son point de terminaison public.

```
aws keyspace list-tables --keyspace-name 'myKeyspace'
```

Le résultat affiche une liste des tables Amazon Keyspaces contenues dans l'espace de touches spécifié. Si vous n'avez aucune table, la liste est vide.

```
{
  "tables": [
    {
      "keyspaceName": "myKeyspace",
      "tableName": "myTable1",
```

```

        "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
catalog/table/myTable1"
      },
      {
        "keyspaceName": "myKeyspace",
        "tableName": "myTable2",
        "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
catalog/table/myTable2"
      }
    ]
  }
}

```

2. Vérifiez qu'Amazon Keyspaces est un service disponible pour créer des points de terminaison VPC dans la région actuelle. AWS (la commande est affichée en gras, suivie d'un exemple de sortie).

```

aws ec2 describe-vpc-endpoint-services

{
  "ServiceNames": [
    "com.amazonaws.us-east-1.cassandra",
    "com.amazonaws.us-east-1.cassandra-fips"
    "api.aws.us-east-1.cassandra-streams"
  ]
}

```

Si Amazon Keyspaces est l'un des services disponibles dans le résultat de la commande, vous pouvez procéder à la création d'un point de terminaison VPC.

3. Pour vous connecter à Amazon Keyspaces à l'aide de points de terminaison à double pile compatibles IPv6, vérifiez que votre VPC prend en charge IPv6 et configurez des sous-réseaux compatibles IPv6. Pour ajouter le support IPv6 à un VPC existant qui ne prend actuellement en charge que l'IPv4, consultez le [support IPv6 pour votre VPC](#) dans le guide de l'utilisateur Amazon VPC ;
4. Déterminez votre identifiant de VPC.

```

aws ec2 describe-vpcs

{
  "Vpcs": [
    {
      "OwnerId": "111122223333",

```

```

    "InstanceTenancy": "default",
    "Ipv6CidrBlockAssociationSet": [
      {
        "AssociationId": "vpc-cidr-assoc-0000aaa0a00a00aa0",
        "Ipv6CidrBlock": "2600:1f18:e19:7d00::/56",
        "Ipv6CidrBlockState": {
          "State": "associated"
        },
        "NetworkBorderGroup": "us-east-1",
        "Ipv6Pool": "Amazon",
        "Ipv6AddressAttribute": "public",
        "IpSource": "amazon"
      }
    ],
    "CidrBlockAssociationSet": [
      {
        "AssociationId": "vpc-cidr-assoc-00a0000a",
        "CidrBlock": "111.11.0.0/16",
        "CidrBlockState": {
          "State": "associated"
        }
      }
    ],
    "IsDefault": true,
    "BlockPublicAccessStates": {
      "InternetGatewayBlockMode": "off"
    },
    "VpcId": "vpc-a1234bcd",
    "State": "available",
    "CidrBlock": "111.11.0.0/16",
    "DhcpOptionsId": "dopt-a00aaaaa"
  }
]
}

```

Dans l'exemple de sortie, l'ID de VPC est `vpc-a1234bcd`.

5. Utilisez un filtre pour recueillir des informations sur les sous-réseaux du VPC.

```

aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-a1234bcd"

{
  "Subnets": [
    {

```



```

    "AvailabilityZoneId": "use1-az1",
    "MapCustomerOwnedIpOnLaunch": false,
    "OwnerId": "111122223333",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [
      {
        "AssociationId": "subnet-cidr-assoc-05d75732736740283",
        "Ipv6CidrBlock": "*****",
        "Ipv6CidrBlockState": {
          "State": "associated"
        },
        "Ipv6AddressAttribute": "public",
        "IpSource": "amazon"
      }
    ],
    "SubnetArn": "arn:aws:ec2:us-east-1:111122223333:subnet/
subnet-70b24b16",
    "EnableDns64": false,
    "Ipv6Native": false,
    "PrivateDnsNameOptionsOnLaunch": {
      "HostnameType": "ip-name",
      "EnableResourceNameDnsARecord": false,
      "EnableResourceNameDnsAAAARecord": false
    },
    "BlockPublicAccessStates": {
      "InternetGatewayBlockMode": "off"
    },
    "SubnetId": "subnet-70b24b16",
    "State": "available",
    "VpcId": "vpc-a1234bcd",
    "CidrBlock": "*****/20",
    "AvailableIpAddressCount": 4089,
    "AvailabilityZone": "us-east-1a",
    "DefaultForAz": true,
    "MapPublicIpOnLaunch": true
  },
  {
    "AvailabilityZoneId": "use1-az2",
    "MapCustomerOwnedIpOnLaunch": false,
    "OwnerId": "111122223333",
    "AssignIpv6AddressOnCreation": false,
    "Ipv6CidrBlockAssociationSet": [
      {
        "AssociationId": "subnet-cidr-assoc-0ec6fb253e05b17eb",

```

```

        "Ipv6CidrBlock": "*****",
        "Ipv6CidrBlockState": {
            "State": "associated"
        },
        "Ipv6AddressAttribute": "public",
        "IpSource": "amazon"
    },
    ],
    "SubnetArn": "arn:aws:ec2:us-east-1:111122223333:subnet/subnet-
c63ffbe7",
    "EnableDns64": false,
    "Ipv6Native": false,
    "PrivateDnsNameOptionsOnLaunch": {
        "HostnameType": "ip-name",
        "EnableResourceNameDnsARecord": false,
        "EnableResourceNameDnsAAAARecord": false
    },
    "BlockPublicAccessStates": {
        "InternetGatewayBlockMode": "off"
    },
    "SubnetId": "subnet-c63ffbe7",
    "State": "available",
    "VpcId": "vpc-a1234bcd",
    "CidrBlock": "*****/20",
    "AvailableIpAddressCount": 4087,
    "AvailabilityZone": "us-east-1b",
    "DefaultForAz": true,
    "MapPublicIpOnLaunch": true
}
]
}

```

Dans l'exemple de sortie, deux ID de sous-réseau sont disponibles : subnet-70b24b16 et subnet-c63ffbe7.

6. Créez le point de terminaison de VPC. Pour le paramètre `--vpc-id`, spécifiez l'ID de VPC de l'étape précédente. Pour le `--subnet-ids` paramètre, spécifiez les ID de sous-réseau de l'étape précédente. Utilisez le `--vpc-endpoint-type` paramètre pour définir le point de terminaison en tant qu'interface. Pour créer un point de terminaison à double pile, utilisez `--ip-address-type dualstack`. Pour plus d'informations sur la commande, reportez-vous [create-vpc-endpoint](#) à la référence des AWS CLI commandes.

```
aws ec2 create-vpc-endpoint \
  --vpc-endpoint-type Interface \
  --vpc-id vpc-a1234bcd \
  --ip-address-type dualstack \
  --service-name com.amazonaws.us-east-1.cassandra \
  --subnet-ids subnet-70b24b16 subnet-c63ffbe7

{
  "VpcEndpoint": {
    "VpcEndpointId": "vpce-0000000abc111d2ef3",
    "VpcEndpointType": "Interface",
    "VpcId": "vpc-a1234bcd",
    "ServiceName": "com.amazonaws.us-east-1.cassandra",
    "State": "pending",
    "RouteTableIds": [],
    "SubnetIds": [
      "subnet-70b24b16",
      "subnet-c63ffbe7"
    ],
    "Groups": [
      {
        "GroupId": "sg-0123456789",
        "GroupName": "default"
      }
    ],
    "IpAddressType": "dualstack",
    "DnsOptions": {
      "DnsRecordIpType": "dualstack"
    },
    "PrivateDnsEnabled": true,
    "RequesterManaged": false,
    "NetworkInterfaceIds": [
      "eni-08cd525f72ea6f1fa",
      "eni-07b1f6c895169d8fb"
    ],
    "DnsEntries": [
      {
        "DnsName": "vpce-000000000000-1234567.cassandra.us-
east-1.vpce.amazonaws.com",
        "HostedZoneId": "Z7HUB22UULQXV"
      }
    ]
  }
}
```

```

        "DnsName": "vpce-0000000000-1234567-us-east-1a.cassandra.us-
east-1.vpce.amazonaws.com",
        "HostedZoneId": "Z7HUB22UULQXV"
    },
    {
        "DnsName": "cassandra.us-east-1.amazonaws.com",
        "HostedZoneId": "ZONEIDPENDING"
    },
    {
        "DnsName": "cassandra.us-east-1.api.aws",
        "HostedZoneId": "ZONEIDPENDING"
    }
],
"CreationTimestamp": "2025-09-19T15:19:19.266000+00:00",
"OwnerId": "111122223333",
"ServiceRegion": "us-east-1"
}
}

```

Étape 4 : configurer les autorisations pour la connexion du point de terminaison VPC

Les procédures de cette étape montrent comment configurer les règles et les autorisations pour utiliser le point de terminaison VPC avec Amazon Keyspaces.

Pour configurer une règle entrante pour le nouveau point de terminaison afin d'autoriser le trafic entrant TCP

1. Dans la console Amazon VPC, sur le panneau de gauche, choisissez Endpoints et choisissez le point de terminaison que vous avez créé à l'étape précédente.
2. Choisissez Gérer les groupes de sécurité, puis choisissez le groupe de sécurité associé à ce point de terminaison.
3. Choisissez Règles entrantes, puis sélectionnez Modifier les règles entrantes.
4. Ajoutez une règle entrante dont le type est CQLSH/CASSANDRA. Cela définit automatiquement la plage de ports à 9142.
5. Pour enregistrer la nouvelle règle de trafic entrant, choisissez Enregistrer les règles.

Pour configurer les autorisations des utilisateurs IAM

1. Vérifiez que l'utilisateur IAM utilisé pour se connecter à Amazon Keyspaces dispose des autorisations appropriées. Dans Gestion des identités et des accès AWS (IAM), vous pouvez utiliser la politique AWS gérée `AmazonKeyspacesReadOnlyAccess` pour accorder à l'utilisateur IAM un accès en lecture à Amazon Keyspaces.
 - a. Connectez-vous à la console IAM Console de gestion AWS et ouvrez-la à <https://console.aws.amazon.com/iam/> l'adresse.
 - b. Dans le tableau de bord de la console IAM, choisissez Users (Utilisateurs), puis votre utilisateur IAM dans la liste.
 - c. Sur la page Récapitulatif, choisissez Ajouter des autorisations.
 - d. Choisissez Attacher directement les stratégies existantes.
 - e. Dans la liste des politiques, choisissez `AmazonKeyspacesReadOnlyAccess`, puis cliquez sur Suivant : Réviser.
 - f. Choisissez Ajouter des autorisations.
2. Vérifiez que vous pouvez accéder à Amazon Keyspaces via le point de terminaison VPC.

```
aws keyspaces list-tables --keyspace-name 'my_KeySpace'
```

Si vous le souhaitez, vous pouvez essayer d'autres AWS CLI commandes pour Amazon Keyspaces. Pour plus d'informations, consultez la référence de la commande [AWS CLI](#).

Note

Les autorisations minimales requises pour qu'un utilisateur ou un rôle IAM puisse accéder à Amazon Keyspaces sont des autorisations de lecture sur la table système, comme indiqué dans la politique suivante. Pour plus d'informations sur les autorisations basées sur des politiques, consultez [the section called “Identity-based examples de politiques”](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```

    "Action": [
      "cassandra:Select"
    ],
    "Resource": [
      "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
    ]
  }
]
}

```

3. Accordez à l'utilisateur IAM un accès en lecture à l'instance Amazon EC2 avec le VPC.

Lorsque vous utilisez Amazon Keyspaces avec des points de terminaison VPC, vous devez accorder à l'utilisateur ou au rôle IAM qui accède à Amazon Keyspaces des autorisations en lecture seule sur votre instance Amazon EC2 et au VPC pour recueillir les données des points de terminaison et d'interface réseau. Amazon Keyspaces enregistre ces informations dans le `system.peers` tableau et les utilise pour gérer les connexions.

Note

Les politiques gérées `AmazonKeyspacesFullAccess` incluent `AmazonKeyspacesReadOnlyAccess_v2` les autorisations requises pour permettre à Amazon Keyspaces d'accéder à l'instance Amazon EC2 afin de lire les informations sur les points de terminaison VPC d'interface disponibles.

- a. Connectez-vous à la console IAM Console de gestion AWS et ouvrez-la à <https://console.aws.amazon.com/iam/> l'adresse.
- b. Sur le tableau de bord de la console IAM, sélectionnez Politiques.
- c. Choisissez Create policy, puis sélectionnez l'onglet JSON.
- d. Copiez la politique suivante et choisissez Next : Tags.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListVPCEndpoints",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeNetworkInterfaces",

```

```

        "ec2:DescribeVpcEndpoints"
      ],
      "Resource": "*"
    }
  ]
}

```

- e. Choisissez Suivant : Réviser, entrez le nom `keyspacesVPCendpoint` de la politique, puis choisissez Créer une politique.
 - f. Dans le tableau de bord de la console IAM, choisissez Users (Utilisateurs), puis votre utilisateur IAM dans la liste.
 - g. Sur la page Récapitulatif, choisissez Ajouter des autorisations.
 - h. Choisissez Attacher directement les stratégies existantes.
 - i. Dans la liste des politiques, choisissez `keyspacesVPCEndpoint`, puis Next : Review.
 - j. Choisissez Ajouter des autorisations.
4. Pour vérifier que la `system.peers` table Amazon Keyspaces est mise à jour avec les informations VPC, exécutez la requête suivante depuis votre instance Amazon EC2 à l'aide de `cqlsh`. Si vous ne l'avez pas déjà installé `cqlsh` sur votre instance Amazon EC2 à l'étape 2, suivez les instructions fournies dans [the section called "Utilisation de l'`cqlsh`-expansion"](#)

```
SELECT * FROM system.peers;
```

La sortie renvoie des nœuds dotés d'adresses IP IPv6 privées, en fonction de la configuration de votre VPC et de votre sous-réseau dans votre région. AWS

```

peer                               | data_center | host_id
      | preferred_ip                               | rack      | release_version
      | rpc_address                               | schema_version |
tokens
-----+-----
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
      2600:1111:2222:3333:283b:8e6:d04f      | us-east-1 | dddddd-7a22-3582-
a73d-49338a686a53 | 2600:1111:2222:3333:283b:8e6:d04f      | us-east-1 |
      3.11.2 | 2600:1111:2222:3333:283b:8e6:d04f      | 05deae2d-6405-494d-a965-
c0e5836bcb3c | {'85070591730234615865843651857942052863'}

```

```
2600:1111:2222:4444:7d26:5a09:1b44 | us-east-1 | 66666666-035d-37ef-
a247-19a6a867ab09 | 2600:1111:2222:4444:7d26:5a09:1b44 | us-east-1 |
3.11.2 | 2600:1111:2222:4444:7d26:5a09:1b44 | 05deae2d-6405-494d-a965-
c0e5836bcb3c | {'170141183460469231731687303715884105726'}
```

Note

Vous devez utiliser une `cqlsh` connexion à Amazon Keyspaces pour confirmer que le point de terminaison de votre VPC a été correctement configuré. Si vous utilisez votre environnement local ou l'éditeur CQL d'Amazon Keyspaces dans le Console de gestion AWS, la connexion passe automatiquement par le point de terminaison public plutôt que par votre point de terminaison VPC. Si vous voyez neuf adresses IP, il s'agit des entrées qu'Amazon Keyspaces écrit automatiquement dans le `system.peers` tableau pour les connexions aux points de terminaison publics.

Étape 5 : Configuration de la surveillance avec CloudWatch

Cette étape explique comment utiliser Amazon CloudWatch pour surveiller la connexion du point de terminaison VPC à Amazon Keyspaces.

AWS PrivateLink publie des points de données CloudWatch concernant les points de terminaison de votre interface. Vous pouvez utiliser les métriques pour vérifier que le système fonctionne comme prévu. L'espace de `AWS/PrivateLinkEndpoints` noms CloudWatch inclut les métriques pour les points de terminaison de l'interface. Pour plus d'informations, consultez [AWS PrivateLink les CloudWatch statistiques](#) du AWS PrivateLink Guide.

Pour créer un CloudWatch tableau de bord avec des métriques de point de terminaison VPC

1. Ouvrez la CloudWatch console à l'adresse <https://console.aws.amazon.com/cloudwatch/>.
2. Dans le panneau de navigation, choisissez Dashboards (Tableaux de bord). Choisissez ensuite Créer un tableau de bord. Entrez un nom pour le tableau de bord et choisissez Create.
3. Sous Ajouter un widget, choisissez Numéro.
4. Sous Métriques, sélectionnez AWS/PrivateLinkEndpoints.
5. Choisissez le type de point de terminaison, le nom du service, l'ID de point de terminaison VPC, l'ID de VPC.

6. Sélectionnez les métriques `ActiveConnections` et `NewConnections` choisissez `Create Widget`.
7. Enregistrez le tableau de bord.

La `ActiveConnections` métrique est définie comme le nombre de connexions actives simultanées que le point de terminaison a reçues au cours de la dernière minute. La `NewConnections` métrique est définie comme le nombre de nouvelles connexions établies via le point de terminaison au cours de la dernière minute.

Pour plus d'informations sur la création de tableaux de bord, voir [Créer un tableau de bord](#) dans le guide de CloudWatch l'utilisateur.

Étape 6 : (Facultatif) Meilleures pratiques pour configurer la taille du pool de connexions pour votre application

Dans cette section, nous expliquons comment déterminer la taille idéale du pool de connexions en fonction des exigences de débit de requêtes de votre application.

Amazon Keyspaces autorise un maximum de 3 000 requêtes CQL par seconde et par connexion TCP. Il n'y a donc pratiquement aucune limite quant au nombre de connexions qu'un conducteur peut établir avec Amazon Keyspaces. Toutefois, nous vous recommandons d'adapter la taille du pool de connexions aux exigences de votre application et de prendre en compte les points de terminaison disponibles lorsque vous utilisez Amazon Keyspaces avec des connexions de point de terminaison VPC.

Vous configurez la taille du pool de connexions dans le pilote client. Par exemple, sur la base d'un pool local de 2 et d'un point de terminaison d'interface VPC créé dans 3 zones de disponibilité, le pilote établit 6 connexions pour les requêtes (7 au total, y compris une connexion de contrôle). À l'aide de ces 6 connexions, vous pouvez prendre en charge un maximum de 18 000 requêtes CQL par seconde.

Si votre application doit prendre en charge 40 000 requêtes CQL par seconde, revenez au nombre de requêtes nécessaires pour déterminer la taille du pool de connexions requis. Pour prendre en charge 40 000 requêtes CQL par seconde, vous devez configurer la taille du pool local sur au moins 5, ce qui prend en charge un minimum de 45 000 requêtes CQL par seconde.

Vous pouvez contrôler si vous dépassez le quota du nombre maximum d'opérations par seconde et par connexion en utilisant la `PerConnectionRequestRateExceeded` CloudWatch métrique

dans l'espace de AWS/Cassandra noms. La `PerConnectionRequestRateExceeded` métrique indique le nombre de demandes adressées à Amazon Keyspaces qui dépassent le quota de taux de demandes par connexion.

Les exemples de code présentés dans cette étape montrent comment estimer et configurer le regroupement de connexions lorsque vous utilisez des points de terminaison VPC d'interface.

Java

Vous pouvez configurer le nombre de connexions par pool dans le pilote Java. Pour un exemple complet de connexion d'un pilote client Java, consultez [the section called “Utilisation d'un pilote client Java Cassandra”](#).

Lorsque le pilote client est démarré, la connexion de contrôle est d'abord établie pour les tâches administratives, telles que les modifications de schéma et de topologie. Ensuite, les connexions supplémentaires sont créées.

Dans l'exemple suivant, la configuration du pilote de taille de pool local est spécifiée comme 2. Si le point de terminaison du VPC est créé sur 3 sous-réseaux au sein du VPC, cela donne 7 points CloudWatch pour le point de terminaison de l'interface, comme indiqué `NewConnections` dans la formule suivante.

```
NewConnections = 3 (VPC subnet endpoints created across) * 2 (pool size) + 1
( control connection)
```

```
datastax-java-driver {

    basic.contact-points = [ "cassandra.us-east-1.amazonaws.com:9142"]
    advanced.auth-provider{
        class = PlainTextAuthProvider
        username = "ServiceUserName"
        password = "ServicePassword"
    }
    basic.load-balancing-policy {
        local-datacenter = "us-east-1"
        slow-replica-avoidance = false
    }

    advanced.ssl-engine-factory {
        class = DefaultSslEngineFactory
        truststore-path = "./src/main/resources/cassandra_truststore.jks"
        truststore-password = "my_password"
```

```

        hostname-validation = false
    }
    advanced.connection {
        pool.local.size = 2
    }
}

```

Si le nombre de connexions actives ne correspond pas à la taille de pool que vous avez configurée (agrégation entre sous-réseaux) +1 connexion de contrôle, cela signifie que quelque chose empêche la création des connexions.

Node.js

Vous pouvez configurer le nombre de connexions par pool dans le Node.js pilote. Pour un exemple complet de connexion de pilote Node.js client, consultez [the section called “Utilisation d'un pilote Node.js client Cassandra”](#).

L'exemple de code ci-dessous utilise le *keyspaces-bundle.pem* fichier du certificat numérique. Pour plus d'informations, consultez [the section called “Avant de commencer”](#).

Dans l'exemple de code suivant, la configuration du pilote de taille de pool local est spécifiée comme 1. Si le point de terminaison du VPC est créé sur 4 sous-réseaux au sein du VPC, cela donne 5 points CloudWatch pour le point de terminaison de l'interface, comme indiqué `NewConnections` dans la formule suivante.

```

NewConnections = 4 (VPC subnet endpoints created across) * 1 (pool size) + 1
( control connection)

```

```

const cassandra = require('cassandra-driver');
const fs = require('fs');
const types = cassandra.types;
const auth = new cassandra.auth.PlainTextAuthProvider('ServiceUserName',
  'ServicePassword');
const sslOptions1 = {
    ca: [
        fs.readFileSync('/home/ec2-user/keyspaces-bundle.pem',
            'utf-8')],
    host: 'cassandra.us-east-1.amazonaws.com',
    rejectUnauthorized: true
};
const client = new cassandra.Client({
    contactPoints: ['cassandra.us-east-1.amazonaws.com'],

```

```

1 } },
    localDataCenter: 'us-east-1',
    pooling: { coreConnectionsPerHost: { [types.distance.local]:

    consistency: types.consistencies.localQuorum,
    queryOptions: { isIdempotent: true },
    authProvider: auth,
    sslOptions: sslOptions1,
    protocolOptions: { port: 9142 }

});

```

Étape 7 : (Facultatif) Nettoyer

Si vous souhaitez supprimer les ressources que vous avez créées dans ce didacticiel, suivez ces procédures.

Pour supprimer votre point de terminaison VPC pour Amazon Keyspaces

1. Connectez-vous à votre instance Amazon EC2.
2. Déterminez l'ID de point de terminaison VPC utilisé pour Amazon Keyspaces. Si vous omettez les `grep` paramètres, les informations de point de terminaison VPC sont affichées pour tous les services.

```

aws ec2 describe-vpc-endpoint-services | grep ServiceName | grep cassandra

{
  "VpcEndpoint": {
    "PolicyDocument": "{\"Version\":\"2000-00-00\",\"Statement\":[{\"Effect\":\"Allow\", \"Principal\": \"*\", \"Action\": \"*\", \"Resource\": \"*\"}]}\",
    "VpcId": "vpc-a1234bcd",
    "State": "available",
    "ServiceName": "com.amazonaws.us-east-1.cassandra",
    "RouteTableIds": [],
    "VpcEndpointId": "vpce-1a23b4c5",
    "CreationTimestamp": "2025-07-26T22:00:14Z"
  }
}

```

Dans l'exemple de sortie, l'ID du point de terminaison de VPC est *vpce-1a23b4c5*. Assurez-vous de remplacer cette valeur par la vôtre.

3. Supprimez le point de terminaison de VPC.

```
aws ec2 delete-vpc-endpoints --vpc-endpoint-ids vpce-1a23b4c5

{
  "Unsuccessful": []
}
```

Le tableau vide [] indique le succès de l'opération (il n'y a pas eu de demande infructueuse).

Pour résilier votre instance Amazon EC2

1. Ouvrez la console Amazon EC2 à l'adresse <https://console.aws.amazon.com/ec2/>.
2. Dans le panneau de navigation, choisissez Instances.
3. Sélectionnez votre instance Amazon EC2.
4. Choisissez Actions, choisissez Instance State, puis Terminate.
5. Dans la fenêtre de confirmation, choisissez Yes, Terminate (Oui, Résilier).

Tutoriel : Intégrer à Apache Spark pour importer ou exporter des données

Apache Spark est un moteur open source pour l'analyse de données à grande échelle. Apache Spark vous permet d'analyser plus efficacement les données stockées dans Amazon Keyspaces. Vous pouvez également utiliser Amazon Keyspaces pour fournir aux applications un accès en lecture cohérent, à un chiffre et à une milliseconde, aux données d'analyse de Spark. Le connecteur open source Spark Cassandra simplifie la lecture et l'écriture de données entre Amazon Keyspaces et Spark.

La prise en charge par Amazon Keyspaces du connecteur Spark Cassandra rationalise l'exécution des charges de travail Cassandra dans les pipelines d' Spark-based analyse en utilisant un service de base de données entièrement géré et sans serveur. Avec Amazon Keyspaces, vous n'avez pas à craindre que Spark soit en concurrence pour les mêmes ressources d'infrastructure sous-jacentes que vos tables. Les tables Amazon Keyspaces augmentent ou diminuent automatiquement en fonction du trafic de votre application.

Le didacticiel suivant explique les étapes et les meilleures pratiques requises pour lire et écrire des données sur Amazon Keyspaces à l'aide du connecteur Spark Cassandra. Ce didacticiel explique

comment migrer des données vers Amazon Keyspaces en chargeant des données à partir d'un fichier avec le connecteur Spark Cassandra et en les écrivant dans une table Amazon Keyspaces. Le didacticiel explique ensuite comment relire les données d'Amazon Keyspaces à l'aide du connecteur Spark Cassandra. Vous devez procéder ainsi pour exécuter des charges de travail Cassandra dans des pipelines Spark-based d'analyse.

Rubriques

- [Conditions préalables à l'établissement de connexions à Amazon Keyspaces avec le connecteur Spark Cassandra](#)
- [Étape 1 : configurer Amazon Keyspaces pour l'intégration au connecteur Apache Cassandra Spark](#)
- [Étape 2 : Configuration du connecteur Apache Cassandra Spark](#)
- [Étape 3 : Création du fichier de configuration de l'application](#)
- [Étape 4 : préparer les données source et la table cible dans Amazon Keyspaces](#)
- [Étape 5 : Écrire et lire les données Amazon Keyspaces à l'aide du connecteur Apache Cassandra Spark](#)
- [Résolution des erreurs courantes lors de l'utilisation du connecteur Spark Cassandra avec Amazon Keyspaces](#)

Conditions préalables à l'établissement de connexions à Amazon Keyspaces avec le connecteur Spark Cassandra

Avant de vous connecter à Amazon Keyspaces avec le connecteur Spark Cassandra, vous devez vous assurer que vous avez installé les éléments suivants. La compatibilité d'Amazon Keyspaces avec le connecteur Spark Cassandra a été testée avec les versions recommandées suivantes :

- Version 8 de Java
- Scala 2.12
- Spark 3.4
- Cassandra Connector 2.5 et versions supérieures
- pilote Cassandra 4.12

1. Pour installer Scala, suivez les instructions sur <https://www.scala-lang.org/download/scala2.html>.
2. Pour installer Spark 3.4.1, suivez cet exemple.

```
curl -o spark-3.4.1-bin-hadoop3.tgz -k https://d1cdn.apache.org/spark/spark-3.4.1/
spark-3.4.1-bin-hadoop3.tgz

# now to untar
tar -zxvf spark-3.4.1-bin-hadoop3.tgz

# set this variable.
export SPARK_HOME=$PWD/spark-3.4.1-bin-hadoop3
...
```

Étape 1 : configurer Amazon Keyspaces pour l'intégration au connecteur Apache Cassandra Spark

Au cours de cette étape, vous confirmez que le partitionneur de votre compte est compatible avec le connecteur Apache Spark et vous configurez les autorisations IAM requises. Les meilleures pratiques suivantes vous aident à fournir une read/write capacité suffisante pour la table.

1. Vérifiez que le `Murmur3Partitioner` partitionneur est le partitionneur par défaut pour votre compte. Ce partitionneur est compatible avec le connecteur Spark Cassandra. Pour plus d'informations sur les partitionneurs et sur la façon de les modifier, consultez [the section called "Utilisation de partitionneurs"](#)
2. Configurez vos autorisations IAM pour Amazon Keyspaces, à l'aide des points de terminaison VPC de l'interface, avec Apache Spark.
 - Attribuez l' read/write accès à la table utilisateur et l'accès en lecture aux tables système, comme indiqué dans l'exemple de politique IAM ci-dessous.
 - [Les clients accédant à Amazon Keyspaces avec Spark via des points de terminaison VPC d'interface VPC doivent remplir le tableau `system.peers`.](#)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cassandra:Select",
        "cassandra:Modify"
```

```

    ],
    "Resource": [
      "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/
mytable",
      "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
    ]
  },
  {
    "Sid": "ListVPCEndpoints",
    "Effect": "Allow",
    "Action": [
      "ec2:DescribeNetworkInterfaces",
      "ec2:DescribeVpcEndpoints"
    ],
    "Resource": "*"
  }
]
}

```

3. Tenez compte des meilleures pratiques suivantes pour configurer une capacité de read/write débit suffisante pour que votre table Amazon Keyspaces prenne en charge le trafic provenant du connecteur Spark Cassandra.
 - Commencez à utiliser la capacité à la demande pour vous aider à tester le scénario.
 - Pour optimiser le coût du débit des tables pour les environnements de production, utilisez un limiteur de débit pour le trafic provenant du connecteur et configurez votre table pour utiliser la capacité allouée avec un dimensionnement automatique. Pour de plus amples informations, veuillez consulter [the section called “Gérez la capacité de débit avec la mise à l'échelle automatique”](#).
 - Vous pouvez utiliser un limiteur de débit fixe fourni avec le pilote Cassandra. Certains [limiteurs de débit sont adaptés à Amazon Keyspaces](#) dans [AWS le](#) référentiel d'exemples.
 - Pour plus d'informations sur la gestion des capacités, consultez [the section called “Configuration des modes read/write de capacité”](#).

Étape 2 : Configuration du connecteur Apache Cassandra Spark

Apache Spark est une plateforme informatique polyvalente que vous pouvez configurer de différentes manières. Pour configurer Spark et le connecteur Spark Cassandra en vue de leur intégration à Amazon Keyspaces, nous vous recommandons de commencer par les paramètres de configuration

minimaux décrits dans la section suivante, puis de les augmenter ultérieurement en fonction de votre charge de travail.

- Créez des partitions Spark d'une taille inférieure à 8 Mo.

Dans Spark, les partitions représentent un bloc atomique de données qui peut être exécuté en parallèle. Lorsque vous écrivez des données sur Amazon Keyspaces à l'aide du connecteur Spark Cassandra, plus la partition Spark est petite, plus le nombre d'enregistrements que la tâche va écrire est faible. Si une tâche Spark rencontre plusieurs erreurs, elle échoue une fois que le nombre de tentatives indiqué est épuisé. Pour éviter de rejouer des tâches volumineuses et de retraiter un grand nombre de données, limitez la taille de la partition Spark.

- Utilisez un faible nombre d'écritures simultanées par exécuteur avec un grand nombre de tentatives.

Amazon Keyspaces renvoie les erreurs de capacité insuffisante aux pilotes Cassandra en cas d'expiration des délais de fonctionnement. Vous ne pouvez pas résoudre les délais d'attente causés par une capacité insuffisante en modifiant la durée d'expiration configurée, car le connecteur Spark Cassandra tente de réessayer les demandes de manière transparente à l'aide du `MultipleRetryPolicy`. Pour éviter que les nouvelles tentatives ne surchargent le pool de connexions du pilote, utilisez un faible nombre d'écritures simultanées par exécuteur avec un grand nombre de tentatives. L'extrait de code suivant en est un exemple.

```
spark.cassandra.query.retry.count = 500
spark.cassandra.output.concurrent.writes = 3
```

- Décomposez le débit total et répartissez-le sur plusieurs sessions Cassandra.
 - Le connecteur Cassandra Spark crée une session pour chaque exécuteur Spark. Considérez cette session comme l'unité d'échelle permettant de déterminer le débit requis et le nombre de connexions requises.
 - Lorsque vous définissez le nombre de cœurs par exécuteur et le nombre de cœurs par tâche, commencez par un niveau bas et augmentez selon les besoins.
 - Définissez les échecs de tâches Spark pour autoriser le traitement en cas d'erreurs transitoires. Une fois que vous vous êtes familiarisé avec les caractéristiques et les exigences de trafic de votre application, nous vous recommandons `spark.task.maxFailures` de définir une valeur bornée.
 - Par exemple, la configuration suivante peut gérer deux tâches simultanées par exécuteur et par session :

```
spark.executor.instances = configurable -> number of executors for the session.  
spark.executor.cores = 2 -> Number of cores per executor.  
spark.task.cpus = 1 -> Number of cores per task.  
spark.task.maxFailures = -1
```

- Désactivez le traitement par lots.
- Nous vous recommandons de désactiver le traitement par lots pour améliorer les modèles d'accès aléatoire. L'extrait de code suivant en est un exemple.

```
spark.cassandra.output.batch.size.rows = 1 (Default = None)  
spark.cassandra.output.batch.grouping.key = none (Default = Partition)  
spark.cassandra.output.batch.grouping.buffer.size = 100 (Default = 1000)
```

- Réglez **SPARK_LOCAL_DIRS** sur un disque local rapide avec suffisamment d'espace.
- Par défaut, Spark enregistre les fichiers de sortie de carte et les ensembles de données distribués résilients (RDD) dans un /tmp dossier. En fonction de la configuration de votre hôte Spark, cela peut entraîner des erreurs de style qui ne laissent aucun espace sur l'appareil.
- Pour définir la variable d'environnement **SPARK_LOCAL_DIRS** sur un répertoire appelé /example/spark-dir, vous pouvez utiliser la commande suivante.

```
export SPARK_LOCAL_DIRS=/example/spark-dir
```

Étape 3 : Création du fichier de configuration de l'application

Pour utiliser le connecteur open source Spark Cassandra avec Amazon Keyspaces, vous devez fournir un fichier de configuration d'application contenant les paramètres requis pour vous connecter au pilote Java. DataStax Vous pouvez utiliser des informations d'identification spécifiques au service ou le plug-in SigV4 pour vous connecter.

Si ce n'est pas déjà fait, vous devez convertir le certificat numérique utilisé pour créer la connexion TLS en un fichier TrustStore. Vous pouvez suivre les étapes détaillées décrites dans le didacticiel [the section called “Avant de commencer”](#) de connexion au pilote Java. Prenez note du chemin et du mot de passe du fichier TrustStore, car vous avez besoin de ces informations lorsque vous créez le fichier de configuration de l'application.

Connect grâce à l'authentification SigV4

Cette section présente un exemple de `application.conf` fichier que vous pouvez utiliser lors de la connexion à l'aide des AWS informations d'identification et du plug-in SigV4. Si ce n'est pas déjà fait, vous devez générer vos clés d'accès IAM (un identifiant de clé d'accès et une clé d'accès secrète) et les enregistrer dans votre fichier de AWS configuration ou en tant que variables d'environnement. Pour obtenir des instructions complètes, consultez [the section called “Informations d'identification requises pour AWS l'authentification”](#).

Dans l'exemple suivant, remplacez le chemin d'accès à votre fichier TrustStore et remplacez le mot de passe.

```
datastax-java-driver {
  basic.contact-points = ["cassandra.us-east-1.amazonaws.com:9142"]
  basic.load-balancing-policy {
    class = DefaultLoadBalancingPolicy
    local-datacenter = us-east-1
    slow-replica-avoidance = false
  }
  basic.request {
    consistency = LOCAL_QUORUM
  }
  advanced {
    auth-provider = {
      class = software.aws.mcs.auth.SigV4AuthProvider
      aws-region = us-east-1
    }
    ssl-engine-factory {
      class = DefaultSslEngineFactory
      truststore-path = "path_to_file/cassandra_truststore.jks"
      truststore-password = "password"
    }
    hostname-validation=false
  }
  advanced.connection.pool.local.size = 3
}
```

Mettez à jour et enregistrez ce fichier de configuration sous `/home/user1/application.conf`. Les exemples suivants utilisent ce chemin.

Connectez-vous à l'aide d'informations d'identification spécifiques au service

Cette section présente un exemple de `application.conf` fichier que vous pouvez utiliser lorsque vous vous connectez avec des informations d'identification spécifiques au service. Si ce n'est pas déjà fait, vous devez générer des informations d'identification spécifiques au service pour Amazon Keyspaces. Pour obtenir des instructions complètes, consultez [the section called “Créez des informations d'identification spécifiques au service”](#).

Dans l'exemple suivant, remplacez `username` et par vos `password` propres informations d'identification. Remplacez également le chemin du fichier vers votre fichier `TrustStore` et remplacez le mot de passe.

```
datastax-java-driver {
  basic.contact-points = ["cassandra.us-east-1.amazonaws.com:9142"]
  basic.load-balancing-policy {
    class = DefaultLoadBalancingPolicy
    local-datacenter = us-east-1
  }
  basic.request {
    consistency = LOCAL_QUORUM
  }
  advanced {
    auth-provider = {
      class = PlainTextAuthProvider
      username = "username"
      password = "password"
      aws-region = "us-east-1"
    }
    ssl-engine-factory {
      class = DefaultSslEngineFactory
      truststore-path = "path_to_file/cassandra_truststore.jks"
      truststore-password = "password"
      hostname-validation=false
    }
    metadata = {
      schema {
        token-map.enabled = true
      }
    }
  }
}
```

Mettez à jour et enregistrez ce fichier de configuration `/home/user1/application.conf` afin de l'utiliser avec l'exemple de code.

Connect avec un tarif fixe

Pour imposer un taux fixe par exécuteur Spark, vous pouvez définir un limiteur de requêtes. Ce limiteur de demandes limite le taux de demandes par seconde. Le connecteur Spark Cassandra déploie une session Cassandra par exécuteur. L'utilisation de la formule suivante peut vous aider à obtenir un débit constant par rapport à une table.

```
max-request-per-second * numberOfExecutors = total throughput against a table
```

Vous pouvez ajouter cet exemple au fichier de configuration de l'application que vous avez créé précédemment.

```
datastax-java-driver {  
  advanced.throttler {  
    class = RateLimitingRequestThrottler  
  
    max-requests-per-second = 3000  
    max-queue-size = 30000  
    drain-interval = 1 millisecond  
  }  
}
```

Étape 4 : préparer les données source et la table cible dans Amazon Keyspaces

Au cours de cette étape, vous allez créer un fichier source contenant des exemples de données et une table Amazon Keyspaces.

1. Créez le fichier source. Vous pouvez choisir l'une des options suivantes :
 - Pour ce didacticiel, vous utilisez un fichier de valeurs séparées par des virgules (CSV) dont le `keyspaces_sample_table.csv` nom est le fichier source pour la migration des données. Le fichier d'exemple fourni contient quelques lignes de données pour une table portant le nom `book_awards`.

- Téléchargez l'exemple de fichier CSV (`keyspaces_sample_table.csv`) contenu dans le fichier d'archive [samplemigration.zip](#) suivant. Décompressez l'archive et notez le chemin vers `keyspaces_sample_table.csv`.
- Si vous souhaitez utiliser votre propre fichier CSV pour écrire des données sur Amazon Keyspaces, assurez-vous que les données sont aléatoires. Les données lues directement depuis une base de données ou exportées vers des fichiers plats sont généralement classées par partition et clé primaire. L'importation de données commandées dans Amazon Keyspaces peut entraîner leur écriture sur de plus petits segments des partitions Amazon Keyspaces, ce qui entraîne une répartition inégale du trafic. Cela peut entraîner un ralentissement des performances et des taux d'erreur plus élevés.

En revanche, la randomisation des données permet de tirer parti des fonctionnalités d'équilibrage de charge intégrées d'Amazon Keyspaces en répartissant le trafic entre les partitions de manière plus uniforme. Il existe différents outils que vous pouvez utiliser pour randomiser les données. Pour un exemple utilisant l'outil open source [Shuf](#), consultez le didacticiel [the section called “Étape 2 : Préparation des données”](#) de migration de données. L'exemple suivant montre comment mélanger des données sous forme de fichier. DataFrame

```
import org.apache.spark.sql.functions.randval
shuffledDF = dataframe.orderBy(rand())
```

2. Créez le keyspace et le tableau cibles dans Amazon Keyspaces.

- a. Connectez-vous à Amazon Keyspaces à l'aide du `cqlsh-expansion` Pour les instructions d'`cqlsh-expansion` installation, voir [the section called “Utilisation de l’cqlsh-expansion”](#).

Dans l'exemple suivant, remplacez le point de terminaison du service par votre propre valeur.

```
cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 --ssl
```

- b. Créez un nouvel espace de touches avec le `catalog` nom indiqué dans l'exemple suivant.

```
CREATE KEYSPACE catalog WITH REPLICATION = {'class': 'SingleRegionStrategy'};
```

- c. Une fois que le nouveau keyspace est devenu disponible, utilisez le code suivant pour créer la table `book_awards` cible. Pour en savoir plus sur la création de ressources asynchrones et sur la façon de vérifier si une ressource est disponible, consultez [the section called “Vérifier l'état de création de l'espace clé”](#)

```
CREATE TABLE catalog.book_awards (  
  year int,  
  award text,  
  rank int,  
  category text,  
  book_title text,  
  author text,  
  publisher text,  
  PRIMARY KEY ((year, award), category, rank)  
);
```

Étape 5 : Écrire et lire les données Amazon Keyspaces à l'aide du connecteur Apache Cassandra Spark

Au cours de cette étape, vous commencez par charger les données du fichier d'exemple dans un fichier `DataFrame` avec le connecteur Spark Cassandra. Ensuite, vous écrivez les données du `DataFrame` dans votre table Amazon Keyspaces. Vous pouvez également utiliser cette partie indépendamment, par exemple pour migrer des données vers une table Amazon Keyspaces. Enfin, vous pouvez lire les données de votre table dans un `DataFrame` à l'aide du connecteur Spark Cassandra. Vous pouvez également utiliser cette partie indépendamment, par exemple pour lire les données d'une table Amazon Keyspaces afin d'effectuer des analyses de données avec Apache Spark.

1. Démarrez le Spark Shell comme indiqué dans l'exemple suivant. Notez que cet exemple utilise l'authentification SigV4.

```
./spark-shell --files application.conf --conf  
spark.cassandra.connection.config.profile.path=application.conf  
--packages software.aws.mcs:aws-sigv4-auth-cassandra-java-driver-  
plugin:4.0.5,com.datastax.spark:spark-cassandra-connector_2.12:3.1.0 --conf  
spark.sql.extensions=com.datastax.spark.connector.CassandraSparkExtensions
```

2. Importez le connecteur Spark Cassandra avec le code suivant.

```
import org.apache.spark.sql.cassandra._
```

3. Pour lire les données du fichier CSV et les stocker dans un fichier `DataFrame`, vous pouvez utiliser l'exemple de code suivant.

```
var df =
  spark.read.option("header", "true").option("inferSchema", "true").csv("keyspaces_sample_tabl
```

Vous pouvez afficher le résultat à l'aide de la commande suivante.

```
scala> df.show();
```

La sortie doit ressembler à ceci.

```
+-----+-----+-----+-----+-----+-----+
+-----+
|          award|year|  category|rank|          author|          book_title|
|publisher|
+-----+-----+-----+-----+-----+-----+
+-----+
|Kwesi Manu Prize|2020|  Fiction|  1|          Akua Mansa|  Where did you go?|
|SomePublisher|
|Kwesi Manu Prize|2020|  Fiction|  2|          John Stiles|          Yesterday|
|Example Books|
|Kwesi Manu Prize|2020|  Fiction|  3|          Nikki Wolf|Moving to the Cha...|
|AnyPublisher|
|          Wolf|2020|Non-Fiction|  1|          Wang Xiulan|  History of Ideas|
|Example Books|
|          Wolf|2020|Non-Fiction|  2|Ana Carolina Silva|          Science Today|
|SomePublisher|
|          Wolf|2020|Non-Fiction|  3|Shirley Rodriguez|The Future of Sea...|
|AnyPublisher|
|    Richard Roe|2020|  Fiction|  1|Alejandro Rosalez|          Long Summer|
|SomePublisher|
|    Richard Roe|2020|  Fiction|  2|          Arnav Desai|          The Key|
|Example Books|
|    Richard Roe|2020|  Fiction|  3|          Mateo Jackson|  Inside the Whale|
|AnyPublisher|
+-----+-----+-----+-----+-----+-----+
+-----+
```


Vous pouvez confirmer le schéma des données dans le `DataFrame` comme indiqué dans l'exemple suivant.

```
scala> df.printSchema
```

La sortie doit ressembler à ceci.

```
root
|-- award: string (nullable = true)
|-- year: integer (nullable = true)
|-- category: string (nullable = true)
|-- rank: integer (nullable = true)
|-- author: string (nullable = true)
|-- book_title: string (nullable = true)
|-- publisher: string (nullable = true)
```

- Utilisez la commande suivante pour écrire les données dans `DataFrame` le tableau Amazon Keyspaces.

```
df.write.cassandraFormat("book_awards", "catalog").mode("APPEND").save()
```

- Pour confirmer que les données ont été enregistrées, vous pouvez les relire dans une trame de données, comme indiqué dans l'exemple suivant.

```
var newDf = spark.read.cassandraFormat("book_awards", "catalog").load()
```

Vous pouvez ensuite afficher les données qui sont désormais contenues dans le dataframe.

```
scala> newDf.show()
```

Le résultat de cette commande devrait ressembler à ceci.

```
+-----+-----+-----+-----+
+----+----+
|      book_title|      author|      award|  category|
|publisher|rank|year|
+-----+-----+-----+-----+
+----+----+
```

```

|      Long Summer| Alejandro Rosalez|      Richard Roe|      Fiction|
SomePublisher|    1|2020|
|      History of Ideas|      Wang Xiulan|      Wolf|Non-Fiction|Example
Books|    1|2020|
|      Where did you go?|      Akua Mansa|Kwesi Manu Prize|      Fiction|
SomePublisher|    1|2020|
|      Inside the Whale|      Mateo Jackson|      Richard Roe|      Fiction|
AnyPublisher|    3|2020|
|      Yesterday|      John Stiles|Kwesi Manu Prize|      Fiction|Example
Books|    2|2020|
|Moving to the Cha...|      Nikki Wolf|Kwesi Manu Prize|      Fiction|
AnyPublisher|    3|2020|
|The Future of Sea...| Shirley Rodriguez|      Wolf|Non-Fiction|
AnyPublisher|    3|2020|
|      Science Today|Ana Carolina Silva|      Wolf|Non-Fiction|
SomePublisher|    2|2020|
|      The Key|      Arnav Desai|      Richard Roe|      Fiction|Example
Books|    2|2020|
+-----+-----+-----+-----+
+----+----+

```

Résolution des erreurs courantes lors de l'utilisation du connecteur Spark Cassandra avec Amazon Keyspaces

Si vous utilisez Amazon Virtual Private Cloud et que vous vous connectez à Amazon Keyspaces, les erreurs les plus courantes rencontrées lors de l'utilisation du connecteur Spark sont dues aux problèmes de configuration suivants.

- L'utilisateur ou le rôle IAM utilisé dans le VPC ne dispose pas des autorisations requises pour accéder `system.peers` à la table dans Amazon Keyspaces. Pour de plus amples informations, veuillez consulter [the section called “Remplissage des entrées de `system.peers` table avec les informations de point de terminaison VPC de l'interface”](#).
- L'utilisateur ou le rôle IAM ne dispose pas des read/write autorisations requises pour accéder à la table des utilisateurs et ne dispose pas de l'accès en lecture aux tables système dans Amazon Keyspaces. Pour de plus amples informations, veuillez consulter [the section called “Étape 1 : Configuration d'Amazon Keyspaces”](#).
- La configuration du pilote Java ne désactive pas la vérification du nom d'hôte lors de la création de la SSL/TLS connexion. Pour obtenir des exemples, consultez [the section called “Étape 2 : Configuration du pilote”](#).

Pour connaître les étapes détaillées de résolution des problèmes de connexion, voir [the section called “Erreurs de connexion au point de terminaison VPC”](#).

En outre, vous pouvez utiliser CloudWatch les métriques Amazon pour résoudre les problèmes liés à la configuration de votre connecteur Spark Cassandra dans Amazon Keyspaces. Pour en savoir plus sur l'utilisation d'Amazon Keyspaces avec CloudWatch, consultez. [the section called “Surveillance avec CloudWatch”](#)

La section suivante décrit les métriques les plus utiles à observer lorsque vous utilisez le connecteur Spark Cassandra.

PerConnectionRequestRateExceeded

Amazon Keyspaces a un quota de 3 000 demandes par seconde et par connexion. Chaque exécuteur Spark établit une connexion avec Amazon Keyspaces. L'exécution de plusieurs tentatives peut épuiser votre quota de débit de demandes par connexion. Si vous dépassez ce quota, Amazon Keyspaces émet une `PerConnectionRequestRateExceeded` métrique. CloudWatch

Si vous `PerConnectionRequestRateExceeded` constatez la présence d'événements ainsi que d'autres erreurs système ou utilisateur, il est probable que Spark exécute plusieurs tentatives au-delà du nombre de demandes alloué par connexion.

Si vous voyez `PerConnectionRequestRateExceeded` des événements sans autre erreur, vous devrez peut-être augmenter le nombre de connexions dans les paramètres de votre pilote pour augmenter le débit, ou vous devrez peut-être augmenter le nombre d'exécuteurs dans votre tâche Spark.

StoragePartitionThroughputCapacityExceeded

Amazon Keyspaces dispose d'un quota de 1 000 WCU ou WRU par second/3 000 RCU ou RRU par seconde et par partition. Si vous observez `StoragePartitionThroughputCapacityExceeded` CloudWatch des événements, cela peut indiquer que les données ne sont pas randomisées lors du chargement. Pour obtenir des exemples de brassage de données, consultez. [the section called “Étape 4 : Préparation des données source et de la table cible”](#)

Erreurs et avertissements courants

Si vous utilisez Amazon Virtual Private Cloud et que vous vous connectez à Amazon Keyspaces, le pilote Cassandra peut émettre un message d'avertissement concernant le nœud de contrôle lui-même dans le tableau `system.peers`. Pour de plus amples informations, veuillez consulter [the section called “Erreurs et avertissements courants”](#). Vous pouvez ignorer cet avertissement en toute sécurité.

Tutoriel : Connectez-vous depuis une application conteneurisée hébergée sur Amazon Elastic Kubernetes Service

Ce didacticiel explique les étapes nécessaires à la configuration d'un cluster Amazon Elastic Kubernetes Service (Amazon EKS) afin d'héberger une application conteneurisée qui se connecte à Amazon Keyspaces à l'aide de l'authentification Sigv4.

Amazon EKS est un service géré qui élimine le besoin d'installer, d'exploiter et de maintenir votre propre plan de contrôle Kubernetes. [Kubernetes](#) est un système open source qui automatise la gestion, la mise à l'échelle et le déploiement d'applications conteneurisées.

Ce didacticiel fournit des instructions détaillées pour configurer, créer et déployer une application Java conteneurisée sur Amazon EKS. Dans la dernière étape, vous exécutez l'application pour écrire des données dans une table Amazon Keyspaces.

Rubriques

- [Conditions requises pour se connecter d'Amazon EKS à Amazon Keyspaces](#)
- [Étape 1 : configurer le cluster Amazon EKS et configurer les autorisations IAM](#)
- [Étape 2 : Configuration de l'application](#)
- [Étape 3 : créer l'image de l'application et télécharger le fichier Docker dans votre référentiel Amazon ECR](#)
- [Étape 4 : Déployer l'application sur Amazon EKS et écrire des données dans votre table](#)
- [Étape 5 : Nettoyage \(facultatif\)](#)

Conditions requises pour se connecter d'Amazon EKS à Amazon Keyspaces

Créez les AWS ressources avant de commencer le didacticiel

1. Avant de commencer ce didacticiel, suivez les instructions de AWS configuration indiquées dans [Accès à Amazon Keyspaces \(pour Apache Cassandra\)](#). Ces étapes incluent l'inscription AWS et la création d'un Gestion des identités et des accès AWS (IAM) principal ayant accès à Amazon Keyspaces.
2. Créez un espace de touches Amazon Keyspaces avec le nom `aws` et une table avec le nom `user` lequel vous pouvez écrire depuis l'application conteneurisée exécutée dans Amazon EKS plus loin dans ce didacticiel. Vous pouvez le faire avec AWS CLI ou en utilisant `cqlsh`.

AWS CLI

```
aws keyspaces create-keyspace --keyspace-name 'aws'
```

Pour confirmer que le keyspace a été créé, vous pouvez utiliser la commande suivante.

```
aws keyspaces list-keyspaces
```

Pour créer la table, vous pouvez utiliser la commande suivante.

```
aws keyspaces create-table --keyspace-name 'aws' --table-name 'user' --schema-definition 'allColumns=[
    {name=username,type=text}, {name=fname,type=text},
    {name=last_update_date,type=timestamp},{name=lname,type=text}]],
    partitionKeys=[{name=username}]'
```

Pour confirmer que votre table a été créée, vous pouvez utiliser la commande suivante.

```
aws keyspaces list-tables --keyspace-name 'aws'
```

Pour plus d'informations, voir [créer un espace de touches](#) et [créer une table](#) dans la référence des AWS CLI commandes.

cqlsh

```
CREATE KEYSPACE aws WITH replication = {'class': 'SimpleStrategy',
'replication_factor': '3'} AND durable_writes = true;
CREATE TABLE aws.user (
    username text PRIMARY KEY,
    fname text,
    last_update_date timestamp,
    lname text
);
```

Pour vérifier que votre table a été créée, vous pouvez utiliser l'instruction suivante.

```
SELECT * FROM system_schema.tables WHERE keyspace_name = "aws";
```

Votre table doit être répertoriée dans le résultat de cette instruction. Notez qu'il peut y avoir un délai avant que la table ne soit créée. Pour plus d'informations, consultez [the section called "CREATE TABLE"](#).

3. Créez un cluster Amazon EKS avec un nœud de type Fargate - Linux. Fargate est un moteur de calcul sans serveur qui vous permet de déployer des pods Kubernetes sans gérer les instances Amazon Amazon EC2. Pour suivre ce didacticiel sans avoir à mettre à jour le nom du cluster dans tous les exemples de commandes, créez un cluster portant ce nom en `my-eks-cluster` suivant les instructions de la section [Getting started with Amazon EKS](#), `eksctl` dans le guide de l'utilisateur Amazon EKS. Lorsque votre cluster est créé, vérifiez que vos nœuds et les deux pods par défaut fonctionnent et fonctionnent correctement. Vous pouvez le faire à l'aide de la commande suivante.

```
kubectl get pods -A -o wide
```

Vous devriez voir quelque chose de similaire à cette sortie.

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE	IP
NODE				NOMINATED	NODE	READINESS
GATES						
kube-system	coredns-1234567890-abcde	1/1	Running	0	18m	
192.0.2.0	fargate-ip-192-0-2-0.region-code.compute.internal				<none>	
<none>						

```
kube-system    coredns-1234567890-12345    1/1    Running    0    18m
192.0.2.1    fargate-ip-192-0-2-1.region-code.compute.internal    <none>
<none>
```

4. Installez Docker. Pour savoir comment installer Docker sur une instance Amazon EC2, [consultez la section Installer](#) Docker dans le guide de l'utilisateur d'Amazon Elastic Container Registry.

Docker est disponible pour plusieurs systèmes d'exploitation, notamment les distributions Linux les plus modernes, comme Ubuntu et même MacOS et Windows. Pour en savoir plus sur la façon d'installer Docker sur votre système d'exploitation, consultez le [guide d'installation Docker](#).

5. créer un référentiel Amazon ECR ; Amazon ECR est un service de registre d'images de conteneurs AWS géré que vous pouvez utiliser avec votre CLI préférée pour envoyer, extraire et gérer des images Docker. Pour plus d'informations sur les référentiels Amazon ECR, consultez le guide de l'[utilisateur d'Amazon Elastic Container Registry](#). Vous pouvez utiliser la commande suivante pour créer un référentiel portant ce nommy-ecr-repository.

```
aws ecr create-repository --repository-name my-ecr-repository
```

Après avoir effectué les étapes préalables, passez à [the section called “Étape 1 : Configuration du cluster Amazon EKS”](#).

Étape 1 : configurer le cluster Amazon EKS et configurer les autorisations IAM

Configurez le cluster Amazon EKS et créez les ressources IAM nécessaires pour permettre à un compte de service Amazon EKS de se connecter à votre table Amazon Keyspaces

1. Créez un fournisseur Open ID Connect (OIDC) pour le cluster Amazon EKS. Cela est nécessaire pour utiliser les rôles IAM pour les comptes de service. Pour plus d'informations sur les fournisseurs OIDC et sur la façon de les créer, consultez la section [Création d'un fournisseur OIDC IAM pour votre cluster](#) dans le guide de l'utilisateur Amazon EKS.
 - a. Créez votre fournisseur d'identité OIDC IAM pour votre cluster avec la commande suivante. Cet exemple suppose que le nom de votre cluster estmy-eks-cluster. Si vous avez un cluster portant un nom différent, n'oubliez pas de le mettre à jour dans toutes les commandes futures.

```
eksctl utils associate-iam-oidc-provider --cluster my-eks-cluster --approve
```

- b. Vérifiez que le fournisseur d'identité OIDC a été enregistré auprès d'IAM à l'aide de la commande suivante.

```
aws iam list-open-id-connect-providers --region us-east-1
```

La sortie doit ressembler à ceci. Prenez note du nom de ressource Amazon (ARN) de l'OIDC, vous en aurez besoin à l'étape suivante lorsque vous créerez une politique de confiance pour le compte de service.

```
{
  "OpenIDConnectProviderList": [
    ..
    {
      "Arn": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
    }
  ]
}
```

2. Créez un compte de service pour le cluster Amazon EKS. Les comptes de service fournissent une identité pour les processus exécutés dans un pod. Un Pod est l'objet Kubernetes le plus petit et le plus simple que vous puissiez utiliser pour déployer une application conteneurisée. Créez ensuite un rôle IAM que le compte de service peut assumer pour obtenir des autorisations d'accès aux ressources. Vous pouvez accéder à n'importe quel AWS service à partir d'un Pod configuré pour utiliser un compte de service pouvant assumer un rôle IAM avec des autorisations d'accès à ce service.
 - a. Créez un nouvel espace de noms pour le compte de service. Un espace de noms permet d'isoler les ressources du cluster créées pour ce didacticiel. Vous pouvez créer un nouvel espace de noms à l'aide de la commande suivante.

```
kubectl create namespace my-eks-namespace
```

- b. Pour utiliser un espace de noms personnalisé, vous devez l'associer à un profil Fargate. Le code suivant en est un exemple.

```
eksctl create fargateprofile \
```



```
--cluster my-eks-cluster \  
--name my-fargate-profile \  
--namespace my-eks-namespace \  
--labels *=*
```

- c. Créez un compte de service dont le nom figure `my-eks-serviceaccount` dans l'espace de noms `my-eks-namespace` de votre cluster Amazon EKS à l'aide de la commande suivante.

```
cat >my-serviceaccount.yaml <<EOF  
apiVersion: v1  
kind: ServiceAccount  
metadata:  
  name: my-eks-serviceaccount  
  namespace: my-eks-namespace  
EOF  
kubectl apply -f my-serviceaccount.yaml
```

- d. Exécutez la commande suivante pour créer un fichier de politique de confiance qui indique au rôle IAM de faire confiance à votre compte de service. Cette relation de confiance est requise avant qu'un mandant puisse assumer un rôle. Vous devez apporter les modifications suivantes au fichier :
- Pour `lePrincipal`, entrez l'ARN renvoyé par IAM à la `list-open-id-connect-providers` commande. L'ARN contient votre numéro de compte et votre région.
 - Dans l'`conditioninstruction`, remplacez le Région AWS et l'identifiant OIDC.
 - Vérifiez que le nom du compte de service et l'espace de noms sont corrects.

Vous devez joindre le fichier de politique de confiance à l'étape suivante lorsque vous créez le rôle IAM.

```
cat >trust-relationship.json <<EOF  
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Principal": {  
        "Federated": "arn:aws:iam::111122223333:oidc-provider/  
oidc.eks.us-east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
```

```

    },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
      "StringEquals": {
        "oidc.eks.us-east-1.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:my-eks-
namespace:my-eks-serviceaccount",
        "oidc.eks.us-east-1.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
      }
    }
  ]
}
EOF

```

Facultatif : vous pouvez également ajouter plusieurs entrées dans les `StringLike` conditions `StringEquals` ou pour permettre à plusieurs comptes de service ou espaces de noms d'assumer le rôle. Pour permettre à votre compte de service d'assumer un rôle IAM dans un autre AWS compte, consultez les [autorisations Cross-account IAM](#) dans le guide de l'utilisateur Amazon EKS.

3. Créez un rôle IAM avec le nom `my-iam-role` que le compte de service Amazon EKS doit assumer. Joignez le fichier de politique de confiance créé lors de la dernière étape au rôle. La politique de confiance spécifie le compte de service et le fournisseur OIDC auxquels le rôle IAM peut faire confiance.

```
aws iam create-role --role-name my-iam-role --assume-role-policy-document file://
trust-relationship.json --description "EKS service account role"
```

4. Attribuez les autorisations du rôle IAM à Amazon Keyspaces en joignant une politique d'accès.
 - a. Joignez une politique d'accès pour définir les actions que le rôle IAM peut effectuer sur des ressources Amazon Keyspaces spécifiques. Pour ce didacticiel, nous utilisons la politique AWS gérée `AmazonKeyspacesFullAccess`, car notre application va écrire des données dans votre table Amazon Keyspaces. À titre de bonne pratique, il est toutefois recommandé de créer des politiques d'accès personnalisées qui mettent en œuvre le principe du moindre privilège. Pour plus d'informations, consultez [the section called "Comment Amazon Keyspaces fonctionne avec IAM"](#).

```
aws iam attach-role-policy --role-name my-iam-role --policy-arn=arn:aws:iam::aws:policy/AmazonKeyspacesFullAccess
```

Vérifiez que la politique a été correctement attachée au rôle IAM à l'aide de l'instruction suivante.

```
aws iam list-attached-role-policies --role-name my-iam-role
```

La sortie doit ressembler à ceci.

```
{
  "AttachedPolicies": [
    {
      "PolicyName": "AmazonKeyspacesFullAccess",
      "PolicyArn": "arn:aws:iam::aws:policy/AmazonKeyspacesFullAccess"
    }
  ]
}
```

- b. Annotez le compte de service avec l'Amazon Resource Name (ARN) du rôle IAM qu'il peut assumer. Assurez-vous de mettre à jour l'ARN du rôle avec votre identifiant de compte.

```
kubectl annotate serviceaccount -n my-eks-namespace my-eks-serviceaccount eks.amazonaws.com/role-arn=arn:aws:iam::111122223333:role/my-iam-role
```

5. Vérifiez que le rôle IAM et le compte de service sont correctement configurés.

- a. Vérifiez que la politique de confiance du rôle IAM est correctement configurée à l'aide de l'instruction suivante.

```
aws iam get-role --role-name my-iam-role --query Role.AssumeRolePolicyDocument
```

La sortie doit ressembler à ceci.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```

    "Principal": {
      "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.us-east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
    },
    "Action": "sts:AssumeRoleWithWebIdentity",
    "Condition": {
      "StringEquals": {
        "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
        "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:my-eks-
namespace:my-eks-serviceaccount"
      }
    }
  }
]
}

```

- b. Vérifiez que le compte de service Amazon EKS est annoté avec le rôle IAM.

```
kubectl describe serviceaccount my-eks-serviceaccount -n my-eks-namespace
```

La sortie doit ressembler à ceci.

```

Name: my-eks-serviceaccount
Namespace: my-eks-namespace
Labels: <none>
Annotations: eks.amazonaws.com/role-arn: arn:aws:iam::111122223333:role/my-iam-
role
Image pull secrets: <none>
Mountable secrets: <none>
Tokens: <none>
[...]

```

Après avoir créé le compte de service Amazon EKS, le rôle IAM et configuré les relations et les autorisations requises, passez à [the section called “Étape 2 : Configuration de l'application”](#).

Étape 2 : Configuration de l'application

Au cours de cette étape, vous allez créer votre application qui se connecte à Amazon Keyspaces à l'aide du plugin SigV4. [Vous pouvez consulter et télécharger l'exemple d'application Java depuis](#)

[le dépôt d'exemples de code Amazon Keyspaces sur Github](#). Vous pouvez également suivre le processus en utilisant votre propre application, en vous assurant de terminer toutes les étapes de configuration.

Configurez votre application et ajoutez les dépendances requises.

1. Vous pouvez télécharger l'exemple d'application Java en clonant le référentiel Github à l'aide de la commande suivante.

```
git clone https://github.com/aws-samples/amazon-keyspaces-examples.git
```

2. Après avoir téléchargé le dépôt Github, décompressez le fichier téléchargé et accédez au ressource répertoire du fichier. `application.conf`
 - a. Configuration de l'application

Dans cette étape, vous configurez le plugin d'authentification SigV4. Vous pouvez utiliser l'exemple suivant dans votre application. Si ce n'est pas déjà fait, vous devez générer vos clés d'accès IAM (un identifiant de clé d'accès et une clé d'accès secrète) et les enregistrer dans votre fichier de AWS configuration ou en tant que variables d'environnement. Pour obtenir des instructions complètes, consultez [the section called "Informations d'identification requises pour AWS l'authentification"](#). Mettez à jour la AWS région et le point de terminaison du service pour Amazon Keyspaces selon les besoins. Pour plus de points de terminaison de service, voir [the section called "Points de terminaison de service"](#). Remplacez l'emplacement du truststore, le nom du truststore et le mot de passe du truststore par les vôtres.

```
datastax-java-driver {  
  basic.contact-points = ["cassandra.us-east-1.amazonaws.com:9142"]  
  basic.load-balancing-policy.local-datacenter = "us-east-1"  
  advanced.auth-provider {  
    class = software.aws.mcs.auth.SigV4AuthProvider  
    aws-region = "us-east-1"  
  }  
  advanced.ssl-engine-factory {  
    class = DefaultSslEngineFactory  
    truststore-path = "truststore_locationtruststore_name.jks"  
    truststore-password = "truststore_password;"  
  }  
}
```

b. Ajoutez la dépendance du module STS.

Cela permet d'utiliser un `WebIdentityTokenCredentialsProvider` qui renvoie les AWS informations d'identification que l'application doit fournir pour que le compte de service puisse assumer le rôle IAM. Vous pouvez le faire en vous basant sur l'exemple suivant.

```
<dependency>
  <groupId>com.amazonaws</groupId>
  <artifactId>aws-java-sdk-sts</artifactId>
  <version>1.11.717</version>
</dependency>
```

c. Ajoutez la dépendance SigV4.

Ce package implémente le plugin d'authentification SigV4 qui est nécessaire pour s'authentifier auprès d'Amazon Keyspaces

```
<dependency>
  <groupId>software.aws.mcs</groupId>
  <artifactId>aws-sigv4-auth-cassandra-java-driver-plugin</
artifactId>
  <version>4.0.3</version>
</dependency>
```

3. Ajoutez une dépendance de journalisation.

Sans journaux, il est impossible de résoudre les problèmes de connexion. Dans ce didacticiel, nous l'utilisons `slf4j` comme framework de journalisation et nous l'utilisons `logback.xml` pour stocker la sortie du journal. Nous avons défini le niveau de journalisation `debug` sur pour établir la connexion. Vous pouvez utiliser l'exemple suivant pour ajouter la dépendance.

```
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-api</artifactId>
  <version>2.0.5</version>
</dependency>
```

Vous pouvez utiliser l'extrait de code suivant pour configurer la journalisation.

```
<configuration>
  <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
```

```
<encoder>
  <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</
pattern>
</encoder>
</appender>

<root level="debug">
  <appender-ref ref="STDOUT" />
</root>
</configuration>
```

Note

Ce debug niveau est nécessaire pour étudier les défaillances de connexion. Une fois que vous vous êtes connecté avec succès à Amazon Keyspaces depuis votre application, vous pouvez modifier le niveau de journalisation en fonction de vos besoins `warning`.

Étape 3 : créer l'image de l'application et télécharger le fichier Docker dans votre référentiel Amazon ECR

Au cours de cette étape, vous compilez l'exemple d'application, vous créez une image Docker et vous envoyez l'image dans votre référentiel Amazon ECR.

Créez votre application, créez une image Docker et soumettez-la à Amazon Elastic Container Registry

1. Définissez des variables d'environnement pour la version qui définissent votre Région AWS. Remplacez les régions dans les exemples par les vôtres.

```
export CASSANDRA_HOST=cassandra.us-east-1.amazonaws.com:9142
export CASSANDRA_DC=us-east-1
```

2. Compilez votre application avec Apache Maven version 3.6.3 ou supérieure à l'aide de la commande suivante.

```
mvn clean install
```

Cela crée un JAR fichier avec toutes les dépendances incluses dans le target répertoire.

3. Récupérez l'URI de votre référentiel ECR nécessaire pour l'étape suivante à l'aide de la commande suivante. Assurez-vous de mettre à jour la région avec celle que vous avez utilisée.

```
aws ecr describe-repositories --region us-east-1
```

La sortie doit ressembler à celle de l'exemple suivant.

```
"repositories": [  
  {  
    "repositoryArn": "arn:aws:ecr:us-east-1:111122223333:repository/my-ecr-  
repository",  
    "registryId": "111122223333",  
    "repositoryName": "my-ecr-repository",  
    "repositoryUri": "111122223333.dkr.ecr.us-east-1.amazonaws.com/my-ecr-repository",  
    "createdAt": "2023-11-02T03:46:34+00:00",  
    "imageTagMutability": "MUTABLE",  
    "imageScanningConfiguration": {  
      "scanOnPush": false  
    },  
    "encryptionConfiguration": {  
      "encryptionType": "AES256"  
    }  
  },  
]
```

4. À partir du répertoire racine de l'application, créez l'image Docker en utilisant l'URI du référentiel indiqué à la dernière étape. Modifiez le fichier Docker selon vos besoins. Dans la commande build, assurez-vous de remplacer votre identifiant de compte et de Région AWS définir la région dans laquelle se trouve le référentiel my-ecr-repository Amazon ECR.

```
docker build -t 111122223333.dkr.ecr.us-east-1.amazonaws.com/my-ecr-  
repository:latest .
```

5. Récupérez un jeton d'authentification pour envoyer l'image Docker vers Amazon ECR. Vous pouvez le faire à l'aide de la commande suivante.

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --  
password-stdin 111122223333.dkr.ecr.us-east-1.amazonaws.com
```


6. Tout d'abord, vérifiez les images existantes dans votre référentiel Amazon ECR. Vous pouvez utiliser la commande suivante :

```
aws ecr describe-images --repository-name my-ecr-repository --region us-east-1
```

Ensuite, placez l'image Docker dans le dépôt. Vous pouvez utiliser la commande suivante :

```
docker push 111122223333.dkr.ecr.us-east-1.amazonaws.com/my-ecr-repository:latest
```

Étape 4 : Déployer l'application sur Amazon EKS et écrire des données dans votre table

Au cours de cette étape du didacticiel, vous configurez le déploiement d'Amazon EKS pour votre application, puis vous confirmez que l'application est en cours d'exécution et qu'elle peut se connecter à Amazon Keyspaces.

Pour déployer une application sur Amazon EKS, vous devez configurer tous les paramètres pertinents dans un fichier appelé `deployment.yaml`. Ce fichier est ensuite utilisé par Amazon EKS pour déployer l'application. Les métadonnées du fichier doivent contenir les informations suivantes :

- Nom de l'application : nom de l'application. Pour ce tutoriel, nous utilisons `my-keyspaces-app`.
- Espace de noms Kubernetes : espace de noms du cluster Amazon EKS. Pour ce tutoriel, nous utilisons `my-eks-namespace`.
- Le nom du compte de service Amazon EKS est le nom du compte de service Amazon EKS. Pour ce tutoriel, nous utilisons `my-eks-serviceaccount`.
- nom de l'image : nom de l'image de l'application. Pour ce tutoriel, nous utilisons `my-keyspaces-app`.
- URI de l'image : URI de l'image Docker provenant d'Amazon ECR.
- AWS identifiant de compte : votre identifiant de AWS compte.
- ARN du rôle IAM : ARN du rôle IAM créé pour que le compte de service puisse l'assumer. Pour ce tutoriel, nous utilisons `my-iam-role`.
- Région AWS du cluster Amazon EKS dans lequel Région AWS vous avez créé votre cluster Amazon EKS.

Au cours de cette étape, vous déployez et exécutez l'application qui se connecte à Amazon Keyspaces et écrit des données dans la table.

1. Configurez le fichier `deployment.yaml`. Vous devez remplacer les valeurs suivantes :

- `name`
- `namespace`
- `serviceName`
- `image`
- `AWS_ROLE_ARN` value
- Le Région AWS in `CASSANDRA_HOST`
- `AWS_REGION`

Vous pouvez utiliser le fichier suivant comme exemple.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-keyspaces-app
  namespace: my-eks-namespace
spec:
  replicas: 1
  selector:
    matchLabels:
      app: my-keyspaces-app
  template:
    metadata:
      labels:
        app: my-keyspaces-app
    spec:
      serviceName: my-eks-serviceaccount
      containers:
        - name: my-keyspaces-app
          image: 111122223333.dkr.ecr.us-east-1.amazonaws.com/my-ecr-repository:latest
          ports:
            - containerPort: 8080
          env:
            - name: CASSANDRA_HOST
              value: "cassandra.us-east-1.amazonaws.com:9142"
```

```

- name: CASSANDRA_DC
  value: "us-east-1"
- name: AWS_WEB_IDENTITY_TOKEN_FILE
  value: /var/run/secrets/eks.amazonaws.com/serviceaccount/token
- name: AWS_ROLE_ARN
  value: "arn:aws:iam::111122223333:role/my-iam-role"
- name: AWS_REGION
  value: "us-east-1"

```

2. Déployez `deployment.yaml`.

```
kubectl apply -f deployment.yaml
```

La sortie doit ressembler à ceci.

```
deployment.apps/my-keyspaces-app created
```

3. Vérifiez l'état du Pod dans votre espace de noms du cluster Amazon EKS.

```
kubectl get pods -n my-eks-namespace
```

La sortie doit ressembler à cet exemple.

```

NAME                                READY STATUS RESTARTS AGE
my-keyspaces-app-123abcde4f-g5hij  1/1   Running 0    75s

```

Pour plus de détails, vous pouvez utiliser la commande suivante.

```
kubectl describe pod my-keyspaces-app-123abcde4f-g5hij -n my-eks-namespace
```

```

Name:                                my-keyspaces-app-123abcde4f-g5hij
Namespace:                           my-eks-namespace
Priority:                             2000001000
Priority Class Name:                   system-node-critical
Service Account:                      my-eks-serviceaccount
Node:                                 fargate-ip-192-168-102-209.ec2.internal/192.168.102.209
Start Time:                           Thu, 23 Nov 2023 12:15:43 +0000
Labels:                               app=my-keyspaces-app
                                       eks.amazonaws.com/fargate-profile=my-fargate-profile

```

```

pod-template-hash=6c56fccc56
Annotations:      CapacityProvisioned: 0.25vCPU 0.5GB
                  Logging: LoggingDisabled: LOGGING_CONFIGMAP_NOT_FOUND
Status:           Running
IP:               192.168.102.209
IPs:
  IP:             192.168.102.209
Controlled By:    ReplicaSet/my-keyspaces-app-6c56fccc56
Containers:
  my-keyspaces-app:
    Container ID:   containerd://41ff7811d33ae4bc398755800abdc132335d51d74f218ba81da0700a6f8c67b
    Image:          111122223333.dkr.ecr.us-east-1.amazonaws.com/
  my_eks_repository:latest
    Image ID:       111122223333.dkr.ecr.us-east-1.amazonaws.com/
  my_eks_repository@sha256:fd3c6430fc5251661efce99741c72c1b4b03061474940200d0524b84a951439c
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Thu, 23 Nov 2023 12:15:19 +0000
      Finished:     Thu, 23 Nov 2023 12:16:17 +0000
    Ready:          True
    Restart Count:  1
    Environment:
      CASSANDRA_HOST:      cassandra.us-east-1.amazonaws.com:9142
      CASSANDRA_DC:        us-east-1
      AWS_WEB_IDENTITY_TOKEN_FILE: /var/run/secrets/eks.amazonaws.com/
  serviceaccount/token
    AWS_ROLE_ARN:         arn:aws:iam::111122223333:role/my-iam-role
    AWS_REGION:           us-east-1
    AWS_STS_REGIONAL_ENDPOINTS: regional
    Mounts:
      /var/run/secrets/eks.amazonaws.com/serviceaccount from aws-iam-token (ro)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-fssbf (ro)
Conditions:
  Type            Status
  Initialized      True
  Ready           True
  ContainersReady True
  PodScheduled    True
Volumes:
  aws-iam-token:
    Type:          Projected (a volume that contains injected data from
multiple sources)

```

```

    TokenExpirationSeconds: 86400
    kube-api-access-fssbf:
      Type: Projected (a volume that contains injected data from
multiple sources)
      TokenExpirationSeconds: 3607
      ConfigMapName: kube-root-ca.crt
      ConfigMapOptional: <nil>
      DownwardAPI: true
    QoS Class: BestEffort
    Node-Selectors: <none>
    Tolerations: node.kubernetes.io/not-ready:NoExecute op=Exists for
300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for
300s
Events:
  Type    Reason             Age          From          Message
  ----    -
Warning   LoggingDisabled    2m13s        fargate-scheduler Disabled logging
because aws-logging configmap was not found. configmap "aws-logging" not found
Normal    Scheduled          89s          fargate-scheduler Successfully
assigned my-eks-namespace/my-keyspaces-app-6c56fccc56-mgs2m to fargate-
ip-192-168-102-209.ec2.internal
Normal    Pulled             75s          kubelet       Successfully pulled image "111122223333.dkr.ecr.us-east-1.amazonaws.com/
my_eks_repository:latest" in 13.027s (13.027s including waiting)
Normal    Pulling            54s (x2 over 88s) kubelet       Pulling image
"111122223333.dkr.ecr.us-east-1.amazonaws.com/my_eks_repository:latest"
Normal    Created            54s (x2 over 75s) kubelet       Created container
my-keyspaces-app
Normal    Pulled             54s          kubelet       Successfully pulled image "111122223333.dkr.ecr.us-east-1.amazonaws.com/
my_eks_repository:latest" in 222ms (222ms including waiting)
Normal    Started            53s (x2 over 75s) kubelet       Started container
my-keyspaces-app

```

4. Consultez les journaux du Pod pour vérifier que votre application est en cours d'exécution et qu'elle peut se connecter à votre table Amazon Keyspaces. Vous pouvez le faire à l'aide de la commande suivante. Assurez-vous de remplacer le nom de votre déploiement.

```
kubectl logs -f my-keyspaces-app-123abcde4f-g5hij -n my-eks-namespace
```

Vous devriez pouvoir voir les entrées du journal de l'application confirmant la connexion à Amazon Keyspaces, comme dans l'exemple ci-dessous.

```

2:47:20.553 [s0-admin-0] DEBUG c.d.o.d.i.c.metadata.MetadataManager
- [s0] Adding initial contact points [Node(endPoint=cassandra.us-
east-1.amazonaws.com/1.222.333.44:9142, hostId=null, hashCode=e750d92)]
22:47:20.562 [s0-admin-1] DEBUG c.d.o.d.i.c.c.ControlConnection - [s0] Initializing
with event types [SCHEMA_CHANGE, STATUS_CHANGE, TOPOLOGY_CHANGE]
22:47:20.564 [s0-admin-1] DEBUG c.d.o.d.i.core.context.EventBus - [s0] Registering
com.datastax.oss.driver.internal.core.metadata.LoadBalancingPolicyWrapper$$Lambda
$812/0x00000000801105e88@769afb95 for class
com.datastax.oss.driver.internal.core.metadata.NodeStateEvent
22:47:20.566 [s0-admin-1] DEBUG c.d.o.d.i.c.c.ControlConnection -
[s0] Trying to establish a connection to Node(endPoint=cassandra.us-
east-1.amazonaws.com/1.222.333.44:9142, hostId=null, hashCode=e750d92)

```

5. Exécutez la requête CQL suivante sur votre table Amazon Keyspaces pour confirmer qu'une ligne de données a été écrite dans votre table :

```
SELECT * from aws.user;
```

Vous devriez voir la sortie suivante :

fname	lname	username	last_update_date
random	k	test	2023-12-07 13:58:31.57+0000

Étape 5 : Nettoyage (facultatif)

Suivez ces étapes pour supprimer toutes les ressources créées dans ce didacticiel.

Supprimer les ressources créées dans ce didacticiel

1. Supprimez votre déploiement. Pour ce faire, vous pouvez utiliser la commande suivante.

```
kubectl delete deployment my-keyspaces-app -n my-eks-namespace
```

2. Supprimez le cluster Amazon EKS et tous les pods qu'il contient. Cela supprime également les ressources associées telles que le compte de service et le fournisseur d'identité OIDC. Pour ce faire, vous pouvez utiliser la commande suivante.

```
eksctl delete cluster --name my-eks-cluster --region us-east-1
```

3. Supprimez le rôle IAM utilisé pour le compte de service Amazon EKS avec des autorisations d'accès à Amazon Keyspaces. Tout d'abord, vous devez supprimer la politique gérée attachée au rôle.

```
aws iam detach-role-policy --role-name my-iam-role --policy-arn
arn:aws:iam::aws:policy/AmazonKeyspacesFullAccess
```

Vous pouvez ensuite supprimer le rôle à l'aide de la commande suivante.

```
aws iam delete-role --role-name my-iam-role
```

Pour plus d'informations, consultez [la section Suppression d'un rôle IAM \(AWS CLI\)](#) dans le guide de l'utilisateur IAM.

4. Supprimez le référentiel Amazon ECR, y compris toutes les images qui y sont stockées. Vous pouvez le faire à l'aide de la commande suivante.

```
aws ecr delete-repository \
    --repository-name my-ecr-repository \
    --force \
    --region us-east-1
```

Notez que l'option `--force` est nécessaire pour supprimer un dépôt contenant des images. Pour supprimer d'abord votre image, vous pouvez le faire à l'aide de la commande suivante.

```
aws ecr batch-delete-image \
    --repository-name my-ecr-repository \
    --image-ids imageTag=latest \
    --region us-east-1
```

Pour plus d'informations, consultez [Supprimer une image](#) dans le guide de l'utilisateur d'Amazon Elastic Container Registry.

5. Supprimez le keyspace et le tableau Amazon Keyspaces. La suppression de l'espace-clavier entraîne automatiquement la suppression de toutes les tables qu'il contient. Pour ce faire, vous pouvez utiliser l'une des options suivantes.

AWS CLI

```
aws keyspaces delete-keyspace --keyspace-name 'aws'
```

Pour confirmer que le keyspace a été supprimé, vous pouvez utiliser la commande suivante.

```
aws keyspaces list-keyspaces
```

Pour supprimer d'abord le tableau, vous pouvez utiliser la commande suivante.

```
aws keyspaces delete-table --keyspace-name 'aws' --table-name 'user'
```

Pour confirmer que votre table a été supprimée, vous pouvez utiliser la commande suivante.

```
aws keyspaces list-tables --keyspace-name 'aws'
```

Pour plus d'informations, voir [supprimer un espace de touches](#) et [supprimer une table](#) dans le manuel de référence des AWS CLI commandes.

cqlsh

```
DROP KEYSPACE IF EXISTS "aws";
```

Pour vérifier que vos espaces de touches ont été supprimés, vous pouvez utiliser l'instruction suivante.

```
SELECT * FROM system_schema.keyspaces ;
```

Votre espace de touche ne doit pas être répertorié dans le résultat de cette instruction. Notez qu'il peut y avoir un délai avant que les espaces clés ne soient supprimés. Pour plus d'informations, consultez [the section called “DROP KEYSPACE”](#).

Pour supprimer d'abord le tableau, vous pouvez utiliser la commande suivante.

```
DROP TABLE "aws.user"
```

Pour confirmer que votre table a été supprimée, vous pouvez utiliser la commande suivante.


```
SELECT * FROM system_schema.tables WHERE keyspace_name = "aws";
```

Votre table ne doit pas être répertoriée dans le résultat de cette instruction. Notez qu'il peut y avoir un délai avant que la table ne soit supprimée. Pour plus d'informations, consultez [the section called "DROP TABLE"](#).

Tutoriel : Exporter un tableau Amazon Keyspaces vers Amazon S3 à l'aide de AWS Glue

Ce didacticiel explique comment exporter un tableau Amazon Keyspaces vers un compartiment Amazon S3 à l'aide AWS Glue de. Le didacticiel utilise l'outil `keyspaces-bulk-cli` CLI disponible dans le [GitHub](#) référentiel Amazon Keyspaces. Grâce à cet outil, vous pouvez exporter des données Amazon Keyspaces vers Amazon S3 sans avoir à configurer de cluster Spark.

Rubriques

- [Conditions requises pour exporter des données depuis Amazon Keyspaces vers Amazon S3](#)
- [Étape 1 : Bootstrap de l'infrastructure et AWS Glue jobs](#)
- [Étape 2 : Exécuter la tâche d'exportation](#)
- [Étape 3 : \(Facultatif\) Créez un déclencheur pour planifier la tâche d'exportation](#)
- [Étape 4 : Nettoyage \(facultatif\)](#)

Conditions requises pour exporter des données depuis Amazon Keyspaces vers Amazon S3

Vérifiez les prérequis suivants et créez les ressources Amazon Keyspaces avant de commencer le didacticiel.

1. Avant de commencer ce didacticiel, suivez les instructions de AWS configuration figurant dans [Accès à Amazon Keyspaces \(pour Apache Cassandra\)](#). Ces étapes incluent l'inscription AWS et la création d'un principal Gestion des identités et des accès AWS (IAM) avec accès à Amazon Keyspaces.

2. L'outil CLI de ce didacticiel utilise vos informations d'identification et les Région AWS stocke par défaut dans un emplacement connu. Pour de plus amples informations, veuillez consulter [the section called “Gérer des clés d'accès”](#).

L'exemple suivant montre comment stocker les valeurs requises en tant que variables d'environnement pour l'utilisateur par défaut.

```
$ export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY
$ export AWS_DEFAULT_REGION=us-east-1
```

3. Pour exécuter la `keyspaces-bulk-cli` CLI de ce didacticiel, les logiciels et outils suivants doivent être installés sur votre machine :

- [Python 3.8](#) ou version ultérieure
- [Java](#) (JDK 11 ou version ultérieure)
- [Apache Maven](#)
- [Git](#)
- [curl](#)
- [AWS CLI](#)

Java, Git et Maven sont requis par l'étape bootstrap pour télécharger et créer l'assistant de politique de nouvelle tentative. Ce didacticiel a été testé avec AWS CLI 2, Python 3.11, Java 17.0.13 et Apache Maven 3.8.7.

4. Vous avez besoin d'un tableau Amazon Keyspaces contenant des exemples de données à exporter ultérieurement dans ce didacticiel. Vous pouvez utiliser votre propre tableau Amazon Keyspaces ou créer un exemple de tableau en suivant les étapes du [Prise en main](#) didacticiel.
 - a. Pour installer `lecqlsh-expansion`, suivez les étapes décrites à l'adresse [the section called “Utilisation de l’cqlsh-expansion”](#).
 - b. Vérifiez que le `Murmur3Partitioner` partitionneur est le partitionneur par défaut pour votre compte. Ce partitionneur est compatible avec le connecteur Apache Spark Cassandra et avec AWS Glue. Pour plus d'informations sur les partitionneurs, consultez [the section called “Utilisation de partitionneurs”](#)

Pour modifier le partitionneur de votre compte, vous pouvez utiliser la déclaration suivante.

```
SELECT partitioner FROM system.local;  
  
UPDATE system.local set  
partitioner='org.apache.cassandra.dht.Murmur3Partitioner' where key='local';
```

- c. Pour créer un espace de clés Amazon Keyspaces, suivez les étapes décrites à [the section called “Création d'un espace clé”](#)
- d. Pour créer le tableau Amazon Keyspaces, suivez les étapes décrites à [the section called “Création d'une table”](#).
- e. Pour charger des exemples de données dans le tableau afin de les exporter vers Amazon S3, suivez les étapes décrites à la section [the section called “Créer”](#).

Après avoir effectué les étapes préalables, passez à [the section called “Étape 1 : infrastructure et AWS Glue emplois Bootstrap”](#).

Étape 1 : Bootstrap de l'infrastructure et AWS Glue jobs

Au cours de cette étape, vous utilisez l'`keyspaces-bulk-cli` interface de ligne de commande pour créer et configurer toutes les AWS ressources requises pour l'exportation automatique des données d'une table Amazon Keyspaces vers un compartiment Amazon S3 à l'aide AWS Glue de. La `bootstrap` commande exécute toutes les tâches de configuration en une seule étape.

La `bootstrap` commande automatise les tâches suivantes.

1. Crée un compartiment Amazon S3 et un rôle de service IAM à l'aide CloudFormation de.
2. Télécharge le connecteur [Apache Spark Cassandra](#), le plug-in d'authentification [Sigv4 et](#) les extensions [Apache Spark](#).
3. Télécharge et crée l'assistant [Keyspaces Retry Policy à l'aide de Maven](#).
4. Télécharge tous les fichiers JAR et le fichier `keyspaces-application.conf` de configuration dans le compartiment Amazon S3.
5. Déploie trois AWS Glue tâches : exporter (exporter un tableau vers Amazon S3), importer (importer des données depuis Amazon S3 dans un tableau) et compter (compter les lignes d'un tableau).
6. Enregistre la configuration de la pile dans un `.keyspaces-bulk-cli.json` fichier local pour être utilisée par les commandes suivantes.

Pour démarrer l'infrastructure et AWS Glue jobs

1. Clonez les fichiers du [référentiel](#) aws-glue sur votre GitHub machine locale.

```
$ git clone https://github.com/aws-samples/amazon-keyspaces-examples.git
$ cd amazon-keyspaces-examples/scala/datastax-v4/aws-glue
```

2. Exécutez la commande bootstrap. L'exemple suivant utilise le nom de pile par défaut aksglue et spécifie l'espace de touches et la table comme valeurs par défaut pour les tâches déployées. Vous pouvez modifier ces valeurs lorsque vous exécutez des commandes individuelles telles que exportimport, oucount.

```
$ ./keyspaces-bulk-cli bootstrap --stack aksglue --keyspace catalog --
table book_awards
```

Le tableau suivant décrit les paramètres facultatifs que vous pouvez transmettre pour personnaliser le bootstrap.

Option	Par défaut	Description
--stack	aksglue	CloudFormation préfixe du nom de la pile
--bucket	Auto-generated à partir de la pile et du compte	Compartiment Amazon S3 pour les artefacts
--role-name	Auto-generated depuis stack	Nom du rôle de service IAM
--keyspace	mykeyspace	Espace de touches par défaut pour les tâches déployées
--table	mytable	Tableau par défaut pour les tâches déployées
--s3-uri	s3://{bucket}/export	Chemin Amazon S3 par défaut pour les données exportées
--format	parquet	Format de données par défaut

Option	Par défaut	Description
<code>--region</code>	À partir de la AWS configuration	Région AWS
<code>--profile</code>	À partir de la AWS configuration	AWS profil nommé

Pour vérifier que le compartiment Amazon S3 créé par bootstrap existe, vous pouvez utiliser la AWS CLI commande suivante. ***YOURACCOUNTID*** Remplacez-le par votre identifiant de AWS compte.

```
$ aws s3 ls s3://amazon-keyspaces-bulk-cli-aksglue-YOURACCOUNTID
```

Le résultat de la commande ressemble à ce qui suit :

```
PRE conf/
PRE jars/
PRE scripts/
```

Pour confirmer que la tâche AWS Glue d'exportation a été déployée, vous pouvez utiliser la commande suivante.

```
$ aws glue list-jobs
```

La sortie répertorie les tâches déployées :

```
{
  "JobNames": [
    "AmazonKeyspacesExportToS3-aksglue",
    "AmazonKeyspacesImportFromS3-aksglue",
    "AmazonKeyspacesCount-aksglue"
  ]
}
```

Si le processus de CloudFormation pile échoue, vous pouvez consulter les informations d'erreur détaillées concernant la pile défaillante dans la CloudFormation console. Pour réessayer, supprimez

la pile qui a échoué `aws cloudformation delete-stack --stack-name aksglue`, corrigez le problème sous-jacent et exécutez à nouveau la commande `bootstrap`. La commande `bootstrap` est idempotente et ignore les ressources qui existent déjà.

Une fois le `bootstrap` terminé et toutes les ressources créées, passez à [the section called “Étape 2 : Exécuter la tâche d'exportation”](#).

Étape 2 : Exécuter la tâche d'exportation

Dans cette étape, vous exécutez la tâche AWS Glue d'exportation créée à l'étape précédente. Après avoir démarré l'exportation, vous pouvez suivre la progression de la tâche à l'aide `status` des logs commandes et.

Pour exécuter et surveiller la tâche d'exportation

1. Exécutez la commande d'exportation. Spécifiez l'espace clé, la table et le compartiment Amazon S3 vers lesquels exporter. Utilisez le compartiment créé par la commande `bootstrap` à l'étape précédente. *YOURACCOUNTID* Remplacez-le par votre identifiant de AWS compte.

```
$ ./keyspaces-bulk-cli export --keyspace catalog --table book_awards \  
  --s3-uri s3://amazon-keyspaces-bulk-cli-aksglue-YOURACCOUNTID
```

- Vous pouvez modifier des paramètres supplémentaires lors de l'exécution de l'exportation. La commande suivante augmente le nombre de AWS Glue travailleurs. Commencez par la valeur par défaut de 2 travailleurs et augmentez selon les besoins en fonction de la taille de la table et de la durée d'exportation. Surveillez la tâche à l'aide de la `status` commande et augmentez la taille si l'exportation prend plus de temps que prévu.

```
$ ./keyspaces-bulk-cli export --keyspace catalog --table book_awards \  
  --s3-uri s3://amazon-keyspaces-bulk-cli-aksglue-YOURACCOUNTID --workers 8
```

2. Surveillez l'état de la tâche d'exportation. Vous pouvez vérifier l'état d'exécution actuel et consulter les journaux.

```
$ ./keyspaces-bulk-cli status export
```

La sortie affiche les détails de l'exécution de la tâche. Attendez que le `State` champ s'affiche `SUCCEEDED` :

```

Job Run: jr_6f8de24d7131da54...
#####
# Field          # Value                                     #
#####
# Job Name       # AmazonKeyspacesExportToS3-aksglue        #
# State          # SUCCEEDED                                #
# Started        # 2025-01-30 14:54:22                     #
# Completed      # 2025-01-30 14:56:45                     #
# Execution Time # 143s                                         #
# Workers        # 2                                           #
# Worker Type    # G.2X                                         #
#####

```

Pour afficher les journaux de la tâche en cours ou terminée, utilisez la commande suivante.

```
$ ./keyspaces-bulk-cli logs export
```

Pour afficher uniquement les journaux d'erreurs, utilisez la commande suivante.

```
$ ./keyspaces-bulk-cli logs export --log-type error
```

3. Vérifiez que l'exportation est terminée en répertoriant le contenu du compartiment Amazon S3. Selon la taille du tableau, l'exportation peut prendre un certain temps. Une fois la tâche d'exportation terminée, vous pouvez voir les dossiers suivants dans le compartiment.

YOURACCOUNTID Remplacez-le par votre identifiant de AWS compte.

```
$ aws s3 ls s3://amazon-keyspaces-bulk-cli-aksglue-YOURACCOUNTID
```

La sortie montre la structure suivante dans votre compartiment :

```

PRE conf/
PRE export/
PRE jars/
PRE scripts/
PRE spark-logs/

```

Vos fichiers de données exportés se trouvent dans la structure de dossiers suivante (les valeurs de date et d'heure reflètent votre propre cycle d'exportation) :

```

\----- export
    \----- catalog
        \----- book_awards
            \----- snapshot
                \----- year=2025
                    \----- month=01
                        \----- day=30
                            \----- hour=14
                                \----- minute=54
                                    \--- part-00000-*.snappy.parquet

```

Pour planifier la AWS Glue tâche que vous venez d'exécuter manuellement, passez à [the section called “Étape 3 : \(Facultatif\) Planifiez la tâche d'exportation”](#).

Étape 3 : (Facultatif) Créez un déclencheur pour planifier la tâche d'exportation

Pour exécuter régulièrement la tâche d'exportation, vous pouvez créer un déclencheur programmé à l'aide du AWS CLI. Pour plus d'informations, consultez les [AWS Glue déclencheurs](#) dans le Guide du AWS Glue développeur.

Pour planifier le AWS Glue emploi à l'export

1. Pour trouver le nom de la tâche et les paramètres d'une exécution précédente, utilisez l'--json indicateur pour afficher les détails de l'exécution, y compris les arguments utilisés.

```
$ ./keyspaces-bulk-cli runs export --json
```

La sortie inclut le nom de la tâche, la configuration du worker et tous les arguments de chaque exécution :

```

[
  {
    "JobName": "AmazonKeyspacesExportToS3-aksglue",
    "RunId": "jr_2d827b0637a36b25f32f03b83e107cf8...",
    "State": "SUCCEEDED",
    "Started": "2025-01-30T14:54:22.480000-04:00",
    "Duration": 127,
    "Workers": 2,

```



```

    "WorkerType": "G.2X",
    "Arguments": {
      "--KEYSPACE_NAME": "catalog",
      "--DRIVER_CONF": "keyspaces-application.conf",
      "--TABLE_NAME": "book_awards",
      "--S3_URI": "s3://amazon-keyspaces-bulk-cli-aksglue-111122223333",
      "--FORMAT": "parquet"
    }
  }
]

```

Utilisez les Arguments valeurs JobNameWorkers,WorkerType, et de la sortie de l'étape suivante pour créer le déclencheur.

2. La AWS CLI commande suivante crée un déclencheur nommé `KeyspacesExportWeeklyTrigger` qui exécute la tâche AWS Glue d'exportation une fois par semaine le lundi à 12 h UTC. Utilisez les valeurs de la sortie JSON de l'étape précédente pour le nom de la tâche, la configuration du collaborateur et les arguments.

```

aws glue create-trigger \
  --name KeyspacesExportWeeklyTrigger \
  --type SCHEDULED \
  --schedule "cron(0 12 ? * MON *)" \
  --start-on-creation \
  --actions '[{
    "JobName": "AmazonKeyspacesExportToS3-aksglue",
    "NumberOfWorkers": 2,
    "WorkerType": "G.2X",
    "Arguments": {
      "--KEYSPACE_NAME": "catalog",
      "--DRIVER_CONF": "keyspaces-application.conf",
      "--TABLE_NAME": "book_awards",
      "--S3_URI": "s3://amazon-keyspaces-bulk-cli-aksglue-YOURACCOUNTID",
      "--FORMAT": "parquet"
    }
  }]'

```

- Pour modifier les paramètres spécifiques de la tâche planifiée, modifiez les valeurs du Arguments bloc. L'exemple suivant planifie l'exportation d'une table différente avec un plus grand nombre de travailleurs.

```
aws glue create-trigger \
```

```
--name KeyspacesExportWeeklyTrigger \
--type SCHEDULED \
--schedule "cron(0 12 ? * MON *)" \
--start-on-creation \
--actions '[{
  "JobName": "AmazonKeyspacesExportToS3-aksglue",
  "NumberOfWorkers": 8,
  "WorkerType": "G.2X",
  "Arguments": {
    "--KEYSPACE_NAME": "my_keyspace",
    "--DRIVER_CONF": "keyspaces-application.conf",
    "--TABLE_NAME": "my_table",
    "--S3_URI": "s3://amazon-keyspaces-bulk-cli-aksglue-YOURACCOUNTID",
    "--FORMAT": "parquet"
  }
}]'
```

3. Pour confirmer que le déclencheur a été créé, utilisez la commande suivante.

```
$ aws glue list-triggers
```

Le résultat de la commande ressemble à ce qui suit :

```
{
  "TriggerNames": [
    "KeyspacesExportWeeklyTrigger"
  ]
}
```

Pour nettoyer les AWS ressources créées dans ce didacticiel, passez à [the section called “Étape 4 : Nettoyage \(facultatif\)”](#).

Étape 4 : Nettoyage (facultatif)

Suivez ces étapes pour supprimer toutes les AWS ressources créées dans ce didacticiel. Les piles de AWS Glue tâches doivent être supprimées avant la pile d'infrastructure parente car elles font référence à ses valeurs exportées.

Pour supprimer les ressources créées dans ce didacticiel

1. Supprimez le contenu du compartiment Amazon S3 créé par bootstrap.

YOURACCOUNTID Remplacez-le par votre identifiant de AWS compte.

```
$ aws s3 rm s3://amazon-keyspaces-bulk-cli-aksglue-YOURACCOUNTID --recursive
```

2. Supprimez les piles de AWS Glue tâches créées par la commande bootstrap. Ces piles enfants doivent être supprimées avant la pile parent. Vous pouvez utiliser les commandes suivantes.

```
$ aws cloudformation delete-stack --stack-name aksglue-export  
$ aws cloudformation delete-stack --stack-name aksglue-import  
$ aws cloudformation delete-stack --stack-name aksglue-count
```

Attendez que les piles enfants aient fini de supprimer avant de continuer. Vous pouvez vérifier l'état à l'aide de la commande suivante.

```
$ aws cloudformation wait stack-delete-complete --stack-name aksglue-export  
$ aws cloudformation wait stack-delete-complete --stack-name aksglue-import  
$ aws cloudformation wait stack-delete-complete --stack-name aksglue-count
```

3. Supprimez la CloudFormation pile parent. Cela supprime la ressource de compartiment Amazon S3 et le rôle IAM créés dans ce didacticiel.

```
$ aws cloudformation delete-stack --stack-name aksglue
```

4. Si vous avez créé un déclencheur à l'étape précédente, supprimez-le à l'aide de la commande suivante.

```
$ aws glue delete-trigger --name KeyspacesExportWeeklyTrigger
```

5. Supprimez l'espace clé et le tableau Amazon Keyspaces. La suppression de l'espace clé entraîne automatiquement la suppression de toutes les tables qu'il contient. Vous pouvez utiliser l'une des options suivantes.

AWS CLI

```
$ aws keyspaces delete-keyspace --keyspace-name 'catalog'
```

Pour confirmer que l'espace clé a été supprimé, vous pouvez utiliser la commande suivante.

```
$ aws keyspaces list-keyspaces
```

Pour supprimer d'abord le tableau, vous pouvez utiliser la commande suivante.

```
$ aws keyspaces delete-table --keyspace-name 'catalog' --table-name 'book_awards'
```

Pour confirmer que votre table a été supprimée, vous pouvez utiliser la commande suivante.

```
$ aws keyspaces list-tables --keyspace-name 'catalog'
```

Pour plus d'informations, voir [supprimer l'espace clavier](#) et [supprimer la table](#) dans la référence des AWS CLI commandes.

cqlsh

```
DROP KEYSPACE IF EXISTS "catalog";
```

Pour vérifier que votre espace clé a été supprimé, vous pouvez utiliser l'instruction suivante.

```
SELECT * FROM system_schema.keyspaces ;
```

Votre espace clavier ne doit pas figurer dans le résultat de cette instruction. Notez qu'il peut y avoir un certain délai avant que l'espace de touche ne soit supprimé. Pour de plus amples informations, veuillez consulter [the section called “DROP KEYSPACE”](#).

Pour supprimer d'abord le tableau, vous pouvez utiliser la commande suivante.

```
DROP TABLE "catalog". "book_awards";
```

Pour confirmer que votre table a été supprimée, vous pouvez utiliser la commande suivante.

```
SELECT * FROM system_schema.tables WHERE keyspace_name = "catalog";
```

Votre tableau ne doit pas figurer dans le résultat de cette instruction. Notez qu'il peut y avoir un délai avant que la table ne soit supprimée. Pour de plus amples informations, veuillez consulter [the section called “DROP TABLE”](#).

Gestion des ressources sans serveur dans Amazon Keyspaces (pour Apache Cassandra)

Amazon Keyspaces (pour Apache Cassandra) est sans serveur. Au lieu de déployer, de gérer et de maintenir les ressources de stockage et de calcul pour votre charge de travail via les nœuds d'un cluster, Amazon Keyspaces alloue les ressources de stockage et de read/write débit directement aux tables.

Amazon Keyspaces provisionne le stockage automatiquement en fonction des données stockées dans vos tables. Il augmente ou diminue le stockage au fur et à mesure que vous écrivez, mettez à jour et supprimez des données, et vous ne payez que pour le stockage que vous utilisez. Les données sont répliquées sur plusieurs [zones de disponibilité](#) pour une haute disponibilité. Amazon Keyspaces surveille en permanence la taille de vos tables afin de déterminer vos frais de stockage. Pour plus d'informations sur la façon dont Amazon Keyspaces calcule la taille facturable des données, consultez [the section called “Estimer la taille des lignes”](#).

Ce chapitre couvre les principaux aspects de la gestion des ressources dans Amazon Keyspaces.

- Estimation de la taille des lignes : pour estimer la taille codée des lignes dans Amazon Keyspaces, prenez en compte des facteurs tels que les métadonnées des clés de partition, les métadonnées des colonnes en cluster, les identifiants de colonne, les types de données et les métadonnées des lignes. Cette taille de ligne codée est utilisée pour la facturation, la gestion des quotas et la planification de la capacité de débit provisionnée.
- Estimation de la consommation de capacité — Cette section présente des exemples d'estimation de la consommation de capacité de lecture et d'écriture pour des scénarios courants tels que les requêtes par plage, les requêtes de limite, les analyses de tables, les transactions légères, les colonnes statiques et les tables multirégionales. Vous pouvez utiliser Amazon CloudWatch pour surveiller l'utilisation réelle de la capacité. Pour plus d'informations sur la surveillance avec CloudWatch, consultez [the section called “Surveillance avec CloudWatch”](#).
- Configuration des modes de read/write capacité : vous pouvez choisir entre deux modes de capacité pour traiter les lectures et les écritures sur vos tables :
 - On-demand mode (par défaut) — Payez par demande pour le débit de lecture et d'écriture. Amazon Keyspaces peut augmenter instantanément la capacité jusqu'à n'importe quel niveau de trafic atteint précédemment.
 - Mode provisionné — Spécifiez à l'avance le nombre requis d'unités de capacité de lecture et d'écriture. Ce mode permet de maintenir des performances de débit prévisibles.

- Gérez la capacité de débit grâce au dimensionnement automatique : pour les tables provisionnées, vous pouvez activer le dimensionnement automatique afin d'ajuster automatiquement la capacité de débit en fonction du trafic réel des applications. Amazon Keyspaces utilise le suivi des cibles pour augmenter ou diminuer la capacité allouée, tout en maintenant le taux d'utilisation à l'objectif que vous avez spécifié.
- Utilisation efficace de la capacité en rafale : Amazon Keyspaces fournit une capacité en rafale en réservant une partie du débit inutilisé à la gestion des pics de trafic. Cette flexibilité permet des pics d'activité occasionnels au-delà de votre débit provisionné.

Pour résoudre les problèmes de capacité, voir [the section called “Erreurs de capacité sans serveur”](#).

Rubriques

- [Estimer la taille des lignes dans Amazon Keyspaces](#)
- [Estimez la consommation de capacité du débit de lecture et d'écriture dans Amazon Keyspaces](#)
- [Configurer les modes de read/write capacité dans Amazon Keyspaces](#)
- [Gérez automatiquement la capacité de débit grâce à la mise à l'échelle automatique d'Amazon Keyspaces](#)
- [Utilisez efficacement la capacité de pointe dans Amazon Keyspaces](#)

Estimer la taille des lignes dans Amazon Keyspaces

Amazon Keyspaces fournit un stockage entièrement géré qui offre des performances de lecture et d'écriture à un chiffre en millisecondes et stocke les données de manière durable dans plusieurs zones de disponibilité. AWS Amazon Keyspaces associe des métadonnées à toutes les lignes et à toutes les colonnes clés primaires afin de garantir un accès efficace aux données et une haute disponibilité.

Cette rubrique explique comment estimer la taille codée des lignes dans Amazon Keyspaces. La taille de ligne codée est utilisée lors du calcul de votre facture et de votre quota d'utilisation. Vous pouvez également utiliser la taille de ligne codée lors de l'estimation des exigences de capacité de débit allouées pour les tables.

Pour calculer la taille codée des lignes dans Amazon Keyspaces, vous pouvez suivre les directives suivantes.

Rubriques

- [Estimer la taille codée des colonnes](#)
- [Estimer la taille codée des valeurs de données en fonction du type de données](#)
- [Tenez compte de l'impact des fonctionnalités d'Amazon Keyspaces sur la taille des lignes](#)
- [Choisissez la bonne formule pour calculer la taille codée d'une ligne](#)
- [Exemple de calcul de la taille des lignes](#)

Estimer la taille codée des colonnes

Cette section explique comment estimer la taille codée des colonnes dans Amazon Keyspaces.

- **Colonnes normales** : pour les colonnes ordinaires, qui ne sont pas des clés primaires, les colonnes de clustering ou les STATIC colonnes, utilisez la taille brute des données de cellule en fonction du [type de données](#) et ajoutez les métadonnées requises. Les types de données et certaines des principales différences dans la façon dont Amazon Keyspaces stocke les valeurs des types de données et les métadonnées sont répertoriés dans la section suivante.
- **Colonnes de clé de partition** : les clés de partition peuvent contenir jusqu'à 2 048 octets de données. Chaque colonne clé de la clé de partition nécessite jusqu'à 3 octets de métadonnées. Lorsque vous calculez la taille de votre ligne, vous devez partir du principe que chaque colonne de clé de partition utilise les 3 octets complets de métadonnées.
- **Colonnes de clustering** : les colonnes de clustering peuvent stocker jusqu'à 850 octets de données. Outre la taille de la valeur des données, chaque colonne de clustering nécessite jusqu'à 20 % de la taille de la valeur des données pour les métadonnées. Lorsque vous calculez la taille de votre ligne, vous devez ajouter 1 octet de métadonnées pour chaque valeur de 5 octets de données de colonne de clustering.

Note

Pour permettre des requêtes efficaces et une indexation intégrée, Amazon Keyspaces stocke deux fois la valeur des données de chaque clé de partition et de chaque colonne de clé de clustering.

- **Noms de colonne** — L'espace requis pour chaque nom de colonne est stocké à l'aide d'un identifiant de colonne et ajouté à chaque valeur de données stockée dans la colonne. La valeur de stockage de l'identifiant de colonne dépend du nombre total de colonnes de votre table :
 - 1 à 62 colonnes : 1 octet

- 63 à 124 colonnes : 2 octets
- 125 à 186 colonnes : 3 octets

Pour chaque 62 colonnes supplémentaires, ajoutez 1 octet. Notez que dans Amazon Keyspaces, jusqu'à 225 colonnes normales peuvent être modifiées à l'aide d'une seule instruction INSERT ou UPDATE d'une seule instruction. Pour de plus amples informations, veuillez consulter [the section called "Quotas de service Amazon Keyspaces"](#).

Estimer la taille codée des valeurs de données en fonction du type de données

Cette section explique comment estimer la taille codée de différents types de données dans Amazon Keyspaces.

- Types de chaînes : les types de données Cassandra ASCII et VARCHAR string sont tous stockés dans Amazon Keyspaces en Unicode UTF-8 avec un codage binaire. TEXT La taille d'une chaîne dans Amazon Keyspaces est égale au nombre d'octets UTF-8 codés.
- Types numériques : CassandraINT,, BIGINT SMALLINTTINYINT, et les types de VARINT données sont stockés dans Amazon Keyspaces sous forme de valeurs de données de longueur variable, avec un maximum de 38 chiffres significatifs. Les zéros de début et de fin sont tronqués. La taille de chacun de ces types de données est d'environ 1 octet pour deux chiffres significatifs + 1 octet.
- Type de blob : BLOB dans Amazon Keyspaces, A est stocké avec la longueur d'octet brute de la valeur.
- Type booléen — La taille d'une Boolean valeur ou d'une Null valeur est de 1 octet.
- Types de collections : colonne qui stocke des types de données de collection tels que LIST ou MAP nécessitant 3 octets de métadonnées, quel que soit leur contenu. La taille d'un LIST or MAP est (identifiant de colonne) + somme (taille des éléments imbriqués) + (3 octets). La taille d'un LIST or vide MAP est (identifiant de colonne) + (3 octets). Chaque individu LIST ou MAP élément nécessite également 1 octet de métadonnées.
- User-defined types — Un [type défini par l'utilisateur \(UDT\)](#) nécessite 3 octets pour les métadonnées, quel que soit son contenu. Pour chaque élément UDT, Amazon Keyspaces a besoin d'un octet supplémentaire de métadonnées.

Pour calculer la taille codée d'un UDT, commencez par le `field name` et `field value` pour les champs d'un UDT :

- **nom de champ** — Chaque nom de champ de l'UDT de niveau supérieur est stocké à l'aide d'un identifiant. La valeur de stockage de l'identifiant dépend du nombre total de champs dans votre UDT de premier niveau et peut varier entre 1 et 3 octets :
 - 1 à 62 champs : 1 octet
 - 63 à 124 champs : 2 octets
 - 125— champs maximum : 3 octets
- **valeur du champ** — Les octets nécessaires pour stocker les valeurs de champ de l'UDT de niveau supérieur dépendent du type de données stocké :
 - **Type de données scalaire** — Les octets requis pour le stockage sont les mêmes que pour le même type de données stocké dans une colonne normale.
 - **UDT figé** — Pour chaque UDT imbriqué figé, l'UDT imbriqué a la même taille que dans le protocole binaire CQL. Pour un UDT imbriqué, 4 octets sont stockés pour chaque champ (y compris les champs vides) et la valeur du champ stocké est le format de sérialisation du protocole binaire CQL de la valeur du champ.
- **Collections Frozen** :
 - **LIST et SET** — Dans le cas d'un LIST or figé imbriquéSET, 4 octets sont stockés pour chaque élément de la collection, plus le format de sérialisation du protocole binaire CQL de la valeur de la collection.
 - **MAP** — Pour une paire figée imbriquéeMAP, chaque paire clé-valeur a les exigences de stockage suivantes :
 - Pour chaque clé, allouez 4 octets, puis ajoutez le format de sérialisation du protocole binaire CQL de la clé.
 - Pour chaque valeur, allouez 4 octets, puis ajoutez le format de sérialisation du protocole binaire CQL de la valeur.
- **Mot-clé FROZEN** : pour les collections figées imbriquées dans des collections figées, Amazon Keyspaces ne nécessite aucun octet supplémentaire pour les métadonnées.
- **Mot-clé STATIC** : les données des STATIC colonnes ne sont pas prises en compte dans la taille de ligne maximale de 1 Mo. Pour calculer la taille des données des colonnes statiques, consultez [the section called “Calculer la taille de colonne statique par partition logique”](#).

Tenez compte de l'impact des fonctionnalités d'Amazon Keyspaces sur la taille des lignes

Cette section explique l'impact des Client-side horodatages et des fonctionnalités Time to Live (TTL) d'Amazon Keyspaces sur la taille codée d'une ligne.

Lorsque les horodatages côté client sont activés ou que le TTL est utilisé (à l'aide de la `USING TTL` clause), des métadonnées supplémentaires sont stockées pour chaque colonne de chaque ligne de votre table. En outre, des métadonnées peuvent également être stockées pour chaque élément dans le cas de certains types de données de collecte. Le tableau suivant indique le surcoût des métadonnées pour chaque type de données.

Column/Data Type	Client-side horodatages activés	TTL sans horodatage côté client	Client-side horodatages activés avec TTL
Types scalar	18 bytes/column	16 bytes/column	27 bytes/column
Collection Frozen/Frozen UDT/Tuple	18 bytes/column	16 bytes/column	27 bytes/column
Non-frozen Liste	30 bytes/element + 19 bytes/column	15 bytes/element	39 bytes/element + 19 bytes/column
Non-frozen Set	14 bytes/element + 26 bytes/column	12 bytes/element + 7 bytes/column	23 bytes/element + 26 bytes/column
Non-frozen Carte/ Non-frozen UDT	17 bytes/element + 26 bytes/column	15 bytes/element + 7 bytes/column	26 bytes/element + 26 bytes/column
Compteur (région unique)	51 bytes/column	N/A (pas de support TTL)	N/A (pas de support TTL)
Compteur (multirégional)	25 + (N_régions × 26) bytes/column	N/A (pas de support TTL)	N/A (pas de support TTL)

Ces octets de métadonnées sont pris en compte dans votre quota de taille de ligne de 1 Mo. Dans la formule du compteur, N_Regions représente le nombre de AWS régions dans lesquelles la colonne du compteur a reçu au moins une opération d'écriture. Les valeurs des frais généraux du compteur

sont approximatives et peuvent varier légèrement. Pour plus d'informations sur les horodatages côté client, consultez [the section called “Client-side horodatages”](#). Pour plus d'informations sur TTL, consultez [the section called “Expirer les données avec Time to Live”](#).

Choisissez la bonne formule pour calculer la taille codée d'une ligne

Cette section présente les différentes formules que vous pouvez utiliser pour estimer les exigences de stockage ou de capacité de débit pour une ligne de données dans Amazon Keyspaces.

La taille totale codée d'une ligne de données peut être estimée à l'aide de l'une des formules suivantes, en fonction de votre objectif :

- Capacité de débit : pour estimer la taille codée d'une ligne afin d'évaluer les unités de read/write demande (RRUs/WRUs) ou de read/write capacité (RCUs/WCUs) requises :

```
total encoded size of row = partition key columns + clustering columns + regular columns
```

- Taille de stockage : pour estimer la taille codée d'une ligne afin de la prévoir `BillableTableSizeInBytes`, ajoutez les métadonnées requises pour le stockage de la ligne :

```
total encoded size of row = partition key columns + clustering columns + regular columns + row metadata (100 bytes)
```

Important

Toutes les métadonnées de colonne, par exemple les identifiants de colonne, les métadonnées de clé de partition, les métadonnées de colonne de clustering, ainsi que les horodatages côté client, le TTL et les métadonnées de ligne sont prises en compte dans la taille de ligne maximale de 1 Mo.

Exemple de calcul de la taille des lignes

Prenons l'exemple suivant d'une table où toutes les colonnes sont de type entier. La table comporte deux colonnes de clé de partition, deux colonnes de clustering et une colonne normale. Comme ce tableau comporte cinq colonnes, l'espace requis pour l'identifiant du nom de colonne est de 1 octet.

```
CREATE TABLE mykeyspace.mytable(pk_col1 int, pk_col2 int, ck_col1 int, ck_col2 int,
reg_col1 int, primary key((pk_col1, pk_col2),ck_col1, ck_col2));
```

Dans cet exemple, nous calculons la taille des données lorsque nous écrivons une ligne dans le tableau, comme indiqué dans l'instruction suivante :

```
INSERT INTO mykeyspace.mytable (pk_col1, pk_col2, ck_col1, ck_col2, reg_col1)
values(1,2,3,4,5);
```

Pour estimer le nombre total d'octets requis par cette opération d'écriture, vous pouvez suivre les étapes suivantes.

1. Calculez la taille d'une colonne de clé de partition en ajoutant les octets correspondant au type de données stocké dans la colonne et les octets de métadonnées. Répétez cette opération pour toutes les colonnes clés de partition.

- a. Calculez la taille de la première colonne de la clé de partition (pk_col1) :

```
(2 bytes for the integer data type) x 2 + 1 byte for the column id + 3 bytes
for partition key metadata = 8 bytes
```

- b. Calculez la taille de la deuxième colonne de la clé de partition (pk_col2) :

```
(2 bytes for the integer data type) x 2 + 1 byte for the column id + 3 bytes
for partition key metadata = 8 bytes
```

- c. Ajoutez les deux colonnes pour obtenir la taille totale estimée des colonnes clés de partition :

```
8 bytes + 8 bytes = 16 bytes for the partition key columns
```

2. Calculez la taille de la colonne de clustering en ajoutant les octets correspondant au type de données stocké dans la colonne et les octets de métadonnées. Répétez cette opération pour toutes les colonnes de clustering.

- a. Calculez la taille de la première colonne de la colonne de clustering (ck_col1) :

```
(2 bytes for the integer data type) x 2 + 20% of the data value (2 bytes) for
clustering column metadata + 1 byte for the column id = 6 bytes
```

- b. Calculez la taille de la deuxième colonne de la colonne de clustering (ck_col2) :

```
(2 bytes for the integer data type) x 2 + 20% of the data value (2 bytes) for  
clustering column metadata + 1 byte for the column id = 6 bytes
```

- c. Ajoutez les deux colonnes pour obtenir la taille totale estimée des colonnes de clustering :

```
6 bytes + 6 bytes = 12 bytes for the clustering columns
```

3. Ajoutez la taille des colonnes normales. Dans cet exemple, nous n'avons qu'une seule colonne qui stocke un entier à un chiffre, ce qui nécessite 2 octets dont 1 octet pour l'identifiant de colonne.
4. Enfin, pour obtenir la taille totale des lignes codées, additionnez les octets de toutes les colonnes. Pour estimer la taille facturable du stockage, ajoutez les 100 octets supplémentaires pour les métadonnées des lignes :

```
16 bytes for the partition key columns + 12 bytes for clustering columns + 3 bytes  
for the regular column + 100 bytes for row metadata = 131 bytes.
```

Pour savoir comment surveiller les ressources sans serveur avec Amazon CloudWatch, consultez [the section called “Surveillance avec CloudWatch”](#).

Estimez la consommation de capacité du débit de lecture et d'écriture dans Amazon Keyspaces

Lorsque vous lisez ou écrivez des données dans Amazon Keyspaces, le nombre d'unités de read/write demande (RRUs/WRUs) ou d'unités de read/write capacité (RCUs/WCUs) consommées par votre requête dépend de la quantité totale de données qu'Amazon Keyspaces doit traiter pour exécuter la requête. Dans certains cas, les données renvoyées au client peuvent être un sous-ensemble des données qu'Amazon Keyspaces a dû lire pour traiter la requête. Pour les écritures conditionnelles, Amazon Keyspaces consomme de la capacité d'écriture même si la vérification conditionnelle échoue.

Pour estimer la quantité totale de données traitées pour une demande, vous devez tenir compte de la taille codée d'une ligne et du nombre total de lignes. Cette rubrique présente quelques exemples de scénarios et de modèles d'accès courants pour montrer comment Amazon Keyspaces traite les requêtes et comment cela affecte la consommation de capacité. Vous pouvez suivre les exemples

pour estimer les besoins en capacité de vos tables et utiliser Amazon CloudWatch pour observer la consommation de capacité de lecture et d'écriture pour ces cas d'utilisation.

Pour plus d'informations sur le calcul de la taille codée des lignes dans Amazon Keyspaces, consultez [the section called “Estimer la taille des lignes”](#)

Rubriques

- [Estimez la consommation de capacité des requêtes de plage dans Amazon Keyspaces](#)
- [Estimer la consommation de capacité de lecture des requêtes limitées](#)
- [Estimez la consommation de capacité de lecture des scans de tables](#)
- [Estimez la consommation de capacité des transactions légères dans Amazon Keyspaces](#)
- [Estimation de la consommation de capacité pour les colonnes statiques dans Amazon Keyspaces](#)
- [Estimer et provisionner la capacité d'une table multirégionale dans Amazon Keyspaces](#)
- [Estimez la consommation de capacité de lecture et d'écriture avec Amazon CloudWatch dans Amazon Keyspaces](#)

Estimez la consommation de capacité des requêtes de plage dans Amazon Keyspaces

Pour examiner la consommation de capacité de lecture d'une requête de plage, nous utilisons le tableau d'exemple suivant qui utilise le mode capacité à la demande.

```
pk1 | pk2 | pk3 | ck1 | ck2 | ck3 | value
-----+-----+-----+-----+-----+-----+-----
a | b | 1 | a | b | 50 | <any value that results in a row size larger than 4KB>
a | b | 1 | a | b | 60 | value_1
a | b | 1 | a | b | 70 | <any value that results in a row size larger than 4KB>
```

Exécutez maintenant la requête suivante sur cette table.

```
SELECT * FROM amazon_keyspaces.example_table_1 WHERE pk1='a' AND pk2='b' AND pk3=1 AND
ck1='a' AND ck2='b' AND ck3 > 50 AND ck3 < 70;
```

Vous recevez le jeu de résultats suivant à partir de la requête et l'opération de lecture effectuée par Amazon Keyspaces consomme 2 RRU en LOCAL_QUORUM mode cohérence.

```
pk1 | pk2 | pk3 | ck1 | ck2 | ck3 | value
```

```
-----+-----+-----+-----+-----+-----+-----+-----+
a | b | 1 | a | b | 60 | value_1
```

Amazon Keyspaces utilise 2 RRU pour évaluer les lignes contenant les valeurs `ck3=60` et `ck3=70` pour traiter la requête. Cependant, Amazon Keyspaces renvoie uniquement la ligne pour laquelle la WHERE condition spécifiée dans la requête est vraie, c'est-à-dire la ligne contenant une valeur. `ck3=60` Pour évaluer la plage spécifiée dans la requête, Amazon Keyspaces lit la ligne correspondant à la limite supérieure de la plage, dans ce cas `ck3 = 70`, mais ne renvoie pas cette ligne dans le résultat. La consommation de capacité de lecture est basée sur les données lues lors du traitement de la requête, et non sur les données renvoyées.

Estimer la consommation de capacité de lecture des requêtes limitées

Lors du traitement d'une requête qui utilise la LIMIT clause, Amazon Keyspaces lit les lignes jusqu'à la taille de page maximale en essayant de répondre à la condition spécifiée dans la requête. Si Amazon Keyspaces ne trouve pas suffisamment de données correspondantes correspondant à la LIMIT valeur de la première page, un ou plusieurs appels paginés peuvent être nécessaires. Pour poursuivre les lectures sur la page suivante, vous pouvez utiliser un jeton de pagination. La taille de page par défaut est de 1 Mo. Pour réduire la capacité de lecture lorsque vous utilisez LIMIT des clauses, vous pouvez réduire la taille de page. Pour plus d'informations sur la pagination, consultez [the section called "Paginer les résultats"](#).

À titre d'exemple, examinons la requête suivante.

```
SELECT * FROM my_table WHERE partition_key=1234 LIMIT 1;
```

Si vous ne définissez pas le format de page, Amazon Keyspaces lit 1 Mo de données même s'il ne vous renvoie qu'une seule ligne. Pour qu'Amazon Keyspaces ne lise qu'une seule ligne, vous pouvez définir le format de page sur 1 pour cette requête. Dans ce cas, Amazon Keyspaces ne lira qu'une seule ligne, à condition qu'aucune ligne n'ait expiré en fonction des Time-to-live paramètres ou des horodatages côté client.

Le PAGE SIZE paramètre détermine le nombre de lignes qu'Amazon Keyspaces analyse depuis le disque pour chaque demande, et non le nombre de lignes renvoyées par Amazon Keyspaces au client. Amazon Keyspaces applique les filtres que vous fournissez, par exemple l'inégalité sur les colonnes non clés ou LIMIT après avoir scanné les données sur le disque. Si vous ne le définissez pas explicitement PAGE SIZE, Amazon Keyspaces lit jusqu'à 1 Mo de données avant d'appliquer des filtres. Par exemple, si vous utilisez LIMIT 1 sans spécifier le PAGE SIZE, Amazon Keyspaces peut

lire des milliers de lignes depuis le disque avant d'appliquer la clause limite et de ne renvoyer qu'une seule ligne.

Pour éviter de trop lire, réduisez le `PAGE_SIZE` nombre de lignes qu'Amazon Keyspaces analyse à chaque lecture. Par exemple, si vous le `PAGE_SIZE` définissez `LIMIT 5` dans votre requête, définissez une valeur comprise entre 5 et 10 afin qu'Amazon Keyspaces ne scanne que 5 à 10 lignes par appel paginé. Vous pouvez modifier ce nombre pour réduire le nombre d'extractions. Pour les limites supérieures à la taille de page, Amazon Keyspaces conserve le nombre total de résultats avec l'état de pagination. Dans le cas d'un fichier `LIMIT` de 10 000 lignes, Amazon Keyspaces peut récupérer ces résultats sur deux pages de 5 000 lignes chacune. La limite de 1 Mo est la limite supérieure de tout ensemble de taille de page.

Estimez la consommation de capacité de lecture des scans de tables

Les requêtes qui aboutissent à une analyse complète de la table, par exemple les requêtes utilisant l'`ALLOW FILTERING` option, sont un autre exemple de requêtes qui traitent plus de lectures que ce qu'elles renvoient sous forme de résultats. Et la consommation de capacité de lecture est basée sur les données lues, et non sur les données renvoyées.

Pour l'exemple de numérisation de tables, nous utilisons l'exemple de tableau suivant en mode capacité à la demande.

```
pk | ck | value
---+---+-----
pk | 10 | <any value that results in a row size larger than 4KB>
pk | 20 | value_1
pk | 30 | <any value that results in a row size larger than 4KB>
```

Amazon Keyspaces crée une table en mode capacité à la demande avec quatre partitions par défaut. Dans cet exemple de tableau, toutes les données sont stockées dans une partition et les trois partitions restantes sont vides.

Exécutez maintenant la requête suivante sur la table.

```
SELECT * from amazon_keyspaces.example_table_2;
```

Cette requête entraîne une opération d'analyse de table au cours de laquelle Amazon Keyspaces analyse les quatre partitions de la table et consomme 6 RRU en `LOCAL_QUORUM` mode cohérence. Tout d'abord, Amazon Keyspaces consomme 3 RRU pour lire les trois lignes avec. `pk = 'pk'` Amazon

Keyspaces consomme ensuite les 3 RRU supplémentaires pour scanner les trois partitions vides de la table. Comme cette requête entraîne une analyse de table, Amazon Keyspaces analyse toutes les partitions de la table, y compris les partitions ne contenant pas de données.

Estimez la consommation de capacité des transactions légères dans Amazon Keyspaces

Les transactions légères (LWT) vous permettent d'effectuer des opérations d'écriture conditionnelle sur les données de votre table. Les opérations de mise à jour conditionnelle sont utiles lors de l'insertion, de la mise à jour et de la suppression d'enregistrements en fonction de conditions évaluant l'état actuel.

Dans Amazon Keyspaces, toutes les opérations d'écriture nécessitent la cohérence LOCAL_QUORUM et l'utilisation des LWT est gratuite. La différence pour les LWT est que lorsqu'une vérification de l'état des LWT se produit, FALSE Amazon Keyspaces consomme des unités de capacité d'écriture (WCU) ou des unités de demande d'écriture (WRU). Le nombre de lignes WCUs/WRUs consommées dépend de la taille de la ligne.

Par exemple, si la taille de la ligne est de 2 Ko, l'écriture conditionnelle échouée en consomme deux WCUs/WRUs. Si la ligne n'existe pas actuellement dans la table, l'opération en consomme une WCUs/WRUs.

Pour déterminer le nombre de demandes ayant entraîné des échecs de vérification des conditions, vous pouvez surveiller la `ConditionalCheckFailed` métrique dans CloudWatch.

Estimez les coûts LWT pour les tables avec Time to Live (TTL)

Les LWT peuvent nécessiter des unités de capacité de lecture (RCU) ou des unités de demande de lecture (RRU) supplémentaires pour les tables configurées avec TTL qui n'utilisent pas d'horodatage côté client. Lorsque la vérification de l'IF EXISTS état de nos IF NOT EXISTS mots clés est utilisée FALSE, les unités de capacité suivantes sont consommées :

- RCUs/RRUs — Si la ligne existe, les valeurs RCUs/RRUs consommées sont basées sur la taille de la ligne existante.
- RCUs/RRUs — Si la ligne n'existe pas, une ligne RCU/RRU est consommée.

Si la condition évaluée aboutit à une opération d'écriture réussie, WCUs/WRUs ils sont consommés en fonction de la taille de la nouvelle ligne.

Estimation de la consommation de capacité pour les colonnes statiques dans Amazon Keyspaces

Dans un tableau Amazon Keyspaces avec des colonnes de regroupement, vous pouvez utiliser le `STATIC` mot clé pour créer une colonne statique. La valeur stockée dans une colonne statique est partagée entre toutes les lignes d'une partition logique. Lorsque vous mettez à jour la valeur de cette colonne, Amazon Keyspaces applique la modification automatiquement à toutes les lignes de la partition.

Cette section explique comment calculer la taille codée des données lorsque vous écrivez dans des colonnes statiques. Ce processus est géré séparément du processus qui écrit des données dans les colonnes non statiques d'une ligne. Outre les quotas de taille pour les données statiques, les opérations de lecture et d'écriture sur des colonnes statiques ont également une incidence indépendante sur la capacité de mesure et de débit des tables. Pour les différences fonctionnelles avec Apache Cassandra lors de l'utilisation de colonnes statiques et de résultats de lecture par page paginée, voir. [the section called “Pagination”](#)

Rubriques

- [Calculez la taille de colonne statique par partition logique dans Amazon Keyspaces](#)
- [Estimez les exigences de capacité et de débit pour les read/write opérations sur des données statiques dans Amazon Keyspaces](#)

Calculez la taille de colonne statique par partition logique dans Amazon Keyspaces

Cette section explique comment estimer la taille codée des colonnes statiques dans Amazon Keyspaces. La taille codée est utilisée lorsque vous calculez votre facture et votre quota d'utilisation. Vous devez également utiliser la taille codée lorsque vous calculez les exigences de capacité de débit allouées pour les tables. Pour calculer la taille codée des colonnes statiques dans Amazon Keyspaces, vous pouvez suivre les directives suivantes.

- Les clés de partition peuvent contenir jusqu'à 2 048 octets de données. Chaque colonne clé de la clé de partition nécessite jusqu'à 3 octets de métadonnées. Ces octets de métadonnées sont pris en compte dans votre quota de taille de données statiques de 1 Mo par partition. Lorsque vous calculez la taille de vos données statiques, vous devez partir du principe que chaque colonne de clé de partition utilise les 3 octets complets de métadonnées.

- Utilisez la taille brute des valeurs de données des colonnes statiques en fonction du type de données. Pour plus d'informations sur les types de données, consultez [the section called “Types de données”](#),
- Ajoutez 104 octets à la taille des données statiques pour les métadonnées.
- Les colonnes de clustering et les colonnes clés ordinaires non primaires ne sont pas prises en compte dans la taille des données statiques. Pour savoir comment estimer la taille des données non statiques au sein des lignes, consultez [the section called “Estimer la taille des lignes”](#).

La taille totale codée d'une colonne statique est basée sur la formule suivante :

```
partition key columns + static columns + metadata = total encoded size of static data
```

Prenons l'exemple suivant d'une table où toutes les colonnes sont de type entier. La table comporte deux colonnes de clé de partition, deux colonnes de clustering, une colonne normale et une colonne statique.

```
CREATE TABLE mykeyspace.mytable(pk_col1 int, pk_col2 int, ck_col1 int, ck_col2  
int, reg_col1 int, static_col1 int static, primary key((pk_col1, pk_col2),ck_col1,  
ck_col2));
```

Dans cet exemple, nous calculons la taille des données statiques de l'instruction suivante :

```
INSERT INTO mykeyspace.mytable (pk_col1, pk_col2, static_col1) values(1,2,6);
```

Pour estimer le nombre total d'octets requis par cette opération d'écriture, vous pouvez suivre les étapes suivantes.

1. Calculez la taille d'une colonne de clé de partition en ajoutant les octets correspondant au type de données stocké dans la colonne et les octets de métadonnées. Répétez cette opération pour toutes les colonnes clés de partition.

- a. Calculez la taille de la première colonne de la clé de partition (pk_col1) :

```
4 bytes for the integer data type + 3 bytes for partition key metadata = 7  
bytes
```

- b. Calculez la taille de la deuxième colonne de la clé de partition (pk_col2) :

```
4 bytes for the integer data type + 3 bytes for partition key metadata = 7
bytes
```

- c. Ajoutez les deux colonnes pour obtenir la taille totale estimée des colonnes clés de partition :

```
7 bytes + 7 bytes = 14 bytes for the partition key columns
```

2. Ajoutez la taille des colonnes statiques. Dans cet exemple, nous n'avons qu'une seule colonne statique qui stocke un entier (ce qui nécessite 4 octets).
3. Enfin, pour obtenir la taille totale codée des données des colonnes statiques, additionnez les octets pour les colonnes clés primaires et les colonnes statiques, puis ajoutez les 104 octets supplémentaires pour les métadonnées :

```
14 bytes for the partition key columns + 4 bytes for the static column + 104 bytes
for metadata = 122 bytes.
```

Vous pouvez également mettre à jour les données statiques et non statiques avec la même instruction. Pour estimer la taille totale de l'opération d'écriture, vous devez d'abord calculer la taille de la mise à jour des données non statiques. Calculez ensuite la taille de la mise à jour de la ligne comme indiqué dans l'exemple sur [the section called “Estimer la taille des lignes”](#), et ajoutez les résultats.

Dans ce cas, vous pouvez écrire un total de 2 Mo : 1 Mo est le quota de taille de ligne maximum, et 1 Mo est le quota de taille de données statique maximale par partition logique.

Pour calculer la taille totale d'une mise à jour de données statiques et non statiques dans la même instruction, vous pouvez utiliser la formule suivante :

```
(partition key columns + static columns + metadata = total encoded size of static data)
+ (partition key columns + clustering columns + regular columns + row metadata = total
encoded size of row)
= total encoded size of data written
```

Prenons l'exemple suivant d'une table où toutes les colonnes sont de type entier. La table comporte deux colonnes de clé de partition, deux colonnes de clustering, une colonne normale et une colonne statique.

```
CREATE TABLE mykeyspace.mytable(pk_col1 int, pk_col2 int, ck_col1 int, ck_col2
int, reg_col1 int, static_col1 int static, primary key((pk_col1, pk_col2),ck_col1,
ck_col2));
```

Dans cet exemple, nous calculons la taille des données lorsque nous écrivons une ligne dans le tableau, comme indiqué dans l'instruction suivante :

```
INSERT INTO mykeyspace.mytable (pk_col1, pk_col2, ck_col1, ck_col2, reg_col1,
static_col1) values(2,3,4,5,6,7);
```

Pour estimer le nombre total d'octets requis par cette opération d'écriture, vous pouvez suivre les étapes suivantes.

1. Calculez la taille totale codée des données statiques comme indiqué précédemment. Dans cet exemple, il s'agit de 122 octets.
2. Ajoutez la taille de la taille totale codée de la ligne en fonction de la mise à jour des données non statiques, en suivant les étapes décrites dans [the section called “Estimer la taille des lignes”](#). Dans cet exemple, la taille totale de la mise à jour des lignes est de 134 octets.

```
122 bytes for static data + 134 bytes for nonstatic data = 256 bytes.
```

Estimez les exigences de capacité et de débit pour les read/write opérations sur des données statiques dans Amazon Keyspaces

Les données statiques sont associées à des partitions logiques dans Cassandra, et non à des lignes individuelles. Les partitions logiques d'Amazon Keyspaces peuvent avoir une taille pratiquement indépendante en s'étendant sur plusieurs partitions de stockage physiques. Par conséquent, les compteurs Amazon Keyspaces écrivent les opérations sur les données statiques et non statiques séparément. En outre, les écritures qui incluent à la fois des données statiques et non statiques nécessitent des opérations sous-jacentes supplémentaires pour assurer la cohérence des données.

Si vous effectuez une opération d'écriture mixte de données statiques et non statiques, cela entraîne deux opérations d'écriture distinctes, l'une pour les données non statiques et l'autre pour les données statiques. Cela s'applique à la fois aux modes de capacité à la demande et aux modes read/write de capacité provisionnée.

L'exemple suivant explique comment estimer les unités de capacité de lecture (RCU) et d'écriture (WCU) requises lorsque vous calculez les exigences de capacité de débit allouée pour les tables d'Amazon Keyspaces comportant des colonnes statiques. Vous pouvez estimer la capacité dont votre table a besoin pour traiter les écritures incluant à la fois des données statiques et non statiques à l'aide de la formule suivante :

```
2 x WCUs required for nonstatic data + 2 x WCUs required for static data
```

Par exemple, si votre application écrit 27 Ko de données par seconde et que chaque écriture inclut 25,5 Ko de données non statiques et 1,5 Ko de données statiques, votre table nécessite 56 WCU (2 x 26 WCU + 2 x 2 WCU).

Amazon Keyspaces mesure les lectures de données statiques et non statiques de la même manière que les lectures de plusieurs lignes. Par conséquent, le prix de lecture de données statiques et non statiques au cours d'une même opération est basé sur la taille agrégée des données traitées pour effectuer la lecture.

Pour savoir comment surveiller les ressources sans serveur avec Amazon CloudWatch, consultez [the section called “Surveillance avec CloudWatch”](#).

Estimer et provisionner la capacité d'une table multirégionale dans Amazon Keyspaces

Vous pouvez configurer la capacité de débit d'une table multirégionale de deux manières :

- On-demand mode capacité, mesuré en unités de demande d'écriture (WRU)
- Mode capacité provisionnée avec mise à l'échelle automatique, mesurée en unités de capacité d'écriture (WCU)

Vous pouvez utiliser le mode capacité provisionnée avec mise à l'échelle automatique ou le mode capacité à la demande pour garantir qu'une table multirégionale dispose d'une capacité suffisante pour effectuer des écritures répliquées pour toutes les tables. Régions AWS

Note

La modification du mode de capacité de la table dans l'une des régions modifie le mode de capacité de toutes les répliques.

Par défaut, Amazon Keyspaces utilise le mode à la demande pour les tables multirégionales. Avec le mode à la demande, vous n'avez pas besoin de spécifier le débit de lecture et d'écriture que vous souhaitez que votre application exécute. Amazon Keyspaces s'adapte instantanément à vos charges de travail lorsqu'elles augmentent ou diminuent pour atteindre un niveau de trafic atteint précédemment. Si le niveau de trafic d'une charge de travail atteint un nouveau pic, Amazon Keyspaces s'adapte rapidement pour s'adapter à la charge de travail.

Si vous choisissez le mode de capacité provisionnée pour une table, vous devez configurer le nombre d'unités de capacité de lecture (RCU) et d'unités de capacité d'écriture (WCU) par seconde dont votre application a besoin.

Pour planifier les besoins en capacité de débit d'une table multirégionale, vous devez d'abord estimer le nombre de WCU nécessaires par seconde pour chaque région. Vous ajoutez ensuite les écritures provenant de toutes les régions dans lesquelles votre table est répliquée, et vous utilisez la somme pour allouer de la capacité à chaque région. Cela est nécessaire car chaque écriture effectuée dans une région doit également être répétée dans chaque région de réplication.

Si la table ne dispose pas d'une capacité suffisante pour gérer les écritures provenant de toutes les régions, des exceptions de capacité se produiront. En outre, les temps d'attente pour la réplication interrégionale augmenteront.

Par exemple, si vous avez une table multirégionale dans laquelle vous attendez 5 écritures par seconde dans l'est des États-Unis (Virginie du Nord), 10 écritures par seconde dans l'est des États-Unis (Ohio) et 5 écritures par seconde en Europe (Irlande), vous devez vous attendre à ce que la table consomme 20 unités WCU dans chaque région : États-Unis Est (Virginie du Nord), États-Unis Est (Ohio) et Europe (Irlande). Cela signifie que dans cet exemple, vous devez provisionner 20 WCU pour chacune des répliques de la table. Vous pouvez surveiller la consommation de capacité de votre table à l'aide d'Amazon CloudWatch. Pour de plus amples informations, veuillez consulter [the section called “Surveillance avec CloudWatch”](#).

Chaque écriture est facturée comme 1 WCU, vous verrez donc un total de 60 WCU facturés dans cet exemple. Pour plus d'informations sur les tarifs, consultez les tarifs [d'Amazon Keyspaces \(pour Apache Cassandra\)](#).

Pour plus d'informations sur la capacité allouée avec le dimensionnement [the section called “Gérez la capacité de débit avec la mise à l'échelle automatique”](#) automatique d'Amazon Keyspaces, consultez.

 Note

Si une table s'exécute en mode capacité allouée avec mise à l'échelle automatique, la capacité d'écriture allouée est autorisée à flotter dans les limites de ces paramètres de dimensionnement automatique pour chaque région.

Estimez la consommation de capacité de lecture et d'écriture avec Amazon CloudWatch dans Amazon Keyspaces

Pour estimer et surveiller la consommation de capacité de lecture et d'écriture, vous pouvez utiliser un CloudWatch tableau de bord. Pour plus d'informations sur les statistiques disponibles pour Amazon Keyspaces, consultez [the section called “Dimensions et métriques”](#)

Pour surveiller les unités de capacité de lecture et d'écriture consommées par une instruction spécifique avec CloudWatch, vous pouvez suivre ces étapes.

1. Création d'une nouvelle table avec des exemples de données
2. Configurez un tableau de bord Amazon Keyspaces pour le CloudWatch tableau. Pour commencer, vous pouvez utiliser un modèle de tableau de bord disponible sur [Github](#).
3. Exécutez l'instruction CQL, par exemple à l'aide de l'ALLOW FILTERING option, et vérifiez les unités de capacité de lecture consommées pour l'analyse complète du tableau dans le tableau de bord.

Configurer les modes de read/write capacité dans Amazon Keyspaces

Amazon Keyspaces propose deux modes de read/write capacité pour traiter les lectures et les écritures sur vos tables :

- On-demand (par défaut)
- Alloué

Le mode read/write de capacité que vous choisissiez contrôle la façon dont vous êtes facturé pour le débit de lecture et d'écriture et la façon dont la capacité de débit des tables est gérée.

Rubriques

- [Configuration du mode capacité à la demande](#)
- [Configuration du mode de capacité provisionnée](#)
- [Afficher le mode de capacité d'une table dans Amazon Keyspaces](#)
- [Modifier le mode de capacité d'une table dans Amazon Keyspaces](#)
- [Configuration du préchauffage des tables dans Amazon Keyspaces](#)

Configuration du mode capacité à la demande

Le mode de capacité à la demande d'Amazon Keyspaces (pour Apache Cassandra) est une option de facturation flexible capable de traiter des milliers de demandes par seconde sans planification des capacités. Cette option offre une tarification de « paiement par demande » où vous ne payez que les demandes de lecture et d'écriture que vous exécutez.

Lorsque vous choisissez le mode à la demande, Amazon Keyspaces peut augmenter instantanément la capacité de débit de votre table jusqu'à n'importe quel niveau de trafic précédemment atteint, puis redescendre à la baisse lorsque le trafic des applications diminue. Si le niveau de trafic d'une charge globale atteint un nouveau pic, le service s'adapte rapidement pour augmenter la capacité de débit de votre table. Vous pouvez activer le mode de capacité à la demande pour les tables nouvelles et existantes.

On-demand le mode est une bonne option si l'une des conditions suivantes est vraie :

- Vous créez de nouvelles tables avec des charges de travail inconnues.
- Vous avez un trafic imprévisible au niveau de l'application.
- Vous préférez ne payer qu'à l'utilisation.

Pour démarrer avec le mode à la demande, vous pouvez créer une nouvelle table ou mettre à jour une table existante pour utiliser le mode capacité à la demande à l'aide de la console ou avec quelques lignes de code CQL (Cassandra Query Language). Pour de plus amples informations, veuillez consulter [the section called "Tables"](#).

Rubriques

- [Unités de demande de lecture et unités de demande d'écriture](#)
- [Trafic de pointe et propriétés de scalabilité](#)

- [Débit initial pour le mode de capacité à la demande](#)

Unités de demande de lecture et unités de demande d'écriture

Avec les tables du mode de capacité à la demande, vous n'avez pas besoin de spécifier à l'avance le débit de lecture et d'écriture que vous comptez utiliser par votre application. Amazon Keyspaces vous facture les lectures et les écritures que vous effectuez sur vos tables en termes d'unités de demande de lecture (RRU) et d'unités de demande d'écriture (WRU).

- Une RRU représente une LOCAL_QUORUM ou deux demandes de LOCAL_ONE lecture pour une ligne d'une taille maximale de 4 Ko. Si vous devez lire une ligne de plus de 4 Ko, l'opération de lecture utilise des RRU supplémentaires. Le nombre total de RRU requis dépend de la taille de la ligne et de l'utilisation ou pas de la cohérence de lecture LOCAL_QUORUM ou LOCAL_ONE. Par exemple, la lecture d'une ligne de 8 Ko nécessite 2 RRU utilisant la cohérence de lecture LOCAL_QUORUM et 1 RRU si vous choisissez la cohérence de lecture LOCAL_ONE.
- Un WRU représente une écriture pour une ligne d'une taille maximale de 1 Ko. Toutes les écritures utilisent la cohérence LOCAL_QUORUM et il n'y a pas de frais supplémentaires pour l'utilisation de transactions légères (LWT). Si vous devez écrire une ligne supérieure à 1 Ko, l'opération d'écriture utilise des WRU supplémentaires. Le nombre total de WRU requis dépend de la taille de la ligne. Par exemple, si la taille de votre ligne est de 2 Ko, vous avez besoin de 2 WRU pour effectuer une requête d'écriture.

Pour de plus amples informations sur les niveaux de cohérence pris en charge, veuillez consulter [the section called “Niveaux de cohérence Cassandra pris en charge”](#).

Trafic de pointe et propriétés de scalabilité

Les tables Amazon Keyspaces qui utilisent le mode capacité à la demande s'adaptent automatiquement au volume de trafic de votre application. On-demand le mode capacité permet de gérer instantanément jusqu'au double du pic de trafic précédent sur une table. Par exemple, le modèle de trafic de votre application peut varier entre 5 000 et 10 000 lectures de LOCAL_QUORUM par seconde, 10 000 lectures par seconde étant le pic de trafic précédent.

Avec ce modèle, le mode de capacité à la demande prend instantanément en charge un trafic soutenu pouvant atteindre 20 000 lectures par seconde. Si votre application présente un trafic de 20 000 lectures par seconde, ce pic devient votre nouvelle capacité de pointe précédente, permettant ainsi au trafic futur d'atteindre jusqu'à 40 000 lectures par seconde.

Si vous avez besoin de plus du double de votre pic précédent sur une table, Amazon Keyspaces alloue automatiquement plus de capacité à mesure que votre volume de trafic augmente. Cela permet de s'assurer que votre table dispose d'une capacité de débit suffisante pour traiter les demandes supplémentaires. Toutefois, vous pouvez observer des erreurs de capacité de débit insuffisante si vous dépassez le double de votre pic précédent en 30 minutes.

Par exemple, supposons que le modèle de trafic de votre application varie entre 5 000 et 10 000 lectures fortement cohérentes par seconde, où 20 000 lectures par seconde correspondent au pic de trafic atteint précédemment. Dans ce cas, le service vous recommande d'espacer votre croissance de trafic sur au moins 30 minutes avant de conduire jusqu'à 40 000 lectures par seconde.

Pour savoir comment estimer la consommation de capacité de lecture et d'écriture d'une table, voir [the section called “Estimation de la consommation de capacité”](#).

Pour en savoir plus sur les quotas par défaut de votre compte et sur la façon de les augmenter, reportez-vous à la section [Quotas](#).

Débit initial pour le mode de capacité à la demande

Si vous créez une table avec un mode de capacité à la demande ou basculez une table existante vers le mode de capacité à la demande pour la première fois, la table possède les paramètres suivants du trafic de pointe précédent, même si la table n'a pas encore opéré de trafic en mode de capacité à la demande :

- Table nouvellement créée avec mode capacité à la demande : le pic précédent était de 2 000 WRU et 6 000 RRU. Vous pouvez poursuivre jusqu'à doubler le pic précédent immédiatement. Cela permet aux tables à la demande nouvellement créées de servir jusqu'à 4 000 WRU et 12 000 RRU.
- Table existante basculée vers le mode de capacité à la demande : le trafic de pointe précédent équivaut à la moitié des unités de capacité en écriture (WCU) et des unités de capacité en lecture (RCU) allouées à la table ou aux paramètres pour une table nouvellement créée avec le mode de capacité à la demande, selon la valeur la plus élevée.

Configuration du mode de capacité provisionnée

Si vous choisissez le mode de capacité de débit alloué vous spécifiez le nombre de lectures et d'écritures par seconde requis pour votre application. Cela vous permet de gérer votre utilisation d'Amazon Keyspaces afin de rester au niveau ou en dessous d'un taux de demandes défini afin de garantir la prévisibilité. Pour en savoir plus sur la mise à l'échelle automatique du débit alloué,

reportez-vous à la section [the section called “Gérez la capacité de débit avec la mise à l'échelle automatique”](#).

Le mode de capacité de débit alloué est une bonne option si l'une des conditions suivantes est vraie :

- Votre application présente un niveau de trafic prévisible.
- Vous exécutez des applications dont le trafic est constant ou évolue progressivement.
- Vous pouvez prévoir les besoins en capacité.

Unités de capacité en lecture et unités de capacité en écriture

Pour les tables en mode alloué, vous spécifiez votre capacité de débit en termes d'unités de capacité en lecture (RCU) et d'unités de capacité en écriture (WCU) :

- Un RCU représente une lecture de LOCAL_QUORUM par seconde ou deux lectures de LOCAL_ONE par seconde, pour une ligne d'une taille maximale de 4 Ko. Si vous devez lire une ligne de plus de 4 Ko, l'opération de lecture utilise des RCU supplémentaires.

Le nombre total de RCU requis dépend de la taille de la ligne et du nombre de lecture de LOCAL_ONE ou de LOCAL_QUORUM souhaitées. Par exemple, si la taille de votre ligne est de 8 Ko, vous avez besoin de 2 RCU pour supporter une lecture de LOCAL_QUORUM par seconde et d'1 RCU si vous choisissez les lectures de LOCAL_ONE.

- Une WCU représente une écriture par seconde pour une ligne d'une taille maximale de 1 Ko. Toutes les écritures utilisent la cohérence LOCAL_QUORUM et il n'y a pas de frais supplémentaires pour l'utilisation de transactions légères (LWT). Si vous devez écrire une ligne supérieure à 1 Ko, l'opération d'écriture utilise des WCU supplémentaires.

Le nombre total de WCU requis dépend de la taille de la ligne. Par exemple, si la taille de votre ligne est de 2 Ko, vous avez besoin de 2 WCU pour supporter une demande d'écriture par seconde. Pour plus d'informations sur la façon d'estimer la consommation de capacité de lecture et d'écriture d'une table, consultez [the section called “Estimation de la consommation de capacité”](#).

Si votre application lit ou écrit des lignes plus grandes (jusqu'à la taille de ligne maximale d'Amazon Keyspaces de 1 Mo), elle consomme davantage d'unités de capacité. Pour en savoir plus sur l'estimation de la taille des lignes, consultez [the section called “Estimer la taille des lignes”](#). Par exemple, supposons que vous créiez une table allouée avec 6 RCU et 6 WCU. Avec ces paramètres, votre application pourrait effectuer les opérations suivantes :

- Effectuez des LOCAL_QUORUM lectures allant jusqu'à 24 Ko par seconde (4 Ko × 6 RCU).
- Effectuez des lectures de LOCAL_ONE allant jusqu'à 48 Ko par seconde (deux fois plus de débit de lecture).
- Écrivez jusqu'à 6 Ko par seconde (1 Ko × 6 WCU).

Le Débit alloué est la quantité maximum de capacité qu'une application peut consommer à partir d'une table. Si votre application dépasse votre capacité de débit alloué, vous pouvez observer des erreurs de capacité insuffisante.

Par exemple, une demande de lecture dont la capacité de débit est insuffisante échoue avec une `Read_Timeout` exception et est publiée dans la `ReadThrottleEvents` métrique. Une demande d'écriture dont la capacité de débit est insuffisante échoue avec une `Write_Timeout` exception et est publiée dans la `WriteThrottleEvents` métrique.

Vous pouvez utiliser Amazon CloudWatch pour surveiller vos indicateurs de débit provisionnés et réels ainsi que les événements de capacité insuffisante. Pour plus d'informations sur ces métriques, consultez [the section called “Dimensions et métriques”](#).

Note

Des erreurs répétées dues à une capacité insuffisante peuvent entraîner des exceptions spécifiques au pilote côté client, par exemple si le pilote DataStax Java échoue avec un `NoHostAvailableException`

Pour modifier les paramètres de capacité de débit des tables, vous pouvez utiliser l'instruction Console de gestion AWS ou l'`ALTER TABLE` instruction à l'aide de CQL. Pour plus d'informations, voir [the section called “ALTER TABLE”](#)

Pour en savoir plus sur les quotas par défaut de votre compte et sur la façon de les augmenter, reportez-vous à la section [Quotas](#).

Afficher le mode de capacité d'une table dans Amazon Keyspaces

Vous pouvez interroger la table système dans l'espace de touches du système Amazon Keyspaces pour consulter les informations relatives au mode capacité relatives à une table. Vous pouvez également voir si une table utilise le mode de capacité de débit à la demande ou allouée. Si la table

est configurée avec le mode de capacité de débit alloué, vous pouvez voir la capacité de débit alloué pour la table.

Vous pouvez également utiliser le AWS CLI pour afficher le mode de capacité d'une table.

Pour modifier le débit provisionné d'une table, consultez [the section called “Changer le mode de capacité”](#)

Cassandra Query Language (CQL)

Exemple

```
SELECT * from system_schema.tables where keyspace_name = 'mykeyspace' and  
table_name = 'mytable';
```

Une table configurée avec le mode de capacité à la demande renvoie ce qui suit.

```
{  
  "capacity_mode":{  
    "last_update_to_pay_per_request_timestamp":"1579551547603",  
    "throughput_mode":"PAY_PER_REQUEST"  
  }  
}
```

Une table configurée avec le mode de capacité de débit alloué renvoie ce qui suit.

```
{  
  "capacity_mode":{  
    "last_update_to_pay_per_request_timestamp":"1579048006000",  
    "read_capacity_units":"5000",  
    "throughput_mode":"PROVISIONED",  
    "write_capacity_units":"6000"  
  }  
}
```

La valeur `last_update_to_pay_per_request_timestamp` est mesurée en millisecondes.

CLI

Affichez le mode de capacité de débit d'une table à l'aide du AWS CLI

```
aws keyspaces get-table --keyspace-name myKeyspace --table-name myTable
```

La sortie de la commande peut ressembler à ceci pour une table en mode capacité provisionnée.

```
"capacitySpecification": {
  "throughputMode": "PROVISIONED",
  "readCapacityUnits": 4000,
  "writeCapacityUnits": 2000
}
```

La sortie d'une table en mode à la demande ressemble à ceci.

```
"capacitySpecification": {
  "throughputMode": "PAY_PER_REQUEST",
  "lastUpdateToPayPerRequestTimestamp": "2024-10-03T10:48:19.092000+00:00"
}
```

Modifier le mode de capacité d'une table dans Amazon Keyspaces

Lorsque vous passez d'une table du mode capacité allouée au mode capacité à la demande, Amazon Keyspaces apporte plusieurs modifications à la structure de votre table et de vos partitions. Ce processus peut prendre plusieurs minutes. Pendant la période de commutation, votre table fournit un débit compatible avec les montants WCU et RCU précédemment alloués.

Si vous passez du mode de capacité à la demande au mode de capacité allouée, votre table fournit un débit correspondant au trafic de pointe précédent atteint lorsque la table était en mode de capacité à la demande.

Les périodes d'attente suivantes s'appliquent lorsque vous changez de mode de capacité :

- Vous pouvez à tout moment faire passer une table nouvellement créée en mode à la demande en mode capacité provisionnée. Cependant, vous ne pouvez revenir en mode à la demande que 24 heures après l'horodatage de création de la table.
- Vous pouvez à tout moment faire passer une table existante en mode à la demande en mode capacité provisionnée. Toutefois, vous ne pouvez passer du mode provisionné au mode à la demande qu'une seule fois par période de 24 heures.

Cassandra Query Language (CQL)

Modifier le mode de capacité de débit d'une table à l'aide de CQL

1. Pour modifier le mode de capacité d'une table, PROVISIONED vous devez configurer les unités de capacité de lecture et de capacité d'écriture en fonction des valeurs maximales attendues de votre charge de travail. L'énoncé suivant en est un exemple. Vous pouvez également exécuter cette instruction pour ajuster la capacité de lecture ou les unités de capacité d'écriture de la table.

```
ALTER TABLE catalog.book_awards WITH CUSTOM_PROPERTIES={'capacity_mode':  
{'throughput_mode': 'PROVISIONED', 'read_capacity_units': 6000,  
'write_capacity_units': 3000}};
```

Pour configurer le mode de capacité provisionnée avec auto-scaling, consultez [the section called “Configurer le dimensionnement automatique sur une table existante”](#)

2. Pour passer du mode de capacité d'une table au mode à la demande, définissez le mode de débit sur PAY_PER_REQUEST. La déclaration suivante en est un exemple.

```
ALTER TABLE catalog.book_awards WITH CUSTOM_PROPERTIES={'capacity_mode':  
{'throughput_mode': 'PAY_PER_REQUEST'}};
```

3. Vous pouvez utiliser l'instruction suivante pour confirmer le mode de capacité de la table.

```
SELECT * from system_schema.tables where keyspace_name = 'catalog' and  
table_name = 'book_awards';
```

Une table configurée avec le mode de capacité à la demande renvoie ce qui suit.

```
{  
  "capacity_mode":{  
    "last_update_to_pay_per_request_timestamp":"1727952499092",  
    "throughput_mode":"PAY_PER_REQUEST"  
  }  
}
```

La valeur `last_update_to_pay_per_request_timestamp` est mesurée en millisecondes.

CLI

Modifiez le mode de capacité de débit d'une table à l'aide du AWS CLI

1. Pour changer le mode de capacité de la table, PROVISIONED vous devez configurer les unités de capacité de lecture et de capacité d'écriture en fonction des valeurs maximales attendues de votre charge de travail. La commande suivante en est un exemple. Vous pouvez également exécuter cette commande pour ajuster la capacité de lecture ou les unités de capacité d'écriture de la table.

```
aws keyspaces update-table --keyspace-name catalog --table-name book_awards
                           \--capacity-specification
                           throughputMode=PROVISIONED,readCapacityUnits=6000,writeCapacityUnits=3000
```

Pour configurer le mode de capacité provisionnée avec auto-scaling, consultez [the section called “Configurer le dimensionnement automatique sur une table existante”](#)

2. Pour passer du mode de capacité d'une table au mode à la demande, vous devez définir le mode de débit sur PAY_PER_REQUEST. La déclaration suivante en est un exemple.

```
aws keyspaces update-table --keyspace-name catalog --table-name book_awards
                           \--capacity-specification
                           throughputMode=PAY_PER_REQUEST
```

3. Vous pouvez utiliser la commande suivante pour vérifier le mode de capacité configuré pour une table.

```
aws keyspaces get-table --keyspace-name catalog --table-name book_awards
```

La sortie d'une table en mode à la demande ressemble à ceci.

```
"capacitySpecification": {
  "throughputMode": "PAY_PER_REQUEST",
  "lastUpdateToPayPerRequestTimestamp": "2024-10-03T10:48:19.092000+00:00"
}
```

Configuration du préchauffage des tables dans Amazon Keyspaces

Amazon Keyspaces adapte automatiquement les partitions de stockage en fonction du débit à la demande ou provisionné, mais en cas de nouvelles tables ou de pics de débit soudains, l'allocation des partitions de stockage requises peut prendre plus de temps. Pour garantir qu'une table nouvelle ou existante dispose d'une capacité suffisante pour supporter le pic de débit prévu, vous pouvez définir manuellement des valeurs de débit de chauffage spécifiques pour préchauffer votre table.

Le débit chaud fait référence au nombre d'opérations de lecture et d'écriture que votre table Amazon Keyspaces peut prendre en charge instantanément. Ces valeurs sont disponibles par défaut pour toutes les tables nouvelles et existantes. Si vous utilisez le mode à la demande ou si vous mettez à jour votre débit provisionné, Amazon Keyspaces garantit que votre application est en mesure d'émettre des demandes allant jusqu'à ces valeurs instantanément.

Amazon Keyspaces ajuste automatiquement les valeurs de débit à chaud à mesure que votre utilisation augmente. Pour ajuster la capacité de débit en fonction des pics à venir, par exemple lorsque vous migrez des données depuis une autre base de données, ce qui peut nécessiter le chargement de téraoctets de données en peu de temps, vous pouvez augmenter manuellement les valeurs de débit à chaud de vos tables. Cela est utile pour les pics planifiés où les taux de demandes peuvent augmenter de 10, 100 fois ou plus. Tout d'abord, déterminez si le débit chaud actuel est suffisant pour gérer le trafic attendu. [Ensuite, si vous devez préchauffer la table en fonction de la charge de travail maximale prévue, vous pouvez augmenter manuellement la valeur du débit à chaud sans modifier vos paramètres de débit ou votre mode de capacité.](#)

Vous pouvez préchauffer les tables pour les opérations de lecture, les opérations d'écriture ou les deux. Vous pouvez augmenter cette valeur pour les tables à région unique et les tables multirégions nouvelles et existantes, et les paramètres de débit à chaud que vous définissez s'appliquent automatiquement à toutes les répliques des tables multirégions. Il n'y a pas de limite au nombre de tables Amazon Keyspaces que vous pouvez préchauffer à tout moment. Le temps nécessaire pour terminer le préchauffage dépend des valeurs que vous avez définies et de la taille de la table. Vous pouvez soumettre des demandes de préchauffage simultanées et ces demandes n'interfèrent avec aucune opération de table. Vous pouvez préchauffer votre table jusqu'à la limite de quota de table pour votre compte dans cette région. Utilisez la [console Service Quotas](#) pour vérifier vos quotas actuels et les augmenter si nécessaire.

Les valeurs de débit à chaud qu'Amazon Keyspaces ajuste en fonction de votre utilisation à la demande ou de la capacité allouée sont disponibles par défaut pour toutes les tables sans frais supplémentaires. Toutefois, si vous augmentez manuellement les valeurs de débit de préchauffage

par défaut pour préchauffer les tables en cas de pic de trafic, des frais supplémentaires s'appliquent. Pour plus d'informations, consultez les [tarifs d'Amazon Keyspaces](#).

Voici quelques scénarios et bonnes pratiques que vous pourriez prendre en compte lors du préchauffage des tables Amazon Keyspaces.

Débit chaud et modèles d'accès inégaux

Une table peut avoir un débit élevé de 30 000 unités de lecture par seconde et de 10 000 unités d'écriture par seconde, mais vous pouvez tout de même rencontrer des événements de dépassement de capacité lors des lectures ou des écritures avant d'atteindre ces valeurs. Cela est probablement dû à une partition chaude. Même si Amazon Keyspaces peut continuer à évoluer pour prendre en charge un débit pratiquement illimité, chaque partition individuelle est limitée à 1 000 unités d'écriture par seconde et à 3 000 unités de lecture par seconde. Si votre application génère trop de trafic vers une petite partie des partitions de la table, des événements de dépassement de capacité peuvent se produire avant même que vous n'atteigniez les valeurs de débit maximal de la table. Nous vous recommandons de suivre les [bonnes pratiques d'Amazon Keyspaces](#) afin de garantir une évolutivité sans faille et d'éviter les partitions chaudes.

Débit chaud pour une table provisionnée

Prenons l'exemple d'une table provisionnée qui a un débit chaud de 30 000 unités de lecture par seconde et de 10 000 unités d'écriture par seconde, mais qui a actuellement un débit provisionné de 4 000 RCU et 8 000 WCU. Vous pouvez instantanément augmenter le débit provisionné de la table jusqu'à 30 000 RCU ou 10 000 WCU en mettant à jour vos paramètres de débit provisionné. Lorsque vous augmentez le débit provisionné au-delà de ces valeurs, le débit chaud s'ajuste automatiquement aux nouvelles valeurs supérieures, car vous avez établi un nouveau débit de pointe. Par exemple, si vous définissez le débit provisionné à 50 000 RCU, le débit à chaud augmente à 50 000 unités de lecture par seconde.

```
"ProvisionedThroughput":
{
  "ReadCapacityUnits": 4000,
  "WriteCapacityUnits": 8000
}
"WarmThroughput":
{
  "ReadUnitsPerSecond": 30000,
  "WriteUnitsPerSecond": 10000
}
```

Débit chaud pour une table à la demande

Une nouvelle table à la demande utilise au début un débit chaud de 12 000 unités de lecture par seconde et de 4 000 unités d'écriture par seconde. Votre table peut instantanément accueillir un trafic soutenu jusqu'à ces niveaux. Lorsque vos demandes dépassent 12 000 unités de lecture par seconde ou 4 000 unités d'écriture par seconde, le débit chaud s'ajuste automatiquement aux valeurs les plus élevées.

```
"WarmThroughput":
{
  "ReadUnitsPerSecond": 12000,
  "WriteUnitsPerSecond": 4000
}
```

Bonnes pratiques pour préchauffer les tables Amazon Keyspaces

Suivez ces bonnes pratiques lors de la mise en œuvre du préchauffage pour vos tables Amazon Keyspaces :

Estimez avec précision la capacité requise

Comme le préchauffage entraîne un coût unique, calculez soigneusement le débit nécessaire en fonction de la charge de travail attendue afin d'éviter le surprovisionnement.

Tenez compte du schéma de la table

Les tables comportant des lignes plus grandes peuvent nécessiter davantage de partitions pour le même débit. Tenez compte de la taille moyenne de vos rangées lors de l'estimation des besoins de préchauffage.

Surveillez les performances des tables

Après le préchauffage, utilisez CloudWatch des métriques pour vérifier que votre table supporte la charge comme prévu. Pour de plus amples informations, veuillez consulter [the section called “Préchauffage du moniteur avec CloudWatch”](#).

Gérer les quotas

Si votre application nécessite un débit supérieur à celui autorisé par les quotas par défaut (40 000 RCUs/WCUs ou 2 000 partitions), le quota de demandes augmente bien avant l'événement à fort trafic. Pour demander une augmentation d'un quota, vous pouvez utiliser la [console Service Quotas](#).

Optimisez les coûts

Pour les événements temporaires à fort trafic, pensez à utiliser le préchauffage plutôt que de passer en mode provisionné à haute capacité, car cela peut être plus rentable pour les événements de courte durée. Pour plus d'informations sur les tarifs, consultez les tarifs [d'Amazon Keyspaces](#).

Note

Surveillez les indicateurs de performance de votre application pendant la phase de test pour vérifier que votre configuration de préchauffage répond correctement aux exigences de votre charge de travail.

Rubriques

- [Création d'une nouvelle table Amazon Keyspaces avec un débit chaud supérieur](#)
- [Augmentation du débit chaud de votre table Amazon Keyspaces existante](#)
- [Afficher le débit à chaud d'une table Amazon Keyspaces](#)
- [Surveillez les performances d'une table préchauffée à l'aide d'Amazon CloudWatch](#)

Création d'une nouvelle table Amazon Keyspaces avec un débit chaud supérieur

Vous pouvez ajuster les valeurs du débit chaud lorsque vous créez votre table Amazon Keyspaces à l'aide de la console, du CQL ou du AWS CLI

Console

Comment créer une nouvelle table avec des paramètres de débit à chaud

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le panneau de navigation, choisissez Tables, puis Create table (Créer une table).
3. Sur la page Créer une table, dans la section Détails de la table, sélectionnez un espace de touche et attribuez un nom à la nouvelle table.
4. Dans la section Colonnes, créez le schéma de votre table.

5. Dans la section Clé primaire, définissez la clé primaire de la table et sélectionnez les colonnes de clustering facultatives.
6. Dans la section Paramètres du tableau, choisissez Personnaliser les paramètres.
7. Passez aux paramètres Read/write de capacité.
8. Pour le mode Capacité, vous pouvez choisir entre le mode Provisioned On-demand ou le mode Provisioned.
9. Dans la section Pre-warming pour les tables, vous pouvez augmenter les valeurs des unités de lecture par seconde et des unités d'écriture par seconde selon les besoins pour préparer votre table à gérer les pics planifiés.

Les valeurs de débit à chaud qu'Amazon Keyspaces ajuste en fonction de votre utilisation à la demande ou de la capacité allouée sont disponibles par défaut pour toutes les tables sans frais supplémentaires. Notez que si vous augmentez manuellement les valeurs de débit de chauffage par défaut pour préchauffer la table en cas de pic de trafic, des frais supplémentaires s'appliquent.

10. Configurez les autres fonctionnalités facultatives du tableau selon vos besoins. Choisissez ensuite Créer une table.

Cassandra Query Language (CQL)

- Créez une table avec un débit à chaud à l'aide de l'une des méthodes suivantes :
 - Pour le mode provisionné, créez une table et spécifiez la capacité maximale attendue pour les lectures et les écritures à l'aide de la syntaxe CQL suivante :

```
CREATE TABLE catalog.book_awards (  
    year int,  
    award text,  
    rank int,  
    category text,  
    book_title text,  
    author text,  
    publisher text,  
    PRIMARY KEY ((year, award), category, rank))  
WITH CUSTOM_PROPERTIES = {  
    'capacity_mode': {  
        'throughput_mode': 'PROVISIONED',  
        'read_capacity_units': 20000,  
        'write_capacity_units': 10000
```

```
    },  
    'warm_throughput': {  
        'read_units_per_second': 40000,  
        'write_units_per_second': 20000  
    }  
};
```

- Pour le mode à la demande, créez une table et spécifiez la capacité maximale attendue pour les lectures et les écritures à l'aide de la syntaxe CQL suivante :

```
CREATE TABLE catalog.book_awards (  
    year int,  
    award text,  
    rank int,  
    category text,  
    book_title text,  
    author text,  
    publisher text,  
    PRIMARY KEY ((year, award), category, rank))  
WITH CUSTOM_PROPERTIES = {  
    'capacity_mode': {  
        'throughput_mode': 'PAY_PER_REQUEST'  
    },  
    'warm_throughput': {  
        'read_units_per_second': 40000,  
        'write_units_per_second': 20000  
    }  
};
```

Pour confirmer les paramètres de capacité du tableau, voir [the section called “Afficher le débit de chaleur de votre table”](#).

CLI

1. Créez une table avec un débit à chaud en utilisant l'une des méthodes suivantes à l'aide du AWS CLI
 - Créez une nouvelle table en mode provisionné et spécifiez les valeurs de capacité maximale attendues pour les lectures et les écritures pour la nouvelle table. La déclaration suivante en est un exemple.

```
aws keyspaces create-table \
```

```
--keyspace-name 'catalog' \
--table-name 'book_awards' \
--schema-definition 'allColumns=[{name=year,type=int},
{name=award,type=text},{name=rank,type=int},{name=category,type=text},
{name=book_title,type=text},{name=author,type=text},
{name=publisher,type=text}],partitionKeys=[{name=year},
{name=award}],clusteringKeys=[{name=category,orderBy=ASC},
{name=rank,orderBy=ASC}]' \
--capacity-specification
  throughputMode=PROVISIONED,readCapacityUnits=20000,writeCapacityUnits=10000
\
--warm-throughput-specification
  readUnitsPerSecond=40000,writeUnitsPerSecond=20000
```

- Créez une nouvelle table en mode à la demande et spécifiez les valeurs de capacité maximale attendues pour les lectures et les écritures pour la nouvelle table. La déclaration suivante en est un exemple.

```
aws keyspaces create-table \
--keyspace-name 'catalog' \
--table-name 'book_awards' \
--schema-definition 'allColumns=[{name=year,type=int},
{name=award,type=text},{name=rank,type=int},{name=category,type=text},
{name=book_title,type=text},{name=author,type=text},
{name=publisher,type=text}],partitionKeys=[{name=year},
{name=award}],clusteringKeys=[{name=category,orderBy=ASC},
{name=rank,orderBy=ASC}]' \
--warmThroughputSpecification
  readUnitsPerSecond=40000,writeUnitsPerSecond=20000
```

2. La sortie de la commande renvoie l'ARN de la table, comme indiqué dans l'exemple suivant.

```
{
  "resourceArn": "arn:aws::cassandra:us-east-1:111122223333:/keyspace/catalog/
table/book_awards>"
}
```

Pour confirmer les paramètres de capacité du tableau, voir [the section called “Afficher le débit de chaleur de votre table”](#).

Java

Créez une nouvelle table à l'aide du SDK for Java.

- Créez une nouvelle table en mode provisionné et spécifiez les valeurs de capacité maximale attendues pour les lectures et les écritures pour la nouvelle table. L'exemple de code suivant en est un exemple.

```
import software.amazon.awssdk.services.keyspaces.KeyspacesClient;
import software.amazon.awssdk.services.keyspaces.model.*;

public class PreWarmingExample {
    public static void main(String[] args) {
        KeyspacesClient keyspacesClient = KeyspacesClient.builder().build();

        // Define schema
        List<ColumnDefinition> columns = Arrays.asList(
            ColumnDefinition.builder().name("year").type("int").build(),
            ColumnDefinition.builder().name("award").type("text").build(),
            ColumnDefinition.builder().name("rank").type("int").build(),
            ColumnDefinition.builder().name("category").type("text").build(),
            ColumnDefinition.builder().name("book_title").type("text").build(),
            ColumnDefinition.builder().name("author").type("text").build(),
            ColumnDefinition.builder().name("publisher").type("text").build()
        );

        List<PartitionKey> partitionKeys = Arrays.asList(
            PartitionKey.builder().name("year").build(),
            PartitionKey.builder().name("award").build()
        );

        List<ClusteringKey> clusteringKeys = Arrays.asList(
            ClusteringKey.builder().name("category").orderBy("ASC").build(),
            ClusteringKey.builder().name("rank").orderBy("ASC").build()
        );

        SchemaDefinition schema = SchemaDefinition.builder()
            .allColumns(columns)
            .partitionKeys(partitionKeys)
            .clusteringKeys(clusteringKeys)
            .build();

        // Define capacity specification
```

```

        CapacitySpecification capacitySpec = CapacitySpecification.builder()
            .throughputMode(ThroughputMode.PROVISIONED)
            .readCapacityUnits(20000)
            .writeCapacityUnits(10000)
            .build();

        // Define warm throughput specification
        WarmThroughputSpecification warmThroughput =
        WarmThroughputSpecification.builder()
            .readUnitsPerSecond(40000L)
            .writeUnitsPerSecond(20000L)
            .build();

        // Create table with PreWarming
        CreateTableRequest request = CreateTableRequest.builder()
            .keyspaceName("catalog")
            .tableName("book_awards")
            .schemaDefinition(schema)
            .capacitySpecification(capacitySpec)
            .warmThroughputSpecification(warmThroughput)
            .build();

        CreateTableResponse response = keyspacesClient.createTable(request);
        System.out.println("Table created with ARN: " + response.resourceArn());
    }
}

```

Augmentation du débit chaud de votre table Amazon Keyspaces existante

Vous pouvez augmenter les valeurs de débit à chaud actuelles de votre table Amazon Keyspaces à l'aide de la console, du CQL ou du AWS CLI

Console

Comment augmenter les paramètres de préchauffage d'une table à l'aide de la console

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le volet de navigation, choisissez Tables, puis choisissez la table que vous souhaitez mettre à jour.
3. Dans l'onglet Capacité du tableau, passez à « Pre-warming pour les tableaux ».

4. Dans la section Pre-warming pour les tables, choisissez Modifier.
5. Sur la page Modifier le préchauffage pour les tables, vous pouvez mettre à jour les valeurs des unités de lecture par seconde et des unités d'écriture par seconde.
6. Sélectionnez Enregistrer les modifications. Votre table est en train d'être mise à jour avec les paramètres de préchauffage spécifiés.

Cassandra Query Language (CQL)

Augmenter les paramètres de débit à chaud d'une table à l'aide de CQL

- Utilisez l'ALTER TABLE instruction pour augmenter le débit de chaleur d'une table. La déclaration suivante en est un exemple.

```
ALTER TABLE catalog.book_awards
WITH CUSTOM_PROPERTIES = {
    'warm_throughput': {
        'read_units_per_second': 60000,
        'write_units_per_second': 30000
    }
};
```

Pour confirmer les paramètres de capacité mis à jour du tableau, voir [the section called “Afficher le débit de chaleur de votre table”](#).

CLI

Augmentez les réglages de préchauffage d'une table à l'aide du AWS CLI

- Pour augmenter le débit de chauffage de la table, vous pouvez utiliser la commande. `update-table` La déclaration suivante en est un exemple.

```
aws keyspaces update-table \
--keyspace-name 'catalog' \
--table-name 'book_awards' \
--warmThroughputSpecification readUnitsPerSecond=60000,writeUnitsPerSecond=30000
```

Pour confirmer les paramètres de capacité mis à jour du tableau, voir [the section called “Afficher le débit de chaleur de votre table”](#).

Java

Mettez à jour les paramètres de préchauffage d'une table à l'aide du SDK for Java.

- Mettez à jour les paramètres de débit de chaleur d'une table. L'exemple de code suivant en est un exemple.

```
import software.amazon.awssdk.services.keyspaces.KeyspacesClient;
import software.amazon.awssdk.services.keyspaces.model.*;

public class UpdatePreWarmingExample {
    public static void main(String[] args) {
        KeyspacesClient keyspacesClient = KeyspacesClient.builder().build();

        // Define new warm throughput specification
        WarmThroughputSpecification warmThroughput =
        WarmThroughputSpecification.builder()
            .readUnitsPerSecond(60000L)
            .writeUnitsPerSecond(30000L)
            .build();

        // Update table with new PreWarming settings
        UpdateTableRequest request = UpdateTableRequest.builder()
            .keyspaceName("catalog")
            .tableName("book_awards")
            .warmThroughputSpecification(warmThroughput)
            .build();

        UpdateTableResponse response = keyspacesClient.updateTable(request);
        System.out.println("Table update requested: " + response.resourceArn());
    }
}
```

Afficher le débit à chaud d'une table Amazon Keyspaces

Vous pouvez consulter les valeurs de débit à chaud actuelles de votre table Amazon Keyspaces à l'aide de la console, du CQL ou du AWS CLI

Console

Comment consulter les paramètres de préchauffage de votre table à l'aide de la console

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le volet de navigation, choisissez Tables, puis sélectionnez le tableau que vous souhaitez consulter.
3. Dans l'onglet Capacité du tableau, passez à « Pre-warming pour les tableaux ».

Cassandra Query Language (CQL)

Afficher les paramètres de débit à chaud d'une table à l'aide de CQL

- Pour afficher les paramètres de débit à chaud d'une table, vous pouvez utiliser l'instruction CQL suivante.

```
SELECT custom_properties
FROM system_schema_mcs.tables
WHERE keyspace_name='catalog' and table_name='book_awards';

// Output:
...
custom_properties
-----
{
  'warm_throughput':
  {
    'read_units_per_second': '40000',
    'write_units_per_second': '20000',
    'status': 'AVAILABLE'
  }
}
...
```

CLI

Consultez les paramètres de débit de chaleur d'une table à l'aide du AWS CLI

- Vous pouvez afficher les paramètres de débit à chaud d'une table à l'aide de la `get-table` commande, comme indiqué dans l'exemple suivant.

```
aws keyspaces get-table \  
--keyspace-name 'catalog' \  
--table-name 'book_awards'
```

Voici un exemple de sortie de la `get-table` commande pour une table à région unique en mode provisionné.

```
{  
  "keyspaceName": "catalog",  
  "tableName": "book_awards",  
  ... Existing Fields ...,  
  "capacitySpecificationSummary": {  
    "throughputMode": "PROVISIONED",  
    "readCapacityUnits": 20000,  
    "writeCapacityUnits": 10000  
  },  
  "warmThroughputSpecificationSummary": {  
    "readUnitsPerSecond": 40000,  
    "writeUnitsPerSecond": 20000,  
    "status": "AVAILABLE"  
  }  
}
```

Voici un exemple de sortie pour une table à région unique en mode à la demande.

```
{  
  "keyspaceName": "catalog",  
  "tableName": "book_awards_ondemand",  
  ... Existing Fields ...,  
  "capacitySpecification": {  
    "throughputMode": "PAY_PER_REQUEST"  
  },  
  "warmThroughputSpecificationSummary": {  
    "readUnitsPerSecond": 40000,  
    "writeUnitsPerSecond": 20000,  
    "status": "AVAILABLE"  
  }  
}
```

```
        "writeUnitsPerSecond": 20000,  
        "status": "AVAILABLE"  
    }  
}
```

Java

Lisez les paramètres de préchauffage d'une table à l'aide du SDK for Java.

- Lisez les valeurs de débit à chaud d'une table à l'aide de `get-table`. L'exemple de code suivant en est un exemple.

```
import software.amazon.awssdk.services.keyspaces.KeyspacesClient;  
import software.amazon.awssdk.services.keyspaces.model.*;  
  
public class GetTableWithPreWarmingExample {  
    public static void main(String[] args) {  
        KeyspacesClient keyspacesClient = KeyspacesClient.builder().build();  
  
        // Get table details including PreWarming specification  
        GetTableRequest request = GetTableRequest.builder()  
            .keyspaceName("catalog")  
            .tableName("book_awards")  
            .build();  
  
        GetTableResponse response = keyspacesClient.getTable(request);  
  
        // Access PreWarming details  
        if (response.warmThroughputSpecification() != null) {  
            WarmThroughputSpecificationSummary warmThroughputSummary =  
response.warmThroughputSpecification();  
            System.out.println("PreWarming Status: " +  
warmThroughputSummary.status());  
            System.out.println("Read Units: " +  
warmThroughputSummary.readUnitsPerSecond());  
            System.out.println("Write Units: " +  
warmThroughputSummary.writeUnitsPerSecond());  
  
            // Check if PreWarming is active  
            if (warmThroughputSummary.status().equals("AVAILABLE")) {  
                System.out.println("Table is fully pre-warmed and ready for high  
throughput");  
            }  
        }  
    }  
}
```

```
        } else if (warmThroughputSummary.status().equals("UPDATING")) {  
            System.out.println("Table PreWarming is currently being  
updated");  
        }  
    } else {  
        System.out.println("Table does not have PreWarming enabled");  
    }  
}  
}
```

Surveillez les performances d'une table préchauffée à l'aide d'Amazon CloudWatch

Le préchauffage d'Amazon Keyspaces n'introduit pas de nouvelles CloudWatch statistiques, mais vous pouvez surveiller les performances des tables préchauffées à l'aide des statistiques Amazon Keyspaces existantes :

SuccessfulRequestLatency

Surveillez cette métrique pour vérifier que la table préchauffée traite les demandes avec la latence attendue.

WriteThrottleEvents et ReadThrottleEvents

Ces paramètres doivent rester faibles pour une table correctement préchauffée. Si vous constatez des erreurs de capacité insuffisantes malgré le préchauffage, vous devrez peut-être ajuster vos valeurs de débit de chauffage.

ConsumedReadCapacityUnits et ConsumedWriteCapacityUnits

Ces mesures indiquent la consommation réelle de capacité, ce qui peut aider à valider si votre configuration de préchauffage est appropriée.

ProvisionedReadCapacityUnits et ProvisionedWriteCapacityUnits

Pour les tables provisionnées, ces mesures indiquent la capacité actuellement allouée.

Ces métriques peuvent être consultées dans la CloudWatch console ou demandées à l'aide de l'API. CloudWatch Pour de plus amples informations, veuillez consulter [the section called “Surveillance avec CloudWatch”](#).

Gérez automatiquement la capacité de débit grâce à la mise à l'échelle automatique d'Amazon Keyspaces

De nombreuses charges de travail de base de données sont cycliques par nature et difficiles à prévoir. Prenons l'exemple d'une application de réseau social où la plupart des utilisateurs sont actifs pendant la journée. La base de données doit être capable de gérer l'activité durant cette période, mais elle n'a pas besoin des mêmes niveaux de débit pendant la nuit.

Un autre exemple : supposons que vous ayez une nouvelles application de jeux pour appareils mobiles qui est rapidement adoptée par les utilisateurs. Si le jeu devient trop populaire, les ressources de base de données disponibles risquent d'être dépassées, entraînant un ralentissement des performances et le mécontentement des clients. Ce type de charges de travail nécessite souvent une intervention manuelle pour augmenter ou diminuer les ressources de base de données en fonction de la variation des niveaux d'utilisation.

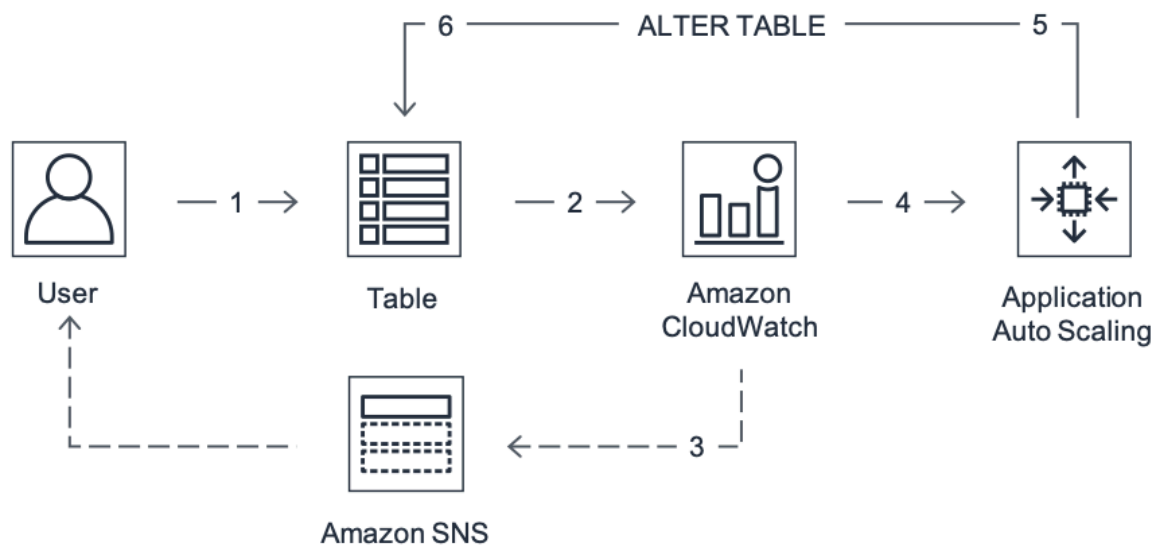
Amazon Keyspaces (pour Apache Cassandra) vous aide à provisionner efficacement la capacité de débit pour des charges de travail variables en ajustant automatiquement la capacité de débit en fonction du trafic applicatif réel. Amazon Keyspaces utilise le service Application Auto Scaling pour augmenter et diminuer la capacité de lecture et d'écriture d'un tableau en votre nom. Pour plus d'informations sur Application Auto Scaling, consultez le Guide de l'utilisateur de [Application Auto Scaling](#).

Note

Pour démarrer rapidement avec la mise à l'échelle automatique d'Amazon Keyspaces, consultez [the section called “Configuration et mise à jour des politiques de dimensionnement automatique”](#).

Comment fonctionne la mise à l'échelle automatique d'Amazon Keyspaces

Le schéma suivant fournit une vue d'ensemble de haut niveau de la manière dont la mise à l'échelle automatique d'Amazon Keyspaces gère la capacité de débit d'une table.



Pour activer la mise à l'échelle automatique d'une table, vous devez créer une stratégie de mise à l'échelle. La stratégie de mise à l'échelle permet de spécifier si vous souhaitez mettre à l'échelle la capacité en lecture ou en écriture (ou les deux), ainsi que les paramètres d'unité de capacité allouée minimum et maximum pour la table.

La stratégie de mise à l'échelle définit également une utilisation cible. L'utilisation cible correspond au ratio unités de capacité consommées/unités de capacité allouées à un point dans le temps, exprimé sous forme de pourcentage. La mise à l'échelle automatique utilise un algorithme de suivi de cible pour ajuster le débit alloué de la table vers le haut ou vers le bas en réponse aux charges de travail réelles. L'utilisation réelle de la capacité reste ainsi proche de votre utilisation cible.

Vous pouvez également définir les valeurs d'utilisation cibles de la mise à l'échelle automatique entre 20 % et 90 % pour votre capacité de lecture et d'écriture. Le taux d'utilisation cible par défaut est de 70 %. Vous pouvez définir l'utilisation cible à un pourcentage inférieur si votre trafic change rapidement et si vous souhaitez que la capacité commence à augmenter plus tôt. Vous pouvez également définir le taux d'utilisation cible sur un taux plus élevé si le trafic de votre application change plus lentement et que vous souhaitez réduire le coût du débit.

Pour plus d'informations sur les politiques de dimensionnement, consultez la section Stratégies de dimensionnement du suivi de [Target pour Application Auto Scaling](#) dans le Guide de l'utilisateur de l'[application Auto Scaling](#).

Lorsque vous créez une politique de dimensionnement, Amazon Keyspaces crée deux paires d'Amazon CloudWatch alarmes Amazon en votre nom. Chaque paire représente vos limites supérieure et

inférieure pour les paramètres de débit alloué. Ces CloudWatch alarmes sont déclenchées lorsque l'utilisation réelle de la table s'écarte de votre utilisation cible pendant une période prolongée. Pour en savoir plus sur Amazon CloudWatch, consultez le guide de CloudWatch l'utilisateur [Amazon](#).

Lorsque l'une des CloudWatch alarmes est déclenchée, Amazon Simple Notification Service (Amazon SNS) vous envoie une notification (si vous l'avez activée). L' CloudWatch alarme appelle ensuite Application Auto Scaling pour évaluer votre politique de dimensionnement. Cela envoie à son tour une demande `Alter Table` à Amazon Keyspaces pour ajuster la capacité provisionnée de la table à la hausse ou à la baisse, selon les besoins. Pour en savoir plus sur les notifications Amazon SNS, consultez la section [Configuration des notifications Amazon SNS](#).

Amazon Keyspaces traite la demande `Alter Table` en augmentant (ou en diminuant) la capacité de débit provisionnée de la table afin qu'elle se rapproche de votre utilisation cible.

Note

Le dimensionnement automatique d'Amazon Keyspaces modifie les paramètres de débit provisionnés uniquement lorsque la charge de travail réelle reste élevée (ou réduite) pendant une période prolongée de plusieurs minutes. L'algorithme de suivi cherche à garder l'utilisation cible au niveau ou proche de la valeur choisie sur le long terme. Les pics soudains de l'activité de lecture sont gérés par la capacité de transmission en mode rafale intégrée de la table.

Fonctionnement de la mise à l'échelle automatique pour les tables multirégions

Pour garantir que la capacité de lecture et d'écriture est toujours suffisante pour toutes les répliques de tables dans l'ensemble Régions AWS d'une table multirégion en mode capacité provisionnée, nous vous recommandons de configurer la mise à l'échelle automatique d'Amazon Keyspaces.

Lorsque vous utilisez une table multirégion en mode provisionné avec mise à l'échelle automatique, vous ne pouvez pas désactiver la mise à l'échelle automatique pour une seule réplique de table. Vous pouvez toutefois ajuster les paramètres de mise à l'échelle automatique de lecture du tableau pour différentes régions. Par exemple, vous pouvez spécifier différents paramètres de capacité de lecture et de mise à l'échelle automatique de lecture pour chaque région dans laquelle la table est répliquée.

Les paramètres de dimensionnement automatique en lecture que vous configurez pour une réplique de table dans une région spécifiée remplacent les paramètres de dimensionnement automatique généraux de la table. La capacité d'écriture doit toutefois rester synchronisée sur toutes les répliques de tables afin de garantir une capacité suffisante pour répliquer les écritures dans toutes les régions.

La mise à l'échelle automatique d'Amazon Keyspaces met à jour indépendamment la capacité provisionnée de la table dans chaque table en Région AWS fonction de l'utilisation dans cette région. Par conséquent, la capacité provisionnée dans chaque région pour une table multirégion peut être différente lorsque la mise à l'échelle automatique est active.

Vous pouvez configurer les paramètres de dimensionnement automatique d'une table multirégion et de ses répliques à l'aide de la console Amazon Keyspaces, de l'API ou du AWS CLI CQL. Pour plus d'informations sur la création et la mise à jour des paramètres de dimensionnement automatique pour les tables multirégions, consultez [the section called “Mettre à jour les paramètres de capacité allouée et de dimensionnement automatique pour une table multirégionale”](#).

Note

Si vous utilisez la mise à l'échelle automatique pour les tables multirégions, vous devez toujours utiliser les opérations de l'API Amazon Keyspaces pour configurer les paramètres de mise à l'échelle automatique. Si vous utilisez directement les opérations de l'API Application Auto Scaling pour configurer les paramètres de dimensionnement automatique, il n'est pas possible Régions AWS de spécifier la table multi-régions. Cela peut entraîner des configurations non prises en charge.

Notes d'utilisation

Avant de commencer à utiliser la mise à l'échelle automatique d'Amazon Keyspaces, vous devez prendre en compte les points suivants :

- La mise à l'échelle automatique d'Amazon Keyspaces n'est pas disponible dans la région Moyen-Orient (Émirats arabes unis).
- Le dimensionnement automatique d'Amazon Keyspaces peut augmenter la capacité de lecture ou la capacité d'écriture aussi souvent que nécessaire, conformément à votre politique de dimensionnement. Tous les quotas Amazon Keyspaces restent en vigueur, comme décrit dans [Quotas](#).

- La mise à l'échelle automatique d'Amazon Keyspaces ne vous empêche pas de modifier manuellement les paramètres de débit provisionnés. Ces ajustements manuels n'affectent aucune CloudWatch alarme existante associée à la politique de dimensionnement.
- Si vous utilisez la console pour créer une table avec une capacité de débit provisionnée, la mise à l'échelle automatique d'Amazon Keyspaces est activée par défaut. Vous pouvez modifier vos paramètres de mise à l'échelle automatique à tout moment. Pour de plus amples informations, veuillez consulter [the section called “Désactiver le dimensionnement automatique d'Amazon Keyspaces pour un tableau”](#).
- Si vous utilisez CloudFormation pour créer des politiques de dimensionnement, vous devez gérer les politiques de dimensionnement de CloudFormation manière à ce que la pile soit synchronisée avec le modèle de pile. Si vous modifiez les politiques de dimensionnement depuis Amazon Keyspaces, elles seront remplacées par les valeurs d'origine du modèle de CloudFormation pile lorsque la pile sera réinitialisée.
- Si vous l'utilisez CloudTrail pour surveiller la mise à l'échelle automatique d'Amazon Keyspace, des alertes peuvent s'afficher pour les appels passés par Application Auto Scaling dans le cadre de son processus de validation de configuration. Vous pouvez filtrer ces alertes en utilisant le `invokedBy` champ qui contient `application-autoscaling.amazonaws.com` ces contrôles de validation.

Configurer et mettre à jour les politiques de dimensionnement automatique d'Amazon Keyspaces

Vous pouvez utiliser la console, le CQL ou le AWS Command Line Interface (AWS CLI) pour configurer le dimensionnement automatique d'Amazon Keyspaces pour les tables nouvelles et existantes. Vous pouvez également modifier les paramètres de mise à l'échelle automatique ou désactiver la mise à l'échelle automatique.

Pour des fonctionnalités plus avancées, telles que la définition des temps de recharge de mise à l'échelle initiale et inférieure, nous vous recommandons d'utiliser CQL ou le pour gérer les politiques de dimensionnement d' AWS CLI Amazon Keyspaces.

Rubriques

- [Configurer les autorisations pour le dimensionnement automatique d'Amazon Keyspaces](#)
- [Création d'un nouveau tableau avec mise à l'échelle automatique](#)
- [Configurer le dimensionnement automatique sur une table existante](#)
- [Afficher la configuration de mise à l'échelle automatique d'Amazon Keyspaces de votre table](#)

- [Désactiver le dimensionnement automatique d'Amazon Keyspaces pour un tableau](#)
- [Afficher l'activité de mise à l'échelle automatique d'une table Amazon Keyspaces dans Amazon CloudWatch](#)

Configurer les autorisations pour le dimensionnement automatique d'Amazon Keyspaces

Pour commencer, vérifiez que le principal dispose des autorisations appropriées pour créer et gérer les paramètres de dimensionnement automatique. Dans Gestion des identités et des accès AWS (IAM), la politique AWS gérée `AmazonKeyspacesFullAccess` est requise pour gérer les politiques de dimensionnement d'Amazon Keyspaces.

Important

Les autorisations `application-autoscaling:*` sont nécessaires pour désactiver la mise à l'échelle automatique sur une table. Vous devez désactiver la mise à l'échelle automatique pour une table avant de pouvoir la supprimer.

Pour configurer un utilisateur ou un rôle IAM pour l'accès à la console Amazon Keyspaces et le dimensionnement automatique d'Amazon Keyspaces, ajoutez la politique suivante.

Pour joindre la **`AmazonKeyspacesFullAccess`** politique

1. Connectez-vous à la console IAM Console de gestion AWS et ouvrez-la à <https://console.aws.amazon.com/iam/> l'adresse.
2. Sur le tableau de bord de la console IAM, choisissez Utilisateurs, puis choisissez votre utilisateur ou votre rôle IAM dans la liste.
3. Sur la page Récapitulatif, choisissez Ajouter des autorisations.
4. Choisissez Attacher directement les stratégies existantes.
5. Dans la liste des politiques, choisissez `AmazonKeyspacesFullAccess`, puis cliquez sur Suivant : Réviser.
6. Choisissez Ajouter des autorisations.

Création d'un nouveau tableau avec mise à l'échelle automatique

Lorsque vous créez une nouvelle table Amazon Keyspaces, vous pouvez activer automatiquement le dimensionnement automatique en fonction de la capacité d'écriture ou de lecture de la table. Cela permet à Amazon Keyspaces de contacter Application Auto Scaling en votre nom pour enregistrer la table en tant que cible évolutive et ajuster la capacité d'écriture ou de lecture allouée.

Pour plus d'informations sur la façon de créer une table multirégionale et de configurer différents paramètres de mise à l'échelle automatique pour les répliques de tables, consultez [the section called “Création d'une table multirégionale en mode provisionné”](#)

Note

Le dimensionnement automatique d'Amazon Keyspaces nécessite la présence d'un rôle lié à un service (AWSRoleForApplicationAutoScaling_CassandraTable) qui exécute des actions de dimensionnement automatique en votre nom. Ce rôle est créé automatiquement pour vous. Pour de plus amples informations, veuillez consulter [the section called “Utilisation des rôles liés à un service”](#).

Console

Créez un nouveau tableau avec la mise à l'échelle automatique activée à l'aide de la console

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le panneau de navigation, choisissez Tables, puis Create table (Créer une table).
3. Sur la page Créer une table, dans la section Détails de la table, sélectionnez un espace de touche et attribuez un nom à la nouvelle table.
4. Dans la section Colonnes, créez le schéma de votre table.
5. Dans la section Clé primaire, définissez la clé primaire de la table et sélectionnez les colonnes de clustering facultatives.
6. Dans la section Paramètres du tableau, choisissez Personnaliser les paramètres.
7. Passez aux paramètres Read/write de capacité.
8. Pour le Mode de capacité, choisissez Provisioned (Alloué).
9. Dans la section Capacité de lecture confirmez que l'option Mettre à l'échelle automatiquement est sélectionnée.

Dans cette étape, vous sélectionnez les unités de capacité de lecture minimale et maximale pour la table, ainsi que l'utilisation cible.

- Unités de capacité minimale : entrez la valeur du niveau de débit minimal que le tableau doit toujours être prêt à prendre en charge. La valeur doit être comprise entre 1 et le quota par seconde du débit maximal pour votre compte (40 000 par défaut).
- Unités de capacité maximale : entrez le débit maximal que vous souhaitez allouer pour le tableau. La valeur doit être comprise entre 1 et le quota par seconde du débit maximal pour votre compte (40 000 par défaut).
- Utilisation cible — Entrez un taux d'utilisation cible compris entre 20 % et 90 %. Lorsque le trafic dépasse le taux d'utilisation cible défini, la capacité est automatiquement mise à l'échelle. Lorsque le trafic tombe en dessous de la cible définie, il est automatiquement réduit de nouveau.

 Note

Pour en savoir plus sur les quotas par défaut de votre compte et sur la façon de les augmenter, reportez-vous à la section [Quotas](#).

10. Dans la section Capacité d'écriture, choisissez les mêmes paramètres que ceux définis à l'étape précédente pour la capacité de lecture, ou configurez les valeurs de capacité manuellement.
11. Choisissez Créer un tableau. Votre table est créée avec les paramètres de mise à l'échelle automatique spécifiés.

Cassandra Query Language (CQL)

Créez une nouvelle table avec le dimensionnement automatique d'Amazon Keyspaces à l'aide de CQL

Pour configurer les paramètres de mise à l'échelle automatique d'une table par programmation, vous devez utiliser l'AUTOSCALING_SETTINGS instruction qui contient les paramètres de mise à l'échelle automatique d'Amazon Keyspaces. Les paramètres définissent les conditions qui obligent Amazon Keyspaces à ajuster le débit provisionné de votre table, ainsi que les actions facultatives supplémentaires à effectuer. Dans cet exemple, vous définissez les paramètres de mise à l'échelle automatique pour mytable.

Cette politique contient les éléments suivants :

- **AUTOSCALING_SETTINGS**— Spécifie si Amazon Keyspaces est autorisé à ajuster la capacité de débit en votre nom. Les valeurs suivantes sont obligatoires :
 - **provisioned_write_capacity_autoscaling_update**:
 - **minimum_units**
 - **maximum_units**
 - **provisioned_read_capacity_autoscaling_update**:
 - **minimum_units**
 - **maximum_units**
- **scaling_policy**— Amazon Keyspaces soutient la politique de suivi des cibles. Pour définir la politique de suivi des cibles, vous devez configurer les paramètres suivants.
 - **target_value**— La mise à l'échelle automatique d'Amazon Keyspaces garantit que le rapport entre la capacité consommée et la capacité allouée reste égal ou proche de cette valeur. Vous définissez **target_value** en tant que pourcentage.
 - **disableScaleIn**: (Facultatif) Un boolean qui indique si la table **scale-in** est désactivée ou activée. Ce paramètre est désactivé par défaut. Pour l'activer **scale-in**, définissez la boolean valeur sur **FALSE**. Cela signifie que la capacité est automatiquement réduite pour une table en votre nom.
 - **scale_out_cooldown**— Une activité de **scale-out** augmente le débit provisionné de votre table. Pour ajouter un temps de stabilisation pour les activités de montée en charge, spécifiez une valeur, en secondes, pour **scale_out_cooldown**. Si vous ne spécifiez aucune valeur, la valeur par défaut est 0. Pour plus d'informations sur le suivi des cibles et les périodes de recharge, consultez les [politiques de dimensionnement de Target Tracking](#) dans le guide de l'utilisateur d'Application Auto Scaling.
 - **scale_in_cooldown**— Une activité de **scale-in** réduit le débit provisionné de votre table. Pour ajouter un temps de stabilisation pour les activités de mise à l'échelle, spécifiez une valeur, en secondes, pour **scale_in_cooldown**. Si vous ne spécifiez aucune valeur, la valeur par défaut est 0. Pour plus d'informations sur le suivi des cibles et les périodes de recharge, consultez les [politiques de dimensionnement de Target Tracking](#) dans le guide de l'utilisateur d'Application Auto Scaling.

 Note

Pour mieux comprendre le fonctionnement de `target_value`, supposons que vous ayez une table avec un paramètre de débit approvisionné de 200 unités de capacité d'écriture. Vous décidez de créer une politique de mise à l'échelle pour cette table, avec une `target_value` de 70 pour cent.

Supposons maintenant que vous commencez à générer du trafic d'écriture vers la table de telle sorte que le débit d'écriture réel est de 150 unités de capacité. Le ratio consommation/approvisionnement ($150 / 200$) est maintenant de 75 %. Ce ratio étant supérieur à votre objectif, l'autoscaling augmente la capacité d'écriture allouée à 215, de sorte que le ratio soit ($150 / 215$), soit 69,77 %, le plus proche `target_value` possible de la vôtre, sans toutefois le dépasser.

Pour `mytable`, vous avez défini `TargetValue` une capacité de lecture et d'écriture de 50 %. Le dimensionnement automatique d'Amazon Keyspaces ajuste le débit provisionné de la table dans une fourchette de 5 à 10 unités de capacité afin que le ratio consommé/provisionné reste égal ou proche de 50 %. Pour la capacité de lecture, vous définissez les valeurs `ScaleInCooldown` pour `ScaleOutCooldown` et sur 60 secondes.

Vous pouvez utiliser l'instruction suivante pour créer une nouvelle table Amazon Keyspaces avec la mise à l'échelle automatique activée.

```
CREATE TABLE mykeyspace.mytable(pk int, ck int, PRIMARY KEY (pk, ck))
WITH CUSTOM_PROPERTIES = {
  'capacity_mode': {
    'throughput_mode': 'PROVISIONED',
    'read_capacity_units': 1,
    'write_capacity_units': 1
  }
} AND AUTOSCALING_SETTINGS = {
  'provisioned_write_capacity_autoscaling_update': {
    'maximum_units': 10,
    'minimum_units': 5,
    'scaling_policy': {
      'target_tracking_scaling_policy_configuration': {
        'target_value': 50
      }
    }
  }
},
```

```
'provisioned_read_capacity_autoscaling_update': {
  'maximum_units': 10,
  'minimum_units': 5,
  'scaling_policy': {
    'target_tracking_scaling_policy_configuration': {
      'target_value': 50,
      'scale_in_cooldown': 60,
      'scale_out_cooldown': 60
    }
  }
}
};
```

CLI

Créez un nouveau tableau avec la mise à l'échelle automatique d'Amazon Keyspaces à l'aide du AWS CLI

Pour configurer les paramètres de dimensionnement automatique d'une table par programmation, vous devez utiliser l'`autoScalingSpecification` action qui définit les paramètres du dimensionnement automatique d'Amazon Keyspaces. Les paramètres définissent les conditions qui obligent Amazon Keyspaces à ajuster le débit provisionné de votre table, ainsi que les actions facultatives supplémentaires à effectuer. Dans cet exemple, vous définissez les paramètres de mise à l'échelle automatique pour `mytable`.

Cette politique contient les éléments suivants :

- `autoScalingSpecification`— Spécifie si Amazon Keyspaces est autorisé à ajuster la capacité et le débit en votre nom. Vous pouvez activer la mise à l'échelle automatique pour la capacité de lecture et pour la capacité d'écriture séparément. Vous devez ensuite définir les paramètres suivants pour `autoScalingSpecification` :
 - `writeCapacityAutoScaling`— Les unités de capacité d'écriture maximale et minimale.
 - `readCapacityAutoScaling`— Les unités de capacité de lecture maximale et minimale.
 - `scalingPolicy`— Amazon Keyspaces soutient la politique de suivi des cibles. Pour définir la politique de suivi des cibles, vous devez configurer les paramètres suivants.
 - `targetValue`— La mise à l'échelle automatique d'Amazon Keyspaces garantit que le rapport entre la capacité consommée et la capacité allouée reste égal ou proche de cette valeur. Vous définissez `targetValue` en tant que pourcentage.

- `disableScaleIn`: (Facultatif) Un boolean qui indique si la table `scale-in` est désactivée ou activée. Ce paramètre est désactivé par défaut. Pour l'activer `scale-in`, définissez la boolean valeur sur `FALSE`. Cela signifie que la capacité est automatiquement réduite pour une table en votre nom.
- `scaleOutCooldown`— Une activité de `scale-out` augmente le débit provisionné de votre table. Pour ajouter un temps de stabilisation pour les activités de montée en charge, spécifiez une valeur, en secondes, pour `ScaleOutCooldown`. La valeur par défaut est 0. Pour plus d'informations sur le suivi des cibles et les périodes de recharge, consultez les [politiques de dimensionnement de Target Tracking](#) dans le guide de l'utilisateur d'Application Auto Scaling.
- `scaleInCooldown`— Une activité de `scale-in` réduit le débit provisionné de votre table. Pour ajouter un temps de stabilisation pour les activités de mise à l'échelle, spécifiez une valeur, en secondes, pour `ScaleInCooldown`. La valeur par défaut est 0. Pour plus d'informations sur le suivi des cibles et les périodes de recharge, consultez les [politiques de dimensionnement de Target Tracking](#) dans le guide de l'utilisateur d'Application Auto Scaling.

 Note

Pour mieux comprendre le fonctionnement de `TargetValue`, supposons que vous ayez une table avec un paramètre de débit approvisionné de 200 unités de capacité d'écriture. Vous décidez de créer une politique de mise à l'échelle pour cette table, avec une `TargetValue` de 70 pour cent.

Supposons maintenant que vous commencez à générer du trafic d'écriture vers la table de telle sorte que le débit d'écriture réel est de 150 unités de capacité. Le ratio consommation/approvisionnement ($150 / 200$) est maintenant de 75 %. Ce ratio étant supérieur à votre objectif, l'autoscaling augmente la capacité d'écriture allouée à 215, de sorte que le ratio soit ($150 / 215$), soit 69,77 %, le plus proche `TargetValue` possible de la vôtre, sans toutefois le dépasser.

Pour `mytable`, vous avez défini `TargetValue` une capacité de lecture et d'écriture de 50 %. Le dimensionnement automatique d'Amazon Keyspaces ajuste le débit provisionné de la table dans une fourchette de 5 à 10 unités de capacité afin que le ratio consommé/provisionné reste égal ou proche de 50 %. Pour la capacité de lecture, vous définissez les valeurs `ScaleInCooldown` pour `ScaleOutCooldown` et sur 60 secondes.

Lorsque vous créez des tables avec des paramètres de mise à l'échelle automatique complexes, il est utile de charger les paramètres de mise à l'échelle automatique à partir d'un fichier JSON. Dans l'exemple suivant, vous pouvez télécharger le fichier JSON d'exemple à partir du fichier [auto-scaling.zip](#) et l'extraire `auto-scaling.json` en prenant note du chemin d'accès au fichier. Dans cet exemple, le fichier JSON se trouve dans le répertoire actuel. Pour connaître les différentes options de chemin de fichier, consultez [Comment charger des paramètres à partir d'un fichier](#).

```
aws keyspaces create-table --keyspace-name mykeyspace --table-name mytable
    \ --schema-definition 'allColumns=[{name=pk,type=int},
{name=ck,type=int}],partitionKeys=[{name=pk},{name=ck}]'
    \ --capacity-specification
throughputMode=PROVISIONED,readCapacityUnits=1,writeCapacityUnits=1
    \ --auto-scaling-specification file://auto-scaling.json
```

Configurer le dimensionnement automatique sur une table existante

Vous pouvez mettre à jour une table Amazon Keyspaces existante pour activer le dimensionnement automatique en fonction de la capacité d'écriture ou de lecture de la table. Si vous mettez à jour une table qui est actuellement en mode capacité à la demande, vous devez d'abord changer le mode capacité de la table en mode capacité provisionnée.

Pour plus d'informations sur la façon de mettre à jour les paramètres de mise à l'échelle automatique pour une table multirégionale, consultez [the section called “Mettre à jour les paramètres de capacité allouée et de dimensionnement automatique pour une table multirégionale”](#).

Le dimensionnement automatique d'Amazon Keyspaces nécessite la présence d'un rôle lié à un service (AWS `ServiceRoleForApplicationAutoScaling_CassandraTable`) qui exécute des actions de dimensionnement automatique en votre nom. Ce rôle est créé automatiquement pour vous. Pour de plus amples informations, veuillez consulter [the section called “Utilisation des rôles liés à un service”](#).

Console

Configurer le dimensionnement automatique d'Amazon Keyspaces pour une table existante

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Choisissez la table avec laquelle vous souhaitez travailler, puis accédez à l'onglet Capacité.

3. Dans la section Paramètres de capacité, choisissez Modifier.
4. En mode Capacité, assurez-vous que la table utilise le mode Capacité provisionnée.
5. Sélectionnez Mettre à l'échelle automatiquement et consultez l'étape 6 dans [the section called "Création d'un nouveau tableau avec mise à l'échelle automatique"](#) pour modifier la capacité de lecture et d'écriture.
6. Lorsque les paramètres de mise à l'échelle automatique sont définis, choisissez Enregistrer.

Cassandra Query Language (CQL)

Configurer une table existante avec le dimensionnement automatique d'Amazon Keyspaces à l'aide de CQL

Vous pouvez utiliser l'`ALTER TABLE` instruction d'une table Amazon Keyspaces existante pour configurer le dimensionnement automatique en fonction de la capacité d'écriture ou de lecture de la table. Si vous mettez à jour une table qui est actuellement en mode capacité à la demande, vous devez la `capacity_mode` définir sur provisionnée. Si votre table est déjà en mode capacité provisionnée, ce champ peut être omis.

Dans l'exemple suivant, l'instruction met à jour la table `mytable`, qui est en mode capacité à la demande. L'instruction fait passer le mode de capacité de la table en mode provisionné avec le dimensionnement automatique activé.

La capacité d'écriture est configurée dans une plage de 5 à 10 unités de capacité avec une valeur cible de 50 %. La capacité de lecture est également configurée dans une plage de 5 à 10 unités de capacité avec une valeur cible de 50 %. Pour la capacité de lecture, vous définissez les valeurs `scale_in_cooldown` pour `scale_out_cooldown` et sur 60 secondes.

```
ALTER TABLE mykeyspace.mytable
WITH CUSTOM_PROPERTIES = {
    'capacity_mode': {
        'throughput_mode': 'PROVISIONED',
        'read_capacity_units': 1,
        'write_capacity_units': 1
    }
} AND AUTOSCALING_SETTINGS = {
    'provisioned_write_capacity_autoscaling_update': {
        'maximum_units': 10,
        'minimum_units': 5,
        'scaling_policy': {
```

```
        'target_tracking_scaling_policy_configuration': {
            'target_value': 50
        }
    },
    'provisioned_read_capacity_autoscaling_update': {
        'maximum_units': 10,
        'minimum_units': 5,
        'scaling_policy': {
            'target_tracking_scaling_policy_configuration': {
                'target_value': 50,
                'scale_in_cooldown': 60,
                'scale_out_cooldown': 60
            }
        }
    }
};
```

CLI

Configurez une table existante avec le dimensionnement automatique d'Amazon Keyspaces à l'aide du AWS CLI

Pour une table Amazon Keyspaces existante, vous pouvez activer le dimensionnement automatique pour la capacité d'écriture ou de lecture de la table à l'UpdateTable aide de cette opération.

Vous pouvez utiliser la commande suivante pour activer le dimensionnement automatique d'Amazon Keyspaces pour une table existante. Les paramètres de mise à l'échelle automatique de la table sont chargés à partir d'un fichier JSON. Dans l'exemple suivant, vous pouvez télécharger le fichier JSON d'exemple à partir du fichier [auto-scaling.zip](#) et l'extraire `auto-scaling.json` en prenant note du chemin d'accès au fichier. Dans cet exemple, le fichier JSON se trouve dans le répertoire actuel. Pour connaître les différentes options de chemin de fichier, consultez [Comment charger des paramètres à partir d'un fichier](#).

Pour plus d'informations sur les paramètres de mise à l'échelle automatique utilisés dans l'exemple suivant, consultez [the section called “Création d'un nouveau tableau avec mise à l'échelle automatique”](#).

```
aws keyspaces update-table --keyspace-name mykeyspace --table-name mytable
    \ --capacity-specification
    throughputMode=PROVISIONED,readCapacityUnits=1,writeCapacityUnits=1
```

```
\ --auto-scaling-specification file://auto-scaling.json
```

Afficher la configuration de mise à l'échelle automatique d'Amazon Keyspaces de votre table

Vous pouvez utiliser la console, le CQL ou le AWS CLI pour afficher et mettre à jour les paramètres de mise à l'échelle automatique d'une table Amazon Keyspaces.

Console

Afficher les paramètres de mise à l'échelle automatique à l'aide de la console

1. Choisissez le tableau que vous souhaitez consulter et accédez à l'onglet Capacité.
2. Dans la section Paramètres de capacité, choisissez Modifier. Vous pouvez désormais modifier les paramètres dans les sections Capacité de lecture ou Capacité d'écriture. Pour plus d'informations sur ces paramètres, consultez la page [the section called “Création d'un nouveau tableau avec mise à l'échelle automatique”](#).

Cassandra Query Language (CQL)

Consultez la politique de dimensionnement automatique d'Amazon Keyspaces de votre table à l'aide de CQL

Pour afficher les détails de la configuration de mise à l'échelle automatique d'une table, utilisez la commande suivante.

```
SELECT * FROM system_schema_mcs.autoscaling WHERE keyspace_name = 'mykeyspace' AND
table_name = 'mytable';
```

La sortie de cette commande ressemble à ceci :

```
keyspace_name | table_name | provisioned_read_capacity_autoscaling_update
|
provisioned_write_capacity_autoscaling_update
-----+-----
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
```



```
mykeyspace | mytable | {'minimum_units': 5, 'maximum_units':
10, 'scaling_policy': {'target_tracking_scaling_policy_configuration':
{'scale_out_cooldown': 60, 'disable_scale_in': false, 'target_value':
50, 'scale_in_cooldown': 60}}} | {'minimum_units': 5, 'maximum_units':
10, 'scaling_policy': {'target_tracking_scaling_policy_configuration':
{'scale_out_cooldown': 0, 'disable_scale_in': false, 'target_value': 50,
'scale_in_cooldown': 0}}}
```

CLI

Consultez la politique de dimensionnement automatique d'Amazon Keyspaces de votre table à l'aide du AWS CLI

Pour afficher la configuration de mise à l'échelle automatique d'une table, vous pouvez utiliser l'opération `get-table-auto-scaling-settings`. La commande CLI suivante en est un exemple.

```
aws keyspaces get-table-auto-scaling-settings --keyspace-name mykeyspace --table-
name mytable
```

La sortie de cette commande ressemble à ceci :

```
{
  "keyspaceName": "mykeyspace",
  "tableName": "mytable",
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/
table/mytable",
  "autoScalingSpecification": {
    "writeCapacityAutoScaling": {
      "autoScalingDisabled": false,
      "minimumUnits": 5,
      "maximumUnits": 10,
      "scalingPolicy": {
        "targetTrackingScalingPolicyConfiguration": {
          "disableScaleIn": false,
          "scaleInCooldown": 0,
          "scaleOutCooldown": 0,
          "targetValue": 50.0
        }
      }
    },
    "readCapacityAutoScaling": {
```

```

    "autoScalingDisabled": false,
    "minimumUnits": 5,
    "maximumUnits": 10,
    "scalingPolicy": {
      "targetTrackingScalingPolicyConfiguration": {
        "disableScaleIn": false,
        "scaleInCooldown": 60,
        "scaleOutCooldown": 60,
        "targetValue": 50.0
      }
    }
  }
}

```

Désactiver le dimensionnement automatique d'Amazon Keyspaces pour un tableau

Vous pouvez désactiver le dimensionnement automatique d'Amazon Keyspaces pour votre table à tout moment. Si vous n'avez plus besoin de redimensionner la capacité de lecture ou d'écriture de votre tableau, vous devriez envisager de désactiver le dimensionnement automatique afin qu'Amazon Keyspaces ne continue pas à modifier les paramètres de capacité de lecture ou d'écriture de votre tableau. Vous pouvez mettre à jour le tableau à l'aide de la console, du CQL ou du AWS CLI.

La désactivation de la mise à l'échelle automatique supprime également les CloudWatch alarmes créées en votre nom.

Pour supprimer le rôle lié à un service utilisé par Application Auto Scaling pour accéder à votre table Amazon Keyspaces, suivez les étapes décrites dans [the section called “Supprimer un rôle lié à un service pour Amazon Keyspaces”](#)

Note

Pour supprimer le rôle lié à un service utilisé par Application Auto Scaling, vous devez désactiver le dimensionnement automatique sur toutes les tables du compte. Régions AWS

Console

Désactiver le redimensionnement automatique d'Amazon Keyspaces pour votre table à l'aide de la console

Utilisation de la console Amazon Keyspaces

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Choisissez le tableau que vous souhaitez mettre à jour et accédez à l'onglet Capacité.
3. Dans la section Paramètres de capacité, choisissez Modifier.
4. Pour désactiver le dimensionnement automatique d'Amazon Keyspaces, décochez la case Scale automatically. La désactivation du dimensionnement automatique annule l'enregistrement de la table en tant que cible évolutive avec Application Auto Scaling.

Cassandra Query Language (CQL)

Désactiver la mise à l'échelle automatique d'Amazon Keyspaces pour votre table à l'aide de CQL

L'instruction suivante désactive la mise à l'échelle automatique de la capacité d'écriture de la table mytable.

```
ALTER TABLE mykeyspace.mytable
WITH AUTOSCALING_SETTINGS = {
    'provisioned_write_capacity_autoscaling_update': {
        'autoscaling_disabled': true
    }
};
```

CLI

Désactivez la mise à l'échelle automatique d'Amazon Keyspaces pour votre tableau à l'aide du AWS CLI

La commande suivante désactive le dimensionnement automatique en fonction de la capacité de lecture de la table. Il supprime également les CloudWatch alarmes créées en votre nom.

```
aws keyspaces update-table --keyspace-name mykeyspace --table-name mytable
    \ --auto-scaling-specification
    readCapacityAutoScaling={autoScalingDisabled=true}
```

Afficher l'activité de mise à l'échelle automatique d'une table Amazon Keyspaces dans Amazon CloudWatch

Vous pouvez surveiller la manière dont le dimensionnement automatique d'Amazon Keyspaces utilise les ressources en utilisant Amazon CloudWatch, qui génère des statistiques relatives à votre utilisation et à vos performances. Suivez les étapes du [guide de l'Application Auto Scaling utilisateur](#) pour créer un CloudWatch tableau de bord.

Utilisez efficacement la capacité de pointe dans Amazon Keyspaces

Amazon Keyspaces offre une certaine flexibilité dans le provisionnement du débit par partition en fournissant une capacité de pointe. Lorsque vous n'utilisez pas pleinement le débit d'une partition, Amazon Keyspaces réserve une partie de cette capacité inutilisée pour des pics de débit ultérieurs afin de gérer les pics d'utilisation.

Amazon Keyspaces conserve actuellement jusqu'à 5 minutes (300 secondes) de capacité de lecture et d'écriture inutilisée. Lors d'une vague occasionnelle d'activité de lecture ou d'écriture, ces unités de capacité supplémentaire peuvent être consommées rapidement, même plus rapidement que la capacité de débit allouée par seconde que vous avez définie pour votre table.

Amazon Keyspaces peut également consommer de la capacité en rafale pour la maintenance en arrière-plan et d'autres tâches sans préavis.

Notez que ces informations relatives à la capacité de transmission en mode rafale sont susceptibles d'évoluer.

Utilisation des fonctionnalités d'Amazon Keyspaces (pour Apache Cassandra)

Ce chapitre fournit des informations sur l'utilisation d'Amazon Keyspaces et sur diverses fonctionnalités de base de données, telles que la sauvegarde et la restauration, le temps de vie et la réplication multirégionale.

- **Durée de vie** : Amazon Keyspaces fait automatiquement expirer les données des tables en fonction de la valeur Time to Live que vous avez définie. Découvrez comment configurer le TTL et comment l'utiliser dans vos tables.
- **PITR** — Protégez vos tables Amazon Keyspaces contre les opérations d'écriture ou de suppression accidentelles en créant des sauvegardes continues des données de vos tables. Découvrez comment configurer le PITR sur vos tables et comment restaurer une table à un moment précis ou comment restaurer une table supprimée accidentellement.
- **Utilisation de tables multirégionales** : Multi-Region les tables d'Amazon Keyspaces doivent avoir une capacité de débit d'écriture configurée en mode à la demande ou en mode capacité provisionnée avec mise à l'échelle automatique. Planifiez les besoins en capacité de débit en estimant les unités de capacité d'écriture (WCU) requises pour chaque région, et fournissez la somme des écritures provenant de toutes les régions afin de garantir une capacité suffisante pour les écritures répliquées.
- **Capture des données de modification (CDC) d'Amazon Keyspaces** — Les flux Amazon Keyspaces CDC enregistrent les événements de modification au niveau des lignes depuis votre table en temps quasi réel. Découvrez comment utiliser la bibliothèque client Kinesis (KCL) pour consommer et traiter les données issues des flux CDC d'Amazon Keyspaces.
- **Requêtes et pagination** : Amazon Keyspaces prend en charge des fonctionnalités de requête avancées, telles que l'utilisation de IN l'opérateur SELECT avec des instructions, le classement des résultats ORDER BY avec et la pagination automatique de grands ensembles de résultats. Cette section explique comment Amazon Keyspaces traite ces requêtes et fournit des exemples.
- **Partitionneurs** — Amazon Keyspaces propose trois partitionneurs : `Murmur3Partitioner` (par défaut), et `RandomPartitioner` `DefaultPartitioner` Vous pouvez modifier le partitionneur par région au niveau du compte à l'aide du langage CQL (Cassandra Query Language). Console de gestion AWS
- **Client-side horodatages** : les Client-side horodatages sont des horodatages Cassandra-compatible conservés par Amazon Keyspaces pour chaque cellule de votre tableau. Utilisez des horodatages

côté client pour résoudre les conflits et pour laisser votre application cliente déterminer l'ordre des écritures.

- **User-defined types (UDT)** : avec les UDT, vous pouvez définir des structures de données dans vos applications qui représentent des hiérarchies de données réelles.
- **Marquage des ressources** : vous pouvez étiqueter les ressources Amazon Keyspaces, telles que les espaces clés et les tableaux, à l'aide de balises. Les balises aident à classer les ressources, à permettre le suivi des coûts et à configurer le contrôle d'accès en fonction des balises. Cette section couvre les restrictions, les opérations et les meilleures pratiques en matière de balisage pour Amazon Keyspaces.
- **CloudFormation modèles** : vous CloudFormation aide à modéliser et à configurer vos espaces clés et vos tables Amazon Keyspaces afin que vous puissiez passer moins de temps à créer et à gérer vos ressources et votre infrastructure.

Rubriques

- [Keyspaces du système dans Amazon Keyspaces](#)
- [User-defined types \(UDT\) dans Amazon Keyspaces](#)
- [Utilisation de requêtes CQL dans Amazon Keyspaces](#)
- [Utilisation des flux de capture des données modifiées \(CDC\) dans Amazon Keyspaces](#)
- [Utilisation des partitionneurs dans Amazon Keyspaces](#)
- [Client-side horodatages dans Amazon Keyspaces](#)
- [Multi-Region réplication pour Amazon Keyspaces \(pour Apache Cassandra\)](#)
- [Backup et restauration des données avec restauration instantanée pour Amazon Keyspaces](#)
- [Expirer les données avec Time to Live \(TTL\) pour Amazon Keyspaces \(pour Apache Cassandra\)](#)
- [L'utilisation de ce service avec un AWS Kit SDK](#)
- [Utilisation de balises et d'étiquettes pour les ressources Amazon Keyspaces](#)
- [Créez des ressources Amazon Keyspaces avec AWS CloudFormation](#)
- [Utilisation de NoSQL Workbench avec Amazon Keyspaces \(pour Apache Cassandra\)](#)

Keyspaces du système dans Amazon Keyspaces

Les espaces de touches système et les tables système d'Amazon Keyspaces (pour Apache Cassandra) sont des ressources en lecture seule qui stockent les métadonnées relatives à vos ressources Amazon Keyspaces. Les espaces de touches du système sont présents partout

Compte AWS, que vous ayez créé des espaces de touches ou des tables. Ils constituent une fonctionnalité de compatibilité avec Apache Cassandra, et ils sont fournis sans frais supplémentaires.

Vous ne pouvez pas modifier ou supprimer les espaces de touches du système. La console Amazon Keyspaces affiche uniquement les espaces de touches créés par les utilisateurs. Les espaces de touches du système sont accessibles par programmation via CQL et apparaissent dans des services tels que et. AWS CloudFormation AWS Config

Amazon Keyspaces utilise quatre espaces de touches système :

- `system`
- `system_schema`
- `system_schema_mcs`
- `system_multiregion_info`

Les sections suivantes fournissent des informations sur les espaces de touches du système et les tables système pris en charge dans Amazon Keyspaces.

system

Il s'agit d'un keyspace de Cassandra. Amazon Keyspaces utilise les tableaux suivants.

Noms des tables	Noms de colonnes	Commentaires
<code>local</code>	<code>key</code> , <code>bootstrap</code> <code>ped</code> , <code>broadcast</code> <code>_address</code> , <code>cluster_n</code> <code>ame</code> , <code>cql_versi</code> <code>on</code> , <code>data_cent</code> <code>er</code> , <code>gossip_ge</code> <code>neration</code> , <code>host_id</code> , <code>listen_address</code> , <code>native_protocol_ve</code> <code>rsion</code> , <code>partition</code> <code>er</code> , <code>rack</code> , <code>release_v</code> <code>ersion</code> , <code>rpc_addre</code> <code>ss</code> , <code>schema_version</code> ,	Informations sur le keyspace local.

Noms des tables	Noms de colonnes	Commentaires
	thrift_version, tokens, truncated_at	
peers	peer, data_center, host_id, preferred _ip, rack, release_v ersion, rpc_addre ss, schema_version, tokens	Consultez ce tableau pour voir les points de terminaison disponibles. Par exemple, si vous vous connectez via un point de terminaison public, une liste de neuf adresses IP disponibles s'affiche. Si vous vous connectez via un point de terminaison FIPS, une liste de trois adresses IP s'affiche. Si vous vous connectez via un point de terminaison AWS PrivateLink VPC, la liste des adresses IP que vous avez configurées s'affiche. Pour de plus amples informations, veuillez consulter the section called “Remplissage des entrées de system.peers table avec les informations de point de terminaison VPC de l'interface” .
size_estimates	keyspace_name, table_name, range_sta rt, range_end, mean_partition_size, partitions_count	Ce tableau définit la taille totale et le nombre de partitions pour chaque plage de jetons pour chaque table. Cela est nécessaire pour le connecteur Apache Cassandra Spark, qui utilise la taille de partition estimée pour distribuer le travail.

Noms des tables	Noms de colonnes	Commentaires
<code>prepared_statements</code>	<code>prepared_id</code> , <code>logged_keyspace</code> , <code>query_string</code>	Ce tableau contient des informations sur les requêtes enregistrées.

system_schema

Il s'agit d'un keyspace de Cassandra. Amazon Keyspaces utilise les tableaux suivants.

Noms des tables	Noms de colonnes	Commentaires
<code>keyspaces</code>	<code>keyspace_name</code> , <code>durable_writes</code> , <code>replication</code>	Informations relatives à un keyspace spécifique.
<code>tables</code>	<code>keyspace_name</code> , <code>table_name</code> , <code>bloom_filter_fp_chance</code> , <code>caching</code> , <code>comment</code> , <code>compaction</code> , <code>compression</code> , <code>crc_check_chance</code> , <code>dclocal_read_repair_chance</code> , <code>default_time_to_live</code> , <code>extensions</code> , <code>flags</code> , <code>gc_grace_seconds</code> , <code>id</code> , <code>max_index_interval</code> , <code>memtable_flush_period_in_ms</code> , <code>min_index_interval</code> , <code>read_repair_chance</code> , <code>speculative_retry</code>	Informations relatives à une table spécifique.

Noms des tables	Noms de colonnes	Commentaires
types	keyspace_name, type_name, field_names, field_types	Informations sur un type défini par l'utilisateur (UDT) spécifique.
columns	keyspace_name, table_name, column_name, clustering_order, column_name_bytes, kind, position, type	Informations relatives à une colonne spécifique.

system_schema_mcs

Il s'agit d'un espace de touches Amazon Keyspaces qui stocke des informations ou des paramètres spécifiques à AWS Amazon Keyspaces.

Noms des tables	Noms de colonnes	Commentaires
keyspaces	keyspace_name, durable_writes, replication	Interrogez cette table pour savoir par programmation si un espace de touches a été créé. Pour de plus amples informations, veuillez consulter the section called “Vérifier l'état de création de l'espace clé” .
tables	keyspace_name, creation_time, speculative_retry, cdc, gc_grace_seconds, crc_check_chance, min_index_interval, bloom_filter_fp_chance,	Interrogez cette table pour connaître le statut d'une table spécifique. Pour de plus amples informations, veuillez consulter the section called “Vérifier l'état de création de la table” .

Noms des tables	Noms de colonnes	Commentaires
	flags, custom_properties, dclocal_read_repair_chance, table_name, caching, default_time_to_live, read_repair_chance, max_index_interval, extensions, compaction, comment, id, compression, memtable_flush_period_in_ms, cdc_specification, latest_stream_arn, status	<p>Vous pouvez également consulter ce tableau pour répertorier les paramètres spécifiques à Amazon Keyspaces et stockés sous le nom de. custom_properties Par exemple :</p> <ul style="list-style-type: none"> • capacity_mode • cdc_specification • client_side_timestamps • encryption_specification • point_in_time_recovery • ttl
tables_history	keyspace_name, table_name, event_time, creation_time, custom_properties, event	Interrogez cette table pour en savoir plus sur les modifications apportées au schéma d'une table spécifique.
columns	keyspace_name, table_name, column_name, clustering_order, column_name_bytes, kind, position, type	Cette table est identique à la table Cassandra dans le system_schema keyspace.

Noms des tables	Noms de colonnes	Commentaires
tags	resource_id, keyspace_name, resource_name, resource_type, tags	Interrogez ce tableau pour savoir si un espace de touches contient des balises. Pour de plus amples informations, veuillez consulter the section called “Afficher les balises du tableau” .
types	keyspace_name, type_name, field_names, field_types, max_nesting_depth, last_modified_timestamp, status, direct_referring_tables, direct_parent_types	Consultez ce tableau pour obtenir des informations sur les types définis par l'utilisateur (UDT). Par exemple, vous pouvez interroger cette table pour répertorier tous les UDT pour un espace de touches donné. Pour de plus amples informations, veuillez consulter the section called “User-defined types (UDT)” .

Noms des tables	Noms de colonnes	Commentaires
autoscaling	keyspace_name, table_name, provisioned_read_capacity_autoscaling_update, provisioned_write_capacity_autoscaling_update	Interrogez cette table pour obtenir les paramètres de dimensionnement automatique d'une table provisionnée. Notez que ces paramètres ne seront pas disponibles tant que le tableau ne sera pas actif. Pour interroger cette table, vous devez spécifier <code>keyspace_name</code> et <code>table_name</code> dans la WHERE clause. Pour de plus amples informations, veuillez consulter the section called “Afficher la configuration de mise à l'échelle automatique d'Amazon Keyspaces de votre table” .

system_multiregion_info

Il s'agit d'un espace de touches Amazon Keyspaces qui stocke des informations sur la réplication multirégionale.

Noms des tables	Noms de colonnes	Commentaires
tables	keyspace_name, table_name, region, status	Ce tableau contient des informations sur les tables multirégionales, par exemple les informations dans Régions AWS auxquelles la table est répliquée et le statut de la table. Vous pouvez également consulter ce tableau pour

Noms des tables	Noms de colonnes	Commentaires
		<p>répertorier les paramètres spécifiques à Amazon Keyspaces stockés sous le nom de <code>custom_properties</code>. Par exemple :</p> <ul style="list-style-type: none"> • <code>capacity_mode</code> <p>Pour interroger cette table, vous devez spécifier <code>keyspace_name</code> et <code>table_name</code> dans la WHERE clause. Pour de plus amples informations, veuillez consulter the section called “Création d'un espace de saisie multirégional”.</p>
<code>keyspaces</code>	<code>keyspace_name</code> , <code>region</code> , <code>status</code> , <code>tables_replication_progress</code>	<p>Ce tableau contient des informations sur la progression d'une ALTER KEYSPACE opération qui ajoute une réplique à un keyspace, par exemple, le nombre de tables déjà créées dans la nouvelle région et le nombre de tables encore en cours d'élaboration. Pour des exemples, voir the section called “Vérifier la progression de la réplication”.</p>

Noms des tables	Noms de colonnes	Commentaires
autoscaling	keyspace_name, table_name, provisioned_read_capacity_autoscaling_update, provisioned_write_capacity_autoscaling_update, region	Interrogez cette table pour obtenir les paramètres de dimensionnement automatique d'une table provisionnée multirégionale. Notez que ces paramètres ne seront pas disponibles tant que le tableau ne sera pas actif. Pour interroger cette table, vous devez spécifier <code>keyspace_name</code> et <code>table_name</code> dans la WHERE clause. Pour de plus amples informations, veuillez consulter the section called “Mettre à jour les paramètres de capacité allouée et de dimensionnement automatique pour une table multirégionale” .
types	keyspace_name, type_name, field_names, field_types, max_nesting_depth, last_modified_timestamp, status, direct_referring_tables, direct_parent_types, region	Consultez ce tableau pour obtenir des informations sur les types définis par l'utilisateur (UDT) dans les espaces clés multirégionaux. Par exemple, vous pouvez interroger cette table pour répertorier toutes les répliques de table et leurs AWS régions respectives qui utilisent des UDT pour un espace clé donné. Pour de plus amples informations, veuillez consulter the section called “User-defined types (UDT)” .

User-defined types (UDT) dans Amazon Keyspaces

Un type défini par l'utilisateur (UDT) est un regroupement de champs et de types de données que vous pouvez utiliser pour définir une seule colonne dans Amazon Keyspaces. Les types de données valides pour les UDT sont tous les types de données Cassandra pris en charge, y compris les collections et autres UDT que vous avez déjà créés dans le même espace clé. Pour plus d'informations sur les types de données Cassandra pris en charge, consultez [the section called “Support des types de données Cassandra”](#).

Vous pouvez utiliser des types définis par l'utilisateur (UDT) dans Amazon Keyspaces pour organiser les données de manière plus efficace. Par exemple, vous pouvez créer des UDT avec des collections imbriquées, ce qui vous permet d'implémenter une modélisation de données plus complexe dans vos applications. Vous pouvez également utiliser le mot clé frozen pour définir les UDT.

Les UDT sont liés à un espace de touches et sont disponibles pour toutes les tables et tous les UDT d'un même espace de touches. Vous pouvez créer des UDT dans des espaces clés à région unique ou multirégionale.

Vous pouvez créer de nouvelles tables ou modifier des tables existantes et ajouter de nouvelles colonnes utilisant un UDT. Pour créer un UDT avec un UDT imbriqué, l'UDT imbriqué doit être figé.

Pour connaître le nombre d'UDT pris en charge par espace de touches, les niveaux d'imbrication pris en charge et les autres valeurs et quotas par défaut liés aux UDT, voir. [the section called “Quotas et valeurs par défaut pour les types définis par l'utilisateur \(UDT\) dans Amazon Keyspaces”](#)

Pour plus d'informations sur le calcul de la taille codée des UDT, consultez [the section called “Estimer la taille codée des valeurs de données en fonction du type de données”](#).

Pour plus d'informations sur la syntaxe CQL, consultez [the section called “Types”](#).

Pour en savoir plus sur les UDT et la restauration instantanée, consultez. [the section called “PITR et UDT”](#)

Rubriques

- [Configurer les autorisations pour travailler avec les types définis par l'utilisateur \(UDT\) dans Amazon Keyspaces](#)
- [Création d'un type défini par l'utilisateur \(UDT\) dans Amazon Keyspaces](#)
- [Afficher les types définis par l'utilisateur \(UDT\) dans Amazon Keyspaces](#)

- [Supprimer un type défini par l'utilisateur \(UDT\) dans Amazon Keyspaces](#)

Configurer les autorisations pour travailler avec les types définis par l'utilisateur (UDT) dans Amazon Keyspaces

Tout comme les tables, les UDT sont liés à un espace de touches spécifique. Mais contrairement aux tables, vous ne pouvez pas définir d'autorisations directement pour les UDT. Les UDT ne sont pas considérés comme des ressources AWS et ne possèdent aucun identifiant unique au format Amazon Resource Name (ARN). Pour donner à un IAM principal les autorisations nécessaires pour effectuer des actions spécifiques sur un UDT, vous devez définir des autorisations pour l'espace de touches auquel l'UDT est lié. Pour utiliser des UDT dans des espaces clés multirégionaux, des autorisations supplémentaires sont requises.

Pour pouvoir créer, afficher ou supprimer des UDT, le principal, par exemple un utilisateur ou un rôle IAM, a besoin des mêmes autorisations que celles requises pour effectuer la même action sur l'espace de touches auquel l'UDT est lié.

Pour plus d'informations sur Gestion des identités et des accès AWS, voir [the section called “Gestion des identités et des accès AWS”](#).

Autorisations pour créer un UDT

Pour créer un UDT dans un espace clé à région unique, le principal a besoin d'Create autorisations pour l'espace clé.

La politique IAM suivante en est un exemple.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cassandra:Create",
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/"
      ]
    }
  ]
}
```

Pour créer un UDT dans un espace clé multirégional, en plus des autorisations, le principal doit également disposer d'`CreateAuthorisations` pour effectuer l'action dans l'espace de touches `CreateMultiRegionResource` spécifié.

La politique IAM suivante en est un exemple.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "cassandra:Create", "cassandra:CreateMultiRegionResource" ],
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/"
      ]
    }
  ]
}
```

Autorisations pour afficher un UDT

Pour afficher ou répertorier les UDT dans un espace clé à région unique, le principal doit disposer d'autorisations de lecture pour l'espace clé du système. Pour de plus amples informations, veuillez consulter [the section called “system_schema_mcs”](#).

La politique IAM suivante en est un exemple.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cassandra:Select",
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
      ]
    }
  ]
}
```

Pour afficher ou répertorier les UDT pour un espace clé multirégional, le principal a besoin d'autorisations pour les actions `SELECT` et `SelectMultiRegionResource` pour l'espace

clé du système. Pour de plus amples informations, veuillez consulter [the section called “system_multiregion_info”](#).

La politique IAM suivante en est un exemple.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["cassandra:Select", "cassandra:SelectMultiRegionResource"],
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
      ]
    }
  ]
}
```

Autorisations pour supprimer un UDT

Pour supprimer un UDT d'un espace de touches à région unique, le principal doit avoir les autorisations nécessaires pour effectuer l'Dropaction dans l'espace de touches spécifié.

La politique IAM suivante en est un exemple.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cassandra:Drop",
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/"
      ]
    }
  ]
}
```

Pour supprimer un UDT d'un espace de touches multirégional, le principal a besoin d'autorisations pour l'Dropaction et pour l'action pour l'DropMultiRegionResourceespace de touches spécifié.

La politique IAM suivante en est un exemple.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [ "cassandra:Drop", "cassandra:DropMultiRegionResource" ],
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/"
      ]
    }
  ]
}
```

Création d'un type défini par l'utilisateur (UDT) dans Amazon Keyspaces

Pour créer un UDT dans un espace de touches à région unique, vous pouvez utiliser l'`CREATE TYPE` instruction dans CQL, la `create-type` commande associée à, ou la console AWS CLI.

Les noms UDT doivent contenir 48 caractères ou moins, doivent commencer par un caractère alphabétique et ne peuvent contenir que des caractères alphanumériques et des traits de soulignement. Amazon Keyspaces convertit automatiquement les majuscules en minuscules.

Vous pouvez également déclarer un nom UDT entre guillemets. Lorsque vous déclarez un nom UDT entre guillemets, Amazon Keyspaces préserve les majuscules et autorise les caractères spéciaux.

Vous pouvez également utiliser des guillemets doubles dans le nom lorsque vous créez l'UDT, mais vous devez éviter chaque guillemet double par un guillemet supplémentaire.

Le tableau suivant présente des exemples de noms UDT autorisés. La première colonne indique comment saisir le nom lorsque vous créez le type, la deuxième colonne montre comment Amazon Keyspaces met en forme le nom en interne. Amazon Keyspaces attend le nom formaté pour des opérations telles que `GetType`

Nom saisi	Nom formaté	Remarque
MY_UDT	my_udt	Sans guillemets, Amazon Keyspaces convertit tous les caractères majuscules en minuscules.
"MY_UDT"	MY_UDT	Avec les guillemets, Amazon Keyspaces respecte les majuscules et supprime les guillemets du nom formaté.

Nom saisi	Nom formaté	Remarque
"1234"	1234	Avec des guillemets, le nom peut commencer par un chiffre, et Amazon Keyspaces supprime les guillemets du nom formaté.
"Special_Ch@r@cter s<>!!"	Special_Ch@r@cter s<>!!	Avec des guillemets, le nom peut contenir des caractères spéciaux, et Amazon Keyspaces supprime les guillemets du nom formaté.
"nested""""quote s"	nested""""quotes	Amazon Keyspaces supprime les guillemets extérieurs et les guillemets d'échappement du nom formaté.

Console

Créez un type défini par l'utilisateur (UDT) avec la console Amazon Keyspaces

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le volet de navigation, choisissez Keyspaces, puis choisissez un keyspace dans la liste.
3. Choisissez l'onglet UDTs.
4. Choisissez Create UDT
5. Sous Détails de l'UDT, entrez le nom de l'UDT. Dans les champs UDT, vous définissez le schéma de l'UDT.
6. Pour terminer, choisissez Create UDT.

Cassandra Query Language (CQL)

Création d'un type défini par l'utilisateur (UDT) avec CQL

Dans cet exemple, nous créons une nouvelle version du tableau des récompenses littéraires utilisé dans [the section called “Création d’une table”](#). Dans ce tableau, nous enregistrons toutes les récompenses qu'un auteur reçoit pour un livre donné. Nous créons deux UDT imbriqués et contenant des informations sur le livre primé.

1. Créez un keyspace avec le nom `catalog`.

```
CREATE KEYSPACE catalog WITH REPLICATION = {'class': 'SingleRegionStrategy'};
```

2. Créez le premier type. Ce type stocke les codes BISAC, qui sont utilisés pour définir le genre des livres. Un code BISAC se compose d'un code alphanumérique et d'un maximum de quatre domaines thématiques.

```
CREATE TYPE catalog.bisac (  
    bisac_code text,  
    subject1 text,  
    subject2 text,  
    subject3 text,  
    subject4 text  
);
```

3. Créez un deuxième type de prix du livre utilisant le premier UDT. L'UDT imbriqué doit être gelé.

```
CREATE TYPE catalog.book (  
    award_title text,  
    book_title text,  
    publication_date date,  
    page_count int,  
    ISBN text,  
    genre FROZEN <bisac>  
);
```

4. Créez un tableau avec une colonne pour le nom de l'auteur et utilisez un type de liste pour les prix du livre. Notez que l'UDT utilisé dans la liste doit être figé.

```
CREATE TABLE catalog.authors (  
    author_name text PRIMARY KEY,  
    awards list <FROZEN <book>>  
);
```

5. Dans cette étape, nous insérons une ligne de données dans le nouveau tableau.

```
CONSISTENCY LOCAL_QUORUM;
```

```
INSERT INTO catalog.authors (author_name, awards) VALUES (  
    'John Stiles' ,
```

```
[{
  award_title: 'Wolf',
  book_title: 'Yesterday',
  publication_date: '2020-10-10',
  page_count: 345,
  ISBN: '026204630X',
  genre: { bisac_code: 'FIC014090', subject1: 'FICTION', subject2:
'Historical', subject3: '20th Century', subject4: 'Post-World War II'}
},
{award_title: 'Richard Roe',
  book_title: 'Who ate the cake?',
  publication_date: '2019-05-13',
  page_count: 193,
  ISBN: '9780262046305',
  genre: { bisac_code: 'FIC022130', subject1: 'FICTION', subject2: 'Mystery &
Detective', subject3: 'Cozy', subject4: 'Culinary'}}
];
```

6. Dans la dernière étape, nous lisons les données du tableau.

```
SELECT * FROM catalog.authors;
```

Le résultat de la commande doit ressembler à ceci.

```
author_name | awards
-----
+-----
John Stiles | [{award_title: 'Wolf', book_title: 'Yesterday', publication_date:
2020-10-10, page_count: 345, isbn: '026204630X', genre: {bisac_code:
'FIC014090', subject1: 'FICTION', subject2: 'Historical', subject3: '20th
Century', subject4: 'Post-World War II'}}, {award_title: 'Richard Roe',
book_title: 'Who ate the cake?', publication_date: 2019-05-13, page_count: 193,
isbn: '9780262046305', genre: {bisac_code: 'FIC022130', subject1: 'FICTION',
subject2: 'Mystery & Detective', subject3: 'Cozy', subject4: 'Culinary'}}]

(1 rows)
```

Pour plus d'informations sur la syntaxe CQL, consultez [the section called "CREATE TYPE"](#).

CLI

Créez un type défini par l'utilisateur (UDT) avec AWS CLI

1. Pour créer un type, vous pouvez utiliser la syntaxe suivante.

```
aws keyspaces create-type
--keyspace-name 'my_keyspace'
--type-name 'my_udt'
--field-definitions
  '['
    {"name" : "field1", "type" : "int"},
    {"name" : "field2", "type" : "text"}
  ']
```

2. Le résultat de cette commande ressemble à celui de cet exemple. Notez que cela typeName renvoie le nom formaté de l'UDT.

```
{
  "keyspaceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
my_keyspace/",
  "typeName": "my_udt"
}
```

Afficher les types définis par l'utilisateur (UDT) dans Amazon Keyspaces

Pour afficher ou répertorier tous les UDT dans un espace de touches à région unique, vous pouvez interroger la table `system_schema_mcs.types` dans l'espace de touches du système à l'aide d'une instruction dans CQL, ou utiliser `list-type` les commandes `get-type` et avec le ou la AWS CLI console.

Quelle que soit l'option, le principal IAM a besoin d'autorisations de lecture sur l'espace de touches du système. Pour de plus amples informations, veuillez consulter [the section called “Configurer les autorisations”](#).

Console

Afficher les types définis par l'utilisateur (UDT) avec la console Amazon Keyspaces

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le volet de navigation, choisissez Keyspaces, puis choisissez un keyspace dans la liste.
3. Choisissez l'onglet UDT pour consulter la liste de tous les UDT dans le keyspace.
4. Pour examiner un UDT en détail, choisissez-en un dans la liste.
5. Dans l'onglet Schéma, vous pouvez consulter le schéma. Dans l'onglet Utilisé dans, vous pouvez voir si cet UDT est utilisé dans des tables ou dans d'autres UDT. Notez que vous ne pouvez supprimer que les UDT qui ne sont pas utilisés par des tables ou par d'autres UDT.

Cassandra Query Language (CQL)

Afficher les types définis par l'utilisateur (UDT) d'un espace de touches à région unique avec CQL

1. Pour voir les types disponibles dans un espace de touches donné, vous pouvez utiliser l'instruction suivante.

```
SELECT type_name
FROM system_schema_mcs.types
WHERE keyspace_name = 'my_keyspace';
```

2. Pour afficher les détails relatifs à un type spécifique, vous pouvez utiliser l'instruction suivante.

```
SELECT
    keyspace_name,
    type_name,
    field_names,
    field_types,
    max_nesting_depth,
    last_modified_timestamp,
    status,
    direct_referring_tables,
    direct_parent_types
FROM system_schema_mcs.types
WHERE keyspace_name = 'my_keyspace' AND type_name = 'my_udt';
```

3. Vous pouvez répertorier tous les UDT présents dans le compte à l'aide `DESC TYPE` de.

```
DESC TYPES;

Keyspace my_keyspace
-----
my_udt1  my_udt2

Keyspace my_keyspace2
-----
my_udt1
```

4. Vous pouvez répertorier tous les UDT dans l'espace de touches actuellement sélectionné en utilisant `DESC TYPE`

```
USE my_keyspace;
my_keyspace DESC TYPES;

my_udt1  my_udt2
```

5. Pour répertorier tous les UDT dans un espace de touches multirégional, vous pouvez interroger la table système `types` dans l'espace de touches `system_multiregion_info`. La requête suivante en est un exemple.

```
SELECT keyspace_name, type_name, region, status FROM
system_multiregion_info.types WHERE keyspace_name = 'mykeyspace' AND table_name
= 'mytable';
```

Le résultat de cette commande ressemble à ceci.

keyspace_name	table_name	region	status
mykeyspace	mytable	us-east-1	ACTIVE
mykeyspace	mytable	ap-southeast-1	ACTIVE
mykeyspace	mytable	eu-west-1	ACTIVE

CLI

Affichez les types définis par l'utilisateur (UDT) à l'aide du AWS CLI

1. Pour répertorier les types disponibles dans un espace de touches, vous pouvez utiliser la `list-types` commande.

```
aws keyspaces list-types
--keyspace-name 'my_keyspace'
```

Le résultat de cette commande ressemble à celui de cet exemple.

```
{
  "types": [
    "my_udt",
    "parent_udt"
  ]
}
```

2. Pour afficher les détails d'un type donné, vous pouvez utiliser la `get-type` commande.

```
aws keyspaces get-type
--type-name 'my_udt'
--keyspace-name 'my_keyspace'
```

Le résultat de cette commande ressemble à celui de cet exemple.

```
{
  "keyspaceName": "my_keyspace",
  "typeName": "my_udt",
  "fieldDefinitions": [
    {
      "name": "a",
      "type": "int"
    },
    {
      "name": "b",
      "type": "text"
    }
  ],
  "lastModifiedTimestamp": 1721328225776,
```

```
"maxNestingDepth": 3
"status": "ACTIVE",
"directReferringTables": [],
"directParentTypes": [
  "parent_udt"
],
"keyspaceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
my_keyspace/"
}
```

Supprimer un type défini par l'utilisateur (UDT) dans Amazon Keyspaces

Pour supprimer un UDT dans un espace de touches, vous pouvez utiliser l'`DROP TYPE` instruction dans CQL, la `delete-type` commande associée à AWS CLI, ou la console.

Console

Supprimer un type défini par l'utilisateur (UDT) avec la console Amazon Keyspaces

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le volet de navigation, choisissez Keyspaces, puis choisissez un keyspace dans la liste.
3. Choisissez l'onglet UDTs.
4. Choisissez l'UDT que vous souhaitez supprimer. Sur Used in, vous pouvez confirmer que le type que vous souhaitez supprimer n'est pas actuellement utilisé par une table ou un autre UDT.
5. Choisissez Supprimer au-dessus du résumé.
6. Tapez `Delete` dans la boîte de dialogue qui apparaît, puis choisissez Supprimer l'UDT.

Cassandra Query Language (CQL)

Supprimer un type défini par l'utilisateur (UDT) avec CQL

- Pour supprimer un type, vous pouvez utiliser l'instruction suivante.

```
DROP TYPE my_keyspace.my_udt;
```

Pour plus d'informations sur la syntaxe CQL, consultez [the section called “DROP TYPE”](#).

CLI

Supprimez un type défini par l'utilisateur (UDT) à l'aide du AWS CLI

1. Pour supprimer un type, vous pouvez utiliser la commande suivante.

```
aws keyspaces delete-type
--keyspace-name 'my_keyspace'
--type-name 'my_udt'
```

2. Le résultat de la commande ressemble à celui de cet exemple.

```
{
  "keyspaceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
my_keyspace/",
  "typeName": "my_udt"
}
```

Utilisation de requêtes CQL dans Amazon Keyspaces

Cette section explique comment utiliser les requêtes dans Amazon Keyspaces (pour Apache Cassandra). Les instructions CQL disponibles pour interroger, transformer et gérer les données sont `SELECT`, `INSERT`, `UPDATE`, et `DELETE`. Les rubriques suivantes décrivent certaines des options les plus complexes disponibles lors de l'utilisation de requêtes. Pour la syntaxe complète du langage avec des exemples, consultez [the section called "Instructions DML"](#).

Rubriques

- [Utiliser l'opérateur IN avec l'instruction SELECT dans une requête dans Amazon Keyspaces](#)
- [Utiliser des instructions par lots dans Amazon Keyspaces](#)
- [Résultats de commande avec ORDER BY dans Amazon Keyspaces](#)
- [Paginer les résultats dans Amazon Keyspaces](#)

Utiliser l'**opérateur** IN avec l'**instruction** SELECT dans une requête dans Amazon Keyspaces

SÉLECTIONNEZ DANS

Vous pouvez interroger les données des tables à l'aide de l'`SELECT` instruction, qui lit une ou plusieurs colonnes pour une ou plusieurs lignes d'un tableau et renvoie un jeu de résultats contenant les lignes correspondant à la demande. Une `SELECT` instruction contient un `select_clause` qui détermine les colonnes à lire et à renvoyer dans le jeu de résultats. La clause peut contenir des instructions pour transformer les données avant de les renvoyer. La `WHERE` clause facultative spécifie quelles lignes doivent être interrogées et est composée de relations sur les colonnes qui font partie de la clé primaire. Amazon Keyspaces prend en charge le `IN` mot clé dans la `WHERE` clause. Cette section utilise des exemples pour montrer comment Amazon Keyspaces traite les `SELECT` instructions utilisant le `IN` mot clé.

Cet exemple montre comment Amazon Keyspaces décompose l'`SELECT` instruction contenant le `IN` mot clé en sous-requêtes. Dans cet exemple, nous utilisons une table portant le nom `my_keyspace.customers`. La table comporte une colonne de clé primaire `department_id`, deux colonnes `sales_region_id` de clustering et une colonne contenant le nom du client dans la `customer_name` colonne. `sales_representative_id`

```
SELECT * FROM my_keyspace.customers;
```

department_id	sales_region_id	sales_representative_id	customer_name
0	0	0	a
0	0	1	b
0	1	0	c
0	1	1	d
1	0	0	e
1	0	1	f
1	1	0	g
1	1	1	h

À l'aide de ce tableau, vous pouvez exécuter l'`SELECT` instruction suivante pour rechercher les clients des départements et des régions de vente qui vous intéressent en utilisant le `IN` mot clé de la `WHERE` clause. La déclaration suivante en est un exemple.

```
SELECT * FROM my_keyspace.customers WHERE department_id IN (0, 1) AND sales_region_id
IN (0, 1);
```

Amazon Keyspaces divise cette instruction en quatre sous-requêtes, comme indiqué dans le résultat suivant.

```
SELECT * FROM my_keyspace.customers WHERE department_id = 0 AND sales_region_id = 0;
```

department_id	sales_region_id	sales_representative_id	customer_name
0	0	0	a
0	0	1	b

```
SELECT * FROM my_keyspace.customers WHERE department_id = 0 AND sales_region_id = 1;
```

department_id	sales_region_id	sales_representative_id	customer_name
0	1	0	c
0	1	1	d

```
SELECT * FROM my_keyspace.customers WHERE department_id = 1 AND sales_region_id = 0;
```

department_id	sales_region_id	sales_representative_id	customer_name
1	0	0	e
1	0	1	f

```
SELECT * FROM my_keyspace.customers WHERE department_id = 1 AND sales_region_id = 1;
```

department_id	sales_region_id	sales_representative_id	customer_name
1	1	0	g
1	1	1	h

Lorsque le IN mot clé est utilisé, Amazon Keyspaces pagine automatiquement les résultats dans les cas suivants :

- Après le traitement d'une sous-requête sur 10.
- Après avoir traité 1 Mo d'E/S logiques.
- Si vous avez configuré unPAGE SIZE, Amazon Keyspaces pagine après avoir lu le nombre de requêtes à traiter en fonction de l'ensemble. PAGE SIZE
- Lorsque vous utilisez le LIMIT mot clé pour réduire le nombre de lignes renvoyées, Amazon Keyspaces pagine après avoir lu le nombre de requêtes à traiter en fonction de l'ensemble. LIMIT

Le tableau suivant est utilisé pour illustrer cela à l'aide d'un exemple.

Pour plus d'informations sur la pagination, consultez[the section called “Pagner les résultats”](#).

```
SELECT * FROM my_keyspace.customers;
```

department_id	sales_region_id	sales_representative_id	customer_name
2	0	0	g
2	1	1	h
2	2	2	i
0	0	0	a
0	1	1	b
0	2	2	c
1	0	0	d
1	1	1	e
1	2	2	f
3	0	0	j
3	1	1	k
3	2	2	l

Vous pouvez exécuter l'instruction suivante sur ce tableau pour voir comment fonctionne la pagination.

```
SELECT * FROM my_keyspace.customers WHERE department_id IN (0, 1, 2, 3) AND
sales_region_id IN (0, 1, 2) AND sales_representative_id IN (0, 1);
```

Amazon Keyspaces traite cette instruction comme 24 sous-requêtes, car la cardinalité du produit cartésien de tous les IN termes contenus dans cette requête est 24.

department_id	sales_region_id	sales_representative_id	customer_name
0	0	0	a
0	1	1	b
1	0	0	d
1	1	1	e

---MORE---

department_id	sales_region_id	sales_representative_id	customer_name
2	0	0	g
2	1	1	h
3	0	0	j

---MORE---

department_id	sales_region_id	sales_representative_id	customer_name
---------------	-----------------	-------------------------	---------------

-----+-----+-----+-----

3 | 1 | 1 | k

Cet exemple montre comment utiliser la ORDER BY clause dans une SELECT instruction avec le IN mot clé.

```
SELECT * FROM my_keyspace.customers WHERE department_id IN (3, 2, 1) ORDER BY sales_region_id DESC;
```

department_id	sales_region_id	sales_representative_id	customer_name
-----+-----+-----+-----			
3	2	2	l
3	1	1	k
3	0	0	j
2	2	2	i
2	1	1	h
2	0	0	g
1	2	2	f
1	1	1	e
1	0	0	d

Les sous-requêtes sont traitées dans l'ordre dans lequel les colonnes de la clé de partition et de la clé de clustering sont présentées dans la requête. Dans l'exemple ci-dessous, les sous-requêtes pour la valeur de clé de partition « 2 » sont traitées en premier, suivies des sous-requêtes pour la valeur de clé de partition « 3 » et « 1 ». Les résultats d'une sous-requête donnée sont classés en fonction de la clause de classement de la requête, si elle est présente, ou de l'ordre de classification de la table défini lors de la création de la table.

```
SELECT * FROM my_keyspace.customers WHERE department_id IN (2, 3, 1) ORDER BY sales_region_id DESC;
```

department_id	sales_region_id	sales_representative_id	customer_name
-----+-----+-----+-----			
2	2	2	i
2	1	1	h
2	0	0	g
3	2	2	l
3	1	1	k
3	0	0	j
1	2	2	f
1	1	1	e

1		0		0		d
---	--	---	--	---	--	---

Utiliser des instructions par lots dans Amazon Keyspaces

Vous pouvez combiner plusieurs DELETE opérations INSERTUPDATE, et dans une BATCH instruction. LOGGEDles lots sont les lots par défaut.

```
batch_statement ::=      BEGIN [ UNLOGGED ] BATCH
                        [ USING update_parameter( AND update_parameter)* ]
                        modification_statement ( ';' modification_statement )*
                        APPLY BATCH
modification_statement ::= insert_statement | update_statement | delete_statement
```

Lorsque vous exécutez une instruction par lots, le pilote combine toutes les instructions du lot en une seule opération par lots.

Pour décider du type d'opération par lots à utiliser, vous pouvez suivre les instructions suivantes.

Utilisez des lots enregistrés dans les cas suivants :

- Vous avez besoin de garanties de transaction atomiques.
- Des latences légèrement plus élevées constituent un compromis acceptable.

Utilisez des lots non enregistrés dans les cas suivants :

- Vous devez optimiser les opérations à partition unique.
- Vous souhaitez réduire la surcharge du réseau.
- Vous avez des exigences en matière de haut débit.

Pour plus d'informations sur les quotas de relevés par lots, consultez [Quotas](#).

Lots non enregistrés

Avec les lots non enregistrés, Amazon Keyspaces traite plusieurs opérations en une seule demande sans tenir à jour de journal des lots. Dans le cas d'une opération par lots non enregistrée, il est possible que certaines actions aboutissent alors que d'autres échouent. Les lots non enregistrés sont utiles lorsque vous souhaitez :

- Optimisez les opérations au sein d'une seule partition.
- Réduisez le trafic réseau en regroupant les demandes associées.

La syntaxe d'un lot non enregistré est similaire à celle d'un lot enregistré, avec l'ajout du UNLOGGED mot clé.

```
BEGIN UNLOGGED BATCH
  INSERT INTO users (id, firstname, lastname) VALUES (1, 'John', 'Doe');
  INSERT INTO users (id, firstname, lastname) VALUES (2, 'Jane', 'Smith');
APPLY BATCH;
```

Lots enregistrés

Un lot enregistré combine plusieurs actions d'écriture en une seule opération atomique. Lorsque vous exécutez un lot enregistré :

- Toutes les actions réussissent ensemble ou échouent ensemble.
- L'opération est synchrone et idempotente.
- Vous pouvez écrire dans plusieurs tables Amazon Keyspaces, à condition qu'elles se trouvent dans le même AWS compte et Région AWS.

Les lots enregistrés peuvent présenter des latences légèrement plus élevées. Pour les applications à haut débit, pensez à utiliser des lots non enregistrés.

L'utilisation des lots enregistrés dans Amazon Keyspaces est gratuite. Vous ne payez que pour les écritures qui font partie de vos opérations par lots. Amazon Keyspaces effectue deux écritures sous-jacentes sur chaque ligne du lot : l'une pour préparer la ligne pour le lot et l'autre pour valider le lot. Lorsque vous planifiez la capacité des tables qui utilisent des lots enregistrés, n'oubliez pas que chaque ligne d'un lot nécessite deux fois la capacité d'une opération d'écriture standard. Par exemple, si votre application exécute un lot enregistré par seconde avec trois lignes de 1 Ko, vous devez provisionner six unités de capacité d'écriture (WCU) contre seulement trois WCU pour les écritures individuelles ou les lots non enregistrés.

Pour plus d'informations sur les tarifs, consultez la rubrique Tarification d'[Amazon Keyspaces \(pour Apache Cassandra\)](#).

Utiliser des instructions préparées dans des lots enregistrés

Les lots enregistrés prennent en charge les instructions préparées (liées) avec une exécution atomique complète (tout ou rien). Vous pouvez combiner les performances des instructions préparées avec les garanties d'atomicité d'un lot enregistré.

Vous pouvez créer un lot consigné d'instructions préparées de deux manières. Les exemples suivants utilisent un pilote Java compatible avec Apache Cassandra.

Dans la première approche, vous préparez chaque instruction individuellement et vous ajoutez les instructions liées à un `BatchStatement`. Cette approche prend en charge les lots qui couvrent plusieurs tables dans le même AWS compte et Région AWS. L'exemple de code Java suivant illustre cette approche :

```
import com.datastax.driver.core.BatchStatement;
import com.datastax.driver.core.PreparedStatement;
import com.datastax.driver.core.Session;

PreparedStatement psInsert = session.prepare(
    "INSERT INTO my_keyspace.orders (order_id, status) VALUES (?, ?)");
PreparedStatement psUpdate = session.prepare(
    "UPDATE my_keyspace.inventory SET qty = ? WHERE sku = ?");

BatchStatement batch = new BatchStatement(BatchStatement.Type.LOGGED);
batch.add(psInsert.bind(orderId, "CONFIRMED"));
batch.add(psUpdate.bind(newQty, sku));

try {
    session.execute(batch);
} catch (Exception e) {
    // Handle the batch execution error.
}
```

Dans la deuxième approche, vous préparez un `BEGIN BATCH ... APPLY BATCH` bloc entier sous la forme d'une instruction unique, puis vous le liez et vous l'exécutez. Cette approche est la mieux adaptée à un lot dont les instructions ciblent une seule table. Pour préparer un lot couvrant plusieurs tables, utilisez la première approche. L'exemple de code Java suivant montre comment préparer un bloc de commandes complet sous la forme d'une seule instruction :

```
import com.datastax.driver.core.PreparedStatement;
import com.datastax.driver.core.Session;

PreparedStatement psBatch = session.prepare(
    "BEGIN BATCH " +
    "  INSERT INTO my_keyspace.orders (order_id, status) VALUES (?, ?); " +
    "  INSERT INTO my_keyspace.orders (order_id, status) VALUES (?, ?); " +
    "APPLY BATCH");
```

```
try {  
    session.execute(psBatch.bind(orderId1, "CONFIRMED", orderId2, "PENDING"));  
} catch (Exception e) {  
    // Handle the batch execution error.  
}
```

Vous pouvez également utiliser des instructions préparées dans un lot enregistré avec les éléments suivants :

- Déclarations conditionnelles (IF NOT EXISTS ou IF)
- Marqueurs de liaison nommés
- Un mélange de relevés préparés et non préparés dans le même lot
- Réutilisation d'une seule instruction préparée avec différentes valeurs limites dans le lot

Meilleures pratiques pour les opérations par lots

Prenez en compte les pratiques recommandées suivantes lorsque vous utilisez les opérations par lots Amazon Keyspaces.

- Activez la mise à l'échelle automatique afin de disposer d'une capacité de débit suffisante pour que vos tables puissent gérer les opérations par lots et répondre aux exigences de débit supplémentaires des lots enregistrés.
- Utilisez des opérations individuelles ou des lots non enregistrés lorsque les opérations peuvent être exécutées indépendamment sans affecter l'exactitude de l'application.
- Concevez votre application de manière à minimiser les mises à jour simultanées des mêmes lignes, car les opérations par lots simultanées peuvent entrer en conflit et échouer.
- Pour l'ingestion de données en masse à haut débit sans exigences d'atomicité, utilisez des opérations d'écriture individuelles ou des lots non enregistrés.

Cohérence et simultanéité

Amazon Keyspaces applique les règles de cohérence et de simultanéité suivantes pour les lots enregistrés :

- Toutes les opérations par lots utilisent un niveau de LOCAL_QUORUM cohérence.
- Les lots simultanés affectant différentes lignes peuvent être exécutés simultanément.

- Les DELETE opérations INSERT UPDATE simultanées ou effectuées sur les lignes impliquées dans un lot en cours échouent en raison d'un conflit.

Opérateurs et conditions pris en charge

Opérateurs de WHERE clause pris en charge :

- Égalité (=)

Opérateurs non pris en charge :

- Opérateurs de plage (>, <, >=, <=)
- Opérateur IN
- Opérateur LIKE
- Opérateur BETWEEN

Non pris en charge dans les lots enregistrés :

- Plusieurs instructions affectant la même ligne
- Opérations de contre-attaque
- Suppressions de plages

Une DELETE instruction d'un lot enregistré doit cibler une seule ligne en spécifiant la clé primaire complète (toutes les colonnes de clé de partition et de clustering) avec des conditions d'égalité.

Une suppression de plage, telle qu'une suppression par clé de partition seule ou sur une plage de colonnes de clustering, n'est pas prise en charge dans un lot enregistré. Pour exécuter une suppression de plage, utilisez une DELETE instruction autonome ou un lot non enregistré.

Conditions de défaillance des relevés de lots enregistrés

Une opération par lots enregistrée peut échouer dans l'un des cas suivants :

- Les expressions conditionnelles (comme IF NOT EXISTS ou IF) sont évaluées à False.
- Une ou plusieurs opérations contiennent des paramètres non valides.
- La demande entre en conflit avec une autre opération par lots exécutée sur les mêmes lignes.
- La table ne dispose pas d'une capacité provisionnée suffisante.
- La taille d'une ligne dépasse la limite de taille maximale.
- Le format des données d'entrée n'est pas valide.

Relevés par lots et réplication multirégionale

Dans les déploiements multirégionaux :

- Les opérations de la région source sont synchrones et atomiques.
- Les opérations de la région de destination sont asynchrones.
- Toutes les opérations par lots sont répliquées vers les régions de destination, mais peuvent ne pas maintenir l'isolement pendant l'application.

Surveiller les opérations par lots

Vous pouvez surveiller les opérations par lots à l'aide CloudWatch des métriques Amazon pour suivre les performances, les erreurs et les modèles d'utilisation. Amazon Keyspaces fournit les CloudWatch mesures suivantes pour surveiller les opérations par lots par table :

- `SuccessfulRequestCount`— Suivez les opérations par lots réussies.
- `Latency`— Mesurez les performances des opérations par lots.
- `ConsumedWriteCapacityUnits`— Surveillez la consommation de capacité des opérations par lots.

Pour de plus amples informations, veuillez consulter [the section called “Métriques Amazon Keyspaces”](#).

Outre les CloudWatch statistiques, vous pouvez les utiliser AWS CloudTrail pour enregistrer toutes les actions de l'API Amazon Keyspaces. Chaque action d'API du lot est enregistrée, CloudTrail ce qui facilite le suivi et l'audit des opérations par lots dans vos tables Amazon Keyspaces.

Exemples d'opérations par lots

Voici un exemple d'instruction de lot enregistrée de base.

```
BEGIN BATCH
  INSERT INTO users (id, firstname, lastname) VALUES (1, 'John', 'Doe');
  INSERT INTO users (id, firstname, lastname) VALUES (2, 'Jane', 'Smith');
APPLY BATCH;
```

Voici un exemple de lot qui inclut des DELETE instructions INSERTUPDATE, et.

```
BEGIN BATCH
  INSERT INTO users (id, firstname, lastname) VALUES (1, 'John', 'Doe');
  UPDATE users SET firstname = 'Johnny' WHERE id = 2;
  DELETE FROM users WHERE id = 3;
APPLY BATCH;
```

Voici un exemple de lot utilisant des horodatages côté client.

```
BEGIN BATCH
  INSERT INTO users (id, firstname, lastname) VALUES (1, 'John', 'Stiles') USING
  TIMESTAMP 1669069624;
  INSERT INTO users (id, firstname, lastname) VALUES (2, 'Jane', 'Doe') USING
  TIMESTAMP 1669069624;
APPLY BATCH;

BEGIN BATCH
  UPDATE users USING TIMESTAMP 1669069624 SET firstname = 'Carlos' WHERE id = 1;
  UPDATE users USING TIMESTAMP 1669069624 SET firstname = 'Diego' WHERE id = 2;
APPLY BATCH;
```

Voici un exemple de lot conditionnel.

```
BEGIN BATCH
  INSERT INTO users (id, firstname, lastname) VALUES (1, 'Jane', 'Doe') IF NOT
  EXISTS;
  INSERT INTO users (id, firstname, lastname) VALUES (2, 'John', 'Doe') IF NOT
  EXISTS;
APPLY BATCH;

BEGIN BATCH
  UPDATE users SET lastname = 'Stiles' WHERE id = 1 IF lastname = 'Doe';
  UPDATE users SET lastname = 'Stiles' WHERE id = 2 IF lastname = 'Doe';
APPLY BATCH;
```

Voici un exemple de lot utilisant Time to Live (TTL).

```
BEGIN BATCH
  INSERT INTO users (id, firstname, lastname) VALUES (1, 'John', 'Doe') USING TTL
  3600;
  INSERT INTO users (id, firstname, lastname) VALUES (2, 'Jane', 'Smith') USING TTL
  7200;
```



```
APPLY BATCH;
```

Voici un exemple d'instruction batch qui met à jour plusieurs tables.

```
BEGIN BATCH
  INSERT INTO users (id, firstname) VALUES (1, 'John');
  INSERT INTO user_emails (user_id, email) VALUES (1, 'john@example.com');
APPLY BATCH;
```

Voici un exemple d'opération par lots utilisant des types définis par l'utilisateur (UDT). L'exemple suppose que l'UDT `address` existe.

```
BEGIN BATCH
  INSERT INTO users (id, firstname, address)
  VALUES (1, 'John', {street: '123 Main St', city: 'NYC', zip: '10001'});
  INSERT INTO users (id, firstname, address)
  VALUES (2, 'Jane', {street: '456 Oak Ave', city: 'LA', zip: '90210'});
APPLY BATCH;

BEGIN BATCH
  UPDATE users SET address.zip = '10002' WHERE id = 1;
  UPDATE users SET address.city = 'Boston' WHERE id = 2;
APPLY BATCH;
```

Résultats de commande avec **ORDER BY** dans Amazon Keyspaces

La `ORDER BY` clause spécifie l'ordre de tri des résultats renvoyés dans une `SELECT` instruction. L'instruction prend une liste de noms de colonnes comme arguments et pour chaque colonne, vous pouvez spécifier l'ordre de tri des données. Vous ne pouvez spécifier des colonnes de clustering que dans les clauses de classement, les colonnes non clustering ne sont pas autorisées.

Les deux options d'ordre de tri disponibles pour les résultats renvoyés concernent `ASC` l'ordre de tri croissant et `DESC` de tri décroissant.

```
SELECT * FROM my_keyspace.my_table ORDER BY (col1 ASC, col2 DESC, col3 ASC);
```

col1	col2	col3
0	6	a
1	5	b
2	4	c

3		3		d
4		2		e
5		1		f
6		0		g

```
SELECT * FROM my_keyspace.my_table ORDER BY (col1 DESC, col2 ASC, col3 DESC);
```

col1		col2		col3
-----+-----+-----				
6		0		g
5		1		f
4		2		e
3		3		d
2		4		c
1		5		b
0		6		a

Si vous ne spécifiez pas l'ordre de tri dans l'instruction de requête, l'ordre par défaut de la colonne de clustering est utilisé.

Les ordres de tri possibles que vous pouvez utiliser dans une clause de classement dépendent de l'ordre de tri attribué à chaque colonne de clustering lors de la création de la table. Les résultats des requêtes ne peuvent être triés que dans l'ordre défini pour toutes les colonnes de clustering lors de la création de la table ou dans l'ordre inverse de l'ordre de tri défini. Les autres combinaisons possibles ne sont pas autorisées.

Par exemple, si la table `CLUSTERING ORDER` est `(col1 ASC, col2 DESC, col3 ASC)`, les paramètres valides pour `ORDER BY` sont soit `(col1 ASC, col2 DESC, col3 ASC)` soit `(col1 DESC, col2 ASC, col3 DESC)`. Pour plus d'informations sur `CLUSTERING ORDER`, voir `table_options` ci-dessous [the section called "CREATE TABLE"](#).

Paginer les résultats dans Amazon Keyspaces

Amazon Keyspaces pagine automatiquement les résultats des `SELECT` instructions lorsque les données lues pour traiter l'`SELECT` instruction dépassent 1 Mo. Avec la pagination, les résultats des `SELECT` relevés sont divisés en « pages » de données d'une taille de 1 Mo (ou moins). Une application peut traiter la première page des résultats, puis la deuxième, et ainsi de suite. Les clients doivent toujours vérifier la présence de jetons de pagination lorsqu'ils traitent `SELECT` des requêtes renvoyant plusieurs lignes.

Si un client fournit un document `PAGE SIZE` qui nécessite la lecture de plus de 1 Mo de données, Amazon Keyspaces divise automatiquement les résultats en plusieurs pages en fonction des incréments de lecture de données de 1 Mo.

Par exemple, si la taille moyenne d'une ligne est de 100 Ko et que vous spécifiez une valeur `PAGE SIZE` de 20, Amazon Keyspaces pagine automatiquement les données après avoir lu 10 lignes (1 000 Ko de données lues).

Comme Amazon Keyspaces pagine les résultats en fonction du nombre de lignes lues pour traiter une demande et non du nombre de lignes renvoyées dans le jeu de résultats, certaines pages peuvent ne contenir aucune ligne si vous exécutez des requêtes filtrées.

Par exemple, si vous définissez `PAGE SIZE` la valeur 10 et que Keyspaces évalue 30 lignes pour traiter votre `SELECT` requête, Amazon Keyspaces renverra trois pages. Si seul un sous-ensemble de lignes correspond à votre requête, certaines pages peuvent contenir moins de 10 lignes. Pour un exemple `PAGE SIZE` de la manière dont les `LIMIT` requêtes peuvent affecter la capacité de lecture, consultez [the section called “Estimer la consommation de capacité de lecture des requêtes limitées”](#).

Pour une comparaison avec la pagination d'Apache Cassandra, voir. [the section called “Pagination”](#)

Utilisation des flux de capture des données modifiées (CDC) dans Amazon Keyspaces

La capture des données de modification (CDC) d'Amazon Keyspaces enregistre les événements de modification au niveau des lignes à partir d'une table Amazon Keyspaces en temps quasi réel.

Amazon Keyspaces CDC permet des cas d'utilisation axés sur des événements tels que l'IoT industriel et la détection des fraudes, ainsi que des cas d'utilisation liés au traitement des données tels que la recherche en texte intégral et l'archivage des données. Les événements de changement capturés par Amazon Keyspaces CDC dans les flux peuvent être utilisés par les applications en aval qui exécutent des fonctions critiques telles que l'analyse des données, la recherche de texte, le machine learning training/inference et les sauvegardes continues de données à des fins d'archivage. Par exemple, vous pouvez transférer des données de flux vers des services AWS d'analyse et de stockage tels qu'Amazon OpenSearch Service, Amazon Redshift et Amazon S3 pour un traitement ultérieur.

Amazon Keyspaces CDC propose des enregistrements de modifications triés dans le temps et dédoublés pour les tables, avec une mise à l'échelle automatique du débit de données et une durée de conservation allant jusqu'à 24 heures.

Les flux CDC d'Amazon Keyspaces sont totalement sans serveur et vous n'avez pas besoin de gérer l'infrastructure de données pour capturer les événements de changement. En outre, Amazon Keyspaces CDC ne consomme aucune capacité de table, que ce soit pour le calcul ou le stockage. Pour de plus amples informations, veuillez consulter [the section called “Comment fonctionnent les flux CDC”](#).

Vous pouvez utiliser l'API Amazon Keyspaces Streams pour créer des applications qui utilisent les flux CDC d'Amazon Keyspaces et prennent des mesures en fonction de leur contenu. Pour les points de terminaison disponibles, voir [the section called “Accédez aux points de terminaison des flux CDC”](#).

Pour une liste complète de toutes les opérations disponibles pour Amazon Keyspaces dans l'API Streams, consultez le manuel de référence de l'API Amazon [Keyspaces](#) Streams.

Rubriques

- [Comment fonctionnent les flux de saisie des données de modification \(CDC\) dans Amazon Keyspaces](#)
- [Comment utiliser les flux de saisie des données de modification \(CDC\) dans Amazon Keyspaces](#)

Comment fonctionnent les flux de saisie des données de modification (CDC) dans Amazon Keyspaces

Cette section fournit une vue d'ensemble du fonctionnement des flux de saisie des données de modification (CDC) dans Amazon Keyspaces.

Amazon Keyspaces Change Data Capture (CDC) enregistre une séquence ordonnée de modifications au niveau des lignes dans les tables Amazon Keyspaces et stocke ces informations dans un journal appelé `stream` pendant 24 heures maximum. Chaque modification au niveau de la ligne génère un nouvel enregistrement CDC qui contient les informations de la colonne clé principale ainsi que les états « avant » et « après » de la ligne, y compris toutes les colonnes. Les applications peuvent accéder au flux et visualiser les mutations en temps quasi réel.

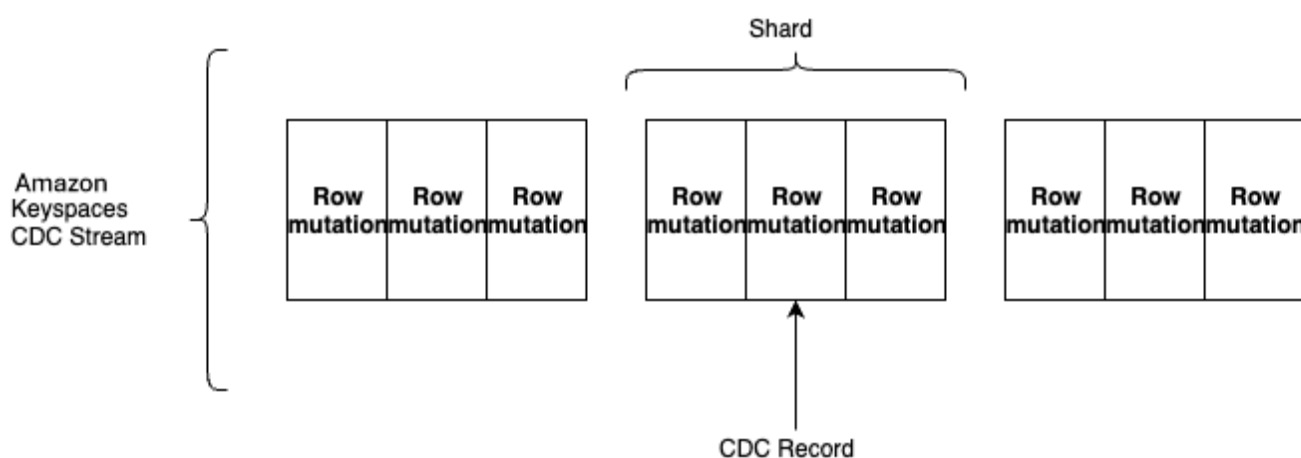
Lorsque vous activez le CDC sur votre table, Amazon Keyspaces crée un nouveau flux CDC et commence à capturer des informations sur chaque modification de la table. Le flux CDC possède un Amazon Resource Name (ARN) au format suivant :

```
arn:${Partition}:cassandra:{Region}:${Account}:/keyspace/${keyspaceName}/table/
${tableName}/stream/${streamLabel}
```

Vous pouvez sélectionner le type d'informations ou le type de vue que le flux CDC collecte pour chaque enregistrement lorsque vous activez le flux CDC pour la première fois. Vous ne pourrez pas modifier le type de vue du flux par la suite. Amazon Keyspaces prend en charge les types de vues suivants :

- **NEW_AND_OLD_IMAGES**— Capture les versions de la ligne avant et après la mutation. Il s'agit de l'option par défaut.
- **NEW_IMAGE**— Capture la version de la ligne après la mutation.
- **OLD_IMAGE**— Capture la version de la ligne avant la mutation.
- **KEYS_ONLY**— Capture les clés de partition et de clustering de la ligne qui a été mutée.

Chaque flux CDC est composé d'enregistrements. Chaque enregistrement représente une modification d'une seule ligne dans un tableau Amazon Keyspaces. Les enregistrements sont organisés de manière logique en groupes appelés « shards ». Ces groupes sont organisés de manière logique par plages de clés primaires (combinaison de clés de partition, plages de clés de clustering) et constituent une structure interne d'Amazon Keyspaces. Chaque fragment fait office de conteneur pour plusieurs enregistrements et contient les informations nécessaires pour accéder à ces enregistrements et les parcourir à chaque fois.



Un numéro de séquence est attribué à chaque enregistrement CDC, reflétant l'ordre dans lequel l'enregistrement a été publié dans le fragment. Il est garanti que le numéro de séquence augmente et qu'il est unique au sein de chaque fragment.

Amazon Keyspaces crée et supprime des partitions automatiquement. En fonction des charges de trafic, Amazon Keyspaces peut également fractionner ou fusionner des fragments au fil du temps. Par exemple, Amazon Keyspaces peut diviser une partition en plusieurs nouvelles partitions ou fusionner des partitions en une seule nouvelle partition. Les API Amazon Keyspaces publient les informations de la partition et du flux CDC pour permettre aux applications utilisatrices de traiter les enregistrements dans le bon ordre en accédant à l'intégralité du graphe de lignage d'une partition.

Amazon Keyspaces CDC repose sur les principes suivants sur lesquels vous pouvez vous appuyer lors de la création de votre application :

- Chaque enregistrement de mutation au niveau de la ligne apparaît exactement une fois dans le flux CDC.
- Lorsque vous consommez des fragments par ordre de lignée, chaque enregistrement de mutation au niveau de la ligne apparaît dans la même séquence que l'ordre de mutation réel sur la clé primaire.

Rubriques

- [Comment fonctionne la conservation des données pour les flux CDC dans Amazon Keyspaces](#)
- [Comment fonctionne l'expiration des données Time to Live \(TTL\) avec les flux CDC dans Amazon Keyspaces](#)
- [Comment fonctionnent les opérations par lots pour les flux CDC dans Amazon Keyspaces](#)
- [Fonctionnement des colonnes statiques dans les flux CDC d'Amazon Keyspaces](#)
- [Comment fonctionne le chiffrement au repos pour les flux CDC dans Amazon Keyspaces](#)
- [Comment fonctionne la réplication multirégionale pour les flux CDC dans Amazon Keyspaces](#)
- [Flux CDC et intégration avec AWS services](#)

Comment fonctionne la conservation des données pour les flux CDC dans Amazon Keyspaces

Amazon Keyspaces conserve les enregistrements du flux CDC pendant 24 heures. Vous ne pouvez pas modifier la durée de conservation. Si vous désactivez le CDC sur une table, les données du flux restent lisibles pendant 24 heures. Passé ce délai, les données expirent et les enregistrements sont automatiquement supprimés.

Comment fonctionne l'expiration des données Time to Live (TTL) avec les flux CDC dans Amazon Keyspaces

Amazon Keyspaces affiche l'heure d'expiration au column/cell niveau et au niveau de la ligne dans un champ de métadonnées appelé `expirationTime` dans les enregistrements de modification du CDC. Lorsqu'Amazon Keyspaces TTL détecte l'expiration d'une cellule, le CDC crée un nouvel enregistrement de modification indiquant TTL comme origine de la modification. Pour plus d'informations sur TTL, consultez [the section called “Expirer les données avec Time to Live”](#).

Comment fonctionnent les opérations par lots pour les flux CDC dans Amazon Keyspaces

Les opérations par lots sont divisées en interne en modifications individuelles au niveau des lignes. Amazon Keyspaces conserve tous les enregistrements des flux CDC au niveau de la ligne, même si la modification a eu lieu lors d'une opération par lots. Amazon Keyspaces maintient l'ordre des enregistrements dans le flux CDC dans le même ordre que l'ordre de mutation qui s'est produit au niveau de la ligne ou sur la clé primaire.

Fonctionnement des colonnes statiques dans les flux CDC d'Amazon Keyspaces

Les valeurs de colonne statiques sont partagées entre toutes les lignes d'une partition dans Cassandra. En raison de ce comportement, Amazon Keyspaces capture toutes les mises à jour apportées à une colonne statique sous la forme d'un enregistrement distinct dans le flux CDC. Les exemples suivants résument le comportement des mutations de colonnes statiques :

- Lorsque seule la colonne statique est mise à jour, le flux CDC contient une modification de ligne pour la colonne statique en tant que seule colonne de la ligne.
- Lorsqu'une ligne est mise à jour sans aucune modification de la colonne statique, le flux CDC contient une modification de ligne qui contient toutes les colonnes à l'exception de la colonne statique.
- Lorsqu'une ligne est mise à jour en même temps que la colonne statique, le flux CDC contient deux modifications de ligne distinctes, l'une pour la colonne statique et l'autre pour le reste de la ligne.

Comment fonctionne le chiffrement au repos pour les flux CDC dans Amazon Keyspaces

Pour chiffrer les données au repos dans le journal ordonné par le CDC, Amazon Keyspaces utilise la même clé de chiffrement que celle déjà utilisée pour la table. Pour plus d'informations sur le chiffrement au repos, consultez [the section called “Chiffrement au repos”](#).

Comment fonctionne la réplication multirégionale pour les flux CDC dans Amazon Keyspaces

Vous pouvez activer et désactiver les flux CDC pour les répliques individuelles d'une table multirégion à l'aide de l'`update-tableAPI` ou de la commande `ALTER TABLE CQL`. En raison de la réplication asynchrone et de la résolution des conflits, les flux CDC pour les tables multirégions ne sont pas cohérents entre les tables. Régions AWS Par conséquent, les enregistrements capturés par Amazon Keyspaces dans le flux peuvent apparaître dans un ordre différent selon les régions.

Pour plus d'informations sur la réplication multirégion, consultez [the section called “Multi-Region réplication”](#).

Flux CDC et intégration avec AWS services

Comment utiliser les points de terminaison VPC pour les flux CDC dans Amazon Keyspaces

Vous pouvez utiliser des points de terminaison VPC pour accéder aux flux CDC d'Amazon Keyspaces. Pour plus d'informations sur la création et l'accès à des points de terminaison VPC pour les flux, consultez. [the section called “Utilisation de points de terminaison VPC d'interface avec des flux”](#)

Comment CloudWatch fonctionne la surveillance des flux CDC dans Amazon Keyspaces

Vous pouvez utiliser Amazon CloudWatch pour surveiller les appels d'API effectués vers le point de terminaison Amazon Keyspaces CDC. Pour plus d'informations sur les mesures disponibles, consultez [the section called “Mesures relatives à la capture des données de modification par Amazon Keyspaces \(CDC\)”](#).

Comment CloudTrail fonctionne la journalisation pour les flux CDC dans Amazon Keyspaces

Amazon Keyspaces CDC est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans Amazon Keyspaces.

CloudTrail capture les appels d'API DDL (Data Definition Language) et les appels d'API DML (Data Manipulation Language) pour Amazon Keyspaces sous forme d'événements. Les appels capturés incluent les appels provenant de la console Amazon Keyspaces et les appels programmatiques vers les opérations de l'API Amazon Keyspaces.

Pour plus d'informations sur les événements CDC capturés par CloudTrail, consultez [the section called “Se connecter avec CloudTrail”](#).

Comment fonctionne le balisage pour les flux CDC dans Amazon Keyspaces

Les flux CDC d'Amazon Keyspaces constituent une ressource balisable. Vous pouvez baliser un flux lorsque vous créez une table par programmation à l'aide de CQL, du AWS SDK ou du AWS CLI. Vous pouvez également baliser des flux existants, supprimer des balises ou afficher les balises d'un flux. Pour de plus amples informations, veuillez consulter [the section called “Étiquetez les espaces clés, les tables et les flux”](#).

Comment utiliser les flux de saisie des données de modification (CDC) dans Amazon Keyspaces

Rubriques

- [Configurer les autorisations pour utiliser les flux CDC dans Amazon Keyspaces](#)
- [Comment accéder aux points de terminaison des flux CDC dans Amazon Keyspaces](#)
- [Activer un flux CDC lors de la création d'une nouvelle table dans Amazon Keyspaces](#)
- [Activer un flux CDC pour une table existante dans Amazon Keyspaces](#)
- [Désactiver un flux CDC dans Amazon Keyspaces](#)
- [Afficher les flux CDC dans Amazon Keyspaces](#)
- [Accédez aux enregistrements dans les flux CDC dans Amazon Keyspaces](#)
- [Utiliser la bibliothèque cliente Kinesis \(KCL\) pour traiter les flux Amazon Keyspaces](#)

Configurer les autorisations pour utiliser les flux CDC dans Amazon Keyspaces

Pour activer les flux CDC, le principal, par exemple un utilisateur ou un rôle IAM, a besoin des autorisations suivantes.

Pour plus d'informations sur Gestion des identités et des accès AWS, consultez [the section called “Gestion des identités et des accès AWS”](#).

Autorisations pour activer un flux CDC pour une table

Pour activer un flux CDC pour une table Amazon Keyspaces, le principal doit d'abord avoir les autorisations nécessaires pour créer ou modifier une table, puis les autorisations nécessaires pour créer le rôle [AWSServiceRoleForAmazonKeyspacesCDC](#) lié au service. Amazon Keyspaces utilise le rôle lié au service pour publier CloudWatch des statistiques sur votre compte en votre nom.

La politique IAM suivante en est un exemple.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cassandra:Create",
        "cassandra:CreateMultiRegionResource",
        "cassandra:Alter",
        "cassandra:AlterMultiRegionResource"
      ],
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/*",
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
      ]
    },
    {
      "Sid": "KeyspacesCDCServiceLinkedRole",
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "arn:aws:iam::*:role/aws-service-role/cassandra-streams.amazonaws.com/AWSServiceRoleForAmazonKeyspacesCDC",
      "Condition": {
        "StringLike": {
          "iam:AWSServiceName": "cassandra-streams.amazonaws.com"
        }
      }
    }
  ]
}
```

Pour désactiver un flux, seules ALTER TABLE des autorisations sont requises.

Autorisations pour afficher un flux CDC

Pour afficher ou répertorier les flux CDC, le principal a besoin d'autorisations de lecture pour l'espace clé du système. Pour de plus amples informations, veuillez consulter [the section called “system_schema_mcs”](#).

La politique IAM suivante en est un exemple.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "cassandra:Select",
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
      ]
    }
  ]
}
```

Pour afficher ou répertorier les flux CDC avec l'API Amazon Keyspaces AWS CLI ou l'API Amazon Keyspaces, le principal a besoin d'autorisations supplémentaires pour les actions `cassandra:ListStreams` et `cassandra:GetStream`.

La politique IAM suivante en est un exemple.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cassandra:Select",
        "cassandra:ListStreams",
        "cassandra:GetStream"
      ],
      "Resource": "*"
    }
  ]
}
```

Autorisations pour lire un flux CDC

Pour lire les flux CDC, le principal a besoin des autorisations suivantes.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cassandra:GetStream",
        "cassandra:GetShardIterator",
        "cassandra:GetRecords"
      ],
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/
table/my_table/stream/stream_label"
      ]
    }
  ]
}
```

Autorisations pour traiter les flux Amazon Keyspaces CDC avec la bibliothèque cliente Kinesis (KCL)

Pour traiter les flux CDC d'Amazon Keyspaces avec KCL, le principal IAM doit disposer des autorisations suivantes.

- Amazon Keyspaces— Read-only accès à un flux Amazon Keyspaces CDC spécifié.
- DynamoDB— Autorisations permettant de créer shard lease des tables, d'accéder aux tables en lecture et en écriture et d'accéder en lecture à l'index, selon les besoins du traitement des flux KCL.
- CloudWatch— Autorisations pour publier des données métriques provenant d'Amazon Keyspaces Le CDC diffuse le traitement avec KCL dans l'espace de noms de votre application cliente KCL sur votre compte. CloudWatch Pour plus d'informations sur la surveillance, consultez [Surveiller la bibliothèque cliente Kinesis avec Amazon CloudWatch](#).

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```

    "Action":[
        "cassandra:GetStream",
        "cassandra:GetShardIterator",
        "cassandra:GetRecords"
    ],
    "Resource":[
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/
table/my_table/stream/stream_label"
    ]
},
{
    "Effect":"Allow",
    "Action":[
        "dynamodb:CreateTable",
        "dynamodb:DescribeTable",
        "dynamodb:UpdateTable",
        "dynamodb:GetItem",
        "dynamodb:UpdateItem",
        "dynamodb:PutItem",
        "dynamodb>DeleteItem",
        "dynamodb:Scan"
    ],
    "Resource":[
        "arn:aws:dynamodb:us-east-1:111122223333:table/KCL_APPLICATION_NAME"
    ]
},
{
    "Effect":"Allow",
    "Action":[
        "dynamodb:CreateTable",
        "dynamodb:DescribeTable",
        "dynamodb:GetItem",
        "dynamodb:UpdateItem",
        "dynamodb:PutItem",
        "dynamodb>DeleteItem",
        "dynamodb:Scan"
    ],
    "Resource":[
        "arn:aws:dynamodb:us-east-1:111122223333:table/KCL_APPLICATION_NAME-
WorkerMetricStats",
        "arn:aws:dynamodb:us-east-1:111122223333:table/KCL_APPLICATION_NAME-
CoordinatorState"
    ]
},

```

```
{
  "Effect": "Allow",
  "Action": [
    "dynamodb:Query"
  ],
  "Resource": [
    "arn:aws:dynamodb:us-east-1:111122223333:table/KCL_APPLICATION_NAME/index/*"
  ]
},
{
  "Effect": "Allow",
  "Action": [
    "cloudwatch:PutMetricData"
  ],
  "Resource": "*"
}
]
```

Comment accéder aux points de terminaison des flux CDC dans Amazon Keyspaces

Amazon Keyspaces gère des [points de terminaison distincts](#) pour keyspaces/tables et pour les flux CDC dans chaque zone Région AWS où Amazon Keyspaces est disponible. Pour accéder à un flux CDC, sélectionnez la région de la table et remplacez le `cassandra` préfixe par le nom du point de terminaison, comme indiqué dans l'exemple suivant : `cassandra-streams`

Nom de la région	Région	Point de terminaison pour les espaces clés et les tableaux	Point de terminaison pour les flux
USA Est (Virginie du Nord)	us-east-1	<i>cassandra</i> .us-east-1.api.aws	<i>cassandra-streams</i> .us-east-1.api.aws

Le tableau suivant contient une liste complète des points de terminaison publics disponibles pour Amazon Keyspaces change data capture streams. Amazon Keyspaces CDC streams prend en charge IPv4 et IPv6. Tous les points de terminaison publics, par exemple `cassandra-streams.us-`

east-1.api.aws, sont des points de terminaison à double pile qui peuvent être configurés pour IPv4 et IPv6.

Nom de la région	Région	Point de terminaison	Protocole	
US East (Ohio)	us-east-2	cassandra-streams.us-east-2.api.aws	HTTPS	
USA Est (Virginie du Nord)	us-east-1	cassandra-streams.us-east-1.api.aws	HTTPS	
USA Ouest (Californie du Nord)	us-west-1	cassandra-streams.us-west-1.api.aws	HTTPS	
US West (Oregon)	us-west-2	cassandra-streams.us-west-2.api.aws	HTTPS	
Afrique (Le Cap)	af-south-1	cassandra-streams.af-south-1.api.aws	HTTPS	
Asie-Pacifique (Hong Kong)	ap-east-1	cassandra-streams.ap-east-1.api.aws	HTTPS	
Asie-Pacifique (Hyderabad)	ap-south-2	cassandra-streams.ap-south-2.api.aws	HTTPS	
Asie-Pacifique (Jakarta)	ap-southeast-3	cassandra-streams.ap-southeast-3.api.aws	HTTPS	

Nom de la région	Région	Point de terminaison	Protocole	
Asie-Pacifique (Malaisie)	ap-southeast-5	cassandra-streams.ap-southeast-5.api.aws	HTTPS	
Asie-Pacifique (Melbourne)	ap-southeast-4	cassandra-streams.ap-southeast-4.api.aws	HTTPS	
Asia Pacific (Mumbai)	ap-south-1	cassandra-streams.ap-south-1.api.aws	HTTPS	
Asie-Pacifique (Nouvelle Zélande)	ap-southeast-6	cassandra-streams.ap-southeast-6.api.aws	HTTPS	
Asie-Pacifique (Osaka)	ap-northeast-3	cassandra-streams.ap-northeast-3.api.aws	HTTPS	
Asia Pacific (Seoul)	ap-northeast-2	cassandra-streams.ap-northeast-2.api.aws	HTTPS	
Asie-Pacifique (Singapour)	ap-southeast-1	cassandra-streams.ap-southeast-1.api.aws	HTTPS	
Asia Pacific (Sydney)	ap-southeast-2	cassandra-streams.ap-southeast-2.api.aws	HTTPS	

Nom de la région	Région	Point de terminaison	Protocole	
Asie-Pacifique (Taïpei)	ap-east-2	cassandra-streams.ap-east-2.api.aws	HTTPS	
Asie-Pacifique (Thaïlande)	ap-southeast-7	cassandra-streams.ap-southeast-7.api.aws	HTTPS	
Asie-Pacifique (Tokyo)	ap-northeast-1	cassandra-streams.ap-northeast-1.api.aws	HTTPS	
Canada (Centre)	ca-central-1	cassandra-streams.ca-central-1.api.aws	HTTPS	
Canada-Ouest (Calgary)	ca-west-1	cassandra-streams.ca-west-1.api.aws	HTTPS	
Europe (Francfort)	eu-central-1	cassandra-streams.eu-central-1.api.aws	HTTPS	
Europe (Irlande)	eu-west-1	cassandra-streams.eu-west-1.api.aws	HTTPS	
Europe (Londres)	eu-west-2	cassandra-streams.eu-west-2.api.aws	HTTPS	
Europe (Milan)	eu-south-1	cassandra-streams.eu-south-1.api.aws	HTTPS	
Europe (Paris)	eu-west-3	cassandra-streams.eu-west-3.api.aws	HTTPS	

Nom de la région	Région	Point de terminaison	Protocole	
Europe (Espagne)	eu-south-2	cassandra-streams.eu-south-2.api.aws	HTTPS	
Europe (Stockholm)	eu-north-1	cassandra-streams.eu-north-1.api.aws	HTTPS	
Europe (Zurich)	eu-central-2	cassandra-streams.eu-central-2.api.aws	HTTPS	
Israël (Tel Aviv)	il-central-1	cassandra-streams.il-central-1.api.aws	HTTPS	
Mexique (Centre)	mx-central-1	cassandra-streams.mx-central-1.api.aws	HTTPS	
Moyen-Orient (Bahreïn)	me-south-1	cassandra-streams.me-south-1.api.aws	HTTPS	
Moyen-Orient (EAU)	me-central-1	cassandra-streams.me-central-1.api.aws	HTTPS	
Amérique du Sud (São Paulo)	sa-east-1	cassandra-streams.sa-east-1.api.aws	HTTPS	
AWS GovCloud (US-East)	us-gov-east-1	cassandra-streams.us-gov-east-1.api.aws	HTTPS	

Nom de la région	Région	Point de terminaison	Protocole	
AWS GovCloud (US-West)	us-gov-west-1	cassandra-streams.us-gov-west-1.api.aws	HTTPS	

Activer un flux CDC lors de la création d'une nouvelle table dans Amazon Keyspaces

Pour activer un flux CDC lorsque vous créez une table, vous pouvez utiliser l'`CREATE TABLE` instruction dans CQL ou la `create-table` commande avec le AWS CLI.

Pour chaque ligne modifiée du tableau, Amazon Keyspaces peut capturer les modifications suivantes en fonction `view_type` de celles `cdc_specification` que vous avez sélectionnées :

- `NEW_AND_OLD_IMAGES`— les deux versions de la ligne, avant et après la modification. Il s'agit de l'option par défaut.
- `NEW_IMAGE`— la version de la ligne après la modification.
- `OLD_IMAGE`— la version de la ligne avant la modification.
- `KEYS_ONLY`— les clés de partition et de clustering de la ligne qui a été modifiée.

Pour plus d'informations sur la façon de baliser un flux, consultez [the section called “Créer un flux avec des balises avec une nouvelle table”](#).

Note

Amazon Keyspaces CDC nécessite la présence d'un rôle lié à un service (`AWSServiceRoleForAmazonKeyspacesCDC`) qui publie les données métriques issues des flux Amazon Keyspaces CDC sur votre CloudWatch compte "`cloudwatch:namespace`": "`AWS/Cassandra`" en votre nom. Ce rôle est créé automatiquement pour vous. Pour de plus amples informations, veuillez consulter [the section called “Flux CDC”](#).

Cassandra Query Language (CQL)

Activez un flux CDC lorsque vous créez une table avec CQL

1.

```
CREATE TABLE mykeyspace.mytable (a text, b text, PRIMARY KEY(a))
WITH CUSTOM_PROPERTIES={'cdc_specification': {'view_type': 'NEW_IMAGE'}} AND CDC
= TRUE;
```
2. Pour confirmer les paramètres de diffusion, vous pouvez utiliser l'instruction suivante.

```
SELECT keyspace_name, table_name, cdc, custom_properties FROM
system_schema_mcs.tables WHERE keyspace_name = 'mykeyspace' AND table_name =
'mytable';
```

Le résultat de cette instruction devrait ressembler à celui-ci.

```
SELECT keyspace_name, table_name, cdc, custom_properties FROM
system_schema_mcs.tables WHERE keyspace_name = 'mykeyspace' AND table_name =
'mytable';keyspace_name | table_name | cdc | custom_properties
-----+-----+-----
+-----+-----+-----+-----
mykeyspace | mytable | True | {'capacity_mode':
{'last_update_to_pay_per_request_timestamp': '1741383893782',
'throughput_mode': 'PAY_PER_REQUEST'}, 'cdc_specification':
{'latest_stream_arn': 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/
mykeyspace/table/mytable/stream/2025-03-07T21:44:53.783', 'status': 'ENABLED',
'view_type': 'NEW_IMAGE'}, 'encryption_specification': {'encryption_type':
'AWS_OWNED_KMS_KEY'}, 'point_in_time_recovery': {'status': 'disabled'}}>
```

CLI

Activez un flux CDC lorsque vous créez une table avec AWS CLI

1. Pour créer un flux, vous pouvez utiliser la syntaxe suivante.

```
aws keyspaces create-table \
--keyspace-name 'mykeyspace' \
--table-name 'mytable' \
--schema-definition 'allColumns=[{name=a,type=text},
{name=b,type=text}],partitionKeys=[{name=a}]' \
```

```
--cdc-specification status=ENABLED,viewType=NEW_IMAGE
```

2. Le résultat de cette commande affiche la `create-table` réponse standard et ressemble à cet exemple.

```
{ "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/  
table/mytable" }
```

Activer un flux CDC pour une table existante dans Amazon Keyspaces

Pour activer un flux CDC pour une table existante, vous pouvez utiliser l'`ALTER TABLE` instruction dans CQL, la `update-table` commande avec le AWS CLI, ou vous pouvez utiliser la console.

Pour chaque ligne modifiée du tableau, Amazon Keyspaces peut capturer les modifications suivantes en fonction `view_type` de celles `cdc_specification` que vous avez sélectionnées :

- `NEW_AND_OLD_IMAGES`— les deux versions de la ligne, avant et après la modification. Il s'agit de l'option par défaut.
- `NEW_IMAGE`— la version de la ligne après la modification.
- `OLD_IMAGE`— la version de la ligne avant la modification.
- `KEYS_ONLY`— les clés de partition et de clustering de la ligne qui a été modifiée.

Pour plus d'informations sur la façon de baliser un flux, consultez [the section called “Ajouter de nouveaux tags à un stream”](#).

Note

Amazon Keyspaces CDC nécessite la présence d'un rôle lié à un service (`AWSServiceRoleForAmazonKeyspacesCDC`) qui publie les données métriques issues des flux Amazon Keyspaces CDC sur votre CloudWatch compte `"cloudwatch:namespace": "AWS/Cassandra"` en votre nom. Ce rôle est créé automatiquement pour vous. Pour de plus amples informations, veuillez consulter [the section called “Flux CDC”](#).

Cassandra Query Language (CQL)

Activer un flux (flux CDC) avec CQL

Vous pouvez l'utiliser `ALTER TABLE` pour activer un flux pour une table existante.

1. L'exemple suivant crée un flux qui capture uniquement les modifications apportées aux clés de partition et de clustering d'une ligne modifiée.

```
ALTER TABLE mykeyspace.mytable
WITH cdc = TRUE
AND CUSTOM_PROPERTIES={'cdc_specification': {'view_type': 'KEYS_ONLY'}};
```

2. Pour vérifier les paramètres de diffusion, vous pouvez utiliser l'instruction suivante.

```
SELECT keyspace_name, table_name, cdc, custom_properties FROM
system_schema_mcs.tables WHERE keyspace_name = 'mykeyspace' AND table_name =
'mytable';
```

Le résultat de l'instruction ressemble à ceci.

```
keyspace_name | table_name | cdc | custom_properties
-----+-----+-----
+-----+-----+-----
mykeyspace | mytable | True | {'capacity_mode':
{'last_update_to_pay_per_request_timestamp': '1741385897045',
'throughput_mode': 'PAY_PER_REQUEST'}, 'cdc_specification':
{'latest_stream_arn': 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/
mykeyspace/table/mytable/stream/2025-03-07T22:20:10.454', 'status': 'ENABLED',
'view_type': 'KEYS_ONLY'}, 'encryption_specification': {'encryption_type':
'AWS_OWNED_KMS_KEY'}, 'point_in_time_recovery': {'status': 'disabled'}}
```

CLI

Créez un flux CDC avec le AWS CLI

1. Pour créer un flux pour une table existante, vous pouvez utiliser la syntaxe suivante.

```
aws keyspaces update-table \
--keyspace-name 'mykeyspace' \
--table-name 'mytable' \
```

```
--cdc-specification status=ENABLED,viewType=NEW_AND_OLD_IMAGES
```

2. Le résultat de cette commande affiche la `create-table` réponse standard et ressemble à cet exemple.

```
{ "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/
table/mytable" }
```

Console

Activez un flux CDC à l'aide de la console Amazon Keyspaces

1. Connectez-vous au Console de gestion AWS, puis ouvrez la console Amazon Keyspaces à <https://console.aws.amazon.com/keyspaces/home> l'adresse.
2. Dans le volet de navigation, choisissez **Tables**, puis choisissez un tableau dans la liste.
3. Choisissez l'onglet **Streams**.
4. Choisissez **Modifier** pour activer un flux.
5. Sélectionnez **Activer les diffusions**.
6. Choisissez le type d'affichage du flux. Les options suivantes sont disponibles. Notez que vous ne pouvez pas modifier le type d'affichage d'un flux une fois celui-ci créé.
 - **Nouvelles et anciennes images** : Amazon Keyspaces capture les deux versions de la ligne, avant et après la modification. Il s'agit de l'option par défaut.
 - **Nouvelle image** : Amazon Keyspaces capture uniquement la version de la ligne après la modification.
 - **Ancienne image** : Amazon Keyspaces capture uniquement la version de la ligne avant la modification.
 - **Clé primaire uniquement** : Amazon Keyspaces capture uniquement les colonnes clés de partition et de clustering de la ligne qui a été modifiée.
7. Pour terminer, choisissez **Enregistrer les modifications**.

Désactiver un flux CDC dans Amazon Keyspaces

Pour désactiver un flux CDC dans un espace de touches, vous pouvez utiliser l'`ALTER TABLE` instruction dans CQL, la `update-table` commande associée à la console AWS CLI ou la console.

Cassandra Query Language (CQL)

Désactiver un flux (flux CDC) avec CQL

1. Pour désactiver un flux, vous pouvez utiliser l'instruction suivante.

```
ALTER TABLE mykeyspace.mytable
WITH cdc = FALSE;
```

2. Pour confirmer que le flux est désactivé, vous pouvez utiliser l'instruction suivante.

```
SELECT keyspace_name, table_name, cdc, custom_properties FROM
system_schema_mcs.tables WHERE keyspace_name = 'mykeyspace' AND table_name =
'mytable';
```

Le résultat de cette instruction ressemble à celui-ci.

```
keyspace_name | table_name | cdc    | custom_properties
-----+-----+-----+-----
+-----+-----+-----+-----
mykeyspace   | mytable   | False  | {'capacity_mode':
{'last_update_to_pay_per_request_timestamp': '1741385668642',
'throughput_mode': 'PAY_PER_REQUEST'}, 'encryption_specification':
{'encryption_type': 'AWS_OWNED_KMS_KEY'}, 'point_in_time_recovery': {'status':
'disabled'}}
```

CLI

Désactivez un flux (flux CDC) à l'aide du AWS CLI

1. Pour désactiver un flux, vous pouvez utiliser la commande suivante.

```
aws keyspaces update-table \
--keyspace-name 'mykeyspace' \
--table-name 'mytable' \
--cdc-specification status=DISABLED
```

2. La sortie de la commande ressemble à celle de cet exemple.

```
{
```



```
"keyspaceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
my_keyspace/",
"streamName": "my_stream"
}
```

Console

Désactiver un flux (flux CDC) à l'aide de la console Amazon Keyspaces

1. Connectez-vous au et ouvrez Console de gestion AWS la console Amazon Keyspaces à <https://console.aws.amazon.com/keyspaces/home> l'adresse.
2. Dans le volet de navigation, choisissez **Tables**, puis choisissez un tableau dans la liste.
3. Choisissez l'onglet **Streams**.
4. Choisissez **Modifier**.
5. Désélectionnez **Activer les diffusions**.
6. Choisissez **Enregistrer les modifications** pour désactiver le flux.

Afficher les flux CDC dans Amazon Keyspaces

Pour afficher ou répertorier tous les flux dans l'espace clé, vous pouvez interroger la table `system_schema_mcs.streams` dans l'espace clé du système à l'aide d'une instruction dans CQL, ou utiliser les `list-stream` commandes `get-stream` et avec la console ou la AWS CLI console.

Pour les autorisations requises, consultez [the section called “Configurer les autorisations”](#).

Cassandra Query Language (CQL)

Afficher les flux CDC avec CQL

- Pour contrôler l'état CDC de votre table, vous pouvez utiliser l'instruction suivante.

```
SELECT custom_properties
FROM system_schema_mcs.tables
WHERE keyspace_name='my_keyspace' and table_name='my_table';
```

Le résultat de la commande ressemble à ceci.

...

```
custom_properties
-----
{'cdc_specification':{'status': 'Enabled', 'view_type': 'NEW_IMAGE',
  'latest_stream_arn': 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/
my_keyspace/table/my_table/stream/stream_label'}}
...
```

CLI

Consultez les flux du CDC avec le AWS CLI

1. Cet exemple montre comment afficher les informations de flux d'une table.

```
aws keyspaces get-table \
--keyspace-name 'my_keyspace' \
--table-name 'my_table'
```

Le résultat de la commande ressemble à ceci.

```
{
  "keyspaceName": "my_keyspace",
  "tableName": "my_table",
  ... Other fields ...,
  "latestStreamArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
my_keyspace/table/my_table/stream/stream_label",
  "cdcSpecification": {
    "status": "ENABLED",
    "viewType": "NEW_AND_OLD_IMAGES"
  }
}
```

2. Vous pouvez répertorier tous les flux de votre compte dans un délai spécifié Région AWS. La commande suivante en est un exemple.

```
aws keyspacestreams list-streams --region us-east-1
```

Le résultat de la commande peut ressembler à ceci.

```
{
  "Streams": [
```

```
{
  "StreamArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
ks_1/table/t1/stream/2023-05-11T21:21:33.291",
  "StreamLabel": "2023-05-11T21:21:33.291",
  "KeyspaceName": "ks_1"
  "TableName": "t1",
},
{
  "StreamArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
ks_1/table/t2/stream/2023-05-11T21:21:33.291",
  "StreamLabel": "2023-05-11T21:21:33.291",
  "KeyspaceName": "ks_1"Create a keyspace with the name catalog. Note
that streams are not supported in multi-Region
keyspaces.
  "TableName": "t2",
},
{
  "StreamArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
ks_2/table/t1/stream/2023-05-11T21:21:33.291",
  "StreamLabel": "2023-05-11T21:21:33.291",
  "KeyspaceName": "ks_3"
  "TableName": "t1",
}
]
}
```

3. Vous pouvez également répertorier les flux CDC pour un espace clé donné à l'aide des paramètres suivants.

```
aws keyspacestreams list-streams --keyspace-name ks_1 --region us-east-1
```

Le résultat de la commande ressemble à ceci.

```
{
  "Streams": [
    {
      "StreamArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
ks_1/table/t1/stream/2023-05-11T21:21:33.291",
      "StreamLabel": "2023-05-11T21:21:33.291",
      "KeyspaceName": "ks_1"
      "TableName": "t1",
    },
    {
```

```

    "StreamArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
ks_1/table/t2/stream/2023-05-11T21:21:33.291",
    "StreamLabel": "2023-05-11T21:21:33.291",
    "KeyspaceName": "ks_1"
    "TableName": "t2",
  }
]
}

```

4. Vous pouvez également répertorier les flux CDC pour une table donnée à l'aide des paramètres suivants.

```

aws keyspacesstreams list-streams --keyspace-name ks_1 --table-name t2 --
region us-east-1

```

Le résultat de la commande ressemble à ceci.

```

{
  "Streams": [
    {
      "StreamArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
ks_1/table/t2/stream/2023-05-11T21:21:33.291",
      "StreamLabel": "2023-05-11T21:21:33.291",
      "KeyspaceName": "ks_1"
      "TableName": "t2",
    }
  ]
}

```

Console

Afficher les flux CDC dans la console Amazon Keyspaces

1. Connectez-vous au et ouvrez Console de gestion AWS la console Amazon Keyspaces à <https://console.aws.amazon.com/keyspaces/home> l'adresse.
2. Dans le volet de navigation, choisissez Tables, puis choisissez un tableau dans la liste.
3. Cliquez sur l'onglet Streams pour consulter les détails du stream.

Accédez aux enregistrements dans les flux CDC dans Amazon Keyspaces

Pour accéder aux enregistrements d'un flux, vous utilisez l'API <https://docs.aws.amazon.com/keyspaces/latest/StreamsAPIReference/Welcome.html> Amazon Keyspaces Streams. La section suivante contient des exemples sur la façon d'accéder aux enregistrements à l'aide du AWS CLI.

Pour les autorisations requises, consultez [the section called “Configurer les autorisations”](#).

Accédez aux enregistrements d'un flux à l'aide du AWS CLI

1. Vous pouvez utiliser l'API Amazon Keyspaces Streams pour accéder aux enregistrements de modifications du flux. Pour plus d'informations, consultez la section Référence [de l'API](#) Amazon Keyspaces Streams. Pour récupérer les fragments du flux, vous pouvez utiliser l'`get-streamAPI` comme indiqué dans l'exemple suivant.

```
aws keyspacesstreams get-stream \  
--stream-arn 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/  
mytable/stream/STREAM_LABEL'
```

Voici un exemple de la sortie.

```
{  
  "StreamArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/  
table/mytable/stream/2023-05-11T21:21:33.291",  
  "StreamStatus": "ENABLED",  
  "StreamViewType": "NEW_AND_OLD_IMAGES",  
  "CreationRequestDateTime": "<CREATION_TIME>",  
  "KeyspaceName": "mykeyspace",  
  "TableName": "mytable",  
  "StreamLabel": "2023-05-11T21:21:33.291",  
  "Shards": [  
    {  
      "SequenceNumberRange": {  
        "EndingSequenceNumber": "<END_SEQUENCE_NUMBER>",  
        "StartingSequenceNumber": "<START_SEQUENCE_NUMBER>"  
      },  
      "ShardId": "<SHARD_ID>"  
    },  
  ],  
}
```

2. Pour récupérer des enregistrements du flux, vous commencez par obtenir un itérateur qui vous fournit le point de départ pour accéder aux enregistrements. Pour ce faire, vous pouvez utiliser les fragments du flux CDC renvoyé par l'API à l'étape précédente. Pour rassembler l'itérateur, vous pouvez utiliser l'`get-shard-iterator` API. Pour cet exemple, vous utilisez un itérateur de type `TRIM_HORIZON` qui extrait à partir du dernier point découpé (ou début) de la partition.

```
aws keyspacesstreams get-shard-iterator \  
--stream-arn 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/  
mytable/stream/STREAM_LABEL' \  
--shard-id 'SHARD_ID' \  
--shard-iterator-type 'TRIM_HORIZON'
```

La sortie de la commande ressemble à celle de l'exemple suivant.

```
{  
  "ShardIterator": "<SHARD_ITERATOR>"  
}
```

3. Pour récupérer les enregistrements CDC à l'aide de l'`get-records` API, vous pouvez utiliser l'itérateur renvoyé à la dernière étape. La commande suivante en est un exemple.

```
aws keyspacesstreams get-records \  
--shard-iterator 'SHARD_ITERATOR' \  
--limit 100
```

Voici un exemple de la sortie.

```
{  
  "changeRecords": [...],  
  "nextShardIterator": "<NEXT_SHARD_ITERATOR>",  
  "iteratorDescription": {  
    "iteratorPosition": "BEHIND_TIP"  
  }  
}
```

Optimisez la fréquence d'interrogation grâce à la position de l'itérateur

La `GetRecords` réponse inclut un `iteratorDescription` champ qui indique la position actuelle de votre consommateur dans le fragment :

- **AT_TIP**— Aucun autre enregistrement n'est actuellement disponible. Envisagez de réduire la fréquence de vos sondages afin d'optimiser les coûts.
- **BEHIND_TIP**— La pointe du flux avance en continu. Des enregistrements supplémentaires peuvent être disponibles. Continuez à voter à la fréquence habituelle.

BEHIND_TIP Avec une `changeRecords` liste vide indique que le flux progresse mais qu'aucun dossier client n'est disponible à cette position. Continuez à voter normalement.

Utiliser la bibliothèque cliente Kinesis (KCL) pour traiter les flux Amazon Keyspaces

Cette rubrique explique comment utiliser la bibliothèque cliente Kinesis (KCL) pour consommer et traiter les données issues des flux CDC (Change Data Capture) d'Amazon Keyspaces.

Au lieu de travailler directement avec l'API Amazon Keyspaces Streams, l'utilisation de la bibliothèque client Kinesis (KCL) offre de nombreux avantages, par exemple :

- Suivi du lignage des fragments et gestion des itérateurs intégrés.
- Équilibrage automatique de la charge entre les employés.
- Tolérance aux pannes et reprise en cas de défaillance du personnel.
- Points de contrôle pour suivre la progression du traitement.
- Adaptation à l'évolution de la capacité des cours d'eau.
- Informatique distribuée simplifiée pour le traitement des dossiers du CDC.

La section suivante explique pourquoi et comment utiliser la bibliothèque cliente Kinesis (KCL) pour traiter des flux et fournit un exemple de traitement d'un flux Amazon Keyspaces CDC avec la KCL.

Pour plus d'informations sur les tarifs, consultez la rubrique Tarification d'[Amazon Keyspaces \(pour Apache Cassandra\)](#).

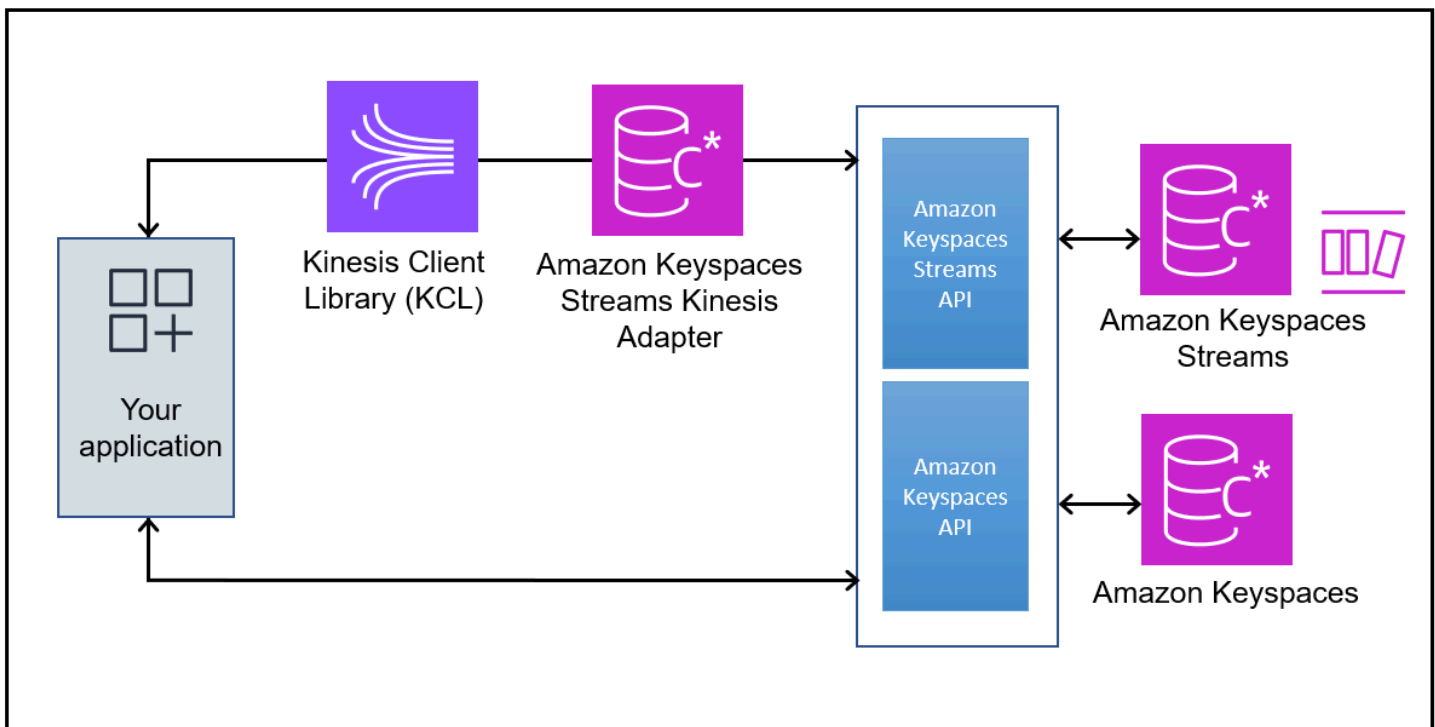
Qu'est-ce que Kinesis Client Library ?

La bibliothèque cliente Kinesis (KCL) est une bibliothèque logicielle Java autonome conçue pour simplifier le processus de consommation et de traitement des données provenant de flux. KCL gère de nombreuses tâches complexes associées à l'informatique distribuée, ce qui vous permet de vous concentrer sur la mise en œuvre de votre logique métier lors du traitement des données de flux. KCL gère des activités telles que l'équilibrage de charge entre plusieurs travailleurs, la réponse aux défaillances des travailleurs, le contrôle des enregistrements traités et la réponse à l'évolution du nombre de partitions dans le flux.

Pour traiter les flux CDC d'Amazon Keyspaces, vous pouvez utiliser les modèles de conception contenus dans la KCL pour travailler avec des fragments de flux et des enregistrements de flux. La KCL simplifie le codage en fournissant des abstractions utiles par-dessus l'API Kinesis Data Streams de bas niveau. Pour plus d'informations sur la KCL, voir [Développer les consommateurs avec KCL](#) dans le guide du développeur Amazon Kinesis Data Streams.

Pour écrire des applications à l'aide de la KCL, vous utilisez l'adaptateur Amazon Keyspaces Streams Kinesis. L'adaptateur Kinesis implémente l'interface Kinesis Data Streams afin que vous puissiez utiliser le KCL pour consommer et traiter des enregistrements provenant des flux Amazon Keyspaces. Pour obtenir des instructions sur la configuration et l'installation de l'adaptateur Kinesis Amazon Keyspaces Streams, consultez le [GitHub](#) référentiel.

Le schéma suivant montre comment ces bibliothèques interagissent les unes avec les autres.



KCL est fréquemment mis à jour pour intégrer de nouvelles versions des bibliothèques sous-jacentes, des améliorations de sécurité et des corrections de bogues. Nous vous recommandons d'utiliser la dernière version de KCL pour éviter les problèmes connus et bénéficier des dernières améliorations. Pour trouver la dernière version de KCL, consultez le référentiel [KCL GitHub](#).

Concepts de KCL

Avant de mettre en œuvre une application grand public à l'aide de KCL, vous devez comprendre les concepts suivants :

Application KCL destinée aux consommateurs

Une application grand public KCL est un programme qui traite les données d'un flux Amazon Keyspaces CDC. Le KCL fait office d'intermédiaire entre le code de votre application client et le flux Amazon Keyspaces CDC.

Travailleur

Un worker est une unité d'exécution de votre application client KCL qui traite les données du flux Amazon Keyspaces CDC. Votre application peut exécuter plusieurs workers répartis sur plusieurs instances.

Processeur d'enregistrement

Un processeur d'enregistrement est la logique de votre application qui traite les données d'une partition dans le flux Amazon Keyspaces CDC. Un processeur d'enregistrement est instancié par un worker pour chaque partition qu'il gère.

Bail

Un contrat de location représente la responsabilité du traitement d'un fragment. Les employés utilisent des contrats de location pour coordonner quel travailleur traite quel fragment. KCL stocke les données de location dans un tableau d'Amazon DynamoDB.

Point de contrôle

Un point de contrôle est un enregistrement de la position dans la partition jusqu'à laquelle le processeur d'enregistrements a traité les enregistrements avec succès. Le point de contrôle permet à votre application de reprendre le traitement là où il s'était arrêté en cas d'échec d'un collaborateur.

Une fois l'adaptateur Amazon Keyspaces Kinesis en place, vous pouvez commencer à développer sur l'interface KCL, les appels d'API étant dirigés de manière fluide vers le point de terminaison du flux Amazon Keyspaces. Pour obtenir la liste des points de terminaison disponibles, reportez-vous à la section [the section called “Accédez aux points de terminaison des flux CDC”](#).

Lorsque votre application démarre, elle appelle la KCL pour instancier un worker. Vous devez fournir au travailleur les informations de configuration de l'application, telles que le descripteur de flux et les AWS informations d'identification, ainsi que le nom de la classe de processeur d'enregistrement que vous fournissez. À mesure qu'il exécute le code dans le processeur d'enregistrements, le worker effectue les tâches suivantes :

- Se connecte au flux
- Énumère les partitions dans le flux.
- Coordonne les associations de partition avec les autres travaux (le cas échéant)
- Instancie un processeur d'enregistrements pour chaque partition qu'il gère
- Extrait des enregistrements du flux
- Pousse les enregistrements sur le processeur d'enregistrements correspondant
- Contrôle les enregistrements traités
- Équilibre les associations partition-worker lorsque le nombre d'instances de worker change
- Équilibre les associations partition-worker quand des partitions sont fractionnées

Implémenter une application KCL destinée aux consommateurs pour les flux CDC d'Amazon Keyspaces

Cette rubrique fournit un guide étape par étape pour implémenter une application client KCL afin de traiter les flux CDC d'Amazon Keyspaces.

1. Prérequis : Avant de commencer, assurez-vous d'avoir :

- Un tableau Amazon Keyspaces avec un flux CDC
- Autorisations IAM requises pour que le principal IAM puisse accéder au flux Amazon Keyspaces CDC, créer et accéder à des tables DynamoDB pour le traitement des flux KCL, et autorisations pour publier des métriques. CloudWatch Pour plus d'informations et un exemple de politique, consultez [the section called “Autorisations pour KCL”](#).
- Assurez-vous que des AWS informations d'identification valides sont configurées dans votre configuration locale. Pour de plus amples informations, veuillez consulter [the section called “Gérer des clés d'accès”](#).
- Kit de développement Java (JDK) 8 ou version ultérieure
- Les exigences sont répertoriées dans le [fichier Readme](#) sur Github.

2. Au cours de cette étape, vous ajoutez la dépendance KCL à votre projet. Pour Maven, ajoutez ce qui suit à votre fichier pom.xml :

```
<dependencies>
  <dependency>
    <groupId>software.amazon.kinesis</groupId>
    <artifactId>amazon-kinesis-client</artifactId>
```

```

        <version>3.1.0</version>
    </dependency>
    <dependency>
        <groupId>software.amazon.keyspaces</groupId>
        <artifactId>keyspaces-streams-kinesis-adapter</artifactId>
        <version>1.0.0</version>
    </dependency>
</dependencies>

```

Note

Vérifiez toujours la dernière version de KCL dans le référentiel [KCL GitHub](#).

3. Créez une classe d'usine qui produit des instances de processeur d'enregistrement :

```

import software.amazon.awssdk.services.keyspacesstreams.model.Record;
import
    software.amazon.keyspaces.streamsadapter.adapter.KeyspacesStreamsClientRecord;
import
    software.amazon.keyspaces.streamsadapter.model.KeyspacesStreamsProcessRecordsInput;
import
    software.amazon.keyspaces.streamsadapter.processor.KeyspacesStreamsShardRecordProcessor;
import software.amazon.kinesis.lifecycle.events.InitializationInput;
import software.amazon.kinesis.lifecycle.events.LeaseLostInput;
import software.amazon.kinesis.lifecycle.events.ShardEndedInput;
import software.amazon.kinesis.lifecycle.events.ShutdownRequestedInput;
import software.amazon.kinesis.processor.RecordProcessorCheckpointer;

public class RecordProcessor implements KeyspacesStreamsShardRecordProcessor {
    private String shardId;

    @Override
    public void initialize(InitializationInput initializationInput) {
        this.shardId = initializationInput.shardId();
        System.out.println("Initializing record processor for shard: " + shardId);
    }

    @Override
    public void processRecords(KeyspacesStreamsProcessRecordsInput
processRecordsInput) {
        try {

```

```

        for (KeyspacesStreamsClientRecord record :
processRecordsInput.records()) {
            Record keyspacesRecord = record.getRecord();
            System.out.println("Received record: " + keyspacesRecord);
        }

        if (!processRecordsInput.records().isEmpty()) {
            RecordProcessorCheckpointier checkpointier =
processRecordsInput.checkpointer();
            try {
                checkpointier.checkpoint();
                System.out.println("Checkpoint successful for shard: " +
shardId);
            } catch (Exception e) {
                System.out.println("Error while checkpointing for shard: " +
shardId + " " + e);
            }
        }
    } catch (Exception e) {
        System.out.println("Error processing records for shard: " + shardId + "
" + e);
    }
}

@Override
public void leaseLost(LeaseLostInput leaseLostInput) {
    System.out.println("Lease lost for shard: " + shardId);
}

@Override
public void shardEnded(ShardEndedInput shardEndedInput) {
    System.out.println("Shard ended: " + shardId);
    try {
        // This is required. Checkpoint at the end of the shard
        shardEndedInput.checkpointer().checkpoint();
        System.out.println("Final checkpoint successful for shard: " +
shardId);
    } catch (Exception e) {
        System.out.println("Error while final checkpointing for shard: " +
shardId + " " + e);
        throw new RuntimeException("Error while final checkpointing", e);
    }
}
}

```

```

@Override
public void shutdownRequested(ShutdownRequestedInput shutdownRequestedInput) {
    System.out.println("Shutdown requested for shard " + shardId);
    try {
        shutdownRequestedInput.checkpointer().checkpoint();
    } catch (Exception e) {
        System.out.println("Error while checkpointing on shutdown for shard: "
+ shardId + " " + e);
    }
}
}

```

4. Créez une fabrique de records comme indiqué dans l'exemple suivant.

```

import software.amazon.kinesis.processor.ShardRecordProcessor;
import software.amazon.kinesis.processor.ShardRecordProcessorFactory;

import java.util.Queue;
import java.util.concurrent.ConcurrentLinkedQueue;

public class RecordProcessorFactory implements ShardRecordProcessorFactory {
    private final Queue<RecordProcessor> processors = new
ConcurrentLinkedQueue<>();

    @Override
    public ShardRecordProcessor shardRecordProcessor() {
        System.out.println("Creating new RecordProcessor");
        RecordProcessor processor = new RecordProcessor();
        processors.add(processor);
        return processor;
    }
}

```

5. Au cours de cette étape, vous allez créer la classe de base pour configurer KCLv3 et l'adaptateur Amazon Keyspaces.

```

import com.example.KCLExample.utils.RecordProcessorFactory;
import
    software.amazon.keyspaces.streamsadapter.AmazonKeyspacesStreamsAdapterClient;
import software.amazon.keyspaces.streamsadapter.StreamsSchedulerFactory;
import java.util.Arrays;
import java.util.List;
import java.util.concurrent.ExecutionException;

```

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchAsyncClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableResponse;
import software.amazon.awssdk.services.keyspacesstreams.KeyspacesStreamsClient;
import software.amazon.awssdk.services.kinesis.KinesisAsyncClient;
import software.amazon.kinesis.common.ConfigsBuilder;
import software.amazon.kinesis.common.InitialPositionInStream;
import software.amazon.kinesis.common.InitialPositionInStreamExtended;
import software.amazon.kinesis.coordinator.CoordinatorConfig;
import software.amazon.kinesis.coordinator.Scheduler;
import software.amazon.kinesis.leases.LeaseManagementConfig;
import software.amazon.kinesis.processor.ProcessorConfig;
import software.amazon.kinesis.processor.StreamTracker;
import software.amazon.kinesis.retrieval.polling.PollingConfig;

public class KCLTestBase {

    protected KeyspacesStreamsClient streamsClient;
    protected KinesisAsyncClient adapterClient;
    protected DynamoDbAsyncClient dynamoDbAsyncClient;
    protected CloudWatchAsyncClient cloudWatchClient;
    protected Region region;
    protected RecordProcessorFactory recordProcessorFactory;
    protected Scheduler scheduler;
    protected Thread schedulerThread;

    public void baseSetUp() {
        recordProcessorFactory = new RecordProcessorFactory();
        setupKCLBase();
    }

    protected void setupKCLBase() {
        region = Region.US_EAST_1;

        streamsClient = KeyspacesStreamsClient.builder()
            .region(region)
            .build();
        adapterClient = new AmazonKeyspacesStreamsAdapterClient(
            streamsClient,
            region);
        dynamoDbAsyncClient = DynamoDbAsyncClient.builder()
```

```

        .region(region)
        .build();
    cloudWatchClient = CloudWatchAsyncClient.builder()
        .region(region)
        .build();
}

protected void startScheduler(Scheduler scheduler) {
    this.scheduler = scheduler;
    schedulerThread = new Thread(() -> scheduler.run());
    schedulerThread.start();
}

protected void shutdownScheduler() {
    if (scheduler != null) {
        scheduler.shutdown();
        try {
            schedulerThread.join(30000);
        } catch (InterruptedException e) {
            System.out.println("Error while shutting down scheduler " + e);
        }
    }
}

protected Scheduler createScheduler(String streamArn, String leaseTableName) {
    String workerId = "worker-" + System.currentTimeMillis();

    // Create ConfigsBuilder
    ConfigsBuilder configsBuilder = createConfigsBuilder(streamArn, workerId,
        leaseTableName);

    // Configure retrieval config for polling
    PollingConfig pollingConfig = new PollingConfig(streamArn, adapterClient);

    // Create the Scheduler
    return StreamsSchedulerFactory.createScheduler(
        configsBuilder.checkpointConfig(),
        configsBuilder.coordinatorConfig(),
        configsBuilder.leaseManagementConfig(),
        configsBuilder.lifecycleConfig(),
        configsBuilder.metricsConfig(),
        configsBuilder.processorConfig(),

        configsBuilder.retrievalConfig().retrievalSpecificConfig(pollingConfig),

```

```
        streamsClient,
        region
    );
}

private ConfigsBuilder createConfigsBuilder(String streamArn, String workerId,
String leaseTableName) {
    ConfigsBuilder configsBuilder = new ConfigsBuilder(
        streamArn,
        leaseTableName,
        adapterClient,
        dynamoDbAsyncClient,
        cloudWatchClient,
        workerId,
        recordProcessorFactory);

    configureCoordinator(configsBuilder.coordinatorConfig());
    configureLeaseManagement(configsBuilder.leaseManagementConfig());
    configureProcessor(configsBuilder.processorConfig());
    configureStreamTracker(configsBuilder, streamArn);

    return configsBuilder;
}

private void configureCoordinator(CoordinatorConfig config) {
    config.skipShardSyncAtWorkerInitializationIfLeasesExist(true)
        .parentShardPollIntervalMillis(1000)
        .shardConsumerDispatchPollIntervalMillis(500);
}

private void configureLeaseManagement(LeaseManagementConfig config) {
    config.shardSyncIntervalMillis(0)
        .leasesRecoveryAuditorInconsistencyConfidenceThreshold(0)
        .leasesRecoveryAuditorExecutionFrequencyMillis(5000)
        .leaseAssignmentIntervalMillis(1000L);
}

private void configureProcessor(ProcessorConfig config) {
    config.callProcessRecordsEvenForEmptyRecordList(true);
}

private void configureStreamTracker(ConfigsBuilder configsBuilder, String
streamArn) {
```



```

        StreamTracker streamTracker =
StreamsSchedulerFactory.createSingleStreamTracker(
            streamArn,

InitialPositionInStreamExtended.newInitialPosition(InitialPositionInStream.TRIM_HORIZON)
        );
        configsBuilder.streamTracker(streamTracker);
    }

    public void deleteAllDdbTables(String baseTableName) {
        List<String> tablesToDelete = Arrays.asList(
            baseTableName,
            baseTableName + "-CoordinatorState",
            baseTableName + "-WorkerMetricStats"
        );

        for (String tableName : tablesToDelete) {
            deleteTable(tableName);
        }
    }

    private void deleteTable(String tableName) {
        DeleteTableRequest deleteTableRequest = DeleteTableRequest.builder()
            .tableName(tableName)
            .build();

        try {
            DeleteTableResponse response =
dynamoDbAsyncClient.deleteTable(deleteTableRequest).get();
            System.out.println("Table deletion response " + response);
        } catch (InterruptedException | ExecutionException e) {
            System.out.println("Error deleting table: " + tableName + " " + e);
        }
    }
}

```

6. Au cours de cette étape, vous implémentez la classe de processeur d'enregistrement pour que votre application commence à traiter les événements de modification.

```

import software.amazon.kinesis.coordinator.Scheduler;

public class KCLTest {

```

```
private static final int APP_RUNTIME_SECONDS = 1800;
private static final int SLEEP_INTERNAL_MS = 60*1000;

public static void main(String[] args) {
    KCLTestBase kclTestBase;

    kclTestBase = new KCLTestBase();
    kclTestBase.baseSetUp();

    // Create and start scheduler
    String leaseTableName = generateUniqueApplicationName();

    // Update below to your Stream ARN
    String streamArn = "arn:aws:cassandra:us-east-1:759151643516:/keyspace/
cdc_sample_test/table/test_kcl_bool/stream/2025-07-01T15:52:57.529";
    Scheduler scheduler = kclTestBase.createScheduler(streamArn,
leaseTableName);
    kclTestBase.startScheduler(scheduler);

    // Wait for specified time before shutting down - KCL applications are
    // designed to run forever, however in this
    // example we will shut it down after APP_RUNTIME_SECONDS
    long startTime = System.currentTimeMillis();
    long endTime = startTime + (APP_RUNTIME_SECONDS * 1000);
    while (System.currentTimeMillis() < endTime) {
        try {
            // Print and sleep every minute
            Thread.sleep(SLEEP_INTERNAL_MS);
            System.out.println("Application is running");
        } catch (InterruptedException e) {
            System.out.println("Interrupted while waiting for records");
            Thread.currentThread().interrupt();
            break;
        }
    }

    // Stop the scheduler
    kclTestBase.shutdownScheduler();
    kclTestBase.deleteAllDdbTables(leaseTableName);
}

public static String generateUniqueApplicationName() {
    String timestamp = String.valueOf(System.currentTimeMillis());
```

```
String randomString = java.util.UUID.randomUUID().toString().substring(0,
8);
return String.format("KCL-App-%s-%s", timestamp, randomString);
}
```

Bonnes pratiques

Suivez ces bonnes pratiques lorsque vous utilisez KCL avec les flux CDC d'Amazon Keyspaces :

Gestion des erreurs

Implémentez une gestion robuste des erreurs dans votre processeur d'enregistrement pour gérer les exceptions avec élégance. Envisagez d'implémenter une logique de nouvelle tentative pour les échecs transitoires.

Fréquence des points de contrôle

Équilibrez la fréquence des points de contrôle afin de minimiser les doublons tout en garantissant un suivi raisonnable des progrès. Des points de contrôle trop fréquents peuvent avoir un impact sur les performances, tandis que des points de contrôle trop peu fréquents peuvent entraîner une augmentation du retraitement en cas d'échec d'un travailleur.

Dimensionnement des effectifs

Augmentez le nombre de travailleurs en fonction du nombre de fragments dans votre flux CDC. Un bon point de départ est d'avoir un travailleur par fragment, mais vous devrez peut-être procéder à des ajustements en fonction de vos besoins de traitement.

Surveillance

Utilisez CloudWatch les statistiques fournies par KCL pour surveiller l'état et les performances de votre application grand public. Les indicateurs clés incluent la latence de traitement, l'âge du point de contrôle et le nombre de baux.

Test

Testez minutieusement votre application grand public, y compris dans des scénarios tels que des pannes de travail, le repartage des flux et des conditions de charge variables.

Utilisation de KCL avec des langages autres que Java

Bien que KCL soit principalement une bibliothèque Java, vous pouvez l'utiliser avec d'autres langages de programmation via le MultiLangDaemon. MultiLangDaemon Il s'agit d'un Java-based démon qui gère l'interaction entre votre processeur d'enregistrement non Java et le KCL.

KCL prend en charge les langues suivantes :

- Python
- Ruby
- Node.js
- .NET

Pour plus d'informations sur l'utilisation de KCL avec des langages autres que Java, consultez la documentation [MultiLangDaemon KCL](#).

Résolution des problèmes

Cette section propose des solutions aux problèmes courants que vous pouvez rencontrer lors de l'utilisation de KCL avec les flux CDC d'Amazon Keyspaces.

Traitement lent

Si votre demande destinée aux consommateurs traite les dossiers lentement, considérez les points suivants :

- Augmenter le nombre d'instances de travail
- Optimisation de la logique de traitement de vos enregistrements
- Vérification de la présence de goulots d'étranglement dans les systèmes en aval

Traitement des doublons

Si vous constatez que des enregistrements sont traités en double, vérifiez votre logique de contrôle. Assurez-vous d'effectuer un point de contrôle après avoir traité avec succès les enregistrements.

Les échecs des travailleurs

Si les employés échouent fréquemment, vérifiez :

- Contraintes de ressources (processeur, mémoire)
- Problèmes liés à la connectivité réseau

- Problèmes d'autorisations

Problèmes liés aux tables de location

Si vous rencontrez des problèmes avec le tableau des baux KCL :

- Vérifiez que votre application dispose des autorisations appropriées pour accéder au tableau Amazon Keyspaces
- Vérifiez que la table dispose d'un débit provisionné suffisant

Utilisation des partitionneurs dans Amazon Keyspaces

Dans Apache Cassandra, les partitionneurs contrôlent les nœuds sur lesquels les données sont stockées dans le cluster. Les partitionneurs créent un jeton numérique à l'aide d'une valeur hachée de la clé de partition. Cassandra utilise ce jeton pour distribuer les données entre les nœuds. Les clients peuvent également utiliser ces jetons dans les `SELECT` opérations et les `WHERE` clauses afin d'optimiser les opérations de lecture et d'écriture. Par exemple, les clients peuvent exécuter efficacement des requêtes parallèles sur de grandes tables en spécifiant des plages de jetons distinctes à interroger dans chaque tâche parallèle.

Amazon Keyspaces propose trois partitionneurs différents.

Murmur3 Partitioner (par défaut)

Apache Cassandra-compatible `Murmur3Partitioner`. `Murmur3Partitioner` Il s'agit du partitionneur Cassandra par défaut dans Amazon Keyspaces et dans Cassandra 1.2 et versions ultérieures.

RandomPartitioner

Apache Cassandra-compatible `RandomPartitioner`. `RandomPartitioner` Il s'agit du partitionneur Cassandra par défaut pour les versions antérieures à Cassandra 1.2.

Partitionneur par défaut de Keyspaces

`DefaultPartitioner` Renvoie les mêmes résultats token de fonction que `leRandomPartitioner`.

Le paramètre du partitionneur est appliqué par région au niveau du compte. Par exemple, si vous modifiez le partitionneur dans l'est des États-Unis (Virginie du Nord), la modification est appliquée à toutes les tables du même compte dans cette région. Vous pouvez modifier votre partitionneur en

toute sécurité à tout moment. Notez que la modification de configuration prend environ 10 minutes. Vous n'avez pas besoin de recharger vos données Amazon Keyspaces lorsque vous modifiez le paramètre du partitionneur. Les clients utiliseront automatiquement le nouveau paramètre du partitionneur lors de leur prochaine connexion.

Comment modifier le partitionneur dans Amazon Keyspaces

Vous pouvez modifier le partitionneur à l'aide du langage CQL (Console de gestion AWS ou Cassandra Query Language).

Console de gestion AWS

Pour modifier le partitionneur à l'aide de la console Amazon Keyspaces

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le panneau de navigation, choisissez Configuration.
3. Sur la page de configuration, accédez à Modifier le partitionneur.
4. Sélectionnez le partitionneur compatible avec votre version de Cassandra. Le changement de partitionneur prend environ 10 minutes pour s'appliquer.

Note

Une fois la modification de configuration terminée, vous devez vous déconnecter et vous reconnecter à Amazon Keyspaces pour les demandes d'utilisation du nouveau partitionneur.

Cassandra Query Language (CQL)

1. Pour voir quel partitionneur est configuré pour le compte, vous pouvez utiliser la requête suivante.

```
SELECT partitioner from system.local;
```

Si le partitionneur n'a pas été modifié, le résultat de la requête est le suivant.

```
partitioner
```

```
-----  
com.amazonaws.cassandra.DefaultPartitioner
```

2. Pour mettre à jour le partitionneur vers le Murmur3 partitionneur, vous pouvez utiliser l'instruction suivante.

```
UPDATE system.local set  
partitioner='org.apache.cassandra.dht.Murmur3Partitioner' where key='local';
```

3. Notez que cette modification de configuration prend environ 10 minutes. Pour confirmer que le partitionneur a été configuré, vous pouvez exécuter à nouveau la SELECT requête. Notez qu'en raison d'une éventuelle cohérence de lecture, la réponse peut ne pas encore refléter les résultats du changement de partitionneur récemment effectué. Si vous répétez l'opération peu de temps après, la réponse doit renvoyer les données les plus récentes.

```
SELECT partitioner from system.local;
```

Note

Vous devez vous déconnecter puis vous reconnecter à Amazon Keyspaces pour que les demandes utilisent le nouveau partitionneur.

Client-side horodatages dans Amazon Keyspaces

Dans Amazon Keyspaces, les horodatages côté client sont des horodatages conservés pour chaque Cassandra-compatible cellule de votre tableau. Vous pouvez utiliser les horodatages côté client pour résoudre les conflits en laissant vos applications clientes déterminer l'ordre des écritures. Par exemple, lorsque les clients d'une application distribuée dans le monde entier mettent à jour les mêmes données, les horodatages côté client conservent l'ordre dans lequel les mises à jour ont été effectuées sur les clients. Amazon Keyspaces utilise ces horodatages pour traiter les écritures.

Les horodatages côté client d'Amazon Keyspaces sont entièrement gérés. Vous n'avez pas à gérer les paramètres système de bas niveau tels que les stratégies de nettoyage et de compactage.

Lorsque vous supprimez des données, les lignes sont marquées d'une pierre tombale pour suppression. Amazon Keyspaces supprime automatiquement les données fondamentales (généralement dans les 10 jours) sans affecter les performances ou la disponibilité de votre

application. Les données Tombstoned ne sont pas disponibles pour les instructions DML (Data Manipulation Language). Alors que vous continuez à effectuer des lectures et des écritures sur des lignes contenant des données gravées, celles-ci continuent d'être prises en compte dans le stockage, les unités de capacité de lecture (RCU) et les unités de capacité d'écriture (WCU) jusqu'à ce qu'elles soient supprimées du stockage.

Une fois que les horodatages côté client ont été activés pour une table, vous pouvez spécifier un horodatage avec la `USING TIMESTAMP` clause dans votre requête CQL DML (Data Manipulation Language). Pour de plus amples informations, veuillez consulter [the section called “Utiliser les horodatages côté client dans les requêtes”](#). Si vous ne spécifiez pas d'horodatage dans votre requête CQL, Amazon Keyspaces utilise l'horodatage transmis par le pilote de votre client. Si le pilote client ne fournit pas d'horodatage, Amazon Keyspaces attribue automatiquement un horodatage au niveau des cellules, car les horodatages ne peuvent pas l'être. NULL Pour demander des horodatages, vous pouvez utiliser la `WRITETIME` fonction de votre instruction DML.

Amazon Keyspaces ne facture aucun supplément pour activer l'horodatage côté client. Toutefois, avec les horodatages côté client, vous stockez et écrivez des données supplémentaires pour chaque valeur de votre ligne. Cela peut entraîner une utilisation supplémentaire du stockage et, dans certains cas, une utilisation du débit supplémentaire. Pour plus d'informations sur les tarifs d'Amazon Keyspaces, consultez les tarifs d'Amazon [Keyspaces \(pour Apache Cassandra\)](#).

Lorsque les horodatages côté client sont activés dans Amazon Keyspaces, des métadonnées supplémentaires sont stockées à côté de vos données de ligne. La surcharge par ligne dépend du type de colonne, du fait que la ligne utilise le protocole TTL et (pour les tables multirégionales) du nombre de régions répliquées. La surcharge peut aller de quelques octets pour les lignes de colonnes scalaires simples à des dizaines d'octets ou plus pour les lignes contenant des collections non figées ou des compteurs multirégionaux. Ces métadonnées sont prises en compte à la fois pour votre coût de stockage et pour votre quota de taille de ligne de 1 Mo.

Pour déterminer l'impact global sur le stockage et le débit, prenez en compte le nombre de colonnes de votre tableau, les types de données utilisés et le nombre d'éléments de collection dans chaque ligne. Par exemple, les lignes contenant de nombreuses colonnes de collection non figées contenant un grand nombre d'éléments auront une surcharge plus élevée que les lignes contenant uniquement des colonnes scalaires. Les métadonnées supplémentaires affectent également le nombre d'unités de capacité d'écriture (WCU) consommées par écriture. Pour plus d'informations sur la manière d'estimer les frais généraux pour votre schéma spécifique, consultez [the section called “Estimer la taille des lignes”](#). Pour plus d'informations sur le calcul de la capacité de lecture et d'écriture, consultez [the section called “Configuration des modes read/write de capacité”](#).

Une fois que les horodatages côté client ont été activés pour un tableau, vous ne pouvez pas le désactiver.

Pour en savoir plus sur l'utilisation des horodatages côté client dans les requêtes, consultez [the section called “Utiliser les horodatages côté client dans les requêtes”](#)

Rubriques

- [Comment les horodatages côté client d'Amazon Keyspaces s'intègrent à AWS services](#)
- [Créez un nouveau tableau avec des horodatages côté client dans Amazon Keyspaces](#)
- [Configurer les horodatages côté client pour une table dans Amazon Keyspaces](#)
- [Utiliser les horodatages côté client dans les requêtes dans Amazon Keyspaces](#)

Comment les horodatages côté client d'Amazon Keyspaces s'intègrent à AWS services

La métrique d'horodatage côté client suivante est disponible sur Amazon CloudWatch pour permettre une surveillance continue.

- `SystemReconciliationDeletes`— Le nombre d'opérations de suppression nécessaires pour supprimer les données tombstoned.

Pour plus d'informations sur le suivi CloudWatch des métriques, consultez [the section called “Surveillance avec CloudWatch”](#).

Lorsque vous l'utilisez CloudFormation, vous pouvez activer les horodatages côté client lors de la création d'une table Amazon Keyspaces. Pour plus d'informations, consultez le [Guide de l'utilisateur AWS CloudFormation](#).

Créez un nouveau tableau avec des horodatages côté client dans Amazon Keyspaces

Suivez ces exemples pour créer une nouvelle table Amazon Keyspaces avec les horodatages côté client activés à l'aide d'Amazon Console de gestion AWS Keyspaces, du Cassandra Query Language (CQL) ou du AWS Command Line Interface

Console

Création d'un nouveau tableau avec des horodatages côté client (console)

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le panneau de navigation, choisissez Tables, puis Create table (Créer une table).
3. Sur la page Créer une table dans la section Détails de la table, sélectionnez un espace de touche et donnez un nom à la nouvelle table.
4. Dans la section Schéma, créez le schéma de votre table.
5. Dans la section Paramètres du tableau, choisissez Personnaliser les paramètres.
6. Passez aux Client-side horodatages.

Choisissez Activer les horodatages côté client pour activer les horodatages côté client pour le tableau.

7. Choisissez Créer un tableau. Votre tableau est créé avec les horodatages côté client activés.

Cassandra Query Language (CQL)

Création d'une nouvelle table à l'aide de CQL

1. Pour créer une nouvelle table avec des horodatages côté client activés à l'aide de CQL, vous pouvez utiliser l'exemple suivant.

```
CREATE TABLE my_keyspace.my_table (  
    userid uuid,  
    time timeuuid,  
    subject text,  
    body text,  
    user inet,  
    PRIMARY KEY (userid, time)  
) WITH CUSTOM_PROPERTIES = {'client_side_timestamps': {'status': 'enabled'}};
```

2. Pour confirmer les paramètres d'horodatage côté client pour la nouvelle table, utilisez une SELECT instruction pour vérifier les paramètres, `custom_properties` comme indiqué dans l'exemple suivant.

```
SELECT custom_properties from system_schema_mcs.tables where keyspace_name =  
'my_keyspace' and table_name = 'my_table';
```

Le résultat de cette instruction indique l'état des horodatages côté client.

```
'client_side_timestamps': {'status': 'enabled'}
```

AWS CLI

Créez une nouvelle table à l'aide du AWS CLI

1. Pour créer une nouvelle table avec les horodatages côté client activés, vous pouvez utiliser l'exemple suivant.

```
./aws keyspaces create-table \  
--keyspace-name my_keyspace \  
--table-name my_table \  
--client-side-timestamps 'status=ENABLED' \  
--schema-definition 'allColumns=[{name=id,type=int},{name=date,type=timestamp},  
{name=name,type=text}],partitionKeys=[{name=id}]'
```

2. Pour vérifier que les horodatages côté client sont activés pour la nouvelle table, exécutez le code suivant.

```
./aws keyspaces get-table \  
--keyspace-name my_keyspace \  
--table-name my_table
```

La sortie doit ressembler à cet exemple.

```
{  
  "keyspaceName": "my_keyspace",  
  "tableName": "my_table",  
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/  
my_keyspace/table/my_table",  
  "creationTimestamp": 1662681206.032,  
  "status": "ACTIVE",  
  "schemaDefinition": {  
    "allColumns": [  

```

```
    {
      "name": "id",
      "type": "int"
    },
    {
      "name": "date",
      "type": "timestamp"
    },
    {
      "name": "name",
      "type": "text"
    }
  ],
  "partitionKeys": [
    {
      "name": "id"
    }
  ],
  "clusteringKeys": [],
  "staticColumns": []
},
"capacitySpecification": {
  "throughputMode": "PAY_PER_REQUEST",
  "lastUpdateToPayPerRequestTimestamp": 1662681206.032
},
"encryptionSpecification": {
  "type": "AWS_OWNED_KMS_KEY"
},
"pointInTimeRecovery": {
  "status": "DISABLED"
},
"clientSideTimestamps": {
  "status": "ENABLED"
},
"ttl": {
  "status": "ENABLED"
},
"defaultTimeToLive": 0,
"comment": {
  "message": ""
}
}
```

Configurer les horodatages côté client pour une table dans Amazon Keyspaces

Suivez ces exemples pour activer les horodatages côté client pour les tables existantes à l'aide d'Amazon Keyspaces Console de gestion AWS, du Cassandra Query Language (CQL) ou du AWS Command Line Interface

Console

Pour activer les horodatages côté client pour une table existante (console)

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Choisissez le tableau que vous souhaitez mettre à jour, puis choisissez l'onglet Paramètres supplémentaires.
3. Dans l'onglet Paramètres supplémentaires, accédez à Modifier les horodatages côté client et sélectionnez Activer les horodatages côté client
4. Choisissez Enregistrer les modifications pour modifier les paramètres du tableau.

Cassandra Query Language (CQL)

Utilisation d'une instruction CQL

1. Activez les horodatages côté client pour une table existante avec l'instruction CQL. ALTER TABLE

```
ALTER TABLE my_table WITH custom_properties = {'client_side_timestamps':  
{'status': 'enabled'}};;
```

2. Pour confirmer les paramètres d'horodatage côté client pour la nouvelle table, utilisez une SELECT instruction pour vérifier les paramètres, custom_properties comme indiqué dans l'exemple suivant.

```
SELECT custom_properties from system_schema_mcs.tables where keyspace_name =  
'my_keyspace' and table_name = 'my_table';
```

Le résultat de cette instruction indique l'état des horodatages côté client.

```
'client_side_timestamps': {'status': 'enabled'}
```

AWS CLI

Utilisation de AWS CLI

1. Vous pouvez activer les horodatages côté client pour une table existante à l'aide de l' AWS CLI exemple suivant.

```
./aws keyspaces update-table \  
--keyspace-name my_keyspace \  
--table-name my_table \  
--client-side-timestamps 'status=ENABLED'
```

2. Pour vérifier que les horodatages côté client sont activés pour la table, exécutez le code suivant.

```
./aws keyspaces get-table \  
--keyspace-name my_keyspace \  
--table-name my_table
```

La sortie doit ressembler à cet exemple et indiquer l'état des horodatages côté client sous la forme. ENABLED

```
{  
  "keyspaceName": "my_keyspace",  
  "tableName": "my_table",  
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/  
my_keyspace/table/my_table",  
  "creationTimestamp": 1662681312.906,  
  "status": "ACTIVE",  
  "schemaDefinition": {  
    "allColumns": [  
      {  
        "name": "id",  
        "type": "int"  
      },  
      {  
        "name": "date",
```

```
        "type": "timestamp"
      },
      {
        "name": "name",
        "type": "text"
      }
    ],
    "partitionKeys": [
      {
        "name": "id"
      }
    ],
    "clusteringKeys": [],
    "staticColumns": []
  },
  "capacitySpecification": {
    "throughputMode": "PAY_PER_REQUEST",
    "lastUpdateToPayPerRequestTimestamp": 1662681312.906
  },
  "encryptionSpecification": {
    "type": "AWS_OWNED_KMS_KEY"
  },
  "pointInTimeRecovery": {
    "status": "DISABLED"
  },
  "clientSideTimestamps": {
    "status": "ENABLED"
  },
  "ttl": {
    "status": "ENABLED"
  },
  "defaultTimeToLive": 0,
  "comment": {
    "message": ""
  }
}
```

Utiliser les horodatages côté client dans les requêtes dans Amazon Keyspaces

Après avoir activé les horodatages côté client, vous pouvez transmettre l'horodatage dans vos instructions `INSERT`, `UPDATE`, et avec la clause `DELETE USING TIMESTAMP`

La valeur d'horodatage `bigint` représente un nombre de microsecondes depuis l'heure de base standard connue sous le nom de epoch : 1er janvier 1970 à 00:00:00 GMT. Un horodatage fourni par le client doit se situer entre 2 jours dans le passé et 5 minutes dans le futur par rapport à l'heure actuelle de l'horloge murale.

Amazon Keyspaces conserve les métadonnées d'horodatage pendant toute la durée de vie des données. Vous pouvez utiliser cette `WRITETIME` fonction pour rechercher les horodatages survenus il y a des années. Pour plus d'informations sur la syntaxe CQL, consultez [the section called “Instructions DML”](#).

L'instruction CQL suivante est un exemple d'utilisation d'un horodatage en tant que `update_parameter`

```
INSERT INTO catalog.book_awards (year, award, rank, category, book_title, author, publisher)
VALUES (2022, 'Wolf', 4, 'Non-Fiction', 'Science Update', 'Ana Carolina Silva', 'SomePublisher')
USING TIMESTAMP 1669069624;
```

Si vous ne spécifiez pas d'horodatage dans votre requête CQL, Amazon Keyspaces utilise l'horodatage transmis par le pilote de votre client. Si aucun horodatage n'est fourni par le pilote client, Amazon Keyspaces attribue un horodatage côté serveur pour votre opération d'écriture.

Pour voir la valeur d'horodatage stockée pour une colonne spécifique, vous pouvez utiliser la `WRITETIME` fonction dans une `SELECT` instruction, comme illustré dans l'exemple suivant.

```
SELECT year, award, rank, category, book_title, author, publisher, WRITETIME(year),
WRITETIME(award), WRITETIME(rank),
WRITETIME(category), WRITETIME(book_title), WRITETIME(author), WRITETIME(publisher)
from catalog.book_awards;
```


Multi-Region réplication pour Amazon Keyspaces (pour Apache Cassandra)

Vous pouvez utiliser la réplication multirégionale d'Amazon Keyspaces pour répliquer vos données grâce à une réplication active-active, automatisée et entièrement gérée sur le site de votre choix. Régions AWS Avec la réplication active-active, chaque région est capable d'effectuer des lectures et des écritures de manière isolée. Vous pouvez améliorer à la fois la disponibilité et la résilience face à la dégradation régionale, tout en bénéficiant de lectures et d'écritures locales à faible latence pour les applications globales.

Avec la réplication multirégionale, Amazon Keyspaces réplique les données de manière asynchrone entre les régions, et les données sont généralement propagées d'une région à l'autre en une seconde. De plus, grâce à la réplication multirégionale, vous n'avez plus à résoudre les conflits et les problèmes de divergence des données, ce qui vous permet de vous concentrer sur votre application.

Par défaut, Amazon Keyspaces réplique les données dans trois [zones de disponibilité](#) au sein d'une même zone Région AWS pour garantir la durabilité et la haute disponibilité. Avec la réplication multirégionale, vous pouvez créer des espaces clés multirégionaux qui répliquent vos tables dans différentes zones géographiques Régions AWS de votre choix.

Rubriques

- [Avantages de l'utilisation de la réplication multirégionale](#)
- [Modes de capacité et tarification](#)
- [Comment fonctionne la réplication multirégionale dans Amazon Keyspaces](#)
- [Notes d'utilisation de la réplication multirégionale Amazon Keyspaces](#)
- [Configurer la réplication multirégionale pour Amazon Keyspaces \(pour Apache Cassandra\)](#)

Avantages de l'utilisation de la réplication multirégionale

Multi-Region la réplication offre les avantages suivants.

- Lectures et écritures globales avec une latence d'un chiffre en millisecondes — Dans Amazon Keyspaces, la réplication est active-active. Vous pouvez effectuer des opérations de lecture et d'écriture localement depuis les régions les plus proches de vos clients avec une latence d'un chiffre en millisecondes, quelle que soit l'échelle. Vous pouvez utiliser les tables multirégionales

d'Amazon Keyspaces pour les applications internationales qui ont besoin d'un temps de réponse rapide partout dans le monde.

- Continuité d'activité améliorée et protection contre la dégradation dans une seule région — Grâce à la réplication multirégionale, vous pouvez remédier à la dégradation en une seule fois Région AWS en redirigeant votre application vers une autre région de votre espace clé multirégional. Amazon Keyspaces proposant une réplication active-active, cela n'a aucun impact sur vos lectures et vos écritures.

Amazon Keyspaces garde une trace de toutes les écritures effectuées sur votre espace de touches multirégional mais qui n'ont pas été propagées à toutes les régions répliques. Une fois que la région est de nouveau en ligne, Amazon Keyspaces synchronise automatiquement toutes les modifications manquantes afin que vous puissiez les récupérer sans aucun impact sur l'application.

- High-speed réplication entre régions : Multi-Region la réplication utilise une réplication physique rapide basée sur le stockage des données entre les régions, avec un délai de réplication généralement inférieur à une seconde.

La réplication dans Amazon Keyspaces n'a que peu ou pas d'impact sur vos requêtes de base de données car elle ne partage pas les ressources de calcul avec votre application. Cela signifie que vous pouvez traiter les cas d'utilisation à haut débit d'écriture ou les cas d'utilisation présentant des pics ou des pics de débit soudains sans aucun impact sur les applications.

- Cohérence et résolution des conflits — Toute modification apportée aux données d'une région est reproduite dans les autres régions dans un espace clé multirégional. Si les applications mettent à jour les mêmes données dans différentes régions en même temps, des conflits peuvent survenir.

Pour garantir une certaine cohérence, Amazon Keyspaces utilise des horodatages au niveau des cellules et le dernier rédacteur obtient le rapprochement entre les mises à jour simultanées. La résolution des conflits est entièrement gérée et s'effectue en arrière-plan sans aucun impact sur les applications.

Pour plus d'informations sur les configurations et fonctionnalités prises en charge, consultez [the section called “Notes d'utilisation”](#).

Modes de capacité et tarification

Pour un espace clé multirégional, vous pouvez utiliser le mode capacité à la demande ou le mode capacité provisionnée. Pour de plus amples informations, veuillez consulter [the section called “Configuration des modes read/write de capacité”](#).

Pour le mode à la demande, une unité de demande d'écriture (WRU) vous est facturée pour écrire jusqu'à 1 Ko de données par ligne, de la même manière que pour les tables à région unique. Mais les écritures vous sont facturées dans chaque région de votre espace de saisie multirégional. Par exemple, l'écriture d'une ligne de 3 Ko de données dans un espace clé multirégional avec deux régions nécessite 6 WRU : $3 * 2 = 6$ WRU. En outre, les écritures qui incluent à la fois des données statiques et non statiques nécessitent des opérations d'écriture supplémentaires.

Pour le mode provisionné, une unité de capacité d'écriture (WCU) vous est facturée pour écrire jusqu'à 1 Ko de données par ligne, de la même manière que pour les tables à région unique. Mais les écritures vous sont facturées dans chaque région de votre espace de saisie multirégional. Par exemple, l'écriture d'une ligne de 3 Ko de données par seconde dans un espace clé multirégional avec deux régions nécessite 6 WCU : $3 * 2 = 6$ WCU. En outre, les écritures qui incluent à la fois des données statiques et non statiques nécessitent des opérations d'écriture supplémentaires.

Pour plus d'informations sur les tarifs, consultez les tarifs [d'Amazon Keyspaces \(pour Apache Cassandra\)](#).

Comment fonctionne la réplication multirégionale dans Amazon Keyspaces

Cette section fournit une vue d'ensemble du fonctionnement de la réplication multirégionale Amazon Keyspaces. Pour plus d'informations sur la tarification, consultez la rubrique Tarification d'[Amazon Keyspaces \(pour Apache Cassandra\)](#).

Rubriques

- [Comment fonctionne la réplication multirégionale dans Amazon Keyspaces](#)
- [Multi-Region résolution des conflits de réplication](#)
- [Multi-Region réplication et reprise après sinistre](#)
- [Multi-Region réplication dans Régions AWS désactivé par défaut](#)
- [Multi-Region réplication et intégration avec restauration instantanée \(PITR\)](#)
- [Multi-Region réplication et intégration avec AWS services](#)

Comment fonctionne la réplication multirégionale dans Amazon Keyspaces

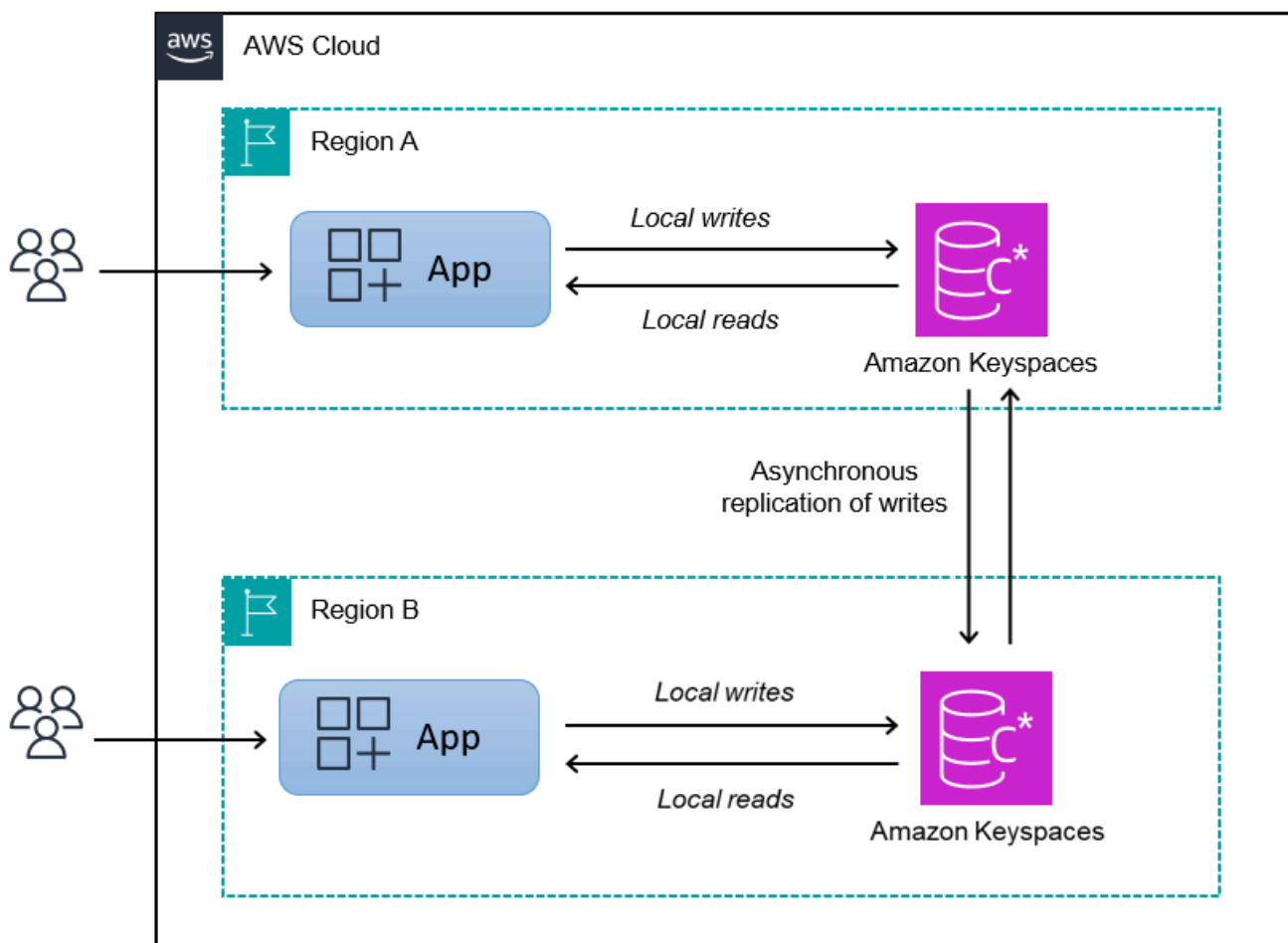
La réplication multirégionale Amazon Keyspaces met en œuvre une architecture de résilience des données qui distribue vos données de manière indépendante et géographiquement distribuée.

Régions AWS Il utilise la réplication active-active, qui fournit une faible latence locale, chaque région pouvant effectuer des lectures et des écritures de manière isolée.

Lorsque vous créez un espace de clés multi-régions Amazon Keyspaces, vous pouvez sélectionner des régions supplémentaires vers lesquelles les données seront répliquées. Chaque table que vous créez dans un espace clé multirégional se compose de plusieurs tables répliques (une par région) qu'Amazon Keyspaces considère comme une seule unité.

Les réplicas a les mêmes nom de table et schéma de clé primaire. Lorsqu'une application écrit des données dans une table locale d'une région, les données sont écrites de manière durable en utilisant le niveau de LOCAL_QUORUM cohérence. Amazon Keyspaces réplique automatiquement les données de manière asynchrone vers les autres régions de réplication. Le délai de réplication entre les régions est généralement inférieur à une seconde et n'a aucune incidence sur les performances ou le débit de votre application.

Une fois les données écrites, vous pouvez les lire dans la table multi-régions d'une autre région de réplication avec les niveaux de LOCAL_ONE/LOCAL_QUORUM cohérence. Pour plus d'informations sur les configurations et fonctionnalités prises en charge, consultez [the section called “Notes d'utilisation”](#).



Multi-Region résolution des conflits de réplication

La réplication multirégionale Amazon Keyspaces est entièrement gérée, ce qui signifie que vous n'avez pas à effectuer de tâches de réplication telles que des opérations de réparation régulières pour résoudre les problèmes de synchronisation des données. Amazon Keyspaces surveille la cohérence des données entre les différentes Régions AWS tables en détectant et en réparant les conflits, et synchronise automatiquement les répliques.

Amazon Keyspaces utilise la méthode `last writer wins` pour le rapprochement des données. Grâce à ce mécanisme de résolution des conflits, toutes les régions d'un espace clé multirégional s'accordent sur la dernière mise à jour et convergent vers un état dans lequel elles disposent toutes de données identiques. Le processus de rapprochement n'a aucune incidence sur les performances de l'application. Pour faciliter la résolution des conflits, les horodatages côté client sont automatiquement activés pour les tables multirégions et ne peuvent pas être désactivés. Pour de plus amples informations, veuillez consulter [the section called “Client-side horodatages”](#).

Multi-Region réplication et reprise après sinistre

Avec la réplication multirégionale Amazon Keyspaces, les écritures sont répliquées de manière asynchrone dans chaque région. Dans les rares cas de dégradation ou de défaillance d'une seule région, la réplication multirégion vous aide à effectuer une reprise après sinistre avec peu ou pas d'impact sur votre application. La reprise après sinistre est généralement mesurée à l'aide des valeurs de l'objectif de temps de restauration (RTO) et de l'objectif de point de restauration (RPO).

Objectif de temps de restauration : temps nécessaire à un système pour revenir à un état de fonctionnement après un sinistre. Le RTO mesure le temps d'arrêt que votre charge de travail peut tolérer, mesuré dans le temps. Pour les plans de reprise après sinistre qui utilisent la réplication multirégionale pour basculer vers une région non affectée, le RTO peut être proche de zéro. Le RTO est limité par la rapidité avec laquelle votre application peut détecter l'état de défaillance et rediriger le trafic vers une autre région.

Objectif du point de restauration : quantité de données pouvant être perdue (mesurée dans le temps). Pour les plans de reprise après sinistre qui utilisent la réplication multirégion pour basculer vers une région non affectée, le RPO est généralement d'un chiffre en secondes. Le RPO est limité par la latence de réplication vers la réplique cible du basculement.

En cas de panne ou de dégradation régionale, vous n'avez pas besoin de promouvoir une région secondaire ni d'effectuer des procédures de basculement de base de données, car la réplication dans Amazon Keyspaces est active-active. Vous pouvez plutôt utiliser Amazon Route 53 pour acheminer

votre application vers la région saine la plus proche. Pour en savoir plus sur la Route 53, consultez [Qu'est-ce qu'Amazon Route 53 ?](#).

Si l'une d'entre Région AWS elles est isolée ou dégradée, votre application peut rediriger le trafic vers une autre région à l'aide de Route 53 pour effectuer des lectures et des écritures sur une autre table de répliques. Vous pouvez également appliquer une logique métier personnalisée pour déterminer quand rediriger les demandes vers d'autres régions. Par exemple, vous pouvez informer votre application des multiples points de terminaison disponibles.

Lorsque la région est de nouveau en ligne, Amazon Keyspaces reprend la propagation des écritures en attente depuis cette région vers les tables de répliques des autres régions. Il reprend également la propagation des écritures des autres tables de réplique vers la région revenue en ligne.

Multi-Region réplication dans Régions AWS désactivé par défaut

La réplication multirégionale Amazon Keyspaces est prise en charge dans les domaines suivants Régions AWS qui sont désactivés par défaut :

- Région Afrique (Le Cap)
- Région du Moyen-Orient (EAU)
- Région Asie-Pacifique (Hong Kong)
- Middle East (Bahrain) Region

Avant de pouvoir utiliser une région désactivée par défaut avec la réplication multi-régions Amazon Keyspaces, vous devez d'abord activer la région. Pour plus d'informations, consultez la section [Activer ou désactiver Régions AWS dans votre compte](#) dans le Guide de [AWS Organizations l'utilisateur](#).

Après avoir activé une région, vous pouvez créer de nouvelles ressources Amazon Keyspaces dans la région et ajouter la région à un espace clé multirégional.

Lorsque vous désactivez une région utilisée par la réplication multirégionale Amazon Keyspaces, Amazon Keyspaces déclenche une période de grâce de 24 heures. Au cours de cette période, vous pouvez vous attendre au comportement suivant :

- Amazon Keyspaces continue d'effectuer des opérations en langage de manipulation des données (DML) dans les régions activées.
- Amazon Keyspaces interrompt la réplication des mises à jour de données depuis les régions activées vers les régions désactivées.

- Amazon Keyspaces bloque toutes les demandes en langage de définition des données (DDL) dans la région désactivée.

Si vous avez désactivé la région par erreur, vous pouvez la réactiver dans les 24 heures. Si vous réactivez la région pendant la période de grâce de 24 heures, Amazon Keyspaces prendra les mesures suivantes :

- Reprenez automatiquement toutes les répliquions vers la région réactivée.
- Répliquez toutes les mises à jour de données qui ont eu lieu dans les régions activées alors que la région était désactivée pour garantir la cohérence des données.
- Poursuivez automatiquement toutes les opérations de répliquion multirégions supplémentaires.

Si la région reste désactivée après la fermeture de la fenêtre de 24 heures, Amazon Keyspaces prend les mesures suivantes pour supprimer définitivement la région de la répliquion multirégionale :

- Supprimez la région désactivée de tous les espaces clés de répliquion multi-régions.
- Convertissez les répliquions de tables de répliquion multirégions dans la région désactivée en espaces clés et tables à région unique.
- Amazon Keyspaces ne supprime aucune ressource de la région désactivée.

Une fois qu'Amazon Keyspaces a définitivement supprimé la région désactivée de l'espace clé multi-régions, vous ne pouvez plus ajouter la région désactivée.

Multi-Region répliquion et intégration avec restauration instantanée (PITR)

Point-in-time la restauration est prise en charge pour les tables multi-régions. Pour restaurer correctement une table multi-régions avec PITR, les conditions suivantes doivent être remplies.

- La table source et la table cible doivent être configurées en tant que tables multi-régions.
- Les régions de répliquion pour l'espace clé de la table source et pour l'espace clé de la table cible doivent être identiques.
- PITR doit être activé sur toutes les répliquions de la table source.

Vous pouvez exécuter l'instruction de restauration depuis n'importe quelle région dans laquelle la table source est disponible. Amazon Keyspaces restaure automatiquement la table cible dans chaque région. Pour plus d'informations sur le PITR, consultez [the section called “Comment ça marche”](#).

Lorsque vous créez une table multi-régions, les paramètres Pitr que vous définissez au cours du processus de création sont automatiquement appliqués à toutes les tables de toutes les régions. Lorsque vous modifiez les paramètres Pitr à l'aide de `ALTER TABLE`, Amazon Keyspaces applique la mise à jour uniquement à la table locale et non aux répliques des autres régions. Pour activer Pitr pour une table multi-régions existante, vous devez répéter l'`ALTER TABLE` instruction pour toutes les répliques.

Multi-Region réplication et intégration avec AWS services

Vous pouvez surveiller les performances de réplication entre des tables de différentes tables à Régions AWS l'aide CloudWatch des métriques Amazon. La métrique suivante fournit une surveillance continue des espaces clés multirégionaux.

- `ReplicationLatency`— Cette métrique mesure le temps nécessaire pour répliquer `updates` `inserts`, ou pour passer d'une table de `letes` de répliques à une autre table de répliques dans un espace clé multirégional.

Pour plus d'informations sur la façon de surveiller CloudWatch les métriques, consultez [the section called “Surveillance avec CloudWatch”](#).

Notes d'utilisation de la réplication multirégionale Amazon Keyspaces

Tenez compte des points suivants lorsque vous utilisez la réplication multirégionale avec Amazon Keyspaces.

- Vous pouvez sélectionner n'importe quel [public disponible](#) Régions AWS. Pour plus d'informations sur Régions AWS [ceux qui sont désactivés par défaut](#), consultez [the section called “Régions désactivées par défaut”](#).
- AWS GovCloud (US) Regions et les régions de Chine ne sont pas prises en charge.
- Envisagez les solutions de contournement suivantes jusqu'à ce que les fonctionnalités soient disponibles :

Configurez le temps de vie (TTL) lors de la création de la table multirégionale. Vous ne pourrez ni activer ni désactiver le TTL, ni ajuster la valeur TTL ultérieurement. Pour de plus amples informations, veuillez consulter [the section called “Expirer les données avec Time to Live”](#).

- Pour le chiffrement au repos, utilisez une clé AWS détenue. Les clés gérées par le client ne sont actuellement pas prises en charge pour les tables multirégionales. Pour plus d'informations, veuillez consulter la rubrique

[the section called “Comment ça marche”](#).

- Vous pouvez l'utiliser ALTER KEYSPACE pour ajouter une région à un espace clé à région unique ou multirégion. Pour de plus amples informations, veuillez consulter [the section called “Ajouter une région à un keyspace”](#).
- Avant d'ajouter une région à un espace clé à région unique, assurez-vous qu'aucune table située sous cet espace clé n'est configurée avec des clés gérées par le client.
- Les balises existantes configurées pour les espaces clés ou les tables ne sont pas répliquées dans la nouvelle région.
- Lorsque vous utilisez la gestion des capacités allouées avec le dimensionnement automatique d'Amazon Keyspaces, veuillez à utiliser les opérations de l'API Amazon Keyspaces pour créer et configurer vos tables multirégionales. Les opérations d'API Application Auto Scaling sous-jacentes qu'Amazon Keyspaces appelle en votre nom ne disposent pas de fonctionnalités multirégionales.

Pour de plus amples informations, veuillez consulter [the section called “Mettre à jour les paramètres de capacité allouée et de dimensionnement automatique pour une table multirégionale”](#). Pour plus d'informations sur la manière d'estimer le débit de capacité d'écriture des tables multirégionales provisionnées, consultez [the section called “Estimation de la capacité d'une table multirégionale”](#).

- Bien que les données soient automatiquement répliquées dans les régions sélectionnées d'une table multirégionale, lorsqu'un client se connecte à un point de terminaison dans une région et interroge la `system.peers` table, la requête renvoie uniquement des informations locales. Le résultat de la requête apparaît comme un cluster de centre de données unique pour le client.
- La réplication multirégionale d'Amazon Keyspaces est asynchrone et permet de garantir la cohérence des écritures. LOCAL_QUORUM LOCAL_QUORUM La cohérence nécessite qu'une mise à jour d'une ligne soit maintenue de manière durable sur deux répliques de la région locale avant de renvoyer le succès au client. La propagation des écritures vers la région (ou les régions) répliquée est ensuite effectuée de manière asynchrone.

La réplication multirégionale d'Amazon Keyspaces ne prend pas en charge la réplication synchrone ou la cohérence. QUORUM

- Lorsque vous créez un espace de touches ou un tableau multirégional, toutes les balises que vous définissez au cours du processus de création sont automatiquement appliquées à tous les espaces clés et tables de toutes les régions. Lorsque vous modifiez les balises existantes à l'aide de ALTER KEYSPACE ou ALTER TABLE, la mise à jour ne s'applique qu'au keyspace ou au tableau de la région dans laquelle vous effectuez la modification.

- Amazon CloudWatch fournit une `ReplicationLatency` métrique pour chaque région répliquée. Il calcule cette métrique en suivant les lignes qui arrivent, en comparant leur heure d'arrivée avec leur temps d'écriture initial et en calculant une moyenne. Les horaires sont enregistrés CloudWatch dans la région source. Pour de plus amples informations, veuillez consulter [the section called “Surveillance avec CloudWatch”](#).

Il peut être utile de consulter les durées moyennes et maximales afin de déterminer le délai de réplication moyen et le plus défavorable. Il n'existe aucun SLA sur cette latence.

- Lorsque vous utilisez une table multirégionale en mode à la demande, vous pouvez observer une augmentation de la latence pour la réplication asynchrone des écritures si une réplique de table connaît un nouveau pic de trafic. De la même manière qu'Amazon Keyspaces adapte automatiquement la capacité d'une table à la demande composée d'une seule région au trafic d'application qu'elle reçoit, Amazon Keyspaces adapte automatiquement la capacité d'une réplique de table à la demande multirégionale au trafic qu'elle reçoit. L'augmentation de la latence de réplication est transitoire car Amazon Keyspaces alloue automatiquement davantage de capacité à mesure que le volume de trafic augmente. Une fois que toutes les répliques se sont adaptées à votre volume de trafic, la latence de réplication devrait revenir à la normale. Pour de plus amples informations, veuillez consulter [the section called “Trafic de pointe et propriétés de scalabilité”](#).
- Lorsque vous utilisez une table multirégionale en mode provisionné, si votre application dépasse la capacité de débit allouée, vous pouvez observer des erreurs de capacité insuffisantes et une augmentation de la latence de réplication. Pour garantir une capacité de lecture et d'écriture suffisante pour toutes les répliques de table dans l'ensemble Régions AWS d'une table multirégionale, nous vous recommandons de configurer le dimensionnement automatique d'Amazon Keyspaces. L'autoscaling d'Amazon Keyspaces vous aide à allouer efficacement la capacité de débit pour les charges de travail variables en ajustant automatiquement la capacité de débit en fonction du trafic réel des applications. Pour de plus amples informations, veuillez consulter [the section called “Fonctionnement de la mise à l'échelle automatique pour les tables multirégions”](#).

Configurer la réplication multirégionale pour Amazon Keyspaces (pour Apache Cassandra)

Vous pouvez utiliser la console, le Cassandra Query Language (CQL), ou le AWS Command Line Interface pour créer et gérer des espaces clés et des tables multirégionaux dans Amazon Keyspaces.

Cette section fournit des exemples de création et de gestion d'espaces clés et de tables multirégionaux. Toutes les tables que vous créez dans un espace de touches multirégional héritent automatiquement des paramètres multirégionaux de l'espace de touches.

Pour plus d'informations sur les configurations et fonctionnalités prises en charge, consultez [the section called “Notes d’utilisation”](#).

Rubriques

- [Configurer les autorisations IAM requises pour créer des espaces clés et des tables multirégionaux](#)
- [Configurer les autorisations IAM requises pour ajouter un Région AWS à un keyspace](#)
- [Création d'un keyspace multirégional dans Amazon Keyspaces](#)
- [Ajouter un Région AWS à un espace de touches dans Amazon Keyspaces](#)
- [Vérifiez la progression de la réplication lors de l'ajout d'une nouvelle région à un keyspace](#)
- [Création d'un tableau multirégional avec les paramètres par défaut dans Amazon Keyspaces](#)
- [Création d'une table multirégionale en mode provisionné avec mise à l'échelle automatique dans Amazon Keyspaces](#)
- [Mettre à jour les paramètres de capacité allouée et de dimensionnement automatique pour une table multirégionale dans Amazon Keyspaces](#)
- [Afficher les paramètres de capacité allouée et de dimensionnement automatique pour une table multirégionale dans Amazon Keyspaces](#)
- [Désactiver le redimensionnement automatique pour un tableau dans Amazon Keyspaces](#)
- [Définissez manuellement la capacité allouée d'une table multirégionale dans Amazon Keyspaces](#)

Configurer les autorisations IAM requises pour créer des espaces clés et des tables multirégionaux

Pour créer correctement des espaces clés et des tables multirégionaux, le principal IAM doit être en mesure de créer un rôle lié à un service. Ce rôle lié à un service est un type unique de rôle IAM prédéfini par Amazon Keyspaces. Il inclut toutes les autorisations dont Amazon Keyspaces a besoin pour effectuer des actions en votre nom. Pour de plus amples informations sur le rôle lié à un service, veuillez consulter [the section called “Multi-Region Réplication”](#).

Pour créer le rôle lié au service requis par la réplication multirégionale, la politique du principal IAM nécessite les éléments suivants :

- `iam:CreateServiceLinkedRole`— L'action que le principal peut effectuer.

- `arn:aws:iam::*:role/aws-service-role/replication.cassandra.amazonaws.com/AWSServiceRoleForKeyspacesReplication`— La ressource sur laquelle l'action peut être exécutée.
- `iam:AWSServiceName`: `"replication.cassandra.amazonaws.com"`— Le seul AWS service auquel ce rôle peut être associé est Amazon Keyspaces.

Voici un exemple de politique qui accorde les autorisations minimales requises à un directeur pour créer des espaces clés et des tables multirégionaux.

```
{
    "Effect": "Allow",
    "Action": "iam:CreateServiceLinkedRole",
    "Resource": "arn:aws:iam::*:role/aws-service-role/
replication.cassandra.amazonaws.com/AWSServiceRoleForKeyspacesReplication",
    "Condition": {"StringLike": {"iam:AWSServiceName":
"replication.cassandra.amazonaws.com"}}
}
```

Pour obtenir des autorisations IAM supplémentaires pour les espaces clés et les tables multirégionaux, consultez les [actions, les ressources et les clés de condition pour Amazon Keyspaces \(pour Apache Cassandra\)](#) dans le Service Authorization Reference.

Configurer les autorisations IAM requises pour ajouter un Région AWS à un keyspace

Pour ajouter une région à un keyspace, le principal IAM a besoin des autorisations suivantes :

- `cassandra:Alter`
- `cassandra:AlterMultiRegionResource`
- `cassandra:Create`
- `cassandra:CreateMultiRegionResource`
- `cassandra:Select`
- `cassandra:SelectMultiRegionResource`
- `cassandra:Modify`
- `cassandra:ModifyMultiRegionResource`

Si la table est configurée en mode provisionné avec le dimensionnement automatique activé, les autorisations supplémentaires suivantes sont nécessaires.

- `application-autoscaling:RegisterScalableTarget`
- `application-autoscaling:DeregisterScalableTarget`
- `application-autoscaling:DescribeScalableTargets`
- `application-autoscaling:PutScalingPolicy`
- `application-autoscaling:DescribeScalingPolicies`

Pour ajouter correctement une région à un espace clé à région unique, le principal IAM doit également être en mesure de créer un rôle lié à un service. Ce rôle lié à un service est un type unique de rôle IAM prédéfini par Amazon Keyspaces. Il inclut toutes les autorisations dont Amazon Keyspaces a besoin pour effectuer des actions en votre nom. Pour de plus amples informations sur le rôle lié à un service, veuillez consulter [the section called “Multi-Region Réplication”](#).

Pour créer le rôle lié au service requis par la réplication multirégionale, la politique du principal IAM nécessite les éléments suivants :

- `iam:CreateServiceLinkedRole`— L'action que le principal peut effectuer.
- `arn:aws:iam::*:role/aws-service-role/replication.cassandra.amazonaws.com/AWSServiceRoleForKeyspacesReplication`— La ressource sur laquelle l'action peut être exécutée.
- `iam:AWSServiceName`: `"replication.cassandra.amazonaws.com"`— Le seul AWS service auquel ce rôle peut être associé est Amazon Keyspaces.

Voici un exemple de politique qui accorde les autorisations minimales requises à un directeur pour ajouter une région à un keyspace.

```
{
    "Effect": "Allow",
    "Action": "iam:CreateServiceLinkedRole",
    "Resource": "arn:aws:iam::*:role/aws-service-role/
replication.cassandra.amazonaws.com/AWSServiceRoleForKeyspacesReplication",
    "Condition": {"StringLike": {"iam:AWSServiceName":
"replication.cassandra.amazonaws.com"}}
}
```

Création d'un keyspace multirégional dans Amazon Keyspaces

Cette section fournit des exemples de création d'un espace de touches multirégional. Vous pouvez le faire sur la console Amazon Keyspaces à l'aide de CQL ou du AWS CLI. Toutes les tables que vous créez dans un espace de touches multirégional héritent automatiquement des paramètres multirégionaux de l'espace de touches.

Note

Lorsque vous créez un espace de saisie multirégional, Amazon Keyspaces crée un rôle lié à un service dont le nom figure dans votre compte. `AWSServiceRoleForAmazonKeyspacesReplication`. Ce rôle permet à Amazon Keyspaces de répliquer les écritures sur toutes les répliques d'une table multirégionale en votre nom. Pour en savoir plus, veuillez consulter la section [the section called “Multi-Region Réplication”](#).

Console

Création d'un espace de touches multirégional (console)

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le panneau de navigation, choisissez Keyspaces, puis Créer un keyspace.
3. Dans le champ Nom de l'espace clé, entrez le nom de l'espace clé.
4. Dans la section de Multi-Region réplication, vous pouvez ajouter les régions supplémentaires disponibles dans la liste.
5. Pour terminer, choisissez Create keyspace.

Cassandra Query Language (CQL)

Création d'un espace de touches multirégional à l'aide de CQL

1. Pour créer un espace de touches multirégional, utilisez cette option `NetworkTopologyStrategy` pour spécifier l'espace de touches dans Régions AWS le quel l'espace de touches doit être répliqué. Vous devez inclure votre région actuelle et au moins une région supplémentaire.

Toutes les tables du keyspace héritent de la stratégie de réplication du keyspace. Vous ne pouvez pas modifier la stratégie de réplication au niveau de la table.

NetworkTopologyStrategy— Le facteur de réplication pour chaque région est de trois, car Amazon Keyspaces réplique les données dans trois [zones de disponibilité](#) au sein d'une même région Région AWS, par défaut.

L'instruction CQL suivante en est un exemple.

```
CREATE KEYSPACE mykeyspace
WITH REPLICATION = {'class':'NetworkTopologyStrategy', 'us-east-1':'3', 'ap-southeast-1':'3', 'eu-west-1':'3' };
```

- Vous pouvez utiliser une instruction CQL pour interroger la table `system_multiregion_info.tables` afin de répertorier par programmation les régions et le statut de la table multirégionale que vous spécifiez. Le code suivant en est un exemple.

```
SELECT * from system_multiregion_info.tables WHERE keyspace_name = 'mykeyspace'
AND table_name = 'mytable';
```

Le résultat de l'instruction ressemble à ce qui suit :

keyspace_name	table_name	region	status
mykeyspace	mytable	us-east-1	ACTIVE
mykeyspace	mytable	ap-southeast-1	ACTIVE
mykeyspace	mytable	eu-west-1	ACTIVE

CLI

Créez un nouvel espace de touches multirégional à l'aide du AWS CLI

- Pour créer un espace de touches multirégional, vous pouvez utiliser l'instruction CLI suivante. Spécifiez votre région actuelle et au moins une région supplémentaire dans `leregionList`.

```
aws keyspaces create-keyspace --keyspace-name mykeyspace \
```

```
--replication-specification replicationStrategy=MULTI_REGION,regionList=us-east-1,eu-west-1
```

Pour créer une table multirégionale, reportez-vous aux sections [the section called “Création d'une table multirégionale avec les paramètres par défaut”](#) et [the section called “Création d'une table multirégionale en mode provisionné”](#).

Ajouter un Région AWS à un espace de touches dans Amazon Keyspaces

Vous pouvez ajouter une nouvelle touche Région AWS à un espace de touches unique ou multirégional. La nouvelle région de réplique est appliquée à toutes les tables du keyspace.

Pour transformer un espace de touches à région unique en espace de touches multirégional, vous devez activer les horodatages côté client pour toutes les tables de l'espace de touches. Pour de plus amples informations, veuillez consulter [the section called “Client-side horodatages”](#).

Si vous ajoutez une région supplémentaire à un espace clé multirégional, Amazon Keyspaces doit répliquer les tables existantes dans la nouvelle région en effectuant une restauration unique entre régions pour chaque table existante. Les frais de restauration pour chaque table sont facturés par Go. Pour plus d'informations, consultez [Backup and restore](#) sur la page de tarification d'Amazon Keyspaces (pour Apache Cassandra). Le transfert de données entre les régions est gratuit pour cette opération de restauration. Outre les données, toutes les propriétés des tables, à l'exception des balises, seront répliquées dans la nouvelle région.

Vous pouvez utiliser l'`ALTER KEYSPACE` instruction dans CQL, la `update-keyspace` commande associée au AWS CLI, ou la console pour ajouter une nouvelle région à un keyspace unique ou multirégional dans Amazon Keyspaces. Pour que le relevé soit correctement exécuté, le compte que vous utilisez doit être situé dans l'une des régions où le keyspace est déjà disponible. Pendant l'ajout de la réplique, vous ne pouvez effectuer aucune autre opération du langage de définition des données (DDL) sur les ressources mises à jour et répliquées.

Pour plus d'informations sur les autorisations requises pour ajouter une région, consultez [the section called “Configurer les autorisations IAM pour ajouter une région”](#).

Note

Lorsque vous ajoutez une région supplémentaire à un espace clé à région unique, Amazon Keyspaces crée un rôle lié à un service dont le nom figure dans votre compte. `AWSServiceRoleForAmazonKeyspacesReplication` Ce rôle permet à Amazon

Keyspaces de répliquer des tables vers de nouvelles régions et de répliquer les écritures d'une table vers toutes les répliques d'une table multirégionale en votre nom. Pour en savoir plus, veuillez consulter la section [the section called “Multi-Region Réplication”](#).

Console

Suivez ces étapes pour ajouter une région à un espace de touches à l'aide de la console Amazon Keyspaces.

Ajouter une région à un espace de touches (console)

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le volet de navigation, choisissez Keyspaces, puis choisissez un espace de touches dans la liste.
3. Cliquez sur l'onglet Régions AWS.
4. Régions AWS Dans l'onglet, choisissez Ajouter une région.
5. Dans la boîte de dialogue Ajouter une région, choisissez la région supplémentaire que vous souhaitez ajouter au keyspace.
6. Pour terminer, choisissez Ajouter.

Cassandra Query Language (CQL)

Ajouter une région à un espace de touches à l'aide de CQL

- Pour ajouter une nouvelle région à un keyspace, vous pouvez utiliser l'instruction suivante. Dans cet exemple, l'espace de touches est déjà disponible dans les régions USA Est (Virginie du Nord) et USA Ouest (Oregon), et l'instruction CQL ajoute la région USA Ouest (Californie du Nord).

```
ALTER KEYSPACE my_keyspace
WITH REPLICATION = {
  'class': 'NetworkTopologyStrategy',
  'us-east-1': '3',
  'us-west-2': '3',
  'us-west-1': '3'
} AND CLIENT_SIDE_TIMESTAMPS = {'status': 'ENABLED'};
```

CLI

Ajoutez une région à un espace de touches à l'aide du AWS CLI

- Pour ajouter une nouvelle région à un espace de touches à l'aide de la CLI, vous pouvez utiliser l'exemple suivant. Notez que la valeur par défaut pour `client-side-timestamps` est `DISABLED`. Avec la `update-keyspace` commande, vous devez modifier la valeur en `ENABLED`.

```
aws keyspace update-keyspace \  
--keyspace-name my_keyspace \  
--replication-specification '{"replicationStrategy": "MULTI_REGION",  
"regionList": ["us-east-1", "eu-west-1", "eu-west-3"] }' \  
--client-side-timestamps '{"status": "ENABLED"}'
```

Vérifiez la progression de la réplication lors de l'ajout d'une nouvelle région à un keyspace

L'ajout d'une nouvelle région à un espace de touches Amazon Keyspaces est une opération de longue haleine. Pour suivre les progrès, vous pouvez utiliser les requêtes présentées dans cette section.

Cassandra Query Language (CQL)

Utilisation de CQL pour vérifier la progression de l'ajout d'une région

- Pour vérifier la progression de la création des nouvelles répliques de table dans un espace de touches donné, vous pouvez interroger la `system_multiregion_info.keyspaces` table. L'instruction CQL suivante en est un exemple.

```
SELECT keyspace_name, region, status, tables_replication_progress  
FROM system_multiregion_info.keyspaces  
WHERE keyspace_name = 'my_keyspace';
```

Pendant qu'une opération de réplication est en cours, le statut indique la progression de la création de la table dans la nouvelle région. Il s'agit d'un exemple où 5 tables sur 10 ont été répliquées dans la nouvelle région.

keyspace_name	region	status	tables_replication_progress
---------------	--------	--------	-----------------------------

```

-----+-----+-----+-----
my_keyspace | us-east-1 | Updating |
my_keyspace | us-west-2 | Updating |
my_keyspace | eu-west-1 | Creating  | 50%

```

Une fois le processus de réplication terminé avec succès, le résultat doit ressembler à cet exemple.

```

keyspace_name | region    | status
-----+-----+-----
my_keyspace   | us-east-1 | Active
my_keyspace   | us-west-2 | Active
my_keyspace   | eu-west-1 | Active

```

CLI

Utilisation du AWS CLI pour vérifier la progression de l'ajout d'une région

- Pour confirmer l'état de la création de répliques de table pour un espace de touches donné, vous pouvez utiliser l'exemple suivant.

```

aws keyspaces get-keyspace \
--keyspace-name my_keyspace

```

La sortie doit ressembler à cet exemple.

```

{
  "keyspaceName": "my_keyspace",
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
my_keyspace/",
  "replicationStrategy": "MULTI_REGION",
  "replicationRegions": [
    "us-east-1",
    "eu-west-1"
  ]
  "replicationGroupStatus": [
    {
      "RegionName": "us-east-1",
      "KeyspaceStatus": "Active"
    },

```

```
{
  "RegionName": "eu-west-1",
  "KeyspaceStatus": "Creating",
  "TablesReplicationProgress": "50.0%"
}
```

Création d'un tableau multirégional avec les paramètres par défaut dans Amazon Keyspaces

Cette section fournit des exemples de création d'une table multirégionale en mode à la demande avec tous les paramètres par défaut. Vous pouvez le faire sur la console Amazon Keyspaces à l'aide de CQL ou du AWS CLI. Toutes les tables que vous créez dans un espace de touches multirégional héritent automatiquement des paramètres multirégionaux de l'espace de touches.

Pour créer un espace de touches multirégional, voir. [the section called “Création d'un espace de saisie multirégional”](#)

Console

Création d'un tableau multirégional avec les paramètres par défaut (console)

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Choisissez un espace de touches multirégional.
3. Dans l'onglet Tables, choisissez Créer une table.
4. Dans Nom de la table, entrez le nom de la table. Le Régions AWS fichier dans lequel cette table est répliquée est indiqué dans la boîte d'information.
5. Continuez avec le schéma de table.
6. Sous Paramètres du tableau, passez à l'option Paramètres par défaut. Notez les paramètres par défaut suivants pour les tables multirégionales.
 - Mode capacité : le mode capacité par défaut est On-demand. Pour plus d'informations sur la configuration du mode provisionné, consultez [the section called “Création d'une table multirégionale en mode provisionné”](#).
 - Gestion des clés de chiffrement : seule l'Clé détenue par AWSOption est prise en charge.

- Client-side horodatages — Cette fonctionnalité est requise pour les tables multirégionales.
- Choisissez Personnaliser les paramètres si vous devez activer Time to Live (TTL) pour le tableau et toutes ses répliques.

 Note

Vous ne pourrez pas modifier les paramètres TTL sur une table multirégionale existante.

7. Pour terminer, choisissez Créer une table.

Cassandra Query Language (CQL)

Création d'une table multirégionale en mode à la demande avec les paramètres par défaut

- Pour créer une table multirégionale avec des paramètres par défaut, vous pouvez utiliser l'instruction CQL suivante.

```
CREATE TABLE mykeyspace.mytable(pk int, ck int, PRIMARY KEY (pk, ck))
  WITH CUSTOM_PROPERTIES = {
    'capacity_mode':{
      'throughput_mode':'PAY_PER_REQUEST'
    },
    'point_in_time_recovery':{
      'status':'enabled'
    },
    'encryption_specification':{
      'encryption_type':'AWS_OWNED_KMS_KEY'
    },
    'client_side_timestamps':{
      'status':'enabled'
    }
  };
```

CLI

À l'aide du AWS CLI

1. Pour créer une table multirégionale avec les paramètres par défaut, il suffit de spécifier le schéma. Vous pouvez utiliser l'exemple suivant.

```
aws keyspaces create-table --keyspace-name mykeyspace --table-name mytable \
--schema-definition 'allColumns=[{name=pk,type=int}],partitionKeys={name= pk}'
```

Le résultat de la commande est le suivant :

```
{
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
mykeyspace/table/mytable"
}
```

2. Pour confirmer les paramètres du tableau, vous pouvez utiliser l'instruction suivante.

```
aws keyspaces get-table --keyspace-name mykeyspace --table-name mytable
```

La sortie affiche tous les paramètres par défaut d'une table multirégionale.

```
{
  "keyspaceName": "mykeyspace",
  "tableName": "mytable",
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
mykeyspace/table/mytable",
  "creationTimestamp": "2023-12-19T16:50:37.639000+00:00",
  "status": "ACTIVE",
  "schemaDefinition": {
    "allColumns": [
      {
        "name": "pk",
        "type": "int"
      }
    ],
    "partitionKeys": [
      {
        "name": "pk"
      }
    ]
  }
}
```

```
    "clusteringKeys": [],
    "staticColumns": []
  },
  "capacitySpecification": {
    "throughputMode": "PAY_PER_REQUEST",
    "lastUpdateToPayPerRequestTimestamp": "2023-12-19T16:50:37.639000+00:00"
  },
  "encryptionSpecification": {
    "type": "AWS_OWNED_KMS_KEY"
  },
  "pointInTimeRecovery": {
    "status": "DISABLED"
  },
  "defaultTimeToLive": 0,
  "comment": {
    "message": ""
  },
  "clientSideTimestamps": {
    "status": "ENABLED"
  },
  "replicaSpecifications": [
    {
      "region": "us-east-1",
      "status": "ACTIVE",
      "capacitySpecification": {
        "throughputMode": "PAY_PER_REQUEST",
        "lastUpdateToPayPerRequestTimestamp": 1702895811.469
      }
    },
    {
      "region": "eu-north-1",
      "status": "ACTIVE",
      "capacitySpecification": {
        "throughputMode": "PAY_PER_REQUEST",
        "lastUpdateToPayPerRequestTimestamp": 1702895811.121
      }
    }
  ]
}
```

Création d'une table multirégionale en mode provisionné avec mise à l'échelle automatique dans Amazon Keyspaces

Cette section fournit des exemples de création d'une table multirégionale en mode provisionné avec mise à l'échelle automatique. Vous pouvez le faire sur la console Amazon Keyspaces à l'aide de CQL ou du AWS CLI

Pour plus d'informations sur les configurations prises en charge et les fonctionnalités de réplication multirégionale, consultez [the section called “Notes d'utilisation”](#).

Pour créer un espace de touches multirégional, voir. [the section called “Création d'un espace de saisie multirégional”](#)

Lorsque vous créez une nouvelle table multirégionale en mode provisionné avec des paramètres de dimensionnement automatique, vous pouvez spécifier les paramètres généraux de la table qui sont valides pour tous les Régions AWS éléments dans lesquels la table est répliquée. Vous pouvez ensuite remplacer les paramètres de capacité de lecture et lire les paramètres de mise à l'échelle automatique pour chaque réplique. La capacité d'écriture reste toutefois synchronisée entre toutes les répliques afin de garantir une capacité suffisante pour répliquer les écritures dans toutes les régions.

Note

Le dimensionnement automatique d'Amazon Keyspaces nécessite la présence d'un rôle lié à un service (AWSServiceRoleForApplicationAutoScaling_CassandraTable) qui exécute des actions de dimensionnement automatique en votre nom. Ce rôle est créé automatiquement pour vous. Pour de plus amples informations, veuillez consulter [the section called “Utilisation des rôles liés à un service”](#).

Console

Création d'une nouvelle table multirégionale avec mise à l'échelle automatique activée

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Choisissez un espace de touches multirégional.
3. Dans l'onglet Tables, choisissez Créer une table.

4. Sur la page Créer une table dans la section Détails de la table, sélectionnez un espace de touche et donnez un nom à la nouvelle table.
5. Dans la section Colonnes, créez le schéma de votre table.
6. Dans la section Clé primaire, définissez la clé primaire de la table et sélectionnez les colonnes de clustering facultatives.
7. Dans la section Paramètres du tableau, choisissez Personnaliser les paramètres.
8. Passez aux paramètres Read/write de capacité.
9. Pour le Mode de capacité, choisissez Provisioned (Alloué).
10. Dans la section Capacité de lecture confirmez que l'option Mettre à l'échelle automatiquement est sélectionnée.

Vous pouvez choisir de configurer les mêmes unités de capacité de lecture pour tous les éléments dans Régions AWS auxquels la table est répliquée. Vous pouvez également décocher la case et configurer différemment la capacité de lecture pour chaque région.

Si vous choisissez de configurer chaque région différemment, vous sélectionnez les unités de capacité de lecture minimale et maximale pour chaque réplique de table, ainsi que l'utilisation cible.

- Unités de capacité minimale : entrez la valeur du niveau de débit minimal que le tableau doit toujours être prêt à prendre en charge. La valeur doit être comprise entre 1 et le quota par seconde du débit maximal pour votre compte (40 000 par défaut).
- Unités de capacité maximale : entrez le débit maximal que vous souhaitez allouer pour la table. La valeur doit être comprise entre 1 et le quota par seconde du débit maximal pour votre compte (40 000 par défaut).
- Utilisation cible — Entrez un taux d'utilisation cible compris entre 20 % et 90 %. Lorsque le trafic dépasse le taux d'utilisation cible défini, la capacité est automatiquement mise à l'échelle. Lorsque le trafic tombe en dessous de la cible définie, il est automatiquement réduit de nouveau.
- Désactivez la case à cocher Echelle automatique si vous souhaitez configurer manuellement la capacité de lecture du tableau. Ce paramètre s'applique à toutes les répliques de la table.

 Note

Pour garantir une capacité de lecture suffisante pour toutes les répliques, nous recommandons le dimensionnement automatique d'Amazon Keyspaces pour les tables multirégionales provisionnées.

 Note

Pour en savoir plus sur les quotas par défaut de votre compte et sur la façon de les augmenter, reportez-vous à la section [Quotas](#).

11. Dans la section Capacité d'écriture, vérifiez que l'option Échelle automatique est sélectionnée. Configurez ensuite les unités de capacité de la table. Les unités de capacité d'écriture restent synchronisées dans l'ensemble Régions AWS afin de garantir une capacité suffisante pour répliquer les événements d'écriture dans les régions.
 - Désélectionnez Scale automatiquement si vous souhaitez provisionner manuellement la capacité d'écriture de la table. Ce paramètre s'applique à toutes les répliques de la table.

 Note

Pour garantir une capacité d'écriture suffisante pour toutes les répliques, nous recommandons le dimensionnement automatique d'Amazon Keyspaces pour les tables multirégionales provisionnées.

12. Choisissez Créer un tableau. Votre table est créée avec les paramètres de mise à l'échelle automatique spécifiés.

Cassandra Query Language (CQL)

Création d'une table multirégionale avec mode capacité provisionnée et mise à l'échelle automatique à l'aide de CQL

- Pour créer une table multirégionale en mode provisionné avec mise à l'échelle automatique, vous devez d'abord spécifier le mode de capacité en le définissant `CUSTOM_PROPERTIES` pour la table. Après avoir spécifié le mode de capacité provisionnée, vous pouvez

configurer les paramètres de dimensionnement automatique de la table à l'aide `AUTOSCALING_SETTINGS` de.

Pour des informations détaillées sur les paramètres de mise à l'échelle automatique, la politique de suivi des cibles, la valeur cible et les paramètres facultatifs, consultez [the section called “Création d'un nouveau tableau avec mise à l'échelle automatique”](#).

Pour définir la capacité de lecture d'une réplique de table dans une région spécifique, vous pouvez configurer les paramètres suivants dans le cadre de la table `replica_updates` :

- La région
- Les unités de capacité de lecture provisionnées (facultatif)
- Paramètres de mise à l'échelle automatique pour la capacité de lecture (facultatif)

L'exemple suivant montre une `CREATE TABLE` instruction pour une table multirégionale en mode provisionné. Les paramètres généraux de mise à l'échelle automatique des capacités d'écriture et de lecture sont les mêmes. Toutefois, les paramètres de mise à l'échelle automatique de lecture spécifient des périodes de recharge supplémentaires de 60 secondes avant d'augmenter ou de diminuer la capacité de lecture du tableau. En outre, les paramètres de mise à l'échelle automatique de la capacité de lecture pour la région USA Est (Virginie du Nord) sont supérieurs à ceux des autres répliques. De plus, la valeur cible est fixée à 70 % au lieu de 50 %.

```
CREATE TABLE mykeyspace.mytable(pk int, ck int, PRIMARY KEY (pk, ck))
WITH CUSTOM_PROPERTIES = {
    'capacity_mode': {
        'throughput_mode': 'PROVISIONED',
        'read_capacity_units': 5,
        'write_capacity_units': 5
    }
} AND AUTOSCALING_SETTINGS = {
    'provisioned_write_capacity_autoscaling_update': {
        'maximum_units': 10,
        'minimum_units': 5,
        'scaling_policy': {
            'target_tracking_scaling_policy_configuration': {
                'target_value': 50
            }
        }
    }
},
```

```

    'provisioned_read_capacity_autoscaling_update': {
      'maximum_units': 10,
      'minimum_units': 5,
      'scaling_policy': {
        'target_tracking_scaling_policy_configuration': {
          'target_value': 50,
          'scale_in_cooldown': 60,
          'scale_out_cooldown': 60
        }
      }
    },
    'replica_updates': {
      'us-east-1': {
        'provisioned_read_capacity_autoscaling_update': {
          'maximum_units': 20,
          'minimum_units': 5,
          'scaling_policy': {
            'target_tracking_scaling_policy_configuration': {
              'target_value': 70
            }
          }
        }
      }
    }
  }
};

```

CLI

Créez une nouvelle table multirégionale en mode provisionné avec mise à l'échelle automatique à l'aide du AWS CLI

- Pour créer une table multirégionale en mode provisionné avec une configuration de dimensionnement automatique, vous pouvez utiliser le. AWS CLI Notez que vous devez utiliser la `create-table` commande Amazon Keyspaces CLI pour configurer les paramètres de mise à l'échelle automatique multirégionale. Cela est dû au fait qu'Application Auto Scaling, le service qu'Amazon Keyspaces utilise pour effectuer le dimensionnement automatique en votre nom, ne prend pas en charge plusieurs régions.

Pour plus d'informations sur les paramètres de mise à l'échelle automatique, la politique de suivi des cibles, la valeur cible et les paramètres facultatifs, consultez [the section called “Création d'un nouveau tableau avec mise à l'échelle automatique”](#).

Pour définir la capacité de lecture d'une réplique de table dans une région spécifique, vous pouvez configurer les paramètres suivants dans le cadre de la table `replicaSpecifications` :

- La région
- Les unités de capacité de lecture provisionnées (facultatif)
- Paramètres de mise à l'échelle automatique pour la capacité de lecture (facultatif)

Lorsque vous créez des tables multirégionales provisionnées avec des paramètres de dimensionnement automatique complexes et différentes configurations pour les répliques de tables, il est utile de charger les paramètres de mise à l'échelle automatique et les configurations de réplique de la table à partir de fichiers JSON.

Pour utiliser l'exemple de code suivant, vous pouvez télécharger les exemples de fichiers JSON depuis [auto-scaling.zip](#), puis extraire `auto-scaling.json` et `replication.json`. Prenez note du chemin d'accès aux fichiers.

Dans cet exemple, les fichiers JSON se trouvent dans le répertoire actuel. Pour connaître les différentes options de chemin de fichier, consultez [Comment charger des paramètres à partir d'un fichier](#).

```
aws keyspaces create-table --keyspace-name mykeyspace --table-name mytable \
--schema-definition 'allColumns=[{name=pk,type=int},
{name=ck,type=int}],partitionKeys=[{name=pk},{name=ck}]' \
--capacity-specification
throughputMode=PROVISIONED,readCapacityUnits=1,writeCapacityUnits=1 \
--auto-scaling-specification file://auto-scaling.json \
--replica-specifications file://replication.json
```

Mettre à jour les paramètres de capacité allouée et de dimensionnement automatique pour une table multirégionale dans Amazon Keyspaces

Cette section inclut des exemples d'utilisation de la console, du CQL, et de gestion des paramètres AWS CLI de dimensionnement automatique d'Amazon Keyspaces pour les tables multirégionales provisionnées. Pour plus d'informations sur les options générales de configuration de l'autoscaling et sur leur fonctionnement, consultez [the section called “Gérez la capacité de débit avec la mise à l'échelle automatique”](#).

Notez que si vous utilisez le mode capacité provisionnée pour les tables multirégionales, vous devez toujours utiliser les appels d'API Amazon Keyspaces pour configurer le dimensionnement automatique. Cela est dû au fait que les opérations sous-jacentes de l'API Application Auto Scaling ne le sont pas Region-aware.

Pour plus d'informations sur la manière d'estimer le débit de capacité d'écriture des tables multirégionales provisionnées, consultez [the section called “Estimation de la capacité d'une table multirégionale”](#)

Pour plus d'informations sur l'API Amazon Keyspaces, consultez le manuel Amazon [Keyspaces](#) API Reference.

Lorsque vous mettez à jour le mode provisionné ou les paramètres de dimensionnement automatique d'une table multirégionale, vous pouvez mettre à jour les paramètres de capacité de lecture et la configuration de dimensionnement automatique de lecture pour chaque réplique de la table.

La capacité d'écriture reste toutefois synchronisée entre toutes les répliques afin de garantir une capacité suffisante pour répliquer les écritures dans toutes les régions.

Cassandra Query Language (CQL)

Mettre à jour les paramètres de capacité provisionnée et de dimensionnement automatique d'une table multirégionale à l'aide de CQL

- Vous pouvez l'utiliser `ALTER TABLE` pour mettre à jour le mode de capacité et les paramètres de mise à l'échelle automatique d'une table existante. Si vous mettez à jour une table actuellement en mode capacité à la demande, `capacity_mode` c'est obligatoire. Si votre table est déjà en mode capacité provisionnée, ce champ peut être omis.

Pour des informations détaillées sur les paramètres de mise à l'échelle automatique, la politique de suivi des cibles, la valeur cible et les paramètres facultatifs, consultez [the section called “Création d'un nouveau tableau avec mise à l'échelle automatique”](#).

Dans la même déclaration, vous pouvez également mettre à jour les paramètres de capacité de lecture et de mise à l'échelle automatique des répliques de tables dans des régions spécifiques en mettant à jour les `replica_updates` propriétés de la table. La déclaration suivante en est un exemple.

```
ALTER TABLE mykeyspace.mytable
WITH CUSTOM_PROPERTIES = {
```

```

    'capacity_mode': {
        'throughput_mode': 'PROVISIONED',
        'read_capacity_units': 1,
        'write_capacity_units': 1
    }
} AND AUTOSCALING_SETTINGS = {
    'provisioned_write_capacity_autoscaling_update': {
        'maximum_units': 10,
        'minimum_units': 5,
        'scaling_policy': {
            'target_tracking_scaling_policy_configuration': {
                'target_value': 50
            }
        }
    },
    'provisioned_read_capacity_autoscaling_update': {
        'maximum_units': 10,
        'minimum_units': 5,
        'scaling_policy': {
            'target_tracking_scaling_policy_configuration': {
                'target_value': 50,
                'scale_in_cooldown': 60,
                'scale_out_cooldown': 60
            }
        }
    },
    'replica_updates': {
        'us-east-1': {
            'provisioned_read_capacity_autoscaling_update': {
                'maximum_units': 20,
                'minimum_units': 5,
                'scaling_policy': {
                    'target_tracking_scaling_policy_configuration': {
                        'target_value': 70
                    }
                }
            }
        }
    }
};

```

CLI

Mettez à jour les paramètres de capacité provisionnée et de dimensionnement automatique d'une table multirégionale à l'aide du AWS CLI

- Pour mettre à jour le mode provisionné et la configuration de dimensionnement automatique d'une table existante, vous pouvez utiliser la AWS CLI `update-table` commande.

Notez que vous devez utiliser les commandes de la CLI Amazon Keyspaces pour créer ou modifier les paramètres de mise à l'échelle automatique multirégionale. Cela est dû au fait qu'Application Auto Scaling, le service qu'Amazon Keyspaces utilise pour effectuer le dimensionnement automatique de la capacité des tables en votre nom, ne prend pas en charge les tables multiples. Régions AWS

Pour mettre à jour la capacité de lecture d'une réplique de table dans une région spécifique, vous pouvez modifier l'un des paramètres facultatifs suivants de la table `replicaSpecifications` :

- Les unités de capacité de lecture provisionnées (facultatif)
- Paramètres de mise à l'échelle automatique pour la capacité de lecture (facultatif)

Lorsque vous mettez à jour des tables multirégionales avec des paramètres de mise à l'échelle automatique complexes et différentes configurations pour les répliques de tables, il est utile de charger les paramètres de mise à l'échelle automatique et les configurations de réplique de la table à partir de fichiers JSON.

Pour utiliser l'exemple de code suivant, vous pouvez télécharger les exemples de fichiers JSON depuis [auto-scaling.zip](#), puis extraire `auto-scaling.json` et `replication.json`. Prenez note du chemin d'accès aux fichiers.

Dans cet exemple, les fichiers JSON se trouvent dans le répertoire actuel. Pour connaître les différentes options de chemin de fichier, consultez [Comment charger des paramètres à partir d'un fichier](#).

```
aws keyspaces update-table --keyspace-name mykeyspace --table-name mytable \
--capacity-specification
    throughputMode=PROVISIONED,readCapacityUnits=1,writeCapacityUnits=1 \
--auto-scaling-specification file://auto-scaling.json \
--replica-specifications file://replication.json
```


Afficher les paramètres de capacité allouée et de dimensionnement automatique pour une table multirégionale dans Amazon Keyspaces

Vous pouvez consulter les paramètres de capacité provisionnée et de dimensionnement automatique d'une table multirégionale sur la console Amazon Keyspaces, à l'aide de CQL ou du AWS CLI. Cette section fournit des exemples illustrant comment procéder à l'aide de CQL et du AWS CLI.

Cassandra Query Language (CQL)

Afficher la capacité allouée et les paramètres de mise à l'échelle automatique d'une table multirégionale à l'aide de CQL

- Pour afficher la configuration de mise à l'échelle automatique d'une table multirégionale, utilisez la commande suivante.

```
SELECT * FROM system_multiregion_info.autoscaling WHERE keyspace_name =
'mykeyspace' AND table_name = 'mytable';
```

Le résultat de cette commande ressemble à ce qui suit :

```
keyspace_name | table_name | region          |
provisioned_read_capacity_autoscaling_update

provisioned_write_capacity_autoscaling_update
-----+-----+-----+
+-----+-----+-----+
mykeyspace    | mytable    | ap-southeast-1 | {'minimum_units': 5,
'maximum_units': 10, 'scaling_policy':
{'target_tracking_scaling_policy_configuration': {'scale_out_cooldown':
60, 'disable_scale_in': false, 'target_value': 50, 'scale_in_cooldown':
60}}} | {'minimum_units': 5, 'maximum_units': 10, 'scaling_policy':
{'target_tracking_scaling_policy_configuration': {'scale_out_cooldown': 0,
'disable_scale_in': false, 'target_value': 50, 'scale_in_cooldown': 0}}}
mykeyspace    | mytable    | us-east-1      | {'minimum_units': 5,
'maximum_units': 20, 'scaling_policy':
{'target_tracking_scaling_policy_configuration': {'scale_out_cooldown':
60, 'disable_scale_in': false, 'target_value': 70, 'scale_in_cooldown':
60}}} | {'minimum_units': 5, 'maximum_units': 10, 'scaling_policy':
```

```
{'target_tracking_scaling_policy_configuration': {'scale_out_cooldown': 0,
'disable_scale_in': false, 'target_value': 50, 'scale_in_cooldown': 0}}}
mykeyspace | mytable | eu-west-1 | {'minimum_units': 5,
'maximum_units': 10, 'scaling_policy':
{'target_tracking_scaling_policy_configuration': {'scale_out_cooldown':
60, 'disable_scale_in': false, 'target_value': 50, 'scale_in_cooldown':
60}}} | {'minimum_units': 5, 'maximum_units': 10, 'scaling_policy':
{'target_tracking_scaling_policy_configuration': {'scale_out_cooldown': 0,
'disable_scale_in': false, 'target_value': 50, 'scale_in_cooldown': 0}}}
```

CLI

Consultez les paramètres de capacité provisionnée et de dimensionnement automatique d'une table multirégionale à l'aide du AWS CLI

- Pour afficher la configuration de mise à l'échelle automatique d'une table multirégionale, vous pouvez utiliser l'opération `get-table-auto-scaling-settings`. La commande CLI suivante en est un exemple.

```
aws keyspaces get-table-auto-scaling-settings --keyspace-name mykeyspace --
table-name mytable
```

Le résultat suivant doit s'afficher.

```
{
  "keyspaceName": "mykeyspace",
  "tableName": "mytable",
  "resourceArn": "arn:aws:cassandra:us-east-1:777788889999:/keyspace/
mykeyspace/table/mytable",
  "autoScalingSpecification": {
    "writeCapacityAutoScaling": {
      "autoScalingDisabled": false,
      "minimumUnits": 5,
      "maximumUnits": 10,
      "scalingPolicy": {
        "targetTrackingScalingPolicyConfiguration": {
          "disableScaleIn": false,
          "scaleInCooldown": 0,
          "scaleOutCooldown": 0,
          "targetValue": 50.0
        }
      }
    }
  }
}
```

```

    }
  },
  "readCapacityAutoScaling": {
    "autoScalingDisabled": false,
    "minimumUnits": 5,
    "maximumUnits": 20,
    "scalingPolicy": {
      "targetTrackingScalingPolicyConfiguration": {
        "disableScaleIn": false,
        "scaleInCooldown": 60,
        "scaleOutCooldown": 60,
        "targetValue": 70.0
      }
    }
  }
},
"replicaSpecifications": [
  {
    "region": "us-east-1",
    "autoScalingSpecification": {
      "writeCapacityAutoScaling": {
        "autoScalingDisabled": false,
        "minimumUnits": 5,
        "maximumUnits": 10,
        "scalingPolicy": {
          "targetTrackingScalingPolicyConfiguration": {
            "disableScaleIn": false,
            "scaleInCooldown": 0,
            "scaleOutCooldown": 0,
            "targetValue": 50.0
          }
        }
      },
      "readCapacityAutoScaling": {
        "autoScalingDisabled": false,
        "minimumUnits": 5,
        "maximumUnits": 20,
        "scalingPolicy": {
          "targetTrackingScalingPolicyConfiguration": {
            "disableScaleIn": false,
            "scaleInCooldown": 60,
            "scaleOutCooldown": 60,
            "targetValue": 70.0
          }
        }
      }
    }
  }
]

```

```

    }
  }
},
{
  "region": "eu-north-1",
  "autoScalingSpecification": {
    "writeCapacityAutoScaling": {
      "autoScalingDisabled": false,
      "minimumUnits": 5,
      "maximumUnits": 10,
      "scalingPolicy": {
        "targetTrackingScalingPolicyConfiguration": {
          "disableScaleIn": false,
          "scaleInCooldown": 0,
          "scaleOutCooldown": 0,
          "targetValue": 50.0
        }
      }
    },
    "readCapacityAutoScaling": {
      "autoScalingDisabled": false,
      "minimumUnits": 5,
      "maximumUnits": 10,
      "scalingPolicy": {
        "targetTrackingScalingPolicyConfiguration": {
          "disableScaleIn": false,
          "scaleInCooldown": 60,
          "scaleOutCooldown": 60,
          "targetValue": 50.0
        }
      }
    }
  }
}
]
}

```

Désactiver le redimensionnement automatique pour un tableau dans Amazon Keyspaces

Cette section fournit des exemples expliquant comment désactiver le dimensionnement automatique pour une table multirégionale en mode capacité allouée. Vous pouvez le faire sur la console Amazon Keyspaces à l'aide de CQL ou du AWS CLI

Important

Nous recommandons d'utiliser le dimensionnement automatique pour les tables multirégionales qui utilisent le mode capacité provisionnée. Pour de plus amples informations, veuillez consulter [the section called “Estimation de la capacité d'une table multirégionale”](#).

Note

Pour supprimer le rôle lié à un service utilisé par Application Auto Scaling, vous devez désactiver le dimensionnement automatique sur toutes les tables du compte. Régions AWS

Console

Désactiver le dimensionnement automatique d'Amazon Keyspaces pour une table multirégionale existante sur la console

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Choisissez la table avec laquelle vous souhaitez travailler, puis cliquez sur l'onglet Capacité.
3. Dans la section Paramètres de capacité, choisissez Modifier.
4. Pour désactiver le dimensionnement automatique d'Amazon Keyspaces, décochez la case Scale automatically. La désactivation du dimensionnement automatique annule l'enregistrement de la table en tant que cible évolutive avec Application Auto Scaling. Pour supprimer le rôle lié à un service utilisé par Application Auto Scaling pour accéder à votre table Amazon Keyspaces, suivez les étapes décrites dans. [the section called “Supprimer un rôle lié à un service pour Amazon Keyspaces”](#)
5. Lorsque les paramètres de mise à l'échelle automatique sont définis, choisissez Enregistrer.

Cassandra Query Language (CQL)

Désactiver la mise à l'échelle automatique pour une table multirégionale à l'aide de CQL

- Vous pouvez l'utiliser `ALTER TABLE` pour désactiver le redimensionnement automatique pour une table existante. Notez que vous ne pouvez pas désactiver le redimensionnement automatique pour une réplique de table individuelle.

Dans l'exemple suivant, la mise à l'échelle automatique est désactivée pour la capacité de lecture de la table.

```
ALTER TABLE mykeyspace.mytable
WITH AUTOSCALING_SETTINGS = {
    'provisioned_read_capacity_autoscaling_update': {
        'autoscaling_disabled': true
    }
};
```

CLI

Désactivez la mise à l'échelle automatique pour une table multirégionale à l'aide du AWS CLI

- Vous pouvez utiliser cette AWS CLI `update-table` commande pour désactiver la mise à l'échelle automatique pour une table existante. Notez que vous ne pouvez pas désactiver le redimensionnement automatique pour une réplique de table individuelle.

Dans l'exemple suivant, la mise à l'échelle automatique est désactivée pour la capacité de lecture de la table.

```
aws keyspaces update-table --keyspace-name mykeyspace --table-name mytable
    \ --auto-scaling-specification
    readCapacityAutoScaling={autoScalingDisabled=true}
```

Définissez manuellement la capacité allouée d'une table multirégionale dans Amazon Keyspaces

Si vous devez désactiver le dimensionnement automatique pour une table multirégionale, vous pouvez configurer manuellement la capacité de lecture de la table pour une table répliquée à l'aide de CQL ou du AWS CLI

Note

Nous recommandons d'utiliser le dimensionnement automatique pour les tables multirégionales qui utilisent le mode capacité provisionnée. Pour de plus amples informations, veuillez consulter [the section called “Estimation de la capacité d'une table multirégionale”](#).

Cassandra Query Language (CQL)

Configuration manuelle de la capacité provisionnée d'une table multirégionale à l'aide de CQL

- Vous pouvez l'utiliser `ALTER TABLE` pour provisionner manuellement la capacité de lecture d'une table répliquée.

```
ALTER TABLE mykeyspace.mytable
WITH CUSTOM_PROPERTIES = {
    'capacity_mode': {
        'throughput_mode': 'PROVISIONED',
        'read_capacity_units': 1,
        'write_capacity_units': 1
    },
    'replica_updates': {
        'us-east-1': {
            'read_capacity_units': 2
        }
    }
};
```

CLI

Définissez manuellement la capacité allouée d'une table multirégionale à l'aide du AWS CLI

- Si vous devez désactiver le dimensionnement automatique pour une table multirégionale, vous pouvez l'utiliser pour `update-table` provisionner manuellement la capacité de lecture de la table pour une table de réplique.

```
aws keyspaces update-table --keyspace-name mykeyspace --table-name mytable \
--capacity-specification
    throughputMode=PROVISIONED,readCapacityUnits=1,writeCapacityUnits=1 \
--replica-specifications region="us-east-1",readCapacityUnits=5
```

Backup et restauration des données avec restauration instantanée pour Amazon Keyspaces

Point-in-time La restauration (PITR) permet de protéger vos tables Amazon Keyspaces contre les opérations d'écriture ou de suppression accidentelles en vous fournissant des sauvegardes continues des données de vos tables.

Supposons, par exemple, qu'un script de test écrit accidentellement dans une table Amazon Keyspaces de production. Grâce à la restauration instantanée, vous pouvez restaurer les données de cette table à tout moment depuis que le PITR a été activé au cours des 35 derniers jours. Si vous supprimez une table pour laquelle la restauration instantanée est activée, vous pouvez rechercher les données de la table supprimée pendant 35 jours (sans frais supplémentaires) et les restaurer dans l'état dans lequel elles se trouvaient juste avant le point de suppression.

Vous pouvez restaurer une table Amazon Keyspaces à un moment précis en utilisant la console, le AWS SDK et le () ou Cassandra Query Language AWS Command Line Interface (CQL AWS CLI). Pour de plus amples informations, veuillez consulter [Utilisez la restauration instantanée dans Amazon Keyspaces](#).

Point-in-time les opérations n'ont aucun impact sur les performances ou la disponibilité de la table de base, et la restauration d'une table ne consomme pas de débit supplémentaire.

Pour plus d'informations sur les quotas PITR, consultez [Quotas](#).

Pour plus d'informations sur les tarifs, consultez les tarifs d'[Amazon Keyspaces \(pour Apache Cassandra\)](#).

Rubriques

- [Comment fonctionne la restauration instantanée dans Amazon Keyspaces](#)
- [Utilisez la restauration instantanée dans Amazon Keyspaces](#)

Comment fonctionne la restauration instantanée dans Amazon Keyspaces

Cette section fournit un aperçu du fonctionnement de la restauration instantanée (PITR) d'Amazon Keyspaces. Pour plus d'informations sur les tarifs, consultez les tarifs [d'Amazon Keyspaces \(pour Apache Cassandra\)](#).

Rubriques

- [Fenêtre temporelle pour les sauvegardes continues PITR](#)
- [Paramètres de restauration PITR](#)
- [Restauration PITR de tables chiffrées](#)
- [Restauration PITR de tables multirégionales](#)
- [Restauration PITR de tables avec des types définis par l'utilisateur \(UDT\)](#)
- [Durée de restauration des tables avec PITR](#)
- [Amazon Keyspaces PITR et intégration aux services AWS](#)

Fenêtre temporelle pour les sauvegardes continues PITR

Amazon Keyspaces PITR utilise deux horodatages pour maintenir la période pendant laquelle les sauvegardes restaurables sont disponibles pour une table.

- **Heure de restauration la plus ancienne** : indique l'heure de la première sauvegarde restaurable. La première sauvegarde restaurable remonte à 35 jours ou à la date d'activation du PITR, selon la date la plus récente. La période de sauvegarde maximale de 35 jours ne peut pas être modifiée.
- **Heure actuelle** — L'horodatage de la dernière sauvegarde restaurable est l'heure actuelle. Si aucun horodatage n'est fourni lors d'une restauration, l'heure actuelle est utilisée.

Lorsque le PITR est activé, vous pouvez effectuer une restauration à n'importe quel moment entre `EarliestRestorableDateTime` et `CurrentTime`. Vous ne pouvez restaurer les données d'une table qu'à une date où le PITR était activé.

Si vous désactivez le PITR puis le réactivez ultérieurement, vous redéfinissez l'heure de début de la première sauvegarde disponible à la date de réactivation du PITR. Cela signifie que la désactivation du PITR efface l'historique de vos sauvegardes.

Note

Les opérations du langage de définition des données (DDL) sur les tables, telles que les modifications de schéma, sont effectuées de manière asynchrone. Vous ne pouvez voir que les opérations terminées dans les données de votre table restaurée, mais vous pouvez voir des actions supplémentaires sur votre table source si elles étaient en cours au moment de la restauration. Pour obtenir la liste des instructions DDL, consultez [the section called “Instructions DDL”](#).

Il n'est pas nécessaire qu'une table soit active pour être restaurée. Vous pouvez également restaurer les tables supprimées si le PITR a été activé sur la table supprimée et si la suppression a eu lieu dans la fenêtre de sauvegarde (ou au cours des 35 derniers jours).

Note

Si une nouvelle table est créée avec le même nom qualifié (par exemple, `mykeyspace.mytable`) qu'une table précédemment supprimée, la table supprimée ne pourra plus être restaurée. Si vous essayez de le faire depuis la console, un avertissement s'affiche.

Paramètres de restauration PITR

Lorsque vous restaurez une table à l'aide de PITR, Amazon Keyspaces restaure le schéma et les données de votre table source dans l'état basé sur l'horodatage `day:hour:minute:second ()` sélectionné pour une nouvelle table. PITR ne remplace pas les tables existantes.

Outre le schéma et les données de la table, PITR les restaure `custom_properties` à partir de la table source. Contrairement aux données de la table, qui sont restaurées en fonction de l'horodatage sélectionné entre l'heure de restauration la plus ancienne et l'heure actuelle, les propriétés personnalisées sont toujours restaurées en fonction des paramètres de la table à l'heure actuelle.

Les paramètres de la table restaurée correspondent aux paramètres de la table source avec l'horodatage du lancement de la restauration. Si vous souhaitez remplacer ces paramètres lors

de la restauration, vous pouvez le faire à l'aide `WITH custom_properties` de. Les propriétés personnalisées incluent les paramètres suivants.

- Read/write mode capacité
- Paramètres de capacité de débit provisionnée
- Réglages PITR

Si la table est en mode capacité allouée avec le dimensionnement automatique activé, l'opération de restauration rétablit également les paramètres de dimensionnement automatique de la table. Vous pouvez les remplacer à l'aide du `autoscaling_settings` paramètre dans CQL ou à l'`autoScalingSpecification` aide de la CLI. Pour plus d'informations sur les paramètres de mise à l'échelle automatique, consultez [the section called “Gérez la capacité de débit avec la mise à l'échelle automatique”](#).

Lorsque vous effectuez une restauration complète de la table, tous les paramètres de la table restaurée proviennent des paramètres actuels de la table source au moment de la restauration.

Supposons par exemple que le débit alloué d'une table vienne d'être abaissé à 50 unités de capacité de lecture et 50 unités de capacité d'écriture. Vous rétablissez ensuite l'état de la table à il y a trois semaines. À cette époque, son débit provisionné était fixé à 100 unités de capacité de lecture et à 100 unités de capacité d'écriture. Dans ce cas, Amazon Keyspaces restaure les données de votre table à ce moment-là, mais utilise les paramètres de débit actuels (50 unités de capacité de lecture et 50 unités de capacité d'écriture).

Les paramètres suivants ne sont pas restaurés et vous devez les configurer manuellement pour la nouvelle table.

- Flux de capture des données de modification (CDC) d'Amazon Keyspaces
- Gestion des identités et des accès AWS politiques (IAM)
- CloudWatch Mesures et alarmes Amazon
- Tags (peuvent être ajoutés à l'`RESTORE` instruction CQL en utilisant `WITH TAGS`)

Restauration PITR de tables chiffrées

Lorsque vous restaurez une table à l'aide du PITR, Amazon Keyspaces restaure les paramètres de chiffrement de votre table source. Si la table a été chiffrée avec un Clé détenue par AWS (par défaut), elle est automatiquement restaurée avec le même paramètre. Si la table que vous souhaitez

restaurer a été chiffrée à l'aide d'une clé gérée par le client, cette même clé gérée par le client doit être accessible à Amazon Keyspaces pour restaurer les données de la table.

Vous pouvez modifier les paramètres de chiffrement de la table au moment de la restauration. Pour passer d'une clé gérée par le client Clé détenue par AWS à une clé gérée par le client, vous devez fournir une clé gérée par le client valide et accessible au moment de la restauration.

Si vous souhaitez passer d'une clé gérée par le client à une Clé détenue par AWS, vérifiez qu'Amazon Keyspaces a accès à la clé gérée par le client de la table source pour restaurer la table avec un. Clé détenue par AWS Pour plus d'informations sur les paramètres de chiffrement au repos des tables, consultez [the section called “Comment ça marche”](#).

Note

Si le tableau a été supprimé parce qu'Amazon Keyspaces a perdu l'accès à votre clé gérée par le client, vous devez vous assurer que la clé gérée par le client est accessible à Amazon Keyspaces avant d'essayer de restaurer le tableau. Une table chiffrée à l'aide d'une clé gérée par le client ne peut pas être restaurée si Amazon Keyspaces n'a pas accès à cette clé. Pour plus d'informations, consultez la section [Résolution des problèmes d'accès aux clés](#) dans le Guide du AWS Key Management Service développeur.

Restauration PITR de tables multirégionales

Vous pouvez restaurer une table multirégionale à l'aide de PITR. Pour que l'opération de restauration soit réussie, le PITR doit être activé sur toutes les répliques de la table source et la table source et la table de destination doivent être répliquées sur la même table. Régions AWS

Amazon Keyspaces rétablit les paramètres de la table source dans chacune des régions répliquées faisant partie du keyspace. Vous pouvez également modifier les paramètres lors de l'opération de restauration. Pour plus d'informations sur les paramètres qui peuvent être modifiés au cours de la restauration, consultez [the section called “Restaurer les paramètres”](#).

Pour plus d'informations sur la réplication multirégionale, consultez [the section called “Comment ça marche”](#).

Restauration PITR de tables avec des types définis par l'utilisateur (UDT)

Vous pouvez restaurer une table qui utilise des UDT. Pour que l'opération de restauration réussisse, les UDT référencés doivent exister et être valides dans le keyspace.

Si un UDT requis est absent lorsque vous tentez de restaurer une table, Amazon Keyspaces essaie de restaurer automatiquement le schéma UDT, puis continue de restaurer la table.

Si vous avez supprimé et recréé l'UDT, Amazon Keyspaces restaure l'UDT avec le nouveau schéma de l'UDT et rejette la demande de restauration de la table à l'aide du schéma UDT d'origine. Dans ce cas, si vous souhaitez restaurer la table avec l'ancien schéma UDT, vous pouvez restaurer la table dans un nouvel espace de touches. Lorsque vous supprimez et créez un UDT, même si le schéma de l'UDT recréé est identique à celui de l'UDT supprimé, l'UDT recréé est considéré comme un nouvel UDT. Dans ce cas, Amazon Keyspaces rejette la demande de restauration de la table avec l'ancien schéma UDT.

Si l'UDT est absent et qu'Amazon Keyspaces tente de le restaurer, la tentative échoue si vous avez atteint le nombre maximum d'UDT pour le compte dans la région.

Pour plus d'informations sur les quotas UDT et les valeurs par défaut, consultez [the section called “Quotas et valeurs par défaut pour les types définis par l'utilisateur \(UDT\) dans Amazon Keyspaces”](#). Pour plus d'informations sur l'utilisation des UDT, consultez [the section called “User-defined types \(UDT\)”](#).

Durée de restauration des tables avec PITR

Le temps nécessaire pour restaurer une table dépend de plusieurs facteurs et n'est pas toujours directement corrélé à la taille de la table.

Voici quelques points à prendre en compte concernant les délais de restauration.

- Vous restaurez vos sauvegardes vers une nouvelle table. Même si la table est vide, la réalisation de toutes les actions nécessaires à la création de la nouvelle table et au lancement du processus de restauration peut prendre jusqu'à 20 minutes.
- Les temps de restauration des grandes tables dotées de modèles de données bien distribués peuvent être de plusieurs heures, voire plus.
- Si votre table source contient des données très asymétriques, le délai de restauration peut augmenter. Par exemple, si la clé primaire de votre table utilise le mois de l'année comme clé de partition et que toutes vos données datent du mois de décembre, vous avez des données asymétriques.

Une bonne pratique de planification de la reprise après sinistre consiste à documenter régulièrement les temps moyens de restauration et à déterminer comment ces délais affectent votre objectif global de temps de récupération.

Amazon Keyspaces PITR et intégration aux services AWS

Les opérations PITR suivantes sont enregistrées AWS CloudTrail pour permettre une surveillance et un audit continu.

- Créez une nouvelle table avec PITR activé ou désactivé.
- Activez ou désactivez le PITR sur une table existante.
- Restaurez une table active ou supprimée.

Pour de plus amples informations, veuillez consulter [Journalisation des appels d'API Amazon Keyspaces avec AWS CloudTrail](#).

Vous pouvez effectuer les actions PITR suivantes à l'aide CloudFormation de.

- Créez une nouvelle table avec PITR activé ou désactivé.
- Activez ou désactivez le PITR sur une table existante.

Pour de plus amples informations, veuillez consulter le document [Référence du type de ressource Cassandra](#) dans le [Guide de l'utilisateur CloudFormation](#).

Utilisez la restauration instantanée dans Amazon Keyspaces

Avec Amazon Keyspaces (pour Apache Cassandra), vous pouvez restaurer des tables à un moment précis à l'aide de Point-in-Time Restore (PITR). Le PITR vous permet de restaurer un état antérieur d'une table au cours des 35 derniers jours, en fournissant des fonctionnalités de protection et de restauration des données. Cette fonctionnalité est utile dans des cas tels que la suppression accidentelle de données, les erreurs d'application ou à des fins de test. Vous pouvez récupérer les données rapidement et efficacement, en minimisant les temps d'arrêt et les pertes de données. Les sections suivantes vous guident tout au long du processus de restauration des tables à l'aide du PITR dans Amazon Keyspaces, afin de garantir l'intégrité des données et la continuité des activités.

Rubriques

- [Configurer les autorisations IAM de la table de restauration pour Amazon Keyspaces PITR](#)
- [Configurer le PITR pour une table dans Amazon Keyspaces](#)
- [Désactiver le PITR pour une table Amazon Keyspaces](#)
- [Restaurer une table à partir d'une sauvegarde à un moment précis dans Amazon Keyspaces](#)
- [Restaurer une table supprimée à l'aide d'Amazon Keyspaces PITR](#)

Configurer les autorisations IAM de la table de restauration pour Amazon Keyspaces PITR

Cette section explique comment configurer les autorisations permettant à un principal Gestion des identités et des accès AWS (IAM) de restaurer les tables Amazon Keyspaces. Dans IAM, la politique AWS gérée `AmazonKeyspacesFullAccess` inclut les autorisations de restauration des tables Amazon Keyspaces. Pour mettre en œuvre une politique personnalisée avec les autorisations minimales requises, tenez compte des exigences décrites dans la section suivante.

Pour restaurer correctement une table, le principal IAM a besoin des autorisations minimales suivantes :

- `cassandra:Restore`— L'action de restauration est requise pour que la table cible soit restaurée.
- `cassandra:Select`— L'action de sélection est requise pour lire dans la table source.
- `cassandra:TagResource`— L'action de balise est facultative et n'est requise que si l'opération de restauration ajoute des balises.

Il s'agit d'un exemple de politique qui accorde les autorisations minimales requises à un utilisateur pour restaurer des tables dans `Keyspacemykeyspace`.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cassandra:Restore",
        "cassandra:Select"
      ],
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/*",
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
      ]
    }
  ]
}
```

Des autorisations supplémentaires pour restaurer une table peuvent être requises en fonction d'autres fonctionnalités sélectionnées. Par exemple, si la table source est chiffrée au repos à

l'aide d'une clé gérée par le client, Amazon Keyspaces doit être autorisé à accéder à la clé gérée par le client de la table source pour pouvoir restaurer correctement la table. Pour de plus amples informations, veuillez consulter [the section called “PITR et tables cryptées”](#).

Si vous utilisez des politiques IAM avec des [clés de condition](#) pour restreindre le trafic entrant vers des sources spécifiques, vous devez vous assurer qu'Amazon Keyspaces est autorisé à effectuer une opération de restauration pour le compte de votre principal. Vous devez ajouter une clé de `aws:ViaAWSService` condition à votre politique IAM si celle-ci limite le trafic entrant à l'une des conditions suivantes :

- Points de terminaison VPC avec `aws:SourceVpce`
- plages d'adresses IP avec `aws:SourceIp`
- VPC avec `aws:SourceVpc`

La clé de `aws:ViaAWSService` condition permet l'accès lorsqu'un AWS service fait une demande en utilisant les informations d'identification du principal. Pour plus d'informations, voir [Éléments de politique IAM JSON : clé de condition](#) dans le guide de l'utilisateur IAM.

Voici un exemple de politique qui limite le trafic source à une adresse IP spécifique et permet à Amazon Keyspaces de restaurer une table pour le compte du principal.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CassandraAccessForCustomIp",
      "Effect": "Allow",
      "Action": "cassandra:*",
      "Resource": "*",
      "Condition": {
        "Bool": {
          "aws:ViaAWSService": "false"
        },
        "ForAnyValue:IpAddress": {
          "aws:SourceIp": [
            "123.45.167.89"
          ]
        }
      }
    }
  ],
}
```



```
{
  "Sid": "CassandraAccessForAwsService",
  "Effect": "Allow",
  "Action": "cassandra:*",
  "Resource": "*",
  "Condition": {
    "Bool": {
      "aws:ViaAWSService": "true"
    }
  }
}
```

Pour un exemple de politique utilisant la clé de condition `aws:ViaAWSService` globale, voir [the section called “Politiques relatives aux terminaux VPC et restauration instantanée \(PITR\) d'Amazon Keyspaces”](#).

Configurer le PITR pour une table dans Amazon Keyspaces

Vous pouvez configurer une table dans Amazon Keyspaces pour les opérations de sauvegarde et de restauration à l'aide de PITR avec la console, le CQL et le AWS CLI

Lorsque vous créez une nouvelle table à l'aide de CQL ou du AWS CLI, vous devez activer explicitement le PITR dans l'instruction `create table`. Lorsque vous créez une nouvelle table à l'aide de la console, PITR est activé par défaut.

Pour savoir comment restaurer une table, consultez [the section called “Restaurer une table à un point dans le temps”](#).

Console

Configurer le PITR pour une table à l'aide de la console

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le volet de navigation, choisissez Tables et sélectionnez la table que vous souhaitez modifier.
3. Dans l'onglet Sauvegardes, choisissez Modifier.
4. Dans la section Modifier les paramètres de restauration ponctuels, sélectionnez Activer la restauration. Point-in-time

5. Sélectionnez Enregistrer les modifications.

Cassandra Query Language (CQL)

Configurer le PITR pour une table à l'aide de CQL

1. Vous pouvez gérer les paramètres PITR des tables à l'aide de la propriété `point_in_time_recovery` personnalisée.

Pour activer le PITR lorsque vous créez une nouvelle table, vous devez définir le statut de `point_in_time_recovery` à `enabled`. Vous pouvez utiliser la commande CQL suivante à titre d'exemple.

```
CREATE TABLE "my_keyspace1"."my_table1"(  
  "id" int,  
  "name" ascii,  
  "date" timestamp,  
  PRIMARY KEY("id"))  
WITH CUSTOM_PROPERTIES = {  
  'capacity_mode':{'throughput_mode':'PAY_PER_REQUEST'},  
  'point_in_time_recovery':{'status':'enabled'}  
}
```

Note

Si aucune propriété personnalisée de restauration instantanée n'est spécifiée, la restauration instantanée est désactivée par défaut.

2. Pour activer le PITR pour une table existante à l'aide de CQL, exécutez la commande CQL suivante.

```
ALTER TABLE mykeyspace.mytable  
WITH custom_properties = {'point_in_time_recovery': {'status': 'enabled'}}
```

CLI

Configurez le PITR pour une table à l'aide du AWS CLI

1. Vous pouvez gérer les paramètres PITR des tables à l'aide de l'`UpdateTableAPI`.

Pour activer le PITR lorsque vous créez une nouvelle table, vous devez l'inclure `point-in-time-recovery 'status=ENABLED'` dans la commande `create table`. Vous pouvez utiliser la AWS CLI commande suivante à titre d'exemple. La commande a été divisée en lignes distinctes pour améliorer la lisibilité.

```
aws keyspaces create-table --keyspace-name 'myKeyspace' --table-name 'myTable'
                           --schema-definition 'allColumns=[{name=id,type=int},
{name=name,type=text},{name=date,type=timestamp}],partitionKeys=[{name=id}]'
                           --point-in-time-recovery 'status=ENABLED'
```

 Note

Si aucune valeur de restauration instantanée n'est spécifiée, la restauration instantanée est désactivée par défaut.

2. Pour confirmer le paramètre de restauration instantanée d'une table, vous pouvez utiliser la commande suivante. AWS CLI

```
aws keyspaces get-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

3. Pour activer le PITR pour une table existante à l'aide de AWS CLI, exécutez la commande suivante.

```
aws keyspaces update-table --keyspace-name 'myKeyspace' --table-name 'myTable'
                           --point-in-time-recovery 'status=ENABLED'
```

Désactiver le PITR pour une table Amazon Keyspaces

Vous pouvez désactiver le PITR pour une table Amazon Keyspaces à tout moment à l'aide de la console, du CQL ou du. AWS CLI

 Important

La désactivation du PITR supprime immédiatement l'historique de vos sauvegardes, même si vous réactivez le PITR sur la table dans les 35 jours.

Pour savoir comment restaurer une table, consultez [the section called “Restaurer une table à un point dans le temps”](#).

Console

Désactiver le PITR pour une table à l'aide de la console

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le volet de navigation, choisissez Tables et sélectionnez la table que vous souhaitez modifier.
3. Dans l'onglet Sauvegardes, choisissez Modifier.
4. Dans la section Modifier les paramètres de restauration ponctuels, décochez la case Activer la Point-in-time restauration.
5. Sélectionnez Enregistrer les modifications.

Cassandra Query Language (CQL)

Désactiver le PITR pour une table à l'aide de CQL

- Pour désactiver le PITR pour une table existante, exécutez la commande CQL suivante.

```
ALTER TABLE mykeyspace.mytable  
WITH custom_properties = {'point_in_time_recovery': {'status': 'disabled'}}
```

CLI

Désactivez le PITR pour une table à l'aide du AWS CLI

- Pour désactiver le PITR pour une table existante, exécutez la AWS CLI commande suivante.

```
aws keyspaces update-table --keyspace-name 'myKeyspace' --table-name 'myTable'  
--point-in-time-recovery 'status=DISABLED'
```

Restaurer une table à partir d'une sauvegarde à un moment précis dans Amazon Keyspaces

La section suivante explique comment restaurer une table Amazon Keyspaces existante à un moment précis.

Note

Cette procédure suppose que la table que vous utilisez a été configurée avec une restauration instantanée. Pour activer le PITR pour une table, consultez [the section called “Configurer PITR”](#).

Important

Pendant qu'une restauration est en cours, ne modifiez ni ne supprimez les politiques Gestion des identités et des accès AWS (IAM) qui accordent au principal IAM (par exemple, utilisateur, groupe ou rôle) l'autorisation d'effectuer la restauration. Sinon, il peut en résulter un comportement inattendu. Par exemple, si vous supprimez les autorisations d'écriture pour une table alors que celle-ci est en cours de restauration, l'`RestoreTableToPointInTime` opération sous-jacente ne peut écrire aucune des données restaurées dans la table.

Vous ne pouvez modifier ou supprimer des autorisations qu'une fois l'opération de restauration terminée.

Console

Restaurer une table à un moment précis à l'aide de la console

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le volet de navigation sur le côté gauche de la console, choisissez Tables.
3. Dans la liste des tables, choisissez la table que vous souhaitez restaurer.
4. Dans l'onglet Sauvegardes du tableau, dans la section Point-in-time de restauration, choisissez Restaurer.

5. Pour le nouveau nom de table, entrez un nouveau nom pour la table restaurée, par exemple **mytable_restored**.
6. Pour définir le moment de l'opération de restauration, vous pouvez choisir entre deux options :
 - Sélectionnez l'heure la plus ancienne préconfigurée.
 - Sélectionnez Spécifier la date et l'heure, puis entrez la date et l'heure auxquelles vous souhaitez restaurer la nouvelle table.

 Note

Vous pouvez effectuer une restauration à n'importe quel moment dans le plus bref délai et à l'heure actuelle. Amazon Keyspaces restaure les données de votre table dans leur état en fonction de la date et de l'heure sélectionnées (day:hour:minute:second).

7. Choisissez Restaurer pour démarrer le processus de restauration.

La table en cours de restauration s'affiche avec le statut Restauration en cours. Une fois le processus de restauration terminé, le statut de la table restaurée passe à Actif.

Cassandra Query Language (CQL)

Restaurer une table à un point dans le temps à l'aide de CQL

1. Vous pouvez restaurer une table active à un point dans le temps compris entre `earliest_restorable_timestamp` et l'heure actuelle. L'heure actuelle est la valeur par défaut.

Pour confirmer que la restauration instantanée est activée pour la table, interrogez le `system_schema_mcs.tables` comme indiqué dans cet exemple.

```
SELECT custom_properties
FROM system_schema_mcs.tables
WHERE keyspace_name = 'mykeyspace' AND table_name = 'mytable';
```

Point-in-time la restauration est activée comme indiqué dans l'exemple de sortie suivant.

```
custom_properties
```

```

-----
{
  ...,
  "point_in_time_recovery": {
    "earliest_restorable_timestamp": "2020-06-30T19:19:21.175Z"
    "status": "enabled"
  }
}

```

2. • Rétablissez le tableau à l'heure actuelle. Lorsque vous omettez la `WITH restore_timestamp = ...` clause, l'horodatage actuel est utilisé.

```

RESTORE TABLE mykeyspace.mytable_restored
FROM TABLE mykeyspace.mytable;

```

- Vous pouvez également effectuer une restauration à un moment précis, défini par un `restore_timestamp` au format ISO 8601. Vous pouvez spécifier n'importe quel instant dans le passé au cours des 35 derniers jours. Par exemple, la commande suivante restaure la table à la `EarliestRestorableDateTime`.

```

RESTORE TABLE mykeyspace.mytable_restored
FROM TABLE mykeyspace.mytable
WITH restore_timestamp = '2020-06-30T19:19:21.175Z';

```

Pour une description complète de la syntaxe, voir [the section called “RESTAURER LA TABLE”](#) la référence du langage.

3. Pour vérifier que la restauration de la table a réussi, interrogez le `system_schema_mcs.tables` pour confirmer l'état de la table.

```

SELECT status
FROM system_schema_mcs.tables
WHERE keyspace_name = 'mykeyspace' AND table_name = 'mytable_restored'

```

La requête affiche le résultat suivant.

```

status
-----
RESTORING

```

La table en cours de restauration s'affiche avec le statut **Restauration en cours**. Une fois le processus de restauration terminé, le statut de la table passe à **Actif**.

CLI

Restaurez un tableau à un point dans le temps à l'aide du AWS CLI

1. Créez une table simple nommée sur `myTable` laquelle le PITR est activé. La commande a été divisée en lignes distinctes pour des raisons de lisibilité.

```
aws keyspaces create-table --keyspace-name 'myKeyspace' --table-name 'myTable'
                           --schema-definition 'allColumns=[{name=id,type=int},
{name=name,type=text},{name=date,type=timestamp}],partitionKeys=[{name=id}]'
                           --point-in-time-recovery 'status=ENABLED'
```

2. Vérifiez les propriétés de la nouvelle table et vérifiez le `earliestRestorableTimestamp` PITR.

```
aws keyspaces get-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

La sortie de cette commande renvoie ce qui suit.

```
{
  "keyspaceName": "myKeyspace",
  "tableName": "myTable",
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
myKeyspace/table/myTable",
  "creationTimestamp": "2022-06-20T14:34:57.049000-07:00",
  "status": "ACTIVE",
  "schemaDefinition": {
    "allColumns": [
      {
        "name": "id",
        "type": "int"
      },
      {
        "name": "date",
        "type": "timestamp"
      },
      {
```



```

        "name": "name",
        "type": "text"
    }
],
"partitionKeys": [
    {
        "name": "id"
    }
],
"clusteringKeys": [],
"staticColumns": []
},
"capacitySpecification": {
    "throughputMode": "PAY_PER_REQUEST",
    "lastUpdateToPayPerRequestTimestamp": "2022-06-20T14:34:57.049000-07:00"
},
"encryptionSpecification": {
    "type": "AWS_OWNED_KMS_KEY"
},
"pointInTimeRecovery": {
    "status": "ENABLED",
    "earliestRestorableTimestamp": "2022-06-20T14:35:13.693000-07:00"
},
"defaultTimeToLive": 0,
"comment": {
    "message": ""
}
}

```

3. • Pour restaurer un tableau à un point dans le temps, spécifiez un `restore_timestamp` au format ISO 8601. Vous pouvez choisir n'importe quel moment au cours des 35 derniers jours à intervalles d'une seconde. Par exemple, la commande suivante restaure la table à la `EarliestRestorableDateTime`.

```
aws keyspaces restore-table --source-keyspace-name 'myKeyspace' --source-table-name 'myTable' --target-keyspace-name 'myKeyspace' --target-table-name 'myTable_restored' --restore-timestamp "2022-06-20 21:35:14.693"
```

La sortie de cette commande renvoie l'ARN de la table restaurée.

```
{
```

```
"restoredTableARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
myKeyspace/table/myTable_restored"
}
```

- Pour rétablir l'heure actuelle de la table, vous pouvez omettre le `restore-timestamp` paramètre.

```
aws keyspaces restore-table --source-keyspace-name 'myKeyspace' --source-
table-name 'myTable' --target-keyspace-name 'myKeyspace' --target-table-name
'myTable_restored1'"
```

Restaurer une table supprimée à l'aide d'Amazon Keyspaces PITR

La procédure suivante indique comment restaurer une table supprimée depuis la sauvegarde jusqu'au moment de sa suppression. Vous pouvez le faire à l'aide de CQL ou du AWS CLI.

Note

Cette procédure suppose que le PITR a été activé sur la table supprimée.

Cassandra Query Language (CQL)

Restaurer une table supprimée à l'aide de CQL

1. Pour confirmer que la restauration instantanée est activée pour une table supprimée, interrogez la table système. Seules les tables pour lesquelles la restauration instantanée est activée sont affichées.

```
SELECT custom_properties
FROM system_schema_mcs.tables_history
WHERE keyspace_name = 'mykeyspace' AND table_name = 'my_table';
```

La requête affiche le résultat suivant.

```
custom_properties
-----
{
    ... ,
```

```
"point_in_time_recovery":{
  "restorable_until_time":"2020-08-04T00:48:58.381Z",
  "status":"enabled"
}
```

2. Restaurez le tableau au moment de sa suppression à l'aide de l'exemple d'instruction suivant.

```
RESTORE TABLE mykeyspace.mytable_restored
FROM TABLE mykeyspace.mytable;
```

CLI

Restaurez une table supprimée à l'aide du AWS CLI

1. Supprimez une table que vous avez créée précédemment et sur laquelle le PITR est activé. Voici un exemple de commande.

```
aws keyspaces delete-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

2. Restaurez la table supprimée au moment de sa suppression à l'aide de la commande suivante.

```
aws keyspaces restore-table --source-keyspace-name 'myKeyspace' --source-
table-name 'myTable' --target-keyspace-name 'myKeyspace' --target-table-name
'myTable_restored2'
```

La sortie de cette commande renvoie l'ARN de la table restaurée.

```
{
  "restoredTableARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
myKeyspace/table/myTable_restored2"
}
```

Expirer les données avec Time to Live (TTL) pour Amazon Keyspaces (pour Apache Cassandra)

Amazon Keyspaces (pour Apache Cassandra) Time to Live (TTL) vous aide à simplifier la logique de votre application et à optimiser le prix du stockage en expirant automatiquement les données des tables. Les données dont vous n'avez plus besoin sont automatiquement supprimées de votre tableau en fonction de la valeur Time to Live que vous avez définie.

Cela facilite le respect des politiques de conservation des données basées sur les exigences commerciales, sectorielles ou réglementaires qui définissent la durée de conservation des données ou spécifient le moment où les données doivent être supprimées.

Par exemple, vous pouvez utiliser le TTL dans une AdTech application pour planifier le moment où les données relatives à des annonces spécifiques expirent et ne sont plus visibles pour les clients. Vous pouvez également utiliser le TTL pour supprimer automatiquement les anciennes données et économiser sur vos coûts de stockage.

Vous pouvez définir une valeur TTL par défaut pour l'ensemble du tableau et remplacer cette valeur pour les lignes et les colonnes individuelles. Les opérations TTL n'ont aucun impact sur les performances de votre application. De plus, le nombre de lignes et de colonnes marquées comme expirant avec le TTL n'affecte pas la disponibilité de votre table.

Amazon Keyspaces filtre automatiquement les données expirées afin qu'elles ne soient pas renvoyées dans les résultats des requêtes ou ne puissent pas être utilisées dans les instructions du langage de manipulation des données (DML). Amazon Keyspaces supprime généralement les données expirées du stockage dans les 10 jours suivant la date d'expiration.

Dans de rares cas, Amazon Keyspaces peut ne pas être en mesure de supprimer les données dans les 10 jours en cas d'activité prolongée sur la partition de stockage sous-jacente afin de protéger la disponibilité. Dans ces cas, Amazon Keyspaces continue de tenter de supprimer les données expirées une fois que le trafic sur la partition diminue.

Une fois les données définitivement supprimées du stockage, vous ne devez plus payer de frais de stockage.

Vous pouvez définir, modifier ou désactiver les paramètres TTL par défaut pour les tables nouvelles et existantes à l'aide de la console, du langage de requête Cassandra (CQL) ou du AWS CLI

Sur les tables où le TTL par défaut est configuré, vous pouvez utiliser des instructions CQL pour remplacer les paramètres TTL par défaut de la table et appliquer des valeurs TTL personnalisées aux

lignes et aux colonnes. Pour plus d'informations, consultez [the section called “INSERTÀ utiliser pour définir un TTL personnalisé pour les nouvelles lignes”](#) et [the section called “UPDATEÀ utiliser pour définir un TTL personnalisé pour les lignes et les colonnes”](#).

La tarification TTL est basée sur la taille des lignes supprimées ou mises à jour à l'aide de Time to Live. Les opérations TTL sont mesurées en unités de. TTL deLetes Une suppression TTL est consommée par Ko de données par ligne supprimée ou mise à jour.

Par exemple, pour mettre à jour une ligne qui stocke 2,5 Ko de données et pour supprimer une ou plusieurs colonnes de la ligne en même temps, trois suppressions TTL sont nécessaires. Ou bien, pour supprimer une ligne entière contenant 3,5 Ko de données, quatre suppressions TTL sont nécessaires.

Une suppression TTL est consommée par Ko de données supprimées par ligne. Pour plus d'informations sur les tarifs, consultez les tarifs [d'Amazon Keyspaces \(pour Apache Cassandra\)](#).

Rubriques

- [Amazon Keyspaces Time to Live et intégration aux services AWS](#)
- [Création d'une nouvelle table avec les paramètres Time to Live \(TTL\) par défaut](#)
- [Mettre à jour la valeur TTL \(Time to Live\) par défaut d'une table](#)
- [Créer un tableau avec les paramètres personnalisés de durée de vie \(TTL\) activés](#)
- [Mettre à jour le tableau avec Time to Live \(TTL\) personnalisé](#)
- [Utilisez l'INSERTinstruction pour définir des valeurs de durée de vie \(TTL\) personnalisées pour les nouvelles lignes](#)
- [Utilisez l'UPDATEinstruction pour modifier les paramètres personnalisés de durée de vie \(TTL\) pour les lignes et les colonnes](#)

Amazon Keyspaces Time to Live et intégration aux services AWS

La métrique TTL suivante est disponible sur Amazon CloudWatch pour permettre une surveillance continue.

- **TTLDeLetes**— Les unités consommées pour supprimer ou mettre à jour des données d'affilée à l'aide du Time to Live (TTL).

Pour plus d'informations sur la façon de surveiller CloudWatch les métriques, consultez [the section called “Surveillance avec CloudWatch”](#).

Lorsque vous l'utilisez CloudFormation, vous pouvez activer le TTL lors de la création d'une table Amazon Keyspaces. Pour plus d'informations, consultez le [Guide de l'utilisateur AWS CloudFormation](#).

Création d'une nouvelle table avec les paramètres Time to Live (TTL) par défaut

Dans Amazon Keyspaces, vous pouvez définir une valeur TTL par défaut pour toutes les lignes d'un tableau lorsque celui-ci est créé.

La valeur TTL par défaut d'une table est zéro, ce qui signifie que les données n'expirent pas automatiquement. Si la valeur TTL par défaut d'une table est supérieure à zéro, un horodatage d'expiration est ajouté à chaque ligne.

Les valeurs TTL sont définies en secondes, et la valeur maximale configurable est de 630 720 000 secondes, soit l'équivalent de 20 ans.

Après la création de la table, vous pouvez remplacer le paramètre TTL par défaut de la table pour des lignes ou des colonnes spécifiques par des instructions CQL DML. Pour plus d'informations, consultez [the section called “INSERTÀ utiliser pour définir un TTL personnalisé pour les nouvelles lignes”](#) et [the section called “UPDATEÀ utiliser pour définir un TTL personnalisé pour les lignes et les colonnes”](#).

Lorsque vous activez le TTL sur une table, Amazon Keyspaces commence à stocker des métadonnées TTL-related supplémentaires pour chaque ligne. En outre, le TTL utilise des horodatages d'expiration pour suivre l'expiration des lignes ou des colonnes. Les horodatages sont stockés sous forme de métadonnées de ligne et contribuent au coût de stockage de la ligne.

Une fois la fonctionnalité TTL activée, vous ne pouvez pas la désactiver pour une table. Le fait de `default_time_to_live` définir le tableau sur 0 désactive les délais d'expiration par défaut pour les nouvelles données, mais cela ne désactive pas la fonctionnalité TTL et ne rétablit pas les métadonnées de stockage ou le comportement d'écriture d'origine du tableau Amazon Keyspaces.

Les exemples suivants montrent comment créer une nouvelle table avec une valeur TTL par défaut.

Console

Créez une nouvelle table avec une valeur Time to Live par défaut à l'aide de la console.

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>

2. Dans le panneau de navigation, choisissez Tables, puis Create table (Créer une table).
3. Sur la page Créer une table dans la section Détails de la table, sélectionnez un espace de touche et donnez un nom à la nouvelle table.
4. Dans la section Schéma, créez le schéma de votre table.
5. Dans la section Paramètres du tableau, choisissez Personnaliser les paramètres.
6. Passez à Time to Live (TTL).

Au cours de cette étape, vous sélectionnez les paramètres TTL par défaut pour le tableau.

Pour la période TTL par défaut, entrez le délai d'expiration et choisissez l'unité de temps que vous avez saisie, par exemple les secondes, les jours ou les années. Amazon Keyspaces stockera la valeur en quelques secondes.

7. Choisissez Créer un tableau. Votre table est créée avec la valeur TTL par défaut spécifiée.

Cassandra Query Language (CQL)

Créez une nouvelle table avec une valeur TTL par défaut à l'aide de CQL

1. L'instruction suivante crée une nouvelle table dont la valeur TTL par défaut est définie sur 3 024 000 secondes, ce qui représente 35 jours.

```
CREATE TABLE my_table (  
    userid uuid,  
    time timeuuid,  
    subject text,  
    body text,  
    user inet,  
    PRIMARY KEY (userid, time)  
    ) WITH default_time_to_live = 3024000;
```

2. Pour confirmer les paramètres TTL de la nouvelle table, utilisez l'`cqlshDESCRIBE` instruction illustrée dans l'exemple suivant. La sortie indique le paramètre TTL par défaut pour le tableau sous `default_time_to_live` la forme.

```
DESC TABLE my_table;
```

```
CREATE TABLE my_keyspace.my_table (  
    userid uuid,
```

```

    time timeuuid,
    body text,
    subject text,
    user inet,
    PRIMARY KEY (userid, time)
) WITH CLUSTERING ORDER BY (time ASC)
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'class': 'com.amazonaws.cassandra.DefaultCaching'}
    AND comment = ''
    AND compaction = {'class': 'com.amazonaws.cassandra.DefaultCompaction'}
    AND compression = {'class': 'com.amazonaws.cassandra.DefaultCompression'}
    AND crc_check_chance = 1.0
    AND dclocal_read_repair_chance = 0.0
    AND default_time_to_live = 3024000
    AND gc_grace_seconds = 7776000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 3600000
    AND min_index_interval = 128
    AND read_repair_chance = 0.0
    AND speculative_retry = '99PERCENTILE';

```

CLI

Créez une nouvelle table avec une valeur TTL par défaut à l'aide du AWS CLI

1. Vous pouvez utiliser la commande suivante pour créer une nouvelle table avec la valeur TTL par défaut définie sur un an.

```

aws keyspaces create-table --keyspace-name 'myKeyspace' --table-name 'myTable' \
    --schema-definition 'allColumns=[{name=id,type=int},
    {name=name,type=text},{name=date,type=timestamp}],partitionKeys=[{name=id}]' \
    --default-time-to-live '31536000'

```

2. Pour confirmer l'état TTL de la table, vous pouvez utiliser la commande suivante.

```

aws keyspaces get-table --keyspace-name 'myKeyspace' --table-name 'myTable'

```

Le résultat de la commande ressemble à celui de l'exemple suivant

```

{
  "keyspaceName": "myKeyspace",

```



```

    "tableName": "myTable",
    "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
myKeyspace/table/myTable",
    "creationTimestamp": "2024-09-02T10:52:22.190000+00:00",
    "status": "ACTIVE",
    "schemaDefinition": {
      "allColumns": [
        {
          "name": "id",
          "type": "int"
        },
        {
          "name": "date",
          "type": "timestamp"
        },
        {
          "name": "name",
          "type": "text"
        }
      ],
      "partitionKeys": [
        {
          "name": "id"
        }
      ],
      "clusteringKeys": [],
      "staticColumns": []
    },
    "capacitySpecification": {
      "throughputMode": "PAY_PER_REQUEST",
      "lastUpdateToPayPerRequestTimestamp": "2024-09-02T10:52:22.190000+00:00"
    },
    "encryptionSpecification": {
      "type": "AWS_OWNED_KMS_KEY"
    },
    "pointInTimeRecovery": {
      "status": "DISABLED"
    },
    "ttl": {
      "status": "ENABLED"
    },
    "defaultTimeToLive": 31536000,
    "comment": {
      "message": ""
    }
  }

```

```
    },  
    "replicaSpecifications": []  
  }  
}
```

Mettre à jour la valeur TTL (Time to Live) par défaut d'une table

Vous pouvez mettre à jour une table existante avec une nouvelle valeur TTL par défaut. Les valeurs TTL sont définies en secondes, et la valeur maximale configurable est de 630 720 000 secondes, soit l'équivalent de 20 ans.

Lorsque vous activez le TTL sur une table, Amazon Keyspaces commence à stocker des métadonnées TTL-related supplémentaires pour chaque ligne. En outre, le TTL utilise des horodatages d'expiration pour suivre l'expiration des lignes ou des colonnes. Les horodatages sont stockés sous forme de métadonnées de ligne et contribuent au coût de stockage de la ligne.

Une fois le protocole TTL activé pour une table, vous pouvez remplacer le paramètre TTL par défaut de la table pour des lignes ou des colonnes spécifiques par des instructions CQL DML. Pour plus d'informations, consultez [the section called “INSERTÀ utiliser pour définir un TTL personnalisé pour les nouvelles lignes”](#) et [the section called “UPDATEÀ utiliser pour définir un TTL personnalisé pour les lignes et les colonnes”](#).

Une fois la fonctionnalité TTL activée, vous ne pouvez pas la désactiver pour une table. Le fait de `default_time_to_live` définir le tableau sur 0 désactive les délais d'expiration par défaut pour les nouvelles données, mais cela ne désactive pas la fonctionnalité TTL et ne rétablit pas les métadonnées de stockage ou le comportement d'écriture d'origine du tableau Amazon Keyspaces.

Procédez comme suit pour mettre à jour les paramètres Time to Live par défaut pour les tables existantes à l'aide de la console, du CQL ou du AWS CLI.

Console

Mettre à jour la valeur TTL par défaut d'une table à l'aide de la console

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Choisissez le tableau que vous souhaitez mettre à jour, puis cliquez sur l'onglet Paramètres supplémentaires.
3. Passez à Time to Live (TTL) et choisissez Modifier.

4. Pour la période TTL par défaut, entrez le délai d'expiration et choisissez l'unité de temps, par exemple les secondes, les jours ou les années. Amazon Keyspaces stockera la valeur en quelques secondes. Cela ne modifie pas la valeur TTL des lignes existantes.
5. Lorsque les paramètres TTL sont définis, choisissez Enregistrer les modifications.

Cassandra Query Language (CQL)

Mettre à jour la valeur TTL par défaut d'une table à l'aide de CQL

1. Vous pouvez l'utiliser `ALTER TABLE` pour modifier les paramètres TTL (Time to Live) par défaut d'une table. Pour mettre à jour les paramètres TTL par défaut de la table à 2 592 000 secondes, ce qui représente 30 jours, vous pouvez utiliser l'instruction suivante.

```
ALTER TABLE my_table WITH default_time_to_live = 2592000;
```

2. Pour confirmer les paramètres TTL de la table mise à jour, utilisez `DESCRIBE` instruction illustrée dans l'exemple suivant. La sortie indique le paramètre TTL par défaut pour le tableau sous `default_time_to_live` la forme.

```
DESC TABLE my_table;
```

Le résultat de l'instruction doit ressembler à celui de cet exemple.

```
CREATE TABLE my_keyspace.my_table (  
  id int PRIMARY KEY,  
  date timestamp,  
  name text  
) WITH bloom_filter_fp_chance = 0.01  
  AND caching = {'class': 'com.amazonaws.cassandra.DefaultCaching'}  
  AND comment = ''  
  AND compaction = {'class': 'com.amazonaws.cassandra.DefaultCompaction'}  
  AND compression = {'class': 'com.amazonaws.cassandra.DefaultCompression'}  
  AND crc_check_chance = 1.0  
  AND dclocal_read_repair_chance = 0.0  
  AND default_time_to_live = 2592000  
  AND gc_grace_seconds = 7776000  
  AND max_index_interval = 2048  
  AND memtable_flush_period_in_ms = 3600000  
  AND min_index_interval = 128  
  AND read_repair_chance = 0.0
```

```
AND speculative_retry = '99PERCENTILE';
```

CLI

Mettez à jour la valeur TTL par défaut d'une table à l'aide du AWS CLI

1. Vous pouvez l'utiliser `update-table` pour modifier la valeur TTL par défaut d'une table. Pour mettre à jour les paramètres TTL par défaut de la table à 2 592 000 secondes, ce qui représente 30 jours, vous pouvez utiliser l'instruction suivante.

```
aws keyspaces update-table --keyspace-name 'myKeyspace' --table-name 'myTable'
--default-time-to-live '2592000'
```

2. Pour confirmer la valeur TTL par défaut mise à jour, vous pouvez utiliser l'instruction suivante.

```
aws keyspaces get-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

Le résultat de l'instruction doit ressembler à celui de l'exemple suivant.

```
{
  "keyspaceName": "myKeyspace",
  "tableName": "myTable",
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
myKeyspace/table/myTable",
  "creationTimestamp": "2024-09-02T10:52:22.190000+00:00",
  "status": "ACTIVE",
  "schemaDefinition": {
    "allColumns": [
      {
        "name": "id",
        "type": "int"
      },
      {
        "name": "date",
        "type": "timestamp"
      },
      {
        "name": "name",
        "type": "text"
      }
    ]
  },
}
```

```
    "partitionKeys": [
      {
        "name": "id"
      }
    ],
    "clusteringKeys": [],
    "staticColumns": []
  },
  "capacitySpecification": {
    "throughputMode": "PAY_PER_REQUEST",
    "lastUpdateToPayPerRequestTimestamp": "2024-09-02T10:52:22.190000+00:00"
  },
  "encryptionSpecification": {
    "type": "AWS_OWNED_KMS_KEY"
  },
  "pointInTimeRecovery": {
    "status": "DISABLED"
  },
  "ttl": {
    "status": "ENABLED"
  },
  "defaultTimeToLive": 2592000,
  "comment": {
    "message": ""
  },
  "replicaSpecifications": []
}
```

Créer un tableau avec les paramètres personnalisés de durée de vie (TTL) activés

Pour créer un nouveau tableau avec les paramètres personnalisés Time to Live qui peuvent être appliqués aux lignes et aux colonnes sans activer les paramètres TTL par défaut pour l'ensemble du tableau, vous pouvez utiliser les commandes suivantes.

Note

Si un tableau est créé avec des paramètres `ttl` personnalisés activés, vous ne pourrez pas les désactiver ultérieurement.

Cassandra Query Language (CQL)

Créez une nouvelle table avec un paramètre TTL personnalisé à l'aide de CQL

- ```
CREATE TABLE my_keyspace.my_table (id int primary key) WITH
CUSTOM_PROPERTIES={'ttl':{'status': 'enabled'}};
```

## CLI

Créez une nouvelle table avec un paramètre TTL personnalisé à l'aide du AWS CLI

1. Vous pouvez utiliser la commande suivante pour créer une nouvelle table avec le TTL activé.

```
aws keyspaces create-table --keyspace-name 'myKeyspace' --table-name 'myTable' \
 --schema-definition
'allColumns=[{name=id,type=int},{name=name,type=text},
{name=date,type=timestamp}],partitionKeys=[{name=id}]' \
 --ttl 'status=ENABLED'
```

2. Pour confirmer que le TTL est activé pour le tableau, vous pouvez utiliser l'instruction suivante.

```
aws keyspaces get-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

Le résultat de l'instruction doit ressembler à celui de l'exemple suivant.

```
{
 "keyspaceName": "myKeyspace",
 "tableName": "myTable",
 "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
myKeyspace/table/myTable",
 "creationTimestamp": "2024-09-02T10:52:22.190000+00:00",
 "status": "ACTIVE",
 "schemaDefinition": {
 "allColumns": [
 {
 "name": "id",
 "type": "int"
 },
 {
 "name": "date",
```

```

 "type": "timestamp"
 },
 {
 "name": "name",
 "type": "text"
 }
],
 "partitionKeys": [
 {
 "name": "id"
 }
],
 "clusteringKeys": [],
 "staticColumns": []
 },
 "capacitySpecification": {
 "throughputMode": "PAY_PER_REQUEST",
 "lastUpdateToPayPerRequestTimestamp": "2024-09-02T11:18:55.796000+00:00"
 },
 "encryptionSpecification": {
 "type": "AWS_OWNED_KMS_KEY"
 },
 "pointInTimeRecovery": {
 "status": "DISABLED"
 },
 "ttl": {
 "status": "ENABLED"
 },
 "defaultTimeToLive": 0,
 "comment": {
 "message": ""
 },
 "replicaSpecifications": []
}

```

## Mettre à jour le tableau avec Time to Live (TTL) personnalisé

Pour activer les paramètres personnalisés Time to Live pour un tableau afin que les valeurs TTL puissent être appliquées à des lignes et colonnes individuelles sans définir de valeur TTL par défaut pour l'ensemble du tableau, vous pouvez utiliser les commandes suivantes.

 Note

Une fois `t1` activé, vous ne pouvez pas le désactiver pour le tableau.

## Cassandra Query Language (CQL)

Activer les paramètres TTL personnalisés pour une table à l'aide de CQL

- ```
ALTER TABLE my_table WITH CUSTOM_PROPERTIES={'ttl':{'status': 'enabled'}};
```

CLI

Activez les paramètres TTL personnalisés pour un tableau à l'aide du AWS CLI

- Vous pouvez utiliser la commande suivante pour mettre à jour le paramètre TTL personnalisé d'une table.

```
aws keyspaces update-table --keyspace-name 'myKeyspace' --table-name 'myTable' --ttl 'status=ENABLED'
```

- Pour confirmer que le TTL est désormais activé pour la table, vous pouvez utiliser l'instruction suivante.

```
aws keyspaces get-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

Le résultat de l'instruction doit ressembler à celui de l'exemple suivant.

```
{
  "keyspaceName": "myKeyspace",
  "tableName": "myTable",
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
myKeyspace/table/myTable",
  "creationTimestamp": "2024-09-02T11:32:27.349000+00:00",
  "status": "ACTIVE",
  "schemaDefinition": {
    "allColumns": [
      {
        "name": "id",
        "type": "int"
```



```
    },
    {
      "name": "date",
      "type": "timestamp"
    },
    {
      "name": "name",
      "type": "text"
    }
  ],
  "partitionKeys": [
    {
      "name": "id"
    }
  ],
  "clusteringKeys": [],
  "staticColumns": []
},
"capacitySpecification": {
  "throughputMode": "PAY_PER_REQUEST",
  "lastUpdateToPayPerRequestTimestamp": "2024-09-02T11:32:27.349000+00:00"
},
"encryptionSpecification": {
  "type": "AWS_OWNED_KMS_KEY"
},
"pointInTimeRecovery": {
  "status": "DISABLED"
},
"ttl": {
  "status": "ENABLED"
},
"defaultTimeToLive": 0,
"comment": {
  "message": ""
},
"replicaSpecifications": []
}
```

Utilisez l'**INSERT** instruction pour définir des valeurs de durée de vie (TTL) personnalisées pour les nouvelles lignes

Note

Avant de pouvoir définir des valeurs TTL personnalisées pour les lignes à l'aide de l'INSERT instruction, vous devez d'abord activer le TTL personnalisé sur la table. Pour de plus amples informations, veuillez consulter [the section called “Mettre à jour le TTL personnalisé du tableau”](#).

Pour remplacer la valeur TTL par défaut d'une table en définissant des dates d'expiration pour des lignes individuelles, vous pouvez utiliser l'INSERT instruction suivante :

- INSERT— Insérez une nouvelle ligne de données avec un ensemble de valeurs TTL.

La définition des valeurs TTL pour les nouvelles lignes à l'aide de l'INSERT instruction a priorité sur le paramètre TTL par défaut de la table.

L'instruction CQL suivante insère une ligne de données dans le tableau et modifie le paramètre TTL par défaut à 259 200 secondes (ce qui équivaut à 3 jours).

```
INSERT INTO my_table (userid, time, subject, body, user)
VALUES (B79CB3BA-745E-5D9A-8903-4A02327A7E09, 96a29100-5e25-11ec-90d7-
b5d91eceda0a, 'Message', 'Hello', '205.212.123.123')
USING TTL 259200;
```

Pour confirmer les paramètres TTL de la ligne insérée, utilisez l'instruction suivante.

```
SELECT TTL (subject) from my_table;
```

Utilisez l'**UPDATE** instruction pour modifier les paramètres personnalisés de durée de vie (TTL) pour les lignes et les colonnes

Note

Avant de pouvoir définir des valeurs TTL personnalisées pour les lignes et les colonnes, vous devez d'abord activer le TTL sur le tableau. Pour de plus amples informations, veuillez consulter [the section called “Mettre à jour le TTL personnalisé du tableau”](#).

Vous pouvez utiliser l'UPDATE instruction pour remplacer la valeur TTL par défaut d'une table en définissant la date d'expiration des lignes et des colonnes individuelles :

- Lignes — Vous pouvez mettre à jour une ligne de données existante avec une valeur TTL personnalisée.
- Colonnes — Vous pouvez mettre à jour un sous-ensemble de colonnes dans des lignes existantes avec une valeur TTL personnalisée.

La définition des valeurs TTL pour les lignes et les colonnes a priorité sur le paramètre TTL par défaut de la table.

Pour modifier les paramètres TTL de la colonne « objet » insérée précédemment de 259 200 secondes (3 jours) à 86 400 secondes (un jour), utilisez l'instruction suivante.

```
UPDATE my_table USING TTL 86400 set subject = 'Updated Message' WHERE userid =
B79CB3BA-745E-5D9A-8903-4A02327A7E09 and time = 96a29100-5e25-11ec-90d7-b5d91eceda0a;
```

Vous pouvez exécuter une simple requête de sélection pour voir l'enregistrement mis à jour avant l'heure d'expiration.

```
SELECT * from my_table;
```

La requête affiche le résultat suivant.

userid	subject	user	time	body
-----+-----+-----				
+-----+-----				

```
b79cb3ba-745e-5d9a-8903-4a02327a7e09 | 96a29100-5e25-11ec-90d7-b5d91eceda0a | Hello |
Updated Message | 205.212.123.123
50554d6e-29bb-11e5-b345-feff819cdc9f | cf03fb21-59b5-11ec-b371-dff626ab9620 | Hello |
Message | 205.212.123.123
```

Pour confirmer que l'expiration a bien été effectuée, réexécutez la même requête après le délai d'expiration configuré.

```
SELECT * from my_table;
```

La requête affiche le résultat suivant une fois que la colonne « objet » a expiré.

```
userid | time | body |
subject | user
-----+-----+-----
+-----+-----+-----
b79cb3ba-745e-5d9a-8903-4a02327a7e09 | 96a29100-5e25-11ec-90d7-b5d91eceda0a | Hello |
null | 205.212.123.123
50554d6e-29bb-11e5-b345-feff819cdc9f | cf03fb21-59b5-11ec-b371-dff626ab9620 | Hello |
Message | 205.212.123.123
```

L'utilisation de ce service avec un AWS Kit SDK

AWS des kits de développement logiciel (SDK) sont disponibles pour de nombreux langages de programmation populaires. Chaque kit SDK fournit une API, des exemples de code et de la documentation qui facilitent la création d'applications par les développeurs dans leur langage préféré.

Documentation SDK	Exemples de code
AWS SDK pour C++	AWS SDK pour C++ exemples de code
AWS CLI	AWS CLI exemples de code
AWS SDK pour Go	AWS SDK pour Go exemples de code
AWS SDK pour Java	AWS SDK pour Java exemples de code
AWS SDK pour JavaScript	AWS SDK pour JavaScript exemples de code
AWS SDK pour Kotlin	AWS SDK pour Kotlin exemples de code

Documentation SDK	Exemples de code
AWS SDK pour .NET	AWS SDK pour .NET exemples de code
AWS SDK pour PHP	AWS SDK pour PHP exemples de code
Outils AWS pour PowerShell	Outils AWS pour PowerShell exemples de code
AWS SDK pour Python v4	AWS Exemples de code SDK pour Python v4
AWS SDK pour Python (Boto3)	AWS SDK pour Python (Boto3) exemples de code
AWS SDK pour Ruby	AWS SDK pour Ruby exemples de code
AWS SDK pour Rust	AWS SDK pour Rust exemples de code
AWS SDK pour SAP ABAP	AWS SDK pour SAP ABAP exemples de code
AWS SDK pour Swift	AWS SDK pour Swift exemples de code

Exemple de disponibilité

Vous n'avez pas trouvé ce dont vous avez besoin ? Demandez un exemple de code en utilisant le lien [Provide feedback](#) (Fournir un commentaire) en bas de cette page.

Utilisation de balises et d'étiquettes pour les ressources Amazon Keyspaces

Vous pouvez étiqueter les ressources Amazon Keyspaces (pour Apache Cassandra) à l'aide de balises. Les balises vous permettent de classer vos ressources de différentes manières, par exemple en fonction de leur objectif, de leur propriétaire, de leur environnement ou d'autres critères. Les balises vous permettent d'effectuer les actions suivantes :

- Identifier rapidement une ressource en fonction des balises que vous lui avez attribuées.
- Voir AWS les factures ventilées par tags.

- Contrôlez l'accès aux ressources Amazon Keyspaces à l'aide de balises. Pour des exemples de politique IAM utilisant des balises, consultez [the section called “Autorisation basée sur les tags Amazon Keyspaces”](#).

Le balisage est pris en charge par AWS des services tels qu'Amazon Elastic Compute Cloud (Amazon EC2), Amazon Simple Storage Service (Amazon S3), Amazon Keyspaces, etc. Un balisage efficace peut fournir un aperçu sur les coûts en vous permettant de créer des rapports sur les services associés à une balise spécifique.

Pour commencer le balisage, procédez comme suit :

1. Comprendre [Restrictions relatives à l'utilisation de balises pour étiqueter les ressources dans Amazon Keyspaces](#).
2. Créer des balises en utilisant [Étiquetez les espaces clés, les tables et les flux dans Amazon Keyspaces](#).
3. [Créez des rapports de répartition des coûts à l'aide de balises pour Amazon Keyspaces](#) À utiliser pour suivre vos AWS coûts par tag actif.

En dernier lieu, il est conseillé de suivre les politique de balisage optimales. Pour plus d'informations, consultez [Politiques d'étiquetage AWS](#).

Restrictions relatives à l'utilisation de balises pour étiqueter les ressources dans Amazon Keyspaces

Chaque étiquette est constituée d'une clé et d'une valeur, que vous définissez. Les restrictions suivantes s'appliquent :

- Chaque espace de touches, table ou flux Amazon Keyspaces ne peut comporter qu'une seule balise avec la même clé. Si vous tentez d'ajouter une balise existante (même clé), la valeur de la balise existante est mise à jour avec la nouvelle valeur.
- Les balises appliquées à un espace de touches ne s'appliquent pas automatiquement aux tables de cet espace de touches. Pour appliquer la même balise à un keyspace et à toutes ses tables, chaque ressource doit être étiquetée individuellement.
- Les balises appliquées à une table ne s'appliquent pas automatiquement au flux de cette table. Pour appliquer les mêmes balises à une table et au flux lors de la création de la table, vous pouvez utiliser l'`PropagateTagsOnEnable` indicateur lors de la création de la table. À l'aide de cet

indicateur, Amazon Keyspaces applique les balises de la table au flux lors de la création du flux. Lorsque le flux est actif, les modifications apportées aux balises du tableau ne s'appliquent pas au flux.

- Lorsque vous créez un espace de touches ou un tableau multirégional, toutes les balises que vous définissez au cours du processus de création sont automatiquement appliquées à tous les espaces clés et tables de toutes les régions. Lorsque vous modifiez des balises existantes à l'aide de `ALTER KEYSPACE` ou `ALTER TABLE`, la mise à jour ne s'applique qu'au keyspace ou au tableau de la région dans laquelle vous effectuez la modification.
- Une valeur agit comme un descripteur au sein d'une catégorie d'étiquette (clé). Dans Amazon Keyspaces, la valeur ne peut pas être vide ou nulle.
- Les clés et les valeurs des balises distinguent les majuscules et minuscules.
- La longueur de clé maximale est de 128 caractères Unicode.
- La longueur de valeur maximale est de 256 caractères Unicode.
- Les caractères autorisés sont les lettres, les espaces blancs et les chiffres, ainsi que les caractères spéciaux suivants : `+ - = . _ : /`
- Le nombre maximum d'identifications par ressource est de 50.
- AWS-les noms et valeurs des balises assignés reçoivent automatiquement le `aws` : préfixe, que vous ne pouvez pas attribuer. AWS-les noms de balises attribués ne sont pas pris en compte dans la limite de 50 balises. User-assigned les noms de balises ont le préfixe `user` : dans le rapport de répartition des coûts.
- Vous ne pouvez pas antidater l'application d'une balise.

Étiquetez les espaces clés, les tables et les flux dans Amazon Keyspaces

Vous pouvez ajouter, répertorier, modifier ou supprimer des balises pour les espaces clés, les tables et les flux à l'aide de la console Amazon Keyspaces, AWS CLI du langage CQL (Cassandra Query Language). Vous pouvez ensuite activer ces balises définies par l'utilisateur afin qu'elles apparaissent sur la AWS Billing and Cost Management console pour le suivi de la répartition des coûts. Pour de plus amples informations, veuillez consulter [Créez des rapports de répartition des coûts à l'aide de balises pour Amazon Keyspaces](#).

Pour les modifications groupées, vous pouvez également utiliser l'éditeur de balises sur la console. Pour plus d'informations, veuillez consulter [Utilisation de Tag Editor](#) dans le Guide de l'utilisateur Groupes de ressources AWS .

Pour plus d'informations sur la structure des balises, consultez [Restrictions relatives à l'utilisation de balises pour étiqueter les ressources dans Amazon Keyspaces](#).

Rubriques

- [Ajouter des tags lors de la création d'un nouvel espace de touches](#)
- [Ajouter des tags à un keyspace](#)
- [Supprimer des tags d'un keyspace](#)
- [Afficher les tags d'un keyspace](#)
- [Ajouter des balises lors de la création d'un nouveau tableau](#)
- [Ajouter des balises à un tableau](#)
- [Supprimer des balises d'un tableau](#)
- [Afficher les balises d'un tableau](#)
- [Ajouter des balises à un nouveau flux lors de la création d'une table](#)
- [Ajouter des balises à un nouveau flux pour une table existante](#)
- [Ajouter de nouveaux tags à un stream](#)
- [Supprimer des balises d'un flux](#)
- [Afficher les tags d'un stream](#)

Ajouter des tags lors de la création d'un nouvel espace de touches

Vous pouvez utiliser la console Amazon Keyspaces, le CQL ou le AWS CLI pour ajouter des balises lorsque vous créez un nouvel espace de touches.

Console

Définissez une balise lors de la création d'un nouvel espace de touches à l'aide de la console

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le panneau de navigation, choisissez Keyspaces, puis Créer un keyspace.
3. Dans la page Créer un keyspace indiquez un nom pour le keyspace.
4. Sous Balises, choisissez Ajouter une nouvelle balise et entrez une clé et une valeur.
5. Choisissez Create keyspace.

Cassandra Query Language (CQL)

Définissez une balise lors de la création d'un nouvel espace de touches à l'aide de CQL

- L'exemple suivant crée un nouveau keyspace avec des balises.

```
CREATE KEYSPACE mykeyspace WITH TAGS = {'key1':'val1', 'key2':'val2'};
```

CLI

Définissez une balise lors de la création d'un nouvel espace de touches à l'aide du AWS CLI

- L'instruction suivante crée un nouvel espace de touches avec des balises.

```
aws keyspaces create-keyspace --keyspace-name 'myKeyspace' --tags  
'key=key1,value=val1' 'key=key2,value=val2'
```

Ajouter des tags à un keyspace

Les exemples suivants montrent comment ajouter des balises à un keyspace dans Amazon Keyspaces.

Console

Ajouter un tag à un espace de touches existant à l'aide de la console

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le panneau de navigation, choisissez Keyspaces.
3. Choisissez un espace de touche dans la liste. Choisissez ensuite l'onglet Tags où vous pouvez voir les tags du keyspace.
4. Choisissez Gérer les balises pour ajouter, modifier ou supprimer des balises.
5. Sélectionnez Enregistrer les modifications.

Cassandra Query Language (CQL)

Ajouter une balise à un espace de touches existant à l'aide de CQL

- ```
ALTER KEYSPACE mykeyspace ADD TAGS {'key1':'val1', 'key2':'val2'};
```

## CLI

Ajoutez une balise à un espace de touches existant à l'aide du AWS CLI

- L'exemple suivant montre comment ajouter de nouvelles balises à un keyspace existant.

```
aws keyspaces tag-resource --resource-arn 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/myKeyspace/' --tags 'key=key3,value=val3'
'key=key4,value=val4'
```

## Supprimer des tags d'un keyspace

### Console

Supprimer un tag d'un espace de touches existant à l'aide de la console

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le panneau de navigation, choisissez Keyspaces.
3. Choisissez un espace de touche dans la liste. Choisissez ensuite l'onglet Tags où vous pouvez voir les tags du keyspace.
4. Choisissez Gérer les tags et supprimez les tags dont vous n'avez plus besoin.
5. Sélectionnez Enregistrer les modifications.

## Cassandra Query Language (CQL)

Supprimer une balise d'un espace de touches existant à l'aide de CQL

- ```
ALTER KEYSPACE mykeyspace DROP TAGS {'key1':'val1', 'key2':'val2'};
```

CLI

Supprimer un tag d'un espace de touches existant à l'aide du AWS CLI

- L'instruction suivante supprime les balises spécifiées d'un keyspace.

```
aws keyspaces untag-resource --resource-arn 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/myKeyspace/' --tags 'key=key3,value=val3'
'key=key4,value=val4'
```

Afficher les tags d'un keyspace

Les exemples suivants montrent comment lire des balises à l'aide de la console, du CQL ou du AWS CLI.

Console

Afficher les tags d'un keyspace à l'aide de la console Amazon Keyspaces

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le panneau de navigation, choisissez Keyspaces.
3. Choisissez un espace de touche dans la liste. Choisissez ensuite l'onglet Tags où vous pouvez voir les tags du keyspace.

Cassandra Query Language (CQL)

Afficher les balises d'un keyspace à l'aide de CQL

Pour lire les balises associées à un keyspace, utilisez l'instruction CQL suivante.

```
SELECT * FROM system_schema_mcs.tags WHERE valid_where_clause;
```

La WHERE clause est obligatoire et doit utiliser l'un des formats suivants :

- `keyspace_name = 'mykeyspace' AND resource_type = 'keyspace'`
- `resource_id = arn`

- L'instruction suivante indique si un espace de touches comporte des balises.

```
SELECT * FROM system_schema_mcs.tags WHERE keyspace_name = 'mykeyspace' AND
resource_type = 'keyspace';
```

La sortie de la commande ressemble à ce qui suit.

```
resource_id | keyspace_name
| resource_name | resource_type | tags
-----
+-----+-----+-----+-----
arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/ | mykeyspace
| mykeyspace | keyspace | {'key1': 'val1', 'key2': 'val2'}
```

CLI

Affichez les balises d'un keyspace à l'aide du AWS CLI

- Cet exemple montre comment répertorier les balises de la ressource spécifiée.

```
aws keyspaces list-tags-for-resource --resource-arn 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/myKeyspace/'
```

Le résultat de la dernière commande ressemble à ceci.

```
{
  "tags": [
    {
      "key": "key1",
      "value": "val1"
    },
    {
      "key": "key2",
      "value": "val2"
    },
    {
      "key": "key3",
      "value": "val3"
    }
  ],
}
```

```
{
  "key": "key4",
  "value": "val4"
}
```

Ajouter des balises lors de la création d'un nouveau tableau

Vous pouvez utiliser la console Amazon Keyspaces, le CQL ou le AWS CLI pour ajouter des balises aux nouvelles tables lorsque vous les créez.

Console

Ajoutez une balise lors de la création d'une nouvelle table à l'aide de la (console)

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le panneau de navigation, choisissez Tables, puis Create table (Créer une table).
3. Sur la page Créer une table, dans la section Détails de la table, sélectionnez un espace de touche et attribuez un nom à la table.
4. Dans la section Schéma, créez le schéma de votre table.
5. Dans la section Paramètres du tableau, choisissez Personnaliser les paramètres.
6. Passez à la section Balises du tableau — section facultative, puis choisissez Ajouter une nouvelle balise pour créer de nouvelles balises.
7. Choisissez Créer un tableau.

Cassandra Query Language (CQL)

Ajouter des balises lors de la création d'une nouvelle table à l'aide de CQL

- L'exemple suivant crée une table avec des balises.

```
CREATE TABLE mytable(...) WITH TAGS = {'key1':'val1', 'key2':'val2'};
```

CLI

Ajoutez des balises lors de la création d'un nouveau tableau à l'aide du AWS CLI

- L'exemple suivant montre comment créer une nouvelle table avec des balises. La commande crée une table MyTable dans un espace de touches MyKeyspace déjà existant. Notez que la commande a été divisée en différentes lignes pour améliorer la lisibilité.

```
aws keyspaces create-table --keyspace-name 'myKeyspace' --table-name 'myTable'
                        --schema-definition 'allColumns=[{name=id,type=int},
{name=name,type=text},{name=date,type=timestamp}],partitionKeys=[{name=id}]'
                        --tags 'key=key1,value=val1' 'key=key2,value=val2'
```

Ajouter des balises à un tableau

Vous pouvez ajouter des balises à une table existante dans Amazon Keyspaces à l'aide de la console, du CQL ou du AWS CLI

Console

Ajouter des tags à un tableau à l'aide de la console Amazon Keyspaces

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le volet de navigation, choisissez Tables.
3. Choisissez un tableau dans la liste, puis cliquez sur l'onglet Tags.
4. Choisissez Gérer les balises pour ajouter des balises au tableau.
5. Sélectionnez Enregistrer les modifications.

Cassandra Query Language (CQL)

Ajouter des balises à une table à l'aide de CQL

- L'instruction suivante indique comment ajouter des balises à une table existante.

```
ALTER TABLE mykeyspace.mytable ADD TAGS {'key1':'val1', 'key2':'val2'};
```

CLI

Ajoutez des balises à un tableau à l'aide du AWS CLI

- L'exemple suivant montre comment ajouter de nouvelles balises à une table existante.

```
aws keyspaces tag-resource --resource-arn 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/myKeyspace/table/myTable' --tags 'key=key3,value=val3' 'key=key4,value=val4'
```

Supprimer des balises d'un tableau

Console

Supprimer des tags d'un tableau à l'aide de la console Amazon Keyspaces

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le volet de navigation, choisissez Tables.
3. Choisissez un tableau dans la liste, puis cliquez sur l'onglet Tags.
4. Choisissez Gérer les balises pour supprimer les balises du tableau.
5. Sélectionnez Enregistrer les modifications.

Cassandra Query Language (CQL)

Supprimer des balises d'une table à l'aide de CQL

- L'instruction suivante indique comment supprimer des balises d'une table existante.

```
ALTER TABLE mytable DROP TAGS {'key3':'val3', 'key4':'val4'};
```

CLI

Ajoutez des balises à un tableau à l'aide du AWS CLI

- L'instruction suivante supprime les balises spécifiées d'un keyspace.

```
aws keyspaces untag-resource --resource-arn 'arn:aws:cassandra:us-east-1:11112223333:/keyspace/myKeyspace/table/myTable' --tags 'key=key3,value=val3' 'key=key4,value=val4'
```

Afficher les balises d'un tableau

Les exemples suivants montrent comment afficher les balises d'une table dans Amazon Keyspaces à l'aide de la console, du CQL ou du AWS CLI

Console

Afficher les balises d'un tableau à l'aide de la console

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le volet de navigation, choisissez Tables.
3. Choisissez un tableau dans la liste, puis cliquez sur l'onglet Tags.

Cassandra Query Language (CQL)

Afficher les balises d'une table à l'aide de CQL

Pour lire les balises associées à une table, utilisez l'instruction CQL suivante.

```
SELECT * FROM system_schema_mcs.tags WHERE valid_where_clause;
```

La WHERE clause est obligatoire et doit utiliser l'un des formats suivants :

- `keyspace_name = 'mykeyspace' AND resource_name = 'mytable'`
- `resource_id = arn`
- La requête suivante renvoie les balises de la table spécifiée.

```
SELECT * FROM system_schema_mcs.tags WHERE keyspace_name = 'mykeyspace' AND resource_name = 'mytable';
```


La sortie de cette requête se présente comme suit :

```
resource_id |
keyspace_name | resource_name | resource_type | tags
-----
+-----+-----+-----+-----+
arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/mytable |
mykeyspace | mytable | table | {'key1': 'val1', 'key2':
'val2'}
```

CLI

Affichez les balises d'un tableau à l'aide du AWS CLI

- Cet exemple montre comment répertorier les balises de la ressource spécifiée.

```
aws keyspaces list-tags-for-resource --resource-arn
'arn:aws:arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/
table/my_table/stream/2025-05-11T21:21:33.291cassandra:us-east-1:111122223333:/
keyspace/myKeyspace/table/myTable'
```

Le résultat de la dernière commande ressemble à ceci.

```
{
  "tags": [
    {
      "key": "key1",
      "value": "val1"
    },
    {
      "key": "key2",
      "value": "val2"
    },
    {
      "key": "key3",
      "value": "val3"
    },
    {
      "key": "key4",
```

```

        "value": "val4"
    }
]
}

```

Ajouter des balises à un nouveau flux lors de la création d'une table

Vous pouvez ajouter des balises aux flux lorsque vous créez une nouvelle table avec un flux à l'aide de CQL ou AWS CLI pour baliser un flux.

Note

Amazon Keyspaces CDC nécessite la présence d'un rôle lié à un service (AWSServiceRoleForAmazonKeyspacesCDC) qui publie les données métriques provenant des flux Amazon Keyspaces CDC sur votre compte en votre nom "cloudwatch:namespace": "AWS/Cassandra". CloudWatch Ce rôle est créé automatiquement pour vous. Pour de plus amples informations, veuillez consulter [the section called "Flux CDC"](#).

Cassandra Query Language (CQL)

Ajouter des balises à un flux lors de la création d'une nouvelle table à l'aide de CQL

1. Pour créer une nouvelle table avec un flux et appliquer automatiquement les balises du tableau au flux, vous pouvez utiliser l'indicateur `'propagate_tags': 'TABLE'`. La déclaration suivante en est un exemple.

```

CREATE TABLE mytable (pk int, ck text, PRIMARY KEY(pk))
WITH TAGS={'key1':'val1', 'key2':'val2'}
AND cdc = TRUE
AND CUSTOM_PROPERTIES={
    'cdc_specification': {
        'view_type': 'NEW_IMAGE',
        'propagate_tags': 'TABLE'
    }
};

```

2. Pour appliquer de nouvelles balises au flux, vous pouvez utiliser l'exemple suivant.

```
CREATE TABLE mytable (pk int, ck text, PRIMARY KEY(pk))
WITH TAGS={'key1':'val1', 'key2':'val2'}
AND cdc = TRUE
AND CUSTOM_PROPERTIES={
    'cdc_specification': {
        'view_type': 'NEW_IMAGE',
        'tags': { 'key': 'string', 'value': 'string' },
    }
};
```

CLI

Ajoutez des balises à un flux lors de la création d'une nouvelle table à l'aide du AWS CLI

1. Pour créer une table avec un flux et appliquer automatiquement les balises du tableau au flux, vous pouvez utiliser l'`propagateTags=Table`indicateur. Le code suivant en est un exemple.

```
aws keyspaces create-table \
--keyspace-name 'my_keyspace' \
--table-name 'my_table' \
--schema-definition 'allColumns=[{name=pk,type=int},
{name=ck,type=text}],clusteringKeys=[{name=ck,orderBy=ASC}],partitionKeys=[{name=pk}]' \
--tags key=tag_key,value=tag_value
--cdc-specification propagateTags=TABLE,status=ENABLED,viewType=NEW_IMAGE
```

2. Pour appliquer différentes balises au flux, vous pouvez utiliser l'exemple suivant.

```
aws keyspaces create-table \
--keyspace-name 'my_keyspace' \
--table-name 'my_table' \
--schema-definition 'allColumns=[{name=pk,type=int},
{name=ck,type=text}],clusteringKeys=[{name=ck,orderBy=ASC}],partitionKeys=[{name=pk}]' \
--tags key=tag_key,value=tag_value
--cdc-specification 'status=ENABLED,viewType=NEW_IMAGE,tags=[{key=tag_key,
value=tag_value}]'
```

Ajouter des balises à un nouveau flux pour une table existante

Vous pouvez ajouter des balises lorsque vous créez un nouveau flux pour une table existante. Vous pouvez utiliser l'indicateur `PropagateTags` pour appliquer les balises de table au flux ou spécifier de nouvelles balises pour le flux. Vous pouvez utiliser le CQL ou le AWS CLI pour baliser un nouveau flux.

Note

Amazon Keyspaces CDC nécessite la présence d'un rôle lié à un service (AWS IAM `AWSServiceRoleForAmazonKeyspacesCDC`) qui publie les données métriques provenant des flux Amazon Keyspaces CDC sur votre compte en votre nom "cloudwatch:namespace": "AWS/Cassandra". CloudWatch Ce rôle est créé automatiquement pour vous. Pour de plus amples informations, veuillez consulter [the section called "Flux CDC"](#).

Console

Ajoutez des balises lors de la création d'un nouveau flux à l'aide de la (console)

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le volet de navigation, choisissez Tables, puis choisissez la table pour laquelle vous souhaitez ajouter un flux.
3. Choisissez l'onglet Streams.
4. Dans la section Détails du flux, choisissez Modifier.
5. Sélectionnez Activer les streams.
6. Sélectionnez le type d'affichage et passez à Tags pour créer des balises pour le flux.
7. Vous pouvez sélectionner l'une des options suivantes :
 - Pas de balises : utilisez cette option si vous ne souhaitez pas créer de balises pour le flux.
 - Copier les balises de la table : utilisez cette option si vous souhaitez copier les balises de la table vers le flux. Après avoir copié les balises, vous pouvez les modifier pour le stream. Notez que cette option n'est disponible que si le tableau comporte des balises.
 - Ajouter de nouveaux tags — Vous pouvez ajouter jusqu'à 50 tags pour le stream en choisissant Ajouter un nouveau tag.

8. Sélectionnez Enregistrer les modifications.

Cassandra Query Language (CQL)

Ajouter des tags lors de la création d'un nouveau stream

1. Pour créer un nouveau flux pour une table existante et appliquer les balises de la table au flux, vous pouvez utiliser l'indicateur `'propagate_tags': 'TABLE'`. Les déclarations suivantes en sont un exemple.

```
ALTER TABLE mytable WITH cdc = TRUE AND CUSTOM_PROPERTIES={ 'cdc_specification':  
  { 'view_type': 'NEW_IMAGE', 'propagate_tags': 'TABLE' } };
```

2. Pour créer un nouveau flux pour une table existante et spécifier de nouvelles balises, vous pouvez utiliser l'exemple suivant.

```
ALTER TABLE mytable WITH cdc = TRUE AND CUSTOM_PROPERTIES={ 'cdc_specification':  
  { 'view_type': 'NEW_IMAGE', 'tags': { 'key': 'string', 'value': 'string' } } };
```

CLI

Ajoutez des balises lors de la création d'un nouveau flux à l'aide du AWS CLI

1. Pour créer un nouveau flux avec des balises, vous pouvez utiliser l'indicateur `propagateTags=TABLE` pour appliquer automatiquement les balises de la table au flux. Le code suivant en est un exemple.

```
aws keyspaces update-table \  
--keyspace-name 'my_keyspace' \  
--table-name 'my_table' \  
--cdc-specification propagateTags=TABLE,status=ENABLED,viewType=NEW_IMAGE
```

2. Pour créer un nouveau flux pour une table existante et spécifier de nouvelles balises, vous pouvez utiliser l'exemple suivant.

```
aws keyspaces update-table \  
--keyspace-name 'my_keyspace' \  
--table-name 'my_table' \  
--cdc-specification propagateTags=TABLE,status=ENABLED,viewType=NEW_IMAGE
```

```
--cdc-specification 'status=ENABLED,viewType=NEW_IMAGE,tags=[{key=tag_key,
value=tag_value}]'
```

Ajouter de nouveaux tags à un stream

Vous pouvez ajouter de nouvelles balises à un flux existant dans Amazon Keyspaces à l'aide du CQL ou du. AWS CLI Vous ne pouvez ajouter des tags qu'au dernier stream.

Console

Ajouter des tags à un flux existant (console)

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le volet de navigation, choisissez Tables, puis choisissez la table contenant le flux que vous souhaitez baliser.
3. Choisissez l'onglet Streams.
4. Dans la section Balises choisissez Gérer les balises.
5. Choisissez Ajouter un nouveau tag pour ajouter un nouveau tag. Vous pouvez créer jusqu'à 50 balises en répétant cette étape.
6. Sélectionnez Enregistrer les modifications.

Cassandra Query Language (CQL)

Ajouter des balises à un flux à l'aide de CQL

- L'instruction suivante indique comment ajouter des balises à un flux existant.

```
ALTER TABLE mykeyspace.mytable ADD TAGS_FOR_CDC {'key1':'val1', 'key2':'val2'};
```

CLI

Ajoutez des balises à un flux à l'aide du AWS CLI

- L'exemple suivant montre comment ajouter de nouvelles balises à un flux existant.

```
aws keyspaces tag-resource --resource-arn 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/table/my_table/stream/2025-05-11T21:21:33.291' --tags 'key=key3,value=val3' 'key=key4,value=val4'
```

Supprimer des balises d'un flux

Pour supprimer des balises d'un flux, vous pouvez utiliser CQL ou le AWS CLI. Vous ne pouvez supprimer les balises que pour le dernier stream.

Console

Supprimer des tags d'un tableau à l'aide de la console Amazon Keyspaces

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le volet de navigation, choisissez Tables.
3. Choisissez un tableau dans la liste, puis cliquez sur l'onglet Streams.
4. Dans la section Tags, choisissez Gérer les tags pour supprimer les tags du tableau.
5. Après le tag que vous souhaitez supprimer, choisissez Supprimer.
6. Sélectionnez Enregistrer les modifications.

Cassandra Query Language (CQL)

Supprimer des balises d'un flux à l'aide de CQL

- L'instruction suivante indique comment supprimer des balises d'un flux existant.

```
ALTER TABLE mytable DROP TAGS_FOR_CDC {'key3':'val3', 'key4':'val4'};
```

CLI

Supprimez les balises d'un flux à l'aide du AWS CLI

- L'instruction suivante supprime les balises spécifiées d'un flux.

```
aws keyspaces untag-resource --resource-arn 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/table/my_table/stream/2025-05-11T21:21:33.291' --tags 'key=key3,value=val3' 'key=key4,value=val4'
```

Afficher les tags d'un stream

Les exemples suivants montrent comment afficher les balises d'un flux dans Amazon Keyspaces à l'aide de CQL ou du AWS CLI

Console

Afficher les tags d'un flux à l'aide de la console

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le volet de navigation, choisissez Tables.
3. Choisissez un tableau dans la liste, puis cliquez sur l'onglet Streams.
4. Vous pouvez consulter les tags du stream dans la section Tags.

Cassandra Query Language (CQL)

Afficher les balises d'un flux à l'aide de CQL

Pour lire les balises associées à un flux, vous devez spécifier l'ARN de la ressource du flux dans la WHERE clause. La syntaxe CQL suivante en est un exemple.

```
SELECT * FROM system_schema_mcs.tags WHERE resource_id = stream_arn;
```

- La requête suivante renvoie les balises pour le flux spécifié.

```
SELECT tags FROM system_schema_mcs.tags WHERE resource_id =  
'arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/table/my_table/  
stream/2025-05-06T17:17:39.800';
```

La sortie de cette requête se présente comme suit :


```

resource_id
                                     | keyspace_name | resource_name
resource_type | tags
-----
+-----+-----+-----+
+-----+
arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/table/my_table/
stream/2025-04-02T23:00:07.052 | singleks | 2025-04-02T23:00:07.052 |
stream | {'tagkey': 'tagval'}

```

CLI

Affichez les tags d'un flux à l'aide du AWS CLI

- Cet exemple montre comment répertorier les balises de tous les flux sous le keyspace spécifié.

```
aws keyspaces list-tags-for-resource --resource-arn 'arn:aws:cassandra:us-
east-1:111122223333:/keyspace/my_keyspace/table/my_table/
stream/2025-05-11T21:21:33.291'
```

Le résultat de la dernière commande ressemble à ceci.

```
{
  "tags": [
    {
      "key": "key1",
      "value": "val1"
    },
    {
      "key": "key2",
      "value": "val2"
    },
    {
      "key": "key3",
      "value": "val3"
    },
    {
      "key": "key4",
      "value": "val4"
    }
  ]
}
```

```
}  
  ]  
}
```

Créez des rapports de répartition des coûts à l'aide de balises pour Amazon Keyspaces

AWS utilise des balises pour organiser les coûts des ressources dans votre rapport de répartition des coûts. AWS fournit deux types de balises de répartition des coûts :

- Une balise AWS générée. AWS définit, crée et applique cette balise pour vous.
- User-defined balises. Vous définissez, créez et appliquez ces étiquettes.

Vous devez activer les deux types d'étiquettes séparément pour qu'elles apparaissent dans Cost Explorer ou sur un rapport de répartition des coûts.

Pour activer les balises AWS générées :

1. Connectez-vous à la console Billing Console de gestion AWS and Cost Management et ouvrez-la à l'adresse <https://console.aws.amazon.com/billing/home#/>.
2. Dans le volet de navigation, choisissez Balises de répartition des coûts.
3. Sous Identifications de répartition des coûts générées par AWS, choisissez Activer.

Pour activer les étiquettes définies par l'utilisateur :

1. Connectez-vous à la console Billing Console de gestion AWS and Cost Management et ouvrez-la à l'adresse <https://console.aws.amazon.com/billing/home#/>.
2. Dans le volet de navigation, choisissez Balises de répartition des coûts.
3. Sous Balises de répartition des User-Defined coûts, sélectionnez Activer.

Après avoir créé et activé les balises, AWS génère un rapport de répartition des coûts avec votre utilisation et vos coûts regroupés en fonction de vos balises actives. Le rapport de répartition des coûts inclut tous vos AWS coûts pour chaque période de facturation. Ce rapport inclut les ressources balisées et non balisées, afin que vous puissiez organiser clairement les frais pour les ressources.

 Note

À l'heure actuelle, les données transférées depuis Amazon Keyspaces ne sont pas ventilées par balises sur les rapports de répartition des coûts.

Pour plus d'informations, consultez [Utilisation des étiquettes d'allocation des coûts](#).

Créez des ressources Amazon Keyspaces avec AWS CloudFormation

Amazon Keyspaces est intégré à un service qui vous aide à modéliser et à configurer vos AWS espaces de touches et vos tables afin que vous puissiez passer moins de temps à créer et à gérer vos ressources et votre infrastructure. AWS CloudFormation Vous créez un modèle qui décrit les espaces clés et les tables que vous souhaitez, et vous vous CloudFormation occupez de l'approvisionnement et de la configuration de ces ressources pour vous.

Lorsque vous l'utilisez CloudFormation, vous pouvez réutiliser votre modèle pour configurer vos ressources Amazon Keyspaces de manière cohérente et répétée. Décrivez simplement vos ressources une seule fois, puis fournissez les mêmes ressources encore et encore dans plusieurs Comptes AWS régions.

Amazon Keyspaces et modèles CloudFormation

[Pour fournir et configurer des ressources pour Amazon Keyspaces, vous devez comprendre CloudFormation les modèles](#). Les modèles sont des fichiers texte formatés en JSON ou YAML. Ces modèles décrivent les ressources que vous souhaitez mettre à disposition dans vos CloudFormation piles. Si vous n'êtes pas familiarisé avec JSON ou YAML, vous pouvez utiliser CloudFormation Designer pour vous aider à démarrer avec les CloudFormation modèles. Pour plus d'informations, voir [Qu'est-ce qu'un CloudFormation designer ?](#) dans le guide de AWS CloudFormation l'utilisateur.

Amazon Keyspaces prend en charge la création d'espaces de touches et de tableaux dans. CloudFormation Pour les tables que vous créez à l'aide de CloudFormation modèles, vous pouvez spécifier le schéma, le read/write mode, les paramètres de débit provisionné et les autres fonctionnalités prises en charge. Pour plus d'informations, notamment des exemples de modèles JSON et YAML pour les espaces clés et les tables, consultez la référence du [type de ressource Amazon Keyspaces \(pour Apache Cassandra\) dans la référence](#) des modèles.AWS CloudFormation

En savoir plus sur CloudFormation

Pour en savoir plus CloudFormation, consultez les ressources suivantes :

- [AWS CloudFormation](#)
- [AWS CloudFormation Guide de l'utilisateur](#)
- [AWS CloudFormation Guide de l'utilisateur de l'interface de ligne de commande](#)

Utilisation de NoSQL Workbench avec Amazon Keyspaces (pour Apache Cassandra)

NoSQL Workbench est une application côté client qui vous permet de concevoir et de visualiser plus facilement des modèles de données non relationnels pour Amazon Keyspaces. Les clients NoSQL Workbench sont disponibles pour Windows, macOS et Linux.

Conception de modèles de données et création automatique de ressources

NoSQL Workbench vous fournit une interface pointer-cliquer pour concevoir et créer des modèles de données Amazon Keyspaces. Vous pouvez facilement créer de nouveaux modèles de données à partir de zéro en définissant des espaces clés, des tableaux et des colonnes. Vous pouvez également importer des modèles de données existants et apporter des modifications (comme ajouter, modifier ou supprimer des colonnes) pour adapter les modèles de données aux nouvelles applications. NoSQL Workbench vous permet ensuite de valider les modèles de données dans Amazon Keyspaces ou Apache Cassandra et de créer automatiquement les espaces clés et les tables. Pour savoir comment créer des modèles de données, reportez-vous aux sections [the section called “Création d'un modèle de données”](#) et [the section called “Modifier un modèle de données”](#).

Visualisation de modèles de données

À l'aide de NoSQL Workbench, vous pouvez visualiser vos modèles de données pour vous assurer qu'ils peuvent prendre en charge les requêtes et les modèles d'accès de votre application. Vous pouvez également enregistrer et exporter vos modèles de données dans différents formats à des fins de collaboration, de documentation et de présentations. Pour de plus amples informations, veuillez consulter [the section called “Visualisez un modèle de données”](#).

Rubriques

- [Téléchargement de NoSQL Workbench](#)
- [Démarez avec NoSQL Workbench](#)
- [Visualisez des modèles de données avec NoSQL Workbench](#)
- [Créez un nouveau modèle de données avec NoSQL Workbench](#)
- [Modifier les modèles de données existants avec NoSQL Workbench](#)
- [Comment valider des modèles de données dans Amazon Keyspaces et Apache Cassandra](#)
- [Exemples de modèles de données dans NoSQL Workbench](#)
- [Historique de version pour NoSQL Workbench](#)

Téléchargement de NoSQL Workbench

Suivez ces instructions pour télécharger et installer NoSQL Workbench.

Pour télécharger et installer NoSQL Workbench

1. Utilisez l'un des liens suivants pour télécharger gratuitement NoSQL Workbench.

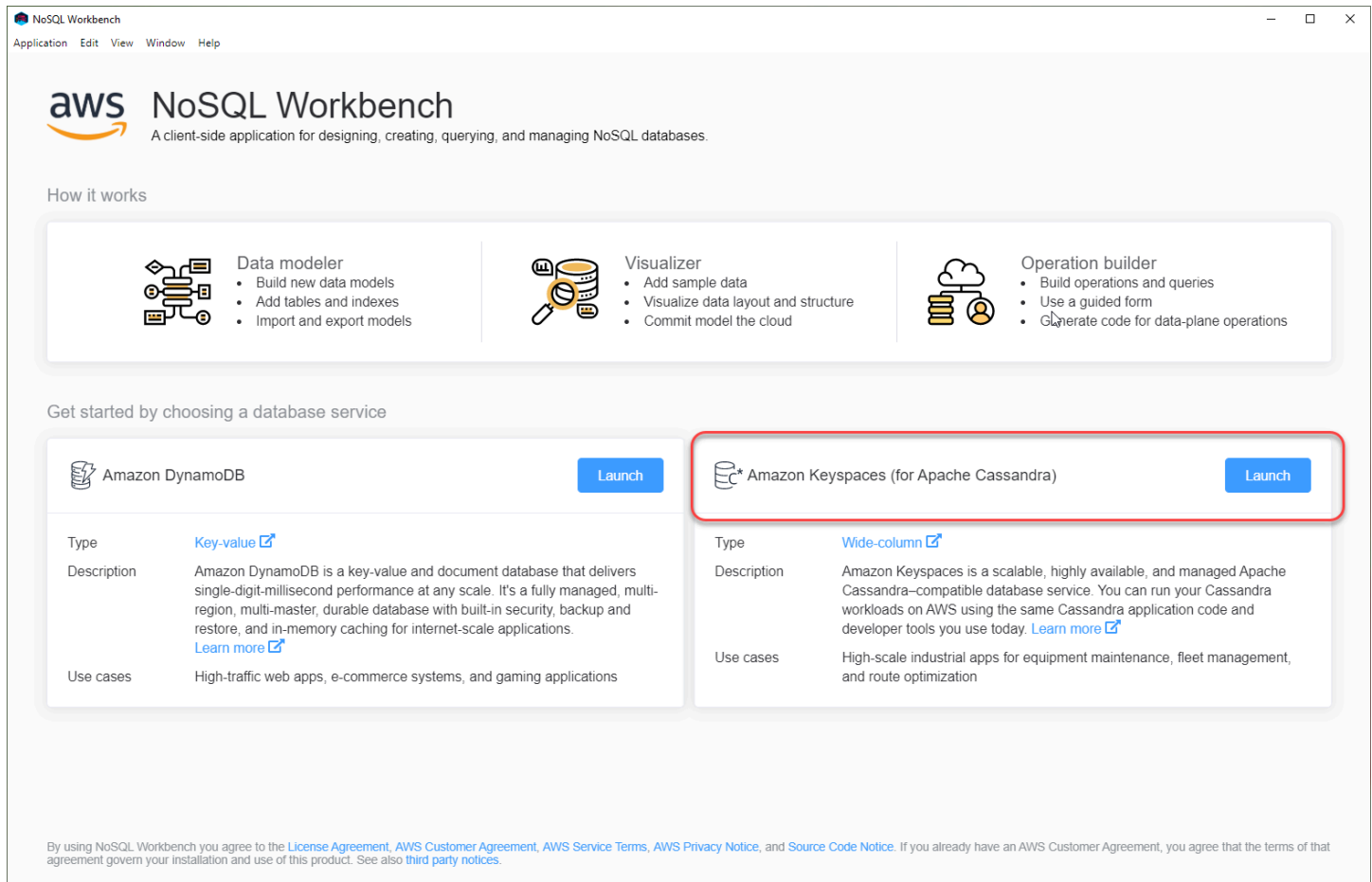
Système d'exploitation	Lien de téléchargement
macOS	Télécharger pour macOS
Linux*	Téléchargement pour Linux
Windows	Téléchargement pour Windows

* NoSQL Workbench prend en charge Ubuntu 12.04, Fedora 21 et Debian 8, ou toute version plus récente de ces distributions Linux.

2. Une fois le téléchargement terminé, démarrez l'application et suivez les instructions à l'écran pour terminer l'installation.

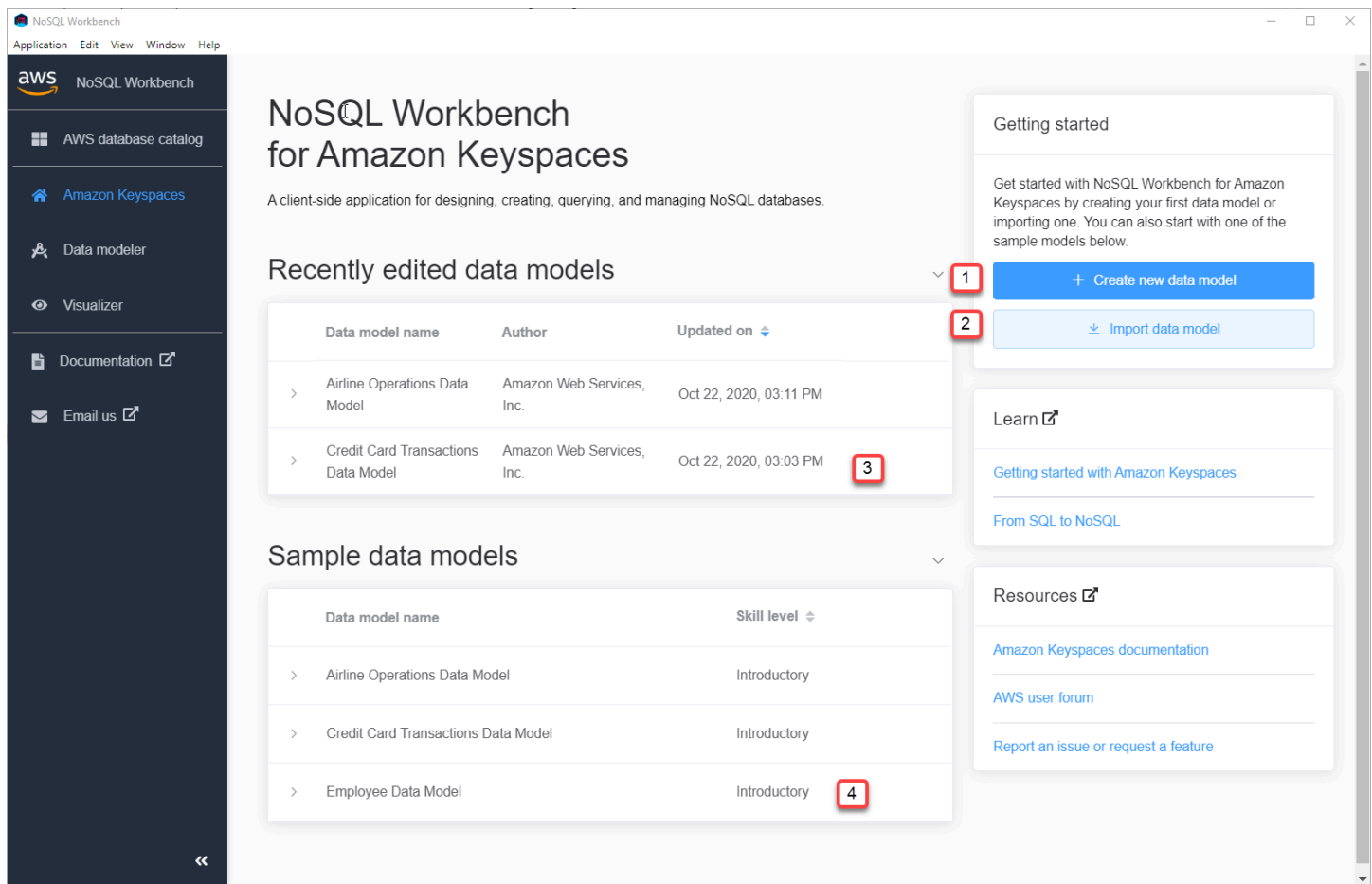
Démarez avec NoSQL Workbench

Pour commencer à utiliser NoSQL Workbench, sur la page Catalogue de bases de données de NoSQL Workbench, sélectionnez Amazon Keyspaces, puis choisissez Lancer.



Cela ouvre la page d'accueil de NoSQL Workbench pour Amazon Keyspaces, où vous disposez des options suivantes pour commencer :

1. Créez un nouveau modèle de données.
2. Importez un modèle de données existant au format JSON.
3. Ouvrez un modèle de données récemment modifié.
4. Ouvrez l'un des exemples de modèles disponibles.



Chacune des options ouvre le modèleur de données NoSQL Workbench. Pour continuer à créer un nouveau modèle de données, consultez [the section called “Création d'un modèle de données”](#). Pour modifier un modèle de données existant, consultez [the section called “Modifier un modèle de données”](#).

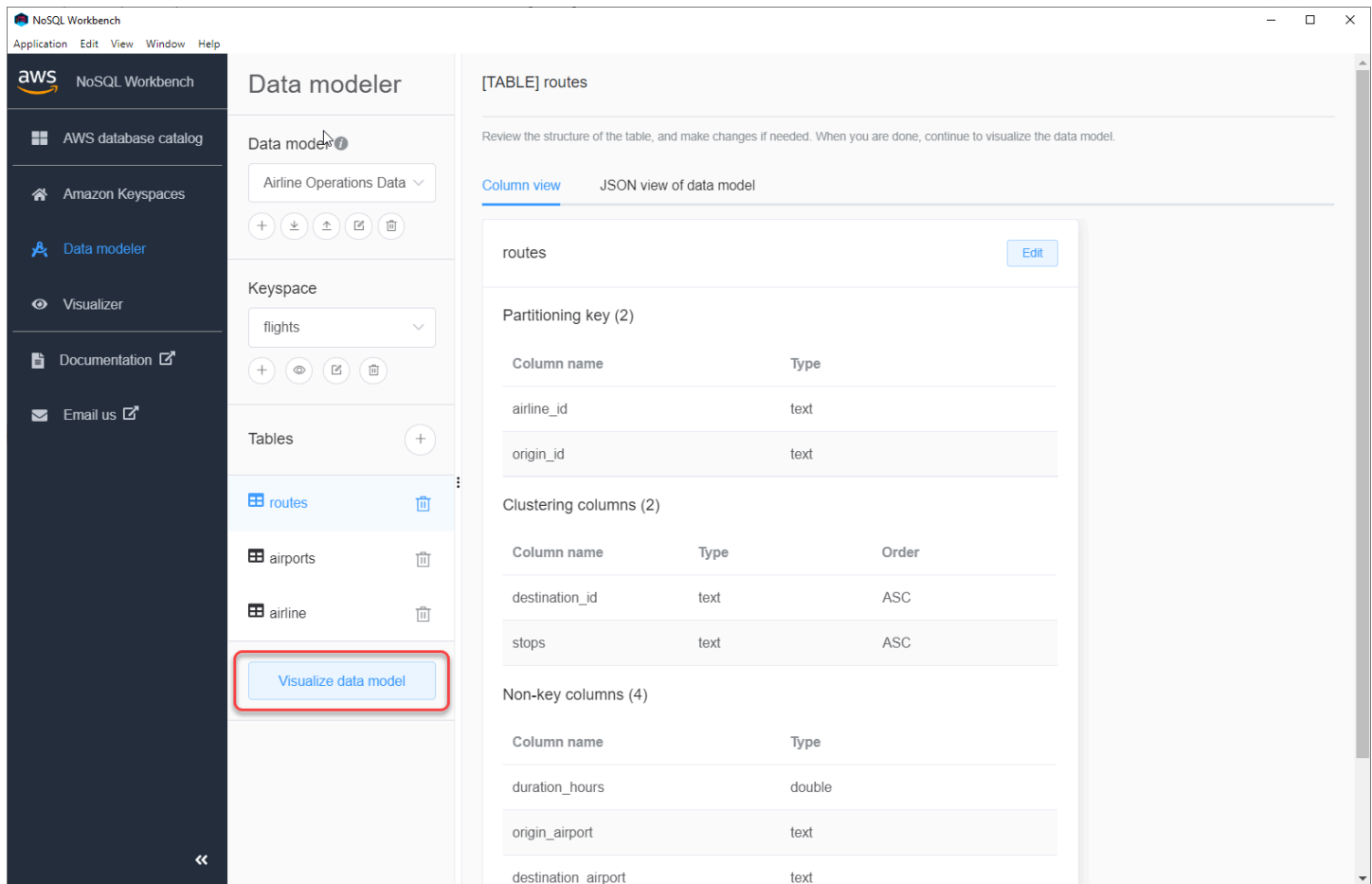
Visualisez des modèles de données avec NoSQL Workbench

À l'aide de NoSQL Workbench, vous pouvez visualiser vos modèles de données pour vous assurer qu'ils peuvent prendre en charge les requêtes et les modèles d'accès de votre application. Vous pouvez également enregistrer et exporter vos modèles de données dans différents formats à des fins de collaboration, de documentation et de présentations.

Après avoir créé un nouveau modèle de données ou modifié un modèle de données existant, vous pouvez visualiser le modèle.

Visualisation de modèles de données avec NoSQL Workbench

Lorsque vous avez terminé le modèle de données dans le modélisateur de données, choisissez **Visualiser le modèle de données**.



Cela vous amène au visualiseur de données dans NoSQL Workbench. Le visualiseur de données fournit une représentation visuelle du schéma de la table et vous permet d'ajouter des exemples de données. Pour ajouter des exemples de données à un tableau, choisissez un tableau dans le modèle, puis choisissez **Modifier**. Pour ajouter une nouvelle ligne de données, choisissez **Ajouter une nouvelle ligne** en bas de l'écran. Choisissez **Save** lorsque vous avez terminé.

The screenshot shows the AWS NoSQL Workbench Visualizer interface. On the left is a sidebar with navigation options: AWS database catalog, Amazon Keyspaces, Data modeler, Visualizer (selected), Documentation, and Email us. The main panel is titled 'Visualizer' and shows the 'Data model' for 'Airline Operations'. The 'Keyspace' is set to 'flights'. Under 'Tables', a list includes 'routes', 'airports', and 'airline'. The 'routes' table is selected, and its schema is displayed in the main area. The schema shows a primary key 'airline_id (text)' and clustering columns 'destination_id (text)' and 'stops (text)'. Non-key columns are 'duration_hours (double)' and 'origin_airport (text)'. Below the schema, a table of sample data is shown with three rows. At the bottom of the main panel, there is a '+ Add new row' button highlighted with a red box. The interface also includes buttons for 'Aggregate view', 'Commit to Amazon Keyspaces', and 'Commit to Apache Cassandra'.

Visualizer

Data model ⓘ

Airline Operations [▼]

Keyspace

flights [▼]

Tables

- routes
- airports
- airline

Aggregate view

Commit to Amazon Keyspaces

Commit to Apache Cassandra

[TABLE] routes

Use this view to get a visual representation of your tables schema, and add sample data. When you are done, commit your model to Amazon Keyspaces.

Primary key			Non-key columns	
Partition key	Clustering columns		duration_hours (double)	origin_airport (text)
airline_id (text)	destination_id (text) ⬇	stops (text) ⬇		
acme_airlines	MCI	1	0.33333333	Newark Liberty
trusted_airlines	MIC	2	0.33333333	Phoenix Sky Ha
freedom_airline	EWR	1	0.33333333	San Francisco

+ Add new row

Vue d'agrégation

Après avoir confirmé le schéma de la table, vous pouvez agréger les visualisations du modèle de données.

Visualizer

Data model ⓘ

Airline Operations Data ▾

↓ ↑

Keyspace

flights ▾

Tables

- routes
- airports
- airline

Aggregate view

Commit to Amazon Keyspaces

Commit to Apache Cassandra

[TABLE] routes Edit

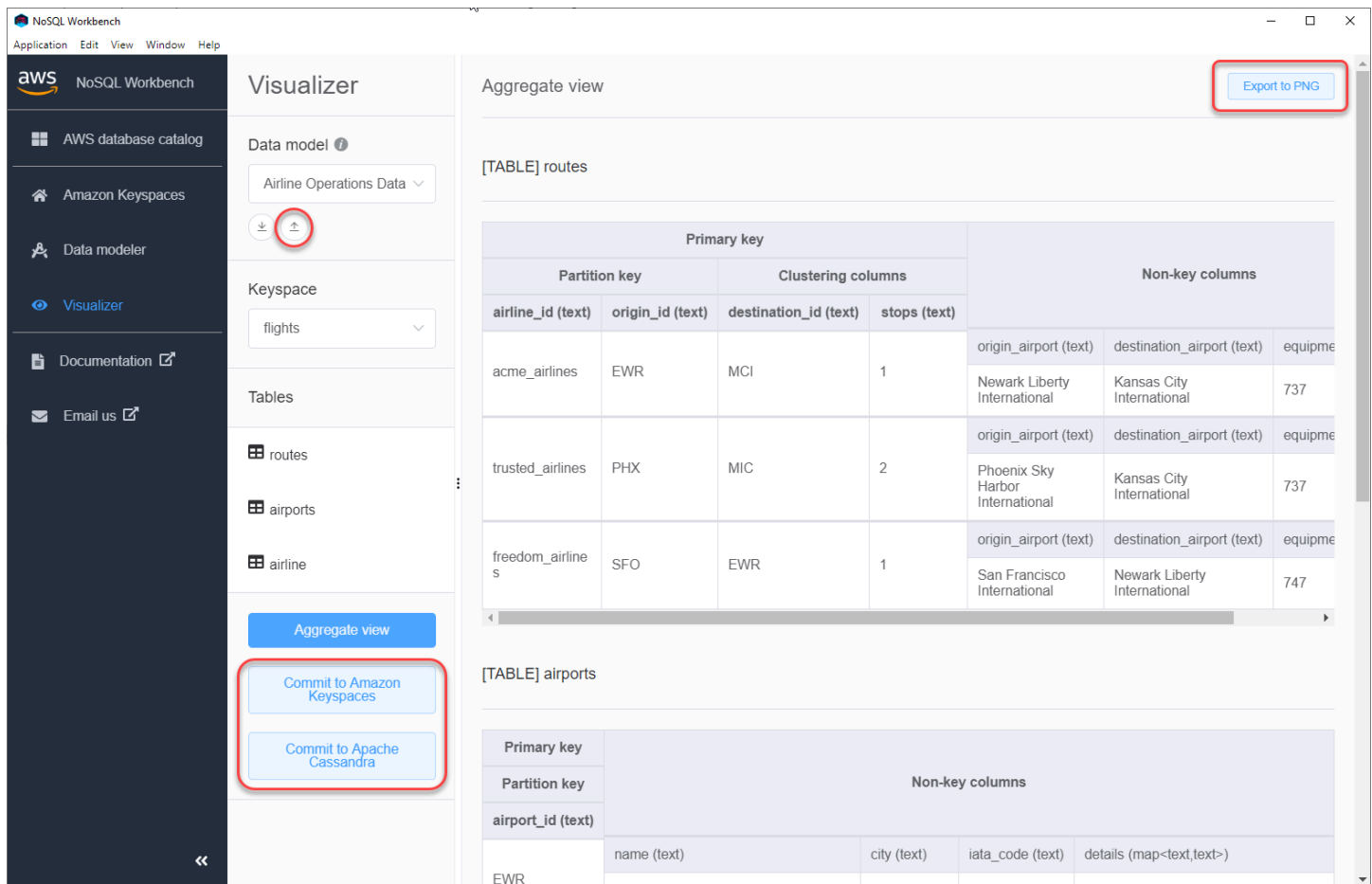
Use this view to get a visual representation of your tables schema, and add sample data. When you are done, commit your model to Amazon Keyspaces.

Primary key				Non-key columns		
Partition key		Clustering columns				
airline_id (text)	origin_id (text)	destination_id (text)	stops (text)	origin_airport (text)	destination_airport (text)	equipment
acme_airlines	EWR	MCI	1	Newark Liberty International	Kansas City International	737
trusted_airlines	PHX	MIC	2	Phoenix Sky Harbor International	Kansas City International	737
freedom_airlines	SFO	EWR	1	San Francisco International	Newark Liberty International	747

Après avoir agrégé la vue du modèle de données, vous pouvez l'exporter vers un fichier PNG. Pour exporter le modèle de données vers un fichier JSON, choisissez le signe de téléchargement sous le nom du modèle de données.

Note

Vous pouvez exporter le modèle de données au format JSON à tout moment au cours du processus de conception.



Les options suivantes s'offrent à vous pour valider les modifications :

- Adhérez à Amazon Keyspaces
- S'engager dans un cluster Apache Cassandra

Pour en savoir plus sur la façon de valider des modifications, consultez [the section called “Commettre un modèle de données”](#).

Créez un nouveau modèle de données avec NoSQL Workbench

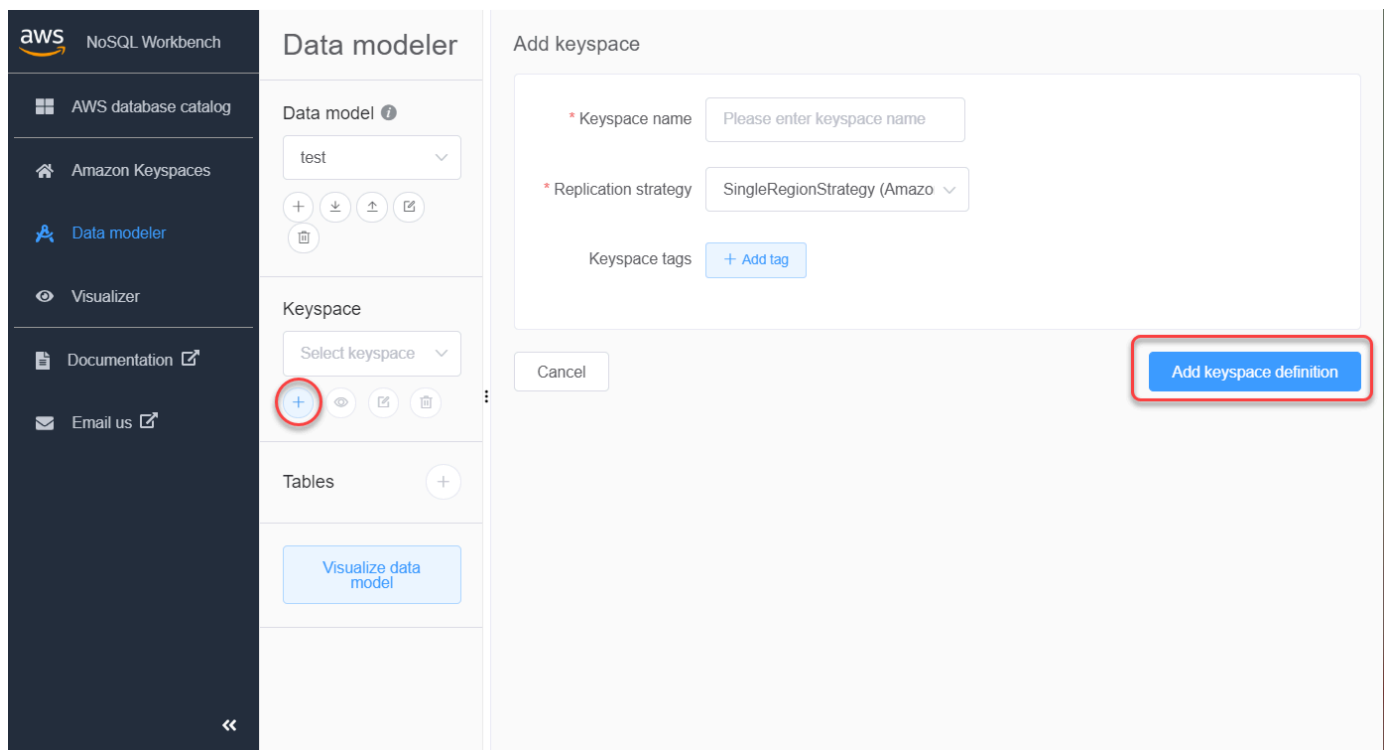
Vous pouvez utiliser le modèleur de données NoSQL Workbench pour concevoir de nouveaux modèles de données en fonction des modèles d'accès aux données de votre application. Pour créer un nouveau modèle de données pour Amazon Keyspaces, vous pouvez utiliser le modèleur de données NoSQL Workbench pour créer des espaces clés, des tableaux et des colonnes. Suivez ces étapes pour créer un nouveau modèle de données.

1. Pour créer un nouvel espace clé, choisissez le signe plus sous **Keyspace**.

Dans cette étape, choisissez les propriétés et paramètres suivants.

- Nom de l'espace clé — Entrez le nom du nouvel espace clé.
- Stratégie de réplication : choisissez la stratégie de réplication pour l'espace clé. Amazon Keyspaces les utilise `SingleRegionStrategy` pour répliquer les données trois fois automatiquement dans plusieurs zones de AWS disponibilité. Si vous envisagez de valider le modèle de données dans un cluster Apache Cassandra, vous pouvez choisir `SimpleStrategy` ou `NetworkTopologyStrategy`.
- Balises Keyspaces : les balises de ressources sont facultatives et vous permettent de classer vos ressources de différentes manières, par exemple par objectif, propriétaire, environnement ou d'autres critères. Pour en savoir plus sur les balises pour les ressources Amazon Keyspaces, consultez [the section called “Utilisation des tags”](#).

2. Choisissez Ajouter une définition d'espace clé pour créer l'espace clé.



3. Pour créer un nouveau tableau, cliquez sur le signe plus à côté de Tableaux. Au cours de cette étape, vous définissez les propriétés et paramètres suivants.

- Nom de la table : nom de la nouvelle table.
- Colonnes — Ajoutez un nom de colonne et choisissez le type de données. Répétez ces étapes pour chaque colonne de votre schéma.

- Clé de partition : choisissez les colonnes pour la clé de partition.
 - Colonnes de regroupement : choisissez des colonnes de regroupement (facultatif).
 - Mode de capacité — Choisissez le mode read/write de capacité pour le tableau. Vous pouvez choisir une capacité provisionnée ou à la demande. Pour en savoir plus sur les modes de capacité, consultez [the section called “Configuration des modes read/write de capacité”](#).
 - Balises de tableau : les balises de ressources sont facultatives et vous permettent de classer vos ressources de différentes manières, par exemple par objectif, propriétaire, environnement ou autres critères. Pour en savoir plus sur les balises pour les ressources Amazon Keyspaces, consultez [the section called “Utilisation des tags”](#).
4. Choisissez Ajouter une définition de table pour créer la nouvelle table.
 5. Répétez ces étapes pour créer des tableaux supplémentaires.
 6. Continuez [the section called “Visualisation d'un modèle de données”](#) à visualiser le modèle de données que vous avez créé.

The screenshot shows the AWS NoSQL Workbench interface. On the left is a dark sidebar with navigation links: AWS database catalog, Amazon Keyspaces, Data modeler (highlighted), Visualizer, Documentation, and Email us. The main area is titled 'Data modeler' and contains a 'Data model' section with a dropdown set to 'test', a 'Keyspace' section with a dropdown set to 'key1', and a 'Tables' section with a red box around a '+' button. Below the 'Tables' section is a 'Visualize data model' button. The right pane is titled 'Add table definition' and contains a form with the following fields: 'Keyspace' (dropdown set to 'key1'), '* Table name' (text input with placeholder 'Please enter table name'), '* Columns' (text input with placeholder 'Enter column name' and a 'Column type' dropdown), '+ Add another column' button, '* Partition key' (dropdown with placeholder 'Select columns to add to partition'), 'Clustering columns' (+ Add clustering column button), 'Capacity mode' (radio buttons for 'Provisioned' and 'On-demand', with 'On-demand' selected), and 'Table tags' (+ Add tag button). At the bottom of the form are 'Cancel' and 'Add table definition' buttons, with the latter highlighted by a red box.

Modifier les modèles de données existants avec NoSQL Workbench

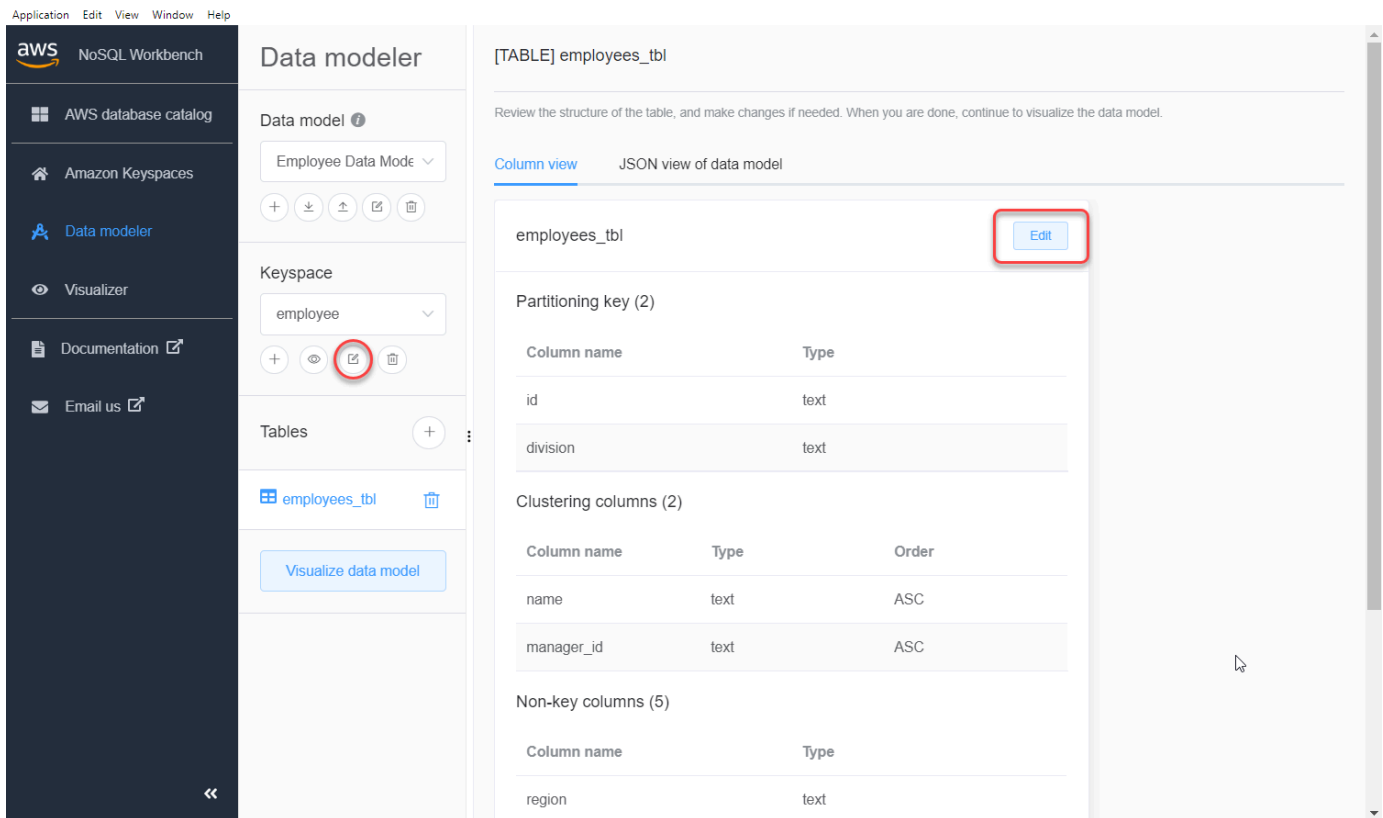
Vous pouvez utiliser le modélisateur de données pour importer et modifier des modèles de données existants créés à l'aide de NoSQL Workbench. Le modélisateur de données inclut également quelques exemples de modèles de données pour vous aider à démarrer avec la modélisation des données. Les modèles de données que vous pouvez modifier avec NoSQL Workbench peuvent être des modèles de données importés depuis un fichier, les exemples de modèles de données fournis ou des modèles de données que vous avez créés précédemment.

1. Pour modifier un espace clé, choisissez le symbole de modification sous Espace clé.

Au cours de cette étape, vous pouvez modifier les propriétés et paramètres suivants.

- Nom de l'espace clé — Entrez le nom du nouvel espace clé.
- Stratégie de réplication : choisissez la stratégie de réplication pour l'espace clé. Amazon Keyspaces les utilise `SingleRegionStrategy` pour répliquer les données trois fois automatiquement dans plusieurs zones de AWS disponibilité. Si vous envisagez de valider le modèle de données dans un cluster Apache Cassandra, vous pouvez choisir `SimpleStrategy` ou `NetworkTopologyStrategy`.
- Balises Keyspaces : les balises de ressources sont facultatives et vous permettent de classer vos ressources de différentes manières, par exemple par objectif, propriétaire, environnement ou d'autres critères. Pour en savoir plus sur les balises pour les ressources Amazon Keyspaces, consultez [the section called “Utilisation des tags”](#).

2. Choisissez Enregistrer les modifications pour mettre à jour l'espace clé.



3. Pour modifier un tableau, choisissez **Modifier** à côté du nom du tableau. Au cours de cette étape, vous pouvez mettre à jour les propriétés et paramètres suivants.
 - Nom de la table : nom de la nouvelle table.
 - Colonnes — Ajoutez un nom de colonne et choisissez le type de données. Répétez ces étapes pour chaque colonne de votre schéma.
 - Clé de partition : choisissez les colonnes pour la clé de partition.
 - Colonnes de regroupement : choisissez des colonnes de regroupement (facultatif).
 - Mode de capacité — Choisissez le mode read/write de capacité pour le tableau. Vous pouvez choisir une capacité provisionnée ou à la demande. Pour en savoir plus sur les modes de capacité, consultez [the section called “Configuration des modes read/write de capacité”](#).
 - Balises de tableau : les balises de ressources sont facultatives et vous permettent de classer vos ressources de différentes manières, par exemple par objectif, propriétaire, environnement ou autres critères. Pour en savoir plus sur les balises pour les ressources Amazon Keyspaces, consultez [the section called “Utilisation des tags”](#).
4. Choisissez **Enregistrer les modifications** pour mettre à jour le tableau.
5. Continuez [the section called “Visualisation d'un modèle de données”](#) à visualiser le modèle de données que vous avez mis à jour.

Comment valider des modèles de données dans Amazon Keyspaces et Apache Cassandra

Cette section explique comment valider des modèles de données complets dans les clusters Amazon Keyspaces et Apache Cassandra. Ce processus crée automatiquement les ressources côté serveur pour les espaces clés et les tables en fonction des paramètres que vous avez définis dans le modèle de données.

The screenshot shows the AWS NoSQL Workbench Visualizer interface. On the left, there is a sidebar with navigation options: AWS database catalog, Amazon Keyspaces, Data modeler, Visualizer (selected), Documentation, and Email us. The main area is titled 'Visualizer' and shows a 'Data model' for 'Airline Operations Data'. Below this, there are sections for 'Keyspace' (set to 'flights') and 'Tables' (listing 'routes', 'airports', and 'airline'). At the bottom of the sidebar, there are three buttons: 'Aggregate view', 'Commit to Amazon Keyspaces' (highlighted with a red box), and 'Commit to Apache Cassandra'. The main content area displays a table schema for 'routes' with columns: 'airline_id (text)', 'origin_id (text)', 'destination_id (text)', 'stops (text)', 'origin_airport (text)', 'destination_airport (text)', and 'equipment'. The table contains three rows of data.

Primary key				Non-key columns		
Partition key		Clustering columns				
airline_id (text)	origin_id (text)	destination_id (text)	stops (text)	origin_airport (text)	destination_airport (text)	equipment
acme_airlines	EWR	MCI	1	Newark Liberty International	Kansas City International	737
trusted_airlines	PHX	MIC	2	Phoenix Sky Harbor International	Kansas City International	737
freedom_airlines	SFO	EWR	1	San Francisco International	Newark Liberty International	747

Rubriques

- [Avant de commencer](#)
- [Connectez-vous à Amazon Keyspaces à l'aide d'informations d'identification spécifiques au service](#)
- [Connectez-vous à Amazon Keyspaces avec Gestion des identités et des accès AWS informations d'identification \(IAM\)](#)
- [Utiliser une connexion enregistrée](#)
- [Engagez-vous auprès d'Apache Cassandra](#)

Avant de commencer

Amazon Keyspaces nécessite l'utilisation du protocole TLS (Transport Layer Security) pour sécuriser les connexions avec les clients. Pour vous connecter à Amazon Keyspaces à l'aide du protocole TLS, vous devez effectuer la tâche suivante avant de commencer.

- Téléchargez les certificats numériques suivants et enregistrez les fichiers localement ou dans votre répertoire personnel.
 1. AmazonRootCA1
 2. AmazonRootCA2
 3. AmazonRootCA3
 4. AmazonRootCA4
 5. Starfield Class 2 Root (en option, pour des raisons de rétrocompatibilité)

Pour télécharger les certificats, vous pouvez utiliser les commandes suivantes.

```
curl -O https://www.amazontrust.com/repository/AmazonRootCA1.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA2.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA3.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA4.pem
curl -O https://certs.secureserver.net/repository/sf-class2-root.crt
```

Note

Amazon Keyspaces utilisait auparavant des certificats TLS ancrés dans le Starfield Class 2 CA. AWS migre tous Régions AWS vers des certificats émis dans le cadre d'Amazon Trust Services (Amazon Root CA 1 à 4). Pendant cette transition, configurez les clients pour qu'ils fassent confiance à la fois aux autorités de certification Amazon Root 1 à 4 et à la racine Starfield afin de garantir la compatibilité entre toutes les régions.

Combinez tous les certificats téléchargés dans un seul pem fichier avec le nom **keyspaces-bundle.pem** indiqué dans nos exemples. Pour ce faire, exécutez la commande suivante : Prenez note du chemin d'accès au fichier, vous en aurez besoin plus tard.

```
cat AmazonRootCA1.pem \
    AmazonRootCA2.pem \
    AmazonRootCA3.pem \
```

```
AmazonRootCA4.pem \  
sf-class2-root.crt \  
> keyspaces-bundle.pem
```

Après avoir enregistré le fichier de certificat, vous pouvez vous connecter à Amazon Keyspaces. L'une des options consiste à se connecter à l'aide d'informations d'identification spécifiques au service. Service-specific les informations d'identification sont un nom d'utilisateur et un mot de passe associés à un utilisateur IAM spécifique et ne peuvent être utilisés qu'avec le service spécifié. La deuxième option consiste à se connecter à l'aide d'informations d'identification IAM qui utilisent le processus [AWS Signature Version 4 \(Sigv4\)](#). Pour en savoir plus sur ces deux options, consultez [the section called “Création d'identifiants d'accès programmatiques”](#).

Pour vous connecter à l'aide d'informations d'identification spécifiques au service, consultez [the section called “Connectez-vous à l'aide d'informations d'identification spécifiques au service”](#)

Pour vous connecter à l'aide des informations d'identification IAM, consultez [the section called “Se connecter avec les informations d'identification IAM”](#).

Connectez-vous à Amazon Keyspaces à l'aide d'informations d'identification spécifiques au service

Cette section explique comment utiliser les informations d'identification spécifiques au service pour valider le modèle de données que vous avez créé ou modifié avec NoSQL Workbench.

1. Pour créer une nouvelle connexion à l'aide d'informations d'identification spécifiques au service, choisissez l'onglet Se connecter à l'aide d'un nom d'utilisateur et d'un mot de passe.
 - Avant de commencer, vous devez créer des informations d'identification spécifiques au service à l'aide du processus décrit à l'adresse [the section called “Créez des informations d'identification spécifiques au service”](#)

Après avoir obtenu les informations d'identification spécifiques au service, vous pouvez continuer à configurer la connexion. Continuez avec l'une des méthodes suivantes :

- Nom d'utilisateur — Entrez le nom d'utilisateur.
- Mot de passe — Entrez le mot de passe.
- Région AWS— Pour les régions disponibles, voir [the section called “Points de terminaison de service”](#).

- **Port** : Amazon Keyspaces utilise le port 9142.

Vous pouvez également importer des informations d'identification enregistrées à partir d'un fichier.

2. Choisissez **Commit** pour mettre à jour Amazon Keyspaces avec le modèle de données.

Commit to Amazon Keyspaces

 On this page, you can create server-side resources such as keyspaces and tables for the chosen data model.



< Use saved connections Connect by using IAM credentials Connect by using user name >



You can generate service-specific credentials to allow your users to access Amazon Keyspaces using AWS Management Console or AWS CLI.

[How to generate Amazon Keyspaces credentials](#)

* User Name

anika

* Password

.....



* AWS Region

us-east-1



* Port

9142

OR

 Import from credential file

Cancel

Reset

Commit

Connectez-vous à Amazon Keyspaces avec Gestion des identités et des accès AWS informations d'identification (IAM)

Cette section montre comment utiliser les informations d'identification IAM pour valider le modèle de données créé ou modifié avec NoSQL Workbench.

1. Pour créer une nouvelle connexion à l'aide des informations d'identification IAM, choisissez l'onglet **Se connecter à l'aide des informations d'identification IAM**.
 - Avant de commencer, vous devez créer des informations d'identification IAM à l'aide de l'une des méthodes suivantes.
 - Pour accéder à la console, utilisez votre nom d'utilisateur IAM et votre mot de passe pour vous connecter à la [Console de gestion AWS](#) depuis la page de connexion IAM. Pour plus d'informations sur les informations d'identification de AWS sécurité, y compris l'accès par programmation et les alternatives aux informations d'identification à long terme, consultez les informations d'identification [AWS de sécurité](#) dans le guide de l'utilisateur IAM. Pour plus de détails sur la connexion à votre Compte AWS compte, consultez la section [Comment vous connecter AWS](#) dans le Guide de Connexion à AWS l'utilisateur.
 - Pour accéder à la CLI, vous avez besoin d'un ID de clé d'accès et d'une clé d'accès secrète. Utilisation des informations d'identification temporaires au lieu des clés d'accès à long terme si possible. Les informations d'identification temporaires incluent un ID de clé d'accès, une clé d'accès secrète et un jeton de sécurité qui indique la date d'expiration des informations d'identification. Pour plus d'informations, consultez la section [Utilisation d'informations d'identification temporaires avec AWS des ressources](#) dans le Guide de l'utilisateur IAM.
 - Pour accéder à l'API, vous avez besoin d'un ID de clé d'accès et d'une clé d'accès secrète. Utilisez des clés d'accès utilisateur IAM plutôt que des clés d'accès Utilisateur racine d'un compte AWS . Pour plus d'informations sur la création de clés d'accès, consultez la section [Gérer les clés d'accès pour les utilisateurs IAM](#) dans le Guide de l'utilisateur IAM.

Pour plus d'informations, consultez [la section Gestion des clés d'accès pour les utilisateurs IAM](#).

Après avoir obtenu les informations d'identification IAM, vous pouvez continuer à configurer la connexion.

- Nom de la connexion : nom de la connexion.
 - Région AWS— Pour les régions disponibles, voir [the section called “Points de terminaison de service”](#).
 - ID de clé d'accès — Entrez l'ID de clé d'accès.
 - Clé d'accès secrète — Entrez la clé d'accès secrète.
 - Port : Amazon Keyspaces utilise le port 9142.
 - AWS certificat public : pointez sur le AWS certificat qui a été téléchargé lors de la première étape.
 - Persister la connexion : cochez cette case si vous souhaitez enregistrer les secrets de AWS connexion localement.
2. Choisissez `Commit` pour mettre à jour Amazon Keyspaces avec le modèle de données.

i On this page, you can create server-side resources such as keyspaces and tables for the chosen data model.

< Use saved connections Connect by using IAM credentials Connect by using user name >

* Connection name

Connection 2

* AWS Region

us-east-2

* Access key ID

AKIAIOSFODNN7EXAMPLE

* Secret access key

.....



* Port

9142

* AWS public certificate

⬇ Choose AWS public certificate

AmazonRootCA1.pem

Choose an AWS public certificate for a Signature Version 4–based connection to Amazon Keyspaces. **i**

Persist connection



If you select this check box, AWS connection secrets will be persisted in

C:\Users\awsuser\aws\credentials

Cancel

Reset

Commit

Utiliser une connexion enregistrée

Si vous avez déjà configuré une connexion à Amazon Keyspaces, vous pouvez l'utiliser comme connexion par défaut pour valider les modifications du modèle de données. Choisissez l'onglet Utiliser les connexions enregistrées et continuez à valider les mises à jour.

Commit to Amazon Keyspaces



On this page, you can create server-side resources such as keyspaces and tables for the chosen data model.

< **Use saved connections** Connect by using IAM credentials Connect by using user name : >

* Saved connections

default



* Port

9142

* AWS public certificate

[Choose AWS public certificate](#)

AmazonRootCA1.pem

Choose an AWS public certificate for a Signature Version 4–based connection to Amazon Keyspaces.

Cancel

Reset

Commit

Engagez-vous auprès d'Apache Cassandra

Cette section explique comment établir les connexions à un cluster Apache Cassandra pour valider le modèle de données créé ou modifié avec NoSQL Workbench.

 Note

Seuls les modèles de données qui ont été créés avec des clusters Apache Cassandra `SimpleStrategy` ou qui `NetworkTopologyStrategy` peuvent être validés dans ces clusters. Pour modifier la stratégie de réplication, modifiez l'espace clé dans le modélisateur de données.

1.
 - Nom d'utilisateur : entrez le nom d'utilisateur si l'authentification est activée sur le cluster.
 - Mot de passe : entrez le mot de passe si l'authentification est activée sur le cluster.
 - Points de contact — Entrez les points de contact.
 - Centre de données local — Entrez le nom du centre de données local.
 - Port : la connexion utilise le port 9042.
2. Choisissez `Commit` pour mettre à jour le cluster Apache Cassandra avec le modèle de données.

Commit to Apache Cassandra



Configure the connection to the Apache Cassandra cluster so that you can commit your data model to the database, including keyspaces, tables, and sample data. The user name and password are not required, and are necessary only if authentication is enabled on the cluster

User name

Password

* Contact points

[+ Add another contact point](#)

* Local data center

* Port

Cancel

Reset

Commit

Exemples de modèles de données dans NoSQL Workbench

La page d'accueil du modèleur et du visualiseur affiche un certain nombre d'exemples de modèles fournis avec NoSQL Workbench. Cette section décrit ces modèles et leurs utilisations potentielles.

Rubriques

- [Modèle de données d'employé](#)
- [Modèle de données sur les transactions par carte de crédit](#)
- [Modèle de données sur les opérations aériennes](#)

Modèle de données d'employé

Ce modèle de données représente un schéma Amazon Keyspaces pour une application de base de données d'employés.

Les applications qui accèdent aux informations sur les employés d'une entreprise donnée peuvent utiliser ce modèle de données.

Les modèles d'accès pris en charge par ce modèle de données sont les suivants :

- Récupération du dossier d'un employé avec un identifiant donné.
- Récupération du dossier d'un employé avec un identifiant et une division donnés.
- Récupération du dossier d'un employé avec un identifiant et un nom donnés.

Modèle de données sur les transactions par carte de crédit

Ce modèle de données représente un schéma Amazon Keyspaces pour les transactions par carte de crédit dans les magasins de détail.

Le stockage des transactions par carte de crédit aide non seulement les magasins à tenir leurs livres, mais aide également les directeurs de magasin à analyser les tendances d'achat, ce qui peut les aider à faire des prévisions et à planifier.

Les modèles d'accès pris en charge par ce modèle de données sont les suivants :

- Récupération des transactions par numéro de carte de crédit, mois, année et date.
- Récupération des transactions par numéro de carte de crédit, catégorie et date.
- Récupération des transactions par catégorie, lieu et numéro de carte de crédit.

- Récupération des transactions par numéro de carte de crédit et statut du litige.

Modèle de données sur les opérations aériennes

Ce modèle de données affiche des données sur les vols en avion, y compris les aéroports, les compagnies aériennes et les itinéraires aériens.

Les principaux composants de la modélisation Amazon Keyspaces présentés sont les paires clé-valeur, les banques de données à colonnes larges, les clés composites et les types de données complexes tels que des cartes illustrant les modèles d'accès aux données NoSQL courants.

Les modèles d'accès pris en charge par ce modèle de données sont les suivants :

- Récupération des itinéraires au départ d'une compagnie aérienne donnée dans un aéroport donné.
- Récupération des itinéraires avec un aéroport de destination donné.
- Récupération des aéroports proposant des vols directs.
- Récupération des informations relatives à l'aéroport et à la compagnie aérienne.

Historique de version pour NoSQL Workbench

Le tableau suivant décrit les modifications importantes apportées à chaque version de l'application côté client NoSQL Workbench.

Modifier	Description	Date
NoSQL Workbench pour Amazon Keyspaces — GA.	NoSQL Workbench pour Amazon Keyspaces est généralement disponible.	28 octobre 2020
Version préliminaire de NoSQL Workbench.	NoSQL Workbench est une application côté client qui vous permet de concevoir et de visualiser plus facilement des modèles de données non relationnels pour Amazon Keyspaces. Les clients NoSQL Workbench sont	5 octobre 2020

Modifier	Description	Date
	disponibles pour Windows, macOS et Linux. Pour plus d'informations, consultez NoSQL Workbench pour Amazon Keyspaces .	

Exemples de code pour Amazon Keyspaces à l'aide de SDK AWS

Les exemples de code suivants montrent comment utiliser Amazon Keyspaces avec un kit de développement AWS logiciel (SDK).

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [L'utilisation de ce service avec un AWS Kit SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit de développement logiciel (SDK).

Exemples de code

- [Exemples de base pour Amazon Keyspaces utilisant des SDK AWS](#)
 - [Bonjour Amazon Keyspaces](#)
 - [Découvrez les bases d'Amazon Keyspaces avec un SDK AWS](#)
 - [Actions pour Amazon Keyspaces à l'aide de kits SDK AWS](#)
 - [Utilisation CreateKeyspace avec un AWS SDK](#)
 - [Utilisation CreateTable avec un AWS SDK](#)
 - [Utilisation DeleteKeyspace avec un AWS SDK](#)
 - [Utilisation DeleteTable avec un AWS SDK](#)
 - [Utilisation GetKeyspace avec un AWS SDK](#)
 - [Utilisation GetTable avec un AWS SDK](#)
 - [Utilisation ListKeyspaces avec un AWS SDK](#)
 - [Utilisation ListTables avec un AWS SDK](#)
 - [Utilisation RestoreTable avec un AWS SDK](#)
 - [Utilisation UpdateTable avec un AWS SDK](#)

Exemples de base pour Amazon Keyspaces utilisant des SDK AWS

Les exemples de code suivants montrent comment utiliser les bases d'Amazon Keyspaces (pour Apache Cassandra) avec des kits AWS SDK.

Exemples

- [Bonjour Amazon Keyspaces](#)
- [Découvrez les bases d'Amazon Keyspaces avec un SDK AWS](#)
- [Actions pour Amazon Keyspaces à l'aide de kits SDK AWS](#)
 - [Utilisation CreateKeyspace avec un AWS SDK](#)
 - [Utilisation CreateTable avec un AWS SDK](#)
 - [Utilisation DeleteKeyspace avec un AWS SDK](#)
 - [Utilisation DeleteTable avec un AWS SDK](#)
 - [Utilisation GetKeyspace avec un AWS SDK](#)
 - [Utilisation GetTable avec un AWS SDK](#)
 - [Utilisation ListKeyspaces avec un AWS SDK](#)
 - [Utilisation ListTables avec un AWS SDK](#)
 - [Utilisation RestoreTable avec un AWS SDK](#)
 - [Utilisation UpdateTable avec un AWS SDK](#)

Bonjour Amazon Keyspaces

Les exemples de code suivants montrent comment démarrer avec Amazon Keyspaces.

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
namespace KeyspacesActions;

public class HelloKeyspaces
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Keyspaces (for Apache
        // Cassandra).
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
                        LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
                        LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonKeyspaces>()
                    .AddTransient<KeyspacesWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloKeyspaces>();

        var keyspacesClient =
            host.Services.GetRequiredService<IAmazonKeyspaces>();
        var keyspacesWrapper = new KeyspacesWrapper(keyspacesClient);

        Console.WriteLine("Hello, Amazon Keyspaces! Let's list your keyspaces:");
        await keyspacesWrapper.ListKeyspaces();
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListKeyspaces](#) à la section Référence des AWS SDK pour .NET API.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.keyspaces.KeyspacesClient;
import software.amazon.awssdk.services.keyspaces.model.KeyspaceSummary;
import software.amazon.awssdk.services.keyspaces.model.KeyspacesException;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesRequest;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesResponse;
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class HelloKeyspaces {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
            .region(region)
            .build();

        listKeyspaces(keyClient);
    }

    public static void listKeyspaces(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest =
                ListKeyspacesRequest.builder()
                    .maxResults(10)
```

```
        .build();

        ListKeyspacesResponse response =
keyClient.listKeyspaces(keyspacesRequest);
        List<KeyspaceSummary> keyspaces = response.keyspaces();
        for (KeyspaceSummary keyspace : keyspaces) {
            System.out.println("The name of the keyspace is " +
keyspace.keyspaceName());
        }

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListKeyspaces](#) à la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
Before running this Kotlin code example, set up your development environment,
including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
*/
```

```
suspend fun main() {
    listKeyspaces()
}

suspend fun listKeyspaces() {
    val keyspacesRequest =
        ListKeyspacesRequest {
            maxResults = 10
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.listKeyspaces(keyspacesRequest)
        response.keyspaces?.forEach { keyspace ->
            println("The name of the keyspace is ${keyspace.keyspaceName}")
        }
    }
}
```

- Pour plus de détails sur l'API, consultez [ListKeyspaces](#) la section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import boto3

def hello_keyspaces(keyspaces_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Keyspaces (for Apache
    Cassandra)
    client and list the keyspaces in your account.
    This example uses the default settings specified in your shared credentials
```

```
and config files.

:param keyspaces_client: A Boto3 Amazon Keyspaces Client object. This object
wraps
                                the low-level Amazon Keyspaces service API.
"""
print("Hello, Amazon Keyspaces! Let's list some of your keyspaces:\n")
for ks in keyspaces_client.list_keyspaces(maxResults=5).get("keyspaces", []):
    print(ks["keyspaceName"])
    print(f"\t{ks['resourceArn']}")

if __name__ == "__main__":
    hello_keyspaces(boto3.client("keyspaces"))
```

- Pour plus de détails sur l'API, consultez [ListKeyspaces](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [L'utilisation de ce service avec un AWS Kit SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Découvrez les bases d'Amazon Keyspaces avec un SDK AWS

Les exemples de code suivants montrent comment :

- créer un espace de clés et une table. Le schéma de table contient des données de film et la reprise ponctuelle est activée ;
- se connecter à l'espace de clés à l'aide d'une connexion TLS sécurisée avec authentification SigV4 ;
- interroger la table. Ajoutez, extrayez et mettez à jour les données des films ;
- mettre à jour la table. Ajoutez une colonne pour suivre les films visionnés ;
- restaurer l'état précédent de la table et nettoyer les ressources.

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
global using System.Security.Cryptography.X509Certificates;
global using Amazon.Keyspaces;
global using Amazon.Keyspaces.Model;
global using KeyspacesActions;
global using KeyspacesScenario;
global using Microsoft.Extensions.Configuration;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;
global using Newtonsoft.Json;

namespace KeyspacesBasics;

/// <summary>
/// Amazon Keyspaces (for Apache Cassandra) scenario. Shows some of the basic
/// actions performed with Amazon Keyspaces.
/// </summary>
public class KeyspacesBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
                        LogLevel.Information)
```

```

        .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
    .ConfigureServices((_, services) =>
services.AddAWSService<IAmazonKeyspaces>()
.AddTransient<KeyspacesWrapper>()
.AddTransient<CassandraWrapper>()
)
    .Build();

logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
    .CreateLogger<KeyspacesBasics>();

var configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from .json file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

var keyspacesWrapper =
host.Services.GetRequiredService<KeyspacesWrapper>();
var uiMethods = new UiMethods();

var keyspaceName = configuration["KeyspaceName"];
var tableName = configuration["TableName"];

bool success; // Used to track the results of some operations.

uiMethods.DisplayOverview();
uiMethods.PressEnter();

// Create the keyspace.
var keyspaceArn = await keyspacesWrapper.CreateKeyspace(keyspaceName);

// Wait for the keyspace to be available. GetKeyspace results in a
// resource not found error until it is ready for use.
try
{
    var getKeyspaceArn = "";
    Console.WriteLine($"Created {keyspaceName}. Waiting for it to become
available. ");
    do
    {

```

```
        getKeyspaceArn = await
keyspacesWrapper.GetKeyspace(keyspaceName);
        Console.WriteLine(". ");
    } while (getKeyspaceArn != keyspaceArn);
}
catch (ResourceNotFoundException)
{
    Console.WriteLine("Waiting for keyspace to be created.");
}

Console.WriteLine($"\\nThe keyspace {keyspaceName} is ready for use.");

uiMethods.PressEnter();

// Create the table.
// First define the schema.
var allColumns = new List<ColumnDefinition>
{
    new ColumnDefinition { Name = "title", Type = "text" },
    new ColumnDefinition { Name = "year", Type = "int" },
    new ColumnDefinition { Name = "release_date", Type = "timestamp" },
    new ColumnDefinition { Name = "plot", Type = "text" },
};

var partitionKeys = new List<PartitionKey>
{
    new PartitionKey { Name = "year", },
    new PartitionKey { Name = "title" },
};

var tableSchema = new SchemaDefinition
{
    AllColumns = allColumns,
    PartitionKeys = partitionKeys,
};

var tableArn = await keyspacesWrapper.CreateTable(keyspaceName,
tableSchema, tableName);

// Wait for the table to be active.
try
{
    var resp = new GetTableResponse();
    Console.WriteLine("Waiting for the new table to be active. ");
}
```

```
do
{
    try
    {
        resp = await keyspacesWrapper.GetTable(keyspaceName,
tableName);

        Console.WriteLine(".");
    }
    catch (ResourceNotFoundException)
    {
        Console.WriteLine(".");
    }
} while (resp.Status != TableStatus.ACTIVE);

// Display the table's schema.
Console.WriteLine($"\\nTable {tableName} has been created in
{keyspaceName}");
Console.WriteLine("Let's take a look at the schema.");
uiMethods.DisplayTitle("All columns");
resp.SchemaDefinition.AllColumns.ForEach(column =>
{
    Console.WriteLine($"{column.Name, -40}\\t{column.Type, -20}");
});

uiMethods.DisplayTitle("Cluster keys");
resp.SchemaDefinition.ClusteringKeys.ForEach(clusterKey =>
{
    Console.WriteLine($"{clusterKey.Name, -40}\\t{clusterKey.OrderBy, -20}");
});

uiMethods.DisplayTitle("Partition keys");
resp.SchemaDefinition.PartitionKeys.ForEach(partitionKey =>
{
    Console.WriteLine($"{partitionKey.Name}");
});

uiMethods.PressEnter();
}
catch (ResourceNotFoundException ex)
{
    Console.WriteLine($"Error: {ex.Message}");
}
```



```
// Access Apache Cassandra using the Cassandra drive for C#.
var cassandraWrapper =
host.Services.GetRequiredService<CassandraWrapper>();
var movieFilePath = configuration["MovieFile"];

Console.WriteLine("Let's add some movies to the table we created.");
var inserted = await cassandraWrapper.InsertIntoMovieTable(keyspaceName,
tableName, movieFilePath);

uiMethods.PressEnter();

Console.WriteLine("Added the following movies to the table:");
var rows = await cassandraWrapper.GetMovies(keyspaceName, tableName);
uiMethods.DisplayTitle("All Movies");

foreach (var row in rows)
{
    var title = row.GetValue<string>("title");
    var year = row.GetValue<int>("year");
    var plot = row.GetValue<string>("plot");
    var release_date = row.GetValue<DateTime>("release_date");
    Console.WriteLine($"{release_date}\t\t{title}\t\t{year}\n{plot}");
    Console.WriteLine(uiMethods.SepBar);
}

// Update the table schema
uiMethods.DisplayTitle("Update table schema");
Console.WriteLine("Now we will update the table to add a boolean field
called watched.");

// First save the current time as a UTC Date so the original
// table can be restored later.
var timeChanged = DateTime.UtcNow;

// Now update the schema.
var resourceArn = await keyspacesWrapper.UpdateTable(keyspaceName,
tableName);
uiMethods.PressEnter();

Console.WriteLine("Now let's mark some of the movies as watched.");

// Pick some files to mark as watched.
var movieToWatch = rows[2].GetValue<string>("title");
var watchedMovieYear = rows[2].GetValue<int>("year");
```

```

        var changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
        tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[6].GetValue<string>("title");
        watchedMovieYear = rows[6].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
        tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[9].GetValue<string>("title");
        watchedMovieYear = rows[9].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
        tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[10].GetValue<string>("title");
        watchedMovieYear = rows[10].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
        tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[13].GetValue<string>("title");
        watchedMovieYear = rows[13].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
        tableName, movieToWatch, watchedMovieYear);

        uiMethods.DisplayTitle("Watched movies");
        Console.WriteLine("These movies have been marked as watched:");
        rows = await cassandraWrapper.GetWatchedMovies(keyspaceName, tableName);
        foreach (var row in rows)
        {
            var title = row.GetValue<string>("title");
            var year = row.GetValue<int>("year");
            Console.WriteLine($"{title,-40}\t{year,8}");
        }
        uiMethods.PressEnter();

        Console.WriteLine("We can restore the table to its previous state but
        that can take up to 20 minutes to complete.");
        string answer;
        do
        {
            Console.WriteLine("Do you want to restore the table? (y/n)");
            answer = Console.ReadLine();
        } while (answer.ToLower() != "y" && answer.ToLower() != "n");

        if (answer == "y")

```

```

    {
        var restoredTableName = $"{tableName}_restored";
        var restoredTableArn = await keyspacesWrapper.RestoreTable(
            keyspaceName,
            tableName,
            restoredTableName,
            timeChanged);
        // Loop and call GetTable until the table is gone. Once it has been
        // deleted completely, GetTable will raise a
ResourceNotFoundException.
        bool wasRestored = false;

        try
        {
            do
            {
                var resp = await keyspacesWrapper.GetTable(keyspaceName,
restoredTableName);
                wasRestored = (resp.Status == TableStatus.ACTIVE);
            } while (!wasRestored);
        }
        catch (ResourceNotFoundException)
        {
            // If the restored table raised an error, it isn't
            // ready yet.
            Console.WriteLine(".");
        }
    }

    uiMethods.DisplayTitle("Clean up resources.");

    // Delete the table.
    success = await keyspacesWrapper.DeleteTable(keyspaceName, tableName);

    Console.WriteLine($"Table {tableName} successfully deleted from
{keyspaceName}.");
    Console.WriteLine("Waiting for the table to be removed completely. ");

    // Loop and call GetTable until the table is gone. Once it has been
    // deleted completely, GetTable will raise a ResourceNotFoundException.
    bool wasDeleted = false;

    try
    {

```

```

        do
        {
            var resp = await keyspacesWrapper.GetTable(keyspaceName,
tableName);
            } while (!wasDeleted);
        }
        catch (ResourceNotFoundException ex)
        {
            wasDeleted = true;
            Console.WriteLine($"{ex.Message} indicates that the table has been
deleted.");
        }

        // Delete the keyspace.
        success = await keyspacesWrapper.DeleteKeyspace(keyspaceName);
        Console.WriteLine("The keyspace has been deleted and the demo is now
complete.");
    }
}

```

```

namespace KeyspacesActions;

/// <summary>
/// Performs Amazon Keyspaces (for Apache Cassandra) actions.
/// </summary>
public class KeyspacesWrapper
{
    private readonly IAmazonKeyspaces _amazonKeyspaces;

    /// <summary>
    /// Constructor for the KeyspaceWrapper.
    /// </summary>
    /// <param name="amazonKeyspaces">An Amazon Keyspaces client object.</param>
    public KeyspacesWrapper(IAmazonKeyspaces amazonKeyspaces)
    {
        _amazonKeyspaces = amazonKeyspaces;
    }

    /// <summary>
    /// Create a new keyspace.
    /// </summary>

```

```

    /// <param name="keyspaceName">The name for the new keyspace.</param>
    /// <returns>The Amazon Resource Name (ARN) of the new keyspace.</returns>
    public async Task<string> CreateKeyspace(string keyspaceName)
    {
        var response =
            await _amazonKeyspaces.CreateKeyspaceAsync(
                new CreateKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.ResourceArn;
    }

    /// <summary>
    /// Create a new Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace where the table will be
    created.</param>
    /// <param name="schema">The schema for the new table.</param>
    /// <param name="tableName">The name of the new table.</param>
    /// <returns>The Amazon Resource Name (ARN) of the new table.</returns>
    public async Task<string> CreateTable(string keyspaceName, SchemaDefinition
    schema, string tableName)
    {
        var request = new CreateTableRequest
        {
            KeyspaceName = keyspaceName,
            SchemaDefinition = schema,
            TableName = tableName,
            PointInTimeRecovery = new PointInTimeRecovery { Status =
    PointInTimeRecoveryStatus.ENABLED };
        };

        var response = await _amazonKeyspaces.CreateTableAsync(request);
        return response.ResourceArn;
    }

    /// <summary>
    /// Delete an existing keyspace.
    /// </summary>
    /// <param name="keyspaceName"></param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteKeyspace(string keyspaceName)
    {
        var response = await _amazonKeyspaces.DeleteKeyspaceAsync(

```

```

        new DeleteKeyspaceRequest { KeyspaceName = keySpaceName });
    return response.HttpStatusCode == HttpStatusCode.OK;
}

/// <summary>
/// Delete an Amazon Keyspaces table.
/// </summary>
/// <param name="keySpaceName">The keyspace containing the table.</param>
/// <param name="tableName">The name of the table to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteTable(string keySpaceName, string tableName)
{
    var response = await _amazonKeyspaces.DeleteTableAsync(
        new DeleteTableRequest { KeyspaceName = keySpaceName, TableName =
tableName });
    return response.HttpStatusCode == HttpStatusCode.OK;
}

/// <summary>
/// Get data about a keyspace.
/// </summary>
/// <param name="keySpaceName">The name of the keyspace.</param>
/// <returns>The Amazon Resource Name (ARN) of the keyspace.</returns>
public async Task<string> GetKeyspace(string keySpaceName)
{
    var response = await _amazonKeyspaces.GetKeyspaceAsync(
        new GetKeyspaceRequest { KeyspaceName = keySpaceName });
    return response.ResourceArn;
}

/// <summary>
/// Get information about an Amazon Keyspaces table.
/// </summary>
/// <param name="keySpaceName">The keyspace containing the table.</param>
/// <param name="tableName">The name of the Amazon Keyspaces table.</param>
/// <returns>The response containing data about the table.</returns>
public async Task<GetTableResponse> GetTable(string keySpaceName, string
tableName)
{
    var response = await _amazonKeyspaces.GetTableAsync(

```

```

        new GetTableRequest { KeyspaceName = keySpaceName, TableName =
tableName });
        return response;
    }

    /// <summary>
    /// Lists all keyspaces for the account.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task ListKeyspaces()
    {
        var paginator = _amazonKeyspaces.Paginators.ListKeyspaces(new
ListKeyspacesRequest());

        Console.WriteLine("{0, -30}\t{1}", "Keyspace name", "Keyspace ARN");
        Console.WriteLine(new string('-', Console.WindowWidth));
        await foreach (var keySpace in paginator.Keyspaces)
        {

Console.WriteLine($"{keySpace.KeyspaceName, -30}\t{keySpace.ResourceArn}");
        }
    }

    /// <summary>
    /// Lists the Amazon Keyspaces tables in a keySpace.
    /// </summary>
    /// <param name="keySpaceName">The name of the keySpace.</param>
    /// <returns>A list of TableSummary objects.</returns>
    public async Task<List<TableSummary>> ListTables(string keySpaceName)
    {
        var response = await _amazonKeyspaces.ListTablesAsync(new
ListTablesRequest { KeyspaceName = keySpaceName });
        response.Tables.ForEach(table =>
        {

Console.WriteLine($"{table.KeyspaceName}\t{table.TableName}\t{table.ResourceArn}");
        });

        return response.Tables;
    }

```

```

    /// <summary>
    /// Restores the specified table to the specified point in time.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to restore.</param>
    /// <param name="timestamp">The time to which the table will be restored.</
param>
    /// <returns>The Amazon Resource Name (ARN) of the restored table.</returns>
    public async Task<string> RestoreTable(string keyspaceName, string tableName,
string restoredTableName, DateTime timestamp)
    {
        var request = new RestoreTableRequest
        {
            RestoreTimestamp = timestamp,
            SourceKeyspaceName = keyspaceName,
            SourceTableName = tableName,
            TargetKeyspaceName = keyspaceName,
            TargetTableName = restoredTableName
        };

        var response = await _amazonKeyspaces.RestoreTableAsync(request);
        return response.RestoredTableARN;
    }

    /// <summary>
    /// Updates the movie table to add a boolean column named watched.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to change.</param>
    /// <returns>The Amazon Resource Name (ARN) of the updated table.</returns>
    public async Task<string> UpdateTable(string keyspaceName, string tableName)
    {
        var newColumn = new ColumnDefinition { Name = "watched", Type =
"boolean" };
        var request = new UpdateTableRequest
        {
            KeyspaceName = keyspaceName,
            TableName = tableName,
            AddColumns = new List<ColumnDefinition> { newColumn }
        };
        var response = await _amazonKeyspaces.UpdateTableAsync(request);
        return response.ResourceArn;
    }

```



```
}
```

```
using System.Net;
using Cassandra;

namespace KeyspacesScenario;

/// <summary>
/// Class to perform CRUD methods on an Amazon Keyspaces (for Apache Cassandra)
/// database.
///
/// NOTE: This sample uses a plain text authenticator for example purposes only.
/// Recommended best practice is to use a SigV4 authentication plugin, if
/// available.
/// </summary>
public class CassandraWrapper
{
    private readonly IConfiguration _configuration;
    private readonly string _localPathToFile;
    private const string _certLocation = "https://certs.secureserver.net/
repository/sf-class2-root.crt";
    private const string _certFileName = "sf-class2-root.crt";
    private readonly X509Certificate2Collection _certCollection;
    private X509Certificate2 _amazoncert;
    private Cluster _cluster;

    // User name and password for the service.
    private string _userName = null!;
    private string _pwd = null!;

    public CassandraWrapper()
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        _localPathToFile = Path.GetTempPath();
    }
}
```

```

        // Get the Starfield digital certificate and save it locally.
        var client = new WebClient();
        client.DownloadFile(_certLocation, $"{_localPathToFile}/
{_certFileName}");

        //var httpClient = new HttpClient();
        //var httpResult = httpClient.Get(fileUrl);
        //using var resultStream = await httpResult.Content.ReadAsStreamAsync();
        //using var fileStream = File.Create(pathToSave);
        //resultStream.CopyTo(fileStream);

        _certCollection = new X509Certificate2Collection();
        _amazoncert = new X509Certificate2($"{_localPathToFile}/
{_certFileName}");

        // Get the user name and password stored in the configuration file.
        _userName = _configuration["UserName"]!;
        _pwd = _configuration["Password"]!;

        // For a list of Service Endpoints for Amazon Keyspaces, see:
        // https://docs.aws.amazon.com/keyspaces/latest/devguide/
        programmatic.endpoints.html
        var awsEndpoint = _configuration["ServiceEndpoint"];

        _cluster = Cluster.Builder()
            .AddContactPoints(awsEndpoint)
            .WithPort(9142)
            .WithAuthProvider(new PlainTextAuthProvider(_userName, _pwd))
            .WithSSL(new SSLOptions().SetCertificateCollection(_certCollection))
            .WithQueryOptions(
                new QueryOptions()
                    .SetConsistencyLevel(ConsistencyLevel.LocalQuorum)
                    .SetSerialConsistencyLevel(ConsistencyLevel.LocalSerial))
            .Build();
    }

    /// <summary>
    /// Loads the contents of a JSON file into a list of movies to be
    /// added to the Apache Cassandra table.
    /// </summary>
    /// <param name="movieFileName">The full path to the JSON file.</param>
    /// <returns>A list of movie objects.</returns>

```

```

    public List<Movie> ImportMoviesFromJson(string movieFileName, int numToImport
= 0)
    {
        if (!File.Exists(movieFileName))
        {
            return null!;
        }

        using var sr = new StreamReader(movieFileName);
        string json = sr.ReadToEnd();

        var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

        // If numToImport = 0, return all movies in the collection.
        if (numToImport == 0)
        {
            // Now return the entire list of movies.
            return allMovies;
        }
        else
        {
            // Now return the first numToImport entries.
            return allMovies.GetRange(0, numToImport);
        }
    }

    /// <summary>
    /// Insert movies into the movie table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="movieTableName">The Amazon Keyspaces table.</param>
    /// <param name="movieFilePath">The path to the resource file containing
    /// movie data to insert into the table.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> InsertIntoMovieTable(string keyspaceName, string
movieTableName, string movieFilePath, int numToImport = 20)
    {
        // Get some movie data from the movies.json file
        var movies = ImportMoviesFromJson(movieFilePath, numToImport);

        var session = _cluster.Connect(keyspaceName);

        string insertCql;

```

```

        RowSet rs;

        // Now we insert the numToImport movies into the table.
        foreach (var movie in movies)
        {
            // Escape single quote characters in the plot.
            insertCql = $"INSERT INTO {keyspaceName}.{movieTableName}
(title, year, release_date, plot) values($${movie.Title}$$, {movie.Year},
'{movie.Info.Release_Date.ToString("yyyy-MM-dd")}', $${movie.Info.Plot}$$)";
            rs = await session.ExecuteAsync(new SimpleStatement(insertCql));
        }

        return true;
    }

    /// <summary>
    /// Gets all of the movies in the movies table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <returns>A list of row objects containing movie data.</returns>
    public async Task<List<Row>> GetMovies(string keyspaceName, string tableName)
    {
        var session = _cluster.Connect();
        RowSet rs;
        try
        {
            rs = await session.ExecuteAsync(new SimpleStatement($"SELECT * FROM
{keyspaceName}.{tableName}"));

            // Extract the row data from the returned RowSet.
            var rows = rs.GetRows().ToList();
            return rows;
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            return null!;
        }
    }

    /// <summary>
    /// Mark a movie in the movie table as watched.
    /// </summary>

```

```

    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <param name="title">The title of the movie to mark as watched.</param>
    /// <param name="year">The year the movie was released.</param>
    /// <returns>A set of rows containing the changed data.</returns>
    public async Task<List<Row>> MarkMovieAsWatched(string keyspaceName, string
tableName, string title, int year)
    {
        var session = _cluster.Connect();
        string updateCql = $"UPDATE {keyspaceName}.{tableName} SET watched=true
WHERE title = ${title} AND year = {year}";
        var rs = await session.ExecuteAsync(new SimpleStatement(updateCql));
        var rows = rs.GetRows().ToList();
        return rows;
    }

    /// <summary>
    /// Retrieve the movies in the movies table where watched is true.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <returns>A list of row objects containing information about movies
    /// where watched is true.</returns>
    public async Task<List<Row>> GetWatchedMovies(string keyspaceName, string
tableName)
    {
        var session = _cluster.Connect();
        RowSet rs;
        try
        {
            rs = await session.ExecuteAsync(new SimpleStatement($"SELECT
title, year, plot FROM {keyspaceName}.{tableName} WHERE watched = true ALLOW
FILTERING"));

            // Extract the row data from the returned RowSet.
            var rows = rs.GetRows().ToList();
            return rows;
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            return null!;
        }
    }
}

```

```
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour .NET .
 - [CreateKeyspace](#)
 - [CreateTable](#)
 - [DeleteKeyspace](#)
 - [DeleteTable](#)
 - [GetKeyspace](#)
 - [GetTable](#)
 - [ListKeyspaces](#)
 - [ListTables](#)
 - [RestoreTable](#)
 - [UpdateTable](#)

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 *
 * Before running this Java code example, you must create a
```

```

* Java keystore (JKS) file and place it in your project's resources folder.
*
* This file is a secure file format used to hold certificate information for
* Java applications. This is required to make a connection to Amazon Keyspaces.
* For more information, see the following documentation topic:
*
* https://docs.aws.amazon.com/keyspaces/latest/devguide/using\_java\_driver.html
*
* This Java example performs the following tasks:
*
* 1. Create a keyspace.
* 2. Check for keyspace existence.
* 3. List keyspaces using a paginator.
* 4. Create a table with a simple movie data schema and enable point-in-time
* recovery.
* 5. Check for the table to be in an Active state.
* 6. List all tables in the keyspace.
* 7. Use a Cassandra driver to insert some records into the Movie table.
* 8. Get all records from the Movie table.
* 9. Get a specific Movie.
* 10. Get a UTC timestamp for the current time.
* 11. Update the table schema to add a 'watched' Boolean column.
* 12. Update an item as watched.
* 13. Query for items with watched = True.
* 14. Restore the table back to the previous state using the timestamp.
* 15. Check for completion of the restore action.
* 16. Delete the table.
* 17. Confirm that both tables are deleted.
* 18. Delete the keyspace.
*/

```

```

public class ScenarioKeyspaces {
    public static final String DASHES = new String(new char[80]).replace("\0",
"-");

    /*
     * Usage:
     * fileName - The name of the JSON file that contains movie data. (Get this
file
     * from the GitHub repo at resources/sample_file.)
     * keyspaceName - The name of the keyspace to create.
     */
    public static void main(String[] args) throws InterruptedException,
IOException {

```

```
String fileName = "<Replace with the JSON file that contains movie
data>";
String keyspaceName = "<Replace with the name of the keyspace to
create>";
String titleUpdate = "The Family";
int yearUpdate = 2013;
String tableName = "Movie";
String tableNameRestore = "MovieRestore";
Region region = Region.US_EAST_1;
KeyspacesClient keyClient = KeyspacesClient.builder()
    .region(region)
    .build();

DriverConfigLoader loader =
DriverConfigLoader.fromClasspath("application.conf");
CqlSession session = CqlSession.builder()
    .withConfigLoader(loader)
    .build();

System.out.println(DASHES);
System.out.println("Welcome to the Amazon Keyspaces example scenario.");
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("1. Create a keyspace.");
createKeySpace(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
Thread.sleep(5000);
System.out.println("2. Check for keyspace existence.");
checkKeyspaceExistence(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("3. List keyspaces using a paginator.");
listKeyspacesPaginator(keyClient);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("4. Create a table with a simple movie data schema and
enable point-in-time recovery.");
createTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);
```



```
System.out.println(DASHES);
System.out.println("5. Check for the table to be in an Active state.");
Thread.sleep(6000);
checkTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. List all tables in the keyspace.");
listTables(keyClient, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Use a Cassandra driver to insert some records into
the Movie table.");
Thread.sleep(6000);
loadData(session, fileName, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("8. Get all records from the Movie table.");
getMovieData(session, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("9. Get a specific Movie.");
getSpecificMovie(session, keyspaceName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("10. Get a UTC timestamp for the current time.");
ZonedDateTime utc = ZonedDateTime.now(ZoneOffset.UTC);
System.out.println("DATETIME = " + Date.from(utc.toInstant()));
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("11. Update the table schema to add a watched Boolean
column.");
updateTable(keyClient, keyspaceName, tableName);
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("12. Update an item as watched.");
Thread.sleep(10000); // Wait 10 secs for the update.
```

```
        updateRecord(session, keyspaceName, titleUpdate, yearUpdate);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Query for items with watched = True.");
        getWatchedData(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Restore the table back to the previous state
using the timestamp.");
        System.out.println("Note that the restore operation can take up to 20
minutes.");
        restoreTable(keyClient, keyspaceName, utc);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Check for completion of the restore action.");
        Thread.sleep(5000);
        checkRestoredTable(keyClient, keyspaceName, "MovieRestore");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete both tables.");
        deleteTable(keyClient, keyspaceName, tableName);
        deleteTable(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Confirm that both tables are deleted.");
        checkTableDelete(keyClient, keyspaceName, tableName);
        checkTableDelete(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Delete the keyspace.");
        deleteKeyspace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The scenario has completed successfully.");
        System.out.println(DASHES);
    }
```

```
public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                        .keyspaceName(keyspaceName)
                        .build();

        keyClient.deleteKeyspace(deleteKeyspaceRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkTableDelete(KeyspacesClient keyClient, String
keyspaceName, String tableName)
    throws InterruptedException {
    try {
        String status;
        GetTableResponse response;
        GetTableRequest tableRequest = GetTableRequest.builder()
                .keyspaceName(keyspaceName)
                .tableName(tableName)
                .build();

        // Keep looping until table cannot be found and a
ResourceNotFoundException is
        // thrown.
        while (true) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println(". The table status is " + status);
            Thread.sleep(500);
        }

    } catch (ResourceNotFoundException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
    System.out.println("The table is deleted");
}
```

```
public static void deleteTable(KeyspacesClient keyClient, String
keyspaceName, String tableName) {
    try {
        DeleteTableRequest tableRequest = DeleteTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        keyClient.deleteTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkRestoredTable(KeyspacesClient keyClient, String
keyspaceName, String tableName)
    throws InterruptedException {
    try {
        boolean tableStatus = false;
        String status;
        GetTableResponse response = null;
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        while (!tableStatus) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println("The table status is " + status);

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true;
            }
            Thread.sleep(500);
        }

        List<ColumnDefinition> cols =
response.schemaDefinition().allColumns();
        for (ColumnDefinition def : cols) {
            System.out.println("The column name is " + def.name());
            System.out.println("The column type is " + def.type());
        }
    }
}
```

```

    }

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void restoreTable(KeyspacesClient keyClient, String
keyspaceName, ZonedDateTime utc) {
    try {
        Instant myTime = utc.toInstant();
        RestoreTableRequest restoreTableRequest =
RestoreTableRequest.builder()
            .restoreTimestamp(myTime)
            .sourceTableName("Movie")
            .targetKeyspaceName(keyspaceName)
            .targetTableName("MovieRestore")
            .sourceKeyspaceName(keyspaceName)
            .build();

        RestoreTableResponse response =
keyClient.restoreTable(restoreTableRequest);
        System.out.println("The ARN of the restored table is " +
response.restoredTableARN());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void getWatchedData(CqlSession session, String keyspaceName) {
    ResultSet resultSet = session
        .execute("SELECT * FROM \"\" + keyspaceName + "\".\"Movie\" WHERE
watched = true ALLOW FILTERING;");
    resultSet.forEach(item -> {
        System.out.println("The Movie title is " + item.getString("title"));
        System.out.println("The Movie year is " + item.getInt("year"));
        System.out.println("The plot is " + item.getString("plot"));
    });
}
}

```

```

    public static void updateRecord(CqlSession session, String keySpace, String
titleUpdate, int yearUpdate) {
        String sqlStatement = "UPDATE \"" + keySpace
            + "\".\"Movie\" SET watched=true WHERE title = :k0 AND year
= :k1;";

        BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
        PreparedStatement preparedStatement = session.prepare(sqlStatement);
        builder.addStatement(preparedStatement.boundStatementBuilder()
            .setString("k0", titleUpdate)
            .setInt("k1", yearUpdate)
            .build());

        BatchStatement batchStatement = builder.build();
        session.execute(batchStatement);
    }

    public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
        try {
            ColumnDefinition def = ColumnDefinition.builder()
                .name("watched")
                .type("boolean")
                .build();

            UpdateTableRequest tableRequest = UpdateTableRequest.builder()
                .keyspaceName(keySpace)
                .tableName(tableName)
                .addColumnns(def)
                .build();

            keyClient.updateTable(tableRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getSpecificMovie(CqlSession session, String keyspaceName)
{
        ResultSet resultSet = session.execute(

```

```

        "SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE title =
        'The Family' ALLOW FILTERING ;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    // Get records from the Movie table.
    public static void getMovieData(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session.execute("SELECT * FROM \"" + keyspaceName +
        "\".\"Movie\";");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    // Load data into the table.
    public static void loadData(CqlSession session, String fileName, String
    keySpace) throws IOException {
        String sqlStatement = "INSERT INTO \"" + keySpace + "\".\"Movie\" (title,
        year, plot) values (:k0, :k1, :k2)";
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
        ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {

            // Add 20 movies to the table.
            if (t == 20)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String plot = currentNode.path("info").path("plot").toString();

            // Insert the data into the Amazon Keyspaces table.

```

```

        BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
        PreparedStatement preparedStatement = session.prepare(sqlStatement);
        builder.addStatement(preparedStatement.boundStatementBuilder()
            .setString("k0", title)
            .setInt("k1", year)
            .setString("k2", plot)
            .build());

        BatchStatement batchStatement = builder.build();
        session.execute(batchStatement);
        t++;
    }

    System.out.println("You have added " + t + " records successfully!");
}

public static void listTables(KeyspacesClient keyClient, String keyspaceName)
{
    try {
        ListTablesRequest tablesRequest = ListTablesRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
        listRes.stream()
            .flatMap(r -> r.tables().stream())
            .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                " Table name: " + content.tableName()));

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
    throws InterruptedException {
    try {
        boolean tableStatus = false;

```



```
String status;
GetTableResponse response = null;
GetTableRequest tableRequest = GetTableRequest.builder()
    .keyspaceName(keyspaceName)
    .tableName(tableName)
    .build();

while (!tableStatus) {
    response = keyClient.getTable(tableRequest);
    status = response.statusAsString();
    System.out.println(". The table status is " + status);

    if (status.compareTo("ACTIVE") == 0) {
        tableStatus = true;
    }
    Thread.sleep(500);
}

List<ColumnDefinition> cols =
response.schemaDefinition().allColumns();
for (ColumnDefinition def : cols) {
    System.out.println("The column name is " + def.name());
    System.out.println("The column type is " + def.type());
}

} catch (KeyspacesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

}

public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        // Set the columns.
        ColumnDefinition defTitle = ColumnDefinition.builder()
            .name("title")
            .type("text")
            .build();

        ColumnDefinition defYear = ColumnDefinition.builder()
            .name("year")
            .type("int")
            .build();
```

```
ColumnDefinition defReleaseDate = ColumnDefinition.builder()
    .name("release_date")
    .type("timestamp")
    .build();

ColumnDefinition defPlot = ColumnDefinition.builder()
    .name("plot")
    .type("text")
    .build();

List<ColumnDefinition> collist = new ArrayList<>();
collist.add(defTitle);
collist.add(defYear);
collist.add(defReleaseDate);
collist.add(defPlot);

// Set the keys.
PartitionKey yearKey = PartitionKey.builder()
    .name("year")
    .build();

PartitionKey titleKey = PartitionKey.builder()
    .name("title")
    .build();

List<PartitionKey> keyList = new ArrayList<>();
keyList.add(yearKey);
keyList.add(titleKey);

SchemaDefinition schemaDefinition = SchemaDefinition.builder()
    .partitionKeys(keyList)
    .allColumns(collist)
    .build();

PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
    .status(PointInTimeRecoveryStatus.ENABLED)
    .build();

CreateTableRequest tableRequest = CreateTableRequest.builder()
    .keyspaceName(keySpace)
    .tableName(tableName)
    .schemaDefinition(schemaDefinition)
    .pointInTimeRecovery(timeRecovery)
```

```
        .build();

        CreateTableResponse response = keyClient.createTable(tableRequest);
        System.out.println("The table ARN is " + response.resourceArn());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
    try {
        ListKeyspacesRequest keyspacesRequest =
ListKeyspacesRequest.builder()
        .maxResults(10)
        .build();

        ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
        listRes.stream()
            .flatMap(r -> r.keyspaces().stream())
            .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        GetKeyspaceResponse response =
keyClient.getKeyspace(keyspaceRequest);
        String name = response.keyspaceName();
        System.out.println("The " + name + " KeySpace is ready");

    } catch (KeyspacesException e) {
```

```
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        CreateKeyspaceRequest keyspaceRequest =
CreateKeyspaceRequest.builder()
                        .keyspaceName(keyspaceName)
                        .build();

        CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
        System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Java 2.x .
 - [CreateKeyspace](#)
 - [CreateTable](#)
 - [DeleteKeyspace](#)
 - [DeleteTable](#)
 - [GetKeyspace](#)
 - [GetTable](#)
 - [ListKeyspaces](#)
 - [ListTables](#)
 - [RestoreTable](#)
 - [UpdateTable](#)

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

`/**`

Before running this Kotlin code example, set up your development environment, including your credentials.

For more information, see the following documentation topic:

<https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html>

This example uses a secure file format to hold certificate information for Kotlin applications. This is required to make a connection to Amazon Keyspaces. For more information, see the following documentation topic:

https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html

This Kotlin example performs the following tasks:

1. Create a keyspace.
2. Check for keyspace existence.
3. List keyspaces using a paginator.
4. Create a table with a simple movie data schema and enable point-in-time recovery.
5. Check for the table to be in an Active state.
6. List all tables in the keyspace.
7. Use a Cassandra driver to insert some records into the Movie table.
8. Get all records from the Movie table.
9. Get a specific Movie.
10. Get a UTC timestamp for the current time.
11. Update the table schema to add a 'watched' Boolean column.
12. Update an item as watched.
13. Query for items with watched = True.
14. Restore the table back to the previous state using the timestamp.

```

15. Check for completion of the restore action.
16. Delete the table.
17. Confirm that both tables are deleted.
18. Delete the keyspace.
*/

/*
Usage:
    fileName - The name of the JSON file that contains movie data. (Get this
file from the GitHub repo at resources/sample_file.)
    keyspaceName - The name of the keyspace to create.
*/
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main() {
    val fileName = "<Replace with the JSON file that contains movie data>"
    val keyspaceName = "<Replace with the name of the keyspace to create>"
    val titleUpdate = "The Family"
    val yearUpdate = 2013
    val tableName = "MovieKotlin"
    val tableNameRestore = "MovieRestore"

    val loader = DriverConfigLoader.fromClasspath("application.conf")
    val session =
        CqlSession
            .builder()
            .withConfigLoader(loader)
            .build()

    println(DASHES)
    println("Welcome to the Amazon Keyspaces example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create a keyspace.")
    createKeySpace(keyspaceName)
    println(DASHES)

    println(DASHES)
    delay(5000)
    println("2. Check for keyspace existence.")
    checkKeyspaceExistence(keyspaceName)
    println(DASHES)

```

```
println(DASHES)
println("3. List keyspaces using a paginator.")
listKeyspacesPaginator()
println(DASHES)

println(DASHES)
println("4. Create a table with a simple movie data schema and enable point-
in-time recovery.")
createTable(keyspaceName, tableName)
println(DASHES)

println(DASHES)
println("5. Check for the table to be in an Active state.")
delay(6000)
checkTable(keyspaceName, tableName)
println(DASHES)

println(DASHES)
println("6. List all tables in the keyspace.")
listTables(keyspaceName)
println(DASHES)

println(DASHES)
println("7. Use a Cassandra driver to insert some records into the Movie
table.")
delay(6000)
loadData(session, fileName, keyspaceName)
println(DASHES)

println(DASHES)
println("8. Get all records from the Movie table.")
getMovieData(session, keyspaceName)
println(DASHES)

println(DASHES)
println("9. Get a specific Movie.")
getSpecificMovie(session, keyspaceName)
println(DASHES)

println(DASHES)
println("10. Get a UTC timestamp for the current time.")
val utc = ZonedDateTime.now(ZoneOffset.UTC)
println("DATETIME = ${Date.from(utc.toInstant())}")
println(DASHES)
```

```
println(DASHES)
println("11. Update the table schema to add a watched Boolean column.")
updateTable(keyspaceName, tableName)
println(DASHES)

println(DASHES)
println("12. Update an item as watched.")
delay(10000) // Wait 10 seconds for the update.
updateRecord(session, keyspaceName, titleUpdate, yearUpdate)
println(DASHES)

println(DASHES)
println("13. Query for items with watched = True.")
getWatchedData(session, keyspaceName)
println(DASHES)

println(DASHES)
println("14. Restore the table back to the previous state using the
timestamp.")
println("Note that the restore operation can take up to 20 minutes.")
restoreTable(keyspaceName, utc)
println(DASHES)

println(DASHES)
println("15. Check for completion of the restore action.")
delay(5000)
checkRestoredTable(keyspaceName, "MovieRestore")
println(DASHES)

println(DASHES)
println("16. Delete both tables.")
deleteTable(keyspaceName, tableName)
deleteTable(keyspaceName, tableNameRestore)
println(DASHES)

println(DASHES)
println("17. Confirm that both tables are deleted.")
checkTableDelete(keyspaceName, tableName)
checkTableDelete(keyspaceName, tableNameRestore)
println(DASHES)

println(DASHES)
println("18. Delete the keyspace.")
```



```
deleteKeyspace(keyspaceName)
println(DASHES)

println(DASHES)
println("The scenario has completed successfully.")
println(DASHES)
}

suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}

suspend fun checkTableDelete(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var status: String
    var response: GetTableResponse
    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    try {
        KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient
->
            // Keep looping until the table cannot be found and a
ResourceNotFoundException is thrown.
            while (true) {
                response = keyClient.getTable(tableRequest)
                status = response.status.toString()
                println(". The table status is $status")
                delay(500)
            }
        }
    } catch (e: ResourceNotFoundException) {
```

```
        println(e.message)
    }
    println("The table is deleted")
}

suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}

suspend fun checkRestoredTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println("The table status is $status")

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
    }
}
```

```

    }

    val cols = response!!.schemaDefinition?.allColumns
    if (cols != null) {
        for (def in cols) {
            println("The column name is ${def.name}")
            println("The column type is ${def.type}")
        }
    }
}

suspend fun restoreTable(
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}

fun getWatchedData(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"${keyspaceName}\".
    \"MovieKotlin\" WHERE watched = true ALLOW FILTERING;")
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
    }
}

```

```
        println("The plot is ${item.getString("plot")}")
    }
}

fun updateRecord(
    session: CqlSession,
    keySpace: String,
    titleUpdate: String?,
    yearUpdate: Int,
) {
    val sqlStatement =
        "UPDATE \"\$keySpace\".\"MovieKotlin\" SET watched=true WHERE title = :k0
AND year = :k1;"
    val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
    builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
    val preparedStatement = session.prepare(sqlStatement)
    builder.addStatement(
        preparedStatement
            .boundStatementBuilder()
            .setString("k0", titleUpdate)
            .setInt("k1", yearUpdate)
            .build(),
    )
    val batchStatement = builder.build()
    session.execute(batchStatement)
}

suspend fun updateTable(
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }
}
```

```

        KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
            keyClient.updateTable(tableRequest)
        }
    }

fun getSpecificMovie(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet =
        session.execute("SELECT * FROM \"\$keyspaceName\".\"MovieKotlin\" WHERE
title = 'The Family' ALLOW FILTERING ;")

    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Get records from the Movie table.
fun getMovieData(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"\$keyspaceName\".
\"MovieKotlin\";")
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Load data into the table.
fun loadData(
    session: CqlSession,
    fileName: String,
    keySpace: String,
) {
    val sqlStatement =
        "INSERT INTO \"\$keySpace\".\"MovieKotlin\" (title, year, plot) values
(:k0, :k1, :k2)"
    val parser = JsonFactory().createParser(File(fileName))

```

```

val rootNode = ObjectMapper().readTree<JsonNode>(parser)
val iter: Iterator<JsonNode> = rootNode.iterator()
var currentNode: ObjectNode

var t = 0
while (iter.hasNext()) {
    if (t == 50) {
        break
    }

    currentNode = iter.next() as ObjectNode
    val year = currentNode.path("year").asInt()
    val title = currentNode.path("title").asText()
    val info = currentNode.path("info").toString()

    // Insert the data into the Amazon Keyspaces table.
    val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
    builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
    val preparedStatement: PreparedStatement = session.prepare(sqlStatement)
    builder.addStatement(
        preparedStatement
            .boundStatementBuilder()
            .setString("k0", title)
            .setInt("k1", year)
            .setString("k2", info)
            .build(),
    )

    val batchStatement = builder.build()
    session.execute(batchStatement)
    t++
}

suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
    }
}

```

```

        .collect { obj ->
            println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
        }
    }
}

suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
        val cols: List<ColumnDefinition>? =
            response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}

suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {

```

```
// Set the columns.
val defTitle =
    ColumnDefinition {
        name = "title"
        type = "text"
    }

val defYear =
    ColumnDefinition {
        name = "year"
        type = "int"
    }

val defReleaseDate =
    ColumnDefinition {
        name = "release_date"
        type = "timestamp"
    }

val defPlot =
    ColumnDefinition {
        name = "plot"
        type = "text"
    }

val collist = ArrayList<ColumnDefinition>()
collist.add(defTitle)
collist.add(defYear)
collist.add(defReleaseDate)
collist.add(defPlot)

// Set the keys.
val yearKey =
    PartitionKey {
        name = "year"
    }

val titleKey =
    PartitionKey {
        name = "title"
    }

val keyList = ArrayList<PartitionKey>()
keyList.add(yearKey)
```



```
keyList.add(titleKey)

val schemaDefinition0b =
    SchemaDefinition {
        partitionKeys = keyList
        allColumns = colList
    }

val timeRecovery =
    PointInTimeRecovery {
        status = PointInTimeRecoveryStatus.Enabled
    }

val tableRequest =
    CreateTableRequest {
        keyspaceName = keySpaceVal
        tableName = tableNameVal
        schemaDefinition = schemaDefinition0b
        pointInTimeRecovery = timeRecovery
    }

KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
    val response = keyClient.createTable(tableRequest)
    println("The table ARN is ${response.resourceArn}")
}

suspend fun listKeyspacesPaginator() {
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
}

suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
```

```
        val response: GetKeyspaceResponse =
            keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}

suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour Kotlin.
 - [CreateKeyspace](#)
 - [CreateTable](#)
 - [DeleteKeyspace](#)
 - [DeleteTable](#)
 - [GetKeyspace](#)
 - [GetTable](#)
 - [ListKeyspaces](#)
 - [ListTables](#)
 - [RestoreTable](#)
 - [UpdateTable](#)

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif à une invite de commande.

```
class KeyspaceScenario:
    """Runs an interactive scenario that shows how to get started using Amazon
    Keyspaces."""

    def __init__(self, ks_wrapper):
        """
        :param ks_wrapper: An object that wraps Amazon Keyspace actions.
        """
        self.ks_wrapper = ks_wrapper

    @demo_func
    def create_keyspace(self):
        """
        1. Creates a keyspace.
        2. Lists up to 10 keyspace in your account.
        """
        print("Let's create a keyspace.")
        ks_name = q.ask(
            "Enter a name for your new keyspace.\nThe name can contain only
letters, "
            "numbers and underscores: ",
            q.non_empty,
        )
        if self.ks_wrapper.exists_keyspace(ks_name):
            print(f"A keyspace named {ks_name} exists.")
        else:
            ks_arn = self.ks_wrapper.create_keyspace(ks_name)
            ks_exists = False
            while not ks_exists:
                wait(3)
                ks_exists = self.ks_wrapper.exists_keyspace(ks_name)
```

```

        print(f"Created a new keyspace.\n\t{ks_arn}.")
    print("The first 10 keyspace in your account are:\n")
    self.ks_wrapper.list_keyspaces(10)

    @demo_func
    def create_table(self):
        """
        1. Creates a table in the keyspace. The table is configured with a schema
to hold
        movie data and has point-in-time recovery enabled.
        2. Waits for the table to be in an active state.
        3. Displays schema information for the table.
        4. Lists tables in the keyspace.
        """
        print("Let's create a table for movies in your keyspace.")
        table_name = q.ask("Enter a name for your table: ", q.non_empty)
        table = self.ks_wrapper.get_table(table_name)
        if table is not None:
            print(
                f"A table named {table_name} already exists in keyspace "
                f"{self.ks_wrapper.ks_name}."
            )
        else:
            table_arn = self.ks_wrapper.create_table(table_name)
            print(f"Created table {table_name}:\n\t{table_arn}")
            table = {"status": None}
            print("Waiting for your table to be ready...")
            while table["status"] != "ACTIVE":
                wait(5)
                table = self.ks_wrapper.get_table(table_name)
            print(f"Your table is {table['status']}. Its schema is:")
            pp(table["schemaDefinition"])
            print("\nThe tables in your keyspace are:\n")
            self.ks_wrapper.list_tables()

    @demo_func
    def ensure_tls_cert(self):
        """
        Ensures you have a TLS certificate available to use to secure the
connection
        to the keyspace. This function downloads a default certificate or lets
you
        specify your own.
        """

```

```

    print("To connect to your keyspace, you must have a TLS certificate.")
    print("Checking for TLS certificate...")
    cert_path = os.path.join(
        os.path.dirname(__file__), QueryManager.DEFAULT_CERT_FILE
    )
    if not os.path.exists(cert_path):
        cert_choice = q.ask(
            f"Press enter to download a certificate from
{QueryManager.CERT_URL} "
            f"or enter the full path to the certificate you want to use: "
        )
        if cert_choice:
            cert_path = cert_choice
        else:
            cert = requests.get(QueryManager.CERT_URL).text
            with open(cert_path, "w") as cert_file:
                cert_file.write(cert)
    else:
        q.ask(f"Certificate {cert_path} found. Press Enter to continue.")
    print(
        f"Certificate {cert_path} will be used to secure the connection to
your keyspace."
    )
    return cert_path

@demo_func
def query_table(self, qm, movie_file):
    """
    1. Adds movies to the table from a sample movie data file.
    2. Gets a list of movies from the table and lets you select one.
    3. Displays more information about the selected movie.
    """
    qm.add_movies(self.ks_wrapper.table_name, movie_file)
    movies = qm.get_movies(self.ks_wrapper.table_name)
    print(f"Added {len(movies)} movies to the table:")
    sel = q.choose("Pick one to learn more about it: ", [m.title for m in
movies])
    movie_choice = qm.get_movie(
        self.ks_wrapper.table_name, movies[sel].title, movies[sel].year
    )
    print(movie_choice.title)
    print(f"\tReleased: {movie_choice.release_date}")
    print(f"\tPlot: {movie_choice.plot}")

```

```

@demo_func
def update_and_restore_table(self, qm):
    """
    1. Updates the table by adding a column to track watched movies.
    2. Marks some of the movies as watched.
    3. Gets the list of watched movies from the table.
    4. Restores to a movies_restored table at a previous point in time.
    5. Gets the list of movies from the restored table.
    """
    print("Let's add a column to record which movies you've watched.")
    pre_update_timestamp = datetime.utcnow()
    print(
        f"Recorded the current UTC time of {pre_update_timestamp} so we can
restore the table later."
    )
    self.ks_wrapper.update_table()
    print("Waiting for your table to update...")
    table = {"status": "UPDATING"}
    while table["status"] != "ACTIVE":
        wait(5)
        table = self.ks_wrapper.get_table(self.ks_wrapper.table_name)
    print("Column 'watched' added to table.")
    q.ask(
        "Let's mark some of the movies as watched. Press Enter when you're
ready.\n"
    )
    movies = qm.get_movies(self.ks_wrapper.table_name)
    for movie in movies[:10]:
        qm.watched_movie(self.ks_wrapper.table_name, movie.title, movie.year)
        print(f"Marked {movie.title} as watched.")
    movies = qm.get_movies(self.ks_wrapper.table_name, watched=True)
    print("-" * 88)
    print("The watched movies in our table are:\n")
    for movie in movies:
        print(movie.title)
    print("-" * 88)
    if q.ask(
        "Do you want to restore the table to the way it was before all of
these\n"
        "updates? Keep in mind, this can take up to 20 minutes. (y/n) ",
        q.is_yesno,
    ):
        starting_table_name = self.ks_wrapper.table_name

```

```

        table_name_restored =
self.ks_wrapper.restore_table(pre_update_timestamp)
        table = {"status": "RESTORING"}
        while table["status"] != "ACTIVE":
            wait(10)
            table = self.ks_wrapper.get_table(table_name_restored)
        print(
            f"Restored {starting_table_name} to {table_name_restored} "
            f"at a point in time of {pre_update_timestamp}."
        )
        movies = qm.get_movies(table_name_restored)
        print("Now the movies in our table are:")
        for movie in movies:
            print(movie.title)

def cleanup(self, cert_path):
    """
    1. Deletes the table and waits for it to be removed.
    2. Deletes the keyspace.

    :param cert_path: The path of the TLS certificate used in the demo. If
the
                        certificate was downloaded during the demo, it is
removed.
    """
    if q.ask(
        f"Do you want to delete your {self.ks_wrapper.table_name} table and "
        f"{self.ks_wrapper.ks_name} keyspace? (y/n) ",
        q.is_yesno,
    ):
        table_name = self.ks_wrapper.table_name
        self.ks_wrapper.delete_table()
        table = self.ks_wrapper.get_table(table_name)
        print("Waiting for the table to be deleted.")
        while table is not None:
            wait(5)
            table = self.ks_wrapper.get_table(table_name)
        print("Table deleted.")
        self.ks_wrapper.delete_keyspace()
        print(
            "Keyspace deleted. If you chose to restore your table during the
"
            "demo, the original table is also deleted."
        )

```

```

        if cert_path == os.path.join(
            os.path.dirname(__file__), QueryManager.DEFAULT_CERT_FILE
        ) and os.path.exists(cert_path):
            os.remove(cert_path)
            print("Removed certificate that was downloaded for this demo.")

    def run_scenario(self):
        logging.basicConfig(level=logging.INFO, format="%(levelname)s:
%(message)s")

        print("-" * 88)
        print("Welcome to the Amazon Keyspaces (for Apache Cassandra) demo.")
        print("-" * 88)

        self.create_keyspace()
        self.create_table()
        cert_file_path = self.ensure_tls_cert()
        # Use a context manager to ensure the connection to the keyspace is
        closed.
        with QueryManager(
            cert_file_path, boto3.DEFAULT_SESSION, self.ks_wrapper.ks_name
        ) as qm:
            self.query_table(qm, "../resources/sample_files/movies.json")
            self.update_and_restore_table(qm)
            self.cleanup(cert_file_path)

        print("\nThanks for watching!")
        print("-" * 88)

if __name__ == "__main__":
    try:
        scenario = KeyspaceScenario(KeyspaceWrapper.from_client())
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")

```

Définissez une classe qui encapsule l'espace de clés et les actions de la table.

```

class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""

```



```
def __init__(self, keyspaces_client):
    """
    :param keyspaces_client: A Boto3 Amazon Keyspaces client.
    """
    self.keyspaces_client = keyspaces_client
    self.ks_name = None
    self.ks_arn = None
    self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)

def create_keyspace(self, name):
    """
    Creates a keyspace.

    :param name: The name to give the keyspace.
    :return: The Amazon Resource Name (ARN) of the new keyspace.
    """
    try:
        response = self.keyspaces_client.create_keyspace(keyspaceName=name)
        self.ks_name = name
        self.ks_arn = response["resourceArn"]
    except ClientError as err:
        logger.error(
            "Couldn't create %s. Here's why: %s: %s",
            name,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
    else:
        return self.ks_arn

def exists_keyspace(self, name):
    """
    Checks whether a keyspace exists.

    :param name: The name of the keyspace to look up.
```

```

        :return: True when the keyspace exists. Otherwise, False.
        """
    try:
        response = self.keyspaces_client.get_keyspace(keyspaceName=name)
        self.ks_name = response["keyspaceName"]
        self.ks_arn = response["resourceArn"]
        exists = True
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.info("Keyspace %s does not exist.", name)
            exists = False
        else:
            logger.error(
                "Couldn't verify %s exists. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
    return exists

def list_keyspaces(self, limit):
    """
    Lists the keyspaces in your account.

    :param limit: The maximum number of keyspaces to list.
    """
    try:
        ks_paginator = self.keyspaces_client.get_paginator("list_keyspaces")
        for page in ks_paginator.paginate(PaginationConfig={"MaxItems":
limit}):
            for ks in page["keyspaces"]:
                print(ks["keyspaceName"])
                print(f"\t{ks['resourceArn']}")
    except ClientError as err:
        logger.error(
            "Couldn't list keyspaces. Here's why: %s: %s",
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise

```

```

def create_table(self, table_name):
    """
    Creates a table in the keyspace.
    The table is created with a schema for storing movie data
    and has point-in-time recovery enabled.

    :param table_name: The name to give the table.
    :return: The ARN of the new table.
    """
    try:
        response = self.keyspaces_client.create_table(
            keyspaceName=self.ks_name,
            tableName=table_name,
            schemaDefinition={
                "allColumns": [
                    {"name": "title", "type": "text"},
                    {"name": "year", "type": "int"},
                    {"name": "release_date", "type": "timestamp"},
                    {"name": "plot", "type": "text"},
                ],
                "partitionKeys": [{"name": "year"}, {"name": "title"}],
            },
            pointInTimeRecovery={"status": "ENABLED"},
        )
    except ClientError as err:
        logger.error(
            "Couldn't create table %s. Here's why: %s: %s",
            table_name,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
    else:
        return response["resourceArn"]

def get_table(self, table_name):
    """
    Gets data about a table in the keyspace.

    :param table_name: The name of the table to look up.
    :return: Data about the table.
    """
    try:

```

```
        response = self.keyspaces_client.get_table(
            keyspaceName=self.ks_name, tableName=table_name
        )
        self.table_name = table_name
    except ClientError as err:
        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.info("Table %s does not exist.", table_name)
            self.table_name = None
            response = None
        else:
            logger.error(
                "Couldn't verify %s exists. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
    return response

def list_tables(self):
    """
    Lists the tables in the keyspace.
    """
    try:
        table_paginator = self.keyspaces_client.get_paginator("list_tables")
        for page in table_paginator.paginate(keyspaceName=self.ks_name):
            for table in page["tables"]:
                print(table["tableName"])
                print(f"\t{table['resourceArn']}")
    except ClientError as err:
        logger.error(
            "Couldn't list tables in keyspace %s. Here's why: %s: %s",
            self.ks_name,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise

def update_table(self):
    """
    Updates the schema of the table.
```

This example updates a table of movie data by adding a new column that tracks whether the movie has been watched.

```

"""
try:
    self.keyspaces_client.update_table(
        keyspaceName=self.ks_name,
        tableName=self.table_name,
        addColumns=[{"name": "watched", "type": "boolean"}],
    )
except ClientError as err:
    logger.error(
        "Couldn't update table %s. Here's why: %s: %s",
        self.table_name,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise

```

```

def restore_table(self, restore_timestamp):

```

```

    """

```

Restores the table to a previous point in time. The table is restored to a new table in the same keyspace.

:param restore_timestamp: The point in time to restore the table. This
time must be in UTC format.

:return: The name of the restored table.

```

    """

```

```

try:
    restored_table_name = f"{self.table_name}_restored"
    self.keyspaces_client.restore_table(
        sourceKeyspaceName=self.ks_name,
        sourceTableName=self.table_name,
        targetKeyspaceName=self.ks_name,
        targetTableName=restored_table_name,
        restoreTimestamp=restore_timestamp,
    )
except ClientError as err:
    logger.error(
        "Couldn't restore table %s. Here's why: %s: %s",
        restore_timestamp,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )

```

```
        )
        raise
    else:
        return restored_table_name

def delete_table(self):
    """
    Deletes the table from the keyspace.
    """
    try:
        self.keyspaces_client.delete_table(
            keyspaceName=self.ks_name, tableName=self.table_name
        )
        self.table_name = None
    except ClientError as err:
        logger.error(
            "Couldn't delete table %s. Here's why: %s: %s",
            self.table_name,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise

def delete_keyspace(self):
    """
    Deletes the keyspace.
    """
    try:
        self.keyspaces_client.delete_keyspace(keyspaceName=self.ks_name)
        self.ks_name = None
    except ClientError as err:
        logger.error(
            "Couldn't delete keyspace %s. Here's why: %s: %s",
            self.ks_name,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

Définissez une classe qui crée une connexion TLS à un espace de clés, s'authentifie avec SigV4 et envoie des requêtes CQL à une table de l'espace de clés.

```
class QueryManager:
    """
    Manages queries to an Amazon Keyspaces (for Apache Cassandra) keyspace.
    Queries are secured by TLS and authenticated by using the Signature V4
    (SigV4)
    AWS signing protocol. This is more secure than sending username and password
    with a plain-text authentication provider.

    This example downloads a default certificate to secure TLS, or lets you
    specify
    your own.

    This example uses a table of movie data to demonstrate basic queries.
    """

    DEFAULT_CERT_FILE = "sf-class2-root.crt"
    CERT_URL = f"https://certs.secureserver.net/repository/sf-class2-root.crt"

    def __init__(self, cert_file_path, boto_session, keyspace_name):
        """
        :param cert_file_path: The path and file name of the certificate used for
        TLS.
        :param boto_session: A Boto3 session. This is used to acquire your AWS
        credentials.
        :param keyspace_name: The name of the keyspace to connect.
        """
        self.cert_file_path = cert_file_path
        self.boto_session = boto_session
        self.ks_name = keyspace_name
        self.cluster = None
        self.session = None

    def __enter__(self):
        """
        Creates a session connection to the keyspace that is secured by TLS and
        authenticated by SigV4.
        """
        ssl_context = SSLContext(PROTOCOL_TLSv1_2)
```

```

        ssl_context.load_verify_locations(self.cert_file_path)
        ssl_context.verify_mode = CERT_REQUIRED
        auth_provider = SigV4AuthProvider(self.boto_session)
        contact_point = f"cassandra.
{self.boto_session.region_name}.amazonaws.com"
        exec_profile = ExecutionProfile(
            consistency_level=ConsistencyLevel.LOCAL_QUORUM,
            load_balancing_policy=DCAwareRoundRobinPolicy(),
        )
        self.cluster = Cluster(
            [contact_point],
            ssl_context=ssl_context,
            auth_provider=auth_provider,
            port=9142,
            execution_profiles={EXEC_PROFILE_DEFAULT: exec_profile},
            protocol_version=4,
        )
        self.cluster.__enter__()
        self.session = self.cluster.connect(self.ks_name)
        return self

def __exit__(self, *args):
    """
    Exits the cluster. This shuts down all existing session connections.
    """
    self.cluster.__exit__(*args)

def add_movies(self, table_name, movie_file_path):
    """
    Gets movies from a JSON file and adds them to a table in the keyspace.

    :param table_name: The name of the table.
    :param movie_file_path: The path and file name of a JSON file that
contains movie data.
    """
    with open(movie_file_path, "r") as movie_file:
        movies = json.loads(movie_file.read())
    stmt = self.session.prepare(
        f"INSERT INTO {table_name} (year, title, release_date, plot) VALUES
(?, ?, ?, ?);"
    )
    for movie in movies[:20]:
        self.session.execute(
            stmt,

```



```

        parameters=[
            movie["year"],
            movie["title"],
            date.fromisoformat(movie["info"]
["release_date"].partition("T")[0]),
            movie["info"]["plot"],
        ],
    )

def get_movies(self, table_name, watched=None):
    """
    Gets the title and year of the full list of movies from the table.

    :param table_name: The name of the movie table.
    :param watched: When specified, the returned list of movies is filtered
to
                    either movies that have been watched or movies that have
not
                    been watched. Otherwise, all movies are returned.
    :return: A list of movies in the table.
    """
    if watched is None:
        stmt = SimpleStatement(f"SELECT title, year from {table_name}")
        params = None
    else:
        stmt = SimpleStatement(
            f"SELECT title, year from {table_name} WHERE watched = %s ALLOW
FILTERING"
        )
        params = [watched]
    return self.session.execute(stmt, parameters=params).all()

def get_movie(self, table_name, title, year):
    """
    Gets a single movie from the table, by title and year.

    :param table_name: The name of the movie table.
    :param title: The title of the movie.
    :param year: The year of the movie's release.
    :return: The requested movie.
    """
    return self.session.execute(
        SimpleStatement(
            f"SELECT * from {table_name} WHERE title = %s AND year = %s"

```

```
        ),
        parameters=[title, year],
    ).one()

def watched_movie(self, table_name, title, year):
    """
    Updates a movie as having been watched.

    :param table_name: The name of the movie table.
    :param title: The title of the movie.
    :param year: The year of the movie's release.
    """
    self.session.execute(
        SimpleStatement(
            f"UPDATE {table_name} SET watched=true WHERE title = %s AND year
= %s"
        ),
        parameters=[title, year],
    )
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Python (Boto3).
 - [CreateKeyspace](#)
 - [CreateTable](#)
 - [DeleteKeyspace](#)
 - [DeleteTable](#)
 - [GetKeyspace](#)
 - [GetTable](#)
 - [ListKeyspaces](#)
 - [ListTables](#)
 - [RestoreTable](#)
 - [UpdateTable](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [L'utilisation de ce service avec un AWS Kit SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Actions pour Amazon Keyspaces à l'aide de kits SDK AWS

Les exemples de code suivants montrent comment effectuer des actions Amazon Keyspaces individuelles avec AWS des SDK. Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions pour configurer et exécuter le code.

Les exemples suivants incluent uniquement les actions les plus couramment utilisées. Pour obtenir la liste complète, consultez la [Référence des API Amazon Keyspaces \(pour Apache Cassandra\)](#).

Exemples

- [Utilisation CreateKeyspace avec un AWS SDK](#)
- [Utilisation CreateTable avec un AWS SDK](#)
- [Utilisation DeleteKeyspace avec un AWS SDK](#)
- [Utilisation DeleteTable avec un AWS SDK](#)
- [Utilisation GetKeyspace avec un AWS SDK](#)
- [Utilisation GetTable avec un AWS SDK](#)
- [Utilisation ListKeyspaces avec un AWS SDK](#)
- [Utilisation ListTables avec un AWS SDK](#)
- [Utilisation RestoreTable avec un AWS SDK](#)
- [Utilisation UpdateTable avec un AWS SDK](#)

Utilisation **CreateKeyspace** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser CreateKeyspace.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Create a new keyspace.
/// </summary>
/// <param name="keyspaceName">The name for the new keyspace.</param>
/// <returns>The Amazon Resource Name (ARN) of the new keyspace.</returns>
public async Task<string> CreateKeyspace(string keyspaceName)
{
    var response =
        await _amazonKeyspaces.CreateKeyspaceAsync(
            new CreateKeyspaceRequest { KeyspaceName = keyspaceName });
    return response.ResourceArn;
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateKeyspace](#) à la section Référence des AWS SDK pour .NET API.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
```

```
try {
    CreateKeyspaceRequest keyspaceRequest =
    CreateKeyspaceRequest.builder()
        .keyspaceName(keyspaceName)
        .build();

    CreateKeyspaceResponse response =
    keyClient.createKeyspace(keyspaceRequest);
    System.out.println("The ARN of the KeySpace is " +
    response.resourceArn());

} catch (KeyspacesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateKeyspace](#) à la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

```
}
```

- Pour plus de détails sur l'API, consultez [CreateKeyspace](#) la section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)

    def create_keyspace(self, name):
        """
        Creates a keyspace.

        :param name: The name to give the keyspace.
        :return: The Amazon Resource Name (ARN) of the new keyspace.
```

```

"""
try:
    response = self.keyspaces_client.create_keyspace(keyspaceName=name)
    self.ks_name = name
    self.ks_arn = response["resourceArn"]
except ClientError as err:
    logger.error(
        "Couldn't create %s. Here's why: %s: %s",
        name,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise
else:
    return self.ks_arn

```

- Pour plus de détails sur l'API, consultez [CreateKeyspace](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
    oo_result = lo_kys->createkeyspace(
        iv_keyspacename = iv_keyspace_name ).
    MESSAGE 'Keyspace created successfully.' TYPE 'I'.
CATCH /aws1/cx_kysconflictexception.
    MESSAGE 'Keyspace already exists.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.

```

```
ENDTRY.
```

- Pour plus de détails sur l'API, consultez [CreateKeyspace](#) la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [L'utilisation de ce service avec un AWS Kit SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **CreateTable** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser CreateTable.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Create a new Amazon Keyspaces table.
/// </summary>
/// <param name="keyspaceName">The keyspace where the table will be
created.</param>
/// <param name="schema">The schema for the new table.</param>
/// <param name="tableName">The name of the new table.</param>
/// <returns>The Amazon Resource Name (ARN) of the new table.</returns>
public async Task<string> CreateTable(string keyspaceName, SchemaDefinition
schema, string tableName)
```



```

{
    var request = new CreateTableRequest
    {
        KeyspaceName = keyspaceName,
        SchemaDefinition = schema,
        TableName = tableName,
        PointInTimeRecovery = new PointInTimeRecovery { Status =
PointInTimeRecoveryStatus.ENABLED }
    };

    var response = await _amazonKeyspaces.CreateTableAsync(request);
    return response.ResourceArn;
}

```

- Pour plus de détails sur l'API, reportez-vous [CreateTable](#) à la section Référence des AWS SDK pour .NET API.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        // Set the columns.
        ColumnDefinition defTitle = ColumnDefinition.builder()
            .name("title")
            .type("text")
            .build();

        ColumnDefinition defYear = ColumnDefinition.builder()
            .name("year")
            .type("int")

```

```
        .build();

ColumnDefinition defReleaseDate = ColumnDefinition.builder()
    .name("release_date")
    .type("timestamp")
    .build();

ColumnDefinition defPlot = ColumnDefinition.builder()
    .name("plot")
    .type("text")
    .build();

List<ColumnDefinition> collist = new ArrayList<>();
collist.add(defTitle);
collist.add(defYear);
collist.add(defReleaseDate);
collist.add(defPlot);

// Set the keys.
PartitionKey yearKey = PartitionKey.builder()
    .name("year")
    .build();

PartitionKey titleKey = PartitionKey.builder()
    .name("title")
    .build();

List<PartitionKey> keyList = new ArrayList<>();
keyList.add(yearKey);
keyList.add(titleKey);

SchemaDefinition schemaDefinition = SchemaDefinition.builder()
    .partitionKeys(keyList)
    .allColumns(collist)
    .build();

PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
    .status(PointInTimeRecoveryStatus.ENABLED)
    .build();

CreateTableRequest tableRequest = CreateTableRequest.builder()
    .keyspaceName(keySpace)
    .tableName(tableName)
    .schemaDefinition(schemaDefinition)
```

```
        .pointInTimeRecovery(timeRecovery)
        .build();

    CreateTableResponse response = keyClient.createTable(tableRequest);
    System.out.println("The table ARN is " + response.resourceArn());

} catch (KeyspacesException e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateTable](#) à la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }
}
```

```
    }

    val defReleaseDate =
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val collist = ArrayList<ColumnDefinition>()
    collist.add(defTitle)
    collist.add(defYear)
    collist.add(defReleaseDate)
    collist.add(defPlot)

    // Set the keys.
    val yearKey =
        PartitionKey {
            name = "year"
        }

    val titleKey =
        PartitionKey {
            name = "title"
        }

    val keyList = ArrayList<PartitionKey>()
    keyList.add(yearKey)
    keyList.add(titleKey)

    val schemaDefinition0b =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = collist
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }
```

```

    }

    val tableRequest =
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionObj
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}

```

- Pour plus de détails sur l'API, consultez [CreateTable](#) la section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None

```

```
self.table_name = None

@classmethod
def from_client(cls):
    keyspaces_client = boto3.client("keyspaces")
    return cls(keyspaces_client)

def create_table(self, table_name):
    """
    Creates a table in the keyspace.
    The table is created with a schema for storing movie data
    and has point-in-time recovery enabled.

    :param table_name: The name to give the table.
    :return: The ARN of the new table.
    """
    try:
        response = self.keyspaces_client.create_table(
            keyspaceName=self.ks_name,
            tableName=table_name,
            schemaDefinition={
                "allColumns": [
                    {"name": "title", "type": "text"},
                    {"name": "year", "type": "int"},
                    {"name": "release_date", "type": "timestamp"},
                    {"name": "plot", "type": "text"},
                ],
                "partitionKeys": [{"name": "year"}, {"name": "title"}],
            },
            pointInTimeRecovery={"status": "ENABLED"},
        )
    except ClientError as err:
        logger.error(
            "Couldn't create table %s. Here's why: %s: %s",
            table_name,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
    else:
        return response["resourceArn"]
```

- Pour plus de détails sur l'API, consultez [CreateTable](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
    " Define schema with columns
    DATA(lt_columns) = VALUE /aws1/
cl_kyscolumnndefinition=>tt_columnndefinitionlist(
    ( NEW /aws1/cl_kyscolumnndefinition( iv_name = 'title' iv_type =
'text' ) )
    ( NEW /aws1/cl_kyscolumnndefinition( iv_name = 'year' iv_type =
'int' ) )
    ( NEW /aws1/cl_kyscolumnndefinition( iv_name = 'release_date' iv_type =
'timestamp' ) )
    ( NEW /aws1/cl_kyscolumnndefinition( iv_name = 'plot' iv_type =
'text' ) )
    ).

    " Define partition keys
    DATA(lt_partition_keys) = VALUE /aws1/
cl_kyspartitionkey=>tt_partitionkeylist(
    ( NEW /aws1/cl_kyspartitionkey( iv_name = 'year' ) )
    ( NEW /aws1/cl_kyspartitionkey( iv_name = 'title' ) )
    ).

    " Create schema definition
    DATA(lo_schema) = NEW /aws1/cl_kysschemadefinition(
        it_allcolumns = lt_columns
        it_partitionkeys = lt_partition_keys ).

```

```

" Enable point-in-time recovery
DATA(lo_pitr) = NEW /aws1/cl_kyspointintimerec(
    iv_status = 'ENABLED' ).

oo_result = lo_kys->createtable(
    iv_keyspacename = iv_keyspace_name
    iv_tablename = iv_table_name
    io_schemadefinition = lo_schema
    io_pointintimerecovery = lo_pitr ).
MESSAGE 'Table created successfully.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- Pour plus de détails sur l'API, consultez [CreateTable](#) la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [L'utilisation de ce service avec un AWS Kit SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **DeleteKeyspace** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser DeleteKeyspace.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Delete an existing keyspace.
/// </summary>
/// <param name="keyspaceName"></param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteKeyspace(string keyspaceName)
{
    var response = await _amazonKeyspaces.DeleteKeyspaceAsync(
        new DeleteKeyspaceRequest { KeyspaceName = keyspaceName });
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteKeyspace](#) à la section Référence des AWS SDK pour .NET API.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
    try {
```

```
        DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
    .keyspaceName(keyspaceName)
    .build();

        keyClient.deleteKeyspace(deleteKeyspaceRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteKeyspace](#) à la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}
```

- Pour plus de détails sur l'API, consultez [DeleteKeyspace](#) la section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)

    def delete_keyspace(self):
        """
        Deletes the keyspace.
        """
        try:
            self.keyspaces_client.delete_keyspace(keyspaceName=self.ks_name)
            self.ks_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
)
raise
```

- Pour plus de détails sur l'API, consultez [DeleteKeyspace](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.
    lo_kys->deletekeyspace(
        iv_keyspacename = iv_keyspace_name ).
    MESSAGE 'Keyspace deleted successfully.' TYPE 'I'.
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- Pour plus de détails sur l'API, consultez [DeleteKeyspace](#) la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [L'utilisation de ce service avec un AWS Kit SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **DeleteTable** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser DeleteTable.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Delete an Amazon Keyspaces table.
/// </summary>
/// <param name="keyspaceName">The keyspace containing the table.</param>
/// <param name="tableName">The name of the table to delete.</param>
/// <returns>A Boolean value indicating the success of the action.</returns>
public async Task<bool> DeleteTable(string keyspaceName, string tableName)
{
    var response = await _amazonKeyspaces.DeleteTableAsync(
        new DeleteTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
    return response.HttpStatusCode == HttpStatusCode.OK;
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteTable](#) à la section Référence des AWS SDK pour .NET API.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void deleteTable(KeyspacesClient keyClient, String
keyspaceName, String tableName) {
    try {
        DeleteTableRequest tableRequest = DeleteTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        keyClient.deleteTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteTable](#) à la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}
```

- Pour plus de détails sur l'API, consultez [DeleteTable](#) la section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None
```

```
@classmethod
def from_client(cls):
    keyspaces_client = boto3.client("keyspaces")
    return cls(keyspaces_client)

def delete_table(self):
    """
    Deletes the table from the keyspace.
    """
    try:
        self.keyspaces_client.delete_table(
            keyspaceName=self.ks_name, tableName=self.table_name
        )
        self.table_name = None
    except ClientError as err:
        logger.error(
            "Couldn't delete table %s. Here's why: %s: %s",
            self.table_name,
            err.response["Error"]["Code"],
            err.response["Error"]["Message"],
        )
        raise
```

- Pour plus de détails sur l'API, consultez [DeleteTable](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

TRY.


```

lo_kys->deletetable(
    iv_keyspacename = iv_keyspace_name
    iv_tablename = iv_table_name ).
MESSAGE 'Table deleted successfully.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- Pour plus de détails sur l'API, consultez [DeleteTable](#) la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [L'utilisation de ce service avec un AWS Kit SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **GetKeyspace** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser GetKeyspace.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

/// <summary>
/// Get data about a keyspace.

```

```

/// </summary>
/// <param name="keyspaceName">The name of the keyspace.</param>
/// <returns>The Amazon Resource Name (ARN) of the keyspace.</returns>
public async Task<string> GetKeyspace(string keyspaceName)
{
    var response = await _amazonKeyspaces.GetKeyspaceAsync(
        new GetKeyspaceRequest { KeyspaceName = keyspaceName });
    return response.ResourceArn;
}

```

- Pour plus de détails sur l'API, reportez-vous [GetKeyspace](#) à la section Référence des AWS SDK pour .NET API.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
    try {
        GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        GetKeyspaceResponse response =
keyClient.getKeyspace(keyspaceRequest);
        String name = response.keyspaceName();
        System.out.println("The " + name + " KeySpace is ready");

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [GetKeyspace](#) à la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {  
    val keyspaceRequest =  
        GetKeyspaceRequest {  
            keyspaceName = keyspaceNameVal  
        }  
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->  
        val response: GetKeyspaceResponse =  
        keyClient.getKeyspace(keyspaceRequest)  
        val name = response.keyspaceName  
        println("The $name KeySpace is ready")  
    }  
}
```

- Pour plus de détails sur l'API, consultez [GetKeyspace](#) la section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)

    def exists_keyspace(self, name):
        """
        Checks whether a keyspace exists.

        :param name: The name of the keyspace to look up.
        :return: True when the keyspace exists. Otherwise, False.
        """
        try:
            response = self.keyspaces_client.get_keyspace(keyspaceName=name)
            self.ks_name = response["keyspaceName"]
            self.ks_arn = response["resourceArn"]
            exists = True
        except ClientError as err:
```

```

        if err.response["Error"]["Code"] == "ResourceNotFoundException":
            logger.info("Keyspace %s does not exist.", name)
            exists = False
        else:
            logger.error(
                "Couldn't verify %s exists. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
    return exists

```

- Pour plus de détails sur l'API, consultez [GetKeyspace](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
    oo_result = lo_kys->getkeyspace(
        iv_keyspacename = iv_keyspace_name ).
    MESSAGE 'Keyspace retrieved successfully.' TYPE 'I'.
CATCH /aws1/cx_kysresourcenotfoundex.
    MESSAGE 'Keyspace does not exist.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- Pour plus de détails sur l'API, consultez [GetKeyspace](#) la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [L'utilisation de ce service avec un AWS Kit SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **GetTable** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser `GetTable`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Get information about an Amazon Keyspaces table.
/// </summary>
/// <param name="keyspaceName">The keyspace containing the table.</param>
/// <param name="tableName">The name of the Amazon Keyspaces table.</param>
/// <returns>The response containing data about the table.</returns>
public async Task<GetTableResponse> GetTable(string keyspaceName, string
tableName)
{
    var response = await _amazonKeyspaces.GetTableAsync(
        new GetTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
```

```
        return response;
    }
```

- Pour plus de détails sur l'API, reportez-vous [GetTable](#) à la section Référence des AWS SDK pour .NET API.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
    throws InterruptedException {
    try {
        boolean tableStatus = false;
        String status;
        GetTableResponse response = null;
        GetTableRequest tableRequest = GetTableRequest.builder()
            .keyspaceName(keyspaceName)
            .tableName(tableName)
            .build();

        while (!tableStatus) {
            response = keyClient.getTable(tableRequest);
            status = response.statusAsString();
            System.out.println(". The table status is " + status);

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true;
            }
            Thread.sleep(500);
        }
    }
```

```

        List<ColumnDefinition> cols =
response.schemaDefinition().allColumns();
        for (ColumnDefinition def : cols) {
            System.out.println("The column name is " + def.name());
            System.out.println("The column type is " + def.type());
        }

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [GetTable](#) à la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->

```



```

        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
        val cols: List<ColumnDefinition>? =
response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}

```

- Pour plus de détails sur l'API, consultez [GetTable](#) la section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.

```

```

        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)

    def get_table(self, table_name):
        """
        Gets data about a table in the keyspace.

        :param table_name: The name of the table to look up.
        :return: Data about the table.
        """
        try:
            response = self.keyspaces_client.get_table(
                keyspaceName=self.ks_name, tableName=table_name
            )
            self.table_name = table_name
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Table %s does not exist.", table_name)
                self.table_name = None
                response = None
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response

```

- Pour plus de détails sur l'API, consultez [GetTable](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.
    oo_result = lo_kys->gettable(
        iv_keyspacename = iv_keyspace_name
        iv_tablename = iv_table_name ).
    MESSAGE 'Table information retrieved successfully.' TYPE 'I'.
CATCH /aws1/cx_kysresourcenotfoundex.
    MESSAGE 'Table does not exist.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
    MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- Pour plus de détails sur l'API, consultez [GetTable](#) la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [L'utilisation de ce service avec un AWS Kit SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **ListKeyspaces** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser ListKeyspaces.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Lists all keyspaces for the account.
/// </summary>
/// <returns>Async task.</returns>
public async Task ListKeyspaces()
{
    var paginator = _amazonKeyspaces.Paginators.ListKeyspaces(new
ListKeyspacesRequest());

    Console.WriteLine("{0, -30}\t{1}", "Keyspace name", "Keyspace ARN");
    Console.WriteLine(new string('-', Console.WindowWidth));
    await foreach (var keyspace in paginator.Keyspaces)
    {
        Console.WriteLine($"{keyspace.KeyspaceName, -30}\t{keyspace.ResourceArn}");
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListKeyspaces](#) à la section Référence des AWS SDK pour .NET API.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
    try {
        ListKeyspacesRequest keyspacesRequest =
            ListKeyspacesRequest.builder()
                .maxResults(10)
                .build();

        ListKeyspacesIterable listRes =
            keyClient.listKeyspacesPaginator(keyspacesRequest);
        listRes.stream()
            .flatMap(r -> r.keyspaces().stream())
            .forEach(content -> System.out.println(" Name: " +
                content.keyspaceName()));
    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListKeyspaces](#) à la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listKeyspacesPaginator() {
    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
}
```

- Pour plus de détails sur l'API, consultez [ListKeyspaces](#) la section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""
```

```

def __init__(self, keyspaces_client):
    """
    :param keyspaces_client: A Boto3 Amazon Keyspaces client.
    """
    self.keyspaces_client = keyspaces_client
    self.ks_name = None
    self.ks_arn = None
    self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)

    def list_keyspaces(self, limit):
        """
        Lists the keyspaces in your account.

        :param limit: The maximum number of keyspaces to list.
        """
        try:
            ks_paginator = self.keyspaces_client.get_paginator("list_keyspaces")
            for page in ks_paginator.paginate(PaginationConfig={"MaxItems":
limit}):
                for ks in page["keyspaces"]:
                    print(ks["keyspaceName"])
                    print(f"\t{ks['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list keyspaces. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

```

- Pour plus de détails sur l'API, consultez [ListKeyspaces](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.  
    oo_result = lo_kys->listkeyspaces(  
        iv_maxresults = iv_max_results ).  
    MESSAGE 'Keyspaces listed successfully.' TYPE 'I'.  
    CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->  
>av_err_msg }|.  
    MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- Pour plus de détails sur l'API, consultez [ListKeyspaces](#) la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [L'utilisation de ce service avec un AWS Kit SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **ListTables** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser `ListTables`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Lists the Amazon Keyspaces tables in a keyspace.
/// </summary>
/// <param name="keyspaceName">The name of the keyspace.</param>
/// <returns>A list of TableSummary objects.</returns>
public async Task<List<TableSummary>> ListTables(string keyspaceName)
{
    var response = await _amazonKeyspaces.ListTablesAsync(new
ListTablesRequest { KeyspaceName = keyspaceName });
    response.Tables.ForEach(table =>
    {

Console.WriteLine($"{table.KeyspaceName}\t{table.TableName}\t{table.ResourceArn}");
    });

    return response.Tables;
}
```

- Pour plus de détails sur l'API, reportez-vous [ListTables](#) à la section Référence des AWS SDK pour .NET API.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void listTables(KeyspacesClient keyClient, String keyspaceName)
{
    try {
        ListTablesRequest tablesRequest = ListTablesRequest.builder()
            .keyspaceName(keyspaceName)
            .build();

        ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
        listRes.stream()
            .flatMap(r -> r.tables().stream())
            .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                " Table name: " + content.tableName()));

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListTables](#) à la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}
```

- Pour plus de détails sur l'API, consultez [ListTables](#) la section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)

    def list_tables(self):
        """
        Lists the tables in the keyspace.
        """
        try:
            table_paginator = self.keyspaces_client.get_paginator("list_tables")
            for page in table_paginator.paginate(keyspaceName=self.ks_name):
                for table in page["tables"]:
                    print(table["tableName"])
                    print(f"\t{table['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list tables in keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise

```

- Pour plus de détails sur l'API, consultez [ListTables](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.  
    oo_result = lo_kys->listtables(  
        iv_keyspacename = iv_keyspace_name ).  
    MESSAGE 'Tables listed successfully.' TYPE 'I'.  
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
    DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.  
    MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- Pour plus de détails sur l'API, consultez [ListTables](#) la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [L'utilisation de ce service avec un AWS Kit SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **RestoreTable** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser `RestoreTable`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Restores the specified table to the specified point in time.
/// </summary>
/// <param name="keyspaceName">The keyspace containing the table.</param>
/// <param name="tableName">The name of the table to restore.</param>
/// <param name="timestamp">The time to which the table will be restored.</
param>
/// <returns>The Amazon Resource Name (ARN) of the restored table.</returns>
public async Task<string> RestoreTable(string keyspaceName, string tableName,
string restoredTableName, DateTime timestamp)
{
    var request = new RestoreTableRequest
    {
        RestoreTimestamp = timestamp,
        SourceKeyspaceName = keyspaceName,
        SourceTableName = tableName,
        TargetKeyspaceName = keyspaceName,
        TargetTableName = restoredTableName
    };

    var response = await _amazonKeyspaces.RestoreTableAsync(request);
    return response.RestoredTableARN;
}
```

- Pour plus de détails sur l'API, reportez-vous [RestoreTable](#) à la section Référence des AWS SDK pour .NET API.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public static void restoreTable(KeyspacesClient keyClient, String
keyspaceName, ZonedDateTime utc) {
    try {
        Instant myTime = utc.toInstant();
        RestoreTableRequest restoreTableRequest =
        RestoreTableRequest.builder()
            .restoreTimestamp(myTime)
            .sourceTableName("Movie")
            .targetKeyspaceName(keyspaceName)
            .targetTableName("MovieRestore")
            .sourceKeyspaceName(keyspaceName)
            .build();

        RestoreTableResponse response =
        keyClient.restoreTable(restoreTableRequest);
        System.out.println("The ARN of the restored table is " +
        response.restoredTableARN());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [RestoreTable](#) à la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun restoreTable(
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}
```

- Pour plus de détails sur l'API, consultez [RestoreTable](#) la section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)

    def restore_table(self, restore_timestamp):
        """
        Restores the table to a previous point in time. The table is restored
        to a new table in the same keyspace.

        :param restore_timestamp: The point in time to restore the table. This
        time must be in UTC format.
        :return: The name of the restored table.
        """
        try:
            restored_table_name = f"{self.table_name}_restored"
            self.keyspaces_client.restore_table(
```

```

        sourceKeyspaceName=self.ks_name,
        sourceTableName=self.table_name,
        targetKeyspaceName=self.ks_name,
        targetTableName=restored_table_name,
        restoreTimestamp=restore_timestamp,
    )
except ClientError as err:
    logger.error(
        "Couldn't restore table %s. Here's why: %s: %s",
        restore_timestamp,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise
else:
    return restored_table_name

```

- Pour plus de détails sur l'API, consultez [RestoreTable](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

TRY.

```

oo_result = lo_kys->restoretable(
    iv_sourcekeyspacename = iv_source_keyspace_name
    iv_sourcetablename = iv_source_table_name
    iv_targetkeyspacename = iv_target_keyspace_name
    iv_targettablename = iv_target_table_name
    iv_restoretimestamp = iv_restore_timestamp ).
MESSAGE 'Table restore initiated successfully.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).

```

```
DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- Pour plus de détails sur l'API, consultez [RestoreTable](#) la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [L'utilisation de ce service avec un AWS Kit SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation **UpdateTable** avec un AWS SDK

Les exemples de code suivants illustrent comment utiliser UpdateTable.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Updates the movie table to add a boolean column named watched.
/// </summary>
/// <param name="keyspaceName">The keyspace containing the table.</param>
/// <param name="tableName">The name of the table to change.</param>
/// <returns>The Amazon Resource Name (ARN) of the updated table.</returns>
public async Task<string> UpdateTable(string keyspaceName, string tableName)
```

```

{
    var newColumn = new ColumnDefinition { Name = "watched", Type =
"boolean" };
    var request = new UpdateTableRequest
    {
        KeyspaceName = keyspaceName,
        TableName = tableName,
        AddColumns = new List<ColumnDefinition> { newColumn }
    };
    var response = await _amazonKeyspaces.UpdateTableAsync(request);
    return response.ResourceArn;
}

```

- Pour plus de détails sur l'API, voir [UpdateTable](#) la section Référence des AWS SDK pour .NET API.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
    try {
        ColumnDefinition def = ColumnDefinition.builder()
            .name("watched")
            .type("boolean")
            .build();

        UpdateTableRequest tableRequest = UpdateTableRequest.builder()
            .keyspaceName(keySpace)
            .tableName(tableName)
            .addColumns(def)
            .build();
    }
}

```

```

        keyClient.updateTable(tableRequest);

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

- Pour plus de détails sur l'API, voir [UpdateTable](#) la section Référence des AWS SDK for Java 2.x API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

suspend fun updateTable(
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->

```

```

        keyClient.updateTable(tableRequest)
    }
}

```

- Pour plus de détails sur l'API, consultez [UpdateTable](#) la section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
    actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)

    def update_table(self):
        """
        Updates the schema of the table.

```

This example updates a table of movie data by adding a new column that tracks whether the movie has been watched.

```
"""
try:
    self.keyspaces_client.update_table(
        keyspaceName=self.ks_name,
        tableName=self.table_name,
        addColumns=[{"name": "watched", "type": "boolean"}],
    )
except ClientError as err:
    logger.error(
        "Couldn't update table %s. Here's why: %s: %s",
        self.table_name,
        err.response["Error"]["Code"],
        err.response["Error"]["Message"],
    )
    raise
```

- Pour plus de détails sur l'API, consultez [UpdateTable](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.
    " Add a new column to track watched movies
    DATA(lt_add_columns) = VALUE /aws1/
cl_kyscolumndefinition=>tt_columndefinitionlist(
    ( NEW /aws1/cl_kyscolumndefinition( iv_name = 'watched' iv_type =
'boolean' ) )
    ).
```

```
oo_result = lo_kys->updatetable(  
    iv_keyspacename = iv_keyspace_name  
    iv_tablename = iv_table_name  
    it_addcolumns = lt_add_columns ).  
MESSAGE 'Table updated successfully.' TYPE 'I'.  
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).  
DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->  
>av_err_msg }|.  
MESSAGE lv_error TYPE 'E'.  
ENDTRY.
```

- Pour plus de détails sur l'API, consultez [UpdateTable](#) la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [L'utilisation de ce service avec un AWS Kit SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Bibliothèques et outils Amazon Keyspaces (pour Apache Cassandra)

Cette section fournit des informations sur les bibliothèques Amazon Keyspaces (pour Apache Cassandra), des exemples de code et des outils.

Rubriques

- [Bibliothèques et exemples](#)
- [Exemples et dépôts d'outils de développement mis en évidence](#)

Bibliothèques et exemples

[Vous trouverez les bibliothèques open source et les outils de développement d'Amazon Keyspaces GitHub dans les dépôts AWS et les exemples de dépôts.](#)

Boîte à outils pour développeurs Amazon Keyspaces (pour Apache Cassandra)

Ce référentiel fournit une image docker contenant des outils de développement utiles pour Amazon Keyspaces. Par exemple, il inclut un fichier CQLSHRC contenant les meilleures pratiques, une extension d'AWS authentication facultative pour cqlsh et des outils d'assistance pour effectuer les tâches courantes. La boîte à outils est optimisée pour Amazon Keyspaces, mais fonctionne également avec les clusters Apache Cassandra.

<https://github.com/aws-samples/amazon-keyspaces-toolkit>.

Exemples d'Amazon Keyspaces (pour Apache Cassandra)

Ce dépôt est notre liste officielle d'exemples de code Amazon Keyspaces. Le dépôt est subdivisé en sections par langue (voir [Exemples](#)). Chaque langue possède sa propre sous-section d'exemples. Ces exemples illustrent les implémentations et modèles courants des services Amazon Keyspaces que vous pouvez utiliser lors de la création d'applications.

<https://github.com/aws-samples/amazon-keyspaces-examples/>.

AWS Plug-ins d'authentification Signature Version 4 (SigV4)

Les plugins vous permettent de gérer l'accès à Amazon Keyspaces en utilisant des utilisateurs et des Gestion des identités et des accès AWS rôles (IAM).

Java : <https://github.com/aws/aws-sigv4-auth-cassandra-java-driver-plugin>.

Node.js: <https://github.com/aws/aws-sigv4-auth-cassandra-nodejs-driver-plugin>.

Python: <https://github.com/aws/aws-sigv4-auth-cassandra-python-driver-plugin>.

Allez : <https://github.com/aws/aws-sigv4-auth-cassandra-gocql-driver-plugin>.

Exemples et dépôts d'outils de développement mis en évidence

Vous trouverez ci-dessous une sélection d'outils communautaires utiles pour Amazon Keyspaces (pour Apache Cassandra).

Tampons de protocole Amazon Keyspaces

Vous pouvez utiliser les Protocol Buffers (Protobuf) avec Amazon Keyspaces pour proposer une alternative aux types définis par l'utilisateur (UDT) d'Apache Cassandra. Protobuf est un format de données multiplateforme gratuit et open source utilisé pour sérialiser des données structurées. Vous pouvez stocker les données Protobuf à l'aide du type de données CQL et BLOB refactoriser les UDT tout en préservant les données structurées dans les applications et les langages de programmation.

Ce référentiel fournit un exemple de code qui se connecte à Amazon Keyspaces, crée une nouvelle table et insère une ligne contenant un message Protobuf. Ensuite, la ligne est lue avec une forte cohérence.

<https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/java/datastax-v4/protobuf-user-defined-types>

CloudFormation modèle pour créer un CloudWatch tableau de bord Amazon pour les métriques Amazon Keyspaces (pour Apache Cassandra)

Ce référentiel fournit des CloudFormation modèles permettant de configurer rapidement CloudWatch des métriques pour Amazon Keyspaces. L'utilisation de ce modèle vous permettra de démarrer

plus facilement en fournissant des CloudWatch tableaux de bord prédéfinis déployables avec des métriques couramment utilisées.

<https://github.com/aws-samples/amazon-keyspaces-cloudwatch-cloudformation-templates>.

Utilisation d'Amazon Keyspaces (pour Apache Cassandra) avec AWS Lambda

Le référentiel contient des exemples qui montrent comment se connecter à Amazon Keyspaces depuis Lambda. Vous trouverez ci-dessous quelques exemples.

C#/.NET : <https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/dotnet/datastax-v3/connection-lambda>

Java : <https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/java/datastax-v4/connection-lambda>.

Un autre exemple Lambda qui montre comment déployer et utiliser Amazon Keyspaces à partir d'un Lambda Python est disponible dans le référentiel suivant.

<https://github.com/aws-samples/aws-keyspaces-lambda-python>

Utilisation d'Amazon Keyspaces (pour Apache Cassandra) avec Spring

Voici un exemple qui vous montre comment utiliser Amazon Keyspaces avec Spring Boot.

<https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/java/datastax-v4/spring>

Utilisation d'Amazon Keyspaces (pour Apache Cassandra) avec Scala

Il s'agit d'un exemple qui montre comment se connecter à Amazon Keyspaces à l'aide du plugin d'authentification SigV4 avec Scala.

<https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/scala/datastax-v4/connection-sigv4>

Utilisation d'Amazon Keyspaces (pour Apache Cassandra) avec AWS Glue

Il s'agit d'un exemple qui montre comment utiliser Amazon Keyspaces avec AWS Glue

<https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/scala/datastax-v4/aws-glue>

Amazon Keyspaces (pour Apache Cassandra) Langage de requête Cassandra (CQL) vers convertisseur CloudFormation

Ce package implémente un outil de ligne de commande pour convertir les scripts Apache Cassandra Query Language (CQL) en modèles CloudFormation (CloudFormation), ce qui permet de gérer facilement les schémas Amazon Keyspaces en piles. CloudFormation

<https://github.com/aws/amazon-keyspaces-cql-to-cfn-converter>.

Amazon Keyspaces (pour Apache Cassandra), aides pour le pilote Apache Cassandra pour Java

Ce référentiel contient des politiques relatives aux pilotes, des exemples et des bonnes pratiques lors de l'utilisation du pilote DataStax Java avec Amazon Keyspaces (pour Apache Cassandra).

<https://github.com/aws-samples/amazon-keyspaces-java-driver-helpers>.

Démo de compression rapide d'Amazon Keyspaces (pour Apache Cassandra)

Ce référentiel explique comment compresser, stocker et stocker des objets read/write volumineux pour améliorer les performances et réduire les coûts de débit et de stockage.

<https://github.com/aws-samples/amazon-keyspaces-compression-example>.

Démo d'Amazon Keyspaces (pour Apache Cassandra) et du codec Amazon S3

Le codec Amazon S3 personnalisé prend en charge le mappage transparent et configurable par l'utilisateur des pointeurs UUID vers les objets Amazon S3.

<https://github.com/aws-samples/amazon-keyspaces-large-object-s3-demo>.

Bonnes pratiques de conception et d'architecture avec Amazon Keyspaces (pour Apache Cassandra)

Utilisez cette section pour trouver rapidement des recommandations visant à optimiser les performances et à minimiser les coûts de débit lorsque vous travaillez avec Amazon Keyspaces.

Table des matières

- [Principales différences et principes de conception de la conception NoSQL](#)
 - [Différences entre le système de base de données relationnelle et le système NoSQL](#)
 - [Deux concepts essentiels de la conception NoSQL](#)
 - [Approche de la conception NoSQL](#)
- [Optimisation des connexions client-pilote pour l'environnement sans serveur](#)
 - [Comment fonctionnent les connexions dans Amazon Keyspaces](#)
 - [Comment configurer les connexions dans Amazon Keyspaces](#)
 - [Comment configurer la politique de nouvelles tentatives pour les connexions dans Amazon Keyspaces](#)
 - [Comment configurer les connexions via les points de terminaison VPC dans Amazon Keyspaces](#)
 - [Comment surveiller les connexions dans Amazon Keyspaces](#)
 - [Comment gérer les erreurs de connexion dans Amazon Keyspaces](#)
- [Bonnes pratiques de modélisation des données : recommandations pour la conception de modèles de données](#)
 - [Comment utiliser efficacement les clés de partition dans Amazon Keyspaces](#)
 - [Utilisez le sharding en écriture pour répartir uniformément les charges de travail entre les partitions](#)
 - [Sharding à l'aide de clés de partition composées et de valeurs aléatoires](#)
 - [Sharding à l'aide de clés de partition composées et de valeurs calculées](#)
- [Optimisation des coûts des tables Amazon Keyspaces](#)
 - [Évaluer les coûts au niveau de la table](#)
 - [Comment consulter les coûts d'un seul tableau Amazon Keyspaces](#)
 - [Vue par défaut de Cost Explorer](#)
 - [Comment utiliser et appliquer des balises de table dans Cost Explorer](#)

- [Évaluez le mode capacité de votre table](#)
 - [Quels sont les modes de capacité de table disponibles ?](#)
 - [Quand sélectionner le mode de capacité à la demande ?](#)
 - [Quand sélectionner le mode de capacité provisionnée ?](#)
 - [Autres facteurs à prendre en compte lors du choix d'un mode de capacité de table](#)
- [Évaluez les paramètres Application Auto Scaling de votre table](#)
 - [Comprendre les paramètres de votre Application Auto Scaling](#)
 - [Comment identifier les tables présentant un faible taux d'utilisation cible \(\$\leq 50\%\$ \)](#)
 - [Comment gérer les charges de travail liées aux variations saisonnières](#)
 - [Comment gérer les pics de charge de travail imprévisibles](#)
 - [Comment gérer les charges de travail avec des applications liées](#)
- [Identifiez vos ressources inutilisées pour optimiser les coûts dans Amazon Keyspaces](#)
 - [Comment identifier les ressources inutilisées](#)
 - [Identification des ressources de table inutilisées](#)
 - [Nettoyage des ressources de table inutilisées](#)
 - [Nettoyage des sauvegardes de restauration ponctuelle \(PITR\) inutilisées](#)
- [Évaluez les modèles d'utilisation de vos tables pour optimiser les performances et les coûts](#)
 - [Effectuer moins d'opérations de lecture fortement cohérente](#)
 - [Activer la durée de vie \(TTL\)](#)
- [Évaluer la capacité allouée pour un dimensionnement approprié](#)
 - [Comment récupérer les statistiques de consommation à partir de vos tableaux Amazon Keyspaces](#)
 - [Comment identifier les tables Amazon Keyspaces sous-approvisionnées](#)
 - [Comment identifier les tables Amazon Keyspaces suraprovisionnées](#)

Principales différences et principes de conception de la conception NoSQL

Les systèmes de base de données NoSQL tels qu'Amazon Keyspaces utilisent des modèles alternatifs pour la gestion des données, tels que les paires clé-valeur ou le stockage de documents.

Conception NoSQL

Lorsque vous passez d'un système de gestion de base de données relationnelle à un système de

base de données NoSQL tel qu'Amazon Keyspaces, il est important de comprendre les principales différences et les approches de conception spécifiques.

Rubriques

- [Différences entre le système de base de données relationnelle et le système NoSQL](#)
- [Deux concepts essentiels de la conception NoSQL](#)
- [Approche de la conception NoSQL](#)

Différences entre le système de base de données relationnelle et le système NoSQL

Les systèmes de gestion de bases de données relationnelles (SGBDR) et les bases de données NoSQL ont chacun leurs forces et leurs faiblesses :

- Dans les SGBDR, les données peuvent être interrogées avec souplesse, mais les requêtes sont relativement coûteuses et ne sont pas bien mises à l'échelle dans les situations de trafic intense (consultez [the section called “Modélisation des données”](#)).
- Dans une base de données NoSQL telle qu'Amazon Keyspaces, les données peuvent être consultées efficacement d'un nombre limité de manières, en dehors desquelles les requêtes peuvent être coûteuses et lentes.

Ces différences entraînent une conception des bases de données très différente d'un système à l'autre :

- Dans un SGBDR, la conception se concentre avant tout sur la flexibilité, sans se préoccuper des détails de l'implémentation ni des performances. En général, l'optimisation des requêtes n'affecte pas la conception du schéma, mais la normalisation est très importante.
- Dans Amazon Keyspaces, vous concevez votre schéma spécifiquement pour que les requêtes les plus courantes et les plus importantes soient aussi rapides et économiques que possible. Les structures de vos données sont adaptées aux exigences spécifiques de vos cas d'utilisation.

Deux concepts essentiels de la conception NoSQL

Pour concevoir un système NoSQL, il faut un autre état d'esprit que pour un SGBDR. Pour un SGBDR, vous pouvez créer un modèle de données normalisé sans réfléchir aux modèles d'accès.

Vous pouvez l'étendre ultérieurement, pour répondre à de nouvelles questions et de nouveaux besoins d'interrogation. Vous pouvez organiser chaque type de données dans sa propre table.

En quoi la conception d'un système NoSQL est différente

- En revanche, vous ne devriez pas commencer à concevoir votre schéma pour Amazon Keyspaces tant que vous ne connaissez pas les questions auxquelles il doit répondre. Il est essentiel d'identifier au préalable les problèmes métier et les cas d'utilisation de l'application.
- Vous devez gérer le moins de tables possible dans une application Amazon Keyspaces. Le fait de disposer de moins de tables permet de rendre les choses plus évolutives, de réduire la gestion des autorisations et de réduire la charge de travail de votre application Amazon Keyspaces. Cela peut également contribuer à maintenir des coûts de sauvegarde globalement plus faibles.

Approche de la conception NoSQL

La première étape de la conception de votre application Amazon Keyspaces consiste à identifier les modèles de requêtes spécifiques auxquels le système doit répondre.

Il est notamment important de comprendre trois propriétés fondamentales des modèles d'accès de votre application avant de commencer :

- Taille des données : connaître la quantité de données qui sera stockée et demandée en une seule fois permet de déterminer le moyen le plus efficace de partitionner les données.
- Forme des données : au lieu de remodeliser les données lors du traitement d'une requête (comme c'est le cas dans un SGBDR), une base de données NoSQL organise les données de manière à ce que leur forme dans la base de données corresponde aux requêtes. C'est un élément clé pour augmenter la vitesse et la scalabilité.
- Vitesse des données : Amazon Keyspaces évolue en augmentant le nombre de partitions physiques disponibles pour traiter les requêtes et en distribuant efficacement les données entre ces partitions. Le fait de connaître à l'avance les pics de charge des requêtes peut aider à déterminer comment partitionner les données de manière à utiliser au mieux I/O la capacité.

Après avoir identifié les besoins de requête spécifiques, vous pouvez organiser les données en fonction des principes généraux qui déterminent la performance :

- Rassemblez les données connexes. Il y a 20 ans, les recherches visant à optimiser les tables de routage ont démontré que l'emplacement des références était le facteur le plus important pour

accélérer le temps de réponse : il convient de rassembler les données connexes au même endroit. C'est toujours le cas dans les systèmes NoSQL aujourd'hui, dans lesquels le fait de conserver les données connexes à proximité a un impact très important sur les coûts et les performances. Au lieu de répartir des éléments de données connexes sur plusieurs tables, il est préférable de conserver les éléments connexes dans votre système NoSQL aussi près que possible les uns des autres.

En règle générale, vous devez gérer le moins de tables possible dans une application Amazon Keyspaces.

Les cas où des données de série chronologique volumineuses sont impliquées ou dans lesquels les ensembles de données ont des modèles d'accès très différents sont des exemples d'exceptions. Une table unique avec des index inversés peut généralement permettre à des requêtes simples de créer et récupérer les structures de données hiérarchiques complexes dont votre application a besoin.

- Utilisez l'ordre de tri. Les éléments connexes peuvent être regroupés et interrogés de manière efficace si la conception des clés permet de les trier ensemble. Il s'agit d'une stratégie de conception NoSQL importante.
- Répartissez les requêtes. Il est également important qu'un volume élevé de requêtes ne soit pas focalisé sur une partie de la base de données, où elles peuvent dépasser la I/O capacité. Au lieu de cela, vous devez concevoir les clés de données de manière à ce que le trafic soit, autant que possible, réparti de manière uniforme sur les partitions, évitant ainsi les « zones sensibles ».

Ces principes généraux se traduisent par des modèles de conception courants que vous pouvez utiliser pour modéliser efficacement les données dans Amazon Keyspaces.

Optimisation des connexions client-pilote pour l'environnement sans serveur

Pour communiquer avec Amazon Keyspaces, vous pouvez utiliser n'importe quel pilote client Apache Cassandra existant de votre choix. Amazon Keyspaces étant un service sans serveur, nous vous recommandons d'optimiser la configuration de connexion de votre pilote client en fonction des besoins de débit de votre application. Cette rubrique présente les meilleures pratiques, notamment le calcul du nombre de connexions requises par votre application, ainsi que la surveillance et le traitement des erreurs des connexions.

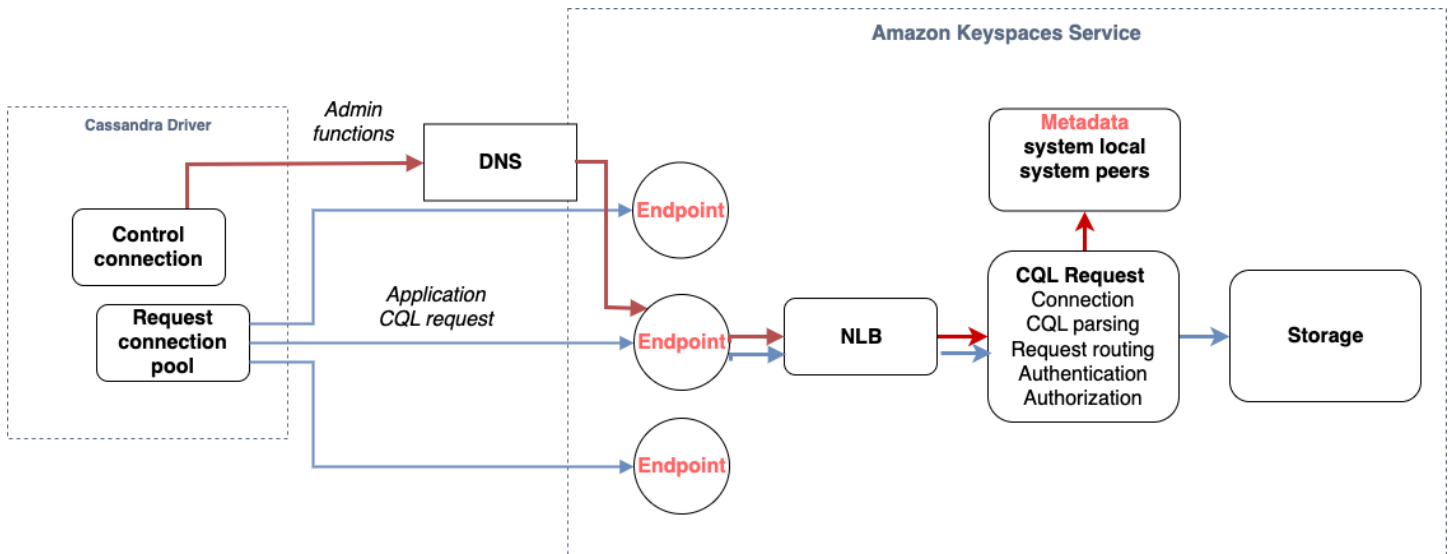
Rubriques

- [Comment fonctionnent les connexions dans Amazon Keyspaces](#)
- [Comment configurer les connexions dans Amazon Keyspaces](#)
- [Comment configurer la politique de nouvelles tentatives pour les connexions dans Amazon Keyspaces](#)
- [Comment configurer les connexions via les points de terminaison VPC dans Amazon Keyspaces](#)
- [Comment surveiller les connexions dans Amazon Keyspaces](#)
- [Comment gérer les erreurs de connexion dans Amazon Keyspaces](#)

Comment fonctionnent les connexions dans Amazon Keyspaces

Cette section donne un aperçu du fonctionnement des connexions client-pilote dans Amazon Keyspaces. Étant donné que la mauvaise configuration du pilote client Cassandra peut entraîner `PerConnectionRequestExceeded` des événements dans Amazon Keyspaces, il est nécessaire de configurer le nombre approprié de connexions dans la configuration du pilote client pour éviter ces erreurs de connexion et d'autres similaires.

Lors de la connexion à Amazon Keyspaces, le pilote a besoin d'un point de terminaison initial pour établir une connexion initiale. Amazon Keyspaces utilise le DNS pour acheminer la connexion initiale vers l'un des nombreux points de terminaison disponibles. Les points de terminaison sont attachés à des équilibres de charge réseau qui établissent à leur tour une connexion avec l'un des gestionnaires de demandes du parc. Une fois la connexion initiale établie, le pilote client collecte des informations sur tous les points de terminaison disponibles dans le `system.peers` tableau. Grâce à ces informations, le pilote client peut créer des connexions supplémentaires aux points de terminaison répertoriés. Le nombre de connexions que le pilote client peut créer est limité par le nombre de connexions locales spécifié dans les paramètres du pilote client. Par défaut, la plupart des pilotes clients établissent une connexion par point de terminaison, établissent un pool de connexions avec Cassandra et équilibrent la charge des requêtes sur ce pool de connexions. Bien que plusieurs connexions puissent être établies vers le même point de terminaison, elles peuvent être connectées derrière l'équilibreur de charge réseau à de nombreux gestionnaires de demandes différents. Lorsque vous vous connectez via le point de terminaison public, l'établissement d'une connexion à chacun des neuf points de terminaison répertoriés dans le `system.peers` tableau entraîne neuf connexions à différents gestionnaires de demandes.



Comment configurer les connexions dans Amazon Keyspaces

Amazon Keyspaces prend en charge jusqu'à 3 000 requêtes CQL par connexion TCP et par seconde. Étant donné qu'il n'existe aucune limite quant au nombre de connexions qu'un pilote peut établir, nous recommandons de ne cibler que 500 demandes CQL par seconde et par connexion afin de prendre en compte les surcharges, les pics de trafic et un meilleur équilibrage de charge. Suivez ces étapes pour vous assurer que la connexion de votre pilote est correctement configurée en fonction des besoins de votre application.

Augmentez le nombre de connexions par adresse IP que votre pilote gère dans son pool de connexions.

- La plupart des pilotes Cassandra établissent un pool de connexions avec Cassandra et équilibrent la charge des requêtes sur ce pool de connexions. Le comportement par défaut de la plupart des pilotes consiste à établir une connexion unique avec chaque point de terminaison. Amazon Keyspaces expose neuf adresses IP homologues aux conducteurs. Ainsi, selon le comportement par défaut de la plupart des conducteurs, cela se traduit par 9 connexions. Amazon Keyspaces prend en charge jusqu'à 3 000 requêtes CQL par connexion TCP par seconde. Par conséquent, le débit de requêtes CQL maximal d'un pilote utilisant les paramètres par défaut est de 27 000 requêtes CQL par seconde. Si vous utilisez les paramètres par défaut du pilote, il se peut qu'une seule connexion doive traiter un débit de requêtes CQL supérieur au débit maximal de 3 000 requêtes CQL par seconde. Cela pourrait entraîner des `PerConnectionRequestExceeded` événements.
- Pour éviter les `PerConnectionRequestExceeded` événements, vous devez configurer le pilote pour créer des connexions supplémentaires par point de terminaison afin de distribuer le débit.

- Pour Amazon Keyspaces, il est recommandé de partir du principe que chaque connexion peut prendre en charge 500 requêtes CQL par seconde.
- Cela signifie que pour une application de production qui doit prendre en charge environ 27 000 requêtes CQL par seconde réparties sur les neuf points de terminaison disponibles, vous devez configurer six connexions par point de terminaison. Cela garantit que chaque connexion ne traite pas plus de 500 demandes par seconde.

Calculez le nombre de connexions par adresse IP que vous devez configurer pour votre pilote en fonction des besoins de votre application.

Pour déterminer le nombre de connexions que vous devez configurer par point de terminaison pour votre application, considérez l'exemple suivant. Vous avez une application qui doit prendre en charge 20 000 requêtes CQL par seconde comprenant 10 000 INSERTSELECT, 5 000 et 5 000 DELETE opérations. L'application Java s'exécute sur trois instances sur Amazon Elastic Container Service (Amazon ECS), où chaque instance établit une session unique avec Amazon Keyspaces. Le calcul que vous pouvez utiliser pour estimer le nombre de connexions que vous devez configurer pour votre pilote utilise l'entrée suivante.

1. Le nombre de demandes par seconde que votre application doit prendre en charge.
2. Le nombre d'instances disponibles, une étant soustraite pour tenir compte de la maintenance ou de la panne.
3. Le nombre de points de terminaison disponibles. Si vous vous connectez via des points de terminaison publics, vous disposez de neuf points de terminaison disponibles. Si vous utilisez des points de terminaison VPC, vous avez entre deux et cinq points de terminaison disponibles, selon la région.
4. Utiliser 500 requêtes CQL par seconde et par connexion est une bonne pratique pour Amazon Keyspaces.
5. Arrondissez le résultat.

Dans cet exemple, la formule ressemble à ceci.

```
20,000 CQL queries / (3 instances - 1 failure) / 9 public endpoints / 500 CQL queries  
per second = ROUND(2.22) = 3
```

Sur la base de ce calcul, vous devez spécifier trois connexions locales par point de terminaison dans la configuration du pilote. Pour les connexions à distance, configurez une seule connexion par point de terminaison.

Comment configurer la politique de nouvelles tentatives pour les connexions dans Amazon Keyspaces

Lorsque vous configurez la politique de nouvelles tentatives pour une connexion à Amazon Keyspaces, nous vous recommandons de mettre en œuvre la politique de nouvelle tentative d'Amazon Keyspaces. `AmazonKeyspacesExponentialRetryPolicy` Cette politique de nouvelle tentative est mieux adaptée pour réessayer sur différentes connexions à Amazon Keyspaces que celle du conducteur. `DefaultRetryPolicy`

Avec le `AmazonKeyspacesExponentialRetryPolicy`, vous pouvez configurer le nombre de tentatives de connexion correspondant à vos besoins. Par défaut, le nombre de nouvelles tentatives pour le `AmazonKeyspacesExponentialRetryPolicy` est défini sur 3.

Un autre avantage est que la politique de réessai d'Amazon Keyspaces renvoie l'exception initiale renvoyée par le service, qui indique pourquoi la tentative de demande a échoué. La politique de nouvelle tentative par défaut renvoie uniquement le générique `NoHostAvailableException`, ce qui peut masquer des informations sur l'échec de la demande.

Pour configurer la politique de nouvelles tentatives de demande à l'aide du `AmazonKeyspacesExponentialRetryPolicy`, nous vous recommandons de configurer un petit nombre de tentatives et de gérer les exceptions renvoyées dans le code de votre application.

Pour des exemples de code implémentant des politiques de nouvelle tentative, consultez les politiques de [nouvelle tentative d'Amazon Keyspaces](#) sur Github.

Comment configurer les connexions via les points de terminaison VPC dans Amazon Keyspaces

Lorsque vous vous connectez via des points de terminaison VPC privés, vous disposez probablement de 3 points de terminaison disponibles. Le nombre de points de terminaison VPC peut être différent par région, en fonction du nombre de zones de disponibilité et du nombre de sous-réseaux dans le VPC attribué. La région USA Est (Virginie du Nord) compte cinq zones de disponibilité et vous pouvez avoir jusqu'à cinq points de terminaison Amazon Keyspaces. La région USA Ouest (Californie du Nord) comporte deux zones de disponibilité et vous pouvez avoir jusqu'à deux points de

terminaison Amazon Keyspaces. Le nombre de points de terminaison n'a aucun impact sur l'échelle, mais il augmente le nombre de connexions que vous devez établir dans la configuration du pilote. Prenez l'exemple de code suivant. Votre application doit prendre en charge 20 000 requêtes CQL et s'exécute sur trois instances sur Amazon ECS, chaque instance établissant une session unique avec Amazon Keyspaces. La seule différence réside dans le nombre de points de terminaison disponibles dans les différents Régions AWS.

Connexions requises dans la région USA Est (Virginie du Nord) :

```
20,000 CQL queries / (3 instances - 1 failure) / 5 private VPC endpoints / 500 CQL queries per second = 4 local connections
```

Connexions requises dans la région de l'ouest des États-Unis (Californie du Nord) :

```
20,000 CQL queries / (3 instances - 1 failure) / 2 private VPC endpoints / 500 CQL queries per second = 10 local connections
```

Important

Lorsque vous utilisez des points de terminaison VPC privés, des autorisations supplémentaires sont nécessaires pour qu'Amazon Keyspaces découvre dynamiquement les points de terminaison VPC disponibles et renseigne le tableau. `system.peers` Pour plus d'informations, consultez [the section called “Remplissage des entrées de `system.peers` table avec les informations de point de terminaison VPC de l'interface”](#).

Lorsque vous accédez à Amazon Keyspaces via un point de terminaison VPC privé en utilisant un autre point de terminaison Compte AWS, il est probable que vous ne voyiez qu'un seul point de terminaison Amazon Keyspaces. Encore une fois, cela n'a aucun impact sur l'ampleur du débit possible vers Amazon Keyspaces, mais cela peut vous obliger à augmenter le nombre de connexions dans la configuration de votre pilote. Cet exemple montre le même calcul pour un seul point de terminaison disponible.

```
20,000 CQL queries / (3 instances - 1 failure) / 1 private VPC endpoints / 500 CQL queries per second = 20 local connections
```

Pour en savoir plus sur l'accès entre comptes à Amazon Keyspaces via un VPC partagé, consultez. [the section called “Configuration de l'accès entre comptes dans un VPC partagé”](#)

Comment surveiller les connexions dans Amazon Keyspaces

Pour identifier le nombre de points de terminaison auxquels votre application est connectée, vous pouvez enregistrer le nombre de paires découverts dans le `system.peers` tableau. L'exemple suivant est un exemple de code Java qui affiche le nombre de paires une fois la connexion établie.

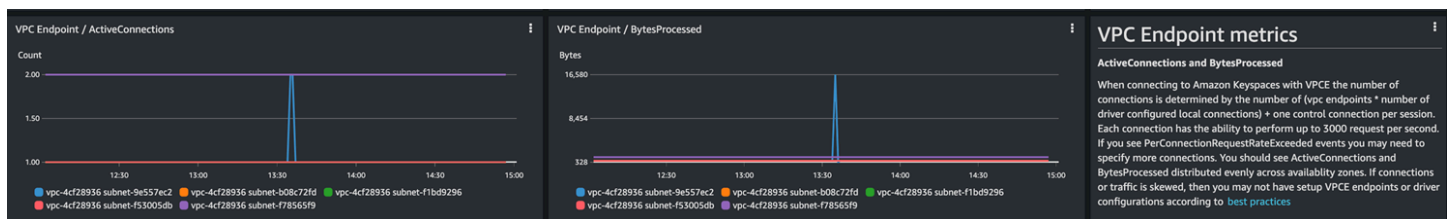
```
ResultSet result = session.execute(new SimpleStatement("SELECT * FROM system.peers"));

logger.info("number of Amazon Keyspaces endpoints:" + result.all().stream().count());
```

Note

La console ou AWS console CQL n'est pas déployée au sein d'un VPC et utilise donc le point de terminaison public. Par conséquent, l'exécution de la `system.peers` requête à partir d'applications situées en dehors du VPCE se traduit souvent par 9 paires. Il peut également être utile d'imprimer les adresses IP de chaque homologue.

Vous pouvez également observer le nombre de paires lorsque vous utilisez un point de terminaison VPC en configurant les métriques Amazon CloudWatch VPCE. Dans CloudWatch, vous pouvez voir le nombre de connexions établies avec le point de terminaison du VPC. Les pilotes Cassandra établissent une connexion pour chaque point de terminaison afin d'envoyer des requêtes CQL et une connexion de contrôle pour recueillir des informations sur les tables système. L'image ci-dessous montre les CloudWatch métriques du point de terminaison VPC après la connexion à Amazon Keyspaces avec une connexion configurée dans les paramètres du pilote. La métrique indique six connexions actives, dont une connexion de contrôle et cinq connexions (une par point de terminaison dans les zones de disponibilité).



Pour commencer à surveiller le nombre de connexions à l'aide d'un CloudWatch graphique, vous pouvez déployer ce CloudFormation modèle disponible GitHub dans le référentiel de [modèles Amazon Keyspaces](#).

Comment gérer les erreurs de connexion dans Amazon Keyspaces

Lorsque le quota de 3 000 demandes par connexion est dépassé, Amazon Keyspaces renvoie un `PerConnectionRequestExceeded` événement et le pilote Cassandra reçoit une `WriteTimeout` exception ou `ReadTimeout`. Vous devez réessayer cette exception avec un retard exponentiel dans votre politique de réessai de Cassandra ou dans votre application. Vous devez prévoir un délai de traitement exponentiel pour éviter d'envoyer des demandes supplémentaires.

La politique de nouvelles tentatives par défaut tente d'intégrer `try next host` le plan de requête. Comme Amazon Keyspaces peut avoir un à trois points de terminaison disponibles lors de la connexion au point de terminaison VPC, vous pouvez également voir les `ReadTimeout` exceptions `NoHostAvailableException` en plus `WriteTimeout` et dans les journaux de vos applications. Vous pouvez utiliser les politiques de réessai fournies par Amazon Keyspaces, qui permettent de réessayer sur le même point de terminaison mais sur des connexions différentes.

Vous trouverez des exemples de politiques de nouvelles tentatives exponentielles pour Java GitHub dans le référentiel d'exemples de code [Java d'Amazon Keyspaces](#). Vous pouvez trouver des exemples de langage supplémentaires sur Github dans le référentiel d'exemples de [code Amazon Keyspaces](#).

Bonnes pratiques de modélisation des données : recommandations pour la conception de modèles de données

Une modélisation des données efficace est essentielle pour optimiser les performances et minimiser les coûts lorsque vous travaillez avec Amazon Keyspaces (pour Apache Cassandra). Cette rubrique présente les principales considérations et recommandations relatives à la conception de modèles de données adaptés aux modèles d'accès aux données de votre application.

- Conception de la clé de partition — La clé de partition joue un rôle essentiel dans la détermination de la manière dont les données sont distribuées entre les partitions d'Amazon Keyspaces. Le choix d'une clé de partition appropriée peut avoir un impact significatif sur les performances des requêtes et les coûts de débit. Cette section décrit les stratégies de conception de clés de partition qui favorisent une répartition uniforme des activités de lecture et d'écriture entre les partitions.
- Considérations clés :
 - Répartition uniforme des activités : visez une activité de lecture et d'écriture uniforme sur toutes les partitions afin de minimiser les coûts de débit et d'exploiter efficacement la capacité de rafale.

- Modèles d'accès — Alignez le design de votre clé de partition avec les principaux modèles d'accès aux données de votre application.
- Taille de partition : évitez de créer des partitions trop grandes, car cela peut avoir un impact sur les performances et augmenter les coûts.

Pour visualiser et concevoir des modèles de données plus facilement, vous pouvez utiliser le [NoSQL Workbench](#).

Rubriques

- [Comment utiliser efficacement les clés de partition dans Amazon Keyspaces](#)

Comment utiliser efficacement les clés de partition dans Amazon Keyspaces

La clé primaire qui identifie de manière unique chaque ligne d'une table Amazon Keyspaces peut être composée d'une ou de plusieurs colonnes de clé de partition, qui déterminent les partitions dans lesquelles les données sont stockées, et d'une ou plusieurs colonnes de clustering facultatives, qui définissent la manière dont les données sont regroupées et triées au sein d'une partition.

Étant donné que la clé de partition définit le nombre de partitions dans lesquelles vos données sont stockées et la manière dont les données sont réparties entre ces partitions, le choix de votre clé de partition peut avoir un impact significatif sur les performances de vos requêtes. En général, vous devez concevoir votre application pour une activité uniforme sur toutes les partitions du disque.

La répartition uniforme des activités de lecture et d'écriture de votre application sur toutes les partitions permet de minimiser les coûts de débit, et cela s'applique aux modes à la demande ainsi qu'aux modes de read/write capacité provisionnée. Par exemple, si vous utilisez le mode capacité provisionnée, vous pouvez déterminer les modèles d'accès dont votre application a besoin et estimer le nombre total d'unités de capacité de lecture (RCU) et d'unités de capacité d'écriture (WCU) requises par chaque table. Amazon Keyspaces prend en charge vos modèles d'accès en utilisant le débit que vous avez fourni, à condition que le trafic sur une partition donnée ne dépasse pas 3 000 RCU et 1 000 WCU.

Lorsqu'une partition présente un débit de lecture ou d'écriture élevé et soutenu, en fonction des modèles de trafic, Amazon Keyspaces peut automatiquement diviser la partition en deux nouvelles partitions. Chaque nouvelle partition contient un sous-ensemble des lignes de la partition d'origine, répartissant le débit uniformément entre les deux partitions.

Amazon Keyspaces offre une flexibilité supplémentaire dans le provisionnement du débit par partition en fournissant une capacité de rafale. Pour plus d'informations, consultez. [the section called “Utiliser la capacité de rafale”](#)

Rubriques

- [Utilisez le sharding en écriture pour répartir uniformément les charges de travail entre les partitions](#)

Utilisez le sharding en écriture pour répartir uniformément les charges de travail entre les partitions

L'un des moyens de mieux répartir les écritures sur une partition dans Amazon Keyspaces consiste à étendre l'espace. Vous pouvez effectuer cette opération de plusieurs manières. Vous pouvez ajouter une colonne de clé de partition supplémentaire dans laquelle vous écrivez des nombres aléatoires pour répartir les lignes entre les partitions. Ou vous pouvez utiliser un nombre qui est calculé en fonction d'une information sur laquelle porte la requête.

Sharding à l'aide de clés de partition composées et de valeurs aléatoires

Une stratégie pour répartir les charges de manière plus uniforme sur une partition consiste à ajouter une colonne de clé de partition supplémentaire dans laquelle vous écrivez des nombres aléatoires. Vous pouvez alors randomiser les écritures sur l'espace plus important.

Par exemple, considérez le tableau suivant qui contient une clé de partition unique représentant une date.

```
CREATE TABLE IF NOT EXISTS tracker.blogs (  
    publish_date date,  
    title text,  
    description int,  
    PRIMARY KEY (publish_date));
```

Pour répartir plus uniformément ce tableau entre les partitions, vous pouvez inclure une colonne de clé de partition supplémentaire `shard` qui stocke des nombres aléatoires. Par exemple :

```
CREATE TABLE IF NOT EXISTS tracker.blogs (  
    publish_date date,  
    shard int,  
    title text,  
    description int,
```

```
PRIMARY KEY ((publish_date, shard));
```

Lorsque vous insérez des données, vous pouvez choisir un nombre aléatoire entre 1 et 200 pour la `shard` colonne. Cela produit des valeurs de clé de partition composées telles que `(2020-07-09, 1)`, `(2020-07-09, 2)`, et ainsi de suite `(2020-07-09, 200)`. Puisque vous randomisez la clé de partition, les écritures quotidiennes sur la table sont réparties uniformément entre plusieurs partitions. Cela permet un meilleur parallélisme et un débit général plus élevé.

Toutefois, pour lire toutes les lignes d'un jour donné, vous devez rechercher toutes les partitions dans les lignes, puis fusionner les résultats. Par exemple, vous devez d'abord émettre une `SELECT` instruction pour la valeur de la clé de partition `(2020-07-09, 1)`. Émettez ensuite une autre `SELECT` déclaration pour `(2020-07-09, 2)`, et ainsi de suite, par le biais `(2020-07-09, 200)`. Enfin, votre application devra fusionner les résultats de toutes ces `SELECT` instructions.

Sharding à l'aide de clés de partition composées et de valeurs calculées

Une stratégie de randomisation peut considérablement améliorer le débit d'écriture. Mais il est difficile de lire une ligne spécifique car vous ne savez pas quelle valeur a été écrite dans la `shard` colonne au moment où la ligne a été écrite. Pour faciliter la lecture des lignes individuelles, vous pouvez utiliser une autre stratégie. Au lieu d'utiliser un nombre aléatoire pour répartir les lignes entre les partitions, utilisez un nombre que vous pouvez calculer en fonction de l'élément sur lequel vous souhaitez effectuer une requête.

Considérons l'exemple précédent, dans lequel une table utilise la date du jour dans la clé de partition. Supposons maintenant que chaque ligne possède une `title` colonne accessible et que vous ayez le plus souvent besoin de rechercher les lignes par titre en plus de leur date. Avant que votre application n'écrive la ligne dans la table, elle peut calculer une valeur de hachage en fonction du titre et l'utiliser pour remplir la `shard` colonne. Le calcul peut se traduire par un nombre compris entre 1 et 200 assez uniformément distribué, à l'image de ce que la stratégie de randomisation produit.

Un simple calcul suffirait probablement, tel que le produit des valeurs des points de UTF-8 code pour les caractères du titre, modulo 200, + 1. La valeur de la clé de partition composée serait alors la combinaison de la date et du résultat du calcul.

Avec cette stratégie, les écritures sont réparties uniformément entre les valeurs de clé de partition, et de ce fait entre les partitions physiques. Vous pouvez facilement exécuter une `SELECT` instruction pour une ligne et une date spécifiques, car vous pouvez calculer la valeur de la clé de partition pour une `title` valeur spécifique.

Pour lire toutes les lignes d'un jour donné, vous devez toujours SELECT chacune des (2020-07-09, N) clés (1 N à 200), et votre application doit ensuite fusionner tous les résultats. L'avantage est que vous évitez qu'une valeur de clé de partition « critique » ne prenne l'ensemble de la charge de travail.

Optimisation des coûts des tables Amazon Keyspaces

Cette section décrit les meilleures pratiques pour optimiser les coûts de vos tables Amazon Keyspaces existantes. Vous devez examiner les stratégies suivantes pour déterminer la stratégie d'optimisation des coûts qui répond le mieux à vos besoins et les aborder de manière itérative. Chaque stratégie fournit une vue d'ensemble de ce qui pourrait avoir un impact sur vos coûts, la manière de rechercher des opportunités d'optimisation des coûts et des conseils prescriptifs sur la manière de mettre en œuvre ces meilleures pratiques pour vous aider à économiser.

Rubriques

- [Évaluer les coûts au niveau de la table](#)
- [Évaluez le mode capacité de votre table](#)
- [Évaluez les paramètres Application Auto Scaling de votre table](#)
- [Identifiez vos ressources inutilisées pour optimiser les coûts dans Amazon Keyspaces](#)
- [Évaluez les modèles d'utilisation de vos tables pour optimiser les performances et les coûts](#)
- [Évaluer la capacité allouée pour un dimensionnement approprié](#)

Évaluer les coûts au niveau de la table

L'outil Cost Explorer qui se trouve dans le Console de gestion AWS permet de visualiser les coûts ventilés par type, par exemple les frais de lecture, d'écriture, de stockage et de sauvegarde. Vous pouvez également voir ces coûts résumés par période, par mois ou par jour.

L'un des problèmes courants de Cost Explorer est que vous ne pouvez pas facilement consulter les coûts d'une seule table en particulier, car Cost Explorer ne vous permet pas de filtrer ou de regrouper les coûts d'une table spécifique. Vous pouvez consulter la taille métrique de la table facturable (octets) de chaque table dans l'onglet Monitor de la console Amazon Keyspaces. Si vous avez besoin de plus d'informations relatives aux coûts par table, cette section explique comment utiliser le [balisage](#) pour effectuer une analyse des coûts de table individuelle dans Cost Explorer.

Rubriques

- [Comment consulter les coûts d'un seul tableau Amazon Keyspaces](#)
- [Vue par défaut de Cost Explorer](#)
- [Comment utiliser et appliquer des balises de table dans Cost Explorer](#)

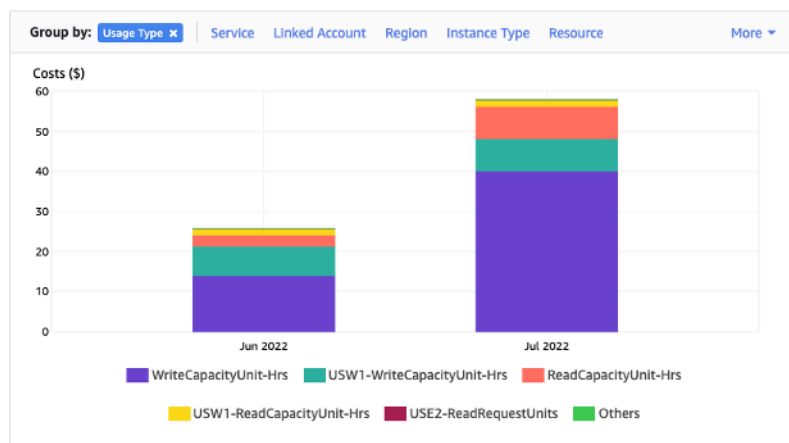
Comment consulter les coûts d'un seul tableau Amazon Keyspaces

Vous pouvez consulter des informations de base sur une table Amazon Keyspaces dans la console, notamment le schéma de clé primaire, la taille de la table facturable et les statistiques relatives à la capacité. Vous pouvez utiliser la taille de la table pour calculer le coût de stockage mensuel de la table. Par exemple, 0,25\$ par Go dans l'est des États-Unis (Virginie du Nord) Région AWS.

Si la table utilise le mode capacité provisionnée, les paramètres actuels de l'unité de capacité de lecture (RCU) et de l'unité de capacité d'écriture (WCU) sont également renvoyés. Vous pouvez utiliser ces informations pour calculer les coûts actuels de lecture et d'écriture de la table. Notez que ces coûts peuvent changer, en particulier si vous avez configuré le tableau avec le dimensionnement automatique d'Amazon Keyspaces.

Vue par défaut de Cost Explorer

La vue par défaut de Cost Explorer fournit des graphiques indiquant le coût des ressources consommées, par exemple le débit et le stockage. Vous pouvez choisir de regrouper ces coûts par période, par exemple les totaux par mois ou par jour. Les coûts de stockage, de lecture, d'écriture et d'autres catégories peuvent également être ventilés et comparés.



Comment utiliser et appliquer des balises de table dans Cost Explorer

Par défaut, Cost Explorer ne fournit pas de résumé des coûts pour une table spécifique, car il combine les coûts de plusieurs tables pour obtenir un total. Vous pouvez toutefois utiliser le [balisage](#)

[des ressources AWS](#) pour identifier chaque table à l'aide d'une balise de métadonnées. Les balises sont des paires clé-valeur que vous pouvez utiliser à diverses fins, par exemple pour identifier toutes les ressources appartenant à un projet ou à un département. Pour plus d'informations, consultez [the section called "Utilisation des tags"](#).

Pour cet exemple, nous utilisons une table portant le nom MyTable.

1. Définissez une balise avec la clé `table_name` et la valeur de `MyTable`
2. [Activez la balise dans Cost Explorer](#), puis filtrez sur la valeur de la balise pour obtenir une meilleure visibilité sur les coûts de chaque table.

Note

Cela peut prendre un ou deux jours pour que la balise commence à apparaître dans Cost Explorer

Vous pouvez définir vous-même les balises de métadonnées dans la console ou par programmation à l'aide de CQL AWS CLI, du SDK ou du SDK. AWS Envisagez d'exiger la définition d'une balise `table_name` dans le cadre du nouveau processus de création de tables de votre organisation. Pour plus d'informations, consultez [the section called "Création de rapports de répartition des coûts"](#).

Évaluez le mode capacité de votre table

Cette section explique comment sélectionner le mode de capacité approprié pour votre table Amazon Keyspaces. Chaque mode est réglé de façon à répondre aux besoins d'une charge de travail différente en termes de réactivité face à l'évolution du débit, ainsi que de facturation de cette utilisation. Vous devez tenir compte de ces facteurs lorsque vous prenez votre décision.

Rubriques

- [Quels sont les modes de capacité de table disponibles ?](#)
- [Quand sélectionner le mode de capacité à la demande ?](#)
- [Quand sélectionner le mode de capacité provisionnée ?](#)
- [Autres facteurs à prendre en compte lors du choix d'un mode de capacité de table](#)

Quels sont les modes de capacité de table disponibles ?

Lorsque vous créez une table Amazon Keyspaces, vous devez sélectionner le mode de capacité à la demande ou le mode de capacité provisionnée. Pour plus d'informations, consultez [the section called "Configuration des modes read/write de capacité"](#).

On-demand mode capacité

Le mode de capacité à la demande est conçu pour éliminer le besoin de planifier ou de provisionner la capacité de votre table Amazon Keyspaces. Dans ce mode, votre table répond instantanément aux demandes sans qu'il soit nécessaire d'augmenter ou de diminuer les ressources (jusqu'à deux fois le débit maximal précédent de la table).

On-demand les tables sont facturées en comptant le nombre de demandes réelles par rapport à la table, de sorte que vous ne payez que pour ce que vous utilisez plutôt que pour ce qui a été provisionné.

Mode de capacité provisionnée

Le mode de capacité provisionnée est un modèle plus traditionnel dans lequel vous pouvez définir la capacité dont la table dispose pour les demandes, soit directement, soit à l'aide d'Application Auto Scaling. Dans la mesure où une capacité spécifique est mise en service pour la table à tout moment, la facturation est basée sur la capacité fournie plutôt que sur le nombre de demandes. Le dépassement de la capacité allouée peut également entraîner le rejet des demandes par la table et réduire l'expérience des utilisateurs de votre application.

Le mode de capacité provisionnée nécessite un équilibre entre le fait de ne pas surprovisionner ou sous-provisionner la table pour atteindre les deux objectifs, la faible occurrence d'erreurs de capacité de débit insuffisante et l'optimisation des coûts.

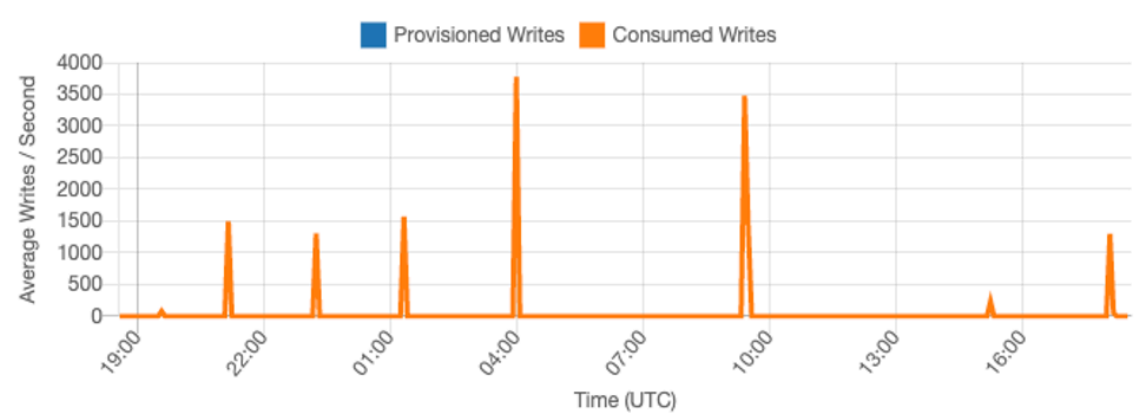
Quand sélectionner le mode de capacité à la demande ?

Lorsque vous optimisez en fonction des coûts, le mode à la demande est votre meilleur choix lorsque vous avez une charge de travail imprévisible similaire à celle illustrée dans le graphique suivant.

Les facteurs suivants contribuent à ce type de charge de travail :

- Caractère imprévisible des demandes (entraînant des pics de trafic)
- Volume variable des demandes (en raison des charges de travail par lots)

- Baisse à zéro ou en dessous de 18 % du pic pendant une heure donnée (en raison des environnements de développement ou de test)



Pour les charges de travail présentant les caractéristiques ci-dessus, l'utilisation d'Application Auto Scaling pour maintenir une capacité suffisante pour que la table puisse répondre aux pics de trafic peut entraîner des résultats indésirables. Il se peut que la table soit surprovisionnée et coûte plus cher que nécessaire, soit qu'elle soit sous-approvisionnée et que les demandes entraînent des erreurs de débit inutiles liées à une faible capacité. Dans de tels cas, les tables à la demande sont le meilleur choix.

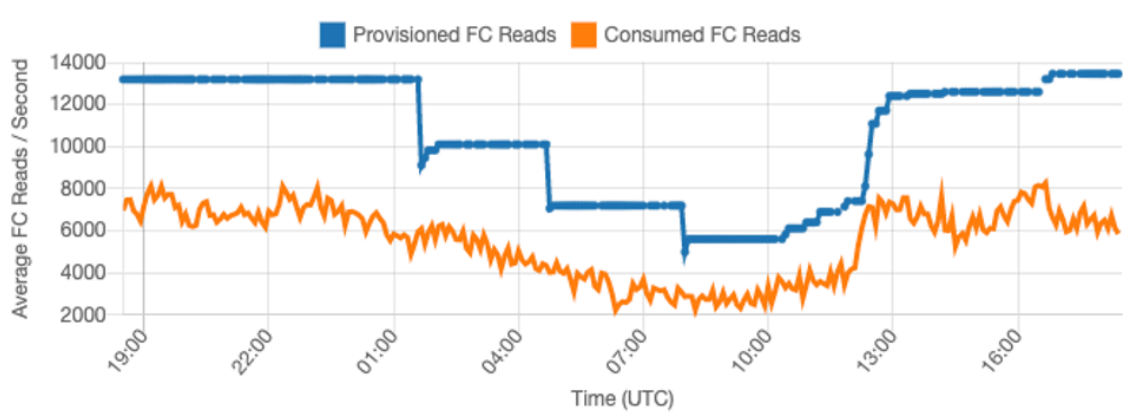
Les tables à la demande étant facturées sur demande, vous n'avez rien d'autre à faire au niveau des tables pour optimiser les coûts. Vous devez régulièrement évaluer vos tables à la demande pour vérifier que la charge de travail présente toujours les caractéristiques ci-dessus. Si la charge de travail s'est stabilisée, envisagez de passer en mode provisionné afin de maintenir l'optimisation des coûts.

Quand sélectionner le mode de capacité provisionnée ?

Une charge de travail idéale pour le mode de capacité allouée est une charge de travail dont le modèle d'utilisation est plus prévisible, comme le montre le graphique ci-dessous.

Les facteurs suivants contribuent à une charge de travail prévisible :

- Predicable/cyclical trafic pour une heure ou une journée donnée
- Rafales de trafic limitées à court terme



Étant donné que les volumes de trafic sont plus stables à une heure ou à une journée donnée, vous pouvez définir la capacité allouée de manière relativement proche de la capacité réellement consommée de la table. L'optimisation des coûts d'un tableau des capacités provisionnées consiste en fin de compte à faire en sorte que la capacité provisionnée (ligne bleue) soit aussi proche que possible de la capacité consommée (ligne orange) sans augmenter le nombre d'`ThrottledRequests` événements associés au tableau. L'espace entre les deux lignes représente à la fois une perte de capacité et une assurance contre une mauvaise expérience utilisateur due à des erreurs de capacité de débit insuffisante.

Amazon Keyspaces fournit Application Auto Scaling pour les tables de capacité provisionnées, qui équilibre automatiquement ces données en votre nom. Vous pouvez suivre votre capacité consommée tout au long de la journée et configurer la capacité provisionnée de la table en fonction de quelques variables.

Unités de capacité minimale

Vous pouvez définir la capacité minimale d'une table pour limiter les erreurs de capacité de débit insuffisante, mais cela ne réduit pas le coût de la table. Si votre table connaît des périodes de faible utilisation suivies d'une augmentation soudaine d'utilisation, le fait de définir la valeur minimale peut empêcher Application Auto Scaling de définir une capacité trop faible de la table.

Unité de capacité maximale

Vous pouvez définir la capacité maximale d'une table afin de limiter la mise à l'échelle d'une table en utilisant une valeur supérieure à celle prévue. Envisagez d'appliquer un maximum pour les tables de développement ou de test, où les tests de charge à grande échelle ne sont pas souhaités. Vous pouvez définir un maximum pour n'importe quelle table, mais veillez à évaluer régulièrement ce paramètre par rapport à la base de référence du tableau lorsque vous l'utilisez en production, afin d'éviter les erreurs accidentelles liées à une capacité de débit insuffisante.

Utilisation cible

La définition de l'utilisation cible de la table est le principal moyen d'optimiser les coûts pour une table à capacité provisionnée. La définition d'une valeur en pourcentage inférieure augmente le surprovisionnement de la table, ce qui augmente les coûts, mais réduit le risque d'erreurs liées à une capacité de débit insuffisante. La définition d'une valeur de pourcentage plus élevée réduit le surprovisionnement de la table, mais augmente le risque d'erreurs liées à une capacité de débit insuffisante.

Autres facteurs à prendre en compte lors du choix d'un mode de capacité de table

Au moment de choisir entre les deux modes de capacité, certains facteurs supplémentaires méritent d'être pris en compte.

Lorsque vous choisissez entre les deux modes de table, considérez dans quelle mesure cette réduction supplémentaire affecte le coût de la table. Dans de nombreux cas, même une charge de travail relativement imprévisible peut être plus rentable à exécuter sur une table de capacité provisionnée surdimensionnée avec des capacités réservées.

Améliorer la prévisibilité de votre charge de travail

Dans certaines situations, une charge de travail peut apparemment présenter à la fois un schéma prévisible et un schéma imprévisible. Bien que cela puisse être facilement pris en charge par un tableau à la demande, les coûts seront probablement inférieurs si les modèles imprévisibles de la charge de travail peuvent être améliorés.

Les importations par lots sont l'une des causes les plus fréquentes de ces tendances. Ce type de trafic peut souvent dépasser la capacité de base de la table à un point tel que des erreurs de capacité de débit insuffisante se produiraient en cas d'exécution. Pour qu'une charge de travail comme celle-ci soit exécutée sur une table à capacité provisionnée, considérez les options suivantes :

- Si le lot se produit à des heures planifiées, vous pouvez planifier une augmentation de la capacité d'autodimensionnement de votre application avant son exécution.
- Si le lot se produit de manière aléatoire, pensez à essayer de prolonger le temps d'exécution plutôt que de l'exécuter le plus rapidement possible.
- Ajoutez une période d'accélération à l'importation, au cours de laquelle la vitesse de l'importation commence lentement mais augmente lentement en quelques minutes jusqu'à ce qu'Application Auto Scaling ait eu l'opportunité de commencer à ajuster la capacité de la table.

Évaluez les paramètres Application Auto Scaling de votre table

Cette section explique comment évaluer les paramètres Application Auto Scaling de vos tables Amazon Keyspaces. [Amazon Keyspaces Application Auto Scaling](#) est une fonctionnalité qui gère le débit des tables en fonction du trafic de votre application et de votre indicateur d'utilisation cible. Cela garantit que vos tables disposent de la capacité requise pour vos modèles d'application.

Le service Application Auto Scaling surveille l'utilisation actuelle de votre table et la compare à la valeur d'utilisation cible :TargetValue. Il vous indique s'il est temps d'augmenter ou de diminuer la capacité allouée.

Rubriques

- [Comprendre les paramètres de votre Application Auto Scaling](#)
- [Comment identifier les tables présentant un faible taux d'utilisation cible \(<= 50 %\)](#)
- [Comment gérer les charges de travail liées aux variations saisonnières](#)
- [Comment gérer les pics de charge de travail imprévisibles](#)
- [Comment gérer les charges de travail avec des applications liées](#)

Comprendre les paramètres de votre Application Auto Scaling

Pour définir la valeur correcte correspondant à l'utilisation cible, ainsi que l'étape initiale et les valeurs finales, vous devez faire appel à l'équipe des opérations. Cela vous permet de définir correctement les valeurs en fonction de l'utilisation historique de l'application, qui est utilisée pour déclencher les politiques Application Auto Scaling. L'objectif d'utilisation est le pourcentage de votre capacité totale qui doit être atteint pendant un certain temps avant que les règles d'Application Auto Scaling ne s'appliquent.

Lorsque vous définissez un objectif d'utilisation élevé (un objectif d'environ 90 %), cela signifie que votre trafic doit être supérieur à 90 % pendant un certain temps avant que l'Application Auto Scaling ne soit activée. Il est préférable de ne pas utiliser une cible d'utilisation élevée, sauf si votre application est très constante et ne fait l'objet d'aucun pic de trafic.

Lorsque vous définissez un taux d'utilisation très faible (un objectif inférieur à 50 %), cela signifie que votre application doit atteindre 50 % de la capacité allouée avant de déclencher une politique Application Auto Scaling. À moins que le trafic de vos applications n'augmente à un rythme très soutenu, cela se traduit généralement par une capacité inutilisée et un gaspillage de ressources.

Comment identifier les tables présentant un faible taux d'utilisation cible ($\leq 50\%$)

Vous pouvez utiliser le AWS CLI ou Console de gestion AWS pour surveiller et identifier les TargetValues politiques de votre Application Auto Scaling dans vos ressources Amazon Keyspaces.

Note

Lorsque vous utilisez des tables multirégionales en mode capacité allouée avec le dimensionnement automatique d'Amazon Keyspaces, veillez à utiliser les opérations de l'API Amazon Keyspaces pour configurer le dimensionnement automatique. Les opérations d'API Application Auto Scaling sous-jacentes qu'Amazon Keyspaces appelle en votre nom ne disposent pas de fonctionnalités multirégionales. Pour plus d'informations, consultez [the section called “Afficher les paramètres de capacité provisionnée et de dimensionnement automatique pour une table multirégionale”](#).

AWS CLI

1. Pour afficher la liste complète des ressources, exécutez la commande suivante :

```
aws application-autoscaling describe-scaling-policies --service-namespace  
cassandra
```

Cette commande renverra la liste complète des politiques Application Auto Scaling émises pour n'importe quelle ressource Amazon Keyspaces. Pour récupérer uniquement les ressources d'une table particulière, vous pouvez ajouter le `--resource-id` parameter. Par exemple :

```
aws application-autoscaling describe-scaling-policies --service-namespace  
cassandra --resource-id "keyspace/keyspace-name/table/table-name"
```

2. Renvoie uniquement les politiques de dimensionnement automatique pour une table particulière en exécutant la commande suivante

```
aws application-autoscaling describe-scaling-policies --service-namespace  
cassandra --resource-id "keyspace/keyspace-name/table/table-name"
```

Les valeurs des politiques Application Auto Scaling sont mises en évidence ci-dessous. Vous devez vous assurer que la valeur cible est supérieure à 50 % pour éviter le surprovisionnement. Le résultat doit ressembler à ce qui suit :

```
{
  "ScalingPolicies": [
    {
      "PolicyARN": "arn:aws:autoscaling:us-
east-1:111122223333:scalingPolicy:<uuid>:resource/keyspaces/table/table-name-
scaling-policy",
      "PolicyName": "$<full-table-name>",
      "ServiceNamespace": "cassandra",
      "ResourceId": "keyspace/keyspace-name/table/table-name",
      "ScalableDimension": "cassandra:table:WriteCapacityUnits",
      "PolicyType": "TargetTrackingScaling",
      "TargetTrackingScalingPolicyConfiguration": {
        "TargetValue": 70.0,
        "PredefinedMetricSpecification": {
          "PredefinedMetricType": "KeyspacesWriteCapacityUtilization"
        }
      },
      "Alarms": [
        ...
      ],
      "CreationTime": "2022-03-04T16:23:48.641000+10:00"
    },
    {
      "PolicyARN": "arn:aws:autoscaling:us-
east-1:111122223333:scalingPolicy:<uuid>:resource/keyspaces/table/table-name-
scaling-policy",
      "PolicyName": "$<full-table-name>",
      "ServiceNamespace": "cassandra",
      "ResourceId": "keyspace/keyspace-name/table/table-name",
      "ScalableDimension": "cassandra:table:ReadCapacityUnits",
      "PolicyType": "TargetTrackingScaling",
      "TargetTrackingScalingPolicyConfiguration": {
        "TargetValue": 70.0,
        "PredefinedMetricSpecification": {
          "PredefinedMetricType": "KeyspacesReadCapacityUtilization"
        }
      },
      "Alarms": [
```

```
    ...  
    ],  
    "CreationTime": "2022-03-04T16:23:47.820000+10:00"  
  }  
]  
}
```

Console de gestion AWS

1. Connectez-vous au Console de gestion AWS et accédez à la page de CloudWatch service sur [Getting Started with the Console de gestion AWS](#). Sélectionnez la solution appropriée Région AWS si nécessaire.
2. Dans le volet de navigation, sélectionnez Tables. Sur la page Tables, sélectionnez le nom de la table.
3. Sur la page Détails de la table de l'onglet Capacity, passez en revue les paramètres Application Auto Scaling de votre table.

Si vos valeurs d'utilisation cibles sont inférieures ou égales à 50 %, explorez les métriques d'utilisation de vos tables pour déterminer si elles sont [sous-provisionnées ou surprovisionnées](#).

Comment gérer les charges de travail liées aux variations saisonnières

Envisagez le scénario suivant : votre application fonctionne en dessous d'une valeur moyenne minimale la plupart du temps, mais la cible d'utilisation est faible. Votre application peut donc réagir rapidement aux événements qui se produisent à certaines heures de la journée et vous disposez d'une capacité suffisante pour éviter les ralentissements. Ce scénario est courant avec les applications qui sont très actives pendant les heures normales de bureau (de 9 h à 17 h), mais qui fonctionnent ensuite à un niveau de base en dehors de cette plage horaire. Étant donné que certains utilisateurs commencent à se connecter avant 9 heures du matin, l'application utilise ce seuil bas pour augmenter rapidement afin d'atteindre la capacité requise aux heures de pointe.

Ce scénario peut se présenter comme suit :

- Entre 17 h et 9 h, les unités ConsumedWriteCapacityUnits restent entre 90 et 100.
- Les utilisateurs commencent à se connecter à l'application avant 9 heures du matin et les unités de capacité augmentent considérablement (la valeur maximale que vous avez vue est de 1 500 WCU).

- En moyenne, l'utilisation de vos applications varie entre 800 et 1 200 pendant les heures de travail.

Si le scénario précédent s'applique à votre application, envisagez d'utiliser le [dimensionnement automatique des applications planifié](#), dans lequel une règle Application Auto Scaling peut toujours être configurée sur votre table, mais avec une utilisation cible moins agressive qui ne fournit la capacité supplémentaire qu'aux intervalles spécifiques dont vous avez besoin.

Vous pouvez utiliser le AWS CLI pour exécuter les étapes suivantes afin de créer une règle de dimensionnement automatique planifiée qui s'exécute en fonction de l'heure du jour et du jour de la semaine.

1. Enregistrez votre table Amazon Keyspaces en tant que cible évolutive avec. Application Auto Scaling Une cible pouvant être mise à l'échelle avec est une ressource dont Application Auto Scaling peut augmenter ou réduire la capacité.

```
aws application-autoscaling register-scalable-target \  
  --service-namespace cassandra \  
  --scalable-dimension cassandra:table:WriteCapacityUnits \  
  --resource-id keyspace/keyspace-name/table/table-name \  
  --min-capacity 90 \  
  --max-capacity 1500
```

2. Configurez les actions planifiées en fonction de vos besoins.

Vous avez besoin de deux règles pour couvrir le scénario : l'une pour augmenter la taille et l'autre pour la réduire. La première règle permettant d'étendre l'action planifiée est illustrée dans l'exemple suivant.

```
aws application-autoscaling put-scheduled-action \  
  --service-namespace cassandra \  
  --scalable-dimension cassandra:table:WriteCapacityUnits \  
  --resource-id keyspace/keyspace-name/table/table-name \  
  --scheduled-action-name my-8-5-scheduled-action \  
  --scalable-target-action MinCapacity=800,MaxCapacity=1500 \  
  --schedule "cron(45 8 ? * MON-FRI *)" \  
  --timezone "Australia/Brisbane"
```

La deuxième règle permettant de réduire l'action planifiée est illustrée dans cet exemple.

```
aws application-autoscaling put-scheduled-action \  
  --service-namespace cassandra \  
  --scalable-dimension cassandra:table:WriteCapacityUnits \  
  --resource-id keyspace/keyspace-name/table/table-name \  
  --scheduled-action-name my-8-5-scheduled-action \  
  --scalable-target-action MinCapacity=800,MaxCapacity=1500 \  
  --schedule "cron(45 8 ? * MON-FRI *)" \  
  --timezone "Australia/Brisbane"
```

```
--service-namespace cassandra \
--scalable-dimension cassandra:table:WriteCapacityUnits \
--resource-id keyspace/keyspace-name/table/table-name \
--scheduled-action-name my-5-8-scheduled-down-action \
--scalable-target-action MinCapacity=90,MaxCapacity=1500 \
--schedule "cron(15 17 ? * MON-FRI *)" \
--timezone "Australia/Brisbane"
```

3. Exécutez la commande suivante pour confirmer que les deux règles ont été activées :

```
aws application-autoscaling describe-scheduled-actions --service-namespace
cassandra
```

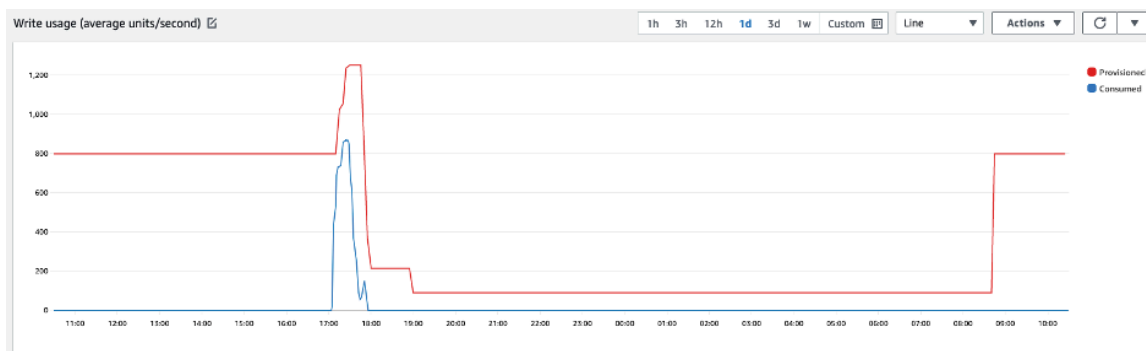
Vous devriez obtenir le résultat suivant :

```
{
  "ScheduledActions": [
    {
      "ScheduledActionName": "my-5-8-scheduled-down-action",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
east-1:111122223333:scheduledAction:<uuid>:resource/keyspaces/table/table-
name:scheduledActionName/my-5-8-scheduled-down-action",
      "ServiceNamespace": "cassandra",
      "Schedule": "cron(15 17 ? * MON-FRI *)",
      "Timezone": "Australia/Brisbane",
      "ResourceId": "keyspace/keyspace-name/table/table-name",
      "ScalableDimension": "cassandra:table:WriteCapacityUnits",
      "ScalableTargetAction": {
        "MinCapacity": 90,
        "MaxCapacity": 1500
      },
      "CreationTime": "2022-03-15T17:30:25.100000+10:00"
    },
    {
      "ScheduledActionName": "my-8-5-scheduled-action",
      "ScheduledActionARN": "arn:aws:autoscaling:us-
east-1:111122223333:scheduledAction:<uuid>:resource/keyspaces/table/table-
name:scheduledActionName/my-8-5-scheduled-action",
      "ServiceNamespace": "cassandra",
      "Schedule": "cron(45 8 ? * MON-FRI *)",
      "Timezone": "Australia/Brisbane",
      "ResourceId": "keyspace/keyspace-name/table/table-name",
      "ScalableDimension": "cassandra:table:WriteCapacityUnits",
```



```
    "ScalableTargetAction": {  
      "MinCapacity": 800,  
      "MaxCapacity": 1500  
    },  
    "CreationTime": "2022-03-15T17:28:57.816000+10:00"  
  }  
]
```

L'image suivante montre un exemple de charge de travail qui maintient en permanence une cible d'utilisation de 70 %. Remarquez que les règles de dimensionnement automatique s'appliquent toujours et que le débit ne diminue pas.



En zoomant, nous pouvons constater qu'un pic dans l'application a déclenché le seuil de 70 %, forçant ainsi la mise à l'échelle automatique à démarrer et à fournir la capacité supplémentaire requise pour la table. L'action de mise à l'échelle automatique planifiée affectera les valeurs maximales et minimales, et il est de votre responsabilité de les configurer.



Comment gérer les pics de charge de travail imprévisibles

Dans ce scénario, l'application utilise un objectif d'utilisation très faible, car vous ne connaissez pas encore les modèles d'application et vous voulez vous assurer que votre charge de travail ne rencontre pas d'erreurs de débit liées à une faible capacité.

Envisagez plutôt d'utiliser le [mode capacité à la demande](#). On-demand les tables sont idéales pour les charges de travail exigeantes où vous ne connaissez pas les modèles de trafic. Avec le mode de capacité à la demande, vous payez à la demande les lectures et écritures de données que votre application effectue sur vos tables. Vous n'avez pas besoin de spécifier le débit de lecture et d'écriture que vous souhaitez que votre application atteigne, car Amazon Keyspaces s'adapte instantanément à vos charges de travail à mesure qu'elles augmentent ou diminuent.

Comment gérer les charges de travail avec des applications liées

Dans ce scénario, l'application dépend d'autres systèmes, tels que les scénarios de traitement par lots dans lesquels vous pouvez avoir de forts pics de trafic en fonction des événements dans la logique de l'application.

Envisagez de développer une logique d'auto-scaling des applications personnalisée qui réagit aux événements susceptibles d'augmenter la capacité des tables et `TargetValues` en fonction de vos besoins spécifiques. Vous pourriez bénéficier Amazon EventBridge et utiliser une combinaison de AWS services tels que `λ` et Step Functions pour répondre aux besoins spécifiques de votre application.

Identifiez vos ressources inutilisées pour optimiser les coûts dans Amazon Keyspaces

Cette section explique comment évaluer régulièrement vos ressources inutilisées. Au fur et à mesure que les exigences de votre application évoluent, vous devez vous assurer qu'aucune ressource n'est inutilisée et qu'elle ne contribue à des coûts inutiles liés à Amazon Keyspaces. Les procédures décrites ci-dessous utilisent CloudWatch les métriques Amazon pour identifier les ressources inutilisées et prendre des mesures pour réduire les coûts.

Vous pouvez surveiller Amazon Keyspaces à l'aide d'Amazon Keyspaces CloudWatch, qui collecte et traite les données brutes d'Amazon Keyspaces pour en faire des indicateurs lisibles en temps quasi réel. Ces statistiques étant conservées pendant un certain temps, vous pouvez accéder aux informations historiques pour acquérir une meilleure compréhension de votre utilisation. Par défaut, les données métriques d'Amazon Keyspaces sont envoyées automatiquement à CloudWatch . Pour plus d'informations, consultez [Qu'est-ce qu'Amazon CloudWatch ?](#) et [conservation des métriques](#) dans le guide de CloudWatch l'utilisateur Amazon.

Rubriques

- [Comment identifier les ressources inutilisées](#)
- [Identification des ressources de table inutilisées](#)
- [Nettoyage des ressources de table inutilisées](#)
- [Nettoyage des sauvegardes de restauration ponctuelle \(PITR\) inutilisées](#)

Comment identifier les ressources inutilisées

Pour identifier les tables inutilisées, vous pouvez examiner les CloudWatch indicateurs suivants sur une période de 30 jours afin de déterminer s'il existe des lectures ou des écritures actives sur une table spécifique :

ConsumedReadCapacityUnits

Nombre d'unités de capacité de lecture consommées sur la période spécifiée, de sorte que vous puissiez suivre la capacité consommée. Vous pouvez récupérer la capacité de lecture totale consommée pour une table.

ConsumedWriteCapacityUnits

Nombre d'unités de capacité d'écriture consommées sur la période spécifiée, de sorte que vous puissiez suivre la capacité consommée. Vous pouvez récupérer la capacité d'écriture totale consommée pour une table.

Identification des ressources de table inutilisées

Amazon CloudWatch est un service de surveillance et d'observabilité qui fournit les statistiques du tableau Amazon Keyspaces que vous pouvez utiliser pour identifier les ressources inutilisées. CloudWatch les métriques peuvent être consultées à Console de gestion AWS la fois par le biais du AWS Command Line Interface.

AWS Command Line Interface

Pour afficher les statistiques de vos tables via le AWS Command Line Interface, vous pouvez utiliser les commandes suivantes.

1. Tout d'abord, évaluez les lectures de votre table :

Note

Si le nom de la table n'est pas unique dans votre compte, vous devez également spécifier le nom du keyspace.

```
aws cloudwatch get-metric-statistics --metric-name  
ConsumedReadCapacityUnits --start-time <start-time> --end-time <end-
```

```
time> --period <period> --namespace AWS/Cassandra --statistics Sum --  
dimensions Name=TableName,Value=<table-name>
```

Pour éviter de faussement identifier une table comme étant inutilisée, évaluez les indicateurs sur une période plus longue. Choisissez une plage d'heures de début et de fin appropriées, par exemple 30 jours, et une période appropriée, telle que 86400.

Dans les données renvoyées, toute somme supérieure à 0 indique que la table que vous évaluez a reçu du trafic de lecture pendant cette période.

Le résultat suivant montre une table recevant du trafic de lecture au cours de la période évaluée :

```
{  
  "Timestamp": "2022-08-25T19:40:00Z",  
  "Sum": 36023355.0,  
  "Unit": "Count"  
},  
{  
  "Timestamp": "2022-08-12T19:40:00Z",  
  "Sum": 38025777.5,  
  "Unit": "Count"  
},
```

Le résultat suivant montre qu'une table n'a pas reçu de trafic de lecture au cours de la période évaluée :

```
{  
  "Timestamp": "2022-08-01T19:50:00Z",  
  "Sum": 0.0,  
  "Unit": "Count"  
},  
{  
  "Timestamp": "2022-08-20T19:50:00Z",  
  "Sum": 0.0,  
  "Unit": "Count"  
},
```

2. Ensuite, évaluez les écritures dans votre table :

```
aws cloudwatch get-metric-statistics --metric-name
```

```
ConsumedWriteCapacityUnits --start-time <start-time> --end-time <end-time> --period <period> --namespace AWS/Cassandra --statistics Sum --dimensions Name=TableName,Value=<table-name>
```

Pour éviter de faussement identifier une table comme étant inutilisée, vous devez évaluer les indicateurs sur une période plus longue. Choisissez une plage d'heure de début et de fin appropriée, par exemple 30 jours, et une période appropriée, telle que 86400.

Dans les données renvoyées, toute somme supérieure à 0 indique que la table que vous évaluez a reçu du trafic de lecture pendant cette période.

Le résultat suivant montre une table recevant du trafic d'écriture au cours de la période évaluée :

```
{
  "Timestamp": "2022-08-19T20:15:00Z",
  "Sum": 41014457.0,
  "Unit": "Count"
},
{
  "Timestamp": "2022-08-18T20:15:00Z",
  "Sum": 40048531.0,
  "Unit": "Count"
},
```

Le résultat suivant montre qu'une table n'a pas reçu de trafic d'écriture au cours de la période évaluée :

```
{
  "Timestamp": "2022-07-31T20:15:00Z",
  "Sum": 0.0,
  "Unit": "Count"
},
{
  "Timestamp": "2022-08-19T20:15:00Z",
  "Sum": 0.0,
  "Unit": "Count"
},
```

Console de gestion AWS

Les étapes suivantes vous permettent d'évaluer l'utilisation de vos ressources à l'aide du Console de gestion AWS.

1. Connectez-vous au Console de gestion AWS et accédez à la page de CloudWatch service à l'adresse <https://console.aws.amazon.com/cloudwatch/>. Sélectionnez l'option appropriée Région AWS en haut à droite de la console, si nécessaire.
2. Dans la barre de navigation de gauche, recherchez la section Mesures et choisissez Toutes les mesures.
3. L'action ci-dessus ouvre un tableau de bord avec deux panneaux. Dans le panneau supérieur, vous pouvez voir les statistiques actuellement représentées graphiquement. En bas, vous pouvez sélectionner les indicateurs disponibles pour le graphe. Choisissez Amazon Keyspaces dans le panneau inférieur.
4. Dans le panneau de sélection des statistiques d'Amazon Keyspaces, choisissez la catégorie Tableau Metrics pour afficher les statistiques relatives à vos tables dans la région actuelle.
5. Identifiez le nom de votre table en faisant défiler le menu vers le bas, puis choisissez les métriques ConsumedReadCapacityUnits et ConsumedWriteCapacityUnits pour votre tableau.
6. Choisissez l'onglet Mesures graphiques (2) et réglez la colonne Statistiques sur Somme.
7. Pour éviter d'identifier à tort une table comme étant inutilisée, évaluez les métriques de la table sur une période plus longue. En haut du panneau graphique, choisissez une période appropriée, par exemple un mois, pour évaluer votre tableau. Choisissez Personnalisé, choisissez 1 mois dans le menu déroulant, puis choisissez Appliquer.
8. Évaluez les métriques représentées graphiquement pour votre table afin de déterminer si elle est utilisée. Les métriques supérieures à 0 indiquent qu'une table a été utilisée pendant la période évaluée. Un graphique plat à 0 en lecture et en écriture indique qu'une table n'est pas utilisée.

Nettoyage des ressources de table inutilisées

Si vous avez identifié des ressources de table inutilisées, vous pouvez réduire leurs coûts permanents de la manière suivante.

 Note

Si vous avez identifié une table non utilisée mais que vous souhaitez la garder à disposition pour un accès futur, pensez à la passer en mode à la demande. Dans le cas contraire, vous pouvez envisager de supprimer le tableau.

Modes de capacité

Amazon Keyspaces facture la lecture, l'écriture et le stockage de données dans vos tables Amazon Keyspaces.

Amazon Keyspaces propose [deux modes de capacité](#), dotés d'options de facturation spécifiques pour le traitement des lectures et des écritures sur vos tables : à la demande et provisionné. Le mode read/write capacité contrôle la façon dont vous êtes facturé pour le débit de lecture et d'écriture et la façon dont vous gérez la capacité.

Pour les tables en mode à la demande, vous devez spécifier le débit de lecture et d'écriture que votre application est supposée atteindre. Amazon Keyspaces vous facture les opérations de lecture et d'écriture effectuées par votre application sur vos tables en termes d'unités de demande de lecture et d'unités de demande d'écriture. S'il n'y a aucune activité sur votre table, vous ne payez pas pour le débit, mais vous devez tout de même payer des frais de stockage.

Suppression des tables

Si vous avez découvert une table inutilisée et que vous souhaitez la supprimer, pensez d'abord à effectuer une sauvegarde ou à exporter les données.

Les sauvegardes effectuées AWS Backup peuvent tirer parti de la hiérarchisation du stockage à froid, ce qui permet de réduire encore les coûts. Reportez-vous à la documentation sur la [gestion des plans de sauvegarde](#) pour savoir comment utiliser un cycle de vie pour transférer votre sauvegarde vers un stockage à froid.

Une fois votre table sauvegardée, vous pouvez choisir de la supprimer via la Console de gestion AWS ou via l' AWS Command Line Interface.

Nettoyage des sauvegardes de restauration ponctuelle (PITR) inutilisées

Amazon Keyspaces propose la Point-in-time restauration, qui fournit des sauvegardes continues pendant 35 jours afin de vous protéger contre les écritures ou les suppressions accidentelles. Les sauvegardes PITR sont associées à des coûts.

Reportez-vous à la documentation [the section called “Backup et restauration avec restauration instantanée”](#) à l'adresse suivante pour déterminer si des sauvegardes sont activées sur vos tables et qu'elles ne sont peut-être plus nécessaires.

Évaluez les modèles d'utilisation de vos tables pour optimiser les performances et les coûts

Cette section explique comment évaluer si vous utilisez efficacement vos tables Amazon Keyspaces. Certains modèles d'utilisation ne sont pas optimaux pour Amazon Keyspaces, et ils peuvent être optimisés tant du point de vue des performances que du point de vue des coûts.

Rubriques

- [Effectuer moins d'opérations de lecture fortement cohérente](#)
- [Activer la durée de vie \(TTL\)](#)

Effectuer moins d'opérations de lecture fortement cohérente

Amazon Keyspaces vous permet de configurer la [cohérence de lecture](#) pour chaque demande. Les demandes de lecture sont « éventuellement cohérentes » par défaut. Finalement, les lectures cohérentes sont facturées à 0,5 RCU pour un maximum de 4 Ko de données.

La plupart des éléments des charges de travail distribuées sont flexibles et peuvent tolérer une éventuelle cohérence. Cependant, certains modèles d'accès peuvent nécessiter des lectures fortement cohérentes. Les lectures très cohérentes sont facturées à 1 RCU pour un maximum de 4 Ko de données, ce qui double essentiellement vos coûts de lecture. Amazon Keyspaces vous offre la possibilité d'utiliser les deux modèles de cohérence sur la même table.

Vous pouvez évaluer votre charge de travail et le code de votre application pour vérifier si les lectures fortement cohérentes ne sont utilisées que lorsque cela est nécessaire.

Activer la durée de vie (TTL)

[Time to Live \(TTL\)](#) vous aide à simplifier la logique de votre application et à optimiser le prix du stockage en expirant automatiquement les données des tables. Les données dont vous n'avez plus besoin sont automatiquement supprimées de votre tableau en fonction de la valeur Time to Live que vous avez définie.

Évaluer la capacité allouée pour un dimensionnement approprié

Cette section explique comment évaluer si vous disposez d'un provisionnement de taille appropriée sur vos tables Amazon Keyspaces. À mesure que votre charge de travail évolue, vous devez modifier vos procédures opérationnelles de manière appropriée, en particulier lorsque votre table Amazon Keyspaces est configurée en mode provisionné et que vous courez le risque de surprovisionner ou de sous-approvisionner vos tables.

Les procédures décrites dans cette section nécessitent des informations statistiques qui doivent être capturées à partir des tables Amazon Keyspaces qui prennent en charge votre application de production. Pour comprendre le comportement de votre application, vous devez définir une période suffisamment importante pour saisir le caractère saisonnier des données de votre application. Par exemple, si votre application repose sur des cycles hebdomadaires, spécifier une période de trois semaines devrait vous laisser suffisamment de marge pour analyser ses besoins en matière de débit.

Si vous ne savez pas par où commencer, utilisez au moins un mois de données pour les calculs ci-dessous.

Lors de l'évaluation de la capacité, pour les tables Amazon Keyspaces, vous pouvez configurer les unités de capacité de lecture (RCU) et les unités de capacité d'écriture (WCU) indépendamment.

Rubriques

- [Comment récupérer les statistiques de consommation à partir de vos tableaux Amazon Keyspaces](#)
- [Comment identifier les tables Amazon Keyspaces sous-approvisionnées](#)
- [Comment identifier les tables Amazon Keyspaces surapprovisionnées](#)

Comment récupérer les statistiques de consommation à partir de vos tableaux Amazon Keyspaces

Pour évaluer la capacité de la table, surveillez les CloudWatch mesures suivantes et sélectionnez la dimension appropriée pour récupérer les informations de la table :

Unités de capacité de lecture	Unités de capacité d'écriture
ConsumedReadCapacityUnits	ConsumedWriteCapacityUnits
ProvisionedReadCapacityUnits	ProvisionedWriteCapacityUnits

Unités de capacité de lecture	Unités de capacité d'écriture
ReadThrottleEvents	WriteThrottleEvents

Vous pouvez le faire par le biais du AWS CLI ou du Console de gestion AWS.

AWS CLI

Avant de récupérer les métriques de consommation du tableau, vous devez commencer par capturer certains points de données historiques à l'aide de l' CloudWatch API.

Commencez par créer deux fichiers : `write-calc.json` et `read-calc.json`. Ces fichiers représentent les calculs de la table. Vous devez mettre à jour certains champs, comme indiqué dans le tableau ci-dessous, pour les adapter à votre environnement.

Note

Si le nom de la table n'est pas unique dans votre compte, vous devez également spécifier le nom du keyspace.

Nom de champ	Définition	Exemple
<table-name>	Le nom de la table que vous analysez	SampleTable
<period>	Période que vous utilisez pour évaluer l'objectif d'utilisation, en secondes	Pour une période d'une heure, vous devez spécifier : 3 600
<start-time>	Début de votre intervalle d'évaluation, spécifié au format ISO8601	21/02/2018 T23:00:00
<end-time>	Fin de votre intervalle d'évaluation, spécifié au format ISO8601	22/02/2018 T06:00:00

Le fichier de calculs d'écriture récupère le nombre de WCU provisionnés et consommés au cours de la période correspondant à la plage de dates spécifiée. Il génère également un pourcentage d'utilisation qui peut être utilisé pour l'analyse. Le contenu complet du `write-calc.json` fichier doit ressembler à celui de l'exemple suivant.

```
{
  "MetricDataQueries": [
    {
      "Id": "provisionedWCU",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/Cassandra",
          "MetricName": "ProvisionedWriteCapacityUnits",
          "Dimensions": [
            {
              "Name": "TableName",
              "Value": "<table-name>"
            }
          ]
        },
        "Period": <period>,
        "Stat": "Average"
      },
      "Label": "Provisioned",
      "ReturnData": false
    },
    {
      "Id": "consumedWCU",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/Cassandra",
          "MetricName": "ConsumedWriteCapacityUnits",
          "Dimensions": [
            {
              "Name": "TableName",
              "Value": "<table-name>"
            }
          ]
        },
        "Period": <period>,
        "Stat": "Sum"
      },
      "Label": "",
```

```

    "ReturnData": false
  },
  {
    "Id": "m1",
    "Expression": "consumedWCU/PERIOD(consumedWCU)",
    "Label": "Consumed WCUs",
    "ReturnData": false
  },
  {
    "Id": "utilizationPercentage",
    "Expression": "100*(m1/provisionedWCU)",
    "Label": "Utilization Percentage",
    "ReturnData": true
  }
],
"StartTime": "<start-time>",
"EndTime": "<end-time>",
"ScanBy": "TimestampDescending",
"MaxDatapoints": 24
}

```

Le fichier de calcul lu utilise une métrique similaire. Ce fichier récupère le nombre de RCU qui ont été provisionnées et consommées pendant la période correspondant à la plage de dates spécifiée. Le contenu du `read-calc.json` fichier doit ressembler à celui de cet exemple.

```

{
  "MetricDataQueries": [
    {
      "Id": "provisionedRCU",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/Cassandra",
          "MetricName": "ProvisionedReadCapacityUnits",
          "Dimensions": [
            {
              "Name": "TableName",
              "Value": "<table-name>"
            }
          ]
        },
        "Period": "<period>",
        "Stat": "Average"
      }
    },
  ],
}

```

```

    "Label": "Provisioned",
    "ReturnData": false
  },
  {
    "Id": "consumedRCU",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/Cassandra",
        "MetricName": "ConsumedReadCapacityUnits",
        "Dimensions": [
          {
            "Name": "TableName",
            "Value": "<table-name>"
          }
        ]
      },
      "Period": <period>,
      "Stat": "Sum"
    },
    "Label": "",
    "ReturnData": false
  },
  {
    "Id": "m1",
    "Expression": "consumedRCU/PERIOD(consumedRCU)",
    "Label": "Consumed RCU",
    "ReturnData": false
  },
  {
    "Id": "utilizationPercentage",
    "Expression": "100*(m1/provisionedRCU)",
    "Label": "Utilization Percentage",
    "ReturnData": true
  }
],
"StartTime": "<start-time>",
"EndTime": "<end-time>",
"ScanBy": "TimestampDescending",
"MaxDatapoints": 24
}

```

Une fois les fichiers créés, vous pouvez commencer à récupérer les données d'utilisation.

1. Pour récupérer les données d'utilisation de l'écriture, exécutez la commande suivante :

```
aws cloudwatch get-metric-data --cli-input-json file://write-calc.json
```

2. Pour récupérer les données d'utilisation en lecture, exécutez la commande suivante :

```
aws cloudwatch get-metric-data --cli-input-json file://read-calc.json
```

Le résultat des deux requêtes est une série de points de données au format JSON qui peuvent être utilisés pour l'analyse. Vos résultats dépendent du nombre de points de données que vous avez spécifiés, de la période et de vos propres données de charge de travail. Cela pourrait ressembler à l'exemple suivant.

```
{
  "MetricDataResults": [
    {
      "Id": "utilizationPercentage",
      "Label": "Utilization Percentage",
      "Timestamps": [
        "2022-02-22T05:00:00+00:00",
        "2022-02-22T04:00:00+00:00",
        "2022-02-22T03:00:00+00:00",
        "2022-02-22T02:00:00+00:00",
        "2022-02-22T01:00:00+00:00",
        "2022-02-22T00:00:00+00:00",
        "2022-02-21T23:00:00+00:00"
      ],
      "Values": [
        91.55364583333333,
        55.066631944444445,
        2.6114930555555556,
        24.9496875,
        40.947256944444445,
        25.618194444444444,
        0.0
      ],
      "StatusCode": "Complete"
    }
  ],
  "Messages": []
}
```

 Note

Si vous spécifiez une courte période et une longue période, vous devrez peut-être modifier la `MaxDatapoints` valeur, qui est définie par défaut sur 24 dans le script. Cela représente un seul point de données par heure et donc 24 points de données par jour.

Console de gestion AWS

1. Connectez-vous au Console de gestion AWS et accédez à la page de CloudWatch service sur [Getting Started with the Console de gestion AWS](#). Sélectionnez la solution appropriée Région AWS si nécessaire.
2. Localisez la section Mesures dans la barre de navigation de gauche et choisissez Toutes les mesures.
3. Cela ouvre un tableau de bord avec deux panneaux. Le panneau supérieur affiche le graphique, tandis que le panneau inférieur contient les indicateurs que vous souhaitez représenter graphiquement. Choisissez le panneau Amazon Keyspaces.
4. Choisissez la catégorie Table Metrics dans les sous-panneaux. Cela vous montre les tables de votre compte actuel Région AWS.
5. Identifiez le nom de votre table en faisant défiler le menu vers le bas, puis sélectionnez les métriques des opérations d'écriture : `ConsumedWriteCapacityUnits` et `ProvisionedWriteCapacityUnits`.

 Note

Cet exemple décrit les métriques des opérations d'écriture, mais vous pouvez également suivre ces étapes pour représenter graphiquement les métriques des opérations de lecture.

6. Sélectionnez l'onglet Graphed metrics (2) (Indicateurs graphiques (2)) pour modifier les formules. Par défaut, CloudWatch choisit la fonction statistique Average pour les graphiques.
7. Lorsque les deux mesures représentées dans le graphique sont sélectionnées (case à cocher sur la gauche), sélectionnez le menu Add math (Ajouter une formule), suivi de Common (Commun), puis sélectionnez la fonction Percentage (Pourcentage). Répétez cette procédure deux fois.

C'est la première fois que vous sélectionnez la fonction Pourcentage.

Sélection de la fonction Pourcentage pour la deuxième fois.

8. À ce stade, vous devriez avoir quatre métriques dans le menu inférieur. Travaillons sur le calcul de `ConsumedWriteCapacityUnits`. Par souci de cohérence, vous devez faire correspondre les noms à ceux que vous avez utilisés dans la AWS CLI section. Cliquez sur l'ID m1 et remplacez cette valeur par `consumedWCU`.
9. Remplacez la statistique Average (Moyenne) par Sum (Somme). Cette action crée automatiquement une autre métrique appelée `ANOMALY_DETECTION_BAND`. Dans le cadre de cette procédure, vous pouvez l'ignorer en décochant la case sur la métrique ad1 nouvellement générée.
10. Répétez l'étape 8 pour remplacer le nom de l'ID m2 par `provisionedWCU`. Conservez la statistique Average (Moyenne).
11. Choisissez l'étiquette Expression1 et mettez à jour la valeur en m1 et l'étiquette en WCU consommées.

 Note

Assurez-vous de n'avoir sélectionné que m1 (case à cocher sur la gauche) et `provisionedWCU` pour visualiser correctement les données. Mettez à jour la formule en cliquant sur Détails et en la remplaçant par `consumedWCU/PERIOD(ConsumedWCU)`. Cette étape peut également générer une autre métrique `ANOMALY_DETECTION_BAND`, mais dans le cadre de cette procédure, vous pouvez l'ignorer.

12. Vous devriez maintenant avoir deux graphiques : l'un qui indique vos WCU provisionnés dans le tableau et l'autre qui indique les WCU consommés.
13. Mettez à jour la formule du pourcentage en sélectionnant le graphique Expression2 (e2). Remplacez le nom des libellés et des identifiants par `utilizationPercent`. Renommez la formule pour qu'elle corresponde à $100 * () \text{ m1/provisionedWCU}$.
14. Supprimez la case à cocher de toutes les métriques, à l'exception de `UtilizationPercentage`, pour visualiser vos modèles d'utilisation. L'intervalle par défaut est fixé à 1 minute, mais n'hésitez pas à le modifier selon vos besoins.

Les résultats que vous obtenez dépendent des données réelles de votre charge de travail. Les intervalles d'utilisation supérieurs à 100 % sont sujets à des événements d'erreur liés à une faible capacité de débit. Amazon Keyspaces propose une [capacité en rafale](#), mais dès qu'elle est

épuisée, toute valeur supérieure à 100 % est confrontée à des événements d'erreur liés à une faible capacité de débit.

Comment identifier les tables Amazon Keyspaces sous-approvisionnées

Pour la plupart des charges de travail, une table est considérée comme sous-provisionnée lorsqu'elle consomme constamment plus de 80 % de sa capacité provisionnée.

La [capacité en rafale](#) est une fonctionnalité d'Amazon Keyspaces qui permet aux clients de consommer temporairement RCUs/WCUs plus que ce qui était initialement prévu (plus que le débit provisionné par seconde défini dans le tableau). La capacité de débordement a été créée pour absorber les augmentations soudaines du trafic dues à des événements spéciaux ou à des pics d'utilisation. Cette capacité de rafale est limitée. Pour plus d'informations, voir [the section called “Utiliser la capacité de rafale”](#). Dès que les RCU et WCU inutilisés sont épuisés, vous pouvez rencontrer des erreurs de débit en cas de faible capacité si vous essayez de consommer plus de capacité que celle allouée. Lorsque le trafic de votre application approche le taux d'utilisation de 80 %, le risque de rencontrer des erreurs liées au débit de faible capacité est nettement plus élevé.

La règle du taux d'utilisation de 80 % varie en fonction de la saisonnalité de vos données et de la croissance du trafic. Réfléchissez aux scénarios suivants :

- Si le trafic est resté stable à un taux d'utilisation d'environ 90 % au cours des 12 derniers mois, votre table dispose de la capacité idéale
- Si le trafic de vos applications augmente à un rythme de 8 % par mois en moins de 3 mois, vous allez atteindre une utilisation de 100 %
- Si le trafic de vos applications augmente à un rythme de 5 % en un peu plus de 4 mois, vous allez tout de même atteindre une utilisation de 100 %

Les résultats des requêtes ci-dessus donnent une idée de votre taux d'utilisation. Utilisez-les comme guide pour évaluer plus en détail d'autres métriques qui pourront vous aider à choisir d'augmenter la capacité de votre table selon vos besoins (par exemple, à un taux de croissance mensuel ou hebdomadaire). Travaillez avec l'équipe des opérations pour définir le pourcentage approprié pour votre charge de travail et vos tables.

Il existe des scénarios spéciaux dans lesquels les données sont faussées lorsque vous les analysez sur une base quotidienne ou hebdomadaire. Par exemple, dans le cas des applications saisonnières dont l'utilisation augmente pendant les heures de travail (mais qui tombent ensuite à presque zéro

en dehors des heures de travail), vous pourriez tirer parti de la [planification de l'auto-scaling des applications](#), dans laquelle vous spécifiez les heures de la journée (et les jours de la semaine) pour augmenter la capacité allouée, ainsi que le moment de la réduire. Au lieu de viser une capacité accrue afin de couvrir les heures de pointe, vous pouvez également bénéficier des configurations d'[auto-scaling des tables Amazon Keyspaces](#) si votre saisonnalité est moins prononcée.

Comment identifier les tables Amazon Keyspaces surapprovisionnées

Les résultats de requête obtenus à partir des scripts ci-dessus fournissent les points de données nécessaires pour effectuer une analyse initiale. Si votre ensemble de données présente des valeurs d'utilisation inférieures à 20 % pendant plusieurs intervalles, votre table est peut-être surprovisionnée. Pour déterminer plus précisément si vous devez réduire le nombre de WCU et de RCU, réexaminez les autres lectures dans les intervalles.

Lorsque votre table contient plusieurs intervalles d'utilisation réduits, vous pouvez tirer parti des politiques Application Auto Scaling, soit en planifiant Application Auto Scaling, soit en configurant simplement les politiques Application Auto Scaling par défaut pour la table en fonction de l'utilisation.

Si votre charge de travail présente un faible rapport utilisation/accélération élevé (Max () /Min (ThrottleEvents) dans l'intervalleThrottleEvents), cela peut se produire lorsque votre charge de travail est très élevée et que le trafic augmente de manière significative certains jours (ou à certaines heures de la journée), mais qu'il est par ailleurs constamment faible. Dans ces scénarios, il peut être avantageux d'utiliser [Application Auto Scaling programmé](#).

Résolution des problèmes liés à Amazon Keyspaces (pour Apache Cassandra)

Ce guide décrit les étapes de résolution des problèmes pour différents scénarios liés à l'utilisation d'Amazon Keyspaces (pour Apache Cassandra). Il contient des informations sur la résolution des erreurs générales, des problèmes de connexion, des problèmes de gestion des capacités et des erreurs du langage de définition des données (DDL).

- Erreurs générales
 - Résolution des exceptions de haut niveau telles que `NoNodeAvailableException`, `NoHostAvailableException`, et `AllNodesFailedException`
 - Isoler les erreurs sous-jacentes des exceptions du pilote Java.
 - Implémentation de politiques de nouvelle tentative et configuration correcte des connexions.
- Problèmes de connexion
 - Résolution des erreurs lors de la connexion aux points de terminaison Amazon Keyspaces à l'aide `cqlsh` de pilotes clients Apache Cassandra.
 - Résolution des problèmes liés aux connexions des points de terminaison VPC, aux Cassandra-stress connexions et aux erreurs de configuration IAM.
 - Gestion des pertes de connexion lors des importations de données.
- Erreurs liées à la gestion des capacités
 - Reconnaissance et résolution des erreurs de capacité insuffisante liées aux tables, aux partitions et aux connexions.
 - Surveillance des statistiques Amazon Keyspaces pertinentes dans Amazon CloudWatch Logs.
 - Optimisation des connexions et du débit pour améliorer les performances.
- Erreurs liées au langage de définition des données (DDL)
 - Résolution des erreurs lors de la création, de l'accès ou de la restauration des espaces clés et des tables.
 - Gestion des défaillances liées aux paramètres personnalisés de durée de vie (TTL), aux limites de colonnes et aux suppressions de plages.
 - Considérations relatives aux lourdes charges de travail de suppression.

Pour obtenir des conseils de dépannage spécifiques à l'accès IAM, consultez [the section called “Résolution des problèmes”](#). Pour plus d'informations sur les meilleures pratiques en matière de sécurité, consultez [the section called “Bonnes pratiques de sécurité”](#).

Rubriques

- [Résolution des erreurs générales dans Amazon Keyspaces](#)
- [Résolution des erreurs de connexion dans Amazon Keyspaces](#)
- [Résolution des erreurs de gestion des capacités dans Amazon Keyspaces](#)
- [Résolution des erreurs linguistiques liées à la définition des données dans Amazon Keyspaces](#)

Résolution des erreurs générales dans Amazon Keyspaces

Vous recevez des erreurs générales ? Voici quelques problèmes courants et comment les résoudre.

Erreurs générales

Vous êtes confronté à l'une des exceptions de premier niveau suivantes, qui peut se produire pour de nombreuses raisons différentes.

- `NoNodeAvailableException`
- `NoHostAvailableException`
- `AllNodesFailedException`

Ces exceptions sont générées par le pilote client et peuvent se produire soit lorsque vous établissez la connexion de contrôle, soit lorsque vous effectuez des requêtes `read/write/prepare/execute/batch`.

Lorsque l'erreur survient alors que vous établissez la connexion de contrôle, cela signifie que tous les points de contact spécifiés dans votre application sont inaccessibles. Lorsque l'erreur se produit lors de l'exécution `prepare/execute` des requêtes `read/write/`, cela indique que toutes les tentatives pour cette demande ont été épuisées. Chaque nouvelle tentative est tentée sur un nœud différent lorsque vous utilisez la politique de nouvelle tentative par défaut.

Comment isoler l'erreur sous-jacente des exceptions du pilote Java de haut niveau

Ces erreurs générales peuvent être causées soit par des problèmes de connexion, soit lors de l'exécution d'`prepare/execute` opérations `read/write` /. Des défaillances transitoires sont prévisibles dans les systèmes distribués et doivent être traitées en réessayant la demande. Le pilote Java ne

réessaie pas automatiquement en cas d'erreur de connexion. Il est donc recommandé de mettre en œuvre la politique de nouvelle tentative lors de l'établissement de la connexion du pilote dans votre application. Pour un aperçu détaillé des meilleures pratiques en matière de connexion, voir [the section called “Connexions”](#).

Par défaut, le pilote Java est défini sur `false` pour toutes les demandes, ce qui signifie qu'il ne réessaie pas automatiquement la demande `read/write /prepare` qui a échoué. Pour configurer `true` et demander au pilote de réessayer les demandes qui ont échoué, vous pouvez le faire de différentes manières. Voici un exemple de la façon dont vous pouvez définir l'idempotence par programmation pour une seule requête dans votre application Java.

```
Statement s = new SimpleStatement("SELECT * FROM my_table WHERE id = 1");
s.setIdempotent(true);
```

Vous pouvez également définir l'idempotence par défaut pour l'ensemble de votre application Java par programmation, comme indiqué dans l'exemple suivant.

```
// Make all statements idempotent by default:
cluster.getConfiguration().getQueryOptions().setDefaultIdempotence(true);
//Set the default idempotency to true in your Cassandra configuration
basic.request.default-idempotence = true
```

Une autre recommandation consiste à créer une politique de nouvelle tentative au niveau de l'application. Dans ce cas, l'application doit intercepter la demande `NoNodeAvailableException` et réessayer. Nous recommandons 10 tentatives avec un retard exponentiel commençant à 10 ms et allant jusqu'à 100 ms avec un temps total de 1 seconde pour toutes les tentatives.

[Une autre option consiste à appliquer la politique de nouvelles tentatives exponentielles d'Amazon Keyspaces lors de l'établissement de la connexion au pilote Java disponible sur Github.](#)

Vérifiez que vous avez établi des connexions à plusieurs nœuds lorsque vous utilisez la politique de nouvelle tentative par défaut. Vous pouvez le faire à l'aide de la requête suivante dans Amazon Keyspaces.

```
SELECT * FROM system.peers;
```

Si la réponse à cette requête est vide, cela indique que vous travaillez avec un seul nœud pour Amazon Keyspaces. Si vous utilisez la politique de nouvelle tentative par défaut, il n'y aura aucune nouvelle tentative car la nouvelle tentative par défaut a toujours lieu sur un autre nœud. Pour en

savoir plus sur l'établissement de connexions via des points de terminaison VPC, consultez. [the section called “Connexions aux points de terminaison VPC”](#)

Pour un didacticiel détaillé expliquant comment établir une connexion à Amazon Keyspaces à l'aide du pilote Cassandra Datastax 4.x, consultez. [the section called “Plugin d'authentification pour Java 4.x”](#)

Résolution des erreurs de connexion dans Amazon Keyspaces

Vous rencontrez des difficultés pour vous connecter ? Voici quelques problèmes courants et comment les résoudre.

Erreurs de connexion à un point de terminaison Amazon Keyspaces

Les connexions échouées et les erreurs de connexion peuvent entraîner des messages d'erreur différents. La section suivante couvre les scénarios les plus courants.

Rubriques

- [Je ne parviens pas à me connecter à Amazon Keyspaces avec cqlsh](#)
- [Je ne parviens pas à me connecter à Amazon Keyspaces à l'aide d'un pilote client Cassandra](#)

Je ne parviens pas à me connecter à Amazon Keyspaces avec cqlsh

Vous essayez de vous connecter à un point de terminaison Amazon Keyspaces à l'aide de cqlsh et la connexion échoue avec un. **Connection error**

Si vous essayez de vous connecter à une table Amazon Keyspaces et que cqlsh n'est pas correctement configuré, la connexion échoue. La section suivante fournit des exemples des problèmes de configuration les plus courants qui entraînent des erreurs de connexion lorsque vous essayez d'établir une connexion à l'aide de cqlsh.

Note

Si vous essayez de vous connecter à Amazon Keyspaces depuis un VPC, des autorisations supplémentaires sont requises. Pour configurer correctement une connexion à l'aide de points de terminaison VPC, suivez les étapes décrites dans le. [the section called “Connexion aux points de terminaison VPC”](#)

Vous essayez de vous connecter à Amazon Keyspaces à l'aide de `cqlsh`, mais une erreur de connexion s'affiche. **timed out**

Cela peut être le cas si vous n'avez pas fourni le port correct, ce qui entraîne l'erreur suivante.

```
# cqlsh cassandra.us-east-1.amazonaws.com 9140 -u "USERNAME" -p "PASSWORD" --ssl
Connection error: ('Unable to connect to any servers', {'3.234.248.199': error(None,
    "Tried connecting to [('3.234.248.199', 9140)]. Last error: timed out")})
```

Pour résoudre ce problème, vérifiez que vous utilisez le port 9142 pour la connexion.

Vous essayez de vous connecter à Amazon Keyspaces à l'aide de `cqlsh`, mais un message d'erreur s'affiche. **Name or service not known**

Cela peut être le cas si vous avez utilisé un point de terminaison mal orthographié ou qui n'existe pas. Dans l'exemple suivant, le nom du point de terminaison est mal orthographié.

```
# cqlsh cassandra.us-east-1.amazon.com 9142 -u "USERNAME" -p "PASSWORD" --ssl
Traceback (most recent call last):
  File "/usr/bin/cqlsh.py", line 2458, in <module>
    main(*read_options(sys.argv[1:], os.environ))
  File "/usr/bin/cqlsh.py", line 2436, in main
    encoding=options.encoding)
  File "/usr/bin/cqlsh.py", line 484, in __init__
    load_balancing_policy=WhiteListRoundRobinPolicy([self.hostname]),
  File "/usr/share/cassandra/lib/cassandra-driver-internal-only-3.11.0-bb96859b.zip/
cassandra-driver-3.11.0-bb96859b/cassandra/policies.py", line 417, in __init__
socket.gaierror: [Errno -2] Name or service not known
```

Pour résoudre ce problème lorsque vous utilisez des points de terminaison publics pour vous connecter, sélectionnez un point de [the section called “Points de terminaison de service”](#) terminaison disponible et vérifiez que le nom du point de terminaison ne contient aucune erreur. Si vous utilisez des points de terminaison VPC pour vous connecter, vérifiez que les informations du point de terminaison VPC sont correctes dans votre configuration `cqlsh`.

Vous essayez de vous connecter à Amazon Keyspaces à l'aide de `cqlsh`, mais un message d'erreur s'affiche. **OperationTimedOut**

Amazon Keyspaces exige que le protocole SSL soit activé pour les connexions afin de garantir une sécurité renforcée. Le paramètre SSL est peut-être absent si le message d'erreur suivant s'affiche.

```
# cqlsh cassandra.us-east-1.amazonaws.com -u "USERNAME" -p "PASSWORD"
```



```
Connection error: ('Unable to connect to any servers', {'3.234.248.192':
  OperationTimedOut('errors=Timed out creating connection (5 seconds),
  last_host=None',)})
#
```

Pour résoudre ce problème, ajoutez l'indicateur suivant à la commande de connexion cqlsh.

```
--ssl
```

Vous essayez de vous connecter à Amazon Keyspaces à l'aide de cqlsh et vous recevez un message d'erreur. **SSL transport factory requires a valid certfile to be specified**

Dans ce cas, le chemin d'accès au SSL/TLS certificat est manquant, ce qui entraîne l'erreur suivante.

```
# cat .cassandra/cqlshrc
[connection]
port = 9142
factory = cqlshlib.ssl.ssl_transport_factory
#

# cqlsh cassandra.us-east-1.amazonaws.com -u "USERNAME" -p "PASSWORD" --ssl
Validation is enabled; SSL transport factory requires a valid certfile to be specified.
Please provide path to the certfile in [ssl] section as 'certfile' option in /
root/.cassandra/cqlshrc (or use [certfiles] section) or set SSL_CERTFILE environment
variable.
#
```

Pour résoudre ce problème, ajoutez le chemin du fichier de certificat sur votre ordinateur. Pour de plus amples informations, veuillez consulter [the section called “Comment configurer manuellement les cqlsh connexions pour le protocole TLS”](#).

```
certfile = path_to_file/keyspaces-bundle.pem
```

Vous essayez de vous connecter à Amazon Keyspaces à l'aide de cqlsh, mais un message d'erreur s'affiche. **No such file or directory**

Cela peut être le cas si le chemin d'accès au fichier de certificat sur votre ordinateur est incorrect, ce qui entraîne l'erreur suivante.

```
# cat .cassandra/cqlshrc
[connection]
port = 9142
factory = cqlshlib.ssl.ssl_transport_factory

[ssl]
validate = true
certfile = /root/wrong_path/keysaces-bundle.pem
#

# cqlsh cassandra.us-east-1.amazonaws.com -u "USERNAME" -p "PASSWORD" --ssl
Connection error: ('Unable to connect to any servers', {'3.234.248.192': IOError(2, 'No
such file or directory')})
#
```

Pour résoudre ce problème, vérifiez que le chemin d'accès au fichier de certificat sur votre ordinateur est correct. Pour de plus amples informations, veuillez consulter [the section called “Comment configurer manuellement les cqlsh connexions pour le protocole TLS”](#).

Vous essayez de vous connecter à Amazon Keyspaces à l'aide de cqlsh, mais un message d'erreur s'affiche. **[X509] PEM lib**

Cela peut être le cas si le pem fichier de SSL/TLS certificat n'est pas valide, ce qui entraîne l'erreur suivante.

```
# cqlsh cassandra.us-east-1.amazonaws.com -u "USERNAME" -p "PASSWORD" --ssl
Connection error: ('Unable to connect to any servers', {'3.234.248.241':
error(185090057, u"Tried connecting to [('3.234.248.241', 9142)]. Last error: [X509]
PEM lib (_ssl.c:3063)"))
#
```

Pour résoudre ce problème, assurez-vous d'avoir téléchargé les certificats numériques requis. Pour de plus amples informations, veuillez consulter [the section called “Comment configurer manuellement les cqlsh connexions pour le protocole TLS”](#).

Vous essayez de vous connecter à Amazon Keyspaces à l'aide de cqlsh, mais vous recevez une erreur SSL. **unknown**

Cela peut être le cas si le pem fichier de SSL/TLS certificat est vide, ce qui entraîne l'erreur suivante.

```
# cqlsh cassandra.us-east-1.amazonaws.com -u "USERNAME" -p "PASSWORD" --ssl
Connection error: ('Unable to connect to any servers', {'3.234.248.220': error(0,
u"Tried connecting to [('3.234.248.220', 9142)]. Last error: unknown error
(_ssl.c:3063)"}))
#
```

Pour résoudre ce problème, assurez-vous d'avoir téléchargé les certificats numériques requis.

Vous pouvez le confirmer en suivant les étapes décrites dans la rubrique suivante [the section called “Comment configurer manuellement les cqlsh connexions pour le protocole TLS”](#).

Vous essayez de vous connecter à Amazon Keyspaces à l'aide de cqlsh, mais un message d'erreur s'affiche. **SSL: CERTIFICATE_VERIFY_FAILED**

Cela peut être le cas si le fichier de SSL/TLS certificat n'a pas pu être vérifié, ce qui entraîne l'erreur suivante.

```
Connection error: ('Unable to connect to any servers', {'3.234.248.223':
error(1, u"Tried connecting to [('3.234.248.223', 9142)]. Last error: [SSL:
CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:727)"}))
```

Pour résoudre ce problème, assurez-vous d'avoir téléchargé les certificats numériques requis.

Vous pouvez le confirmer en suivant les étapes décrites dans la rubrique suivante [the section called “Comment configurer manuellement les cqlsh connexions pour le protocole TLS”](#).

Vous essayez de vous connecter à Amazon Keyspaces à l'aide de cqlsh, mais un message d'erreur s'affiche. **Last error: timed out**

Cela peut être le cas si vous n'avez pas configuré de règle sortante pour Amazon Keyspaces dans votre groupe de sécurité Amazon EC2, ce qui entraîne l'erreur suivante.

```
# cqlsh cassandra.us-east-1.amazonaws.com 9142 -u "USERNAME" -p "PASSWORD" --ssl
Connection error: ('Unable to connect to any servers', {'3.234.248.206': error(None,
"Tried connecting to [('3.234.248.206', 9142)]. Last error: timed out"}))
#
```

Pour confirmer que ce problème est dû à la configuration de l'instance Amazon EC2 et non cqlsh, vous pouvez essayer de vous connecter à votre espace de touches en utilisant, par exemple AWS CLI, la commande suivante.

```
aws keyspace list-tables --keyspace-name 'my_keyspace'
```

Si cette commande expire également, l'instance Amazon EC2 n'est pas correctement configurée.

Pour vérifier que vous disposez des autorisations suffisantes pour accéder à Amazon Keyspaces, vous pouvez utiliser le AWS CloudShell pour vous connecter à `cqlsh`. Si ces connexions sont établies, vous devez configurer l'instance Amazon EC2.

Pour résoudre ce problème, vérifiez que votre instance Amazon EC2 dispose d'une règle sortante qui autorise le trafic vers Amazon Keyspaces. Si ce n'est pas le cas, vous devez créer un nouveau groupe de sécurité pour l'instance EC2 et ajouter une règle autorisant le trafic sortant vers les ressources Amazon Keyspaces. Pour mettre à jour la règle de trafic sortant afin d'autoriser le trafic vers Amazon Keyspaces, CQLSH/CASSANDRA choisissez dans le menu déroulant Type.

Après avoir créé le nouveau groupe de sécurité avec la règle de trafic sortant, vous devez l'ajouter à l'instance. Sélectionnez l'instance, puis sélectionnez Actions, Sécurité, puis Modifier les groupes de sécurité. Ajoutez le nouveau groupe de sécurité avec la règle sortante, mais assurez-vous que le groupe par défaut reste également disponible.

Pour plus d'informations sur la façon d'afficher et de modifier les règles de sortie EC2, consultez la section [Ajouter des règles à un groupe de sécurité dans le guide de l'utilisateur Amazon EC2](#).

Vous essayez de vous connecter à Amazon Keyspaces à l'aide de `cqlsh`, mais un message d'erreur s'affiche. **Unauthorized**

Cela peut être le cas si vous ne disposez pas des autorisations Amazon Keyspaces dans la politique utilisateur d'IAM, ce qui entraîne l'erreur suivante.

```
# cqlsh cassandra.us-east-1.amazonaws.com 9142 -u "testuser-at-12345678910" -p
"PASSWORD" --ssl
Connection error: ('Unable to connect to any servers', {'3.234.248.241':
AuthenticationFailed('Failed to authenticate to 3.234.248.241: Error from server:
code=2100 [Unauthorized] message="User arn:aws:iam::12345678910:user/testuser has no
permissions."',))
#
```

Pour résoudre ce problème, assurez-vous que l'utilisateur IAM `testuser-at-12345678910` est autorisé à accéder à Amazon Keyspaces. Pour des exemples de politiques IAM qui accordent l'accès à Amazon Keyspaces, consultez [the section called “Identity-based exemples de politiques”](#)

Pour obtenir des conseils de résolution des problèmes spécifiques à l'accès IAM, consultez [the section called “Résolution des problèmes”](#).

Vous essayez de vous connecter à Amazon Keyspaces à l'aide de `cqlsh`, mais un message d'erreur s'affiche. **Bad credentials**

Cela peut être le cas si le nom d'utilisateur ou le mot de passe est incorrect, ce qui entraîne l'erreur suivante.

```
# cqlsh cassandra.us-east-1.amazonaws.com 9142 -u "USERNAME" -p "PASSWORD" --ssl
Connection error: ('Unable to connect to any servers', {'3.234.248.248':
  AuthenticationFailed('Failed to authenticate to 3.234.248.248: Error from server:
    code=0100 [Bad credentials] message="Provided username USERNAME and/or password are
    incorrect"', )})
#
```

Pour résoudre ce problème, vérifiez que le code **USERNAME** et le mot de passe correspondent **PASSWORD** au nom d'utilisateur et au mot de passe que vous avez obtenus lorsque vous avez généré des informations d'identification [spécifiques au service](#).

Important

Si vous continuez à voir des erreurs lorsque vous essayez de vous connecter à `cqlsh`, réexécutez la commande avec l'option `--debugoption` et incluez le résultat détaillé lors de la prise de contact. Support

Je ne parviens pas à me connecter à Amazon Keyspaces à l'aide d'un pilote client Cassandra

Les sections suivantes présentent les erreurs les plus courantes lors de la connexion avec un pilote client Cassandra.

Vous essayez de vous connecter à une table Amazon Keyspaces à l'aide du pilote DataStax Java, mais un message d'**NodeUnavailableException** erreur s'affiche.

Si la connexion sur laquelle la demande est tentée est interrompue, cela entraîne l'erreur suivante.

```
[com.datastax.oss.driver.api.core.NodeUnavailableException: No connection
  was available to Node(endPoint=vpce-22ff22f2f22222fff-aa1bb234.cassandra.us-
  west-2.vpce.amazonaws.com/11.1.1111.222:9142, hostId=1a23456b-
  c77d-8888-9d99-146cb22d6ef6, hashCode=123ca4567)]
```

Pour résoudre ce problème, trouvez la valeur du rythme cardiaque et abaissez-la à 30 secondes si elle est supérieure.

```
advanced.heartbeat.interval = 30 seconds
```

Recherchez ensuite le délai d'expiration associé et assurez-vous que la valeur est définie sur au moins 5 secondes.

```
advanced.connection.init-query-timeout = 5 seconds
```

Vous essayez de vous connecter à une table Amazon Keyspaces à l'aide d'un pilote et du plugin SigV4, mais un message d'erreur s'affiche. **AttributeError**

Si les informations d'identification ne sont pas correctement configurées, l'erreur suivante se produit.

```
cassandra.cluster.NoHostAvailable: ('Unable to connect to any servers',  
{'44.234.22.154:9142': AttributeError("'NoneType' object has no attribute  
'access_key'")})
```

Pour résoudre ce problème, vérifiez que vous transmettez les informations d'identification associées à votre utilisateur ou rôle IAM lorsque vous utilisez le plug-in SigV4. Le plug-in SigV4 nécessite les informations d'identification suivantes.

- **AWS_ACCESS_KEY_ID**— Spécifie une clé d'AWS accès associée à un utilisateur ou à un rôle IAM.
- **AWS_SECRET_ACCESS_KEY**— Spécifie la clé secrète associée à la clé d'accès. Il s'agit du « mot de passe » de la clé d'accès.

Pour en savoir plus sur les clés d'accès et le plug-in SigV4, consultez [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#).

Vous essayez de vous connecter à une table Amazon Keyspaces à l'aide d'un pilote, mais un message d'erreur s'affiche. **PartialCredentialsError**

Si ce **AWS_SECRET_ACCESS_KEY** n'est pas le cas, cela peut entraîner l'erreur suivante.

```
cassandra.cluster.NoHostAvailable: ('Unable to connect to any servers',  
{'44.234.22.153:9142':  
PartialCredentialsError('Partial credentials found in config-file, missing:  
aws_secret_access_key')})
```

Pour résoudre ce problème, vérifiez que vous passez à la fois le `AWS_ACCESS_KEY_ID` et le `AWS_SECRET_ACCESS_KEY` lorsque vous utilisez le plug-in SigV4. Pour en savoir plus sur les clés d'accès et le plug-in SigV4, consultez [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#).

Vous essayez de vous connecter à une table Amazon Keyspaces à l'aide d'un pilote, mais un message d'**Invalid signature** erreur s'affiche.

Cela peut être le cas si l'un des composants requis pour la signature est incorrect ou n'est pas correctement défini pour la session.

- `AWS_ACCESS_KEY_ID`
- `AWS_SECRET_ACCESS_KEY`
- `AWS_DEFAULT_REGION`

L'erreur suivante est un exemple de clé d'accès non valide.

```
cassandra.cluster.NoHostAvailable: ('Unable to connect to any servers',
{'11.234.11.234:9142':
AuthenticationFailed('Failed to authenticate to 11.234.11.234:9142: Error from server:
code=0100
[Bad credentials] message="Authentication failure: Invalid signature"')})
```

Pour résoudre ce problème, vérifiez que les clés d'accès Région AWS ont été correctement configurées pour que le plugin SigV4 puisse accéder à Amazon Keyspaces. Pour en savoir plus sur les clés d'accès et le plug-in SigV4, consultez [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#).

La connexion de mon point de terminaison VPC ne fonctionne pas correctement

Vous essayez de vous connecter à Amazon Keyspaces via des points de terminaison VPC, mais vous recevez des erreurs de mappage de jetons ou vous rencontrez un faible débit.

Cela peut être le cas si la connexion du point de terminaison VPC n'est pas correctement configurée.

Pour résoudre ces problèmes, vérifiez les détails de configuration suivants. Pour suivre un didacticiel étape par étape afin d'apprendre à configurer une connexion via des points de terminaison VPC d'interface pour Amazon Keyspaces, consultez. [the section called “Connexion aux points de terminaison VPC”](#)

1. Vérifiez que l'entité IAM utilisée pour se connecter à Amazon Keyspaces read/write a accès à la table utilisateur et dispose d'un accès en lecture aux tables système, comme indiqué dans l'exemple suivant.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cassandra:Select",
        "cassandra:Modify"
      ],
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/mytable",
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
      ]
    }
  ]
}
```

2. Vérifiez que l'entité IAM utilisée pour se connecter à Amazon Keyspaces dispose des autorisations de lecture requises pour accéder aux informations du point de terminaison VPC sur votre instance Amazon EC2, comme indiqué dans l'exemple suivant.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListVPCEndpoints",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeNetworkInterfaces",
        "ec2:DescribeVpcEndpoints"
      ],
      "Resource": "*"
    }
  ]
}
```


Note

Les politiques gérées `AmazonKeyspacesFullAccess` incluent `AmazonKeyspacesReadOnlyAccess_v2` les autorisations requises pour permettre à Amazon Keyspaces d'accéder à l'instance Amazon EC2 afin de lire les informations sur les points de terminaison VPC d'interface disponibles.

Pour plus d'informations sur les points de terminaison VPC, voir [the section called “Utilisation des points de terminaison VPC d'interface pour Amazon Keyspaces”](#)

3. Vérifiez que la configuration SSL du pilote Java définit la validation du nom d'hôte sur `false`, comme indiqué dans cet exemple.

```
hostname-validation = false
```

Pour plus d'informations sur la configuration du pilote, consultez [the section called “Étape 2 : Configuration du pilote”](#).

4. Pour confirmer que le point de terminaison du VPC a été correctement configuré, vous pouvez exécuter l'instruction suivante depuis votre VPC.

Note

Vous ne pouvez pas utiliser votre environnement de développement local ou l'éditeur CQL d'Amazon Keyspaces pour confirmer cette configuration, car ils utilisent le point de terminaison public.

```
SELECT peer FROM system.peers;
```

La sortie doit ressembler à cet exemple et renvoyer entre 2 et 6 nœuds avec des adresses IPv4 privées lors de la connexion depuis un réseau IPv4, en fonction de la configuration et de la région de votre VPC. AWS

```
peer
-----
192.0.2.0.15
```

```
192.0.2.0.24  
192.0.2.0.13  
192.0.2.0.7  
192.0.2.0.8
```

(5 rows)

Je ne parviens pas à me connecter en utilisant **cassandra-stress**

Vous essayez de vous connecter à Amazon Keyspaces à l'aide de la **cassandra-stress** commande, mais un **SSL context** message d'erreur s'affiche.

Cela se produit si vous essayez de vous connecter à Amazon Keyspaces, mais que le TrustStore n'est pas correctement configuré. Amazon Keyspaces nécessite l'utilisation du protocole TLS (Transport Layer Security) pour sécuriser les connexions avec les clients.

Dans ce cas, le message d'erreur suivant s'affiche.

```
Error creating the initializing the SSL Context
```

Pour résoudre ce problème, suivez les instructions pour configurer un TrustStore comme indiqué dans cette rubrique [the section called “Avant de commencer”](#).

Une fois le TrustStore configuré, vous devriez pouvoir vous connecter à l'aide de la commande suivante.

```
./cassandra-stress user profile=./profile.yaml n=100 "ops(insert=1,select=1)"  
cl=LOCAL_QUORUM -node "cassandra.eu-north-1.amazonaws.com" -port native=9142  
-transport ssl-alg="PKIX" truststore="./cassandra_truststore.jks" truststore-  
password="trustStore_pw" -mode native cql3 user="user_name" password="password"
```

Je ne parviens pas à me connecter à l'aide des identités IAM

Vous essayez de vous connecter à une table Amazon Keyspaces en utilisant une identité IAM, mais vous recevez un message d'erreur. **Unauthorized**

Cela se produit si vous essayez de vous connecter à une table Amazon Keyspaces en utilisant une identité IAM (par exemple, un utilisateur IAM) sans mettre en œuvre la politique et sans donner au préalable à l'utilisateur les autorisations requises.

Dans ce cas, le message d'erreur suivant s'affiche.

```
Connection error: ('Unable to connect to any servers', {'3.234.248.202':
AuthenticationFailed('Failed to authenticate to 3.234.248.202:
Error from server: code=2100 [Unauthorized] message="User
arn:aws:iam::1234567890123:user/testuser has no permissions."' ,)})
```

Pour résoudre ce problème, vérifiez les autorisations de l'utilisateur IAM. Pour se connecter à un pilote standard, l'utilisateur doit au moins avoir SELECT accès aux tables du système, car la plupart des pilotes lisent le système keyspaces/tables lorsqu'ils établissent la connexion.

Par exemple, les politiques IAM qui accordent l'accès au système Amazon Keyspaces et aux tables utilisateur, voir. [the section called “Accès aux tables Amazon Keyspaces”](#)

Pour consulter la section de résolution des problèmes spécifique à IAM, consultez [the section called “Résolution des problèmes”](#).

J'essaie d'importer des données avec cqlsh et la connexion à ma table Amazon Keyspaces est perdue

Vous essayez de télécharger des données sur Amazon Keyspaces avec cqlsh, mais vous recevez des erreurs de connexion.

La connexion à Amazon Keyspaces échoue lorsque le client cqlsh a reçu trois erreurs consécutives de tout type de la part du serveur. Le client cqlsh échoue avec le message suivant.

```
Failed to import 1 rows: NoHostAvailable - , will retry later, attempt 3 of 100
```

Pour résoudre cette erreur, vous devez vous assurer que les données à importer correspondent au schéma de table dans Amazon Keyspaces. Vérifiez le fichier d'importation pour détecter les erreurs d'analyse. Vous pouvez essayer d'utiliser une seule ligne de données en utilisant une instruction INSERT pour isoler l'erreur.

Le client tente automatiquement de rétablir la connexion.

Résolution des erreurs de gestion des capacités dans Amazon Keyspaces

Vous rencontrez des problèmes avec la capacité sans serveur ? Voici quelques problèmes courants et comment les résoudre.

Erreurs de capacité sans serveur

Cette section explique comment reconnaître les erreurs liées à la gestion de la capacité sans serveur et comment les résoudre. Par exemple, vous pouvez observer des événements de capacité insuffisante lorsque votre application dépasse la capacité de débit allouée.

Apache Cassandra étant un logiciel basé sur des clusters conçu pour fonctionner sur un parc de nœuds, il ne contient aucun message d'exception lié aux fonctionnalités sans serveur telles que la capacité de débit. La plupart des pilotes ne comprennent que les codes d'erreur disponibles dans Apache Cassandra. Amazon Keyspaces utilise donc le même ensemble de codes d'erreur pour garantir la compatibilité.

Pour associer les erreurs de Cassandra aux événements de capacité sous-jacents, vous pouvez utiliser Amazon CloudWatch pour surveiller les indicateurs Amazon Keyspaces pertinents. Insufficient-capacity les événements qui entraînent des erreurs côté client peuvent être classés dans ces trois groupes en fonction de la ressource à l'origine de l'événement :

- **Tableau** — Si vous choisissez le mode Capacité allouée pour une table et que votre application dépasse le débit alloué, vous pouvez observer des erreurs de capacité insuffisante. Pour de plus amples informations, veuillez consulter [the section called “Configuration des modes read/write de capacité”](#).
- **Partition** — Vous pouvez rencontrer des événements de capacité insuffisante si le trafic sur une partition donnée dépasse 3 000 RCU ou 1 000 WCU. Nous recommandons de répartir le trafic de manière uniforme sur les partitions comme meilleure pratique. Pour de plus amples informations, veuillez consulter [the section called “Modélisation des données”](#).
- **Connexion** : le débit risque d'être insuffisant si vous dépassez le quota du nombre maximal d'opérations par seconde et par connexion. Pour augmenter le débit, vous pouvez augmenter le nombre de connexions par défaut lors de la configuration de la connexion avec le pilote.

Pour savoir comment configurer les connexions pour Amazon Keyspaces, consultez. [the section called “Comment configurer les connexions”](#) Pour plus d'informations sur l'optimisation des connexions via les points de terminaison VPC, consultez. [the section called “Connexions aux points de terminaison VPC”](#)

Pour déterminer quelle ressource est à l'origine de l'événement de capacité insuffisante qui renvoie l'erreur côté client, vous pouvez consulter le tableau de bord dans la console Amazon Keyspaces. Par défaut, la console fournit une vue agrégée des CloudWatch mesures relatives à la capacité et

au trafic les plus courantes dans la section Capacité et mesures associées de l'onglet Capacité du tableau.

Pour créer votre propre tableau de bord à l'aide d'Amazon CloudWatch, consultez les statistiques Amazon Keyspaces suivantes.

- **PerConnectionRequestRateExceeded**— Demandes adressées à Amazon Keyspaces qui dépassent le quota du taux de demandes par connexion. Chaque connexion client à Amazon Keyspaces peut prendre en charge jusqu'à 3 000 demandes CQL par seconde. Vous pouvez effectuer plus de 3 000 demandes par seconde en créant plusieurs connexions.
- **ReadThrottleEvents**— Demandes adressées à Amazon Keyspaces qui dépassent la capacité de lecture d'une table.
- **StoragePartitionThroughputCapacityExceeded**— Demandes adressées à une partition de stockage Amazon Keyspaces qui dépassent la capacité de débit de la partition. Les partitions de stockage Amazon Keyspaces peuvent supporter jusqu'à 1 000 unités WCU/WRU par seconde et 3 000 unités RCU/RRU par seconde. Pour atténuer ces exceptions, nous vous recommandons de revoir votre modèle de données afin de répartir le read/write trafic sur un plus grand nombre de partitions.
- **WriteThrottleEvents**— Demandes adressées à Amazon Keyspaces dont la capacité d'écriture dépasse la capacité d'écriture d'une table.

Pour en savoir plus CloudWatch, consultez [the section called “Surveillance avec CloudWatch”](#). Pour obtenir la liste de toutes les CloudWatch statistiques disponibles pour Amazon Keyspaces, consultez [the section called “Dimensions et métriques”](#)

Note

[Pour commencer à utiliser un tableau de bord personnalisé qui affiche toutes les statistiques fréquemment observées pour Amazon Keyspaces, vous pouvez utiliser un CloudWatch modèle prédéfini disponible GitHub dans le AWS référentiel d'exemples.](#)

Rubriques

- [Je reçois des erreurs de capacité NoHostAvailable insuffisante de la part de mon pilote client](#)
- [Je reçois des erreurs de délai d'écriture lors de l'importation des données](#)
- [Je ne vois pas la taille de stockage réelle d'un keyspace ou d'une table](#)

Je reçois des erreurs de capacité **NoHostAvailable** insuffisante de la part de mon pilote client

Vous voyez **Read_Timeout** des **Write_Timeout** exceptions pour une table.

Les tentatives répétées d'écriture ou de lecture dans une table Amazon Keyspaces dont la capacité est insuffisante peuvent entraîner des erreurs côté client spécifiques au pilote.

CloudWatch À utiliser pour surveiller vos indicateurs de débit provisionnés et réels, ainsi que les événements de capacité insuffisante pour le tableau. Par exemple, une demande de lecture dont la capacité de débit est insuffisante échoue avec une `Read_Timeout` exception et est publiée dans la `ReadThrottleEvents` métrique. Une demande d'écriture dont la capacité de débit est insuffisante échoue avec une `Write_Timeout` exception et est publiée dans la `WriteThrottleEvents` métrique. Pour plus d'informations sur ces métriques, consultez [the section called “Dimensions et métriques”](#).

Pour résoudre ces problèmes, envisagez l'une des options suivantes.

- Augmentez le débit provisionné pour la table, qui correspond à la capacité de débit maximale qu'une application peut consommer. Pour de plus amples informations, veuillez consulter [the section called “Unités de capacité en lecture et unités de capacité en écriture”](#).
- Laissez le service gérer la capacité de débit en votre nom grâce à la mise à l'échelle automatique. Pour de plus amples informations, veuillez consulter [the section called “Gérez la capacité de débit avec la mise à l'échelle automatique”](#).
- Choisissez le mode On-demandcapacité pour la table. Pour de plus amples informations, veuillez consulter [the section called “Configuration du mode capacité à la demande”](#).

Si vous devez augmenter le quota de capacité par défaut de votre compte, consultez [Quotas](#).

Des erreurs liées au dépassement de la capacité de partition s'affichent.

Lorsque le message d'erreur s'affiche, `StoragePartitionThroughputCapacityExceeded` la capacité de la partition est temporairement dépassée. Cela peut être géré automatiquement par la capacité adaptative ou la capacité à la demande. Nous vous recommandons de revoir votre modèle de données afin de répartir read/write le trafic sur un plus grand nombre de partitions afin d'atténuer ces erreurs. Les partitions de stockage Amazon Keyspaces peuvent supporter jusqu'à 1 000 unités WCU/WRU par seconde et 3 000 unités RCU/RRU par seconde. Pour en savoir plus sur la manière

d'améliorer votre modèle de données afin de répartir le read/write trafic sur un plus grand nombre de partitions, consultez [the section called “Modélisation des données”](#).

`Write_Timeout` les exceptions peuvent également être causées par un taux élevé d'opérations d'écriture simultanées qui incluent des données statiques et non statiques dans la même partition logique. Si le trafic est censé exécuter plusieurs opérations d'écriture simultanées incluant des données statiques et non statiques au sein de la même partition logique, nous vous recommandons d'écrire les données statiques et non statiques séparément. L'écriture séparée des données permet également d'optimiser les coûts de débit.

Vous constatez des erreurs liées au dépassement du taux de demandes de connexion.

Cela est `PerConnectionRequestRateExceeded` dû à l'une des causes suivantes.

- Il se peut que vous n'ayez pas assez de connexions configurées par session.
- Il se peut que vous obteniez moins de connexions que vos homologues disponibles, car les autorisations de point de terminaison VPC ne sont pas correctement configurées. Pour plus d'informations sur les politiques relatives aux points de terminaison VPC, consultez [the section called “Utilisation des points de terminaison VPC d'interface pour Amazon Keyspaces”](#)
- Si vous utilisez un pilote 4.x, vérifiez si la validation du nom d'hôte est activée. Le pilote active la vérification du nom d'hôte TLS par défaut. Cette configuration fait apparaître Amazon Keyspaces sous la forme d'un cluster à nœud unique pour le pilote. Nous vous recommandons de désactiver la vérification du nom d'hôte.

Nous vous recommandons de suivre les meilleures pratiques suivantes pour optimiser vos connexions et votre débit :

- Configurez le réglage du débit des requêtes CQL.

Amazon Keyspaces prend en charge jusqu'à 3 000 requêtes CQL par connexion TCP et par seconde, mais le nombre de connexions qu'un pilote peut établir est illimité.

La plupart des pilotes Cassandra open source établissent un pool de connexions avec Cassandra et équilibrent la charge des requêtes sur ce pool de connexions. Amazon Keyspaces expose 9 adresses IP homologues aux conducteurs. Le comportement par défaut de la plupart des pilotes consiste à établir une connexion unique à chaque adresse IP homologue. Par conséquent, le débit maximal de requêtes CQL d'un pilote utilisant les paramètres par défaut sera de 27 000 requêtes CQL par seconde.

Pour augmenter ce nombre, nous vous recommandons d'augmenter le nombre de connexions par adresse IP que votre pilote gère dans son pool de connexions. Par exemple, si vous définissez le nombre maximum de connexions par adresse IP sur 2, vous doublerez le débit maximal de votre pilote pour le porter à 54 000 requêtes CQL par seconde.

- Optimisez vos connexions à nœud unique.

Par défaut, la plupart des pilotes Cassandra open source établissent une ou plusieurs connexions à chaque adresse IP indiquée dans le `system.peers` tableau lors de l'établissement d'une session. Cependant, certaines configurations peuvent conduire un pilote à se connecter à une seule adresse IP Amazon Keyspaces. Cela peut se produire si le pilote tente de valider le nom d'hôte SSL des nœuds homologues (par exemple, les pilotes DataStax Java) ou lorsqu'il se connecte via un point de terminaison VPC.

Pour bénéficier de la même disponibilité et des mêmes performances qu'un pilote connecté à plusieurs adresses IP, nous vous recommandons de procéder comme suit :

- Augmentez le nombre de connexions par IP à 9 ou plus en fonction du débit client souhaité.
- Créez une politique de nouvelles tentatives personnalisée qui garantit que les nouvelles tentatives sont exécutées sur le même nœud. Pour plus d'informations, veuillez consulter la rubrique

[the section called “Comment configurer les politiques de nouvelle tentative”](#).

- Si vous utilisez des points de terminaison VPC, accordez à l'entité IAM utilisée pour se connecter à Amazon Keyspaces l'autorisation d'interroger votre VPC pour obtenir les informations relatives au point de terminaison et à l'interface réseau. Cela améliore l'équilibrage de charge et augmente read/write le débit. Pour de plus amples informations, veuillez consulter [???](#).

Je reçois des erreurs de délai d'écriture lors de l'importation des données

Vous recevez une erreur de temporisation lorsque vous chargez des données à l'aide de la **cqlsh COPY** commande.

```
Failed to import 1 rows: WriteTimeout - Error from server: code=1100 [Coordinator node
timed out waiting for replica nodes' responses]
message="Operation timed out - received only 0 responses." info={'received_responses':
0, 'required_responses': 2, 'write_type': 'SIMPLE', 'consistency':
'LOCAL_QUORUM'}, will retry later, attempt 1 of 100
```


Amazon Keyspaces utilise les `WriteTimeout` exceptions `ReadTimeout` et pour indiquer lorsqu'une demande d'écriture échoue en raison d'une capacité de débit insuffisante. Pour aider à diagnostiquer les exceptions de capacité insuffisante, Amazon Keyspaces publie les statistiques suivantes sur Amazon. CloudWatch

- `WriteThrottleEvents`
- `ReadThrottledEvents`
- `StoragePartitionThroughputCapacityExceeded`

Pour résoudre les erreurs de capacité insuffisante lors du chargement de données, réduisez le taux d'écriture par travailleur ou le taux d'ingestion total, puis réessayez de télécharger les lignes. Pour de plus amples informations, veuillez consulter [the section called “Étape 4 : Configuration des paramètres `cqlsh COPY FROM`”](#). Pour une option de téléchargement de données plus robuste, pensez à utiliser `DSBulk`, qui est disponible dans le [GitHub référentiel](#). Pour obtenir des instructions détaillées, consultez [the section called “Chargement de données à l'aide de `DSBulk`”](#).

Je ne vois pas la taille de stockage réelle d'un keyspace ou d'une table

Vous ne pouvez pas voir la taille de stockage réelle du keyspace ou de la table.

Pour en savoir plus sur la taille de rangement de votre table, consultez [the section called “Évaluer les coûts au niveau de la table”](#). Vous pouvez également estimer la taille de stockage en commençant par calculer la taille des lignes dans un tableau. Des instructions détaillées pour le calcul de la taille des lignes sont disponibles sur [the section called “Estimer la taille des lignes”](#).

Résolution des erreurs linguistiques liées à la définition des données dans Amazon Keyspaces

Vous rencontrez des difficultés pour créer des ressources ? Voici quelques problèmes courants et comment les résoudre.

Erreurs de langage de définition des données

Amazon Keyspaces exécute des opérations de langage de définition de données (DDL) de manière asynchrone, par exemple en créant et en supprimant des espaces clés et des tables. Si une application essaie d'utiliser la ressource avant qu'elle ne soit prête, l'opération échoue.

Vous pouvez surveiller l'état de création de nouveaux espaces de touches et de nouvelles tables dans le Console de gestion AWS, ce qui indique lorsqu'un espace de touches ou une table est en attente ou actif. Vous pouvez également surveiller l'état de création d'un nouvel espace de touches ou d'une nouvelle table par programmation en interrogeant la table du schéma du système. Un espace de touches ou une table devient visible dans le schéma du système lorsqu'il est prêt à être utilisé.

Note

Pour optimiser la création d'espaces clés à l'aide de cet utilitaire CloudFormation, vous pouvez utiliser cet utilitaire pour convertir des scripts CQL en CloudFormation modèles. L'outil est disponible dans le [GitHub référentiel](#).

Rubriques

- [J'ai créé un nouvel espace de touches, mais je ne parviens pas à le voir ou à y accéder](#)
- [J'ai créé une nouvelle table, mais je ne parviens pas à la consulter ou à y accéder](#)
- [J'essaie de restaurer une table à l'aide de la restauration instantanée \(PITR\) d'Amazon Keyspaces, mais la restauration échoue](#)
- [J'essaie de l'utiliser INSERT/UPDATE pour modifier les paramètres personnalisés Time to Live \(TTL\), mais l'opération échoue](#)
- [J'essaie de télécharger des données dans ma table Amazon Keyspaces et je reçois un message d'erreur indiquant que le nombre de colonnes est dépassé](#)
- [J'essaie de supprimer des données de ma table Amazon Keyspaces et la suppression échoue pour la plage](#)

J'ai créé un nouvel espace de touches, mais je ne parviens pas à le voir ou à y accéder

Vous recevez des erreurs de la part de votre application qui tente d'accéder à un nouvel espace de touches.

Si vous essayez d'accéder à un espace de touches Amazon Keyspaces récemment créé mais toujours en cours de création asynchrone, vous recevrez un message d'erreur. L'erreur suivante en est un exemple.

```
InvalidRequest: Error from server: code=2200 [Invalid query] message="unconfigured
keyspace mykeyspace"
```

Le modèle de conception recommandé pour vérifier qu'un nouvel espace de touches est prêt à être utilisé consiste à interroger les tables de schéma du système Amazon Keyspaces (system_schema_mcs.*).

Pour de plus amples informations, veuillez consulter [the section called “Vérifier l'état de création de l'espace clé”](#).

J'ai créé une nouvelle table, mais je ne parviens pas à la consulter ou à y accéder

Vous recevez des erreurs de la part de votre application qui tente d'accéder à une nouvelle table.

Si vous essayez d'accéder à une table Amazon Keyspaces récemment créée mais toujours en cours de création asynchrone, vous recevrez un message d'erreur. Par exemple, une tentative d'interrogation d'une table qui n'est pas encore disponible échoue avec une unconfigured table erreur.

```
InvalidRequest: Error from server: code=2200 [Invalid query] message="unconfigured
table mykeyspace.mytable"
```

La tentative d'affichage de la table avec sync_table() échoue avec unKeyError.

```
KeyError: 'mytable'
```

Le modèle de conception recommandé pour vérifier qu'une nouvelle table est prête à être utilisée consiste à interroger les tables du schéma du système Amazon Keyspaces (system_schema_mcs.*).

Il s'agit de l'exemple de sortie d'une table en cours de création.

```
user-at-123@cqlsh:system_schema_mcs> select table_name,status from
system_schema_mcs.tables where keyspace_name='example_keyspace' and
table_name='example_table';
```

```
table_name | status
```

```
-----+-----
```

```
example_table | CREATING
```

```
(1 rows)
```

Il s'agit de l'exemple de sortie d'une table active.

```
user-at-123@cqlsh:system_schema_mcs> select table_name,status from
system_schema_mcs.tables where keyspace_name='example_keyspace' and
table_name='example_table';
```

```
table_name | status
```

```
-----+-----
```

```
example_table | ACTIVE
```

```
(1 rows)
```

Pour de plus amples informations, veuillez consulter [the section called “Vérifier l'état de création de la table”](#).

J'essaie de restaurer une table à l'aide de la restauration instantanée (PITR) d'Amazon Keyspaces, mais la restauration échoue

Si vous essayez de restaurer une table Amazon Keyspaces avec une restauration instantanée (PITR) et que vous constatez que le processus de restauration commence mais ne se termine pas correctement, vous n'avez peut-être pas configuré toutes les autorisations requises pour le processus de restauration pour cette table en particulier.

Outre les autorisations des utilisateurs, Amazon Keyspaces peut avoir besoin d'autorisations pour effectuer des actions au nom de votre principal pendant le processus de restauration. C'est le cas si la table est chiffrée à l'aide d'une clé gérée par le client ou si vous utilisez des politiques IAM qui limitent le trafic entrant.

Par exemple, si vous utilisez des clés de condition dans votre politique IAM pour restreindre le trafic source à des points de terminaison ou à des plages d'adresses IP spécifiques, l'opération de restauration échoue. Pour permettre à Amazon Keyspaces d'effectuer l'opération de restauration des tables pour le compte de votre principal, vous devez ajouter une clé de condition `aws:ViaAWSService` globale dans la politique IAM.

Pour plus d'informations sur les autorisations de restauration de tables, consultez [the section called “Configurer les autorisations IAM pour la restauration”](#).

J'essaie de l'utiliser INSERT/UPDATE pour modifier les paramètres personnalisés Time to Live (TTL), mais l'opération échoue

Si vous essayez d'insérer ou de mettre à jour une valeur TTL personnalisée, l'opération risque d'échouer avec l'erreur suivante.

```
TTL is not yet supported.
```

Pour spécifier des valeurs TTL personnalisées pour les lignes ou les colonnes à l'aide UPDATE des opérations INSERT OR, vous devez d'abord activer le TTL pour la table. Vous pouvez activer le TTL pour un tableau à l'aide de la propriété `t_t1` personnalisée.

Pour plus d'informations sur l'activation des paramètres TTL personnalisés pour les tables, consultez [the section called “Mettre à jour le TTL personnalisé du tableau”](#).

J'essaie de télécharger des données dans ma table Amazon Keyspaces et je reçois un message d'erreur indiquant que le nombre de colonnes est dépassé

Vous téléchargez des données et vous avez dépassé le nombre de colonnes pouvant être mises à jour simultanément.

Cette erreur se produit lorsque le schéma de votre table dépasse la taille maximale de 350 Ko. Pour de plus amples informations, veuillez consulter [Quotas](#).

J'essaie de supprimer des données de ma table Amazon Keyspaces et la suppression échoue pour la plage

Vous essayez de supprimer des données par clé de partition et vous recevez un message d'erreur de suppression de plage.

Cette erreur se produit lorsque vous essayez de supprimer plus de 1 000 lignes en une seule opération de suppression.

```
Range delete requests are limited by the amount of items that can be deleted in a single range.
```

Pour de plus amples informations, veuillez consulter [the section called “Supprimer une plage”](#).

Pour supprimer plus de 1 000 lignes au sein d'une même partition, considérez les options suivantes.

- Supprimer par partition — Si la majorité des partitions comportent moins de 1 000 lignes, vous pouvez essayer de supprimer les données par partition. Si les partitions contiennent plus de 1 000 lignes, essayez plutôt de les supprimer par la colonne de clustering.
- Supprimer par colonne de clustering : si votre modèle contient plusieurs colonnes de clustering, vous pouvez utiliser la hiérarchie des colonnes pour supprimer plusieurs lignes. Les colonnes de clustering sont une structure imbriquée et vous pouvez supprimer de nombreuses lignes en agissant sur la colonne de niveau supérieur.
- Supprimer par ligne individuelle — Vous pouvez parcourir les lignes par itération et supprimer chaque ligne à l'aide de sa clé primaire complète (colonnes de partition et colonnes de clustering).
- Il est recommandé de diviser vos lignes sur des partitions. Dans Amazon Keyspaces, nous vous recommandons de répartir votre débit sur des partitions de table. Cela répartit les données et l'accès de manière uniforme sur les ressources physiques, ce qui permet d'obtenir le meilleur débit. Pour de plus amples informations, veuillez consulter [the section called “Modélisation des données”](#).

Tenez également compte des recommandations suivantes lorsque vous planifiez des opérations de suppression pour des charges de travail importantes.

- Avec Amazon Keyspaces, les partitions peuvent contenir un nombre de lignes pratiquement illimité. Cela vous permet de redimensionner les partitions de manière « plus large » que le guide Cassandra traditionnel de 100 Mo. Il n'est pas rare que les séries chronologiques ou les registres augmentent de plus d'un gigaoctet de données au fil du temps.
- Avec Amazon Keyspaces, il n'y a aucune stratégie de compactage ni aucune pierre angulaire à prendre en compte lorsque vous devez effectuer des opérations de suppression pour des charges de travail importantes. Vous pouvez supprimer autant de données que vous le souhaitez sans affecter les performances de lecture.

Surveillance d'Amazon Keyspaces (pour Apache Cassandra)

La surveillance joue un rôle important dans le maintien de la fiabilité, de la disponibilité et des performances d'Amazon Keyspaces et de vos autres AWS solutions. AWS fournit les outils de surveillance suivants pour surveiller Amazon Keyspaces, signaler un problème et prendre des mesures automatiques le cas échéant :

- Amazon Keyspaces propose un tableau de bord préconfiguré Console de gestion AWS indiquant la latence et les erreurs agrégées dans toutes les tables du compte.
- Amazon CloudWatch surveille vos AWS ressources et les applications que vous utilisez AWS en temps réel. Vous pouvez collecter et suivre les métriques à l'aide de tableaux de bord personnalisés. Par exemple, vous pouvez créer une base de référence pour les performances normales d'Amazon Keyspaces dans votre environnement en mesurant les performances à différents moments et dans différentes conditions de charge. Lorsque vous surveillez Amazon Keyspaces, stockez les données de surveillance historiques afin de pouvoir les comparer aux données de performance actuelles, d'identifier les modèles de performances normaux et les anomalies de performance, et de concevoir des méthodes pour résoudre les problèmes. Pour établir une base de référence, vous devez au minimum surveiller les erreurs système. Pour plus d'informations, consultez le [guide de CloudWatch l'utilisateur Amazon](#).
- CloudWatch Les alarmes Amazon surveillent une seule métrique sur une période que vous spécifiez et exécutent une ou plusieurs actions en fonction de la valeur de la métrique par rapport à un seuil donné sur un certain nombre de périodes. Par exemple, si vous utilisez Amazon Keyspaces en mode provisionné avec mise à l'échelle automatique des applications, l'action est une notification envoyée par Amazon Simple Notification Service (Amazon SNS) pour évaluer une politique d'Application Auto Scaling.

CloudWatch les alarmes n'appellent pas d'actions simplement parce qu'elles sont dans un état particulier. L'état doit avoir changé et avoir été maintenu pendant un nombre de périodes spécifié. Pour de plus amples informations, veuillez consulter [Surveillance d'Amazon Keyspaces avec Amazon CloudWatch](#).

- Amazon CloudWatch Logs vous permet de surveiller, de stocker et d'accéder à vos fichiers journaux à partir de tables Amazon Keyspaces et d'autres sources. CloudTrail CloudWatch Les journaux peuvent surveiller les informations contenues dans les fichiers journaux et vous avertir lorsque certains seuils sont atteints. Vous pouvez également archiver vos données de journaux

dans une solution de stockage hautement durable. Pour plus d'informations, consultez le [guide de l'utilisateur d'Amazon CloudWatch Logs](#).

- AWS CloudTrail capture les appels d'API et les événements associés créés par votre Compte AWS ou au nom de celui-ci et livre les fichiers journaux dans un compartiment Amazon S3 que vous spécifiez. Vous pouvez identifier les utilisateurs et les comptes qui ont appelé AWS, l'adresse IP source à partir de laquelle les appels ont été émis, ainsi que le moment où les appels ont eu lieu. Pour de plus amples informations, veuillez consulter le [Guide de l'utilisateur AWS CloudTrail](#).

Amazon EventBridge est un service de bus d'événements sans serveur qui permet de connecter facilement vos applications à des données provenant de diverses sources. EventBridge fournit un flux de données en temps réel à partir de vos propres applications, applications Software-as-a-Service (SaaS) et AWS services et achemine ces données vers des cibles telles que Lambda. Cela vous permet de surveiller les événements qui se produisent dans les services et de créer des architectures basées sur les événements. Pour plus d'informations, consultez le [guide de EventBridge l'utilisateur Amazon](#).

Rubriques

- [Surveillance d'Amazon Keyspaces avec Amazon CloudWatch](#)
- [Journalisation des appels d'API Amazon Keyspaces avec AWS CloudTrail](#)

Surveillance d'Amazon Keyspaces avec Amazon CloudWatch

Vous pouvez surveiller Amazon Keyspaces à l'aide d'Amazon CloudWatch, qui collecte des données brutes et les transforme en indicateurs lisibles en temps quasi réel. Ces statistiques sont enregistrées pour une durée de 15 mois ; par conséquent, vous pouvez accéder aux informations historiques et acquérir un meilleur point de vue de la façon dont votre service ou application web s'exécute.

Vous pouvez également définir des alarmes qui surveillent certains seuils et envoient des notifications ou prennent des mesures lorsque ces seuils sont atteints. Pour plus d'informations, consultez le [guide de CloudWatch l'utilisateur Amazon](#).

Note

Pour démarrer rapidement avec un tableau de CloudWatch bord préconfiguré présentant les indicateurs courants pour Amazon Keyspaces, vous pouvez utiliser CloudFormation un

modèle disponible sur. <https://github.com/aws-samples/amazon-keyspaces-cloudwatch-cloudformation-templates>

Rubriques

- [Comment utiliser les statistiques d'Amazon Keyspaces ?](#)
- [Statistiques et dimensions d'Amazon Keyspaces](#)
- [Afficher les statistiques d'Amazon Keyspaces dans CloudWatch](#)
- [Création d' CloudWatch alarmes pour surveiller Amazon Keyspaces](#)

Comment utiliser les statistiques d'Amazon Keyspaces ?

Les statistiques rapportées par Amazon Keyspaces fournissent des informations que vous pouvez analyser de différentes manières. La liste suivante présente certaines utilisations courantes des métriques. Voici quelques suggestions pour vous aider à démarrer, qui ne forment pas une liste exhaustive. Pour plus d'informations sur les mesures et la rétention, consultez la section [Mesures](#).

Comment puis-je ?	Métriques pertinentes
Comment déterminer si des erreurs système se sont produites	Vous pouvez superviser <code>SystemErrors</code> pour déterminer si des demandes ont entraîné un code erreur du serveur. En règle générale, cette métrique doit être égale à zéro. Si tel n'est pas le cas, vous devez enquêter.
Comment puis-je comparer la capacité moyenne de lecture allouée à la capacité de lecture consommée ?	<p>Pour surveiller la capacité moyenne de lecture allouée et la capacité de lecture consommée</p> <ol style="list-style-type: none"> 1. Définissez la période <code>ConsumedReadCapacityUnits</code> et <code>ProvisionedReadCapacityUnits</code> l'intervalle que vous souhaitez surveiller. 2. Modifiez la statistique pour <code>ConsumedReadCapacityUnits</code> de <code>Average</code> en <code>Sum</code>. 3. Créez une nouvelle Expression Mathvide. 4. Dans la section Détails de la nouvelle expression mathématique, entrez l'identifiant de la métrique <code>ConsumedReadCapacityUnits</code> et divisez-le par la fonction

Comment puis-je ?	Métriques pertinentes
	<p>CloudWatch PERIOD de la métrique (metric_id/ (PERIOD (metric_id))).</p> <p>5. Désélectionnez ConsumedReadCapacityUnits .</p> <p>Vous pouvez maintenant comparer votre capacité de lecture consommée moyenne à votre capacité allouée. Pour plus d'informations sur les fonctions arithmétiques de base et sur la création d'une série chronologique, voir Utilisation des mathématiques métriques.</p>
<p>Comment puis-je comparer l'écriture allouée moyenne à la capacité d'écriture consommée ?</p>	<p>Pour surveiller la capacité moyenne d'écriture allouée et la capacité d'écriture consommée</p> <ol style="list-style-type: none"> 1. Définissez la période ConsumedWriteCapacityUnits et ProvisionedWriteCapacityUnits l'intervalle que vous souhaitez surveiller. 2. Modifiez la statistique pour ConsumedWriteCapacityUnits de Average en Sum. 3. Créez une nouvelle Expression Mathvide. 4. Dans la section Détails de la nouvelle expression mathématique, entrez l'identifiant de la métrique ConsumedWriteCapacityUnits et divisez-le par la fonction CloudWatch PERIOD de la métrique (metric_id/ (PERIOD (metric_id))). 5. Désélectionnez ConsumedWriteCapacityUnits . <p>Vous pouvez maintenant comparer votre capacité d'écriture consommée moyenne à votre capacité allouée. Pour plus d'informations sur les fonctions arithmétiques de base et sur la création d'une série chronologique, voir Utilisation des mathématiques métriques.</p>

Statistiques et dimensions d'Amazon Keyspaces

Lorsque vous interagissez avec Amazon Keyspaces, celui-ci envoie des statistiques et des dimensions à Amazon CloudWatch. Toutes les métriques sont agrégées et signalées chaque minute. Les sections suivantes présentent un aperçu de toutes les statistiques disponibles pour Amazon Keyspaces.

Rubriques

- [Métriques Amazon Keyspaces](#)
- [Dimensions des statistiques Amazon Keyspaces](#)
- [Mesures relatives à la capture des données de modification par Amazon Keyspaces \(CDC\)](#)
- [Dimensions des métriques de capture des données de modification \(CDC\) d'Amazon Keyspaces](#)

Métriques Amazon Keyspaces

Amazon CloudWatch agrège les statistiques Amazon Keyspaces à intervalles d'une minute.

Certaines statistiques, telles que `Average` ou `Sum`, s'appliquent à chaque métrique. Cependant, toutes ces valeurs sont disponibles via la console Amazon Keyspaces, ou en utilisant la CloudWatch console ou des AWS SDK pour toutes les métriques. AWS CLI Dans le tableau suivant, chaque métrique dispose d'une liste de statistiques valides applicables à cette métrique.

Métrique	Description
<code>AccountMaxTableLevelReads</code>	<p>Nombre maximal d'unités de capacité en lecture pouvant être utilisées par la table d'un compte. Pour les tables à la demande, cette limite définit le nombre maximal d'unités de demande de lecture qu'une table peut utiliser.</p> <p>Unités Count:</p> <p>Statistiques valides :</p> <ul style="list-style-type: none"> • <code>Maximum</code>— Le nombre maximum d'unités de capacité de lecture pouvant être utilisées par une table du compte.

Métrique	Description
AccountMaxTableLevelWrites	<p>Nombre maximal d'unités de capacité en écriture pouvant être utilisées par la table d'un compte. Pour les tables à la demande, cette limite définit le nombre maximal d'unités de demande d'écriture qu'une table peut utiliser.</p> <p>Unités Count:</p> <p>Statistiques valides :</p> <ul style="list-style-type: none"> • Maximum— Le nombre maximum d'unités de capacité d'écriture pouvant être utilisées par une table du compte.
AccountProvisionedReadCapacityUtilization	<p>Pourcentage d'unités de capacité de lecture approuvée qu'un compte utilise.</p> <p>Unités : Percent</p> <p>Statistiques valides :</p> <ul style="list-style-type: none"> • Maximum – Pourcentage maximum d'unités de capacité de lecture approuvée que le compte utilise. • Minimum – Pourcentage minimum d'unités de capacité de lecture approuvée que le compte utilise. • Average – Pourcentage moyen d'unités de capacité de lecture approuvée que le compte utilise. La métrique est publiée à intervalles de cinq minutes. Par conséquent, si vous ajustez rapidement les unités de capacité de lecture approuvée, il se peut que cette statistique ne reflète pas la moyenne réelle.

Métrique	Description
AccountProvisionedWriteCapacityUtilization	<p>Pourcentage d'unités de capacité d'écriture approvisionnée qu'un compte utilise.</p> <p>Unités : Percent</p> <p>Statistiques valides :</p> <ul style="list-style-type: none"> • Maximum – Pourcentage maximum d'unités de capacité d'écriture approvisionnée que le compte utilise. • Minimum – Pourcentage minimum d'unités de capacité d'écriture approvisionnée que le compte utilise. • Average – Pourcentage moyen d'unités de capacité d'écriture approvisionnée que le compte utilise. La métrique est publiée à intervalles de cinq minutes. Par conséquent, si vous ajustez rapidement les unités de capacité d'écriture approvisionnée, il se peut que cette statistique ne reflète pas la moyenne réelle.
BillableTableSizeInBytes	<p>Taille facturable du tableau en octets. Il s'agit de la somme de la taille codée de toutes les lignes de la table. Cette métrique vous permet de suivre les coûts de stockage de votre table au fil du temps.</p> <p>Unités : Bytes</p> <p>Dimensions : Keyspace, TableName</p> <p>Statistiques valides :</p> <ul style="list-style-type: none"> • Maximum— La taille de stockage maximale de la table. • Minimum— La taille de stockage minimale de la table. • Average— Taille de stockage moyenne de la table. Cette métrique est calculée sur des intervalles de 4 à 6 heures.

Métrique	Description
<code>ConditionalCheckFailedRequests</code>	<p>Nombre de demandes d'écriture de transactions légères ayant échoué. Les opérations INSERT, UPDATE et DELETE vous permettent de fournir une condition logique dont l'évaluation doit être true avant que l'opération puisse continuer. Si cette condition est fausse, elle <code>ConditionalCheckFailedRequests</code> est incrémentée d'une unité. Les contrôles de condition qui évaluent de manière erronée consomment des unités de capacité d'écriture en fonction de la taille de la ligne. Pour de plus amples informations, veuillez consulter the section called “Estimation de la consommation de capacité du LWT”.</p> <p>Unités : Count</p> <p>Dimensions : Keyspace, TableName</p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• Minimum• Maximum• Average• SampleCount• Sum

Métrique	Description
ConsumedReadCapacityUnits	<p>Nombre d'unités de capacité de lecture consommées au cours de la période spécifiée. Pour plus d'informations, consultez Read/Write la section Mode capacité.</p> <div><p> Note</p><p>Pour comprendre votre utilisation moyenne du débit par seconde, utilisez la statistique Sum pour calculer le débit consommé pour la période d'une minute. Puis divisez la somme par le nombre de secondes en une minute (60) pour calculer la ConsumedReadCapacityUnits moyenne par seconde (en sachant que cette moyenne ne mettra pas en évidence les grands pics d'activité de lecture). Pour plus d'informations sur la comparaison de la capacité de lecture moyenne consommée à la capacité de lecture allouée, voir the section called “Utilisation de métriques”</p></div>

Unités : Count

Dimensions : Keyspace, TableName

Statistiques valides :

- **Minimum**— Le nombre minimal d'unités de capacité de lecture consommées par chaque demande individuelle envoyée à la table.
- **Maximum**— Le nombre maximal d'unités de capacité de lecture consommées par une requête individuelle envoyée à la table.
- **Average** – Capacité de lecture moyenne par demande consommée.

Métrique	Description
	<div> Note La valeur <code>Average</code> est influencée par des périodes d'inactivité où la valeur de l'échantillon est nulle.</div> <ul style="list-style-type: none">• <code>Sum</code> – Nombre total d'unités de capacité de lecture consommées. Il s'agit de la statistique la plus utile pour la métrique <code>ConsumedReadCapacityUnits</code> .• <code>SampleCount</code> — Le nombre de demandes adressées à Amazon Keyspaces, même si aucune capacité de lecture n'a été consommée. <div> Note La valeur <code>SampleCount</code> est influencée par des périodes d'inactivité où la valeur de l'échantillon est nulle.</div>

Métrique	Description
ConsumedWriteCapacityUnits	<p>Nombre d'unités de capacité d'écriture consommées au cours de la période spécifiée. Vous pouvez récupérer la capacité d'écriture totale consommée pour une table. Pour plus d'informations, consultez Read/Write la section Mode capacité.</p> <div><p> Note</p><p>Pour comprendre votre utilisation moyenne du débit par seconde, utilisez la statistique Sum pour calculer le débit consommé pour la période d'une minute. Puis divisez la somme par le nombre de secondes en une minute (60) pour calculer la ConsumedWriteCapacityUnits moyenne par seconde (en sachant que cette moyenne ne mettra pas en évidence les grands pics d'activité d'écriture). Pour plus d'informations sur la comparaison de la capacité d'écriture moyenne consommée à la capacité d'écriture allouée, voir the section called “Utilisation de métriques”</p></div> <p>Unités : Count</p> <p>Dimensions : Keyspace, TableName</p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• Minimum— Le nombre minimal d'unités de capacité d'écriture consommées par chaque demande individuelle envoyée à la table.• Maximum— Le nombre maximal d'unités de capacité d'écriture consommées par une requête individuelle envoyée à la table.

Métrique	Description
	<ul style="list-style-type: none">Average – Capacité d'écriture moyenne par demande consommée. <div> Note La valeur Average est influencée par des périodes d'inactivité où la valeur de l'échantillon est nulle.</div> <ul style="list-style-type: none">Sum – Nombre total d'unités de capacité d'écriture consommées. Il s'agit de la statistique la plus utile pour la métrique ConsumedWriteCapacityUnits .SampleCount — Le nombre de demandes adressées à Amazon Keyspaces, même si aucune capacité d'écriture n'a été consommée. <div> Note La valeur SampleCount est influencée par des périodes d'inactivité où la valeur de l'échantillon est nulle.</div>

Métrique	Description
MaxProvisionedTableReadCapacityUtilization	<p>Pourcentage moyen d'unités de capacité en lecture allouées utilisées par la table en lecture allouée la plus élevée du compte.</p> <p>Unités Percent:</p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• Maximum – Pourcentage maximum d'unités de capacité de lecture approvisionnée qu'utilise la table de lecture la plus approvisionnée du compte.• Minimum – Pourcentage minimum d'unités de capacité de lecture approvisionnée qu'utilise la table de lecture la plus approvisionnée du compte.• Average – Pourcentage moyen d'unités de capacité de lecture approvisionnée qu'utilise la table de lecture la plus approvisionnée du compte. La métrique est publiée à intervalles de cinq minutes. Par conséquent, si vous ajustez rapidement les unités de capacité de lecture approvisionnée, il se peut que cette statistique ne reflète pas la moyenne réelle.

Métrique	Description
MaxProvisionedTableWriteCapacityUtilization	<p>Pourcentage de capacité en écriture allouée utilisée par la table en écriture allouée la plus élevée d'un compte.</p> <p>Unités Percent:</p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• Maximum— Le pourcentage maximal d'unités de capacité d'écriture provisionnées utilisées par la table d'écriture provisionnée la plus élevée du compte.• Minimum— Le pourcentage minimum d'unités de capacité d'écriture provisionnées utilisées par la table d'écriture provisionnée la plus élevée du compte.• Average— Le pourcentage moyen d'unités de capacité d'écriture provisionnées utilisées par la table d'écriture provisionnée la plus élevée du compte. La métrique est publiée à intervalles de cinq minutes. Par conséquent, si vous ajustez rapidement les unités de capacité d'écriture approvisionnée, il se peut que cette statistique ne reflète pas la moyenne réelle.

Métrique	Description
<code>PerConnectionRequestRateExceeded</code>	<p>Demandes adressées à Amazon Keyspaces qui dépassent le quota de demandes par connexion. Chaque connexion client à Amazon Keyspaces peut prendre en charge jusqu'à 3 000 demandes CQL par seconde. Les clients peuvent créer plusieurs connexions pour augmenter le débit.</p> <p>Lorsque vous utilisez la réplication multirégionale, chaque écriture répliquée contribue également à ce quota. Il est recommandé d'augmenter le nombre de connexions à vos tables pour éviter les <code>PerConnectionRequestRateExceeded</code> erreurs. Il n'y a aucune limite au nombre de connexions que vous pouvez avoir sur Amazon Keyspaces.</p> <p>Unités : Count</p> <p>Dimensions : Keyspace, TableName, Operation</p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• <code>SampleCount</code>• <code>Sum</code>

Métrique	Description
ProvisionedReadCapacityUnits	<p>Nombre d'unités de capacité de lecture allouées pour une table.</p> <p>La dimension <code>TableName</code> renvoie la valeur <code>ProvisionedReadCapacityUnits</code> de la table.</p> <p>Unités : Count</p> <p>Dimensions : Keyspace, TableName</p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• Minimum – Paramètre le plus bas pour la capacité de lecture approvisionnée. Si vous utilisez <code>ALTER TABLE</code> pour augmenter la capacité de lecture, cette métrique indique la valeur <code>ReadCapacityUnits</code> approvisionnée la plus faible pendant cette période.• Maximum – Paramètre le plus élevé pour la capacité de lecture approvisionnée. Si vous utilisez <code>ALTER TABLE</code> pour réduire la capacité de lecture, cette métrique indique la valeur <code>ReadCapacityUnits</code> approvisionnée la plus élevée pendant cette période.• Average – Capacité de lecture approvisionnée moyenne. La métrique <code>ProvisionedReadCapacityUnits</code> est publiée à intervalles de cinq minutes. Par conséquent, si vous ajustez rapidement les unités de capacité de lecture provisionnée, il se peut que cette statistique ne reflète pas la moyenne réelle.

Métrique	Description
ProvisionedWriteCapacityUnits	<p>Nombre d'unités de capacité d'écriture allouées pour une table.</p> <p>La dimension <code>TableName</code> renvoie la valeur <code>ProvisionedWriteCapacityUnits</code> de la table.</p> <p>Unités : Count</p> <p>Dimensions : Keyspace, TableName</p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• Minimum – Paramètre le plus bas pour la capacité d'écriture approvisionnée. Si vous utilisez <code>ALTER TABLE</code> pour augmenter la capacité d'écriture, cette métrique indique la valeur <code>WriteCapacityUnits</code> approvisionnée la plus faible pendant cette période.• Maximum – Paramètre le plus élevé pour la capacité d'écriture approvisionnée. Si vous utilisez <code>ALTER TABLE</code> pour réduire la capacité d'écriture, cette métrique indique la valeur <code>WriteCapacityUnits</code> approvisionnée la plus élevée pendant cette période.• Average – Capacité d'écriture approvisionnée moyenne. La métrique <code>ProvisionedWriteCapacityUnits</code> est publiée à intervalles de cinq minutes. Par conséquent, si vous ajustez rapidement les unités de capacité d'écriture provisionnée, il se peut que cette statistique ne reflète pas la moyenne réelle.

Métrique	Description
ReadThrottleEvents	<p>Demandes adressées à Amazon Keyspaces qui dépassent la capacité de lecture allouée pour une table, ou les quotas au niveau du compte, les quotas de demande par connexion ou les quotas au niveau de la partition.</p> <p>Unités : Count</p> <p>Dimensions : Keyspace, TableName, Operation</p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• SampleCount• Sum
ReplicationLatency	<p>Cette métrique s'applique uniquement aux espaces clés multirégionaux et mesure le temps nécessaire à la réplication updatesinserts, ou le temps nécessaire pour passer d'une table deletes de réplique à une autre table de réplication dans un espace de saisie multirégional.</p> <p>Unités : Millisecond</p> <p>Dimensions : TableName, ReceivingRegion</p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• Average• Maximum• Minimum

Métrique	Description
ReturnedItemCountBySelect	<p>Nombre de lignes renvoyées par des SELECT requêtes à plusieurs lignes pendant la période spécifiée. Multi-row SELECTles requêtes sont des requêtes qui ne contiennent pas la clé primaire complète, telles que les scans complets de tables et les requêtes par plage.</p> <p>Notez que le nombre d'éléments retournés n'est pas nécessairement le même que le nombre d'éléments ayant été évalués. Par exemple, supposons que vous ayez demandé une sélection <code>SELECT *</code> avec une clause <code>ALLOW FILTERING</code> sur une table qui comportait 100 lignes, mais que vous ayez spécifié une clause <code>WHERE</code> afin de réduire les résultats à 15 lignes. Dans ce cas, la réponse de SELECT contiendra un <code>ScanCount</code> de 100 et un <code>Count</code> de 15 lignes retournées.</p> <p>Unités : Count</p> <p>Dimensions : Keyspace, TableName, Operation</p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• Minimum• Maximum• Average• SampleCount• Sum

Métrique	Description
<code>StoragePartitionThroughputCapacityExceeded</code>	<p>Demandes adressées à une partition de stockage Amazon Keyspaces qui dépassent la capacité de débit de la partition. Les partitions de stockage Amazon Keyspaces peuvent supporter jusqu'à 1 000 unités WCU/WRU par seconde et 3 000 unités RCU/RRU par seconde. Nous vous recommandons de revoir votre modèle de données afin de répartir read/write le trafic sur un plus grand nombre de partitions afin d'atténuer ces exceptions.</p> <div> Note Les partitions logiques d'Amazon Keyspaces peuvent s'étendre sur plusieurs partitions de stockage et leur taille est pratiquement illimitée.</div> <p>Unités : Count</p> <p>Dimensions : Keyspace, TableName, Operation</p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• SampleCount• Sum
<code>SuccessfulRequestCount</code>	<p>Le nombre de demandes traitées avec succès au cours de la période spécifiée.</p> <p>Unités : Count</p> <p>Dimensions : Keyspace, TableName, Operation</p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• SampleCount

Métrique	Description
SuccessfulRequestLatency	<p>Les demandes envoyées à Amazon Keyspaces avec succès au cours de la période spécifiée. <code>SuccessfulRequestLatency</code> peut fournir deux types d'informations différents :</p> <ul style="list-style-type: none">• Temps écoulé pour les demandes réussies (<code>Minimum</code>, <code>Maximum</code>, <code>Sum</code> ou <code>Average</code>).• Nombre de requêtes réussies (<code>SampleCount</code>). <p><code>SuccessfulRequestLatency</code> reflète uniquement l'activité au sein d'Amazon Keyspaces et ne prend pas en compte la latence du réseau ou l'activité côté client.</p> <p>Unités : <code>Milliseconds</code></p> <p>Dimensions : <code>Keyspace</code>, <code>TableName</code>, <code>Operation</code></p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• <code>Minimum</code>• <code>Maximum</code>• <code>Average</code>• <code>SampleCount</code>

Métrique	Description
SystemErrors	<p>Les demandes adressées à Amazon Keyspaces qui génèrent un <code>ServerError</code> pendant la période spécifiée. Une erreur <code>ServerError</code> indique généralement une erreur de service interne.</p> <p>Unités : Count</p> <p>Dimensions : Keyspace, TableName, Operation</p> <p>Statistiques valides :</p> <ul style="list-style-type: none"> Sum SampleCount
SystemReconciliationDeletes	<p>Les unités consommées pour supprimer les données tombstone lorsque les horodatages côté client sont activés. Chacune <code>SystemReconciliationDelete</code> fournit une capacité suffisante pour supprimer ou mettre à jour jusqu'à 1 Ko de données par ligne. Par exemple, pour mettre à jour une ligne qui stocke 2,5 Ko de données et pour supprimer une ou plusieurs colonnes de la ligne en même temps, il faut 3 <code>SystemReconciliationDeletes</code>. Ou bien, pour supprimer une ligne entière contenant 3,5 Ko de données, 4 sont nécessaires <code>SystemReconciliationDeletes</code>.</p> <p>Unités : Count</p> <p>Dimensions : Keyspace, TableName</p> <p>Statistiques valides :</p> <ul style="list-style-type: none"> Sum— Le nombre total de produits <code>SystemReconciliationDeletes</code> consommés au cours d'une période donnée.

Métrique	Description
TTLDeletes	<p>Unités consommées pour supprimer ou mettre à jour des données d'affilée à l'aide de Time to Live (TTL). Chacune TTLDelete fournit une capacité suffisante pour supprimer ou mettre à jour jusqu'à 1 Ko de données par ligne. Par exemple, pour mettre à jour une ligne qui stocke 2,5 Ko de données et pour supprimer simultanément une ou plusieurs colonnes de la ligne, 3 suppressions TTL sont nécessaires. Ou bien, pour supprimer une ligne entière contenant 3,5 Ko de données, 4 suppressions TTL sont nécessaires.</p> <p>Unités : Count</p> <p>Dimensions : Keyspace, TableName</p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• Sum— Le nombre total de produits TTLDeletes consommés au cours d'une période donnée.

Métrique	Description
UserErrors	<p>Demandes adressées à Amazon Keyspaces qui génèrent une <code>InvalidRequest</code> erreur pendant la période spécifiée.</p> <p><code>InvalidRequest</code> les erreurs représentent généralement une erreur côté client, par exemple une combinaison de paramètres non valide, une tentative de mise à jour d'une table inexistante ou une signature de demande incorrecte.</p> <p><code>UserErrors</code> représentent l'ensemble des demandes non valides pour le courant Région AWS et le courant Compte AWS.</p> <p>Unités : Count</p> <p>Dimensions : <code>Keyspace</code>, <code>TableName</code>, <code>Operation</code></p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• Sum• SampleCount
WriteThrottleEvents	<p>Demandes adressées à Amazon Keyspaces qui dépassent la capacité d'écriture allouée pour une table, ou les quotas au niveau du compte, les quotas de demande par connexion ou les quotas au niveau de la partition.</p> <p>Unités : Count</p> <p>Dimensions : <code>Keyspace</code>, <code>TableName</code>, <code>Operation</code></p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• SampleCount• Sum

Dimensions des statistiques Amazon Keyspaces

Les statistiques d'Amazon Keyspaces sont qualifiées par les valeurs du compte, du nom de la table ou de l'opération. Vous pouvez utiliser la CloudWatch console pour récupérer les données Amazon Keyspaces selon l'une des dimensions indiquées dans le tableau suivant.

Dimension	Description
Keyspace	Cette dimension limite les données à un keyspace spécifique. Cette valeur peut être n'importe quel espace-clé de la région actuelle et de la région actuelle Compte AWS.
Operation	Cette dimension limite les données à l'une des opérations CQL d'Amazon Keyspaces, telle que INSERT les opérations. SELECT
TableName	Cette dimension limite les données à une table spécifique. Cette valeur peut être n'importe quel nom de table de la région actuelle et de la région actuelle Compte AWS. Si le nom de la table n'est pas unique dans le compte, vous devez également spécifier Keyspace.

Mesures relatives à la capture des données de modification par Amazon Keyspaces (CDC)

Les statistiques suivantes sont disponibles dans votre CloudWatch compte pour surveiller les appels d'API vers le point de terminaison de capture des données de modification (CDC) d'Amazon Keyspaces.

Métrique	Description
ReturnedRecordsCount	<p>Le nombre d'enregistrements de flux renvoyés par GetRecords les opérations au cours de la période spécifiée.</p> <p>Unités : Count</p> <p>Dimensions : Keyspace, TableName, Operation</p>

Métrique	Description
	<p>Statistiques valides :</p> <ul style="list-style-type: none">• <code>Minimum</code>• <code>Maximum</code>• <code>Average</code>• <code>SampleCount</code>• <code>Sum</code>
<code>SuccessfulRequestLatency</code>	<p>Les demandes envoyées à Amazon Keyspaces avec succès au cours de la période spécifiée. <code>SuccessfulRequestLatency</code> peut fournir deux types d'informations différents :</p> <ul style="list-style-type: none">• Temps écoulé pour les demandes réussies (<code>Minimum</code>, <code>Maximum</code>, <code>Sum</code> ou <code>Average</code>).• Nombre de requêtes réussies (<code>SampleCount</code>). <p><code>SuccessfulRequestLatency</code> reflète uniquement l'activité au sein des API CDC d'Amazon Keyspaces et ne prend pas en compte la latence du réseau ou l'activité côté client.</p> <p>Unités : <code>Milliseconds</code></p> <p>Dimensions : <code>Keyspace</code>, <code>TableName</code>, <code>Operation</code></p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• <code>Minimum</code>• <code>Maximum</code>• <code>Average</code>• <code>SampleCount</code>

Métrique	Description
SystemErrors	<p>Les demandes adressées au CDC d'Amazon Keyspaces qui génèrent un code de HTTP 500 statut pendant la période spécifiée. Un code d'HTTP 500 état indique généralement une erreur de service interne.</p> <p>Unités : Count</p> <p>Dimensions : Keyspace, TableName, Operation</p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• Sum• SampleCount
UserErrors	<p>Demandes adressées au CDC d'Amazon Keyspaces qui génèrent un code de HTTP 400 statut pendant la période spécifiée.</p> <p>UserErrors représente l'ensemble des HTTP 400 erreurs enregistrées par Amazon Keyspaces CDC dans les versions actuelles Région AWS et actuelles. Compte AWS</p> <p>Unités : Count</p> <p>Dimensions : Keyspace, TableName, Operation</p> <p>Statistiques valides :</p> <ul style="list-style-type: none">• Sum• SampleCount

Dimensions des métriques de capture des données de modification (CDC) d'Amazon Keyspaces

Les statistiques d'Amazon Keyspaces sont qualifiées par les valeurs du compte, du nom de la table ou de l'opération. Vous pouvez utiliser la CloudWatch console pour récupérer les données Amazon Keyspaces selon l'une des dimensions indiquées dans le tableau suivant.

Dimension	Description
<code>StreamLabel</code>	Cette dimension limite les données à un flux de capture de données de modification (CDC) spécifique lorsque <code>KeyspaceName</code> et <code>TableName</code> sont spécifiées.
<code>Keyspace</code>	Cette dimension limite les données à un keyspace spécifique. Cette valeur peut être n'importe quel espace-clé de la région actuelle et de la région actuelle Compte AWS.
<code>TableName</code>	Cette dimension limite les données à une table spécifique. Cette valeur peut être n'importe quel nom de table de la région actuelle et de la région actuelle Compte AWS. Si le nom de la table n'est pas unique dans le compte, vous devez également spécifier <code>Keyspace</code> .
<code>Operation</code>	Cette dimension limite les données à l'une des opérations d'API Amazon Keyspaces Streams suivantes : <code>GetRecords</code> , <code>GetStream</code> , <code>GetShardIterator</code> , et <code>ListStreams</code> .

Afficher les statistiques d'Amazon Keyspaces dans CloudWatch

Lorsque vous interagissez avec Amazon Keyspaces, celui-ci envoie les statistiques et dimensions suivantes à Amazon CloudWatch. Toutes les métriques sont agrégées et signalées chaque minute. Vous pouvez utiliser les procédures suivantes pour consulter les statistiques d'Amazon Keyspaces.

Pour afficher les métriques à l'aide de la CloudWatch console

Les métriques sont d'abord regroupées par espace de noms de service, puis par les différentes combinaisons de dimension au sein de chaque espace de noms.

1. Ouvrez la CloudWatch console à l'adresse <https://console.aws.amazon.com/cloudwatch/>.
2. Si nécessaire, changez la Région AWS. Dans la barre de navigation, choisissez la région dans laquelle se trouvent vos AWS ressources. Pour plus d'informations, consultez [Points de terminaison du service AWS](#).
3. Dans le panneau de navigation, sélectionnez Métriques.
4. Sous l'onglet Toutes les métriques, choisissez AWS/Cassandra..

Pour afficher les métriques à l'aide de la AWS CLI

- À partir d'une invite de commande, utilisez la commande suivante :

```
aws cloudwatch list-metrics --namespace "AWS/Cassandra"
```

Création d' CloudWatch alarmes pour surveiller Amazon Keyspaces

Vous pouvez créer une CloudWatch alarme Amazon pour Amazon Keyspaces qui envoie un message Amazon Simple Notification Service (Amazon SNS) lorsque l'alarme change d'état. Une alarme surveille une seule métrique pendant la période que vous spécifiez. Elle réalise une ou plusieurs actions en fonction de la valeur de la métrique par rapport à un seuil donné sur un certain nombre de périodes. L'action est une notification envoyée à une rubrique Amazon SNS ou à une politique Application Auto Scaling.

Lorsque vous utilisez Amazon Keyspaces en mode provisionné avec Application Auto Scaling, le service crée deux paires d' CloudWatch alarmes en votre nom. Chaque paire représente vos limites supérieure et inférieure pour les paramètres de débit alloué. Ces CloudWatch alarmes sont déclenchées lorsque l'utilisation réelle de la table s'écarte de votre utilisation cible pendant une période prolongée. Pour en savoir plus sur les CloudWatch alarmes créées par Application Auto Scaling, consultez [the section called “Comment fonctionne la mise à l'échelle automatique d'Amazon Keyspaces”](#).

Les alarmes déclenchent des actions uniquement pour les changements d'état prolongés. CloudWatch les alarmes n'appellent pas d'actions simplement parce qu'elles sont dans un état particulier. L'état doit avoir changé et avoir été maintenu pendant un nombre de périodes spécifié.

Pour plus d'informations sur la création d' CloudWatch alarmes, consultez la section [Utilisation des CloudWatch alarmes Amazon](#) dans le guide de CloudWatch l'utilisateur Amazon.

Journalisation des appels d'API Amazon Keyspaces avec AWS CloudTrail

Amazon Keyspaces est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions effectuées par un utilisateur, un rôle ou un AWS service dans Amazon Keyspaces. CloudTrail capture les appels d'API DDL (Data Definition Language) et les appels d'API DML (Data Manipulation Language) pour Amazon Keyspaces sous forme d'événements. Les appels capturés incluent les appels provenant de la console Amazon Keyspaces et les appels programmatiques vers les opérations de l'API Amazon Keyspaces.

Si vous créez un suivi, vous pouvez activer la diffusion continue des CloudTrail événements vers un bucket Amazon Simple Storage Service (Amazon S3), y compris les événements pour Amazon Keyspaces.

Si vous ne configurez pas de suivi, vous pouvez toujours consulter les derniers événements pris en charge sur la CloudTrail console dans Historique des événements. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite à Amazon Keyspaces, l'adresse IP à partir de laquelle la demande a été faite, l'auteur de la demande, la date à laquelle elle a été faite, ainsi que des informations supplémentaires.

Pour en savoir plus CloudTrail, consultez le [guide de AWS CloudTrail l'utilisateur](#).

Rubriques

- [Configuration des entrées du fichier journal Amazon Keyspaces dans CloudTrail](#)
- [Informations sur le langage de définition des données \(DDL\) Amazon Keyspaces dans CloudTrail](#)
- [Informations sur le langage de manipulation des données \(DML\) d'Amazon Keyspaces dans CloudTrail](#)
- [Comprendre les entrées des fichiers journaux Amazon Keyspaces](#)

Configuration des entrées du fichier journal Amazon Keyspaces dans CloudTrail

Chaque action d'API Amazon Keyspaces connectée CloudTrail inclut des paramètres de demande exprimés dans le langage de requête CQL. Pour de plus amples informations, veuillez consulter [Référence du langage CQL](#).

Vous pouvez afficher, rechercher et télécharger les événements récents dans votre Compte AWS. Pour plus d'informations, consultez la section [Affichage des événements avec l'historique des CloudTrail événements](#).

Pour un enregistrement continu des événements survenus dans votre environnement Compte AWS, y compris des événements pour Amazon Keyspaces, créez un historique. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un parcours dans la console, celui-ci s'applique à toutes les AWS régions. Le journal enregistre les événements de toutes les régions de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. En outre, vous pouvez configurer d'autres AWS services pour analyser plus en détail les données d'événements collectées dans les CloudTrail journaux et agir en conséquence.

Pour plus d'informations, consultez les rubriques suivantes dans le AWS CloudTrail Guide de l'utilisateur :

- [Présentation de la création d'un journal d'activité](#)
- [CloudTrail services et intégrations pris en charge](#)
- [Configuration des notifications Amazon SNS pour CloudTrail](#)
- [Réception de fichiers CloudTrail journaux provenant de plusieurs régions](#)
- [Réception de fichiers CloudTrail journaux provenant de plusieurs comptes](#)

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été faite avec les informations d'identification de l'utilisateur root ou Gestion des identités et des accès AWS (IAM).
- Si la demande a été effectuée avec les informations d'identification de sécurité temporaires d'un rôle ou d'un utilisateur fédéré.
- Si la demande a été faite par un autre AWS service.

Pour de plus amples informations, veuillez consulter l'[élément userIdentity CloudTrail](#) .

Informations sur le langage de définition des données (DDL) Amazon Keyspaces dans CloudTrail

CloudTrail est activé sur votre compte Compte AWS lorsque vous créez le compte. Lorsqu'une activité DDL se produit dans Amazon Keyspaces, cette activité est automatiquement enregistrée en CloudTrail tant qu'événement avec les AWS autres événements de service dans l'historique des événements. Le tableau suivant présente les instructions DDL enregistrées pour Amazon Keyspaces.

CloudTrail eventName	Instruction	Action CQL	AWS Action du SDK
CreateKeyspace	DDL	CREATE KEYSPACE	CreateKeyspace
AlterKeyspace	DDL	ALTER KEYSPACE	UpdateKeyspace
DropKeyspace	DDL	DROP KEYSPACE	DeleteKeyspace
CreateTable	DDL	CREATE TABLE	CreateTable
DropTable	DDL	DROP TABLE	DeleteTable
AlterTable	DDL	ALTER TABLE	UpdateTable , TagResource , UntagResource
CreateUdt	DDL	CREATE TYPE	CreateType
DropUdt	DDL	DROP TYPE	DeleteType
GetStream	DDL	aucune action CQL disponible	GetStream
ListStreams	DDL	aucune action CQL disponible	ListStreams

Informations sur le langage de manipulation des données (DML) d'Amazon Keyspaces dans CloudTrail

Pour activer la journalisation des instructions DML d'Amazon Keyspaces avec CloudTrail, vous devez d'abord activer la journalisation de l'activité de l'API du plan de données dans CloudTrail. Vous pouvez commencer à enregistrer les événements DML Amazon Keyspaces dans des pistes nouvelles ou existantes en choisissant de consigner l'activité pour le type d'événement de données (table Cassandra) à l'aide de la CloudTrail console, ou en définissant la `resources.type` valeur à l'aide de `AWS::Cassandra::Table` la AWS CLI ou des opérations d'API. CloudTrail Pour plus d'informations, veuillez consulter [Journalisation des événements de données](#).

Pour plus d'informations et un exemple montrant comment créer des alarmes pour des événements liés aux données, consultez le billet suivant sur le blog de AWS base de données [Using DML auditing for Amazon Keyspaces \(pour Apache Cassandra\)](#).

Le tableau suivant indique les événements de données enregistrés CloudTrail par Cassandra table.

CloudTrail eventName	Instruction	Action CQL	AWS Actions du SDK
Select	DML	SELECT	GetKeyspace , GetTable, GetType, ListKeyspaces , ListTables , ListTypes , ListTagsForResource
Insert	DML	INSERT	aucune action du AWS SDK disponible
Mettre à jour	DML	UPDATE	aucune action du AWS SDK disponible
Suppression	DML	DELETE	aucune action du AWS SDK disponible

CloudTrail eventName	Instruction	Action CQL	AWS Actions du SDK
GetShardIterator	DML	aucune action CQL disponible	GetShardIterator
GetRecords	DML	aucune action CQL disponible	GetRecords

Comprendre les entrées des fichiers journaux Amazon Keyspaces

CloudTrail les fichiers journaux contiennent une ou plusieurs entrées de journal. Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'action demandée, la date et l'heure de l'action, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics, ils n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre une entrée de CloudTrail journal qui illustre les DropTable actions CreateKeyspace DropKeyspaceCreateTable,, et :

```
{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE:alice",
        "arn": "arn:aws:sts::111122223333:assumed-role/users/alice",
        "accountId": "111122223333",
        "sessionContext": {
          "sessionIssuer": {
            "type": "Role",
            "principalId": "AKIAIOSFODNN7EXAMPLE",
            "arn": "arn:aws:iam::111122223333:role/Admin",
            "accountId": "111122223333",
            "userName": "Admin"
          },
          "webIdFederationData": {},
          "attributes": {
            "mfaAuthenticated": "false",
```



```

        "creationDate": "2020-01-15T18:47:56Z"
      }
    },
    "eventTime": "2020-01-15T18:53:04Z",
    "eventSource": "cassandra.amazonaws.com",
    "eventName": "CreateKeyspace",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "10.24.34.01",
    "userAgent": "Cassandra Client/ProtocolV4",
    "requestParameters": {
      "rawQuery": "\n\tCREATE KEYSPACE \"mykeyspace\"\n\tWITH\n\t\tREPLICATION =
{'class': 'SingleRegionStrategy'}\n\t\t",
      "keyspaceName": "mykeyspace"
    },
    "responseElements": null,
    "requestID": "bfa3e75d-bf4d-4fc0-be5e-89d15850eb41",
    "eventID": "d25beae8-f611-4229-877a-921557a07bb9",
    "readOnly": false,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::Cassandra::Keyspace",
        "ARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/"
      }
    ],
    "eventType": "AwsApiCall",
    "apiVersion": "3.4.4",
    "recipientAccountId": "111122223333",
    "managementEvent": true,
    "eventCategory": "Management",
    "tlsDetails": {
      "tlsVersion": "TLSv1.2",
      "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
      "clientProvidedHostHeader": "cassandra.us-east-1.amazonaws.com"
    }
  },
  {
    "eventVersion": "1.05",
    "userIdentity": {
      "type": "AssumedRole",
      "principalId": "AKIAIOSFODNN7EXAMPLE:alice",
      "arn": "arn:aws:sts::111122223333:assumed-role/users/alice",
      "accountId": "111122223333",
      "sessionContext": {

```

```

    "sessionIssuer": {
      "type": "Role",
      "principalId": "AKIAIOSFODNN7EXAMPLE",
      "arn": "arn:aws:iam::111122223333:role/Admin",
      "accountId": "111122223333",
      "userName": "Admin"
    },
    "webIdFederationData": {},
    "attributes": {
      "mfaAuthenticated": "false",
      "creationDate": "2020-01-15T18:47:56Z"
    }
  }
},
"eventTime": "2020-01-15T19:28:39Z",
"eventSource": "cassandra.amazonaws.com",
"eventName": "DropKeyspace",
"awsRegion": "us-east-1",
"sourceIPAddress": "10.24.34.01",
"userAgent": "Cassandra Client/ProtocolV4",
"requestParameters": {
  "rawQuery": "DROP KEYSPACE \"mykeyspace\"",
  "keyspaceName": "mykeyspace"
},
"responseElements": null,
"requestID": "66f3d86a-56ae-4c29-b46f-abcd489ed86b",
"eventID": "e5aebec-e1dd-41e3-a515-84fe6aaabd7b",
"readOnly": false,
"resources": [
  {
    "accountId": "111122223333",
    "type": "AWS::Cassandra::Keyspace",
    "ARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/"
  }
],
"eventType": "AwsApiCall",
"apiVersion": "3.4.4",
"recipientAccountId": "111122223333",
"managementEvent": true,
"eventCategory": "Management",
"tlsDetails": {
  "tlsVersion": "TLSv1.2",
  "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
  "clientProvidedHostHeader": "cassandra.us-east-1.amazonaws.com"
}

```

```

},
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAIOSFODNN7EXAMPLE:alice",
    "arn": "arn:aws:sts::111122223333:assumed-role/users/alice",
    "accountId": "111122223333",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/Admin",
        "accountId": "111122223333",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-01-15T18:47:56Z"
      }
    }
  },
  "eventTime": "2020-01-15T18:55:24Z",
  "eventSource": "cassandra.amazonaws.com",
  "eventName": "CreateTable",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "10.24.34.01",
  "userAgent": "Cassandra Client/ProtocolV4",
  "requestParameters": {
    "rawQuery": "\n\tCREATE TABLE \"mykeyspace\".\"mytable\"(\n\t\t\"ID\" int,\n\t\t\"username\" text,\n\t\t\"email\" text,\n\t\t\"post_type\" text,\n\t\tPRIMARY\n\tKEY(\"ID\", \"username\", \"email\"))",
    "keyspaceName": "mykeyspace",
    "tableName": "mytable"
  },
  "responseElements": null,
  "requestID": "5f845963-70ea-4988-8a7a-2e66d061aacb",
  "eventID": "fe0dbd2b-7b34-4675-a30c-740f9d8d73f9",
  "readOnly": false,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::Cassandra::Table",

```

```

    "ARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/
mytable"
  }
],
"eventType": "AwsApiCall",
"apiVersion": "3.4.4",
"recipientAccountId": "111122223333",
"managementEvent": true,
"eventCategory": "Management",
"tlsDetails": {
  "tlsVersion": "TLSv1.2",
  "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
  "clientProvidedHostHeader": "cassandra.us-east-1.amazonaws.com"
},
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AKIAIOSFODNN7EXAMPLE:alice",
    "arn": "arn:aws:sts::111122223333:assumed-role/users/alice",
    "accountId": "111122223333",
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/Admin",
        "accountId": "111122223333",
        "userName": "Admin"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2020-01-15T18:47:56Z"
      }
    }
  },
  "eventTime": "2020-01-15T19:27:59Z",
  "eventSource": "cassandra.amazonaws.com",
  "eventName": "DropTable",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "10.24.34.01",
  "userAgent": "Cassandra Client/ProtocolV4",
  "requestParameters": {
    "rawQuery": "DROP TABLE \"mykeyspace\".\"mytable\"",

```

```

    "keyspaceName": "mykeyspace",
    "tableName": "mytable"
  },
  "responseElements": null,
  "requestID": "025501b0-3582-437e-9d18-8939e9ef262f",
  "eventID": "1a5cbedc-4e38-4889-8475-3eab98de0ffd",
  "readOnly": false,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::Cassandra::Table",
      "ARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/
mytable"
    }
  ],
  "eventType": "AwsApiCall",
  "apiVersion": "3.4.4",
  "recipientAccountId": "111122223333",
  "managementEvent": true,
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.2",
    "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
    "clientProvidedHostHeader": "cassandra.us-east-1.amazonaws.com"
  }
}
]
}

```

Le fichier journal suivant montre un exemple d'SELECT instruction.

```

{
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/alice",
    "accountId": "111122223333",
    "userName": "alice"
  },
  "eventTime": "2023-11-17T10:38:04Z",
  "eventSource": "cassandra.amazonaws.com",
  "eventName": "Select",
  "awsRegion": "us-east-1",

```

```

"sourceIPAddress": "10.24.34.01",
"userAgent": "Cassandra Client/ProtocolV4",
"requestParameters": {
  "keyspaceName": "my_keyspace",
  "tableName": "my_table",
  "conditions": [
    "pk = **(Redacted)",
    "ck < 3**(Redacted)0",
    "region = 't**(Redacted)t'"
  ],
  "select": [
    "pk",
    "ck",
    "region"
  ],
  "allowFiltering": true
},
"responseElements": null,
"requestID": "6d83bbf0-a3d0-4d49-b1d9-e31779a28628",
"eventID": "e00552d3-34e9-4092-931a-912c4e08ba17",
"readOnly": true,
"resources": [
  {
    "accountId": "111122223333",
    "type": "AWS::Cassandra::Table",
    "ARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/
table/my_table"
  }
],
"eventType": "AwsApiCall",
"apiVersion": "3.4.4",
"managementEvent": false,
"recipientAccountId": "111122223333",
"eventCategory": "Data",
"tlsDetails": {
  "tlsVersion": "TLSv1.3",
  "cipherSuite": "TLS_AES_128_GCM_SHA256",
  "clientProvidedHostHeader": "cassandra.us-east-1.amazonaws.com"
}
}

```

Le fichier journal suivant montre un exemple d'INSERT instruction.

```

{
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/alice",
    "accountId": "111122223333",
    "userName": "alice"
  },
  "eventTime": "2023-12-01T22:11:43Z",
  "eventSource": "cassandra.amazonaws.com",
  "eventName": "Insert",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "10.24.34.01",
  "userAgent": "Cassandra Client/ProtocolV4",
  "requestParameters": {
    "keyspaceName": "my_keyspace",
    "tableName": "my_table",
    "primaryKeys": {
      "pk": "**(Redacted)",
      "ck": "1**(Redacted)8"
    },
    "columnNames": [
      "pk",
      "ck",
      "region"
    ],
    "updateParameters": {
      "TTL": "2**(Redacted)0"
    }
  },
  "responseElements": null,
  "requestID": "edf8af47-2f87-4432-864d-a960ac35e471",
  "eventID": "81b56a1c-9bdd-4c92-bb8e-92776b5a3bf1",
  "readOnly": false,
  "resources": [
    {
      "accountId": "111122223333",
      "type": "AWS::Cassandra::Table",
      "ARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/table/my_table"
    }
  ]
}

```

```

],
"eventType": "AwsApiCall",
"apiVersion": "3.4.4",
"managementEvent": false,
"recipientAccountId": "111122223333",
"eventCategory": "Data",
"tlsDetails": {
  "tlsVersion": "TLSv1.3",
  "cipherSuite": "TLS_AES_128_GCM_SHA256",
  "clientProvidedHostHeader": "cassandra.us-east-1.amazonaws.com"
}
}

```

Le fichier journal suivant montre un exemple d'UPDATE instruction.

```

{
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/alice",
    "accountId": "111122223333",
    "userName": "alice"
  },
  "eventTime": "2023-12-01T22:11:43Z",
  "eventSource": "cassandra.amazonaws.com",
  "eventName": "Update",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "10.24.34.01",
  "userAgent": "Cassandra Client/ProtocolV4",
  "requestParameters": {
    "keyspaceName": "my_keyspace",
    "tableName": "my_table",
    "primaryKeys": {
      "pk": "'t**(Redacted)t'",
      "ck": "'s**(Redacted)g'"
    },
    "assignmentColumnNames": [
      "nonkey"
    ],
    "conditions": [
      "nonkey < 1**(Redacted)7"
    ]
  }
}

```



```

    },
    "responseElements": null,
    "requestID": "edf8af47-2f87-4432-864d-a960ac35e471",
    "eventID": "81b56a1c-9bdd-4c92-bb8e-92776b5a3bf1",
    "readOnly": false,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::Cassandra::Table",
        "ARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/table/
my_table"
      }
    ],
    "eventType": "AwsApiCall",
    "apiVersion": "3.4.4",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data",
    "tlsDetails": {
      "tlsVersion": "TLSv1.3",
      "cipherSuite": "TLS_AES_128_GCM_SHA256",
      "clientProvidedHostHeader": "cassandra.us-east-1.amazonaws.com"
    }
  }
}

```

Le fichier journal suivant montre un exemple d'DELETE instruction.

```

{
  "eventVersion": "1.09",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::111122223333:user/alice",
    "accountId": "111122223333",
    "userName": "alice",
  },
  "eventTime": "2023-10-23T13:59:05Z",
  "eventSource": "cassandra.amazonaws.com",
  "eventName": "Delete",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "10.24.34.01",
  "userAgent": "Cassandra Client/ProtocolV4",
  "requestParameters": {

```

```

    "keyspaceName": "my_keyspace",
    "tableName": "my_table",
    "primaryKeys": {
        "pk": "**(Redacted)",
        "ck": "**(Redacted)"
    },
    "conditions": [],
    "deleteColumnNames": [
        "m",
        "s"
    ],
    "updateParameters": {}
},
"responseElements": null,
"requestID": "3d45e63b-c0c8-48e2-bc64-31afc5b4f49d",
"eventID": "499da055-c642-4762-8775-d91757f06512",
"readOnly": false,
"resources": [
    {
        "accountId": "111122223333",
        "type": "AWS::Cassandra::Table",
        "ARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/table/
my_table"
    }
],
"eventType": "AwsApiCall",
"apiVersion": "3.4.4",
"managementEvent": false,
"recipientAccountId": "111122223333",
"eventCategory": "Data",
"tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "cassandra.us-east-1.amazonaws.com"
}
}

```

La sécurité dans Amazon Keyspaces (pour Apache Cassandra)

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le [modèle de responsabilité partagée](#) décrit ceci comme la sécurité du cloud et la sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS Cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. L'efficacité de notre sécurité est régulièrement testée et vérifiée par des auditeurs tiers dans le cadre des [programmes de conformité AWS](#). Pour en savoir plus sur les programmes de conformité qui s'appliquent à Amazon Keyspaces, consultez la section [AWS Services concernés par programme de conformité](#).
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris la sensibilité de vos données, les exigences de votre organisation, et la législation et la réglementation applicables.

Cette documentation vous aidera à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation d'Amazon Keyspaces. Les rubriques suivantes expliquent comment configurer Amazon Keyspaces pour répondre à vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres AWS services qui peuvent vous aider à surveiller et à sécuriser vos ressources Amazon Keyspaces.

Rubriques

- [Protection des données dans Amazon Keyspaces](#)
- [Gestion des identités et des accès AWS pour Amazon Keyspaces](#)
- [Validation de conformité pour Amazon Keyspaces \(pour Apache Cassandra\)](#)
- [Résilience et reprise après sinistre dans Amazon Keyspaces](#)
- [Sécurité de l'infrastructure dans Amazon Keyspaces](#)
- [Analyse de configuration et de vulnérabilité pour Amazon Keyspaces](#)
- [Bonnes pratiques en matière de sécurité pour Amazon Keyspaces](#)

Protection des données dans Amazon Keyspaces

Le modèle de [responsabilité AWS partagée \(modèle de \)](#) s'applique à la protection des données dans Amazon Keyspaces (pour Apache Cassandra). Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez la [FAQ sur la confidentialité des données](#) et les . Pour plus d'informations sur la protection des données en Europe, consultez le [Centre du règlement général sur la protection des données \(RGPD\)](#).

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou Gestion des identités et des accès AWS (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- SSL/TLS À utiliser pour communiquer avec AWS les ressources. Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail. Pour plus d'informations sur l'utilisation des CloudTrail sentiers pour capturer AWS des activités, consultez la section [Utilisation des CloudTrail sentiers](#) dans le guide de AWS CloudTrail l'utilisateur.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-3 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS disponibles, consultez [Norme FIPS \(Federal Information Processing Standard\) 140-3](#).

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Nom. Cela inclut lorsque vous travaillez avec Amazon Keyspaces ou d'autres utilisateurs à Services AWS l'aide de la console, de l'API ou AWS des AWS CLI SDK. Toutes les données

que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Rubriques

- [Le chiffrement est au repos dans Amazon Keyspaces](#)
- [Chiffrement en transit dans Amazon Keyspaces](#)
- [Confidentialité du trafic interréseau dans Amazon Keyspaces](#)

Le chiffrement est au repos dans Amazon Keyspaces

[Le chiffrement au repos d'Amazon Keyspaces \(pour Apache Cassandra\) améliore la sécurité en chiffrant toutes vos données au repos à l'aide des clés de chiffrement stockées dans \(\).AWS Key Management ServiceAWS KMS](#) Cette fonctionnalité réduit la lourdeur opérationnelle et la complexité induites par la protection des données sensibles. Avec le chiffrement au repos, vous pouvez créer des applications sensibles à la sécurité qui répondent aux exigences réglementaires et de conformité strictes en matière de protection des données.

Le chiffrement au repos d'Amazon Keyspaces chiffre vos données à l'aide de la norme de chiffrement avancée 256 bits (). AES-256 Cela permet de sécuriser vos données contre tout accès non autorisé au stockage sous-jacent.

Amazon Keyspaces chiffre et déchiffre les données contenues dans les tables et les flux de manière transparente. Amazon Keyspaces utilise le chiffrement des enveloppes et une hiérarchie de clés pour protéger les clés de chiffrement des données. Il s'intègre au stockage et AWS KMS à la gestion de la clé de chiffrement racine. Pour plus d'informations sur la hiérarchie des clés de chiffrement, consultez [the section called “Comment ça marche”](#). Pour plus d'informations sur des AWS KMS concepts tels que le chiffrement des enveloppes, consultez [AWS KMS la section Concepts des services de gestion](#) dans le Guide du AWS Key Management Service développeur.

Lorsque vous créez une nouvelle table, vous pouvez choisir l'une des AWS KMS clés suivantes (clés KMS) :

- Clé détenue par AWS — Il s'agit du type de cryptage par défaut. La clé appartient à Amazon Keyspaces (sans frais supplémentaires).

- Clé gérée par le client — Cette clé est stockée dans votre compte et vous l'avez créée, détenue et gérée par vous. Vous avez le contrôle total de la clé gérée par le client (AWS KMS des frais s'appliquent).

Amazon Keyspaces chiffre automatiquement les flux de capture des données de modification (CDC) avec la même clé que la table sous-jacente. Pour plus d'informations sur le CDC, voir [the section called “Travailler avec les flux CDC”](#).

Vous pouvez basculer entre la clé gérée par le client Clé détenue par AWS et la clé gérée par le client à tout moment. Vous pouvez spécifier une clé gérée par le client lorsque vous créez une nouvelle table ou que vous modifiez la clé KMS d'une table existante à l'aide de la console ou par programmation à l'aide d'instructions CQL. Pour savoir comment procéder, consultez [Chiffrement au repos : comment utiliser les clés gérées par le client pour chiffrer des tables dans Amazon Keyspaces](#).

Le chiffrement au repos à l'aide de l'option par défaut de Clés détenues par AWS est proposé sans frais supplémentaires. Toutefois, des frais AWS KMS s'appliquent les clés gérées par le client. Pour plus d'informations sur la tarification, consultez [Tarification d'AWS KMS](#).

Le chiffrement au repos d'Amazon Keyspaces est disponible dans toutes les régions Régions AWS, y compris en AWS Chine (Pékin) et en AWS Chine (Ningxia). Pour de plus amples informations, veuillez consulter [Chiffrement au repos : comment fonctionne-t-il dans Amazon Keyspaces](#).

Rubriques

- [Chiffrement au repos : comment fonctionne-t-il dans Amazon Keyspaces](#)
- [Chiffrement au repos : comment utiliser les clés gérées par le client pour chiffrer des tables dans Amazon Keyspaces](#)

Chiffrement au repos : comment fonctionne-t-il dans Amazon Keyspaces

Le chiffrement au repos d'Amazon Keyspaces (pour Apache Cassandra) chiffre vos données à l'aide de la norme Advanced Encryption Standard (AES) 256 bits. Cela permet de sécuriser vos données contre tout accès non autorisé au stockage sous-jacent. Toutes les données clients figurant dans les tables Amazon Keyspaces sont cryptées au repos par défaut, et le chiffrement côté serveur est transparent, ce qui signifie qu'aucune modification des applications n'est requise.

Le chiffrement au repos s'intègre à AWS Key Management Service (AWS KMS) pour gérer la clé de chiffrement utilisée pour chiffrer vos tables. Lors de la création d'une nouvelle table ou de la mise à jour d'une table existante, vous pouvez choisir l'une des options AWS KMS clés suivantes :

- Clé détenue par AWS — Il s'agit du type de cryptage par défaut. La clé appartient à Amazon Keyspaces (sans frais supplémentaires).
- Clé gérée par le client : cette clé est stockée dans votre compte et est créée, détenue et gérée par vous. Vous avez le contrôle total de la clé gérée par le client (AWS KMS des frais s'appliquent).

AWS KMS clé (clé KMS)

Le chiffrement au repos protège toutes vos données Amazon Keyspaces à l'aide d'une AWS KMS clé. Par défaut, Amazon Keyspaces utilise une [Clé détenue par AWS](#) clé de chiffrement multi-tenant créée et gérée dans un compte de service Amazon Keyspaces.

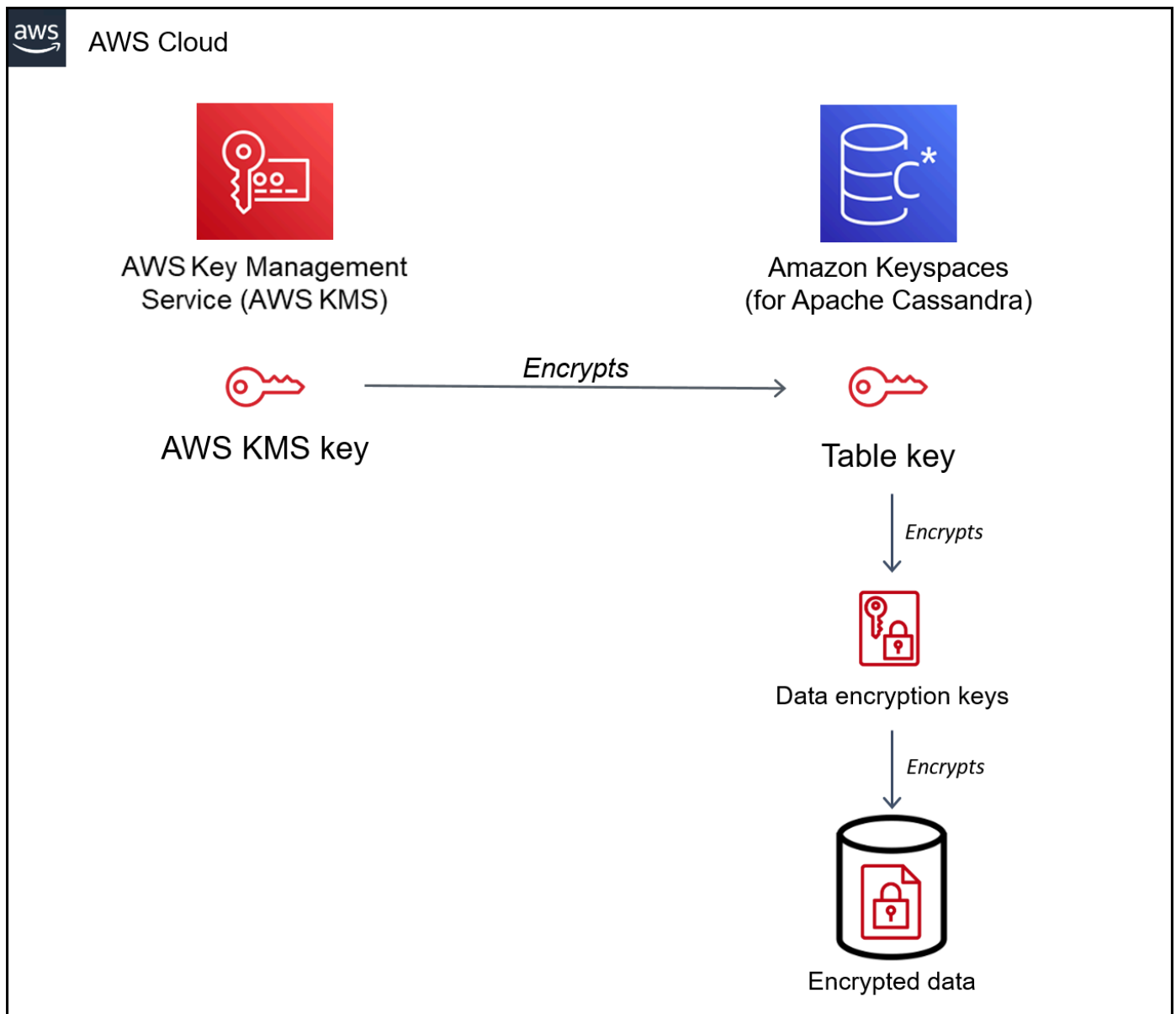
Cependant, vous pouvez crypter vos tables Amazon Keyspaces à l'aide d'une clé gérée par [le client](#) dans votre. Compte AWS Vous pouvez sélectionner une clé KMS différente pour chaque table d'un espace de touches. La clé KMS que vous sélectionnez pour une table est également utilisée pour chiffrer toutes ses métadonnées et ses sauvegardes restaurables.

Vous sélectionnez la clé KMS pour une table lorsque vous créez ou mettez à jour la table. Vous pouvez modifier la clé KMS d'une table à tout moment, soit dans la console Amazon Keyspaces, soit à l'aide de l'[instruction](#) ALTER TABLE. Le processus de changement de clé KMS est fluide et ne nécessite pas de temps d'arrêt ni de dégradation du service.

Hiérarchie des clés

Amazon Keyspaces utilise une hiérarchie de clés pour chiffrer les données. Dans cette hiérarchie de clés, la clé KMS est la clé racine. Il est utilisé pour chiffrer et déchiffrer la clé de chiffrement de la table Amazon Keyspaces. La clé de chiffrement de table est utilisée pour chiffrer les clés de chiffrement utilisées en interne par Amazon Keyspaces pour chiffrer et déchiffrer les données lors des opérations de lecture et d'écriture.

Grâce à la hiérarchie des clés de chiffrement, vous pouvez apporter des modifications à la clé KMS sans avoir à rechiffrer les données ni affecter les applications et les opérations de données en cours.



Clé de table

La clé de table Amazon Keyspaces est utilisée comme clé de chiffrement. Amazon Keyspaces utilise la clé de table pour protéger les clés de chiffrement des données internes utilisées pour chiffrer les données stockées dans les tables, les fichiers journaux et les sauvegardes restaurables. Amazon Keyspaces génère une clé de chiffrement de données unique pour chaque structure sous-jacente d'une table. Cependant, plusieurs lignes de table peuvent être protégées par la même clé de chiffrement des données.

Lorsque vous définissez pour la première fois la clé KMS sur une clé gérée par le client, une clé de données est AWS KMS générée. La clé AWS KMS de données fait référence à la clé de table dans Amazon Keyspaces.

Lorsque vous accédez à une table chiffrée, Amazon Keyspaces envoie une demande AWS KMS pour utiliser la clé KMS pour déchiffrer la clé de table. Ensuite, il utilise la clé de table en texte brut pour déchiffrer les clés de chiffrement des données Amazon Keyspaces, et il utilise les clés de cryptage des données en texte brut pour déchiffrer les données de la table.

Amazon Keyspaces utilise et stocke la clé de table et les clés de chiffrement des données en dehors de AWS KMS. Il protège toutes les clés avec les clés de chiffrement [Advanced Encryption Standard](#) (AES) et 256 bits. Ensuite, il stocke les clés cryptées avec les données cryptées afin qu'elles soient disponibles pour déchiffrer les données de la table à la demande.

Mise en cache des clés de table

Pour éviter de demander AWS KMS chaque opération Amazon Keyspaces, Amazon Keyspaces met en cache les clés de table en texte brut pour chaque connexion en mémoire. Si Amazon Keyspaces reçoit une demande concernant la clé de table mise en cache après cinq minutes d'inactivité, il envoie une nouvelle demande pour déchiffrer la clé AWS KMS de table. Cet appel capture toutes les modifications apportées aux politiques d'accès de la clé KMS dans AWS KMS ou Gestion des identités et des accès AWS (IAM) depuis la dernière demande de déchiffrement de la clé de table.

Chiffrement d'enveloppe

Si vous modifiez la clé gérée par le client pour votre table, Amazon Keyspaces génère une nouvelle clé de table. Il utilise ensuite la nouvelle clé de table pour rechiffrer les clés de chiffrement des données. Il utilise également la nouvelle clé de table pour chiffrer les clés de table précédentes qui sont utilisées pour protéger les sauvegardes restaurables. Ce processus s'appelle le chiffrement des enveloppes. Cela garantit que vous pouvez accéder à des sauvegardes restaurables même si vous alternez la clé gérée par le client. Pour plus d'informations sur le chiffrement des enveloppes, consultez la section [Envelope Encryption](#) dans le Guide du AWS Key Management Service développeur.

Rubriques

- [AWS clés possédées](#)
- [Clés gérées par le client](#)
- [Notes d'utilisation du chiffrement au repos](#)

AWS clés possédées

Clés détenues par AWS ne sont pas stockés dans votre Compte AWS. Elles font partie d'une collection de clés KMS que les AWS possède et les gère pour une utilisation multiple Comptes AWS. AWS services que vous pouvez Clés détenues par AWS utiliser pour protéger vos données.

Vous ne pouvez pas consulter, gérer Clés détenues par AWS, utiliser ou auditer leur utilisation. Cependant, vous n'avez pas besoin d'effectuer de travaux ni de modifier de programme pour protéger les clés qui chiffrent vos données.

Vous n'êtes pas facturé de frais mensuels ni de frais d'utilisation pour l'utilisation de Clés détenues par AWS, et ils ne sont pas pris en compte dans les AWS KMS quotas de votre compte.

Clés gérées par le client

Les clés gérées par le client sont des clés Compte AWS que vous créez, possédez et gérez. Vous avez le contrôle total de ces clés KMS.

Utilisez une clé gérée par le client pour obtenir les fonctions suivantes.

- Vous créez et gérez la clé gérée par le client, notamment en définissant et en tenant à jour les politiques [clés](#), les politiques <https://docs.aws.amazon.com/kms/latest/developerguide/iam-policies.html> IAM et les autorisations permettant [de](#) contrôler l'accès à la clé gérée par le client. Vous pouvez [activer et désactiver](#) la clé gérée par le client, activer et désactiver la rotation <https://docs.aws.amazon.com/kms/latest/developerguide/rotate-keys.html> automatique des clés et [planifier la suppression de la clé](#) gérée par le client lorsqu'elle n'est plus utilisée. Vous pouvez créer des balises et des alias pour les clés gérées par les clients que vous gérez.
- Vous pouvez utiliser une clé gérée par le client avec un [élément de clé importé](#) ou dans un [magasin de clés personnalisé](#) que vous possédez et gérez.
- Vous pouvez utiliser AWS CloudTrail Amazon CloudWatch Logs pour suivre les demandes qu'Amazon Keyspaces envoie en votre AWS KMS nom. Pour de plus amples informations, veuillez consulter [the section called “Étape 6 : Configuration de la surveillance avec AWS CloudTrail”](#).

Les clés gérées par le client [sont facturées](#) pour chaque appel d'API, et AWS KMS des quotas s'appliquent à ces clés KMS. Pour plus d'informations, consultez la section Quotas de [AWS KMS ressources ou de demandes](#).

Lorsque vous spécifiez une clé gérée par le client comme clé de chiffrement racine d'une table, les sauvegardes restaurables sont chiffrées avec la même clé de chiffrement que celle spécifiée pour

la table au moment de la création de la sauvegarde. En cas de rotation de la clé KMS de la table, la mise sous enveloppe des touches garantit que la dernière clé KMS a accès à toutes les sauvegardes restaurables.

Amazon Keyspaces doit avoir accès à votre clé gérée par le client pour vous permettre d'accéder aux données de votre table. Si l'état de la clé de chiffrement est désactivé ou si sa suppression est programmée, Amazon Keyspaces ne peut ni chiffrer ni déchiffrer les données. Par conséquent, vous ne pouvez pas effectuer d'opérations de lecture et d'écriture sur la table. Dès que le service détecte que votre clé de chiffrement n'est pas accessible, Amazon Keyspaces envoie une notification par e-mail pour vous avertir.

Vous devez rétablir l'accès à votre clé de chiffrement dans les sept jours, sinon Amazon Keyspaces supprimera automatiquement votre tableau. Par mesure de précaution, Amazon Keyspaces crée une sauvegarde restaurable des données de votre tableau avant de le supprimer. Amazon Keyspaces conserve la sauvegarde restaurable pendant 35 jours. Après 35 jours, vous ne pouvez plus restaurer les données de votre table. La sauvegarde restaurable ne vous est pas facturée, mais des frais de [restauration standard s'appliquent](#).

Vous pouvez utiliser cette sauvegarde restaurable pour restaurer vos données dans une nouvelle table. Pour lancer la restauration, la dernière clé gérée par le client utilisée pour la table doit être activée et Amazon Keyspaces doit y avoir accès.

Note

Lorsque vous créez une table chiffrée à l'aide d'une clé gérée par le client qui est inaccessible ou dont la suppression est programmée avant la fin du processus de création, une erreur se produit. L'opération de création de table échoue et vous recevez une notification par e-mail.

Notes d'utilisation du chiffrement au repos

Lorsque vous utilisez le chiffrement au repos dans Amazon Keyspaces, tenez compte des points suivants.

- Server-side le chiffrement au repos est activé sur toutes les tables Amazon Keyspaces et ne peut pas être désactivé. La table entière est cryptée au repos, vous ne pouvez pas sélectionner de colonnes ou de lignes spécifiques pour le cryptage.

- Par défaut, Amazon Keyspaces utilise une clé par défaut à service unique (Clé détenue par AWS) pour chiffrer toutes vos tables. Si cette clé n'existe pas, elle est créée pour vous. Les clés de service par défaut ne peuvent pas être désactivées.
- Le chiffrement au repos ne chiffre les données que lorsqu'elles sont statiques (au repos) sur un support de stockage persistant. Si la sécurité des données est un élément important pour les données en transit ou en cours d'utilisation, vous devez prendre des mesures supplémentaires :
 - Données en transit : toutes vos données stockées dans Amazon Keyspaces sont chiffrées en transit. Par défaut, les communications depuis et vers Amazon Keyspaces sont protégées à l'aide du chiffrement SSL (Secure Sockets Layer) /Transport Layer Security (TLS).
 - Données en cours d'utilisation : protégez vos données avant de les envoyer à Amazon Keyspaces en utilisant un cryptage côté client.
 - Clés gérées par le client : les données stockées dans vos tableaux sont toujours cryptées à l'aide de vos clés gérées par le client. Cependant, les opérations qui effectuent des mises à jour atomiques de plusieurs lignes chiffrent temporairement les données Clés détenues par AWS pendant le traitement. Cela inclut les opérations de suppression de pages et les opérations qui accèdent simultanément à des données statiques et non statiques.
- Une seule clé gérée par le client peut générer jusqu'à 50 000 [subventions](#). Chaque table Amazon Keyspaces associée à une clé gérée par le client consomme 2 subventions. Une subvention est libérée lorsque la table est supprimée. La deuxième autorisation est utilisée pour créer un instantané automatique de la table afin de prévenir toute perte de données au cas où Amazon Keyspaces perdrait l'accès à la clé gérée par le client par inadvertance. Cette subvention est accordée 42 jours après la suppression du tableau.

Chiffrement au repos : comment utiliser les clés gérées par le client pour chiffrer des tables dans Amazon Keyspaces

Vous pouvez utiliser la console ou les instructions CQL AWS KMS key pour spécifier les nouvelles tables et mettre à jour les clés de chiffrement des tables existantes dans Amazon Keyspaces. La rubrique suivante explique comment implémenter des clés gérées par le client pour les tables nouvelles et existantes.

Rubriques

- [Conditions préalables : créer une clé gérée par le client en utilisant AWS KMS et octroyer des autorisations à Amazon Keyspaces](#)
- [Étape 3 : Spécifier une clé gérée par le client pour une nouvelle table](#)

- [Étape 4 : Mettre à jour la clé de chiffrement d'une table existante](#)
- [Étape 5 : Utiliser le contexte de chiffrement Amazon Keyspaces dans les journaux](#)
- [Étape 6 : Configuration de la surveillance avec AWS CloudTrail](#)

Conditions préalables : créer une clé gérée par le client en utilisant AWS KMS et octroyer des autorisations à Amazon Keyspaces

Avant de protéger une table Amazon Keyspaces avec une [clé gérée par le client](#), vous devez d'abord créer la clé dans AWS Key Management Service (AWS KMS), puis autoriser Amazon Keyspaces à utiliser cette clé.

Étape 1 : Créez une clé gérée par le client à l'aide de AWS KMS

Pour créer une clé gérée par le client à utiliser pour protéger une table Amazon Keyspaces, vous pouvez suivre les étapes décrites dans [Création de clés KMS de chiffrement symétriques](#) à l'aide de la console ou de l'API. AWS

Étape 2 : Autoriser l'utilisation de votre clé gérée par le client

Avant de pouvoir choisir une [clé gérée par le client](#) pour protéger une table Amazon Keyspaces, les politiques relatives à cette clé gérée par le client doivent autoriser Amazon Keyspaces à l'utiliser en votre nom. Vous avez un contrôle total sur les politiques et les autorisations relatives à la clé gérée par le client. Vous pouvez fournir ces autorisations dans une [politique de clé](#), une [politique IAM](#) ou un [octroi](#).

Amazon Keyspaces n'a pas besoin d'autorisation supplémentaire pour utiliser la valeur par défaut [Clé détenue par AWS](#) afin de protéger les tables Amazon Keyspaces de votre compte. AWS

Les rubriques suivantes expliquent comment configurer les autorisations requises à l'aide de politiques et d'autorisations IAM qui autorisent les tables Amazon Keyspaces à utiliser une clé gérée par le client.

Rubriques

- [Politique clé concernant les clés gérées par le client](#)
- [Exemple de politique de clé](#)
- [Utiliser des subventions pour autoriser Amazon Keyspaces](#)

Politique clé concernant les clés gérées par le client

Lorsque vous sélectionnez une [clé gérée par le client](#) pour protéger une table Amazon Keyspaces, Amazon Keyspaces obtient l'autorisation d'utiliser la clé gérée par le client au nom du principal qui effectue la sélection. Ce principal, qu'il s'agisse d'un utilisateur ou d'un rôle, doit disposer des autorisations requises par Amazon Keyspaces sur la clé gérée par le client.

Amazon Keyspaces requiert au minimum les autorisations suivantes sur une clé gérée par le client :

- [kms:Encrypt](#)
- [kms:Decrypt](#)
- [kms : ReEncrypt](#) * (pour kms : ReEncryptFrom et kms :ReEncryptTo)
- kms : GenerateDataKey * (pour [kms : GenerateDataKey](#) et [kms : GenerateDataKeyWithoutPlaintext](#))
- [km : DescribeKey](#)
- [km : CreateGrant](#)

Exemple de politique de clé

Par exemple, l'exemple de stratégie de clé suivant fournit uniquement les autorisations requises. La politique a les effets suivants :

- Permet à Amazon Keyspaces d'utiliser la clé gérée par le client dans le cadre d'opérations cryptographiques et de créer des subventions, mais uniquement lorsqu'elle agit pour le compte des responsables du compte autorisés à utiliser Amazon Keyspaces. Si les principaux responsables spécifiés dans la déclaration de politique ne sont pas autorisés à utiliser Amazon Keyspaces, l'appel échoue, même s'il provient du service Amazon Keyspaces.
- La clé de ViaService condition [kms :](#) autorise les autorisations uniquement lorsque la demande provient d'Amazon Keyspaces au nom des principaux acteurs répertoriés dans la déclaration de politique. Ces principaux ne peuvent pas appeler ces opérations directement. Notez que la commande valeur kms:ViaService, cassandra.*.amazonaws.com, possède un astérisque (*) dans la position Région. Amazon Keyspaces a besoin d'une autorisation pour être indépendant de tout particulier. Région AWS
- Permet aux administrateurs de clés gérés par le client (utilisateurs qui peuvent assumer le db-team rôle) d'accéder en lecture seule à la clé gérée par le client et de révoquer les autorisations, y compris celles [dont Amazon Keyspaces a besoin](#) pour protéger la table.

- Permet à Amazon Keyspaces d'accéder en lecture seule à la clé gérée par le client. Dans ce cas, Amazon Keyspaces peut appeler ces opérations directement. Il n'est pas obligé d'agir pour le compte du principal d'un compte.

Avant d'utiliser un exemple de politique clé, remplacez les exemples de principes par des principes réels provenant de votre. Compte AWS

```
{
  "Id": "key-policy-cassandra",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow access through Amazon Keyspaces for all principals in the account
that are authorized to use Amazon Keyspaces",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::111122223333:user/db-lead"},
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey",
        "kms:CreateGrant"
      ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "kms:ViaService": "cassandra.*.amazonaws.com"
        }
      }
    },
    {
      "Sid": "Allow administrators to view the customer managed key and revoke
grants",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/db-team"
      },
      "Action": [
        "kms:Describe*",
        "kms:Get*",
        "kms:List*",
```

```
    "kms:RevokeGrant"  
  ],  
  "Resource": "*"   
}   
]   
}
```

Utiliser des subventions pour autoriser Amazon Keyspaces

Outre les politiques clés, Amazon Keyspaces utilise des autorisations pour définir des autorisations sur une clé gérée par le client. Pour visualiser les octrois sur une clé gérée par le client dans votre compte, utilisez l'opération [ListGrants](#). Amazon Keyspaces n'a pas besoin de subventions, ni d'autorisations supplémentaires, pour utiliser le [Clé détenue par AWS](#) afin de protéger votre table.

Amazon Keyspaces utilise les autorisations d'octroi lorsqu'il effectue des tâches de maintenance du système en arrière-plan et de protection continue des données. Il utilise également des octrois pour générer les clés de table.

Chaque octroi est spécifique à une table. Si le compte inclut plusieurs tables chiffrées sous la même clé gérée par le client, chaque type de table est attribué à chaque table. La subvention est limitée par le contexte de [chiffrement Amazon Keyspaces](#), qui inclut le nom de la table et Compte AWS l'ID. La subvention inclut l'autorisation de [retirer la subvention](#) si elle n'est plus nécessaire.

Pour créer les subventions, Amazon Keyspaces doit être autorisé à appeler `CreateGrant` au nom de l'utilisateur qui a créé la table cryptée.

La politique de clé peut également permettre au compte de [révoquer l'octroi](#) sur la clé gérée par le client. Toutefois, si vous révoquez l'autorisation sur une table cryptée active, Amazon Keyspaces ne sera pas en mesure de protéger et de gérer la table.

Étape 3 : Spécifier une clé gérée par le client pour une nouvelle table

Suivez ces étapes pour spécifier la clé gérée par le client sur une nouvelle table à l'aide de la console Amazon Keyspaces ou du CQL.

Création d'une table chiffrée à l'aide d'une clé gérée par le client (console)

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le panneau de navigation, choisissez Tables, puis Create table (Créer une table).

3. Sur la page Créer une table, dans la section Détails de la table, sélectionnez un espace de touche et attribuez un nom à la nouvelle table.
4. Dans la section Schéma, créez le schéma de votre table.
5. Dans la section Paramètres du tableau, choisissez Personnaliser les paramètres.
6. Passez aux paramètres de chiffrement.

Au cours de cette étape, vous devez sélectionner les paramètres de chiffrement de la table.

Dans la section Chiffrement au repos AWS KMS key, sous Choisir une clé KMS, choisissez l'option Choisir une autre clé KMS (avancée), puis dans le champ de recherche, choisissez AWS KMS key ou entrez un nom de ressource Amazon (ARN).

 Note

Si la clé que vous avez sélectionnée n'est pas accessible ou ne dispose pas des autorisations requises, consultez la section [Résolution des problèmes d'accès à la clé](#) dans le Guide du AWS Key Management Service développeur.

7. Choisissez Créer pour créer la table chiffrée.

Création d'une nouvelle table à l'aide d'une clé gérée par le client pour le chiffrement au repos (CQL)

Pour créer une nouvelle table qui utilise une clé gérée par le client pour le chiffrement au repos, vous pouvez utiliser l'`CREATE TABLE` instruction comme indiqué dans l'exemple suivant. Assurez-vous de remplacer l'ARN de la clé par un ARN pour une clé valide avec des autorisations accordées à Amazon Keyspaces.

```
CREATE TABLE my_keyspace.my_table(id bigint, name text, place text STATIC, PRIMARY
KEY(id, name)) WITH CUSTOM_PROPERTIES = {
    'encryption_specification':{
        'encryption_type': 'CUSTOMER_MANAGED_KMS_KEY',
        'kms_key_identifier': 'arn:aws:kms:eu-west-1:555555555555:key/11111111-1111-111-1111-111111111111'
    }
};
```

Si vous recevez un `Invalid Request Exception`, vous devez confirmer que la clé gérée par le client est valide et qu'Amazon Keyspaces dispose des autorisations requises. Pour vérifier que la clé

a été correctement configurée, consultez la section [Résolution des problèmes d'accès à la clé](#) dans le Guide du AWS Key Management Service développeur.

Étape 4 : Mettre à jour la clé de chiffrement d'une table existante

Vous pouvez également utiliser la console Amazon Keyspaces ou le CQL pour modifier les clés de chiffrement d'une table existante entre une clé KMS gérée par le client Clé détenue par AWS et une clé KMS gérée par le client à tout moment.

Mettre à jour une table existante avec la nouvelle clé gérée par le client (console)

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Dans le volet de navigation, choisissez Tables.
3. Choisissez le tableau que vous souhaitez mettre à jour, puis cliquez sur l'onglet Paramètres supplémentaires.
4. Dans la section Chiffrement au repos, choisissez Gérer le chiffrement pour modifier les paramètres de chiffrement de la table.

Sous Choisir une AWS KMS key, choisissez l'option Choisir une autre clé KMS (avancée), puis dans le champ de recherche, choisissez une AWS KMS key ou entrez un Amazon Resource Name (ARN).

Note

Si la clé que vous avez sélectionnée n'est pas valide, consultez la section [Résolution des problèmes d'accès aux clés](#) dans le Guide du AWS Key Management Service développeur.

Vous pouvez également choisir un Clé détenue par AWS pour une table chiffrée à l'aide d'une clé gérée par le client.

5. Choisissez Enregistrer les modifications pour enregistrer les modifications apportées au tableau.

Mettre à jour la clé de chiffrement utilisée pour une table existante

Pour modifier la clé de chiffrement d'une table existante, vous devez utiliser l'ALTER TABLE instruction pour spécifier une clé gérée par le client pour le chiffrement au repos. Assurez-

vous de remplacer l'ARN de la clé par un ARN pour une clé valide avec des autorisations accordées à Amazon Keyspaces.

```
ALTER TABLE my_keyspace.my_table WITH CUSTOM_PROPERTIES = {  
    'encryption_specification':{  
        'encryption_type': 'CUSTOMER_MANAGED_KMS_KEY',  
        'kms_key_identifier': 'arn:aws:kms:eu-west-1:555555555555:key/11111111-1111-111-1111-111111111111'  
    }  
};
```

Si vous recevez un `Invalid Request Exception`, vous devez confirmer que la clé gérée par le client est valide et qu'Amazon Keyspaces dispose des autorisations requises. Pour vérifier que la clé a été correctement configurée, consultez la section [Résolution des problèmes d'accès à la clé](#) dans le Guide du AWS Key Management Service développeur.

Pour rétablir l'option de chiffrement par défaut de la clé de chiffrement au repos avec Clés détenues par AWS, vous pouvez utiliser l'`ALTER TABLE` instruction comme indiqué dans l'exemple suivant.

```
ALTER TABLE my_keyspace.my_table WITH CUSTOM_PROPERTIES = {  
    'encryption_specification':{  
        'encryption_type' : 'AWS_OWNED_KMS_KEY'  
    }  
};
```

Étape 5 : Utiliser le contexte de chiffrement Amazon Keyspaces dans les journaux

Un [contexte de chiffrement](#) est un ensemble de paires clé-valeur qui contiennent des données non secrètes arbitraires. Lorsque vous incluez un contexte de chiffrement dans une demande de chiffrement de données, lie AWS KMS cryptographiquement le contexte de chiffrement aux données chiffrées. Pour déchiffrer les données, vous devez transmettre le même contexte de chiffrement.

Amazon Keyspaces utilise le même contexte de chiffrement dans toutes les opérations AWS KMS cryptographiques. Si vous utilisez une [clé gérée par le client](#) pour protéger votre table Amazon Keyspaces, vous pouvez utiliser le contexte de chiffrement pour identifier l'utilisation de la clé gérée par le client dans les dossiers d'audit et les journaux. Il apparaît également en texte clair dans les journaux, tels que dans les journaux pour [AWS CloudTrail](#) et [Amazon CloudWatch Logs](#).

Dans ses demandes adressées à AWS KMS, Amazon Keyspaces utilise un contexte de chiffrement comportant trois paires clé-valeur.

```
"encryptionContextSubset": {  
  "aws:cassandra:keyspaceName": "my_keyspace",  
  "aws:cassandra:tableName": "mytable"  
  "aws:cassandra:subscriberId": "111122223333"  
}
```

- **Keyspace** : la première paire clé-valeur identifie l'espace clé qui inclut la table cryptée par Amazon Keyspaces. La clé est `aws:cassandra:keyspaceName`. La valeur est le nom du keyspace.

```
"aws:cassandra:keyspaceName": "<keyspace-name>"
```

Par exemple :

```
"aws:cassandra:keyspaceName": "my_keyspace"
```

- **Table** — La deuxième paire clé-valeur identifie la table cryptée par Amazon Keyspaces. La clé est `aws:cassandra:tableName`. La valeur correspond au nom de la table.

```
"aws:cassandra:tableName": "<table-name>"
```

Par exemple :

```
"aws:cassandra:tableName": "my_table"
```

- **Compte** — La troisième paire clé-valeur identifie le. Compte AWS La clé est `aws:cassandra:subscriberId`. La valeur correspond à l'ID de compte.

```
"aws:cassandra:subscriberId": "<account-id>"
```

Par exemple :

```
"aws:cassandra:subscriberId": "111122223333"
```

Étape 6 : Configuration de la surveillance avec AWS CloudTrail

Si vous utilisez une [clé gérée par le client](#) pour protéger vos tables Amazon Keyspaces, vous pouvez utiliser AWS CloudTrail les journaux pour suivre les demandes qu'Amazon Keyspaces envoie en votre nom. AWS KMS

Les `CreateGrant` demandes `GenerateDataKeyDescribeKey`, `Decrypt`, et sont abordées dans cette section. En outre, Amazon Keyspaces utilise une [RetireGrant](#) opération pour supprimer une autorisation lorsque vous supprimez une table.

Note

Lorsque vous travaillez avec Amazon Keyspaces, certaines opérations peuvent générer des CloudTrail événements dont le `invokedBy` champ est égal à `dynamodb.amazonaws.com`. Cela est attendu et se produit car Amazon Keyspaces s'intègre à Amazon DynamoDB pour fournir son service.

GenerateDataKey

Amazon Keyspaces crée une clé de table unique pour chiffrer les données au repos. Il envoie une [GenerateDataKey](#) demande AWS KMS qui spécifie la clé KMS de la table.

L'événement qui enregistre l'opération `GenerateDataKey` est similaire à l'exemple d'événement suivant. L'utilisateur est le compte du service Amazon Keyspaces. Les paramètres incluent l'Amazon Resource Name (ARN) de la clé gérée par le client, un spécificateur de clé qui nécessite une clé de 256 bits et le [contexte de chiffrement](#) qui identifie l'espace clé, la table et le Compte AWS

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AWSService",
    "invokedBy": "AWS Internal"
  },
  "eventTime": "2021-04-16T04:56:05Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "GenerateDataKey",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "AWS Internal",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "keySpec": "AES_256",
    "encryptionContext": {
      "aws:cassandra:keyspaceName": "my_keyspace",
      "aws:cassandra:tableName": "my_table",
      "aws:cassandra:subscriberId": "123SAMPLE012"
    }
  }
}
```

```

    },
    "keyId": "arn:aws:kms:eu-west-1:555555555555:key/11111111-1111-111-1111-111111111111"
  },
  "responseElements": null,
  "requestID": "5e8e9cb5-9194-4334-aacc-9dd7d50fe246",
  "eventID": "49fccab9-2448-4b97-a89d-7d5c39318d6f",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123SAMPLE012",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:eu-west-1:555555555555:key/11111111-1111-111-1111-111111111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "123SAMPLE012",
  "sharedEventID": "84fbaaf0-9641-4e32-9147-57d2cb08792e"
}

```

DescribeKey

Amazon Keyspaces utilise une [DescribeKey](#) opération pour déterminer si la clé KMS que vous avez sélectionnée existe dans le compte et dans la région.

L'événement qui enregistre l'opération `DescribeKey` est similaire à l'exemple d'événement suivant. L'utilisateur est le compte du service Amazon Keyspaces. Les paramètres incluent l'ARN de la clé gérée par le client et un spécificateur de clé qui nécessite une clé de 256 bits.

```

{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAZ3FNIIVIZZ6H7CFQG",
    "arn": "arn:aws:iam::123SAMPLE012:user/admin",
    "accountId": "123SAMPLE012",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "admin",
    "sessionContext": {
      "sessionIssuer": {},

```

```

        "webIdFederationData": {},
        "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2021-04-16T04:55:42Z"
        }
    },
    "invokedBy": "AWS Internal"
},
"eventTime": "2021-04-16T04:55:58Z",
"eventSource": "kms.amazonaws.com",
"eventName": "DescribeKey",
"awsRegion": "us-east-1",
"sourceIPAddress": "AWS Internal",
"userAgent": "AWS Internal",
"requestParameters": {
    "keyId": "arn:aws:kms:eu-
west-1:555555555555:key/11111111-1111-111-1111-111111111111"
},
"responseElements": null,
"requestID": "c25a8105-050b-4f52-8358-6e872fb03a6c",
"eventID": "0d96420e-707e-41b9-9118-56585a669658",
"readOnly": true,
"resources": [
    {
        "accountId": "123SAMPLE012",
        "type": "AWS::KMS::Key",
        "ARN": "arn:aws:kms:eu-
west-1:555555555555:key/11111111-1111-111-1111-111111111111"
    }
],
"eventType": "AwsApiCall",
"managementEvent": true,
"eventCategory": "Management",
"recipientAccountId": "123SAMPLE012"
}

```

Decrypt

Lorsque vous accédez à une table Amazon Keyspaces, Amazon Keyspaces doit déchiffrer la clé de la table afin de pouvoir déchiffrer les clés situées en dessous dans la hiérarchie. Il déchiffre ensuite les données de la table. Pour déchiffrer la clé de table, Amazon Keyspaces envoie [une](#) demande AWS KMS de déchiffrement indiquant la clé KMS de la table.

L'événement qui enregistre l'opération Decrypt est similaire à l'exemple d'événement suivant. L'utilisateur principal est Compte AWS celui qui accède à la table. Les paramètres incluent la clé de table cryptée (sous forme de blob de texte chiffré) et le [contexte de chiffrement](#) qui identifie la table et le. Compte AWS AWS KMS déduit l'ID de la clé gérée par le client à partir du texte chiffré.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AWSService",
    "invokedBy": "AWS Internal"
  },
  "eventTime": "2021-04-16T05:29:44Z",
  "eventSource": "kms.amazonaws.com",
  "eventName": "Decrypt",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "AWS Internal",
  "userAgent": "AWS Internal",
  "requestParameters": {
    "encryptionContext": {
      "aws:cassandra:keyspaceName": "my_keyspace",
      "aws:cassandra:tableName": "my_table",
      "aws:cassandra:subscriberId": "123SAMPLE012"
    },
    "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
  },
  "responseElements": null,
  "requestID": "50e80373-83c9-4034-8226-5439e1c9b259",
  "eventID": "8db9788f-04a5-4ae2-90c9-15c79c411b6b",
  "readOnly": true,
  "resources": [
    {
      "accountId": "123SAMPLE012",
      "type": "AWS::KMS::Key",
      "ARN": "arn:aws:kms:eu-west-1:555555555555:key/11111111-1111-111-1111-111111111111"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "eventCategory": "Management",
  "recipientAccountId": "123SAMPLE012",
  "sharedEventID": "7ed99e2d-910a-4708-a4e3-0180d8dbb68e"
}
```


}

CreateGrant

Lorsque vous utilisez une [clé gérée par le client](#) pour protéger votre table Amazon Keyspaces, Amazon Keyspaces utilise des [subventions](#) pour permettre au service d'effectuer des tâches continues de protection, de maintenance et de durabilité des données. Ces subventions ne sont pas requises sur [Clés détenues par AWS](#).

Les subventions créées par Amazon Keyspaces sont spécifiques à une table. Le principal de la [CreateGrant](#) demande est l'utilisateur qui a créé la table.

L'événement qui enregistre l'opération CreateGrant est similaire à l'exemple d'événement suivant. Les paramètres incluent l'ARN de la clé gérée par le client pour la table, le principal du bénéficiaire et le principal sortant (le service Amazon Keyspaces), ainsi que les opérations couvertes par la subvention. Elle inclut également une contrainte qui exige que toutes les opérations de chiffrement utilisent le [contexte de chiffrement](#) spécifié.

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAZ3FNIIVIZZ6H7CFQG",
    "arn": "arn:aws:iam::arn:aws:kms:eu-west-1:555555555555:key/11111111-1111-111-1111-111111111111:user/admin",
    "accountId": "arn:aws:kms:eu-west-1:555555555555:key/11111111-1111-111-1111-111111111111",
    "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
    "userName": "admin",
    "sessionContext": {
      "sessionIssuer": {},
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2021-04-16T04:55:42Z"
      }
    }
  },
  "invokedBy": "AWS Internal"
},
"eventTime": "2021-04-16T05:11:10Z",
"eventSource": "kms.amazonaws.com",
"eventName": "CreateGrant",
```

```

"awsRegion": "us-east-1",
"sourceIPAddress": "AWS Internal",
"userAgent": "AWS Internal",
"requestParameters": {
  "keyId": "a7d328af-215e-4661-9a69-88c858909f20",
  "operations": [
    "DescribeKey",
    "GenerateDataKey",
    "Decrypt",
    "Encrypt",
    "ReEncryptFrom",
    "ReEncryptTo",
    "RetireGrant"
  ],
  "constraints": {
    "encryptionContextSubset": {
      "aws:cassandra:keyspaceName": "my_keyspace",
      "aws:cassandra:tableName": "my_table",
      "aws:cassandra:subscriberId": "123SAMPLE012"
    }
  },
  "retiringPrincipal": "cassandratest.us-east-1.amazonaws.com",
  "granteePrincipal": "cassandratest.us-east-1.amazonaws.com"
},
"responseElements": {
  "grantId":
"18e4235f1b07f289762a31a1886cb5efd225f069280d4f76cd83b9b9b5501013"
},
"requestID": "b379a767-1f9b-48c3-b731-fb23e865e7f7",
"eventID": "29ee1fd4-28f2-416f-a419-551910d20291",
"readOnly": false,
"resources": [
  {
    "accountId": "123SAMPLE012",
    "type": "AWS::KMS::Key",
    "ARN": "arn:aws:kms:eu-
west-1:555555555555:key/11111111-1111-111-1111-111111111111"
  }
],
"eventType": "AwsApiCall",
"managementEvent": true,
"eventCategory": "Management",
"recipientAccountId": "123SAMPLE012"

```

```
}
```

Chiffrement en transit dans Amazon Keyspaces

Amazon Keyspaces accepte uniquement les connexions sécurisées utilisant le protocole TLS (Transport Layer Security). Le chiffrement en transit fournit une couche supplémentaire de protection des données en chiffrant vos données lorsqu'elles sont acheminées vers et depuis Amazon Keyspaces. Les politiques organisationnelles, les réglementations sectorielles ou gouvernementales et les exigences de conformité nécessitent souvent l'utilisation du chiffrement en transit pour renforcer la sécurité des données de vos applications lorsqu'elles transmettent des données sur le réseau.

Pour savoir comment chiffrer les `cqlsh` connexions à Amazon Keyspaces à l'aide du protocole TLS, consultez [the section called “Comment configurer manuellement les `cqlsh` connexions pour le protocole TLS”](#) Pour savoir comment utiliser le chiffrement TLS avec les pilotes clients, consultez [the section called “Utilisation d'un pilote client Cassandra”](#).

Confidentialité du trafic interréseau dans Amazon Keyspaces

Cette rubrique décrit comment Amazon Keyspaces (pour Apache Cassandra) sécurise les connexions entre les applications sur site et Amazon Keyspaces et entre Amazon Keyspaces AWS et les autres ressources de ce dernier. Région AWS

Trafic entre les clients de service et sur site et les applications

Vous disposez de deux options de connectivité entre votre réseau privé et AWS :

- Une AWS Site-to-Site VPN connexion. Pour plus d'informations, consultez [Présentation de AWS Site-to-Site VPN](#) dans le Guide de l'utilisateur AWS Site-to-Site VPN .
- Une Direct Connect connexion. Pour plus d'informations, consultez [Présentation de Direct Connect](#) dans le Guide de l'utilisateur Direct Connect .

En tant que service géré, Amazon Keyspaces (pour Apache Cassandra) est protégé par la sécurité du réseau AWS mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section [Sécurité du AWS cloud](#). Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section [Protection de l'infrastructure](#) dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à Amazon Keyspaces via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Des suites de chiffrement dotées d'un secret de transmission parfait (PFS), telles que DHE (Ephemeral) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

Amazon Keyspaces prend en charge deux méthodes pour authentifier les demandes des clients. La première méthode utilise des informations d'identification spécifiques au service, qui sont des informations d'identification basées sur un mot de passe générées pour un utilisateur IAM spécifique. Vous pouvez créer et gérer le mot de passe à l'aide de la console IAM, de l' AWS CLI API ou de l' AWS API. Pour plus d'informations, consultez la section [Utilisation d'IAM avec Amazon Keyspaces](#).

La seconde méthode utilise un plugin d'authentification pour le pilote DataStax Java open source pour Cassandra. Ce plugin permet aux [utilisateurs, aux rôles et aux identités fédérées IAM](#) d'ajouter des informations d'authentification aux demandes d'API Amazon Keyspaces (pour Apache Cassandra) à l'aide du processus [AWS Signature Version 4 \(SigV4\)](#). Pour de plus amples informations, veuillez consulter [the section called “Création d'informations d'identification IAM pour l' AWS authentification”](#).

Trafic entre les AWS ressources d'une même région

Les points de terminaison VPC d'interface permettent une communication privée entre votre cloud privé virtuel (VPC) exécuté dans Amazon VPC et Amazon Keyspaces. Les points de terminaison VPC d'interface sont alimentés par AWS PrivateLink un AWS service qui permet une communication privée entre les VPC et les services. AWS PrivateLink permet cela en utilisant une interface réseau élastique avec des adresses IP privées dans votre VPC afin que le trafic réseau ne quitte pas le réseau Amazon. Les points de terminaison VPC d'interface ne nécessitent pas de passerelle Internet, de périphérie NAT, de connexion VPN ou de connexion. Direct Connect Pour plus d'informations, consultez [Amazon Virtual Private Cloud](#) and [Interface VPC endpoints](#) ().AWS PrivateLink Pour obtenir des exemples de politiques, consultez [the section called “Utilisation des points de terminaison VPC d'interface pour Amazon Keyspaces”](#).

Gestion des identités et des accès AWS pour Amazon Keyspaces

Gestion des identités et des accès AWS (IAM) est un outil Service AWS qui permet à un administrateur de contrôler en toute sécurité l'accès aux AWS ressources. Les administrateurs IAM

contrôlent qui peut être authentifié (connecté) et autorisé (autorisé) à utiliser les ressources Amazon Keyspaces. IAM est un Service AWS outil que vous pouvez utiliser sans frais supplémentaires.

Rubriques

- [Public ciblé](#)
- [Authentification par des identités](#)
- [Gestion de l'accès à l'aide de politiques](#)
- [Comment Amazon Keyspaces fonctionne avec IAM](#)
- [Exemples de politiques basées sur l'identité d'Amazon Keyspaces](#)
- [AWS politiques gérées pour Amazon Keyspaces](#)
- [Résolution des problèmes d'identité et d'accès à Amazon Keyspaces](#)
- [Utilisation de rôles liés à un service pour Amazon Keyspaces](#)

Public ciblé

La façon dont vous utilisez Gestion des identités et des accès AWS (IAM) varie en fonction de votre rôle :

- Utilisateur du service : demandez des autorisations à votre administrateur si vous ne pouvez pas accéder aux fonctionnalités (voir [Résolution des problèmes d'identité et d'accès à Amazon Keyspaces](#))
- Administrateur du service : déterminez l'accès des utilisateurs et soumettez les demandes d'autorisation (voir [Comment Amazon Keyspaces fonctionne avec IAM](#))
- Administrateur IAM : rédigez des politiques pour gérer l'accès (voir [Exemples de politiques basées sur l'identité d'Amazon Keyspaces](#))

Authentification par des identités

L'authentification est la façon dont vous vous connectez à AWS l'aide de vos informations d'identification. Vous devez être authentifié en tant qu'utilisateur IAM ou en assumant un rôle IAM. Utilisateur racine d'un compte AWS

Vous pouvez vous connecter en tant qu'identité fédérée à l'aide d'informations d'identification provenant d'une source d'identité telle que AWS IAM Identity Center (IAM Identity Center), d'une

authentification unique ou d'informations d'identification. Google/Facebook Pour plus d'informations sur la connexion, consultez [Connexion à votre Compte AWS](#) dans le Guide de l'utilisateur Connexion à AWS .

Pour l'accès par programmation, AWS fournit un SDK et une CLI pour signer les demandes de manière cryptographique. Pour plus d'informations, consultez [Signature AWS Version 4 pour les demandes d'API](#) dans le Guide de l'utilisateur IAM.

Compte AWS utilisateur root

Lorsque vous créez un Compte AWS, vous commencez par une seule identité de connexion appelée utilisateur Compte AWS root qui dispose d'un accès complet à toutes Services AWS les ressources. Il est vivement déconseillé d'utiliser l'utilisateur racine pour vos tâches quotidiennes. Pour les tâches qui requièrent des informations d'identification de l'utilisateur racine, consultez [Tâches qui requièrent les informations d'identification de l'utilisateur racine](#) dans le Guide de l'utilisateur IAM.

Utilisateurs et groupes IAM

Un [utilisateur IAM](#) est une identité qui dispose d'autorisations spécifiques pour une seule personne ou application. Nous vous recommandons d'utiliser ces informations d'identification temporaires au lieu des utilisateurs IAM avec des informations d'identification à long terme. Pour plus d'informations, voir [Exiger des utilisateurs humains qu'ils utilisent la fédération avec un fournisseur d'identité pour accéder à AWS l'aide d'informations d'identification temporaires](#) dans le guide de l'utilisateur IAM.

[Les groupes IAM](#) spécifient une collection d'utilisateurs IAM et permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Pour plus d'informations, consultez [Cas d'utilisation pour les utilisateurs IAM](#) dans le Guide de l'utilisateur IAM.

Rôles IAM

Un [rôle IAM](#) est une identité dotée d'autorisations spécifiques qui fournit des informations d'identification temporaires. Vous pouvez assumer un rôle en [passant d'un rôle utilisateur à un rôle IAM \(console\)](#) ou en appelant une opération AWS CLI ou AWS API. Pour plus d'informations, consultez [Méthodes pour endosser un rôle](#) dans le Guide de l'utilisateur IAM.

Les rôles IAM sont utiles pour l'accès des utilisateurs fédérés, les autorisations temporaires des utilisateurs IAM, les accès intercompte, les accès entre services et les applications exécutées sur Amazon EC2. Pour plus d'informations, consultez [Accès intercompte aux ressources dans IAM](#) dans le Guide de l'utilisateur IAM.

Gestion de l'accès à l'aide de politiques

Vous contrôlez l'accès en AWS créant des politiques et en les associant à AWS des identités ou à des ressources. Une politique définit les autorisations lorsqu'elles sont associées à une identité ou à une ressource. AWS évalue ces politiques lorsqu'un directeur fait une demande. La plupart des politiques sont stockées AWS sous forme de documents JSON. Pour plus d'informations les documents de politique JSON, consultez [Vue d'ensemble des politiques JSON](#) dans le Guide de l'utilisateur IAM.

À l'aide de politiques, les administrateurs précisent qui a accès à quoi en définissant quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

Par défaut, les utilisateurs et les rôles ne disposent d'aucune autorisation. Un administrateur IAM crée des politiques IAM et les ajoute aux rôles, que les utilisateurs peuvent ensuite assumer. Les politiques IAM définissent les autorisations quelle que soit la méthode que vous utilisez pour exécuter l'opération.

Identity-based politiques

Identity-based les politiques sont des documents de politique d'autorisation JSON que vous attachez à une identité (utilisateur, groupe ou rôle). Ces politiques contrôlent les actions que peuvent exécuter ces identités, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez [Définition d'autorisations IAM personnalisées avec des politiques gérées par le client](#) dans le Guide de l'utilisateur IAM.

Identity-based les politiques peuvent être des politiques intégrées (intégrées directement dans une seule identité) ou des politiques gérées (politiques autonomes associées à plusieurs identités). Pour découvrir comment choisir entre des politiques gérées et en ligne, consultez [Choix entre les politiques gérées et les politiques en ligne](#) dans le Guide de l'utilisateur IAM.

Resource-based politiques

Resource-based les politiques sont des documents de politique JSON que vous attachez à une ressource. Les exemples incluent les politiques de confiance de rôle IAM et les stratégies de compartiment Amazon S3. Dans les services qui sont compatibles avec les politiques basées sur les ressources, les administrateurs de service peuvent les utiliser pour contrôler l'accès à une ressource spécifique. Vous devez [spécifier un principal](#) dans une politique basée sur les ressources.

Resource-based les politiques sont des politiques intégrées qui se trouvent dans ce service. Vous ne pouvez pas utiliser les politiques AWS gérées par IAM dans une stratégie basée sur les ressources.

Listes de contrôle d'accès (ACL)

Les listes de contrôle d'accès (ACL) vérifie quels principaux (membres de compte, utilisateurs ou rôles) ont l'autorisation d'accéder à une ressource. Les listes de contrôle d'accès sont similaires aux politiques basées sur les ressources, bien qu'elles n'utilisent pas le format de document de politique JSON.

Amazon S3 et Amazon VPC sont des exemples de services qui prennent en charge les ACL. AWS WAF Pour en savoir plus sur les listes de contrôle d'accès, consultez [Vue d'ensemble des listes de contrôle d'accès \(ACL\)](#) dans le Guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge des types de politiques supplémentaires qui peuvent définir les autorisations maximales accordées par les types de politiques les plus courants :

- Limites d'autorisations : une limite des autorisations définit le nombre maximum d'autorisations qu'une politique basée sur l'identité peut accorder à une entité IAM. Pour plus d'informations, consultez [Limites d'autorisations pour des entités IAM](#) dans le Guide de l'utilisateur IAM.
- Politiques de contrôle des services (SCP) : spécifient les autorisations maximales pour une organisation ou une unité organisationnelle dans AWS Organizations. Pour plus d'informations, consultez [Politiques de contrôle de service](#) dans le Guide de l'utilisateur AWS Organizations .
- Politiques de contrôle des ressources (RCP) : définissent les autorisations maximales disponibles pour les ressources de votre organisation. Pour plus d'informations, consultez [Politiques de contrôle des ressources \(RCP\)](#) dans le Guide de l'utilisateur AWS Organizations .
- Politiques de session : politiques avancées que vous passez en tant que paramètre lorsque vous créez par programmation une session temporaire pour un rôle ou un utilisateur fédéré. Pour plus d'informations, consultez [Politiques de session](#) dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations en résultant sont plus compliquées à comprendre. Pour savoir comment AWS déterminer s'il faut autoriser une demande lorsque plusieurs types de politiques sont impliqués, consultez la section [Logique d'évaluation des politiques](#) dans le guide de l'utilisateur IAM.

Comment Amazon Keyspaces fonctionne avec IAM

Avant d'utiliser IAM pour gérer l'accès à Amazon Keyspaces, vous devez connaître les fonctionnalités IAM disponibles avec Amazon Keyspaces. Pour obtenir une vue d'ensemble de la façon dont Amazon Keyspaces et les autres AWS services fonctionnent avec IAM, consultez les [AWS services compatibles avec IAM dans le guide de l'utilisateur d'IAM](#).

Rubriques

- [Politiques basées sur l'identité d'Amazon Keyspaces](#)
- [Politiques basées sur les ressources d'Amazon Keyspaces](#)
- [Autorisation basée sur les tags Amazon Keyspaces](#)
- [Rôles IAM chez Amazon Keyspaces](#)

Politiques basées sur l'identité d'Amazon Keyspaces

Avec les politiques IAM basées sur l'identité, vous pouvez spécifier des actions et ressources autorisées ou refusées, ainsi que les conditions dans lesquelles les actions sont autorisées ou refusées. Amazon Keyspaces prend en charge des actions et des ressources spécifiques, ainsi que des clés de condition. Pour en savoir plus sur tous les éléments que vous utilisez dans une politique JSON, consultez [Références des éléments de politique JSON IAM](#) dans le Guide de l'utilisateur IAM.

Pour voir les ressources et actions spécifiques au service Amazon Keyspaces, ainsi que les clés contextuelles de condition qui peuvent être utilisées pour les politiques d'autorisation IAM, consultez les [actions, les ressources et les clés de condition pour Amazon Keyspaces \(pour Apache Cassandra\)](#) dans le Service Authorization Reference.

Actions

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément `Action` d'une politique JSON décrit les actions que vous pouvez utiliser pour autoriser ou refuser l'accès à une politique. Intégration d'actions dans une politique afin d'accorder l'autorisation d'exécuter les opérations associées.

Les actions politiques dans Amazon Keyspaces utilisent le préfixe suivant avant l'action : `cassandra:` Par exemple, pour autoriser quelqu'un à créer un espace de touches Amazon

Keyspaces à l'aide de l'instruction Amazon Keyspaces CREATE CQL, vous devez inclure l'action dans sa politique. `cassandra:Create` Les déclarations de politique doivent inclure un élément `Action` ou `NotAction`. Amazon Keyspaces définit son propre ensemble d'actions décrivant les tâches que vous pouvez effectuer avec ce service.

Pour spécifier plusieurs actions dans une seule déclaration, séparez-les par des virgules comme suit :

```
"Action": [  
    "cassandra:CREATE",  
    "cassandra:MODIFY"  
]
```

Pour consulter la liste des actions Amazon Keyspaces, consultez la section [Actions définies par Amazon Keyspaces \(pour Apache Cassandra\)](#) dans le Service Authorization Reference.

Ressources

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément de politique JSON `Resource` indique le ou les objets auxquels l'action s'applique. Il est recommandé de définir une ressource à l'aide de son [Amazon Resource Name \(ARN\)](#). Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*" 
```

Dans Amazon Keyspaces, les espaces clés, les tables et les flux peuvent être utilisés dans le cadre des autorisations Resource IAM.

Note

Pour accéder aux espaces clés et aux tables des utilisateurs dans Amazon Keyspaces, votre politique IAM doit `cassandra:Select` inclure des autorisations sur les tables système :

```
arn:${Partition}:cassandra:${Region}:${Account}:/keyspace/system*
```

Cela s'applique aux scénarios suivants :

- AWS Accès à la console de gestion
- les opérations sur les ressources du SDKGetKeyspace, par exemple GetTableListKeyspaces, et ListTables
- Connexions au pilote client Apache Cassandra standard, car les pilotes lisent automatiquement les tables système lors de l'initialisation de la connexion

Les tables système sont en lecture seule et ne peuvent pas être modifiées.

La ressource keyspace Amazon Keyspaces possède l'ARN suivant :

```
arn:${Partition}:cassandra:${Region}:${Account}:/keyspace/${keyspaceName}/
```

La ressource de table Amazon Keyspaces possède l'ARN suivant :

```
arn:${Partition}:cassandra:${Region}:${Account}:/keyspace/${keyspaceName}/table/${tableName}
```

La ressource de flux Amazon Keyspaces possède l'ARN suivant :

```
arn:${Partition}:cassandra:{Region}:${Account}:/keyspace/${keyspaceName}/table/${tableName}/stream/${streamLabel}
```

Pour plus d'informations sur le format des ARN, consultez [Amazon Resource Names \(ARN\) et espaces de noms de AWS services](#).

Par exemple, pour spécifier le keyspace mykeyspace dans votre déclaration, utilisez l'ARN suivant :

```
"Resource": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/"
```

Pour spécifier toutes les instances qui appartiennent à un compte spécifique, utilisez le caractère générique (*) :

```
"Resource": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/*"
```

Certaines actions Amazon Keyspaces, telles que celles relatives à la création de ressources, ne peuvent pas être effectuées sur une ressource spécifique. Dans ces cas-là, vous devez utiliser le caractère générique (*).

```
"Resource": "*"
```

Par exemple, pour accorder SELECT des autorisations à un principal IAM pour mytable inmykeyspace, le principal doit être autorisé à lire à la fois mytable etkeyspace/system*. Pour spécifier plusieurs ressources dans une seule instruction, séparez leurs ARN par des virgules.

```
"Resource": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/mytable",  
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system"
```

Pour consulter la liste des types de ressources Amazon Keyspaces et de leurs ARN, consultez la section [Resources Defined by Amazon Keyspaces \(pour Apache Cassandra\)](#) dans le Service Authorization Reference. Pour savoir avec quelles actions vous pouvez spécifier l'ARN de chaque ressource, consultez [Actions définies par Amazon Keyspaces \(pour Apache Cassandra\)](#).

Clés de condition

Les administrateurs peuvent utiliser les politiques AWS JSON pour spécifier qui a accès à quoi. C'est-à-dire, quel principal peut effectuer des actions sur quelles ressources et dans quelles conditions.

L'élément Condition indique à quel moment les instructions s'exécutent en fonction de critères définis. Vous pouvez créer des expressions conditionnelles qui utilisent des [opérateurs de condition](#), tels que les signes égal ou inférieur à, pour faire correspondre la condition de la politique aux valeurs de la demande. Pour voir toutes les clés de condition AWS globales, voir les clés de [contexte de condition AWS globales](#) dans le guide de l'utilisateur IAM.

Amazon Keyspaces définit son propre ensemble de clés de condition et prend également en charge l'utilisation de certaines clés de condition globales. Pour voir toutes les clés de condition AWS globales, voir les clés de [contexte de condition AWS globales](#) dans le guide de l'utilisateur IAM.

Toutes les actions Amazon Keyspaces prennent en charge les clés aws:RequestTag/\${TagKey}aws:ResourceTag/\${TagKey}, le et les clés de aws:TagKeys condition. Pour

de plus amples informations, veuillez consulter [the section called “Accès aux ressources Amazon Keyspaces basé sur des balises”](#).

Pour consulter la liste des clés de condition Amazon Keyspaces, consultez la section [Clés de condition pour Amazon Keyspaces \(pour Apache Cassandra\)](#) dans le Service Authorization Reference. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, consultez [Actions définies par Amazon Keyspaces \(pour Apache Cassandra\)](#).

Exemples

Pour consulter des exemples de politiques basées sur l'identité d'Amazon Keyspaces, consultez [Exemples de politiques basées sur l'identité d'Amazon Keyspaces](#)

Politiques basées sur les ressources d'Amazon Keyspaces

Amazon Keyspaces ne prend pas en charge les politiques basées sur les ressources. Pour afficher un exemple de page de stratégie basée sur les ressources détaillée, consultez <https://docs.aws.amazon.com/lambda/latest/dg/access-control-resource-based.html>.

Autorisation basée sur les tags Amazon Keyspaces

Vous pouvez gérer l'accès à vos ressources Amazon Keyspaces à l'aide de balises. Pour gérer l'accès aux ressources basé sur des balises, vous devez fournir les informations de balise dans l'[élément Condition](#) d'une stratégie utilisant les clés de condition `cassandra:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`. Pour plus d'informations sur le balisage des ressources Amazon Keyspaces, consultez [the section called “Utilisation des tags”](#)

Pour visualiser un exemple de stratégie basée sur l'identité permettant de limiter l'accès à une ressource en fonction des balises de cette ressource, veuillez consulter [Accès aux ressources Amazon Keyspaces basé sur des balises](#).

Rôles IAM chez Amazon Keyspaces

Un [rôle IAM](#) est une entité au sein de votre Compte AWS qui possède des autorisations spécifiques.

Utilisation d'informations d'identification temporaires avec Amazon Keyspaces

Vous pouvez utiliser des informations d'identification temporaires pour vous connecter à l'aide de la fédération pour endosser un rôle IAM ou bien un rôle entre comptes. Vous obtenez des informations

d'identification de sécurité temporaires en appelant des opérations d' AWS STS API telles que [AssumeRole](#) ou [GetFederationToken](#).

Amazon Keyspaces prend en charge l'utilisation d'informations d'identification temporaires avec le plugin d'authentification AWS Signature Version 4 (SigV4) disponible sur le référentiel Github pour les langues suivantes :

- Java : <https://github.com/aws/aws-sigv4-auth-cassandra-java-driver-plugin>.
- Node.js: <https://github.com/aws/aws-sigv4-auth-cassandra-nodejs-driver-plugin>.
- Python: <https://github.com/aws/aws-sigv4-auth-cassandra-python-driver-plugin>.
- Allez : <https://github.com/aws/aws-sigv4-auth-cassandra-gocql-driver-plugin>.

Pour des exemples et des didacticiels qui implémentent le plugin d'authentification pour accéder à Amazon Keyspaces par programmation, consultez. [the section called “Utilisation d'un pilote client Cassandra”](#)

Service-linked rôles

[Service-linked les rôles](#) permettent aux AWS services d'accéder aux ressources d'autres services pour effectuer une action en votre nom. Service-linked les rôles apparaissent dans votre compte IAM et appartiennent au service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

Pour en savoir plus sur la création ou la gestion des rôles liés aux services Amazon Keyspaces, consultez. [the section called “Utilisation des rôles liés à un service”](#)

Rôles du service

Amazon Keyspaces ne prend pas en charge les rôles de service.

Exemples de politiques basées sur l'identité d'Amazon Keyspaces

Par défaut, les utilisateurs et les rôles IAM ne sont pas autorisés à créer ou à modifier les ressources Amazon Keyspaces. Ils ne peuvent pas non plus effectuer de tâches à l'aide de la console, du CQLSH ou AWS de AWS CLI l'API. Un administrateur IAM doit créer des politiques IAM autorisant les utilisateurs et les rôles à exécuter des opérations d'API spécifiques sur les ressources spécifiées dont ils ont besoin. Il doit ensuite attacher ces politiques aux utilisateurs ou aux groupes IAM ayant besoin de ces autorisations.

Pour apprendre à créer une politique basée sur l'identité IAM à l'aide de ces exemples de documents de politique JSON, veuillez consulter [Création de politiques dans l'onglet JSON](#) dans le Guide de l'utilisateur IAM.

Rubriques

- [Bonnes pratiques en matière de politiques](#)
- [Utilisation de la console Amazon Keyspaces](#)
- [Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations](#)
- [Accès aux tables Amazon Keyspaces](#)
- [Accès aux ressources Amazon Keyspaces basé sur des balises](#)

Bonnes pratiques en matière de politiques

Identity-based les politiques déterminent si quelqu'un peut créer, accéder ou supprimer les ressources Amazon Keyspaces de votre compte. Ces actions peuvent entraîner des frais pour votre Compte AWS. Lorsque vous créez ou modifiez des politiques basées sur l'identité, suivez ces instructions et recommandations :

- Commencez AWS par les politiques gérées et passez aux autorisations du moindre privilège : pour commencer à accorder des autorisations à vos utilisateurs et à vos charges de travail, utilisez les politiques AWS gérées qui accordent des autorisations pour de nombreux cas d'utilisation courants. Ils sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire davantage les autorisations en définissant des politiques gérées par les AWS clients spécifiques à vos cas d'utilisation. Pour plus d'informations, consultez [politiques gérées par AWS](#) ou [politiques gérées par AWS pour les activités professionnelles](#) dans le Guide de l'utilisateur IAM.
- Accordez les autorisations de moindre privilège : lorsque vous définissez des autorisations avec des politiques IAM, accordez uniquement les autorisations nécessaires à l'exécution d'une seule tâche. Pour ce faire, vous définissez les actions qui peuvent être entreprises sur des ressources spécifiques dans des conditions spécifiques, également appelées autorisations de moindre privilège. Pour plus d'informations sur l'utilisation d'IAM pour appliquer des autorisations, consultez [politiques et autorisations dans IAM](#) dans le Guide de l'utilisateur IAM.
- Utilisez des conditions dans les politiques IAM pour restreindre davantage l'accès : vous pouvez ajouter une condition à vos politiques afin de limiter l'accès aux actions et aux ressources. Par exemple, vous pouvez écrire une condition de politique pour spécifier que toutes les demandes doivent être envoyées via SSL. Vous pouvez également utiliser des conditions pour accorder l'accès aux actions de service si elles sont utilisées par le biais d'un service spécifique Service

AWS, tel que CloudFormation. Pour plus d'informations, consultez [Conditions pour éléments de politique JSON IAM](#) dans le Guide de l'utilisateur IAM.

- Utilisez l'Analyseur d'accès IAM pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles : l'Analyseur d'accès IAM valide les politiques nouvelles et existantes de manière à ce que les politiques IAM respectent le langage de politique IAM (JSON) et les bonnes pratiques IAM. IAM Access Analyzer fournit plus de 100 vérifications de politiques et des recommandations exploitables pour vous aider à créer des politiques sécurisées et fonctionnelles. Pour plus d'informations, consultez [Validation de politiques avec IAM Access Analyzer](#) dans le Guide de l'utilisateur IAM.
- Exiger l'authentification multifactorielle (MFA) : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root, activez l'authentification MFA pour une sécurité accrue. Compte AWS Pour exiger la MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez [Sécurisation de l'accès aux API avec MFA](#) dans le Guide de l'utilisateur IAM.

Pour plus d'informations sur les bonnes pratiques dans IAM, consultez [Bonnes pratiques de sécurité dans IAM](#) dans le Guide de l'utilisateur IAM.

Utilisation de la console Amazon Keyspaces

Amazon Keyspaces n'a pas besoin d'autorisations spécifiques pour accéder à la console Amazon Keyspaces. Vous devez disposer d'au moins des autorisations en lecture seule pour répertorier et consulter les informations relatives aux ressources Amazon Keyspaces dans votre. Compte AWS Si vous créez une politique basée sur l'identité qui est plus restrictive que les autorisations minimales requises, la console ne fonctionnera pas comme prévu pour les entités (utilisateurs et rôles IAM) tributaires de cette politique.

Deux politiques AWS gérées sont à la disposition des entités pour l'accès à la console Amazon Keyspaces.

- [AmazonKeyspacesReadOnlyAccess_v2](#) — Cette politique accorde un accès en lecture seule à Amazon Keyspaces.
- [AmazonKeyspacesFullAccess](#) — Cette politique accorde l'autorisation d'utiliser Amazon Keyspaces avec un accès complet à toutes les fonctionnalités.

Pour plus d'informations sur les politiques gérées par Amazon Keyspaces, consultez [the section called “AWS politiques gérées”](#)

Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Cet exemple montre comment créer une politique qui permet aux utilisateurs IAM d'afficher les politiques en ligne et gérées attachées à leur identité d'utilisateur. Cette politique inclut les autorisations permettant d'effectuer cette action sur la console ou par programmation à l'aide de l'API AWS CLI or AWS .

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
      ],
      "Resource": ["arn:aws:iam::*:user/${aws:username}"]
    },
    {
      "Sid": "NavigateInConsole",
      "Effect": "Allow",
      "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
      ],
      "Resource": "*"
    }
  ]
}
```

Accès aux tables Amazon Keyspaces

Note

Pour accéder aux espaces clés et aux tables des utilisateurs dans Amazon Keyspaces, votre politique IAM doit `cassandra:Select` inclure des autorisations sur les tables système :

```
arn:${Partition}:cassandra:${Region}:${Account}:/keyspace/system*
```

Cela s'applique aux scénarios suivants :

- AWS Accès à la console de gestion
- les opérations sur les ressources du SDK `GetKeyspace`, par exemple `GetTableListKeyspaces`, et `ListTables`
- Connexions au pilote client Apache Cassandra standard, car les pilotes lisent automatiquement les tables système lors de l'initialisation de la connexion

Les tables système sont en lecture seule et ne peuvent pas être modifiées.

Voici un exemple de politique qui accorde un accès en lecture seule (SELECT) aux tables du système Amazon Keyspaces. Pour tous les exemples, remplacez la région et l'ID de compte dans le nom de ressource Amazon (ARN) par les vôtres.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cassandra:Select"
      ],
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
      ]
    }
  ]
}
```

L'exemple de politique suivant ajoute un accès en lecture seule à la table utilisateur `mytable` dans le `keyspace`. `mykeyspace`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cassandra:Select"
      ],
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/mytable",
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
      ]
    }
  ]
}
```

L'exemple de politique suivant attribue `read/write` l'accès à une table utilisateur et un accès en lecture aux tables système.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cassandra:Select",
        "cassandra:Modify"
      ],
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/mytable",
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
      ]
    }
  ]
}
```

L'exemple de stratégie suivant permet à un utilisateur de créer des tables dans le keyspace mykeyspace.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cassandra:Create",
        "cassandra:Select"
      ],
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/*",
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
      ]
    }
  ]
}
```

L'exemple de politique suivant attribue un accès en lecture aux tables système, mais restreint l'accès SELECT (lecture) et MODIFY (écriture) à la table utilisateur. mytable

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cassandra:Select"
      ],
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
      ]
    },
    {
      "Effect": "Deny",
      "Action": [
        "cassandra:Select",
        "cassandra:Modify"
      ],
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/mytable"
      ]
    }
  ]
}
```

```
    ]
  }
]
}
```

Accès aux ressources Amazon Keyspaces basé sur des balises

Vous pouvez utiliser les conditions de votre politique basée sur l'identité pour contrôler l'accès aux ressources Amazon Keyspaces en fonction de balises. Ces politiques contrôlent la visibilité des espaces clés et des tables du compte. Notez que les autorisations basées sur des balises pour les tables système se comportent différemment lorsque les demandes sont effectuées à l'aide du AWS SDK par rapport aux appels d'API Cassandra Query Language (CQL) via les pilotes Cassandra et les outils de développement.

- Pour effectuer Get des demandes List et y affecter des ressources avec le AWS SDK lors de l'utilisation de l'accès basé sur des balises, l'appelant doit disposer d'un accès en lecture aux tables système. Par exemple, des autorisations Select d'action sont nécessaires pour lire les données des tables système via l'GetTableopération. Si l'appelant ne dispose que d'un accès basé sur des balises à une table spécifique, une opération nécessitant un accès supplémentaire à une table système échouera.
- Pour des raisons de compatibilité avec le comportement établi du pilote Cassandra, les politiques d'autorisation basées sur les balises ne sont pas appliquées lors de l'exécution d'opérations sur les tables système à l'aide d'appels d'API CQL (Cassandra Query Language) via des pilotes Cassandra et des outils de développement.

L'exemple suivant montre comment créer une stratégie qui accorde des autorisations à un utilisateur pour afficher une table si son Owner contient la valeur du nom d'utilisateur de cet utilisateur. Dans cet exemple, vous accordez également un accès en lecture aux tables système.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadOnlyAccessTaggedTables",
      "Effect": "Allow",
      "Action": "cassandra:Select",
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/*",
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
      ]
    }
  ]
}
```

```

    ],
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/Owner": "${aws:username}"
      }
    }
  }
]
}

```

Vous pouvez rattacher cette politique aux utilisateurs IAM de votre compte. Si un utilisateur nommé `richard-roe` tente de consulter une table Amazon Keyspaces, la table doit être `Owner=richard-roe` balisée ou `owner=richard-roe`. Dans le cas contraire, l'utilisateur se voit refuser l'accès. La clé de condition d'étiquette `Owner` correspond à la fois à `Owner` et à `owner`, car les noms de clé de condition ne sont pas sensibles à la casse. Pour plus d'informations, consultez [Conditions pour éléments de politique JSON IAM](#) dans le Guide de l'utilisateur IAM.

La stratégie suivante autorise un utilisateur à créer des tables avec des balises si l'`Owner` de la table contient la valeur du nom d'utilisateur de cet utilisateur.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CreateTagTableUser",
      "Effect": "Allow",
      "Action": [
        "cassandra:Create",
        "cassandra:TagResource"
      ],
      "Resource": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/*",
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/Owner": "${aws:username}"
        }
      }
    }
  ]
}

```

AWS politiques gérées pour Amazon Keyspaces

Une politique AWS gérée est une politique autonome créée et administrée par AWS. AWS les politiques gérées sont conçues pour fournir des autorisations pour de nombreux cas d'utilisation courants afin que vous puissiez commencer à attribuer des autorisations aux utilisateurs, aux groupes et aux rôles.

N'oubliez pas que les politiques AWS gérées peuvent ne pas accorder d'autorisations de moindre privilège pour vos cas d'utilisation spécifiques, car elles sont accessibles à tous les AWS clients. Nous vous recommandons de réduire encore les autorisations en définissant des [politiques gérées par le client](#) qui sont propres à vos cas d'utilisation.

Vous ne pouvez pas modifier les autorisations définies dans les politiques AWS gérées. Si les autorisations définies dans une politique AWS gérée sont AWS mises à jour, la mise à jour affecte toutes les identités principales (utilisateurs, groupes et rôles) auxquelles la politique est attachée. AWS est le plus susceptible de mettre à jour une politique AWS gérée lorsqu'une nouvelle Service AWS est lancée ou lorsque de nouvelles opérations d'API sont disponibles pour les services existants.

Pour plus d'informations, consultez [Politiques gérées par AWS](#) dans le Guide de l'utilisateur IAM.

AWS politique gérée : AmazonKeyspacesReadOnlyAccess _v2

Vous pouvez associer la politique AmazonKeyspacesReadOnlyAccess_v2 à vos identités IAM.

Cette politique accorde un accès en lecture seule à Amazon Keyspaces et inclut les autorisations requises lors de la connexion via des points de terminaison VPC privés.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

- Amazon Keyspaces— Fournit un accès en lecture seule à Amazon Keyspaces.

- **Amazon Keyspaces CDC streams**— Permet aux directeurs d'écran de consulter les flux CDC d'Amazon Keyspaces.
- **Application Auto Scaling** : permet aux principaux de visualiser les configurations à partir d'Application Auto Scaling. Cela est nécessaire pour que les utilisateurs puissent afficher les politiques de mise à l'échelle automatique associées à une table.
- **CloudWatch**— Permet aux principaux de consulter les données métriques et les alarmes configurées dans CloudWatch. Cela est nécessaire pour que les utilisateurs puissent voir la taille de la table facturable et les CloudWatch alarmes configurées pour une table.
- **AWS KMS**— Permet aux principaux d'afficher les clés configurées dans AWS KMS. Cela est nécessaire pour que les utilisateurs puissent consulter AWS KMS les clés qu'ils créent et gèrent dans leur compte afin de confirmer que la clé attribuée à Amazon Keyspaces est une clé de chiffrement symétrique activée.
- **Amazon EC2**— Permet aux principaux se connectant à Amazon Keyspaces via des points de terminaison VPC d'interroger le VPC de votre instance Amazon EC2 pour obtenir des informations sur le point de terminaison et l'interface réseau. Cet accès en lecture seule à l'instance Amazon EC2 est requis pour qu'Amazon Keyspaces puisse rechercher et stocker les points de terminaison VPC d'interface disponibles dans le tableau utilisé pour l'équilibrage de charge de connexion.
`system.peers`

Pour consulter le JSON format de la politique, consultez [AmazonKeyspacesReadOnlyAccess_v2](#).

AWS politique gérée : AmazonKeyspacesReadOnlyAccess

Vous pouvez associer la politique AmazonKeyspacesReadOnlyAccess à vos identités IAM.

Cette politique accorde un accès en lecture seule à Amazon Keyspaces.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

- **Amazon Keyspaces**— Fournit un accès en lecture seule à Amazon Keyspaces.
- **Amazon Keyspaces CDC streams**— Permet aux directeurs d'écran de consulter les flux CDC d'Amazon Keyspaces.

- **Application Auto Scaling** : permet aux principaux de visualiser les configurations à partir d'Application Auto Scaling. Cela est nécessaire pour que les utilisateurs puissent afficher les politiques de mise à l'échelle automatique associées à une table.
- **CloudWatch**— Permet aux principaux de consulter les données métriques et les alarmes configurées dans CloudWatch. Cela est nécessaire pour que les utilisateurs puissent voir la taille de la table facturable et les CloudWatch alarmes configurées pour une table.
- **AWS KMS**— Permet aux principaux d'afficher les clés configurées dans AWS KMS. Cela est nécessaire pour que les utilisateurs puissent consulter AWS KMS les clés qu'ils créent et gèrent dans leur compte afin de confirmer que la clé attribuée à Amazon Keyspaces est une clé de chiffrement symétrique activée.

Pour consulter le JSON format de la politique, voir [AmazonKeyspacesReadOnlyAccess](#).

AWS politique gérée : AmazonKeyspacesFullAccess

Vous pouvez associer la politique AmazonKeyspacesFullAccess à vos identités IAM.

Cette politique accorde des autorisations administratives qui permettent à vos administrateurs d'accéder sans restriction à Amazon Keyspaces.

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

- **Amazon Keyspaces**— Permet aux directeurs d'accéder à n'importe quelle ressource Amazon Keyspaces et d'effectuer toutes les actions.
- **Application Auto Scaling**— Permet aux principaux de créer, de consulter et de supprimer des politiques de dimensionnement automatique pour les tables Amazon Keyspaces. Cela est nécessaire pour que les administrateurs puissent gérer les politiques de dimensionnement automatique pour les tables Amazon Keyspaces.
- **CloudWatch**— Permet aux donneurs d'ordre de voir la taille de la table facturable ainsi que de créer, de consulter et de supprimer des CloudWatch alarmes conformément aux politiques de dimensionnement automatique d'Amazon Keyspaces. Cela est nécessaire pour que les

administrateurs puissent consulter la taille de la table facturable et créer un CloudWatch tableau de bord.

- IAM— Permet à Amazon Keyspaces de créer automatiquement des rôles liés à un service avec IAM lorsque les fonctionnalités suivantes sont activées :
 - Amazon Keyspaces CDC streams— Lorsqu'un administrateur active un flux pour une table, Amazon Keyspaces crée le rôle lié au service [AWSServiceRoleForAmazonKeyspacesCDC](#) pour publier des CloudWatch statistiques sur votre compte en votre nom.
 - Application Auto Scaling— Lorsqu'un administrateur active Application Auto Scaling pour une table, Amazon Keyspaces crée le rôle lié au service [AWSServiceRoleForApplicationAutoScaling_CassandraTable](#) pour effectuer des actions de dimensionnement automatique en votre nom.
 - Amazon Keyspaces multi-Region replication— Lorsqu'un administrateur crée un nouvel espace clé multirégional ou en ajoute un nouveau Région AWS à un espace clé mono-régional existant, Amazon Keyspaces crée le [AWSServiceRoleForAmazonKeyspacesReplication](#) rôle lié à un service pour effectuer la réplication des tables, des données et des métadonnées dans les régions sélectionnées en votre nom.
- AWS KMS : permet aux principaux d'afficher les clés configurées dans AWS KMS. Cela est nécessaire pour que les utilisateurs puissent consulter AWS KMS les clés qu'ils créent et gèrent dans leur compte afin de confirmer que la clé attribuée à Amazon Keyspaces est une clé de chiffrement symétrique activée.
- Amazon EC2— Permet aux principaux se connectant à Amazon Keyspaces via des points de terminaison VPC d'interroger le VPC de votre instance Amazon EC2 pour obtenir des informations sur le point de terminaison et l'interface réseau. Cet accès en lecture seule à l'instance Amazon EC2 est requis pour qu'Amazon Keyspaces puisse rechercher et stocker les points de terminaison VPC d'interface disponibles dans le tableau utilisé pour l'équilibrage de charge de connexion.
`system.peers`

Pour consulter le JSON format de la politique, voir [AmazonKeyspacesFullAccess](#).

AWS politique gérée : KeyspacesCDCServiceRolePolicy

Vous ne pouvez pas joindre de KeyspacesCDCServiceRolePolicy à vos entités IAM. Cette politique est associée à un rôle lié à un service qui permet à Amazon Keyspaces d'effectuer des

actions en votre nom. Pour de plus amples informations, veuillez consulter [the section called “Flux CDC”](#).

Cette politique accorde les autorisations requises au rôle lié au service pour publier en votre `AWSServiceRoleForAmazonKeyspacesCDC` nom les données de métriques de flux CDC d'Amazon Keyspaces. CloudWatch

Détails de l'autorisation

Cette politique inclut les autorisations suivantes.

- CloudWatch— Permet au rôle lié au service de publier [AWSServiceRoleForAmazonKeyspacesCDC](#) les données métriques des flux CDC d'Amazon Keyspaces sur votre compte en votre nom. `"cloudwatch:namespace": "AWS/Cassandra"` CloudWatch

Pour consulter le JSON format de la politique, voir [KeyspacesCDCServiceRolePolicy](#).

Amazon Keyspaces met à jour les politiques gérées AWS

Consultez les informations relatives aux mises à jour des politiques AWS gérées pour Amazon Keyspaces depuis que ce service a commencé à suivre ces modifications. Pour obtenir des alertes automatiques sur les modifications apportées à cette page, abonnez-vous au flux RSS de la page [Historique de la documentation](#).

Modifier	Description	Date
KeyspacesCDCServiceRolePolicy : nouvelle politique	Amazon Keyspaces a ajouté une nouvelle politique gérée <code>KeyspacesCDCServiceRolePolicy</code> qui accorde les autorisations requises au rôle lié à un service pour	2 juillet 2025

Modifier	Description	Date
	publier en votre nom les données des métriques de flux CDC d'AWSServiceRoleForAmazonKeyspacesCDC Amazon Keyspaces. CloudWatch Pour de plus amples informations, veuillez consulter the section called “Flux CDC” .	
AmazonKeyspacesReadOnlyAccess_v2 — Mise à jour d'une politique existante	Amazon Keyspaces a ajouté de nouvelles autorisations pour permettre aux responsables de l'IAM de consulter les streams du CDC d'Amazon Keyspaces. Pour de plus amples informations, veuillez consulter the section called “Afficher les flux du CDC” .	2 juillet 2025
AmazonKeyspacesReadOnlyAccess : mise à jour d'une politique existante	Amazon Keyspaces a ajouté de nouvelles autorisations pour permettre aux responsables de l'IAM de consulter les streams du CDC d'Amazon Keyspaces. Pour de plus amples informations, veuillez consulter the section called “Afficher les flux du CDC” .	2 juillet 2025

Modifier	Description	Date
AmazonKeyspacesFullAccess : mise à jour d'une politique existante	<p>Amazon Keyspaces a créé la politique Keyspaces CDCServiceRolePolicy gérée pour le rôle lié au service AWSServiceRoleForAmazonKeyspacesCDC afin d'ajouter les autorisations requises lorsqu'un administrateur active un flux pour une table.</p> <p>Amazon Keyspaces utilise le rôle lié au service AWSServiceRoleForAmazonKeyspacesCDC pour publier des CloudWatch statistiques sur votre compte en votre nom. Pour de plus amples informations, veuillez consulter the section called “Flux CDC”.</p>	2 juillet 2025

Modifier	Description	Date
AmazonKeyspacesFullAccess : mise à jour d'une politique existante	<p>Amazon Keyspaces a mis à jour le Keyspaces ReplicationService RolePolicy rôle lié au service AWSServiceRoleForAmazonKeyspacesReplication afin d'ajouter les autorisations requises lorsqu'un administrateur ajoute un nouveau rôle Région AWS à un espace clé unique ou multirégional.</p> <p>Amazon Keyspaces utilise le rôle lié au service AWSServiceRoleForAmazonKeyspacesReplication pour répliquer les tables, leurs paramètres et les données en votre nom. Pour de plus amples informations, veuillez consulter the section called “Multi-Region Réplication”.</p>	19 novembre 2024

Modifier	Description	Date
AmazonKeyspacesFullAccess : mise à jour d'une politique existante	<p>Amazon Keyspaces a ajouté de nouvelles autorisations pour permettre à Amazon Keyspaces de créer un rôle lié à un service lorsqu'un administrateur ajoute une nouvelle région à un espace clé unique ou multirégional.</p> <p>Amazon Keyspaces utilise le rôle lié au service pour effectuer des tâches de réplication de données en votre nom. Pour de plus amples informations, veuillez consulter the section called "Multi-Region Réplication".</p>	3 octobre 2023

Modifier	Description	Date
AmazonKeyspacesReadOnlyAccess_v2 — Nouvelle politique	<p>Amazon Keyspaces a créé une nouvelle politique visant à ajouter des autorisations en lecture seule pour les clients se connectant à Amazon Keyspaces via des points de terminaison VPC d'interface afin d'accéder à l'instance Amazon EC2 afin de consulter des informations réseau.</p> <p>Amazon Keyspaces stocke les points de terminaison VPC d'interface disponibles dans le <code>system.peers</code> tableau pour l'équilibrage de charge de connexion. Pour de plus amples informations, veuillez consulter the section called “Utilisation des points de terminaison de VPC d'interface”.</p>	12 septembre 2023

Modifier	Description	Date
AmazonKeyspacesFullAccess : mise à jour d'une politique existante	<p>Amazon Keyspaces a ajouté de nouvelles autorisations pour permettre à Amazon Keyspaces de créer un rôle lié à un service lorsqu'un administrateur crée un espace de touches multirégional.</p> <p>Amazon Keyspaces utilise le rôle lié au service <code>AWSServiceRoleForAmazonKeyspacesReplication</code> pour effectuer des tâches de réplication de données en votre nom. Pour de plus amples informations, veuillez consulter the section called “Multi-Region Réplication”.</p>	5 juin 2023
AmazonKeyspacesReadOnlyAccess : mise à jour d'une politique existante	<p>Amazon Keyspaces a ajouté de nouvelles autorisations pour permettre aux utilisateurs de consulter la taille facturable d'une table en utilisant CloudWatch</p> <p>Amazon Keyspaces s'intègre CloudWatch à Amazon pour vous permettre de contrôler la taille des tables facturables. Pour de plus amples informations, veuillez consulter the section called “Métriques Amazon Keyspaces”.</p>	7 juillet 2022

Modifier	Description	Date
AmazonKeyspacesFullAccess : mise à jour d'une politique existante	<p>Amazon Keyspaces a ajouté de nouvelles autorisations pour permettre aux utilisateurs de consulter la taille facturée d'une table en utilisant CloudWatch.</p> <p>Amazon Keyspaces s'intègre CloudWatch à Amazon pour vous permettre de contrôler la taille des tables facturables. Pour de plus amples informations, veuillez consulter the section called "Métriques Amazon Keyspaces".</p>	7 juillet 2022
AmazonKeyspacesReadOnlyAccess : mise à jour d'une politique existante	<p>Amazon Keyspaces a ajouté de nouvelles autorisations pour permettre aux utilisateurs de consulter AWS KMS les clés configurées pour le chiffrement Amazon Keyspaces au repos.</p> <p>Le chiffrement au repos d'Amazon Keyspaces s'intègre AWS KMS à la protection et à la gestion des clés de chiffrement utilisées pour chiffrer les données au repos. Pour afficher la AWS KMS clé configurée pour Amazon Keyspaces, des autorisations de lecture seule ont été ajoutées.</p>	1er juin 2021

Modifier	Description	Date
AmazonKeyspacesFullAccess : mise à jour d'une politique existante	<p>Amazon Keyspaces a ajouté de nouvelles autorisations pour permettre aux utilisateurs de consulter AWS KMS les clés configurées pour le chiffrement Amazon Keyspaces au repos.</p> <p>Le chiffrement au repos d'Amazon Keyspaces s'intègre AWS KMS à la protection et à la gestion des clés de chiffrement utilisées pour chiffrer les données au repos. Pour afficher la AWS KMS clé configurée pour Amazon Keyspaces, des autorisations de lecture seule ont été ajoutées.</p>	1er juin 2021
Amazon Keyspaces a commencé à suivre les modifications	Amazon Keyspaces a commencé à suivre les modifications apportées à ses politiques AWS gérées.	1er juin 2021

Résolution des problèmes d'identité et d'accès à Amazon Keyspaces

Utilisez les informations suivantes pour vous aider à diagnostiquer et à résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec Amazon Keyspaces et IAM.

Rubriques

- [Je ne suis pas autorisé à effectuer une action dans Amazon Keyspaces](#)
- [J'ai modifié un utilisateur ou un rôle IAM et les modifications n'ont pas pris effet immédiatement](#)

- [Je ne parviens pas à restaurer une table à l'aide de la restauration instantanée \(PITR\) d'Amazon Keyspaces](#)
- [Je ne suis pas autorisé à effectuer iam : PassRole](#)
- [Je suis administrateur et je souhaite autoriser d'autres personnes à accéder à Amazon Keyspaces](#)
- [Je souhaite autoriser des personnes extérieures à moi Compte AWS à accéder à mes ressources Amazon Keyspaces](#)

Je ne suis pas autorisé à effectuer une action dans Amazon Keyspaces

S'il vous Console de gestion AWS indique que vous n'êtes pas autorisé à effectuer une action, vous devez contacter votre administrateur pour obtenir de l'aide. Votre administrateur est la personne qui vous a fourni votre nom d'utilisateur et votre mot de passe.

L'exemple d'erreur suivant se produit lorsque l'utilisateur mateojackson IAM essaie d'utiliser la console pour afficher les détails d'une table *table* mais ne dispose pas `cassandra:Select` des autorisations nécessaires.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
cassandra:Select on resource: mytable
```

Dans ce cas, Mateo demande à son administrateur de mettre à jour ses politiques pour lui permettre d'accéder à la ressource *mytable* à l'aide de l'action `cassandra:Select`.

J'ai modifié un utilisateur ou un rôle IAM et les modifications n'ont pas pris effet immédiatement

Les modifications de la politique IAM peuvent prendre jusqu'à 10 minutes pour prendre effet pour les applications disposant de connexions existantes et établies à Amazon Keyspaces. Les modifications de politique IAM prennent effet immédiatement lorsque les applications établissent une nouvelle connexion. Si vous avez apporté des modifications à un utilisateur ou à un rôle IAM existant et que cela n'a pas pris effet immédiatement, attendez 10 minutes ou déconnectez-vous et reconnectez-vous à Amazon Keyspaces.

Je ne parviens pas à restaurer une table à l'aide de la restauration instantanée (PITR) d'Amazon Keyspaces

Si vous essayez de restaurer une table Amazon Keyspaces avec une restauration instantanée (PITR) et que vous constatez que le processus de restauration commence mais ne se termine pas

correctement, vous n'avez peut-être pas configuré toutes les autorisations requises pour le processus de restauration. Vous devez contacter votre administrateur pour obtenir de l'aide et lui demander de mettre à jour vos politiques afin de vous permettre de restaurer une table dans Amazon Keyspaces.

Outre les autorisations des utilisateurs, Amazon Keyspaces peut avoir besoin d'autorisations pour effectuer des actions au nom de votre principal pendant le processus de restauration. C'est le cas si la table est chiffrée à l'aide d'une clé gérée par le client ou si vous utilisez des politiques IAM qui limitent le trafic entrant. Par exemple, si vous utilisez des clés de condition dans votre politique IAM pour limiter le trafic source à des points de terminaison ou à des plages d'adresses IP spécifiques, l'opération de restauration échoue. Pour permettre à Amazon Keyspaces d'effectuer l'opération de restauration des tables pour le compte de votre principal, vous devez ajouter une clé de condition `aws:ViaAWSService` globale dans la politique IAM.

Pour plus d'informations sur les autorisations de restauration de tables, consultez [the section called “Configurer les autorisations IAM pour la restauration”](#).

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez un message d'erreur indiquant que vous n'êtes pas autorisé à effectuer l'`iam:PassRole` action, vos politiques doivent être mises à jour pour vous permettre de transmettre un rôle à Amazon Keyspaces.

Certains services AWS permettent de transmettre un rôle existant à ce service au lieu de créer un nouveau rôle de service ou un rôle lié à un service. Pour ce faire, vous devez disposer des autorisations nécessaires pour transmettre le rôle au service.

L'exemple d'erreur suivant se produit lorsqu'un utilisateur IAM nommé `marymajor` essaie d'utiliser la console pour effectuer une action dans Amazon Keyspaces. Toutefois, l'action nécessite que le service ait des autorisations accordées par un rôle de service. Mary n'est pas autorisée à transmettre le rôle au service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

Dans ce cas, les politiques de Mary doivent être mises à jour pour lui permettre d'exécuter l'action `iam:PassRole`.

Si vous avez besoin d'aide, contactez votre AWS administrateur. Votre administrateur vous a fourni vos informations d'identification de connexion.

Je suis administrateur et je souhaite autoriser d'autres personnes à accéder à Amazon Keyspaces

Pour autoriser d'autres personnes à accéder à Amazon Keyspaces, vous devez accorder l'autorisation aux personnes ou aux applications qui ont besoin d'y accéder. Si vous utilisez AWS IAM Identity Center pour gérer des personnes et des applications, vous attribuez des ensembles d'autorisations aux utilisateurs ou aux groupes afin de définir leur niveau d'accès. Les ensembles d'autorisations créent et attribuent automatiquement des politiques IAM aux rôles IAM associés à la personne ou à l'application. Pour plus d'informations, consultez la section [Ensembles d'autorisations](#) dans le guide de AWS IAM Identity Center l'utilisateur.

Si vous n'utilisez pas IAM Identity Center, vous devez créer des entités IAM (utilisateurs ou rôles) pour les personnes ou les applications qui ont besoin d'un accès. Vous devez ensuite joindre une politique à l'entité qui lui accorde les autorisations appropriées dans Amazon Keyspaces. Une fois les autorisations accordées, fournissez les informations d'identification à l'utilisateur ou au développeur de l'application. Ils utiliseront ces informations d'identification pour y accéder AWS. Pour en savoir plus sur la création d'utilisateurs, de groupes, de politiques et d'autorisations [IAM, consultez la section Identités, politiques et autorisations IAM dans le guide de l'utilisateur IAM](#).

Je souhaite autoriser des personnes extérieures à moi Compte AWS à accéder à mes ressources Amazon Keyspaces

Vous pouvez créer un rôle que les utilisateurs provenant d'autres comptes ou les personnes extérieures à votre organisation pourront utiliser pour accéder à vos ressources. Vous pouvez spécifier qui est autorisé à assumer le rôle. Pour les services qui prennent en charge les politiques basées sur les ressources ou les listes de contrôle d'accès (ACL), vous pouvez utiliser ces politiques pour donner l'accès à vos ressources.

Pour plus d'informations, consultez les éléments suivants :

- Pour savoir si Amazon Keyspaces prend en charge ces fonctionnalités, consultez. [Comment Amazon Keyspaces fonctionne avec IAM](#)
- Pour savoir comment fournir l'accès à vos ressources sur celles Comptes AWS que vous possédez, consultez la section [Fournir l'accès à un utilisateur IAM dans un autre utilisateur Compte AWS que vous possédez](#) dans le Guide de l'utilisateur IAM.
- Pour savoir comment fournir l'accès à vos ressources à des tiers Comptes AWS, consultez la section [Fournir un accès à des ressources Comptes AWS détenues par des tiers](#) dans le guide de l'utilisateur IAM.

- Pour savoir comment fournir un accès par le biais de la fédération d'identité, consultez [Fournir un accès à des utilisateurs authentifiés en externe \(fédération d'identité\)](#) dans le Guide de l'utilisateur IAM.
- Pour en savoir plus sur la différence entre l'utilisation des rôles et des politiques basées sur les ressources pour l'accès intercompte, consultez [Accès intercompte aux ressources dans IAM](#) dans le Guide de l'utilisateur IAM.

Utilisation de rôles liés à un service pour Amazon Keyspaces

[Amazon Keyspaces \(pour Apache Cassandra\) utilise des rôles liés à un service Gestion des identités et des accès AWS \(IAM\)](#). Un rôle lié à un service est un type unique de rôle IAM directement lié à Amazon Keyspaces. Service-linked les rôles sont prédéfinis par Amazon Keyspaces et incluent toutes les autorisations dont le service a besoin pour appeler d'autres AWS services en votre nom.

Pour plus d'informations sur les autres services qui prennent en charge les rôles liés à un service, consultez la section [AWS Services qui fonctionnent avec IAM](#) et recherchez les services dont la valeur est Oui dans la Service-linked colonne des rôles. Choisissez un Oui ayant un lien permettant de consulter les détails du rôle pour ce service.

Rubriques

- [Utilisation des rôles pour le dimensionnement automatique des applications Amazon Keyspaces](#)
- [Utilisation des rôles pour Amazon Keyspaces Replication Multi-Region](#)
- [Utilisation de rôles pour les flux CDC d'Amazon Keyspaces](#)

Utilisation des rôles pour le dimensionnement automatique des applications Amazon Keyspaces

[Amazon Keyspaces \(pour Apache Cassandra\) utilise des rôles liés à un service Gestion des identités et des accès AWS \(IAM\)](#). Un rôle lié à un service est un type unique de rôle IAM directement lié à Amazon Keyspaces. Service-linked les rôles sont prédéfinis par Amazon Keyspaces et incluent toutes les autorisations dont le service a besoin pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration d'Amazon Keyspaces, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Amazon Keyspaces définit les autorisations associées à ses rôles liés aux services et, sauf indication contraire, seul Amazon Keyspaces peut assumer ses rôles. Les autorisations définies comprennent la politique de confiance et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Cela protège vos ressources Amazon Keyspaces, car vous ne pouvez pas supprimer par inadvertance l'autorisation d'accès aux ressources.

Service-linked autorisations de rôle pour Amazon Keyspaces

Amazon Keyspaces utilise le rôle lié au service nommé `AWSServiceRoleForApplicationAutoScaling_CassandraTable` pour permettre à Application Auto Scaling d'appeler Amazon Keyspaces et Amazon en votre nom. CloudWatch

Le rôle `CassandraTable` lié au service `AWSServiceRoleForApplicationAutoScaling` _ fait confiance aux services suivants pour assumer le rôle :

- `cassandra.application-autoscaling.amazonaws.com`

La politique d'autorisation des rôles permet à Application Auto Scaling d'effectuer les actions suivantes sur les ressources Amazon Keyspaces spécifiées :

- Action : `cassandra:Select` sur `arn:*:cassandra:*:*:/keyspace/system/table/*`
- Action : `cassandra:Select` sur la ressource `arn:*:cassandra:*:*:/keyspace/system_schema/table/*`
- Action : `cassandra:Select` sur la ressource `arn:*:cassandra:*:*:/keyspace/system_schema_mcs/table/*`
- Action : `cassandra:Alter` sur la ressource `arn:*:cassandra:*:*:""`

Création d'un rôle lié à un service pour Amazon Keyspaces

Il n'est pas nécessaire de créer manuellement un rôle lié à un service pour le dimensionnement automatique d'Amazon Keyspaces. Lorsque vous activez le dimensionnement automatique d'Amazon Keyspaces sur une table avec le CQL Console de gestion AWS, le ou l' AWS API AWS CLI, Application Auto Scaling crée le rôle lié au service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous activez le dimensionnement automatique d'Amazon Keyspaces pour une table, Application Auto Scaling crée à nouveau le rôle lié au service pour vous.

 Important

Ce rôle lié à un service peut apparaître dans votre compte si vous avez effectué une action dans un autre service qui utilise les fonctions prises en charge par ce rôle. Pour en savoir plus, voir [Un nouveau rôle est apparu dans mon Compte AWS](#).

Modification d'un rôle lié à un service pour Amazon Keyspaces

Amazon Keyspaces ne vous permet pas de modifier le rôle lié au CassandraTable service AWSServiceRoleForApplicationAutoScaling. Après avoir créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez [Modification d'un rôle lié à un service](#) dans le Guide de l'utilisateur IAM.

Supprimer un rôle lié à un service pour Amazon Keyspaces

Si vous n'avez plus besoin d'utiliser une fonction ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, aucune entité inutilisée n'est surveillée ou gérée activement. Toutefois, vous devez d'abord désactiver le dimensionnement automatique sur toutes les tables du compte Régions AWS avant de pouvoir supprimer manuellement le rôle lié à un service. Pour désactiver le dimensionnement automatique sur les tables Amazon Keyspaces, consultez [the section called "Désactiver le dimensionnement automatique d'Amazon Keyspaces pour un tableau"](#)

 Note

Si le dimensionnement automatique d'Amazon Keyspaces utilise le rôle lorsque vous essayez de modifier les ressources, la désinscription risque d'échouer. Si cela se produit, patientez quelques minutes et réessayez.

Pour supprimer manuellement le rôle lié au service à l'aide d'IAM

Utilisez la console IAM, le AWS CLI, ou l' AWS API pour supprimer le rôle lié au CassandraTable service AWSServiceRoleForApplicationAutoScaling. Pour plus d'informations, consultez [la section Suppression d'un Service-Linked rôle](#) dans le guide de l'utilisateur IAM.

 Note

Pour supprimer le rôle lié au service utilisé par le dimensionnement automatique d'Amazon Keyspaces, vous devez d'abord désactiver le dimensionnement automatique sur toutes les tables du compte.

Régions prises en charge pour les rôles liés au service Amazon Keyspaces

Amazon Keyspaces prend en charge l'utilisation de rôles liés à un service dans toutes les régions où le service est disponible. Pour plus d'informations, consultez la section [Points de terminaison de service pour Amazon Keyspaces](#).

Utilisation des rôles pour Amazon Keyspaces Replication Multi-Region

[Amazon Keyspaces \(pour Apache Cassandra\) utilise des rôles liés à un service Gestion des identités et des accès AWS \(IAM\)](#). Un rôle lié à un service est un type unique de rôle IAM directement lié à Amazon Keyspaces. Service-linked les rôles sont prédéfinis par Amazon Keyspaces et incluent toutes les autorisations dont le service a besoin pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration d'Amazon Keyspaces, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Amazon Keyspaces définit les autorisations associées à ses rôles liés aux services et, sauf indication contraire, seul Amazon Keyspaces peut assumer ses rôles. Les autorisations définies comprennent la politique de confiance et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous pouvez supprimer un rôle lié à un service uniquement après la suppression préalable de ses ressources connexes. Cela protège vos ressources Amazon Keyspaces, car vous ne pouvez pas supprimer par inadvertance l'autorisation d'accès aux ressources.

Service-linked autorisations de rôle pour Amazon Keyspaces

Amazon Keyspaces utilise le rôle lié au service nommé pour permettre à `AWSServiceRoleForAmazonKeyspacesReplication` Amazon Keyspaces d'en ajouter un nouveau Régions AWS à un espace de touches en votre nom, et de répliquer les tables ainsi que toutes leurs données et paramètres dans la nouvelle région. Ce rôle permet également à Amazon Keyspaces de répliquer les écritures sur les tables de toutes les régions en votre nom.

Le rôle `AWSServiceRoleForAmazonKeyspacesReplication` lié à un service fait confiance aux services suivants pour assumer le rôle :

- `replication.cassandra.amazonaws.com`

La politique d'autorisation des rôles nommée `KeyspacesReplicationServiceRolePolicy` permet à Amazon Keyspaces d'effectuer les actions suivantes :

- Action : `cassandra:Select`
- Action : `cassandra:SelectMultiRegionResource`
- Action : `cassandra:Modify`
- Action : `cassandra:ModifyMultiRegionResource`
- Action : `cassandra:AlterMultiRegionResource`
- Action : `application-autoscaling:RegisterScalableTarget` — Amazon Keyspaces utilise les autorisations de dimensionnement automatique de l'application lorsque vous ajoutez une réplique à une seule table de régions en mode provisionné avec le dimensionnement automatique activé.
- Action : `application-autoscaling:DeregisterScalableTarget`
- Action : `application-autoscaling:DescribeScalableTargets`
- Action : `application-autoscaling:PutScalingPolicy`
- Action : `application-autoscaling:DescribeScalingPolicies`
- Action : `cassandra:Alter`
- Action : `cloudwatch>DeleteAlarms`
- Action : `cloudwatch:DescribeAlarms`
- Action : `cloudwatch:PutMetricAlarm`

Bien que le rôle lié au service Amazon Keyspaces `AWSServiceRoleForAmazonKeyspacesReplication` fournisse les autorisations suivantes : « Action : » pour le nom de ressource Amazon (ARN) « `arn : *` » spécifié dans la politique, Amazon Keyspaces fournit l'ARN de votre compte.

Les autorisations permettant de créer le rôle lié au service `AWSServiceRoleForAmazonKeyspacesReplication` sont incluses dans la politique `AmazonKeyspacesFullAccess` gérée. Pour de plus amples informations, veuillez consulter [the section called “AmazonKeyspacesFullAccess”](#).

Vous devez configurer les autorisations de manière à permettre à vos utilisateurs, groupes ou rôles de créer, modifier ou supprimer un rôle lié à un service. Pour plus d'informations, consultez les [autorisations relatives aux Service-linked rôles](#) dans le guide de l'utilisateur IAM.

Création d'un rôle lié à un service pour Amazon Keyspaces

Vous ne pouvez pas créer manuellement un rôle lié à un service. Lorsque vous créez un espace de saisie multirégional dans l'API Console de gestion AWS, le ou l' AWS API AWS CLI, Amazon Keyspaces crée le rôle lié à un service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous créez un espace clé multirégional, Amazon Keyspaces crée à nouveau le rôle lié à un service pour vous.

Modification d'un rôle lié à un service pour Amazon Keyspaces

Amazon Keyspaces ne vous permet pas de modifier le rôle lié au `AWSServiceRoleForAmazonKeyspacesReplication` service. Après avoir créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez [Modification d'un rôle lié à un service](#) dans le Guide de l'utilisateur IAM.

Supprimer un rôle lié à un service pour Amazon Keyspaces

Si vous n'avez plus besoin d'utiliser une fonctionnalité ou un service qui nécessite un rôle lié à un service, nous vous recommandons de supprimer ce rôle. De cette façon, vous n'avez aucune entité inutilisée qui n'est pas surveillée ou gérée activement. Toutefois, vous devez d'abord supprimer tous les espaces clés multirégionaux du compte Régions AWS avant de pouvoir supprimer manuellement le rôle lié à un service.

Nettoyage d'un rôle lié à un service

Avant de pouvoir utiliser IAM pour supprimer un rôle lié à un service, vous devez d'abord supprimer les espaces clés et les tables multirégionaux utilisés par le rôle.

Note

Si le service Amazon Keyspaces utilise le rôle lorsque vous essayez de supprimer les ressources, la suppression risque d'échouer. Si cela se produit, patientez quelques minutes et réessayez.

Pour supprimer les ressources Amazon Keyspaces utilisées par `AWSServiceRoleForAmazonKeyspacesReplication` (console)

1. Connectez-vous à la Console de gestion AWS console Amazon Keyspaces et ouvrez-la à l'adresse. <https://console.aws.amazon.com/keyspaces/home>
2. Choisissez Keyspaces dans le panneau de gauche.
3. Sélectionnez tous les espaces clés multirégionaux dans la liste.
4. Choisissez Supprimer, confirmez la suppression, puis sélectionnez Supprimer les espaces clés.

Vous pouvez également supprimer des espaces clés multirégionaux par programmation à l'aide de l'une des méthodes suivantes.

- L'instruction Cassandra Query Language (CQL). [???](#)
- L'opération [delete-keyspace](#) de la CLI. AWS
- [DeleteKeyspace](#)Fonctionnement de l'API Amazon Keyspaces.

Suppression manuelle du rôle lié au service

Utilisez la console IAM, le AWS CLI, ou l' AWS API pour supprimer le rôle lié au `AWSServiceRoleForAmazonKeyspacesReplication` service. Pour plus d'informations, consultez la section [Suppression d'un rôle lié à un service](#) dans le Guide de l'utilisateur IAM.

Régions prises en charge pour les rôles liés au service Amazon Keyspaces

Amazon Keyspaces ne prend pas en charge l'utilisation de rôles liés à un service dans toutes les régions où le service est disponible. Vous pouvez utiliser le `AWSServiceRoleForAmazonKeyspacesReplication` rôle dans les régions suivantes.

Nom de la région	Identité de la région	Support sur Amazon Keyspaces
USA Est (Virginie du Nord)	us-east-1	Oui
USA Est (Ohio)	us-east-2	Oui
USA Ouest (Californie du Nord)	us-west-1	Oui

Nom de la région	Identité de la région	Support sur Amazon Keyspaces
USA Ouest (Oregon)	us-west-2	Oui
Asie-Pacifique (Mumbai)	ap-south-1	Oui
Asie-Pacifique (Osaka)	ap-northeast-3	Oui
Asie-Pacifique (Séoul)	ap-northeast-2	Oui
Asie-Pacifique (Singapour)	ap-southeast-1	Oui
Asie-Pacifique (Sydney)	ap-southeast-2	Oui
Asie-Pacifique (Tokyo)	ap-northeast-1	Oui
Canada (Centre)	ca-central-1	Oui
Europe (Francfort)	eu-central-1	Oui
Europe (Irlande)	eu-west-1	Oui
Europe (Londres)	eu-west-2	Oui
Europe (Paris)	eu-west-3	Oui
Afrique (Le Cap)	af-south-1	Oui
Amérique du Sud (São Paulo)	sa-east-1	Oui
AWS GovCloud (US-East)	us-gov-east-1	Non
AWS GovCloud (US-West)	us-gov-west-1	Non

Utilisation de rôles pour les flux CDC d'Amazon Keyspaces

[Amazon Keyspaces \(pour Apache Cassandra\) utilise des rôles liés à un service Gestion des identités et des accès AWS \(IAM\)](#). Un rôle lié à un service est un type unique de rôle IAM directement lié à Amazon Keyspaces. Service-linked les rôles sont prédéfinis par Amazon Keyspaces et incluent toutes les autorisations dont le service a besoin pour appeler d'autres AWS services en votre nom.

Un rôle lié à un service facilite la configuration d'Amazon Keyspaces, car vous n'avez pas à ajouter manuellement les autorisations nécessaires. Amazon Keyspaces définit les autorisations associées à ses rôles liés aux services et, sauf indication contraire, seul Amazon Keyspaces peut assumer ses rôles. Les autorisations définies comprennent la politique de confiance et la politique d'autorisation. De plus, cette politique d'autorisation ne peut pas être attachée à une autre entité IAM.

Vous ne pouvez pas supprimer le rôle lié à un service.

Service-linked autorisations de rôle pour Amazon Keyspaces

Amazon Keyspaces utilise le rôle lié au service nommé pour permettre aux flux `AWSServiceRoleForAmazonKeyspacesCDC` Amazon Keyspaces CDC de publier des CloudWatch statistiques sur votre compte en votre nom.

Le rôle `AWSServiceRoleForAmazonKeyspacesCDC` lié à un service fait confiance au service suivant pour assumer le rôle :

- `cassandra-streams.amazonaws.com`

La politique d'autorisation des rôles nommée [KeyspacesCDCServiceRolePolicy](#) permet à Amazon Keyspaces d'effectuer l'action suivante sur les ressources de l' CloudWatch espace de noms : AWS/Cassandra

- Action : `cloudwatch:PutMetricData` sur *

`AWSServiceRoleForAmazonKeyspacesCDC` Fournit les autorisations : Action : `cloudwatch:PutMetricData` sur toutes les ressources répondant à la condition suivante : `"cloudwatch:namespace": "AWS/Cassandra"`.

Pour plus d'informations sur `KeyspacesCDCServiceRolePolicy`, voir [the section called "KeyspacesCDCServiceRolePolicy"](#).

Pour activer les flux CDC pour une table, qui crée automatiquement le rôle lié à un service `AWSServiceRoleForAmazonKeyspacesCDC`, le principal IAM a besoin des autorisations suivantes.

```
{
  "Sid": "KeyspacesCDCServiceLinkedRole",
  "Effect": "Allow",
  "Action": "iam:CreateServiceLinkedRole",
```

```
"Resource": "arn:aws:iam::*:role/aws-service-role/cassandra-streams.amazonaws.com/AWSServiceRoleForAmazonKeyspacesCDC",
"Condition": {
  "StringLike": {
    "iam:AWSServiceName": "cassandra-streams.amazonaws.com"
  }
}
```

Les autorisations permettant de créer le rôle lié au service `AWSServiceRoleForAmazonKeyspacesCDC` sont incluses dans la politique `AmazonKeyspacesFullAccess` gérée. Pour de plus amples informations, veuillez consulter [the section called “AmazonKeyspacesFullAccess”](#).

Création d'un rôle lié à un service pour Amazon Keyspaces

Il n'est pas nécessaire de créer manuellement un rôle lié à un service pour les flux CDC d'Amazon Keyspaces. Lorsque vous activez les flux CDC d'Amazon Keyspaces sur une table avec le CQL Console de gestion AWS, le ou l'API AWS CLI AWS, Amazon Keyspaces crée le rôle lié au service pour vous.

Si vous supprimez ce rôle lié à un service et que vous avez ensuite besoin de le recréer, vous pouvez utiliser la même procédure pour recréer le rôle dans votre compte. Lorsque vous activez les flux CDC d'Amazon Keyspaces pour une table, Amazon Keyspaces crée à nouveau le rôle lié à un service pour vous.

Modification d'un rôle lié à un service pour Amazon Keyspaces

Amazon Keyspaces ne vous permet pas de modifier le rôle lié au `AWSServiceRoleForAmazonKeyspacesCDC` service. Après avoir créé un rôle lié à un service, vous ne pouvez pas changer le nom du rôle, car plusieurs entités peuvent faire référence à ce rôle. Néanmoins, vous pouvez modifier la description du rôle à l'aide d'IAM. Pour plus d'informations, consultez [Modification d'un rôle lié à un service](#) dans le Guide de l'utilisateur IAM.

Régions prises en charge pour les rôles liés au service Amazon Keyspaces

Amazon Keyspaces prend en charge l'utilisation de rôles liés à un service dans toutes les régions où le service est disponible. Pour plus d'informations, consultez [AWS Régions et points de terminaison](#).

Validation de conformité pour Amazon Keyspaces (pour Apache Cassandra)

Third-party les auditeurs évaluent la sécurité et la conformité d'Amazon Keyspaces (pour Apache Cassandra) dans le cadre de multiples programmes de AWS conformité. Il s'agit des licences suivantes :

- ISO/IEC 27001:2013, 27017:2015, 27018:2019 et 9001:2015. ISO/IEC Pour plus d'informations, consultez les certifications et services [AWS ISO et CSA STAR](#).
- System and Organization Controls (SOC)
- Payment Card Industry (PCI)
- Programme fédéral de gestion des risques et des autorisations (FedRAMP) Élevé
- Health Insurance Portability and Accountability Act (HIPAA)

Notre nouvelle expérience AWS d'inscription n'est pas conçue pour les charges de travail réglementées. Si vous utilisez notre nouvelle interface d'AWS inscription, mais que vous souhaitez l'utiliser AWS pour des charges de travail réglementées, vous pouvez vous [inscrire à des fonctionnalités AWS \(avancées\)](#) ou [activer des fonctionnalités avancées](#) pour votre AWS environnement.

Pour savoir si un Service AWS inscrit dans le cadre de programmes de conformité spécifiques, consultez [Services AWS la section Étendue par programme de conformité](#) et choisissez le programme de conformité qui vous intéresse. Pour des informations générales, voir Programmes de [AWS conformité Programmes](#) .

Vous pouvez télécharger des rapports d'audit tiers à l'aide de AWS Artifact. Pour plus d'informations, voir [Téléchargement de rapports dans AWS Artifact](#) .

Votre responsabilité en matière de conformité lors de l'utilisation Services AWS dépend de la sensibilité de vos données, des objectifs de conformité de votre entreprise et des lois et réglementations applicables. Pour plus d'informations sur votre responsabilité en matière de conformité lors de l'utilisation Services AWS, consultez [AWS la documentation de sécurité](#).

Résilience et reprise après sinistre dans Amazon Keyspaces

L'infrastructure AWS mondiale est construite autour Régions AWS de zones de disponibilité. Régions AWS fournissent plusieurs zones de disponibilité physiquement séparées et isolées,

connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone de disponibilité à l'autre sans interruption. Les zones de disponibilité sont plus hautement disponibles, tolérantes aux pannes et évolutives que les infrastructures traditionnelles à un ou plusieurs centres de données.

Amazon Keyspaces réplique automatiquement les données à trois reprises dans plusieurs zones de AWS disponibilité au sein d'une même zone Région AWS pour garantir durabilité et haute disponibilité.

Pour plus d'informations sur les zones de disponibilité Régions AWS et les zones de disponibilité, consultez la section [Infrastructure AWS globale](#).

Outre l'infrastructure AWS mondiale, Amazon Keyspaces propose plusieurs fonctionnalités pour répondre à vos besoins en matière de résilience et de sauvegarde des données.

Réplication multirégionale

Amazon Keyspaces propose une réplication multirégionale si vous devez répliquer vos données ou vos applications sur de plus grandes distances géographiques. Vous pouvez reproduire vos tables Amazon Keyspaces sur les Régions AWS différentes tables de votre choix. Pour de plus amples informations, veuillez consulter [the section called “Multi-Region réplication”](#).

Point-in-time rétablissement (PITR)

PITR aide à protéger vos tables Amazon Keyspaces contre les opérations d'écriture ou de suppression accidentelles en vous fournissant des sauvegardes continues des données de vos tables. Pour plus d'informations, consultez la section [Point-in-timeRecovery for Amazon Keyspaces](#).

Sécurité de l'infrastructure dans Amazon Keyspaces

En tant que service géré, Amazon Keyspaces (pour Apache Cassandra) est protégé par la sécurité du réseau AWS mondial. Pour plus d'informations sur les services AWS de sécurité et sur la manière dont AWS l'infrastructure est protégée, consultez la section [Sécurité du AWS cloud](#). Pour concevoir votre AWS environnement en utilisant les meilleures pratiques en matière de sécurité de l'infrastructure, consultez la section [Protection de l'infrastructure](#) dans le cadre AWS bien architecturé du pilier de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder à Amazon Keyspaces via le réseau. Les clients doivent prendre en charge les éléments suivants :

- Protocole TLS (Transport Layer Security). Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Des suites de chiffrement dotées d'un secret de transmission parfait (PFS), telles que DHE (Ephemeral) ou ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Diffie-Hellman La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

Amazon Keyspaces prend en charge deux méthodes pour authentifier les demandes des clients. La première méthode utilise des informations d'identification spécifiques au service, qui sont des informations d'identification basées sur un mot de passe générées pour un utilisateur IAM spécifique. Vous pouvez créer et gérer le mot de passe à l'aide de la console IAM, de l' AWS CLI API ou de l' AWS API. Pour plus d'informations, consultez [Utilisation d'IAM avec Amazon Keyspaces](#).

La seconde méthode utilise un plugin d'authentification pour le pilote DataStax Java open source pour Cassandra. Ce plugin permet aux [utilisateurs, aux rôles et aux identités fédérées IAM](#) d'ajouter des informations d'authentification aux demandes d'API Amazon Keyspaces (pour Apache Cassandra) à l'aide du processus [AWS Signature Version 4 \(SigV4\)](#). Pour de plus amples informations, veuillez consulter [the section called “Création d'informations d'identification IAM pour l' AWS authentication”](#).

Vous pouvez appeler ces opérations d'API depuis n'importe quel emplacement réseau, mais Amazon Keyspaces prend en charge les politiques d'accès basées sur les ressources, qui peuvent inclure des restrictions basées sur l'adresse IP source. Vous pouvez également utiliser les politiques d'Amazon Keyspaces pour contrôler l'accès depuis des points de terminaison Amazon Virtual Private Cloud (Amazon VPC) ou des VPC spécifiques. En fait, cela isole l'accès réseau à une ressource Amazon Keyspaces donnée uniquement du VPC spécifique au sein du réseau. AWS

Vous pouvez utiliser un point de terminaison VPC d'interface pour empêcher le trafic entre votre Amazon VPC et Amazon Keyspaces de quitter le réseau Amazon. Les points de terminaison VPC d'interface sont alimentés par AWS PrivateLink une AWS technologie qui permet une communication privée entre les AWS services à l'aide d'une interface réseau élastique avec des adresses IP privées dans votre Amazon VPC. Pour de plus amples informations, veuillez consulter [the section called “Utilisation des points de terminaison de VPC d'interface ”](#).

Utilisation d'Amazon Keyspaces avec des points de terminaison VPC d'interface

Les points de terminaison VPC d'interface permettent une communication privée entre votre cloud privé virtuel (VPC) exécuté dans Amazon VPC et Amazon Keyspaces. Les points de terminaison VPC d'interface sont alimentés par AWS PrivateLink un AWS service qui permet une communication privée entre les VPC et les services. AWS

AWS PrivateLink permet cela en utilisant une interface réseau élastique avec des adresses IP privées dans votre VPC afin que le trafic réseau ne quitte pas le réseau Amazon. Les points de terminaison de VPC d'interface n'ont pas besoin d'une passerelle Internet, d'un périphérique NAT, d'une connexion VPN ou d'une connexion Direct Connect . Pour plus d'informations, consultez [Amazon Virtual Private Cloud](#) and [Interface VPC endpoints](#) ().AWS PrivateLink

Rubriques

- [Utilisation des points de terminaison VPC d'interface pour Amazon Keyspaces](#)
- [Remplissage des entrées de system.peers table avec les informations de point de terminaison VPC de l'interface](#)
- [Contrôle de l'accès aux points de terminaison VPC de l'interface pour Amazon Keyspaces](#)
- [Disponibilité](#)
- [Politiques relatives aux terminaux VPC et restauration instantanée \(PITR\) d'Amazon Keyspaces](#)
- [Erreurs et avertissements courants](#)

Utilisation des points de terminaison VPC d'interface pour Amazon Keyspaces

Vous pouvez créer un point de terminaison VPC d'interface afin que le trafic entre Amazon Keyspaces et vos ressources Amazon VPC commence à passer par le point de terminaison VPC d'interface. Pour commencer, suivez les étapes pour [créer un point de terminaison d'interface](#). Ensuite, modifiez le groupe de sécurité associé au point de terminaison que vous avez créé à l'étape précédente et configurez une règle entrante pour le port 9142. Pour plus d'informations, consultez la section [Ajout, suppression et mise à jour de règles](#).

Pour un didacticiel étape par étape sur la configuration d'une connexion à Amazon Keyspaces via un point de terminaison VPC, consultez. [the section called “Connexion aux points de terminaison VPC”](#) Pour savoir comment configurer l'accès entre comptes pour les ressources Amazon Keyspaces

séparément des applications d'un VPC, consultez. Comptes AWS [the section called “Configuration de l'accès entre comptes”](#)

Remplissage des entrées de **system.peers** table avec les informations de point de terminaison VPC de l'interface

Les pilotes Apache Cassandra utilisent la `system.peers` table pour demander des informations sur les nœuds du cluster. Les pilotes Cassandra utilisent les informations du nœud pour équilibrer la charge des connexions et relancer les opérations. Amazon Keyspaces remplit automatiquement neuf entrées du `system.peers` tableau pour les clients qui se connectent via le point de terminaison public.

Pour fournir aux clients qui se connectent via des points de terminaison VPC d'interface des fonctionnalités similaires, Amazon Keyspaces remplit le `system.peers` tableau de votre compte avec une entrée pour chaque zone de disponibilité dans laquelle un point de terminaison VPC est disponible. Pour rechercher et stocker les points de terminaison VPC d'interface disponibles dans le tableau `system.peers`, Amazon Keyspaces exige que vous accordiez à l'entité IAM utilisée pour se connecter à Amazon Keyspaces les autorisations d'accès lui permettant de demander à votre VPC les informations relatives au point de terminaison et à l'interface réseau.

Important

Le remplissage du `system.peers` tableau avec les points de terminaison VPC de votre interface disponibles améliore l'équilibrage de charge et augmente le débit. read/write Il est recommandé à tous les clients accédant à Amazon Keyspaces via des points de terminaison VPC d'interface et est requis pour Apache Spark.

Pour accorder à l'entité IAM utilisée pour se connecter à Amazon Keyspaces l'autorisation de rechercher les informations nécessaires sur le point de terminaison VPC de l'interface, vous pouvez mettre à jour votre rôle IAM ou votre politique utilisateur existants, ou créer une nouvelle politique IAM comme indiqué dans l'exemple suivant.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ListVPCEndpoints",
      "Effect": "Allow",
```

```
    "Action": [
      "ec2:DescribeNetworkInterfaces",
      "ec2:DescribeVpcEndpoints"
    ],
    "Resource": "*"
  }
]
```

Note

Les politiques gérées `AmazonKeyspacesFullAccess` incluent `AmazonKeyspacesReadOnlyAccess_v2` les autorisations requises pour permettre à Amazon Keyspaces d'accéder à l'instance Amazon EC2 afin de lire les informations sur les points de terminaison VPC d'interface disponibles.

Pour vérifier que la politique a été correctement configurée, interrogez le `system.peers` tableau pour consulter les informations réseau. Si le `system.peers` tableau est vide, cela peut indiquer que la politique n'a pas été configurée correctement ou que vous avez dépassé le quota de demandes `DescribeNetworkInterfaces` et les actions `DescribeVPCEndpoints` d'API. `DescribeVPCEndpoints` entre dans `Describe*` cette catégorie et est considérée comme une action non mutante. `DescribeNetworkInterfaces` appartient au sous-ensemble des actions non mutantes non filtrées et non paginées, et différents quotas s'appliquent. Pour plus d'informations, consultez la section [Taille des compartiments de jetons de demande et taux de recharge](#) dans le manuel Amazon EC2 API Reference.

Si vous voyez un tableau vide, réessayez quelques minutes plus tard pour exclure tout problème de quota de taux de demandes. Pour vérifier que vous avez correctement configuré les points de terminaison VPC, consultez [the section called “Erreurs de connexion au point de terminaison VPC”](#). Si votre requête renvoie les résultats du tableau, cela signifie que votre politique a été correctement configurée.

Contrôle de l'accès aux points de terminaison VPC de l'interface pour Amazon Keyspaces

Avec les politiques de point de terminaison VPC, vous pouvez contrôler l'accès aux ressources de deux manières :

- Politique IAM : vous pouvez contrôler les demandes, les utilisateurs ou les groupes autorisés à accéder à Amazon Keyspaces via un point de terminaison VPC spécifique. Pour ce faire, utilisez une [clé de condition](#) dans la stratégie attachée à un utilisateur, un groupe ou un rôle IAM.
- Politique VPC : vous pouvez contrôler quels points de terminaison VPC ont accès à vos ressources Amazon Keyspaces en leur associant des politiques. Pour restreindre l'accès à un keyspace ou à une table spécifique afin d'autoriser uniquement le trafic passant par un point de terminaison VPC spécifique, modifiez la stratégie IAM existante qui restreint l'accès aux ressources et ajoutez ce point de terminaison VPC.

Vous trouverez ci-dessous des exemples de politiques relatives aux terminaux pour accéder aux ressources Amazon Keyspaces.

- Exemple de politique IAM : limitez tout accès à une table Amazon Keyspaces spécifique, sauf si le trafic provient du point de terminaison VPC spécifié. Cet exemple de politique peut être associé à un utilisateur, un rôle ou un groupe IAM. Il restreint l'accès à une table Amazon Keyspaces spécifiée, sauf si le trafic entrant provient d'un point de terminaison VPC spécifié.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UserOrRolePolicyToDenyAccess",
      "Action": "cassandra:*",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/mytable",
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
      ],
      "Condition": { "StringNotEquals" : { "aws:sourceVpce": "vpce-abc123" } }
    }
  ]
}
```

Note

Pour restreindre l'accès à une table spécifique, vous devez également inclure l'accès aux tables système. Les tables système sont en lecture seule.

- Exemple de politique VPC : Read-only accès — Cet exemple de politique peut être attaché à un point de terminaison VPC. (Pour plus d'informations, consultez la section [Contrôle de l'accès aux ressources Amazon VPC](#)). Il limite les actions à l'accès en lecture seule aux ressources Amazon Keyspaces via le point de terminaison VPC auquel il est connecté.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadOnly",
      "Principal": "*",
      "Action": [
        "cassandra:Select"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

- Exemple de politique VPC : restreindre l'accès à une table Amazon Keyspaces spécifique — Cet exemple de politique peut être attaché à un point de terminaison VPC. Il limite l'accès à une table spécifique via le point de terminaison VPC auquel il est attaché.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "RestrictAccessToTable",
      "Principal": "*",
      "Action": "cassandra:*",
      "Effect": "Allow",
      "Resource": [
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/mytable",
        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
      ]
    }
  ]
}
```


 Note

Pour restreindre l'accès à une table spécifique, vous devez également inclure l'accès aux tables système. Les tables système sont en lecture seule.

Disponibilité

Amazon Keyspaces prend en charge l'utilisation des points de terminaison VPC d'interface partout Régions AWS où le service est disponible. Pour de plus amples informations, veuillez consulter [???](#).

Politiques relatives aux terminaux VPC et restauration instantanée (PITR) d'Amazon Keyspaces

Si vous utilisez des politiques IAM avec des [clés de condition](#) pour restreindre le trafic entrant, l'opération de restauration des tables risque d'échouer. Par exemple, si vous limitez le trafic source à des points de terminaison VPC spécifiques à l'aide de clés de `aws:SourceVpce` condition, l'opération de restauration de table échoue. Pour autoriser Amazon Keyspaces à effectuer une opération de restauration pour le compte de votre mandant, vous devez ajouter une clé de `aws:ViaAWSService` condition à votre politique IAM. La clé de `aws:ViaAWSService` condition permet l'accès lorsqu'un AWS service fait une demande en utilisant les informations d'identification du principal. Pour plus d'informations, voir [Éléments de politique IAM JSON : clé de condition](#) dans le guide de l'utilisateur IAM. La politique suivante en est un exemple.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "CassandraAccessForVPCE",
      "Effect": "Allow",
      "Action": "cassandra:*",
      "Resource": "*",
      "Condition": {
        "Bool": {
          "aws:ViaAWSService": "false"
        },
        "StringEquals": {
          "aws:SourceVpce": [
            "vpce-12345678901234567"
          ]
        }
      }
    }
  ]
}
```

```

    ]
  }
}
},
{
  "Sid": "CassandraAccessForAwsService",
  "Effect": "Allow",
  "Action": "cassandra:*",
  "Resource": "*",
  "Condition": {
    "Bool": {
      "aws:ViaAWSService": "true"
    }
  }
}
]
}

```

Erreurs et avertissements courants

Si vous utilisez Amazon Virtual Private Cloud et que vous vous connectez à Amazon Keyspaces, l'avertissement suivant peut s'afficher.

```

Control node cassandra.us-east-1.amazonaws.com/1.111.111.111:9142 has an entry
for itself in system.peers: this entry will be ignored. This is likely due to a
misconfiguration;
please verify your rpc_address configuration in cassandra.yaml on all nodes in your
cluster.

```

Cet avertissement se produit car le `system.peers` tableau contient des entrées pour tous les points de terminaison Amazon VPC qu'Amazon Keyspaces est autorisé à consulter, y compris le point de terminaison Amazon VPC via lequel vous êtes connecté. Vous pouvez ignorer cet avertissement en toute sécurité.

Pour les autres erreurs, consultez [the section called “Erreurs de connexion au point de terminaison VPC”](#).

Utilisation des flux CDC d'Amazon Keyspaces avec des points de terminaison VPC d'interface

Les points de terminaison VPC d'interface permettent une communication privée entre votre cloud privé virtuel (VPC) exécuté dans Amazon VPC et Amazon Keyspaces. Les points de terminaison VPC d'interface sont alimentés par AWS PrivateLink un AWS service qui permet une communication privée entre les VPC et les services. AWS

AWS PrivateLink permet cela en utilisant une interface réseau élastique avec des adresses IP privées dans votre VPC afin que le trafic réseau ne quitte pas le réseau Amazon. Les points de terminaison de VPC d'interface n'ont pas besoin d'une passerelle Internet, d'un périphérique NAT, d'une connexion VPN ou d'une connexion Direct Connect . Pour plus d'informations, consultez [Amazon Virtual Private Cloud](#) and [Interface VPC endpoints](#) ().AWS PrivateLink

Rubriques

- [Utilisation des points de terminaison VPC de l'interface pour les flux CDC d'Amazon Keyspaces](#)
- [Points de terminaison VPC de l'interface de diffusion \(CDC\) d'Amazon Keyspaces](#)
- [Création d'un point de terminaison VPC avec l'interface de flux CDC d'Amazon Keyspaces](#)
- [Mettre à jour un point de terminaison \(VPC\) de l'interface de flux CDC d'Amazon Keyspaces](#)
- [Répertorier les flux à l'aide d'une interface de flux CDC Amazon Keyspaces \(point de terminaison VPC\)](#)
- [Création d'une politique pour un point de terminaison \(VPC\) de l'interface de flux CDC d'Amazon Keyspaces](#)

Utilisation des points de terminaison VPC de l'interface pour les flux CDC d'Amazon Keyspaces

Vous pouvez utiliser un point de terminaison VPC d'interface afin que le trafic entre les flux CDC d'Amazon Keyspaces et vos ressources Amazon VPC commence à passer par le point de terminaison VPC de l'interface. Vous pouvez utiliser les politiques de point de terminaison VPC pour restreindre l'accès à vos flux CDC.

Pour plus d'informations sur les flux CDC d'Amazon Keyspaces, consultez. [the section called “Travailler avec les flux CDC”](#)

Points de terminaison VPC de l'interface de diffusion (CDC) d'Amazon Keyspaces

Lorsque vous créez un point de terminaison d'interface, Amazon Keyspaces CDC streams génère deux types de noms DNS spécifiques au point de terminaison pour le flux : régional et zonal.

Régional

Le nom DNS régional inclut les informations suivantes :

- un identifiant de point de terminaison Amazon VPC unique
- un identifiant de service
- le Région AWS
- le `vpce.amazonaws.com` suffixe

Pour un point de terminaison Amazon VPC avec l'ID `vpce-1a2b3c4d`, le nom DNS généré peut ressembler à l'exemple suivant : `vpce-1a2b3c4d-5e6f.cassandra-streams.us-east-1.vpce.amazonaws.com`

Zonal

Le nom DNS de zone inclut la [zone de disponibilité](#) en plus des informations contenues dans le nom DNS régional. Le nom DNS généré pour le point de terminaison Amazon VPC avec l'ID `vpce-1a2b3c4d` ressemble à celui de l'exemple suivant. Notez que la zone de disponibilité Région AWS est désormais incluse : `vpce-1a2b3c4d-5e6f-us-east-1a.cassandra-streams.us-east-1.vpce.amazonaws.com`

Vous pouvez utiliser cette option si votre architecture isole des zones de disponibilité. Par exemple, vous pouvez l'utiliser pour contenir les pannes ou réduire les coûts de transfert de données Régionaux.

Note

Pour obtenir une fiabilité optimale, nous vous recommandons de déployer votre service dans un minimum de trois zones de disponibilité.

Création d'un point de terminaison VPC avec l'interface de flux CDC d'Amazon Keyspaces

Vous pouvez utiliser le AWS CLI ou le AWS SDK pour accéder aux opérations de l'API Amazon Keyspaces CDC Streams via les points de terminaison de l'interface Amazon Keyspaces CDC

Streams. Pour obtenir la liste complète de toutes les opérations d'API disponibles, consultez le manuel de référence des [API Amazon Keyspaces Streams](#).

Pour plus d'informations sur la création de points de terminaison VPC, consultez la section [Création d'un point de terminaison d'interface](#) dans le guide de l'utilisateur Amazon VPC.

Pour créer un point de terminaison VPC, vous pouvez utiliser la syntaxe de l'exemple suivant.

```
aws ec2 create-vpc-endpoint \  
  --region us-east-1 \  
  --service-name api.aws.us-east-1.cassandra-streams \  
  --vpc-id client-vpc-id \  
  --subnet-ids client-subnet-id \  
  --vpc-endpoint-type Interface \  
  --security-group-ids client-sg-id
```

Mettre à jour un point de terminaison (VPC) de l'interface de flux CDC d'Amazon Keyspaces

Pour mettre à jour un point de terminaison VPC, vous pouvez utiliser la syntaxe de l'exemple suivant.

```
aws ec2 modify-vpc-endpoint \  
  --region us-east-1 \  
  --vpc-endpoint-id client-vpc-id \  
  --policy-document policy-document \ #example optional parameter  
  --add-security-group-ids security-group-ids \ #example optional parameter
```

Répertorier les flux à l'aide d'une interface de flux CDC Amazon Keyspaces (point de terminaison VPC)

Pour répertorier les flux qui utilisent un point de terminaison VPC, vous pouvez utiliser la syntaxe de l'exemple suivant. Assurez-vous de remplacer la région et le nom DNS de l'ID de point de terminaison du VPC par vos propres informations.

```
aws keyspacesstreams \  
  --endpoint https://vpce-1a2b3c4d-5e6f.cassandra-streams.us-east-1.vpce.amazonaws.com  
 \  
  --region us-east-1 \  
  list-streams
```

Création d'une politique pour un point de terminaison (VPC) de l'interface de flux CDC d'Amazon Keyspaces

Vous pouvez associer une politique de point de terminaison à votre point de terminaison Amazon VPC qui contrôle l'accès aux flux CDC d'Amazon Keyspaces. La politique spécifie les informations suivantes :

- Le principal Gestion des identités et des accès AWS (IAM) qui peut effectuer des actions
- Les actions qui peuvent être effectuées.
- Les ressources sur lesquelles les actions peuvent être exécutées.

Pour restreindre l'accès à des flux CDC Amazon Keyspaces spécifiques afin de n'autoriser l'accès qu'à des AWS services spécifiques dans votre Amazon VPC, vous pouvez utiliser l'exemple suivant.

La politique de flux suivante accorde l'accès à n'importe quel principal IAM pour les actions `cassandra:GetStream` et `cassandra:GetRecords` pour le flux spécifié `2025-02-20T11:22:33.444` attaché à la ressource `/keyspace/mykeyspace/table/mytable/` appartenant au compte `123456788901`. Pour utiliser cette politique de point de terminaison, assurez-vous de remplacer la région, l'ID de compte et la ressource par le libellé du flux.

```
{
  "Version": "2012-10-17",
  "Id": "Policy1216114807515",
  "Statement": [
    { "Sid": "Access-to-specific-stream-only",
      "Principal": "*",
      "Action": [
        "cassandra:GetStream",
        "cassandra:GetRecords"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/
table/mytable/stream/2025-02-20T11:22:33.444"]
    }
  ]
}
```

 Note

Amazon Keyspaces ne prend pas en charge les points de terminaison Gateway pour les flux CDC.

Analyse de configuration et de vulnérabilité pour Amazon Keyspaces

AWS gère les tâches de sécurité de base telles que l'application de correctifs au système d'exploitation client (OS) et aux bases de données, la configuration du pare-feu et la reprise après sinistre. Ces procédures ont été vérifiées et certifiées par les tiers appropriés. Pour plus de détails, consultez les ressources suivantes :

- Modèle de [responsabilité partagée](#) [Modèle](#)
- [Amazon Web Services : présentation des procédures de sécurité](#) (livre blanc)

Bonnes pratiques en matière de sécurité pour Amazon Keyspaces

Amazon Keyspaces (pour Apache Cassandra) fournit un certain nombre de fonctionnalités de sécurité à prendre en compte lors de l'élaboration et de la mise en œuvre de vos propres politiques de sécurité. Les bonnes pratiques suivantes doivent être considérées comme des instructions générales et ne représentent pas une solution de sécurité complète. Étant donné que ces bonnes pratiques peuvent ne pas être appropriées ou suffisantes pour votre environnement, considérez-les comme des remarques utiles plutôt que comme des recommandations.

Rubriques

- [Bonnes pratiques de sécurité préventive pour Amazon Keyspaces](#)
- [Les meilleures pratiques de Detective en matière de sécurité pour Amazon Keyspaces](#)

Bonnes pratiques de sécurité préventive pour Amazon Keyspaces

Les bonnes pratiques de sécurité suivantes sont considérées comme préventives car elles peuvent vous aider à anticiper et à prévenir les incidents de sécurité dans Amazon Keyspaces.

Utiliser le chiffrement au repos

[Amazon Keyspaces chiffre au repos toutes les données utilisateur stockées dans des tables à l'aide des clés de chiffrement stockées dans AWS Key Management Service \(\).AWS KMS](#) Cela fournit une couche supplémentaire de protection des données en sécurisant vos données contre tout accès non autorisé au stockage sous-jacent.

Par défaut, Amazon Keyspaces utilise une Clé détenue par AWS pour chiffrer toutes vos tables. Si cette clé n'existe pas, elle a été créée pour vous. Les clés par défaut du service ne peuvent pas être désactivées.

Vous pouvez également utiliser une [clé gérée par le client](#) pour le chiffrement au repos. Pour plus d'informations, consultez [Amazon Keyspaces Encryption at Rest](#).

Utiliser les rôles IAM pour authentifier l'accès à Amazon Keyspaces

Pour que les utilisateurs, les applications et les autres AWS services puissent accéder à Amazon Keyspaces, ils doivent inclure des AWS informations d'identification valides dans leurs demandes d' AWS API. Vous ne devez pas stocker les AWS informations d'identification directement dans l'application ou l'instance EC2. Il s'agit d'informations d'identification à long terme qui ne font pas l'objet d'une rotation automatique, et dont la compromission pourrait avoir un impact considérable sur l'activité. Un rôle IAM vous permet d'obtenir des clés d'accès temporaires qui peuvent être utilisées pour accéder aux ressources et services AWS .

Pour en savoir plus, consultez [Rôles IAM](#).

Utiliser les politiques IAM pour l'autorisation de base Amazon Keyspaces

Lorsque vous accordez des autorisations, vous décidez qui les obtient, pour quelles API Amazon Keyspaces ils obtiennent des autorisations et quelles actions spécifiques vous souhaitez autoriser sur ces ressources. La mise en œuvre du principe du moindre privilège est essentielle pour réduire les risques de sécurité et l'impact pouvant résulter d'erreurs ou d'intentions malveillantes.

Associez des politiques d'autorisation aux identités IAM (c'est-à-dire aux utilisateurs, aux groupes et aux rôles) et accordez ainsi des autorisations pour effectuer des opérations sur les ressources Amazon Keyspaces.

Pour ce faire, utilisez les ressources suivantes :

- [AWS politiques gérées \(prédéfinies\)](#)
- [Politiques gérées par le client](#)

Utiliser des conditions de politique IAM pour un contrôle d'accès précis

Lorsque vous accordez des autorisations dans Amazon Keyspaces, vous pouvez spécifier les conditions qui déterminent la manière dont une politique d'autorisation prend effet. La mise en œuvre du principe du moindre privilège est essentielle pour réduire les risques de sécurité et l'impact pouvant résulter d'erreurs ou d'intentions malveillantes.

Vous pouvez spécifier des conditions lors de l'octroi d'autorisations à l'aide d'une politique IAM. Par exemple, vous pouvez effectuer les opérations suivantes :

- Accordez des autorisations pour permettre aux utilisateurs d'accéder en lecture seule à des espaces clés ou à des tables spécifiques.
- Accordez des autorisations permettant à un utilisateur d'accéder en écriture à une certaine table, en fonction de son identité.

Pour plus d'informations, consultez la section [Exemples Identity-Based de politiques](#).

Envisager un chiffrement côté client

Si vous stockez des données sensibles ou confidentielles dans Amazon Keyspaces, vous souhaitez peut-être chiffrer ces données le plus près possible de leur origine afin de les protéger tout au long de leur cycle de vie. Le chiffrement de vos données sensibles en transit et au repos contribue à garantir que vos données en texte brut ne sont pas accessibles à un tiers.

Les meilleures pratiques de Detective en matière de sécurité pour Amazon Keyspaces

Les meilleures pratiques de sécurité suivantes sont considérées comme détectives car elles peuvent vous aider à détecter les faiblesses et les incidents de sécurité potentiels.

AWS CloudTrail À utiliser pour surveiller l'utilisation des AWS KMS clés AWS Key Management Service (AWS KMS)

Si vous utilisez une [AWS KMS clé gérée par le client](#) pour le chiffrement au repos, l'utilisation de cette clé est connectée AWS CloudTrail. CloudTrail fournit une visibilité sur l'activité des utilisateurs en enregistrant les actions effectuées sur votre compte. CloudTrail enregistre des informations importantes sur chaque action, notamment l'auteur de la demande, les services utilisés, les actions effectuées, les paramètres des actions et les éléments de réponse renvoyés par le AWS service. Ces informations vous aident à suivre les modifications apportées à vos

AWS ressources et à résoudre les problèmes opérationnels. CloudTrail permet de garantir plus facilement le respect des politiques internes et des normes réglementaires.

Vous pouvez l'utiliser CloudTrail pour auditer l'utilisation des clés. CloudTrail crée des fichiers journaux contenant l'historique des appels d' AWS API et des événements associés à votre compte. Ces fichiers journaux incluent toutes les demandes d' AWS KMS API effectuées à l'aide de la console, AWS des SDK et des outils de ligne de commande, en plus de celles effectuées via les AWS services intégrés. Vous pouvez utiliser ces fichiers journaux pour obtenir des informations sur le moment où la AWS KMS clé a été utilisée, l'opération demandée, l'identité du demandeur, l'adresse IP d'origine de la demande, etc. Pour plus d'informations, consultez [Journalisation des appels d'API AWS Key Management Service avec AWS CloudTrail](#) dans le [Guide de l'utilisateur AWS CloudTrail](#).

CloudTrail À utiliser pour surveiller les opérations du langage de définition de données (DDL) Amazon Keyspaces

CloudTrail fournit une visibilité sur l'activité des utilisateurs en enregistrant les actions effectuées sur votre compte. CloudTrail enregistre des informations importantes sur chaque action, notamment l'auteur de la demande, les services utilisés, les actions effectuées, les paramètres des actions et les éléments de réponse renvoyés par le AWS service. Ces informations vous aident à suivre les modifications apportées à vos AWS ressources et à résoudre les problèmes opérationnels. CloudTrail permet de garantir plus facilement le respect des politiques internes et des normes réglementaires.

Toutes les [opérations DDL d'Amazon Keyspaces sont automatiquement](#) enregistrées. CloudTrail Les opérations DDL vous permettent de créer et de gérer les espaces de touches et les tables Amazon Keyspaces.

Lorsqu'une activité a lieu dans Amazon Keyspaces, cette activité est enregistrée dans un CloudTrail événement avec d'autres événements liés au AWS service dans l'historique des événements. Pour plus d'informations, consultez la section [Enregistrement des opérations Amazon Keyspaces à l'aide de](#). AWS CloudTrail Vous pouvez consulter, rechercher et télécharger les événements récents dans votre Compte AWS. Pour plus d'informations, consultez la section [Affichage des événements avec l'historique des CloudTrail événements](#) dans le Guide de AWS CloudTrail l'utilisateur.

[Pour un enregistrement continu des événements survenus dans votre environnement Compte AWS, y compris des événements pour Amazon Keyspaces, créez un historique.](#) Un suivi permet CloudTrail de transférer des fichiers journaux vers un compartiment Amazon Simple

Storage Service (Amazon S3). Par défaut, lorsque vous créez un parcours sur la console, celui-ci s'applique à tous Régions AWS. Le journal d'activité consigne les événements de toutes les régions dans la partition AWS et livre les fichiers journaux dans le compartiment S3 de votre choix. En outre, vous pouvez configurer d'autres AWS services pour analyser plus en détail les données d'événements collectées dans les CloudTrail journaux et agir en conséquence.

Étiquetez vos ressources Amazon Keyspaces à des fins d'identification et d'automatisation

Vous pouvez attribuer des métadonnées à vos AWS ressources sous forme de balises. Chaque balise est une étiquette simple composée d'une clé définie par le client et d'une valeur facultative qui facilite la gestion, la recherche et le filtrage des ressources.

L'étiquetage permet l'implémentation de contrôles groupés. Bien qu'il n'existe pas de types intrinsèques d'étiquettes, celles-ci vous permettent de catégoriser des ressources par objectif, par propriétaire, par environnement ou selon d'autres critères. Voici quelques exemples :

- Accès : utilisé pour contrôler l'accès aux ressources Amazon Keyspaces en fonction de balises. Pour de plus amples informations, veuillez consulter [the section called “Autorisation basée sur les tags Amazon Keyspaces”](#).
- Sécurité — Utilisé pour déterminer les exigences telles que les paramètres de protection des données.
- Confidentialité — Identifiant du niveau spécifique de confidentialité des données pris en charge par une ressource.
- Environnement : utilisé pour différencier les infrastructures de développement, de test et de production.

Pour plus d'informations, consultez les sections [Stratégies de AWS balisage](#) et [Ajout de balises et d'étiquettes aux ressources](#).

Référence du langage CQL pour Amazon Keyspaces (pour Apache Cassandra)

Après vous être connecté à un point de terminaison Amazon Keyspaces, vous utilisez le langage de requête Cassandra (CQL) pour travailler avec votre base de données. CQL est similaire à de nombreux égards au langage SQL (Structured Query Language).

- **Éléments CQL** — Cette section couvre les éléments fondamentaux du CQL pris en charge par Amazon Keyspaces, notamment les identifiants, les constantes, les termes et les types de données. Il explique des concepts tels que les types de chaînes, les types numériques, les types de collection, etc.
- **Langage de définition des données (DDL)** : les instructions DDL sont utilisées pour gérer les structures de données telles que les espaces clés et les tables dans Amazon Keyspaces. Cette section couvre les instructions relatives à la création, à la modification et à la suppression d'espaces clés et de tables, ainsi qu'à la restauration de tables à partir d'une sauvegarde instantanée.
- **Langage de manipulation des données (DML)** : les instructions DML sont utilisées pour gérer les données contenues dans les tables. Cette section couvre les instructions relatives à la sélection, à l'insertion, à la mise à jour et à la suppression de données. Il explique également les fonctionnalités de requête avancées telles que l'utilisation de l'INopérateur, le classement des résultats et la pagination.
- **Built-in fonctions** — Amazon Keyspaces prend en charge une variété de fonctions scalaires intégrées que vous pouvez utiliser dans les instructions CQL. Cette section fournit un aperçu de ces fonctions, y compris des exemples de leur utilisation.

Tout au long de cette rubrique, vous trouverez une syntaxe détaillée, des exemples et les meilleures pratiques pour utiliser efficacement le CQL dans Amazon Keyspaces.

Rubriques

- [Éléments du langage de requête Cassandra \(CQL\) dans Amazon Keyspaces](#)
- [Instructions DDL \(langage de définition des données\) dans Amazon Keyspaces](#)
- [Instructions DML \(langage de manipulation de données\) dans Amazon Keyspaces](#)
- [Built-in fonctions dans Amazon Keyspaces](#)

Éléments du langage de requête Cassandra (CQL) dans Amazon Keyspaces

Découvrez les éléments du langage de requête Cassandra (CQL) pris en charge par Amazon Keyspaces, notamment les identifiants, les constantes, les termes et les types de données.

Rubriques

- [Identifiants](#)
- [Constantes](#)
- [Termes](#)
- [Types de données](#)
- [Encodage JSON des types de données Amazon Keyspaces](#)

Identifiants

Les identificateurs (ou noms) sont utilisés pour identifier les tables, colonnes et autres objets. Un identifiant peut être entre apostrophes ou pas. Les conditions suivantes s'appliquent :

```

identifiant      ::=  unquoted_identifiant | quoted_identifiant
unquoted_identifiant ::=  re('[a-zA-Z][a-zA-Z0-9]*')
quoted_identifiant ::=  "'" (any character where " can appear if doubled)+ "'"
  
```

Constantes

Les constantes suivantes sont définies.

```

constant ::=  string | integer | float | boolean | uuid | blob | NULL
string    ::=  '\' (any character where ' can appear if doubled)+ '\''
            | '$$' (any character other than '$$') '$$'
integer   ::=  re('-?[0-9]+')
float     ::=  re('-?[0-9]+(\.[0-9]*)?([eE][+-]?[0-9+]?)') | NAN | INFINITY
boolean   ::=  TRUE | FALSE
uuid      ::=  hex{8}-hex{4}-hex{4}-hex{4}-hex{12}
hex       ::=  re("[0-9a-fA-F]")
blob      ::=  '0' ('x' | 'X') hex+
  
```

Termes

Un terme désigne le type de valeurs prises en charge. Les termes sont définis par ce qui suit.

```
term          ::= constant | literal | function_call | arithmetic_operation |
               type_hint | bind_marker
literal       ::= collection_literal | tuple_literal
function_call ::= identifiant '(' [ term (',' term)* ] ')'
arithmetic_operation ::= '-' term | term ('+' | '-' | '*' | '/' | '%') term
```

Types de données

Amazon Keyspaces prend en charge les types de données suivants :

Types de chaîne

Type de données	Description
ascii	Représente une chaîne de caractères ASCII.
text	Représente une chaîne UTF-8 codée.
varchar	Représente une chaîne UTF-8 codée (varcharil s'agit d'un alias pour text).

Types numériques

Type de données	Description
bigint	Représente un entier long signé 64 bits.
counter	Représente un compteur d'entier signé 64 bits. Pour plus d'informations, consultez the section called "Compteurs" .
decimal	Représente une décimale de précision variable.
double	Représente une virgule flottante IEEE 754 64 bits.

Type de données	Description
float	Représente une virgule flottante IEEE 754 32 bits.
int	Représente un entier signé 32 bits.
varint	Représente une valeur entière comprise entre +/-10 ³⁸ .

Compteurs

Une colonne `counter` contient un entier signé 64 bits. La valeur du compteur est incrémentée ou décrémentée à l'aide de l'instruction [the section called “UPDATE”](#), et ne peut pas être définie directement. Cela rend les colonnes `counter` utiles pour le suivi des dénombrements. Par exemple, vous pouvez utiliser des compteurs pour suivre le nombre d'entrées dans un fichier journal ou le nombre de fois qu'une publication a été consultée sur un réseau social. Les restrictions ci-après s'appliquent aux colonnes `counter` :

- Une colonne de type `counter` ne peut pas faire partie de la `primary key` d'une table.
- Dans une table qui contient une ou plusieurs colonnes de type `counter`, toutes les colonnes de cette table doivent être de type `counter`.

Dans les cas où la mise à jour d'un compteur échoue (par exemple, en raison d'un délai d'attente ou d'une perte de connexion avec Amazon Keyspaces), le client ne sait pas si la valeur du compteur a été mise à jour. Si la mise à jour est réessayée, la mise à jour de la valeur du compteur peut être appliquée une deuxième fois.

Type de blob

Type de données	Description
blob	Représente les octets arbitraires.

Type Boolean

Type de données	Description
boolean	Représente true ou false.

Time-related types

Type de données	Description
date	Une chaîne au format<yyyy> - <mm> - <dd> .
timestamp	Entier signé de 64 bits représentant la date et l'heure depuis l'époque (1er janvier 1970 à 00:00:00 GMT) en millisecondes.
timeuuid	Représente un UUID version 1 .

Types de collections

Type de données	Description
list	Représente une collection ordonnée d'éléments littéraux.
map	Représente une collection non ordonnée de paires clé-valeur.
set	Représente une collection non ordonnée d'un ou plusieurs éléments littéraux.

Vous déclarez une colonne de collection en utilisant le type de collection suivi d'un autre type de données (par exemple, TEXT ou INT) entre crochets. Vous pouvez créer une colonne avec un SET de TEXT, ou vous pouvez créer une paire clé-valeur TEXT et une MAP paire INT clé-valeur, comme indiqué dans l'exemple suivant.


```
SET <TEXT>  
MAP <TEXT, INT>
```

Une collection non figée vous permet de mettre à jour chaque élément de collection individuel. Client-side les horodatages et les paramètres de durée de vie (TTL) sont enregistrés pour les éléments individuels.

Lorsque vous utilisez le FROZEN mot clé sur un type de collection, les valeurs de la collection sont sérialisées en une seule valeur immuable, et Amazon Keyspaces les traite comme un. BLOB Il s'agit d'une collection surgelée. Une UPDATE déclaration INSERT OR remplace l'ensemble de la collection Frozen. Vous ne pouvez pas mettre à jour des éléments individuels d'une collection figée.

Client-side les paramètres d'horodatage et de durée de vie (TTL) s'appliquent à l'ensemble de la collection figée, et non à des éléments individuels. Frozen les colonnes de collection peuvent faire partie PRIMARY KEY d'un tableau.

Vous pouvez imbriquer des collections congelées. Par exemple, vous pouvez définir un MAP dans un SET si le FROZEN mot clé MAP est utilisé, comme illustré dans l'exemple suivant.

```
SET <FROZEN> <MAP <TEXT, INT>>>
```

Amazon Keyspaces prend en charge l'imbrication d'un maximum de 8 niveaux de collections figées par défaut. Pour plus d'informations, consultez [the section called “Quotas de service Amazon Keyspaces”](#). Pour plus d'informations sur les différences fonctionnelles avec Apache Cassandra, consultez [the section called “FROZEN collections”](#). Pour plus d'informations sur la syntaxe CQL, consultez [the section called “CREATE TABLE”](#) et [the section called “ALTER TABLE”](#).

Type de tuple

Le type de tuple données représente un groupe limité d'éléments littéraux. Vous pouvez utiliser un tuple comme alternative à un `user defined type`. Il n'est pas nécessaire d'utiliser le FROZEN mot-clé pour les tuples. En effet, un tuple est toujours figé et vous ne pouvez pas mettre à jour les éléments individuellement.

Autres types

Type de données	Description
inet	Chaîne représentant une adresse IP, au format IPv4 ou IPv6.

Statique

Dans un tableau Amazon Keyspaces avec des colonnes de regroupement, vous pouvez utiliser le `STATIC` mot clé pour créer une colonne statique de n'importe quel type.

La déclaration suivante en est un exemple.

```
my_column INT STATIC
```

Pour plus d'informations sur l'utilisation de colonnes statiques, consultez [the section called “Estimation de la consommation de capacité des colonnes statiques”](#).

User-defined types (UDT)

Amazon Keyspaces prend en charge les types définis par l'utilisateur (UDT). Vous pouvez utiliser n'importe quel type de données Amazon Keyspaces valide pour créer un UDT, y compris les collections et les autres UDT existants. Vous créez des UDT dans un espace de touches et vous pouvez les utiliser pour définir des colonnes dans n'importe quel tableau de l'espace de touches.

Pour plus d'informations sur la syntaxe CQL, consultez [the section called “Types”](#). Pour plus d'informations sur l'utilisation des UDT, consultez [the section called “User-defined types \(UDT\)”](#).

Pour connaître le nombre d'UDT pris en charge par espace de touches, les niveaux d'imbrication pris en charge, ainsi que les autres valeurs et quotas par défaut liés aux UDT, voir [the section called “Quotas et valeurs par défaut pour les types définis par l'utilisateur \(UDT\) dans Amazon Keyspaces”](#)

Encodage JSON des types de données Amazon Keyspaces

Amazon Keyspaces propose les mêmes mappages de types de données JSON qu'Apache Cassandra. Le tableau suivant décrit les types de données qu'Amazon Keyspaces accepte dans les

INSERT JSON instructions et les types de données qu'Amazon Keyspaces utilise lorsqu'il renvoie des données avec l'instruction. **SELECT JSON**

Pour les types de données à champ unique tels que `floatint`, `UUID`, et `date`, vous pouvez également insérer des données sous forme `destring`. Pour les types de données composés et les collections, tels que `tuplemap`, et `list`, vous pouvez également insérer des données au format JSON ou sous forme codéeJSON `string`.

Type de données JSON	Types de données acceptés dans les INSERT JSON relevés	Types de données renvoyés dans les SELECT JSON relevés	Remarques
<code>ascii</code>	<code>string</code>	<code>string</code>	Utilise l'échappement de caractères JSON\
<code>bigint</code>	<code>integer</code> , <code>string</code>	<code>integer</code>	La chaîne doit être un entier de 64 bits valide.
<code>blob</code>	<code>string</code>	<code>string</code>	La chaîne doit commencer 0x par un nombre pair de chiffres hexadécimaux.
<code>boolean</code>	<code>boolean</code> , <code>string</code>	<code>boolean</code>	La chaîne doit être <code>true</code> soit <code>false</code> .
<code>date</code>	<code>string</code>	<code>string</code>	Date au formatYYYY-MM-DD , fuseau horaire UTC.
<code>decimal</code>	<code>integer</code> , <code>float</code> , <code>string</code>	<code>float</code>	Peut dépasser la précision à virgule IEEE-754 flottante de 32 bits ou 64 bits

Type de données JSON	Types de données acceptés dans les INSERT JSON relevés	Types de données renvoyés dans les SELECT JSON relevés	Remarques
			dans le décodeur côté client.
double	integer, float, string	float	La chaîne doit être un entier ou un nombre flottant valide.
float	integer, float, string	float	La chaîne doit être un entier ou un nombre flottant valide.
inet	string	string	Adresse IPv4 ou IPv6.
int	integer, string	integer	La chaîne doit être un entier 32 bits valide.
list	list, string	list	Utilise la représentation de liste JSON native.
map	map, string	map	Utilise la représentation cartographique JSON native.
smallint	integer, string	integer	La chaîne doit être un entier de 16 bits valide.
set	list, string	list	Utilise la représentation de liste JSON native.

Type de données JSON	Types de données acceptés dans les INSERT JSON relevés	Types de données renvoyés dans les SELECT JSON relevés	Remarques
text	string	string	Utilise l'échappement de caractères JSON\
time	string	string	Heure du jour au formatHH-MM-SS[.fffffffffff] .
timestamp	integer, string	string	Horodatage. Les constantes de chaîne vous permettent de stocker des horodatages sous forme de dates. Les horodatages avec format YYYY-MM-DD HH:MM:SS. SSS sont renvoyés.
timeuuid	string	string	Tapez 1 UUID. Voir constants pour le format UUID.
tinyint	integer, string	integer	La chaîne doit être un entier de 8 bits valide.
tuple	list, string	list	Utilise la représentation de liste JSON native.

Type de données JSON	Types de données acceptés dans les INSERT JSON relevés	Types de données renvoyés dans les SELECT JSON relevés	Remarques
UDT	map, string	map	Utilise la représentation cartographique JSON native avec les noms de champs comme clés.
uuid	string	string	Voir constants pour le format UUID.
varchar	string	string	Utilise l'échappement de caractères JSON\.
varint	integer, string	integer	Longueur variable ; peut dépasser les nombres entiers de 32 bits ou 64 bits dans le décodeur côté client.

Instructions DDL (langage de définition des données) dans Amazon Keyspaces

Le langage de définition des données (DDL) est l'ensemble des instructions Cassandra Query Language (CQL) que vous utilisez pour gérer les structures de données dans Amazon Keyspaces (pour Apache Cassandra), telles que les espaces de touches et les tables. Vous utilisez DDL pour créer ces structures de données, les modifier après leur création et les supprimer lorsqu'elles ne sont plus utilisées. Amazon Keyspaces exécute des opérations DDL de manière asynchrone. Pour plus d'informations sur la façon de confirmer qu'une opération asynchrone est terminée, consultez [the section called “Création et suppression asynchrones d'espaces clés et de tables”](#)

Les instructions DDL suivantes sont prises en charge :

- [CREATE KEYSPACE](#)
- [ALTER KEYSPACE](#)
- [DROP KEYSPACE](#)
- [UTILISER](#)
- [CRÉER UNE TABLE](#)
- [ALTER TABLE](#)
- [RESTAURER LA TABLE](#)
- [SUPPRIMER LA TABLE](#)
- [CRÉER UN TYPE](#)
- [TYPE DE CHUTE](#)

Rubriques

- [Keyspaces](#)
- [Tables](#)
- [User-defined types \(UDT\)](#)

Keyspaces

Un keyspace regroupe les tables associées qui sont pertinentes pour une ou plusieurs applications. En termes de système de gestion de base de données relationnelle (SGBDR), les keyspaces sont à peu près similaires aux bases de données, tablespaces ou constructions similaires.

Note

Dans Apache Cassandra, les keyspaces déterminent la réplique des données entre plusieurs nœuds de stockage. Cependant, Amazon Keyspaces est un service entièrement géré : les détails de sa couche de stockage sont gérés en votre nom. Pour cette raison, les espaces clés d'Amazon Keyspaces sont uniquement des constructions logiques et ne sont pas liés au stockage physique sous-jacent.

Pour plus d'informations sur les limites de quota et les contraintes pour les espaces de clés Amazon Keyspaces, consultez. [Quotas](#)

Déclarations pour les espaces clés

- [CREATE KEYSPACE](#)
- [ALTER KEYSPACE](#)
- [DROP KEYSPACE](#)
- [USE](#)

CREATE KEYSPACE

Utilisez l'instruction `CREATE KEYSPACE` pour créer un nouveau keyspace.

Syntaxe

```
create_keyspace_statement ::=  
  CREATE KEYSPACE [ IF NOT EXISTS ] keyspace_name  
  WITH options
```

Où :

- *keyspace_name* est le nom du keyspace à créer.
- Les options peuvent être les suivantes :
 - **REPLICATION**— Une carte qui indique la stratégie de réplication pour le keyspace :
 - **SingleRegionStrategy**— Pour un espace de touches à région unique. (Obligatoire)
 - **NetworkTopologyStrategy**— Spécifiez au moins deux Régions AWS. Le facteur de réplication pour chaque région est de trois. (Facultatif)
 - **DURABLE_WRITES**— Les écritures sur Amazon Keyspaces sont toujours durables, cette option n'est donc pas requise. Toutefois, si elle est spécifiée, la valeur doit être `true`.
 - **TAGS**— Une liste de balises de paires clé-valeur à associer à la ressource lorsque vous la créez. (Facultatif)

Exemple

Créez un keyspace comme suit.

```
CREATE KEYSPACE my_keyspace  
  WITH REPLICATION = {'class': 'SingleRegionStrategy'} and TAGS ={'key1':'val1',  
  'key2':'val2'} ;
```


Pour créer un espace de touches multirégional, spécifiez-en `NetworkTopologyStrategy` et incluez-en au moins deux. Régions AWS Le facteur de réplication pour chaque région est de trois.

```
CREATE KEYSPACE my_keyspace
  WITH REPLICATION = {'class': 'NetworkTopologyStrategy', 'us-east-1': '3', 'ap-
southeast-1': '3', 'eu-west-1': '3'};
```

ALTER KEYSPACE

Vous pouvez utiliser l'`ALTER KEYSPACE WITH` instruction pour les options suivantes

- **REPLICATION**— Utilisez cette option pour ajouter une nouvelle Région AWS réplique à un keyspace. Vous pouvez ajouter une nouvelle région à un keyspace à région unique ou à un espace clé multirégional.
- **TAGS**— Utilisez cette option pour ajouter ou supprimer des balises dans un espace de touches.

Syntaxe

```
alter_keyspace_statement ::=
  ALTER KEYSPACE keyspace_name
  WITH options
```

Où :

- *keyspace_name* est le nom du keyspace à modifier.
- les options sont l'une des suivantes :
 - **ADD | DROP TAGS**— Une liste de balises de paires clé-valeur à ajouter ou à supprimer de l'espace clé.
 - **REPLICATION**— Une carte qui indique la stratégie de réplication pour le keyspace ;
 - **class**— `NetworkTopologyStrategy` définit le keyspace comme étant un keyspace multirégional.
 - **region**— Spécifiez une touche supplémentaire Région AWS pour cet espace de touches. Le facteur de réplication pour chaque région est de trois.
 - **CLIENT_SIDE_TIMESTAMPS**— La valeur par défaut est `DISABLED`. Vous ne pouvez modifier le statut que sur `ENABLED`.

Exemples

Modifiez un espace de touche comme indiqué dans l'exemple suivant pour ajouter des balises.

```
ALTER KEYSPACE my_keyspace ADD TAGS {'key1':'val1', 'key2':'val2'};
```

Pour ajouter une troisième région à un espace de touches multirégional, vous pouvez utiliser l'instruction suivante.

```
ALTER KEYSPACE my_keyspace  
WITH REPLICATION = {  
  'class': 'NetworkTopologyStrategy',  
  'us-east-1': '3',  
  'us-west-2': '3',  
  'us-west-1': '3'  
} AND CLIENT_SIDE_TIMESTAMPS = {'status': 'ENABLED'};
```

DROP KEYSPACE

Utilisez l'`DROP KEYSPACE` instruction pour supprimer un espace de touches, y compris l'ensemble de son contenu, tel que les tableaux.

Syntaxe

```
drop_keyspace_statement ::=  
DROP KEYSPACE [ IF EXISTS ] keyspace_name
```

Où :

- *keyspace_name* est le nom du keyspace à supprimer.

Exemple

```
DROP KEYSPACE my_keyspace;
```

USE

Utilisez l'`USE` instruction pour définir le keyspace actuel. Cela vous permet de faire référence à des objets liés à un espace de touches spécifique, par exemple des tables et des types, sans utiliser le nom complet qui inclut le préfixe d'espace-clé.

Syntaxe

```
use_statement ::=  
    USE keyspace_name
```

Où :

- `keyspace_name` est le nom du keyspace à utiliser.

Exemple

```
USE my_keyspace;
```

Tables

Les tables sont les principales structures de données d'Amazon Keyspaces. Les données d'une table sont organisées en lignes et en colonnes. Un sous-ensemble de ces colonnes est utilisé pour déterminer le partitionnement (et finalement le placement des données) via la spécification d'une clé de partition.

Un autre ensemble de colonnes peut être défini en colonnes de clustering, ce qui signifie qu'elles peuvent participer en tant que prédicats à l'exécution de la requête.

Par défaut, de nouvelles tables sont créées avec une capacité de débit à la demande. Vous pouvez modifier le mode de capacité pour les tables nouvelles et existantes. Pour plus d'informations sur les modes read/write de débit de capacité, consultez [the section called “Configuration des modes read/write de capacité”](#).

Pour les tables en mode provisionné, vous pouvez configurer en option `AUTOSCALING_SETTINGS`. Pour plus d'informations sur le dimensionnement automatique d'Amazon Keyspaces et les options disponibles, consultez. [the section called “Configurer le dimensionnement automatique sur une table existante”](#)

Pour plus d'informations sur les limites de quota et les contraintes pour les tables Amazon Keyspaces, consultez. [Quotas](#)

Relevés pour tableaux

- [CREATE TABLE](#)

- [ALTER TABLE](#)
- [RESTAURER LA TABLE](#)
- [DROP TABLE](#)

CREATE TABLE

Utilisez l'instruction `CREATE TABLE` pour créer une nouvelle table.

Syntaxe

```
create_table_statement ::= CREATE TABLE [ IF NOT EXISTS ] table_name
    '('
        column_definition
        ( ',' column_definition )*
        [ ',' PRIMARY KEY '(' primary_key ')' ]
    ')' [ WITH table_options ]

column_definition      ::= column_name cql_type [ FROZEN ][ STATIC ][ PRIMARY KEY]

primary_key           ::= partition_key [ ',' clustering_columns ]

partition_key         ::= column_name
                        | '(' column_name ( ',' column_name )* ')'

clustering_columns    ::= column_name ( ',' column_name )*

table_options         ::= [table_options]
                        | CLUSTERING ORDER BY '(' clustering_order
    ')' [ AND table_options ]
                        | cdc
                        | CUSTOM_PROPERTIES
                        | AUTOSCALING_SETTINGS
                        | default_time_to_live
                        | TAGS

clustering_order      ::= column_name (ASC | DESC) ( ',' column_name (ASC | DESC) )*
```

Où :

- *table_name* est le nom de la table à créer. Le nom complet inclut le préfixe keyspace. Vous pouvez également définir le keyspace actuel à l'aide de l'instruction `USE keyspace`.

- *column_definition* comprend les éléments suivants :
 - *column_name*— Nom de la colonne.
 - *cql_type*— Un type de données Amazon Keyspaces (voir [Types de données](#)).
 - *FROZEN*— Désigne cette colonne définie par l'utilisateur ou de type collection (par exemple, LISTSET, ou MAP) comme figée. Une collection figée est sérialisée en une seule valeur immuable et traitée comme une. BLOB Pour plus d'informations, consultez [the section called "Types de collections"](#).
 - *STATIC*— Désigne cette colonne comme statique. Les colonnes statiques stockent les valeurs partagées par toutes les lignes d'une même partition.
 - *PRIMARY KEY*— Désigne cette colonne comme clé primaire de la table.
- *primary_key* comprend les éléments suivants :
 - *partition_key*
 - *clustering_columns*
- *partition_key*:
 - La clé de partition peut être une seule colonne ; il peut aussi s'agir d'une valeur composée de deux colonnes ou plus. La partie clé de partition de la clé primaire est obligatoire et détermine la manière dont Amazon Keyspaces stocke vos données.
- *clustering_columns*:
 - La partie facultative de la colonne de clustering de votre clé primaire détermine la façon dont les données sont regroupées et triées dans chaque partition.
- *table_options* se composent des éléments suivants :
 - *CLUSTERING ORDER BY*— L'ORDRE DE CLUSTERING par défaut sur une table est composé de vos clés de clustering dans le sens de ASC tri (croissant). Spécifiez-le pour remplacer le comportement de tri par défaut.
 - *cdc*— Un booléen qui indique si Amazon Keyspaces crée un flux de capture des données de modification (CDC) pour la table. La valeur par défaut est false. Pour spécifier le view type lors de l'activation d'un flux, définissez le *cdc_specification* avec CUSTOM_PROPERTIES.
 - *CUSTOM_PROPERTIES*— Une carte des paramètres spécifiques à Amazon Keyspaces.
 - *capacity_mode*: Spécifie le mode de capacité de read/write débit pour la table. Les options sont *throughput_mode:PAY_PER_REQUEST* et *throughput_mode:PROVISIONED*. Le mode de capacité allouée nécessite *read_capacity_units* et *write_capacity_units* en tant qu'entrées. La valeur par défaut est *throughput_mode:PAY_PER_REQUEST*.
 - *cdc_specification*: Spécifie *view_type* le flux CDC. Les options sont :

- `NEW_AND_OLD_IMAGES`— les deux versions de la ligne, avant et après la modification. Il s'agit de l'option par défaut.
- `NEW_IMAGE`— la version de la ligne après la modification.
- `OLD_IMAGE`— la version de la ligne avant la modification.
- `KEYS_ONLY`— les clés de partition et de clustering de la ligne qui a été modifiée.

Pour plus d'informations sur les flux CDC, consultez [the section called “Travailler avec les flux CDC”](#). Pour des exemples de code, consultez [the section called “Activer un flux CDC pour une nouvelle table”](#).

- `client_side_timestamps`: Spécifie si les horodatages côté client sont activés ou désactivés pour la table. Les options sont `{'status': 'enabled'}` et `{'status': 'disabled'}`. S'il n'est pas spécifié, la valeur par défaut est `status:disabled`. Une fois que les horodatages côté client sont activés pour une table, ce paramètre ne peut pas être désactivé.
- `encryption_specification`: Spécifie les options de chiffrement pour le chiffrement au repos. S'il n'est pas spécifié, la valeur par défaut est `encryption_type:AWS_OWNED_KMS_KEY`. L'option de chiffrement (clé gérée par le client) nécessite la AWS KMS clé au format Amazon Resource Name (ARN) comme entrée : `kms_key_identifieur:ARN:kms_key_identifieur:ARN`.
- `point_in_time_recovery`: Spécifie si la restauration instantanée est activée ou désactivée pour la table. Les options sont `status:enabled` et `status:disabled`. S'il n'est pas spécifié, la valeur par défaut est `status:disabled`.
- `replica_updates`: Spécifie les paramètres d'une table multirégionale spécifiques à un Région AWS. Pour une table multirégionale, vous pouvez configurer la capacité de lecture de la table différemment selon Région AWS les régions. Vous pouvez le faire en configurant les paramètres suivants. Pour plus d'informations et d'exemples, consultez [the section called “Création d'une table multirégionale en mode provisionné”](#).
 - `region`— La réplique Région AWS de la table avec les paramètres suivants :
 - `read_capacity_units`
- `TTL`: active les paramètres personnalisés Time to Live pour le tableau. Pour l'activer, utilisez `status:enabled`. La valeur par défaut est `status:disabled`. Une fois TTL activé, vous ne pouvez pas le désactiver pour le tableau.

- **AUTOSCALING_SETTINGS** inclut les paramètres facultatifs suivants pour les tables en mode provisionné. Pour plus d'informations et d'exemples, consultez [the section called “Création d'un nouveau tableau avec mise à l'échelle automatique”](#).
- **provisioned_write_capacity_autoscaling_update**:
 - **autoscaling_disabled**— Pour activer la mise à l'échelle automatique en fonction de la capacité d'écriture, définissez la valeur sur `false`. La valeur par défaut est `true`. (Facultatif)
 - **minimum_units**— Le niveau minimum de débit d'écriture que la table doit toujours être prête à prendre en charge. La valeur doit être comprise entre 1 et le quota de débit maximal par seconde pour votre compte (40 000 par défaut).
 - **maximum_units**— Le niveau maximal de débit d'écriture que la table doit toujours être prête à prendre en charge. La valeur doit être comprise entre 1 et le quota de débit maximal par seconde pour votre compte (40 000 par défaut).
 - **scaling_policy**— Amazon Keyspaces soutient la politique de suivi des cibles. La cible de dimensionnement automatique est la capacité d'écriture allouée à la table.
 - **target_tracking_scaling_policy_configuration**— Pour définir la politique de suivi de la cible, vous devez définir la valeur cible. Pour plus d'informations sur le suivi des cibles et les périodes de recharge, consultez les [politiques de dimensionnement de Target Tracking](#) dans le guide de l'utilisateur d'Application Auto Scaling.
 - **target_value**— Le taux d'utilisation cible de la table. Le dimensionnement automatique d'Amazon Keyspaces garantit que le rapport entre la capacité consommée et la capacité allouée reste égal ou proche de cette valeur. Vous définissez **target_value** en tant que pourcentage. Un double entre 20 et 90. (Obligatoire)
 - **scale_in_cooldown**— Un temps de recharge en secondes entre les activités de mise à l'échelle, qui permet à la table de se stabiliser avant qu'une autre activité d'échelle ne commence. Si aucune valeur n'est fournie, la valeur par défaut est 0. (Facultatif)
 - **scale_out_cooldown**— Un temps de recharge en secondes entre les activités de redimensionnement qui permet à la table de se stabiliser avant le début d'une autre activité de redimensionnement. Si aucune valeur n'est fournie, la valeur par défaut est 0. (Facultatif)
 - **disable_scale_in**: A boolean qui indique si la table `scale-in` est désactivée ou activée. Ce paramètre est désactivé par défaut. Pour l'activer `scale-in`, définissez la boolean valeur sur `FALSE`. Cela signifie que la capacité est automatiquement réduite pour une table en votre nom. (Facultatif)

- `provisioned_read_capacity_autoscaling_update`:
 - `autoscaling_disabled`— Pour activer la mise à l'échelle automatique en fonction de la capacité de lecture, définissez la valeur sur `false`. La valeur par défaut est `true`. (Facultatif)
 - `minimum_units`— Le niveau de débit minimal que la table doit toujours être prête à supporter. La valeur doit être comprise entre 1 et le quota de débit maximal par seconde pour votre compte (40 000 par défaut).
 - `maximum_units`— Le niveau de débit maximal que la table doit toujours être prête à supporter. La valeur doit être comprise entre 1 et le quota de débit maximal par seconde pour votre compte (40 000 par défaut).
 - `scaling_policy`— Amazon Keyspaces soutient la politique de suivi des cibles. L'objectif de dimensionnement automatique est la capacité de lecture allouée à la table.
 - `target_tracking_scaling_policy_configuration`— Pour définir la politique de suivi de la cible, vous devez définir la valeur cible. Pour plus d'informations sur le suivi des cibles et les périodes de recharge, consultez les [politiques de dimensionnement de Target Tracking](#) dans le guide de l'utilisateur d'Application Auto Scaling.
 - `target_value`— Le taux d'utilisation cible de la table. Le dimensionnement automatique d'Amazon Keyspaces garantit que le rapport entre la capacité consommée et la capacité allouée reste égal ou proche de cette valeur. Vous définissez `target_value` en tant que pourcentage. Un double entre 20 et 90. (Obligatoire)
 - `scale_in_cooldown`— Un temps de recharge en secondes entre les activités de mise à l'échelle, qui permet à la table de se stabiliser avant qu'une autre activité d'échelle ne commence. Si aucune valeur n'est fournie, la valeur par défaut est 0. (Facultatif)
 - `scale_out_cooldown`— Un temps de recharge en secondes entre les activités de redimensionnement qui permet à la table de se stabiliser avant le début d'une autre activité de redimensionnement. Si aucune valeur n'est fournie, la valeur par défaut est 0. (Facultatif)
 - `disable_scale_in`: A boolean qui indique si la table `scale-in` est désactivée ou activée. Ce paramètre est désactivé par défaut. Pour l'activer `scale-in`, définissez la boolean valeur sur `FALSE`. Cela signifie que la capacité est automatiquement réduite pour une table en votre nom. (Facultatif)
- `replica_updates`: Spécifie les paramètres de mise à l'échelle automatique Région AWS spécifiques d'une table multirégionale. Pour une table multirégionale, vous pouvez configurer

la capacité de lecture de la table différemment selon Région AWS les régions. Vous pouvez le faire en configurant les paramètres suivants. Pour plus d'informations et d'exemples, consultez [the section called “Mettre à jour les paramètres de capacité allouée et de dimensionnement automatique pour une table multirégionale”](#).

- `region`— La réplique Région AWS de la table avec les paramètres suivants :
- `provisioned_read_capacity_autoscaling_update`
 - `autoscaling_disabled`— Pour activer la mise à l'échelle automatique en fonction de la capacité de lecture de la table, définissez la valeur sur `false`. La valeur par défaut est `true`. (Facultatif)

 Note

Le dimensionnement automatique d'une table multirégionale doit être activé ou désactivé pour toutes les répliques de la table.

- `minimum_units`— Le niveau minimum de débit de lecture que la table doit toujours être prête à prendre en charge. La valeur doit être comprise entre 1 et le quota de débit maximal par seconde pour votre compte (40 000 par défaut).
- `maximum_units`— Le niveau maximal de débit de lecture que la table doit toujours être prête à prendre en charge. La valeur doit être comprise entre 1 et le quota de débit maximal par seconde pour votre compte (40 000 par défaut).
- `scaling_policy`— Amazon Keyspaces soutient la politique de suivi des cibles. L'objectif de dimensionnement automatique est la capacité de lecture allouée à la table.
- `target_tracking_scaling_policy_configuration`— Pour définir la politique de suivi de la cible, vous devez définir la valeur cible. Pour plus d'informations sur le suivi des cibles et les périodes de recharge, consultez les [politiques de dimensionnement de Target Tracking](#) dans le guide de l'utilisateur d'Application Auto Scaling.
- `target_value`— Le taux d'utilisation cible de la table. Le dimensionnement automatique d'Amazon Keyspaces garantit que le rapport entre la capacité de lecture consommée et la capacité de lecture allouée reste égal ou proche de cette valeur. Vous définissez `target_value` en tant que pourcentage. Un double entre 20 et 90. (Obligatoire)
- `scale_in_cooldown`— Un temps de recharge en secondes entre les activités de mise à l'échelle, qui permet à la table de se stabiliser avant qu'une autre activité

d'échelle ne commence. Si aucune valeur n'est fournie, la valeur par défaut est 0. (Facultatif)

- `scale_out_cooldown`— Un temps de recharge en secondes entre les activités de redimensionnement qui permet à la table de se stabiliser avant le début d'une autre activité de redimensionnement. Si aucune valeur n'est fournie, la valeur par défaut est 0. (Facultatif)
- `disable_scale_in`: A boolean qui indique si la table `scale-in` est désactivée ou activée. Ce paramètre est désactivé par défaut. Pour l'activer `scale-in`, définissez la boolean valeur sur `FALSE`. Cela signifie que la capacité de lecture est automatiquement réduite pour une table en votre nom. (Facultatif)
- `default_time_to_live`— Le paramètre Durée de vie par défaut en secondes pour le tableau.
- `TAGS`— Une liste de balises de paires clé-valeur à associer à la ressource lors de sa création.
- `clustering_order` comprend les éléments suivants :
 - `column_name`— Nom de la colonne.
 - `ASC / DESC`— Définit le modificateur d'ordre ascendant (ASC) ou descendant (DESC). S'il n'est pas spécifié, l'ordre par défaut est ASC.

Exemple

```
CREATE TABLE IF NOT EXISTS my_keyspace.my_table (
    id text,
    name text,
    region text,
    division text,
    project text,
    role text,
    pay_scale int,
    vacation_hrs float,
    manager_id text,
    PRIMARY KEY (id,division))
WITH CUSTOM_PROPERTIES={
    'capacity_mode':{
        'throughput_mode':
'PROVISIONED', 'read_capacity_units': 10, 'write_capacity_units': 20
    },
    'point_in_time_recovery':{'status':
'enabled'},
```

```

                                'encryption_specification':{
                                    'encryption_type':
'CUSTOMER_MANAGED_KMS_KEY',

                                'kms_key_identifieur': 'arn:aws:kms:eu-
west-1:555555555555:key/11111111-1111-111-1111-111111111111'
                                }
                                }
                                AND CLUSTERING ORDER BY (division ASC)
                                AND TAGS={'key1':'val1', 'key2':'val2'}
                                AND default_time_to_live = 3024000;

```

Dans une table qui utilise des colonnes de clustering, les colonnes non clusterisées peuvent être déclarées statiques dans la définition de la table. Pour plus d'informations sur les colonnes statiques, consultez [the section called “Estimation de la consommation de capacité des colonnes statiques”](#).

Exemple

```

CREATE TABLE my_keyspace.my_table (
                                id int,
                                name text,
                                region text,
                                division text,
                                project text STATIC,
                                PRIMARY KEY (id,division));

```

Vous pouvez créer une table avec une colonne utilisant un type défini par l'utilisateur (UDT). La première instruction des exemples crée un type, la seconde crée une table avec une colonne qui utilise le type.

Exemple

```

CREATE TYPE my_keyspace."udt""N@ME" (my_field int);
CREATE TABLE my_keyspace.my_table (my_col1 int pri key, my_col2 "udt""N@ME");

```

ALTER TABLE

Utilisez l'ALTER TABLE instruction pour ajouter de nouvelles colonnes, ajouter des balises ou modifier les propriétés personnalisées du tableau.

Syntaxe

```

alter_table_statement ::= ALTER TABLE table_name

    [ ADD ( column_definition | column_definition_list) ]
    [[ADD | DROP] TAGS {'key1':'val1', 'key2':'val2'}]
    [ WITH table_options [ , ... ] ] ;

column_definition      ::= column_name cql_type

```

Où :

- *table_name* est le nom de la table à modifier.
- *column_definition* est le nom de la colonne et le type de données à ajouter.
- *column_definition_list* est une liste de colonnes séparées par des virgules placées entre parenthèses.
- *table_options* se composent des éléments suivants :
 - *AUTOSCALING_SETTINGS* inclut les paramètres de mise à l'échelle automatique facultatifs pour les tables provisionnées. Pour la syntaxe et les descriptions détaillées, voir [the section called “CREATE TABLE”](#). Pour obtenir des exemples, consultez [the section called “Configurer le dimensionnement automatique sur une table existante”](#).
 - *cdc*— Un booléen qui indique si Amazon Keyspaces crée un flux de capture des données de modification (CDC) pour la table. La valeur par défaut est `false`. Pour spécifier le `view_type` lors de l'activation d'un flux, définissez le `cdc_specification` avec `CUSTOM_PROPERTIES`.
 - *CUSTOM_PROPERTIES*— Une carte des paramètres spécifiques à Amazon Keyspaces.
 - `capacity_mode`: Spécifie le mode de capacité de read/write débit pour la table. Les options sont `throughput_mode:PAY_PER_REQUEST` et `throughput_mode:PROVISIONED`. Le mode de capacité allouée nécessite `read_capacity_units` et `write_capacity_units` en tant qu'entrées. La valeur par défaut est `throughput_mode:PAY_PER_REQUEST`.
 - `cdc_specification`: Spécifie `view_type` le flux CDC. Les options sont :
 - `NEW_AND_OLD_IMAGES`— les deux versions de la ligne, avant et après la modification. Il s'agit de l'option par défaut.
 - `NEW_IMAGE`— la version de la ligne après la modification.
 - `OLD_IMAGE`— la version de la ligne avant la modification.
 - `KEYS_ONLY`— les clés de partition et de clustering de la ligne qui a été modifiée.

Pour plus d'informations sur les flux CDC, consultez [the section called “Travailler avec les flux CDC”](#). Pour des exemples de code, consultez [the section called “Activer un flux CDC pour une table existante”](#).

- `client_side_timestamps`: Spécifie si les horodatages côté client sont activés ou désactivés pour la table. Les options sont `{ 'status': 'enabled' }` et `{ 'status': 'disabled' }`. S'il n'est pas spécifié, la valeur par défaut est `status:disabled`. Une fois que les horodatages côté client sont activés pour une table, ce paramètre ne peut pas être désactivé.
- `encryption_specification`: Spécifie l'option de chiffrement pour le chiffrement au repos. Les options sont `encryption_type:AWS_OWNED_KMS_KEY` et `encryption_type:CUSTOMER_MANAGED_KMS_KEY`. L'option de chiffrement (clé gérée par le client) nécessite la AWS KMS clé au format Amazon Resource Name (ARN) comme entrée `:kms_key_identifieur:ARN`.
- `point_in_time_recovery`: Spécifie si la restauration instantanée est activée ou désactivée pour la table. Les options sont `status:enabled` et `status:disabled`. La valeur par défaut est `status:disabled`.
- `replica_updates`: Spécifie les paramètres Région AWS spécifiques d'une table multirégionale. Pour une table multirégionale, vous pouvez configurer la capacité de lecture de la table différemment selon Région AWS les régions. Vous pouvez le faire en configurant les paramètres suivants. Pour plus d'informations et d'exemples, consultez [the section called “Mettre à jour les paramètres de capacité allouée et de dimensionnement automatique pour une table multirégionale”](#).
 - `region`— La réplique Région AWS de la table avec les paramètres suivants :
 - `read_capacity_units`
 - `ttl`: active les paramètres personnalisés Time to Live pour le tableau. Pour l'activer, utilisez `status:enabled`. La valeur par défaut est `status:disabled`. Une fois `ttl` activé, vous ne pouvez pas le désactiver pour le tableau.
- `default_time_to_live`: paramètre de durée de vie par défaut en secondes pour le tableau.
- `TAGS` est une liste de balises de paire clé-valeur à attacher à la ressource.

 Note

Avec ALTER TABLE, vous ne pouvez modifier qu'une seule propriété personnalisée. Vous ne pouvez pas combiner plusieurs commandes ALTER TABLE dans la même instruction.

Exemples

L'instruction suivante indique comment ajouter une colonne à une table existante.

```
ALTER TABLE mykeyspace.mytable ADD (ID int);
```

Cette instruction indique comment ajouter deux colonnes de collection à une table existante :

- Colonne de collection figée `col_frozen_list` contenant une collection figée imbriquée
- Colonne de collection non figée `col_map` contenant une collection figée imbriquée

```
ALTER TABLE my_Table ADD(col_frozen_list FROZEN<LIST<FROZEN<SET<TEXT>>>>, col_map  
MAP<INT, FROZEN<SET<INT>>>>);
```

L'exemple suivant montre comment ajouter une colonne utilisant un type défini par l'utilisateur (UDT) à une table.

```
ALTER TABLE my_keyspace.my_table ADD (my_column, my_udt);
```

Pour modifier le mode de capacité d'une table et spécifier les unités de capacité de lecture et d'écriture, vous pouvez utiliser l'instruction suivante.

```
ALTER TABLE mykeyspace.mytable WITH CUSTOM_PROPERTIES={'capacity_mode':  
{'throughput_mode': 'PROVISIONED', 'read_capacity_units': 10, 'write_capacity_units':  
20}};
```

L'instruction suivante indique une clé KMS gérée par le client pour la table.

```
ALTER TABLE mykeyspace.mytable WITH CUSTOM_PROPERTIES={  
    'encryption_specification':{  
        'encryption_type': 'CUSTOMER_MANAGED_KMS_KEY',
```

```

        'kms_key_identifiant': 'arn:aws:kms:eu-
west-1:555555555555:key/11111111-1111-111-1111-111111111111'
    }
};

```

Pour activer la restauration instantanée d'une table, vous pouvez utiliser l'instruction suivante.

```

ALTER TABLE mykeyspace.mytable WITH CUSTOM_PROPERTIES={'point_in_time_recovery':
{'status': 'enabled'}};

```

Pour définir une valeur Time to Live par défaut en secondes pour une table, vous pouvez utiliser l'instruction suivante.

```

ALTER TABLE my_table WITH default_time_to_live = 2592000;

```

Cette instruction active les paramètres Time to Live personnalisés pour une table.

```

ALTER TABLE mytable WITH CUSTOM_PROPERTIES={'ttl':{'status': 'enabled'}};

```

RESTAURER LA TABLE

Utilisez l'`RESTORE TABLE` instruction pour rétablir une table à un point précis dans le temps. Cette instruction exige que la restauration instantanée soit activée sur une table. Pour plus d'informations, consultez [the section called “Backup et restauration avec restauration instantanée”](#).

Syntaxe

```

restore_table_statement ::=
    RESTORE TABLE restored_table_name FROM TABLE source_table_name
        [ WITH table_options [ , ... ] ];

```

Où :

- *restored_table_name* est le nom de la table restaurée.
- *source_table_name* est le nom de la table source.
- *table_options* comprend les éléments suivants :
 - *restore_timestamp* est l'heure du point de restauration au format ISO 8601. S'il n'est pas spécifié, l'horodatage actuel est utilisé.
 - *CUSTOM_PROPERTIES*— Une carte des paramètres spécifiques à Amazon Keyspaces.

- `capacity_mode`: Spécifie le mode de capacité de read/write débit pour la table. Les options sont `throughput_mode:PAY_PER_REQUEST` et `throughput_mode:PROVISIONED`. Le mode de capacité allouée nécessite `read_capacity_units` et `write_capacity_units` en tant qu'entrées. La valeur par défaut est le paramètre actuel de la table source.
- `encryption_specification`: Spécifie l'option de chiffrement pour le chiffrement au repos. Les options sont `encryption_type:AWS_OWNED_KMS_KEY` et `encryption_type:CUSTOMER_MANAGED_KMS_KEY`. L'option de chiffrement (clé gérée par le client) nécessite la AWS KMS clé au format Amazon Resource Name (ARN) comme entrée :`kms_key_identifier:ARN`. Pour restaurer une table chiffrée avec une clé gérée par le client vers une table chiffrée avec un Clé détenue par AWS, Amazon Keyspaces a besoin d'accéder à la AWS KMS clé de la table source.
- `point_in_time_recovery`: Spécifie si la restauration instantanée est activée ou désactivée pour la table. Les options sont `status:enabled` et `status:disabled`. Contrairement à ce qui se passe lorsque vous créez de nouvelles tables, le statut par défaut des tables restaurées est `status:enabled` dû au fait que le paramètre est hérité de la table source. Pour désactiver le PITR pour les tables restaurées, vous devez définir `status:disabled` explicitement.
- `replica_updates`: Spécifie les paramètres Région AWS spécifiques d'une table multirégionale. Pour une table multirégionale, vous pouvez configurer la capacité de lecture de la table différemment selon Région AWS les régions. Vous pouvez le faire en configurant les paramètres suivants.
 - `region`— La réplique Région AWS de la table avec les paramètres suivants :
 - `read_capacity_units`
- `AUTOSCALING_SETTINGS` inclut les paramètres de mise à l'échelle automatique facultatifs pour les tables provisionnées. Pour une syntaxe et des descriptions détaillées, voir [the section called "CREATE TABLE"](#).
- `TAGS` est une liste de balises de paire clé-valeur à attacher à la ressource.

Note

Les tables supprimées ne peuvent être restaurées qu'au moment de leur suppression.

Exemple


```
RESTORE TABLE mykeyspace.mytable_restored from table mykeyspace.my_table
WITH restore_timestamp = '2020-06-30T04:05:00+0000'
AND custom_properties = {'point_in_time_recovery':{'status':'disabled'},
  'capacity_mode':{'throughput_mode': 'PROVISIONED', 'read_capacity_units': 10,
  'write_capacity_units': 20}}
AND TAGS={'key1':'val1', 'key2':'val2'};
```

DROP TABLE

Utilisez l'instruction `DROP TABLE` pour supprimer une table du keyspace.

Syntaxe

```
drop_table_statement ::=
  DROP TABLE [ IF EXISTS ] table_name
```

Où :

- `IF EXISTS` empêche `DROP TABLE` d'échouer si la table n'existe pas. (Facultatif)
- *table_name* est le nom de la table à supprimer.

Exemple

```
DROP TABLE my_keyspace.my_table;
```

User-defined types (UDT)

UDT : regroupement de champs et de types de données que vous pouvez utiliser pour définir une seule colonne dans Amazon Keyspaces. Les types de données valides pour les UDT sont tous les types de données Cassandra pris en charge, y compris les collections et autres UDT que vous avez déjà créés dans le même espace clé. Pour plus d'informations sur les types de données Cassandra pris en charge, consultez [the section called “Support des types de données Cassandra”](#).

```
user_defined_type::= udt_name
udt_name::= [ keyspace_name '.' ] identifier
```

Déclarations pour les types

- [CREATE TYPE](#)
- [DROP TYPE](#)

CREATE TYPE

Utilisez l'`CREATE TYPE` instruction pour créer un nouveau type.

Syntaxe

```
create_type_statement ::= CREATE TYPE [ IF NOT EXISTS ] udt_name  
    '(' field_definition ( ',' field_definition ) * ')'   
    field_definition ::= identifieur cql_type
```

Où :

- `IF NOT EXISTS` `CREATE TYPE` empêche l'échec si le type existe déjà. (Facultatif)
- *udt_name* est le nom complet de l'UDT au format type, par exemple. `my_keyspace.my_type` Si vous définissez le keyspace actuel avec l'`USE` instruction, il n'est pas nécessaire de spécifier le nom du keyspace.
- *field_definition* se compose d'un nom et d'un type.

Le tableau suivant présente des exemples de noms UDT autorisés. La première colonne indique comment saisir le nom lorsque vous créez le type, la deuxième colonne montre comment Amazon Keyspaces met en forme le nom en interne. Amazon Keyspaces attend le nom formaté pour des opérations telles que. `GetType`

Nom saisi	Nom formaté	Remarque
MY_UDT	my_udt	Sans guillemets, Amazon Keyspaces convertit tous les caractères majuscules en minuscules.
"MY_UDT"	MY_UDT	Avec les guillemets, Amazon Keyspaces respecte les majuscules et supprime les guillemets du nom formaté.
"1234"	1234	Avec des guillemets, le nom peut commencer par un chiffre, et Amazon Keyspaces supprime les guillemets du nom formaté.

Nom saisi	Nom formaté	Remarque
"Special_Ch@r@cter s<>!!"	Special_Ch@r@cter s<>!!	Avec des guillemets, le nom peut contenir des caractères spéciaux, et Amazon Keyspaces supprime les guillemets du nom formaté.
"nested""""quote s"	nested""""quotes	Amazon Keyspaces supprime les guillemets extérieurs et les guillemets d'échappement du nom formaté.

Exemples

```
CREATE TYPE my_keyspace.phone (
    country_code int,
    number text
);
```

Vous pouvez imbriquer des UDT si l'UDT imbriqué est gelé. Pour plus d'informations sur les valeurs par défaut et les quotas pour les types, consultez [the section called “Quotas UDT et valeurs par défaut d'Amazon Keyspaces”](#).

```
CREATE TYPE my_keyspace.user (
    first_name text,
    last_name text,
    phones FROZEN<phone>
);
```

Pour d'autres exemples de code montrant comment créer des UDT, consultez [the section called “User-defined types \(UDT\)”](#).

DROP TYPE

Utilisez l'`DROP TYPE` instruction pour supprimer un UDT. Vous ne pouvez supprimer qu'un type qui n'est pas utilisé par un autre type ou une autre table.

Syntaxe

```
drop_type_statement ::= DROP TYPE [ IF EXISTS ] udt_name
```

Où :

- `IF EXISTS DROP TYPE` empêche l'échec si le type n'existe pas. (Facultatif)
- `udt_name` est le nom complet de l'UDT au format type, par exemple. `my_keyspace.my_type` Si vous définissez le keyspace actuel avec l'`USE` instruction, il n'est pas nécessaire de spécifier le nom du keyspace.

Exemple

```
DROP TYPE udt_name;
```

Instructions DML (langage de manipulation de données) dans Amazon Keyspaces

Le langage de manipulation de données (DML) est l'ensemble des instructions CQL (Cassandra Query Language) que vous utilisez pour gérer les données dans les tables Amazon Keyspaces (pour Apache Cassandra). Vous utilisez des instructions DML pour ajouter, modifier ou supprimer des données dans une table.

Vous utilisez également les instructions DML pour interroger les données d'une table. (Notez que CQL ne prend pas en charge les jointures ou les sous-requêtes.)

Rubriques

- [SELECT](#)
- [INSERT](#)
- [UPDATE](#)
- [DELETE](#)

SELECT

Utilisez une instruction `SELECT` pour interroger les données.

Syntaxe

```
select_statement ::= SELECT [ JSON ] ( select_clause | '*' )  
                        FROM table_name
```

```

[ WHERE 'where_clause' ]
[ ORDER BY 'ordering_clause' ]
[ LIMIT (integer | bind_marker) ]
[ ALLOW FILTERING ]
select_clause ::= selector [ AS identifier ] ( ',' selector [ AS identifier ] )
selector ::= column_name
               | term
               | CAST '(' selector AS cql_type ')'
               | function_name '(' [ selector ( ',' selector )* ] ')'
where_clause ::= relation ( AND relation )*
relation ::= column_name operator term
               TOKEN
operator ::= '=' | '<' | '>' | '<=' | '>=' | IN | CONTAINS | CONTAINS KEY
ordering_clause ::= column_name [ ASC | DESC ] ( ',' column_name [ ASC | DESC ] )*
```

Exemples

```

SELECT name, id, manager_id FROM "myGSGKeyspace".employees_tbl ;

SELECT JSON name, id, manager_id FROM "myGSGKeyspace".employees_tbl ;
```

Pour un tableau qui fait correspondre JSON-encoded les types de données aux types de données Amazon Keyspaces, consultez. [the section called “Encodage JSON des types de données Amazon Keyspaces”](#)

Utilisation du **IN** mot clé

Le IN mot clé indique l'égalité pour une ou plusieurs valeurs. Il peut être appliqué à la clé de partition et à la colonne de clustering. Les résultats sont renvoyés dans l'ordre dans lequel les clés sont présentées dans le SELECT relevé.

Exemples

```

SELECT * from mykeyspace.mytable WHERE primary.key1 IN (1,2) and clustering.key1 = 2;
SELECT * from mykeyspace.mytable WHERE primary.key1 IN (1,2) and clustering.key1 <= 2;
SELECT * from mykeyspace.mytable WHERE primary.key1 = 1 and clustering.key1 IN (1, 2);
SELECT * from mykeyspace.mytable WHERE primary.key1 <= 2 and clustering.key1 IN (1, 2)
ALLOW FILTERING;
```

Pour plus d'informations sur le IN mot clé et sur la manière dont Amazon Keyspaces traite l'instruction, consultez. [the section called “Utiliser IN SELECT”](#)

Résultats de commande

La `ORDER BY` clause indique l'ordre de tri des résultats renvoyés. Il prend comme arguments une liste de noms de colonnes ainsi que l'ordre de tri de chaque colonne. Vous ne pouvez spécifier des colonnes de clustering que dans les clauses de commande. Non-clusteringles colonnes ne sont pas autorisées. Les options d'ordre de tri concernent `ASC` l'ordre de tri croissant et `DESC` décroissant. Si l'ordre de tri est omis, l'ordre par défaut de la colonne de clustering est utilisé. Pour les ordres de tri possibles, voir [the section called “Résultats de la commande”](#).

Exemple

```
SELECT name, id, division, manager_id FROM "myGSGKeyspace".employees_tbl WHERE id =  
'012-34-5678' ORDER BY division;
```

Lorsque vous utilisez `ORDER BY` le `IN` mot clé, les résultats sont classés dans une page. La réorganisation complète avec pagination désactivée n'est pas prise en charge.

JETON

Vous pouvez appliquer la `TOKEN` fonction à la `PARTITION KEY` colonne dans `SELECT` et `WHERE` aux clauses. Avec cette `TOKEN` fonction, Amazon Keyspaces renvoie des lignes en fonction de la valeur du jeton mappé `PARTITION_KEY` plutôt que de la valeur du `PARTITION KEY`.

`TOKEN`Les relations ne sont pas prises en charge avec le `IN` mot clé.

Exemples

```
SELECT TOKEN(id) from my_table;  
  
SELECT TOKEN(id) from my_table WHERE TOKEN(id) > 100 and TOKEN(id) < 10000;
```

Fonction TTL

Vous pouvez utiliser la `TTL` fonction avec l'`SELECT` instruction pour récupérer le délai d'expiration en secondes enregistré pour une colonne. Si aucune `TTL` valeur n'est définie, la fonction revient à `null`.

Exemple

```
SELECT TTL(my_column) from my_table;
```

La TTL fonction ne peut pas être utilisée sur des colonnes à plusieurs cellules telles que des collections.

WRITETIMEfonction

Vous pouvez utiliser la WRITETIME fonction avec l'SELECTinstruction pour récupérer l'horodatage stocké sous forme de métadonnées pour la valeur d'une colonne uniquement si la table utilise des horodatages côté client. Pour plus d'informations, consultez [the section called “Client-side horodatages”](#).

```
SELECT WRITETIME(my_column) from my_table;
```

La WRITETIME fonction ne peut pas être utilisée sur des colonnes à plusieurs cellules telles que des collections.

Note

Pour des raisons de compatibilité avec le comportement établi du pilote Cassandra, les politiques d'autorisation basées sur les balises ne sont pas appliquées lorsque vous effectuez des opérations sur les tables système à l'aide d'appels d'API Cassandra Query Language (CQL) via les pilotes Cassandra et les outils de développement. Pour plus d'informations, consultez [the section called “Accès aux ressources Amazon Keyspaces basé sur des balises”](#).

INSERT

Utilisez l'instruction INSERT pour ajouter une ligne à une table.

Syntaxe

```
insert_statement ::= INSERT INTO table_name ( names_values | json_clause )  
                    [ IF NOT EXISTS ]  
                    [ USING update_parameter ( AND update_parameter )* ]  
names_values      ::= names VALUES tuple_literal  
json_clause       ::= JSON string [ DEFAULT ( NULL | UNSET ) ]  
names             ::= '(' column_name ( ',' column_name )* ')'
```

Exemple

```
INSERT INTO "myGSGKeyspace".employees_tbl (id, name, project, region, division, role,
    pay_scale, vacation_hrs, manager_id)
VALUES ('012-34-5678', 'Russ', 'NightFlight', 'US', 'Engineering', 'IC', 3, 12.5,
    '234-56-7890') ;
```

Paramètres de mise à jour

INSERT prend en charge les valeurs suivantes comme suit `update_parameter` :

- **TTL**— Une valeur temporelle en secondes. La valeur maximale configurable est de 630 720 000 secondes, soit l'équivalent de 20 ans.
- **TIMESTAMP**— Une `bigint` valeur représentant le nombre de microsecondes écoulées depuis l'heure de base standard connue sous le nom de epoch : 1er janvier 1970 à 00h00 GMT. Dans Amazon Keyspaces, un horodatage doit être compris entre 2 jours dans le passé et 5 minutes dans le futur.

Exemple

```
INSERT INTO my_table (userid, time, subject, body, user)
    VALUES (B79CB3BA-745E-5D9A-8903-4A02327A7E09, 96a29100-5e25-11ec-90d7-
b5d91eceda0a, 'Message', 'Hello', '205.212.123.123')
    USING TTL 259200;
```

Support JSON

Pour un tableau qui fait correspondre JSON-encoded les types de données aux types de données Amazon Keyspaces, consultez [the section called “Encodage JSON des types de données Amazon Keyspaces”](#)

Vous pouvez utiliser le JSON mot clé pour insérer une carte JSON codée sous forme de ligne unique. Pour les colonnes qui existent dans le tableau mais qui sont omises dans l'instruction d'insertion JSON, utilisez-les `DEFAULT` `UNSET` pour conserver les valeurs existantes. `DEFAULT NULL` à utiliser pour écrire une valeur NULL dans chaque ligne de colonnes omises et remplacer les valeurs existantes (les frais d'écriture standard s'appliquent). `DEFAULT NULL` est l'option par défaut.

Exemple

```
INSERT INTO "myGSGKeyspace".employees_tbl JSON '{"id":"012-34-5678",
```



```
"name": "Russ",
"project": "NightFlight",
"region": "US",
"division": "Engineering",
"role": "IC",
"pay_scale": 3,
"vacation_hrs": 12.5,
"manager_id": "234-56-7890"}';
```

Si les données JSON contiennent des clés dupliquées, Amazon Keyspaces stocke la dernière valeur de la clé (similaire à Apache Cassandra). Dans l'exemple suivant, où se trouve la clé dupliquée `id`, la valeur 234-56-7890 est utilisée.

Exemple

```
INSERT INTO "myGSGKeyspace".employees_tbl JSON '{"id":"012-34-5678",
"name": "Russ",
"project": "NightFlight",
"region": "US",
"division": "Engineering",
"role": "IC",
"pay_scale": 3,
"vacation_hrs": 12.5,
"id": "234-56-7890"}';
```

UPDATE

Utilisez l'instruction UPDATE pour modifier une ligne d'une table.

Syntaxe

```
update_statement ::= UPDATE table_name
                        [ USING update_parameter ( AND update_parameter )* ]
                        SET assignment ( ',' assignment )*
                        WHERE where_clause
                        [ IF ( EXISTS | condition ( AND condition )* ) ]
update_parameter ::= ( integer | bind_marker )
assignment       ::= simple_selection '=' term
                        | column_name '=' column_name ( '+' | '-' ) term
                        | column_name '=' list_literal '+' column_name
simple_selection ::= column_name
                        | column_name '[' term ']'
```

```

| column_name '.' `field_name`
condition ::= simple_selection operator term

```

Exemple

```

UPDATE "myGSGKeyspace".employees_tbl SET pay_scale = 5 WHERE id = '567-89-0123' AND
division = 'Marketing' ;

```

Pour incrémenter un counter, utilisez la syntaxe suivante. Pour plus d'informations, consultez [the section called “Compteurs”](#).

```

UPDATE ActiveUsers SET counter = counter + 1 WHERE user = A70FE1C0-5408-4AE3-
BE34-8733E5K09F14 AND action = 'click';

```

Paramètres de mise à jour

UPDATE prend en charge les valeurs suivantes comme suit `update_parameter` :

- **TTL**— Une valeur temporelle en secondes. La valeur maximale configurable est de 630 720 000 secondes, soit l'équivalent de 20 ans.
- **TIMESTAMP**— Une bigint valeur représentant le nombre de microsecondes écoulées depuis l'heure de base standard connue sous le nom de epoch : 1er janvier 1970 à 00h00 GMT. Dans Amazon Keyspaces, un horodatage doit être compris entre 2 jours dans le passé et 5 minutes dans le futur.

Exemple

```

UPDATE my_table (userid, time, subject, body, user)
VALUES (B79CB3BA-745E-5D9A-8903-4A02327A7E09, 96a29100-5e25-11ec-90d7-
b5d91eceda0a, 'Message', 'Hello again', '205.212.123.123')
USING TIMESTAMP '2022-11-03 13:30:54+0400';

```

DELETE

Utilisez l'instruction DELETE pour supprimer une ligne d'une table.

Syntaxe

```

delete_statement ::= DELETE [ simple_selection ( ',' simple_selection ) ]
                        FROM table_name
                        [ USING update_parameter ( AND update_parameter )* ]
                        WHERE where_clause
                        [ IF ( EXISTS | condition ( AND condition )*) ]

simple_selection ::= column_name
                    | column_name '[' term ']'
                    | column_name '.' `field_name`

condition        ::= simple_selection operator term

```

Où :

- *table_name* est la table qui contient la ligne à supprimer.

Exemple

```

DELETE manager_id FROM "myGSGKeyspace".employees_tbl WHERE id='789-01-2345' AND
division='Executive' ;

```

DELETE prend en charge la valeur suivante en tant que *update_parameter* :

- **TIMESTAMP**— Une *bigint* valeur représentant le nombre de microsecondes écoulées depuis l'heure de base standard connue sous le nom de epoch : 1er janvier 1970 à 00h00 GMT.

Built-in fonctions dans Amazon Keyspaces

Amazon Keyspaces (pour Apache Cassandra) prend en charge diverses fonctions intégrées que vous pouvez utiliser dans les instructions CQL (Cassandra Query Language).

Rubriques

- [Fonctions scalaires](#)

Fonctions scalaires

Une fonction scalaire effectue un calcul sur une valeur unique et renvoie le résultat sous forme de valeur unique. Amazon Keyspaces prend en charge les fonctions scalaires suivantes.

Fonction	Description
<code>blobAsType</code>	Renvoie une valeur du type de données spécifié.
<code>cast</code>	Convertit un type de données natif en un autre type de données natif.
<code>currentDate</code>	Renvoie le courant date/time sous forme de date.
<code>currentTime</code>	Renvoie le courant date/time sous forme d'heure.
<code>currentTimestamp</code>	Renvoie le courant date/time sous forme d'horodatage.
<code>currentTimeUUID</code>	Renvoie le courant date/time sous la forme d'untimeuuid.
<code>fromJson</code>	Convertit la chaîne JSON dans le type de données de la colonne sélectionnée.
<code>maxTimeuuid</code>	Renvoie la plus grande valeur possible timeuuid pour l'horodatage ou la chaîne de date.
<code>minTimeuuid</code>	Renvoie le plus petit possible timeuuid pour l'horodatage ou la chaîne de date.
<code>now</code>	Renvoie un nouveau timeuuid unique.
<code>toDate</code>	Convertit un timeuuid ou un horodatage en un type de date.
<code>toJson</code>	Renvoie la valeur de colonne de la colonne sélectionnée au format JSON.

Fonction	Description
<code>token</code>	Renvoie la valeur de hachage de la clé de partition.
<code>toTimestamp</code>	Convertit un <code>timeuuid</code> ou une date en horodatage.
<code>TTL</code>	Renvoie le délai d'expiration en secondes pour une colonne.
<code>typeAsBlob</code>	Convertit le type de données spécifié en <code>blob</code> .
<code>toUnixTimestamp</code>	Convertit un <code>timeuuid</code> ou un horodatage en <code>bigInt</code> .
<code>uuid</code>	Renvoie un UUID de version 4 aléatoire.
<code>writetime</code>	Renvoie l'horodatage de la valeur de la colonne spécifiée.
<code>dateOf</code>	(Obsolète) Extrait l'horodatage d'un <code>timeuuid</code> et renvoie la valeur sous forme de date.
<code>unixTimestampOf</code>	(Obsolète) Extrait l'horodatage d'un <code>timeuuid</code> et renvoie la valeur sous la forme d'un horodatage entier brut 64 bits.

Quotas pour Amazon Keyspaces (pour Apache Cassandra)

Cette section décrit les quotas actuels et les valeurs par défaut pour Amazon Keyspaces (pour Apache Cassandra).

Rubriques

- [Quotas de service Amazon Keyspaces](#)
- [Augmentation ou diminution du débit \(pour les tables allouées\)](#)
- [Le chiffrement d'Amazon Keyspaces est au repos](#)
- [Quotas et valeurs par défaut pour les types définis par l'utilisateur \(UDT\) dans Amazon Keyspaces](#)

Quotas de service Amazon Keyspaces

Le tableau suivant contient les quotas Amazon Keyspaces (pour Apache Cassandra) et les valeurs par défaut. Les informations sur les quotas pouvant être ajustés sont disponibles dans la console [Service Quotas](#), où vous pouvez également demander des augmentations de quotas. Pour plus d'informations sur les quotas, contactez AWS Support.

Quota	Description	Amazon Keyspaces par défaut
Nombre maximum d'espaces de touches par Région AWS	Nombre maximal de keyspaces pour cet abonné par région. Vous pouvez ajuster cette valeur par défaut dans la console Service Quotas .	256
Nombre maximum de tables par Région AWS	Nombre maximal de tables dans tous les keyspaces pour cet abonné par région. Vous pouvez ajuster cette valeur par défaut dans la console Service Quotas .	256

Quota	Description	Amazon Keyspaces par défaut
Quantité maximale de données restaurées à l'aide d'ADD REGIONopérations	Taille maximale des données pouvant être restaurées simultanément par ADD REGION des opérations. Pour augmenter la quantité de données à restaurer simultanément, contactez Support.	10 To
Taille maximale du schéma de table	Taille maximale d'un schéma de table.	350 KO
Nombre maximal d'opérations DDL simultanées	Le nombre maximum d'opérations DDL simultanées autorisées pour cet abonné par région.	50
Nombre maximal de requêtes par connexion	Nombre maximal de requêtes CQL pouvant être traitées par une seule connexion TCP client par seconde.	3000
Taille maximale des lignes	Taille maximale d'une ligne, à l'exclusion des données de colonne statiques. Pour en savoir plus, consultez the section called “Estimer la taille des lignes” .	1 Mo

Quota	Description	Amazon Keyspaces par défaut
Nombre maximal de colonnes dans INSERT et de UPDATE relevés	Le nombre maximum de colonnes autorisées dans le CQL INSERT ou UPDATE les instructions. Une UPDATE instruction INSERT or prend en charge jusqu'à 225 colonnes normales lorsque Time to Live (TTL) est désactivé. Si le TTL est activé, il est possible de modifier jusqu'à 166 colonnes normales en une seule opération.	225/166
Nombre maximal de données statiques par partition logique	Taille agrégée maximale des données statiques dans une partition logique. Pour en savoir plus, consultez the section called “Calculer la taille de colonne statique par partition logique” .	1 Mo
Nombre maximum de sous-requêtes par instruction IN SELECT	Nombre maximal de sous-requêtes que vous pouvez utiliser pour le IN mot clé dans une SELECT instruction. Vous pouvez ajuster cette valeur par défaut dans la console Service Quotas .	100

Quota	Description	Amazon Keyspaces par défaut
Nombre maximum de collections congelées imbriquées par Région AWS	Le nombre maximum de collections imbriquées prises en charge lorsque vous utilisez le FROZEN mot clé pour une colonne contenant un type de données de collection. Pour plus d'informations sur les collections congelées, consultez the section called “Types de collections” . Pour augmenter le niveau de nidification, contactez Support.	8
Nombre maximal d'instructions dans un lot enregistré	Le nombre maximum d'instructions CQL prises en charge lorsque vous utilisez une instruction par lots enregistrée. Pour plus d'informations sur les relevés par lots, consultez the section called “Utiliser des relevés par lots” .	100
Nombre maximal d'instructions dans un lot enregistré utilisant des colonnes statiques	Nombre maximal d'instructions CQL prises en charge lorsque vous utilisez des colonnes statiques dans une instruction par lots enregistrée. Pour plus d'informations sur les relevés par lots, consultez the section called “Utiliser des relevés par lots” .	50

Quota	Description	Amazon Keyspaces par défaut
Taille maximale du relevé de lot enregistré	Taille maximale d'un relevé de lot enregistré. Pour plus d'informations sur les relevés par lots, consultez the section called “Utiliser des relevés par lots” .	4 MO
Nombre maximal d'instructions dans un lot non enregistré	Nombre maximal d'instructions CQL prises en charge lorsque vous utilisez une instruction par lots non journalisée. Pour plus d'informations sur les relevés par lots, consultez the section called “Utiliser des relevés par lots” .	30
Débit de lecture maximal par seconde	Débit de lecture maximal par seconde (unités de demande de lecture (RRU) ou unités de capacité de lecture (RCU) pouvant être allouées à une table par région. Vous pouvez ajuster cette valeur par défaut dans la console Service Quotas .	40 000
Débit d'écriture maximal par seconde	Débit d'écriture maximal par seconde (unités de demande d'écriture (WRU) ou unités de capacité d'écriture (WCU) pouvant être alloué à une table par région. Vous pouvez ajuster cette valeur par défaut dans la console Service Quotas .	40 000

Quota	Description	Amazon Keyspaces par défaut
Débit d'écriture maximal pour une table utilisant change-data-capture (CDC)	Capacité d'écriture maximale pour une table Amazon Keyspaces lorsque le CDC est activé. Ce quota par défaut s'applique uniquement aux tables en mode read/write capacité provisionnée. Vous pouvez ajuster cette valeur par défaut dans la console Service Quotas .	40 000
Account-level débit de lecture (provisionné)	Le nombre maximum d'unités de capacité de lecture agrégée (RCU) allouées au compte par région. Cela ne s'applique qu'aux tables en mode read/write capacité allouée. Vous pouvez ajuster cette valeur par défaut dans la console Service Quotas .	80 000
Account-level débit d'écriture (provisionné)	Le nombre maximum d'unités de capacité d'écriture agrégée (WCU) allouées au compte par région. Cela ne s'applique qu'aux tables en mode read/write capacité allouée. Vous pouvez ajuster cette valeur par défaut dans la console Service Quotas .	80 000

Quota	Description	Amazon Keyspaces par défaut
Nombre maximum de cibles évolutives par région et par compte	Le nombre maximum d'objets évolutifs pour le compte par région. Une table Amazon Keyspaces est considérée comme une cible évolutive si le dimensionnement automatique est activé pour la capacité de lecture, et comme une autre cible évolutive si le dimensionnement automatique est activé pour la capacité d'écriture. Vous pouvez ajuster cette valeur par défaut dans la console Service Quotas pour Application Auto Scaling en choisissant des cibles évolutives pour Amazon Keyspaces.	1 500
Taille maximale de la clé de partition	Taille maximale de la clé de partition composée. Jusqu'à 3 octets de stockage supplémentaire sont ajoutés à la taille brute de chaque colonne incluse dans la clé de partition pour les métadonnées.	2048 bytes
Taille maximale de la clé de clustering	Taille combinée maximale de toutes les colonnes de mise en cluster. Jusqu'à 4 octets de stockage supplémentaire sont ajoutés à la taille brute de chaque colonne de clustering pour les métadonnées.	850 octets

Quota	Description	Amazon Keyspaces par défaut
Nombre maximal de restaurations simultanées de tables à l'aide Point-in-time de Recovery (PITR)	Le nombre maximum de restaurations de tables simultanées à l'aide du PITR par abonné est de 4. Vous pouvez ajuster cette valeur par défaut dans la console Service Quotas .	4
Quantité maximale de données restaurées à l'aide de la restauration instantanée (PITR)	Taille maximale des données pouvant être restaurées à l'aide du PITR dans les 24 heures. Vous pouvez ajuster cette valeur par défaut dans la console Service Quotas .	5 To

Augmentation ou diminution du débit (pour les tables allouées)

Augmentation du débit alloué

Vous pouvez augmenter `ReadCapacityUnits` ou `WriteCapacityUnits` aussi souvent que nécessaire à l'aide de la console ou de l'`ALTER TABLE` instruction. Les nouveaux paramètres ne prennent effet qu'après que l'opération `ALTER TABLE` est achevée.

Vous ne pouvez pas dépasser vos quotas par compte lorsque vous ajoutez de la capacité provisionnée. Et vous pouvez augmenter la capacité allouée à vos tables autant que vous le souhaitez. Pour de plus amples informations sur les quotas par compte, veuillez consulter la section précédente, [the section called “Quotas de service Amazon Keyspaces”](#).

Diminution du débit alloué

Pour chaque tableau d'un `ALTER TABLE` relevé, vous pouvez diminuer `ReadCapacityUnits` ou `WriteCapacityUnits` (ou les deux). Les nouveaux paramètres ne prennent effet qu'après que l'opération `ALTER TABLE` est achevée.

Vous pouvez effectuer jusqu'à quatre diminutions à n'importe quel moment de la journée. Une journée est définie conformément à l'heure UTC (Universal Coordinated Time). De plus, si aucune

réduction n'a été effectuée au cours de la dernière heure, une réduction supplémentaire est autorisée. Cela porte effectivement à 27 le nombre maximum de diminutions par jour (4 diminutions au cours de la première heure et 1 diminution pour chacune des fenêtres d'une heure suivantes par jour).

Le chiffrement d'Amazon Keyspaces est au repos

Vous pouvez modifier les options de chiffrement entre une AWS KMS clé AWS détenue et une AWS KMS clé gérée par le client jusqu'à quatre fois par période de 24 heures, par table, à compter de la création de la table. Et si aucune modification n'est intervenue au cours des six dernières heures, une modification supplémentaire est autorisée. Cela porte ainsi à huit le nombre maximum de changements par jour (quatre changements au cours des six premières heures et un changement pour chacune des fenêtres de six heures suivantes de la journée).

Vous pouvez modifier l'option de chiffrement pour utiliser une AWS KMS clé AWS détenue aussi souvent que nécessaire, même si le quota précédent a été épuisé.

Voici les quotas, à moins que vous ne demandiez un seuil supérieur. Pour demander une augmentation du quota de service, consultez [Support](#).

Quotas et valeurs par défaut pour les types définis par l'utilisateur (UDT) dans Amazon Keyspaces

Quotas UDT et valeurs par défaut d'Amazon Keyspaces

Le tableau suivant contient les quotas et les valeurs par défaut relatifs aux UDT dans Amazon Keyspaces. Les informations sur les quotas pouvant être ajustés sont disponibles dans la console [Service Quotas](#), où vous pouvez également demander des augmentations de quotas. Pour plus d'informations sur ces quotas, contactez AWS Support.

Quota	Description	Amazon Keyspaces par défaut
Nombre maximum d'UDT par Région AWS	Le nombre maximum d'UDT dans tous les espaces clés pour cet abonné par région.	256

Quota	Description	Amazon Keyspaces par défaut
Nombre maximum de tables par UDT	Nombre maximal de tables pouvant référencer le même UDT.	100
Nombre maximum d'UDT par table	Le nombre maximum d'UDT auxquels une table peut faire référence.	50
Niveau maximal d'imbrication des UDT	Profondeur d'imbrication maximale prise en charge pour les UDT.	8
Nombre maximum d'UDT enfants directs par UDT	Le nombre maximum d'UDT enfants pris en charge pour un UDT.	10
Nombre maximum d'UDT parents directs par UDT	Le nombre maximum d'UDT parents pris en charge pour un UDT.	10
Taille maximale du schéma UDT	Taille maximale du schéma pour un UDT.	25 KO
Longueur maximale du nom UDT	Le nombre maximum de caractères dans le nom UDT.	48
Longueur maximale du nom de champ UDT	Nombre maximal de caractères dans le nom d'un champ UDT.	128

Historique du document pour Amazon Keyspaces (pour Apache Cassandra)

Le tableau suivant décrit les modifications importantes apportées à la documentation depuis la dernière version d'Amazon Keyspaces (pour Apache Cassandra). Pour recevoir les notifications de mise à jour de cette documentation, abonnez-vous à un flux RSS.

- Dernière mise à jour de la documentation : 28 janvier 2026

Modification	Description	Date
GetRecords L'API renvoie désormais la description de l'itérateur	La réponse de l'GetRecords API inclut désormais un <code>iteratorDescription</code> champ contenant une <code>iteratorPosition</code> valeur indiquant la position actuelle de l'itérateur dans le shard, ce qui vous permet d'optimiser la fréquence des interrogations.	2 juin 2026
Vous pouvez désormais préchauffer des tables nouvelles et existantes dans Amazon Keyspaces	Vous pouvez désormais configurer <code>warm-throughput</code> les tables nouvelles et existantes dans Amazon Keyspaces afin de les préchauffer en cas de pics de capacité prévus.	28 janvier 2026
Amazon Keyspaces propose désormais une expérience intégrée AWS CloudShell	Vous pouvez désormais lancer un shell pré-authentifié directement depuis la console Amazon Keyspaces.	26 janvier 2026

<u>Amazon Keyspaces prend désormais en charge les lots enregistrés</u>	Vous pouvez désormais combiner plusieurs instructions dans des lots enregistrés dans Amazon Keyspaces.	11 novembre 2025
<u>Modifications apportées aux certificats TLS d'Amazon Keyspaces</u>	Instructions mises à jour pour inclure les certificats émis dans le cadre de la chaîne Amazon Trust Services (ATS).	25 octobre 2025
<u>Amazon Keyspaces prend désormais en charge les requêtes IPv4 et IPv6</u>	Les points de terminaison à double pile Amazon Keyspaces prennent en charge les configurations IPv4 et IPv6.	30 septembre 2025
<u>NOWet UUID sont désormais pris en charge dans toutes les instructions CQL d'Amazon Keyspaces</u>	Vous pouvez désormais utiliser les UUID fonctions NOW et dans toutes les instructions CQL d'Amazon Keyspaces.	8 septembre 2025
<u>L'adaptateur Kinesis d'Amazon Keyspaces Streams est désormais disponible</u>	Vous pouvez désormais traiter les flux CDC d'Amazon Keyspaces avec la Kinesis Client Library (KCL).	2 juillet 2025

[Nouvelle politique gérée d'Amazon Keyspaces](#)

Amazon Keyspaces a ajouté une nouvelle politique gérée KeyspacesCDCServic eRolePolicy qui accorde les autorisations requises au rôle lié à un service pour publier en votre nom les données des métriques de flux CDC d'AWS Servic eRoleForAmazonKeys pacesCDC Amazon Keyspaces. CloudWatch Pour plus d'informations, consultez la section [Utilisation des rôles pour les flux CDC d'Amazon Keyspaces](#).

2 juillet 2025

[Mise à jour des politiques gérées par Amazon Keyspaces](#)

Amazon Keyspaces a ajouté de nouvelles autorisations `AmazonKeyspacesReadOnlyAccess_v2` et a ajouté de nouvelles autorisations aux politiques `AmazonKeyspacesReadOnlyAccess` gérées afin de permettre aux responsables d'IAM de consulter les flux. Amazon Keyspaces a mis à jour la politique `AmazonKeyspacesFullAccess` gérée avec les autorisations nécessaires pour créer un rôle lié à un service `AWSServiceRoleForAmazonKeyspacesCDC` qui publie des `CloudWatch` statistiques pour le compte des principaux.

2 juillet 2025

[Les flux de capture des données de modification \(CDC\) d'Amazon Keyspaces sont désormais disponibles](#)

Activez les flux CDC pour enregistrer les modifications dans vos tables Amazon Keyspaces.

2 juillet 2025

[La réplication multirégionale d'Amazon Keyspaces prend désormais en charge la région Afrique \(Le Cap\).](#)

Vous pouvez désormais définir la région Afrique (Le Cap) pour les espaces clés multirégionaux.

31 mars 2025

[La réplication multirégionale d'Amazon Keyspaces prend désormais en charge un nombre illimité. Régions AWS](#)

Vous pouvez désormais définir autant de régions que vous le souhaitez pour les espaces clés multirégionaux.

25 mars 2025

Amazon Keyspaces prend en charge les types définis par l'utilisateur (UDT) dans les espaces clés multirégionaux	Vous pouvez désormais créer et utiliser des UDT dans des espaces clés multirégionaux.	24 mars 2025
Mise à jour des politiques gérées par Amazon Keyspaces	Amazon Keyspaces a ajouté de nouvelles autorisations à la politique AmazonKeyspacesFullAccess gérée afin de permettre aux responsables d'IAM d'ajouter de nouvelles régions à un keyspace existant. Cela inclut une mise à jour du rôle lié au service. AWSServiceRoleForAmazonKeyspacesReplication	19 novembre 2024
Ajouter des répliques à des tables multirégionales dans Amazon Keyspaces	Vous pouvez désormais ajouter de nouveaux Régions AWS espaces clés à une ou plusieurs régions existants.	19 novembre 2024
Support pour les types définis par l'utilisateur (UDT) dans Amazon Keyspaces	Grâce à la prise en charge native des UDT dans Amazon Keyspaces, vous pouvez désormais définir des structures de données dans vos applications qui représentent des hiérarchies de données réelles.	30 octobre 2024
Ajout de la prise en charge de la réplication multirégionale d'Amazon Keyspaces	Amazon Keyspaces prend désormais en charge les modifications de schéma pour les tables multirégionales.	17 septembre 2024

[Conseils de migration mis à jour pour Amazon Keyspaces](#)

Le guide de migration mis à jour explique comment créer un plan de migration pour réussir la migration d'Apache Cassandra vers Amazon Keyspaces, y compris différentes stratégies pour les migrations hors ligne et en ligne.

28 juin 2024

[Connectez-vous à Amazon Keyspaces depuis Amazon Elastic Kubernetes Service](#)

Vous pouvez désormais suivre un didacticiel étape par étape pour vous connecter à Amazon Keyspaces depuis Amazon EKS.

7 février 2024

[Amazon Keyspaces : support de réplication multirégionale pour les tables provisionnées](#)

Amazon Keyspaces prend désormais en charge le mode de capacité provisionnée pour les tables multirégionales.

23 janvier 2024

[API de dimensionnement automatique Amazon Keyspaces pour les tables provisionnées](#)

Amazon Keyspaces propose désormais un CQL support AWS API pour configurer le dimensionnement automatique avec le mode de capacité provisionnée.

23 janvier 2024

[Activité DML d'Amazon Keyspaces incluse dans les journaux CloudTrail](#)

Vous pouvez désormais auditer les appels d'API Amazon Keyspaces Data Manipulation Language (DML) entrants. AWS CloudTrail

20 décembre 2023

[Support d'Amazon Keyspaces pour le mot clé FROZEN](#)

Amazon Keyspaces prend désormais en charge le FROZEN mot clé pour les types de données de collecte.

15 novembre 2023

[Mise à jour des politiques gérées par Amazon Keyspaces](#)

Amazon Keyspaces a ajouté de nouvelles autorisations à la politique AmazonKeyspacesFullAccess gérée afin de permettre aux clients se connectant à Amazon Keyspaces via des points de terminaison VPC d'interface d'accéder à l'instance Amazon EC2 afin de mettre à jour la table Amazon system.permissions Keyspaces avec les informations réseau provenant du VPC.

3 octobre 2023

[Mise à jour des politiques gérées par Amazon Keyspaces](#)

Amazon Keyspaces a créé une nouvelle politique AmazonKeyspacesReadOnlyAccess_v2 gérée pour permettre aux clients se connectant à Amazon Keyspaces via des points de terminaison VPC d'interface d'accéder à l'instance Amazon EC2 afin de mettre à jour la table Amazon system.permissions Keyspaces avec les informations réseau provenant du VPC.

12 septembre 2023

Bonnes pratiques pour créer des connexions dans Amazon Keyspaces	Découvrez comment améliorer et optimiser les configurations des pilotes clients dans Amazon Keyspaces.	30 juin 2023
Les espaces de touches du système sont désormais documentés pour Amazon Keyspaces	Découvrez ce qui est stocké dans les espaces de touches du système et comment leur demander des informations utiles dans Amazon Keyspaces.	21 juin 2023
Amazon Keyspaces prend désormais en charge la réplication multirégionale	La réplication multirégionale d'Amazon Keyspaces vous aide à gérer des applications distribuées dans le monde entier en améliorant la tolérance aux pannes, la stabilité et la résilience.	5 juin 2023
Mise à jour des politiques gérées par Amazon Keyspaces	Amazon Keyspaces a ajouté de nouvelles autorisations à la politique AmazonKeyspacesFullAccess gérée afin de permettre à Amazon Keyspaces de créer un rôle lié à un service lorsqu'un administrateur crée un espace clé multirégional.	5 juin 2023
Support d'Amazon Keyspaces pour le mot clé IN	Amazon Keyspaces prend désormais en charge le mot clé dans SELECT les relevés.	25 avril 2023

Cross-account accès aux Amazon Keyspaces et aux points de terminaison VPC d'interface	Découvrez comment implémenter l'accès entre comptes pour Amazon Keyspaces avec des points de terminaison VPC.	20 avril 2023
Support d'Amazon Keyspaces pour les horodatages côté client	Les horodatages côté client d'Amazon Keyspaces sont des Cassandra-compatible horodatages au niveau des cellules qui aident les applications distribuées à déterminer l'ordre des opérations d'écriture lorsque différents clients modifient les mêmes données.	14 mars 2023
Commencer à utiliser Amazon Keyspaces et les points de terminaison VPC d'interface	Dans ce didacticiel étape par étape, découvrez comment vous connecter à Amazon Keyspaces depuis un VPC.	1er mars 2023
Optimisation des coûts des tables Amazon Keyspaces	Les meilleures pratiques et conseils sont disponibles pour vous aider à identifier des stratégies permettant d'optimiser les coûts de vos tables Amazon Keyspaces existantes.	17 février 2023
Murmur3Partitioner C'est désormais la valeur par défaut	Murmur3Partitioner Il s'agit désormais du partitionneur par défaut dans Amazon Keyspaces.	17 novembre 2022
Amazon Keyspaces prend désormais en charge Murmur3Partitioner	Murmur3Partitioner Il est désormais disponible sur Amazon Keyspaces.	9 novembre 2022

[Support mis à jour pour les chaînes vides et les valeurs blob](#)

Amazon Keyspaces prend désormais également en charge les chaînes vides et les valeurs blob en tant que valeurs de colonnes de clustering.

19 octobre 2022

[Amazon Keyspaces est désormais disponible dans AWS GovCloud \(US\)](#)

Amazon Keyspaces est désormais disponible dans le AWS GovCloud (US) Region et est en cours de FedRAMP-High conformité. Pour plus d'informations sur les points de terminaison disponibles, consultez la section Points de terminaison [AWS GovCloud \(US\) Region FIPS](#).

4 août 2022

[Surveillez les coûts de stockage des tables Amazon Keyspaces avec Amazon CloudWatch](#)

Amazon Keyspaces vous permet désormais de surveiller et de suivre les coûts de stockage des tables au fil du `BillableTableSizeInBytes` CloudWatch temps grâce à cette métrique.

14 juin 2022

[Amazon Keyspaces prend désormais en charge Terraform](#)

Vous pouvez désormais utiliser Terraform pour effectuer des opérations de langage de définition de données (DDL) dans Amazon Keyspaces.

9 juin 2022

[Support de la fonction Amazon Keyspaces token](#)

Amazon Keyspaces vous aide désormais à optimiser les requêtes des applications en utilisant cette fonction. `token`

19 avril 2022

Intégration d'Amazon Keyspaces à Apache Spark	Amazon Keyspaces vous permet désormais de lire et d'écrire des données dans Apache Spark plus facilement en utilisant le connecteur open source Spark Cassandra.	19 avril 2022
Référence de l'API Amazon Keyspaces	Amazon Keyspaces prend en charge les opérations du plan de contrôle pour gérer les espaces de touches et les tables à l'aide du AWS SDK et. AWS CLI Le guide de référence de l'API décrit en détail les opérations du plan de contrôle prises en charge.	2 mars 2022
Comment résoudre les problèmes de configuration courants lors de l'utilisation d'Amazon Keyspaces.	Découvrez comment résoudre les problèmes de configuration courants que vous pouvez rencontrer lors de l'utilisation d'Amazon Keyspaces.	22 novembre 2021
Amazon Keyspaces prend en charge Time to Live (TTL).	Amazon Keyspaces Time to Live (TTL) vous aide à simplifier la logique de votre application et à optimiser le prix du stockage en expirant automatiquement les données des tables.	18 octobre 2021
Migration de données vers Amazon Keyspaces à l'aide de DSBulk.	Step-by-step tutoriel pour migrer des données d'Apache Cassandra vers Amazon Keyspaces à l'aide du DataStax Bulk Loader (DSBulk).	9 août 2021

[Amazon Keyspaces prend en charge les entrées de point de terminaison VPC dans le tableau. system.peers](#)

Amazon Keyspaces vous permet de renseigner le `system.peers` tableau avec les informations disponibles sur les points de terminais on VPC de l'interface afin d'améliorer l'équilibrage de charge et d'augmenter le débit. read/write

29 juillet 2021

[Mise à jour des politiques gérées par IAM pour prendre en charge les AWS KMS clés gérées par le client.](#)

Les politiques gérées par IAM pour Amazon Keyspaces incluent désormais des autorisations permettant de répertorier et de consulter les clés AWS KMS gérées par les clients disponibles stockées dans. AWS KMS

1er juin 2021

[Support d'Amazon Keyspaces pour les clés gérées par AWS KMS le client.](#)

Amazon Keyspaces vous permet de prendre le contrôle des AWS KMS clés gérées par les clients qui sont stockées à des AWS KMS fins de chiffrement au repos.

1er juin 2021

[Support d'Amazon Keyspaces pour la syntaxe JSON](#)

Amazon Keyspaces vous aide à lire et à écrire des documents JSON plus facilement en prenant en charge la syntaxe JSON pour les opérations INSERT et SELECT.

21 janvier 2021

[Support d'Amazon Keyspaces pour les colonnes statiques](#)

Amazon Keyspaces vous permet désormais de mettre à jour et de stocker efficacement les données communes entre plusieurs lignes en utilisant des colonnes statiques.

9 novembre 2020

[Publication générale du support de NoSQL Workbench pour Amazon Keyspaces](#)

NoSQL Workbench est une application côté client qui vous permet de concevoir et de visualiser plus facilement des modèles de données non relationnels pour Amazon Keyspaces. Les clients NoSQL Workbench sont disponibles pour Windows, macOS et Linux.

28 octobre 2020

[Version préliminaire du support de NoSQL Workbench pour Amazon Keyspaces](#)

NoSQL Workbench est une application côté client qui vous permet de concevoir et de visualiser plus facilement des modèles de données non relationnels pour Amazon Keyspaces. Les clients NoSQL Workbench sont disponibles pour Windows, macOS et Linux.

5 octobre 2020

[Nouveaux exemples de code pour l'accès programmatique à Amazon Keyspaces](#)

Nous continuons d'ajouter des exemples de code pour l'accès programmatique à Amazon Keyspaces. Des exemples sont désormais disponibles pour les pilotes Java, Python, Go, C# et Perl Cassandra compatibles avec la version 3.11.2 d'Apache Cassandra.

17 juillet 2020

[Restauration instantanée \(PITR\) d'Amazon Keyspaces](#)

Amazon Keyspaces propose désormais une restauration instantanée (PITR) pour protéger vos tables contre les opérations d'écriture ou de suppression accidentelles en vous fournissant des sauvegardes continues des données de vos tables.

9 juillet 2020

[Disponibilité générale d'Amazon Keyspaces](#)

Avec Amazon Keyspaces, anciennement connu lors de la version préliminaire sous le nom d'Amazon Managed Apache Cassandra Service (MCS), vous pouvez utiliser le code Cassandra Query Language (CQL), les pilotes Cassandra sous licence Apache 2.0 et les outils de développement que vous utilisez déjà aujourd'hui.

23 avril 2020

<u>Mise à l'échelle automatique d'Amazon Keyspaces</u>	Amazon Keyspaces (pour Apache Cassandra) s'intègre à Application Auto Scaling pour vous aider à fournir une capacité de débit efficace pour des charges de travail variables en réponse au trafic réel des applications en ajustant automatiquement la capacité de débit.	23 avril 2020
<u>Interface : points de terminaison de cloud privé virtuel (VPC) pour Amazon Keyspaces</u>	Amazon Keyspaces propose une communication privée entre le service et votre VPC afin que le trafic réseau ne quitte pas le réseau Amazon.	16 avril 2020
<u>Tag-based politiques d'accès</u>	Vous pouvez désormais utiliser des balises de ressources dans les politiques IAM pour gérer l'accès à Amazon Keyspaces.	8 avril 2020
<u>Type de données du compteur</u>	Amazon Keyspaces vous permet désormais de coordonner les incréments et les diminutions par rapport aux valeurs des colonnes à l'aide de compteurs.	7 avril 2020
<u>Balisage des ressources</u>	Amazon Keyspaces vous permet désormais d'étiqueter et de classer les ressources à l'aide de balises.	31 mars 2020

CloudFormation soutien	Amazon Keyspaces vous aide désormais à automatiser la création et la gestion des ressources en utilisant CloudFormation	25 mars 2020
Support des rôles et politiques IAM et de l'authentification SigV4	Ajout d'informations sur la façon dont vous pouvez utiliser Gestion des identités et des accès AWS (IAM) pour gérer les autorisations d'accès et mettre en œuvre des politiques de sécurité pour Amazon Keyspaces et comment utiliser le plug-in d'authentification pour le pilote Java pour DataStax Cassandra afin d'accéder par programmation à Amazon Keyspaces à l'aide de rôles IAM et d'identités fédérées.	17 mars 2020
Read/write mode capacité	Amazon Keyspaces prend désormais en charge deux modes de capacité de read/write débit. Le mode read/write capacité contrôle la façon dont vous êtes facturé pour le débit de lecture et d'écriture et la façon dont la capacité de débit des tables est gérée.	20 février 2020
Première version	Cette documentation couvre la version initiale d'Amazon Keyspaces (pour Apache Cassandra).	3 décembre 2019

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.