Manuel du développeur pour le service géré pour Apache Flink

Service géré pour Apache Flink

Service géré pour Apache Flink: Manuel du développeur pour le service géré pour Apache Flink

Table of Contents

..... xvi
Qu’est-ce que le Service géré pour Apache Flink ? 1
Choisir un service géré pour Apache Flink ou un service géré pour Apache Flink Studio 1
Choix des API Apache Flink à utiliser dans le service géré pour Apache Flink 3
Choisir une API Flink 3
Démarrage 4
Comment ça marche 6
Programmation de votre application Apache Flink 6
API Datastream 6
API de table 7
Création de votre application de service géré pour Apache Flink 8
Création d'applications 8
Création de votre code d’application de service géré pour Apache Flink 8
Création de votre application de service géré pour Apache Flink 10
Démarrage de votre application de service géré pour Apache Flink 11
Vérification de votre application de service géré pour Apache Flink 12
Exécution d’applications 12
État de l’application et de la tâche 12
Les charges de travail par lot 14
Ressources d'application 14
Ressources d’application du service géré pour Apache Flink 14
Ressources d'application Apache Flink 15
API Datastream 16
DataStream Connecteurs API 16
Opérateurs d'API DataStream 30
Horodatages de l'API DataStream 31
API de table 32
Connecteurs de l'API de table 32
Attributs temporels de l'API de table 34
Utilisation de Python 35
Programmation d’une application 35
Création d'une application 39
Surveillance 40
Propriétés d'exécution 41

Utilisation des propriétés d'exécution dans la console 42
Utilisation des propriétés d'exécution dans l'interface CLI 42
Accès aux propriétés d'exécution d'une application de service géré pour Apache Flink 45
Tolérance aux pannes 46
Configuration du point de contrôle 47
Exemples d'API de création de points de contrôle 48
Instantanés 50
Mise à l'échelle 56
Configuration du parallélisme des applications et du KPU ParallelismPer 57
Allocation d'unités de traitement Kinesis 57
Mise à jour du parallélisme de votre application 58
Mise à l'échelle automatique 60
Identification 62
Ajout de balises lorsqu'une application est créée 63
Ajout ou mise à jour des balises pour une application existante 63
Répertorier les balises d'une application 64
Suppression des balises d'une application 64
Utilisation de CloudFormation avec le service géré pour Apache Flink 64
Avant de commencer 64
Écriture d'une fonction Lambda 64
Création d'un rôle Lambda 67
Invocation de fonctions Lambda 67
Invocation de fonctions Lambda 68
Tableau de bord Apache Flink 74
Accès au tableau de bord Apache Flink de votre application 75
Versions 77
Service géré Amazon pour Apache Flink version 1.15.2 77
Modifications apportées au service géré Amazon pour Apache Flink avec Apache Flink 1.15 79
Composants 79
Blocs-notes Studio 81
Création d'un bloc-notes Studio 82
Analyse interactive des données de streaming 83
Interprètes Flink 84
Variables d'environnement de table Apache Flink 85
Déploiement en tant qu'application à état durable 85

Critères Scala/Python 87
Critères SQL 87
Autorisations IAM 88
Connecteurs et dépendances 88
Connecteurs par défaut 89
Dépendances et connecteurs personnalisés 90
Fonctions définies par l'utilisateur 91
Considérations relatives aux fonctions définies par l'utilisateur 92
Activation de la création de points de contrôle 94
Réglage de l'intervalle entre les points de contrôle 94
Configuration du type de point de contrôle 94
Utilisation de AWS Glue 95
Propriétés de tableau 95
Exemples et didacticiels 97
Didacticiel de création d'un bloc-notes Studio 98
Didacticiel sur le déploiement en tant qu'application à état durable 118
Exemples 122
Résolution des problèmes 134
Arrêter une application bloquée 134
Déploiement en tant qu'application à état durable dans un VPC sans accès à Internet 134
eploy-as-app Taille D et réduction du temps de fabrication 135
Annulation de tâches 137
Redémarrage de l'interprète Apache Flink 138
Annexe : création de politiques IAM personnalisées 138
AWS Glue 139
CloudWatch Journaux 140
Flux Kinesis 141
Clusters Amazon MSK 143
Mise en route (DataStream API) 144
Composants de l'application 144
Prérequis 145
Étape 1 : configuration d'un compte 145
S’inscrire à un Compte AWS 145
Création d’un utilisateur administratif 146
Octroi d’un accès par programmation 147
Étape suivante 149

Étape 2 : configuration de AWS CLI 150
Étape suivante 151
Étape 3 : création d'une application 151
Création de deux flux de données Amazon Kinesis Data Streams 152
Écriture d'exemples d'enregistrements dans le flux d'entrée 153
Téléchargement et examen du code Java Apache Flink 154
Compilation du code d'application 155
Chargement du code Java Apache Flink 156
Création et exécution de l’application de service géré pour Apache Flink 156
Étape suivante 169
Étape 4 : nettoyage 169
Suppression de votre application de service géré pour Apache Flink 169
Supprimer vos flux de données Kinesis 169
Supprimer votre objet et votre compartiment Amazon S3 170
Supprimer vos ressources IAM 170
Supprimer vos CloudWatch ressources 170
Étape suivante 170
Étape 5 : étapes suivantes 171
Mise en route (API de table) 173
Composants de l'application 173
Prérequis 174
Création d'une application 174
Création de ressources dépendantes 174
Écriture d'exemples d'enregistrements dans le flux d'entrée 176
Téléchargement et examen du code Java Apache Flink 177
Compilation du code d'application 179
Chargement du code Java Apache Flink 179
Création et exécution de l’application de service géré pour Apache Flink 180
Étape suivante 185
Nettoyage 185
Suppression de votre application de service géré pour Apache Flink 185
Supprimer votre cluster Amazon MSK 186
Supprimer votre VPC 186
Supprimer vos objets et votre compartiment Amazon S3 186
Supprimer vos ressources IAM 186
Supprimer vos CloudWatch ressources 187

Étape suivante 187
Étapes suivantes 187
Mise en route (Python) 188
Mise en route avec Pyflink - L'interpréteur Python pour Apache | Amazon Web Services 188
Composants de l'application 189
Prérequis 189
Création d'une application 190
Création de ressources dépendantes 190
Écriture d'exemples d'enregistrements dans le flux d'entrée 192
Création et examen du code Python Apache Flink 193
Ajouter des dépendances tierces aux applications Python 195
Chargement du code Python Apache Flink 196
Création et exécution de l’application de service géré pour Apache Flink 198
Étape suivante 202
Nettoyage 202
Suppression de votre application de service géré pour Apache Flink 203
Supprimer vos flux de données Kinesis 203
Supprimer vos objets et votre compartiment Amazon S3 203
Supprimer vos ressources IAM 204
Supprimer vos CloudWatch ressources 204
Mise en route (Scala) 205
Création de ressources dépendantes 205
Écriture d'exemples d'enregistrements dans le flux d'entrée 207
Téléchargez et examinez le code de l'application 208
Avant d'utiliser le service géré pour Apache Flink pour la première fois, exécutez les tâches suivantes : 209
Création et exécution de l'application (console) 211
Pour créer l'application 211
Configuration de l'application 211
Modification de la stratégie IAM 213
Exécution de l'application 215
Arrêt de l'application 215
Création et exécution de l'application (CLI) 215
Créer une stratégie d'autorisations 215
Créer une politique IAM 217
Pour créer l'application 219

Démarrage de l'application ... 220
Arrêt de l'application ... 221
Ajouter une option de journalisation CloudWatch ... 221
Mettre à jour des propriétés d'environnement ... 221
Mise à jour du code de l’application ... 222
Nettoyage ... 223
Suppression de votre application de service géré pour Apache Flink ... 224
Supprimer vos flux de données Kinesis ... 224
Supprimer votre objet et votre compartiment Amazon S3 ... 224
Supprimer vos ressources IAM ... 224
Supprimer vos ressources CloudWatch ... 225
Utilisation d'Apache Beam ... 226
Utilisation d’Apache Beam avec le service géré pour Apache Flink ... 226
Capacités Beam ... 226
Création d'une application à l'aide d'Apache Beam ... 227
Création de ressources dépendantes ... 227
Écriture d'exemples d'enregistrements dans le flux d'entrée ... 228
Téléchargez et examinez le code de l'application ... 229
Compilation du code d'application ... 230
Chargement du code Java Apache Flink ... 231
Création et exécution de l’application de service géré pour Apache Flink ... 231
Nettoyage ... 235
Étapes suivantes ... 237
Ateliers de formation, laboratoires et mises en œuvre de solutions ... 238
Développement local des applications Apache Flink en local avant de les déployer vers le service géré pour Apache Flink ... 238
Détection des événements à l'aide du service géré pour Apache Flink Studio ... 238
Solution de streaming de données AWS ... 239
Clickstream Lab ... 239
Mise à l'échelle personnalisée ... 239
CloudWatch Tableau de bord ... 240
Amazon MSK ... 240
Plus de services gérés pour les solutions Apache Flink sur GitHub ... 240
Utilitaires ... 241
Gestionnaire d’instantanés ... 241
Analyse comparative ... 241

Exemples 242
DataStream Exemples d'API 242
Fenêtre bascule 243
Fenêtre défilante 252
Récepteur S3 262
Réplication MSK 277
Consommateur EFO 284
Récepteur Kinesis Data Firehose 295
Intercompte 311
Truststore personnalisé 320
Exemples Python 330
Fenêtre bascule 330
Fenêtre défilante 341
Récepteur S3 351
Exemples Scala : 363
Fenêtre bascule 363
Fenêtre défilante 380
Récepteur S3 398
Sécurité 416
Protection des données 417
Chiffrement des données 417
Gestion des identités et des accès 418
Public ciblé 418
Authentification par des identités 419
Gestion des accès à l’aide de politiques 423
Utilisation du service géré Amazon pour Apache Flink avec IAM 426
Exemples de politiques basées sur l’identité 434
Résolution des problèmes 438
Prévention du problème de l’adjoint confus entre services 440
Surveillance 441
Validation de la conformité 442
FedRAMP 442
Résilience 443
Reprise après sinistre 443
Gestion des versions 444
Sécurité de l’infrastructure 444

Bonnes pratiques de sécurité 445
Implémentation d'un accès sur la base du moindre privilège 445
Utilisation de rôles IAM pour accéder à d'autres services Amazon 445
Implémentation d'un chiffrement côté serveur dans des ressources dépendantes 446
CloudTrail À utiliser pour surveiller les appels d'API 446
Journalisation et surveillance 447
Journalisation 448
Interrogation des journaux avec CloudWatch Logs Insights 448
Surveillance 448
Configuration de la journalisation 450
Configuration de la CloudWatch journalisation à l'aide de la console 451
Configuration de la CloudWatch journalisation à l'aide de la CLI 451
Niveaux de surveillance des applications 456
Bonnes pratiques de journalisation 457
Journalisation de dépannage 458
Étape suivante 458
Analyse des journaux 458
Exécuter un exemple de requête 459
Exemples de requêtes 460
Métriques et dimensions dans le service géré pour Apache Flink 463
Métriques d'application 463
Métriques du connecteur Kinesis Data Streams 493
Métriques du connecteur Amazon MSK 494
Métriques d'Apache Zeppelin 497
Affichage des CloudWatch métriques 498
Métriques 498
Métriques personnalisées 500
Alertes 504
Écriture de messages personnalisés 516
Écrire dans les CloudWatch journaux à l'aide de Log4J 516
Écrire dans les CloudWatch journaux à l'aide de SLF4J 517
Utiliser AWS CloudTrail 518
Service géré pour les informations d'Apache Flink dans CloudTrail 518
Compréhension des entrées du fichier journal du service géré pour Apache Flink 519
Performances 522
Dépannage des performances 522

Le chemin des données ... 522
Solutions de résolution des problèmes de performances ... 523
Bonnes pratiques en matière de performances ... 525
Gérer correctement la mise à l'échelle ... 525
Surveiller l'utilisation des ressources de dépendance externe ... 528
Exécuter votre application Apache Flink localement ... 528
Surveillance des performances ... 528
Surveillance des performances à l'aide des métriques CloudWatch ... 528
Surveillance des performances à l'aide des journaux et alarmes CloudWatch ... 528
Quota ... 530
Maintenance ... 532
Définir un UUID pour tous les opérateurs ... 534
Préparation à la production ... 535
Tests de charge des applications ... 535
Parallélisme max. ... 535
Définir un UUID pour tous les opérateurs ... 536
Bonnes pratiques ... 537
Tolérance aux pannes : points de contrôle et points de sauvegarde ... 537
Versions de connecteurs non prises en charge ... 538
Performances et parallélisme ... 538
Configuration du parallélisme par opérateur ... 539
Journalisation ... 540
Codage ... 540
Gestion des informations d'identification ... 541
Lecture à partir de sources contenant peu de partitions ... 541
Intervalle d'actualisation des blocs-notes Studio ... 542
Performances optimales du bloc-notes Studio ... 542
Comment les stratégies de filigrane et les partitions inactives affectent les fenêtres temporelles ... 542
Récapitulatif ... 544
Exemple ... 544
Définir un UUID pour tous les opérateurs ... 553
Ajouter ServiceResourceTransformer au plugin Maven Shade ... 554
Stateful Functions pour Apache Flink ... 556
Modèle d'application Apache Flink ... 556
Emplacement de la configuration du module ... 557

Versions antérieures 558
Utilisation du connecteur Kinesis Streams d'Apache Flink avec les versions précédentes d'Apache Flink 559
Création d'applications avec Apache Flink 1.8.2 560
Création d'applications avec Apache Flink 1.6.2 561
Mise à niveau des applications 562
Connecteurs disponibles dans Apache Flink 1.6.2 et 1.8.2 562
Mise en route : Flink 1.13.2 563
Composants de l'application 563
Prérequis 564
Étape 1 : configuration d'un compte 564
Étape suivante 568
Étape 2 : configuration de AWS CLI 569
Étape 3 : création d'une application 570
Étape 4 : nettoyage 588
Étape 5 : étapes suivantes 589
Mise en route : Flink 1.11.1 590
Composants de l'application 591
Prérequis 592
Étape 1 : configuration d'un compte 592
Étape 2 : configuration de AWS CLI 596
Étape 3 : création d'une application 597
Étape 4 : nettoyage 615
Étape 5 : étapes suivantes 617
Mise en route : Flink 1.8.2 618
Composants de l'application 144
Prérequis 619
Étape 1 : configuration d'un compte 620
Étape 2 : configuration de AWS CLI 623
Étape 3 : création d'une application 625
Étape 4 : nettoyage 643
Mise en route : Flink 1.6.2 644
Composants de l'application 645
Prérequis 645
Étape 1 : configuration d'un compte 646
Étape 2 : configuration de AWS CLI 649

Étape 3 : création d'une application 651
Étape 4 : nettoyage 668
Paramètres Flink 671
Configuration d'Apache Flink 671
Backend d'État 672
Point de contrôle 672
Point de sauvegarde 674
Tailles des tas 674
Dégonflement de la mémoire tampon 675
Propriétés de configuration Flink modifiables 675
Tolérance aux pannes 675
Points de contrôle et backends d'État 675
Point de contrôle 675
Métriques natives de RockSDB 675
Options avancées pour les backends d'état 677
Options complètes du gestionnaire de tâches 677
Configuration de mémoire 678
RPC/Akka 678
Client 678
Options avancées du cluster 678
Configurations du système de fichiers 679
Options avancées de tolérance aux pannes 679
Configuration de mémoire 678
Métriques 679
Options avancées pour le point de terminaison et le client REST 679
Options de sécurité SSL avancées 679
Options de planification avancées 679
Options avancées pour l'interface utilisateur Web de Flink 679
Affichage des propriétés Flink configurées 680
Utilisation d'un VPC Amazon 681
Concepts Amazon VPC 681
Autorisations d'application VPC 682
Stratégie d'autorisations pour accéder à un VPC Amazon 683
Accès à Internet et aux services 684
Informations connexes 685
API VPC 685

CreateApplication 685
AddApplicationVpcConfiguration 686
DeleteApplicationVpcConfiguration 687
UpdateApplication 687
Exemple : utilisation d'un VPC 688
Résolution des problèmes 689
Résolutions des problèmes de développement 689
Graphiques en flamme pour Apache Flink 689
Problème lié au fournisseur d'informations d'identification avec le connecteur EFO 1.15.2 ... 690
Applications dotées de connecteurs Kinesis non pris en charge 690
Erreur de compilation : « Impossible de résoudre les dépendances du projet » 693
Choix non valide : « kinesisanalyticsv2 » 693
UpdateApplication L'action ne consiste pas à recharger le code de l'application 694
S3 StreamingFileSink FileNotFoundExceptions 694
FlinkKafkaConsumer problème avec l'arrêt avec le point de sauvegarde 696
Flink 1.15 Async Sink Deadlock 696
Traitement de la source Amazon Kinesis Data Streams désordonné lors du repartitionnement 706
Résolution des problèmes d'exécution 707
Outils de dépannage 708
Problèmes d'applications 708
L'application est en train de redémarrer 713
Le débit est trop lent 716
Croissance illimitée de l'état 717
Opérateurs liés aux E/S 718
Limitation en amont ou à la source à partir d'un flux de données Kinesis 719
Points de contrôle 720
Expiration du délai du point de contrôle 727
Échec de point de contrôle (Beam) 728
Contre-pression 730
Asymétrie de données 731
Asymétrie d'état 732
Intégration de ressources de différentes régions 733
Historique du document 734
Exemple de code d'API 741
AddApplicationCloudWatchLoggingOption 742

AddApplicationInput 742
AddApplicationInputProcessingConfiguration 743
AddApplicationOutput 744
AddApplicationReferenceDataSource 744
AddApplicationVpcConfiguration 745
CreateApplication 746
CreateApplicationSnapshot 747
DeleteApplication 747
DeleteApplicationCloudWatchLoggingOption 747
DeleteApplicationInputProcessingConfiguration 748
DeleteApplicationOutput 748
DeleteApplicationReferenceDataSource 748
DeleteApplicationSnapshot 749
DeleteApplicationVpcConfiguration 749
DescribeApplication 749
DescribeApplicationSnapshot 749
DiscoverInputSchema 750
ListApplications 750
ListApplicationSnapshots 751
StartApplication 751
StopApplication 751
UpdateApplication 752
Référence API 753

Le service géré Amazon pour Apache Flink était auparavant connu sous le nom d'Amazon Kinesis Data Analytics pour Apache Flink.

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.

Qu’est-ce que le service géré Amazon pour Apache Flink ?

Avec Amazon Managed Service pour Apache Flink, vous pouvez utiliser Java, Scala, Python ou SQL pour traiter et analyser les données de streaming. Le service vous permet de créer et d'exécuter du code à partir de sources de streaming et de sources statiques pour effectuer des analyses de séries chronologiques, alimenter des tableaux de bord en temps réel et des métriques.

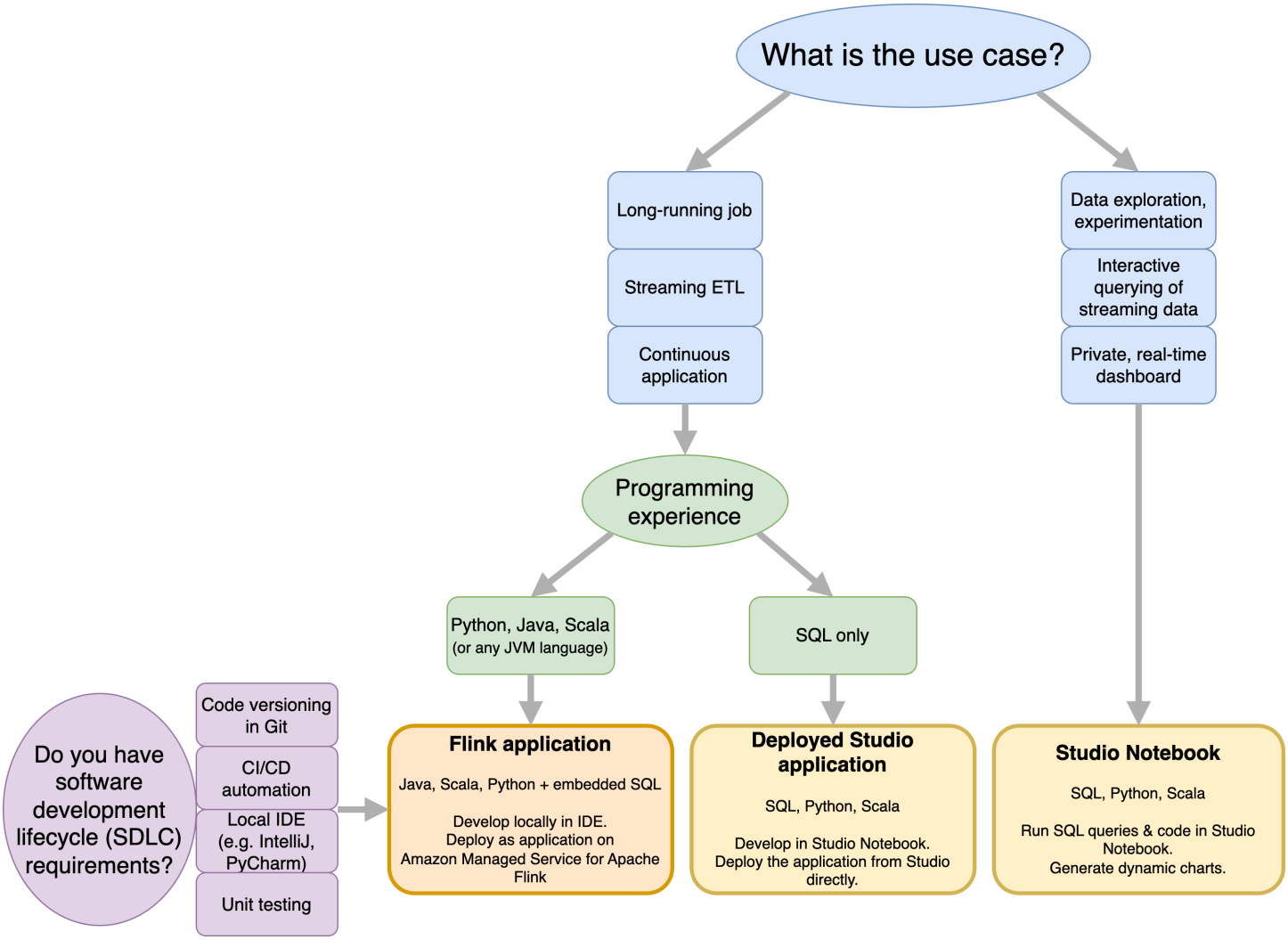
Vous pouvez créer des applications avec le langage de votre choix dans Managed Service for Apache Flink à l'aide de bibliothèques open source basées sur Apache Flink. Apache Flink est un framework et un moteur populaires permettant de traiter des flux de données.

Le service géré pour Apache Flink fournit l'infrastructure sous-jacente pour vos applications Apache Flink. Il gère des fonctionnalités de base telles que le provisionnement des ressources informatiques, la résilience au basculement en mode AZ, le calcul parallèle, le dimensionnement automatique et les sauvegardes d'applications (mises en œuvre sous forme de points de contrôle et de snapshots). Vous pouvez utiliser les fonctionnalités de programmation de haut niveau de Flink (telles que les opérateurs, les fonctions, les sources et les récepteurs) de la même manière que lorsque vous hébergez vous-même l'infrastructure Flink.

Choisir un service géré pour Apache Flink ou un service géré pour Apache Flink Studio

Vous avez deux options pour exécuter vos tâches Flink avec Amazon Managed Service pour Apache Flink. Avec Managed Service for Apache Flink, vous créez des applications Flink en Java, Scala ou Python (et en SQL intégré) à l'aide de l'IDE de votre choix et des API Apache Flink Datastream ou Table. Avec le service géré pour Apache Flink Studio, vous pouvez interroger des flux de données de manière interactive en temps réel et créer et exécuter facilement des applications de traitement de flux à l'aide de SQL, Python et Scala standard.

Vous pouvez sélectionner la méthode qui convient le mieux à votre cas d'utilisation. En cas de doute, cette section propose des conseils de haut niveau pour vous aider.



Avant de décider d'utiliser Amazon Managed Service pour Apache Flink ou Amazon Managed Service pour Apache Flink Studio, vous devez prendre en compte votre cas d'utilisation.

Si vous envisagez d'exploiter une application de longue durée qui prendra en charge des charges de travail telles que le streaming ETL ou les applications continues, vous devriez envisager d'utiliser le service géré pour Apache Flink. En effet, vous pouvez créer votre application Flink à l'aide des API Flink directement dans l'IDE de votre choix. Le développement local avec votre IDE vous permet également de tirer parti des processus et outils courants du cycle de vie du développement logiciel (SDLC) tels que le versionnement du code dans Git, l'automatisation CI/CD ou les tests unitaires.

Si vous êtes intéressé par l'exploration de données ad hoc, si vous souhaitez interroger des données de streaming de manière interactive ou créer des tableaux de bord privés en temps réel, le service géré pour Apache Flink Studio vous aidera à atteindre ces objectifs en quelques clics. Les utilisateurs familiarisés avec SQL peuvent envisager de déployer une application de longue durée directement depuis Studio.

> **Note**
>
> Vous pouvez transformer votre bloc-notes Studio en une application de longue durée. Toutefois, si vous souhaitez intégrer vos outils SDLC tels que la gestion des versions de code sur Git et l'automatisation CI/CD, ou des techniques telles que les tests unitaires, nous recommandons Managed Service pour Apache Flink en utilisant l'IDE de votre choix.

Choix des API Apache Flink à utiliser dans le service géré pour Apache Flink

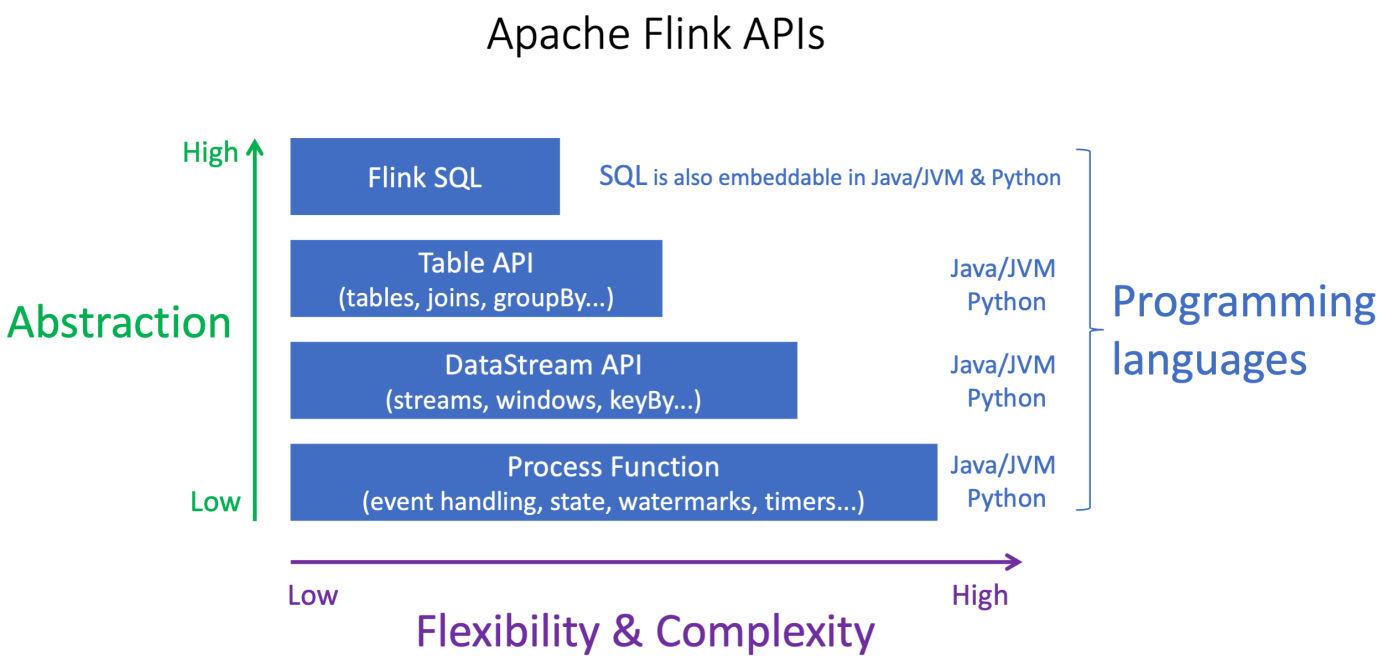
Vous pouvez créer des applications à l'aide de Java, Python et Scala dans Managed Service pour Apache Flink à l'aide des API Apache Flink dans l'IDE de votre choix. Vous trouverez des conseils sur la façon de créer des applications à l'aide de Flink Datastream et de l'API Table dans la documentation. Vous pouvez sélectionner la langue dans laquelle vous créez votre application Flink et les API que vous utilisez pour répondre au mieux aux besoins de votre application et de vos opérations. En cas de doute, cette section fournit des conseils de haut niveau pour vous aider.

Choisir une API Flink

Les API Apache Flink ont différents niveaux d'abstraction qui peuvent affecter la façon dont vous décidez de créer votre application. Ils sont expressifs et flexibles et peuvent être utilisés ensemble pour créer votre application. Vous n'êtes pas obligé d'utiliser une seule API Flink. Pour en savoir plus sur les API Flink, consultez la documentation d'Apache Flink.

Flink propose quatre niveaux d'abstraction d'API : Flink SQL, Table API, DataStream API et Process Function, qui sont utilisés conjointement avec l' DataStream API. Ils sont tous pris en charge dans Amazon Managed Service pour Apache Flink. Il est conseillé de commencer par un niveau d'abstraction plus élevé lorsque cela est possible, mais certaines fonctionnalités de Flink ne sont disponibles qu'avec l'API Datastream, où vous pouvez créer votre application en Java, Python ou Scala. Vous devriez envisager d'utiliser l'API Datastream si :

- Vous avez besoin d'un contrôle précis de l'État
- Vous souhaitez tirer parti de la possibilité d'appeler une base de données externe ou un point de terminaison de manière asynchrone (par exemple pour l'inférence)
- Vous souhaitez utiliser des minuteries personnalisées



> **Note**
>
> Choix d'une langue avec l'API Datastream :
>
> - Le SQL peut être intégré dans n'importe quelle application Flink, quel que soit le langage de programmation choisi.
> - Si vous envisagez d'utiliser l' DataStream API, tous les connecteurs ne sont pas pris en charge en Python.
> - Si vous avez besoin d'une faible latence ou d'un débit élevé, vous devriez envisager Java/Scala quelle que soit l'API.
> - Si vous envisagez d'utiliser Async IO dans l'API Process Functions, vous devez utiliser Java.

Démarrage

Vous pouvez commencer par créer une application de service géré pour Apache Flink qui lit et traite en continu les données en streaming. Créez ensuite votre code à l'aide de l'IDE de votre choix et testez-le avec des données de diffusion en direct. Vous pouvez également configurer des destinations où vous voulez que le service géré pour Apache Flink envoie les résultats.

Nous vous recommandons de commencer par lire les sections suivantes :

- Service géré pour Apache Flink : fonctionnement
- Commencer à utiliser Amazon Managed Service pour Apache Flink (DataStream API)

Vous pouvez également commencer par créer un service géré pour le bloc-notes Apache Flink Studio qui vous permet d'interroger des flux de données de manière interactive en temps réel, et de créer et d'exécuter facilement des applications de traitement de flux à l'aide de SQL, Python et Scala standard. En quelques clics dans l’AWS Management Console, vous pouvez lancer un bloc-notes sans serveur pour interroger des flux de données et obtenir des résultats en quelques secondes. Nous vous recommandons de commencer par lire les sections suivantes :

- Utilisation d’un bloc-notes Studio avec le service géré pour Apache Flink Studio
- Création d’un bloc-notes Studio

Service géré pour Apache Flink : fonctionnement

Le service géré pour Apache Flink est un service Amazon entièrement géré qui vous permet de créer une application Apache Flink pour traiter les données en streaming.

Programmation de votre application Apache Flink

Une application Apache Flink est une application Java ou Scala créée avec l'environnement Apache Flink. Vous créez votre application Apache Flink localement.

Les applications utilisent principalement l'API DataStream ou l'API de table. Les autres API Apache Flink sont également à votre disposition, mais elles sont moins couramment utilisées pour créer des applications de streaming.

Les fonctionnalités des deux API sont les suivantes :

API Datastream

Le modèle de programmation de l'API Apache Flink DataStream repose sur deux composants :

- Flux de données : représentation structurée d'un flux continu d'enregistrements de données.
- Opérateur de transformation : prend un ou plusieurs flux de données en entrée et produit un ou plusieurs flux de données en sortie.

Les applications créées avec l'API DataStream effectuent les opérations suivantes :

- Lire les données d'une source de données (telle qu'un flux Kinesis ou une rubrique Amazon MSK).
- Appliquer des transformations aux données, telles que le filtrage, l'agrégation ou l'enrichissement.
- Écrire les données transformées dans un récepteur de données.

Les applications qui utilisent l'API DataStream peuvent être écrites en Java ou en Scala, et peuvent être lues à partir d'un flux de données Kinesis, d'une rubrique Amazon MSK ou d'une source personnalisée.

Votre application traite les données à l'aide d'un connecteur. Apache Flink utilise les types de connecteurs suivants :

- Source : connecteur utilisé pour lire des données externes.
- Récepteur : connecteur utilisé pour écrire sur des emplacements externes.
- Opérateur : connecteur utilisé pour traiter les données au sein de l'application.

Une application classique comprend au moins un flux de données avec une source, un flux de données avec un ou plusieurs opérateurs et au moins un récepteur de données.

Pour en savoir plus sur l'utilisation de l'API DataStream, consultez API Datastream.

API de table

Le modèle de programmation de l'API de table Apache Flink repose sur deux composants :

- Environnement de table : interface permettant d'accéder aux données sous-jacentes que vous utilisez pour créer et héberger une ou plusieurs tables.
- Table : objet donnant accès à une table ou à une vue SQL.
- Source de table : utilisée pour lire des données provenant d'une source externe, telle qu'une rubrique Amazon MSK.
- Fonction de table : requête SQL ou appel d'API utilisé pour transformer des données.
- Récepteur de table : utilisé pour écrire des données dans un emplacement externe, tel qu'un compartiment Amazon S3.

Les applications créées avec l'API de table effectuent les opérations suivantes :

- Créer un `TableEnvironment` en vous connectant à une `Table Source`.
- Créer une table dans l'`TableEnvironment` à l'aide de requêtes SQL ou de fonctions de l'API de table.
- Exécuter une requête sur la table à l'aide de l'API de table ou de SQL.
- Appliquer des transformations aux résultats de la requête à l'aide de fonctions de table ou de requêtes SQL.
- Écrire les résultats de la requête ou de la fonction dans un `Table Sink`.

Les applications qui utilisent l'API de table peuvent être écrites en Java ou en Scala et peuvent interroger des données à l'aide d'appels d'API ou de requêtes SQL.

Pour plus d'informations sur l'utilisation de l'API de table, consultez API de table.

Création de votre application de service géré pour Apache Flink

Le service géré pour Apache Flink est un service AWS qui crée un environnement pour héberger votre application Apache Flink et lui fournit les paramètres suivants :

- Propriétés d'exécution : paramètres que vous pouvez fournir à votre application. Vous pouvez modifier ces paramètres sans recompiler le code de votre application.
- Tolérance aux pannes : comment votre application se rétablit après une interruption ou un redémarrage.
- Journalisation et surveillance : comment votre application enregistre des événements dans CloudWatch Logs.
- Mise à l'échelle : comment votre application provisionne les ressources informatiques.

Vous pouvez créer votre application de service géré pour Apache Flink à l'aide de la console ou de l’interface AWS CLI. Pour commencer à créer une application de service géré pour Apache Flink, consultez Mise en route (DataStream API).

Création d’une application de service géré pour Apache Flink

Cette rubrique contient des informations sur la création d'un service géré pour Apache Flink.

Cette rubrique contient les sections suivantes :

- Création de votre code d’application de service géré pour Apache Flink
- Création de votre application de service géré pour Apache Flink
- Démarrage de votre application de service géré pour Apache Flink
- Vérification de votre application de service géré pour Apache Flink

Création de votre code d’application de service géré pour Apache Flink

Cette section décrit les composants que vous utilisez pour créer le code d'application pour votre application de service géré pour Apache Flink.

Nous vous recommandons d'utiliser la dernière version prise en charge d'Apache Flink pour le code de votre application. La dernière version d'Apache Flink prise en charge par le service géré pour

Apache Flink est la version 1.15.2. Pour plus d'informations sur la mise à niveau de l'application de service géré pour Apache Flink, consultez Mise à niveau des applications.

Vous créez le code de votre application à l'aide d'Apache Maven. Un projet Apache Maven utilise un fichier `pom.xml` pour spécifier les versions des composants qu'il utilise.

> **Note**
>
> Le service géré pour Apache Flink prend en charge les fichiers JAR d'une taille maximale de 512 Mo. Si vous utilisez un fichier JAR plus volumineux, votre application ne démarrera pas.

Utilisez les versons de composants suivants pour les applications de service géré pour Apache Flink :

Composant	Version
Java	11 (recommandée)
Scala	Voir la note de découplage Scala ci-dessous
Service géré pour Apache Flink Runtime () aws-kinesisanalytics-runtime	1.2.0
AWSConnecteur Kinesis () flink-connector-kinesis	1.15.2
Apache Beam (applications Beam uniquement)	2.33.0, avec la version Jackson 2.12.2

À partir de la version 1.15, Flink n'utilise plus Scala. Les applications peuvent désormais utiliser l'API Java depuis n'importe quelle version de Scala. Vous devrez intégrer la bibliothèque standard Scala de votre choix à vos applications Scala.

Pour un exemple de fichier `pom.xml` pour une application de service géré pour Apache Flink utilisant Apache Flink version 1.15.2, consultez Managed Service for Apache Flink Getting Started Application.

Pour plus d'informations sur la création d'une application de service géré pour Apache Flink utilisant Apache Beam, consultez Utilisation d'Apache Beam.

Spécification de la version d'Apache Flink de votre application

Lorsque vous utilisez l'exécution de service géré pour Apache Flink version 1.1.0 ou ultérieure, vous spécifiez la version d'Apache Flink utilisée par votre application lorsque vous compilez votre application. Vous devez fournir le paramètre `-Dflink.version` suivant à la version d'Apache Flink :

```
mvn package -Dflink.version=1.15.3
```

Pour créer des applications avec les anciennes versions d'Apache Flink, consultez Versions antérieures.

Création de votre application de service géré pour Apache Flink

Une fois que vous avez créé le code de votre application, procédez comme suit pour créer votre application de service géré pour Apache Flink :

- Charger votre code d'application : chargez votre code d'application sur un compartiment Amazon S3. Vous spécifiez le nom du compartiment S3 et le nom d'objet du code d'application lorsque vous créez votre application. Pour un didacticiel expliquant comment télécharger le code de votre application, consultez the section called “Chargement du code Java Apache Flink” dans le didacticiel Mise en route (DataStream API).
- Créer votre application de service géré pour Apache Flink : utilisez l'une des méthodes suivantes pour créer votre application de service géré pour Apache Flink :
  - Créer votre application de service géré pour Apache Flink à l'aide de la consoleAWS : vous pouvez créer et configurer votre application à l'aide de la console AWS.

    Lorsque vous créez votre application à l'aide de la console, les ressources dépendantes de votre application (telles que CloudWatch les flux de journaux, les rôles IAM et les politiques IAM) sont créées pour vous.

    Lorsque vous créez votre application à l'aide de la console, vous spécifiez la version d'Apache Flink utilisée par votre application en la sélectionnant dans le menu déroulant de la page Service géré pour Apache Flink - Créer une application.

    Pour obtenir un didacticiel sur l'utilisation de la console pour créer une application, consultez the section called “Création et exécution de l'application (console)” dans le didacticiel Mise en route (DataStream API).

- Créer votre application de service géré pour Apache Flink à l'aide de AWS CLI : vous pouvez créer et configurer votre application à l'aide de AWS CLI.

  Lorsque vous créez votre application à l'aide de la CLI, vous devez également créer les ressources dépendantes de votre application (telles que CloudWatch les flux de journaux, les rôles IAM et les politiques IAM) manuellement.

  Lorsque vous créez votre application à l'aide de l'interface CLI, vous spécifiez la version d'Apache Flink utilisée par votre application en utilisant le paramètre `RuntimeEnvironment` de l'action `CreateApplication`.

  Pour obtenir un didacticiel sur l'utilisation de l'interface CLI pour créer une application, consultez the section called "Création et exécution d'une application à l'aide de l'interface CLI" dans le didacticiel Mise en route (DataStream API).

> **Note**
>
> Vous ne pouvez pas modifier le `RuntimeEnvironment` d'une application existante. Si vous devez modifier le `RuntimeEnvironment` d'une application existante, vous devez la supprimer et la créer à nouveau.

Démarrage de votre application de service géré pour Apache Flink

Après avoir créé le code de votre application, l'avoir chargé dans S3 et créé votre application de service géré pour Apache Flink, vous pouvez démarrer votre application. Le démarrage d'une application de service géré pour Apache Flink prend généralement plusieurs minutes.

Utilisez l'une des méthodes suivantes pour démarrer votre application :

- Démarrer votre application de service géré pour Apache Flink à l'aide de la console AWS : vous pouvez exécuter votre application en choisissant Exécuter sur la page de votre application dans la console AWS.
- Démarrez votre application Managed Service for Apache Flink à l'aide de l'AWSAPI : vous pouvez exécuter votre application à l'aide de l'StartApplicationaction.

Vérification de votre application de service géré pour Apache Flink

Vous pouvez vérifier que votre application fonctionne de la manière suivante :

- Utilisation CloudWatch des journaux : vous pouvez utiliser CloudWatch Logs et CloudWatch Logs Insights pour vérifier que votre application fonctionne correctement. Pour plus d'informations sur l'utilisation CloudWatch des journaux avec votre application Managed Service for Apache Flink, consultezJournalisation et surveillance.
- Utilisation CloudWatch des métriques : vous pouvez utiliser CloudWatch les métriques pour surveiller l'activité de votre application, ou l'activité des ressources qu'elle utilise pour les entrées ou les sorties (telles que les flux Kinesis, les flux Kinesis Data Firehose ou les compartiments Amazon S3). Pour plus d'informations sur CloudWatch les métriques, consultez Working with Metrics dans le guide de CloudWatch l'utilisateur Amazon.
- Surveillance des emplacements de sortie : si votre application écrit la sortie vers un emplacement (tel qu'un compartiment ou une base de données Amazon S3), vous pouvez surveiller cet emplacement pour détecter les données écrites.

Exécution d’une application de service géré pour l'application Apache Flink

Cette rubrique contient des informations sur l’exécution d'un service géré pour Apache Flink.

Lorsque vous exécutez votre application de service géré pour Apache Flink, le service crée une tâche Apache Flink. Une tâche Apache Flink correspond au cycle de vie d'exécution de votre application de service géré pour Apache Flink. L'exécution de la tâche et les ressources qu'elle utilise sont gérées par le gestionnaire de tâches. Le gestionnaire de tâches divise l'exécution de l'application en tâches. Chaque tâche est gérée par un gestionnaire de tâches. Lorsque vous surveillez les performances de votre application, vous pouvez examiner les performances de chaque gestionnaire de tâches ou du gestionnaire de tâches global.

Pour plus d'informations sur les tâches Apache Flink, consultez la section Jobs and Scheduling dans la documentation Apache Flink.

État de l’application et de la tâche

Votre application et la tâche de l'application ont tous deux un état d'exécution actuel :

- État de l'application : l'état actuel de votre application décrit sa phase d'exécution. Les états de l'application sont les suivants :
  - États d'application stables : votre application conserve généralement ces états jusqu'à ce que vous modifiiez son état :
    - PRÊT : une application nouvelle ou arrêtée affiche l'état PRÊT jusqu'à ce que vous l'exécutiez.
    - EN COURS D'EXÉCUTION : une application démarrée avec succès est en cours d'exécution.
  - États d'application transitoires : une application présentant ces états est généralement en train de passer à un autre état. Si une application reste dans un état transitoire pendant un certain temps, vous pouvez l'arrêter à l'aide de l'action StopApplication avec le paramètre `Force` défini sur `true`. Ces états incluent les éléments suivants :
    - `STARTING`: survient après l'action StartApplication. L'application est en train de passer de l'état `READY` à l'état `RUNNING`.
    - `STOPPING`: survient après l'action StopApplication. L'application est en train de passer de l'état `RUNNING` à l'état `READY`.
    - `DELETING`: survient après l'action DeleteApplication. L'application est en cours de suppression.
    - `UPDATING`: survient après l'action UpdateApplication. L'application est en cours de mise à jour et va revenir à l'état `RUNNING` ou `READY`.
    - `AUTOSCALING`: La propriété `AutoScalingEnabled` de ParallelismConfiguration de l'application est définie sur `true`, et le service augmente le parallélisme de l'application. Lorsque l'application est dans cet état, la seule action d'API valide que vous pouvez utiliser est l'action StopApplication avec le paramètre `Force` défini sur `true`. Pour des informations sur la mise à l'échelle automatique, consultez Mise à l'échelle automatique.
    - `FORCE_STOPPING`: survient après l'appel de l'action StopApplication avec le paramètre `Force` défini sur `true`. Un arrêt forcé de l'application est en cours. L'application est en train de passer de l'état `STARTING`, `UPDATING`, `STOPPING`, ou `AUTOSCALING` à l'état `READY`.
    - `ROLLING_BACK`: survient après l'appel de l'action RollbackApplication. L'application est en train de revenir à une version précédente. L'application est en train de passer de l'état `UPDATING` ou `AUTOSCALING` à l'état `RUNNING`.
    - `ROLLED_BACK`: Lorsque vous réussissez à annuler une application, cet état devient celui de la version à partir de laquelle vous l'avez annulée. Pour des informations sur l'annulation d'une application, consultez RollbackApplication.
    - `MAINTENANCE`: survient lorsque le service géré pour Apache Flink applique des correctifs à votre application. Pour de plus amples informations, veuillez consulter Maintenance.

Vous pouvez vérifier l'état de votre application à l'aide de la console ou à l'aide de l'action DescribeApplication.

- État de la tâche : lorsque votre application est à l'état `RUNNING`, votre tâche a un état qui décrit sa phase d'exécution en cours. Une tâche commence avec le statut `CREATED`, puis passe à l'état `RUNNING` une fois qu'elle a démarré. En cas d'erreur, votre application passe au statut suivant :
  - Pour les applications utilisant Apache Flink 1.11 et versions ultérieures, votre application entre dans l'état `RESTARTING`.
  - Pour les applications utilisant Apache Flink 1.8 et versions antérieures, votre application entre dans l'état `FAILING`.

  L'application passe ensuite à l'état `RESTARTING` ou `FAILED`, selon que la tâche peut être redémarrée ou non.

  Vous pouvez vérifier le statut de la tâche en consultant le journal CloudWatch de votre application pour vérifier les changements d'état.

Les charges de travail par lot

Le service géré Apache Flink prend en charge l'exécution de charges de travail par lots Apache Flink. Dans un traitement par lots, lorsqu'une tâche Apache Flink atteint l'état TERMINÉ, l'état de l'application de service géré pour Apache Flink est défini sur PRÊT. Pour plus d'informations sur les états des tâches Flink, consultez la section Jobs and Scheduling.

Ressources d'application

Cette section décrit les ressources système utilisées par votre application. Comprendre comment le service géré pour Apache Flink fournit et utilise les ressources vous aidera à concevoir, créer et maintenir un service géré performant et stable pour l'application Apache Flink.

Ressources d'application du service géré pour Apache Flink

Le service géré pour Apache Flink est un service AWS qui crée un environnement pour héberger votre application Apache Flink. Le service géré pour Apache Flink fournit des ressources à l'aide d'unités appelées unités de traitement Kinesis (KPU).

Un KPU représente les ressources système suivantes :

- Un cœur de processeur
- 4 Go de mémoire, dont 1 Go de mémoire native et 3 Go de mémoire de tas
- 50 Go d'espace disque

Les KPU exécutent les applications dans des unités d'exécution distinctes appelées tâches et sous-tâches. Vous pouvez comparer une sous-tâche à un fil d'actualité.

Le nombre de KPU disponibles pour une application est égal au paramètre `Parallelism` de l'application, divisé par le paramètre `ParallelismPerKPU` de l'application.

Pour de plus amples informations sur le parallélisme d’application, veuillez consulter Mise à l'échelle.

Ressources d'application Apache Flink

L'environnement Apache Flink alloue des ressources à votre application à l'aide d'unités appelées emplacements de tâche. Lorsque le service géré pour Apache Flink alloue des ressources à votre application, il attribue un ou plusieurs emplacements de tâche Apache Flink à un seul KPU. Le nombre d'emplacements attribués à un seul KPU est égal au paramètre `ParallelismPerKPU` de votre application. Pour plus d'informations sur la planification des tâches, consultez la section Job Scheduling dans la documentation d'Apache Flink.

Parallélisme de l’opérateur

Vous pouvez définir le nombre maximum de sous-tâches qu'un opérateur peut utiliser. Cette valeur s'appelle le parallélisme de l’opérateur. Par défaut, le parallélisme de chaque opérateur de votre application est égal au parallélisme de l'application. Cela signifie que par défaut, chaque opérateur de votre application peut utiliser toutes les sous-tâches disponibles dans l'application si nécessaire.

Vous pouvez définir le parallélisme des opérateurs de votre application à l'aide de la méthode `setParallelism`. Grâce à cette méthode, vous pouvez contrôler le nombre de sous-tâches que chaque opérateur peut utiliser simultanément.

Pour plus d'informations sur le chaînage d'opérateurs, consultez la section Task chaining and resource groups dans la documentation Apache Flink.

Chainage des opérateurs

Normalement, chaque opérateur utilise une sous-tâche distincte à exécuter, mais si plusieurs opérateurs s'exécutent toujours en séquence, le moteur d'exécution peut les affecter tous à la même tâche. Ce processus s'appelle le chaînage d'opérateurs.

Plusieurs opérateurs séquentiels peuvent être enchaînés dans une même tâche s'ils opèrent tous sur les mêmes données. Voici quelques-uns des critères nécessaires pour que cela soit vrai :

- Les opérateurs effectuent un transfert simple de 1 à 1.
- Les opérateurs ont tous le même parallélisme d'opérateur.

Lorsque votre application regroupe les opérateurs dans une seule sous-tâche, elle économise les ressources du système, car le service n'a pas besoin d'effectuer des opérations réseau et d'allouer des sous-tâches à chaque opérateur. Pour déterminer si votre application utilise le chaînage d'opérateurs, examinez le graphique des tâches dans la console du service géré pour Apache Flink. Chaque sommet de l'application représente un ou plusieurs opérateurs. Le graphique montre les opérateurs qui ont été enchaînés en tant que sommet unique.

API Datastream

Votre application Apache Flink utilise l'API Apache Flink DataStream pour transformer les données en flux de données.

Cette section contient les rubriques suivantes :

- Utilisation de connecteurs pour déplacer des données dans le service géré pour Apache Flink avec l'API DataStream : ces composants déplacent les données entre votre application et les sources de données et destinations externes.
- Transformation de données à l’aide d’opérateurs dans le service géré pour Apache Flink avec l'API DataStream : ces composants transforment ou regroupent des éléments de données au sein de votre application.
- Suivi des événements dans le service géré pour Apache Flink à l'aide de l'API DataStream : cette rubrique décrit comment le service géré pour Apache Flink suit les événements lors de l'utilisation de l'API DataStream.

Utilisation de connecteurs pour déplacer des données dans le service géré pour Apache Flink avec l'API DataStream

Dans l' DataStream API Amazon Managed Service for Apache Flink, les connecteurs sont des composants logiciels qui déplacent les données vers et depuis une application Managed Service for Apache Flink. Les connecteurs sont des intégrations flexibles qui vous permettent de lire des fichiers

et des répertoires. Les connecteurs sont constitués de modules complets permettant d'interagir avec les services Amazon et les systèmes tiers.

Les types de connecteurs sont les suivants :

- Sources : fournit des données à votre application à partir d'un flux de données Kinesis, d'un fichier ou d'une autre source de données.
- Récepteurs : envoie des données depuis votre application vers un flux de données Kinesis, un flux Kinesis Data Firehose ou une autre destination de données.
- E/S Asynchrone : fournit un accès asynchrone à une source de données (telle qu'une base de données) pour enrichir les événements de flux.

Connecteurs disponibles

L'environnement Apache Flink contient des connecteurs permettant d'accéder aux données provenant de diverses sources. Pour plus d'informations sur les connecteurs disponibles dans l'environnement Apache Flink, voir Connectors dans la documentation Apache Flink.

Warning

Si vous avez des applications exécutées sous Flink 1.6, 1.8, 1.11 ou 1.13 et que vous souhaitez les exécuter dans les régions Moyen-Orient (EAU), Asie-Pacifique (Hyderabad), Israël (Tel-Aviv), Europe (Zurich), Moyen-Orient (EAU), Asie-Pacifique (Melbourne) ou Asie-Pacifique (Jakarta), vous devrez peut-être reconstruire les archives de vos applications à l'aide d'un connecteur mis à jour ou passer à Flink 1.15. Les directives suivantes sont recommandées :

Améliorations de connecteurs

V Fl	Connecteur utilisé	Résolution
1, - 1,	Firehose	Votre application dépend d'une version

V Fl	Connecteur utilisé	Résolution
		obsolète du connecteur Firehose qui ne prend pas en compte les nouvelles régions AWS. Reconstruisez les archives de votre application à l'aide du connecteur Firehose version 2.1 [v2.1.0]

V Fl	Connecteur utilisé	Résolution
1.	Kinesis	Votre application dépend d'une version obsolète du connecteur Kineses Flink qui ne prend pas en compte les nouvelles régions AWS. Reconstruisez les archives de votre application à l'aide du

V Fl	Connecteur utilisé	Résolution
		connecteur Kinesis Flink version 1.6 https://github.com/awslabs/amazon-kinesis-connector-flink/tree/1.6.1

V Fl	Connecteur utilisé	Résolution
1.	Kinesis	Votre application dépend d'une version obsolète du connecteur Kineses Flink qui ne prend pas en compte les nouvelles régions AWS. Reconstruisez les archives de votre application à l'aide du

V Fl	Connecteur utilisé	Résolution
		connecteur Kinesis Flink version 2.4 https://github.com/awslabs/amazon-kinesis-connector-flink/tree/2.4.1

V Fl	Connecteur utilisé	Résolution
1. et 1.	Kinesis	Votre application dépend d'une version obsolète du connecteur Kineses Flink qui ne prend pas en compte les nouvelles régions AWS. Malheureusement, Flink ne publie plus de correctifs ou de

V Fl	Connecteur utilisé	Résolution
		corrections de bogues pour les connecteurs 1.6/1.13 Nous vous suggérons de passer à Flink 1.15 en reconstruisant l'archive de votre application avec Flink 1.15.

Ajout de sources de données de streaming au service géré pour Apache Flink

Apache Flink fournit des connecteurs pour lire à partir de fichiers, de sockets, de collections et de sources personnalisées. Dans le code de votre application, vous utilisez une source Apache Flink pour recevoir les données d'un flux. Cette section décrit les sources disponibles pour les services Amazon.

Kinesis Data Streams

La source `FlinkKinesisConsumer` fournit des données de streaming à votre application à partir d'un flux de données Amazon Kinesis.

Création d'un **`FlinkKinesisConsumer`**

L'exemple de code suivant illustre la création d'un `FlinkKinesisConsumer` :

```
Properties inputProperties = new Properties();
inputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);
inputProperties.setProperty(ConsumerConfigConstants.STREAM_INITIAL_POSITION, "LATEST");

DataStream<string> input = env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
 new SimpleStringSchema(), inputProperties));
```

Pour plus d'informations sur l'utilisation d'un `FlinkKinesisConsumer`, consultez Téléchargement et examen du code Java Apache Flink.

Création d'un **`FlinkKinesisConsumer`** qui utilise un consommateur EFO

FlinkKinesisConsumer Il est désormais compatible avec Enhanced Fan-Out (EFO).

Si un client Kinesis utilise EFO, le service Kinesis Data Streams lui fournit sa propre bande passante dédiée, au lieu que le consommateur partage la bande passante fixe du flux avec les autres consommateurs lisant le flux.

Pour plus d'informations sur l'utilisation d'EFO avec le consommateur Kinesis, consultez FLIP-128: Enhanced Fan Out for AWS Kinesis Consumers.

Vous activez le consommateur EFO en définissant les paramètres suivants sur le consommateur Kinesis :

- RECORD_PUBLISHER_TYPE : définissez ce paramètre sur EFO pour que votre application utilise un consommateur EFO pour accéder aux données du flux de données Kinesis.
- EFO_CONSUMER_NAME : définissez ce paramètre sur une valeur de chaîne unique parmi les consommateurs de ce flux. La réutilisation d'un nom de consommateur dans le même flux de données Kinesis entraînera la résiliation du client qui utilisait ce nom précédemment.

Pour configurer un `FlinkKinesisConsumer` afin d'utiliser EFO, ajoutez les paramètres suivants au consommateur :

```
consumerConfig.putIfAbsent(RECORD_PUBLISHER_TYPE, "EFO");
consumerConfig.putIfAbsent(EFO_CONSUMER_NAME, "basic-efo-flink-app");
```

Pour un exemple de service géré pour une application Apache Flink utilisant un consommateur EFO, consultez. Consommateur EFO

Amazon MSK

La source `KafkaSource` fournit des données de streaming à votre application à partir d'une rubrique Amazon MSK.

Création d'un **`KafkaSource`**

L'exemple de code suivant illustre la création d'un `KafkaSource` :

```
KafkaSource<String> source = KafkaSource.<String>builder()
    .setBootstrapServers(brokers)
    .setTopics("input-topic")
    .setGroupId("my-group")
    .setStartingOffsets(OffsetsInitializer.earliest())
    .setValueOnlyDeserializer(new SimpleStringSchema())
    .build();

env.fromSource(source, WatermarkStrategy.noWatermarks(), "Kafka Source");
```

Pour plus d'informations sur l'utilisation d'un `KafkaSource`, consultez Réplication MSK.

Écriture de données à l'aide de récepteurs dans le service géré pour Apache Flink

Dans le code de votre application, vous utilisez un récepteur Apache Flink pour écrire des données d'un flux Apache Flink vers un service AWS, tel que Kinesis Data Streams.

Apache Flink fournit des récepteurs pour les fichiers, les sockets et les récepteurs personnalisés. Les récepteurs suivantes sont disponibles pour AWS :

Kinesis Data Streams

Apache Flink fournit des informations sur le connecteur Kinesis Data Streams dans la documentation d'Apache Flink.

Pour un exemple d'application qui utilise un flux de données Kinesis pour l'entrée et la sortie, consultez Mise en route (DataStream API).

Amazon S3

Vous pouvez utiliser le `StreamingFileSink` Apache Flink pour écrire des objets dans un compartiment Amazon S3.

Pour un exemple sur la façon d'écrire des objets dans S3, consultezthe section called “Récepteur S3”.

Kinesis Data Firehose

Le `FlinkKinesisFirehoseProducer` est un récepteur Apache Flink fiable et évolutif permettant de stocker les résultats des applications à l'aide du service Kinesis Data Firehose. Cette section décrit comment configurer un projet Maven pour créer et utiliser un `FlinkKinesisFirehoseProducer`.

Rubriques

- Création d'un FlinkKinesisFirehoseProducer
- Exemple de code FlinkKinesisFirehoseProducer

Création d'un **`FlinkKinesisFirehoseProducer`**

L'exemple de code suivant illustre la création d’un `FlinkKinesisFirehoseProducer` :

```
Properties outputProperties = new Properties();
outputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);

FlinkKinesisFirehoseProducer<String> sink = new
 FlinkKinesisFirehoseProducer<>(outputStreamName, new SimpleStringSchema(),
 outputProperties);
```

Exemple de code **`FlinkKinesisFirehoseProducer`**

L'exemple de code suivant montre comment créer et configurer un `FlinkKinesisFirehoseProducer` et envoyer des donnéesà partir d’un flux de données Apache Flink vers le service Kinesis Data Firehose.

```
package com.amazonaws.services.kinesisanalytics;

import
 com.amazonaws.services.kinesisanalytics.flink.connectors.config.ProducerConfigConstants;
```

```
import
 com.amazonaws.services.kinesisanalytics.flink.connectors.producer.FlinkKinesisFirehoseProducer
import com.amazonaws.services.kinesisanalytics.runtime.KinesisAnalyticsRuntime;
import org.apache.flink.api.common.serialization.SimpleStringSchema;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer;
import org.apache.flink.streaming.connectors.kinesis.FlinkKinesisProducer;

import org.apache.flink.streaming.connectors.kinesis.config.ConsumerConfigConstants;

import java.io.IOException;
import java.util.Map;
import java.util.Properties;

public class StreamingJob {

 private static final String region = "us-east-1";
 private static final String inputStreamName = "ExampleInputStream";
 private static final String outputStreamName = "ExampleOutputStream";

 private static DataStream<String>
 createSourceFromStaticConfig(StreamExecutionEnvironment env) {
  Properties inputProperties = new Properties();
  inputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);
  inputProperties.setProperty(ConsumerConfigConstants.STREAM_INITIAL_POSITION,
 "LATEST");

  return env.addSource(new FlinkKinesisConsumer<>(inputStreamName, new
 SimpleStringSchema(), inputProperties));
 }

 private static DataStream<String>
 createSourceFromApplicationProperties(StreamExecutionEnvironment env)
   throws IOException {
  Map<String, Properties> applicationProperties =
 KinesisAnalyticsRuntime.getApplicationProperties();
  return env.addSource(new FlinkKinesisConsumer<>(inputStreamName, new
 SimpleStringSchema(),
    applicationProperties.get("ConsumerConfigProperties")));
 }

 private static FlinkKinesisFirehoseProducer<String>
 createFirehoseSinkFromStaticConfig() {
```

```
 /*
  * com.amazonaws.services.kinesisanalytics.flink.connectors.config.
  * ProducerConfigConstants
  * lists of all of the properties that firehose sink can be configured with.
  */

 Properties outputProperties = new Properties();
 outputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);

 FlinkKinesisFirehoseProducer<String> sink = new
FlinkKinesisFirehoseProducer<>(outputStreamName,
   new SimpleStringSchema(), outputProperties);
 ProducerConfigConstants config = new ProducerConfigConstants();
 return sink;
}

private static FlinkKinesisFirehoseProducer<String>
createFirehoseSinkFromApplicationProperties() throws IOException {
 /*
  * com.amazonaws.services.kinesisanalytics.flink.connectors.config.
  * ProducerConfigConstants
  * lists of all of the properties that firehose sink can be configured with.
  */

 Map<String, Properties> applicationProperties =
KinesisAnalyticsRuntime.getApplicationProperties();
 FlinkKinesisFirehoseProducer<String> sink = new
FlinkKinesisFirehoseProducer<>(outputStreamName,
   new SimpleStringSchema(),
   applicationProperties.get("ProducerConfigProperties"));
 return sink;
}

public static void main(String[] args) throws Exception {
 // set up the streaming execution environment
 final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

 /*
  * if you would like to use runtime configuration properties, uncomment the
  * lines below
  * DataStream<String> input = createSourceFromApplicationProperties(env);
  */
```

```
  DataStream<String> input = createSourceFromStaticConfig(env);

  // Kinesis Firehose sink
  input.addSink(createFirehoseSinkFromStaticConfig());

  // If you would like to use runtime configuration properties, uncomment the
  // lines below
  // input.addSink(createFirehoseSinkFromApplicationProperties());

  env.execute("Flink Streaming Java API Skeleton");
 }
}
```

Pour un didacticiel complet sur l'utilisation du récepteur Kinesis Data Firehose, consultez the section called “Récepteur Kinesis Data Firehose”.

Utilisation des E/S asynchrones dans le service géré pour Apache Flink

Un opérateur d'E/S asynchrone enrichit les données de flux à l'aide d'une source de données externe telle qu'une base de données. Le service géré pour Apache Flink enrichit les événements du flux de manière asynchrone afin que les demandes puissent être groupées pour une plus grande efficacité.

Pour de plus amples informations, veuillez consulter Asynchronous I/O dans la documentation Apache Flink.

Transformation de données à l’aide d’opérateurs dans le service géré pour Apache Flink avec l'API DataStream

Pour transformer les données entrantes dans un service géré pour Apache Flink, vous devez utiliser un opérateur Apache Flink. Un opérateur Apache Flink transforme un ou plusieurs flux de données en un nouveau flux de données. Le nouveau flux de données contient des données modifiées par rapport au flux de données d'origine. Apache Flink fournit plus de 25 opérateurs de traitement de flux prédéfinis. Pour de plus amples informations, veuillez consulter Operators dans la documentation Apache Flink.

Cette rubrique contient les sections suivantes :

- Transformer les opérateurs
- Opérateur d’aggregation

Transformer les opérateurs

Voici un exemple de transformation de texte simple sur l'un des champs d'un flux de données JSON.

Ce code crée un flux de données transformé. Le nouveau flux de données contient les mêmes données que le flux d'origine, la chaîne « Company » étant ajoutée au contenu du champ `TICKER`.

```
DataStream<ObjectNode> output = input.map(
    new MapFunction<ObjectNode, ObjectNode>() {
        @Override
        public ObjectNode map(ObjectNode value) throws Exception {
            return value.put("TICKER", value.get("TICKER").asText() + " Company");
        }
    }
);
```

Opérateur d'aggregation

Voici un exemple d'opérateur d'agrégation. Le code crée un flux de données agrégé. L'opérateur crée une fenêtre variable de 5 secondes et renvoie la somme des valeurs `PRICE` des enregistrements de la fenêtre avec la même valeur `TICKER`.

```
DataStream<ObjectNode> output = input.keyBy(node -> node.get("TICKER").asText())
    .window(TumblingProcessingTimeWindows.of(Time.seconds(5)))
    .reduce((node1, node2) -> {
        double priceTotal = node1.get("PRICE").asDouble() +
 node2.get("PRICE").asDouble();
        node1.replace("PRICE", JsonNodeFactory.instance.numberNode(priceTotal));
    return node1;
});
```

Pour un exemple de code complet utilisant des opérateurs, consultez Mise en route (DataStream API). Le code source de l'application Getting Started est disponible sur Getting Started dans le référentiel GitHub Managed Service for Apache Flink Java Examples.

Suivi des événements dans le service géré pour Apache Flink à l'aide de l'API DataStream

Le service géré pour Apache Flink suit les événements à l'aide des horodatages suivants :

- Heure de traitement : fait référence à l'heure système de la machine qui exécute l'opération correspondante.
- Heure de l'événement : fait référence à l'heure à laquelle chaque événement individuel s'est produit sur son appareil producteur.
- Heure d'ingestion : fait référence à l'heure à laquelle les événements entrent dans le service géré pour Apache Flink.

Vous réglez l'heure utilisée par l'environnement de streaming en utilisant `setStreamTimeCharacteristic`:

```
env.setStreamTimeCharacteristic(TimeCharacteristic.ProcessingTime);
env.setStreamTimeCharacteristic(TimeCharacteristic.IngestionTime);
env.setStreamTimeCharacteristic(TimeCharacteristic.EventTime);
```

Pour en savoir plus, consultez Event Time dans la documentation Apache Flink.

API de table

Votre application Apache Flink utilise l'API de table Apache Flink pour interagir avec les données d'un flux à l'aide d'un modèle relationnel. Vous utilisez l'API de table pour accéder aux données à l'aide des sources de table, puis vous utilisez les fonctions de table pour transformer et filtrer les données de table. Vous pouvez transformer et filtrer les données de table à l'aide de fonctions d'API ou de commandes SQL.

Cette section contient les rubriques suivantes :

- Connecteurs de l'API de table : ces composants déplacent les données entre votre application et les sources de données et destinations externes.
- Attributs temporels de l'API de table : cette rubrique décrit comment le service géré pour Apache Flink suit les événements lors de l'utilisation de l'API de table.

Connecteurs de l'API de table

Dans le modèle de programmation Apache Flink, les connecteurs sont des composants que votre application utilise pour lire ou écrire des données provenant de sources externes, telles que d'autres services AWS.

Avec l'API de table Apache Flink, vous pouvez utiliser les types de connecteurs suivants :

- Sources de l'API de table : vous utilisez les connecteurs source de l'API de table pour créer des tables dans votre `TableEnvironment` à l'aide d'appels d'API ou de requêtes SQL.
- Récepteurs de l'API de table : vous utilisez des commandes SQL pour écrire des données de table dans des sources externes telles qu'une rubrique Amazon MSK ou un compartiment Amazon S3.

Sources de l'API de table

Vous créez une source de table à partir d'un flux de données. Le code suivant crée une table à partir d'une rubrique Amazon MSK :

```
//create the table
    final FlinkKafkaConsumer<StockRecord> consumer = new
 FlinkKafkaConsumer<StockRecord>(kafkaTopic, new KafkaEventDeserializationSchema(),
 kafkaProperties);
    consumer.setStartFromEarliest();
    //Obtain stream
    DataStream<StockRecord> events = env.addSource(consumer);

    Table table = streamTableEnvironment.fromDataStream(events);
```

Pour plus d'informations sur les sources de table, consultez Table & Connectors dans la documentation Apache Flink.

Récepteurs de l'API de table

Pour écrire des données de table dans un récepteur, vous créez le récepteur en SQL, puis vous exécutez le récepteur basé sur SQL sur l'objet `StreamTableEnvironment`.

L'exemple de code suivant illustre comment écrire des données de table sur un récepteur Amazon S3 :

```
final String s3Sink = "CREATE TABLE sink_table (" +
    "event_time TIMESTAMP," +
    "ticker STRING," +
    "price DOUBLE," +
    "dt STRING," +
    "hr STRING" +
    ")" +
    " PARTITIONED BY (ticker,dt,hr)" +
```

```
    " WITH" +
    "(" +
    " 'connector' = 'filesystem'," +
    " 'path' = '" + s3Path + "'," +
    " 'format' = 'json'" +
    ") ";

    //send to s3
    streamTableEnvironment.executeSql(s3Sink);
    filteredTable.executeInsert("sink_table");
```

Vous pouvez utiliser le paramètre `format` pour contrôler le format utilisé par le service géré pour Apache Flink pour écrire la sortie sur le récepteur. Pour plus d'informations sur les formats, consultez la section Formats dans la documentation Apache Flink.

Pour plus d'informations sur les récepteurs de table, consultez Table & Connectors dans la documentation Apache Flink.

Sources et récepteurs définis par l'utilisateur

Vous pouvez utiliser les connecteurs Apache Kafka existants pour envoyer des données vers et depuis d'autres services AWS, tels qu'Amazon MSK et Amazon S3. Pour interagir avec d'autres sources de données et destinations, vous pouvez définir vos propres sources et récepteurs. Pour plus d'informations, consultez User-Defined Sources and Sinks dans la documentation Apache Flink.

Attributs temporels de l'API de table

Chaque enregistrement d'un flux de données possède plusieurs horodatages qui définissent le moment où les événements liés à l'enregistrement se sont produits :

- Heure de l'événement : horodatage défini par l'utilisateur qui définit le moment où l'événement à l'origine de l'enregistrement s'est produit.
- Heure d'ingestion : heure à laquelle votre application a extrait l'enregistrement du flux de données.
- Heure de traitement : heure à laquelle votre demande a traité l'enregistrement.

Lorsque l'API de table Apache Flink crée des fenêtres basées sur des heures d'enregistrement, vous définissez lequel de ces horodatages elle utilise à l'aide de la méthode setStreamTimeCharacteristic.

Pour plus d'informations sur l'utilisation des horodatages avec l'API de table, consultez la section Time Attributes dans la documentation Apache Flink.

Utilisation de Python avec le service géré Amazon pour Apache Flink

> **Note**
>
> Si vous développez l'application Python Flink sur un nouveau Mac équipé d'une puce Apple Silicon, vous pouvez rencontrer des problèmes connus avec les dépendances Python de PyFlink 1.15. Dans ce cas, nous recommandons d'exécuter l'interpréteur Python dans Docker. Pour obtenir des instructions pas à pas, consultez PyFlink 1.15 development on Apple Silicon Mac.

La version 1.15.2 d'Apache Flink prend en charge la création d'applications à l'aide de la version 3.8 de Python, en utilisant la bibliothèque PyFlink. Pour créer une application de service géré pour Apache Flink à l'aide de Python, procédez comme suit :

- Créez le code de votre application Python sous forme de fichier texte avec une méthode `main`.
- Regroupez le fichier de code de votre application et toutes les dépendances Python ou Java dans un fichier zip, puis chargez-le dans un compartiment Amazon S3.
- Créez votre application de service géré pour Apache Flink, en spécifiant l'emplacement de votre code Amazon S3, les propriétés de l'application et les paramètres de l'application.

À un niveau élevé, l'API de table Python est un encapsuleur autour de l'API de table Java. Pour plus d'informations sur l'API de table Python, consultez Intro to the Python Table API dans la documentation Apache Flink.

Programmation de votre service géré pour Apache Flink dédié à l’application Python

Vous codez votre service géré pour l'application Apache Flink pour Python à l'aide de l'API de table Apache Flink Python. Le moteur Apache Flink traduit les instructions de l'API de table Python (exécutées dans la machine virtuelle Python) en instructions de l'API de table Java (exécutées dans la machine virtuelle Java).

Pour utiliser l'API de table Python, procédez comme suit :

- Créez une référence vers l’`StreamTableEnvironment`.

- Créez des objets `table` à partir de vos données de streaming source en exécutant des requêtes sur la référence `StreamTableEnvironment`.
- Exécutez des requêtes sur vos objets `table` pour créer des tables de sortie.
- Rédigez vos tables de sortie vers vos destinations à l'aide d'un `StatementSet`.

Pour commencer à utiliser l'API de table Python dans le service géré pour Apache Flink, consultez Mise en route avec le service géré Amazon pour Apache Flink pour Python.

Lecture et écriture de données en streaming

Pour lire et écrire des données en streaming, vous devez exécuter des requêtes SQL dans l'environnement de table.

Création d'une table

L'exemple de code suivant illustre une fonction définie par l'utilisateur qui crée une requête SQL. La requête SQL crée une table qui interagit avec un flux Kinesis :

```
def create_table(table_name, stream_name, region, stream_initpos):
   return """ CREATE TABLE {0} (
                `record_id` VARCHAR(64) NOT NULL,
                `event_time` BIGINT NOT NULL,
                `record_number` BIGINT NOT NULL,
                `num_retries` BIGINT NOT NULL,
                `verified` BOOLEAN NOT NULL
              )
              PARTITIONED BY (record_id)
              WITH (
                'connector' = 'kinesis',
                'stream' = '{1}',
                'aws.region' = '{2}',
                'scan.stream.initpos' = '{3}',
                'sink.partitioner-field-delimiter' = ';',
                'sink.producer.collection-max-count' = '100',
                'format' = 'json',
                'json.timestamp-format.standard' = 'ISO-8601'
              ) """.format(table_name, stream_name, region, stream_initpos)
```

Lecture de données en streaming

L'exemple de code suivant montre comment utiliser la requête SQL `CreateTable` précédente sur une référence d'environnement de table pour lire des données :

```
  table_env.execute_sql(create_table(input_table, input_stream, input_region,
stream_initpos))
```

Écriture de données en streaming

L'exemple de code suivant montre comment utiliser la requête SQL de l'exemple `CreateTable` pour créer une référence de table de sortie, et comment utiliser un `StatementSet` pour interagir avec les tables afin d'écrire des données dans un flux Kinesis de destination :

```
  table_result = table_env.execute_sql("INSERT INTO {0} SELECT * FROM {1}"
                    .format(output_table_name, input_table_name))
```

Lecture des propriétés d'exécution

Vous pouvez utiliser les propriétés d'exécution pour configurer votre application sans changer le code de votre application.

Vous spécifiez les propriétés de votre application de la même manière qu'avec un service géré pour Apache Flink pour une application Java. Vous pouvez spécifier des propriétés d'exécution de différentes manières :

- À l'aide de l'action CreateApplication.
- À l'aide de l'action UpdateApplication.
- En Configurant votre application à l'aide de la console.

Vous pouvez récupérer les propriétés de l'application dans le code en lisant un fichier JSON appelé `application_properties.json` créé par l'éxécution du service géré pour Apache Flink.

L'exemple de code suivant montre comment lire les propriétés d'une application à partir du fichier `application_properties.json` :

```
file_path = '/etc/flink/application_properties.json'
  if os.path.isfile(file_path):
```

```
        with open(file_path, 'r') as file:
            contents = file.read()
            properties = json.loads(contents)
```

L'exemple de code de fonction défini par l'utilisateur suivant illustre la lecture d'un groupe de propriétés à partir de l'objet des propriétés de l'application : récupère :

```
def property_map(properties, property_group_id):
   for prop in props:
       if prop["PropertyGroupId"] == property_group_id:
           return prop["PropertyMap"]
```

L'exemple de code suivant illustre la lecture d'une propriété appelée INPUT_STREAM_KEY à partir d'un groupe de propriétés renvoyé par l'exemple précédent :

```
input_stream = input_property_map[INPUT_STREAM_KEY]
```

Création du package de code de votre application

Une fois que vous avez créé votre application Python, vous regroupez votre fichier de code et ses dépendances dans un fichier zip.

Votre fichier zip doit contenir un script python avec une méthode `main` et peut éventuellement contenir les éléments suivants :

- Fichiers de code Python supplémentaires
- Code Java défini par l'utilisateur dans les fichiers JAR
- Bibliothèques Java dans des fichiers JAR

> **Note**
>
> Le fichier zip de votre application doit contenir toutes les dépendances de votre application. Vous ne pouvez pas référencer des bibliothèques provenant d'autres sources pour votre application.

Création de votre application de service géré pour Apache Flink Python

Spécification de vos fichiers de code

Une fois que vous avez créé le package de code de votre application, vous le chargez dans un compartiment Amazon S3. Vous pouvez ensuite créer votre application à l'aide de la console ou de l'action CreateApplication.

Lorsque vous créez votre application à l'aide de l'action CreateApplication, vous spécifiez les fichiers de code et les archives de votre fichier zip à l'aide d'un groupe de propriétés d'application spécial appelé `kinesis.analytics.flink.run.options`. Vous pouvez définir les types de fichiers suivants :

- python : fichier texte contenant une méthode principale de Python.
- jarfile : fichier JAR Java contenant des fonctions Java définies par l'utilisateur.
- pyFiles : fichier de ressources Python contenant les ressources à utiliser par l'application.
- pyArchives : fichier zip contenant les fichiers de ressources de l'application.

Pour plus d'informations sur les types de fichiers de code Python d'Apache Flink, consultez la section Command Line Usage dans la documentation Apache Flink.

> **Note**
>
> Le service géré pour Apache Flink ne prend pas en charge les types de fichiers `pyModule`, `pyExecutable` ou `pyRequirements`. L'ensemble du code, des exigences et des dépendances doivent se trouver dans votre fichier zip. Vous ne pouvez pas spécifier les dépendances à installer à l'aide de pip.

L'exemple d'extrait de code JSON suivant montre comment spécifier l'emplacement des fichiers dans le fichier zip de votre application :

```
"ApplicationConfiguration": {
    "EnvironmentProperties": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "kinesis.analytics.flink.run.options",
```

```
        "PropertyMap": {
          "python": "MyApplication/main.py",
          "jarfile": "MyApplication/lib/myJarFile.jar",
          "pyFiles": "MyApplication/lib/myDependentFile.py",
          "pyArchives": "MyApplication/lib/myArchive.zip"
        }
      },
```

Surveillance de votre application de service géré pour Apache Flink Python

Utilisez le journal CloudWatch de votre application pour surveiller votre application de service géré pour Apache Flink Python.

Le service géré pour Apache Flink enregistre les messages suivants pour les applications Python :

- Messages écrits sur la console à l'aide de `print()` dans la méthode `main` de l'application.
- Messages envoyés dans le cadre de fonctions définies par l'utilisateur à l'aide du package `logging`. L'exemple de code suivant illustre l'écriture dans le journal des applications à partir d'une fonction définie par l'utilisateur :

  ```
  import logging

  @udf(input_types=[DataTypes.BIGINT()], result_type=DataTypes.BIGINT())
  def doNothingUdf(i):
      logging.info("Got {} in the doNothingUdf".format(str(i)))
      return i
  ```

- Messages d'erreur émis par l'application.

  Si l'application génère une exception dans la fonction `main`, elle apparaîtra dans les journaux de votre application.

  L'exemple suivant illustre une entrée de journal pour une exception émise à partir du code Python :

  ```
  2021-03-15 16:21:20.000   -------------------------- Python Process Started
   --------------------------
  2021-03-15 16:21:21.000   Traceback (most recent call last):
  2021-03-15 16:21:21.000   "  File ""/tmp/flink-
  web-6118109b-1cd2-439c-9dcd-218874197fa9/flink-web-upload/4390b233-75cb-4205-
  a532-441a2de83db3_code/PythonKinesisSink/PythonUdfUndeclared.py"", line 101, in
   <module>"
  ```

```
2021-03-15 16:21:21.000       main()
2021-03-15 16:21:21.000   "  File ""/tmp/flink-
web-6118109b-1cd2-439c-9dcd-218874197fa9/flink-web-upload/4390b233-75cb-4205-
a532-441a2de83db3_code/PythonKinesisSink/PythonUdfUndeclared.py"", line 54, in main"
2021-03-15 16:21:21.000   "    table_env.register_function(""doNothingUdf"",
 doNothingUdf)"
2021-03-15 16:21:21.000   NameError: name 'doNothingUdf' is not defined
2021-03-15 16:21:21.000   -------------------------- Python Process Exited
 --------------------------
2021-03-15 16:21:21.000   Run python process failed
2021-03-15 16:21:21.000   Error occurred when trying to start the job
```

> **Note**
>
> En raison de problèmes de performances, nous vous recommandons de n'utiliser que des messages de journal personnalisés lors du développement de l'application.

Interrogation des journaux à l'aide de CloudWatch Insight

La requête CloudWatch Insights suivante recherche les journaux créés par le point d'entrée Python lors de l'exécution de la fonction principale de votre application :

```
fields @timestamp, message
| sort @timestamp asc
| filter logger like /PythonDriver/
| limit 1000
```

Propriétés d’exécution dans le service géré pour Apache Flink

Vous pouvez utiliser les propriétés d'exécution pour configurer votre application sans recompiler le code de votre application.

Cette rubrique contient les sections suivantes :

- Utilisation des propriétés d'exécution dans la console
- Utilisation des propriétés d'exécution dans l’interface CLI
- Accès aux propriétés d’exécution d’une application de service géré pour Apache Flink

Utilisation des propriétés d'exécution dans la console

Vous pouvez ajouter, mettre à jour ou supprimer des propriétés d'exécution dans votre application de service géré pour Apache Flink à l'aide de la console.

> **Note**
>
> Vous ne pouvez pas ajouter de propriétés d'exécution lorsque vous créez une application dans la console du service géré pour Apache Flink.

Mettre à jour les propriétés d’exécution d’une application de service géré pour l'application Apache Flink

1. Ouvrez la console du service géré pour Apache Flink à l’adresse https://console.aws.amazon.com/flink
2. Choisissez votre application de service géré pour Apache Flink. Choisissez Détails de l'application.
3. Sur la page de votre application, choisissez Configurer.
4. Développez la section Propriétés.
5. Utilisez les commandes de la section Propriétés pour définir un groupe de propriétés avec des paires clé-valeur. Utilisez ces commandes pour ajouter, mettre à jour ou supprimer des groupes de propriétés et des propriétés d'exécution.
6. Choisissez Mettre à jour.

Utilisation des propriétés d'exécution dans l’interface CLI

Vous pouvez ajouter, mettre à jour ou supprimer des propriétés d'exécution à l'aide de l’interface AWS CLI.

Cette section inclut des exemples de demandes d'actions d'API pour configurer les propriétés d’exécution pour une application. Pour plus d'informations sur l'utilisation d'un fichier JSON comme entrée pour une action d'API, veuillez consulter Exemple de code pour d’API pour le service géré pour Apache Flink.

> **Note**
>
> Remplacez l'exemple d'ID de compte (*012345678901*) dans les exemples suivants par votre ID de compte.

Ajout de propriétés d'exécution lors de la création d'une application

L'exemple de demande d'action `CreateApplication` suivant ajoute deux groupes de propriétés d'exécution (`ProducerConfigProperties` et `ConsumerConfigProperties`) lorsque vous créez une application :

```
{
    "ApplicationName": "MyApplication",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "java-getting-started-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
```

```
            }
         ]
      }
   }
}
```

Ajout et mise à jour des propriétés d'exécution dans une application existante

L'exemple de demande d'action `UpdateApplication` suivant ajoute ou met à jour les propriétés d'exécution d'une application existante :

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 2,
  "ApplicationConfigurationUpdate": {
    "EnvironmentPropertyUpdates": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap" : {
            "flink.stream.initpos" : "LATEST",
            "aws.region" : "us-west-2",
            "AggregationEnabled" : "false"
          }
        },
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap" : {
            "aws.region" : "us-west-2"
          }
        }
      ]
    }
  }
}
```

> **Note**
>
> Si vous utilisez une clé qui n'a aucune propriété d'exécution correspondante dans un groupe de propriétés, le service géré pour Apache Flink ajoute la paire clé-valeur en tant que

nouvelle propriété. Si vous utilisez une clé pour une propriété d'exécution existante dans un groupe de propriétés, le service géré pour Apache Flink met à jour la valeur de la propriété.

Suppression des propriétés d'exécution

L'exemple de demande d'action `UpdateApplication` suivant supprime toutes les propriétés d'exécution et tous les groupes de propriétés d'une application existante :

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 3,
  "ApplicationConfigurationUpdate": {
    "EnvironmentPropertyUpdates": {
      "PropertyGroups": []
    }
  }
}
```

Important

Si vous omettez un groupe de propriétés existant ou une clé de propriété existante dans un groupe de propriétés, ce groupe de propriétés ou cette propriété est supprimé.

Accès aux propriétés d’exécution d’une application de service géré pour Apache Flink

Vous pouvez récupérer les propriétés d'exécution dans le code de votre application Java à l'aide de la méthode `KinesisAnalyticsRuntime.getApplicationProperties()` statique, qui renvoie un objet `Map<String, Properties>`.

L'exemple de code Java suivant permet de récupérer les propriétés d'exécution pour votre application :

```
Map<String, Properties> applicationProperties =
KinesisAnalyticsRuntime.getApplicationProperties();
```

Vous pouvez récupérer un groupe de propriétés (sous forme d'objet `Java.Util.Properties`) comme suit :

```
Properties consumerProperties = applicationProperties.get("ConsumerConfigProperties");
```

Vous configurez généralement une source ou un récepteur Apache Flink en transmettant l'objet `Properties` sans avoir à récupérer les propriétés individuelles. L'exemple de code suivant montre comment créer une source Flink en transmettant un objet `Properties` extrait des propriétés d'exécution :

```
private static FlinkKinesisProducer<String> createSinkFromApplicationProperties()
 throws IOException {
  Map<String, Properties> applicationProperties =
 KinesisAnalyticsRuntime.getApplicationProperties();
  FlinkKinesisProducer<String> sink = new FlinkKinesisProducer<String>(new
 SimpleStringSchema(),
    applicationProperties.get("ProducerConfigProperties"));

  sink.setDefaultStream(outputStreamName);
  sink.setDefaultPartition("0");
  return sink;
}
```

Pour un exemple de code complet utilisant des propriétés d'exécution, consultez Mise en route (DataStream API). Le code source de l'application Getting Started est disponible sur Getting Started dans le référentiel GitHub Managed Service for Apache Flink Java Examples.

Implémentation de la tolérance aux pannes dans le service géré pour Apache Flink

Le point de contrôle est la méthode utilisée pour implémenter la tolérance aux pannes dans le service géré Amazon pour Apache Flink. Un point de contrôle est une sauvegarde à jour d'une application en cours d'exécution qui est utilisée pour effectuer une restauration immédiate en cas d'interruption ou de basculement imprévu d'une application.

Pour plus de détails sur les points de contrôle dans les applications Apache Flink, consultez Checkpoints dans la documentation Apache Flink.

Un instantané est une sauvegarde créée et gérée manuellement de l'état de l'application. Les instantanés vous permettent de restaurer l'état antérieur de votre application en appelant `UpdateApplication`. Pour de plus amples informations, veuillez consulter Gestion des sauvegardes d'application à l'aide d'instantanés.

Si le point de contrôle est activé pour votre application, le service assure la tolérance aux pannes en créant et en chargeant des sauvegardes des données de l'application en cas de redémarrage inattendu de l'application. Ces redémarrages inattendus d'application peuvent être provoqués par des redémarrages de tâche inattendus, des échecs d'instance, etc. Cela donne à l'application la même sémantique qu'une exécution sans échec lors de ces redémarrages.

Si les instantanés sont activés pour l'application et configurés à l'aide de la configuration ApplicationRestoreConfiguration de l'application, le service fournit une sémantique de traitement unique lors des mises à jour de l'application ou lors de la mise à l'échelle ou de la maintenance liés au service.

Configuration du point de contrôle dans le service géré pour Apache Flink

Vous pouvez configurer le comportement de point de contrôle de votre application. Vous pouvez définir si elle conserve l'état de point de contrôle, à quelle fréquence elle enregistre son état dans les points de contrôle et l'intervalle minimum entre la fin d'une opération de point de contrôle et le début d'une autre.

Vous configurez les paramètres suivants à l'aide des opérations d'API `CreateApplication` ou `UpdateApplication` :

- `CheckpointingEnabled` : indique si le point de contrôle est activé dans l'application.
- `CheckpointInterval` : contient le temps en millisecondes entre les opérations de point de contrôle (persistance).
- `ConfigurationType` : définissez cette valeur sur `DEFAULT` pour utiliser le comportement de point de contrôle par défaut. Définissez cette valeur sur `CUSTOM` pour configurer d'autres valeurs.

> **Note**
>
> Le comportement du point de contrôle par défaut est le suivant :
>
> - CheckpointingEnabled : true
> - CheckpointInterval : 60000
> - MinPauseBetweenCheckpoints : 5000

Si ConfigurationType a défini sur `DEFAULT`, les valeurs précédentes seront utilisées, même si d'autres valeurs leur sont affectées à l'aide de l'AWS Command Line Interface ou si des valeurs sont définies dans le code d'application.

> **Note**
>
> À partir de Flink 1.15, le service géré pour Apache Flink utilise `stop-with-savepoint` lors de la création automatique d'instantanés, c'est-à-dire lors de la mise à jour, de la mise à l'échelle ou de l'arrêt de l'application.

- `MinPauseBetweenCheckpoints` : durée minimale en millisecondes entre la fin d'une opération de point de contrôle et le début d'une autre. La définition de cette propriété empêche l'application de créer un point de contrôle continu lorsque l'opération de contrôle dure plus de temps que `CheckpointInterval`.

Exemples d'API de création de points de contrôle

Cette section inclut des exemples de demandes d'actions d'API pour configurer les points de contrôle pour une application. Pour plus d'informations sur l'utilisation d'un fichier JSON comme entrée pour une action d'API, veuillez consulter Exemple de code pour d'API pour le service géré pour Apache Flink.

Configurer les points de contrôle pour une nouvelle application

L'exemple de demande d'action `CreateApplication` suivant configure les points de contrôle lorsque vous créez une application :

```
{
   "ApplicationName": "MyApplication",
   "RuntimeEnvironment":"FLINK-1_15",
   "ServiceExecutionRole":"arn:aws:iam::123456789123:role/myrole",
   "ApplicationConfiguration": {
      "ApplicationCodeConfiguration":{
      "CodeContent":{
        "S3ContentLocation":{
          "BucketARN":"arn:aws:s3:::mybucket",
          "FileKey":"myflink.jar",
          "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
```

```
        }
      },
      "FlinkApplicationConfiguration": {
         "CheckpointConfiguration": {
            "CheckpointingEnabled": "true",
            "CheckpointInterval": 20000,
            "ConfigurationType": "CUSTOM",
            "MinPauseBetweenCheckpoints": 10000
         }
      }
}
```

Désactiver les points de contrôle pour une nouvelle application

L'exemple de demande d'action `CreateApplication` suivant désactive les points de contrôle lorsque vous créez une application :

```
{
   "ApplicationName": "MyApplication",
   "RuntimeEnvironment":"FLINK-1_15",
   "ServiceExecutionRole":"arn:aws:iam::123456789123:role/myrole",
   "ApplicationConfiguration": {
      "ApplicationCodeConfiguration":{
      "CodeContent":{
        "S3ContentLocation":{
          "BucketARN":"arn:aws:s3:::mybucket",
          "FileKey":"myflink.jar",
          "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "FlinkApplicationConfiguration": {
         "CheckpointConfiguration": {
            "CheckpointingEnabled": "false"
         }
      }
}
```

Configurer les points de contrôle pour une application existante

L'exemple de demande d'action `UpdateApplication` suivant configure les points de contrôle pour une application existante :

```
{
   "ApplicationName": "MyApplication",
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "CheckpointConfigurationUpdate": {
            "CheckpointingEnabledUpdate": true,
            "CheckpointIntervalUpdate": 20000,
            "ConfigurationTypeUpdate": "CUSTOM",
            "MinPauseBetweenCheckpointsUpdate": 10000
         }
      }
   }
}
```

Désactiver les points de contrôle pour une application existante

L'exemple de demande d'action `UpdateApplication` suivant désactive les points de contrôle pour une application existante :

```
{
   "ApplicationName": "MyApplication",
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "CheckpointConfigurationUpdate": {
            "CheckpointingEnabledUpdate": false,
            "CheckpointIntervalUpdate": 20000,
            "ConfigurationTypeUpdate": "CUSTOM",
            "MinPauseBetweenCheckpointsUpdate": 10000
         }
      }
   }
}
```

Gestion des sauvegardes d'application à l'aide d’instantanés

Un instantané est l'implémentation d'un point de sauvegarde Apache Flink par le service géré pour Apache Flink. Un instantané est une sauvegarde de l'état de l'application déclenchée, créée et gérée par un utilisateur ou un service. Pour des informations sur les points de sauvegarde Apache Flink, consultez Savepoints dans la documentation Apache Flink. À l'aide des instantanés, vous pouvez redémarrer une application à partir d'un instantané spécifique de l'état de l'application.

> **Note**
>
> Nous recommandons que votre application crée un instantané plusieurs fois par jour pour redémarrer correctement avec des données d'état correctes. La fréquence correcte pour vos instantanés dépend de la logique métier de votre application. La prise fréquente d'instantané vous permet de récupérer des données plus récentes, mais cela augmente les coûts et nécessite davantage de ressources système.

Dans le service géré pour Apache Flink, vous pouvez gérer les instantanés à l'aide des actions API suivantes :

- `CreateApplicationSnapshot`
- `DeleteApplicationSnapshot`
- `DescribeApplicationSnapshot`
- `ListApplicationSnapshots`

Pour connaître la limite du nombre d'instantanés par application, consultez Quota. Si votre application atteint la limite d'intantanés, la création manuelle d'un instantané échoue avec une `LimitExceededException`.

Le service géré pour Apache Flink ne supprime jamais les instantanés. Vous devez supprimer les instantanés manuellement à l'aide de l'action `DeleteApplicationSnapshot`.

Pour charger un instantané enregistré de l'état de l'application lors du démarrage d'une application, utilisez le paramètre `ApplicationRestoreConfiguration` de l'action `StartApplication` ou `UpdateApplication`.

Cette rubrique contient les sections suivantes :

- Création automatique d'instantanés
- Restauration à partir d'un instantané contenant des données d'état incompatibles
- Exemples d'API d'instantané

Création automatique d'instantanés

Si `SnapshotsEnabled` est défini sur `true` dans ApplicationSnapshotConfiguration de l'application, le service géré pour Apache Flink crée et utilise automatiquement des instantanés lorsque

l'application est mise à jour, mise à l'échelle ou arrêtée afin de fournir une sémantique de traitement unique.

> **Note**
>
> La définition de `ApplicationSnapshotConfiguration::SnapshotsEnabled` sur `false` entraînera une perte de données lors des mises à jour de l'application.

> **Note**
>
> Le service géré pour Apache Flink déclenche des points de sauvegarde intermédiaires lors de la création automatique d'instantanés. Pour la version 1.15 ou ultérieure de Flink, les points de sauvegarde intermédiaires ne provoquent plus d'effets secondaires. Consultez Triggering savepoints

Les instantanés créés automatiquement présentent les qualités suivantes :

- L'instantané est géré par le service, mais vous pouvez le voir à l'aide de l'action ListApplicationSnapshots. Les instantanés créés automatiquement sont pris en compte dans votre limite d'instantanés.
- Si votre application dépasse la limite d'instantanés, les instantanés créés manuellement échoueront, mais le service géré pour Apache Flink créera toujours des instantanés lorsque l'application sera mise à jour, mise à l'échelle ou arrêtée. Vous devez supprimer manuellement les instantanés à l'aide de l'action DeleteApplicationSnapshot avant de créer manuellement d'autres instantanés.

Restauration à partir d'un instantané contenant des données d'état incompatibles

Les instantanés contenant des informations sur les opérateurs, la restauration des données d'état à partir d'un instantané d'un opérateur qui a changé depuis la version précédente de l'application peut avoir des résultats inattendus. Une application rencontrera un échec si elle tente de restaurer les données d'état à partir d'un instantané qui ne correspond pas à l'opérateur actuel. De plus, l'application sera bloquée à l'état `STOPPING` ou `UPDATING`.

Pour autoriser une application à effectuer une restauration à partir d'un instantané contenant des données d'état incompatibles, définissez le paramètre `AllowNonRestoredState` de FlinkRunConfiguration sur `true` à l'aide de l'action UpdateApplication.

Vous constaterez le comportement suivant lorsqu'une application est restaurée à partir d'un instantané obsolète :

- Opérateur ajouté : si un nouvel opérateur est ajouté, le point de sauvegarde ne contient aucune donnée d'état pour le nouvel opérateur. Aucun défaut ne se produira et il n'est pas nécessaire de définir `AllowNonRestoredState`.
- Opérateur supprimé : si un opérateur existant est supprimé, le point de sauvegarde contient les données d'état de l'opérateur manquant. Une erreur se produira à moins que `AllowNonRestoredState` ne soit défini sur `true`.
- Modifié par l'opérateur : si des modifications compatibles sont apportées, telles que le remplacement du type d'un paramètre par un type compatible, l'application peut effectuer une restauration à partir de l'instantané obsolète. Pour plus d'informations sur la restauration à partir d'instantanés, consultez la section Savepoints de la documentation Apache Flink. Une application qui utilise Apache Flink version 1.8 ou ultérieure peut éventuellement être restaurée à partir d'un instantané avec un schéma différent. Une application qui utilise Apache Flink version 1.6 ne peut pas être restaurée. Pour les récepteurs de validation en deux phases, nous recommandons d'utiliser un instantané du système (SwS) au lieu d'un instantané créé par l'utilisateur (CreateApplicationSnapshot).

  Pour Flink, le service géré pour Apache Flink déclenche des points de sauvegarde intermédiaires lors de la création automatique d'instantanés. À partir de la version 1.15 de Flink, les points de sauvegarde intermédiaires ne provoquent plus d'effets secondaires. Consultez Triggering savepoints.

Si vous devez reprendre une application incompatible avec les données de point de sauvegarde existantes, nous vous recommandons d'ignorer la restauration à partir de l'instantané en définissant le paramètre `ApplicationRestoreType` de l'action StartApplication sur `SKIP_RESTORE_FROM_SNAPSHOT`.

Pour plus d'informations sur la façon dont Apache Flink gère les données d'état incompatibles, voir State Schema Evolution dans la documentation Apache Flink.

Exemples d'API d’instantané

Cette section inclut des exemples de demandes d'actions d'API pour utiliser des instantanés avec une application. Pour plus d'informations sur l'utilisation d'un fichier JSON comme entrée pour une action d'API, veuillez consulter Exemple de code pour d’API pour le service géré pour Apache Flink.

Activer les instantanés pour une application

L'exemple de demande suivant pour l'action `UpdateApplication` active les instantanés pour une application :

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "ApplicationSnapshotConfigurationUpdate": {
         "SnapshotsEnabledUpdate": "true"
       }
    }
}
```

Créer un instantané

L'exemple de code de demande suivant pour l'action `CreateApplicationSnapshot` crée un instantané de l’état actuel de l’application :

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MyCustomSnapshot"
}
```

Répertorier les instantanés pour une application

L'exemple de code de demande suivant pour l'action `ListApplicationSnapshots` répertorie les 50 premiers instantanés de l’état actuel de l’application :

```
{
   "ApplicationName": "MyApplication",
   "Limit": 50
}
```

Répertorier les informations sur l'instantané d'une application

L'exemple de demande suivant pour l'action DescribeApplicationSnapshot répertorie les informations spécifiques à un instantané d'application :

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MyCustomSnapshot"
}
```

Supprimer un instantané

L'exemple de demande d'action DeleteApplicationSnapshot suivant supprime un instantané précédemment enregistré. Vous pouvez obtenir la valeur SnapshotCreationTimestamp en utilisant ListApplicationSnapshots ou DeleteApplicationSnapshot :

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MyCustomSnapshot",
   "SnapshotCreationTimestamp": 12345678901.0,
}
```

Redémarrer une application à l'aide d'un instantané nommé

L'exemple de demande d'action StartApplication suivant démarre l'application en utilisant l'état enregistré à partir d'un instantané spécifique :

```
{
   "ApplicationName": "MyApplication",
   "RunConfiguration": {
      "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_CUSTOM_SNAPSHOT",
         "SnapshotName": "MyCustomSnapshot"
      }
   }
}
```

Redémarrer une application à l'aide de l'instantané le plus récent

L'exemple de demande d'action StartApplication suivant démarre l'application en utilisant l'instantané le plus récent :

```
{
   "ApplicationName": "MyApplication",
   "RunConfiguration": {
      "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
      }
   }
}
```

Redémarrer une application sans instantané

L'exemple de demande d'action `StartApplication`suivant démarre l'application sans charger l'état de l'application, même si un instantané est présent :

```
{
   "ApplicationName": "MyApplication",
   "RunConfiguration": {
      "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "SKIP_RESTORE_FROM_SNAPSHOT"
      }
   }
}
```

Mise à l’échelle d’application dans le service géré pour Apache Flink

Vous pouvez configurer l'exécution parallèle des tâches et l'allocation de ressources pour le service géré Amazon pour Apache Flink afin d’implémenter la mise à l’échelle. Pour des informations sur la manière dont Apache Flink planifie les instances parallèles des tâches, veuillez consulter Parallel Execution dans la documentation Apache Flink.

Rubriques

- Configuration du parallélisme des applications et du KPU ParallelismPer
- Allocation d'unités de traitement Kinesis
- Mise à jour du parallélisme de votre application
- Mise à l’échelle automatique

Configuration du parallélisme des applications et du KPU ParallelismPer

Vous configurez l'exécution parallèle des tâches votre application de service géré pour Apache Flink (telles que la lecture depuis une source ou l'exécution d'un opérateur) à l'aide des propriétés `ParallelismConfiguration` suivantes :

- `Parallelism` : utilisez cette propriété pour définir le parallélisme par défaut de l'application Apache Flink. Tous les opérateurs, sources et récepteurs s'exécutent avec ce parallélisme, sauf s'ils sont remplacés dans le code de l'application. La valeur par défaut est 1, et la valeur maximale est 256.
- `ParallelismPerKPU` : utilisez cette propriété pour définir le nombre de tâches parallèles qui peuvent être planifiées par unité de traitement Kinesis (KPU) de votre application. La valeur par défaut est 1, et la valeur maximale est 8. Pour les applications comportant des opérations de blocage (par exemple, des E/S), une valeur plus élevée de `ParallelismPerKPU` entraîne une utilisation complète des ressources KPU.

> **Note**
>
> La limite pour `Parallelism` est égale à `ParallelismPerKPU` fois la limite pour les KPU (64 par défaut). La limite de KPU peut être augmentée en demandant une augmentation de limite. Pour des instructions pour demander une augmentation de cette limite, consultez « Pour demander une augmentation de limite » dans Service Quotas.

Pour des informations sur la définition du parallélisme des tâches pour un opérateur spécifique, consultez la section Setting the Parallelism: Operator dans la documentation Apache Flink.

Allocation d'unités de traitement Kinesis

Le service géré pour Apache Flink fournit la capacité en KPU. qUne seule unité KPU vous fournit 1 vCPU et 4 Go de mémoire. Pour chaque unité KPU allouée, 50 Go de stockage des applications en cours d'exécution sont également fournis.

Le service géré pour Apache Flink calcule les KPU nécessaires pour exécuter votre application à l'aide des propriétés `Parallelism` et `ParallelismPerKPU`, comme suit :

```
Allocated KPUs for the application = Parallelism/ParallelismPerKPU
```

Le service géré pour Apache Flink fournit rapidement des ressources à vos applications en réponse aux pics de débit ou d'activité de traitement. Il supprime progressivement les ressources de votre application une fois le pic d'activité passé. Pour désactiver l'allocation automatique des ressources, définissez la valeur `AutoScalingEnabled` sur `false`, comme décrit plus loin dans Mise à jour du parallélisme de votre application.

La limite par défaut est de 64 pour les KPU pour votre application. Pour des instructions pour demander une augmentation de cette limite, consultez « Pour demander une augmentation de limite » dans Service Quotas.

> **Note**
>
> Un KPU supplémentaire est facturé à des fins d'orchestration. Pour plus d'informations, consultez Tarification du service géré pour Apache Flink.

Mise à jour du parallélisme de votre application

Cette section contient des exemples de demande d'action d'API qui définissent le parallélisme d'une application. Pour plus d'exemples et d'instructions sur l'utilisation de blocs de requête avec des actions d'API, consultez Exemple de code pour d'API pour le service géré pour Apache Flink.

L'exemple de demande d'action `CreateApplication` suivant définit le parallélisme lorsque vous créez une application :

```
{
   "ApplicationName": "string",
   "RuntimeEnvironment":"FLINK-1_15",
   "ServiceExecutionRole":"arn:aws:iam::123456789123:role/myrole",
   "ApplicationConfiguration": {
      "ApplicationCodeConfiguration":{
      "CodeContent":{
         "S3ContentLocation":{
            "BucketARN":"arn:aws:s3:::mybucket",
            "FileKey":"myflink.jar",
            "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
            }
         },
      "CodeContentType":"ZIPFILE"
   },
```

```
      "FlinkApplicationConfiguration": {
         "ParallelismConfiguration": {
            "AutoScalingEnabled": "true",
            "ConfigurationType": "CUSTOM",
            "Parallelism": 4,
            "ParallelismPerKPU": 4
         }
      }
   }
}
```

L'exemple de demande d'action `UpdateApplication` suivant définit le parallélisme pour une application existante :

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "ParallelismConfigurationUpdate": {
            "AutoScalingEnabledUpdate": "true",
            "ConfigurationTypeUpdate": "CUSTOM",
            "ParallelismPerKPUUpdate": 4,
            "ParallelismUpdate": 4
         }
      }
   }
}
```

L'exemple de demande d'action `UpdateApplication` suivant désactive le parallélisme pour une application existante :

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "ParallelismConfigurationUpdate": {
            "AutoScalingEnabledUpdate": "false"
         }
      }
   }
```

```
}
```

Mise à l'échelle automatique

Le service géré pour Apache Flink adapte de manière élastique le parallélisme de votre application pour s'adapter au débit de données de votre source et à la complexité de votre opérateur dans la plupart des scénarios. Le service géré pour Apache Flink surveille l'utilisation des ressources (processeur) de votre application et adapte de manière élastique le parallélisme de votre application à la hausse ou à la baisse en conséquence :

- Votre application augmente (augmente le parallélisme) si la CloudWatch métrique `containerCPUUtilization` est supérieure ou égale à 75 % pendant 15 minutes. Cela signifie que l'action `ScaleUp` est déclenchée lorsqu'il existe 15 points de données consécutifs avec une période d'une minute égale ou supérieure à 75 %.
- Votre application réduit (diminue le parallélisme) lorsque l'utilisation de votre processeur reste inférieure à 10 % pendant six heures. Cela signifie que l'action `ScaleDown` est déclenchée lorsqu'il existe 360 points de données consécutifs avec une période d'une minute égale ou supérieure à 10 %.

> **Note**
>
> Une `containerCPUUtilization` max sur une période de plus d'une minute peut être référencée pour trouver la corrélation avec un point de données utilisé pour l'action de mise à l'échelle, mais il n'est pas nécessaire de refléter le moment exact où l'action est déclenchée.

Le service géré pour Apache Flink ne réduira pas la valeur `CurrentParallelism` de votre application à un niveau inférieur au paramètre `Parallelism` de votre application.

Lorsque le service géré pour Apache Flink met à l'échelle votre application, elle passe à l'état `AUTOSCALING`. Vous pouvez vérifier l'état actuel de votre candidature à l'aide ListApplicationsdes actions DescribeApplicationou. Pendant que le service fait évoluer votre application, la seule action d'API valide que vous pouvez utiliser est celle StopApplicationdont le `Force` paramètre est défini sur`true`.

Vous pouvez utiliser la propriété `AutoScalingEnabled` (partie de `FlinkApplicationConfiguration`) pour activer ou désactiver le comportement de l'autoscaling.

Votre compte AWS est débité pour les KPU fournis par le service géré pour Apache Flink, en fonction des paramètres `parallelism` et `parallelismPerKPU` de votre application. Un pic d'activité augmente les coûts de votre service géré pour Apache Flink.

Pour des d'informations sur la tarification, consultez Tarification du service géré pour Apache Flink.

Notez les informations suivantes à propos de la mise à l'échelle de l'application :

- La mise à l'échelle automatique est activée par défaut.
- La mise à l'échelle ne s'applique pas aux blocs-notes Studio. Toutefois, si vous déployez un bloc-notes Studio en tant qu'application à état durable, la mise à l'échelle s'appliquera à l'application déployée.
- La limite par défaut de votre application est de 64 KPU. Pour plus d'informations, consultez Quota.
- Lorsque la mise à l'échelle automatique met à jour le parallélisme des applications, celles-ci subissent des interruptions. Pour éviter ces interruptions, procédez comme suit :
  - Désactiver la mise à l'échelle automatique
  - Configurez `parallelism` et utilisez l'UpdateApplicationaction `parallelismPerKPU` de votre application. Pour plus d'informations sur la définition des paramètres de parallélisme de votre application, consultez the section called “Mise à jour du parallélisme de votre application”.
  - Surveillez régulièrement l'utilisation des ressources de votre application pour vérifier qu'elle dispose des paramètres de parallélisme adaptés à sa charge de travail. Pour des informations sur la surveillance de l'allocation des ressources, consultez the section called “Métriques et dimensions dans le service géré pour Apache Flink”.

Considérations relatives à maxParallelism

- La logique de mise à l'échelle automatique empêchera la mise à l'échelle d'une tâche Flink jusqu'à un parallélisme susceptible de provoquer des interférences avec la tâche et l'opérateur `maxParallelism`. Par exemple, s'il s'agit d'une tâche simple avec uniquement une source et un récepteur dont la source a `maxParallelism` 16 et le `sink` 8, nous n'allons pas redimensionner automatiquement la tâche au-dessus de 8.
- Si `maxParallelism` n'est pas défini pour une tâche, Flink utilisera par défaut 128. Par conséquent, si vous pensez qu'une tâche devra être exécutée avec un parallélisme supérieur à 128, vous devrez définir ce nombre pour votre application.
- Si vous vous attendez à voir votre tâche se mettre à l'échelle automatiquement, mais que ce n'est pas le cas, assurez-vous que vos valeurs `maxParallelism` le permettent.

Pour plus d'informations, consultez Surveillance améliorée et mise à l’échelle automatique pour Apache Flink

Pour un exemple, voir kda-flink-app-autoscaling.

Utilisation du balisage

Cette section décrit comment ajouter des balises de métadonnées clé-valeur à des applications de service géré pour Apache Flink. Ces balises peuvent être utilisées aux fins suivantes :

- Déterminer la facturation des applications individuelles de service géré pour Apache Flink. Pour plus d’informations, consultez Utilisation des balises de répartition des coûts dans le Guide Facturation et gestion des coûts.
- Contrôle de l'accès aux ressources d’application basé sur des balises. Pour de plus amples informations, veuillez consulter Contrôle de l’accès à l’aide de balises dans le AWS Identity and Access ManagementGuide de l’utilisateur.
- À des fins définies par l'utilisateur. Vous pouvez définir des fonctionnalités d'application en fonction de la présence de balises utilisateur.

Notez les informations suivantes concernant le balisage :

- Le nombre maximal de balises d'application inclut les balises système. Le nombre maximal de balises d'application définies par l'utilisateur est de 50.
- Si une action inclut une liste de balises qui comporte des valeurs `Key` en double, le service émet une `InvalidArgumentException`.

Cette rubrique contient les sections suivantes :

- Ajout de balises lorsqu'une application est créée
- Ajout ou mise à jour des balises pour une application existante
- Répertorier les balises d'une application
- Suppression des balises d'une application

Ajout de balises lorsqu'une application est créée

Vous pouvez ajouter des balises lors de la création d'une application à l'aide du paramètre `tags` de l'action CreateApplication.

L'exemple de demande suivant illustre le nœud `Tags` pour une demande `CreateApplication` :

```
"Tags": [
    {
        "Key": "Key1",
        "Value": "Value1"
    },
    {
        "Key": "Key2",
        "Value": "Value2"
    }
]
```

Ajout ou mise à jour des balises pour une application existante

Vous pouvez ajouter des balises à une application à l'aide de l'action TagResource. Vous ne pouvez pas ajouter de balises à une application à l'aide de l'action UpdateApplication.

Pour mettre à jour une balise existante, ajoutez une balise avec la même clé que la balise existante.

L'exemple de demande suivant pour l'action `TagResource` ajoute de nouvelles balises ou met à jour des balises existantes :

```
{
   "ResourceARN": "string",
   "Tags": [
      {
         "Key": "NewTagKey",
         "Value": "NewTagValue"
      },
      {
         "Key": "ExistingKeyOfTagToUpdate",
         "Value": "NewValueForExistingTag"
      }
   ]
}
```

Répertorier les balises d'une application

Pour répertorier les balises existantes, utilisez l'action ListTagsForResource.

L'exemple de demande suivant pour l'action `ListTagsForResource` répertorie les balises pour une application :

```
{
   "ResourceARN": "arn:aws:kinesisanalyticsus-west-2:012345678901:application/
MyApplication"
}
```

Suppression des balises d'une application

Pour supprimer des balises d'une application, vous devez utiliser l’action UntagResource.

L'exemple de demande suivant pour l'action `UntagResource` supprime des balises d'une application :

```
{
   "ResourceARN": "arn:aws:kinesisanalyticsus-west-2:012345678901:application/
MyApplication",
   "TagKeys": [ "KeyOfFirstTagToRemove", "KeyOfSecondTagToRemove" ]
}
```

Utilisation de CloudFormation avec le service géré pour Apache Flink

Les exercices suivants montrent comment démarrer une application Flink créée via AWS CloudFormation à l’aide d'une fonction Lambda dans la même pile.

Avant de commencer

Avant de commencer cet exercice, suivez les étapes de création d'une application Flink via AWS CloudFormation sur AWS::KinesisAnalytics::Application.

Écriture d’une fonction Lambda

Pour démarrer une application Flink après sa création ou sa mise à jour, nous utilisons l'API kinesisanalyticsv2 start-application. L'appel sera déclenché par un événement AWS CloudFormation

après la création de l'application Flink. Nous verrons comment configurer la pile pour déclencher la fonction Lambda plus loin dans cet exercice, mais nous allons d'abord nous concentrer sur l'instruction de la fonction Lambda et son code. Nous utilisons l'exécution `Python3.8` dans cet exemple.

```
StartApplicationLambda:
    Type: AWS::Lambda::Function
    DependsOn: StartApplicationLambdaRole
    Properties:
      Description: Starts an application when invoked.
      Runtime: python3.8
      Role: !GetAtt StartApplicationLambdaRole.Arn
      Handler: index.lambda_handler
      Timeout: 30
      Code:
        ZipFile: |
          import logging
          import cfnresponse
          import boto3

          logger = logging.getLogger()
          logger.setLevel(logging.INFO)

          def lambda_handler(event, context):
            logger.info('Incoming CFN event {}'.format(event))

            try:
              application_name = event['ResourceProperties']['ApplicationName']

              # filter out events other than Create or Update,
              # you can also omit Update in order to start an application on Create
 only.
              if event['RequestType'] not in ["Create", "Update"]:
                logger.info('No-op for Application {} because CFN RequestType {} is
 filtered'.format(application_name, event['RequestType']))
                cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

                return

              # use kinesisanalyticsv2 API to start an application.
              client_kda = boto3.client('kinesisanalyticsv2',
 region_name=event['ResourceProperties']['Region'])
```

```
            # get application status.
            describe_response =
 client_kda.describe_application(ApplicationName=application_name)
            application_status = describe_response['ApplicationDetail']
['ApplicationStatus']

            # an application can be started from 'READY' status only.
            if application_status != 'READY':
              logger.info('No-op for Application {} because ApplicationStatus {} is
 filtered'.format(application_name, application_status))
              cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

              return

            # create RunConfiguration.
            run_configuration = {
              'ApplicationRestoreConfiguration': {
                'ApplicationRestoreType': 'RESTORE_FROM_LATEST_SNAPSHOT',
              }
            }

            logger.info('RunConfiguration for Application {}:
 {}'.format(application_name, run_configuration))

            # this call doesn't wait for an application to transfer to 'RUNNING'
 state.
            client_kda.start_application(ApplicationName=application_name,
 RunConfiguration=run_configuration)

            logger.info('Started Application: {}'.format(application_name))
            cfnresponse.send(event, context, cfnresponse.SUCCESS, {})
          except Exception as err:
            logger.error(err)
            cfnresponse.send(event,context, cfnresponse.FAILED, {"Data": str(err)})
```

Dans le code précédent, Lambda traitera les événements AWS CloudFormation entrants, filtrera tout à part `Create` et `Update`, obtiendra l'état de l'application et la démarrera si l'état est `READY`. Afin de connaître l'état de l'application, vous devez créer le rôle Lambda, comme indiqué ci-dessous :

Création d'un rôle Lambda

Vous créez un rôle pour que Lambda puisse « communiquer » avec l'application et écrire dans les journaux. Ce rôle utilisera des politiques gérées par défaut, mais vous souhaiterez peut-être le limiter à l'aide de politiques personnalisées.

```
StartApplicationLambdaRole:
    Type: AWS::IAM::Role
    DependsOn: TestFlinkApplication
    Properties:
      Description: A role for lambda to use while interacting with an application.
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - lambda.amazonaws.com
            Action:
              - sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/Amazonmanaged-flinkFullAccess
        - arn:aws:iam::aws:policy/CloudWatchLogsFullAccess
      Path: /
```

Notez que les ressources Lambda seront créées après la création de l'application Flink dans la même pile, car elles en dépendent.

Invocation de fonctions Lambda

Il ne reste plus qu'à invoquer la fonction Lambda. Cela se fait à l'aide d'une ressource personnalisée.

```
StartApplicationLambdaInvoke:
    Description: Invokes StartApplicationLambda to start an application.
    Type: AWS::CloudFormation::CustomResource
    DependsOn: StartApplicationLambda
    Version: "1.0"
    Properties:
      ServiceToken: !GetAtt StartApplicationLambda.Arn
      Region: !Ref AWS::Region
      ApplicationName: !Ref TestFlinkApplication
```

C'est tout ce dont vous avez besoin pour démarrer votre application Flink avec Lambda. Vous êtes maintenant prêt à créer votre propre pile ou à utiliser l'exemple complet ci-dessous pour voir comment toutes ces étapes fonctionnent dans la pratique.

Exemple complet

L'exemple suivant est une version légèrement étendue des étapes ci-dessus avec des ajustements `RunConfiguration` supplémentaires effectuées via les paramètres du modèle. Il s'agit d'une pile fonctionnelle que vous pouvez essayer. Assurez-vous de lire les notes d'accompagnement :

stack.yaml

```
Description: 'kinesisanalyticsv2 CloudFormation Test Application'
Parameters:
  ApplicationRestoreType:
    Description: ApplicationRestoreConfiguration option, can
 be SKIP_RESTORE_FROM_SNAPSHOT, RESTORE_FROM_LATEST_SNAPSHOT or
 RESTORE_FROM_CUSTOM_SNAPSHOT.
    Type: String
    Default: SKIP_RESTORE_FROM_SNAPSHOT
    AllowedValues: [ SKIP_RESTORE_FROM_SNAPSHOT, RESTORE_FROM_LATEST_SNAPSHOT,
 RESTORE_FROM_CUSTOM_SNAPSHOT ]
  SnapshotName:
    Description: ApplicationRestoreConfiguration option, name of a snapshot to restore
 to, used with RESTORE_FROM_CUSTOM_SNAPSHOT ApplicationRestoreType.
    Type: String
    Default: ''
  AllowNonRestoredState:
    Description: FlinkRunConfiguration option, can be true or false.
    Default: true
    Type: String
    AllowedValues: [ true, false ]
  CodeContentBucketArn:
    Description: ARN of a bucket with application code.
    Type: String
  CodeContentFileKey:
    Description: A jar filename with an application code inside a bucket.
    Type: String
Conditions:
  IsSnapshotNameEmpty: !Equals [ !Ref SnapshotName, '' ]
Resources:
  TestServiceExecutionRole:
    Type: AWS::IAM::Role
```

```
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - kinesisanlaytics.amazonaws.com
            Action: sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AmazonKinesisFullAccess
        - arn:aws:iam::aws:policy/AmazonS3FullAccess
      Path: /
  InputKinesisStream:
    Type: AWS::Kinesis::Stream
    Properties:
      ShardCount: 1
  OutputKinesisStream:
    Type: AWS::Kinesis::Stream
    Properties:
      ShardCount: 1
  TestFlinkApplication:
    Type: 'AWS::kinesisanalyticsv2::Application'
    Properties:
      ApplicationName: 'CFNTestFlinkApplication'
      ApplicationDescription: 'Test Flink Application'
      RuntimeEnvironment: 'FLINK-1_15'
      ServiceExecutionRole: !GetAtt TestServiceExecutionRole.Arn
      ApplicationConfiguration:
        EnvironmentProperties:
          PropertyGroups:
            - PropertyGroupId: 'KinesisStreams'
              PropertyMap:
                INPUT_STREAM_NAME: !Ref InputKinesisStream
                OUTPUT_STREAM_NAME: !Ref OutputKinesisStream
                AWS_REGION: !Ref AWS::Region
        FlinkApplicationConfiguration:
          CheckpointConfiguration:
            ConfigurationType: 'CUSTOM'
            CheckpointingEnabled: True
            CheckpointInterval: 1500
            MinPauseBetweenCheckpoints: 500
          MonitoringConfiguration:
            ConfigurationType: 'CUSTOM'
```

```
            MetricsLevel: 'APPLICATION'
            LogLevel: 'INFO'
          ParallelismConfiguration:
            ConfigurationType: 'CUSTOM'
            Parallelism: 1
            ParallelismPerKPU: 1
            AutoScalingEnabled: True
        ApplicationSnapshotConfiguration:
          SnapshotsEnabled: True
        ApplicationCodeConfiguration:
          CodeContent:
            S3ContentLocation:
              BucketARN: !Ref CodeContentBucketArn
              FileKey: !Ref CodeContentFileKey
          CodeContentType: 'ZIPFILE'
  StartApplicationLambdaRole:
    Type: AWS::IAM::Role
    DependsOn: TestFlinkApplication
    Properties:
      Description: A role for lambda to use while interacting with an application.
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - lambda.amazonaws.com
            Action:
              - sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/Amazonmanaged-flinkFullAccess
        - arn:aws:iam::aws:policy/CloudWatchLogsFullAccess
      Path: /
  StartApplicationLambda:
    Type: AWS::Lambda::Function
    DependsOn: StartApplicationLambdaRole
    Properties:
      Description: Starts an application when invoked.
      Runtime: python3.8
      Role: !GetAtt StartApplicationLambdaRole.Arn
      Handler: index.lambda_handler
      Timeout: 30
      Code:
        ZipFile: |
```

```
          import logging
          import cfnresponse
          import boto3

          logger = logging.getLogger()
          logger.setLevel(logging.INFO)

          def lambda_handler(event, context):
            logger.info('Incoming CFN event {}'.format(event))

            try:
              application_name = event['ResourceProperties']['ApplicationName']

              # filter out events other than Create or Update,
              # you can also omit Update in order to start an application on Create
 only.
              if event['RequestType'] not in ["Create", "Update"]:
                logger.info('No-op for Application {} because CFN RequestType {} is
 filtered'.format(application_name, event['RequestType']))
                cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

                return

              # use kinesisanalyticsv2 API to start an application.
              client_kda = boto3.client('kinesisanalyticsv2',
 region_name=event['ResourceProperties']['Region'])

              # get application status.
              describe_response =
 client_kda.describe_application(ApplicationName=application_name)
              application_status = describe_response['ApplicationDetail']
['ApplicationStatus']

              # an application can be started from 'READY' status only.
              if application_status != 'READY':
                logger.info('No-op for Application {} because ApplicationStatus {} is
 filtered'.format(application_name, application_status))
                cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

                return

              # create RunConfiguration from passed parameters.
              run_configuration = {
                'FlinkRunConfiguration': {
```

```
                  'AllowNonRestoredState': event['ResourceProperties']
['AllowNonRestoredState'] == 'true'
                },
                'ApplicationRestoreConfiguration': {
                  'ApplicationRestoreType': event['ResourceProperties']
['ApplicationRestoreType'],
                }
              }

              # add SnapshotName to RunConfiguration if specified.
              if event['ResourceProperties']['SnapshotName'] != '':
                run_configuration['ApplicationRestoreConfiguration']['SnapshotName'] =
 event['ResourceProperties']['SnapshotName']

              logger.info('RunConfiguration for Application {}:
 {}'.format(application_name, run_configuration))

              # this call doesn't wait for an application to transfer to 'RUNNING'
 state.
              client_kda.start_application(ApplicationName=application_name,
 RunConfiguration=run_configuration)

              logger.info('Started Application: {}'.format(application_name))
              cfnresponse.send(event, context, cfnresponse.SUCCESS, {})
            except Exception as err:
              logger.error(err)
              cfnresponse.send(event,context, cfnresponse.FAILED, {"Data": str(err)})
  StartApplicationLambdaInvoke:
    Description: Invokes StartApplicationLambda to start an application.
    Type: AWS::CloudFormation::CustomResource
    DependsOn: StartApplicationLambda
    Version: "1.0"
    Properties:
      ServiceToken: !GetAtt StartApplicationLambda.Arn
      Region: !Ref AWS::Region
      ApplicationName: !Ref TestFlinkApplication
      ApplicationRestoreType: !Ref ApplicationRestoreType
      SnapshotName: !Ref SnapshotName
      AllowNonRestoredState: !Ref AllowNonRestoredState
```

Encore une fois, vous souhaiterez peut-être ajuster les rôles pour Lambda ainsi que pour une application elle-même.

Avant de créer la pile ci-dessus, n'oubliez pas de spécifier vos paramètres.

paramètres.json

```
[
  {
    "ParameterKey": "CodeContentBucketArn",
    "ParameterValue": "YOUR_BUCKET_ARN"
  },
  {
    "ParameterKey": "CodeContentFileKey",
    "ParameterValue": "YOUR_JAR"
  },
  {
    "ParameterKey": "ApplicationRestoreType",
    "ParameterValue": "SKIP_RESTORE_FROM_SNAPSHOT"
  },
  {
    "ParameterKey": "AllowNonRestoredState",
    "ParameterValue": "true"
  }
]
```

Remplacez YOUR_BUCKET_ARN et YOUR_JAR selon vos besoins spécifiques. Vous pouvez suivre ce guide pour créer un compartiment Amazon S3 et un fichier JAR d'application.

Créez maintenant la pile (remplacez YOUR_REGION par une région de votre choix, par exemple us-east-1) :

```
aws cloudformation create-stack --region YOUR_REGION --template-body "file://
stack.yaml" --parameters "file://parameters.json" --stack-name "TestManaged Service for
 Apache FlinkStack" --capabilities CAPABILITY_NAMED_IAM
```

Vous pouvez maintenant accéder à https://console.aws.amazon.com/cloudformation et consulter la progression. Une fois votre application Flink créée, vous devriez la voir à l'état `Starting`. Cela peut prendre quelques minutes avant qu'elle ne soit `Running`.

Pour plus d'informations, consultez les ressources suivantes :

- Quatre méthodes pour récupérer n'importe quelle propriété de service AWS à l'aide d'AWS CloudFormation (partie 1 sur 3).

- Procédure : recherche des ID d'Amazon Machine Image.

Utilisation du tableau de bord Apache Flink avec le service géré pour Apache Flink

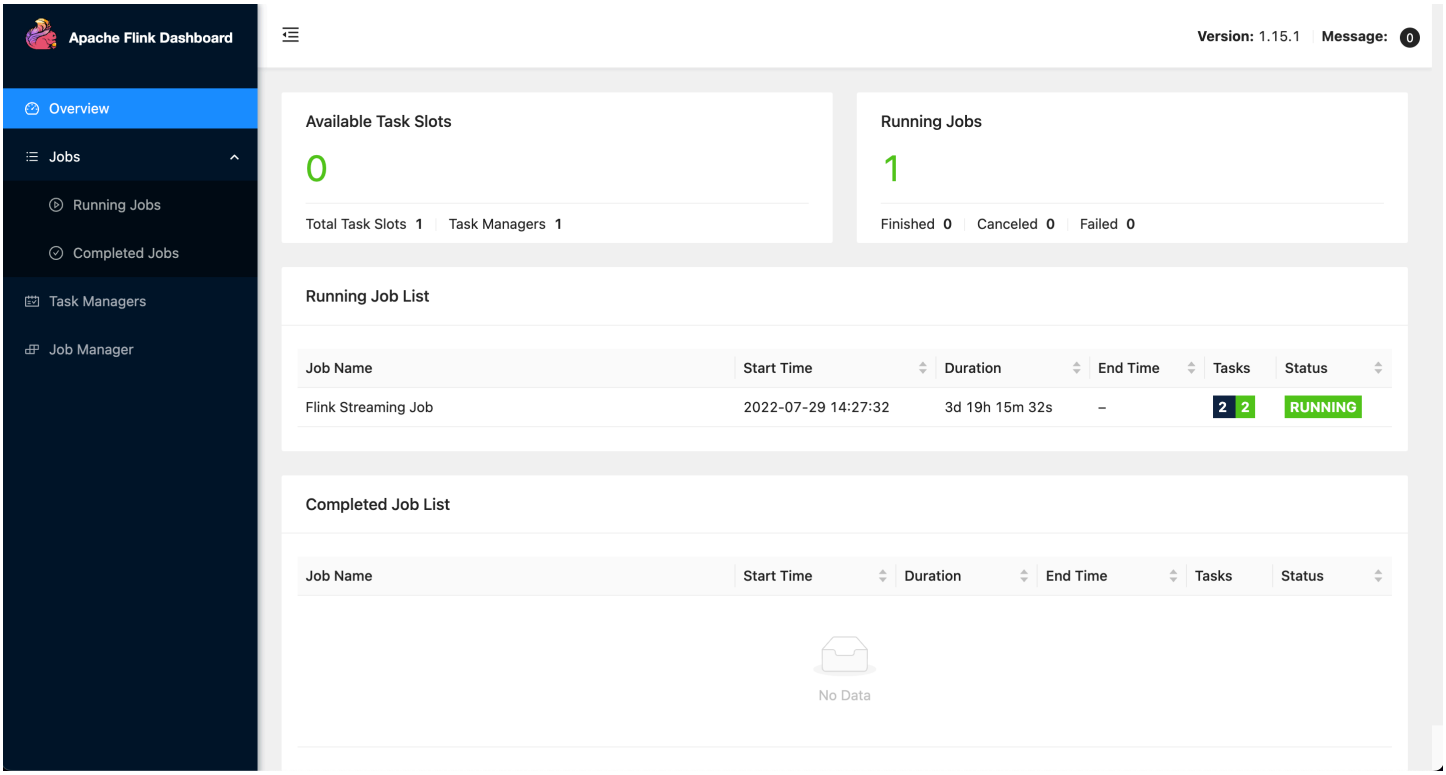
Vous pouvez utiliser le tableau de bord Apache Flink de votre application pour surveiller l'état de santé de votre application de service géré pour Apache Flink. Le tableau de bord de votre application affiche les informations suivantes :

- Ressources utilisées, y compris les gestionnaires de tâches et les emplacements de tâches.
- Informations sur les tâches, y compris celles en cours d'exécution, terminées, annulées ou ayant échoué.

Pour plus d'informations sur les gestionnaires de tâches, les emplacements de tâches et les tâches d'Apache Flink, consultez Apache Flink Architecture sur le site Web d'Apache Flink.

Notez ce qui suit à propos de l'utilisation du tableau de bord Apache Flink avec le service géré pour Apache Flink :

- Le tableau de bord Apache Flink pour les applications de service géré pour Apache Flink est en lecture seule. Vous ne pouvez pas modifier votre application de service géré pour Apache Flink à l’aide du tableau de bord Apache Flink.
- Le tableau de bord Apache Flink n'est pas compatible avec Microsoft Internet Explorer.



Accès au tableau de bord Apache Flink de votre application

Vous pouvez accéder au tableau de bord Apache Flink de votre application via la console du service géré pour Apache Flink ou en demandant un point de terminaison URL sécurisé à l'aide de l’interface CLI.

Accès au tableau de bord de votre application Apache Flink à l'aide de la console du service géré pour Apache Flink

Pour accéder au tableau de bord de votre application Apache Flink depuis la console, choisissez Tableau de bord Apache Flink sur la page de votre application.

> **Note**
>
> Lorsque vous ouvrez le tableau de bord depuis la console du service géré pour Apache Flink, l'URL générée par la console sera valide pendant 12 heures.

Accès au tableau de bord de votre application Apache Flink à l'aide de l’interface CLI du service géré pour Apache Flink

Vous pouvez utiliser l’interface CLI du service géré pour Apache Flink pour générer une URL permettant d'accéder au tableau de bord de votre application. L'URL que vous générez est valide pour une durée déterminée.

> **Note**
>
> Si vous n'accédez pas à l'URL générée dans les trois minutes, elle ne sera plus valide.

Vous générez l'URL de votre tableau de bord à l'aide de l'action CreateApplicationPresignedUrl. Vous pouvez spécifier les paramètres suivants pour l’action :

- Le nom de l'application
- Durée en secondes pendant laquelle l'URL sera valide
- Vous spécifiez `FLINK_DASHBOARD_URL` comme type d'URL.

Versions

Cette rubrique contient des informations sur les fonctionnalités prises en charge et les versions de composants recommandées pour chaque version du service géré pour Apache Flink.

Service géré Amazon pour Apache Flink version 1.15.2

Le service géré pour Apache Flink prend en charge les nouvelles fonctionnalités suivantes dans Apache 1.15.2

Fonction	Description	Référence Apache FLIP
Récepteur asynchrone	Un environnement AWS contributif pour la création de destinations asynchrones qui permet aux développeurs de créer des connecteurs AWS personnalisés avec moins de la moitié de l'effort précédent. Pour de plus amples informations, veuillez consulter The Generic Asynchronous Base Sink.	FLIP-171: Async Sink.
Récepteur Kinesis Data Firehose	AWS a contribué à un nouveau récepteur Amazon Kinesis Firehose utilisant l'environnement asynchrone.	Récepteur Amazon Kinesis Data Firehose
Arrêter avec point de sauvegarde	Arrêter avec point de sauvegarde garantit un fonctionnement sans faille et surtout en garantissant une sémantique unique pour les clients qui comptent dessus.	FLIP-34: Terminate/Suspend Job with Savepoint.

Fonction	Description	Référence Apache FLIP
Découplage Scala	Les utilisateurs peuvent désormais tirer parti de l'API Java depuis n'importe quelle version de Scala, y compris Scala 3. Les clients devront intégrer la bibliothèque standard Scala de leur choix à leurs applications Scala.	FLIP-28: Long-term goal of making flink-table Scala-free.
Scala	Voir le découplage de Scala ci-dessus	FLIP-28: Long-term goal of making flink-table Scala-free.
Métriques du connecteur unifié	Flink a défini des métriques standard pour les tâches et les opérateurs. Le service géré pour Apache Flink continuer a à prendre en charge les métriques du récepteur et de la source et introduira `numRestarts` en parallèle avec `fullRestarts` dans la version 1.15 pour les métriques de disponibilité.	FLIP-33: Standardize Connector Metrics et FLIP-179: Expose Standardized Operator Metrics.
Point de contrôle des tâches terminées	Cette fonctionnalité est activée par défaut dans Flink 1.15 et permet de continuer à effectuer des points de contrôle même si certaines parties du graphique de tâches ont fini de traiter toutes les données, ce qui peut se produire s'il contient des sources limitées (par lots).	FLIP-147: Support Checkpoints After Tasks Finished.

Modifications apportées au service géré Amazon pour Apache Flink avec Apache Flink 1.15

Blocs-notes Studio

Le service géré pour Apache Flink prend désormais en charge Apache Flink 1.15. Le service géré pour Apache Flink Studio utilise les blocs-notes Apache Zeppelin pour fournir une expérience de développement à interface unique pour le développement, le débogage de code et l'exécution d'applications de traitement de flux Apache Flink. Vous pouvez en savoir plus sur le service géré pour Apache Flink Studio et sur la façon de démarrer sur Utilisation d'un bloc-notes Studio avec le service géré pour Apache Flink Studio.

Connecteur EFO

Lors de la mise à niveau vers la version 1.15 du service géré pour Apache Flink, assurez-vous que vous utilisez le connecteur EFO le plus récent, à savoir la version 1.15.3 ou une version ultérieure. Pour plus d'informations sur les raisons, consultez FLINK-29324.

Découplage Scala

Pour commencer avec Flink 1.15.2, vous devrez intégrer la bibliothèque standard Scala de votre choix à vos applications Scala.

Récepteur Kinesis Data Firehose

Lors de la mise à niveau vers la version 1.15 du service géré pour Apache Flink, assurez-vous que vous utilisez le récepteur Amazon Kinesis Data Firehose le plus récent.

Connecteurs Kafka

Lors de la mise à niveau vers la version 1.15 du service géré Amazon pour Apache Flink, assurez-vous que vous utilisez les API de connecteur Kafka les plus récents. Apache Flink déconseille FlinkKafkaConsumer et FlinkKafkaProducer. Ces API pour le récepteur Kafka ne peuvent pas être validées dans Kafka pour Flink 1.15. Assurez-vous d'utiliser KafkaSource et KafkaSink.

Composants

Composant	Version
Java	11 (recommandée)

Composant	Version
Scala	2.12
Service géré pour Apache Flink pour l'éxécution Flink (aws-kinesisanalytics-runtime)	1.2.0
AWSConnecteur Kinesis (flink-connector-kinesis)	1.15.4
Apache Beam (applications Beam uniquement)	2.33.0, avec la version Jackson 2.12.2

Utilisation d’un bloc-notes Studio avec le service géré pour Apache Flink Studio

Les blocs-notes Studio pour le service géré pour Apache Flink vous permettent d’interroger des flux de données de manière interactive en temps réel, et de créer et d’exécuter facilement des applications de traitement de flux à l’aide des normes SQL, Python et Scala. En quelques clics dans la console AWS Management, vous pouvez lancer un bloc-notes sans serveur pour interroger des flux de données et obtenir des résultats en quelques secondes.

Un bloc-notes est un environnement de développement basé sur le Web. Grâce aux blocs-notes, vous bénéficiez d’une expérience de développement interactive simple associée aux capacités avancées fournies par Apache Flink. Les blocs-notes Studio utilisent des blocs-notes alimentés par Apache Zeppelin et se servent d’Apache Flink comme moteur de traitement des flux. Les blocs-notes Studio combinent parfaitement ces technologies pour rendre les analyses avancées sur les flux de données accessibles aux développeurs de tous niveaux de compétences.

Apache Zeppelin fournit à vos blocs-notes Studio une suite complète d’outils d’analyse, y compris les outils suivants :

- Visualisation des données
- Exportation de données dans des fichiers
- Contrôle du format de sortie pour une analyse simplifiée

Pour commencer à utiliser le service géré pour Apache Flink et Apache Zeppelin, consultez Didacticiel de création d'un bloc-notes Studio. Pour plus d'informations sur Apache Zeppelin, consultez la documentation Apache Zeppelin.

Avec un bloc-notes, vous modélisez des requêtes à l'aide de l'API Apache Flink Table et SQL dans SQL, Python ou Scala, ou de DataStream l'API dans Scala. En quelques clics, vous pouvez ensuite transformer le bloc-notes Studio en une application de traitement de flux de service géré en continu et non interactive pour Apache Flink pour vos charges de travail de production.

Cette rubrique contient les sections suivantes :

- Création d’un bloc-notes Studio
- Analyse interactive des données de streaming
- Déploiement en tant qu'application à état durable

- Autorisations IAM pour les blocs-notes Studio
- Connecteurs et dépendances
- Fonctions définies par l'utilisateur
- Activation de la création de points de contrôle
- Utilisation de AWS Glue
- Exemples et didacticiels
- Résolution des problèmes
- Annexe : création de politiques IAM personnalisées

Création d’un bloc-notes Studio

Un bloc-notes Studio contient des requêtes ou des programmes écrits en SQL, Python ou Scala qui s'exécutent sur des données de streaming et renvoient des résultats analytiques. Vous créez votre application à l'aide de la console ou de l’interface CLI et vous fournissez des requêtes pour analyser les données de votre source de données.

Votre application comporte les composants suivants :

- Une source de données, telle qu'un cluster Amazon MSK, un flux de données Kinesis ou un compartiment Amazon S3.
- Une base de données AWS Glue. Cette base de données contient des tables qui stockent vos schémas et points de terminaison de source de données et de destination. Pour plus d’informations, consultez Utilisation de AWS Glue.
- Votre code d'application. Votre code implémente votre requête ou votre programme d'analyse.
- Les paramètres et les propriétés d'exécution de votre application. Pour plus d'informations sur les paramètres et les propriétés d'exécution de votre application, consultez les rubriques suivantes dans le Guide du développeur pour les applications Apache Flink :
  - Parallélisme et mise à l’échelle de l’application : vous utilisez le paramètre de parallélisme de votre application pour contrôler le nombre de requêtes que votre application peut exécuter simultanément. Vos requêtes peuvent également tirer parti d'un parallélisme accru si elles comportent plusieurs chemins d'exécution, par exemple dans les circonstances suivantes :
    - Lors du traitement de plusieurs partitions d'un flux de données Kinesis
    - Lorsque vous partitionnez des données à l'aide de l'opérateur `KeyBy`.
    - Lors de l'utilisation de plusieurs opérateurs de fenêtre

Pour plus d'informations sur la mise à l'échelle des applications, consultez Mise à l'échelle des applications dans le service géré pour Apache Flink.

- Journalisation et surveillance : pour plus d'informations sur la journalisation et la surveillance des applications, consultez la section Journalisation et surveillance dans le service géré Amazon pour Apache Flink.
- Votre application utilise des points de contrôle et des points de sauvegarde pour la tolérance aux pannes. Les points de contrôle et de sauvegarde ne sont pas activés par défaut pour les blocs-notes Studio.

Vous pouvez créer votre bloc-notes Studio à l'aide de la AWS Management Console ou de l'interface AWS CLI.

Lorsque vous créez l'application à partir de la console, vous disposez des options suivantes :

- Dans la console Amazon MSK, choisissez votre cluster, puis choisissez Traiter les données en temps réel.
- Dans la console Kinesis Data Streams, choisissez votre flux de données, puis dans l'onglet Applications, choisissez Traiter les données en temps réel.
- Dans la console du service géré pour Apache Flink, choisissez l'onglet Studio, puis sélectionnez Créer un bloc-notes Studio.

Pour un didacticiel, voir Détection d'événements avec le service géré pour Apache Flink.

Pour un exemple de solution de bloc-notes Studio plus avancée, consultez Studio sur le service géré Amazon pour Apache Flink.

Analyse interactive des données de streaming

Vous utilisez un bloc-notes sans serveur alimenté par Apache Zeppelin pour interagir avec vos données de streaming. Votre bloc-notes peut contenir plusieurs notes, et chaque note peut comporter un ou plusieurs paragraphes dans lesquels vous pouvez écrire votre code.

L'exemple de requête SQL suivant montre comment récupérer des données à partir d'une source de données :

```
%flink.ssql(type=update)
```

```
select * from stock;
```

Pour d'autres exemples de requêtes SQL Flink Streaming, consultez Exemples et didacticiels et Queries de la documentation Apache Flink.

Vous pouvez utiliser les requêtes SQL Flink dans le bloc-notes Studio pour interroger des données de streaming. Vous pouvez également utiliser Python (API de table) et Scala (API de table et Datastream) pour écrire des programmes permettant d'interroger vos données de streaming de manière interactive. Vous pouvez consulter les résultats de vos requêtes ou de vos programmes, les mettre à jour en quelques secondes et les réexécuter pour afficher les résultats mis à jour.

Interprètes Flink

Vous spécifiez le langage que le service géré pour Apache Flink utilise pour exécuter votre application à l'aide d'un interprète. Vous pouvez utiliser les interpréteurs suivants avec le service géré pour Apache Flink :

Nom	Classe	Description
%flink	FlinkInterpreter	Creates ExecutionEnvironment/StreamExecutionEnvironment/BatchTableEnvironment/StreamTableEnvironment and provides a Scala environment
%flink.pyflink	PyFlinkInterpreter	Provides a python environment
%flink.ipyflink	IPyFlinkInterpreter	Provides an ipython environment
%flink.ssql	FlinkStreamSqlInterpreter	Provides a stream sql environment
%flink.bsql	FlinkBatchSqlInterpreter	Provides a batch sql environment

Pour plus d'informations sur les interprètes Flink, consultez la section Interprète Flink pour Apache Zeppelin.

Si vous utilisez `%flink.pyflink` ou `%flink.ipyflink` en tant qu'interprète, vous devrez utiliser le `ZeppelinContext` pour visualiser les résultats dans le bloc-notes.

Pour PyFlink des exemples plus spécifiques, voir Interroger vos flux de données de manière interactive à l'aide du service géré pour Apache Flink Studio et Python.

Variables d'environnement de table Apache Flink

Apache Zeppelin permet d'accéder aux ressources de l'environnement des tables à l'aide de variables d'environnement.

Vous accédez aux ressources de l'environnement des tables Scala avec les variables suivantes :

Variable	Ressource
`senv`	`StreamExecutionEnvironment`
`stenv`	`StreamTableEnvironment pour Blink Planner`

Vous accédez aux ressources de l'environnement des tables Python avec les variables suivantes :

Variable	Ressource
`s_env`	`StreamExecutionEnvironment`
`st_env`	`StreamTableEnvironment pour Blink Planner`

Pour plus d'informations sur l'utilisation des environnements de tables, consultez Create a TableEnvironment dans la documentation Apache Flink.

Déploiement en tant qu'application à état durable

Vous pouvez créer votre code et l'exporter vers Amazon S3. Vous pouvez promouvoir le code que vous avez écrit dans votre note vers une application de traitement des flux en cours d'exécution continue. Il existe deux modes d'exécution d'une application Apache Flink sur service géré pour

Apache Flink : avec un bloc-notes Studio, vous pouvez développer votre code de manière interactive, voir les résultats de votre code en temps réel et le visualiser dans votre note. Après avoir déployé une note pour qu'elle s'exécute en mode streaming, le service géré pour Apache Flink crée pour vous une application qui s'exécute en continu, lit les données de vos sources, écrit sur vos destinations, maintient l'état de l'application à long terme et s'adapte automatiquement en fonction du débit de vos flux sources.

> **Note**
>
> Le compartiment S3 vers lequel vous exportez le code de votre application doit se trouver dans la même région que votre bloc-notes Studio.

Vous ne pouvez déployer une note depuis votre bloc-notes Studio que si elle répond aux critères suivants :

- Les paragraphes doivent être classés dans l'ordre séquentiel. Lorsque vous déployez votre application, tous les paragraphes d'une note sont exécutés séquentiellement (left-to-right, top-to-bottom) tels qu'ils apparaissent dans votre note. Vous pouvez vérifier cet ordre en choisissant Exécuter tous les paragraphes dans votre note.
- Votre code est une combinaison de Python et de SQL ou de Scala et de SQL. Nous ne prenons pas en charge Python et Scala ensemble pour le moment pour deploy-as-application.
- Votre note ne doit avoir que les interprètes suivants :`%flink`, `%flink.ssql`, `%flink.pyflink`, `%flink.ipyflink`, `%md`.
- L'utilisation de l'objet z du Contexte Zeppelin n'est pas prise en charge. Les méthodes qui ne renvoient rien ne feront rien d'autre que de consigner un avertissement. D'autres méthodes déclencheront des exceptions Python ou échoueront à compiler dans Scala.
- Une note doit aboutir à une seule tâche Apache Flink.
- Les notes contenant des formulaires dynamiques ne sont pas prises en charge pour le déploiement en tant qu'application.
- Les paragraphes %md (Markdown) seront ignorés lors du déploiement en tant qu'application, car ils sont censés contenir une documentation lisible par l'homme qui ne convient pas à l'exécution dans le cadre de l'application résultante.
- Les paragraphes désactivés pour être exécutés dans Zeppelin seront ignorés lors du déploiement en tant qu'application. Même si un paragraphe désactivé utilise un interprète incompatible, par exemple `%flink.ipyflink` dans une note comportant les interprètes `%flink` and

`%flink.ssql`, il sera ignoré lors du déploiement de la note en tant qu'application et n'entraînera aucune erreur.

- Pour que le déploiement de l'application réussisse, il doit y avoir au moins un paragraphe contenant le code source (Flink SQL PyFlink ou Flink Scala) activé pour fonctionner.
- La définition du parallélisme dans la directive de l'interprète au sein d'un paragraphe (par exemple `%flink.ssql(parallelism=32)`) sera ignorée dans les applications déployées à partir d'une note. Vous pouvez plutôt mettre à jour l'application déployée via l'AWS Management ConsoleAWSAPI AWS Command Line Interface ou pour modifier les paramètres de parallélisme et/ou de ParallelismPer KPU en fonction du niveau de parallélisme requis par votre application, ou vous pouvez activer le dimensionnement automatique pour votre application déployée.
- Si vous effectuez un déploiement en tant qu'application à état durable, votre VPC doit disposer d'un accès à Internet. Si votre VPC n'a pas accès à Internet, consultez Déploiement en tant qu'application à état durable dans un VPC sans accès à Internet.

Critères Scala/Python

- Dans votre code Scala ou Python, utilisez le planificateur Blink (`senv`, `stenv` pour Scala ; `s_env`, `st_env` pour Python) et non l'ancien planificateur « Flink » (`stenv_2` pour Scala, `st_env_2` pour Python). Le projet Apache Flink recommande l'utilisation du planificateur Blink pour les cas d'utilisation en production. Il s'agit du planificateur par défaut dans Zeppelin et dans Flink.
- Vos paragraphes Python ne doivent pas utiliser d'invocations/affecations shell utilisant `!` ou des commandes magiques IPython, comme `%timeit` ou `%conda` dans des notes destinées à être déployées en tant qu'applications.
- Vous ne pouvez pas utiliser les classes de cas Scala comme paramètres de fonctions transmises à des opérateurs de flux de données d'ordre supérieur tels que `map` et `filter`. Pour plus d'informations sur les classes de cas Scala, consultez CASE CLASSES dans la documentation Scala.

Critères SQL

- Les instructions SELECT simples ne sont pas autorisées, car il n'existe nulle part d'équivalent à la section de sortie d'un paragraphe dans laquelle les données peuvent être fournies.
- Dans un paragraphe donné, les instructions DDL (`USE`, `CREATE`, `ALTER`, `DROP`, `SET`, `RESET`) doivent précéder les instructions DML (`INSERT`). Cela est dû au fait que les instructions DML d'un paragraphe doivent être soumises ensemble sous la forme d'une seule tâche Flink.

- Il doit y avoir au maximum un paragraphe contenant des instructions DML. En effet, pour cette deploy-as-application fonctionnalité, nous ne prenons en charge que la soumission d'une seule tâche à Flink.

Pour plus d'informations et un exemple, consultez Traduire, rédiger et analyser des données de streaming en utilisant les fonctions SQL avec le service géré Amazon pour Apache Flink, Amazon Translate et Amazon Comprehend.

Autorisations IAM pour les blocs-notes Studio

Le service géré pour Apache Flink vous crée un rôle IAM lorsque vous créez un bloc-notes Studio via la AWS Management Console. Il associe également à ce rôle une politique qui autorise les accès suivants :

Service	Accès
CloudWatch Journaux	Liste
Amazon EC2	Liste
AWS Glue	Lire, écrire
Service géré pour Apache Flink	Lecture
Service géré pour Apache Flink V2	Lecture
Amazon S3	Lire, écrire

Connecteurs et dépendances

Les connecteurs vous permettent de lire et d'écrire des données à travers différentes technologies. Le service géré pour Apache Flink intègre trois connecteurs par défaut à votre bloc-notes Studio. Vous pouvez également utiliser des connecteurs personnalisés. Pour plus d'informations sur les connecteurs, consultez Table & SQL Connectors dans la documentation Apache Flink.

Connecteurs par défaut

Si vous utilisez la AWS Management Console pour créer votre bloc-notes Studio, le service géré pour Apache Flink inclut par défaut les connecteurs personnalisés suivants : `flink-sql-connector-flink`, `flink-connector-kafka_2.12` et `aws-msk-iam-auth`. Pour créer un bloc-notes Studio via la console sans ces connecteurs personnalisés, choisissez l'option Créer avec des paramètres personnalisés. Ensuite, lorsque vous arrivez sur la page Configurations, décochez les cases à côté des deux connecteurs.

Si vous utilisez l'CreateApplicationAPI pour créer votre bloc-notes Studio, les `flink-connector-kafka` connecteurs `flink-sql-connector-flink` et ne sont pas inclus par défaut. Pour les ajouter, spécifiez-les en tant que `MavenRefernce` dans le type de données `CustomArtifactsConfiguration`, comme indiqué dans les exemples suivants.

Le connecteur `aws-msk-iam-auth` est le connecteur à utiliser avec Amazon MSK qui inclut la fonctionnalité d'authentification automatique auprès d'IAM.

> **Note**
>
> Les versions de connecteur présentées dans l'exemple suivant sont les seules que nous prenons en charge.

```
For the Kinesis connector:

"CustomArtifactsConfiguration": [{
"ArtifactType": "DEPENDENCY_JAR",
   "MavenReference": {
"GroupId": "org.apache.flink",

      "ArtifactId": "flink-sql-connector-kinesis",
      "Version": "1.15.4"

   }
}]

For authenticating with AWS MSK through AWS IAM:

"CustomArtifactsConfiguration": [{
"ArtifactType": "DEPENDENCY_JAR",
```

```
   "MavenReference": {
"GroupId": "software.amazon.msk",
      "ArtifactId": "aws-msk-iam-auth",
      "Version": "1.1.6"
   }
}]

For the Apache Kafka connector:

"CustomArtifactsConfiguration": [{
"ArtifactType": "DEPENDENCY_JAR",
   "MavenReference": {
"GroupId": "org.apache.flink",

      "ArtifactId": "flink-connector-kafka",
      "Version": "1.15.4"

   }
}]
```

Pour ajouter ces connecteurs à un bloc-notes existant, utilisez l'opération UpdateApplicationAPI et spécifiez-les `MavenReference` en tant que type de `CustomArtifactsConfigurationUpdate` données.

> **Note**
>
> Vous pouvez définir `failOnError` sur true pour le connecteur `flink-sql-connector-kinesis` dans l'API de table.

Dépendances et connecteurs personnalisés

Pour utiliser la AWS Management Console pour ajouter une dépendance ou un connecteur personnalisé à votre bloc-notes Studio, procédez comme suit :

1. Chargez le fichier de votre connecteur personnalisé sur Amazon S3.
2. Dans la AWS Management Console, choisissez l'option Création personnalisée pour créer votre bloc-notes Studio.
3. Suivez le processus de création du bloc-notes Studio jusqu'à ce que vous arriviez à l'étape Configurations.

4. Dans la section Connecteurs personnalisés, choisissez Ajouter un connecteur personnalisé.
5. Spécifiez l'emplacement Amazon S3 de la dépendance ou du connecteur personnalisé.
6. Sélectionnez Enregistrer les modifications.

Pour ajouter un fichier JAR de dépendance ou un connecteur personnalisé lorsque vous créez un nouveau bloc-notes Studio à l'aide de l'CreateApplicationAPI, spécifiez l'emplacement Amazon S3 du fichier JAR de dépendance ou du connecteur personnalisé dans le type de `CustomArtifactsConfiguration` données. Pour ajouter une dépendance ou un connecteur personnalisé à un bloc-notes Studio existant, appelez l'opération d'UpdateApplicationAPI et spécifiez l'emplacement Amazon S3 du JAR de dépendance ou du connecteur personnalisé dans le type de `CustomArtifactsConfigurationUpdate` données.

> **Note**
>
> Lorsque vous incluez une dépendance ou un connecteur personnalisé, vous devez également inclure toutes ses dépendances transitives qui ne sont pas regroupées en son sein.

Fonctions définies par l'utilisateur

Les fonctions définies par l'utilisateur (UDF) sont des points d'extension qui vous permettent d'appeler une logique fréquemment utilisée ou une logique personnalisée qui ne peut être exprimée autrement dans les requêtes. Vous pouvez utiliser Python ou un langage JVM tel que Java ou Scala pour implémenter vos UDF dans les paragraphes de votre bloc-notes Studio. Vous pouvez également ajouter à votre bloc-notes Studio des fichiers JAR externes contenant des UDF implémentés dans un langage JVM.

Lorsque vous implémentez des fichiers JAR qui enregistrent des classes abstraites qui sous-classent `UserDefinedFunction` (ou vos propres classes abstraites), utilisez la portée fournie dans Apache Maven, les instructions de dépendance `compileOnly` dans Gradle, la portée fournie dans SBT ou une directive équivalente dans la configuration de construction de votre projet UDF. Cela permet au code source UDF de se compiler par rapport aux API Flink, mais les classes d'API Flink ne sont pas elles-mêmes incluses dans les artefacts de construction. Reportez-vous à ce pom tiré de l'exemple de fichier JAR UDF qui respecte ces prérequis dans un projet Maven.

> **Note**
>
> Pour un exemple de configuration, consultez Traduire, rédiger et analyser des données de streaming en utilisant les fonctions SQL avec le service géré Amazon pour Apache Flink, Amazon Translate et Amazon Comprehend dans le blog AWS Machine Learning.

Pour utiliser la console afin d'ajouter des fichiers JAR UDF à votre bloc-notes Studio, procédez comme suit :

1. Chargez vos fichiers JAR UDF sur Amazon S3.
2. Dans la AWS Management Console, choisissez l'option Création personnalisée pour créer votre bloc-notes Studio.
3. Suivez le processus de création du bloc-notes Studio jusqu'à ce que vous arriviez à l'étape Configurations.
4. Dans la section Fonctions définies par l'utilisateur, choisissez Ajouter une fonction définie par l'utilisateur.
5. Spécifiez l'emplacement Amazon S3 du fichier JAR ou du fichier ZIP contenant l'implémentation de votre UDF.
6. Sélectionnez Enregistrer les modifications.

Pour ajouter un JAR UDF lorsque vous créez un nouveau bloc-notes Studio à l'aide de l'CreateApplicationAPI, spécifiez l'emplacement du JAR dans le type de `CustomArtifactConfiguration` données. Pour ajouter un fichier JAR UDF à un bloc-notes Studio existant, appelez l'opération d'UpdateApplicationAPI et spécifiez l'emplacement du fichier JAR dans le type de `CustomArtifactsConfigurationUpdate` données. Vous pouvez également utiliser la AWS Management Console pour ajouter des fichiers JAR UDF à votre bloc-notes Studio.

Considérations relatives aux fonctions définies par l'utilisateur

- Le service géré pour Apache Flink Studio utilise la terminologie d'Apache Zeppelin selon laquelle un bloc-notes est une instance Zeppelin pouvant contenir plusieurs notes. Chaque note peut ensuite contenir plusieurs paragraphes. Avec le service géré pour Apache Flink Studio, le processus d'interprétation est partagé entre toutes les notes du bloc-notes. Ainsi, si vous effectuez un enregistrement de fonction explicite à l'aide de createTemporarySystemFunction dans une note, celle-ci peut être référencée telle quelle dans une autre note du même bloc-notes.

L'opération Déployer en tant qu'application fonctionne toutefois sur une note individuelle et non sur toutes les notes du bloc-notes. Lorsque vous effectuez un déploiement en tant qu'application, seul le contenu d'une note active est utilisé pour générer l'application. Tout enregistrement de fonction explicite effectué dans d'autres blocs-notes ne fait pas partie des dépendances d'application générées. En outre, lors de l'option Déployer en tant qu'application, un enregistrement de fonction implicite se produit en convertissant le nom de classe principal de JAR en une chaîne minuscule.

Par exemple, si `TextAnalyticsUDF` est la classe principale pour le fichier JAR UDF, un enregistrement implicite donnera le nom de fonction `textanalyticsudf`. Ainsi, si un enregistrement de fonction explicite dans la note 1 de Studio se produit comme suit, toutes les autres notes de ce bloc-notes (disons la note 2) peuvent faire référence à la fonction par son nom `myNewFuncNameForClass` grâce à l'interprète partagé :

```
stenv.createTemporarySystemFunction("myNewFuncNameForClass", new
TextAnalyticsUDF())
```

Toutefois, lors de l'opération de déploiement en tant qu'application mentionnée à la note 2, cet enregistrement explicite ne sera pas inclus dans les dépendances et, par conséquent, l'application déployée ne fonctionnera pas comme prévu. En raison de l'enregistrement implicite, par défaut, toutes les références à cette fonction sont supposées être avec `textanalyticsudf` et non `myNewFuncNameForClass`.

S'il est nécessaire d'enregistrer un nom de fonction personnalisé, la note 2 elle-même devrait contenir un autre paragraphe pour effectuer un autre enregistrement explicite comme suit :

```
%flink(parallelism=l)
import com.amazonaws.kinesis.udf.textanalytics.TextAnalyticsUDF
# re-register the JAR for UDF with custom name
stenv.createTemporarySystemFunction("myNewFuncNameForClass", new TextAnalyticsUDF())
```

```
%flink. ssql(type=update, parallelism=1)
INSERT INTO
    table2
SELECT
    myNewFuncNameForClass(column_name)
FROM
    table1
;
```

- Si votre fichier JAR UDF inclut des SDK Flink, configurez votre projet Java de manière à ce que le code source UDF puisse être compilé par rapport aux SDK Flink, mais que les classes du SDK Flink ne soient pas elles-mêmes incluses dans l'artefact de construction, par exemple le fichier JAR.

  Vous pouvez utiliser la portée `provided` dans Apache Maven, les instructions de dépendance `compileOnly` dans Gradle, la portée `provided` dans SBT ou une directive équivalente dans la configuration de construction de leur projet UDF. Reportez-vous à ce pom tiré de l'exemple de fichier JAR UDF qui respecte ces prérequis dans un projet Maven. Pour un step-by-step didacticiel complet, consultez ce guide Translate, redact et analysez les données de streaming à l'aide des fonctions SQL avec Amazon Managed Service pour Apache Flink, Amazon Translate et Amazon Comprehend.

Activation de la création de points de contrôle

Vous activez le point de contrôle à l'aide des paramètres d'environnement. Pour plus d'informations sur le point de contrôle, consultez Tolérance aux pannes dans le guide du développeur du service géré pour Apache Flink.

Réglage de l'intervalle entre les points de contrôle

L'exemple de code Scala suivant définit l'intervalle de point de contrôle de votre application à une minute :

```
// start a checkpoint every 1 minute
stenv.enableCheckpointing(60000)
```

L'exemple de code Python suivant définit l'intervalle de point de contrôle de votre application à une minute :

```
st_env.get_config().get_configuration().set_string(
    "execution.checkpointing.interval", "1min"
)
```

Configuration du type de point de contrôle

L'exemple de code Scala suivant définit le mode point de contrôle de votre application sur `EXACTLY_ONCE` (valeur par défaut) :

```
// set mode to exactly-once (this is the default)
stenv.getCheckpointConfig.setCheckpointingMode(CheckpointingMode.EXACTLY_ONCE)
```

L'exemple de code Python suivant définit le mode point de contrôle de votre application sur `EXACTLY_ONCE` (valeur par défaut) :

```
st_env.get_config().get_configuration().set_string(
    "execution.checkpointing.mode", "EXACTLY_ONCE"
)
```

Utilisation de AWS Glue

Votre bloc-notes Studio stocke et obtient des informations sur ses sources de données et ses récepteurs à partir de AWS Glue. Lorsque vous créez votre bloc-notes Studio, vous spécifiez la base de données AWS Glue qui contient vos informations de connexion. Lorsque vous accédez à vos sources de données et à vos récepteurs, vous spécifiez les tables AWS Glue contenues dans la base de données. Vos tables AWS Glue permettent d'accéder aux connexions AWS Glue qui définissent les emplacements, les schémas et les paramètres de vos sources de données et de vos destinations.

Les blocs-notes Studio utilisent les propriétés des tables pour stocker des données spécifiques aux applications. Pour plus d'informations, consultez Propriétés de tableau.

Pour un exemple de configuration d'une connexion AWS Glue, d'une base de données et d'une table à utiliser avec les blocs-notes Studio, consultez Créer une base de données AWS Glue dans le didacticiel Didacticiel de création d'un bloc-notes Studio.

Propriétés de tableau

Outre les champs de données, vos tables AWS Glue fournissent d'autres informations à votre bloc-notes Studio à l'aide des propriétés de table. Le service géré pour Apache Flink utilise les propriétés de table AWS Glue suivantes :

- Utilisation des valeurs temporelles d'Apache Flink : ces propriétés définissent la manière dont le service géré pour Apache Flink émet les valeurs de temps de traitement des données internes d'Apache Flink.
- Utilisation du connecteur Flink et des propriétés de format : ces propriétés fournissent des informations sur vos flux de données.

Pour ajouter une propriété à une table AWS Glue, procédez comme suit :

1. Connectez-vous à la AWS Management Console et ouvrez la console AWS Glue à l'adresse https://console.aws.amazon.com/glue/.
2. Dans la liste des tables, choisissez la table que votre application utilise pour stocker ses informations de connexion de données. Choisissez Action, puis Modifier les détails de la table.
3. Sous Propriétés de la table, saisissez **`managed-flink.proctime`** pour la clé et **`user_action_time`** pour la valeur.

Utilisation des valeurs temporelles d'Apache Flink

Apache Flink fournit des valeurs temporelles qui décrivent le moment où les événements de traitement des flux se sont produits, tels que le temps de traitement et l'heure de l'événement. Pour inclure ces valeurs dans la sortie de votre application, vous définissez des propriétés sur votre table AWS Glue qui indiquent à l'exécution du service géré pour Apache Flink d'émettre ces valeurs dans les champs spécifiés.

Les clés et les valeurs que vous utilisez dans les propriétés de votre table sont les suivantes :

Type d'horodatage	Clé	Valeur
Temps de traitement	managed-flink.proctime	The column name that AWS Glue will use to expose the value. This column name does not correspond to an existing table column.
Heure de l'événement	managed-flink.rowtime	The column name that AWS Glue will use to expose the value. This column name corresponds to an existing table column.
	managed-flink.watermark.*column_name*.milliseconds	The watermark interval in milliseconds

Utilisation du connecteur Flink et des propriétés de format

Vous fournissez des informations sur vos sources de données aux connecteurs Flink de votre application à l'aide des propriétés de table AWS Glue. Voici quelques exemples des propriétés que le service géré pour Apache Flink utilise pour les connecteurs :

Type de connecteur	Clé	Valeur
Kafka	`format`	The format used to deserialize and serialize Kafka messages, e.g. `json` or `csv`.
	`scan.startup.mode`	The startup mode for the Kafka consumer, e.g. `décalage le plus précoce` or `timestamp` .
Kinesis	`format`	The format used to deseriali ze and serialize Kinesis data stream records, e.g. `json` or `csv`.
	`aws.region`	The AWS region where the stream is defined.
S3 (système de fichiers)	format	The format used to deserialize and serialize files, e.g. `json` or `csv`.
	`path`	The Amazon S3 path, e.g. `s3://mybucket/` .

Pour plus d'informations sur les autres connecteurs autres que Kinesis et Apache Kafka, consultez la documentation de votre connecteur.

Exemples et didacticiels

Rubriques

- Didacticiel : création d'un bloc-notes Studio dans le service géré pour Apache Flink
- Didacticiel : déploiement en tant qu'application à état durable
- Exemples

Didacticiel : création d'un bloc-notes Studio dans le service géré pour Apache Flink

Le didacticiel suivant explique comment créer un bloc-notes Studio qui lit les données d'un flux de données Kinesis ou d'un cluster Amazon MSK.

Ce didacticiel contient les sections suivantes :

- Installation
- Créer une base de données AWS Glue
- Étapes suivantes
- Création d'un bloc-notes Studio avec Kinesis Data Streams
- Création d'un bloc-notes Studio avec Amazon MSK
- Nettoyage de votre application et des ressources dépendantes

Installation

Assurez-vous que votre interface AWS CLI exécute la version 2 ou une version ultérieure. Pour installer la dernière interface AWS CLI, consultez Installation, mise à jour et désinstallation de l'interface AWS CLI version 2.

Créer une base de données AWS Glue

Votre bloc-notes Studio utilise une base de données AWS Glue pour les métadonnées relatives à votre source de données Amazon MSK.

Créer une base de données AWS Glue

1. Ouvrez la console AWS Glue, à l'adresse https://console.aws.amazon.com/glue/.
2. Choisissez Ajouter une base de données. Dans la fenêtre Ajouter une base de données, saisissez **`default`** comme nom de la base de données. Choisissez Créer.

Étapes suivantes

Avec ce didacticiel, vous pouvez créer un bloc-notes Studio qui utilise Kinesis Data Streams ou Amazon MSK :

- Kinesis Data Streams : avec Kinesis Data Streams, vous créez rapidement une application qui utilise un flux de données Kinesis comme source. Il vous suffit de créer un flux de données Kinesis en tant que ressource dépendante.
- Amazon MSK : avec Amazon MSK, vous créez une application qui utilise un cluster Amazon MSK comme source. Vous devez créer un Amazon VPC, une instance client Amazon EC2 et un cluster Amazon MSK en tant que ressources dépendantes.

Création d'un bloc-notes Studio avec Kinesis Data Streams

Ce didacticiel explique comment créer un bloc-notes Studio qui utilise un flux de données Kinesis comme source.

Ce didacticiel contient les sections suivantes :

- Configuration
- Création d'une table AWS Glue
- Créer un bloc-notes Studio avec Kinesis Data Streams
- Envoyer des données vers votre flux de données Kinesis
- Tester votre bloc-notes Studio

Configuration

Avant de créer un bloc-notes Studio, créez un flux de données Kinesis (`ExampleInputStream`). Votre application utilise ce flux comme source de l'application.

Vous pouvez créer ce flux à l'aide de la console Amazon Kinesis ou de la commande AWS CLI suivante. Pour obtenir des instructions sur la console, consultez Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams. Nommez le flux **`ExampleInputStream`** et définissez le Nombre de partitions ouvertes sur **`1`**.

Pour créer le flux (`ExampleInputStream`) à l'aide de la commande AWS CLI, utilisez la commande `create-stream` de l'interface AWS CLI Amazon Kinesis suivante.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-east-1 \
--profile adminuser
```

Création d'une table AWS Glue

Votre bloc-notes Studio utilise une base de données AWS Glue pour les métadonnées relatives à votre source de données Kinesis Data Streams.

> **Note**
>
> Vous pouvez d'abord créer la base de données manuellement ou laisser le service géré pour Apache Flink la créer pour vous lors de la création du bloc-notes. De même, vous pouvez soit créer la table manuellement comme décrit dans cette section, soit utiliser le code du connecteur de création de table pour le service géré pour Apache Flink dans votre bloc-notes dans Apache Zeppelin pour créer votre table via une instruction DDL. Vous pouvez ensuite vérifier dans AWS Glue que la table a été créée correctement.

Création d'une table

1. Connectez-vous à la AWS Management Console et ouvrez la console AWS Glue à l'adresse https://console.aws.amazon.com/glue/.
2. Si vous n'avez pas encore de base de données AWS Glue, choisissez Bases de données dans la barre de navigation de gauche. Choisissez Ajouter une base de données. Dans la fenêtre Ajouter une base de données, saisissez **`default`** comme nom de la base de données. Choisissez Créer.
3. Dans la barre de navigation de gauche, choisissez Tables. Sur la page Tables, choisissez Ajouter des tables, Ajouter une table manuellement.
4. Sur la page Configurer les propriétés de votre table, saisissez **`stock`** pour Nom de la table. Assurez-vous de sélectionner la base de données que vous avez créée précédemment. Choisissez Suivant.
5. Sur la page Ajouter un magasin de données, choisissez Kinesis. Pour le Nom du flux, saisissez **`ExampleInputStream`**. Pour URL source Kinesis, saisissez **`https://kinesis.us-`**

`east-1.amazonaws.com`. Si vous copiez et collez l'URL source Kinesis, veillez à supprimer les espaces de début ou de fin. Choisissez Suivant.

6. Sur la page Classification, choisissez JSON. Choisissez Suivant.
7. Sur la page Définir un schéma, choisissez Ajouter une colonne pour ajouter une colonne. Ajoutez des colonnes avec les propriétés suivantes :

Nom de la colonne	Type de données
`symbole boursier`	**`chaîne`**
`prix`	**`double`**

   Choisissez Suivant.
8. Sur la page suivante, vérifiez vos paramètres, puis choisissez Terminer.
9. Choisissez la table que vous venez de créer dans la liste des tables.
10. Choisissez Modifier la table et ajoutez une propriété avec la clé `managed-flink.proctime` et la valeur `proctime`.
11. Choisissez Appliquer.

Créer un bloc-notes Studio avec Kinesis Data Streams

Maintenant que vous avez créé les ressources utilisées par votre application, vous pouvez créer votre bloc-notes Studio.

Pour créer votre application, vous pouvez utiliser la AWS Management Console ou l’interface AWS CLI.

- Créer un bloc-notes Studio à l'aide de la AWS Management Console
- Créer un bloc-notes Studio à l'aide de l’interface AWS CLI

Créer un bloc-notes Studio à l'aide de la AWS Management Console

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard.
2. Sur la page Applications de service géré pour Apache Flink, choisissez l'onglet Studio. Choisissez Créer un bloc-notes Studio.

> **Note**
>
> Vous pouvez aussi créer un bloc-notes Studio à partir des consoles Amazon MSK ou Kinesis Data Streams en sélectionnant votre cluster Amazon MSK ou votre flux de données Kinesis d'entrée, puis en choisissant Traiter les données en temps réel.

3. Sur la page Créer un bloc-notes Studio, fournissez les informations suivantes :

   - Saisissez **`MyNotebook`** comme nom du bloc-notes.
   - Pour Base de données AWS Glue, choissiez Par défaut.

   Choisissez Créer un bloc-notes Studio.
4. Sur la MyNotebookpage, choisissez Exécuter. Attendez que État indique En cours d'exécution. Des frais s'appliquent lorsque le bloc-notes fonctionne.

Créer un bloc-notes Studio à l'aide de l’interface AWS CLI

Pour créer votre bloc-notes Studio à l'aide de l’interface AWS CLI, procédez comme suit :

1. Vérifiez votre identifiant de compte. Vous avez besoin de cette valeur pour créer votre application.
2. Créez le rôle `arn:aws:iam::`*`AccountID`*`:role/ZeppelinRole` et ajoutez les autorisations suivantes au rôle créé automatiquement par la console.

   ```
   "kinesis:GetShardIterator",

   "kinesis:GetRecords",

   "kinesis:ListShards"
   ```
3. Créez un fichier nommé `create.json` avec le contenu suivant. Remplacez les valeurs des espaces réservés par vos informations.

   ```
   {
       "ApplicationName": "MyNotebook",
       "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
       "ApplicationMode": "INTERACTIVE",
       "ServiceExecutionRole": "arn:aws:iam::AccountID:role/ZeppelinRole",
       "ApplicationConfiguration": {
   ```

```
        "ApplicationSnapshotConfiguration": {
            "SnapshotsEnabled": false
        },
        "ZeppelinApplicationConfiguration": {
            "CatalogConfiguration": {
                "GlueDataCatalogConfiguration": {
                    "DatabaseARN": "arn:aws:glue:us-east-1:AccountID:database/
default"
                }
            }
        }
    }
}
```

4. Pour créer votre application, exécutez la commande suivante.

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create.json
```

5. Lorsque la commande est terminée, vous voyez une sortie contenant les détails de votre nouveau bloc-notes Studio. Voici un exemple de la sortie.

```
{
    "ApplicationDetail": {
        "ApplicationARN": "arn:aws:kinesisanalyticsus-
east-1:012345678901:application/MyNotebook",
        "ApplicationName": "MyNotebook",
        "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
        "ApplicationMode": "INTERACTIVE",
        "ServiceExecutionRole": "arn:aws:iam::012345678901:role/ZeppelinRole",
...
```

6. Pour lancer votre application, exécutez la commande suivante. Remplacez les exemples de valeur par l'identifiant de votre compte.

```
aws kinesisanalyticsv2 start-application --application-arn
 arn:aws:kinesisanalyticsus-east-1:012345678901:application/MyNotebook\
```

Envoyer des données vers votre flux de données Kinesis

Pour envoyer des données de test vers votre flux de données Kinesis, procédez comme suit :

1. Utilisez le Kinesis Data Generator.

2. Choisissez Créer un utilisateur Cognito avec. CloudFormation
3. La console AWS CloudFormation s'ouvre avec le modèle Kinesis Data Generator. Choisissez Suivant.
4. Sur la page Spécifier les détails de la pile, saisissez le nom d'utilisateur et le mot de passe de votre utilisateur Cognito. Choisissez Suivant.
5. Sur la page Configurer les options de pile, choisissez Suivant.
6. Sur la page Review Kinesis-Data-Generator-Cognito-User, choisissez l'option Je reconnais qui pourrait créer des ressources IAM. AWS CloudFormation case à cocher. Sélectionnez Créer une pile.
7. Attendez la fin de la création de la pile AWS CloudFormation. Une fois la pile terminée, ouvrez la pile Kinesis-Data-Generator-Cognito-User dans la console AWS CloudFormation et choisissez l'onglet Sorties. Ouvrez l'URL répertoriée pour la valeur KinesisDataGeneratorUrlde sortie.
8. Sur la page Amazon Kinesis Data Generator, connectez-vous avec les informations d'identification que vous avez créées à l'étape 4.
9. Sur la page suivante, renseignez les valeurs suivantes :

Région	**us-east-1**
Flux/Flux Kinesis Data Firehose	**ExampleInputStream**
Enregistrements par seconde	**1**

Pour Modèle d’enregistrement, collez le code suivant :

```
{
    "ticker": "{{random.arrayElement(
        ["AMZN","MSFT","GOOG"]
    )}}",
    "price": {{random.number(
        {
            "min":10,
            "max":150
        }
    )}}
}
```

10. Choisissez Envoyer les données.

11. Le générateur enverra les données à votre flux de données Kinesis.

    Laissez le générateur tourner pendant que vous terminez la section suivante.

Tester votre bloc-notes Studio

Dans cette section, vous utilisez votre bloc-notes Studio pour interroger les données de votre flux de données Kinesis.

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard.
2. Sur la page Applications de service géré pour Apache Flink, choisissez l'onglet Bloc-notes Studio. Choisissez MyNotebook.
3. Sur la MyNotebookpage, choisissez Ouvrir dans Apache Zeppelin.

   L'interface Apache Zeppelin s'ouvre dans un nouvel onglet.
4. Sur la page Bienvenue sur Zeppelin !, choisissez Note Zeppelin.
5. Sur la page Note Zeppelin, entrez la requête suivante dans une nouvelle note :

   ```
   %flink.ssql(type=update)
   select * from stock
   ```

   Choisissez l'icône d'exécution.

   Après un court instant, la note affiche les données du flux de données Kinesis.

Pour ouvrir le tableau de bord Apache Flink de votre application afin de visualiser les aspects opérationnels, choisissez FLINK JOB. Pour plus d'informations sur le tableau de bord Flink, consultez Tableau de bord Apache Flink dans le guide du développeur du service géré pour Apache Flink.

Pour d'autres exemples de requêtes SQL Flink Streaming, consultez la section Queries de la documentation Apache Flink.

Création d'un bloc-notes Studio avec Amazon MSK

Ce didacticiel explique comment créer un bloc-notes Studio qui utilise un cluster Amazon MSK comme source.

Ce didacticiel contient les sections suivantes :

- Installation
- Ajoutez une passerelle NAT à votre VPC
- Pour créer une table et une connexion AWS Glue
- Créer un bloc-notes Studio avec Amazon MSK
- Envoyer des données à votre cluster Amazon MSK
- Tester votre bloc-notes Studio

Installation

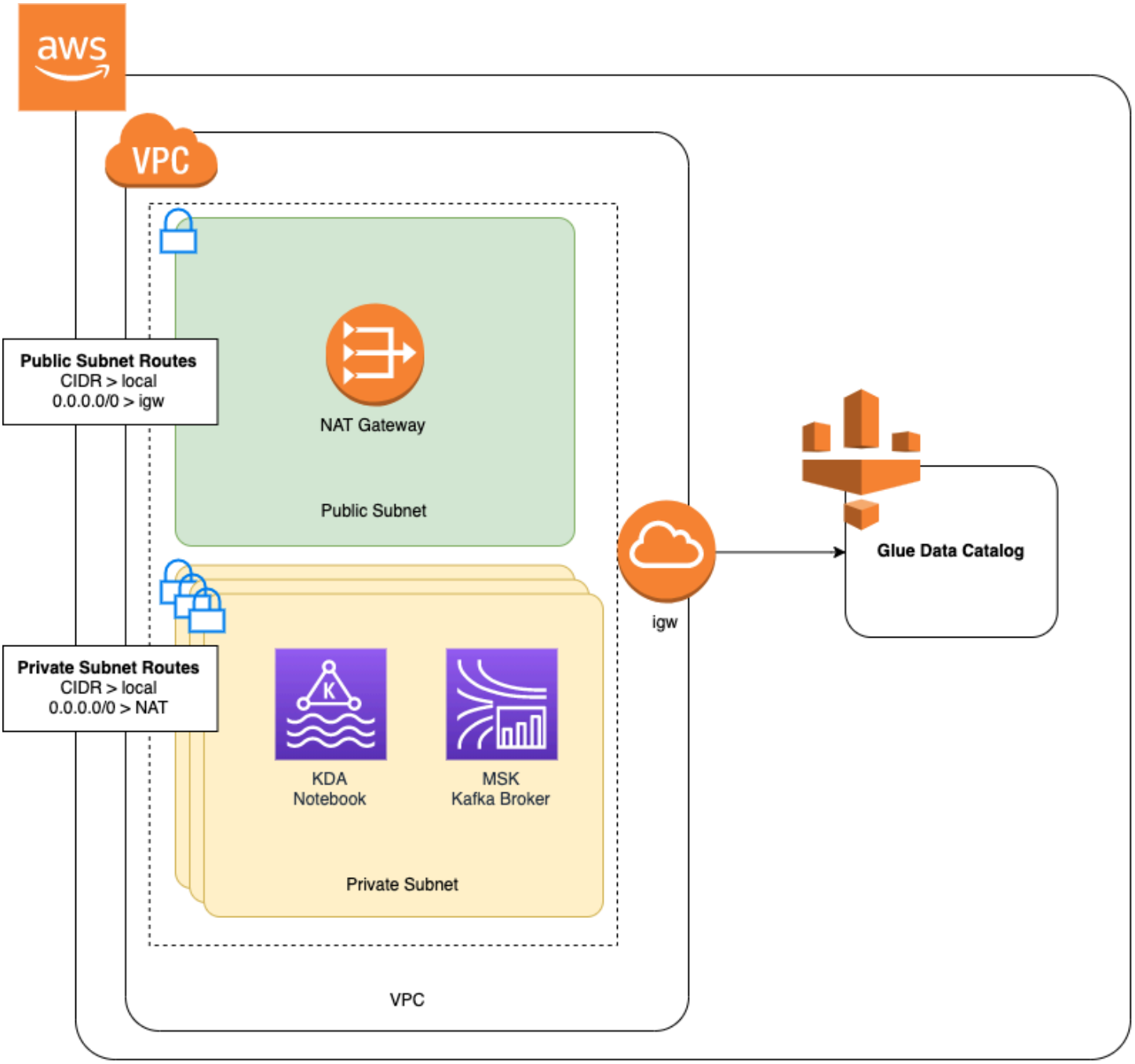
Pour ce didacticiel, vous avez besoin d'un cluster Amazon MSK qui autorise l'accès en texte brut. Si vous n'avez pas encore configuré de cluster Amazon MSK, suivez le didacticiel Mise en route avec Amazon MSK pour créer un VPC Amazon, un cluster Amazon MSK, une rubrique et une instance client Amazon EC2.

Lorsque vous suivez le didacticiel, procédez comme suit :

- Dans l'étape 3 : créer un cluster Amazon MSK, à l'étape 4, modifiez la valeur `ClientBroker` de `TLS` à **`PLAINTEXT`**.

Ajoutez une passerelle NAT à votre VPC

Si vous avez créé un cluster Amazon MSK en suivant le didacticiel Mise en route avec Amazon MSK, ou si votre VPC Amazon existant ne possède pas encore de passerelle NAT pour ses sous-réseaux privés, vous devez ajouter une passerelle NAT à votre VPC Amazon. Le schéma suivant illustre l'architecture.



Pour créer une passerelle NAT pour votre VPC Amazon, procédez comme suit :

1. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
2. Dans la barre de navigation de gauche, choisissez Passerelles NAT.
3. Sur la page Passerelles NAT, choisissez Créer une passerelle NAT.
4. Sur la page Créer une passerelle NAT, renseignez les valeurs suivantes :

Nom (facultatif)	**`ZeppelinGateway`**
Sous-réseau	AWSKafkaTutorialSubnet1
ID d'allocation d'IP Elastic	Choose an available Elastic IP. If there are no Elastic IPs available, choose Attribuer une adresse IP Elastic, and then choose the Elasic IP that the console creates.

Choisissez Créer une passerelle NAT.

5. Dans le volet de navigation de gauche, choisissez Tables de routage.
6. Choisissez Créer une table de routage.
7. Sur la page Créer une table de routage, fournissez les informations suivantes :

   - Balise de nom : **`ZeppelinRouteTable`**
   - VPC : choisissez votre VPC (par exemple AWSKafkaTutorialVPC).

   Choisissez Créer.
8. Dans la liste des tables de routage, choisissez ZeppelinRouteTable. Choisissez l'onglet Routes, puis Modifier les routes.
9. Sur la page Modifier les routes, choisissez Ajouter une route.
10. Dans le Pour Destination, saisissez **`0.0.0.0/0`**. Pour Cible, choisissez Passerelle NAT, ZeppelinGateway. Choisissez Enregistrer les routes. Choisissez Fermer.
11. Sur la page Tables de routage, avec ZeppelinRouteTable sélectionné, choisissez l'onglet Associations de sous-réseaux. Choisissez Modifier les associations de sous-réseaux.
12. Sur la page Modifier les associations de sous-réseaux, choisissez AWSKafkaTutorialSubnet2 et AWSKafkaTutorialSubnet3. Choisissez Enregistrer.

Pour créer une table et une connexion AWS Glue

Votre bloc-notes Studio utilise une base de données AWS Glue pour les métadonnées relatives à votre source de données Amazon MSK. Dans cette section, vous allez créer une connexion AWS Glue qui décrit comment accéder à votre cluster Amazon MSK et une table AWS Glue qui décrit

comment présenter les données de votre source de données à des clients tels que votre bloc-notes Studio.

Créer une connexion

1. Connectez-vous à la AWS Management Console et ouvrez la console AWS Glue à l'adresse https://console.aws.amazon.com/glue/.
2. Si vous n'avez pas encore de base de données AWS Glue, choisissez Bases de données dans la barre de navigation de gauche. Choisissez Ajouter une base de données. Dans la fenêtre Ajouter une base de données, saisissez **`default`** comme nom de la base de données. Choisissez Créer.
3. Dans le menu de navigation de gauche, sélectionnez Connexions. Choisissez Ajouter une connexion.
4. Dans la fenêtre Ajouter une connexion, indiquez les valeurs suivantes :

   - Pour Nom de connexion, saisissez **`ZeppelinConnection`**.
   - Pour Type de connexion, choisissez Kafka.
   - Pour les URL du serveur d'amorçage Kafka, fournissez la chaîne d'agent d'amorçage de votre cluster. Vous pouvez obtenir les agents d'amorçage depuis la console MSK ou en saisissant la commande CLI suivante :

     ```
     aws kafka get-bootstrap-brokers --region us-east-1 --cluster-arn ClusterArn
     ```

   - Décochez la case Exiger une connexion SSL.

   Choisissez Suivant.
5. Sur la page VPC, renseignez les valeurs suivantes :

   - Pour VPC, choisissez le nom de votre VPC (par exemple, AWSKafkaTutorialVPC).
   - Pour Sous-réseau, choisissez AWSKafkaTutorialSubnet2.
   - Pour Groupes de sécurité, sélectionnez tous les groupes disponibles.

   Choisissez Suivant.
6. Sur la page Propriétés de la connexion/Accès à la connexion, choisissez Terminer.

Création d'une table

> **Note**
>
> Vous pouvez soit créer la table manuellement comme décrit dans les étapes suivantes, soit utiliser le code du connecteur de création de table pour le service géré pour Apache Flink dans votre bloc-notes dans Apache Zeppelin pour créer votre table via une instruction DDL. Vous pouvez ensuite vérifier dans AWS Glue que la table a été créée correctement.

1. Dans la barre de navigation de gauche, choisissez Tables. Sur la page Tables, choisissez Ajouter des tables, Ajouter une table manuellement.
2. Sur la page Configurer les propriétés de votre table, saisissez **`stock`** pour Nom de la table. Assurez-vous de sélectionner la base de données que vous avez créée précédemment. Choisissez Suivant.
3. Sur la page Ajouter un magasin de données, choisissez Kafka. Pour Nom de la rubrique, saisissez le nom de votre rubrique (par exemple, AWSKafkaTutorialTopic). Pour Connexion, choisissez ZeppelinConnection.
4. Sur la page Classification, choisissez JSON. Choisissez Suivant.
5. Sur la page Définir un schéma, choisissez Ajouter une colonne pour ajouter une colonne. Ajoutez des colonnes avec les propriétés suivantes :

Nom de la colonne	Type de données
`symbole boursier`	**`chaîne`**
`prix`	**`double`**

   Choisissez Suivant.
6. Sur la page suivante, vérifiez vos paramètres, puis choisissez Terminer.
7. Choisissez la table que vous venez de créer dans la liste des tables.
8. Choisissez Modifier la table et ajoutez une propriété avec la clé `managed-flink.proctime` et la valeur `proctime`.
9. Choisissez Appliquer.

Créer un bloc-notes Studio avec Amazon MSK

Maintenant que vous avez créé les ressources utilisées par votre application, vous pouvez créer votre bloc-notes Studio.

Vous pouvez créer votre application à l'aide de la AWS Management Console ou de l'interface AWS CLI.

- Créer un bloc-notes Studio à l'aide de la AWS Management Console
- Créer un bloc-notes Studio à l'aide de l'interface AWS CLI

> **Note**
>
> Vous pouvez également créer un bloc-notes Studio à partir de la console Amazon MSK en choisissant un cluster existant, puis en choisissant Traiter les données en temps réel.

Créer un bloc-notes Studio à l'aide de la AWS Management Console

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard.
2. Sur la page Applications de service géré pour Apache Flink, choisissez l'onglet Studio. Choisissez Créer un bloc-notes Studio.

   > **Note**
   >
   > Pour créer un bloc-notes Studio à partir des consoles Amazon MSK ou Kinesis Data Streams, sélectionnez votre cluster Amazon MSK ou votre flux de données Kinesis d'entrée, puis choisissez Traiter les données en temps réel.

3. Sur la page Créer un bloc-notes Studio, fournissez les informations suivantes :

   - Pour Nom du bloc-notes Studio, saisissez **`MyNotebook`**.
   - Pour Base de données AWS Glue, choisissez Par défaut.

   Choisissez Créer un bloc-notes Studio.
4. Sur la page MonBloc-notes, choisissez l'onglet Configuration. Dans la section Mise en réseau, choisissez Modifier.

5. Sur la page Modifier le réseau pour MonBloc-notes, choisissez Configuration VPC basée sur le cluster Amazon MSK. Choisissez votre cluster Amazon MSK pour Cluster Amazon MSK. Choisissez Enregistrer les modifications.
6. Sur la page MonBloc-notes, choisissez Exécuter. Attendez que État indique En cours d'exécution.

Créer un bloc-notes Studio à l'aide de l’interface AWS CLI

Pour créer votre bloc-notes Studio à l'aide de l’interface AWS CLI, procédez comme suit :

1. Assurez-vous de disposer des informations suivantes. Vous avez besoin de ces valeurs pour créer votre application.
   - Votre ID de compte.
   - ID de sous-réseau et l'ID du groupe de sécurité pour le VPC Amazon qui contient votre cluster Amazon MSK.
2. Créez un fichier nommé `create.json` avec le contenu suivant. Remplacez les valeurs des espaces réservés par vos informations.

```
{
    "ApplicationName": "MyNotebook",
    "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
    "ApplicationMode": "INTERACTIVE",
    "ServiceExecutionRole": "arn:aws:iam::AccountID:role/ZeppelinRole",
    "ApplicationConfiguration": {
        "ApplicationSnapshotConfiguration": {
            "SnapshotsEnabled": false
        },
        "VpcConfigurations": [
            {
                "SubnetIds": [
                    "SubnetID 1",
                    "SubnetID 2",
                    "SubnetID 3"
                ],
                "SecurityGroupIds": [
                    "VPC Security Group ID"
                ]
            }
        ],
```

```
        "ZeppelinApplicationConfiguration": {
            "CatalogConfiguration": {
                "GlueDataCatalogConfiguration": {
                    "DatabaseARN": "arn:aws:glue:us-east-1:AccountID:database/
default"
                }
            }
        }
    }
}
```

3. Pour créer votre application, exécutez la commande suivante.

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create.json
```

4. Lorsque la commande est terminée, vous devriez obtenir une sortie similaire à celle qui suit, montrant les détails de votre nouveau bloc-notes Studio :

```
{
    "ApplicationDetail": {
        "ApplicationARN": "arn:aws:kinesisanalyticsus-
east-1:012345678901:application/MyNotebook",
        "ApplicationName": "MyNotebook",
        "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
        "ApplicationMode": "INTERACTIVE",
        "ServiceExecutionRole": "arn:aws:iam::012345678901:role/ZeppelinRole",
...
```

5. Pour lancer votre application, exécutez la commande suivante. Remplacez les exemples de valeur par l'identifiant de votre compte.

```
aws kinesisanalyticsv2 start-application --application-arn
 arn:aws:kinesisanalyticsus-east-1:012345678901:application/MyNotebook\
```

Envoyer des données à votre cluster Amazon MSK

Dans cette section, vous allez exécuter un script Python dans votre client Amazon EC2 pour envoyer des données à votre source de données Amazon MSK.

1. Connectez-vous à votre client Amazon EC2.

2. Exécutez les commandes suivantes pour installer Python version 3, Pip et le package Kafka pour Python, puis confirmez les actions :

```
sudo yum install python37
curl -O https://bootstrap.pypa.io/get-pip.py
python3 get-pip.py --user
pip install kafka-python
```

3. Configurez l'interface AWS CLI sur votre machine cliente en saisissant la commande suivante :

```
aws configure
```

Fournissez les informations d'identification de votre compte, et **`us-east-1`** pour la `region`.

4. Créez un fichier nommé `stock.py` avec le contenu suivant. Remplacez l'exemple de valeur par la chaîne d'agent d'amorçage de votre cluster Amazon MSK et mettez à jour le nom de la rubrique si celui-ci n'est pas AWSKafkaTutorialTopic :

```
from kafka import KafkaProducer
import json
import random
from datetime import datetime

BROKERS = "<<Bootstrap Broker List>>"
producer = KafkaProducer(
    bootstrap_servers=BROKERS,
    value_serializer=lambda v: json.dumps(v).encode('utf-8'),
    retry_backoff_ms=500,
    request_timeout_ms=20000,
    security_protocol='PLAINTEXT')


def getStock():
    data = {}
    now = datetime.now()
    str_now = now.strftime("%Y-%m-%d %H:%M:%S")
    data['event_time'] = str_now
    data['ticker'] = random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV'])
    price = random.random() * 100
    data['price'] = round(price, 2)
    return data
```

```
while True:
    data =getStock()
    # print(data)
    try:
        future = producer.send("AWSKafkaTutorialTopic", value=data)
        producer.flush()
        record_metadata = future.get(timeout=10)
        print("sent event to Kafka! topic {} partition {} offset
 {}".format(record_metadata.topic, record_metadata.partition,
 record_metadata.offset))
    except Exception as e:
        print(e.with_traceback())
```

5. Exécutez le script avec la commande suivante :

```
$ python3 stock.py
```

6. Laissez le script s'exécuter pendant que vous complétez la section suivante.

Tester votre bloc-notes Studio

Dans cette section, vous utilisez votre bloc-notes Studio pour interroger les données de votre cluster Amazon MSK.

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard.
2. Sur la page Applications de service géré pour Apache Flink, choisissez l'onglet Bloc-notes Studio. Choisissez MonBloc-notes.
3. Sur la page MonBloc-notes, choisissez Ouvrir dans Apache Zeppelin.

   L'interface Apache Zeppelin s'ouvre dans un nouvel onglet.
4. Sur la page Bienvenue sur Zeppelin !, choisissez Nouvelle note Zeppelin.
5. Sur la page Note Zeppelin, entrez la requête suivante dans une nouvelle note :

   ```
   %flink.ssql(type=update)
   select * from stock
   ```

   Choisissez l'icône d'exécution.

L'application affiche les données du cluster Amazon MSK.

Pour ouvrir le tableau de bord Apache Flink de votre application afin de visualiser les aspects opérationnels, choisissez FLINK JOB. Pour plus d'informations sur le tableau de bord Flink, consultez Tableau de bord Apache Flink dans le guide du développeur du service géré pour Apache Flink.

Pour d'autres exemples de requêtes SQL Flink Streaming, consultez la section Queries de la documentation Apache Flink.

Nettoyage de votre application et des ressources dépendantes

Configurez votre bloc-notes Studio.

1. Ouvrez la console du service géré pour Apache Flink.
2. Choisissez MonBloc-notes.
3. Choisissez Actions, puis Supprimer.

Supprimer votre base de données AWS Glue et votre connexion

1. Ouvrez la console AWS Glue, à l'adresse https://console.aws.amazon.com/glue/.
2. Dans la barre de navigation de gauche, choisissez Bases de données. Cochez la case à côté de Défaut pour le sélectionner. Choisissez Action, Supprimer la base de données. Confirmez votre sélection.
3. Dans le menu de navigation de gauche, sélectionnez Connexions. Cochez la case à côté de ZeppelinConnection pour le sélectionner. Choisissez Action, puis Supprimer la connexion. Confirmez votre sélection.

Pour supprimer la stratégie et le rôle IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le menu de navigation de gauche, choisissez Rôles.
3. Utilisez la barre de recherche pour rechercher le rôle ZeppelinRole.
4. Choisissez le rôle ZeppelinRole. Choisissez Supprimer le rôle. Confirmez la suppression.

Supprimer votre groupe de journaux CloudWatch

La console crée un groupe de journaux CloudWatch et un flux de journaux pour vous lorsque vous créez votre application à l'aide de la console. Vous n'avez pas de groupe ni flux de journaux si vous avez créé votre application à l'aide de l'interface AWS CLI.

1. Ouvrez la console CloudWatch à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans le menu de navigation de gauche, choisissez Groupe de journaux.
3. Choisissez le groupe de journaux /AWS/KinesisAnalytics/MyNotebook.
4. Sélectionnez Actions, Supprimer le ou les groupes de journaux. Confirmez la suppression.

Nettoyer les ressources Kinesis Data Streams

Pour supprimer votre flux Kinesis, ouvrez la console Kinesis Data Streams, sélectionnez votre flux Kinesis, puis choisissez Actions, Supprimer.

Nettoyage des ressources MSK

Suivez les étapes de cette section si vous avez créé un cluster Amazon MSK pour ce didacticiel. Cette section contient des instructions pour nettoyer votre instance client Amazon EC2, Amazon VPC et votre cluster Amazon MSK.

Supprimer votre cluster Amazon MSK

Suivez ces étapes si vous avez créé un cluster Amazon MSK pour ce didacticiel.

1. Ouvrez la console Amazon MSK à l'adresse https://console.aws.amazon.com/msk/home?region=us-east-1#/home/.
2. Choisissez AWS KafkaTutorialCluster. Choisissez Supprimer. Saisissez **`delete`** dans la fenêtre qui apparaît et confirmez votre sélection.

Résilier une instance client

Suivez ces étapes si vous avez créé une instance de client Amazon EC2 pour ce didacticiel.

1. Ouvrez la console Amazon EC2 sur https://console.aws.amazon.com/ec2/.
2. Dans la barre de navigation de gauche, choisissez Instances.
3. Cochez la case à côté de ZeppelinClient pour le sélectionner.

4. Choisissez État de l'instance, Résilier l'instance.

Supprimer votre Amazon VPC

Suivez ces étapes si vous avez créé un Amazon VPC pour ce didacticiel.

1. Ouvrez la console Amazon EC2 sur https://console.aws.amazon.com/ec2/.
2. Choisissez Interfaces réseau dans la barre de navigation de gauche.
3. Saisissez l'identifiant de VPC dans la barre de recherche, puis appuyez sur Entrée.
4. Cochez la case dans l'en-tête du tableau pour sélectionner toutes les interfaces réseau affichées.
5. Sélectionnez Actions, Détacher. Dans la fenêtre qui apparaît, choisissez Activer sous Forcer le détachement. Choisissez Détacher et attendez que toutes les interfaces réseau atteignent l'état Disponible.
6. Cochez la case dans l'en-tête du tableau pour sélectionner à nouveau toutes les interfaces réseau affichées.
7. Sélectionnez Actions, Supprimer. Confirmez l'action.
8. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
9. Sélectionnez AWS KafkaTutorialVPC. Choisissez Actions, Supprimer le VPC. Saisissez **`delete`** et confirmez la suppression.

Didacticiel : déploiement en tant qu'application à état durable

Le didacticiel suivant explique comment déployer un bloc-notes Studio en tant qu'application état durable de service géré pour Apache Flink.

Ce didacticiel contient les sections suivantes :

- Installation
- Déployer une application à état durable à l'aide de la AWS Management Console
- Déployer une application à état durable à l'aide de l'interface AWS CLI

Installation

Créez un nouveau bloc-notes Studio en suivant Didacticiel de création d'un bloc-notes Studio, à l'aide de Kinesis Data Streams ou d'Amazon MSK. Nommez le bloc-notes Studio `ExampleTestDeploy`.

Déployer une application à état durable à l'aide de la AWS Management Console

1. Ajoutez un emplacement de compartiment S3 là où vous souhaitez que le code empaqueté soit stocké sous Emplacement du code d'application (facultatif) dans la console. Cela permet de suivre les étapes de déploiement et d'exécution de votre application directement depuis le bloc-notes.
2. Ajoutez les autorisations requises au rôle d'application pour activer le rôle que vous utilisez pour lire et écrire dans un compartiment Amazon S3 et pour lancer une application de service géré pour Apache Flink :

   - AmazonS3FullAccess
   - Amazonmanaged-flinkFullAccess
   - Accédez à vos sources, destinations et VPC, le cas échéant. Pour de plus amples informations, veuillez consulter Autorisations IAM pour les blocs-notes Studio.
3. Utilisez l'exemple de code suivant :

```
%flink.ssql(type=update)
CREATE TABLE exampleoutput (
  'ticket' VARCHAR,
  'price' DOUBLE
)
WITH (
  'connector' = 'kinesis',
  'stream' = 'ExampleOutputStream',
  'aws.region' = 'us-east-1',
  'scan.stream.initpos' = 'LATEST',
  'format' = 'json'
);

INSERT INTO exampleoutput SELECT ticker, price FROM exampleinputstream
```

4. Avec le lancement de cette fonctionnalité, vous verrez une nouvelle liste déroulante dans le coin supérieur droit de chaque note de votre bloc-notes avec le nom du bloc-notes. Vous pouvez effectuer les actions suivantes :

   - Consultez les paramètres du bloc-notes Studio dans la AWS Management Console.
   - Créez votre note Zeppelin et exportez-la sur Amazon S3. À ce stade, donnez un nom à votre application et choisissez Générer et exporter. Vous recevrez une notification lorsque l'exportation sera terminée.

- Si nécessaire, vous pouvez consulter et exécuter des tests supplémentaires sur l'exécutable dans Amazon S3.
- Une fois la compilation terminée, vous pourrez déployer votre code sous la forme d'une application de streaming Kinesis dotée d'un état durable et de l’autoscaling.
- Utilisez le menu déroulant et choisissez Déployer la note Zeppelin en tant qu'application de streaming Kinesis. Vérifiez le nom de l'application et choisissez Déployer via la console AWS.
- Cela vous mènera à la page de la AWS Management Console de création d'une application de service géré pour Apache Flink. Notez que le nom de l'application, le parallélisme, l'emplacement du code, Glue DB par défaut, le VPC (le cas échéant) et les rôles IAM ont été préremplis. Vérifiez que les rôles IAM disposent des autorisations requises pour accéder à vos sources et destinations. Les instantanés sont activés par défaut pour la gestion des applications à état durable.
- Choisissez Créer une application.
- Vous pouvez choisir de configurer et de modifier tous les paramètres, puis choisir Exécuter pour démarrer votre application de streaming.

Déployer une application à état durable à l'aide de l’interface AWS CLI

Pour déployer une application à l'aide de l’interface AWS CLI, vous devez mettre à jour votre interface AWS CLI pour utiliser le modèle de service fourni dans vos informations sur la version bêta 2. Pour de plus amples informations sur l'utilisation du modèle de service mis à jour, veuillez consulter Installation.

L'exemple suivant permet de créer un nouveau bloc-notes Studio :

```
aws kinesisanalyticsv2 create-application \
     --application-name <app-name> \
     --runtime-environment ZEPPELIN-FLINK-3_0 \
     --application-mode INTERACTIVE \
     --service-execution-role <iam-role>
     --application-configuration '{
       "ZeppelinApplicationConfiguration": {
         "CatalogConfiguration": {
           "GlueDataCatalogConfiguration": {
             "DatabaseARN": "arn:aws:glue:us-east-1:<account>:database/<glue-database-
name>"
           }
         }
```

```
      },
      "FlinkApplicationConfiguration": {
        "ParallelismConfiguration": {
          "ConfigurationType": "CUSTOM",
          "Parallelism": 4,
          "ParallelismPerKPU": 4
        }
      },
      "DeployAsApplicationConfiguration": {
           "S3ContentLocation": {
              "BucketARN": "arn:aws:s3:::<s3bucket>",
              "BasePath": "/something/"
           }
       },
      "VpcConfigurations": [
        {
          "SecurityGroupIds": [
            "<security-group>"
          ],
          "SubnetIds": [
            "<subnet-1>",
            "<subnet-2>"
          ]
        }
      ]
    }' \
    --region us-east-1
```

L'exemple suivant permet de lancer un bloc-notes Studio :

```
aws kinesisanalyticsv2 start-application \
    --application-name <app-name> \
    --region us-east-1 \
    --no-verify-ssl
```

Le code suivant renvoie l'URL de la page du bloc-notes Apache Zeppelin d'une application :

```
aws kinesisanalyticsv2 create-application-presigned-url \
    --application-name <app-name> \
    --url-type ZEPPELIN_UI_URL \

    --region us-east-1 \
```

```
    --no-verify-ssl
```

Exemples

Les exemples de requêtes suivants montrent comment analyser des données à l'aide de requêtes Windows dans un bloc-notes Studio.

- Création de tables avec Amazon MSK/Apache Kafka
- Création de tables avec Kinesis
- Fenêtre bascule
- Fenêtre défilante
- Requête SQL interactive
- BlackHole Connecteur SQL
- Générateur de données
- Requête Scala interactive
- Requête Python interactive
- Requêtes Python, SQL et Scala interactives
- Flux de données Kinesis intercompte

Pour plus d'informations sur les paramètres de requête SQL d'Apache Flink, consultez Flink on Zeppelin Notebooks for Interactive Data Analysis.

Pour afficher votre applicaion dans le tableau de bord Apache Flink, choisissez FLINK JOB sur la page Note Zeppelin de votre application.

Pour plus d'informations sur les requêtes de fenêtre, consultez Windows dans la documentation Apache Flink.

Pour d'autres exemples de requêtes SQL Apache Flink Streaming, consultez la section Queries de la documentation Apache Flink.

Création de tables avec Amazon MSK/Apache Kafka

Vous pouvez utiliser le connecteur Amazon MSK Flink avec le service géré pour Apache Flink Studio afin d'authentifier votre connexion à l'aide de l'authentification en texte brut, SSL ou IAM. Créez vos tables en utilisant les propriétés spécifiques selon vos besoins.

```
-- Plaintext connection
```

```
CREATE TABLE your_table (
  `column1` STRING,
  `column2` BIGINT
) WITH (
  'connector' = 'kafka',
  'topic' = 'your_topic',
  'properties.bootstrap.servers' = '<bootstrap servers>',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json'
);

-- SSL connection

CREATE TABLE your_table (
  `column1` STRING,
  `column2` BIGINT
) WITH (
  'connector' = 'kafka',
   'topic' = 'your_topic',
  'properties.bootstrap.servers' = '<bootstrap servers>',
  'properties.security.protocol' = 'SSL',
  'properties.ssl.truststore.location' = '/usr/lib/jvm/java-11-amazon-corretto/lib/
security/cacerts',
  'properties.ssl.truststore.password' = 'changeit',
  'properties.group.id' = 'myGroup',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json'
);

-- IAM connection (or for MSK Serverless)

CREATE TABLE your_table (
  `column1` STRING,
  `column2` BIGINT
) WITH (
  'connector' = 'kafka',
  'topic' = 'your_topic',
  'properties.bootstrap.servers' = '<bootstrap servers>',
  'properties.security.protocol' = 'SASL_SSL',
  'properties.sasl.mechanism' = 'AWS_MSK_IAM',
  'properties.sasl.jaas.config' = 'software.amazon.msk.auth.iam.IAMLoginModule
 required;',
```

```
  'properties.sasl.client.callback.handler.class' =
 'software.amazon.msk.auth.iam.IAMClientCallbackHandler',
  'properties.group.id' = 'myGroup',
  'scan.startup.mode' = 'earliest-offset',
  'format' = 'json'
);
```

Vous pouvez les combiner avec d'autres propriétés sur le connecteur SQL Apache Kafka.

Création de tables avec Kinesis

Dans l'exemple suivant, vous créez une table à l'aide de Kinesis :

```
CREATE TABLE KinesisTable (
  `column1` BIGINT,
  `column2` BIGINT,
  `column3` BIGINT,
  `column4` STRING,
  `ts` TIMESTAMP(3)
)
PARTITIONED BY (column1, column2)
WITH (
  'connector' = 'kinesis',
  'stream' = 'test_stream',
  'aws.region' = '<region>',
  'scan.stream.initpos' = 'LATEST',
  'format' = 'csv'
);
```

Pour plus d'informations sur les autres propriétés que vous pouvez utiliser, consultez Amazon Kinesis Data Streams SQL Connector.

Fenêtre bascule

La requête SQL Flink Streaming suivante sélectionne le prix le plus élevé pour chaque fenêtre bascule de cinq secondes dans la table `ZeppelinTopic` :

```
%flink.ssql(type=update)
SELECT TUMBLE_END(event_time, INTERVAL '5' SECOND) as winend, MAX(price) as
 five_second_high, ticker
FROM ZeppelinTopic
```

```
GROUP BY ticker, TUMBLE(event_time, INTERVAL '5' SECOND)
```

Fenêtre défilante

La requête SQL Apache Flink Streaming suivante sélectionne le prix le plus élevé pour chaque fenêtre défilante de cinq secondes dans la table `ZeppelinTopic` :

```
%flink.ssql(type=update)
SELECT HOP_END(event_time, INTERVAL '3' SECOND, INTERVAL '5' SECOND) AS winend,
 MAX(price) AS sliding_five_second_max
FROM ZeppelinTopic//or your table name in AWS Glue
GROUP BY HOP(event_time, INTERVAL '3' SECOND, INTERVAL '5' SECOND)
```

Requête SQL interactive

Cet exemple imprime la durée maximale de l'événement et le temps de traitement, ainsi que la somme des valeurs de la table des valeurs clés. Assurez-vous que vous disposez de l'exemple de script de génération de données de l'exécution the section called “Générateur de données”. Pour essayer d'autres requêtes SQL telles que le filtrage et les jointures dans votre bloc-notes Studio, consultez Queries dans la documentation Apache Flink.

```
%flink.ssql(type=single, parallelism=4, refreshInterval=1000, template=<h1>{2}</h1>
 records seen until <h1>Processing Time: {1}</h1> and <h1>Event Time: {0}</h1>)

-- An interactive query prints how many records from the `key-value-stream` we have
 seen so far, along with the current processing and event time.
SELECT
  MAX(`et`) as `et`,
  MAX(`pt`) as `pt`,
  SUM(`value`) as `sum`
FROM
  `key-values`
```

```
%flink.ssql(type=update, parallelism=4, refreshInterval=1000)

-- An interactive tumbling window query that displays the number of records observed
 per (event time) second.
-- Browse through the chart views to see different visualizations of the streaming
 result.
SELECT
```

```
  TUMBLE_START(`et`, INTERVAL '1' SECONDS) as `window`,
  `key`,
  SUM(`value`) as `sum`
FROM
  `key-values`
GROUP BY
  TUMBLE(`et`, INTERVAL '1' SECONDS),
  `key`;
```

BlackHole Connecteur SQL

Le connecteur BlackHole SQL ne vous oblige pas à créer un flux de données Kinesis ou un cluster Amazon MSK pour tester vos requêtes. Pour plus d'informations sur le connecteur BlackHole SQL, consultez la section Connecteur BlackHole SQL dans la documentation d'Apache Flink. Dans cet exemple, le catalogue par défaut est un catalogue en mémoire.

```
%flink.ssql

CREATE TABLE default_catalog.default_database.blackhole_table (
 `key` BIGINT,
 `value` BIGINT,
 `et` TIMESTAMP(3)
) WITH (
 'connector' = 'blackhole'
)
```

```
%flink.ssql(parallelism=1)

INSERT INTO `test-target`
SELECT
  `key`,
  `value`,
  `et`
FROM
  `test-source`
WHERE
  `key` > 3
```

```
%flink.ssql(parallelism=2)

INSERT INTO `default_catalog`.`default_database`.`blackhole_table`
```

```
SELECT
  `key`,
  `value`,
  `et`
FROM
  `test-target`
WHERE
  `key` > 7
```

Générateur de données

Cet exemple utilise Scala pour générer des exemples de données. Vous pouvez utiliser ces exemples de données pour tester différentes requêtes. Utilisez l'instruction create table pour créer la table des valeurs clés.

```
import org.apache.flink.streaming.api.functions.source.datagen.DataGeneratorSource
import org.apache.flink.streaming.api.functions.source.datagen.RandomGenerator
import org.apache.flink.streaming.api.scala.DataStream

import java.sql.Timestamp

// ad-hoc convenience methods to be defined on Table
implicit class TableOps[T](table: DataStream[T]) {
    def asView(name: String): DataStream[T] = {
      if (stenv.listTemporaryViews.contains(name)) {
        stenv.dropTemporaryView("`" + name + "`")
      }
      stenv.createTemporaryView("`" + name + "`", table)
      return table;
    }
}
```

```
%flink(parallelism=4)
val stream = senv
 .addSource(new DataGeneratorSource(RandomGenerator.intGenerator(1, 10), 1000))
 .map(key => (key, 1, new Timestamp(System.currentTimeMillis)))
 .asView("key-values-data-generator")
```

```
%flink.ssql(parallelism=4)
-- no need to define the paragraph type with explicit parallelism (such as
 "%flink.ssql(parallelism=2)")
```

```
-- in this case the INSERT query will inherit the parallelism of the of the above
 paragraph
INSERT INTO `key-values`
SELECT
 `_1` as `key`,
 `_2` as `value`,
 `_3` as `et`
FROM
 `key-values-data-generator`
```

Requête Scala interactive

Il s'agit de la traduction Scala de the section called “Requête SQL interactive”. Pour d'autres exemples de Scala, consultez Table API dans la documentation d'Apache Flink.

```
%flink
import org.apache.flink.api.scala._
import org.apache.flink.table.api._
import org.apache.flink.table.api.bridge.scala._

// ad-hoc convenience methods to be defined on Table
implicit class TableOps(table: Table) {
    def asView(name: String): Table = {
      if (stenv.listTemporaryViews.contains(name)) {
        stenv.dropTemporaryView(name)
      }
      stenv.createTemporaryView(name, table)
      return table;
    }
}
```

```
%flink(parallelism=4)

// A view that computes many records from the `key-values` we have seen so far, along
 with the current processing and event time.
val query01 = stenv
  .from("`key-values`")
  .select(
    $"et".max().as("et"),
    $"pt".max().as("pt"),
    $"value".sum().as("sum")
  ).asView("query01")
```

```
%flink.ssql(type=single, parallelism=16, refreshInterval=1000, template=<h1>{2}</h1>
 records seen until <h1>Processing Time: {1}</h1> and <h1>Event Time: {0}</h1>)

-- An interactive query prints the query01 output.
SELECT * FROM query01
```

```
%flink(parallelism=4)

// An tumbling window view that displays the number of records observed per (event
 time) second.
val query02 = stenv
  .from("`key-values`")
  .window(Tumble over 1.seconds on $"et" as $"w")
  .groupBy($"w", $"key")
  .select(
    $"w".start.as("window"),
    $"key",
    $"value".sum().as("sum")
  ).asView("query02")
```

```
%flink.ssql(type=update, parallelism=4, refreshInterval=1000)

-- An interactive query prints the query02 output.
-- Browse through the chart views to see different visualizations of the streaming
 result.
SELECT * FROM `query02`
```

Requête Python interactive

Il s'agit de la traduction Python de the section called “Requête SQL interactive”. Pour d'autres exemples de Python, consultez Table API dans la documentation d'Apache Flink.

```
%flink.pyflink
from pyflink.table.table import Table

def as_view(table, name):
  if (name in st_env.list_temporary_views()):
    st_env.drop_temporary_view(name)
  st_env.create_temporary_view(name, table)
  return table
```

```
Table.as_view = as_view
```

```
%flink.pyflink(parallelism=16)

# A view that computes many records from the `key-values` we have seen so far, along
 with the current processing and event time
st_env \
  .from_path("`keyvalues`") \
  .select(", ".join([
    "max(et) as et",
    "max(pt) as pt",
    "sum(value) as sum"
  ])) \
  .as_view("query01")
```

```
%flink.ssql(type=single, parallelism=16, refreshInterval=1000, template=<h1>{2}</h1>
 records seen until <h1>Processing Time: {1}</h1> and <h1>Event Time: {0}</h1>)

-- An interactive query prints the query01 output.
SELECT * FROM query01
```

```
%flink.pyflink(parallelism=16)

# A view that computes many records from the `key-values` we have seen so far, along
 with the current processing and event time
st_env \
  .from_path("`key-values`") \
  .window(Tumble.over("1.seconds").on("et").alias("w")) \
  .group_by("w, key") \
  .select(", ".join([
    "w.start as window",
    "key",
    "sum(value) as sum"
  ])) \
  .as_view("query02")
```

```
%flink.ssql(type=update, parallelism=16, refreshInterval=1000)

-- An interactive query prints the query02 output.
-- Browse through the chart views to see different visualizations of the streaming
 result.
```

```
SELECT * FROM `query02`
```

Requêtes Python, SQL et Scala interactives

Vous pouvez utiliser n'importe quelle combinaison de SQL, Python et Scala dans votre bloc-notes pour une analyse interactive. Dans un bloc-notes Studio que vous envisagez de déployer en tant qu'application à état durable, vous pouvez utiliser une combinaison de SQL et de Scala. Cet exemple montre les sections qui sont ignorées et celles qui sont déployées dans l'application à état durable.

```
%flink.ssql
CREATE TABLE `default_catalog`.`default_database`.`my-test-source` (
  `key` BIGINT NOT NULL,
  `value` BIGINT NOT NULL,
  `et` TIMESTAMP(3) NOT NULL,
  `pt` AS PROCTIME(),
  WATERMARK FOR `et` AS `et` - INTERVAL '5' SECOND
)
WITH (
  'connector' = 'kinesis',
  'stream' = 'kda-notebook-example-test-source-stream',
  'aws.region' = 'eu-west-1',
  'scan.stream.initpos' = 'LATEST',
  'format' = 'json',
  'json.timestamp-format.standard' = 'ISO-8601'
)
```

```
%flink.ssql
CREATE TABLE `default_catalog`.`default_database`.`my-test-target` (
  `key` BIGINT NOT NULL,
  `value` BIGINT NOT NULL,
  `et` TIMESTAMP(3) NOT NULL,
  `pt` AS PROCTIME(),
  WATERMARK FOR `et` AS `et` - INTERVAL '5' SECOND
)
WITH (
  'connector' = 'kinesis',
  'stream' = 'kda-notebook-example-test-target-stream',
  'aws.region' = 'eu-west-1',
  'scan.stream.initpos' = 'LATEST',
  'format' = 'json',
  'json.timestamp-format.standard' = 'ISO-8601'
)
```

```
%flink()

// ad-hoc convenience methods to be defined on Table
implicit class TableOps(table: Table) {
  def asView(name: String): Table = {
    if (stenv.listTemporaryViews.contains(name)) {
      stenv.dropTemporaryView(name)
    }
    stenv.createTemporaryView(name, table)
    return table;
  }
}
```

```
%flink(parallelism=1)
val table = stenv
  .from("`default_catalog`.`default_database`.`my-test-source`")
  .select($"key", $"value", $"et")
  .filter($"key" > 10)
  .asView("query01")
```

```
%flink.ssql(parallelism=1)

-- forward data
INSERT INTO `default_catalog`.`default_database`.`my-test-target`
SELECT * FROM `query01`
```

```
%flink.ssql(type=update, parallelism=1, refreshInterval=1000)

-- forward data to local stream (ignored when deployed as application)
SELECT * FROM `query01`
```

```
%flink

// tell me the meaning of life (ignored when deployed as application!)
print("42!")
```

Flux de données Kinesis intercompte

Pour utiliser un flux de données Kinesis se trouvant dans un compte autre que le compte associé à votre bloc-notes Studio, créez un rôle d'exécution de service dans le compte sur lequel votre

bloc-notes Studio est exécuté et une politique d'approbation des rôles dans le compte contenant le flux de données. Utilisez `aws.credentials.provider`, `aws.credentials.role.arn` et `aws.credentials.role.sessionName` dans le connecteur Kinesis dans votre instruction DDL de création de table pour créer une table par rapport au flux de données.

Utilisez le rôle d'exécution de service suivant pour le compte de bloc-notes Studio.

```
{
 "Sid": "AllowNotebookToAssumeRole",
 "Effect": "Allow",
 "Action": "sts:AssumeRole"
 "Resource": "*"
}
```

Utilisez la politique `AmazonKinesisFullAccess` et la politique d'approbation des rôles suivante pour le compte de flux de données.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::<accountID>:root"
 },
 "Action": "sts:AssumeRole",
 "Condition": {}
 }
 ]
}
```

Utilisez le paragraphe suivant pour l'instruction de création de table.

```
%flink.ssql
CREATE TABLE test1 (
name VARCHAR,
age BIGINT
) WITH (
'connector' = 'kinesis',
'stream' = 'stream-assume-role-test',
'aws.region' = 'us-east-1',
'aws.credentials.provider' = 'ASSUME_ROLE',
```

```
'aws.credentials.role.arn' = 'arn:aws:iam::<accountID>:role/stream-assume-role-test-
role',
'aws.credentials.role.sessionName' = 'stream-assume-role-test-session',
'scan.stream.initpos' = 'TRIM_HORIZON',
'format' = 'json'
)
```

Résolution des problèmes

Cette section contient des informations de dépannage pour les blocs-notes Studio.

Arrêter une application bloquée

Pour arrêter une application bloquée dans un état transitoire, appelez l'StopApplicationaction avec le `Force` paramètre défini sur. `true` Pour plus d'informations, consultez Application en cours d'exécution dans le Guide du développeur du service géré pour Apache Flink.

Déploiement en tant qu'application à état durable dans un VPC sans accès à Internet

La deploy-as-application fonction Managed Service for Apache Flink Studio ne prend pas en charge les applications VPC sans accès à Internet. Nous vous recommandons de créer votre application dans Studio, puis d'utiliser le service géré pour Apache Flink pour créer manuellement une application Flink et sélectionner le fichier zip que vous avez créé dans votre bloc-notes.

La procédure suivante montre comment procéder :

1. Créez et exportez votre application Studio vers Amazon S3. Il doit s'agir d'un fichier zip.
2. Créez manuellement une application de service géré pour Apache Flink avec un chemin de code faisant référence à l'emplacement du fichier zip dans Amazon S3. En outre, vous devrez configurer l'application avec les variables `env` suivantes (2 `groupID`, 3 `var` au total) :
3. kinesis.analytics.flink.run.options
    a. python : source/note.py
    b. fichier jar : PythonApplicationDependencies lib/ .jar
4. managed.deploy_as_app.options
    - DatabaseARN : <glue database ARN (Amazon Resource Name)>

5. Vous devrez peut-être accorder des autorisations aux rôles IAM du service géré pour Apache Flink Studio et du service géré pour Apache Flink pour les services utilisés par votre application. Vous pouvez utiliser le même rôle IAM pour les deux applications.

eploy-as-app Taille D et réduction du temps de fabrication

Les applications Studio deploy-as-app for Python regroupent tout ce qui est disponible dans l'environnement Python, car nous ne pouvons pas déterminer les bibliothèques dont vous avez besoin. Cela peut entraîner une taille plus grande que nécessaire deploy-as-app . La procédure suivante montre comment réduire la taille de l'application deploy-as-app Python en désinstallant les dépendances.

Si vous créez une application Python avec des deploy-as-app fonctionnalités de Studio, vous pouvez envisager de supprimer les packages Python préinstallés du système si vos applications n'en dépendent pas. Cela permettra non seulement de réduire la taille finale de l'artefact afin d'éviter de dépasser la limite de service pour la taille de l'application, mais également d'améliorer le temps de création des applications dotées de cette fonctionnalité. deploy-as-app

Vous pouvez exécuter la commande suivante pour répertorier tous les packages Python installés avec leur taille d'installation respective et supprimer de manière sélective les packages volumineux.

```
%flink.pyflink

!pip list --format freeze | awk -F = {'print $1'} | xargs pip show | grep -E
 'Location:|Name:' | cut -d ' ' -f 2 | paste -d ' ' - - | awk '{gsub("-","_",$1); print
 $2 "/" tolower($1)}' | xargs du -sh 2> /dev/null | sort -hr
```

> **Note**
>
> `apache-beam` est requis pour que Flink Python fonctionne. Vous ne devez jamais supprimer ce package et ses dépendances.

Voici la liste des packages Python préinstallés dans Studio V2 dont la suppression peut être envisagée :

```
scipy
statsmodels
```

```
plotnine
seaborn
llvmlite
bokeh
pandas
matplotlib
botocore
boto3
numba
```

Pour supprimer un package Python du bloc-notes Zeppelin :

1. Vérifiez si votre application dépend du package, ou de l'un de ses packages consommateurs, avant de le supprimer. Vous pouvez identifier les dépendances d'un package à l'aide de pipdeptree.
2. Exécution de la commande suivante pour supprimer un package :

```
%flink.pyflink
!pip uninstall -y <package-to-remove>
```

3. Si vous devez récupérer un package que vous avez supprimé par erreur, exécutez la commande suivante :

```
%flink.pyflink
!pip install <package-to-install>
```

Example Exemple : supprimez le **`scipy`** package avant de déployer votre application Python avec deploy-as-app une fonctionnalité.

1. Utilisez `pipdeptree` pour découvrir tous les consommateurs `scipy` et vérifier si vous pouvez supprimer `scipy` en toute sécurité.
   - Installez l'outil via un bloc-notes :

```
%flink.pyflink
!pip install pipdeptree
```

   - Obtenez l'arbre de dépendance inversé de `scipy` en exécutant :

```
%flink.pyflink
```

```
!pip -r -p scipy
```

Vous devriez voir des résultats similaires à ce qui suit (condensés par souci de brièveté) :

```
...
------------------------------------------------------------------------
scipy==1.8.0
### plotnine==0.5.1 [requires: scipy>=1.0.0]
### seaborn==0.9.0 [requires: scipy>=0.14.0]
### statsmodels==0.12.2 [requires: scipy>=1.1]
    ### plotnine==0.5.1 [requires: statsmodels>=0.8.0]
```

2. Inspectez soigneusement l'utilisation de `seaborn`, `statsmodels` et `plotnine` dans vos applications. Si vos applications ne dépendent d'aucun des packages `scipy`, `seaborn`, `statemodels` ou `plotnine`, vous pouvez tous les supprimer ou uniquement ceux dont vos applications n'ont pas besoin.
3. Supprimez le package en exécutant :

```
!pip uninstall -y scipy plotnine seaborn statemodels
```

Annulation de tâches

Cette section explique comment annuler des tâches Apache Flink auxquelles vous ne pouvez pas accéder depuis Apache Zeppelin. Si vous souhaitez annuler une telle tâche, rendez-vous sur le tableau de bord Apache Flink, copiez l'ID de la tâche, puis utilisez-le dans l'un des exemples suivants.

Pour annuler une seule tâche :

```
%flink.pyflink
import requests

requests.patch("https://zeppelin-flink:8082/jobs/[job_id]", verify=False)
```

Pour annuler toutes les tâches en cours :

```
%flink.pyflink
import requests
```

```
r = requests.get("https://zeppelin-flink:8082/jobs", verify=False)
jobs = r.json()['jobs']

for job in jobs:
    if (job["status"] == "RUNNING"):
        print(requests.patch("https://zeppelin-flink:8082/jobs/{}".format(job["id"]),
 verify=False))
```

Pour annuler toutes les tâches :

```
%flink.pyflink
import requests

r = requests.get("https://zeppelin-flink:8082/jobs", verify=False)
jobs = r.json()['jobs']

for job in jobs:
    requests.patch("https://zeppelin-flink:8082/jobs/{}".format(job["id"]),
 verify=False)
```

Redémarrage de l'interprète Apache Flink

Pour redémarrer l'interprète Apache Flink dans votre bloc-notes Studio

1. Choisissez Configuration dans le coin supérieur droit de l'écran.
2. Choisissez Interprète.
3. Choisissez Redémarrer, puis OK.

Annexe : création de politiques IAM personnalisées

Vous utilisez normalement des politiques IAM gérées pour permettre à votre application d'accéder à des ressources dépendantes. Si vous avez besoin d'un contrôle plus précis des autorisations de votre application, vous pouvez utiliser une politique IAM personnalisée. Cette section contient des exemples de politiques IAM personnalisées.

> **Note**
>
> Dans les exemples de politique suivants, remplacez le texte de l'espace réservé par les valeurs de votre application.

Cette rubrique contient les sections suivantes :

- AWS Glue
- CloudWatch Journaux
- Flux Kinesis
- Clusters Amazon MSK

AWS Glue

L'exemple de politique suivant donne l'autorisations d'accéder à la base de données AWS Glue.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GlueTable",
            "Effect": "Allow",
            "Action": [
                "glue:GetConnection",
                "glue:GetTable",
                "glue:GetTables",
                "glue:GetDatabase",
                "glue:CreateTable",
                "glue:UpdateTable"
            ],
            "Resource": [
                "arn:aws:glue:<region>:<accountId>:connection/*",
                "arn:aws:glue:<region>:<accountId>:table/<database-name>/*",
                "arn:aws:glue:<region>:<accountId>:database/<database-name>",
                "arn:aws:glue:<region>:<accountId>:database/hive",
                "arn:aws:glue:<region>:<accountId>:catalog"
            ]
        },
        {
            "Sid": "GlueDatabase",
            "Effect": "Allow",
            "Action": "glue:GetDatabases",
            "Resource": "*"
        }
    ]
}
```

CloudWatch Journaux

La politique suivante accorde des autorisations d'accès aux CloudWatch journaux :

```
{
      "Sid": "ListCloudwatchLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:<region>:<accountId>:log-group:*"
      ]
    },
    {
      "Sid": "ListCloudwatchLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "<logGroupArn>:log-stream:*"
      ]
    },
    {
      "Sid": "PutCloudwatchLogs",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
        "<logStreamArn>"
      ]
    }
```

> **Note**
>
> Si vous créez votre application à l'aide de la console, celle-ci ajoute les politiques nécessaires pour accéder aux CloudWatch journaux à votre rôle d'application.

Flux Kinesis

Votre application peut utiliser un flux Kinesis pour une source ou une destination. Votre application a besoin d'autorisations de lecture pour lire à partir d'un flux source et d'autorisations d'écriture pour écrire dans un flux de destination.

La politique suivante accorde des autorisations de lecture à partir d'un flux Kinesis utilisé comme source :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "KinesisShardDiscovery",
      "Effect": "Allow",
      "Action": "kinesis:ListShards",
      "Resource": "*"
    },
    {
      "Sid": "KinesisShardConsumption",
      "Effect": "Allow",
      "Action": [
        "kinesis:GetShardIterator",
        "kinesis:GetRecords",
        "kinesis:DescribeStream",
        "kinesis:DescribeStreamSummary",
        "kinesis:RegisterStreamConsumer",
        "kinesis:DeregisterStreamConsumer"
      ],
      "Resource": "arn:aws:kinesis:<region>:<accountId>:stream/<stream-name>"
    },
    {
      "Sid": "KinesisEfoConsumer",
      "Effect": "Allow",
      "Action": [
        "kinesis:DescribeStreamConsumer",
        "kinesis:SubscribeToShard"
      ],
      "Resource": "arn:aws:kinesis:<region>:<account>:stream/<stream-name>/consumer/*"
    }
  ]
}
```

La politique suivante accorde des autorisations d'écriture dans un flux Kinesis utilisé comme destination :

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Sid": "KinesisStreamSink",
            "Effect": "Allow",
            "Action": [
                "kinesis:PutRecord",
                "kinesis:PutRecords",
                "kinesis:DescribeStreamSummary",
                "kinesis:DescribeStream"
            ],
            "Resource": "arn:aws:kinesis:<region>:<accountId>:stream/<stream-name>"
        }
    ]
}
```

Si votre application accède à un flux Kinesis crypté, vous devez accorder des autorisations supplémentaires pour accéder au flux et à la clé de chiffrement du flux.

La politique suivante accorde des autorisations pour accéder à un flux source chiffré et à la clé de chiffrement du flux :

```
{
      "Sid": "ReadEncryptedKinesisStreamSource",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": [
        "<inputStreamKeyArn>"
      ]
    }
    ,
```

La politique suivante accorde des autorisations pour accéder à un flux de destination chiffré et à la clé de chiffrement du flux :

```
{
```

```
      "Sid": "WriteEncryptedKinesisStreamSink",
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "<outputStreamKeyArn>"
      ]
    }
```

Clusters Amazon MSK

Pour accorder l'accès à un cluster Amazon MSK, vous accordez l'accès au VPC du cluster. Pour des exemples de politiques d'accès à un VPC Amazon, consultez la section Autorisations d'application VPC.

Commencer à utiliser Amazon Managed Service pour Apache Flink (DataStream API)

Cette section présente les concepts fondamentaux du service géré pour Apache Flink et de l' DataStream API. Elle décrit les options disponibles pour créer et tester vos applications. Elle fournit également des instructions pour installer les outils nécessaires pour suivre les didacticiels de ce guide et pour créer votre première application.

Rubriques

- Composants d’une application de service géré pour Apache Flink
- Prérequis pour effectuer les exercices
- Étape 1 : configuration d'un compte AWS et création d'un administrateur
- Étape 2 : configuration de AWS Command Line Interface (AWS CLI)
- Étape 3 : création et exécution d'une application de service géré pour Apache Flink.
- Etape 4 : nettoyage des ressources AWS
- Étape 5 : étapes suivantes

Composants d’une application de service géré pour Apache Flink

Pour traiter les données, votre application de service géré pour Apache Flink utilise une application Java/Apache Maven ou Scala qui traite les entrées et produit des sorties à l'aide de l’exécution Apache Flink.

Une application de service géré for Apache Flink comprend les composants suivants :

- Propriétés d'exécution : vous pouvez utiliser les propriétés d'exécution pour configurer votre application sans recompiler le code de votre application.
- Source : l'application consomme des données en utilisant une source. Un connecteur source lit les données d'un flux de données Kinesis, d'un compartiment Amazon S3, etc. Pour plus d’informations, consultez Sources.
- Opérateurs : l'application traite les données à l'aide d'un ou de plusieurs opérateurs. Un opérateur peut transformer, enrichir ou agréger des données. Pour plus d’informations, consultez Opérateurs d'API DataStream.

- Récepteur : l'application produit des données vers des sources externes à l'aide de récepteurs. Un connecteur récepteur écrit des données dans un flux de données Kinesis, un flux Kinesis Data Firehose, un compartiment Amazon S3, etc. Pour plus d'informations, consultez Récepteurs.

Après avoir créé, compilé et empaqueté votre code d'application, vous chargez le package de code dans un compartiment Amazon Simple Storage Service (Amazon S3). Vous créez ensuite une application de service géré pour Apache Flink. Vous transmettez l'emplacement du package de code, un flux de données Kinesis comme source de données de streaming et généralement un emplacement de streaming ou de fichier qui reçoit les données traitées par l'application.

Prérequis pour effectuer les exercices

Pour exécuter la procédure indiquée dans ce guide, vous devez disposer des éléments suivants :

- Kit de développement Java (JDK) version 11. Définissez la variable d'environnement `JAVA_HOME` pour qu'elle pointe vers l'emplacement d'installation de votre JDK.
- Nous vous recommandons d'utiliser un environnement de développement (par exemple Eclipse Java Neon ou IntelliJ Idea) pour développer et compiler votre application.
- Client Git. Installez le client Git si vous ne l'avez pas déjà fait.
- Apache Maven Compiler Plugin. Maven doit être installé dans votre chemin de travail. Pour tester votre installation Apache Maven, saisissez les informations suivantes :

```
$ mvn -version
```

Pour démarrer, accédez à Étape 1 : configuration d'un compte AWS et création d'un administrateur.

Étape 1 : configuration d'un compte AWS et création d'un administrateur

Avant d'utiliser le service géré pour Apache Flink pour la première fois, exécutez les tâches suivantes :

S'inscrire à un Compte AWS

Si vous n'avez pas de compte Compte AWS, procédez comme suit pour en créer un.

Pour s’inscrire à un Compte AWS

1. Ouvrez https://portal.aws.amazon.com/billing/signup.
2. Suivez les instructions en ligne.

   Dans le cadre de la procédure d’inscription, vous recevrez un appel téléphonique et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

   Lorsque vous souscrivez à un Compte AWS, un Utilisateur racine d'un compte AWS est créé. Par défaut, seul l’utilisateur root a accès à l’ensemble des Services AWS et des ressources de ce compte. La meilleure pratique de sécurité consiste à attribuer un accès administratif à un utilisateur administratif, et à uniquement utiliser l’utilisateur root pour effectuer tâches nécessitant un accès utilisateur root.

AWS vous envoie un e-mail de confirmation lorsque le processus d’inscription est terminé. Vous pouvez afficher l’activité en cours de votre compte et gérer votre compte à tout moment en accédant à https://aws.amazon.com/ et en cliquant sur Mon compte.

Création d’un utilisateur administratif

Après vous être inscrit à un Compte AWS, sécurisez votre Utilisateur racine d'un compte AWS, activez AWS IAM Identity Center, puis créez un utilisateur administratif afin de ne pas utiliser l’utilisateur root pour les tâches quotidiennes.

Sécurisation de votre Utilisateur racine d'un compte AWS

1. Connectez-vous à la AWS Management Console en tant que propriétaire du compte en sélectionnant Root user (utilisateur root) et en saisissant l’adresse e-mail de Compte AWS. Sur la page suivante, saisissez votre mot de passe.

   Pour obtenir de l’aide pour vous connecter en utilisant l’utilisateur root, consultez Connexion en tant qu’utilisateur root dans le Guide de l’utilisateur Connexion à AWS.
2. Activez l’authentification multifactorielle (MFA) pour votre utilisateur root.

   Pour obtenir des instructions, consultez Activation d’un dispositif MFA virtuel pour l’utilisateur root de votre Compte AWS (console) dans le Guide de l’utilisateur IAM.

Création d'un utilisateur administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Configuration d'AWS IAM Identity Center dans le guide de l'utilisateur AWS IAM Identity Center.
2. Dans IAM Identity Center, octroyez un accès administratif à un utilisateur administratif.

   Pour profiter d'un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, consultez Configuration de l'accès utilisateur avec le répertoire Répertoire IAM Identity Center par défaut dans le Guide de l'utilisateur AWS IAM Identity Center.

Connexion en tant qu'utilisateur administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l'URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l'utilisateur IAM Identity Center.

  Pour obtenir de l'aide pour vous connecter à l'aide d'un utilisateur IAM Identity Center, consultez Connexion au portail d'accès AWS dans le Guide de l'utilisateur Connexion à AWS.

Octroi d'un accès par programmation

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS en dehors de la AWS Management Console. La manière d'octroyer un accès par programmation dépend du type d'utilisateur qui accède à AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l'une des options suivantes.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
Identité de la main-d'œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour l'AWS CLI, veuillez consulter la rubrique Configuration de l'AWS

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
		CLI pour l'utilisation d'AWS IAM Identity Center dans le Guide de l'utilisateur AWS Command Line Interface. • Pour les kits AWS SDK, les outils et les API AWS, consultez Authentification IAM Identity Center dans le Guide de référence des kits SDK et des outils AWS.
IAM	Utilisez des informations d'identification temporaires pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec des ressources AWS dans le Guide de l'utilisateur IAM.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour l'AWS CLI, veuillez consulter la rubrique Authentification à l'aide des informations d'identification d'utilisateur IAM dans le Guide de l'utilisateur AWS Command Line Interface. • Pour les kits SDK et les outils AWS, veuillez consulter la rubrique Authentification à l'aide d'informations d'identification à long terme dans le Guide de référence des kits SDK et des outils AWS. • Pour les API AWS, veuillez consulter la rubrique Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Étape suivante

Étape 2 : configuration de AWS Command Line Interface (AWS CLI)

Étape 2 : configuration de AWS Command Line Interface (AWS CLI)

Dans cette étape, vous allez télécharger et configurer la AWS CLI pour l'utiliser avec le service géré pour Apache Flink.

> **Note**
>
> Les exercices de mise en route de ce guide supposent que vous utilisez les informations d'identification d'administrateur (`adminuser`) de votre compte pour effectuer les opérations.

> **Note**
>
> Si l'interface AWS CLI est déjà installée, vous pouvez avoir besoin de procéder à une mise à niveau pour obtenir les dernières fonctionnalités. Pour plus d'informations, consultez Installation d'AWS Command Line Interface dans le Guide de l'utilisateur AWS Command Line Interface. Pour vérifier la version de l'interface AWS CLI, exécutez la commande suivante :
>
> ```
> aws --version
> ```
>
> Les exercices de ce didacticiel nécessitent la version AWS CLI suivante ou une version ultérieure :
>
> ```
> aws-cli/1.16.63
> ```

Pour configurer l’interface AWS CLI

1. Téléchargez et configurez l’interface AWS CLI. Pour obtenir des instructions, consultez les rubriques suivantes dans le Guide de l'utilisateur de l'interface AWS Command Line Interface :
   - Installation de AWS Command Line Interface
   - Configuration de l'interface AWS CLI (français non garanti)
2. Ajoutez un profil désigné pour l'utilisateur administrateur dans le fichier `config` de l’interface AWS CLI. Vous utiliserez ce profil lorsque vous exécuterez les commandes AWS CLI. Pour plus

d'informations sur les profils nommés, consultez la rubrique Profils nommés dans le Guide de l'utilisateur AWS Command Line Interface.

```
[profile adminuser]
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
```

Pour connaître la liste des régions AWS disponibles, consultez Régions et points de terminaison du manuel Référence générale d'Amazon Web Services.

> **Note**
>
> Les exemples de code et de commandes présentés dans ce didacticiel utilisent la région USA Ouest (Oregon). Pour utiliser une autre région, remplacez la région dans le code et les commandes de ce didacticiel par la région que vous souhaitez utiliser.

3. Vérifiez la configuration en saisissant la commande d'aide suivante à l'invite de commande :

```
aws help
```

Après avoir configuré un AWS compteAWS CLI, vous pouvez passer à l'exercice suivant, dans lequel vous configurez un exemple d'application et testez la end-to-end configuration.

Étape suivante

Étape 3 : création et exécution d'une application de service géré pour Apache Flink.

Étape 3 : création et exécution d'une application de service géré pour Apache Flink.

Dans cet exercice, vous allez créer une application de service géré pour Apache Flink avec des flux de données comme source et comme récepteur.

Cette section contient les étapes suivantes :

- Création de deux flux de données Amazon Kinesis Data Streams
- Écriture d'exemples d'enregistrements dans le flux d'entrée

- Téléchargement et examen du code Java Apache Flink
- Compilation du code d'application
- Chargement du code Java Apache Flink
- Création et exécution de l’application de service géré pour Apache Flink
- Étape suivante

Création de deux flux de données Amazon Kinesis Data Streams

Avant de créer une application de service géré pour Apache Flink dans le cadre de cet exercice, commencez par créer deux flux de données Kinesis (`ExampleInputStream` et `ExampleOutputStream`). Votre application utilise ces flux pour les flux source et de destination de l'application.

Vous pouvez créer ces flux à l'aide de la console Amazon Kinesis ou de la commande AWS CLI suivante. Pour obtenir des instructions sur la console, consultez Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams.

Pour créer les flux de données (AWS CLI)

1. Pour créer le premier flux (`ExampleInputStream`) utilisez la commande AWS CLI `create-stream` Amazon Kinesis suivante.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Pour créer le second flux utilisé par l'application pour écrire la sortie, exécutez la même commande en remplaçant le nom du flux par `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Écriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette section, vous utilisez un script Python pour écrire les exemples d'enregistrements dans le flux pour que l'application les traite.

> **Note**
>
> Cette section nécessite le kit AWS SDK for Python (Boto).

1. Créez un fichier nommé `stock.py` avec le contenu suivant :

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")


if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Plus loin dans ce didacticiel, vous exécutez le script `stock.py` pour envoyer des données à l'application.

```
$ python stock.py
```

Téléchargement et examen du code Java Apache Flink

Le code de l'application Java pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/GettingStarted`.

Notez les informations suivantes à propos du code d'application :

- Un fichier de modèle d'objet du projet (pom.xml) contient des informations sur la configuration et les dépendances de l'application, y compris les bibliothèques du service géré pour Apache Flink.
- Le fichier `BasicStreamingJob.java` contient la méthode `main` qui définit la fonctionnalité de l'application.
- L'application utilise une source Kinesis pour lire à partir du flux source. L'extrait de code suivant crée la source Kinesis :

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- Votre application crée les connecteurs source et récepteur pour accéder aux ressources externes à l'aide d'un objet `StreamExecutionEnvironment`.
- L'application crée les connecteurs source et récepteur à l'aide de propriétés statiques. Pour utiliser les propriétés de l'application dynamique, utilisez les méthodes `createSourceFromApplicationProperties` et `createSinkFromApplicationProperties` pour créer les connecteurs. Ces méthodes lisent les propriétés de l'application pour configurer les connecteurs.

Pour de plus amples informations sur les propriétés d'exécution, veuillez consulter Propriétés d'exécution.

Compilation du code d'application

Dans cette section, vous allez utiliser le compilateur Apache Maven pour créer le code Java pour l'application. Pour de plus amples informations sur l'installation d'Apache Maven et sur le kit de développement Java (JDK), veuillez consulter Prérequis pour effectuer les exercices.

Pour compiler le code d'application

1. Pour utiliser votre code d'application, vous le compilez et l'intégrez dans un fichier JAR. Vous pouvez compiler et intégrer votre code de deux manières :

   - À l'aide de l'outil de ligne de commande Maven. Créez votre fichier JAR en exécutant la commande suivante dans le répertoire qui contient le fichier `pom.xml` :

     ```
     mvn package -Dflink.version=1.15.3
     ```

   - À l'aide de votre environnement de développement. Consultez la documentation de votre environnement de développement pour plus de détails.

     > **Note**
     >
     > Le code source fourni repose sur les bibliothèques de Java 11.

   Vous pouvez charger votre package en tant que fichier JAR, ou compresser le package et le charger en tant que fichier ZIP. Si vous créez votre application à l'aide de l'interface AWS CLI, vous spécifiez le type de contenu de votre code (JAR ou ZIP).

2. En cas d'erreur lors de la compilation, vérifiez que votre variable d'environnement `JAVA_HOME` est correctement définie.

Si la compilation de l'application aboutit, le fichier suivant est créé :

`target/aws-kinesis-analytics-java-apps-1.0.jar`

Chargement du code Java Apache Flink

Dans cette section, vous allez créer un compartiment Amazon Simple Storage Service (Amazon S3) et charger votre code d'application.

Pour charger le code d'application

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez Créer un compartiment.
3. Saisissez **`ka-app-code-<username>`** dans le champ Nom du compartiment. Ajoutez un suffixe au nom du compartiment, par exemple votre nom d'utilisateur, pour qu'il soit unique. Choisissez Suivant.
4. À l'étape Configurer les options, conservez les paramètres, puis choisissez Suivant.
5. À l'étape Définir des autorisations, conservez les paramètres, puis choisissez Suivant.
6. Choisissez Créer un compartiment.
7. Dans la console Amazon S3, choisissez le <username>compartiment ka-app-code-, puis Upload.
8. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `aws-kinesis-analytics-java-apps-1.0.jar` que vous avez créé à l'étape précédente. Choisissez Suivant.
9. Vous n'avez pas besoin de modifier les paramètres de l'objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l'application de service géré pour Apache Flink

Vous pouvez créer et exécuter une application de service géré pour Apache Flink à l'aide de la console ou de l'interface AWS CLI.

> **Note**
>
> Lorsque vous créez l'application à l'aide de la console, vos ressources AWS Identity and Access Management (IAM) et Amazon CloudWatch Logs sont créées pour vous. Lorsque vous créez l'application à l'aide de l'interface AWS CLI, vous créez ces ressources séparément.

Rubriques

- Création et exécution de l'application (console)
- Création et exécution de l'application (AWS CLI)

Création et exécution de l'application (console)

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Description, saisissez **`My java test app`**.
   - Pour Exécution, choisissez Apache Flink.
   - Laissez le menu déroulant de la version sur Apache Flink version 1.15.2 (version recommandée).
4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-MyApplication-us-west-2`

- Rôle : `kinesisanalytics-MyApplication-us-west-2`

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder aux flux de données Kinesis.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **`kinesis-analytics-service-MyApplication-us-west-2`** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*012345678901*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
```

```
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Configuration de l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:

- Pour le compartiment Amazon S3, saisissez **ka-app-code-*<username>***.
- Pour le chemin de l'objet Amazon S3, saisissez **aws-kinesis-analytics-java-apps-1.0.jar**.

3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Sous Propriétés, pour ID de groupe, saisissez **ProducerConfigProperties**.
5. Entrez les valeurs et propriétés d'application suivantes :

ID du groupe	Clé	Valeur
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

6. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
7. Pour la CloudWatch journalisation, cochez la case Activer.
8. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation Amazon, Managed Service for Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`

Exécution de l'application

Le graphique des tâches Flink peut être visualisé en exécutant l'application, en ouvrant le tableau de bord Apache Flink et en choisissant la tâche Flink souhaitée.

Arrêt de l'application

Sur la MyApplicationpage, choisissez Stop. Confirmez l'action.

Mise à jour de l'application

À l'aide de la console, vous pouvez mettre à jour les paramètres d'application tels que les paramètres de surveillance, les propriétés d'application et l'emplacement ou le nom du fichier JAR de l'application. Vous pouvez également recharger le fichier JAR de l'application à partir du compartiment Amazon S3 si vous avez besoin de mettre à jour le code de l'application.

Sur la MyApplicationpage, choisissez Configurer. Mettez à jour les paramètres de l'application, puis choisissez Mettre à jour.

Création et exécution de l'application (AWS CLI)

Dans cette section, vous allez utiliser l'interface AWS CLI pour créer et exécuter l'application de service géré pour Apache Flink. Le service géré pour Apache Flink utilise la commande AWS CLI `kinesisanalyticsv2` pour créer et interagir avec les applications de service géré pour Apache Flink.

Créer une stratégie d'autorisations

> **Note**
>
> Vous devez créer une stratégie d'autorisations et un rôle pour votre application. Si vous ne créez pas ces ressources IAM, votre application ne peut pas accéder à ses flux de données et de journaux.

Vous commencez par créer une stratégie d'autorisations avec deux instructions : une qui accorde des autorisations pour l'action `read` sur le flux source et une autre qui accorde des autorisations pour les actions `write` sur le flux récepteur. Vous attachez ensuite la politique à un rôle IAM (que vous allez créer dans la section suivante). Ainsi, lorsque le service géré pour Apache Flink assume le rôle, le

service dispose des autorisations nécessaires pour lire à partir du flux source et écrire dans le flux récepteur.

Utilisez le code suivant pour créer la stratégie d'autorisations `AKReadSourceStreamWriteSinkStream`. Remplacez *`username`* par le nom d'utilisateur que vous avez utilisé pour créer le compartiment Amazon S3 pour stocker le code d'application. Remplacez l'ID de compte dans Amazon Resource Names (ARN) (*`012345678901`*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": ["arn:aws:s3:::ka-app-code-username",
                "arn:aws:s3:::ka-app-code-username/*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Pour step-by-step obtenir des instructions sur la création d'une politique d'autorisations, voir Tutoriel : créer et joindre votre première politique gérée par le client dans le guide de l'utilisateur IAM.

> **Note**
>
> Pour accéder à d'autres services Amazon, vous pouvez utiliser le AWS SDK for Java. Le service géré pour Apache Flink définit automatiquement les informations d'identification requises par le kit SDK en fonction du rôle IAM d'exécution du service associé à votre application. Aucune étape supplémentaire n'est nécessaire.

Créer un rôle IAM

Dans cette section, vous créez un rôle IAM que l'application de service géré pour Apache Flink peut assumer pour lire un flux source et écrire dans le flux récepteur.

Le service géré pour Apache Flink ne peut pas accéder à votre flux sans autorisation. Vous utilisez un rôle IAM pour accorder ces autorisations. Deux politiques sont attachées à chaque rôle IAM. La politique d'approbation accorde au service géré pour Apache Flink l'autorisation d'assumer le rôle, et la politique d'autorisation détermine ce que le service géré pour Apache Flink peut faire après avoir assumé le rôle.

Vous attachez la stratégie d'autorisations que vous avez créée dans la section précédente à ce rôle.

Pour créer un rôle IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, choisissez Rôles, puis Créer un rôle.
3. Sous Sélectionner le type d'identité approuvée, choisissez Service AWS. Sous Choisir le service qui utilisera ce rôle, choisissez EC2. Sous Sélectionner votre cas d'utilisation, choisissez Kinesis Analytics.

   Sélectionnez Next: Permissions (Étape suivante : autorisations).
4. Dans la page Attacher des stratégies d'autorisations, choisissez Suivant : vérification. Vous attachez des stratégies d'autorisations après avoir créé le rôle.
5. Sur la page Créer un rôle, saisissez **`MF-stream-rw-role`** pour le Nom du rôle. Sélectionnez Créer un rôle.

   Vous venez de créer un nouveau rôle IAM appelé `MF-stream-rw-role`. Ensuite, vous mettez à jour les stratégies d'approbation et d'autorisation pour le rôle.
6. Attachez la politique d'autorisation au rôle.

> **Note**
>
> Dans le cadre de cet exercice, le service géré pour Apache Flink assume ce rôle à la fois pour la lecture des données à partir d'un flux de données Kinesis (source) et pour l'écriture des résultats dans un autre flux de données Kinesis. Vous attachez donc la stratégie que vous avez créée à l'étape précédente, the section called “Créer une stratégie d'autorisations”.

a. Sur la page Récapitulatif, choisissez l'onglet Autorisations.

b. Choisissez Attacher des stratégies.

c. Dans la zone de recherche, saisissez **`AKReadSourceStreamWriteSinkStream`** (la stratégie que vous avez créée dans la section précédente).

d. Choisissez la ReadSourceStreamWriteSinkStream politique AK, puis choisissez Attach policy.

Vous avez maintenant créé le rôle d'exécution de service que votre application utilise pour accéder aux ressources. Notez l'ARN du nouveau rôle.

Pour step-by-step obtenir des instructions sur la création d'un rôle, consultez la section Création d'un rôle IAM (console) dans le guide de l'utilisateur IAM.

Création de l’application de service géré pour Apache Flink

1. Copiez le code JSON suivant dans un fichier nommé `create_request.json`. Remplacez l'exemple d'ARN du rôle par l'ARN du rôle que vous avez créé précédemment. Remplacez le suffixe de l'ARN du compartiment (*`username`*) par le suffixe que vous avez choisi dans la section précédente. Remplacez l'exemple d'ID de compte (*`012345678901`*) dans le rôle d'exécution de service par votre ID de compte.

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
```

```
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "aws-kinesis-analytics-java-apps-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
    }
}
```

2. Exécutez l'action [CreateApplication](#) avec la demande précédente pour créer l'application :

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

L'application est maintenant créée. Vous démarrez l'application dans l'étape suivante.

Démarrage de l'application

Dans cette section, vous utilisez l'action [StartApplication](#) pour démarrer l'application.

Pour démarrer l'application

1. Copiez le code JSON suivant dans un fichier nommé `start_request.json`.

```
{
    "ApplicationName": "test",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. Exécutez l'action `StartApplication` avec la demande précédente pour démarrer l'application :

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

L'application est maintenant en cours d'exécution. Vous pouvez consulter les métriques du service géré pour Apache Flink sur la CloudWatch console Amazon pour vérifier que l'application fonctionne.

Arrêt de l'application

Dans cette section, vous allez utiliser l'action `StopApplication` pour arrêter l'application.

Pour arrêter l'application

1. Copiez le code JSON suivant dans un fichier nommé `stop_request.json`.

```
{
   "ApplicationName": "test"
}
```

2. Exécutez l'action `StopApplication` avec la demande suivante pour arrêter l'application :

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

L'application est maintenant arrêtée.

Ajouter une option de CloudWatch journalisation

Vous pouvez utiliser le AWS CLI pour ajouter un flux de CloudWatch journal Amazon à votre application. Pour plus d'informations sur l'utilisation de CloudWatch Logs avec votre application, consultezthe section called “Configuration de la journalisation”.

Mettre à jour des propriétés d'environnement

Dans cette section, vous utilisez l'action `UpdateApplication` pour modifier les propriétés d'environnement de l'application sans recompiler le code de l'application. Dans cet exemple, vous modifiez la région des flux source et de destination.

Pour mettre à jour des propriétés d'environnement pour l'application

1. Copiez le code JSON suivant dans un fichier nommé `update_properties_request.json`.

```
{"ApplicationName": "test",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
   }
}
```

2. Exécutez l'action `UpdateApplication` avec la demande précédente pour mettre à jour les propriétés de l'environnement :

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Mise à jour du code de l'application

Lorsque vous devez mettre à jour le code de votre application avec une nouvelle version de votre package de code, vous utilisez l'action`UpdateApplication` de l'interface AWS CLI.

> **Note**
>
> Pour charger une nouvelle version du code de l'application portant le même nom de fichier, vous devez spécifier la nouvelle version de l'objet. Pour de plus amples informations sur l'utilisation des versions d'objet Amazon S3, veuillez consulter Activation et désactivation de la gestion des versions.

Pour utiliser l'interface AWS CLI, supprimez votre ancien package de code de votre compartiment Amazon S3, téléchargez la nouvelle version et appelez `UpdateApplication` en spécifiant le même compartiment Amazon S3 et le même nom d'objet, ainsi que la nouvelle version de l'objet. L'application redémarrera avec le nouveau package de code.

L'exemple de demande d'action `UpdateApplication` suivant recharge le code de l'application et redémarre l'application. Mettez à jour l'`CurrentApplicationVersionId` à la version actuelle de l'application. Vous pouvez vérifier la version actuelle de l'application à l'aide des actions `ListApplications` ou `DescribeApplication`. Mettez à jour le suffixe du nom du compartiment (<username>) avec le suffixe que vous avez choisi dans la section the section called "Création de deux flux de données Amazon Kinesis Data Streams".

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "aws-kinesis-analytics-java-apps-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
```

```
                }
            }
        }
    }
}
```

Étape suivante

Etape 4 : nettoyage des ressources AWS

Etape 4 : nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWSCréées dans le didacticiel Mise en route.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer vos flux de données Kinesis
- Supprimer votre objet et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources
- Étape suivante

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Managed Service for Apache Flink, choisissez MyApplication.
3. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer vos flux de données Kinesis

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le panneau Kinesis Data Streams, ExampleInputStreamsélectionnez.

3. Sur la ExampleInputStreampage, choisissez Supprimer Kinesis Stream, puis confirmez la suppression.
4. Sur la page Kinesis Streams, choisissez le ExampleOutputStream, choisissez Actions, choisissez Supprimer, puis confirmez la suppression.

Supprimer votre objet et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code -. <username>
3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.
8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Étape suivante

Étape 5 : étapes suivantes

Étape 5 : étapes suivantes

Maintenant que vous avez créé et exécuté une application de service géré de base pour Apache Flink, consultez les ressources suivantes pour des solutions de service géré plus avancées pour Apache Flink.

- Solution de streaming de données AWS pour Amazon Kinesis : la solution de streaming de données AWS pour Amazon Kinesis configure automatiquement les services AWS nécessaires pour capturer, stocker, traiter et diffuser facilement des données de streaming. La solution propose plusieurs options pour résoudre les problèmes d'utilisation de données en streaming. L'option Managed Service for Apache Flink fournit un exemple de end-to-end streaming ETL illustrant une application réelle qui exécute des opérations analytiques sur des données de taxis simulées à New York. La solution met en place toutes les AWS ressources nécessaires, telles que les rôles et les politiques IAM, un CloudWatch tableau de bord et des CloudWatch alarmes.
- Solution de données de streaming AWS pour Amazon MSK :la solution de données de streaming AWS pour Amazon MSK fournit des modèles AWS CloudFormation dans lesquels les données circulent entre les producteurs, le stockage en streaming, les consommateurs et les destinations.
- Clickstream Lab avec Apache Flink et Apache Kafka : un laboratoire de bout en bout pour les cas d'utilisation d’Amazon Managed Streaming for Apache Kafka pour le stockage de streaming et le service géré pour Apache Flink pour les applications Apache Flink pour le traitement des flux.
- Amazon Managed Service for Apache Flink Workshop : dans cet atelier, vous allez créer une architecture de end-to-end streaming pour ingérer, analyser et visualiser les données de streaming en temps quasi réel. Vous avez décidé d'améliorer les opérations d'une compagnie de taxi à New York. Vous analysez les données de télémétrie d'une flotte de taxis à New York en temps quasi réel afin d'optimiser le fonctionnement de la flotte.
- Service géré Amazon pour Apache Flink : exemples : cette section de ce guide du développeur fournit des exemples de création et d'utilisation d'applications dans le service géré pour Apache Flink. Ils incluent des exemples de code et des step-by-step instructions pour vous aider à créer un service géré pour les applications Apache Flink et à tester vos résultats.
- Learn Flink : Hands On Training : formation d'introduction officielle à Apache Flink qui vous permet de commencer à écrire des applications ETL, analytiques et axées sur les événements évolutives pour le streaming.

> **Note**
>
> Sachez que le service géré pour Apache Flink ne prend pas en charge la version Apache Flink (1.12) utilisée dans cette formation. Vous pouvez utiliser Flink 1.15.2 dans le service géré Flink pour Apache Flink.

Mise en route avec le service géré Amazon pour Apache Flink (API de table)

Cette section présente les concepts fondamentaux du service géré pour Apache Flink et de l'API de table. Elle décrit les options disponibles pour créer et tester vos applications. Elle fournit également des instructions pour installer les outils nécessaires pour suivre les didacticiels de ce guide et pour créer votre première application.

Rubriques

- Composants d’une application de service géré pour Apache Flink
- Prérequis
- Création et exécution d'une application de service géré pour Apache Flink.
- Nettoyage des ressources AWS
- Étapes suivantes

Composants d’une application de service géré pour Apache Flink

Pour traiter les données, votre application de service géré pour Apache Flink utilise une application Java/Apache Maven ou Scala qui traite les entrées et produit des sorties à l'aide de l’exécution Apache Flink.

L’application de service géré pour Apache Flink comprend les composants suivants :

- Propriétés d'exécution : vous pouvez utiliser les propriétés d'exécution pour configurer votre application sans recompiler le code de votre application.
- Source de la table : l'application consomme des données en utilisant une source. Un connecteur source lit les données d'un flux de données Kinesis, d'une rubrique Amazon MSK, etc. Pour plus d’informations, consultez Sources de l’API de table.
- Fonctions : l'application traite les données à l'aide d'une ou de plusieurs fonctions. Une fonction peut transformer, enrichir ou agréger des données.
- Récepteur : l'application produit des données vers des sources externes à l'aide de récepteurs. Un connecteur récepteur écrit des données dans un flux de données Kinesis, un flux Kinesis Data Firehose, une rubrique Amazon MSK, un compartiment Amazon S3, etc. Pour plus d’informations, consultez Récepteurs de l'API de table.

Après avoir créé, compilé et empaqueté votre code d'application, vous chargez le package de code dans un compartiment Amazon S3. Vous créez ensuite une application de service géré pour Apache Flink. Vous transmettez l'emplacement du package de code, une rubrique Amazon MSK comme flux de données de streaming et généralement un emplacement de streaming ou de fichier qui reçoit les données traitées par l'application.

Prérequis

Avant de commencer ce didacticiel, suivez les deux premières étapes de Commencer à utiliser Amazon Managed Service pour Apache Flink (DataStream API) :

- Étape 1 : configuration d'un compte AWS et création d'un administrateur
- Étape 2 : configuration de AWS Command Line Interface (AWS CLI)

Consultez Création d'une application pour démarrer.

Création et exécution d'une application de service géré pour Apache Flink.

Dans cet exercice, vous allez créer une application de service géré pour Apache Flink avec une rubrique Amazon MSK comme source et un compartiment Amazon S3 comme récepteur.

Cette section contient les étapes suivantes.

- Création de ressources dépendantes
- Écriture d'exemples d'enregistrements dans le flux d'entrée
- Téléchargement et examen du code Java Apache Flink
- Compilation du code d'application
- Chargement du code Java Apache Flink
- Création et exécution de l’application de service géré pour Apache Flink
- Étape suivante

Création de ressources dépendantes

Avant de créer un service géré pour Apache Flink dans le cadre de cet exercice, vous commencez par créer les ressources dépendantes suivantes :

- Un cloud privé virtuel (VPC) basé sur Amazon VPC et un cluster Amazon MSK
- Un compartiment Amazon S3 pour stocker le code et la sortie de l'application (`ka-app-code-<username>`)

Création d'un VPC et d'un cluster Amazon MSK

Pour créer un VPC et un cluster Amazon MSK auxquels vous pourrez accéder depuis votre application de service géré pour Apache Flink, suivez le didacticiel Mise en route avec Amazon MSK.

Lorsque vous avez terminé le didacticiel, notez ce qui suit :

- Enregistrez la liste des serveurs bootstrap de votre cluster. Vous pouvez obtenir la liste des serveurs bootstrap à l'aide de la commande suivante, en remplaçant *ClusterArn* par l'Amazon Resource Name (ARN) de votre cluster MSK :

```
aws kafka get-bootstrap-brokers --region us-west-2 --cluster-arn ClusterArn
{...
    "BootstrapBrokerStringTls": "b-2.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-1.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-3.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094"
}
```

- Lorsque vous suivez les étapes décrites dans les didacticiels, veillez à utiliser la région AWS sélectionnée dans le code, les commandes et les entrées de console.

Créer un compartiment Amazon S3

Vous pouvez créer un compartiment Amazon S3 à l'aide de la console. Pour obtenir les instructions relatives à la création de cette ressource, consultez les rubriques suivantes :

- Comment créer un compartiment S3 ? dans le Guide de l'utilisateur de la console Amazon Simple Storage Service. Donnez au compartiment Amazon S3 un nom unique au monde en ajoutant votre nom de connexion, tel que **ka-app-code-*<username>***.

Autres ressources

Lorsque vous créez votre application, Managed Service for Apache Flink crée les CloudWatch ressources Amazon suivantes si elles n'existent pas déjà :

- Un groupe de journaux appelé `/AWS/KinesisAnalytics-java/MyApplication`.
- Un flux de journaux appelé `kinesis-analytics-log-stream`

Écriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette section, vous utilisez un script Python pour écrire les exemples d'enregistrements dans la rubrique Amazon MSK pour que l'application les traite.

1. Connectez-vous à l'instance client que vous avez créée à l'étape 4 : créer un ordinateur client du didacticiel Mise en route avec Amazon MSK.
2. Installez Python3, Pip et la bibliothèque Kafka Python :

```
$ sudo yum install python37
$ curl -O https://bootstrap.pypa.io/get-pip.py
$ python3 get-pip.py --user
$ pip install kafka-python
```

3. Créez un fichier nommé `stock.py` avec les contenus suivants. Remplacez la valeur `BROKERS` par votre liste d'agents bootstrap que vous avez enregistrée précédemment.

```
import datetime
    import json
    import random
    import boto3

    STREAM_NAME = "ExampleInputStream"


    def get_data():
        return {
            'event_time': datetime.datetime.now().isoformat(),
            'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
            'price': round(random.random() * 100, 2)}


    def generate(stream_name, kinesis_client):
        while True:
            data = get_data()
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
```

```
                Data=json.dumps(data),
                PartitionKey="partitionkey")


    if __name__ == '__main__':
        generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

4. Plus loin dans ce didacticiel, vous exécutez le script `stock.py` pour envoyer des données à l'application.

```
$ python3 stock.py
```

Téléchargement et examen du code Java Apache Flink

Le code de l'application Java pour cet exemple est disponible sur GitHub.

Pour télécharger le code d'application Java

1. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/GettingStartedTable`.

Notez les informations suivantes à propos du code d'application :

- Un fichier de modèle d'objet du projet (pom.xml) contient des informations sur la configuration et les dépendances de l'application, y compris les bibliothèques du service géré pour Apache Flink.
- Le fichier `StreamingJob.java` contient la méthode `main` qui définit la fonctionnalité de l'application.
- L'application utilise un `FlinkKafkaConsumer` pour lire la rubrique Amazon MSK. L'extrait de code suivant crée un objet `FlinkKafkaConsumer` :

```
final FlinkKafkaConsumer<StockRecord> consumer = new
 FlinkKafkaConsumer<StockRecord>(kafkaTopic, new KafkaEventDeserializationSchema(),
 kafkaProps);
```

- Votre application crée les connecteurs source et récepteur pour accéder aux ressources externes à l'aide des objets `StreamExecutionEnvironment` et `TableEnvironment`.
- L'application crée des connecteurs source et récepteur à l'aide des propriétés dynamiques de l'application. Vous pouvez ainsi spécifier les paramètres de votre application (tels que votre compartiment S3) sans recompiler le code.

```
//read the parameters from the Managed Service for Apache Flink environment
Map<String, Properties> applicationProperties =
 KinesisAnalyticsRuntime.getApplicationProperties();
Properties flinkProperties = null;

String kafkaTopic = parameter.get("kafka-topic", "AWSKafkaTutorialTopic");
String brokers = parameter.get("brokers", "");
String s3Path = parameter.get("s3Path", "");

if (applicationProperties != null) {
    flinkProperties = applicationProperties.get("FlinkApplicationProperties");
}

if (flinkProperties != null) {
    kafkaTopic = flinkProperties.get("kafka-topic").toString();
    brokers = flinkProperties.get("brokers").toString();
    s3Path = flinkProperties.get("s3Path").toString();
}
```

Pour de plus amples informations sur les propriétés d'exécution, veuillez consulter Propriétés d'exécution.

> **Note**
>
> Lorsque vous créez votre application, nous vous conseillons vivement de créer et d'exécuter l'application de service gérée pour Apache Flink dans la même région que le cluster Amazon MSK. Cela est dû au fait que le connecteur Kafka Flink est optimisé par défaut pour un environnement à faible latence. Si vous devez consommer à partir d'un cluster Kafka interrégional, pensez à augmenter la valeur de configuration pour `receive.buffer.byte`, par exemple 2097152.
> Pour de plus amples informations, veuillez consulter Configurations MSK personnalisées.

Compilation du code d'application

Dans cette section, vous allez utiliser le compilateur Apache Maven pour créer le code Java pour l'application. Pour de plus amples informations sur l'installation d'Apache Maven et sur le kit de développement Java (JDK), veuillez consulter Prérequis pour effectuer les exercices.

Pour compiler le code d'application

1. Pour utiliser votre code d'application, vous le compilez et l'intégrez dans un fichier JAR. Vous pouvez compiler et intégrer votre code de deux manières :

   - À l'aide de l'outil de ligne de commande Maven. Créez votre fichier JAR en exécutant la commande suivante dans le répertoire qui contient le fichier `pom.xml` :

     ```
     mvn package -Dflink.version=1.15.3
     ```

   - À l'aide de votre environnement de développement. Consultez la documentation de votre environnement de développement pour plus de détails.

     > **Note**
     >
     > Le code source fourni repose sur les bibliothèques de Java 11.

   Vous pouvez charger votre package en tant que fichier JAR, ou compresser le package et le charger en tant que fichier ZIP. Si vous créez votre application à l'aide de l'interface AWS CLI, vous spécifiez le type de contenu de votre code (JAR ou ZIP).

2. En cas d'erreur lors de la compilation, vérifiez que votre variable d'environnement `JAVA_HOME` est correctement définie.

Si la compilation de l'application aboutit, le fichier suivant est créé :

`target/aws-kinesis-analytics-java-apps-1.0.jar`

Chargement du code Java Apache Flink

Dans cette section, vous allez créer un compartiment Amazon S3 et charger votre code d'application.

Pour charger le code d'application

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez Créer un compartiment.
3. Saisissez **ka-app-code-*<username>*** dans le champ Nom du compartiment. Ajoutez un suffixe au nom du compartiment, par exemple votre nom d'utilisateur, pour qu'il soit unique. Choisissez Suivant.
4. À l'étape Configurer les options, conservez les paramètres, puis choisissez Suivant.
5. À l'étape Définir des autorisations, conservez les paramètres, puis choisissez Suivant.
6. Choisissez Créer un compartiment.
7. Dans la console Amazon S3, choisissez le <username>compartiment ka-app-code-, puis Upload.
8. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `aws-kinesis-analytics-java-apps-1.0.jar` que vous avez créé à l'étape précédente. Choisissez Suivant.
9. Vous n'avez pas besoin de modifier les paramètres de l'objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l'application de service géré pour Apache Flink

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Description, saisissez **`My java test app`**.
   - Pour Exécution, choisissez Apache Flink.

- Laissez la version sur Apache Flink 1.15.2 (version recommandée).

4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **kinesis-analytics-MyApplication-us-west-2**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-MyApplication-us-west-2`
> - Rôle : `kinesisanalytics-MyApplication-us-west-2`

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder au compartiment Amazon S3.

Pour modifier la politique IAM afin d'ajouter des autorisations au compartiment S3

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **kinesis-analytics-service-MyApplication-us-west-2** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Sid": "S3",
        "Effect": "Allow",
        "Action": [
            "s3:Abort*",
            "s3:DeleteObject*",
```

```
            "s3:GetObject*",
            "s3:GetBucket*",
            "s3:List*",
            "s3:ListBucket",
            "s3:PutObject"
        ],
        "Resource": [
            "arn:aws:s3:::ka-app-code-<username>",
            "arn:aws:s3:::ka-app-code-<username>/*"
        ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        }
    ]
```

```
}
```

Configuration de l'application

Procédez comme suit pour configurer l'application.

Pour configurer l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **ka-app-code-*<username>***.
   - Pour le chemin de l'objet Amazon S3, saisissez **aws-kinesis-analytics-java-apps-1.0.jar**.
3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Sous Propriétés, sélectionnez Créer un groupe.
5. Saisissez :

ID du groupe	Clé	Valeur
FlinkApplicationProperties	**kafka-topic**	**AWSKafkaTutorialTopic**
FlinkApplicationProperties	**brokers**	***Your Amazon MSK cluster's Bootstrap Brokers list***
FlinkApplicationProperties	**s3Path**	**ka-app-code- *<username>***
FlinkApplicationProperties	**security.protocol**	**SSL**
FlinkApplicationProperties	**ssl.truststore.location**	**/usr/lib/jvm/java-11-amazon-corretto**

ID du groupe	Clé	Valeur
		/lib/security/cacerts
FlinkApplicationProperties	**ssl.truststore.password**	**changeit**

6. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
7. Pour la CloudWatch journalisation, cochez la case Activer.
8. Dans la section cloud privé virtuel (VPC), choisissez Configuration VPC basée sur le cluster Amazon MSK. Choisissez AWSKafkaTutorialCluster.
9. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation Amazon, Managed Service for Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`

Exécution de l'application

Procédez comme suit pour exécuter l'application.

Pour exécuter l'application

1. Sur la MyApplicationpage, choisissez Exécuter. Confirmez l'action.
2. Lorsque l'application est en cours d'exécution, actualisez la page. La console affiche le graphique de l'application.
3. Depuis votre client Amazon EC2, exécutez le script Python que vous avez créé précédemment pour écrire des enregistrements dans le cluster Amazon MSK afin que votre application puisse les traiter :

```
$ python3 stock.py
```

Arrêt de l'application

Pour arrêter l'application, sur la MyApplicationpage, choisissez Arrêter. Confirmez l'action.

Étape suivante

Nettoyage des ressources AWS

Nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Mise en route (API de table).

Cette rubrique contient les sections suivantes.

- Suppression de votre application de service géré pour Apache Flink
- Supprimer votre cluster Amazon MSK
- Supprimer votre VPC
- Supprimer vos objets et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources
- Étape suivante

Suppression de votre application de service géré pour Apache Flink

Procédez comme suit pour supprimer l'application.

Pour supprimer l’application

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Managed Service for Apache Flink, choisissez MyApplication.
3. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer votre cluster Amazon MSK

Pour supprimer votre cluster Amazon MSK, suivez Étape 8 : supprimer le cluster Amazon MSK dans le guide du développeur Amazon Managed Streaming for Apache Kafka.

Supprimer votre VPC

Pour supprimer votre Amazon VPC, procédez comme suit :

- Ouvrez la console VPC Amazon.
- Choisissez Vos VPC.
- Pour Actions, choisissez Supprimer le VPC.

Supprimer vos objets et votre compartiment Amazon S3

Procédez comme suit pour supprimer vos objets et votre compartiment S3.

Pour supprimer vos objets et votre compartiment S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le <username>compartiment ka-app-code-.
3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

Utilisez la procédure suivante pour supprimer vos ressources IAM.

Pour supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.

8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos CloudWatch ressources

Pour supprimer vos CloudWatch ressources, procédez comme suit.

Pour supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Étape suivante

Étapes suivantes

Étapes suivantes

Maintenant que vous avez créé et exécuté une application de service géré pour Apache Flink qui utilise l'API de table, consultez Étape 5 : étapes suivantes dans Commencer à utiliser Amazon Managed Service pour Apache Flink (DataStream API).

Mise en route avec le service géré Amazon pour Apache Flink pour Python

Cette section présente les concepts fondamentaux d'un service géré pour Apache Flink à l'aide de Python et de l'API de table. Elle décrit les options disponibles pour créer et tester vos applications. Elle fournit également des instructions pour installer les outils nécessaires pour suivre les didacticiels de ce guide et pour créer votre première application.

Rubriques

- Mise en route avec Pyflink - L'interpréteur Python pour Apache | Amazon Web Services
- Composants d’une application de service géré pour Apache Flink
- Prérequis
- Création et exécution d'un service géré pour Apache Flink dédié à l’application Python
- Nettoyage des ressources AWS

> **Note**
>
> Si vous développez l'application Python Flink sur un nouveau Mac équipé d'une puce Apple Silicon, vous pouvez rencontrer des problèmes connus liés aux dépendances Python de la version PyFlink 1.15. Dans ce cas, nous recommandons d'exécuter l'interpréteur Python dans Docker. Pour step-by-step obtenir des instructions, reportez-vous à la section Développement de la version PyFlink 1.15 sur Apple Silicon Mac.

Mise en route avec Pyflink - L'interpréteur Python pour Apache | Amazon Web Services

Avant de commencer, nous vous invitons à regarder la vidéo suivante :

Mise en route avec Pyflink - L'interpréteur Python pour Apache | Amazon Web Services

Composants d’une application de service géré pour Apache Flink

Pour traiter les données, votre application de service géré pour Apache Flink utilise une application Python qui traite les entrées et produit des sorties à l'aide de l’exécution Apache Flink.

L’application de service géré pour Apache Flink comprend les composants suivants :

- Propriétés d'exécution : vous pouvez utiliser les propriétés d'exécution pour configurer votre application sans recompiler le code de votre application.
- Source de la table : l'application consomme des données en utilisant une source. Un connecteur source lit les données d'un flux de données Kinesis, d'une rubrique Amazon MSK, etc. Pour plus d’informations, consultez Sources de l’API de table.
- Fonctions : l'application traite les données à l'aide d'une ou de plusieurs fonctions. Une fonction peut transformer, enrichir ou agréger des données.
- Récepteur : l'application produit des données vers des sources externes à l'aide de récepteurs. Un connecteur récepteur écrit des données dans un flux de données Kinesis, un flux Kinesis Data Firehose, une rubrique Amazon MSK, un compartiment Amazon S3, etc. Pour plus d’informations, consultez Récepteurs de l'API de table.

Après avoir créé et empaqueté votre code d'application, vous chargez le package de code dans un compartiment Amazon S3. Vous créez ensuite une application de service géré pour Apache Flink. Vous transmettez l'emplacement du package de code, un flux de données de streaming et généralement un emplacement de streaming ou de fichier qui reçoit les données traitées par l'application.

Prérequis

Avant de commencer ce didacticiel, suivez les deux premières étapes de Commencer à utiliser Amazon Managed Service pour Apache Flink (DataStream API) :

- Étape 1 : configuration d'un compte AWS et création d'un administrateur
- Étape 2 : configuration de AWS Command Line Interface (AWS CLI)

Consultez Création d'une application pour démarrer.

Création et exécution d'un service géré pour Apache Flink dédié à l’application Python

Dans cet exercice, vous allez créer une application de service géré pour Apache Flink dédiée à l’application Python avec un flux Kinesis comme source et comme récepteur.

Cette section contient les étapes suivantes.

- Création de ressources dépendantes
- Écriture d'exemples d'enregistrements dans le flux d'entrée
- Création et examen du code Python Apache Flink
- Ajouter des dépendances tierces aux applications Python
- Chargement du code Python Apache Flink
- Création et exécution de l’application de service géré pour Apache Flink
- Étape suivante

Création de ressources dépendantes

Avant de créer un service géré pour Apache Flink dans le cadre de cet exercice, vous commencez par créer les ressources dépendantes suivantes :

- Deux flux Kinesis pour l'entrée et la sortie.
- Un compartiment Amazon S3 pour stocker le code et la sortie de l'application (`ka-app-code-<username>`)

Création de deux flux Kinesis

Avant de créer une application de service géré pour Apache Flink dans le cadre de cet exercice, commencez par créer deux flux de données Kinesis (`ExampleInputStream` et `ExampleOutputStream`). Votre application utilise ces flux pour les flux source et de destination de l'application.

Vous pouvez créer ces flux à l'aide de la console Amazon Kinesis ou de la commande AWS CLI suivante. Pour obtenir des instructions sur la console, consultez Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams.

Pour créer les flux de données (AWS CLI)

1. Pour créer le premier flux (`ExampleInputStream`) utilisez la commande AWS CLI `create-stream` Amazon Kinesis suivante.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Pour créer le second flux utilisé par l'application pour écrire la sortie, exécutez la même commande en remplaçant le nom du flux par `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Créer un compartiment Amazon S3

Vous pouvez créer un compartiment Amazon S3 à l'aide de la console. Pour obtenir les instructions relatives à la création de cette ressource, consultez les rubriques suivantes :

- Comment créer un compartiment S3 ? dans le Guide de l'utilisateur de la console Amazon Simple Storage Service. Donnez au compartiment Amazon S3 un nom unique au monde en ajoutant votre nom de connexion, tel que **`ka-app-code-`*`<username>`***.

Autres ressources

Lorsque vous créez votre application, Managed Service for Apache Flink crée les CloudWatch ressources Amazon suivantes si elles n'existent pas déjà :

- Un groupe de journaux appelé `/AWS/KinesisAnalytics-java/MyApplication`.
- Un flux de journaux appelé `kinesis-analytics-log-stream`

Écriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette section, vous utilisez un script Python pour écrire les exemples d'enregistrements dans le flux pour que l'application les traite.

> **Note**
>
> Cette section nécessite le kit AWS SDK for Python (Boto).

> **Note**
>
> Le script Python de cette section utilise l'interface AWS CLI. Vous devez configurer votre interface AWS CLI pour utiliser les informations d'identification de votre compte et la région par défaut. Pour configurer votre interface AWS CLI, saisissez :
>
> ```
> aws configure
> ```

1. Créez un fichier nommé `stock.py` avec le contenu suivant :

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
```

```
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey")


    if __name__ == '__main__':
        generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Exécutez le script `stock.py` :

```
$ python stock.py
```

   Maintenez le script en cours d'exécution pendant que vous terminez le reste du didacticiel.

Création et examen du code Python Apache Flink

Le code de l'application Python pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Installez le client Git si vous ne l'avez pas déjà fait. Pour de plus amples informations, veuillez consulter Installation de Git.
2. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/python/GettingStarted`.

Le code d'application est situé dans le fichier `getting_started.py`. Notez les informations suivantes à propos du code d'application :

- L'application utilise une source de table Kinesis pour lire à partir du flux source. L'extrait de code suivant appelle la fonction `create_table` permettant de créer la source de table Kinesis :

```
table_env.execute_sql(
        create_table(output_table_name, output_stream, output_region)
```

La fonction `create_table` utilise une commande SQL pour créer une table soutenue par la source de streaming :

```
def create_table(table_name, stream_name, region, stream_initpos = None):
    init_pos = "\n'scan.stream.initpos' = '{0}',".format(stream_initpos) if
 stream_initpos is not None else ''

    return """ CREATE TABLE {0} (
                ticker VARCHAR(6),
                price DOUBLE,
                event_time TIMESTAMP(3),
                WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND
              )
              PARTITIONED BY (ticker)
              WITH (
                'connector' = 'kinesis',
                'stream' = '{1}',
                'aws.region' = '{2}',{3}
                'format' = 'json',
                'json.timestamp-format.standard' = 'ISO-8601'
              ) """.format(table_name, stream_name, region, init_pos)
 }
```

- L'application crée deux tables, puis écrit le contenu d'une table dans l'autre.

```
    # 2. Creates a source table from a Kinesis Data Stream
    table_env.execute_sql(
        create_table(input_table_name, input_stream, input_region)
    )

    # 3. Creates a sink table writing to a Kinesis Data Stream
    table_env.execute_sql(
        create_table(output_table_name, output_stream, output_region, stream_initpos)
    )

    # 4. Inserts the source table data into the sink table
    table_result = table_env.execute_sql("INSERT INTO {0} SELECT * FROM {1}"
                         .format(output_table_name, input_table_name))
```

- L'application utilise le connecteur Flink, issu du fichier flink- sql-connector-kinesis _2.12/1.15.2.

Ajouter des dépendances tierces aux applications Python

Lorsque vous utilisez des packages Python tiers (tels que boto3), vous devez ajouter leurs dépendances transitives et les propriétés requises pour cibler ces dépendances. À un niveau élevé, pour PyPi les dépendances, vous pouvez copier les fichiers et dossiers situés dans `site-packages` le dossier de votre environnement Python pour créer une structure de répertoires comme ci-dessous :

```
PythonPackages
#   README.md
#   python-packages.py
#
####my_deps
       ####boto3
       #   #   session.py
       #   #   utils.py
       #   #   ...
       #
       ####botocore
       #   #   args.py
       #   #   auth.py
       #    ...
       ####mynonpypimodule
       #   #   mymodulefile1.py
       #   #   mymodulefile2.py
            ...
####lib
#    # flink-sql-connector-kinesis-1.15.2.jar
#    # ...
...
```

Pour ajouter le boto3 en tant que dépendance tierce :

1. Créez un environnement Python autonome (conda ou similaire) sur votre machine locale avec les dépendances requises.
2. Notez la liste initiale des packages dans le dossier `site_packages` de cet environnement.
3. `pip-install` toutes les dépendances requises pour votre application.
4. Notez les packages qui ont été ajoutés au dossier `site_packages` après l'étape 3 ci-dessus. Il s'agit des dossiers que vous devez inclure dans votre package (sous le dossier `my_deps`), organisés comme indiqué ci-dessus. Cela vous permettra de saisir la différence des packages

entre les étapes 2 et 3 afin d'identifier les dépendances de package appropriées pour votre application.

5. Fournissez `my_deps/` comme argument pour la propriété `pyFiles` du groupe de propriétés `kinesis.analytics.flink.run.options`, comme décrit ci-dessous pour la propriété `jarfiles`. Flink vous permet également de spécifier les dépendances Python à l'aide de la fonction add_python_file, mais il est important de garder à l'esprit que vous devez uniquement spécifier l'une ou l'autre, pas les deux.

> **Note**
>
> Il n'est pas nécessaire de donner un nom au dossier `my_deps`. L'important est d'enregistrer les dépendances à l'aide de `pyFiles` ou `add_python_file`. Un exemple peut être trouvé dans How to use boto3 within pyFlink.

Chargement du code Python Apache Flink

Dans cette section, vous allez créer un compartiment Amazon S3 et charger votre code d'application.

Pour télécharger le code de l'application à l'aide de la console :

1. Utilisez votre application de compression préférée pour compresser les fichiers `getting-started.py` et https://mvnrepository.com/artifact/org.apache.flink/flink - sql-connector-kinesis _2.12/1.15.2. Nommez l'archive `myapp.zip`. Si vous incluez le dossier externe dans votre archive, vous devez l'inclure dans le chemin avec le code de vos fichiers de configuration : `GettingStarted/getting-started.py`.
2. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
3. Choisissez Créer un compartiment.
4. Saisissez **`ka-app-code-<username>`** dans le champ Nom du compartiment. Ajoutez un suffixe au nom du compartiment, par exemple votre nom d'utilisateur, pour qu'il soit unique. Choisissez Suivant.
5. À l'étape Configurer les options, conservez les paramètres, puis choisissez Suivant.
6. À l'étape Définir des autorisations, conservez les paramètres, puis choisissez Suivant.
7. Choisissez Créer un compartiment.
8. Dans la console Amazon S3, choisissez le <username>compartiment ka-app-code-, puis Upload.

9. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `myapp.zip` que vous avez créé à l'étape précédente. Choisissez Suivant.
10. Vous n'avez pas besoin de modifier les paramètres de l'objet, donc choisissez Charger.

Pour charger le code d'application à l'aide de l'interface AWS CLI :

> **Note**
>
> N'utilisez pas les fonctionnalités de compression de Finder (macOS) ou de l'Explorateur Windows (Windows) pour créer l'archive `myapp.zip`. Il peut en résulter un code d'application non valide.

1. Utilisez votre application de compression préférée pour compresser les fichiers `streaming-file-sink.py` et https://mvnrepository.com/artifact/org.apache.flink/flink - sql-connector-kinesis _2.12/1.15.2.

   > **Note**
   >
   > N'utilisez pas les fonctionnalités de compression de Finder (macOS) ou de l'Explorateur Windows (Windows) pour créer l'archive myapp.zip. Il peut en résulter un code d'application non valide.

2. Utilisez votre application de compression préférée pour compresser les fichiers `getting-started.py` et https://mvnrepository.com/artifact/org.apache.flink/ flink-sql-connector-kinesis /1.15.2. Nommez l'archive `myapp.zip`. Si vous incluez le dossier externe dans votre archive, vous devez l'inclure dans le chemin avec le code de vos fichiers de configuration : `GettingStarted/getting-started.py`.
3. Exécutez la commande suivante :

   ```
   $ aws s3 --region aws region cp myapp.zip s3://ka-app-code-<username>
   ```

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l'application de service géré pour Apache Flink

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Description, saisissez **`My java test app`**.
   - Pour Exécution, choisissez Apache Flink.
   - Laissez la version sur Apache Flink 1.15.2 (version recommandée).
4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rôle : `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Configuration de l'application

Procédez comme suit pour configurer l'application.

Pour configurer l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **ka-app-code-*<username>***.
   - Pour le chemin de l'objet Amazon S3, saisissez **myapp.zip**.
3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Sous Propriétés, sélectionnez Ajouter un groupe.
5. Saisissez :

ID du groupe	Clé	Valeur
consumer.config.0	**input.stream.name**	**ExampleInputStream**
consumer.config.0	**aws.region**	**us-west-2**
consumer.config.0	**scan.stream.initpos**	**LATEST**

   Choisissez Enregistrer.
6. Sous Propriétés, sélectionnez à nouveau Ajouter un groupe.
7. Saisissez :

ID du groupe	Clé	Valeur
producer.config.0	**output.stream.name**	**ExampleOutputStream**
producer.config.0	**aws.region**	**us-west-2**
producer.config.0	**shard.count**	**1**

8. Sous Propriétés, sélectionnez à nouveau Ajouter un groupe. Pour ID du groupe, saisissez **kinesis.analytics.flink.run.options**. Ce groupe de propriétés spécial indique à votre application où se trouvent ses ressources de code. Pour plus d'informations, consultez Spécification de vos fichiers de code.

9. Saisissez :

ID du groupe	Clé	Valeur
kinesis.analytics.flink.run.options	**python**	**getting-started.py**
kinesis.analytics.flink.run.options	**jarfile**	**flink-sql-connector-kinesis-1.15.2.jar**

10. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
11. Pour la CloudWatch journalisation, cochez la case Activer.
12. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation Amazon, Managed Service for Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder au compartiment Amazon S3.

Pour modifier la politique IAM afin d'ajouter des autorisations au compartiment S3

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **kinesis-analytics-service-MyApplication-us-west-2** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.

4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*012345678901*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/myapp.zip"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
```

```
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Exécution de l'application

Le graphique des tâches Flink peut être visualisé en exécutant l'application, en ouvrant le tableau de bord Apache Flink et en choisissant la tâche Flink souhaitée.

Arrêt de l'application

Pour arrêter l'application, sur la MyApplicationpage, choisissez Arrêter. Confirmez l'action.

Étape suivante

Nettoyage des ressources AWS

Nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Mise en route (Python).

Cette rubrique contient les sections suivantes.

- Suppression de votre application de service géré pour Apache Flink
- Supprimer vos flux de données Kinesis
- Supprimer vos objets et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources

Suppression de votre application de service géré pour Apache Flink

Procédez comme suit pour supprimer l'application.

Pour supprimer l’application

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Managed Service for Apache Flink, choisissez MyApplication.
3. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer vos flux de données Kinesis

1. Ouvrez la console du service géré pour Apache Flink à l’adresse https://console.aws.amazon.com/flink
2. Dans le panneau Kinesis Data Streams, ExampleInputStreamsélectionnez.
3. Sur la ExampleInputStreampage, choisissez Supprimer Kinesis Stream, puis confirmez la suppression.
4. Sur la page Kinesis Streams, choisissez le ExampleOutputStream, choisissez Actions, choisissez Supprimer, puis confirmez la suppression.

Supprimer vos objets et votre compartiment Amazon S3

Procédez comme suit pour supprimer vos objets et votre compartiment S3.

Pour supprimer vos objets et votre compartiment S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le <username>compartiment ka-app-code-.

3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

Utilisez la procédure suivante pour supprimer vos ressources IAM.

Pour supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.
8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos CloudWatch ressources

Pour supprimer vos CloudWatch ressources, procédez comme suit.

Pour supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Mise en route (Scala)

> **Note**
>
> À partir de la version 1.15, Flink n'utilise plus Scala. Les applications peuvent désormais utiliser l'API Java depuis n'importe quelle version de Scala. Flink utilise toujours Scala dans quelques composants clés en interne, mais n'expose pas Scala dans le chargeur de classes du code de l'utilisateur. Pour cette raison, les utilisateurs doivent ajouter des dépendances Scala dans leurs archives JAR.
>
> Pour plus d'informations sur les modifications apportées à Scala dans Flink 1.15, consultez Scala Free in One Fifteen.

Dans cet exercice, vous allez créer une application de service géré pour Apache Flink dédiée à Scala avec un flux Kinesis comme source et comme récepteur.

Cette rubrique contient les sections suivantes :

- Création de ressources dépendantes
- Écriture d'exemples d'enregistrements dans le flux d'entrée
- Téléchargez et examinez le code de l'application
- Avant d'utiliser le service géré pour Apache Flink pour la première fois, exécutez les tâches suivantes :
- Création et exécution de l'application (console)
- Création et exécution de l'application (CLI)
- Nettoyage des ressources AWS

Création de ressources dépendantes

Avant de créer une application de service géré pour Apache Flink dans le cadre de cet exercice, vous commencez par créer les ressources dépendantes suivantes :

- Deux flux Kinesis pour l'entrée et la sortie.
- Un compartiment Amazon S3 pour stocker le code de l'application (`ka-app-code-<username>`)

Vous pouvez créer les flux Kinesis et un compartiment S3 à l'aide de la console. Pour obtenir des instructions sur la création de ces ressources, consultez les rubriques suivantes :

- Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams. Nommez vos flux de données **`ExampleInputStream`** et **`ExampleOutputStream`**.

  Pour créer les flux de données (AWS CLI)

  - Pour créer le premier flux (`ExampleInputStream`) utilisez la commande AWS CLI Amazon Kinesis create-stream suivante.

    ```
    aws kinesis create-stream \
        --stream-name ExampleInputStream \
        --shard-count 1 \
        --region us-west-2 \
        --profile adminuser
    ```

  - Pour créer le second flux utilisé par l'application pour écrire la sortie, exécutez la même commande en remplaçant le nom du flux par `ExampleOutputStream`.

    ```
    aws kinesis create-stream \
        --stream-name ExampleOutputStream \
        --shard-count 1 \
        --region us-west-2 \
        --profile adminuser
    ```

- Comment créer un compartiment S3 ? dans le Guide de l'utilisateur de la console Amazon Simple Storage Service. Donnez au compartiment Amazon S3 un nom unique au monde en ajoutant votre nom de connexion, tel que **`ka-app-code-<username>`**.

Autres ressources

Lorsque vous créez votre application, le service géré pour Apache Flink crée les ressources Amazon CloudWatch suivantes si elles n'existent pas déjà :

- Un groupe de journaux appelé `/AWS/KinesisAnalytics-java/MyApplication`
- Un flux de journaux appelé `kinesis-analytics-log-stream`

Écriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette section, vous utilisez un script Python pour écrire les exemples d'enregistrements dans le flux pour que l'application les traite.

> **Note**
>
> Cette section nécessite le kit AWS SDK for Python (Boto).

> **Note**
>
> Le script Python de cette section utilise l'interface AWS CLI. Vous devez configurer votre interface AWS CLI pour utiliser les informations d'identification de votre compte et la région par défaut. Pour configurer votre interface AWS CLI, saisissez :
>
> ```
> aws configure
> ```

1. Créez un fichier nommé `stock.py` avec le contenu suivant :

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
```

```
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")


if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Exécutez le script `stock.py` :

```
$ python stock.py
```

   Maintenez le script en cours d'exécution pendant que vous terminez le reste du didacticiel.

Téléchargez et examinez le code de l'application

Le code d'application Python pour cet exemple est disponible à partir de GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Installez le client Git si vous ne l'avez pas déjà fait. Pour de plus amples informations, veuillez consulter Installation de Git.
2. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/scala/GettingStarted`.

Notez les informations suivantes à propos du code d'application :

- Un fichier `build.sbt` contient des informations sur la configuration et les dépendances de l'application, y compris les bibliothèques du service géré pour Apache Flink.
- Le fichier `BasicStreamingJob.scala` contient la méthode principale qui définit la fonctionnalité de l'application.
- L'application utilise une source Kinesis pour lire à partir du flux source. L'extrait de code suivant crée la source Kinesis :

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
 defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

L'application utilise également un récepteur Kinesis pour écrire dans le flux de résultats. L'extrait de code suivant crée le récepteur Kinesis :

```
private def createSink: KinesisStreamsSink[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val outputProperties = applicationProperties.get("ProducerConfigProperties")

  KinesisStreamsSink.builder[String]
    .setKinesisClientProperties(outputProperties)
    .setSerializationSchema(new SimpleStringSchema)
    .setStreamName(outputProperties.getProperty(streamNameKey,
 defaultOutputStreamName))
    .setPartitionKeyGenerator((element: String) => String.valueOf(element.hashCode))
    .build
}
```

- L’application crée les connecteurs source et récepteur pour accéder aux ressources externes à l'aide d'un objet StreamExecutionEnvironment.
- L'application crée les connecteurs source et récepteur à l'aide de propriétés d’application dynamiques. Les propriétés d’exécution de l'application sont lues pour configurer les connecteurs. Pour de plus amples informations sur les propriétés d’exécution, veuillez consulter Runtime Properties.

Avant d'utiliser le service géré pour Apache Flink pour la première fois, exécutez les tâches suivantes :

Dans cette section, vous allez compiler et charger votre code d'application dans le compartiment Amazon S3 que vous avez créé dans la section Création de ressources dépendantes.

Compilation du code d'application

Dans cette section, vous utilisez l'outil de construction SBT pour créer le code Scala de l'application. Pour installer SBT, voir Install sbt with cs setup. Vous devez également installer le kit de développement Java (JDK). Consultez Prerequisites for Completing the Exercises.

1. Pour utiliser votre code d'application, vous le compilez et l'intégrez dans un fichier JAR. Vous pouvez compiler et empaqueter votre code avec SBT :

```
sbt assembly
```

2. Si la compilation de l'application aboutit, le fichier suivant est créé :

```
target/scala-3.2.0/getting-started-scala-1.0.jar
```

Chargement du code Scala Apache Flink

Dans cette section, vous allez créer un compartiment Amazon S3 et charger votre code d'application.

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez Créer un compartiment.
3. Saisissez `ka-app-code-<username>` dans le champ Nom du compartiment. Ajoutez un suffixe au nom du compartiment, par exemple votre nom d'utilisateur, pour qu'il soit unique. Choisissez Suivant.
4. À l'étape Configurer les options, conservez les paramètres, puis choisissez Suivant.
5. À l'étape Définir des autorisations, conservez les paramètres, puis choisissez Suivant.
6. Choisissez Créer un compartiment.
7. Ouvrez le compartiment `ka-app-code-<username>`, puis choisissez Charger.
8. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `getting-started-scala-1.0.jar` que vous avez créé à l'étape précédente.
9. Vous n'avez pas besoin de modifier les paramètres de l’objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l'application (console)

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Description, saisissez **`My scala test app`**.
   - Pour Exécution, choisissez Apache Flink.
   - Laissez la version sur Apache Flink 1.15.2 (version recommandée).
4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rôle : `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Configuration de l'application

Procédez comme suit pour configurer l'application.

Pour configurer l'application

1. Sur la page MonApplication, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **ka-app-code-*<username>***.
   - Pour le chemin de l'objet Amazon S3, saisissez **getting-started-scala-1.0.jar.**.
3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Sous Propriétés, sélectionnez Ajouter un groupe.
5. Entrez ce qui suit :

ID du groupe	Clé	Valeur
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

   Choisissez Enregistrer.
6. Sous Propriétés, sélectionnez à nouveau Ajouter un groupe.
7. Entrez ce qui suit :

ID du groupe	Clé	Valeur
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**
ProducerConfigProperties	**aws.region**	**us-west-2**

8. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
9. Pour Journalisation CloudWatch, cochez la case Activer.
10. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la journalisation Amazon CloudWatch, le service géré pour Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder au compartiment Amazon S3.

Pour modifier la politique IAM afin d'ajouter des autorisations au compartiment S3

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **`kinesis-analytics-service-MyApplication-us-west-2`** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*`012345678901`*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
```

```
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/getting-started-scala-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
```

```
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Exécution de l'application

Le graphique des tâches Flink peut être visualisé en exécutant l'application, en ouvrant le tableau de bord Apache Flink et en choisissant la tâche Flink souhaitée.

Arrêt de l'application

Pour arrêter l'application, sur la page MonApplication, choisissez Arrêter. Confirmez l'action.

Création et exécution de l'application (CLI)

Dans cette section, vous allez utiliser l'interface AWS Command Line Interface pour créer et exécuter l'application de service géré pour Apache Flink. Utilisez la commande AWS CLI kinesisanalyticsv2 pour créer et interagir avec les applications de service géré pour Apache Flink.

Créer une stratégie d'autorisations

> **Note**
>
> Vous devez créer une stratégie d'autorisations et un rôle pour votre application. Si vous ne créez pas ces ressources IAM, votre application ne peut pas accéder à ses flux de données et de journaux.

Vous commencez par créer une stratégie d'autorisations avec deux instructions : une qui accorde des autorisations pour l'action de lecture sur le flux source et une autre qui accorde des autorisations pour les actions d'écriture sur le flux récepteur. Vous attachez ensuite la politique à un rôle IAM (que vous allez créer dans la section suivante). Ainsi, lorsque le service géré pour Apache Flink assume le rôle, le service dispose des autorisations nécessaires pour lire à partir du flux source et écrire dans le flux récepteur.

Utilisez le code suivant pour créer la stratégie d'autorisations `AKReadSourceStreamWriteSinkStream`. Remplacez **`username`** par le nom d'utilisateur que vous avez utilisé pour créer le compartiment Amazon S3 pour stocker le code d'application. Remplacez l'ID de compte dans Amazon Resource Names (ARN) (**`(012345678901)`**) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/getting-started-scala-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
```

```
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Pour obtenir des instructions étape par étape pour créer une stratégie d'autorisations, veuillez consulter Didacticiel : créer et attacher votre première stratégie gérée par le client dans le Guide de l'utilisateur IAM.

Créer une politique IAM

Dans cette section, vous créez un rôle IAM que l'application de service géré pour Apache Flink peut assumer pour lire un flux source et écrire dans le flux récepteur.

Le service géré pour Apache Flink ne peut pas accéder à votre flux sans autorisation. Vous utilisez un rôle IAM pour accorder ces autorisations. Deux politiques sont attachées à chaque rôle IAM. La politique d'approbation accorde à Managed Service for Apache Flink l'autorisation d'assumer le rôle, et la politique d'autorisation détermine ce que Managed Service for Apache Flink peut faire après avoir assumé le rôle.

Vous attachez la stratégie d'autorisations que vous avez créée dans la section précédente à ce rôle.

Pour créer un rôle IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, choisissez Rôles, puis Créer un Rôle.
3. Sous Sélectionner le type d'identité approuvée, choisissez Service AWS.
4. Sous Choisir le service qui utilisera ce rôle, choisissez EC2.
5. Sous Sélectionnez votre cas d'utilisation, choisissez service géré pour Apache Flink.
6. Sélectionnez Suivant : autorisations.
7. Dans la page Attacher des stratégies d'autorisations, choisissez Suivant : vérification. Vous attachez des stratégies d'autorisations après avoir créé le rôle.
8. Sur la page Créer un rôle, saisissez **`MF-stream-rw-role`** pour le Nom du rôle. Sélectionnez Créer un rôle.

   Vous venez de créer un nouveau rôle IAM appelé `MF-stream-rw-role`. Ensuite, vous mettez à jour les stratégies d'approbation et d'autorisation pour le rôle.
9. Attachez la politique d'autorisation au rôle.

   > **Note**
   >
   > Dans le cadre de cet exercice, le service géré pour Apache Flink assume ce rôle à la fois pour la lecture des données à partir d'un flux de données Kinesis (source) et pour l'écriture des résultats dans un autre flux de données Kinesis. Vous attachez donc la stratégie que vous avez créée à l'étape précédente, Créer une stratégie d'autorisations.

   a. Sur la page Récapitulatif, choisissez l'onglet Autorisations.
   b. Choisissez Attacher des stratégies.
   c. Dans la zone de recherche, saisissez **`AKReadSourceStreamWriteSinkStream`** (la stratégie que vous avez créée dans la section précédente).
   d. Sélectionnez Actions, puis Attacher une stratégie.

Vous avez maintenant créé le rôle d'exécution de service que votre application utilise pour accéder aux ressources. Notez l'ARN du nouveau rôle.

Pour obtenir des instructions étape par étape sur la création d'un rôle, consultez la rubrique Création d'un rôle IAM (console) dans le Guide de l'utilisateur IAM.

Pour créer l'application

Copiez le code JSON suivant dans un fichier nommé `create_request.json`. Remplacez l'exemple d'ARN du rôle par l'ARN du rôle que vous avez créé précédemment. Remplacez le suffixe de l'ARN du compartiment (nom d'utilisateur) par le suffixe que vous avez choisi dans la section précédente. Remplacez l'exemple d'ID de compte (012345678901) dans le rôle d'exécution de service par votre ID de compte.

```
{
    "ApplicationName": "getting_started",
    "ApplicationDescription": "Scala getting started application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "getting-started-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
```

```
            }
         ]
      }
    },
    "CloudWatchLoggingOptions": [
      {
         "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
      }
   ]
}
```

Exécutez CreateApplication avec la requête suivante pour créer l'application :

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

L'application est maintenant créée. Vous démarrez l'application dans l'étape suivante.

Démarrage de l'application

Dans cette section, vous utilisez l'action StartApplication pour démarrer l'application.

Pour démarrer l'application

1. Copiez le code JSON suivant dans un fichier nommé `start_request.json`.

   ```
   {
       "ApplicationName": "getting_started",
       "RunConfiguration": {
           "ApplicationRestoreConfiguration": {
            "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
            }
       }
   }
   ```

2. Exécutez l'action `StartApplication` avec la demande précédente pour démarrer l'application :

   ```
   aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
   ```

L'application est maintenant en cours d'exécution. Vous pouvez consulter les métriques du service géré pour Apache Flink dans la console Amazon CloudWatch pour vérifier que l'application fonctionne.

Arrêt de l'application

Dans cette section, vous allez utiliser l'action StopApplication pour arrêter l'application.

Pour arrêter l'application

1. Copiez le code JSON suivant dans un fichier nommé `stop_request.json`.

```
{
   "ApplicationName": "s3_sink"
}
```

2. Exécutez l'action `StopApplication` avec la demande précédente pour arrêter l'application :

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

L'application est maintenant arrêtée.

Ajouter une option de journalisation CloudWatch

Vous pouvez utiliser l'interface AWS CLI pour ajouter un flux de journaux Amazon CloudWatch à votre application. Pour obtenir des informations sur l'utilisation de CloudWatch Logs avec votre application, veuillez consulter Setting Up Application Logging.

Mettre à jour des propriétés d'environnement

Dans cette section, vous utilisez l'action UpdateApplication pour modifier les propriétés d'environnement de l'application sans recompiler le code de l'application. Dans cet exemple, vous modifiez la région des flux source et de destination.

Pour mettre à jour des propriétés d'environnement pour l'application

1. Copiez le code JSON suivant dans un fichier nommé `update_properties_request.json`.

```
{
      "ApplicationName": "getting_started",
```

```
       "CurrentApplicationVersionId": 1,
       "ApplicationConfigurationUpdate": {
        "EnvironmentPropertyUpdates": {
           "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
            }
           ]
        }
    }
```

2. Exécutez l'action `UpdateApplication` avec la demande précédente pour mettre à jour les propriétés de l'environnement :

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Mise à jour du code de l’application

Lorsque vous devez mettre à jour le code de votre application avec une nouvelle version de votre package de code, vous utilisez l'action UpdateApplication de l’interface CLI.

> **Note**
>
> Pour charger une nouvelle version du code de l'application portant le même nom de fichier, vous devez spécifier la nouvelle version de l'objet. Pour de plus amples informations sur l’utilisation des versions d’objet Amazon S3, veuillez consulter Activation et désactivation de la gestion des versions.

Pour utiliser l'interface AWS CLI, supprimez votre ancien package de code de votre compartiment Amazon S3, téléchargez la nouvelle version et appelez `UpdateApplication` en spécifiant le même compartiment Amazon S3 et le même nom d'objet, ainsi que la nouvelle version de l'objet. L'application redémarrera avec le nouveau package de code.

L'exemple de demande d'action `UpdateApplication` suivant recharge le code de l'application et redémarre l'application. Mettez à jour l'`CurrentApplicationVersionId` à la version actuelle de l'application. Vous pouvez vérifier la version actuelle de l'application à l'aide des actions `ListApplications` ou `DescribeApplication`. Mettez à jour le suffixe du nom du compartiment (<username>) avec le suffixe que vous avez choisi dans la section Création de ressources dépendantes.

```
{{
    "ApplicationName": "getting_started",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-<username>",
                    "FileKeyUpdate": "getting-started-scala-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Fenêtre bascule.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer vos flux de données Kinesis
- Supprimer votre objet et votre compartiment Amazon S3
- Supprimer vos ressources IAM

- Supprimer vos ressources CloudWatch

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. dans le panneau Service géré pour Apache Flink, sélectionnez MonApplication.
3. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer vos flux de données Kinesis

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Kinesis Data Streams, sélectionnez ExampleInputStream.
3. Sur la page ExampleInputStream, choisissez Supprimer le flux Kinesis, puis confirmez la suppression.
4. Sur la page Flux Kinesis, sélectionnez ExampleOutputStream, Actions, Supprimer, puis confirmez la suppression.

Supprimer votre objet et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code-***<username>***.
3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la stratégie Kinesis-Analytics-Service-MyApplication-US-West-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez la stratégie Kinesis-Analytics-Service-MyApplication-US-West-2.

8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos ressources CloudWatch

1. Ouvrez la console CloudWatch à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux /aws/kinesis-analytics/MyApplication.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Création d’applications de service géré pour Apache Flink avec Apache Beam

Vous pouvez utiliser l’environnement Apache Beam avec votre application de service géré pour Apache Flink pour traiter les données de streaming. Les applications de service géré pour Apache Flink qui utilisent Apache Beam utilisent l’exécuteur Apache Flink pour exécuter les pipelines Beam.

Pour un didacticiel sur l’utilisation d’Apache Beam dans une application de service géré pour Apache Flink, consultez Utilisation de CloudFormation avec le service géré pour Apache Flink.

Cette rubrique contient les sections suivantes :

- Utilisation d’Apache Beam avec le service géré pour Apache Flink
- Capacités Beam
- Création d'une application à l'aide d'Apache Beam

Utilisation d’Apache Beam avec le service géré pour Apache Flink

Notez ce qui suit à propos de l'utilisation de l'exécuteur Apache Flink avec le service géré pour Apache Flink :

- Les métriques Apache Beam ne sont pas visibles dans le service géré pour Apache Flink.
- Apache Beam est uniquement pris en charge avec les applications de service géré pour Apache Flink qui utilisent Apache Flink version 1.8 ou ultérieure. Apache Beam n’est pas pris en charge avec les applications de service géré pour Apache Flink qui utilisent Apache Flink version 1.6.

Capacités Beam

Le service géré pour Apache Flink prend en charge les mêmes fonctionnalités d'Apache Beam que l’exécuteur Apache Flink. Pour plus d'informations sur les fonctionnalités prises en charge par l'exécuteur Apache Flink, consultez la matrice de compatibilité Beam.

Nous vous recommandons de tester votre application Apache Flink dans le service géré pour Apache Flink afin de vérifier que nous prenons en charge toutes les fonctionnalités dont votre application a besoin.

Création d'une application à l'aide d'Apache Beam

Dans cet exercice, vous allez créer une application de service géré pour Apache Flink qui transforme les données à l'aide d'Apache Beam. Apache Beam est un modèle de programmation pour le traitement des données de streaming. Pour des informations sur l'utilisation d'Apache Beam avec le service géré pour Apache Flink, consultez Utilisation d'Apache Beam.

> **Note**
>
> Pour configurer les prérequis requis pour cet exercice, commencez par terminer l'exercice Mise en route (DataStream API).

Cette rubrique contient les sections suivantes :

- Création de ressources dépendantes
- Écriture d'exemples d'enregistrements dans le flux d'entrée
- Téléchargez et examinez le code de l'application
- Compilation du code d'application
- Chargement du code Java Apache Flink
- Création et exécution de l'application de service géré pour Apache Flink
- Nettoyage des ressources AWS
- Étapes suivantes

Création de ressources dépendantes

Avant de créer une application de service géré pour Apache Flink dans le cadre de cet exercice, vous commencez par créer les ressources dépendantes suivantes :

- Deux flux de données Kinesis (`ExampleInputStream` et `ExampleOutputStream`)
- Un compartiment Amazon S3 pour stocker le code de l'application (`ka-app-code-<username>`)

Vous pouvez créer les flux Kinesis et un compartiment S3 à l'aide de la console. Pour obtenir des instructions sur la création de ces ressources, consultez les rubriques suivantes :

- Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams. Nommez vos flux de données **`ExampleInputStream`** et **`ExampleOutputStream`**.
- Comment créer un compartiment S3 ? dans le Guide de l'utilisateur de la console Amazon Simple Storage Service. Donnez au compartiment Amazon S3 un nom unique au monde en ajoutant votre nom de connexion, tel que **`ka-app-code-<username>`**.

Écriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette section, vous utilisez un script Python pour écrire des chaînes aléatoires dans le flux pour que l'application les traite.

> **Note**
>
> Cette section nécessite le kit AWS SDK for Python (Boto).

1. Créez un fichier nommé `ping.py` avec le contenu suivant :

```
import json
import boto3
import random

kinesis = boto3.client('kinesis')

while True:
        data = random.choice(['ping', 'telnet', 'ftp', 'tracert', 'netstat'])
        print(data)
        kinesis.put_record(
                StreamName="ExampleInputStream",
                Data=data,
                PartitionKey="partitionkey")
```

2. Exécutez le script `ping.py` :

```
$ python ping.py
```

   Maintenez le script en cours d'exécution pendant que vous terminez le reste du didacticiel.

Téléchargez et examinez le code de l'application

Le code d'application Java pour cet exemple est disponible à partir de GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Installez le client Git si vous ne l'avez pas déjà fait. Pour de plus amples informations, veuillez consulter Installation de Git.
2. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/Beam`.

Le code d'application est situé dans le fichier `BasicBeamStreamingJob.java`. Notez les informations suivantes à propos du code d'application :

- L'application utilise Apache Beam ParDo pour traiter les enregistrements entrants en invoquant une fonction de transformation personnalisée appelée `PingPongFn`.

  Le code pour appeler la fonction `PingPongFn` est le suivant :

```
.apply("Pong transform",
    ParDo.of(new PingPongFn())
```

- Les applications de service géré pour Apache Flink qui utilisent Apache Beam requièrent les composants suivants. Si vous n'incluez pas ces composants et versions dans votre fichier `pom.xml`, votre application charge des versions incorrectes à partir des dépendances de l'environnement, et comme les versions ne correspondent pas, votre application se bloque au moment de l'exécution.

```
<jackson.version>2.10.2</jackson.version>
...
<dependency>
    <groupId>com.fasterxml.jackson.module</groupId>
    <artifactId>jackson-module-jaxb-annotations</artifactId>
    <version>2.10.2</version>
</dependency>
```

- La fonction de transformation `PingPongFn` transmet les données d'entrée dans le flux de sortie, sauf si les données d'entrée sont un ping, auquel cas elle émet la chaîne pong\n vers le flux de sortie.

  Le code de la fonction de transformation est le suivant :

```
    private static class PingPongFn extends DoFn<KinesisRecord, byte[]> {
    private static final Logger LOG = LoggerFactory.getLogger(PingPongFn.class);

    @ProcessElement
    public void processElement(ProcessContext c) {
        String content = new String(c.element().getDataAsBytes(),
 StandardCharsets.UTF_8);
        if (content.trim().equalsIgnoreCase("ping")) {
            LOG.info("Ponged!");
            c.output("pong\n".getBytes(StandardCharsets.UTF_8));
        } else {
            LOG.info("No action for: " + content);
            c.output(c.element().getDataAsBytes());
        }
    }
}
```

Compilation du code d'application

Pour compiler l'application, procédez comme suit :

1. Installez Java et Maven si ce n'est pas déjà fait. Pour plus d'informations, consultez Prérequis dans le didacticiel Mise en route (DataStream API).
2. Compilez l'application à l'aide de la commande suivante :

```
mvn package -Dflink.version=1.15.3 -Dflink.version.minor=1.8
```

> **Note**
>
> Le code source fourni repose sur les bibliothèques de Java 11.

La compilation de l'application crée le fichier JAR de l'application (`target/basic-beam-app-1.0.jar`).

Chargement du code Java Apache Flink

Dans cette section, vous allez charger votre code d'application dans le compartiment Amazon S3 que vous avez créé dans la section Création de ressources dépendantes.

1. Dans la console Amazon S3, choisissez le compartiment ka-app-code-***<username>***, puis choisissez Charger.
2. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `basic-beam-app-1.0.jar` que vous avez créé à l'étape précédente.
3. Vous n'avez pas besoin de modifier les paramètres de l'objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l'application de service géré pour Apache Flink

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Exécution, choisissez Apache Flink.

> **Note**
>
> Le service géré pour Apache Flink utilise Apache Flink version 1.15.2.

- Laissez le menu déroulant de la version sur Apache Flink version 1.15.2 (version recommandée).

4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **kinesis-analytics-MyApplication-us-west-2**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-MyApplication-us-west-2`
> - Rôle : `kinesis-analytics-MyApplication-us-west-2`

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder aux flux de données Kinesis.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **kinesis-analytics-service-MyApplication-us-west-2** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*012345678901*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
```

```
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/basic-beam-app-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
```

```
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Configuration de l'application

1. Sur la page MonApplication, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **ka-app-code-*<username>***.
   - Pour le chemin de l'objet Amazon S3, saisissez **basic-beam-app-1.0.jar**.
3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Entrez ce qui suit :

ID du groupe	Clé	Valeur
BeamApplicationProperties	**InputStreamName**	**ExampleInputStream**
BeamApplicationProperties	**OutputStreamName**	**ExampleOutputStream**
BeamApplicationProperties	**AwsRegion**	**us-west-2**

5. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
6. Pour Journalisation CloudWatch, cochez la case Activer.
7. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la journalisation CloudWatch, Managed Service for Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`
>
> Ce flux de journaux est utilisé pour surveiller l'application. Il ne s'agit pas du même flux de journaux que celui utilisé par l'application pour envoyer les résultats.

Exécution de l'application

Le graphique des tâches Flink peut être visualisé en exécutant l'application, en ouvrant le tableau de bord Apache Flink et en choisissant la tâche Flink souhaitée.

Vous pouvez consulter les métriques du service géré pour Apache Flink dans la console CloudWatch pour vérifier que l'application fonctionne.

Nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Fenêtre bascule.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer vos flux de données Kinesis
- Supprimer votre objet et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos ressources CloudWatch

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink

2. dans le panneau Service géré pour Apache Flink, sélectionnez MonApplication.
3. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer vos flux de données Kinesis

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Kinesis Data Streams, sélectionnez ExampleInputStream.
3. Sur la page ExampleInputStream, choisissez Supprimer le flux Kinesis, puis confirmez la suppression.
4. Sur la page Flux Kinesis, sélectionnez ExampleOutputStream, Actions, Supprimer, puis confirmez la suppression.

Supprimer votre objet et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code-***<username>***.
3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la stratégie Kinesis-Analytics-Service-MyApplication-US-West-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez la stratégie Kinesis-Analytics-Service-MyApplication-US-West-2.
8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos ressources CloudWatch

1. Ouvrez la console CloudWatch à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.

3. Choisissez le groupe de journaux /aws/kinesis-analytics/MyApplication.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Étapes suivantes

Maintenant que vous avez créé et exécuté une application basique de service géré pour Apache Flink qui transforme les données à l'aide d'Apache Beam, consultez l'application suivante pour un exemple de solution plus avancée de service géré pour Apache Flink.

- Atelier de streaming Beam sur le service géré pour Apache Flink : dans cet atelier, nous explorons un exemple de bout en bout qui combine les aspects de lots et de streaming dans un pipeline Apache Beam uniforme.

Ateliers de formation, laboratoires et mises en œuvre de solutions

Les end-to-end exemples suivants illustrent le service géré avancé pour les solutions Apache Flink.

Rubriques

- Développement local des applications Apache Flink en local avant de les déployer vers le service géré pour Apache Flink
- Détection des événements à l'aide du service géré pour Apache Flink Studio
- Solution de streaming de données AWS pour Amazon Kinesis
- Clickstream Lab avec Apache Flink et Apache Kafka
- Mise à l'échelle personnalisée avec Application Auto Scaling
- Tableau de CloudWatch bord Amazon
- AWS Streaming Data Solution pour Amazon MSK
- Plus de services gérés pour les solutions Apache Flink sur GitHub

Développement local des applications Apache Flink en local avant de les déployer vers le service géré pour Apache Flink

Cet atelier vous montrera les bases pour démarrer et commencer à développer des applications Apache Flink localement dans le but à long terme de les déployer sur le service géré pour Apache Flink.

La solution se trouve ici : Guide de démarrage du développement local avec Apache Flink

Détection des événements à l'aide du service géré pour Apache Flink Studio

Cet atelier décrit la détection des événements à l'aide du service géré pour Apache Flink Studio et son déploiement en tant qu'application de service géré pour Apache Flink

La solution se trouve ici : Détection d'événements avec le service géré pour Apache Flink

Solution de streaming de données AWS pour Amazon Kinesis

La solution de streaming de données AWS pour Amazon Kinesis configure automatiquement les services AWS nécessaires pour capturer, stocker, traiter et diffuser facilement des données de streaming. La solution propose plusieurs options pour résoudre les problèmes d'utilisation de données en streaming. L'option Managed Service for Apache Flink fournit un exemple de end-to-end streaming ETL illustrant une application réelle qui exécute des opérations analytiques sur des données de taxis simulées à New York.

Chaque solution comprend les composants suivants :

- Un package AWS CloudFormation pour déployer l'exemple complet.
- Un CloudWatch tableau de bord pour afficher les métriques des applications.
- CloudWatch des alarmes sur les métriques les plus pertinentes de l'application.
- Tous les rôles et politiques IAM nécessaires.

La solution se trouve ici : Solution de streaming de données pour Amazon Kinesis

Clickstream Lab avec Apache Flink et Apache Kafka

Un laboratoire de bout en bout pour les cas d'utilisation d’Amazon Managed Streaming for Apache Kafka pour le stockage de streaming et le service géré pour Apache Flink pour les applications Apache Flink pour le traitement des flux.

La solution se trouve ici : Clickstream Lab

Mise à l’échelle personnalisée avec Application Auto Scaling

Exemple qui aide les utilisateurs à dimensionner automatiquement leur service géré pour les applications Apache Flink à l'aide d'Application Auto Scaling. Cela permet aux utilisateurs de configurer des politiques de mise à l’échelle personnalisées et des attributs de mise à l’échelle personnalisés.

Les solutions peuvent être trouvées ici :

- Service géré pour la mise à l'échelle automatique de l'application Apache Flink
- Mise à l’échelle planifiée

Pour plus d'informations sur la possibilité d'effectuer un dimensionnement personnalisé, consultez Activer le dimensionnement planifié et basé sur des métriques pour Amazon Managed Service pour Apache Flink.

Tableau de CloudWatch bord Amazon

Exemple de CloudWatch tableau de bord pour surveiller le service géré pour les applications Apache Flink. L'exemple de tableau de bord inclut également une application de démonstration pour aider à démontrer les fonctionnalités du tableau de bord.

La solution se trouve ici : Tableau de bord des métriques dans le service géré pour Apache Flink

AWS Streaming Data Solution pour Amazon MSK

AWS Streaming Data Solution pour Amazon MSK fournit des modèles AWS CloudFormation dans lesquels les données circulent entre les producteurs, le stockage en streaming, les consommateurs et les destinations.

La solution se trouve ici : AWS Streaming Data Solution pour Amazon MSK

Plus de services gérés pour les solutions Apache Flink sur GitHub

Les end-to-end exemples suivants illustrent le service géré avancé pour les solutions Apache Flink et sont disponibles sur GitHub :

- Service géré Amazon pour Apache Flink — Utilitaire de comparaison
- Gestionnaire d'instantané – Service géré Amazon pour Apache Flink
- Streaming ETL avec Apache Flink et le service géré Amazon pour Apache Flink
- Analyse des sentiments en temps réel sur la base des commentaires des clients

Utilitaires

Les utilitaires suivants peuvent faciliter l'utilisation du service géré pour Apache Flink :

Rubriques

- Gestionnaire d’instantanés
- Analyse comparative

Gestionnaire d’instantanés

Il est recommandé que les applications Flink déclenchent régulièrement des points de sauvegarde/des instantanés afin de permettre une reprise en cas de panne plus fluide. Le gestionnaire d’instantanés automatise cette tâche et offre les avantages suivants :

- prend un nouvel instantané d'un service géré pour Apache Flink en cours d'exécution
- obtient le nombre d'instantanés d'application
- vérifie si le nombre d'instantanés est supérieur au nombre requis
- supprime les anciens instantanés qui sont plus anciens que le nombre requis

Pour un exemple, voir Snapshot Manager.

Analyse comparative

L'utilitaire d’analyse comparative Flink du service géré pour Apache Flink permet de planifier les capacités, de tester les intégrations et de comparer des services gérés pour Apache Flink pour les applications Apache Flink.

Pour un exemple, consultez Benchmarking

Service géré Amazon pour Apache Flink : exemples

Cette section fournit des exemples de création et d'utilisation d'applications dans le service géré pour Apache Flink. Ils incluent des exemples de code et des step-by-step instructions pour vous aider à créer un service géré pour les applications Apache Flink et à tester vos résultats.

Avant d'explorer ces exemples, nous vous recommandons de consulter les éléments suivants :

- Comment ça marche
- Mise en route (DataStream API)

> **Note**
>
> Ces exemples supposent que vous utilisez la région USA Ouest (Oregon) (`us-west-2`). Si vous utilisez une autre région, mettez à jour le code de votre application, les commandes et les rôles IAM de manière appropriée.

Rubriques

- DataStream Exemples d'API
- Exemples Python
- Exemples Scala :

DataStream Exemples d'API

Les exemples suivants montrent comment créer des applications à l'aide de l' DataStream API Apache Flink.

Rubriques

- Exemple : fenêtre bascule
- Exemple : fenêtre défilante
- Exemple : écriture dans un compartiment Amazon S3
- Didacticiel : utilisation d'une application de service géré pour Apache Flink afin de répliquer des données d'une rubrique dans un cluster MSK à une autre dans un VPC

- Exemple : utilisation d'un consommateur EFO avec un flux de données Kinesis
- Exemple : écrire dans Kinesis Data Firehose
- Exemple : lire à partir d'un flux Kinesis dans un autre compte
- Didacticiel : utilisation d'un truststore personnalisé avec Amazon MSK

Exemple : fenêtre bascule

Dans cet exercice, vous allez créer une application de service géré pour Apache Flink qui agrège les données à l'aide d'une fenêtre bascule. L'agrégation est activée par défaut dans Flink. Pour la désactiver, utilisez la commande suivante :

```
sink.producer.aggregation-enabled' = 'false'
```

> **Note**
>
> Pour configurer les prérequis requis pour cet exercice, commencez par terminer l'exercice Mise en route (DataStream API).

Cette rubrique contient les sections suivantes :

- Création de ressources dépendantes
- Écriture d'exemples d'enregistrements dans le flux d'entrée
- Téléchargez et examinez le code de l'application
- Compilation du code d'application
- Chargement du code Java Apache Flink
- Création et exécution de l'application de service géré pour Apache Flink
- Nettoyage des ressources AWS

Création de ressources dépendantes

Avant de créer une application de service géré pour Apache Flink dans le cadre de cet exercice, vous commencez par créer les ressources dépendantes suivantes :

- Deux flux de données Kinesis (`ExampleInputStream` et `ExampleOutputStream`)
- Un compartiment Amazon S3 pour stocker le code de l'application (`ka-app-code-<username>`)

Vous pouvez créer les flux Kinesis et un compartiment S3 à l'aide de la console. Pour obtenir des instructions sur la création de ces ressources, consultez les rubriques suivantes :

- Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams. Nommez votre flux de données **ExampleInputStream** et**ExampleOutputStream**.
- Comment créer un compartiment S3 ? dans le Guide de l'utilisateur de la console Amazon Simple Storage Service. Donnez au compartiment Amazon S3 un nom unique au monde en ajoutant votre nom de connexion, tel que **ka-app-code-*<username>***.

Écriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette section, vous utilisez un script Python pour écrire les exemples d'enregistrements dans le flux pour que l'application les traite.

> **Note**
>
> Cette section nécessite le kit AWS SDK for Python (Boto).

1. Créez un fichier nommé `stock.py` avec le contenu suivant :

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
```

```
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey")


    if __name__ == '__main__':
        generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Exécutez le script `stock.py` :

```
$ python stock.py
```

Maintenez le script en cours d'exécution pendant que vous terminez le reste du didacticiel.

Téléchargez et examinez le code de l'application

Le code de l'application Java pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Installez le client Git si vous ne l'avez pas déjà fait. Pour de plus amples informations, veuillez consulter Installation de Git.
2. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/TumblingWindow`.

Le code d'application est situé dans le fichier `TumblingWindowStreamingJob.java`. Notez les informations suivantes à propos du code d'application :

- L'application utilise une source Kinesis pour lire à partir du flux source. L'extrait de code suivant crée la source Kinesis :

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- Ajoutez l'instruction d’importation suivante :

```
import
 org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows; //
flink 1.13 onward
```

- L'application utilise l'opérateur `timeWindow` pour déterminer le nombre de valeurs de chaque symbole boursier sur une fenêtre de 5 secondes. Le code suivant crée l'opérateur et envoie les données agrégées vers un nouveau récepteur Kinesis Data Streams :

```
input.flatMap(new Tokenizer()) // Tokenizer for generating words
                    .keyBy(0) // Logically partition the stream for each word

                    .window(TumblingProcessingTimeWindows.of(Time.seconds(5))) //
Flink 1.13 onward
                    .sum(1) // Sum the number of words per partition
                    .map(value -> value.f0 + "," + value.f1.toString() + "\n")
                    .addSink(createSinkFromStaticConfig());
```

Compilation du code d'application

Pour compiler l'application, procédez comme suit :

1. Installez Java et Maven si ce n'est pas déjà fait. Pour plus d'informations, consultez Prérequis dans le didacticiel Mise en route (DataStream API).
2. Compilez l'application à l'aide de la commande suivante :

```
mvn package -Dflink.version=1.15.3
```

> **Note**
>
> Le code source fourni repose sur les bibliothèques de Java 11.

La compilation de l'application crée le fichier JAR de l'application (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

Chargement du code Java Apache Flink

Dans cette section, vous allez charger votre code d'application dans le compartiment Amazon S3 que vous avez créé dans la section Création de ressources dépendantes.

1. Dans la console Amazon S3, choisissez le <username>compartiment ka-app-code-, puis Upload.
2. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `aws-kinesis-analytics-java-apps-1.0.jar` que vous avez créé à l'étape précédente.
3. Vous n'avez pas besoin de modifier les paramètres de l'objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l'application de service géré pour Apache Flink

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Exécution, choisissez Apache Flink.

     > **Note**
     >
     > Le service géré pour Apache Flink utilise Apache Flink version 1.15.2.

   - Laissez le menu déroulant de la version sur Apache Flink version 1.15.2 (version recommandée).
4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-MyApplication-us-west-2`
> - Rôle : `kinesisanalytics-MyApplication-us-west-2`

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder aux flux de données Kinesis.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **`kinesis-analytics-service-MyApplication-us-west-2`** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*012345678901*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
```

```
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Configuration de l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **`ka-app-code-<username>`**.
   - Pour le chemin de l'objet Amazon S3, saisissez **`aws-kinesis-analytics-java-apps-1.0.jar`**.
3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **`kinesis-analytics-MyApplication-us-west-2`**.
4. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
5. Pour la CloudWatch journalisation, cochez la case Activer.
6. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation, le service géré pour Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`
>
> Ce flux de journaux est utilisé pour surveiller l'application. Il ne s'agit pas du même flux de journaux que celui utilisé par l'application pour envoyer les résultats.

Exécution de l'application

1. Sur la MyApplicationpage, choisissez Exécuter. Laissez l'option Exécuter sans instantané sélectionnée et confirmez l'action.
2. Lorsque l'application est en cours d'exécution, actualisez la page. La console affiche le graphique de l'application.

Vous pouvez vérifier les métriques du service géré pour Apache Flink sur la CloudWatch console pour vérifier que l'application fonctionne.

Nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Fenêtre bascule.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer vos flux de données Kinesis
- Supprimer votre objet et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. dans le panneau Managed Service for Apache Flink, sélectionnez MyApplication.
3. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer vos flux de données Kinesis

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Kinesis Data Streams, ExampleInputStreamsélectionnez.
3. Sur la ExampleInputStreampage, choisissez Supprimer Kinesis Stream, puis confirmez la suppression.
4. Sur la page Kinesis Streams, choisissez le ExampleOutputStream, choisissez Actions, choisissez Supprimer, puis confirmez la suppression.

Supprimer votre objet et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code -. <username>

3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.
8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Exemple : fenêtre défilante

> **Note**
>
> Pour configurer les prérequis requis pour cet exercice, commencez par terminer l'exercice Mise en route (DataStream API).

Cette rubrique contient les sections suivantes :

- Création de ressources dépendantes
- Écriture d'exemples d'enregistrements dans le flux d'entrée
- Téléchargez et examinez le code de l'application
- Compilation du code d'application

- Chargement du code Java Apache Flink
- Création et exécution de l'application de service géré pour Apache Flink
- Nettoyage des ressources AWS

Création de ressources dépendantes

Avant de créer une application de service géré pour Apache Flink dans le cadre de cet exercice, vous commencez par créer les ressources dépendantes suivantes :

- Deux flux de données Kinesis (`ExampleInputStream` et `ExampleOutputStream`).
- Un compartiment Amazon S3 pour stocker le code de l'application (`ka-app-code-<username>`)

Vous pouvez créer les flux Kinesis et un compartiment S3 à l'aide de la console. Pour obtenir des instructions sur la création de ces ressources, consultez les rubriques suivantes :

- Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams. Nommez vos flux de données **`ExampleInputStream`** et **`ExampleOutputStream`**.
- Comment créer un compartiment S3 ? dans le Guide de l'utilisateur de la console Amazon Simple Storage Service. Donnez au compartiment Amazon S3 un nom unique au monde en ajoutant votre nom de connexion, tel que **`ka-app-code-<username>`**.

Écriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette section, vous utilisez un script Python pour écrire les exemples d'enregistrements dans le flux pour que l'application les traite.

> **Note**
>
> Cette section nécessite le kit AWS SDK for Python (Boto).

1. Créez un fichier nommé `stock.py` avec le contenu suivant :

```
import datetime
import json
import random
```

```
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

2. Exécutez le script `stock.py` :

```
$ python stock.py
```

Maintenez le script en cours d'exécution pendant que vous terminez le reste du didacticiel.

Téléchargez et examinez le code de l'application

Le code de l'application Java pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Installez le client Git si vous ne l'avez pas déjà fait. Pour de plus amples informations, veuillez consulter Installation de Git.
2. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/SlidingWindow`.

Le code d'application est situé dans le fichier `SlidingWindowStreamingJobWithParallelism.java`. Notez les informations suivantes à propos du code d'application :

- L'application utilise une source Kinesis pour lire à partir du flux source. L'extrait de code suivant crée la source Kinesis :

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- L'application utilise l'opérateur `timeWindow` pour trouver la valeur minimale de chaque symbole boursier sur une fenêtre de 10 secondes qui glisse de 5 secondes. Le code suivant crée l'opérateur et envoie les données agrégées vers un nouveau récepteur Kinesis Data Streams :
- Ajoutez l'instruction d’importation suivante :

```
import
 org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows; //
flink 1.13 onward
```

- L'application utilise l'opérateur `timeWindow` pour déterminer le nombre de valeurs de chaque symbole boursier sur une fenêtre de 5 secondes. Le code suivant crée l'opérateur et envoie les données agrégées vers un nouveau récepteur Kinesis Data Streams :

```
input.flatMap(new Tokenizer()) // Tokenizer for generating words
                .keyBy(0) // Logically partition the stream for each word

  .window(TumblingProcessingTimeWindows.of(Time.seconds(5))) //Flink 1.13 onward
                .sum(1) // Sum the number of words per partition
                .map(value -> value.f0 + "," + value.f1.toString() + "\n")
                .addSink(createSinkFromStaticConfig());
```

Compilation du code d'application

Pour compiler l'application, procédez comme suit :

1. Installez Java et Maven si ce n'est pas déjà fait. Pour plus d'informations, consultez Prérequis dans le didacticiel Mise en route (DataStream API).
2. Compilez l'application à l'aide de la commande suivante :

```
mvn package -Dflink.version=1.15.3
```

> **Note**
>
> Le code source fourni repose sur les bibliothèques de Java 11.

La compilation de l'application crée le fichier JAR de l'application (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

Chargement du code Java Apache Flink

Dans cette section, vous allez charger votre code d'application dans le compartiment Amazon S3 que vous avez créé dans la section Création de ressources dépendantes.

1. Dans la console Amazon S3, choisissez le <username>compartiment ka-app-code-, puis choisissez Upload.
2. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `aws-kinesis-analytics-java-apps-1.0.jar` que vous avez créé à l'étape précédente.
3. Vous n'avez pas besoin de modifier les paramètres de l'objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l'application de service géré pour Apache Flink

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Exécution, choisissez Apache Flink.
   - Laissez le menu déroulant de la version sur Apache Flink version 1.15.2 (version recommandée).
4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rôle : `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder aux flux de données Kinesis.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **`kinesis-analytics-service-MyApplication-us-west-2`** créée pour vous par la console dans la section précédente.

3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*012345678901*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
```

```
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Configuration de l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **ka-app-code-*<username>***.
   - Pour le chemin de l'objet Amazon S3, saisissez **aws-kinesis-analytics-java-apps-1.0.jar**.
3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
5. Pour la CloudWatch journalisation, cochez la case Activer.
6. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation Amazon, Managed Service for Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`
>
> Ce flux de journaux est utilisé pour surveiller l'application. Il ne s'agit pas du même flux de journaux que celui utilisé par l'application pour envoyer les résultats.

Configuration du parallélisme de l’application

Cet exemple d'application utilise l'exécution parallèle de tâches. Le code d'application suivant définit le parallélisme de l'opérateur `min` :

```
.setParallelism(3) // Set parallelism for the min operator
```

Le parallélisme de l’application ne peut pas être supérieur au parallélisme provisionné, dont la valeur par défaut est 1. Pour augmenter le parallélisme de votre application, utilisez l’action AWS CLI suivante :

```
aws kinesisanalyticsv2 update-application
      --application-name MyApplication
      --current-application-version-id <VersionId>
      --application-configuration-update "{\"FlinkApplicationConfigurationUpdate
\": { \"ParallelismConfigurationUpdate\": {\"ParallelismUpdate\": 5,
 \"ConfigurationTypeUpdate\": \"CUSTOM\" }}}"
```

Vous pouvez récupérer l'ID de version actuel de l'application à l'aide ListApplicationsdes actions DescribeApplicationou.

Exécution de l'application

Le graphique des tâches Flink peut être visualisé en exécutant l'application, en ouvrant le tableau de bord Apache Flink et en choisissant la tâche Flink souhaitée.

Vous pouvez vérifier les métriques du service géré pour Apache Flink sur la CloudWatch console pour vérifier que l'application fonctionne.

Nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Fenêtre défilante.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer vos flux de données Kinesis
- Supprimer votre objet et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le panneau Managed Service for Apache Flink, choisissez MyApplication.
3. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer vos flux de données Kinesis

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Kinesis Data Streams, ExampleInputStreamsélectionnez.
3. Sur la ExampleInputStreampage, choisissez Supprimer Kinesis Stream, puis confirmez la suppression.
4. Sur la page Kinesis Streams, choisissez le ExampleOutputStream, choisissez Actions, choisissez Supprimer, puis confirmez la suppression.

Supprimer votre objet et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code -. <username>

3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.
8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Exemple : écriture dans un compartiment Amazon S3

Dans cet exercice, vous allez créer une application de service géré pour Apache Flink qui utilise un flux de données Kinesis comme source et un compartiment Amazon S3 comme récepteur. À l'aide du récepteur, vous pouvez vérifier la sortie de l'application dans la console Amazon S3.

> **Note**
>
> Pour configurer les prérequis requis pour cet exercice, commencez par terminer l'exercice Mise en route (DataStream API).

Cette rubrique contient les sections suivantes :

- Création de ressources dépendantes

- Écriture d'exemples d'enregistrements dans le flux d'entrée
- Téléchargez et examinez le code de l'application
- Modification du code de l'application
- Compilation du code d'application
- Chargement du code Java Apache Flink
- Création et exécution de l’application de service géré pour Apache Flink
- Vérifier la sortie de l'application
- Facultatif : personnaliser la source et le récepteur
- Nettoyage des ressources AWS

Création de ressources dépendantes

Avant de créer un service géré pour Apache Flink dans le cadre de cet exercice, vous commencez par créer les ressources dépendantes suivantes :

- Un flux de données Kinesis (`ExampleInputStream`).
- Un compartiment Amazon S3 pour stocker le code et la sortie de l'application (`ka-app-code-<username>`)

> **Note**
>
> Le service géré pour Apache Flink ne peut pas écrire de données sur Amazon S3 lorsque le chiffrement côté serveur est activé sur le service géré pour Apache Flink.

Vous pouvez créer un flu Kinesis et un compartiment S3 à l'aide de la console. Pour obtenir des instructions sur la création de ces ressources, consultez les rubriques suivantes :

- Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams. Attribuez un nom à votre flux de données **`ExampleInputStream`**.
- Comment créer un compartiment S3 ? dans le Guide de l'utilisateur de la console Amazon Simple Storage Service. Donnez au compartiment Amazon S3 un nom unique au monde en ajoutant votre nom de connexion, tel que **`ka-app-code-<username>`**. Créez deux dossiers (**`code`** et **`data`**) dans le compartiment Amazon S3.

L'application crée les CloudWatch ressources suivantes si elles n'existent pas déjà :

- Un groupe de journaux appelé `/AWS/KinesisAnalytics-java/MyApplication`.
- Un flux de journaux appelé `kinesis-analytics-log-stream`

Écriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette section, vous utilisez un script Python pour écrire les exemples d'enregistrements dans le flux pour que l'application les traite.

> **Note**
>
> Cette section nécessite le kit AWS SDK for Python (Boto).

1. Créez un fichier nommé `stock.py` avec le contenu suivant :

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")
```

```
    if __name__ == '__main__':
        generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Exécutez le script `stock.py` :

```
$ python stock.py
```

Maintenez le script en cours d'exécution pendant que vous terminez le reste du didacticiel.

Téléchargez et examinez le code de l'application

Le code de l'application Java pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Installez le client Git si vous ne l'avez pas déjà fait. Pour de plus amples informations, veuillez consulter Installation de Git.
2. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/S3Sink`.

Le code d'application est situé dans le fichier `S3StreamingSinkJob.java`. Notez les informations suivantes à propos du code d'application :

- L'application utilise une source Kinesis pour lire à partir du flux source. L'extrait de code suivant crée la source Kinesis :

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- Vous devez ajouter l'instruction d’importation suivante :

```
import
 org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows;
```

- L'application utilise un récepteur Apache Flink S3 pour écrire sur Amazon S3.

Le récepteur lit les messages dans une fenêtre défilante, code les messages dans des objets du compartiment S3 et envoie les objets codés au récepteur S3. Le code suivant code les objets à envoyer à Amazon S3 :

```
input.map(value -> { // Parse the JSON
                JsonNode jsonNode = jsonParser.readValue(value, JsonNode.class);
                return new Tuple2<>(jsonNode.get("ticker").toString(), 1);
            }).returns(Types.TUPLE(Types.STRING, Types.INT))
            .keyBy(v -> v.f0) // Logically partition the stream for each word
            .window(TumblingProcessingTimeWindows.of(Time.minutes(1)))
            .sum(1) // Count the appearances by ticker per partition
            .map(value -> value.f0 + " count: " + value.f1.toString() + "\n")
            .addSink(createS3SinkFromStaticConfig());
```

> **Note**
>
> L'application utilise un objet `StreamingFileSink` Flink pour écrire sur Amazon S3. Pour plus d'informations à ce sujet`StreamingFileSink`, consultez StreamingFileSinkla documentation d'Apache Flink.

Modification du code de l'application

Dans cette section, vous modifiez le code de l'application pour écrire le résultat dans votre compartiment Amazon S3.

Mettez à jour la ligne suivante avec votre nom d'utilisateur pour spécifier l'emplacement de sortie de l'application :

```
private static final String s3SinkPath = "s3a://ka-app-code-<username>/data";
```

Compilation du code d'application

Pour compiler l'application, procédez comme suit :

1. Installez Java et Maven si ce n'est pas déjà fait. Pour plus d'informations, consultez Prérequis dans le didacticiel Mise en route (DataStream API).
2. Compilez l'application à l’aide de la commande suivante :

```
mvn package -Dflink.version=1.15.3
```

La compilation de l'application crée le fichier JAR de l'application (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

> **Note**
>
> Le code source fourni repose sur les bibliothèques de Java 11.

Chargement du code Java Apache Flink

Dans cette section, vous allez charger votre code d'application dans le compartiment Amazon S3 que vous avez créé dans la section Création de ressources dépendantes.

1. Dans la console Amazon S3, choisissez le <username>compartiment ka-app-code-, accédez au dossier de code, puis choisissez Upload.
2. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `aws-kinesis-analytics-java-apps-1.0.jar` que vous avez créé à l'étape précédente.
3. Vous n'avez pas besoin de modifier les paramètres de l'objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l'application de service géré pour Apache Flink

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :

- Pour Nom de l'application, saisissez **`MyApplication`**.
- Pour Exécution, choisissez Apache Flink.
- Laissez le menu déroulant de la version sur Apache Flink version 1.15.2 (version recommandée).

4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Pour Nom de l'application, saisissez **`MyApplication`**.
> - Pour Exécution, choisissez Apache Flink.
> - Laissez la version sur Apache Flink 1.15.2 (version recommandée).

6. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
7. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez un service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rôle : `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder aux flux de données Kinesis.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **`kinesis-analytics-service-MyApplication-us-west-2`** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*012345678901*) par votre ID de compte. Remplacez <username> par votre nom d'utilisateur.

```
{
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:Abort*",
                "s3:DeleteObject*",
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-<username>",
                "arn:aws:s3:::ka-app-code-<username>/*"
            ]
          },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:*"
            ]
        },
        {
            "Sid": "ListCloudwatchLogStreams",
            "Effect": "Allow",
```

```
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:%LOG_GROUP_PLACEHOLDER
%:log-stream:*"
            ]
        },
        {
            "Sid": "PutCloudwatchLogs",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:region:account-id:log-group:%LOG_GROUP_PLACEHOLDER
%:log-stream:%LOG_STREAM_PLACEHOLDER%"
            ]
        }
        ,
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },

    ]
}
```

Configuration de l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **ka-app-code-*<username>***.
   - Pour le chemin de l'objet Amazon S3, saisissez **code/aws-kinesis-analytics-java-apps-1.0.jar**.
3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **kinesis-analytics-MyApplication-us-west-2**.

4. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
5. Pour la CloudWatch journalisation, cochez la case Activer.
6. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation, le service géré pour Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`
>
> Ce flux de journaux est utilisé pour surveiller l'application. Il ne s'agit pas du même flux de journaux que celui utilisé par l'application pour envoyer les résultats.

Exécution de l'application

1. Sur la MyApplicationpage, choisissez Exécuter. Laissez l'option Exécuter sans instantané sélectionnée et confirmez l'action.
2. Lorsque l'application est en cours d'exécution, actualisez la page. La console affiche le graphique de l'application.

Vérifier la sortie de l'application

Dans la console Amazon S3, ouvrez le dossier de données dans votre compartiment S3.

Au bout de quelques minutes, des objets contenant des données agrégées provenant de l'application apparaîtront.

> **Note**
>
> L'agrégation est activée par défaut dans Flink. Pour la désactiver, utilisez la commande suivante :

```
sink.producer.aggregation-enabled' = 'false'
```

Facultatif : personnaliser la source et le récepteur

Dans cette section, vous allez personnaliser les paramètres des objets source et récepteur.

> **Note**
>
> Après avoir modifié les sections de code décrites dans les sections suivantes, procédez comme suit pour recharger le code de l'application :
>
> - Répétez les étapes décrites dans la section the section called “Compilation du code d'application” pour compiler le code d'application mis à jour.
> - Répétez les étapes décrites dans la section the section called “Chargement du code Java Apache Flink” pour télécharger le code d'application mis à jour.
> - Sur la page de l'application dans la console, choisissez Configurer, puis choisissez Mettre à jour pour recharger le code d'application mis à jour dans votre application.

Cette section contient les sections suivantes :

- Configurer le partitionnement des données
- Configuration de la fréquence de lecture
- Configuration de la mise en mémoire tampon d'écriture

Configurer le partitionnement des données

Dans cette section, vous allez configurer les noms des dossiers créés par le récepteur de fichiers de streaming dans le compartiment S3. Pour ce faire, ajoutez un assignateur de compartiment au récepteur de fichiers de streaming.

Pour personnaliser les noms de dossiers créés dans le compartiment S3, procédez comme suit :

1. Ajoutez les instructions d'importation suivantes au début du fichier `S3StreamingSinkJob.java` :

```
import
 org.apache.flink.streaming.api.functions.sink.filesystem.rollingpolicies.DefaultRollingPol
import
 org.apache.flink.streaming.api.functions.sink.filesystem.bucketassigners.DateTimeBucketAss
```

2. Mettez à jour la méthode `createS3SinkFromStaticConfig()` dans le code pour qu'elle se présente comme suit :

```
private static StreamingFileSink<String> createS3SinkFromStaticConfig() {

    final StreamingFileSink<String> sink = StreamingFileSink
        .forRowFormat(new Path(s3SinkPath), new
 SimpleStringEncoder<String>("UTF-8"))
        .withBucketAssigner(new DateTimeBucketAssigner("yyyy-MM-dd--HH"))
        .withRollingPolicy(DefaultRollingPolicy.create().build())
        .build();
    return sink;
}
```

L'exemple de code précédent utilise le `DateTimeBucketAssigner` avec un format de date personnalisé pour créer des dossiers dans le compartiment S3. Le `DateTimeBucketAssigner` utilise l'heure actuelle du système pour créer les noms des compartiments. Si vous souhaitez créer un assignateur de compartiment personnalisé afin de personnaliser davantage les noms de dossiers créés, vous pouvez créer une classe qui implémente BucketAssigner. Vous implémentez votre logique personnalisée à l'aide de la méthode `getBucketId`.

Une implémentation personnalisée du `BucketAssigner` peut utiliser le paramètre Context pour obtenir plus d'informations sur un enregistrement afin de déterminer son dossier de destination.

Configuration de la fréquence de lecture

Dans cette section, vous allez configurer la fréquence des lectures sur le flux source.

Par défaut, le consommateur Kinesis Streams lit le flux source cinq fois par seconde. Cette fréquence peut entraîner des problèmes si plusieurs clients lisent le flux ou si l'application doit réessayer de lire un enregistrement. Vous pouvez éviter ces problèmes en définissant la fréquence de lecture du consommateur.

Pour définir la fréquence de lecture du client Kinesis, vous devez définir le paramètre `SHARD_GETRECORDS_INTERVAL_MILLIS`.

L'exemple de code suivant définit le paramètre `SHARD_GETRECORDS_INTERVAL_MILLIS` sur une seconde :

```
kinesisConsumerConfig.setProperty(ConsumerConfigConstants.SHARD_GETRECORDS_INTERVAL_MILLIS,
 "1000");
```

Configuration de la mise en mémoire tampon d'écriture

Dans cette section, vous allez configurer la fréquence d'écriture et les autres paramètres du récepteur.

Par défaut, l'application écrit dans le compartiment de destination toutes les minutes. Vous pouvez modifier cet intervalle et d'autres paramètres en configurant l'objet `DefaultRollingPolicy`.

> **Note**
>
> Le récepteur de fichiers de streaming Apache Flink écrit dans son compartiment de sortie chaque fois que l'application crée un point de contrôle. L'application crée un point de contrôle toutes les minutes par défaut. Pour augmenter l'intervalle d'écriture du récepteur S3, vous devez également augmenter l'intervalle de point de contrôle.

Pour configurer l'objet `DefaultRollingPolicy`, procédez comme suit :

1. Augmentez le paramètre `CheckpointInterval` de l'application. L'entrée suivante pour l'UpdateApplicationaction définit l'intervalle entre les points de contrôle à 10 minutes :

```
{
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "CheckpointConfigurationUpdate": {
            "ConfigurationTypeUpdate" : "CUSTOM",
            "CheckpointIntervalUpdate": 600000
         }
      }
   },
   "ApplicationName": "MyApplication",
```

```
   "CurrentApplicationVersionId": 5
}
```

Pour utiliser le code précédent, spécifiez la version actuelle de l'application. Vous pouvez récupérer la version de l'application à l'aide de l'ListApplicationsaction.

2. Ajoutez l'instruction d'importation suivante au début du fichier `S3StreamingSinkJob.java` :

```
import java.util.concurrent.TimeUnit;
```

3. Mettez à jour la méthode `createS3SinkFromStaticConfig` dans le fichier `S3StreamingSinkJob.java` pour qu'elle se présente comme suit :

```
private static StreamingFileSink<String> createS3SinkFromStaticConfig() {

        final StreamingFileSink<String> sink = StreamingFileSink
                .forRowFormat(new Path(s3SinkPath), new
 SimpleStringEncoder<String>("UTF-8"))
                .withBucketAssigner(new DateTimeBucketAssigner("yyyy-MM-dd--HH"))
                .withRollingPolicy(
                        DefaultRollingPolicy.create()
                            .withRolloverInterval(TimeUnit.MINUTES.toMillis(8))
                            .withInactivityInterval(TimeUnit.MINUTES.toMillis(5))
                            .withMaxPartSize(1024 * 1024 * 1024)
                            .build())
                .build();
        return sink;
    }
```

L'exemple de code précédent définit la fréquence des écritures dans le compartiment Amazon S3 à 8 minutes.

Pour plus d'informations sur la configuration du récepteur de fichiers de streaming Apache Flink, consultez la section Row-encoded Formats dans la documentation d'Apache Flink.

Nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWSque vous avez créées dans le didacticiel Amazon S3.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer votre flux de données Kinesis
- Supprimer vos objets et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le panneau Managed Service for Apache Flink, choisissez MyApplication.
3. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer votre flux de données Kinesis

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Kinesis Data Streams, ExampleInputStreamsélectionnez.
3. Sur la ExampleInputStreampage, choisissez Supprimer Kinesis Stream, puis confirmez la suppression.

Supprimer vos objets et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code -. <username>
3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.

6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.
8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Didacticiel : utilisation d'une application de service géré pour Apache Flink afin de répliquer des données d'une rubrique dans un cluster MSK à une autre dans un VPC

Le didacticiel suivant explique comment créer un Amazon VPC avec un cluster Amazon MSK et deux rubriques, et comment créer une application de service géré pour une Apache Flink qui lit une rubrique Amazon MSK et écrit dans une autre.

> **Note**
>
> Pour configurer les prérequis requis pour cet exercice, commencez par terminer l'exercice Mise en route (DataStream API).

Ce didacticiel contient les sections suivantes :

- Créer un Amazon VPC avec un cluster Amazon MSK
- Création du code de l'application
- Chargement du code Java Apache Flink
- Pour créer l'application
- Configuration de l'application
- Exécution de l'application
- Test de l'application

Créer un Amazon VPC avec un cluster Amazon MSK

Pour créer un exemple de VPC et de cluster Amazon MSK auquel accéder depuis une application de service géré pour Apache Flink, suivez le didacticiel Mise en route avec Amazon MSK.

Lorsque vous avez terminé le didacticiel, notez ce qui suit :

- À l'étape 3 : création d'une rubrique, répétez la commande `kafka-topics.sh --create` pour créer une rubrique de destination nommée `AWSKafkaTutorialTopicDestination` :

```
bin/kafka-topics.sh --create --zookeeper ZooKeeperConnectionString --replication-
factor 3 --partitions 1 --topic AWSKafkaTutorialTopicDestination
```

- Enregistrez la liste des serveurs bootstrap de votre cluster. Vous pouvez obtenir la liste des serveurs bootstrap à l'aide de la commande suivante (remplacez-la *ClusterArn*par l'ARN de votre cluster MSK) :

```
aws kafka get-bootstrap-brokers --region us-west-2 --cluster-arn ClusterArn
{...
    "BootstrapBrokerStringTls": "b-2.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-1.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-3.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094"
}
```

- Lorsque vous suivez les étapes décrites dans les didacticiels, veillez à utiliser la région AWS sélectionnée dans le code, les commandes et les entrées de console.

Création du code de l'application

Dans cette section, vous allez télécharger et compiler le fichier JAR de l'application. Nous vous recommandons d'utiliser Java 11.

Le code de l'application Java pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Installez le client Git si vous ne l'avez pas déjà fait. Pour de plus amples informations, veuillez consulter Installation de Git.
2. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Le code d'application est situé dans le fichier `amazon-kinesis-data-analytics-java-examples/KafkaConnectors/KafkaGettingStartedJob.java`. Vous pouvez examiner le code pour vous familiariser avec la structure du code de l'application de service géré pour Apache Flink.
4. Utilisez l'outil de ligne de commande Maven ou votre environnement de développement préféré pour créer le fichier JAR. Pour compiler le fichier JAR à l'aide de l'outil de ligne de commande Maven, saisissez ce qui suit :

   ```
   mvn package -Dflink.version=1.15.3
   ```

   En cas de succès de la génération, le fichier suivant est créé :

   ```
   target/KafkaGettingStartedJob-1.0.jar
   ```

   > **Note**
   >
   > Le code source fourni repose sur les bibliothèques de Java 11. Si vous utilisez un environnement de développement,

Chargement du code Java Apache Flink

Dans cette section, vous allez charger votre code d'application dans le compartiment Amazon S3 que vous avez créé dans le didacticiel Mise en route (DataStream API).

> **Note**
>
> Si vous avez supprimé le compartiment Amazon S3 du didacticiel Mise en route, suivez à nouveau l'étape the section called "Chargement du code Java Apache Flink".

1. Dans la console Amazon S3, choisissez le <username>compartiment ka-app-code-, puis Upload.
2. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `KafkaGettingStartedJob-1.0.jar` que vous avez créé à l'étape précédente.

3. Vous n'avez pas besoin de modifier les paramètres de l'objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Exécution, choisissez Apache Flink 1.15.2.
4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rôle : `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Configuration de l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:

- Pour le compartiment Amazon S3, saisissez **ka-app-code-*<username>***.
- Pour le chemin de l'objet Amazon S3, saisissez **KafkaGettingStartedJob-1.0.jar**.

3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **kinesis-analytics-MyApplication-us-west-2**.

> **Note**
>
> Lorsque vous spécifiez des ressources d'application à l'aide de la console (comme CloudWatch Logs ou un Amazon VPC), la console modifie votre rôle d'exécution d'application pour autoriser l'accès à ces ressources.

4. Sous Propriétés, sélectionnez Ajouter un groupe. Saisissez les propriétés suivantes :

ID du groupe	Clé	Valeur
KafkaSource	topic	AWSKafkaTutorialTopic
KafkaSource	bootstrap.servers	*La liste des serveurs bootstrap que vous avez enregistrée précédemment*
KafkaSource	security.protocol	SSL
KafkaSource	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSource	ssl.truststore.password	changeit

> **Note**
>
> Le ssl.truststore.password du certificat par défaut est « changeit » ; vous n'avez pas besoin de modifier cette valeur si vous utilisez le certificat par défaut.

Choisissez à nouveau Ajouter un groupe. Saisissez les propriétés suivantes :

ID du groupe	Clé	Valeur
KafkaSink	topic	AWSKafkaTutorialTopicDestination
KafkaSink	bootstrap.servers	*La liste des serveurs bootstrap que vous avez enregistrée précédemment*
KafkaSink	security.protocol	SSL
KafkaSink	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSink	ssl.truststore.password	changeit
KafkaSink	transaction.timeout.ms	1 000

Le code de l'application lit les propriétés de l'application ci-dessus pour configurer la source et le récepteur utilisés pour interagir avec votre VPC et votre cluster Amazon MSK. Pour obtenir plus d'informations sur l'utilisation des propriétés, consultez Propriétés d'exécution.

5. Sous Instantanés, choisissez Désactiver. Cela facilitera la mise à jour de l'application sans charger de données d'état de l'application non valides.
6. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
7. Pour la CloudWatch journalisation, cochez la case Activer.
8. Dans la section Cloud privé virtuel (VPC), choisissez le VPC à associer à votre application. Choisissez les sous-réseaux et le groupe de sécurité associés à votre VPC que vous souhaitez que l'application utilise pour accéder aux ressources du VPC.
9. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation, le service géré pour Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`
>
> Ce flux de journaux est utilisé pour surveiller l'application.

Exécution de l'application

Le graphique des tâches Flink peut être visualisé en exécutant l'application, en ouvrant le tableau de bord Apache Flink et en choisissant la tâche Flink souhaitée.

Test de l'application

Dans cette section, vous allez écrire des enregistrements dans a rubrique source. L'application lit les enregistrements de la rubrique source et les écrit dans la rubrique de destination. Vous vérifiez que l'application fonctionne en écrivant des enregistrements dans la rubrique source et en lisant des enregistrements dans la rubrique de destination.

Pour écrire et lire des enregistrements issus des rubriques, suivez les étapes décrites à l'Étape 6 : produire et consommer des données du didacticiel Mise en route avec Amazon MSK.

Pour lire la rubrique de destination, utilisez le nom de la rubrique de destination au lieu de la rubrique source lors de votre deuxième connexion au cluster :

```
bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString --
consumer.config client.properties --topic AWSKafkaTutorialTopicDestination --from-
beginning
```

Si aucun enregistrement n'apparaît dans la rubrique de destination, consultez la sectionImpossible d’accéder aux ressources d'un VPC de la rubrique Résolution des problèmes.

Exemple : utilisation d'un consommateur EFO avec un flux de données Kinesis

Dans cet exercice, vous allez créer une application de service géré pour Apache Flink qui lit un flux de données Kinesis à l'aide d'un consommateur Enhanced Fan-Out (EFO). Si un client Kinesis utilise EFO, le service Kinesis Data Streams lui fournit sa propre bande passante dédiée, au lieu que le consommateur partage la bande passante fixe du flux avec les autres consommateurs lisant le flux.

Pour plus d'informations sur l'utilisation d'EFO avec le consommateur Kinesis, consultez FLIP-128: Enhanced Fan Out for Kinesis Consumers.

L'application que vous créez dans cet exemple utilise AWS Kinesis Connector (flink-connector-kinesis) 1.15.3.

> **Note**
>
> Pour configurer les prérequis requis pour cet exercice, commencez par terminer l'exercice Mise en route (DataStream API).

Cette rubrique contient les sections suivantes :

- Création de ressources dépendantes
- Écriture d'exemples d'enregistrements dans le flux d'entrée
- Téléchargez et examinez le code de l'application
- Compilation du code d'application
- Chargement du code Java Apache Flink
- Création et exécution de l'application de service géré pour Apache Flink
- Nettoyage des ressources AWS

Création de ressources dépendantes

Avant de créer une application de service géré pour Apache Flink dans le cadre de cet exercice, vous commencez par créer les ressources dépendantes suivantes :

- Deux flux de données Kinesis (`ExampleInputStream` et `ExampleOutputStream`)

- Un compartiment Amazon S3 pour stocker le code de l'application (`ka-app-code-<username>`)

Vous pouvez créer les flux Kinesis et un compartiment S3 à l'aide de la console. Pour obtenir des instructions sur la création de ces ressources, consultez les rubriques suivantes :

- Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams. Nommez votre flux de données **ExampleInputStream** et **ExampleOutputStream**.
- Comment créer un compartiment S3 ? dans le Guide de l'utilisateur de la console Amazon Simple Storage Service. Donnez au compartiment Amazon S3 un nom unique au monde en ajoutant votre nom de connexion, tel que **ka-app-code-*<username>***.

Écriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette section, vous utilisez un script Python pour écrire les exemples d'enregistrements dans le flux pour que l'application les traite.

> **Note**
>
> Cette section nécessite le kit AWS SDK for Python (Boto).

1. Créez un fichier nommé `stock.py` avec le contenu suivant :

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
```

```
        while True:
            data = get_data()
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey")


    if __name__ == '__main__':
        generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Exécutez le script `stock.py` :

```
$ python stock.py
```

Maintenez le script en cours d'exécution pendant que vous terminez le reste du didacticiel.

Téléchargez et examinez le code de l'application

Le code de l'application Java pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Installez le client Git si vous ne l'avez pas déjà fait. Pour de plus amples informations, veuillez consulter Installation de Git.
2. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/EfoConsumer`.

Le code d'application est situé dans le fichier `EfoApplication.java`. Notez les informations suivantes à propos du code d'application :

- Vous activez le consommateur EFO en définissant les paramètres suivants sur le consommateur Kinesis :
  - RECORD_PUBLISHER_TYPE : définissez ce paramètre sur EFO pour que votre application utilise un consommateur EFO pour accéder aux données du flux de données Kinesis.

- EFO_CONSUMER_NAME : définissez ce paramètre sur une valeur de chaîne unique parmi les consommateurs de ce flux. La réutilisation d'un nom de consommateur dans le même flux de données Kinesis entraînera la résiliation du client qui utilisait ce nom précédemment.
- L'exemple de code suivant montre comment attribuer des valeurs aux propriétés de configuration du consommateur afin d'utiliser un consommateur EFO pour lire à partir du flux source :

```
consumerConfig.putIfAbsent(RECORD_PUBLISHER_TYPE, "EFO");
consumerConfig.putIfAbsent(EFO_CONSUMER_NAME, "basic-efo-flink-app");
```

Compilation du code d'application

Pour compiler l'application, procédez comme suit :

1. Installez Java et Maven si ce n'est pas déjà fait. Pour plus d'informations, consultez Prérequis dans le didacticiel Mise en route (DataStream API).
2. Compilez l'application à l'aide de la commande suivante :

```
mvn package -Dflink.version=1.15.3
```

> **Note**
>
> Le code source fourni repose sur les bibliothèques de Java 11.

La compilation de l'application crée le fichier JAR de l'application (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

Chargement du code Java Apache Flink

Dans cette section, vous allez charger votre code d'application dans le compartiment Amazon S3 que vous avez créé dans la section Création de ressources dépendantes.

1. Dans la console Amazon S3, choisissez le <username>compartiment ka-app-code-, puis Upload.
2. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `aws-kinesis-analytics-java-apps-1.0.jar` que vous avez créé à l'étape précédente.
3. Vous n'avez pas besoin de modifier les paramètres de l'objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l’application de service géré pour Apache Flink

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l’adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Exécution, choisissez Apache Flink.

     > **Note**
     >
     > Le service géré pour Apache Flink utilise Apache Flink version 1.15.2.

   - Laissez le menu déroulant de la version sur Apache Flink version 1.15.2 (version recommandée).
4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*

- Rôle : `kinesisanalytics-MyApplication-us-west-2`

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder aux flux de données Kinesis.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **`kinesis-analytics-service-MyApplication-us-west-2`** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*012345678901*) par votre ID de compte.

> **Note**
>
> Ces autorisations permettent à l'application d'accéder au consommateur EFO.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
```

```
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "AllStreams",
            "Effect": "Allow",
            "Action": [
                "kinesis:ListShards",
                "kinesis:ListStreamConsumers",
                "kinesis:DescribeStreamSummary"
            ],
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/*"
        },
        {
            "Sid": "Stream",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:RegisterStreamConsumer",
                "kinesis:DeregisterStreamConsumer"
            ],
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
```

```
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        },
        {
            "Sid": "Consumer",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStreamConsumer",
                "kinesis:SubscribeToShard"
            ],
            "Resource": [
                "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream/
consumer/my-efo-flink-app",
                "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream/
consumer/my-efo-flink-app:*"
            ]
        }
    ]
}
```

Configuration de l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **ka-app-code-*<username>***.
   - Pour le chemin de l'objet Amazon S3, saisissez **aws-kinesis-analytics-java-apps-1.0.jar**.
3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Sous Propriétés, sélectionnez Créer un groupe.
5. Entrez les valeurs et propriétés d'application suivantes :

ID du groupe	Clé	Valeur
ConsumerConfigProperties	**flink.stream.recordpublisher**	**EFO**

ID du groupe	Clé	Valeur
ConsumerConfigProperties	**flink.stream.efo.consumername**	**basic-efo-flink-app**
ConsumerConfigProperties	**INPUT_STREAM**	**ExampleInputStream**
ConsumerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ConsumerConfigProperties	**AWS_REGION**	**us-west-2**

6. Sous Propriétés, sélectionnez Créer un groupe.
7. Entrez les valeurs et propriétés d'application suivantes :

ID du groupe	Clé	Valeur
ProducerConfigProperties	**OUTPUT_STREAM**	**ExampleOutputStream**
ProducerConfigProperties	**AWS_REGION**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

8. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
9. Pour la CloudWatch journalisation, cochez la case Activer.
10. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation, le service géré pour Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :

- Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
- Flux de journaux : `kinesis-analytics-log-stream`

Ce flux de journaux est utilisé pour surveiller l'application. Il ne s'agit pas du même flux de journaux que celui utilisé par l'application pour envoyer les résultats.

Exécution de l'application

Le graphique des tâches Flink peut être visualisé en exécutant l'application, en ouvrant le tableau de bord Apache Flink et en choisissant la tâche Flink souhaitée.

Vous pouvez vérifier les métriques du service géré pour Apache Flink sur la CloudWatch console pour vérifier que l'application fonctionne.

Vous pouvez également consulter la console Kinesis Data Streams, dans l'onglet Enhanced fan-out du flux de données, pour trouver le nom de votre client (). basic-efo-flink-app

Nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Fenêtre EFO.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer vos flux de données Kinesis
- Supprimer votre objet et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. dans le panneau Managed Service for Apache Flink, sélectionnez MyApplication.

3. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer vos flux de données Kinesis

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Kinesis Data Streams, ExampleInputStreamsélectionnez.
3. Sur la ExampleInputStreampage, choisissez Supprimer Kinesis Stream, puis confirmez la suppression.
4. Sur la page Kinesis Streams, choisissez le ExampleOutputStream, choisissez Actions, choisissez Supprimer, puis confirmez la suppression.

Supprimer votre objet et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code -. <username>
3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l’adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.
8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.

4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Exemple : écrire dans Kinesis Data Firehose

Dans cet exercice, vous créez une application de service géré pour Apache Flink qui a un flux de données Kinesis comme source et un flux Kinesis Data Firehose comme récepteur. À l'aide du récepteur, vous pouvez vérifier la sortie de l'application dans un compartiment Amazon S3.

> **Note**
>
> Pour configurer les prérequis requis pour cet exercice, commencez par terminer l'exercice Mise en route (DataStream API).

Cette section contient les étapes suivantes :

- Création de ressources dépendantes
- Écriture d'exemples d'enregistrements dans le flux d'entrée
- Téléchargement et examen du code Java Apache Flink
- Compilation du code d'application
- Chargement du code Java Apache Flink
- Création et exécution de l'application de service géré pour Apache Flink
- Nettoyage des ressources AWS

Création de ressources dépendantes

Avant de créer un service géré pour Apache Flink dans le cadre de cet exercice, vous commencez par créer les ressources dépendantes suivantes :

- Un flux de données Kinesis (`ExampleInputStream`)
- Un flux Kinesis Data Firehose dans lequel l'application écrit la sortie (`ExampleDeliveryStream`).
- Un compartiment Amazon S3 pour stocker le code de l'application (`ka-app-code-<username>`)

Vous pouvez créer le flux Kinesis, les compartiments Amazon S3 et le flux Kinesis Data Firehose à l'aide de la console. Pour obtenir des instructions sur la création de ces ressources, consultez les rubriques suivantes :

- Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams. Attribuez un nom à votre flux de données **`ExampleInputStream`**.
- Création d'un flux de diffusion Amazon Kinesis Data Firehose dans le Guide du développeur Amazon Kinesis Data Firehose. Nommez votre flux de diffusion Kinesis Data Firehose **`ExampleDeliveryStream`**. Lorsque vous créez le flux Kinesis Data Firehose, créez également la destination S3 et le rôle IAM du flux Kinesis Data Firehose.
- Comment créer un compartiment S3 ? dans le Guide de l'utilisateur de la console Amazon Simple Storage Service. Donnez au compartiment Amazon S3 un nom unique au monde en ajoutant votre nom de connexion, tel que **`ka-app-code-<username>`**.

Écriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette section, vous utilisez un script Python pour écrire les exemples d'enregistrements dans le flux pour que l'application les traite.

> **Note**
>
> Cette section nécessite le kit AWS SDK for Python (Boto).

1. Créez un fichier nommé `stock.py` avec le contenu suivant :

```
import datetime
   import json
   import random
   import boto3

   STREAM_NAME = "ExampleInputStream"


   def get_data():
       return {
           'event_time': datetime.datetime.now().isoformat(),
           'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
           'price': round(random.random() * 100, 2)}


   def generate(stream_name, kinesis_client):
       while True:
```

```
            data = get_data()
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey")


    if __name__ == '__main__':
        generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Exécutez le script `stock.py` :

```
$ python stock.py
```

Maintenez le script en cours d'exécution pendant que vous terminez le reste du didacticiel.

Téléchargement et examen du code Java Apache Flink

Le code de l'application Java pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/FirehoseSink`.

Le code d'application est situé dans le fichier `FirehoseSinkStreamingJob.java`. Notez les informations suivantes à propos du code d'application :

- L'application utilise une source Kinesis pour lire à partir du flux source. L'extrait de code suivant crée la source Kinesis :

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- L'application utilise un récepteur Kinesis Data Firehose pour écrire des données dans un flux Kinesis Data Firehose. L'extrait de code suivant crée le récepteur Kinesis Data Firehose :

```
private static KinesisFirehoseSink<String> createFirehoseSinkFromStaticConfig() {
        Properties sinkProperties = new Properties();
        sinkProperties.setProperty(AWS_REGION, region);

        return KinesisFirehoseSink.<String>builder()
                .setFirehoseClientProperties(sinkProperties)
                .setSerializationSchema(new SimpleStringSchema())
                .setDeliveryStreamName(outputDeliveryStreamName)
                .build();
    }
```

Compilation du code d'application

Pour compiler l'application, procédez comme suit :

1. Installez Java et Maven si ce n'est pas déjà fait. Pour plus d'informations, consultez Prérequis dans le didacticiel Mise en route (DataStream API).
2. Pour utiliser le connecteur Kinesis pour l'application suivante, vous devez télécharger, construire et installer Apache Maven. Pour plus d'informations, consultez the section called “Utilisation du connecteur Kinesis Streams d’Apache Flink avec les versions précédentes d'Apache Flink”.
3. Compilez l'application à l’aide de la commande suivante :

```
mvn package -Dflink.version=1.15.3
```

> **Note**
>
> Le code source fourni repose sur les bibliothèques de Java 11.

La compilation de l'application crée le fichier JAR de l'application (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

Chargement du code Java Apache Flink

Dans cette section, vous allez charger votre code d'application dans le compartiment Amazon S3 que vous avez créé dans la section Création de ressources dépendantes.

Pour charger le code d'application

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Dans la console, choisissez le <username>compartiment ka-app-code-, puis choisissez Upload.
3. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `java-getting-started-1.0.jar` que vous avez créé à l'étape précédente.
4. Vous n'avez pas besoin de modifier les paramètres de l'objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l'application de service géré pour Apache Flink

Vous pouvez créer et exécuter une application de service géré pour Apache Flink à l'aide de la console ou de l'interface AWS CLI.

> **Note**
>
> Lorsque vous créez l'application à l'aide de la console, vos ressources AWS Identity and Access Management (IAM) et Amazon CloudWatch Logs sont créées pour vous. Lorsque vous créez l'application à l'aide de l'interface AWS CLI, vous créez ces ressources séparément.

Rubriques

- Création et exécution de l'application (console)
- Création et exécution de l'application (AWS CLI)

Création et exécution de l'application (console)

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.

3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :

   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Description, saisissez **`My java test app`**.
   - Pour Exécution, choisissez Apache Flink.

     > **Note**
     >
     > Le service géré pour Apache Flink utilise Apache Flink version 1.15.2.

   - Laissez le menu déroulant de la version sur Apache Flink version 1.15.2 (version recommandée).
4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez l'application à l'aide de la console, un rôle et une politique IAM peuvent être créés pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rôle : `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder aux flux de données Kinesis et au flux Kinesis Data Firehose.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **`kinesis-analytics-service-MyApplication-us-west-2`** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.

4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez toutes les instances de l'exemple d'ID de compte (*012345678901*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/java-getting-started-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
           ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
```

```
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteDeliveryStream",
            "Effect": "Allow",
            "Action": "firehose:*",
            "Resource": "arn:aws:firehose:us-west-2:012345678901:deliverystream/
ExampleDeliveryStream"
        }
    ]
}
```

Configuration de l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **ka-app-code-*<username>***.
   - Pour le chemin de l'objet Amazon S3, saisissez **java-getting-started-1.0.jar**.
3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
5. Pour la CloudWatch journalisation, cochez la case Activer.
6. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation, le service géré pour Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`

Exécution de l'application

Le graphique des tâches Flink peut être visualisé en exécutant l'application, en ouvrant le tableau de bord Apache Flink et en choisissant la tâche Flink souhaitée.

Arrêt de l'application

Sur la MyApplicationpage, choisissez Stop. Confirmez l'action.

Mise à jour de l'application

À l'aide de la console, vous pouvez mettre à jour les paramètres d'application tels que les paramètres de surveillance, les propriétés d'application et l'emplacement ou le nom du fichier JAR de l'application.

Sur la MyApplicationpage, choisissez Configurer. Mettez à jour les paramètres de l'application, puis choisissez Mettre à jour.

> **Note**
>
> Pour mettre à jour le code de l'application dans la console, vous devez soit modifier le nom de l'objet du fichier JAR, soit utiliser un autre compartiment S3, soit utiliser l'interface AWS CLI comme décrit dans la section the section called "Mise à jour du code de l'application". Si le nom du fichier ou le compartiment ne change pas, le code de l'application n'est pas rechargé lorsque vous choisissez Mettre à jour sur la page Configurer.

Création et exécution de l'application (AWS CLI)

Dans cette section, vous allez utiliser la AWS CLI pour créer et exécuter l'application Managed Service for Apache Flink.

Créer une stratégie d'autorisations

Vous commencez par créer une stratégie d'autorisations avec deux instructions : une qui accorde des autorisations pour l'action `read` sur le flux source et une autre qui accorde des autorisations pour les actions `write` sur le flux récepteur. Vous attachez ensuite la politique à un rôle IAM (que vous allez créer dans la section suivante). Ainsi, lorsque le service géré pour Apache Flink assume le rôle, le service dispose des autorisations nécessaires pour lire à partir du flux source et écrire dans le flux récepteur.

Utilisez le code suivant pour créer la stratégie d'autorisations `AKReadSourceStreamWriteSinkStream`. Remplacez *`username`* par le nom d'utilisateur que vous utiliserez pour créer le compartiment Amazon S3 pour stocker le code d'application. Remplacez l'ID de compte dans Amazon Resource Names (ARN) (*`012345678901`*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": ["arn:aws:s3:::ka-app-code-username",
                "arn:aws:s3:::ka-app-code-username/*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteDeliveryStream",
            "Effect": "Allow",
            "Action": "firehose:*",
            "Resource": "arn:aws:firehose:us-west-2:012345678901:deliverystream/
ExampleDeliveryStream"
        }
```

```
    ]
}
```

Pour step-by-step obtenir des instructions sur la création d'une politique d'autorisations, voir Tutoriel : créer et joindre votre première politique gérée par le client dans le guide de l'utilisateur IAM.

> **Note**
>
> Pour accéder à d'autres services Amazon, vous pouvez utiliser le AWS SDK for Java. Le service géré pour Apache Flink définit automatiquement les informations d'identification requises par le kit SDK en fonction du rôle IAM d'exécution du service associé à votre application. Aucune étape supplémentaire n'est nécessaire.

Créer un rôle IAM

Dans cette section, vous créez un rôle IAM que l'application de service géré pour Apache Flink peut assumer pour lire un flux source et écrire dans le flux récepteur.

Le service géré Managed Service for Apache Flink ne peut pas accéder à votre flux sans autorisation. Vous utilisez un rôle IAM pour accorder ces autorisations. Deux politiques sont attachées à chaque rôle IAM. La politique d'approbation accorde au service géré pour Apache Flink l'autorisation d'assumer le rôle. La politique d'autorisation détermine ce que le service géré pour Apache Flink peut faire après avoir assumé le rôle.

Vous attachez la stratégie d'autorisations que vous avez créée dans la section précédente à ce rôle.

Pour créer un rôle IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, choisissez Rôles, puis Créer un rôle.
3. Sous Sélectionner le type d'identité approuvée, choisissez Service AWS. Sous Choisir le service qui utilisera ce rôle, choisissez EC2. Sous Sélectionner votre cas d'utilisation, choisissez Kinesis Analytics.

   Sélectionnez Next: Permissions (Étape suivante : autorisations).
4. Dans la page Attacher des stratégies d'autorisations, choisissez Suivant : vérification. Vous attachez des stratégies d'autorisations après avoir créé le rôle.

5. Sur la page Créer un rôle, saisissez **`MF-stream-rw-role`** pour le Nom du rôle. Sélectionnez Créer un rôle.

   Vous venez de créer un nouveau rôle IAM appelé `MF-stream-rw-role`. Ensuite, vous mettez à jour les stratégies d'approbation et d'autorisation pour le rôle.

6. Attachez la politique d'autorisation au rôle.

   > **Note**
   >
   > Dans le cadre de cet exercice, le service géré pour Apache Flink assume ce rôle à la fois pour la lecture des données à partir d'un flux de données Kinesis (source) et pour l'écriture des résultats dans un autre flux de données Kinesis. Vous attachez donc la stratégie que vous avez créée à l'étape précédente, the section called “Créer une stratégie d'autorisations”.

   a. Sur la page Récapitulatif, choisissez l'onglet Autorisations.

   b. Choisissez Attacher des stratégies.

   c. Dans la zone de recherche, saisissez **`AKReadSourceStreamWriteSinkStream`** (la stratégie que vous avez créée dans la section précédente).

   d. Choisissez la ReadSourceStreamWriteSinkStream politique AK, puis choisissez Attach policy.

Vous avez maintenant créé le rôle d'exécution de service que votre application utilisera pour accéder aux ressources. Notez l'ARN du nouveau rôle.

Pour step-by-step obtenir des instructions sur la création d'un rôle, consultez la section Création d'un rôle IAM (console) dans le guide de l'utilisateur IAM.

Création de l’application de service géré pour Apache Flink

1. Copiez le code JSON suivant dans un fichier nommé `create_request.json`. Remplacez l'exemple d'ARN du rôle par l'ARN du rôle que vous avez créé précédemment. Remplacez le suffixe de l'ARN du compartiment par le suffixe que vous avez choisi dans la section the section called “Création de ressources dépendantes” (`ka-app-code-<username>`). Remplacez l'exemple d'ID de compte (*`012345678901`*) dans le rôle d'exécution de service par votre ID de compte.

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "java-getting-started-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        }
      }
    }
}
```

2. Exécutez l'action `CreateApplication` avec la demande précédente pour créer l'application :

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

L'application est maintenant créée. Vous démarrez l'application dans l'étape suivante.

Démarrage de l'application

Dans cette section, vous utilisez l'action `StartApplication` pour démarrer l'application.

Pour démarrer l'application

1. Copiez le code JSON suivant dans un fichier nommé `start_request.json`.

```
{
    "ApplicationName": "test",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
```

```
    }
}
```

2. Exécutez l'action StartApplication avec la demande précédente pour démarrer l'application :

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

L'application est maintenant en cours d'exécution. Vous pouvez consulter les métriques du service géré pour Apache Flink sur la CloudWatch console Amazon pour vérifier que l'application fonctionne.

Arrêt de l'application

Dans cette section, vous allez utiliser l'action StopApplication pour arrêter l'application.

Pour arrêter l'application

1. Copiez le code JSON suivant dans un fichier nommé `stop_request.json`.

```
{
    "ApplicationName": "test"
}
```

2. Exécutez l'action StopApplication avec la demande suivante pour arrêter l'application :

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

L'application est maintenant arrêtée.

Ajouter une option de CloudWatch journalisation

Vous pouvez utiliser le AWS CLI pour ajouter un flux de CloudWatch journal Amazon à votre application. Pour plus d'informations sur l'utilisation de CloudWatch Logs avec votre application, consultezthe section called “Configuration de la journalisation”.

Mise à jour du code de l’application

Lorsque vous devez mettre à jour le code de votre application avec une nouvelle version de votre package de code, vous utilisez l'actionUpdateApplication de l’interface AWS CLI.

Pour utiliser l'interface AWS CLI, supprimez votre ancien package de code de votre compartiment Amazon S3, téléchargez la nouvelle version et appelez `UpdateApplication` en spécifiant le même compartiment Amazon S3 et le même nom d'objet.

L'exemple de demande d'action `UpdateApplication` suivant recharge le code de l'application et redémarre l'application. Mettez à jour l'`CurrentApplicationVersionId` à la version actuelle de l'application. Vous pouvez vérifier la version actuelle de l'application à l'aide des actions `ListApplications` ou `DescribeApplication`. Mettez à jour le suffixe du nom du compartiment ((**<*username*>**) avec le suffixe que vous avez choisi dans la section the section called "Création de ressources dépendantes".

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "java-getting-started-1.0.jar"
                }
            }
        }
    }
}
```

Nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Mise en route.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer votre flux de données Kinesis
- Supprimer votre flux Kinesis Data Firehose
- Supprimer votre objet et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le panneau Managed Service for Apache Flink, choisissez MyApplication.
3. Choisissez Configurer.
4. Dans la section Instantanés, choisissez Désactiver, puis sélectionnez Mettre à jour.
5. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer votre flux de données Kinesis

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Kinesis Data Streams, ExampleInputStreamsélectionnez.
3. Sur la ExampleInputStreampage, choisissez Supprimer Kinesis Stream, puis confirmez la suppression.

Supprimer votre flux Kinesis Data Firehose

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Kinesis Data Firehose, choisissez. ExampleDeliveryStream
3. Sur la ExampleDeliveryStreampage, choisissez Supprimer le flux Kinesis Data Firehose, puis confirmez la suppression.

Supprimer votre objet et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code -. <username>
3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.
4. Si vous avez créé un compartiment Amazon S3 pour la destination de votre flux Kinesis Data Firehose, supprimez également ce compartiment.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.

2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Si vous avez créé une nouvelle politique pour votre flux Kinesis Data Firehose, supprimez-la également.
7. Dans la barre de navigation, choisissez Rôles.
8. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.
9. Choisissez Supprimer le rôle, puis confirmez la suppression.
10. Si vous avez créé un nouveau rôle pour votre flux Kinesis Data Firehose, supprimez-le également.

Supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Exemple : lire à partir d'un flux Kinesis dans un autre compte

Cet exemple montre comment créer une application de service géré pour Apache Flink qui lit les données d'un flux Kinesis dans un autre compte. Dans cet exemple, vous allez utiliser un compte pour le flux Kinesis source et un second compte pour l'application de service géré pour Apache Flink et le flux Kinesis récepteur.

Cette rubrique contient les sections suivantes :

- Prérequis
- Configuration
- Créer un flux Kinesis source
- Création et mise à jour des rôles et politiques IAM
- Mettre à jour le script Python
- Mettre à jour l’application Java

- Créez, mettez à jour et exécutez l'application.

Prérequis

- Dans ce didacticiel, vous allez modifier l'exemple Mise en route pour lire les données d'un flux Kinesis dans un autre compte. Terminez le didacticiel Mise en route (DataStream API) avant de continuer.
- Vous avez besoin de deux comptes AWS pour suivre ce didacticiel : un pour le flux source, et un pour l'application et le flux récepteur. Utilisez le compte AWS que vous avez utilisé pour le didacticiel Mise en route pour l'application et le flux récepteur. Utilisez un autre compte AWS pour le flux source.

Configuration

Vous accéderez à vos deux comptes AWS en utilisant des profils nommés. Modifiez vos informations d'identification AWS et vos fichiers de configuration pour inclure deux profils contenant les informations de région et de connexion pour vos deux comptes.

L'exemple de fichier d'informations d'identification suivant contient deux profils nommés, `ka-source-stream-account-profile` et `ka-sink-stream-account-profile`. Utilisez le compte que vous avez utilisé pour le didacticiel Mise en route pour le compte du flux récepteur.

```
[ka-source-stream-account-profile]
aws_access_key_id=AKIAIOSFODNN7EXAMPLE
aws_secret_access_key=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

[ka-sink-stream-account-profile]
aws_access_key_id=AKIAI44QH8DHBEXAMPLE
aws_secret_access_key=je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
```

L'exemple de fichier de configuration suivant contient des profils portant le même nom avec des informations de région et de format de sortie.

```
[profile ka-source-stream-account-profile]
region=us-west-2
output=json

[profile ka-sink-stream-account-profile]
region=us-west-2
```

```
output=json
```

> **Note**
>
> Ce didacticiel n'utilise pas le `ka-sink-stream-account-profile`. Il est inclus à titre d'exemple de la façon d'accéder à deux comptes AWS différents à l'aide de profils.

Pour de plus amples informations sur l'utilisation de profils nommés avec l'interface AWS CLI, consultez Profils nommés dans la documentation AWS Command Line Interface.

Créer un flux Kinesis source

Dans cette section, vous allez créer le flux Kinesis dans le compte source.

Saisissez la commande suivante pour créer le flux Kinesis que l'application utilisera pour l'entrée. Notez que le paramètre `--profile` indique le profil de compte à utiliser.

```
$ aws kinesis create-stream \
--stream-name SourceAccountExampleInputStream \
--shard-count 1 \
--profile ka-source-stream-account-profile
```

Création et mise à jour des rôles et politiques IAM

Pour autoriser l'accès aux objets entre les comptes AWS, vous devez créer un rôle et une politique IAM dans le compte source. Ensuite, vous modifiez la politique IAM dans le compte récepteur. Pour des informations sur la création des rôles et politiques IAM, veuillez consulter les rubriques suivantes dans le Guide de l'utilisateur AWS Identity and Access Management :

- Création de rôles IAM
- Création de politiques IAM

Rôles et politiques du compte récepteur

1. Modifiez la politique `kinesis-analytics-service-MyApplication-us-west-2` dans le didacticiel de mise en route. Cette politique permet d'assumer le rôle dans le compte source afin de lire le flux source.

> **Note**
>
> Lorsque vous utilisez la console pour créer votre application, la console crée une politique appelée `kinesis-analytics-service-<application name>-<application region>` et un rôle appelé `kinesisanalytics-<application name>-<application region>`.

Ajoutez la section surlignée ci-dessous à la politique. Remplacez l'exemple d'ID de compte (*SOURCE01234567*) par l'ID du compte que vous utiliserez pour le flux source.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AssumeRoleInSourceAccount",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::SOURCE01234567:role/KA-Source-Stream-Role"
        },
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:SINK012345678:log-group:*"
            ]
```

```
        },
        {
            "Sid": "ListCloudwatchLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:SINK012345678:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutCloudwatchLogs",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:SINK012345678:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        }
    ]
}
```

2. Ouvrez le rôle `kinesis-analytics-MyApplication-us-west-2` et notez son Amazon Resource Name (ARN). Vous en aurez besoin pour la section suivante. Le rôle ARN ressemble à ceci :

```
arn:aws:iam::SINK012345678:role/service-role/kinesis-analytics-MyApplication-us-
west-2
```

Rôles et politiques du compte source

1. Créez une politique appelée `KA-Source-Stream-Policy` dans le compte source. Utilisez le JSON suivant pour la politique. Remplacez l'exemple de numéro de compte par le numéro du compte source.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": [
                "kinesis:DescribeStream",
                "kinesis:GetRecords",
                "kinesis:GetShardIterator",
                "kinesis:ListShards"
            ],
            "Resource":
               "arn:aws:kinesis:us-west-2:SOURCE123456784:stream/
SourceAccountExampleInputStream"
        }
    ]
}
```

2. Créez un rôle appelé `MF-Source-Stream-Role` dans le compte source. Procédez comme suit pour créer le rôle à l'aide du cas d'utilisation Managed Flink :

    1. Dans IAM Management Console, choisissez Créer un rôle.
    2. Sur la page Créer un rôle, choisissez Service AWS. Dans la liste des services, choisissez Kinesis.
    3. Sous Sélectionnez votre cas d'utilisation, choisissez service géré pour Apache Flink.
    4. Sélectionnez Next: Permissions (Étape suivante : autorisations).
    5. Ajoutez la politique d'autorisations `KA-Source-Stream-Policy` que vous avez créée à l'étape précédente. Choisissez Suivant : balises.
    6. Choisissez Suivant : vérification.
    7. Nommez le rôle `KA-Source-Stream-Role`. Votre application utilisera ce rôle pour accéder au flux source.

3. Ajoutez l'ARN `kinesis-analytics-MyApplication-us-west-2` du compte récepteur à la relation d'approbation du rôle `KA-Source-Stream-Role` dans le compte source :

    1. Ouvrez le `KA-Source-Stream-Role` dans la console IAM.
    2. Choisissez l'onglet Relations d'approbation.
    3. Choisissez Modifier la relation d'approbation.
    4. Utilisez le code suivant pour la relation d'approbation. Remplacez l'exemple d'ID de compte (*SINK012345678*) par l'ID de votre compte récepteur.

```
{
    "Version": "2012-10-17",
    "Statement": [
      {
        "Effect": "Allow",
        "Principal": {
          "AWS": "arn:aws:iam::SINK012345678:role/service-role/kinesis-analytics-
MyApplication-us-west-2"
        },
        "Action": "sts:AssumeRole"
      }
    ]
}
```

Mettre à jour le script Python

Dans cette section, vous allez mettre à jour le script Python qui génère des exemples de données afin d'utiliser le profil du compte source.

Mettez à jour le script `stock.py` avec les modifications mises en évidence ci-dessous.

```
import json
import boto3
import random
import datetime
import os

os.environ['AWS_PROFILE'] ='ka-source-stream-account-profile'
os.environ['AWS_DEFAULT_REGION'] = 'us-west-2'


kinesis = boto3.client('kinesis')
def getReferrer():
    data = {}
    now = datetime.datetime.now()
    str_now = now.isoformat()
    data['event_time'] = str_now
    data['ticker'] = random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV'])
    price = random.random() * 100
    data['price'] = round(price, 2)
    return data
```

```
while True:
        data = json.dumps(getReferrer())
        print(data)
        kinesis.put_record(
                StreamName="SourceAccountExampleInputStream",
                Data=data,
                PartitionKey="partitionkey")
```

Mettre à jour l'application Java

Dans cette section, vous allez mettre à jour le code de l'application Java pour qu'il assume le rôle de compte source lors de la lecture depuis le flux source.

Apportez les modifications suivantes au fichier `BasicStreamingJob.java`. Remplacez le numéro de compte source d'exemple (*SOURCE01234567*) par le numéro de votre compte source.

```
package com.amazonaws.services.managed-flink;

import com.amazonaws.services.managed-flink.runtime.KinesisAnalyticsRuntime;
import org.apache.flink.api.common.serialization.SimpleStringSchema;
import org.apache.flink.streaming.api.datastream.DataStream;
import org.apache.flink.streaming.api.environment.StreamExecutionEnvironment;
import org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer;
import org.apache.flink.streaming.connectors.kinesis.FlinkKinesisProducer;
import org.apache.flink.streaming.connectors.kinesis.config.ConsumerConfigConstants;
import org.apache.flink.streaming.connectors.kinesis.config.AWSConfigConstants;

import java.io.IOException;
import java.util.Map;
import java.util.Properties;

 /**
 * A basic Managed Service for Apache Flink for Java application with Kinesis data
 streams
 * as source and sink.
 */
public class BasicStreamingJob {
    private static final String region = "us-west-2";
    private static final String inputStreamName = "SourceAccountExampleInputStream";
    private static final String outputStreamName = ExampleOutputStream;
    private static final String roleArn = "arn:aws:iam::SOURCE01234567:role/KA-Source-
Stream-Role";
```

```
   private static final String roleSessionName = "ksassumedrolesession";

   private static DataStream<String>
createSourceFromStaticConfig(StreamExecutionEnvironment env) {
       Properties inputProperties = new Properties();
       inputProperties.setProperty(AWSConfigConstants.AWS_CREDENTIALS_PROVIDER,
"ASSUME_ROLE");
       inputProperties.setProperty(AWSConfigConstants.AWS_ROLE_ARN, roleArn);
       inputProperties.setProperty(AWSConfigConstants.AWS_ROLE_SESSION_NAME,
roleSessionName);
       inputProperties.setProperty(ConsumerConfigConstants.AWS_REGION, region);
       inputProperties.setProperty(ConsumerConfigConstants.STREAM_INITIAL_POSITION,
"LATEST");

       return env.addSource(new FlinkKinesisConsumer<>(inputStreamName, new
SimpleStringSchema(), inputProperties));
   }

   private static KinesisStreamsSink<String> createSinkFromStaticConfig() {
       Properties outputProperties = new Properties();
       outputProperties.setProperty(AWSConfigConstants.AWS_REGION, region);

       return KinesisStreamsSink.<String>builder()
               .setKinesisClientProperties(outputProperties)
               .setSerializationSchema(new SimpleStringSchema())
               .setStreamName(outputProperties.getProperty("OUTPUT_STREAM",
"ExampleOutputStream"))
               .setPartitionKeyGenerator(element ->
String.valueOf(element.hashCode()))
               .build();
   }

   public static void main(String[] args) throws Exception {
       // set up the streaming execution environment
       final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

       DataStream<String> input = createSourceFromStaticConfig(env);

       input.addSink(createSinkFromStaticConfig());

       env.execute("Flink Streaming Java API Skeleton");
   }
```

```
}
```

Créez, mettez à jour et exécutez l'application.

Procédez comme suit pour mettre à jour et exécuter l'application :

1. Créez à nouveau l'application en exécutant la commande suivante dans le répertoire contenant le fichier `pom.xml`.

   ```
   mvn package -Dflink.version=1.15.3
   ```

2. Supprimez le précédent fichier JAR de votre compartiment Amazon Simple Storage Service (Amazon S3), puis chargez le nouveau fichier `aws-kinesis-analytics-java-apps-1.0.jar` dans le compartiment S3.
3. Sur la page de l'application, dans la console du service géré pour Apache Flink, choisissez Configurer, Mettre à jour pour recharger le fichier JAR de l'application.
4. Exécutez le script `stock.py` pour envoyer les données au flux source.

   ```
   python stock.py
   ```

L'application lit désormais les données du flux Kinesis dans l'autre compte.

Vous pouvez vérifier que l'application fonctionne en vérifier la métrique `PutRecords.Bytes` du flux `ExampleOutputStream`. Si vous voyez de l'activité dans le flux de sortie, l'application fonctionne correctement.

Didacticiel : utilisation d'un truststore personnalisé avec Amazon MSK

API de source de données actuelles

Si vous utilisez les API de source de données actuelles, votre application peut tirer parti de l'utilitaire MSK Config Providers décrit ici. Cela permet à votre KafkaSource fonction d'accéder à votre keystore et à votre truststore pour le protocole TLS mutuel dans Amazon S3.

```
...
// define names of config providers:
builder.setProperty("config.providers", "secretsmanager,s3import");
```

```
// provide implementation classes for each provider:
builder.setProperty("config.providers.secretsmanager.class",
 "com.amazonaws.kafka.config.providers.SecretsManagerConfigProvider");
builder.setProperty("config.providers.s3import.class",
 "com.amazonaws.kafka.config.providers.S3ImportConfigProvider");

String region = appProperties.get(Helpers.S3_BUCKET_REGION_KEY).toString();
String keystoreS3Bucket = appProperties.get(Helpers.KEYSTORE_S3_BUCKET_KEY).toString();
String keystoreS3Path = appProperties.get(Helpers.KEYSTORE_S3_PATH_KEY).toString();
String truststoreS3Bucket =
 appProperties.get(Helpers.TRUSTSTORE_S3_BUCKET_KEY).toString();
String truststoreS3Path = appProperties.get(Helpers.TRUSTSTORE_S3_PATH_KEY).toString();
String keystorePassSecret =
 appProperties.get(Helpers.KEYSTORE_PASS_SECRET_KEY).toString();
String keystorePassSecretField =
 appProperties.get(Helpers.KEYSTORE_PASS_SECRET_FIELD_KEY).toString();

// region, etc..
builder.setProperty("config.providers.s3import.param.region", region);

// properties
builder.setProperty("ssl.truststore.location", "${s3import:" + region + ":" +
 truststoreS3Bucket + "/" + truststoreS3Path + "}");
builder.setProperty("ssl.keystore.type", "PKCS12");
builder.setProperty("ssl.keystore.location", "${s3import:" + region + ":" +
 keystoreS3Bucket + "/" + keystoreS3Path + "}");
builder.setProperty("ssl.keystore.password", "${secretsmanager:" + keystorePassSecret +
 ":" + keystorePassSecretField + "}");
builder.setProperty("ssl.key.password", "${secretsmanager:" + keystorePassSecret + ":"
 + keystorePassSecretField + "}");
...
```

Vous trouverez plus de détails et une procédure détaillée ici.

SourceFunction API héritées

Si vous utilisez les anciennes SourceFunction API, votre application utilisera des schémas de sérialisation et de désérialisation personnalisés qui remplacent la `open` méthode de chargement du truststore personnalisé. Cela met le truststore à la disposition de l'application après le redémarrage de l'application ou le remplacement des threads.

Le truststore personnalisé est récupéré et stocké à l'aide du code suivant :

```
public static void initializeKafkaTruststore() {
    ClassLoader classLoader = Thread.currentThread().getContextClassLoader();
    URL inputUrl = classLoader.getResource("kafka.client.truststore.jks");
    File dest = new File("/tmp/kafka.client.truststore.jks");

    try {
        FileUtils.copyURLToFile(inputUrl, dest);
    } catch (Exception ex) {
        throw new FlinkRuntimeException("Failed to initialize Kakfa truststore", ex);
    }
}
```

> **Note**
>
> Apache Flink nécessite que le truststore soit au format JKS.

> **Note**
>
> Pour configurer les prérequis requis pour cet exercice, commencez par terminer l'exercice Mise en route (DataStream API).

Le didacticiel suivant explique comment se connecter en toute sécurité (chiffrement en transit) à un cluster Kafka qui utilise des certificats de serveur émis par une autorité de certification (CA) personnalisée, privée ou même auto-hébergée.

Pour connecter un client Kafka en toute sécurité via TLS à un cluster Kafka, le client Kafka (comme l'exemple de l'application Flink) doit faire confiance à la chaîne d'approbation complète présentée par les certificats de serveur du cluster Kafka (de l'autorité de certification émettrice à l'autorité de certification de niveau racine). À titre d'exemple pour un truststore personnalisé, nous utiliserons un cluster Amazon MSK avec l'authentification mutuelle TLS (MTLS) activée. Cela implique que les nœuds du cluster MSK utilisent des certificats de serveur émis par une AWS Certificate Manager Private Certificate Authority (ACM Private CA) qui est privée à votre compte et votre région et qui n'est donc pas approuvée par le truststore par défaut de la machine virtuelle Java (JVM) exécutant l'application Flink.

> **Note**
>
> - Un keystore est utilisé pour stocker les clés privées et les certificats d'identité qu'une application doit présenter au serveur ou au client pour vérification.
> - Un truststore est utilisé pour stocker les certificats des autorités certifiées (CA) qui vérifient le certificat présenté par le serveur dans le cadre d'une connexion SSL.

Vous pouvez également utiliser la technique décrite dans ce didacticiel pour les interactions entre une application de service géré pour Apache Flink et d'autres sources Apache Kafka, telles que :

- Un cluster Apache Kafka personnalisé hébergé dans AWS (Amazon EC2 ou Amazon EKS)
- Un cluster Confluent Kafka hébergé dans AWS
- Un cluster Kafka sur site accessible via AWS Direct Connect ou un VPN

Ce didacticiel contient les sections suivantes :

- Créer un VPC avec un cluster Amazon MSK
- Créer un truststore personnalisé et l'appliquer à votre cluster
- Création du code de l'application
- Chargement du code Java Apache Flink
- Pour créer l'application
- Configuration de l'application
- Exécution de l'application
- Test de l'application

Créer un VPC avec un cluster Amazon MSK

Pour créer un exemple de VPC et de cluster Amazon MSK auquel accéder depuis une application de service géré pour Apache Flink, suivez le didacticiel Mise en route avec Amazon MSK.

Lorsque vous avez terminé le didacticiel, effectuez ce qui suit :

- À l'étape 3 : création d'une rubrique, répétez la commande `kafka-topics.sh --create` pour créer une rubrique de destination nommée `AWSKafkaTutorialTopicDestination` :

```
bin/kafka-topics.sh --create --bootstrap-server ZooKeeperConnectionString --
replication-factor 3 --partitions 1 --topic AWSKafkaTutorialTopicDestination
```

> **Note**
>
> Si la `kafka-topics.sh` commande renvoie une `ZooKeeperClientTimeoutException`, vérifiez que le groupe de sécurité du cluster Kafka dispose d'une règle entrante autorisant tout le trafic provenant de l'adresse IP privée de l'instance cliente.

- Enregistrez la liste des serveurs bootstrap de votre cluster. Vous pouvez obtenir la liste des serveurs bootstrap à l'aide de la commande suivante (remplacez-la *ClusterArn*par l'ARN de votre cluster MSK) :

```
aws kafka get-bootstrap-brokers --region us-west-2 --cluster-arn ClusterArn
{...
    "BootstrapBrokerStringTls": "b-2.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-1.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-3.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094"
}
```

- Lorsque vous suivez les étapes décrites dans de didacticiel et les didacticiels sur les prérequis, veillez à utiliser la région AWS sélectionnée dans le code, les commandes et les entrées de console.

Créer un truststore personnalisé et l’appliquer à votre cluster

Dans cette section, vous allez créer une autorité de certification (CA) personnalisée, l'utiliser pour générer un truststore personnalisé et l'appliquer à votre cluster MSK.

Pour créer et appliquer votre truststore personnalisé, suivez le didacticiel Authentification client figurant dans le guide du développeur Amazon Managed Streaming for Apache Kafka.

Création du code de l'application

Dans cette section, vous allez télécharger et compiler le fichier JAR de l'application.

Le code de l'application Java pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Installez le client Git si vous ne l'avez pas déjà fait. Pour de plus amples informations, veuillez consulter Installation de Git.
2. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Le code d'application est situé dans le fichier `amazon-kinesis-data-analytics-java-examples/CustomKeystore`. Vous pouvez examiner le code pour vous familiariser avec la structure du code du service géré pour Apache Flink.
4. Utilisez l'outil de ligne de commande Maven ou votre environnement de développement préféré pour créer le fichier JAR. Pour compiler le fichier JAR à l'aide de l'outil de ligne de commande Maven, saisissez ce qui suit :

```
mvn package -Dflink.version=1.15.3
```

En cas de succès de la génération, le fichier suivant est créé :

```
target/flink-app-1.0-SNAPSHOT.jar
```

> **Note**
>
> Le code source fourni repose sur les bibliothèques de Java 11.

Chargement du code Java Apache Flink

Dans cette section, vous allez charger votre code d'application dans le compartiment Amazon S3 que vous avez créé dans le didacticiel Mise en route (DataStream API).

> **Note**
>
> Si vous avez supprimé le compartiment Amazon S3 du didacticiel Mise en route, suivez à nouveau l'étape the section called "Chargement du code Java Apache Flink".

1. Dans la console Amazon S3, choisissez le <username>compartiment ka-app-code-, puis Upload.
2. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `flink-app-1.0-SNAPSHOT.jar` que vous avez créé à l'étape précédente.
3. Vous n'avez pas besoin de modifier les paramètres de l'objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Exécution, choisissez Apache Flink 1.15.2.
4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez un service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rôle : `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Configuration de l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **`ka-app-code-<username>`**.
   - Pour le chemin de l'objet Amazon S3, saisissez **`flink-app-1.0-SNAPSHOT.jar`**.
3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **`kinesis-analytics-MyApplication-us-west-2`**.

> **Note**
>
> Lorsque vous spécifiez des ressources d'application à l'aide de la console (comme les journaux ou un VPC), la console modifie le rôle d'exécution de votre application pour autoriser l'accès à ces ressources.

4. Sous Propriétés, sélectionnez Ajouter un groupe. Saisissez les propriétés suivantes :

ID du groupe	Clé	Valeur
`KafkaSource`	topic	AWSKafkaTutorialTopic
`KafkaSource`	bootstrap.servers	*La liste des serveurs bootstrap que vous avez enregistrée précédemment*
`KafkaSource`	security.protocol	SSL
`KafkaSource`	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
`KafkaSource`	ssl.truststore.password	changeit

> **Note**
>
> Le ssl.truststore.password du certificat par défaut est « changeit » ; vous n'avez pas besoin de modifier cette valeur si vous utilisez le certificat par défaut.

Choisissez à nouveau Ajouter un groupe. Saisissez les propriétés suivantes :

ID du groupe	Clé	Valeur
KafkaSink	topic	AWSKafkaTutorialTopicDestination
KafkaSink	bootstrap.servers	*La liste des serveurs bootstrap que vous avez enregistrée précédemment*
KafkaSink	security.protocol	SSL
KafkaSink	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSink	ssl.truststore.password	changeit
KafkaSink	transaction.timeout.ms	1 000

Le code de l'application lit les propriétés de l'application ci-dessus pour configurer la source et le récepteur utilisés pour interagir avec votre VPC et votre cluster Amazon MSK. Pour obtenir plus d'informations sur l'utilisation des propriétés, consultez Propriétés d'exécution.

5. Sous Instantanés, choisissez Désactiver. Cela facilitera la mise à jour de l'application sans charger de données d'état de l'application non valides.
6. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
7. Pour la CloudWatch journalisation, cochez la case Activer.

8. Dans la section Cloud privé virtuel (VPC), choisissez le VPC à associer à votre application. Choisissez les sous-réseaux et le groupe de sécurité associés à votre VPC que vous souhaitez que l'application utilise pour accéder aux ressources du VPC.
9. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation, le service géré pour Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`
>
> Ce flux de journaux est utilisé pour surveiller l'application.

Exécution de l'application

Le graphique des tâches Flink peut être visualisé en exécutant l'application, en ouvrant le tableau de bord Apache Flink et en choisissant la tâche Flink souhaitée.

Test de l'application

Dans cette section, vous allez écrire des enregistrements dans a rubrique source. L'application lit les enregistrements de la rubrique source et les écrit dans la rubrique de destination. Vous vérifiez que l'application fonctionne en écrivant des enregistrements dans la rubrique source et en lisant des enregistrements dans la rubrique de destination.

Pour écrire et lire des enregistrements issus des rubriques, suivez les étapes décrites à l'Étape 6 : produire et consommer des données du didacticiel Mise en route avec Amazon MSK.

Pour lire la rubrique de destination, utilisez le nom de la rubrique de destination au lieu de la rubrique source lors de votre deuxième connexion au cluster :

```
bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString --
consumer.config client.properties --topic AWSKafkaTutorialTopicDestination --from-
beginning
```

Si aucun enregistrement n'apparaît dans la rubrique de destination, consultez la sectionImpossible d'accéder aux ressources d'un VPC de la rubrique Résolution des problèmes.

Exemples Python

Les exemples suivants montrent comment créer des applications à l'aide de Python avec l'API de table Apache Flink.

Rubriques

- Exemple : création d'une fenêtre bascule dans Python
- Exemple : création d'une fenêtre défilante dans Python
- Exemple : envoyer des données de streaming vers Amazon S3 dans Python

Exemple : création d'une fenêtre bascule dans Python

Dans cet exercice, vous allez créer une application de service géré Python pour Apache Flink qui agrège les données à l'aide d'une fenêtre bascule.

> **Note**
>
> Pour configurer les prérequis requis pour cet exercice, commencez par terminer l'exercice Mise en route (Python).

Cette rubrique contient les sections suivantes :

- Création de ressources dépendantes
- Écriture d'exemples d'enregistrements dans le flux d'entrée
- Téléchargez et examinez le code de l'application
- Compression et chargement du code Python Apache Flink
- Création et exécution de l'application de service géré pour Apache Flink
- Nettoyage des ressources AWS

Création de ressources dépendantes

Avant de créer une application de service géré pour Apache Flink dans le cadre de cet exercice, vous commencez par créer les ressources dépendantes suivantes :

- Deux flux de données Kinesis (`ExampleInputStream` et `ExampleOutputStream`)
- Un compartiment Amazon S3 pour stocker le code de l'application (`ka-app-code-<username>`)

Vous pouvez créer les flux Kinesis et un compartiment S3 à l'aide de la console. Pour obtenir des instructions sur la création de ces ressources, consultez les rubriques suivantes :

- Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams. Nommez vos flux de données **`ExampleInputStream`** et **`ExampleOutputStream`**.
- Comment créer un compartiment S3 ? dans le Guide de l'utilisateur de la console Amazon Simple Storage Service. Donnez au compartiment Amazon S3 un nom unique au monde en ajoutant votre nom de connexion, tel que **`ka-app-code-<username>`**.

Écriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette section, vous utilisez un script Python pour écrire les exemples d'enregistrements dans le flux pour que l'application les traite.

> **Note**
>
> Cette section nécessite le kit AWS SDK for Python (Boto).

> **Note**
>
> Le script Python de cette section utilise l'interface AWS CLI. Vous devez configurer votre interface AWS CLI pour utiliser les informations d'identification de votre compte et la région par défaut. Pour configurer votre interface AWS CLI, saisissez :
>
> ```
> aws configure
> ```

1. Créez un fichier nommé `stock.py` avec le contenu suivant :

```
import datetime
import json
import random
```

```
    import boto3

    STREAM_NAME = "ExampleInputStream"


    def get_data():
        return {
            'event_time': datetime.datetime.now().isoformat(),
            'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
            'price': round(random.random() * 100, 2)}


    def generate(stream_name, kinesis_client):
        while True:
            data = get_data()
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey")


    if __name__ == '__main__':
        generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Exécutez le script `stock.py` :

```
$ python stock.py
```

Maintenez le script en cours d'exécution pendant que vous terminez le reste du didacticiel.

Téléchargez et examinez le code de l'application

Le code de l'application Python pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Installez le client Git si vous ne l'avez pas déjà fait. Pour de plus amples informations, veuillez consulter Installation de Git.
2. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/python/TumblingWindow`.

Le code d'application est situé dans le fichier `tumbling-windows.py`. Notez les informations suivantes à propos du code d'application :

- L'application utilise une source de table Kinesis pour lire à partir du flux source. L'extrait de code suivant appelle la fonction `create_table` permettant de créer la source de table Kinesis :

```
table_env.execute_sql(
        create_input_table(input_table_name, input_stream, input_region,
 stream_initpos)
    )
```

  La fonction `create_table` utilise une commande SQL pour créer une table soutenue par la source de streaming :

```
def create_input_table(table_name, stream_name, region, stream_initpos):
    return """ CREATE TABLE {0} (
                ticker VARCHAR(6),
                price DOUBLE,
                event_time TIMESTAMP(3),
                WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND
              )
              PARTITIONED BY (ticker)
              WITH (
                'connector' = 'kinesis',
                'stream' = '{1}',
                'aws.region' = '{2}',
                'scan.stream.initpos' = '{3}',
                'format' = 'json',
                'json.timestamp-format.standard' = 'ISO-8601'
              ) """.format(table_name, stream_name, region, stream_initpos)
```

- L'application utilise l'opérateur `Tumble` pour agréger les enregistrements dans une fenêtre bascule spécifiée et renvoyer les enregistrements agrégés sous forme d'objet de table :

```
tumbling_window_table = (
        input_table.window(
            Tumble.over("10.seconds").on("event_time").alias("ten_second_window")
        )
```

```
        .group_by("ticker, ten_second_window")
        .select("ticker, price.min as price, to_string(ten_second_window.end) as
event_time")
```

- L'application utilise le connecteur Kinesis Flink du `flink-sql-connector-kinesis-1.15.2.jar`.

Compression et chargement du code Python Apache Flink

Dans cette section, vous allez charger votre code d'application dans le compartiment Amazon S3 que vous avez créé dans la section Création de ressources dépendantes.

1. Utilisez votre application de compression préférée pour compresser les fichiers `tumbling-windows.py` et `flink-sql-connector-kinesis-1.15.2.jar`. Nommez l'archive `myapp.zip`.
2. Dans la console Amazon S3, choisissez le <username>compartiment ka-app-code-, puis Upload.
3. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `myapp.zip` que vous avez créé à l'étape précédente.
4. Vous n'avez pas besoin de modifier les paramètres de l'objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l'application de service géré pour Apache Flink

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.

- Pour Exécution, choisissez Apache Flink.

> **Note**
>
> Le service géré pour Apache Flink utilise Apache Flink version 1.15.2.

- Laissez le menu déroulant de la version sur Apache Flink version 1.15.2 (version recommandée).

4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-MyApplication-us-west-2`
> - Rôle : `kinesisanalytics-MyApplication-us-west-2`

Configuration de l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **`ka-app-code-<username>`**.
   - Pour le chemin de l'objet Amazon S3, saisissez **`myapp.zip`**.
3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **`kinesis-analytics-MyApplication-us-west-2`**.
4. Sous Propriétés, sélectionnez Ajouter un groupe.
5. Saisissez :

ID du groupe	Clé	Valeur
consumer.config.0	**input.stream.name**	**ExampleInputStream**
consumer.config.0	**aws.region**	**us-west-2**
consumer.config.0	**scan.stream.initpos**	**LATEST**

Choisissez Enregistrer.

6. Sous Propriétés, sélectionnez à nouveau Ajouter un groupe.
7. Saisissez :

ID du groupe	Clé	Valeur
producer.config.0	**output.stream.name**	**ExampleOutputStream**
producer.config.0	**aws.region**	**us-west-2**
producer.config.0	**shard.count**	**1**

8. Sous Propriétés, sélectionnez à nouveau Ajouter un groupe. Pour ID du groupe, saisissez **kinesis.analytics.flink.run.options**. Ce groupe de propriétés spécial indique à votre application où se trouvent ses ressources de code. Pour plus d'informations, consultez Spécification de vos fichiers de code.
9. Saisissez :

ID du groupe	Clé	Valeur
kinesis.analytics.flink.run.options	**python**	**tumbling-windows.py**
kinesis.analytics.flink.run.options	**jarfile**	**flink-sql-connector-kinesis-1.15.2.jar**

10. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
11. Pour la CloudWatch journalisation, cochez la case Activer.
12. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation, le service géré pour Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`
>
> Ce flux de journaux est utilisé pour surveiller l'application. Il ne s'agit pas du même flux de journaux que celui utilisé par l'application pour envoyer les résultats.

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder aux flux de données Kinesis.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **`kinesis-analytics-service-MyApplication-us-west-2`** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*`012345678901`*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
```

```
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/myapp.zip"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
```

```
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Exécution de l'application

Le graphique des tâches Flink peut être visualisé en exécutant l'application, en ouvrant le tableau de bord Apache Flink et en choisissant la tâche Flink souhaitée.

Vous pouvez vérifier les métriques du service géré pour Apache Flink sur la CloudWatch console pour vérifier que l'application fonctionne.

Nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Fenêtre bascule.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer vos flux de données Kinesis
- Supprimer votre objet et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. dans le panneau Managed Service for Apache Flink, sélectionnez MyApplication.
3. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer vos flux de données Kinesis

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.

2. Dans le panneau Kinesis Data Streams, ExampleInputStreamsélectionnez.
3. Sur la ExampleInputStreampage, choisissez Supprimer Kinesis Stream, puis confirmez la suppression.
4. Sur la page Kinesis Streams, choisissez le ExampleOutputStream, choisissez Actions, choisissez Supprimer, puis confirmez la suppression.

Supprimer votre objet et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code -. <username>
3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.
8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Exemple : création d'une fenêtre défilante dans Python

> **Note**
>
> Pour configurer les prérequis requis pour cet exercice, commencez par terminer l'exercice Mise en route (Python).

Cette rubrique contient les sections suivantes :

- Création de ressources dépendantes
- Écriture d'exemples d'enregistrements dans le flux d'entrée
- Téléchargez et examinez le code de l'application
- Compression et chargement du code Python Apache Flink
- Création et exécution de l'application de service géré pour Apache Flink
- Nettoyage des ressources AWS

Création de ressources dépendantes

Avant de créer une application de service géré pour Apache Flink dans le cadre de cet exercice, vous commencez par créer les ressources dépendantes suivantes :

- Deux flux de données Kinesis (`ExampleInputStream` et `ExampleOutputStream`)
- Un compartiment Amazon S3 pour stocker le code de l'application (`ka-app-code-<username>`)

Vous pouvez créer les flux Kinesis et un compartiment S3 à l'aide de la console. Pour obtenir des instructions sur la création de ces ressources, consultez les rubriques suivantes :

- Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams. Nommez vos flux de données **`ExampleInputStream`** et **`ExampleOutputStream`**.
- Comment créer un compartiment S3 ? dans le Guide de l'utilisateur de la console Amazon Simple Storage Service. Donnez au compartiment Amazon S3 un nom unique au monde en ajoutant votre nom de connexion, tel que **`ka-app-code-<username>`**.

Écriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette section, vous utilisez un script Python pour écrire les exemples d'enregistrements dans le flux pour que l'application les traite.

> **Note**
>
> Cette section nécessite le kit AWS SDK for Python (Boto).

> **Note**
>
> Le script Python de cette section utilise l'interface AWS CLI. Vous devez configurer votre interface AWS CLI pour utiliser les informations d'identification de votre compte et la région par défaut. Pour configurer votre interface AWS CLI, saisissez :
>
> ```
> aws configure
> ```

1. Créez un fichier nommé `stock.py` avec le contenu suivant :

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
```

```
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey")


    if __name__ == '__main__':
        generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Exécutez le script `stock.py` :

```
$ python stock.py
```

   Maintenez le script en cours d'exécution pendant que vous terminez le reste du didacticiel.

Téléchargez et examinez le code de l'application

Le code de l'application Python pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Installez le client Git si vous ne l'avez pas déjà fait. Pour de plus amples informations, veuillez consulter Installation de Git.
2. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/>amazon-kinesis-data-analytics-java-
examples
```

3. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/python/SlidingWindow`.

Le code d'application est situé dans le fichier `sliding-windows.py`. Notez les informations suivantes à propos du code d'application :

- L'application utilise une source de table Kinesis pour lire à partir du flux source. L'extrait de code suivant appelle la fonction `create_input_table` permettant de créer la source de table Kinesis :

```
table_env.execute_sql(
        create_input_table(input_table_name, input_stream, input_region,
 stream_initpos)
```

```
    )
```

La fonction `create_input_table` utilise une commande SQL pour créer une table soutenue par la source de streaming :

```
def create_input_table(table_name, stream_name, region, stream_initpos):
    return """ CREATE TABLE {0} (
                ticker VARCHAR(6),
                price DOUBLE,
                event_time TIMESTAMP(3),
                WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND
              )
              PARTITIONED BY (ticker)
              WITH (
                'connector' = 'kinesis',
                'stream' = '{1}',
                'aws.region' = '{2}',
                'scan.stream.initpos' = '{3}',
                'format' = 'json',
                'json.timestamp-format.standard' = 'ISO-8601'
              ) """.format(table_name, stream_name, region, stream_initpos)
 }
```

- L'application utilise l'opérateur `Slide` pour agréger les enregistrements dans une fenêtre défilante spécifiée et renvoyer les enregistrements agrégés sous forme d'objet de table :

```
sliding_window_table = (
        input_table
            .window(
                Slide.over("10.seconds")
                .every("5.seconds")
                .on("event_time")
                .alias("ten_second_window")
            )
            .group_by("ticker, ten_second_window")
            .select("ticker, price.min as price, to_string(ten_second_window.end) as
 event_time")
    )
```

- L'application utilise le connecteur Kinesis Flink, issu du fichier -1.15.2.jar. flink-sql-connector-kinesis

Compression et chargement du code Python Apache Flink

Dans cette section, vous allez charger votre code d'application dans le compartiment Amazon S3 que vous avez créé dans la section Création de ressources dépendantes.

Cette section décrit comment empaqueter votre application Python.

1. Utilisez votre application de compression préférée pour compresser les fichiers `sliding-windows.py` et `flink-sql-connector-kinesis-1.15.2.jar`. Nommez l'archive `myapp.zip`.
2. Dans la console Amazon S3, choisissez le <username>compartiment ka-app-code-, puis Upload.
3. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `myapp.zip` que vous avez créé à l'étape précédente.
4. Vous n'avez pas besoin de modifier les paramètres de l'objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l'application de service géré pour Apache Flink

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Exécution, choisissez Apache Flink.

> **Note**
>
> Le service géré pour Apache Flink utilise Apache Flink version 1.15.2.

- Laissez le menu déroulant de la version sur Apache Flink version 1.15.2 (version recommandée).

4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **kinesis-analytics-MyApplication-us-west-2**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-MyApplication-us-west-2`
> - Rôle : `kinesisanalytics-MyApplication-us-west-2`

Configuration de l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **ka-app-code-*<username>***.
   - Pour le chemin de l'objet Amazon S3, saisissez **myapp.zip**.
3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Sous Propriétés, sélectionnez Ajouter un groupe.
5. Entrez les valeurs et propriétés d'application suivantes :

ID du groupe	Clé	Valeur
consumer.config.0	**input.stream.name**	**ExampleInputStream**
consumer.config.0	**aws.region**	**us-west-2**

ID du groupe	Clé	Valeur
consumer.config.0	**scan.stream.initpos**	**LATEST**

Choisissez Enregistrer.

6. Sous Propriétés, sélectionnez à nouveau Ajouter un groupe.
7. Entrez les valeurs et propriétés d'application suivantes :

ID du groupe	Clé	Valeur
producer.config.0	**output.stream.name**	**ExampleOutputStream**
producer.config.0	**aws.region**	**us-west-2**
producer.config.0	**shard.count**	**1**

8. Sous Propriétés, sélectionnez à nouveau Ajouter un groupe. Pour ID du groupe, saisissez **kinesis.analytics.flink.run.options**. Ce groupe de propriétés spécial indique à votre application où se trouvent ses ressources de code. Pour plus d'informations, consultez Spécification de vos fichiers de code.
9. Entrez les valeurs et propriétés d'application suivantes :

ID du groupe	Clé	Valeur
kinesis.analytics.flink.run.options	**python**	**sliding-windows.py**
kinesis.analytics.flink.run.options	**jarfile**	**flink-sql-connector-kinesis_1.15.2.jar**

10. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
11. Pour la CloudWatch journalisation, cochez la case Activer.
12. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation, le service géré pour Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`
>
> Ce flux de journaux est utilisé pour surveiller l'application. Il ne s'agit pas du même flux de journaux que celui utilisé par l'application pour envoyer les résultats.

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder aux flux de données Kinesis.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **`kinesis-analytics-service-MyApplication-us-west-2`** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*`012345678901`*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/myapp.zip"
            ]
```

```
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Exécution de l'application

Le graphique des tâches Flink peut être visualisé en exécutant l'application, en ouvrant le tableau de bord Apache Flink et en choisissant la tâche Flink souhaitée.

Vous pouvez vérifier les métriques du service géré pour Apache Flink sur la CloudWatch console pour vérifier que l'application fonctionne.

Nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Fenêtre défilante.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer vos flux de données Kinesis
- Supprimer votre objet et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. dans le panneau Managed Service for Apache Flink, sélectionnez MyApplication.
3. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer vos flux de données Kinesis

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Kinesis Data Streams, ExampleInputStreamsélectionnez.
3. Sur la ExampleInputStreampage, choisissez Supprimer Kinesis Stream, puis confirmez la suppression.
4. Sur la page Kinesis Streams, choisissez le ExampleOutputStream, choisissez Actions, choisissez Supprimer, puis confirmez la suppression.

Supprimer votre objet et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code -. <username>
3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.
8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Exemple : envoyer des données de streaming vers Amazon S3 dans Python

Dans cet exercice, vous allez créer une application de service géré Python pour Apache Flink qui diffuse des données vers un récepteur Amazon Simple Storage Service.

> **Note**
>
> Pour configurer les prérequis requis pour cet exercice, commencez par terminer l'exercice Mise en route (Python).

Cette rubrique contient les sections suivantes :

- Création de ressources dépendantes
- Écriture d'exemples d'enregistrements dans le flux d'entrée
- Téléchargez et examinez le code de l'application
- Compression et chargement du code Python Apache Flink
- Création et exécution de l’application de service géré pour Apache Flink
- Nettoyage des ressources AWS

Création de ressources dépendantes

Avant de créer une application de service géré pour Apache Flink dans le cadre de cet exercice, vous commencez par créer les ressources dépendantes suivantes :

- Un flux de données Kinesis (`ExampleInputStream`)
- Un compartiment Amazon S3 pour stocker le code et la sortie de l'application (`ka-app-code-<username>`)

> **Note**
>
> Le service géré pour Apache Flink ne peut pas écrire de données sur Amazon S3 lorsque le chiffrement côté serveur est activé sur le service géré pour Apache Flink.

Vous pouvez créer un flu Kinesis et un compartiment S3 à l'aide de la console. Pour obtenir des instructions sur la création de ces ressources, consultez les rubriques suivantes :

- Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams. Attribuez un nom à votre flux de données **`ExampleInputStream`**.
- Comment créer un compartiment S3 ? dans le Guide de l'utilisateur de la console Amazon Simple Storage Service. Donnez au compartiment Amazon S3 un nom unique au monde en ajoutant votre nom de connexion, tel que **`ka-app-code-<username>`**.

Écriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette section, vous utilisez un script Python pour écrire les exemples d'enregistrements dans le flux pour que l'application les traite.

> **Note**
>
> Cette section nécessite le kit AWS SDK for Python (Boto).

> **Note**
>
> Le script Python de cette section utilise l'interface AWS CLI. Vous devez configurer votre interface AWS CLI pour utiliser les informations d'identification de votre compte et la région par défaut. Pour configurer votre interface AWS CLI, saisissez :
>
> ```
> aws configure
> ```

1. Créez un fichier nommé `stock.py` avec le contenu suivant :

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
```

```
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey")


    if __name__ == '__main__':
        generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Exécutez le script `stock.py` :

```
$ python stock.py
```

   Maintenez le script en cours d'exécution pendant que vous terminez le reste du didacticiel.

Téléchargez et examinez le code de l'application

Le code de l'application Python pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Installez le client Git si vous ne l'avez pas déjà fait. Pour de plus amples informations, veuillez consulter Installation de Git.
2. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/python/S3Sink`.

Le code d'application est situé dans le fichier `streaming-file-sink.py`. Notez les informations suivantes à propos du code d'application :

- L'application utilise une source de table Kinesis pour lire à partir du flux source. L'extrait de code suivant appelle la fonction `create_source_table` permettant de créer la source de table Kinesis :

```
table_env.execute_sql(
        create_source_table(input_table_name, input_stream, input_region,
 stream_initpos)
```

```
    )
```

La fonction `create_source_table` utilise une commande SQL pour créer une table soutenue par la source de streaming

```
import datetime
    import json
    import random
    import boto3

    STREAM_NAME = "ExampleInputStream"


    def get_data():
        return {
            'event_time': datetime.datetime.now().isoformat(),
            'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
            'price': round(random.random() * 100, 2)}


    def generate(stream_name, kinesis_client):
        while True:
            data = get_data()
            print(data)
            kinesis_client.put_record(
                StreamName=stream_name,
                Data=json.dumps(data),
                PartitionKey="partitionkey")


    if __name__ == '__main__':
        generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

- L'application utilise le connecteur `filesystem` pour envoyer des enregistrements vers un compartiment Amazon S3 :

```
def create_sink_table(table_name, bucket_name):
    return """ CREATE TABLE {0} (
                ticker VARCHAR(6),
                price DOUBLE,
                event_time VARCHAR(64)
              )
```

```
            PARTITIONED BY (ticker)
            WITH (
                'connector'='filesystem',
                'path'='s3a://{1}/',
                'format'='json',
                'sink.partition-commit.policy.kind'='success-file',
                'sink.partition-commit.delay' = '1 min'
            ) """.format(table_name, bucket_name)
```

- L'application utilise le connecteur Kinesis Flink, issu du fichier -1.15.2.jar. flink-sql-connector-kinesis

Compression et chargement du code Python Apache Flink

Dans cette section, vous allez charger votre code d'application dans le compartiment Amazon S3 que vous avez créé dans la section Création de ressources dépendantes.

1. Utilisez votre application de compression préférée pour compresser les fichiers `streaming-file-sink.py` et flink-sql-connector-kinesis-1.15.2.jar. Nommez l'archive `myapp.zip`.
2. Dans la console Amazon S3, choisissez le <username>compartiment ka-app-code-, puis Upload.
3. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `myapp.zip` que vous avez créé à l'étape précédente.
4. Vous n'avez pas besoin de modifier les paramètres de l'objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l'application de service géré pour Apache Flink

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :

- Pour Nom de l'application, saisissez **`MyApplication`**.
- Pour Exécution, choisissez Apache Flink.

> **Note**
>
> Le service géré pour Apache Flink utilise Apache Flink version 1.15.2.

- Laissez le menu déroulant de la version sur Apache Flink version 1.15.2 (version recommandée).

4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rôle : `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Configuration de l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:

- Pour le compartiment Amazon S3, saisissez **`ka-app-code-`*`<username>`***.
- Pour le chemin de l'objet Amazon S3, saisissez **`myapp.zip`**.

3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **`kinesis-analytics-MyApplication-us-west-2`**.
4. Sous Propriétés, sélectionnez Ajouter un groupe.

5. Entrez les valeurs et propriétés d'application suivantes :

ID du groupe	Clé	Valeur
consumer.config.0	**input.stream.name**	**ExampleInputStream**
consumer.config.0	**aws.region**	**us-west-2**
consumer.config.0	**scan.stream.initpos**	**LATEST**

Choisissez Enregistrer.

6. Sous Propriétés, sélectionnez à nouveau Ajouter un groupe. Pour ID du groupe, saisissez **kinesis.analytics.flink.run.options**. Ce groupe de propriétés spécial indique à votre application où se trouvent ses ressources de code. Pour plus d'informations, consultez Spécification de vos fichiers de code.
7. Entrez les valeurs et propriétés d'application suivantes :

ID du groupe	Clé	Valeur
kinesis.analytics.flink.run.options	**python**	**streaming-file-sink.py**
kinesis.analytics.flink.run.options	**jarfile**	**S3Sink/lib/flink-sql-connector-kinesis-1.15.2.jar**

8. Sous Propriétés, sélectionnez à nouveau Ajouter un groupe. Pour ID du groupe, saisissez **sink.config.0**. Ce groupe de propriétés spécial indique à votre application où se trouvent ses ressources de code. Pour plus d'informations, consultez Spécification de vos fichiers de code.
9. Saisissez les propriétés et valeurs d'application suivantes : (remplacez *bucket-name* par le nom réel de votre compartiment Amazon S3.)

ID du groupe	Clé	Valeur
sink.config.0	**output.bucket.name**	***bucket-name***

10. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
11. Pour la CloudWatch journalisation, cochez la case Activer.
12. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation, le service géré pour Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`
>
> Ce flux de journaux est utilisé pour surveiller l'application. Il ne s'agit pas du même flux de journaux que celui utilisé par l'application pour envoyer les résultats.

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder aux flux de données Kinesis.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **`kinesis-analytics-service-MyApplication-us-west-2`** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*`012345678901`*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
```

```
                "logs:DescribeLogGroups",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*",
                "arn:aws:s3:::ka-app-code-<username>/myapp.zip"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": "logs:DescribeLogStreams",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": "logs:PutLogEvents",
            "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
        },
        {
            "Sid": "ListCloudwatchLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteObjects",
            "Effect": "Allow",
            "Action": [
                "s3:Abort*",
```

```
                "s3:DeleteObject*",
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-<username>",
                "arn:aws:s3:::ka-app-code-<username>/*"
            ]
        }
    ]
}
```

Exécution de l'application

Le graphique des tâches Flink peut être visualisé en exécutant l'application, en ouvrant le tableau de bord Apache Flink et en choisissant la tâche Flink souhaitée.

Vous pouvez vérifier les métriques du service géré pour Apache Flink sur la CloudWatch console pour vérifier que l'application fonctionne.

Nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Fenêtre défilante.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer votre flux de données Kinesis
- Supprimer vos objets et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink

2. dans le panneau Managed Service for Apache Flink, sélectionnez MyApplication.
3. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer votre flux de données Kinesis

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Kinesis Data Streams, ExampleInputStreamsélectionnez.
3. Sur la ExampleInputStreampage, choisissez Supprimer Kinesis Stream, puis confirmez la suppression.

Supprimer vos objets et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code -. <username>
3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l’adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.
8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Exemples Scala :

Les exemples suivants montrent comment créer des applications à l'aide de Scala avec Apache Flink.

Rubriques

- Exemple : création d'une fenêtre bascule dans Scala
- Exemple : création d'une fenêtre défilante dans Scala
- Exemple : envoyer des données de streaming vers Amazon S3 dans Scala

Exemple : création d'une fenêtre bascule dans Scala

> **Note**
>
> À partir de la version 1.15, Flink n'utilise plus Scala. Les applications peuvent désormais utiliser l'API Java depuis n'importe quelle version de Scala. Flink utilise toujours Scala dans quelques composants clés en interne, mais n'expose pas Scala dans le chargeur de classes du code de l'utilisateur. Pour cette raison, les utilisateurs doivent ajouter des dépendances Scala dans leurs archives JAR.
> Pour plus d'informations sur les modifications apportées à Scala dans Flink 1.15, consultez Scala Free in One Fifteen.

Dans cet exercice, vous allez créer une application de streaming simple qui utilise Scala 3.2.0 et l'API Java de Flink. DataStream L'application lit les données du flux Kinesis, les agrège à l'aide de fenêtres défilantes et écrit les résultats pour générer le flux Kinesis.

> **Note**
>
> Pour configurer les prérequis requis pour cet exercice, commencez par terminer l'exercice Mise en route (Scala).

Cette rubrique contient les sections suivantes :

- Téléchargez et examinez le code de l'application
- Compiler et charger le code d'application

- Création et exécution de l'application (console)
- Création et exécution de l'application (CLI)
- Mise à jour du code de l'application
- Nettoyage des ressources AWS

Téléchargez et examinez le code de l'application

Le code de l'application Python pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Installez le client Git si vous ne l'avez pas déjà fait. Pour de plus amples informations, veuillez consulter Installation de Git.
2. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/scala/TumblingWindow`.

Notez les informations suivantes à propos du code d'application :

- Un fichier `build.sbt` contient des informations sur la configuration et les dépendances de l'application, y compris les bibliothèques du service géré pour Apache Flink.
- Le fichier `BasicStreamingJob.scala` contient la méthode principale qui définit la fonctionnalité de l'application.
- L'application utilise une source Kinesis pour lire à partir du flux source. L'extrait de code suivant crée la source Kinesis :

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
 defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

L'application utilise également un récepteur Kinesis pour écrire dans le flux de résultats. L'extrait de code suivant crée le récepteur Kinesis :

```
private def createSink: KinesisStreamsSink[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val outputProperties = applicationProperties.get("ProducerConfigProperties")

  KinesisStreamsSink.builder[String]
    .setKinesisClientProperties(outputProperties)
    .setSerializationSchema(new SimpleStringSchema)
    .setStreamName(outputProperties.getProperty(streamNameKey,
 defaultOutputStreamName))
    .setPartitionKeyGenerator((element: String) => String.valueOf(element.hashCode))
    .build
}
```

- L'application utilise l'opérateur de fenêtre pour déterminer le nombre de valeurs de chaque symbole boursier sur une fenêtre bascule de 5 secondes. Le code suivant crée l'opérateur et envoie les données agrégées vers un nouveau récepteur Kinesis Data Streams :

```
environment.addSource(createSource)
  .map { value =>
    val jsonNode = jsonParser.readValue(value, classOf[JsonNode])
    new Tuple2[String, Int](jsonNode.get("ticker").toString, 1)
  }
  .returns(Types.TUPLE(Types.STRING, Types.INT))
  .keyBy(v => v.f0) // Logically partition the stream for each ticker
  .window(TumblingProcessingTimeWindows.of(Time.seconds(10)))
  .sum(1) // Sum the number of tickers per partition
  .map { value => value.f0 + "," + value.f1.toString + "\n" }
  .sinkTo(createSink)
```

- L'application crée des connecteurs source et récepteur pour accéder à des ressources externes à l'aide d'un StreamExecutionEnvironment objet.
- L'application crée les connecteurs source et récepteur à l'aide de propriétés d'application dynamiques. Les propriétés d'exécution de l'application sont lues pour configurer les connecteurs. Pour de plus amples informations sur les propriétés d'exécution, veuillez consulter Runtime Properties.

Compiler et charger le code d'application

Dans cette section, vous compilez et charger votre code d'application dans un compartiment Amazon S3.

Compilation du code d'application

Utilisez l'outil de construction SBT pour créer le code Scala de l'application. Pour installer SBT, voir Install sbt with cs setup. Vous devez également installer le kit de développement Java (JDK). Consultez Prerequisites for Completing the Exercises.

1. Pour utiliser votre code d'application, vous le compilez et l'intégrez dans un fichier JAR. Vous pouvez compiler et empaqueter votre code avec SBT :

```
sbt assembly
```

2. Si la compilation de l'application aboutit, le fichier suivant est créé :

```
target/scala-3.2.0/tumbling-window-scala-1.0.jar
```

Chargement du code Scala Apache Flink

Dans cette section, vous allez créer un compartiment Amazon S3 et charger votre code d'application.

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez Créer un compartiment.
3. Saisissez `ka-app-code-<username>` dans le champ Nom du compartiment. Ajoutez un suffixe au nom du compartiment, par exemple votre nom d'utilisateur, pour qu'il soit unique. Choisissez Suivant.
4. À l'étape Configurer les options, conservez les paramètres, puis choisissez Suivant.
5. À l'étape Définir des autorisations, conservez les paramètres, puis choisissez Suivant.
6. Choisissez Créer un compartiment.
7. Ouvrez le compartiment `ka-app-code-<username>`, puis choisissez Charger.
8. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `tumbling-window-scala-1.0.jar` que vous avez créé à l'étape précédente.
9. Vous n'avez pas besoin de modifier les paramètres de l’objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l'application (console)

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Description, saisissez **`My Scala test app`**.
   - Pour Exécution, choisissez Apache Flink.
   - Laissez la version sur Apache Flink 1.15.2 (version recommandée).
4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rôle : `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Configuration de l'application

Procédez comme suit pour configurer l'application.

Pour configurer l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **ka-app-code-*<username>***.
   - Pour le chemin de l'objet Amazon S3, saisissez **tumbling-window-scala-1.0.jar**.
3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Sous Propriétés, sélectionnez Ajouter un groupe.
5. Saisissez :

ID du groupe	Clé	Valeur
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

   Choisissez Enregistrer.
6. Sous Propriétés, sélectionnez à nouveau Ajouter un groupe.
7. Saisissez :

ID du groupe	Clé	Valeur
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**

ID du groupe	Clé	Valeur
ProducerConfigProperties	**aws.region**	**us-west-2**

8. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
9. Pour la CloudWatch journalisation, cochez la case Activer.
10. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation Amazon, Managed Service for Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder au compartiment Amazon S3.

Pour modifier la politique IAM afin d'ajouter des autorisations au compartiment S3

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **kinesis-analytics-service-MyApplication-us-west-2** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*012345678901*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
```

```
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/tumbling-window-scala-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
```

```
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Exécution de l'application

Le graphique des tâches Flink peut être visualisé en exécutant l'application, en ouvrant le tableau de bord Apache Flink et en choisissant la tâche Flink souhaitée.

Arrêt de l'application

Pour arrêter l'application, sur la MyApplicationpage, choisissez Arrêter. Confirmez l'action.

Création et exécution de l'application (CLI)

Dans cette section, vous allez utiliser l’interface AWS Command Line Interface pour créer et exécuter l'application de service géré pour Apache Flink. Utilisez la commande AWS CLI kinesisanalyticsv2 pour créer et interagir avec les applications de service géré pour Apache Flink.

Créer une stratégie d’autorisations

> **Note**
>
> Vous devez créer une stratégie d'autorisations et un rôle pour votre application. Si vous ne créez pas ces ressources IAM, votre application ne peut pas accéder à ses flux de données et de journaux.

Vous commencez par créer une stratégie d'autorisations avec deux instructions : une qui accorde des autorisations pour l'action de lecture sur le flux source et une autre qui accorde des autorisations pour

les actions d'écriture sur le flux récepteur. Vous attachez ensuite la politique à un rôle IAM (que vous allez créer dans la section suivante). Ainsi, lorsque le service géré pour Apache Flink assume le rôle, le service dispose des autorisations nécessaires pour lire à partir du flux source et écrire dans le flux récepteur.

Utilisez le code suivant pour créer la stratégie d'autorisations `AKReadSourceStreamWriteSinkStream`. Remplacez **`username`** par le nom d'utilisateur que vous avez utilisé pour créer le compartiment Amazon S3 pour stocker le code d'application. Remplacez l'ID de compte dans Amazon Resource Names (ARN) (**`(012345678901)`**) par votre ID de compte. Le rôle d'exécution du service **`MF-stream-rw-role`** doit être adapté au rôle spécifique du client.

```
{
    "ApplicationName": "tumbling_window",
    "ApplicationDescription": "Scala tumbling window application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "tumbling-window-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
```

```
                }
            }
        ]
      }
    },
    "CloudWatchLoggingOptions": [
      {
         "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
      }
   ]
}
```

Pour step-by-step obtenir des instructions sur la création d'une politique d'autorisations, voir Tutoriel : créer et joindre votre première politique gérée par le client dans le guide de l'utilisateur IAM.

Créer un rôle IAM

Dans cette section, vous créez un rôle IAM que l'application de service géré pour Apache Flink peut assumer pour lire un flux source et écrire dans le flux récepteur.

Le service géré pour Apache Flink ne peut pas accéder à votre flux sans autorisation. Vous utilisez un rôle IAM pour accorder ces autorisations. Deux politiques sont attachées à chaque rôle IAM. La politique d'approbation accorde au service géré pour Apache Flink l'autorisation d'assumer le rôle, et la politique d'autorisation détermine ce que le service géré pour Apache Flink peut faire après avoir assumé le rôle.

Vous attachez la stratégie d'autorisations que vous avez créée dans la section précédente à ce rôle.

Pour créer un rôle IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, choisissez Rôles, puis Créer un Rôle.
3. Sous Sélectionner le type d'identité approuvée, choisissez Service AWS.
4. Sous Choisir le service qui utilisera ce rôle, choisissez EC2.
5. Sous Sélectionnez votre cas d'utilisation, choisissez service géré pour Apache Flink.
6. Sélectionnez Next: Permissions (Étape suivante : autorisations).
7. Dans la page Attacher des stratégies d'autorisations, choisissez Suivant : vérification. Vous attachez des stratégies d'autorisations après avoir créé le rôle.

8. Sur la page Créer un rôle, saisissez **`MF-stream-rw-role`** pour le Nom du rôle. Sélectionnez Créer un rôle.

   Vous venez de créer un nouveau rôle IAM appelé `MF-stream-rw-role`. Ensuite, vous mettez à jour les stratégies d'approbation et d'autorisation pour le rôle.

9. Attachez la politique d'autorisation au rôle.

   > **Note**
   >
   > Dans le cadre de cet exercice, le service géré pour Apache Flink assume ce rôle à la fois pour la lecture des données à partir d'un flux de données Kinesis (source) et pour l'écriture des résultats dans un autre flux de données Kinesis. Vous attachez donc la stratégie que vous avez créée à l'étape précédente, Créer une stratégie d'autorisations.

   a. Sur la page Récapitulatif, choisissez l'onglet Autorisations.

   b. Choisissez Attacher des stratégies.

   c. Dans la zone de recherche, saisissez **`AKReadSourceStreamWriteSinkStream`** (la stratégie que vous avez créée dans la section précédente).

   d. Sélectionnez Actions, puis Attacher une stratégie.

Vous avez maintenant créé le rôle d'exécution de service que votre application utilise pour accéder aux ressources. Notez l'ARN du nouveau rôle.

Pour step-by-step obtenir des instructions sur la création d'un rôle, consultez la section Création d'un rôle IAM (console) dans le guide de l'utilisateur IAM.

Pour créer l'application

Copiez le code JSON suivant dans un fichier nommé `create_request.json`. Remplacez l'exemple d'ARN du rôle par l'ARN du rôle que vous avez créé précédemment. Remplacez le suffixe de l'ARN du compartiment (nom d'utilisateur) par le suffixe que vous avez choisi dans la section précédente. Remplacez l'exemple d'ID de compte (012345678901) dans le rôle d'exécution de service par votre ID de compte. Le `ServiceExecutionRole` doit inclure le rôle d'utilisateur IAM que vous avez créé dans la section précédente.

```
"ApplicationName": "tumbling_window",
    "ApplicationDescription": "Scala getting started application",
```

```
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "tumbling-window-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
            }
         ]
      }
    },
    "CloudWatchLoggingOptions": [
      {
         "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
      }
   ]
}
```

Exécutez le CreateApplicationavec la requête suivante pour créer l'application :

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

L'application est maintenant créée. Vous démarrez l'application dans l'étape suivante.

Démarrage de l'application

Dans cette section, vous utilisez l'action StartApplication pour démarrer l'application.

Pour démarrer l'application

1. Copiez le code JSON suivant dans un fichier nommé `start_request.json`.

```
{
    "ApplicationName": "tumbling_window",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. Exécutez l'action `StartApplication` avec la demande précédente pour démarrer l'application :

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

L'application est maintenant en cours d'exécution. Vous pouvez consulter les métriques du service géré pour Apache Flink sur la CloudWatch console Amazon pour vérifier que l'application fonctionne.

Arrêt de l'application

Dans cette section, vous allez utiliser l'action StopApplication pour arrêter l'application.

Pour arrêter l'application

1. Copiez le code JSON suivant dans un fichier nommé `stop_request.json`.

```
{
   "ApplicationName": "tumbling_window"
}
```

2. Exécutez l'action `StopApplication` avec la demande précédente pour arrêter l'application :

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

L'application est maintenant arrêtée.

Ajouter une option de CloudWatch journalisation

Vous pouvez utiliser le AWS CLI pour ajouter un flux de CloudWatch journal Amazon à votre application. Pour plus d'informations sur l'utilisation CloudWatch des journaux avec votre application, consultez la section Configuration de la journalisation des applications.

Mettre à jour des propriétés d'environnement

Dans cette section, vous utilisez l'action UpdateApplication pour modifier les propriétés d'environnement de l'application sans recompiler le code de l'application. Dans cet exemple, vous modifiez la région des flux source et de destination.

Pour mettre à jour des propriétés d'environnement pour l'application

1. Copiez le code JSON suivant dans un fichier nommé `update_properties_request.json`.

```
{"ApplicationName": "tumbling_window",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
            }
         ]
```

```
        }
    }
}
```

2. Exécutez l'action `UpdateApplication` avec la demande précédente pour mettre à jour les propriétés de l'environnement :

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Mise à jour du code de l’application

Lorsque vous devez mettre à jour le code de votre application avec une nouvelle version de votre package de code, vous utilisez l'action UpdateApplicationCLI.

> **Note**
>
> Pour charger une nouvelle version du code de l'application portant le même nom de fichier, vous devez spécifier la nouvelle version de l'objet. Pour de plus amples informations sur l’utilisation des versions d’objet Amazon S3, veuillez consulter Activation et désactivation de la gestion des versions.

Pour utiliser l’interface AWS CLI, supprimez votre ancien package de code de votre compartiment Amazon S3, téléchargez la nouvelle version et appelez `UpdateApplication` en spécifiant le même compartiment Amazon S3 et le même nom d'objet, ainsi que la nouvelle version de l'objet. L'application redémarrera avec le nouveau package de code.

L'exemple de demande d'action `UpdateApplication` suivant recharge le code de l'application et redémarre l'application. Mettez à jour l’`CurrentApplicationVersionId` à la version actuelle de l'application. Vous pouvez vérifier la version actuelle de l'application à l'aide des actions `ListApplications` ou `DescribeApplication`. Mettez à jour le suffixe du nom du compartiment (<username>) avec le suffixe que vous avez choisi dans la section Création de ressources dépendantes.

```
{
    "ApplicationName": "tumbling_window",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
```

```
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "tumbling-window-scala-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Fenêtre bascule.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer vos flux de données Kinesis
- Supprimer votre objet et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. dans le panneau Managed Service for Apache Flink, sélectionnez MyApplication.
3. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer vos flux de données Kinesis

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Kinesis Data Streams, ExampleInputStreamsélectionnez.
3. Sur la ExampleInputStreampage, choisissez Supprimer Kinesis Stream, puis confirmez la suppression.

4. Sur la page Kinesis Streams, choisissez le ExampleOutputStream, choisissez Actions, choisissez Supprimer, puis confirmez la suppression.

Supprimer votre objet et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code -. <username>
3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.
8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Exemple : création d'une fenêtre défilante dans Scala

> **Note**
>
> À partir de la version 1.15, Flink n'utilise plus Scala. Les applications peuvent désormais utiliser l'API Java depuis n'importe quelle version de Scala. Flink utilise toujours Scala dans

quelques composants clés en interne, mais n'expose pas Scala dans le chargeur de classes du code de l'utilisateur. Pour cette raison, les utilisateurs doivent ajouter des dépendances Scala dans leurs archives JAR.
Pour plus d'informations sur les modifications apportées à Scala dans Flink 1.15, consultez Scala Free in One Fifteen.

Dans cet exercice, vous allez créer une application de streaming simple qui utilise Scala 3.2.0 et l'API Java de Flink. DataStream L'application lit les données du flux Kinesis, les agrège à l'aide de fenêtres défilantes et écrit les résultats pour générer le flux Kinesis.

> **Note**
>
> Pour configurer les prérequis requis pour cet exercice, commencez par terminer l'exercice Mise en route (Scala).

Cette rubrique contient les sections suivantes :

- Téléchargez et examinez le code de l'application
- Compiler et charger le code d'application
- Création et exécution de l'application (console)
- Création et exécution de l'application (CLI)
- Mise à jour du code de l'application
- Nettoyage des ressources AWS

Téléchargez et examinez le code de l'application

Le code de l'application Python pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Installez le client Git si vous ne l'avez pas déjà fait. Pour de plus amples informations, veuillez consulter Installation de Git.
2. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/scala/SlidingWindow`.

Notez les informations suivantes à propos du code d'application :

- Un fichier `build.sbt` contient des informations sur la configuration et les dépendances de l'application, y compris les bibliothèques du service géré pour Apache Flink.
- Le fichier `BasicStreamingJob.scala` contient la méthode principale qui définit la fonctionnalité de l'application.
- L'application utilise une source Kinesis pour lire à partir du flux source. L'extrait de code suivant crée la source Kinesis :

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
 defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

  L'application utilise également un récepteur Kinesis pour écrire dans le flux de résultats. L'extrait de code suivant crée le récepteur Kinesis :

```
private def createSink: KinesisStreamsSink[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val outputProperties = applicationProperties.get("ProducerConfigProperties")

  KinesisStreamsSink.builder[String]
    .setKinesisClientProperties(outputProperties)
    .setSerializationSchema(new SimpleStringSchema)
    .setStreamName(outputProperties.getProperty(streamNameKey,
 defaultOutputStreamName))
    .setPartitionKeyGenerator((element: String) => String.valueOf(element.hashCode))
    .build
}
```

- L'application utilise l'opérateur de fenêtre pour trouver le nombre de valeurs pour chaque symbole boursier sur une fenêtre de 10 secondes qui glisse de 5 secondes. Le code suivant crée l'opérateur et envoie les données agrégées vers un nouveau récepteur Kinesis Data Streams :

```
environment.addSource(createSource)
    .map { value =>
      val jsonNode = jsonParser.readValue(value, classOf[JsonNode])
      new Tuple2[String, Double](jsonNode.get("ticker").toString,
 jsonNode.get("price").asDouble)
    }
    .returns(Types.TUPLE(Types.STRING, Types.DOUBLE))
    .keyBy(v => v.f0) // Logically partition the stream for each word
    .window(SlidingProcessingTimeWindows.of(Time.seconds(10), Time.seconds(5)))
    .min(1) // Calculate minimum price per ticker over the window
    .map { value => value.f0 + String.format(",%.2f", value.f1) + "\n" }
    .sinkTo(createSink)
```

- L'application crée des connecteurs source et récepteur pour accéder à des ressources externes à l'aide d'un StreamExecutionEnvironment objet.
- L'application crée les connecteurs source et récepteur à l'aide de propriétés d'application dynamiques. Les propriétés d'exécution de l'application sont lues pour configurer les connecteurs. Pour de plus amples informations sur les propriétés d'exécution, veuillez consulter Runtime Properties.

Compiler et charger le code d'application

Dans cette section, vous compilez et charger votre code d'application dans un compartiment Amazon S3.

Compilation du code d'application

Utilisez l'outil de construction SBT pour créer le code Scala de l'application. Pour installer SBT, voir Install sbt with cs setup. Vous devez également installer le kit de développement Java (JDK). Consultez Prerequisites for Completing the Exercises.

1. Pour utiliser votre code d'application, vous le compilez et l'intégrez dans un fichier JAR. Vous pouvez compiler et empaqueter votre code avec SBT :

   ```
   sbt assembly
   ```

2. Si la compilation de l'application aboutit, le fichier suivant est créé :

   ```
   target/scala-3.2.0/sliding-window-scala-1.0.jar
   ```

Chargement du code Scala Apache Flink

Dans cette section, vous allez créer un compartiment Amazon S3 et charger votre code d'application.

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez Créer un compartiment.
3. Saisissez `ka-app-code-<username>` dans le champ Nom du compartiment. Ajoutez un suffixe au nom du compartiment, par exemple votre nom d'utilisateur, pour qu'il soit unique. Choisissez Suivant.
4. À l'étape Configurer les options, conservez les paramètres, puis choisissez Suivant.
5. À l'étape Définir des autorisations, conservez les paramètres, puis choisissez Suivant.
6. Choisissez Créer un compartiment.
7. Ouvrez le compartiment `ka-app-code-<username>`, puis choisissez Charger.
8. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `sliding-window-scala-1.0.jar` que vous avez créé à l'étape précédente.
9. Vous n'avez pas besoin de modifier les paramètres de l'objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l'application (console)

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Description, saisissez **`My Scala test app`**.
   - Pour Exécution, choisissez Apache Flink.

- Laissez la version sur Apache Flink 1.15.2 (version recommandée).

4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **kinesis-analytics-MyApplication-us-west-2**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-MyApplication-us-west-2`
> - Rôle : `kinesisanalytics-MyApplication-us-west-2`

Configuration de l'application

Procédez comme suit pour configurer l'application.

Pour configurer l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **ka-app-code-*<username>***.
   - Pour le chemin de l'objet Amazon S3, saisissez **sliding-window-scala-1.0.jar.**.
3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Sous Propriétés, sélectionnez Ajouter un groupe.
5. Saisissez :

ID du groupe	Clé	Valeur
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

Choisissez Enregistrer.

6. Sous Propriétés, sélectionnez à nouveau Ajouter un groupe.
7. Saisissez :

ID du groupe	Clé	Valeur
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**
ProducerConfigProperties	**aws.region**	**us-west-2**

8. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
9. Pour la CloudWatch journalisation, cochez la case Activer.
10. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation Amazon, Managed Service for Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`

- Flux de journaux : `kinesis-analytics-log-stream`

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder au compartiment Amazon S3.

Pour modifier la politique IAM afin d'ajouter des autorisations au compartiment S3

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **`kinesis-analytics-service-MyApplication-us-west-2`** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*012345678901*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/sliding-window-scala-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
```

```
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Exécution de l'application

Le graphique des tâches Flink peut être visualisé en exécutant l'application, en ouvrant le tableau de bord Apache Flink et en choisissant la tâche Flink souhaitée.

Arrêt de l'application

Pour arrêter l'application, sur la MyApplicationpage, choisissez Arrêter. Confirmez l'action.

Création et exécution de l'application (CLI)

Dans cette section, vous allez utiliser l’interface AWS Command Line Interface pour créer et exécuter l'application de service géré pour Apache Flink. Utilisez la commande AWS CLI kinesisanalyticsv2 pour créer et interagir avec les applications de service géré pour Apache Flink.

Créer une stratégie d’autorisations

> **Note**
>
> Vous devez créer une stratégie d'autorisations et un rôle pour votre application. Si vous ne créez pas ces ressources IAM, votre application ne peut pas accéder à ses flux de données et de journaux.

Vous commencez par créer une stratégie d'autorisations avec deux instructions : une qui accorde des autorisations pour l'action de lecture sur le flux source et une autre qui accorde des autorisations pour les actions d’écriture sur le flux récepteur. Vous attachez ensuite la politique à un rôle IAM (que vous allez créer dans la section suivante). Ainsi, lorsque le service géré pour Apache Flink assume le rôle, le service dispose des autorisations nécessaires pour lire à partir du flux source et écrire dans le flux récepteur.

Utilisez le code suivant pour créer la stratégie d'autorisations `AKReadSourceStreamWriteSinkStream`. Remplacez **`username`** par le nom d'utilisateur que vous avez utilisé pour créer le compartiment Amazon S3 pour stocker le code d'application. Remplacez l'ID de compte dans Amazon Resource Names (ARN) (**`(012345678901)`**) par votre ID de compte.

```
{
    "ApplicationName": "sliding_window",
    "ApplicationDescription": "Scala sliding window application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
```

```
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "sliding-window-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
            }
         ]
      }
    },
    "CloudWatchLoggingOptions": [
      {
         "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
      }
   ]
}
```

Pour step-by-step obtenir des instructions sur la création d'une politique d'autorisations, voir Tutoriel : créer et joindre votre première politique gérée par le client dans le guide de l'utilisateur IAM.

Créer un rôle IAM

Dans cette section, vous créez un rôle IAM que l’application de service géré pour Apache Flink peut assumer pour lire un flux source et écrire dans le flux récepteur.

Le service géré pour Apache Flink ne peut pas accéder à votre flux sans autorisation. Vous utilisez un rôle IAM pour accorder ces autorisations. Deux politiques sont attachées à chaque rôle IAM. La politique d'approbation accorde au service géré pour Apache Flink l'autorisation d'assumer le rôle, et la politique d'autorisation détermine ce que le service géré pour Apache Flink peut faire après avoir assumé le rôle.

Vous attachez la stratégie d'autorisations que vous avez créée dans la section précédente à ce rôle.

Pour créer un rôle IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, choisissez Rôles, puis Créer un Rôle.
3. Sous Sélectionner le type d'identité approuvée, choisissez Service AWS.
4. Sous Choisir le service qui utilisera ce rôle, choisissez EC2.
5. Sous Sélectionnez votre cas d'utilisation, choisissez service géré pour Apache Flink.
6. Sélectionnez Next: Permissions (Étape suivante : autorisations).
7. Dans la page Attacher des stratégies d'autorisations, choisissez Suivant : vérification. Vous attachez des stratégies d'autorisations après avoir créé le rôle.
8. Sur la page Créer un rôle, saisissez **`MF-stream-rw-role`** pour le Nom du rôle. Sélectionnez Créer un rôle.

   Vous venez de créer un nouveau rôle IAM appelé `MF-stream-rw-role`. Ensuite, vous mettez à jour les stratégies d'approbation et d'autorisation pour le rôle.
9. Attachez la politique d'autorisation au rôle.

   > **Note**
   >
   > Dans le cadre de cet exercice, le service géré pour Apache Flink assume ce rôle à la fois pour la lecture des données à partir d'un flux de données Kinesis (source) et pour l'écriture des résultats dans un autre flux de données Kinesis. Vous attachez donc la stratégie que vous avez créée à l'étape précédente, Créer une stratégie d'autorisations.

   a. Sur la page Récapitulatif, choisissez l'onglet Autorisations.

   b. Choisissez Attacher des stratégies.

c. Dans la zone de recherche, saisissez **`AKReadSourceStreamWriteSinkStream`** (la stratégie que vous avez créée dans la section précédente).

d. Sélectionnez Actions, puis Attacher une stratégie.

Vous avez maintenant créé le rôle d'exécution de service que votre application utilise pour accéder aux ressources. Notez l'ARN du nouveau rôle.

Pour step-by-step obtenir des instructions sur la création d'un rôle, consultez la section Création d'un rôle IAM (console) dans le guide de l'utilisateur IAM.

Pour créer l'application

Copiez le code JSON suivant dans un fichier nommé `create_request.json`. Remplacez l'exemple d'ARN du rôle par l'ARN du rôle que vous avez créé précédemment. Remplacez le suffixe de l'ARN du compartiment (nom d'utilisateur) par le suffixe que vous avez choisi dans la section précédente. Remplacez l'exemple d'ID de compte (012345678901) dans le rôle d'exécution de service par votre ID de compte.

```
{
    "ApplicationName": "sliding_window",
    "ApplicationDescription": "Scala sliding_window application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "sliding-window-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
```

```
                }
            },
            {
                "PropertyGroupId": "ProducerConfigProperties",
                "PropertyMap" : {
                     "aws.region" : "us-west-2",
                     "stream.name" : "ExampleOutputStream"
                }
            }
         ]
      }
    },
    "CloudWatchLoggingOptions": [
      {
         "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
      }
   ]
}
```

Exécutez le CreateApplicationavec la requête suivante pour créer l'application :

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

L'application est maintenant créée. Vous démarrez l'application dans l'étape suivante.

Démarrage de l'application

Dans cette section, vous utilisez l'action StartApplication pour démarrer l'application.

Pour démarrer l'application

1. Copiez le code JSON suivant dans un fichier nommé `start_request.json`.

```
{
    "ApplicationName": "sliding_window",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. Exécutez l'action `StartApplication` avec la demande précédente pour démarrer l'application :

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

L'application est maintenant en cours d'exécution. Vous pouvez consulter les métriques du service géré pour Apache Flink sur la CloudWatch console Amazon pour vérifier que l'application fonctionne.

Arrêt de l'application

Dans cette section, vous allez utiliser l'action StopApplication pour arrêter l'application.

Pour arrêter l'application

1. Copiez le code JSON suivant dans un fichier nommé `stop_request.json`.

```
{
   "ApplicationName": "sliding_window"
}
```

2. Exécutez l'action `StopApplication` avec la demande précédente pour arrêter l'application :

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

L'application est maintenant arrêtée.

Ajouter une option de CloudWatch journalisation

Vous pouvez utiliser le AWS CLI pour ajouter un flux de CloudWatch journal Amazon à votre application. Pour plus d'informations sur l'utilisation CloudWatch des journaux avec votre application, consultez la section Configuration de la journalisation des applications.

Mettre à jour des propriétés d'environnement

Dans cette section, vous utilisez l'action UpdateApplication pour modifier les propriétés d'environnement de l'application sans recompiler le code de l'application. Dans cet exemple, vous modifiez la région des flux source et de destination.

Pour mettre à jour des propriétés d'environnement pour l'application

1. Copiez le code JSON suivant dans un fichier nommé `update_properties_request.json`.

```
{"ApplicationName": "sliding_window",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleOutputStream"
               }
            }
         ]
      }
   }
}
```

2. Exécutez l'action `UpdateApplication` avec la demande précédente pour mettre à jour les propriétés de l'environnement :

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Mise à jour du code de l’application

Lorsque vous devez mettre à jour le code de votre application avec une nouvelle version de votre package de code, vous utilisez l'action UpdateApplicationCLI.

> **Note**
>
> Pour charger une nouvelle version du code de l'application portant le même nom de fichier, vous devez spécifier la nouvelle version de l'objet. Pour de plus amples informations sur l'utilisation des versions d'objet Amazon S3, veuillez consulter Activation et désactivation de la gestion des versions.

Pour utiliser l'interface AWS CLI, supprimez votre ancien package de code de votre compartiment Amazon S3, téléchargez la nouvelle version et appelez `UpdateApplication` en spécifiant le même compartiment Amazon S3 et le même nom d'objet, ainsi que la nouvelle version de l'objet. L'application redémarrera avec le nouveau package de code.

L'exemple de demande d'action `UpdateApplication` suivant recharge le code de l'application et redémarre l'application. Mettez à jour l'`CurrentApplicationVersionId` à la version actuelle de l'application. Vous pouvez vérifier la version actuelle de l'application à l'aide des actions `ListApplications` ou `DescribeApplication`. Mettez à jour le suffixe du nom du compartiment (<username>) avec le suffixe que vous avez choisi dans la section Création de ressources dépendantes.

```
{
    "ApplicationName": "sliding_window",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Fenêtre défilante.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer vos flux de données Kinesis
- Supprimer votre objet et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. dans le panneau Managed Service for Apache Flink, sélectionnez MyApplication.
3. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer vos flux de données Kinesis

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Kinesis Data Streams, ExampleInputStreamsélectionnez.
3. Sur la ExampleInputStreampage, choisissez Supprimer Kinesis Stream, puis confirmez la suppression.
4. Sur la page Kinesis Streams, choisissez le ExampleOutputStream, choisissez Actions, choisissez Supprimer, puis confirmez la suppression.

Supprimer votre objet et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code -. <username>
3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.

4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.
8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Exemple : envoyer des données de streaming vers Amazon S3 dans Scala

> **Note**
>
> À partir de la version 1.15, Flink n'utilise plus Scala. Les applications peuvent désormais utiliser l'API Java depuis n'importe quelle version de Scala. Flink utilise toujours Scala dans quelques composants clés en interne, mais n'expose pas Scala dans le chargeur de classes du code de l'utilisateur. Pour cette raison, les utilisateurs doivent ajouter des dépendances Scala dans leurs archives JAR.
> Pour plus d'informations sur les modifications apportées à Scala dans Flink 1.15, consultez Scala Free in One Fifteen.

Dans cet exercice, vous allez créer une application de streaming simple qui utilise Scala 3.2.0 et l'API Java de Flink. DataStream L'application lit les données du flux Kinesis, les agrège à l'aide de fenêtres défilantes et écrit les résultats dans S3.

> **Note**
>
> Pour configurer les prérequis requis pour cet exercice, commencez par terminer l'exercice Mise en route (Scala). Il vous suffit de créer un dossier supplémentaire **`data/`** dans le compartiment Amazon S3 ka-app-code-<username>.

Cette rubrique contient les sections suivantes :

- Téléchargez et examinez le code de l'application
- Compiler et charger le code d'application
- Création et exécution de l'application (console)
- Création et exécution de l'application (CLI)
- Mise à jour du code de l’application
- Nettoyage des ressources AWS

Téléchargez et examinez le code de l'application

Le code de l'application Python pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Installez le client Git si vous ne l'avez pas déjà fait. Pour de plus amples informations, veuillez consulter Installation de Git.
2. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/scala/S3Sink`.

Notez les informations suivantes à propos du code d'application :

- Un fichier `build.sbt` contient des informations sur la configuration et les dépendances de l'application, y compris les bibliothèques du service géré pour Apache Flink.
- Le fichier `BasicStreamingJob.scala` contient la méthode principale qui définit la fonctionnalité de l'application.
- L'application utilise une source Kinesis pour lire à partir du flux source. L'extrait de code suivant crée la source Kinesis :

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")
```

```
  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
 defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

L'application utilise également un StreamingFileSink pour écrire dans un compartiment Amazon S3 : `

```
def createSink: StreamingFileSink[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val s3SinkPath =
 applicationProperties.get("ProducerConfigProperties").getProperty("s3.sink.path")

  StreamingFileSink
    .forRowFormat(new Path(s3SinkPath), new SimpleStringEncoder[String]("UTF-8"))
    .build()
}
```

- L'application crée des connecteurs source et récepteur pour accéder à des ressources externes à l'aide d'un StreamExecutionEnvironment objet.
- L'application crée les connecteurs source et récepteur à l'aide de propriétés d'application dynamiques. Les propriétés d'exécution de l'application sont lues pour configurer les connecteurs. Pour de plus amples informations sur les propriétés d'exécution, veuillez consulter Runtime Properties.

Compiler et charger le code d'application

Dans cette section, vous compilez et charger votre code d'application dans un compartiment Amazon S3.

Compilation du code d'application

Utilisez l'outil de construction SBT pour créer le code Scala de l'application. Pour installer SBT, voir Install sbt with cs setup. Vous devez également installer le kit de développement Java (JDK). Consultez Prerequisites for Completing the Exercises.

1. Pour utiliser votre code d'application, vous le compilez et l'intégrez dans un fichier JAR. Vous pouvez compiler et empaqueter votre code avec SBT :

```
sbt assembly
```

2. Si la compilation de l'application aboutit, le fichier suivant est créé :

```
target/scala-3.2.0/s3-sink-scala-1.0.jar
```

Chargement du code Scala Apache Flink

Dans cette section, vous allez créer un compartiment Amazon S3 et charger votre code d'application.

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez Créer un compartiment.
3. Saisissez `ka-app-code-<username>` dans le champ Nom du compartiment. Ajoutez un suffixe au nom du compartiment, par exemple votre nom d'utilisateur, pour qu'il soit unique. Choisissez Suivant.
4. À l'étape Configurer les options, conservez les paramètres, puis choisissez Suivant.
5. À l'étape Définir des autorisations, conservez les paramètres, puis choisissez Suivant.
6. Choisissez Créer un compartiment.
7. Ouvrez le compartiment `ka-app-code-<username>`, puis choisissez Charger.
8. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `s3-sink-scala-1.0.jar` que vous avez créé à l'étape précédente.
9. Vous n'avez pas besoin de modifier les paramètres de l’objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l'application (console)

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l’adresse https://console.aws.amazon.com/flink

2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Description, saisissez **`My java test app`**.
   - Pour Exécution, choisissez Apache Flink.
   - Laissez la version sur Apache Flink 1.15.2 (version recommandée).
4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rôle : `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Configuration de l'application

Procédez comme suit pour configurer l'application.

Pour configurer l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **`ka-app-code-`***`<username>`*.
   - Pour le chemin de l'objet Amazon S3, saisissez **`s3-sink-scala-1.0.jar`**.

3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **`kinesis-analytics-MyApplication-us-west-2`**.
4. Sous Propriétés, sélectionnez Ajouter un groupe.
5. Saisissez :

ID du groupe	Clé	Valeur
`ConsumerConfigProperties`	**`input.stream.name`**	**`ExampleInputStream`**
`ConsumerConfigProperties`	**`aws.region`**	**`us-west-2`**
`ConsumerConfigProperties`	**`flink.stream.initpos`**	**`LATEST`**

   Choisissez Enregistrer.
6. Sous Propriétés, sélectionnez Ajouter un groupe.
7. Saisissez :

ID du groupe	Clé	Valeur
`ProducerConfigProperties`	**`s3.sink.path`**	**`s3a://ka-app-code-`*`<user-name>`*` /data`**

8. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
9. Pour la CloudWatch journalisation, cochez la case Activer.
10. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation Amazon, Managed Service for Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :

- Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
- Flux de journaux : `kinesis-analytics-log-stream`

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder au compartiment Amazon S3.

Pour modifier la politique IAM afin d'ajouter des autorisations au compartiment S3

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **`kinesis-analytics-service-MyApplication-us-west-2`** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*`012345678901`*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:Abort*",
                "s3:DeleteObject*",
                "s3:GetObject*",
                "s3:GetBucket*",
                "s3:List*",
                "s3:ListBucket",
                "s3:PutObject"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-<username>",
                "arn:aws:s3:::ka-app-code-<username>/*"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
```

```
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        }
    ]
}
```

Exécution de l'application

Le graphique des tâches Flink peut être visualisé en exécutant l'application, en ouvrant le tableau de bord Apache Flink et en choisissant la tâche Flink souhaitée.

Arrêt de l'application

Pour arrêter l'application, sur la MyApplicationpage, choisissez Arrêter. Confirmez l'action.

Création et exécution de l'application (CLI)

Dans cette section, vous allez utiliser l'interface AWS Command Line Interface pour créer et exécuter l'application de service géré pour Apache Flink. Utilisez la commande AWS CLI kinesisanalyticsv2 pour créer et interagir avec les applications de service géré pour Apache Flink.

Créer une stratégie d'autorisations

> **Note**
>
> Vous devez créer une stratégie d'autorisations et un rôle pour votre application. Si vous ne créez pas ces ressources IAM, votre application ne peut pas accéder à ses flux de données et de journaux.

Vous commencez par créer une stratégie d'autorisations avec deux instructions : une qui accorde des autorisations pour l'action de lecture sur le flux source et une autre qui accorde des autorisations pour les actions d'écriture sur le flux récepteur. Vous attachez ensuite la politique à un rôle IAM (que vous allez créer dans la section suivante). Ainsi, lorsque le service géré pour Apache Flink assume le rôle, le service dispose des autorisations nécessaires pour lire à partir du flux source et écrire dans le flux récepteur.

Utilisez le code suivant pour créer la stratégie d'autorisations `AKReadSourceStreamWriteSinkStream`. Remplacez **`username`** par le nom d'utilisateur que vous avez utilisé pour créer le compartiment Amazon S3 pour stocker le code d'application. Remplacez l'ID de compte dans Amazon Resource Names (ARN) (**`(012345678901)`**) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/getting-started-scala-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-analytics/
MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
```

```
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Pour step-by-step obtenir des instructions sur la création d'une politique d'autorisations, voir Tutoriel : créer et joindre votre première politique gérée par le client dans le guide de l'utilisateur IAM.

Créer un rôle IAM

Dans cette section, vous créez un rôle IAM que l'application de service géré pour Apache Flink peut assumer pour lire un flux source et écrire dans le flux récepteur.

Le service géré pour Apache Flink ne peut pas accéder à votre flux sans autorisation. Vous utilisez un rôle IAM pour accorder ces autorisations. Deux politiques sont attachées à chaque rôle IAM. La politique d'approbation accorde au service géré pour Apache Flink l'autorisation d'assumer le rôle, et la politique d'autorisation détermine ce que le service géré pour Apache Flink peut faire après avoir assumé le rôle.

Vous attachez la stratégie d'autorisations que vous avez créée dans la section précédente à ce rôle.

Pour créer un rôle IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, choisissez Rôles, puis Créer un Rôle.
3. Sous Sélectionner le type d'identité approuvée, choisissez Service AWS.
4. Sous Choisir le service qui utilisera ce rôle, choisissez EC2.
5. Sous Sélectionnez votre cas d'utilisation, choisissez service géré pour Apache Flink.
6. Sélectionnez Next: Permissions (Étape suivante : autorisations).

7. Dans la page Attacher des stratégies d'autorisations, choisissez Suivant : vérification. Vous attachez des stratégies d'autorisations après avoir créé le rôle.
8. Sur la page Créer un rôle, saisissez **`MF-stream-rw-role`** pour le Nom du rôle. Sélectionnez Créer un rôle.

   Vous venez de créer un nouveau rôle IAM appelé `MF-stream-rw-role`. Ensuite, vous mettez à jour les stratégies d'approbation et d'autorisation pour le rôle.
9. Attachez la politique d'autorisation au rôle.

   > **Note**
   >
   > Dans le cadre de cet exercice, le service géré pour Apache Flink assume ce rôle à la fois pour la lecture des données à partir d'un flux de données Kinesis (source) et pour l'écriture des résultats dans un autre flux de données Kinesis. Vous attachez donc la stratégie que vous avez créée à l'étape précédente, Créer une stratégie d'autorisations.

   a. Sur la page Récapitulatif, choisissez l'onglet Autorisations.
   b. Choisissez Attacher des stratégies.
   c. Dans la zone de recherche, saisissez **`AKReadSourceStreamWriteSinkStream`** (la stratégie que vous avez créée dans la section précédente).
   d. Sélectionnez Actions, puis Attacher une stratégie.

Vous avez maintenant créé le rôle d'exécution de service que votre application utilise pour accéder aux ressources. Notez l'ARN du nouveau rôle.

Pour step-by-step obtenir des instructions sur la création d'un rôle, consultez la section Création d'un rôle IAM (console) dans le guide de l'utilisateur IAM.

Pour créer l'application

Copiez le code JSON suivant dans un fichier nommé `create_request.json`. Remplacez l'exemple d'ARN du rôle par l'ARN du rôle que vous avez créé précédemment. Remplacez le suffixe de l'ARN du compartiment (nom d'utilisateur) par le suffixe que vous avez choisi dans la section précédente. Remplacez l'exemple d'ID de compte (012345678901) dans le rôle d'exécution de service par votre ID de compte.

```
{
    "ApplicationName": "s3_sink",
    "ApplicationDescription": "Scala tumbling window application",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "s3-sink-scala-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "s3.sink.path" : "s3a://ka-app-code-<username>/data"
               }
            }
         ]
      }
    },
    "CloudWatchLoggingOptions": [
      {
         "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
      }
   ]
}
```

Exécutez le CreateApplicationavec la requête suivante pour créer l'application :

```
aws kinesisanalyticsv2 create-application --cli-input-json file://create_request.json
```

L'application est maintenant créée. Vous démarrez l'application dans l'étape suivante.

Démarrage de l'application

Dans cette section, vous utilisez l'action StartApplication pour démarrer l'application.

Pour démarrer l'application

1. Copiez le code JSON suivant dans un fichier nommé `start_request.json`.

```
{{
    "ApplicationName": "s3_sink",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. Exécutez l'action `StartApplication` avec la demande précédente pour démarrer l'application :

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

L'application est maintenant en cours d'exécution. Vous pouvez consulter les métriques du service géré pour Apache Flink sur la CloudWatch console Amazon pour vérifier que l'application fonctionne.

Arrêt de l'application

Dans cette section, vous allez utiliser l'action StopApplication pour arrêter l'application.

Pour arrêter l'application

1. Copiez le code JSON suivant dans un fichier nommé `stop_request.json`.

```
{
   "ApplicationName": "s3_sink"
}
```

2. Exécutez l'action `StopApplication` avec la demande précédente pour arrêter l'application :

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

L'application est maintenant arrêtée.

Ajouter une option de CloudWatch journalisation

Vous pouvez utiliser le AWS CLI pour ajouter un flux de CloudWatch journal Amazon à votre application. Pour plus d'informations sur l'utilisation CloudWatch des journaux avec votre application, consultez la section Configuration de la journalisation des applications.

Mettre à jour des propriétés d'environnement

Dans cette section, vous utilisez l'action UpdateApplication pour modifier les propriétés d'environnement de l'application sans recompiler le code de l'application. Dans cet exemple, vous modifiez la région des flux source et de destination.

Pour mettre à jour des propriétés d'environnement pour l'application

1. Copiez le code JSON suivant dans un fichier nommé `update_properties_request.json`.

```
{"ApplicationName": "s3_sink",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2",
                    "stream.name" : "ExampleInputStream",
                    "flink.stream.initpos" : "LATEST"
               }
            },
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "s3.sink.path" : "s3a://ka-app-code-<username>/data"
               }
            }
         ]
```

```
        }
    }
}
```

2. Exécutez l'action `UpdateApplication` avec la demande précédente pour mettre à jour les propriétés de l'environnement :

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Mise à jour du code de l’application

Lorsque vous devez mettre à jour le code de votre application avec une nouvelle version de votre package de code, vous utilisez l'action UpdateApplicationCLI.

> **Note**
>
> Pour charger une nouvelle version du code de l'application portant le même nom de fichier, vous devez spécifier la nouvelle version de l'objet. Pour de plus amples informations sur l’utilisation des versions d’objet Amazon S3, veuillez consulter Activation et désactivation de la gestion des versions.

Pour utiliser l’interface AWS CLI, supprimez votre ancien package de code de votre compartiment Amazon S3, téléchargez la nouvelle version et appelez `UpdateApplication` en spécifiant le même compartiment Amazon S3 et le même nom d'objet, ainsi que la nouvelle version de l'objet. L'application redémarrera avec le nouveau package de code.

L'exemple de demande d'action `UpdateApplication` suivant recharge le code de l'application et redémarre l'application. Mettez à jour l’`CurrentApplicationVersionId` à la version actuelle de l'application. Vous pouvez vérifier la version actuelle de l'application à l'aide des actions `ListApplications` ou `DescribeApplication`. Mettez à jour le suffixe du nom du compartiment (<username>) avec le suffixe que vous avez choisi dans la section Création de ressources dépendantes.

```
{
    "ApplicationName": "s3_sink",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
```

```
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "s3-sink-scala-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Fenêtre bascule.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer vos flux de données Kinesis
- Supprimer votre objet et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. dans le panneau Managed Service for Apache Flink, sélectionnez MyApplication.
3. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer vos flux de données Kinesis

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Kinesis Data Streams, ExampleInputStreamsélectionnez.
3. Sur la ExampleInputStreampage, choisissez Supprimer Kinesis Stream, puis confirmez la suppression.

4. Sur la page Kinesis Streams, choisissez le ExampleOutputStream, choisissez Actions, choisissez Supprimer, puis confirmez la suppression.

Supprimer votre objet et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code -. <username>
3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.
8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Sécurité du service géré Amazon pour Apache Flink

Chez AWS, la sécurité dans le cloud est la priorité numéro 1. En tant que client d'AWS, 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.

La sécurité est une responsabilité partagée entre AWS et vous-même. Le modèle de responsabilité partagée décrit cette notion par les termes sécurité du cloud et sécurité dans le cloud :

- Sécurité du cloud : AWS est responsable de la protection de l'infrastructure qui exécute des services AWS dans le cloud AWS. AWS vous fournit également les 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 au service géré pour Apache Flink, consultez Services AWS concernés par le programme de conformité.
- Sécurité dans le cloud : votre responsabilité est déterminée par le service AWS 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 aide à comprendre comment appliquer le modèle de responsabilité partagée lors de l'utilisation du service géré pour Apache Flink. Les rubriques suivantes vous montrent comment configurer le service géré pour Apache Flink pour répondre à vos objectifs de sécurité et de conformité. Vous pouvez également apprendre à utiliser d'autres services Amazon qui vous permettent de surveiller et de sécuriser les ressources de votre service géré pour Apache Flink.

Rubriques

- Protection des données dans le service géré Amazon pour Apache Flink
- Gestion de l'identité et des accès dans le service géré Amazon pour Apache Flink
- Surveillance dans le service géré pour Apache Flink
- Validation de la conformité pour le service géré pour Apache Flink
- Résilience dans le service géré Amazon pour Apache Flink
- Sécurité de l'infrastructure du service géré Amazon pour Apache Flink
- Bonnes pratiques de sécurité pour le service géré pour Apache Flink

Protection des données dans le service géré Amazon pour Apache Flink

Vous pouvez protéger vos données à l'aide des outils fournis par AWS. Le service géré pour Apache Flink peut fonctionner avec des services qui prennent en charge le chiffrement des données, notamment Kinesis Data Firehose et Amazon S3.

Chiffrement des données dans le service géré pour Apache Flink

Chiffrement au repos

Notez les éléments suivants à propos du chiffrement des données au repos avec le service géré pour Apache Flink :

- Vous pouvez chiffrer les données du flux de données Kinesis entrant à l'aide de. StartStreamEncryption Pour plus d'informations, consultez Qu'est-ce que le chiffrement côté serveur pour Kinesis Streams Data ?.
- Les données de sortie peuvent être chiffrées au repos avec Kinesis Data Firehose pour stocker les données dans un compartiment Amazon S3 chiffré. Vous pouvez spécifier la clé de chiffrement que le compartiment Amazon S3 utilise. Pour plus d'informations, consultez Protection des données avec le chiffrement côté serveur avec des clés gérées par KMS (SSE-KMS).
- Le service géré pour Apache Flink peut lire à partir de n'importe quelle source de streaming et écrire sur n'importe quelle destination de streaming ou de base de données. Assurez-vous que vos sources et destinations chiffrent toutes les données en transit et les données au repos.
- Le code de votre application est chiffré au repos.
- Le stockage durable des applications est chiffré au repos.
- Le stockage des applications en cours est chiffré au repos.

Chiffrement en transit

Le service géré pour Apache Flink chiffre toutes les données en transit. Le chiffrement en transit est activé pour toutes les applications de service géré pour Apache Flink et ne peut pas être désactivé.

Le service géré pour Apache Flink chiffre toutes les données en transit dans les scénarios suivants :

- Données en transit entre Kinesis Data Streams et le service géré pour Apache Flink.

- Données en transit entre les composants internes au sein du service géré pour Apache Flink.
- Données en transit entre le service géré pour Apache Flink et Kinesis Data Firehose.

Gestion des clés

Le chiffrement des données dans le service géré pour Apache Flink utilise des clés gérées par le service. Les clés gérées par le client ne sont pas prises en charge.

Gestion de l'identité et des accès dans le service géré Amazon pour Apache Flink

AWS Identity and Access Management (IAM) est un Service AWS qui aide un administrateur à contrôler en toute sécurité l’accès aux ressources AWS. Des administrateurs IAM contrôlent les personnes qui s'authentifient (sont connectées) et sont autorisées (disposent d'autorisations) à utiliser des ressources du service géré pour Apache Flink. IAM est un Service AWS que vous pouvez utiliser sans frais supplémentaires.

Rubriques

- Public ciblé
- Authentification par des identités
- Gestion des accès à l’aide de politiques
- Utilisation du service géré Amazon pour Apache Flink avec IAM
- Exemples de politiques basées sur l'identité pour le service géré Amazon pour Apache Flink
- Résolution des problèmes liés à l'identité et aux accès dans le service géré Amazon pour Apache Flink
- Prévention du problème de l’adjoint confus entre services

Public ciblé

La façon dont vous utilisez AWS Identity and Access Management (IAM) diffère selon la tâche que vous accomplissez dans le service géré pour Apache Flink.

Utilisateur du service : si vous utilisez le service géré pour Apache Flink pour effectuer votre tâche, votre administrateur vous fournit les informations d'identification et les autorisations dont vous

avez besoin. Au fur et à mesure que vous utilisez les fonctionnalités du service géré pour Apache Flink pour la recherche pour effectuer vos tâches, il se peut que vous ayez besoin d'autorisations supplémentaires. En comprenant bien la gestion des accès, vous saurez demander les autorisations appropriées à votre administrateur. Si vous ne pouvez pas accéder à une fonctionnalité dans le service géré pour Apache Flink, consultez la section Résolution des problèmes liés à l'identité et aux accès dans le service géré Amazon pour Apache Flink.

Administrateur du service : si vous êtes le responsable des ressources du service géré pour Apache Flink de votre entreprise, vous bénéficiez probablement d'un accès total au service géré pour Apache Flink. C'est à vous de déterminer les fonctionnalités et les ressources du service géré pour Apache Flink auxquelles les utilisateurs de votre service doivent avoir accès. Vous devez ensuite soumettre les demandes à votre administrateur IAM pour modifier les autorisations des utilisateurs de votre service. Consultez les informations sur cette page pour comprendre les concepts de base d'IAM. Pour découvrir la façon dont votre entreprise peut utiliser IAM avec le service géré pour Apache Flink, consultez la section Utilisation du service géré Amazon pour Apache Flink avec IAM.

Administrateur IAM : si vous êtes un administrateur IAM, vous souhaiterez peut-être obtenir des informations sur la façon dont vous pouvez écrire des politiques pour gérer l'accès au service géré pour Apache Flink. Pour afficher des exemples de politiques basées sur l'identité du service géré pour Apache Flink que vous pouvez utiliser dans IAM, consultez la section Exemples de politiques basées sur l'identité pour le service géré Amazon pour Apache Flink.

Authentification par des identités

L'authentification correspond au processus par lequel vous vous connectez à AWS avec vos informations d'identification. Vous devez vous authentifier (être connecté à AWS) en tant qu'Utilisateur racine d'un compte AWS, en tant qu'utilisateur IAM ou en endossant un rôle IAM.

Vous pouvez vous connecter à AWS en tant qu'identité fédérée à l'aide des informations d'identification fournies par le biais d'une source d'identité. AWS IAM Identity Center Les utilisateurs (IAM Identity Center), l'authentification de connexion unique de votre entreprise et vos informations d'identification Google ou Facebook sont des exemples d'identités fédérées. Lorsque vous vous connectez avec une identité fédérée, votre administrateur aura précédemment configuré une fédération d'identités avec des rôles IAM. Lorsque vous accédez à AWS en utilisant la fédération, vous endossez indirectement un rôle.

Selon le type d'utilisateur que vous êtes, vous pouvez vous connecter à la AWS Management Console ou au portail d'accès AWS. Pour plus d'informations sur la connexion à AWS, consultez Connexion à votre Compte AWSdans le Guide de l'utilisateurConnexion à AWS.

Si vous accédez à AWS par programmation, AWS fournit un kit de développement logiciel (SDK) et une interface de ligne de commande (CLI) pour signer cryptographiquement vos demandes en utilisant vos informations d'identification. Si vous n'utilisez pas les outils AWS, vous devez signer les requêtes vous-même. Pour plus d'informations sur l'utilisation de la méthode recommandée pour signer des demandes vous-même, consultez Signature des demandes d'API AWS dans le Guide de l'utilisateur IAM.

Quelle que soit la méthode d'authentification que vous utilisez, vous devrez peut-être fournir des informations de sécurité supplémentaires. Par exemple, AWS vous recommande d'utiliser l'authentification multifactorielle (MFA) pour améliorer la sécurité de votre compte. Pour en savoir plus, veuillez consulter Authentification multifactorielle dans le Guide de l'utilisateur AWS IAM Identity Center et Utilisation de l'authentification multifactorielle (MFA) dans l'interface AWS dans le Guide de l'utilisateur IAM.

Utilisateur root Compte AWS

Lorsque vous créez un Compte AWS, vous commencez avec une seule identité de connexion disposant d'un accès complet à tous les Services AWS et ressources du compte. Cette identité est appelée utilisateur root du Compte AWS. Vous pouvez y accéder en vous connectant à l'aide de l'adresse électronique et du mot de passe que vous avez utilisés pour créer le compte. Il est vivement recommandé de ne pas utiliser l'utilisateur root pour vos tâches quotidiennes. Protégez vos informations d'identification d'utilisateur root et utilisez-les pour effectuer les tâches que seul l'utilisateur root peut effectuer. Pour obtenir la liste complète des tâches qui vous imposent de vous connecter en tant qu'utilisateur root, consultez Tâches nécessitant des informations d'identification d'utilisateur root dans le Guide de l'utilisateur IAM.

Identité fédérée

Demandez aux utilisateurs humains, et notamment aux utilisateurs qui nécessitent un accès administrateur, d'appliquer la bonne pratique consistant à utiliser une fédération avec fournisseur d'identité pour accéder à Services AWS en utilisant des informations d'identification temporaires.

Une identité fédérée est un utilisateur de l'annuaire des utilisateurs de votre entreprise, un fournisseur d'identité Web, l'AWS Directory Service, l'annuaire Identity Center ou tout utilisateur qui accède à Services AWS en utilisant des informations d'identification fournies via une source d'identité. Quand des identités fédérées accèdent à Comptes AWS, elles endossent des rôles, ces derniers fournissant des informations d'identification temporaires.

Pour une gestion des accès centralisée, nous vous recommandons d'utiliser AWS IAM Identity Center. Vous pouvez créer des utilisateurs et des groupes dans IAM Identity Center, ou vous connecter et vous synchroniser avec un ensemble d'utilisateurs et de groupes dans votre propre source d'identité pour une utilisation sur l'ensemble de vos applications et de vos Comptes AWS. Pour obtenir des informations sur IAM Identity Center, consultez Qu'est-ce que IAM Identity Center ? dans le Guide de l'utilisateur AWS IAM Identity Center.

Utilisateurs et groupes IAM

Un utilisateur IAM est une identité dans votre Compte AWS qui dispose d'autorisations spécifiques pour une seule personne ou application. Dans la mesure du possible, nous vous recommandons de vous appuyer sur des informations d'identification temporaires plutôt que de créer des utilisateurs IAM ayant des informations d'identification à long terme tels que les clés d'accès. Toutefois, si certains cas d'utilisation spécifiques nécessitent des informations d'identification à long terme avec les utilisateurs IAM, nous vous recommandons de faire pivoter les clés d'accès. Pour plus d'informations, consultez Rotation régulière des clés d'accès pour les cas d'utilisation nécessitant des informations d'identification dans le Guide de l'utilisateur IAM.

Un groupe IAM est une identité qui concerne un ensemble d'utilisateurs IAM. Vous ne pouvez pas vous connecter en tant que groupe. Vous pouvez utiliser les groupes pour spécifier des autorisations pour plusieurs utilisateurs à la fois. Les groupes permettent de gérer plus facilement les autorisations pour de grands ensembles d'utilisateurs. Par exemple, vous pouvez avoir un groupe nommé IAMAdmins et accorder à ce groupe les autorisations d'administrer des ressources IAM.

Les utilisateurs sont différents des rôles. Un utilisateur est associé de manière unique à une personne ou une application, alors qu'un rôle est conçu pour être endossé par tout utilisateur qui en a besoin. Les utilisateurs disposent d'informations d'identification permanentes, mais les rôles fournissent des informations d'identification temporaires. Pour en savoir plus, consultez Quand créer un utilisateur IAM (au lieu d'un rôle) dans le Guide de l'utilisateur IAM.

Rôles IAM

Un rôle IAM est une entité au sein de votre Compte AWS qui dispose d'autorisations spécifiques. Le concept ressemble à celui d'utilisateur IAM, mais le rôle IAM n'est pas associé à une personne en particulier. Vous pouvez temporairement endosser un rôle IAM dans la AWS Management Console en changeant de rôle. Vous pouvez obtenir un rôle en appelant une opération d'API AWS CLI ou AWS à l'aide d'une URL personnalisée. Pour plus d'informations sur les méthodes d'utilisation des rôles, consultez Utilisation de rôles IAM dans le Guide de l'utilisateur IAM.

Les rôles IAM avec des informations d'identification temporaires sont utiles dans les cas suivants :

- Accès utilisateur fédéré – Pour attribuer des autorisations à une identité fédérée, vous créez un rôle et définissez des autorisations pour le rôle. Quand une identité externe s'authentifie, l'identité est associée au rôle et reçoit les autorisations qui sont définies par celui-ci. Pour obtenir des informations sur les rôles pour la fédération, consultez Création d'un rôle pour un fournisseur d'identité tiers (fédération) dans le Guide de l'utilisateur IAM. Si vous utilisez IAM Identity Center, vous configurez un jeu d'autorisations. IAM Identity Center met en corrélation le jeu d'autorisations avec un rôle dans IAM afin de contrôler à quoi vos identités peuvent accéder après leur authentification. Pour plus d'informations sur les jeux d'autorisations, consultez Jeux d'autorisations dans le Guide de l'utilisateur AWS IAM Identity Center.
- Autorisations d'utilisateur IAM temporaires : un rôle ou un utilisateur IAM peut endosser un rôle IAM pour profiter temporairement d'autorisations différentes pour une tâche spécifique.
- Accès intercompte : vous pouvez utiliser un rôle IAM pour permettre à un utilisateur (principal de confiance) d'un compte différent d'accéder aux ressources de votre compte. Les rôles constituent le principal moyen d'accorder l'accès intercompte. Toutefois, certains Services AWS vous permettent d'attacher une politique directement à une ressource (au lieu d'utiliser un rôle en tant que proxy). Pour en savoir plus sur la différence entre les rôles et les politiques basées sur les ressources pour l'accès intercompte, consultez Différence entre les rôles IAM et les politiques basées sur les ressources dans le Guide de l'utilisateur IAM.
- Accès interservices : certains Services AWS utilisent des fonctions dans d'autres Services AWS. Par exemple, lorsque vous effectuez un appel dans un service, il est courant que ce service exécute des applications dans Amazon EC2 ou stocke des objets dans Amazon S3. Un service peut le faire en utilisant les autorisations d'appel du principal, une fonction du service ou un rôle lié au service.
  - Forward access sessions (FAS) – Lorsque vous utilisez un utilisateur ou un rôle IAM pour effectuer des actions dans AWS, vous êtes considéré comme un principal. Lorsque vous utilisez certains services, vous pouvez effectuer une action qui déclenche une autre action dans un autre service. FAS utilise les autorisations du principal appelant Service AWS, combinées à la demande Service AWS pour effectuer des demandes aux services en aval. Les demandes FAS ne sont formulées que lorsqu'un service reçoit une demande qui, pour aboutir, a besoin d'interagir avec d'autres ressources ou Services AWS. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur la politique relative à la transmission de demandes FAS, consultez Sessions de transmission d'accès.
  - Fonction du service : il s'agit d'un rôle IAM attribué à un service afin de réaliser des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction du service

à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

- Rôle lié au service – Un rôle lié au service est un type de fonction du service lié à un Service AWS. Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service s'affichent dans votre Compte AWS et sont détenus par le service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

- Applications s'exécutant sur Amazon EC2 : vous pouvez utiliser un rôle IAM pour gérer des informations d'identification temporaires pour les applications s'exécutant sur une instance EC2 et effectuant des demandes d'API AWS CLI ou AWS. Cette solution est préférable au stockage des clés d'accès au sein de l'instance EC2. Pour attribuer un rôle AWS à une instance EC2 et le rendre disponible à toutes les applications associées, vous pouvez créer un profil d'instance attaché à l'instance. Un profil d'instance contient le rôle et permet aux programmes qui s'exécutent sur l'instance EC2 d'obtenir des informations d'identification temporaires. Pour plus d'informations, consultez Utilisation d'un rôle IAM pour accorder des autorisations à des applications s'exécutant sur des instances Amazon EC2 dans le Guide de l'utilisateur IAM.

Pour savoir dans quel cas utiliser des rôles ou des utilisateurs IAM, consultez Quand créer un rôle IAM (au lieu d'un utilisateur) dans le Guide de l'utilisateur IAM.

Gestion des accès à l'aide de politiques

Vous contrôlez les accès dans AWS en créant des politiques et en les attachant à des identités AWS ou à des ressources. Une politique est un objet dans AWS qui, lorsqu'il est associé à une identité ou à une ressource, définit les autorisations de ces dernières. AWS évalue ces politiques lorsqu'un principal (utilisateur, utilisateur racine ou séance de rôle) envoie une demande. Les autorisations dans les politiques déterminent si la demande est autorisée ou refusée. La plupart des politiques sont stockées dans AWS en tant que documents JSON. Pour plus d'informations sur la structure et le contenu des documents de politique JSON, consultez Présentation des politiques JSON dans le Guide de l'utilisateur IAM.

Les administrateurs peuvent utiliser les politiques JSON AWS pour spécifier qui a accès à quoi. C'est-à-dire, 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. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un

administrateur IAM peut créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Les politiques IAM définissent les autorisations d'une action, quelle que soit la méthode que vous utilisez pour exécuter l'opération. Par exemple, supposons que vous disposiez d'une politique qui autorise l'action `iam:GetRole`. Un utilisateur avec cette politique peut obtenir des informations utilisateur à partir de la AWS Management Console, de la AWS CLI ou de l'API AWS.

Politiques basées sur l'identité

Les politiques basées sur l'identité sont des documents de politique d'autorisations JSON que vous pouvez attacher à une identité telle qu'un utilisateur, un groupe d'utilisateurs ou un rôle IAM. Ces politiques contrôlent quel type d'actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Création de politiques IAM dans le Guide de l'utilisateur IAM.

Les politiques basées sur l'identité peuvent être classées comme des politiques en ligne ou des politiques gérées. Les politiques en ligne sont intégrées directement à un utilisateur, groupe ou rôle. Les politiques gérées sont des politiques autonomes que vous pouvez attacher à plusieurs utilisateurs, groupes et rôles dans votre Compte AWS. Les politiques gérées incluent les politiques gérées par AWS et les politiques gérées par le client. Pour découvrir comment choisir entre une politique gérée et une politique en ligne, consultez Choix entre les politiques gérées et les politiques en ligne dans le Guide de l'utilisateur IAM.

politiques basées sur les ressources

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Des politiques basées sur les ressources sont, par exemple, les politiques de confiance de rôle IAM et des politiques 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. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou des Services AWS.

Les politiques basées sur les ressources sont des politiques en ligne situées dans ce service. Vous ne pouvez pas utiliser les politiques gérées AWS depuis IAM dans une politique basée sur une ressource.

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, AWS WAF et Amazon VPC sont des exemples de services prenant en charge les ACL. Pour en savoir plus sur les listes de contrôle d’accès, consultez Présentation des listes de contrôle d’accès (ACL) dans le Guide du développeur Amazon Simple Storage Service.

Autres types de politique

AWS prend en charge d’autres types de politiques moins courantes. Ces types de politiques peuvent définir le nombre maximum d’autorisations qui vous sont accordées par des types de politiques plus courants.

- Limite d’autorisations : une limite d’autorisations est une fonction avancée dans laquelle vous définissez le nombre maximal d’autorisations qu’une politique basée sur l’identité peut accorder à une entité IAM (utilisateur ou rôle IAM). Vous pouvez définir une limite d’autorisations pour une entité. Les autorisations qui en résultent représentent la combinaison des politiques basées sur l’identité d’une entité et de ses limites d’autorisation. Les politiques basées sur les ressources qui spécifient l’utilisateur ou le rôle dans le champ `Principal` ne sont pas limitées par les limites d’autorisations. Un refus explicite dans l’une de ces politiques remplace l’autorisation. Pour plus d’informations sur les limites d’autorisations, consultez Limites d’autorisations pour des entités IAM dans le Guide de l’utilisateur IAM.
- Politiques de contrôle des services (SCP) - les SCP sont des politiques JSON qui spécifient le nombre maximal d’autorisations pour une organisation ou une unité d’organisation (OU) dans AWS Organizations. AWS Organizations est un service qui vous permet de regrouper et de gérer de façon centralisée plusieurs Comptes AWS détenus par votre entreprise. Si vous activez toutes les fonctions d’une organisation, vous pouvez appliquer les politiques de contrôle des services (SCP) à l’un ou à l’ensemble de vos comptes. La SCP limite les autorisations pour les entités dans les comptes membres, y compris dans chaque Utilisateur racine d'un compte AWS. Pour plus d’informations sur les organisations et les SCP, consultez Fonctionnement des SCP dans le Guide de l’utilisateur AWS Organizations.
- politiques de séance : les politiques de séance sont des politiques avancées que vous utilisez en tant que paramètre lorsque vous créez par programmation une séance temporaire pour un rôle ou un utilisateur fédéré. Les autorisations de la séance obtenue sont une combinaison des politiques

basées sur l'identité de l'utilisateur ou du rôle et des politiques de séance. Les autorisations peuvent également provenir d'une politique basée sur les ressources. Un refus explicite dans l'une de ces politiques remplace l'autorisation. Pour plus d'informations, consultez Politiques de séance dans le Guide de l'utilisateur IAM.

Plusieurs types de politique

Lorsque plusieurs types de politiques s'appliquent à la requête, les autorisations obtenues sont plus compliquées à comprendre. Pour découvrir la façon dont AWS détermine s'il convient d'autoriser une demande en présence de plusieurs types de politiques, veuillez consulter Logique d'évaluation de politiques dans le Guide de l'utilisateur IAM.

Utilisation du service géré Amazon pour Apache Flink avec IAM

Avant d'utiliser IAM pour gérer l'accès au service géré pour Apache Flink, découvrez les fonctionnalités IAM qui peuvent être utilisées avec le service géré pour Apache Flink.

Fonctionnalités IAM pouvant être utilisées avec le service géré Amazon pour Apache Flink

Fonctionnalité IAM	Prise en charge dans le service géré pour Apache Flink
Politiques basées sur l'identité	Oui
Politiques basées sur les ressources	Non
Actions de politique	Oui
Ressources de politique	Oui
Clés de condition d'une politique	Non
ACL	Non
ABAC (étiquettes dans les politiques)	Oui
Informations d'identification temporaires	Oui
Autorisations de principal	Oui

Fonctionnalité IAM	Prise en charge dans le service géré pour Apache Flink
Fonctions du service	Non
Rôles liés à un service	Non

Pour obtenir une vue d'ensemble de la façon dont le service géré pour Apache Flink et d'autres services AWS fonctionnent avec IAM, consultez les Services AWS qui fonctionnent avec IAM dans le Guide de l'utilisateur IAM.

Politiques basées sur l'identité pour le service géré pour Apache Flink

Prend en charge les politiques basées sur une identité	Oui

Les politiques basées sur l’identité sont des documents de politique d’autorisations JSON que vous pouvez attacher à une identité telle qu’un utilisateur, un Groupes d’utilisateurs IAM ou un rôle IAM. Ces politiques contrôlent quel type d’actions des utilisateurs et des rôles peuvent exécuter, sur quelles ressources et dans quelles conditions. Pour découvrir comment créer une politique basée sur l'identité, consultez Création de politiques IAM dans le Guide de l’utilisateur IAM.

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. Vous ne pouvez pas spécifier le principal dans une politique basée sur une identité car celle-ci s’applique à l’utilisateur ou au rôle auquel elle est attachée. Pour découvrir 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.

Exemples de politiques basées sur l'identité pour le service géré pour Apache Flink

Pour afficher des exemples de politiques basées sur l'identité du service géré pour Apache Flink, consultez Exemples de politiques basées sur l'identité pour le service géré Amazon pour Apache Flink.

Politiques basées sur les ressources dans le service géré pour Apache Flink

Prend en charge les politiques basées sur une ressource	Oui

Les politiques basées sur les ressources sont des documents de politique JSON que vous attachez à une ressource. Des politiques basées sur les ressources sont, par exemple, les politiques de confiance de rôle IAM et des politiques 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. Pour la ressource dans laquelle se trouve la politique, cette dernière définit quel type d'actions un principal spécifié peut effectuer sur cette ressource et dans quelles conditions. Vous devez spécifier un principal dans une politique basée sur les ressources. Les principaux peuvent inclure des comptes, des utilisateurs, des rôles, des utilisateurs fédérés ou des Services AWS.

Pour permettre un accès intercompte, vous pouvez spécifier un compte entier ou des entités IAM dans un autre compte en tant que principal dans une politique basée sur les ressources. L'ajout d'un principal entre comptes à une politique basée sur les ressources ne représente qu'une partie de l'instauration de la relation d'approbation. Quand le principal et la ressource se trouvent dans des Comptes AWS différents, un administrateur IAM dans le compte approuvé doit également accorder à l'entité principal (utilisateur ou rôle) l'autorisation d'accéder à la ressource. Pour ce faire, il attache une politique basée sur une identité à l'entité. Toutefois, si une politique basée sur des ressources accorde l'accès à un principal dans le même compte, aucune autre politique basée sur l'identité n'est requise. Pour plus d'informations, consultez Différence entre les rôles IAM et les politiques basées sur une ressource dans le Guide de l'utilisateur IAM.

Actions de politique pour le service géré pour Apache Flink

Prend en charge les actions de politique	Oui

Les administrateurs peuvent utiliser les politiques JSON AWS 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. Les actions de politique possèdent généralement le même nom

que l'opération d'API AWS associée. Il existe quelques exceptions, telles que les actions avec autorisations uniquement qui n'ont pas d'opération API correspondante. Certaines opérations nécessitent également plusieurs actions dans une politique. Ces actions supplémentaires sont nommées actions dépendantes.

Intégration d'actions dans une stratégie afin d'accorder l'autorisation d'exécuter les opérations associées.

Pour afficher la liste des actions dans le service géré pour Apache Flink, consultez Actions définies par le service géré Amazon pour Apache Flink dans Référence de l'autorisation de service.

Les actions de politique dans le service géré pour Apache Flink utilisent le préfixe suivant avant l'action :

```
Kinesis Analytics
```

Pour indiquer plusieurs actions dans une seule déclaration, séparez-les par des virgules.

```
"Action": [
      "Kinesis Analytics:action1",
      "Kinesis Analytics:action2"
         ]
```

Vous pouvez aussi spécifier plusieurs actions à l'aide de caractères génériques (*). Par exemple, pour spécifier toutes les actions qui commencent par le mot `Describe`, incluez l'action suivante :

```
"Action": "Kinesis Analytics:Describe*"
```

Pour afficher des exemples de politiques basées sur l'identité du service géré pour Apache Flink, consultez Exemples de politiques basées sur l'identité pour le service géré Amazon pour Apache Flink.

Ressources de politique pour le service géré pour Apache Flink

Prend en charge les ressources de politique	Oui

Les administrateurs peuvent utiliser les politiques JSON AWS 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 pour lesquels l'action s'applique. Les instructions doivent inclure un élément `Resource` ou `NotResource`. Il est recommandé de définir une ressource à l'aide de son Amazon Resource Name (ARN). Vous pouvez le faire pour des actions qui prennent en charge un type de ressource spécifique, connu sous la dénomination autorisations de niveau ressource.

Pour les actions qui ne sont pas compatibles avec les autorisations de niveau ressource, telles que les opérations de liste, utilisez un caractère générique (*) afin d'indiquer que l'instruction s'applique à toutes les ressources.

```
"Resource": "*"
```

Pour afficher la liste des types de ressources du service géré pour Apache Flink et leurs ARN, consultez Ressources définies par le service géré Amazon pour Apache Flink dans la Référence de l'autorisation de service. Pour savoir les actions avec lesquelles vous pouvez spécifier l'ARN de chaque ressource, consultez Actions définies par le service géré Amazon pour Apache Flink.

Pour afficher des exemples de politiques basées sur l'identité du service géré pour Apache Flink, consultez Exemples de politiques basées sur l'identité pour le service géré Amazon pour Apache Flink.

Clés de condition d'une politique pour le service géré pour Apache Flink

Prise en charge des clés de condition de stratégie spécifiques au service	Oui

Les administrateurs peuvent utiliser les politiques JSON AWS 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` (ou le bloc `Condition`) vous permet de spécifier des conditions lorsqu'une instruction est appliquée. L'élément `Condition` est facultatif. 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.

Si vous spécifiez plusieurs éléments `Condition` dans une instruction, ou plusieurs clés dans un seul élément `Condition`, AWS les évalue à l'aide d'une opération `AND` logique. Si vous spécifiez plusieurs valeurs pour une seule clé de condition, AWS évalue la condition à l'aide d'une opération `OR` logique. Toutes les conditions doivent être remplies avant que les autorisations associées à l'instruction ne soient accordées.

Vous pouvez aussi utiliser des variables d'espace réservé quand vous spécifiez des conditions. Par exemple, vous pouvez accorder à un utilisateur IAM l'autorisation d'accéder à une ressource uniquement si elle est balisée avec son nom d'utilisateur IAM. Pour plus d'informations, consultez Éléments d'une politique IAM : variables et identifications dans le Guide de l'utilisateur IAM.

AWS prend en charge les clés de condition globales et les clés de condition spécifiques à un service. Pour afficher toutes les clés de condition globales AWS, consultez Clés de contexte de condition globale AWS dans le Guide de l'utilisateur IAM.

Pour voir la liste des clés de condition pour le service géré pour Apache Flink, consultez Clés de condition pour le service géré Amazon pour Apache Flink dans la Référence de l'autorisation de service. Pour savoir avec quelles actions et ressources vous pouvez utiliser une clé de condition, consultez Actions définies par le service géré Amazon pour Apache Flink.

Pour afficher des exemples de politiques basées sur l'identité du service géré pour Apache Flink, consultez Exemples de politiques basées sur l'identité pour le service géré Amazon pour Apache Flink.

Listes de contrôle d'accès (ACL) dans le service géré pour Apache Flink

Prend en charge les listes ACL	Non

Les listes de contrôle d'accès (ACL) vérifient 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.

Contrôle d'accès par attributs (ABAC) avec le service géré pour Apache Flink

Prend en charge ABAC (étiquettes dans les politiques)	Oui

Le contrôle d’accès par attributs (ABAC) est une stratégie d’autorisation qui définit des autorisations en fonction des attributs. Dans AWS, ces attributs sont appelés étiquettes. Vous pouvez attacher des étiquettes à des entités IAM (utilisateurs ou rôles), ainsi qu’à de nombreuses ressources AWS. L’étiquetage des entités et des ressources est la première étape d’ABAC. Vous concevez ensuite des politiques ABAC pour autoriser des opérations quand l’identification du principal correspond à celle de la ressource à laquelle il tente d’accéder.

L’ABAC est utile dans les environnements qui connaissent une croissance rapide et pour les cas où la gestion des politiques devient fastidieuse.

Pour contrôler l’accès basé sur des balises, vous devez fournir les informations de balise dans l’élément de condition d’une politique utilisant les clés de condition `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` ou `aws:TagKeys`.

Si un service prend en charge les trois clés de condition pour tous les types de ressources, alors la valeur pour ce service est Oui. Si un service prend en charge les trois clés de condition pour certains types de ressources uniquement, la valeur est Partielle.

Pour plus d’informations sur l’ABAC, consultez Qu’est-ce que le contrôle d’accès basé sur les attributs (ABAC) ? dans le Guide de l’utilisateur IAM. Pour accéder à un didacticiel décrivant les étapes de configuration de l'ABAC, consultez Utilisation du contrôle d'accès par attributs (ABAC) dans le Guide de l'utilisateur IAM.

Utilisation d'informations d'identification temporaires avec le service géré pour Apache Flink

Prend en charge les informations d’identification temporaires	Oui

Certains Services AWS ne fonctionnent pas quand vous vous connectez à l’aide d’informations d’identification temporaires. Pour plus d’informations, notamment sur les Services AWS qui

fonctionnent avec des informations d'identification temporaires, consultez Services AWS qui fonctionnent avec IAM dans le Guide de l'utilisateur IAM.

Vous utilisez des informations d'identification temporaires quand vous vous connectez à la AWS Management Console en utilisant toute méthode autre qu'un nom d'utilisateur et un mot de passe. Par exemple, lorsque vous accédez à AWS en utilisant le lien d'authentification unique (SSO) de votre société, ce processus crée automatiquement des informations d'identification temporaires. Vous créez également automatiquement des informations d'identification temporaires lorsque vous vous connectez à la console en tant qu'utilisateur, puis changez de rôle. Pour plus d'informations sur le changement de rôle, consultez Changement de rôle (console) dans le Guide de l'utilisateur IAM.

Vous pouvez créer manuellement des informations d'identification temporaires à l'aide d'AWS CLI ou de l'API AWS. Vous pouvez ensuite utiliser ces informations d'identification temporaires pour accéder à AWS. AWS recommande de générer des informations d'identification temporaires de façon dynamique au lieu d'utiliser des clés d'accès à long terme. Pour plus d'informations, consultez Informations d'identification de sécurité temporaires dans IAM.

Autorisations de principal entre services pour le service géré pour Apache Flink

Prend en charge les sessions d'accès direct (FAS)	Oui

Lorsque vous vous servez d'un utilisateur IAM ou d'un rôle IAM pour accomplir des actions dans AWS, vous êtes considéré comme un principal. Lorsque vous utilisez certains services, l'action que vous effectuez est susceptible de lancer une autre action dans un autre service. FAS utilise les autorisations du principal appelant Service AWS, combinées à la demande Service AWS pour effectuer des demandes aux services en aval. Les demandes FAS ne sont formulées que lorsqu'un service reçoit une demande qui, pour aboutir, a besoin d'interagir avec d'autres ressources ou Services AWS. Dans ce cas, vous devez disposer d'autorisations nécessaires pour effectuer les deux actions. Pour plus de détails sur une politique lors de la formulation de demandes FAS, consultez Transmission des sessions d'accès.

Fonctions du service pour le service géré pour Apache Flink

Prend en charge les fonctions du service	Oui

Une fonction du service est un rôle IAM qu'un service endosse pour accomplir des actions en votre nom. Un administrateur IAM peut créer, modifier et supprimer une fonction du service à partir d'IAM. Pour plus d'informations, consultez Création d'un rôle pour la délégation d'autorisations à un Service AWS dans le Guide de l'utilisateur IAM.

> **Warning**
>
> La modification des autorisations d'une fonction du service peut altérer la fonctionnalité du service géré pour Apache Flink. Ne modifiez des fonctions du service que quand le service géré pour Apache Flink vous le conseille.

Rôles liés à un service pour le service géré pour Apache Flink

Prend en charge les rôles liés à un service.	Oui

Un rôle lié à un service est un type de fonction du service lié à un Service AWS. Le service peut endosser le rôle afin d'effectuer une action en votre nom. Les rôles liés à un service s'affichent dans votre Compte AWS et sont détenus par le service. Un administrateur IAM peut consulter, mais ne peut pas modifier, les autorisations concernant les rôles liés à un service.

Pour plus d'informations sur la création ou la gestion des rôles liés à un service, consultez Services AWS qui fonctionnent avec IAM. Recherchez un service dans le tableau qui inclut un `Yes` dans la colonne Rôle lié à un service. Choisissez le lien Oui pour consulter la documentation du rôle lié à ce service.

Exemples de politiques basées sur l'identité pour le service géré Amazon pour Apache Flink

Par défaut, les utilisateurs et les rôles ne sont pas autorisés à créer ou à modifier des ressources du service géré pour Apache Flink. Ils ne peuvent pas non plus exécuter des tâches à l'aide de la AWS Management Console, de l'AWS Command Line Interface (AWS CLI) ou de l'API AWS. Pour octroyer aux utilisateurs des autorisations d'effectuer des actions sur les ressources dont ils ont besoin, un administrateur IAM doit créer des politiques IAM. L'administrateur peut ensuite ajouter les politiques IAM aux rôles et les utilisateurs peuvent assumer les rôles.

Pour apprendre à créer une politique basée sur l'identité IAM à l'aide de ces exemples de documents de politique JSON, consultez Création de politiques IAM dans le Guide de l'utilisateur IAM.

Pour plus de détails sur les actions et les types de ressources définis par le service géré pour Apache Flink, y compris le format des ARN pour chacun des types de ressources, consultez Actions, ressources et clés de condition pour le service géré Amazon pour Apache Flink dans la Référence de l'autorisation de service.

Rubriques

- Bonnes pratiques en matière de politiques
- Utilisation de la console du service géré pour Apache Flink
- Autorisation accordée aux utilisateurs pour afficher leurs propres autorisations

Bonnes pratiques en matière de politiques

Les stratégies basées sur l'identité déterminent si une personne peut créer, consulter ou supprimer des ressources du service géré pour Apache Flink dans 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 :

- Démarrer avec AWS gérées et évoluez vers les autorisations de moindre privilège - Pour commencer à accorder des autorisations à vos utilisateurs et charges de travail, utilisez les politiques gérées AWS qui accordent des autorisations dans de nombreux cas d'utilisation courants. Elles sont disponibles dans votre Compte AWS. Nous vous recommandons de réduire encore les autorisations en définissant des politiques gérées par le client AWS qui sont spécifiques à vos cas d'utilisation. Pour de plus amples informations, consultez AWS Politiques gérées ou AWS Politiques gérées pour les activités professionnelles dans le Guide de l'utilisateur IAM.
- Accorder 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.
- Utiliser 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 via un Service AWS spécifique, comme AWS CloudFormation. Pour plus d'informations, consultez Conditions pour éléments de politique JSON IAM dans le Guide de l'utilisateur IAM.

- Utilisez IAM Access Analyzer pour valider vos politiques IAM afin de garantir des autorisations sécurisées et fonctionnelles - IAM Access Analyzer 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 politique IAM Access Analyzer dans le Guide de l'utilisateur IAM.
- Authentification multifactorielle (MFA) nécessaire : si vous avez un scénario qui nécessite des utilisateurs IAM ou un utilisateur root dans votre Compte AWS, activez l'authentification multifactorielle pour une sécurité renforcée. Pour exiger le MFA lorsque des opérations d'API sont appelées, ajoutez des conditions MFA à vos politiques. Pour plus d'informations, consultez Configuration de l'accès aux API protégé par 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 du service géré pour Apache Flink

Pour accéder à la console du service géré pour Apache Flink, vous devez disposer d'un ensemble minimum d'autorisations. Ces autorisations doivent vous permettre de répertorier et de consulter des informations sur les ressources du service géré pour Apache Flink dans votre Compte AWS. Si vous créez une stratégie basée sur l'identité qui est plus restrictive que l'ensemble minimum d'autorisations requis, la console ne fonctionnera pas comme prévu pour les entités (utilisateurs ou rôles) tributaires de cette stratégie.

Vous n'avez pas besoin d'accorder les autorisations minimales de console pour les utilisateurs qui effectuent des appels uniquement à AWS CLI ou à l'API AWS. Autorisez plutôt l'accès à uniquement aux actions qui correspondent à l'opération d'API qu'ils tentent d'effectuer.

Pour vous assurer que les utilisateurs et les rôles peuvent continuer à utiliser la console du service géré pour Apache Flink, attachez également la politique gérée du service géré pour Apache Flink `ConsoleAccess` ou `ReadOnly` AWS aux entités. Pour plus d'informations, consultez Ajout d'autorisations à un utilisateur dans le Guide de l'utilisateur IAM.

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 nécessaires pour réaliser cette action sur la console ou par programmation à l'aide de l'interface AWS CLI ou de l'API 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": "*"
        }
    ]
}
```

Résolution des problèmes liés à l'identité et aux accès dans le service géré Amazon pour Apache Flink

Utilisez les informations suivantes pour identifier et résoudre les problèmes courants que vous pouvez rencontrer lorsque vous travaillez avec le service géré pour Apache Flink et IAM.

Rubriques

- Je ne suis pas autorisé à effectuer une action dans le service géré pour Apache Flink
- Je ne suis pas autorisé à effectuer iam : PassRole
- Je souhaite autoriser des personnes n'appartenant pas à mon compte AWS à accéder aux ressources de mon service géré pour Apache Flink.

Je ne suis pas autorisé à effectuer une action dans le service géré pour Apache Flink

Si la AWS Management Console indique que vous n'êtes pas autorisé à exécuter 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 quand l'utilisateur `mateojackson` tente d'utiliser la console pour afficher des informations détaillées sur une ressource *`my-example-widget`* fictive, mais ne dispose pas des autorisations `Kinesis Analytics:`*`GetWidget`* fictives.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: Kinesis
 Analytics:GetWidget on resource: my-example-widget
```

Dans ce cas, Mateo demande à son administrateur de mettre à jour ses politiques pour lui permettre d'accéder à la ressource *`my-example-widget`* à l'aide de l'action `Kinesis Analytics:`*`GetWidget`*.

Je ne suis pas autorisé à effectuer iam : PassRole

Si vous recevez une erreur selon laquelle vous n'êtes pas autorisé à exécuter l'action `iam:PassRole`, vos politiques doivent être mises à jour pour vous permettre de transmettre un rôle au service géré pour Apache Flink.

Certains Services AWS vous permettent de transmettre un rôle existant à ce service, au lieu de créer une nouvelle fonction du service ou rôle lié à un service. Pour ce faire, un utilisateur doit 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 le service géré pour Apache Flink. Toutefois, l'action nécessite que le service ait des autorisations accordées par une fonction du service. Mary ne dispose pas des autorisations nécessaires pour transférer 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 encore besoin d'aide, contactez votre administrateur AWS. Votre administrateur vous a fourni vos informations de connexion.

Je souhaite autoriser des personnes n'appartenant pas à mon compte AWS à accéder aux ressources de mon service géré pour Apache Flink.

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 en savoir plus, consultez les éléments suivants :

- Pour savoir si le service géré pour Apache Flink prend en charge ces fonctionnalités, consultez Utilisation du service géré Amazon pour Apache Flink avec IAM.
- Pour savoir comment octroyer l'accès à vos ressources à des Comptes AWS dont vous êtes propriétaire, consultez la section Fournir l'accès à un utilisateur IAM dans un autre Compte AWS que vous possédez dans le Guide de l'utilisateur IAM.
- Pour savoir comment octroyer l'accès à vos ressources à des tiers Comptes AWS, consultez Fournir l'accès aux Comptes AWS appartenant à 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 découvrir quelle est la différence entre l'utilisation des rôles et l'utilisation des politiques basées sur les ressources pour l'accès entre comptes, consultez Différence entre les rôles IAM et les politiques basées sur les ressources dans le Guide de l'utilisateur IAM.

Prévention du problème de l'adjoint confus entre services

Dans AWS, l'usurpation d'identité entre services peut se produire lorsqu'un service (le service appelant) appelle un autre service (le service appelé). Le service appelant peut être manipulé pour agir sur les ressources d'un autre client, même s'il ne devrait pas avoir les autorisations appropriées, ce qui se traduit par un problème d'adjoint confus.

Pour éviter cela, AWS fournit des outils qui vous aident à protéger vos données pour tous les services utilisant des principaux de service qui ont eu accès aux ressources de votre compte. Cette section se concentre sur la prévention du problème de l'adjoint confus entre services spécifique au service géré pour Apache Flink. Cependant, vous pouvez en savoir plus à ce sujet dans la section Le problème de l'adjoint confus du Guide de l'utilisateur IAM.

Dans le contexte du service géré pour Apache Flink, nous vous recommandons d'utiliser les clés de contexte de condition SourceAccount globale aws : SourceArn et aws : dans votre politique de confiance en matière de rôle afin de limiter l'accès au rôle aux seules demandes générées par les ressources attendues.

Utilisez `aws:SourceArn` si vous souhaitez qu'une seule ressource soit associée à l'accès entre services. Utilisez `aws:SourceAccount` si vous souhaitez autoriser l'association d'une ressource de ce compte à l'utilisation interservices.

La valeur de la clé `aws:SourceArn` doit être l'ARN de la ressource utilisée par le service géré pour Apache Flink, qui est spécifié au format suivant : `arn:aws:kinesisanalytics:region:account:resource`.

Le moyen le plus efficace de se protéger contre le problème d'adjoint confus consiste à utiliser la clé de contexte de condition globale `aws:SourceArn` avec l'ARN complet de la ressource.

Si vous ne connaissez pas l'ARN complet de la ressource ou si vous indiquez plusieurs ressources, utilisez la clé `aws:SourceArn` avec des caractères génériques (*) pour les parties inconnues de l'ARN. Par exemple : `arn:aws:kinesisanalytics::111122223333:*`.

Les politiques relatives aux rôles que vous fournissez au service géré pour Apache Flink ainsi que les politiques d'approbation relatives aux rôles générés pour vous peuvent utiliser ces clés.

Afin de vous protéger contre le problème d'adjoint confus, effectuez les tâches suivantes :

Pour lutter contre le problème de l'adjoint confus

1. Connectez-vous à la console de gestion AWS et ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Rôles, puis choisissez le rôle que vous souhaitez modifier.
3. Choisissez Modifier la politique.
4. Sur la page Modifier la stratégie d'approbation, remplacez la stratégie JSON par défaut par une stratégie qui utilise l'une des clés de contexte de condition globale `aws:SourceArn` et `aws:SourceAccount` ou les deux. Voir l'exemple de stratégie suivant :
5. Choisissez Mettre à jour une politique.

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Principal":{
            "Service":"kinesisanalytics.amazonaws.com"
         },
         "Action":"sts:AssumeRole",
         "Condition":{
            "StringEquals":{
               "aws:SourceAccount":"Account ID"
            },
            "ArnEquals":{
               "aws:SourceArn":"arn:aws:kinesisanalytics:us-
east-1:123456789012:application/my-app"
            }
         }
      }
   ]
}
```

Surveillance dans le service géré pour Apache Flink

Le service géré pour Apache Flink fournit des fonctionnalités de surveillance pour vos applications. Pour plus d'informations, consultez Journalisation et surveillance.

Validation de la conformité pour le service géré pour Apache Flink

Des auditeurs tiers évaluent la sécurité et la conformité du service géré Amazon pour Apache Flink dans le cadre de plusieurs programmes de conformité AWS. Il s'agit notamment des certifications SOC, PCI, HIPAA.

Pour obtenir la liste des services concernés par des programmes de conformité spécifiques, consultez . Pour obtenir des informations générales, consultez Programmes de conformité AWS.

Vous pouvez télécharger les rapports de l'audit externe avec AWS Artifact. Pour plus d'informations, consultez Téléchargement des rapports dans AWS Artifact.

Votre responsabilité de conformité lors de l'utilisation du service géré pour Apache Flink est déterminée par la sensibilité de vos données, les objectifs de conformité de votre entreprise, ainsi que par la législation et la réglementation applicables. Si votre utilisation du service géré pour Apache Flink est soumise à la conformité aux normes HIPAA ou PCI, fournit des ressources pour vous aider :

- Guides Quick Start de la sécurité et de la conformité : ces guides de déploiement traitent de considérations architecturales et indiquent les étapes à suivre pour déployer des environnements de référence centrés sur la sécurité et la conformité dans AWS.
- Architecting for HIPAA Security and Compliance on Amazon Web Services Ce livre blanc décrit comment les entreprises peuvent utiliser AWS pour créer des applications conformes à la loi HIPAA.
- Ressources de conformité AWS : cet ensemble de manuels et de guides peut s’appliquer à votre secteur et à votre emplacement.
- AWS Config : ce service AWS permet d’évaluer la conformité des configurations de vos ressources par rapport à des pratiques internes, règlementations et autres directives sectorielles.
- AWS Security Hub : ce service AWS fournit une vue complète de votre état de sécurité au sein d’AWS qui vous permet de vérifier votre conformité aux normes du secteur et aux bonnes pratiques de sécurité.

FedRAMP

Le programme de conformité FedRAMP d'AWS inclut le service géré pour Apache Flink comme service autorisé pour FedRAMP. Si vous êtes un client fédéral ou commercial, vous pouvez utiliser le service pour traiter et stocker des charges de travail sensibles dans la limite d'autorisation de la région AWS GovCloud (États-Unis) contenant des données jusqu'à un niveau d'impact élevé, ainsi

que dans les régions USA Est (Virginie du Nord), USA Est (Ohio), USA Ouest (Californie du Nord), USA Ouest (Oregon) avec des données jusqu'à un niveau modéré.

Vous pouvez demander à accéder aux packages de sécurité FedRAMP d'AWS via le PMO FedRAMP ou le responsable de votre compte AWS. Vous pouvez également les télécharger sur AWS Artifact à l'adresse AWS Artifact.

Pour plus d'informations, consultez FedRAMP.

Résilience dans le service géré Amazon pour Apache Flink

L'infrastructure mondiale AWS s'articule autours de régions et de zones de disponibilité AWS. AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, reliées par un réseau à latence faible, 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.

Pour plus d'informations sur les régions et les zones de disponibilité AWS, consultez AWS Infrastructure mondiale.

Outre l'infrastructure globale AWS, le service géré pour Apache Flink propose plusieurs fonctionnalités qui contribuent à la prise en charge des vos besoins en matière de résilience et de sauvegarde de données.

Reprise après sinistre

Le service géré pour Apache Flink s'exécute en mode sans serveur et s'occupe des dégradations de l'hôte, de la disponibilité des zones de disponibilité et d'autres problèmes liés à l'infrastructure en effectuant une migration automatique. Le service géré pour Apache Flink atteint cet objectif grâce à de multiples mécanismes redondants. Chaque application du service géré pour Apache Flink s'exécute dans un cluster Apache Flink à locataire unique. Le cluster Apache Flink est exécuté en mode haute disponibilité JobMananger à l'aide de Zookeeper sur plusieurs zones de disponibilité. Le service géré pour Apache Flink déploie Apache Flink à l'aide d'Amazon EKS. Plusieurs pods Kubernetes sont utilisés dans Amazon EKS pour chaque région AWS dans les zones de disponibilité. En cas d'échec, le service géré pour Apache Flink essaie d'abord de récupérer l'application au sein du cluster Apache Flink en cours d'exécution en utilisant les points de contrôle de votre application, s'ils sont disponibles.

Le service géré pour Apache Flink sauvegarde l'état de l'application à l'aide de points de contrôle et d'instantanés :

- Les points de contrôle sont des sauvegardes de l'état de l'application que le service géré pour Apache Flink crée automatiquement de façon périodique et utilise pour restaurer les données en cas de panne.
- Les instantanés sont des sauvegardes de l'état de l'application que vous créez et restaurez manuellement.

Pour en savoir plus sur les points de contrôle et les instantanés, consultez Tolérance aux pannes.

Gestion des versions

Les versions stockées de l'état de l'application sont gérées comme suit :

- Les points de contrôle sont automatiquement versionnés par le service. Si le service utilise un point de contrôle pour redémarrer l'application, le dernier point de contrôle sera utilisé.
- Les points de sauvegarde sont versionnés à l'aide du SnapshotNameparamètre de l'CreateApplicationSnapshotaction.

Le service géré pour Apache Flink chiffre les données stockées dans les points de contrôle et de sauvegarde.

Sécurité de l'infrastructure du service géré Amazon pour Apache Flink

En tant que service géré, le service géré pour Apache Flink est protégé par les procédures de sécurité du réseau mondial AWS qui sont décrites dans le livre blanc Amazon Web Services : présentation des processus de sécurité.

Vous utilisez les appels d'API publiés par AWS pour accéder au service géré pour Apache Flink via le réseau. Tous les appels d'API au service géré pour Apache Flink sont sécurisés via le protocole TLS (Transport Layer Security) et authentifiés via IAM. Les clients doivent prendre en charge le protocole TLS 1.2 ou une version ultérieure. Les clients doivent aussi prendre en charge les suites de chiffrement PFS (Perfect Forward Secrecy) comme Ephemeral Diffie-Hellman (DHE) ou Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La plupart des systèmes modernes tels que Java 7 et les versions ultérieures prennent en charge ces modes.

En outre, les demandes doivent être signées à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Bonnes pratiques de sécurité pour le service géré pour Apache Flink

Le service géré Amazon pour Apache Flink pour la recherche offre 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.

Implémentation d'un accès sur la base du moindre privilège

Lorsque vous accordez des autorisations, vous sélectionnez qui obtient les autorisations pour telles ou telles ressources du service géré pour Apache Flink. Vous activez des actions spécifiques que vous souhaitez autoriser sur ces ressources. Par conséquent, vous devez accorder uniquement les autorisations qui sont requises pour exécuter une tâche. L'implémentation d'un accès sur la base du moindre privilège est fondamentale pour réduire les risques en matière de sécurité et l'impact que pourraient avoir des d'erreurs ou des actes de malveillance.

Utilisation de rôles IAM pour accéder à d'autres services Amazon

Votre application de service géré pour Apache Flink doit disposer d'informations d'identification valides pour accéder aux ressources dans d'autres services, comme les flux de données Kinesis, les flux Kinesis Data Firehose ou les compartiments Amazon S3. Vous ne devez pas stocker les informations d'identification AWS directement dans l'application ni dans un compartiment Amazon S3. Il s'agit d'autorisations à long terme qui ne font pas automatiquement l'objet d'une rotation et qui pourraient avoir un impact commercial important si elles étaient compromises.

Vous devez plutôt utiliser un rôle IAM pour gérer des informations d'identification temporaires afin que votre application accède à d'autres ressources. Lorsque vous utilisez un rôle, vous n'avez pas à utiliser d'informations d'identification à long terme pour accéder à d'autres ressources.

Pour plus d'informations, veuillez consulter les rubriques suivantes dans le Guide de l'utilisateur IAM :

- Rôles IAM
- Scénarios courants pour les rôles : utilisateurs, applications et services.

Implémentation d'un chiffrement côté serveur dans des ressources dépendantes

Les données au repos et les données en transit sont chiffrées dans le service géré pour Apache Flink, et ce chiffrement ne peut pas être désactivé. Vous devez implémenter un chiffrement côté serveur dans vos ressources dépendantes, comme les flux de données Kinesis, les flux Kinesis Data Firehose et les compartiments Amazon S3. Pour plus d'informations sur l'implémentation du chiffrement côté serveur dans les ressources dépendantes, reportez-vous à Protection des données.

CloudTrail À utiliser pour surveiller les appels d'API

Le service géré pour Apache Flink est intégré à AWS CloudTrail, un service qui enregistre les actions effectuées par un utilisateur, un rôle ou un service Amazon dans le service géré pour Apache Flink.

À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite au service géré pour Apache Flink, l'adresse IP à partir de laquelle la demande a été faite, qui a fait la demande, quand elle a été faite et des informations supplémentaires.

Pour plus d’informations, consultez the section called “Utiliser AWS CloudTrail”.

Journalisation et surveillance dans le service géré Amazon pour Apache Flink

La surveillance est essentielle pour assurer la fiabilité, la disponibilité et les performances de vos applications de service géré pour Apache Flink. Vous devez recueillir les données de surveillance de toutes les parties de votre solution AWS afin de pouvoir déboguer plus facilement une éventuelle défaillance à plusieurs points.

Avant de commencer la surveillance du service géré pour Apache Flink, vous devez créer un plan de surveillance qui contient les réponses aux questions suivantes :

- Quels sont les objectifs de la surveillance ?
- Quelles sont les ressources à surveiller ?
- À quelle fréquence les ressources doivent-elles être surveillées ?
- Quels outils de surveillance utiliser ?
- Qui exécute les tâches de supervision ?
- Qui doit être informé en cas de problème ?

La prochaine étape consiste à établir une base pour les performances normales du service géré pour Apache Flink dans votre environnement. Pour cela, vous devez mesurer les performances à différents moments et sous différentes conditions de charge. Lorsque vous surveillez le service géré pour Apache Flink, vous pouvez stocker des données de surveillance historiques. Vous pouvez ainsi les comparer avec les données de performances actuelles, identifier des modèles de performances normales et des anomalies de performances, ainsi que concevoir des méthodes pour les résoudre.

Rubriques

- Journalisation
- Surveillance
- Configuration de la journalisation des applications
- Analyse des journaux avec CloudWatch Logs Insights
- Affichage des métriques et des dimensions dans le service géré pour Apache Flink
- Écrire des messages personnalisés dans les CloudWatch journaux
- Appels d'API pour la journalisation du service géré pour Apache Flink avec AWS CloudTrail

Journalisation

La journalisation est importante pour permettre aux applications de production de comprendre les erreurs et les défaillances. Cependant, le sous-système de journalisation doit collecter et transférer les entrées du journal vers les CloudWatch journaux. Bien qu'une certaine journalisation soit acceptable et souhaitable, une journalisation prolongée peut surcharger le service et entraîner un retard de l'application Flink. Enregistrer les exceptions et les avertissements est certainement une bonne idée. Mais vous ne pouvez pas générer de message journal pour chaque message traité par l'application Flink. Flink est optimisé pour une latence élevée et une faible latence, ce qui n'est pas le cas du sous-système de journalisation. S'il est vraiment nécessaire de générer une sortie de journal pour chaque message traité, utilisez un récepteur supplémentaire DataStream dans l'application Flink et un récepteur approprié pour envoyer les données à Amazon S3 ou CloudWatch. N'utilisez pas le système de journalisation Java à cette fin. De plus, le paramètre`Debug Monitoring Log Level` du service géré pour Apache Flink génère un trafic important, ce qui peut créer une contre-pression. Vous ne devez l'utiliser que lorsque vous étudiez activement les problèmes liés à l'application.

Interrogation des journaux avec CloudWatch Logs Insights

CloudWatch Logs Insights est un puissant service permettant d'interroger les journaux à grande échelle. Les clients doivent tirer parti de ses capacités pour effectuer rapidement des recherches dans les journaux afin d'identifier et de limiter les erreurs lors d'événements opérationnels.

La requête suivante recherche les exceptions dans tous les journaux du gestionnaire de tâches et les classe en fonction de l'heure à laquelle elles se sont produites.

```
fields @timestamp, @message
| filter isPresent(throwableInformation.0) or isPresent(throwableInformation) or
 @message like /(Error|Exception)/
| sort @timestamp desc
```

Pour d'autres requêtes utiles, consultez la section Exemples de requêtes.

Surveillance

Lorsque vous exécutez des applications de streaming en production, vous souhaitez exécuter l'application en continu et indéfiniment. Il est crucial de mettre en œuvre une surveillance et une alarme appropriées de tous les composants, et pas seulement de l'application Flink. Sinon, vous risquez de passer à côté des problèmes émergents dès le début et de ne vous rendre compte d'un

événement opérationnel trop tard, quand il est beaucoup plus difficile de les atténuer. Les éléments généraux à surveiller incluent :

- La source ingère-t-elle des données ?
- Les données sont-elles lues depuis la source (du point de vue de la source) ?
- L'application Flink reçoit-elle des données ?
- L'application Flink parvient-elle à suivre le rythme ou prend-elle du retard ?
- L'application Flink permet-elle de conserver les données dans le récepteur (du point de vue de l'application) ?
- Le récepteur reçoit-il des données ?

Des métriques plus spécifiques doivent ensuite être prises en compte pour l'application Flink. Ce CloudWatch tableau de bord constitue un bon point de départ. Pour plus d'informations sur les métriques à surveiller pour les applications de production, consultez Utilisation des CloudWatch alarmes avec Amazon Managed Service pour Apache Flink. Ces métriques incluent :

- records_lag_max et millisbehindLatest : si l'application consomme depuis Kinesis ou Kafka, ces métriques indiquent si l'application prend du retard et doit être redimensionnée afin de suivre le rythme de charge actuel. Il s'agit d'une bonne métrique générique facile à suivre pour tous les types d'applications. Mais elle ne peut être utilisée que pour une mise à l'échelle réactive, c'est-à-dire lorsque l'application a déjà pris du retard.
- CPUUtilization heapMemoryUtilizationet — Ces mesures donnent une bonne indication de l'utilisation globale des ressources de l'application et peuvent être utilisées pour un dimensionnement proactif, sauf si l'application est liée aux E/S.
- downtime : un temps d'arrêt supérieur à zéro indique une défaillance de l'application. Si la valeur est supérieure à 0, l'application ne traite aucune donnée.
- lastCheckpointSizeet lastCheckpointDuration— Ces indicateurs surveillent la quantité de données stockées en état et le temps nécessaire pour passer un point de contrôle. Si les points de contrôle augmentent ou prennent du temps, l'application passe continuellement du temps à effectuer les points de contrôle et réduit le nombre de cycles de traitement réels. À certains moments, les points de contrôle peuvent devenir trop grands ou prendre tellement de temps qu'ils échouent. En plus de surveiller les valeurs absolues, les clients devraient également envisager de surveiller le taux de variation avec `RATE(lastCheckpointSize)` et `RATE(lastCheckpointDuration)`.
- numberOfFailedPoints de contrôle : cette métrique compte le nombre de points de contrôle ayant échoué depuis le démarrage de l'application. Selon l'application, il peut être tolérable que les points

de contrôle échouent de temps en temps. Mais si les points de contrôle échouent régulièrement, l'application est probablement malsaine et nécessite une attention accrue. Nous recommandons de surveiller `RATE(numberOfFailedCheckpoints)` pour être alerté en fonction de la pente et non en fonction des valeurs absolues.

Configuration de la journalisation des applications

En ajoutant une option de CloudWatch journalisation Amazon à votre application Managed Service for Apache Flink, vous pouvez surveiller les événements liés à l'application ou les problèmes de configuration.

Cette rubrique décrit comment configurer votre application pour écrire les événements de l'application dans un flux CloudWatch Logs. Une option de CloudWatch journalisation est un ensemble de paramètres et d'autorisations d'application que votre application utilise pour configurer la manière dont elle écrit les événements de l'application dans les CloudWatch journaux. Vous pouvez ajouter et configurer une option de CloudWatch journalisation en utilisant le AWS Management Console ou le AWS Command Line Interface (AWS CLI).

Notez ce qui suit à propos de l'ajout d'une option de CloudWatch journalisation à votre application :

- Lorsque vous ajoutez une option de CloudWatch journalisation à l'aide de la console, Managed Service for Apache Flink crée le groupe de CloudWatch journaux et le flux de journaux pour vous et ajoute les autorisations dont votre application a besoin pour écrire dans le flux de journaux.
- Lorsque vous ajoutez une option de CloudWatch journalisation à l'aide de l'API, vous devez également créer le groupe de journaux et le flux de journaux de l'application, et ajouter les autorisations dont votre application a besoin pour écrire dans le flux de journaux.

Cette rubrique contient les sections suivantes :

- Configuration de la CloudWatch journalisation à l'aide de la console
- Configuration de la CloudWatch journalisation à l'aide de la CLI
- Niveaux de surveillance des applications
- Bonnes pratiques de journalisation
- Journalisation de dépannage
- Étape suivante

Configuration de la CloudWatch journalisation à l'aide de la console

Lorsque vous activez la CloudWatch journalisation pour votre application dans la console, un groupe de CloudWatch journaux et un flux de journaux sont créés pour vous. De plus, la politique d'autorisation de votre application est mise à jour avec les autorisations d'écriture dans le flux.

Le service géré pour Apache Flink crée un groupe de journaux nommé selon la convention suivante, où *ApplicationName* est le nom de votre application.

```
/AWS/KinesisAnalytics/ApplicationName
```

Le service géré pour Apache Flink crée un flux de journaux dans le nouveau groupe de journaux avec le nom suivant.

```
kinesis-analytics-log-stream
```

Vous définissez le niveau des métriques de surveillance de l'application et le niveau du journal de surveillance à l'aide de la section Niveau du journal de surveillance de la page Configurer l'application. Pour plus d'informations sur les journaux d'application, consultez the section called “Niveaux de surveillance des applications”.

Configuration de la CloudWatch journalisation à l'aide de la CLI

Pour ajouter une option de CloudWatch journalisation à l'aide duAWS CLI, procédez comme suit :

- Créez un groupe de CloudWatch journaux et un flux de journaux.
- Ajoutez une option de journalisation lorsque vous créez une application à l'aide de l'CreateApplicationaction, ou ajoutez une option de journalisation à une application existante à l'aide de l'AddApplicationCloudWatchLoggingOptionaction.
- Ajoutez des autorisations à la politique de votre application pour écrire dans les journaux.

Cette section contient les rubriques suivantes :

- Création d'un groupe de CloudWatch journaux et d'un flux de journaux
- Utilisation des options de CloudWatch journalisation des applications
- Ajouter des autorisations d'écriture dans le flux de CloudWatch log

Création d'un groupe de CloudWatch journaux et d'un flux de journaux

Vous créez un groupe de CloudWatch journaux et diffusez à l'aide de la console CloudWatch Logs ou de l'API. Pour plus d'informations sur la création d'un groupe de CloudWatch journaux et d'un flux de journaux, consultez la section Utilisation des groupes de journaux et des flux de journaux.

Utilisation des options de CloudWatch journalisation des applications

Utilisez les actions d'API suivantes pour ajouter une option de CloudWatch journal à une application nouvelle ou existante ou pour modifier une option de journal pour une application existante. Pour plus d'informations sur l'utilisation d'un fichier JSON comme entrée pour une action d'API, veuillez consulter Exemple de code pour d'API pour le service géré pour Apache Flink.

Ajouter une option de CloudWatch journal lors de la création d'une application

L'exemple suivant montre comment utiliser l'`CreateApplication`action pour ajouter une option de CloudWatch journal lorsque vous créez une application. Dans l'exemple, remplacez *Amazon Resource Name (ARN) du flux de CloudWatch log à ajouter à la nouvelle application* par vos propres informations. Pour plus d'informations sur cette action, consultez `CreateApplication`.

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "test-application-description",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation":{
                            "BucketARN": "arn:aws:s3:::mybucket",
                            "FileKey": "myflink.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        }
    },
    "CloudWatchLoggingOptions": [{
      "LogStreamARN": "<Amazon Resource Name (ARN) of the CloudWatch log stream to add
 to the new application>"
 }]
}
```

Ajouter une option de CloudWatch journalisation à une application existante

L'exemple suivant montre comment utiliser l'`AddApplicationCloudWatchLoggingOption`action pour ajouter une option de CloudWatch journalisation à une application existante. Dans l'exemple, remplacez chaque *`espace réservé pour l'entrée utilisateur`* par vos propres informations. Pour plus d'informations sur cette action, consultez [AddApplicationCloudWatchLoggingOption](#).

```
{
   "ApplicationName": "<Name of the application to add the log option to>",
   "CloudWatchLoggingOption": {
      "LogStreamARN": "<ARN of the log stream to add to the application>"
   },
   "CurrentApplicationVersionId": <Version of the application to add the log to>
}
```

Mettre à jour une option de CloudWatch journal existante

L'exemple suivant montre comment utiliser l'`UpdateApplication`action pour modifier une option de CloudWatch journal existante. Dans l'exemple, remplacez chaque *`espace réservé pour l'entrée utilisateur`* par vos propres informations. Pour plus d'informations sur cette action, consultez [UpdateApplication](#).

```
{
   "ApplicationName": "<Name of the application to update the log option for>",
   "CloudWatchLoggingOptionUpdates": [
         {
            "CloudWatchLoggingOptionId": "<ID of the logging option to modify>",
            "LogStreamARNUpdate": "<ARN of the new log stream to use>"
         }
      ],
   "CurrentApplicationVersionId": <ID of the application version to modify>
}
```

Supprimer une option de CloudWatch journalisation d'une application

L'exemple suivant montre comment utiliser l'`DeleteApplicationCloudWatchLoggingOption`action pour supprimer une option de

CloudWatch journal existante. Dans l'exemple, remplacez chaque *espace réservé pour l'entrée utilisateur* par vos propres informations. Pour plus d'informations sur cette action, consultez DeleteApplicationCloudWatchLoggingOption.

```
{
   "ApplicationName": "<Name of application to delete log option from>",
   "CloudWatchLoggingOptionId": "<ID of the application log option to delete>",
   "CurrentApplicationVersionId": <Version of the application to delete the log option
 from>
}
```

Configuration du niveau de journalisation de l'application

Pour définir le niveau de journalisation de l'application, utilisez le paramètre MonitoringConfiguration de l'action CreateApplication ou le paramètre MonitoringConfigurationUpdate de l'action UpdateApplication.

Pour plus d'informations sur les journaux d'application, consultez the section called "Niveaux de surveillance des applications".

Définissez le niveau de journalisation de l'application lors de la création d'une application

L'exemple de demande d'action CreateApplication suivant définit le niveau du journalisation de l'application sur INFO.

```
{
   "ApplicationName": "MyApplication",
   "ApplicationDescription": "My Application Description",
   "ApplicationConfiguration": {
      "ApplicationCodeConfiguration":{
      "CodeContent":{
        "S3ContentLocation":{
          "BucketARN":"arn:aws:s3:::mybucket",
          "FileKey":"myflink.jar",
          "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "CodeContentType":"ZIPFILE"
      },
      "FlinkApplicationConfiguration":
```

```
         "MonitoringConfiguration": {
            "ConfigurationType": "CUSTOM",
            "LogLevel": "INFO"
         }
      },
   "RuntimeEnvironment": "FLINK-1_15",
   "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole"
}
```

Mettre à jour le niveau de journalisation de l'application

L'exemple de demande d'action `UpdateApplication` suivant définit le niveau du journalisation de l'application sur `INFO`.

```
{
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "MonitoringConfigurationUpdate": {
            "ConfigurationTypeUpdate": "CUSTOM",
            "LogLevelUpdate": "INFO"
         }
      }
   }
}
```

Ajouter des autorisations d'écriture dans le flux de CloudWatch log

Le service géré pour Apache Flink a besoin d'autorisations pour y écrire des erreurs de configuration. CloudWatch Vous pouvez ajouter ces autorisations AWS Identity and Access Management au (rôle IAM) assumé par le service géré pour Apache Flink.

Pour plus d'informations sur l'utilisation de rôle IAM avec le service géré pour Apache Flink, consultez Gestion de l'identité et des accès dans le service géré Amazon pour Apache Flink.

Stratégie d'approbation

Pour autoriser le service géré pour Apache Flink à assumer un rôle IAM, vous pouvez attacher la stratégie d'approbation au rôle d'exécution du service.

```
{
  "Version": "2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanlaytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Stratégie d'autorisations

Pour autoriser une application à écrire les événements du journal CloudWatch depuis une ressource Managed Service for Apache Flink, vous pouvez utiliser la politique d'autorisation IAM suivante. Fournissez les Amazon Resource Names (ARN) corrects pour votre groupe de journaux et votre flux.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt0123456789000",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-east-1:123456789012:log-group:my-log-group:log-
stream:my-log-stream*",
                "arn:aws:logs:us-east-1:123456789012:log-group:my-log-group:*",
                "arn:aws:logs:us-east-1:123456789012:log-group:*",
            ]
        }
    ]
}
```

Niveaux de surveillance des applications

Vous contrôlez la génération des messages du journal de l'application à l'aide du niveau de surveillance des métriques et du niveau de surveillance des journaux de l'application.

Le niveau de surveillance des métriques de l'application contrôle la granularité des messages du journal. Les niveaux de surveillance des métriques sont définis comme suit :

- Application : les métriques s'appliquent à l'ensemble de l'application.
- Tâche : les mesures sont adaptées à chaque tâche. Pour plus d'informations sur les tâches, consultez the section called “Mise à l'échelle”.
- Opérateur : les métriques sont limitées à chaque opérateur. Pour des informations sur les opérateurs, consultez the section called “Opérateurs d'API DataStream”.
- Parallélisme : les métriques sont limitées au parallélisme des applications. Vous ne pouvez définir ce niveau de métrique qu'à l'aide du MonitoringConfigurationUpdateparamètre de l' UpdateApplicationAPI. Vous ne pouvez pas définir ce niveau de métriques à l'aide de la console. Pour de plus amples informations sur lee parallèlisme, veuillez consulter the section called “Mise à l'échelle”.

Le niveau de surveillance des journaux de l'application contrôle la verbosité du journal de l'application. Les niveaux de surveillance des journaux sont définis comme suit :

- Erreur : événements catastrophiques potentiels de l'application.
- Avertissement : situations potentiellement dangereuses de l'application.
- Information : événements de défaillance informatifs et temporaires de l'application. Nous vous recommandons d'utiliser ce niveau de journalisation.
- Débogage : événements informatifs précis qui sont particulièrement utiles pour déboguer une application. Remarque : utilisez ce niveau uniquement à des fins de débogage temporaire.

Bonnes pratiques de journalisation

Nous recommandons que votre application utilise le niveau de journalisation Information. Nous recommandons ce niveau pour garantir que vous voyez les erreurs Apache Flink, qui sont journalisées au niveau Information plutôt qu'au niveau Erreur.

Nous vous recommandons de n'utiliser le niveau Débogage que temporairement lorsque vous étudiez les problèmes liés à l'application. Revenez au niveau Information lorsque le problème est résolu. L'utilisation du niveau de journalisation Débogage affectera de manière significative les performances de votre application.

Une journalisation excessive peut également avoir un impact significatif sur les performances de l'application. Nous vous recommandons de ne pas écrire d'entrée de journal pour chaque enregistrement traité, par exemple. Une journalisation excessive peut entraîner de graves blocages dans le traitement des données et entraîner une contre-pression lors de la lecture des données provenant des sources.

Journalisation de dépannage

Si les journaux de l'application ne sont pas écrits dans le flux de journaux, vérifiez les points suivants :

- Vérifiez que le rôle et les politiques IAM de votre application sont corrects. La politique de votre application nécessite les autorisations suivantes pour accéder à votre flux de journaux :
  - `logs:PutLogEvents`
  - `logs:DescribeLogGroups`
  - `logs:DescribeLogStreams`

  Pour plus d'informations, consultez the section called "Ajouter des autorisations d'écriture dans le flux de CloudWatch log".
- Vérifiez que votre application est en cours d'exécution Pour vérifier le statut de votre application, consultez la page de votre application dans la console ou utilisez les ListApplicationsactions DescribeApplicationou.
- Surveillez CloudWatch les métriques, par exemple `downtime` pour diagnostiquer d'autres problèmes liés aux applications. Pour plus d'informations sur CloudWatch les mesures de lecture, consultezMétriques et dimensions dans le service géré pour Apache Flink.

Étape suivante

Après avoir activé la CloudWatch connexion à votre application, vous pouvez utiliser CloudWatch Logs Insights pour analyser les journaux de votre application. Pour plus d'informations, consultez the section called "Analyse des journaux".

Analyse des journaux avec CloudWatch Logs Insights

Après avoir ajouté une option de CloudWatch journalisation à votre application, comme décrit dans la section précédente, vous pouvez utiliser CloudWatch Logs Insights pour interroger vos flux de journaux afin de détecter des événements ou des erreurs spécifiques.

CloudWatch Logs Insights vous permet de rechercher et d'analyser de manière interactive les données de vos CloudWatch journaux dans Logs.

Pour savoir comment démarrer avec CloudWatch Logs Insights, voir Analyser les données des CloudWatch journaux avec Logs Insights.

Exécuter un exemple de requête

Cette section décrit comment exécuter un exemple de requête CloudWatch Logs Insights.

Prérequis

- Groupes de journaux et flux de journaux existants configurés dans CloudWatch Logs.
- Journaux existants stockés dans CloudWatch Logs.

Si vous utilisez des services tels qu'AWS CloudTrailAmazon Route 53 ou Amazon VPC, vous avez probablement déjà configuré les journaux de ces services pour qu'ils soient accessibles dans Logs. CloudWatch Pour plus d'informations sur l'envoi de CloudWatch journaux à Logs, consultez Getting Started with CloudWatch Logs.

Les requêtes dans CloudWatch Logs Insights renvoient soit un ensemble de champs provenant des événements du journal, soit le résultat d'une agrégation mathématique ou d'une autre opération effectuée sur les événements du journal. Cette section illustre une requête qui renvoie une liste d'événements du journal.

Pour exécuter un exemple de requête CloudWatch Logs Insights

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans le panneau de navigation, choisissez Insights.
3. L'éditeur de requête en haut de l'écran contient une requête par défaut qui renvoie les 20 événements du journal les plus récents. Sélectionnez un groupe de journaux à interroger, au-dessus de l'éditeur de requête.

   Lorsque vous sélectionnez un groupe de CloudWatch journaux, Logs Insights détecte automatiquement les champs des données du groupe de journaux et les affiche dans la section Champs découverts dans le volet droit. Il affiche également un graphique à barres des événements de journaux dans ce groupe de journaux au fil du temps. Ce graphique à barres

montre la distribution des événements dans le groupe de journaux correspondant à vos requête et plage de temps, pas seulement les événements affichés dans le tableau.

4. Choisissez Exécuter la requête.

   Les résultats de la requête s'affichent. Dans cet exemple, les résultats sont les 20 évènements de journaux les plus récents (tous types confondus).

5. Pour afficher tous les champs renvoyés de l'un des événements de journaux, choisissez la flèche à gauche de cet événement de journal.

Pour plus d'informations sur l'exécution et la modification CloudWatch des requêtes Logs Insights, voir Exécuter et modifier un exemple de requête.

Exemples de requêtes

Cette section contient des exemples de requêtes CloudWatch Logs Insights pour analyser les journaux de l'application Managed Service for Apache Flink. Ces requêtes recherchent plusieurs exemples de conditions d'erreur et servent de modèles pour écrire des requêtes qui détectent d'autres conditions d'erreur.

> **Note**
>
> Remplacez la région (*us-west-2*), l'ID de compte (*012345678901*) et le nom de l'application (*YourApplication*) dans les exemples de requêtes suivants par la région de votre application et votre numéro de compte.

Cette rubrique contient les sections suivantes :

- Analyser les opérations : répartition des tâches
- Analyser les opérations : modification du parallélisme
- Analyser les erreurs : Access Denied
- Analyser les erreurs : Source or Sink Not Found
- Analyser les erreurs : Application Task-Related Failures

Analyser les opérations : répartition des tâches

La requête CloudWatch Logs Insights suivante renvoie le nombre de tâches que le gestionnaire de tâches Apache Flink distribue entre les gestionnaires de tâches. Vous devez définir le délai de la requête pour qu'il corresponde à une tâche exécutée afin que la requête ne renvoie pas les tâches des tâches précédentes. Pour de plus amples informations sur le parallélisme, veuillez consulter Mise à l'échelle.

```
fields @timestamp, message
| filter message like /Deploying/
| parse message " to flink-taskmanager-*" as @tmid
| stats count(*) by @tmid
| sort @timestamp desc
| limit 2000
```

La requête CloudWatch Logs Insights suivante renvoie les sous-tâches assignées à chaque gestionnaire de tâches. Le nombre total de sous-tâches est la somme du parallélisme de chaque tâche. Le parallélisme des tâches est dérivé du parallélisme des opérateurs et est identique au parallélisme de l'application par défaut, sauf si vous le modifiez dans le code en spécifiant `setParallelism`. Pour plus d'informations sur la définition du parallélisme des opérateurs, consultez la section Définition du parallélisme : niveau opérateur dans la documentation d'Apache Flink.

```
fields @timestamp, @tmid, @subtask
| filter message like /Deploying/
| parse message "Deploying * to flink-taskmanager-*" as @subtask, @tmid
| sort @timestamp desc
| limit 2000
```

Pour plus d'informations sur la planification des tâches, consultez la section Tâches et planification dans la documentation d'Apache Flink.

Analyser les opérations : modification du parallélisme

La requête CloudWatch Logs Insights suivante renvoie les modifications apportées au parallélisme d'une application (par exemple, en raison du dimensionnement automatique). Cette requête renvoie également les modifications manuelles apportées au parallélisme de l'application. Pour plus d'informations sur la mise à l'échelle automatique, consultez the section called “Mise à l'échelle automatique”.

```
fields @timestamp, @parallelism
| filter message like /property: parallelism.default, /
| parse message "default, *" as @parallelism
| sort @timestamp asc
```

Analyser les erreurs : Access Denied

La requête CloudWatch Logs Insights suivante renvoie `Access Denied` des journaux.

```
fields @timestamp, @message, @messageType
| filter applicationARN like /arn:aws:kinesisanalytics us-
west-2:012345678901:application\/YourApplication/
| filter @message like /AccessDenied/
| sort @timestamp desc
```

Analyser les erreurs : Source or Sink Not Found

La requête CloudWatch Logs Insights suivante renvoie `ResourceNotFound` des journaux. `ResourceNotFound`enregistre le résultat si aucune source ou récepteur Kinesis n'est trouvé.

```
fields @timestamp,@message
| filter applicationARN like /arn:aws:kinesisanalytics us-
west-2:012345678901:application\/YourApplication/
| filter @message like /ResourceNotFoundException/
| sort @timestamp desc
```

Analyser les erreurs : Application Task-Related Failures

La requête CloudWatch Logs Insights suivante renvoie les journaux d'échec liés aux tâches d'une application. Ces journaux sont générés lorsque le statut d'une application passe de `RUNNING` à `RESTARTING`.

```
fields @timestamp,@message
| filter applicationARN like /arn:aws:kinesisanalytics us-
west-2:012345678901:application\/YourApplication/
| filter @message like /switched from RUNNING to RESTARTING/
| sort @timestamp desc
```

Pour les applications utilisant Apache Flink version 1.8.2 et antérieures, les échecs liés aux tâches entraîneront le passage de l'état de l'application de `RUNNING` à `FAILED`. Lorsque vous utilisez

Apache Flink 1.8.2 et versions antérieures, utilisez la requête suivante pour rechercher les échecs liés aux tâches de l'application :

```
fields @timestamp,@message
| filter applicationARN like /arn:aws:kinesisanalytics us-
west-2:012345678901:application\/YourApplication/
| filter @message like /switched from RUNNING to FAILED/
| sort @timestamp desc
```

Affichage des métriques et des dimensions dans le service géré pour Apache Flink

Cette rubrique contient les sections suivantes :

- Métriques d'application
- Métriques du connecteur Kinesis Data Streams
- Métriques du connecteur Amazon MSK
- Métriques d'Apache Zeppelin
- Affichage des CloudWatch métriques
- Définition des niveaux CloudWatch de rapport des métriques
- Utilisation des métriques personnalisées avec le service géré Amazon pour Apache Flink
- Utilisation des CloudWatch alarmes avec Amazon Managed Service pour Apache Flink

Lorsque votre service géré pour Apache Flink traite une source de données, le service géré pour Apache Flink communique les mesures et dimensions suivantes à Amazon. CloudWatch

Métriques d'application

Mesure	Unit	Description	Niveau	Notes d'utilisation	
`backPressuredTimeMsPerSecond*`	Millisecondes	Durée (en millisecondes) pendant laquelle cette tâche ou cet	Tâche, opérateur, parallélisme	*Disponible pour les applications de service géré pour Apache	

Mesure	Unit	Description	Niveau	Notes d'utilisation	
		opérateur subit une contre-pression par seconde.		Flink exécutant la version 1.13 de Flink uniquement. Ces mesures peuvent être utiles pour identifier les goulots d'étranglement d'une application.	
`busyTimeMsPerSecond*`	Millisecondes	Durée (en millisecondes) pendant laquelle cette tâche ou cet opérateur est occupé (ni inactif ni en train de subir une contre-pression) par seconde. Peut être NaN, si la valeur n'a pas pu être calculée.	Tâche, opérateur, parallélisme	*Disponible pour les applications de service géré pour Apache Flink exécutant la version 1.13 de Flink uniquement. Ces mesures peuvent être utiles pour identifier les goulots d'étranglement d'une application.	

Mesure	Unit	Description	Niveau	Notes d'utilisation	
`cpuUtilization`	Pourcentage	Pourcentage global d'utilisation du processus dans les gestionnaires de tâches. Par exemple, s'il existe cinq gestionnaires de tâches, le service géré pour Apache Flink publie cinq échantillons de cette métrique par intervalle de reporting.	Application	Vous pouvez utiliser cette métrique pour surveiller l'utilisation minimale, moyenne et maximale du processeur dans votre application. La `CPUUtilization` métrique prend uniquement en compte l'utilisation du processeur par le processus TaskManager JVM exécuté dans le conteneur.	

Mesure	Unit	Description	Niveau	Notes d'utilisation
`containerCPUUtilization`	Pourcentage	Pourcentage global d'utilisation du processeur dans les conteneurs du gestionnaire de tâches du cluster d'applications Flink. Par exemple, s'il existe cinq gestionnaires de tâches, il y a donc cinq TaskManager conteneurs et Managed Service for Apache Flink publie 2* cinq échantillons de cette métrique par intervalle de rapport d'une minute.	Application	Calculé par conteneur comme suit : Temps CPU total (en secondes) consommé par le conteneur x 100/limite CPU du conteneur (en CPU/secondes) La `CPUUtilization` métrique prend uniquement en compte l'utilisation du processeur par le processus TaskManager JVM exécuté dans le conteneur. D'autres composants s'exécutent en dehors de JVM dans le même conteneur. La métrique `container`

Mesure	Unit	Description	Niveau	Notes d'utilisation	
				`CPUUtilization` vous donne une image plus complète, y compris tous les processus en termes d'épuisement du processeur au niveau du conteneur et les échecs qui en résultent.	

Mesure	Unit	Description	Niveau	Notes d'utilisation
`containerMemoryUtilization`	Pourcentage	Pourcentage global d'utilisation de la mémoire dans les conteneurs du gestionnaire de tâches du cluster d'applications Flink. Par exemple, s'il existe cinq gestionnaires de tâches, il y a donc cinq TaskManager conteneurs et Managed Service for Apache Flink publie 2* cinq échantillons de cette métrique par intervalle de rapport d'une minute.	Application	Calculé par conteneur comme suit : Utilisation de la mémoire du conteneur (octets) x 100/limite de mémoire du conteneur selon les spécifications de déploiement du pod (en octets) Les `ManagedMemoryUtilzations` métriques `HeapMemoryUtilization` et ne prennent en compte que des mesures de mémoire spécifiques, telles que l'utilisation de la mémoire par segment de

Mesure	Unit	Description	Niveau	Notes d'utilisation	
				mémoire de la TaskManager JVM ou la mémoire gérée (utilisation de la mémoire en dehors de la JVM pour des processus natifs tels que RockSDB State Backend). La métrique `containerMemoryUtilization` vous donne une image plus complète en incluant la mémoire de travail, qui permet de mieux suivre l'épuisement total de la mémoire. Une fois épuisée, elle se retrouvera dans `Out of Memory Error` la	

Mesure	Unit	Description	Niveau	Notes d'utilisation	
				TaskManager capsule.	
`containerDiskUtilization`	Pourcentage	Pourcentage global d'utilisation du disque dans les conteneurs du gestionnaire de tâches du cluster d'applications Flink. Par exemple, s'il existe cinq gestionnaires de tâches, il y a donc cinq TaskManager conteneurs et Managed Service for Apache Flink publie 2* cinq échantillons de cette métrique par intervalle de rapport d'une minute.	Application	Calculé par conteneur comme suit : Utilisation du disque en octets* 100/limite de disque pour le conteneur en octets Pour les conteneurs, cela représente l'utilisation du système de fichiers sur lequel le volume racine du conteneur est configuré.	

Mesure	Unit	Description	Niveau	Notes d'utilisation	
`currentInputWatermark`	Millisecondes	Le dernier filigrane que cette application/opérateur/tâche/thread a reçu	Application, opérateur, tâche, parallélisme	Cet enregistrement n'est émis que pour les dimensions à deux entrées. Il s'agit de la valeur minimale des derniers filigranes reçus.	
`currentOutputWatermark`	Millisecondes	Le dernier filigrane que cette application/opérateur/tâche/thread a émis	Application, opérateur, tâche, parallélisme		

Mesure	Unit	Description	Niveau	Notes d'utilisation	
`downtime`	Millisecondes	Pour les tâches actuellement en situation d'échec ou de récupération, le temps écoulé pendant cette panne.	Application	Cette métrique mesure le temps écoulé pendant l'échec ou la récupération d'une tâche. Cette métrique renvoie 0 pour les tâches en cours d'exécution et -1 pour les tâches terminées. Si cette métrique n'est pas égale à 0 ou -1, cela indique que la tâche Apache Flink de l'application n'a pas pu être exécuté.	

Mesure	Unit	Description	Niveau	Notes d'utilisation	
`fullRestarts`	Nombre	Nombre total de fois que cette tâche a été complètement redémarrée depuis son envoi. Cette métrique ne mesure pas les redémarrages affinés.	Application	Vous pouvez utiliser cette métrique pour évaluer l'état général de l'application. Les redémarrages peuvent avoir lieu pendant la maintenance interne par le service géré pour Apache Flink. Un nombre de redémarrages plus élevé que d'habitude peut indiquer un problème lié à l'application.	

Mesure	Unit	Description	Niveau	Notes d'utilisation	
`heapMemoryUtilization`	Pourcentage	Utilisation globale de la mémoire de tas dans les gestionnaires de tâches. Par exemple, s'il existe cinq gestionnaires de tâches, le service géré pour Apache Flink publie cinq échantillons de cette métrique par intervalle de reporting.	Application	Vous pouvez utiliser cette métrique pour surveiller l'utilisation minimale, moyenne et maximale de l'utilisation de la mémoire de tas dans votre application. Le `HeapMemoryUtilization` seul prend en compte des métriques de mémoire spécifiques, telles que l'utilisation de la mémoire par segment de mémoire de la TaskManager JVM.	

Mesure	Unit	Description	Niveau	Notes d’utilisation	
`idleTimeMsPerSecond*`	Millisecondes	Durée (en millisecondes) pendant laquelle cette tâche ou cet opérateur est inactif (n’a aucune donnée à traiter) par seconde. Le temps d'inactivité exclut le temps de contre-pression. Ainsi, si la tâche est contre-pressée, elle n'est pas inactive.	Tâche, opérateur, parallélisme	*Disponible pour les applications de service géré pour Apache Flink exécutant la version 1.13 de Flink uniquement. Ces mesures peuvent être utiles pour identifier les goulots d'étranglement d'une application.	

Mesure	Unit	Description	Niveau	Notes d'utilisation	
`lastCheckpointSize`	Octets	La taille totale du dernier point de contrôle	Application	Vous pouvez utiliser cette métrique pour déterminer l'utilisation du stockage des applications en cours d'exécution. Si la valeur de cette métrique augmente, cela peut indiquer un problème lié à votre application, tel qu'une fuite de mémoire ou un goulot d'étranglement.	

Mesure	Unit	Description	Niveau	Notes d'utilisation	
`lastCheckpointDuration`	Millisecondes	Le temps qu'il a fallu pour terminer le dernier point de contrôle	Application	Cette métrique mesure le temps nécessaire pour terminer le point de contrôle le plus récent. Si la valeur de cette métrique augmente, cela peut indiquer un problème lié à votre application, tel qu'une fuite de mémoire ou un goulot d'étranglement. Dans certains cas, vous pouvez résoudre ce problème en désactivant le point de contrôle.	

Mesure	Unit	Description	Niveau	Notes d'utilisation	
managedMemoryUsed*	Octets	Quantité de mémoire gérée actuellement en cours d'utilisation.	Application, opérateur, tâche, parallélisme	*Disponible pour les applications de service géré pour Apache Flink exécutant la version 1.13 de Flink uniquement. Cela concerne la mémoire gérée par Flink en dehors du tas de Java. Elle est utilisée pour le backend d'état RockSDB et est également disponible pour les applications.	

Mesure	Unit	Description	Niveau	Notes d'utilisation
`managedMemoryTotal` *	Octets	Quantité totale de mémoire gérée.	Application, opérateur, tâche, parallélisme	*Disponible pour les applications de service géré pour Apache Flink exécutant la version 1.13 de Flink uniquement. Cela concerne la mémoire gérée par Flink en dehors du tas de Java. Elle est utilisée pour le backend d'état RockSDB et est également disponible pour les applications. La métrique `ManagedMemoryUtilzations` ne prend en compte que des mesures de mémoire spécifiques telles que la mémoire gérée

Mesure	Unit	Description	Niveau	Notes d'utilisation
				(utilisation de la mémoire en dehors de JVM pour les processus natifs tels que RockSDB State Backend)
`managedMemoryUtilization*`	Pourcentage	Dérivé par managedMemoryUsed/ managedMemoryTotal	Application, opérateur, tâche, parallélisme	*Disponible pour les applications de service géré pour Apache Flink exécutant la version 1.13 de Flink uniquement. Cela concerne la mémoire gérée par Flink en dehors du tas de Java. Elle est utilisée pour le backend d'état RockSDB et est également disponible pour les applications.

Mesure	Unit	Description	Niveau	Notes d'utilisation	
`numberOfFailedCheckpoints`	Nombre	Nombre de fois que le point de contrôle a échoué.	Application	Vous pouvez utiliser cette métrique pour surveiller l'état et la progression des applications. Les points de contrôle peuvent échouer en raison de problèmes d'application, tels que des problèmes de débit ou d'autorisation.	

Mesure	Unit	Description	Niveau	Notes d'utilisation	
`numRecordsIn*`	Nombre	Nombre total d'enregistrements reçus par cette application, cet opérateur ou cette tâche.	Application, opérateur, tâche, parallélisme	*Pour appliquer la statistique SUM sur une période donnée (seconde/minute) : • Sélectionnez la métrique au niveau approprié. Si vous suivez la métrique d'un opérateur, vous devez sélectionner les métriques d'opérateur correspondantes. • Comme le service géré pour Apache Flink prend 4 instantanés de métrique par minute, les calculs métriques suivants	

Mesure	Unit	Description	Niveau	Notes d'utilisation	
				doivent être utilisés : m1/4 où m1 est la statistique SUM sur une période (seconde/minute) Le niveau de la métrique indique si cette métrique mesure le nombre total d'enregistrements reçus par l'ensemble de l'application, un opérateur spécifique ou une tâche spécifique.	

Mesure	Unit	Description	Niveau	Notes d'utilisation	
`numRecordsInPerSecond*`	Nombre/seconde	Nombre total d'enregistrements reçus par cette application, cet opérateur ou cette tâche par seconde.	Application, opérateur, tâche, parallélisme	*Pour appliquer la statistique SUM sur une période donnée (seconde/minute) : • Sélectionnez la métrique au niveau approprié. Si vous suivez la métrique d'un opérateur, vous devez sélectionner les métriques d'opérateur correspondantes. • Comme le service géré pour Apache Flink prend 4 instantanés de métrique par minute, les calculs métriques suivants	

Mesure	Unit	Description	Niveau	Notes d'utilisation	
				doivent être utilisés : m1/4 où m1 est la statistique SUM sur une période (seconde/minute) Le niveau de la métrique indique si cette métrique mesure le nombre total d'enregistrements reçus par l'ensemble de l'application, un opérateur spécifique ou une tâche spécifique par seconde.	

Mesure	Unit	Description	Niveau	Notes d'utilisation	
`numRecordsOut*`	Nombre	Nombre total d'enregistrements émis par cette application, cet opérateur ou cette tâche.	Application, opérateur, tâche, parallélisme	*Pour appliquer la statistique SUM sur une période donnée (seconde/minute) : • Sélectionnez la métrique au niveau approprié. Si vous suivez la métrique d'un opérateur, vous devez sélectionner les métriques d'opérateur correspondantes. • Comme le service géré pour Apache Flink prend 4 instantanés de métrique par minute, les calculs métriques suivants	

Mesure	Unit	Description	Niveau	Notes d'utilisation	
				doivent être utilisés : m1/4 où m1 est la statistique SUM sur une période (seconde/minute) Le niveau de la métrique indique si cette métrique mesure le nombre total d'enregistrements émis par l'ensemble de l'application, un opérateur spécifique ou une tâche spécifique.	

Mesure	Unit	Description	Niveau	Notes d'utilisation	
`numLateRecordsDropped*`	Nombre	Application, opérateur, tâche, parallélisme		*Pour appliquer la statistique SUM sur une période donnée (seconde/minute) : • Sélectionnez la métrique au niveau approprié. Si vous suivez la métrique d'un opérateur, vous devez sélectionner les métriques d'opérateur correspondantes. • Comme le service géré pour Apache Flink prend 4 instantanés de métrique par minute, les calculs métriques suivants	

Mesure	Unit	Description	Niveau	Notes d'utilisation	
				doivent être utilisés : m1/4 où m1 est la statistique SUM sur une période (seconde/minute) Le nombre d'enregistrements que cet opérateur ou cette tâche a perdus en raison de son arrivée tardive.	

Mesure	Unit	Description	Niveau	Notes d'utilisation
`numRecordsOutPerSecond*`	Nombre/seconde	Nombre total d'enregistrements émis par cette application, cet opérateur ou cette tâche par seconde.	Application, opérateur, tâche, parallélisme	*Pour appliquer la statistique SUM sur une période donnée (seconde/minute) : • Sélectionnez la métrique au niveau approprié. Si vous suivez la métrique d'un opérateur, vous devez sélectionner les métriques d'opérateur correspondantes. • Comme le service géré pour Apache Flink prend 4 instantanés de métrique par minute, les calculs métriques suivants

Mesure	Unit	Description	Niveau	Notes d'utilisation	
				doivent être utilisés : m1/4 où m1 est la statistique SUM sur une période (seconde/minute) Le niveau de la métrique indique si cette métrique mesure le nombre total d'enregistrements émis par l'ensemble de l'application, un opérateur spécifique ou une tâche spécifique par seconde.	

Mesure	Unit	Description	Niveau	Notes d'utilisation
`oldGenerationGCCount`	Nombre	Le nombre total d'anciennes opérations de récupération de mémoire qui ont eu lieu dans tous les gestionnaires de tâches.	Application	
`oldGenerationGCTime`	Millisecondes	Le temps total passé à effectuer d'anciennes opérations de récupération de mémoire.	Application	Vous pouvez utiliser cette métrique pour surveiller la durée totale, moyenne et maximale de récupération de mémoire.
`threadCount`	Nombre	Nombre total de threads actifs utilisés par l'application.	Application	Cette métrique mesure le nombre de threads utilisés par le code de l'application. Ce n'est pas la même chose que le parallélisme des applications.

Mesure	Unit	Description	Niveau	Notes d'utilisation	
`uptime`	Millisecondes	Durée pendant laquelle la tâche a été exécutée sans interruption.	Application	Vous pouvez utiliser cette métrique pour déterminer si une tâche s'exécute correctement. Cette métrique renvoie -1 pour les tâches terminées.	

Métriques du connecteur Kinesis Data Streams

AWS émet tous les enregistrements pour Kinesis Data Streams, outre les suivants :

Mesure	Unit	Description	Niveau	Notes d'utilisation
`millisbehindLatest`	Millisecondes	Le nombre de millisecondes où le consommateur est en retard par rapport au début du flux, qui indique le retard que subit le consommateur.	Application (pour Stream), Parallélisme (pour) ShardId	• Une valeur égale à 0 indique que le traitement des enregistrements est terminé et qu'il ne reste plus d'enregistrements à traiter pour le moment. La métrique d'une partition

Mesure	Unit	Description	Niveau	Notes d'utilisation
				particulière peut être spécifiée par le nom du flux et l'identifiant de la partition. • La valeur -1 indique que le service n'a pas encore indiqué de valeur pour la métrique.
`bytesRequestedPerFetch`	Octets	Les octets demandés lors d'un seul appel de `getRecords` .	Application (pour Stream), Parallélisme (pour) ShardId	

Métriques du connecteur Amazon MSK

AWS émet tous les enregistrements pour Amazon MSK, outre les suivants :

Mesure	Unit	Description	Niveau	Notes d'utilisation
`currentoffsets`	N/A	Le décalage de lecture actuel du consommateur, pour chaque partition. La métrique d'une partition particuli	Application (pour le sujet), Parallélisme (pour) PartitionId	

Mesure	Unit	Description	Niveau	Notes d'utilisation
		ère peut être spécifiée par le nom de la rubrique et l'identifiant de la partition.		
`commitsFailed`	N/A	Le nombre total d'échecs de validation de décalage pour Kafka, si la validation de décalage et le point de contrôle sont activés.	Application, opérateur, tâche, parallélisme	La réattribution des validations de décalage à Kafka n'est qu'un moyen de révéler les progrès réalisés par les consommateurs. Un échec de validation n'affecte donc pas l'intégrité des décalages de partition à points de contrôle de Flink.

Mesure	Unit	Description	Niveau	Notes d'utilisation
`commitsSucceeded`	N/A	Le nombre total de validations de décalage réussies dans Kafka, si la validation de décalage et les points de contrôle sont activés.	Application, opérateur, tâche, parallélisme	
`committed offsets`	N/A	Le dernier décalage correctement validé dans Kafka, pour chaque partition. La métrique d'une partition particulière peut être spécifiée par le nom de la rubrique et l'identifiant de la partition.	Application (pour le sujet), Parallélisme (pour) PartitionId	
`records_lag_max`	Nombre	Le décalage maximal en termes de nombre d'enregistrements pour chaque partition de cette fenêtre	Application, opérateur, tâche, parallélisme	

Mesure	Unit	Description	Niveau	Notes d'utilisation
bytes_consumed_rate	Octets	Nombre moyen d'octets consommés par seconde pour une rubrique	Application, opérateur, tâche, parallélisme	

Métriques d'Apache Zeppelin

Pour les blocs-notes Studio, AWS émet les mesures suivantes au niveau de l'application : KPUs, cpuUtilization, heapMemoryUtilization, oldGenerationGCTime, oldGenerationGCCount et threadCount. En outre, il émet les métriques indiquées dans le tableau suivant, également au niveau de l'application.

Mesure	Unit	Description	Nom Prometheus
zeppelinCpuUtilization	Pourcentage	Pourcentage global d'utilisation du processeur sur le serveur Apache Zeppelin.	process_cpu_usage
zeppelinHeapMemoryUtilization	Pourcentage	Pourcentage global d'utilisation de la mémoire de tas pour le serveur Apache Zeppelin.	jvm_memory_used_bytes
zeppelinThreadCount	Nombre	Le nombre total de threads actifs utilisés par le serveur Apache Zeppelin.	jvm_threads_live_threads

Mesure	Unit	Description	Nom Prometheus
`zeppelinWaitingJobs`	Nombre	Le nombre de tâches Apache Zeppelin en attente d'un thread.	`jetty_threads_jobs`
`zeppelinServerUptime`	Secondes	Durée totale pendant laquelle le serveur a été opérationnel.	`process_uptime_seconds`

Affichage des CloudWatch métriques

Vous pouvez consulter CloudWatch les statistiques de votre application à l'aide de la CloudWatch console Amazon ou duAWS CLI.

Pour afficher les métriques à l'aide de la CloudWatch console

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans le panneau de navigation, sélectionnez Metrics (Métriques).
3. Dans le volet CloudWatch Mesures par catégorie du service géré pour Apache Flink, choisissez une catégorie de mesures.
4. Dans le volet supérieur, faites défiler l'écran pour afficher la liste complète des métriques.

Pour afficher les métriques à l'aide de AWS CLI

- A partir d'une invite de commande, utilisez la commande suivante.

```
aws cloudwatch list-metrics --namespace "AWS/KinesisAnalytics" --region region
```

Définition des niveaux CloudWatch de rapport des métriques

Vous pouvez contrôler le niveau des métriques d'application créées par votre application. Le service géré pour Apache Flink prend en charge les niveaux de métriques suivants :

- Application : l'application crée un rapport uniquement pour le plus haut niveau de métriques pour chaque application. Les métriques du service géré pour Apache Flink sont publiées par défaut au niveau de l'application.
- Tâche : l'application crée un rapport pour les dimensions de métrique spécifiques à la tâche pour les métriques définies avec le niveau de rapport des métriques Tâche, telles que le nombre d'enregistrements entrants et sortants de l'application par seconde.
- Opérateur : l'application crée un rapport pour les dimensions de métrique spécifiques à l'opérateur pour les métriques définies avec le niveau de rapport des métriques Opérateur, telles que les métriques pour chaque filtre ou opération cartographique.
- Parallélisme : l'application crée un rapport pour les niveaux de métrique `Task` et `Operator` pour chaque thread d'exécution. Ce niveau de création de rapport n'est pas recommandé pour les applications avec un parallélisme supérieur à 64 en raison de coûts excessifs.

> **Note**
>
> Vous ne devez utiliser ce niveau de métrique que pour le dépannage en raison de la quantité de données métriques générées par le service. Vous ne pouvez définir ce niveau de métriques qu'à l'aide de l'interface CLI. Ce niveau de métrique n'est pas disponible dans la console.

Le niveau par défaut est Application. L'application fournit des statistiques au niveau actuel et à tous les niveaux supérieurs. Par exemple, si le niveau de création de rapport est défini sur Opérateur, l'application crée un rapport pour les métriques Application, Tâche et Opérateur.

Vous définissez le niveau de rapport CloudWatch des métriques en utilisant le `MonitoringConfiguration` paramètre de l'`CreateApplication`action, ou le `MonitoringConfigurationUpdate` paramètre de l'`UpdateApplication`action. L'exemple de demande d'`UpdateApplication`action suivant définit le niveau de rapport CloudWatch des métriques sur Task :

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "MonitoringConfigurationUpdate": {
            "ConfigurationTypeUpdate": "CUSTOM",
```

```
            "MetricsLevelUpdate": "TASK"
         }
      }
   }
}
```

Vous pouvez également configurer le niveau de journalisation à l'aide du paramètre `LogLevel` de l'action `CreateApplication`, ou du paramètre `LogLevelUpdate` de l'action `UpdateApplication`. Vous pouvez utiliser les niveaux de journalisation suivants :

- `ERROR` : enregistre les événements d'erreur potentiellement récupérables.
- `WARN` : enregistre les événements d'avertissement susceptibles de provoquer une erreur.
- `INFO` : enregistre les événements informatifs.
- `DEBUG` : enregistre les événements de débogage généraux.

Pour plus d'informations sur les niveaux de journalisation Log4j, consultez la section Niveaux de journalisation personnalisés dans la documentation d'Apache Log4j.

Utilisation des métriques personnalisées avec le service géré Amazon pour Apache Flink

Le service géré pour Apache Flink expose 19 mesures CloudWatch, y compris des mesures relatives à l'utilisation des ressources et au débit. En outre, vous pouvez créer vos propres métriques pour suivre des données spécifiques à l'application, telles que le traitement des événements ou l'accès à des ressources externes.

Cette rubrique contient les sections suivantes :

- Comment ça marche
- Exemples
- Affichage des métriques personnalisées

Comment ça marche

Les métriques personnalisées du service géré pour Apache Flink utilisent le système de métrique Apache Flink. Les métriques Apache Flink ont les attributs suivants :

- Type : le type d'une métrique décrit la manière dont elle mesure et crée des rapport sur les données. Les types de métriques Apache Flink disponibles incluent Nombre, Jauge, Histogramme et Mètre. Pour plus d'informations sur les types de métriques Apache Flink, consultez la section Metric Types.

  > **Note**
  >
  > AWS CloudWatch Metrics ne prend pas en charge le type de métrique Histogram Apache Flink. CloudWatch ne peut afficher que les métriques Apache Flink des types Count, Gauge et Meter.

- Portée : la portée d'une métrique comprend son identifiant et un ensemble de paires clé-valeur qui indiquent comment la métrique sera signalée. CloudWatch L'identifiant d'une métrique se compose des éléments suivants :
  - Une portée système, qui indique le niveau auquel la métrique est signalée (par exemple, Opérateur).
  - Une portée utilisateur, qui définit des attributs tels que les variables utilisateur ou les noms de groupes de métriques. Ces attributs sont définis à l'aide de `MetricGroup.addGroup(key, value)` ou `MetricGroup.addGroup(name)`.

  Pour plus d'informations sur la portée des métriques, consultez Scope.

Pour plus d'informations sur les métriques d'Apache Flink, consultez Metrics de la documentation d'Apache Flink.

Pour créer une métrique personnalisée dans votre service géré pour Apache Flink, vous pouvez accéder au système de métrique Apache Flink à partir de n'importe quelle fonction utilisateur qui étend `RichFunction` en appelant `GetMetricGroup`. Cette méthode renvoie un MetricGroupobjet que vous pouvez utiliser pour créer et enregistrer des métriques personnalisées. Le service géré pour Apache Flink indique toutes les métriques créées avec la clé de groupe `KinesisAnalytics` to CloudWatch. Les métriques personnalisées que vous définissez présentent les caractéristiques suivantes :

- Votre métrique personnalisée possède un nom de métrique et un nom de groupe. Ces noms doivent être composés de caractères alphanumériques.
- Les attributs que vous définissez dans le champ d'application utilisateur (à l'exception du `KinesisAnalytics` groupe de mesures) sont publiés sous forme de CloudWatch dimensions.

- Les métriques personnalisées sont publiées au niveau `Application` par défaut.
- Les dimensions (Tâche/Opérateur/Parallélisme) sont ajoutées à la métrique en fonction du niveau de surveillance de l'application. Vous définissez le niveau de surveillance de l'application à l'aide du MonitoringConfigurationparamètre de l'CreateApplicationaction ou du MonitoringConfigurationUpdateparamètre ou de l'UpdateApplicationaction.

Exemples

Les exemples de code suivants montrent comment créer une classe de mappage qui crée et incrémente une métrique personnalisée, et comment implémenter la classe de mappage dans votre application en l'ajoutant à un objet `DataStream`.

Métrique personnalisée Nombre d'enregistrements

L'exemple de code suivant montre comment créer une classe de mappage qui crée une métrique qui compte les enregistrements dans un flux de données (fonctionnalité identique à celle de la métrique `numRecordsIn`) :

```
    private static class NoOpMapperFunction extends RichMapFunction<String, String> {
        private transient int valueToExpose = 0;
        private final String customMetricName;

        public NoOpMapperFunction(final String customMetricName) {
            this.customMetricName = customMetricName;
        }

        @Override
        public void open(Configuration config) {
            getRuntimeContext().getMetricGroup()
                    .addGroup("KinesisAnalytics")
                    .addGroup("Program", "RecordCountApplication")
                    .addGroup("NoOpMapperFunction")
                    .gauge(customMetricName, (Gauge<Integer>) () -> valueToExpose);
        }

        @Override
        public String map(String value) throws Exception {
            valueToExpose++;
            return value;
        }
    }
```

Dans l'exemple précédent, la variable `valueToExpose` est incrémentée pour chaque enregistrement traité par l'application.

Après avoir défini votre classe de mappage, vous créez un flux intégré à l'application qui implémente la carte :

```
DataStream<String> noopMapperFunctionAfterFilter =
    kinesisProcessed.map(new NoOpMapperFunction("FilteredRecords"));
```

Pour le code complet de cette application, consultez Record Count Custom Metric Application.

Métrique personnalisée Nombre de mots

L'exemple de code suivant montre comment créer une classe de mappage qui crée une métrique qui compte les mots dans un flux de données :

```
private static final class Tokenizer extends RichFlatMapFunction<String, Tuple2<String,
 Integer>> {

            private transient Counter counter;

            @Override
            public void open(Configuration config) {
                this.counter = getRuntimeContext().getMetricGroup()
                        .addGroup("KinesisAnalytics")
                        .addGroup("Service", "WordCountApplication")
                        .addGroup("Tokenizer")
                        .counter("TotalWords");
            }

            @Override
            public void flatMap(String value, Collector<Tuple2<String, Integer>>out) {
                // normalize and split the line
                String[] tokens = value.toLowerCase().split("\\W+");

                // emit the pairs
                for (String token : tokens) {
                    if (token.length() > 0) {
                        counter.inc();
                        out.collect(new Tuple2<>(token, 1));
                    }
                }
            }
```

```
        }
```

Dans l'exemple précédent, la variable `counter` est incrémentée pour chaque mot traité par l'application.

Après avoir défini votre classe de mappage, vous créez un flux intégré à l'application qui implémente la carte :

```
// Split up the lines in pairs (2-tuples) containing: (word,1), and
// group by the tuple field "0" and sum up tuple field "1"
DataStream<Tuple2<String, Integer>> wordCountStream = input.flatMap(new
 Tokenizer()).keyBy(0).sum(1);

// Serialize the tuple to string format, and publish the output to kinesis sink
wordCountStream.map(tuple -> tuple.toString()).addSink(createSinkFromStaticConfig());
```

Pour le code complet de cette application, voir Word Count Custom Metric Application.

Affichage des métriques personnalisées

Les métriques personnalisées pour votre application apparaissent dans la console CloudWatch Metrics du AWS/KinesisAnalyticstableau de bord, sous le groupe de métriques Application.

Utilisation des CloudWatch alarmes avec Amazon Managed Service pour Apache Flink

À l'aide des alarmes CloudWatch métriques Amazon, vous observez une CloudWatch métrique sur une période que vous spécifiez. L'alarme réalise une ou plusieurs actions en fonction de la valeur de la métrique ou de l'expression par rapport à un seuil sur un certain nombre de périodes. Par exemple, l'alarme pourrait envoyer une notification à une rubrique Amazon Simple Notification Service (Amazon SNS).

Pour plus d'informations sur les CloudWatch alarmes, consultez la section Utilisation d'Amazon CloudWatch Alarms.

Alarmes recommandées

Cette section contient les alarmes recommandées pour surveiller les applications de service géré pour Apache Flink.

Le tableau décrit les alarmes recommandées et comporte les colonnes suivantes :

- Expression métrique : métrique ou expression métrique à tester par rapport au seuil.
- Statistique : statistique utilisée pour vérifier la métrique, par exemple, Moyenne.
- Seuil : l'utilisation de cette alarme vous oblige à déterminer un seuil qui définit la limite des performances attendues de l'application. Vous devez déterminer ce seuil en surveillant votre application dans des conditions normales.
- Description : causes susceptibles de déclencher cette alarme et solutions possibles à ce problème.

Expression de métrique	Statistique	Seuil	Description
`temps d’arrêt > 0`	Average	0	A downtime greater than zero indicates that the application has failed. If the value is larger than 0, the application is not processing any data. Recommended for all applications. The `Temps d'arrêt` metric measures the duration of an outage. A downtime greater than zero indicates that the applicati on has failed. For troubleshooting, see L'application est en train de redémarrer.
`TARIF (numberOf Failedpoints de contrôle) > 0`	Average	0	This metric counts the number of failed checkpoints since the application started. Depending on the application, it can be

Expression de métrique	Statistique	Seuil	Description
			tolerable if checkpoints fail occasionally. But if checkpoints are regularly failing, the application is likely unhealthy and needs further attention. We recommend monitoring RATE(numberOfFailedCheckpoints) to alarm on the gradient and not on absolute values. Recommended for all applications. Use this metric to monitor application health and checkpointing progress. The application saves state data to checkpoints when it's healthy. Checkpointing can fail due to timeouts if the application isn't making progress in processing the input data. For troubleshooting, see Le délai du point de contrôle est expiré.

Expression de métrique	Statistique	Seuil	Description
`Opérateur . numRecord sOutPerSecond` < seuil	Average	The minimum number of records emitted from the applicati on during normal conditions.	Recommended for all applications. Falling below this threshold can indicate that the application isn't making expected progress on the input data. For troublesh ooting, see Le débit est trop lent.

Expression de métrique	Statistique	Seuil	Description
`records_lag_max|millisbehindLatest` > seuil	Maximum	The maximum expected latency during normal conditions.	If the application is consuming from Kinesis or Kafka, these metrics indicate if the application is falling behind and needs to be scaled in order to keep up with the current load. This is a good generic metric that is easy to track for all kinds of applications. But it can only be used for reactive scaling, i.e., when the applicati on has already fallen behind. Recommend ed for all applications. Use the `records_lag_max` metric for a Kafka source, or the `millisbeh indLatest` for a Kinesis stream source. Rising above this threshold can indicate that the application isn't making expected progress on the input data. For troublesh

Expression de métrique	Statistique	Seuil	Description
			ooting, see Le débit est trop lent.

Expression de métrique	Statistique	Seuil	Description
`lastCheckpointDuration` > seuil	Maximum	The maximum expected checkpoint duration during normal conditions.	Monitors how much data is stored in state and how long it takes to take a checkpoint. If checkpoints grow or take long, the application is continuously spending time on checkpointing and has less cycles for actual processing. At some points, checkpoints may grow too large or take so long that they fail. In addition to monitoring absolute values, customers should also considering monitoring the change rate with `TARIF (lastCheckpointSize)` and `TARIF (lastCheckpointDuration)`. If the `lastCheckpointDuration` continuously increases, rising above this threshold can indicate that the application isn't making expected

Expression de métrique	Statistique	Seuil	Description
			progress on the input data, or that there are problems with application health such as backpress ure. For troublesh ooting, see Croissanc e illimitée de l'état.

Expression de métrique	Statistique	Seuil	Description
`lastCheckpointSize` > seuil	Maximum	The maximum expected checkpoint size during normal conditions.	Monitors how much data is stored in state and how long it takes to take a checkpoint. If checkpoints grow or take long, the application is continuously spending time on checkpointing and has less cycles for actual processing. At some points, checkpoints may grow too large or take so long that they fail. In addition to monitoring absolute values, customers should also considering monitoring the change rate with `TARIF (lastCheckpointSize)` and `TARIF (lastCheckpointDuration)`. If the `lastCheckpointSize` continuously increases, rising above this threshold can indicate that the application is accumulating state

Expression de métrique	Statistique	Seuil	Description
			data. If the state data becomes too large, the application can run out of memory when recovering from a checkpoint, or recovering from a checkpoint might take too long. For troublesh ooting, see Croissanc e illimitée de l'état.
`heapMemor yUtilization` > seuil	Maximum	This gives a good indication of the overall resource utilization of the application and can be used for proactive scaling unless the application is I/O bound. The maximum expected `heapMemor yUtilization` size during normal conditions, with a recommended value of 90 percent.	You can use this metric to monitor the maximum memory utilization of task managers across the application. If the application reaches this threshold, you need to provision more resources . You do this by enabling automatic scaling or increasin g the application parallelism. For more information about increasing resources, see Mise à l'échelle.

Expression de métrique	Statistique	Seuil	Description
`cpuUtilization` > seuil	Maximum	This gives a good indication of the overall resource utilization of the application and can be used for proactive scaling unless the application is I/O bound. The maximum expected `cpuUtilization` size during normal conditions, with a recommended value of 80 percent.	You can use this metric to monitor the maximum CPU utilization of task managers across the application. If the application reaches this threshold, you need to provision more resources You do this by enabling automatic scaling or increasing the application parallelism. For more information about increasing resources, see Mise à l'échelle.
`threadsCount` > seuil	Maximum	The maximum expected `threadsCount` size during normal conditions.	You can use this metric to watch for thread leaks in task managers across the application. If this metric reaches this threshold, check your application code for threads being created without being closed.

Expression de métrique	Statistique	Seuil	Description
`(oldGarbageCollectionDurée * 100) /60_000 sur une période de 1 min')` > seuil	Maximum	The maximum expected `oldGarbageCollectionHeure` duration. We recommend setting a threshold such that typical garbage collection time is 60 percent of the specified threshold, but the correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that there is a memory leak in task managers across the application.
`TAUX (oldGarbageCollectionNombre)` > seuil	Maximum	The maximum expected `oldGarbageCollectionCompter` under normal conditions. The correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that there is a memory leak in task managers across the application.
`Opérateur. currentOutputWatermark - Opérateur. currentInputWatermark` > seuil	Minimum	The minimum expected watermark increment under normal conditions. The correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that either the application is processing increasingly older events, or that an upstream subtask has not sent a watermark in an increasingly long time.

Écrire des messages personnalisés dans les CloudWatch journaux

Vous pouvez écrire des messages personnalisés dans le CloudWatch journal de votre application Managed Service for Apache Flink. Pour ce faire, utilisez la bibliothèque `log4j` d'Apache ou la bibliothèque `Simple Logging Facade for Java (SLF4J)`.

Rubriques

- Écrire dans les CloudWatch journaux à l'aide de Log4J
- Écrire dans les CloudWatch journaux à l'aide de SLF4J

Écrire dans les CloudWatch journaux à l'aide de Log4J

1. Ajoutez les dépendances suivantes au fichier de `pom.xml` votre application :

```
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-api</artifactId>
    <version>2.6.1</version>
</dependency>
<dependency>
    <groupId>org.apache.logging.log4j</groupId>
    <artifactId>log4j-core</artifactId>
    <version>2.6.1</version>
</dependency>
```

2. Incluez l'objet de la bibliothèque :

```
import org.apache.logging.log4j.Logger;
```

3. Instanciez l'objet `Logger` en lui transmettant votre classe d'application :

```
private static final Logger log =
 LogManager.getLogger.getLogger(YourApplicationClass.class);
```

4. Écrivez dans le journal en utilisant `log.info`. Un grand nombre de messages sont écrits dans le journal de l'application. Pour faciliter le filtrage de vos messages personnalisés, utilisez le niveau du journal `INFO` de l'application.

```
log.info("This message will be written to the application's CloudWatch log");
```

L'application écrit un enregistrement dans le journal contenant un message similaire au message suivant :

```
{
  "locationInformation": "com.amazonaws.services.managed-
flink.StreamingJob.main(StreamingJob.java:95)",
  "logger": "com.amazonaws.services.managed-flink.StreamingJob",
  "message": "This message will be written to the application's CloudWatch log",
  "threadName": "Flink-DispatcherRestEndpoint-thread-2",
  "applicationARN": "arn:aws:kinesisanalyticsus-east-1:123456789012:application/test",
  "applicationVersionId": "1", "messageSchemaVersion": "1",
  "messageType": "INFO"
}
```

Écrire dans les CloudWatch journaux à l'aide de SLF4J

1. Ajoutez la dépendance suivante au fichier de `pom.xml` votre application :

   ```
   <dependency>
       <groupId>org.slf4j</groupId>
       <artifactId>slf4j-log4j12</artifactId>
       <version>1.7.7</version>
       <scope>runtime</scope>
   </dependency>
   ```

2. Incluez les objets de la bibliothèque :

   ```
   import org.slf4j.Logger;
   import org.slf4j.LoggerFactory;
   ```

3. Instanciez l'objet `Logger` en lui transmettant votre classe d'application :

   ```
   private static final Logger log =
    LoggerFactory.getLogger(YourApplicationClass.class);
   ```

4. Écrivez dans le journal en utilisant `log.info`. Un grand nombre de messages sont écrits dans le journal de l'application. Pour faciliter le filtrage de vos messages personnalisés, utilisez le niveau du journal `INFO` de l'application.

   ```
   log.info("This message will be written to the application's CloudWatch log");
   ```

L'application écrit un enregistrement dans le journal contenant un message similaire au message suivant :

```
{
  "locationInformation": "com.amazonaws.services.managed-
flink.StreamingJob.main(StreamingJob.java:95)",
  "logger": "com.amazonaws.services.managed-flink.StreamingJob",
  "message": "This message will be written to the application's CloudWatch log",
  "threadName": "Flink-DispatcherRestEndpoint-thread-2",
  "applicationARN": "arn:aws:kinesisanalyticsus-east-1:123456789012:application/test",
  "applicationVersionId": "1", "messageSchemaVersion": "1",
  "messageType": "INFO"
}
```

Appels d’API pour la journalisation du service géré pour Apache Flink avec AWS CloudTrail

Le service géré pour Apache Flink est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans Managed Service for Apache Flink. CloudTrail capture tous les appels d'API pour Managed Service for Apache Flink sous forme d'événements. Les appels capturés incluent les appels à partir de la console du service géré pour Apache Flink et les appels de code vers les opérations d'API du service géré pour Apache Flink. Si vous créez un suivi, vous pouvez activer la diffusion continue d' CloudTrail événements vers un compartiment Amazon S3, y compris des événements pour le service géré pour Apache Flink. Si vous ne configurez pas de suivi, vous pouvez toujours consulter les événements les plus récents dans la CloudTrail console dans Historique des événements. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite au service géré pour Apache Flink, l'adresse IP à partir de laquelle la demande a été faite, qui a fait la demande, quand elle a été faite et des informations supplémentaires.

Pour en savoir plus CloudTrail, consultez le guide de AWS CloudTrail l'utilisateur.

Service géré pour les informations d'Apache Flink dans CloudTrail

CloudTrail est activé sur votre AWS compte lorsque vous le créez. Lorsqu'une activité se produit dans Managed Service for Apache Flink, cette activité est enregistrée dans un CloudTrail événement avec d'autres événements de AWS service dans l'historique des événements. Vous pouvez afficher, rechercher et télécharger les événements récents dans votre compte AWS. Pour plus d'informations, consultez la section Affichage des événements à l'aide de l'historique des CloudTrail événements.

Pour un enregistrement continu des événements dans votre compte AWS, notamment des événements pour le service géré pour Apache Flink, créez un journal d'activité. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un journal de suivi dans la console, il s'applique à toutes les 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 Amazon 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. Pour en savoir plus, consultez les ressources suivantes :

- 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 de plusieurs régions et réception de fichiers CloudTrail journaux de plusieurs comptes

Toutes les actions du service géré pour Apache Flink sont enregistrées CloudTrail et documentées dans la référence de l'API du service géré pour Apache Flink. Par exemple, les appels aux `UpdateApplication` actions `CreateApplication` et génèrent des entrées dans les fichiers CloudTrail journaux.

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 effectuée avec les informations d'identification utilisateur racine ou AWS Identity and Access Management (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 effectuée par un autre service AWS.

Pour plus d'informations, consultez la section Élément userIdentity CloudTrail .

Compréhension des entrées du fichier journal du service géré pour Apache Flink

Un suivi est une configuration qui permet de transmettre des événements sous forme de fichiers journaux à un compartiment Amazon S3 que vous spécifiez. 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 DescribeApplicationactions AddApplicationCloudWatchLoggingOptionet.

```
{
    "Records": [
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::012345678910:user/Alice",
                "accountId": "012345678910",
                "accessKeyId": "EXAMPLE_KEY_ID",
                "userName": "Alice"
            },
            "eventTime": "2019-03-07T01:19:47Z",
            "eventSource": "kinesisanlaytics.amazonaws.com",
            "eventName": "AddApplicationCloudWatchLoggingOption",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
            "requestParameters": {
                "applicationName": "cloudtrail-test",
                "currentApplicationVersionId": 1,
                "cloudWatchLoggingOption": {
                    "logStreamARN": "arn:aws:logs:us-east-1:012345678910:log-
group:cloudtrail-test:log-stream:flink-cloudwatch"
                }
            },
            "responseElements": {
                "cloudWatchLoggingOptionDescriptions": [
                    {
                        "cloudWatchLoggingOptionId": "2.1",
                        "logStreamARN": "arn:aws:logs:us-east-1:012345678910:log-
group:cloudtrail-test:log-stream:flink-cloudwatch"
                    }
                ],
```

```
                "applicationVersionId": 2,
                "applicationARN": "arn:aws:kinesisanalyticsus-
east-1:012345678910:application/cloudtrail-test"
            },
            "requestID": "18dfb315-4077-11e9-afd3-67f7af21e34f",
            "eventID": "d3c9e467-db1d-4cab-a628-c21258385124",
            "eventType": "AwsApiCall",
            "apiVersion": "2018-05-23",
            "recipientAccountId": "012345678910"
        },
        {
            "eventVersion": "1.05",
            "userIdentity": {
                "type": "IAMUser",
                "principalId": "EX_PRINCIPAL_ID",
                "arn": "arn:aws:iam::012345678910:user/Alice",
                "accountId": "012345678910",
                "accessKeyId": "EXAMPLE_KEY_ID",
                "userName": "Alice"
            },
            "eventTime": "2019-03-12T02:40:48Z",
            "eventSource": "kinesisanlaytics.amazonaws.com",
            "eventName": "DescribeApplication",
            "awsRegion": "us-east-1",
            "sourceIPAddress": "127.0.0.1",
            "userAgent": "aws-sdk-java/unknown-version Linux/x.xx",
            "requestParameters": {
                "applicationName": "sample-app"
            },
            "responseElements": null,
            "requestID": "3e82dc3e-4470-11e9-9d01-e789c4e9a3ca",
            "eventID": "90ffe8e4-9e47-48c9-84e1-4f2d427d98a5",
            "eventType": "AwsApiCall",
            "apiVersion": "2018-05-23",
            "recipientAccountId": "012345678910"
        }
    ]
}
```

Réglage des performances dans le service géré Amazon pour Apache Flink

Cette rubrique décrit des techniques pour surveiller et améliorer les performances de votre application de service géré pour Apache Flink.

Rubriques

- Dépannage des performances
- Bonnes pratiques en matière de performances
- Surveillance des performances

Dépannage des performances

Cette section contient une liste de symptômes que vous pouvez vérifier pour diagnostiquer et résoudre les problèmes de performances.

Si votre source de données est un flux Kinesis, les problèmes de performances se présentent généralement sous la forme d'une métrique `millisbehindLatest` élevée ou croissante. Pour les autres sources, vous pouvez vérifier une métrique similaire qui représente le retard de lecture depuis la source.

Le chemin des données

Lorsque vous étudiez un problème de performance lié à votre application, considérez le chemin complet emprunté par vos données. Les composants d'application suivants peuvent devenir des goulots d'étranglement en termes de performances et créer une contre-pression s'ils ne sont pas correctement conçus ou fournis :

- Sources de données et destinations : assurez-vous que les ressources externes avec lesquelles votre application interagit sont correctement provisionnées en fonction du débit dont bénéficiera votre application.
- Données d'état : assurez-vous que votre application n'interagit pas trop fréquemment avec le magasin d'état.

  Vous pouvez optimiser le sérialiseur utilisé par votre application. Le sérialiseur Kryo par défaut peut gérer n'importe quel type sérialisable, mais vous pouvez utiliser un sérialiseur plus performant si

votre application ne stocke des données que dans des types POJO. Pour plus d'informations sur les sérialiseurs Apache Flink, consultez Data Types & Serialization dans la documentation Apache Flink.

- Opérateurs : assurez-vous que la logique métier mise en œuvre par vos opérateurs n'est pas trop compliquée ou que vous ne créez ni n'utilisez pas de ressources pour chaque enregistrement traité. Assurez-vous également que votre application ne crée pas de fenêtres défilantes ou bascule trop fréquemment.

Solutions de résolution des problèmes de performances

Cette section contient des solutions potentielles aux problèmes de performances.

Rubriques

- Niveaux de surveillance CloudWatch
- Métriques du processeur d'application
- Parallélisme de l’application
- Journalisation d'application
- Parallélisme de l’opérateur
- Logique de l'application
- Mémoire d'application

Niveaux de surveillance CloudWatch

Vérifiez que les niveaux de surveillance de CloudWatch ne sont pas définis sur un paramètre trop détaillé.

Le paramètre de niveau de surveillance des journaux `Debug` génère une grande quantité de trafic, ce qui peut créer une contre-pression. Vous ne devez l'utiliser que lorsque vous étudiez activement les problèmes liés à l'application.

Si votre application possède un paramètre `Parallelism` élevé, l'utilisation du niveau de surveillance des métriques `Parallelism` générera également un volume de trafic important pouvant entraîner une contre-pression. Utilisez ce niveau de métrique uniquement lorsque le `Parallelism` pour votre application est faible ou lorsque vous étudiez des problèmes liés à l'application.

Pour de plus amples informations, veuillez consulter Niveaux de surveillance des applications.

Métriques du processeur d'application

Vérifiez la métrique CPU de l'application. Si cette métrique est supérieure à 75 %, vous pouvez autoriser l'application à s'allouer davantage de ressources en activant l’autoscaling.

Si l’autoscaling est activé, l'application alloue davantage de ressources si l'utilisation du processeur est supérieure à 75 % pendant 15 minutes. Pour plus d'informations sur la mise à l’échelle, consultez la section Gérer correctement la mise à l'échelle suivante et Mise à l'échelle.

> **Note**
>
> Une application n’utilise l’autoscaling qu'en fonction de l'utilisation du processeur. L'application n’utilisera pas l’autoscaling en réponse à d'autres métriques du système, tels que l’`heapMemoryUtilization`. Si votre application utilise beaucoup d'autres métriques, augmentez manuellement le parallélisme de votre application.

Parallélisme de l’application

Augmentez le parallélisme de l'application. Vous mettez à jour le parallélisme de l'application à l'aide du paramètre `ParallelismConfigurationUpdate` de l'action UpdateApplication.

Le nombre maximal de KPU pour une application est de 64 par défaut et peut être augmenté en demandant une augmentation de limite.

Il est également important d'attribuer le parallélisme à chaque opérateur en fonction de sa charge de travail, plutôt que de simplement augmenter le parallélisme de l’application. Consultez Parallélisme de l’opérateur ci-dessous.

Journalisation d'application

Vérifiez si l'application enregistre une entrée pour chaque enregistrement en cours de traitement. La rédaction d'une entrée de journal pour chaque enregistrement pendant les périodes où le débit de l'application est élevé entraîne de graves blocages dans le traitement des données. Pour vérifier cette condition, recherchez dans vos journaux les entrées de journal que votre application écrit avec chaque enregistrement qu'elle traite. Pour plus d'informations sur la lecture des journaux d’application, consultez the section called “Analyse des journaux”.

Parallélisme de l’opérateur

Vérifiez que la charge de travail de votre application est répartie uniformément entre les processus de travail.

Pour plus d'informations sur le réglage de la charge de travail des opérateurs de votre application, consultez Mise à l’échelle des opérateurs.

Logique de l'application

Examinez la logique de votre application pour détecter les opérations inefficaces ou non performantes, telles que l'accès à une dépendance externe (telle qu'une base de données ou un service Web), l'accès à l'état de l'application, etc. Une dépendance externe peut également nuire aux performances si elle n'est pas performante ou si elle n'est pas accessible de manière fiable, ce qui peut pousser la dépendance externe à renvoyer des erreurs `HTTP 500`.

Si votre application utilise une dépendance externe pour enrichir ou traiter les données entrantes, envisagez plutôt d'utiliser des E/S asynchrones. Pour de plus amples informations, veuillez consulter Async I/O dans la documentation Apache Flink.

Mémoire d'application

Vérifiez que votre application ne présente aucune fuite de ressources. Si votre application n'élimine pas correctement les threads ou la mémoire, il est possible que les métriques `millisbehindLatest`, `CheckpointSize` et `CheckpointDuration` connaissent des pics ou augmentent progressivement. Cette condition peut également entraîner des échecs du gestionnaire de tâches.

Bonnes pratiques en matière de performances

Cette section décrit les considérations spéciales relatives à la conception d'une application axée sur les performances.

Gérer correctement la mise à l'échelle

Cette section contient des informations sur la gestion de la mise à l’échelle au niveau de l'application et au niveau de l'opérateur.

Cette section contient les rubriques suivantes :

- Gérer correctement la mise à l’échelle des applications
- Gérez correctement la mise à l’échelle des opérateurs

Gérer correctement la mise à l’échelle des applications

Vous pouvez utiliser la mise à l’autoscaling pour gérer les pics inattendus d'activité des applications. Les KPU de votre application augmenteront automatiquement si les critères suivants sont réunis :

- L’autoscaling est activé pour l'application.
- L'utilisation du processeur reste supérieure à 75 % pendant 15 minutes.

Si l’autoscaling est activé, mais que l'utilisation du processeur ne reste pas à ce seuil, l'application n'augmentera pas les KPU. Si vous constatez un pic d'utilisation du processeur qui n'atteint pas ce seuil, ou un pic lié à une métrique d'utilisation différente, par exemple `heapMemoryUtilization`, augmentez la mise à l’échelle manuellement pour permettre à votre application de gérer les pics d'activité.

> **Note**
>
> Si l'application a automatiquement ajouté des ressources supplémentaires par le biais de l’autoscaling, l'application libérera les nouvelles ressources après une période d'inactivité. La réduction des ressources affectera temporairement les performances.

Pour plus d'informations sur la mise à l’échelle, consultez Mise à l'échelle.

Gérez correctement la mise à l’échelle des opérateurs

Vous pouvez améliorer les performances de votre application en vérifiant que la charge de travail de votre application est répartie uniformément entre les processus de travail et que les opérateurs de votre application disposent des ressources système dont ils ont besoin pour être stables et performants.

Vous pouvez définir le parallélisme pour chaque opérateur du code de votre application à l'aide du paramètre `parallelism`. Si vous ne définissez pas le parallélisme pour un opérateur, celui-ci utilisera le paramètre de parallélisme au niveau de l'application. Les opérateurs qui utilisent le paramètre de parallélisme au niveau de l'application peuvent potentiellement utiliser toutes les ressources système disponibles pour l'application, ce qui la rend instable.

Pour déterminer au mieux le parallélisme de chaque opérateur, considérez les besoins relatifs en ressources de l'opérateur par rapport aux autres opérateurs de l'application. Définissez un paramètre de parallélisme des opérateurs plus gourmands en ressources pour les opérateurs plus gourmands que pour les opérateurs moins gourmands.

Le parallélisme total des opérateurs de l'application est la somme du parallélisme de tous les opérateurs de l'application. Vous ajustez le parallélisme total des opérateurs pour votre application en déterminant le meilleur rapport entre celui-ci et le nombre total d'emplacements de tâches disponibles pour votre application. Un rapport stable typique entre le parallélisme total des opérateurs et les emplacements de tâches est de 4:1, c'est-à-dire que l'application dispose d'un emplacement de tâches disponible pour quatre sous-tâches d'opérateur disponibles. Une application avec des opérateurs plus gourmands en ressources peut avoir besoin d'un ratio de 3:1 ou 2:1, tandis qu'une application avec des opérateurs moins gourmands en ressources peut être stable avec un ratio de 10:1.

Vous pouvez définir le ratio pour l'opérateur à l'aide de Propriétés d'exécution, afin de pouvoir ajuster le parallélisme de l'opérateur sans compiler ni télécharger le code de votre application.

L'exemple de code suivant montre comment définir le parallélisme des opérateurs en tant que ratio réglable du parallélisme de l'application actuelle :

```
Map<String, Properties> applicationProperties =
 KinesisAnalyticsRuntime.getApplicationProperties();
operatorParallelism =
    StreamExecutionEnvironment.getParallelism() /
    Integer.getInteger(

 applicationProperties.get("OperatorProperties").getProperty("MyOperatorParallelismRatio")
    );
```

Pour plus d'informations sur les sous-tâches, les emplacements de tâches et les autres ressources de l'application, consultez Ressources d'application.

Pour contrôler la répartition de la charge de travail entre les processus de travail de votre application, utilisez le paramètre `Parallelism` et la méthode de partition `KeyBy`. Pour plus d'informations, consultez les rubriques suivantes dans la documentation Apache Flink :

- Exécution en parallèle
- Transformations de DataStream

Surveiller l'utilisation des ressources de dépendance externe

En cas de problème de performance dans une destination (telle que Kinesis Streams, Kinesis Data Firehose, DynamoDB ou OpenSearch Service), votre application subira une contre-pression. Vérifiez que vos dépendances externes sont correctement configurées pour le débit de votre application.

> **Note**
>
> Les échecs d'autres services peuvent entraîner des défaillances dans votre application. Si vous constatez des échecs dans votre application, consultez les journaux CloudWatch de vos services de destination pour détecter les échecs.

Exécuter votre application Apache Flink localement

Pour résoudre les problèmes de mémoire, vous pouvez exécuter votre application dans une installation locale de Flink. Cela vous donnera accès à des outils de débogage tels que la trace de pile et le vidage de tas, qui ne sont pas disponibles lors de l'exécution de votre application dans le service géré pour Apache Flink.

Pour plus d'informations sur la création d'une installation locale de Flink, consultez Local Setup Tutorial dans la documentation Apache Flink.

Surveillance des performances

Cette section décrit les outils permettant de surveiller les performances d'une application.

Surveillance des performances à l'aide des métriques CloudWatch

Vous surveillez l'utilisation des ressources, le débit, les points de contrôle et les interruptions de votre application à l'aide des métriques CloudWatch. Pour obtenir des informations sur l'utilisation des métriques CloudWatch avec votre application de service géré pour Apache Flink, veuillez consulter Métriques et dimensions dans le service géré pour Apache Flink.

Surveillance des performances à l'aide des journaux et alarmes CloudWatch

Vous surveillez les conditions d'erreur susceptibles de provoquer des problèmes de performances à l'aide des journaux CloudWatch.

Des conditions d'erreur apparaissent dans les entrées du journal lorsque l’état de la tâche Apache Flink passe de `RUNNING` à `FAILED`.

Vous utilisez les alarmes CloudWatch pour créer des notifications en cas de problèmes de performances, tels que l'utilisation des ressources ou les mesures de point de contrôle supérieures à un seuil de sécurité, ou des modifications inattendues de l'état des applications.

Pour plus d'informations sur la création d’alarmes CloudWatch pour l’application de service géré pour Apache Flink consultez Alertes.

Service géré pour Apache Flink et quota de blocs-notes Studio

Lorsque vous travaillez avec le service géré Amazon pour Apache Flink, prenez en compte le quota suivant :

- Vous pouvez créer jusqu'à 50 services gérés pour les applications Apache Flink par région dans votre compte. Vous pouvez demander des applications supplémentaires via le formulaire d'augmentation de quota de service. Pour plus d'informations, consultez le Centre AWS Support.

  Pour obtenir la liste des régions qui prennent en charge le service géré pour Apache Flink, consultez Régions et points de terminaison du service géré pour Apache Flink.

- Le nombre d'unités de traitement Kinesis (KPU) est limité à 64 par défaut. Pour des instructions pour demander une augmentation de ce quota, consultez Pour demander une augmentation de quota dans Service Quotas. Assurez-vous de spécifier le préfixe d'application auquel la nouvelle limite de KPU doit être appliquée.

  Avec le service géré pour Apache Flink, votre compte AWS est débité pour les ressources allouées, plutôt que pour les ressources utilisées par votre application. Vous êtes facturé selon un taux horaire basé sur le nombre maximal d'unités KPU utilisées pour exécuter votre application de traitement de flux. Une seule unité KPU vous fournit 1 vCPU et 4 GiO de mémoire. Pour chaque KPU, le service fournit également 50 GiO de stockage des applications en cours d'exécution.

- Vous pouvez créer jusqu'à 1 000 services gérés pour Apache Flink Instantanés par application.

- Vous pouvez attribuer jusqu'à 50 balises par application.

- La taille maximale d'un fichier JAR d'application est de 512 MiO. Si vous dépassez ce quota, votre application ne pourra pas démarrer.

Pour les blocs-notes Studio, les quotas suivants s'appliquent. Pour demander des quotas plus élevés, créez un dossier de support.

- `websocketMessageSize` = 5 MiO
- `noteSize` = 5 MiO
- `noteCount` = 1 000
- `Max cumulative UDF size` = 100 MiO
- `Max cumulative dependency jar size` = 300 MiO

Maintenance du service géré pour Apache Flink

Le service géré pour Apache Flink corrige régulièrement vos applications avec des mises à jour de sécurité du système d'exploitation et des mises à jour de sécurité des images de conteneur afin de garantir la conformité et d'atteindre les objectifs de sécurité AWS. Le tableau suivant répertorie la période par défaut pendant laquelle le service géré pour Apache Flink effectue ce type de maintenance. La maintenance de votre application peut avoir lieu à tout moment pendant la période correspondant à votre région. Votre application peut subir une interruption de 10 à 30 secondes au cours de ce processus de maintenance. Cependant, la durée réelle de l'interruption dépend de l'état de l'application. Pour plus d'informations sur la manière de minimiser l'impact de ces interruptions, consultez the section called “Tolérance aux pannes : points de contrôle et points de sauvegarde”.

Pour modifier la période pendant laquelle le service géré pour Apache Flink effectue la maintenance de votre application, utilisez l'API UpdateApplicationMaintenanceConfiguration.

Région	Fenêtre horaire de maintenance
AWS GovCloud (US, côte ouest)	06:00–14:00 UTC
AWS GovCloud (US, côte est)	03:00–11:00 UTC
USA Est (Virginie du Nord)	03:00–11:00 UTC
USA Est (Ohio)	03:00–11:00 UTC
USA Ouest (Californie du Nord)	06:00–14:00 UTC
USA Ouest (Oregon)	06:00–14:00 UTC
Asie-Pacifique (Hong Kong)	13:00–21:00 UTC
Asie-Pacifique (Mumbai)	16:30–00:30 UTC
Asie-Pacifique (Hyderabad)	16:30–00:30 UTC
Asie-Pacifique (Séoul)	13:00–21:00 UTC
Asie-Pacifique (Singapour)	14:00–22:00 UTC

Région	Fenêtre horaire de maintenance
Asie-Pacifique (Sydney)	12:00–20:00 UTC
Asie-Pacifique (Jakarta)	15:00–23:00 UTC
Asie Pacifique (Tokyo)	13:00–21:00 UTC
Canada (Centre)	03:00–11:00 UTC
Chine (Beijing)	13:00–21:00 UTC
Chine (Ningxia)	13:00–21:00 UTC
Europe (Francfort)	06:00–14:00 UTC
Europe (Zurich)	20:00–04:00 UTC
Europe (Irlande)	22:00–06:00 UTC
Europe (Londres)	22:00–06:00 UTC
Europe (Stockholm)	23:00–07:00 UTC
Europe (Milan)	21:00–05:00 UTC
Europe (Espagne)	21:00–05:00 UTC
Afrique (Le Cap)	20:00–04:00 UTC
Europe (Irlande)	22:00–06:00 UTC
Europe (Londres)	23:00–07:00 UTC
Europe (Paris)	23:00–07:00 UTC
Europe (Stockholm)	23:00–07:00 UTC
Moyen-Orient (Bahreïn)	13:00–21:00 UTC
Moyen-Orient (EAU)	18:00–02:00 UTC

Région	Fenêtre horaire de maintenance
Amérique du Sud (São Paulo)	19:00–03:00 UTC
Israël (Tel Aviv)	20:00–04:00 UTC

Définir un UUID pour tous les opérateurs

Lorsque le service géré pour Apache Flink lance une tâche Flink pour une application avec un instantané, la tâche Flink peut ne pas démarrer en raison de certains problèmes. L'un d'eux est la non-concordance des identifiants d'opérateur. Flink attend des identifiants d'opérateur explicites et cohérents pour les opérateurs de graphiques de tâches Flink. S'il n'est pas défini explicitement, Flink génère automatiquement un identifiant pour les opérateurs. Cela est dû au fait que Flink utilise ces identifiants d'opérateur pour identifier de manière unique les opérateurs dans un graphe de tâches et les utilise pour stocker l'état de chaque opérateur dans un point de sauvegarde.

Le problème de non-concordance des identifiants d'opérateur se produit lorsque Flink ne trouve pas de correspondance 1:1 entre les identifiants d'opérateur d'un graphique de tâches et les identifiants d'opérateur définis dans un point de sauvegarde. Cela se produit lorsque des identifiants d'opérateur cohérents et explicites ne sont pas définis et que Flink génère automatiquement des identifiants d'opérateur qui peuvent ne pas être cohérents à chaque création de graphique de tâches. La probabilité que les applications rencontrent ce problème est élevée lors des opérations de maintenance. Pour éviter cela, nous recommandons aux clients de définir l'UUID pour tous les opérateurs en code Flink. Pour plus d'informations, consultez la rubrique Définir un UUID pour tous les opérateurs dans la section Préparation à la production.

Préparation à la production

Il s'agit d'un ensemble d'aspects importants de l'exécution d'applications de production sur le service géré pour Apache Flink. Il ne s'agit pas d'une liste exhaustive, mais du strict minimum de ce à quoi vous devez faire attention avant de mettre une application en production.

Tests de charge des applications

Certains problèmes liés aux applications ne se manifestent que sous une charge importante. Nous avons vu des cas où des applications semblaient saines et où un événement opérationnel a considérablement amplifié la charge sur l'application. Cela peut se produire indépendamment de l'application elle-même : si la source de données ou le récepteur de données n'est pas disponible pendant quelques heures, l'application Flink ne peut pas progresser. Une fois ce problème résolu, une file d’attente de données non traitées s'accumule et peut épuiser complètement les ressources disponibles. La charge peut alors amplifier les bogues ou les problèmes de performance qui ne sont pas apparus auparavant.

Il est donc essentiel d'effectuer des tests de charge appropriés pour les applications de production. Les questions auxquelles il convient de répondre lors de ces tests de charge sont les suivantes :

- L'application est-elle stable sous une charge élevée prolongée ?
- L'application peut-elle toujours prendre un point de sauvegarde en cas de charge maximale ?
- Combien de temps faut-il pour traiter une file d’attente d'une heure ? Et pendant combien de temps pour 24 heures (en fonction de la rétention maximale des données dans le flux) ?
- Le débit de l'application augmente-t-il lorsque l'application est mise à l'échelle ?

Lors de la consommation à partir d'un flux de données, ces scénarios peuvent être simulés en les intégrant au flux pendant un certain temps. Démarrez ensuite l'application et faites-lui consommer les données depuis le début du temps, par exemple, utilisez une position de départ de `TRIM_HORIZON` dans le cas d'un flux de données Kinesis.

Parallélisme max.

Le parallélisme max définit le parallélisme maximal qu'une application avec état peut atteindre. Ceci est défini lorsque l'état est créé pour la première fois et il n'existe aucun moyen de mettre à l’échelle l'opérateur au-delà de ce maximum sans supprimer l'état.

Le parallélisme maximal est défini lorsque l'état est créé pour la première fois.

Par défaut, le paramètre max est défini sur :

- 128, si le parallélisme est <= 128
- `MIN(nextPowerOfTwo(parallelism + (parallelism / 2)), 2^15)` : si le parallélisme est > 128

Si vous prévoyez de mettre à l'échelle votre application à un parallélisme > 128, vous devez définir explicitement le parallélisme maximal.

Le parallélisme maximal peut être défini au niveau de l'application, avec `env.setMaxParallelism(x)` ou un seul opérateur. Sauf indication contraire, tous les opérateurs héritent du parallélisme maximal de l'application.

Pour plus d'informations, consultez Set An Explicit Max Parallelism dans la documentation Flink.

Définir un UUID pour tous les opérateurs

Un UUID est utilisé dans l'opération au cours de laquelle Flink mappe un point de sauvegarde à un opérateur individuel. La définition d'un UUID spécifique pour chaque opérateur fournit un mappage stable pour le processus de restauration du point de sauvegarde.

```
.map(...).uid("my-map-function")
```

Pour plus d'informations, consultez Production Readiness Checklist.

Bonnes pratiques pour le service géré pour Apache Flink

Cette section contient des informations et des recommandations pour développer des applications de service géré pour Apache Flink stables et performantes.

Rubriques

- Tolérance aux pannes : points de contrôle et points de sauvegarde
- Versions de connecteurs non prises en charge
- Performances et parallélisme
- Configuration du parallélisme par opérateur
- Journalisation
- Codage
- Gestion des informations d'identification
- Lecture à partir de sources contenant peu de partitions
- Intervalle d'actualisation des blocs-notes Studio
- Performances optimales du bloc-notes Studio
- Comment les stratégies de filigrane et les partitions inactives affectent les fenêtres temporelles
- Définir un UUID pour tous les opérateurs
- Ajouter ServiceResourceTransformer au plugin Maven Shade

Tolérance aux pannes : points de contrôle et points de sauvegarde

Utilisez des points de contrôle et des points de sauvegarde pour implémenter la tolérance aux pannes dans votre application de service géré pour Apache Flink. Gardez les points suivants à l'esprit lorsque vous développez et maintenez votre application :

- Nous vous recommandons de laisser le point de contrôle activé pour votre application. Le point de contrôle assure la tolérance aux pannes de votre application lors de la maintenance planifiée, ainsi qu'en cas d'échecs inattendus dus à des problèmes de service, à des défaillances de dépendance des applications ou à d'autres problèmes. Pour des informations sur la maintenance, veuillez consulter Maintenance.
- Réglez ApplicationSnapshotConfiguration: : SnapshotsEnabled sur `false` pendant le développement ou le dépannage de l'application. Un instantané est créé à chaque arrêt de

l'application, ce qui peut entraîner des problèmes si l'application est en mauvais état ou si elle n'est pas performante. Définissez `SnapshotsEnabled` sur `true` une fois que l'application est en production et qu'elle est stable.

> **Note**
>
> Nous recommandons que votre application crée un instantané plusieurs fois par jour pour redémarrer correctement avec des données d'état correctes. La fréquence correcte pour vos instantanés dépend de la logique métier de votre application. La prise fréquente d'instantané vous permet de récupérer des données plus récentes, mais cela augmente les coûts et nécessite davantage de ressources système.

Pour plus d'informations sur la surveillance des interruptions d'application, consultez Métriques et dimensions dans le service géré pour Apache Flink.

Pour plus d'informations sur la tolérance aux pannes d'implémentation, consultez Tolérance aux pannes.

Versions de connecteurs non prises en charge

La version 1.15 du service géré pour Apache Flink empêchera automatiquement le démarrage ou la mise à jour des applications si elles utilisent des versions de Kinesis Connector non prises en charge (regroupées dans les fichiers JAR de l'application). Lors de la mise à niveau vers la version 1.15 du service géré pour Apache Flink, assurez-vous que vous utilisez le connecteur Kinesis le plus récent. Il s'agit de toute version égale ou ultérieure à la version 1.15.2. Toutes les autres versions ne seront pas prises en charge par le service géré pour Apache Flink, car elles peuvent entraîner des problèmes de cohérence ou des échecs avec la fonctionnalité Arrêt avec point de sauvegarde, empêchant les opérations d'arrêt/de mise à jour.

Performances et parallélisme

Votre application peut être mise à l'échelle pour répondre à tous les niveaux de débit en ajustant le parallélisme de vos applications et en évitant les problèmes de performances. Gardez les points suivants à l'esprit lorsque vous développez et maintenez votre application :

- Vérifiez que toutes les sources et tous les récepteurs de votre application sont suffisamment approvisionnés et ne sont pas limités. Si les sources et les récepteurs sont d'autres AWS services, surveillez l'utilisation de ces services CloudWatch.
- Pour les applications présentant un parallélisme très élevé, vérifiez si les niveaux élevés de parallélisme sont appliqués à tous les opérateurs de l'application. Par défaut, Apache Flink applique le même parallélisme d'application à tous les opérateurs du graphique d'application. Cela peut entraîner des problèmes d'approvisionnement sur les sources ou les récepteurs, ou des blocages dans le traitement des données par les opérateurs. Vous pouvez modifier le parallélisme de chaque opérateur dans le code avec SetParallelism.
- Cherchez à comprendre la signification des paramètres de parallélisme pour les opérateurs de votre application. Si vous modifiez le parallélisme d'un opérateur, il se peut que vous ne puissiez pas restaurer l'application à partir d'un instantané créé alors que l'opérateur avait un parallélisme incompatible avec les paramètres actuels. Pour plus d'informations sur la définition du parallélisme des opérateurs, consultez Set maximum parallelism for operators explicitly.

Pour plus d'informations sur l'implémentation de la mise à l'échelle, consultez Mise à l'échelle.

Configuration du parallélisme par opérateur

Par défaut, le parallélisme est défini pour tous les opérateurs au niveau de l'application. Vous pouvez annuler le parallélisme d'un seul opérateur à l'aide de l'API à l'aide de DataStream . `.setParallelism(x)` Vous pouvez définir le parallélisme d'un opérateur sur n'importe quel parallélisme égal ou inférieur au parallélisme de l'application.

Si possible, définissez le parallélisme des opérateurs en fonction du parallélisme de l'application. De cette façon, le parallélisme des opérateurs variera en fonction du parallélisme de l'application. Si vous utilisez l'autoscaling, par exemple, le parallélisme de tous les opérateurs variera dans les mêmes proportions :

```
int appParallelism = env.getParallelism();
...
...ops.setParalleism(appParallelism/2);
```

Dans certains cas, vous pouvez définir le parallélisme de l'opérateur sur une constante. Par exemple, définir le parallélisme d'un flux Kinesis source en fonction du nombre de partitions. Dans ces cas, vous devez envisager de transmettre le parallélisme de l'opérateur comme paramètre de

configuration de l'application, afin de le modifier sans modifier le code, si vous devez, par exemple, redéfinir le flux source.

Journalisation

Vous pouvez surveiller les performances de votre application et les conditions d'erreur à l'aide CloudWatch des journaux. Gardez les points suivants à l'esprit lorsque vous configurez la journalisation pour votre application :

- Activez la CloudWatch journalisation de l'application afin que tout problème d'exécution puisse être résolu.
- Ne créez pas d'entrée de journal pour chaque enregistrement traité dans l'application. Cela cause de graves blocages lors du traitement et peut entraîner une contre-pression dans le traitement des données.
- Créez des CloudWatch alarmes pour vous avertir lorsque votre application ne fonctionne pas correctement. Pour de plus amples informations, veuillez consulter la page Alertes.

Pour plus d'informations sur l'implémentation de la journalisation, consultez Journalisation et surveillance.

Codage

Vous pouvez rendre votre application performante et stable en utilisant les pratiques de programmation recommandées. Gardez les points suivants à l'esprit lorsque vous écrivez le code d'application :

- N'utilisez pas `system.exit()` dans le code de votre application, ni dans la méthode `main` de votre application, ni dans les fonctions définies par l'utilisateur. Si vous souhaitez arrêter votre application depuis le code, lancez une exception dérivée de `Exception` ou `RuntimeException` contenant un message indiquant le problème rencontré par l'application.

  Notez ce qui suit concernant la façon dont le service gère cette exception :

  - Si l'exception provient de la méthode `main` de votre application, le service l'intégrera dans une `ProgramInvocationException` lorsque l'application passera à l'état `RUNNING`, et le gestionnaire de tâches ne soumettra pas la tâche.
  - Si l'exception provient d'une fonction définie par l'utilisateur, le gestionnaire de tâches échouera et la redémarrera, et les détails de l'exception seront écrits dans le journal des exceptions.

- Pensez à griser le fichier JAR de votre application et ses dépendances incluses. Le grisage est recommandé en cas de conflits potentiels entre les noms de packages de votre application et l'exécution Apache Flink. En cas de conflit, les journaux de votre application peuvent contenir une exception de type `java.util.concurrent.ExecutionException`. Pour plus d'informations sur le grisage du fichier JAR de votre application, consultez Apache Maven Shade Plugin.

Gestion des informations d'identification

Vous ne devez pas intégrer d'informations d'identification à long terme à des applications de production (ou à toute autre application). Les informations d'identification à long terme sont susceptibles d'être enregistrées dans un système de contrôle de version et peuvent facilement être perdues. Vous pouvez plutôt associer un rôle à l'application de service géré pour Apache Flink et accorder des privilèges à ce rôle. L'application Flink en cours d'exécution peut ensuite récupérer des informations d'identification temporaires avec les privilèges correspondants auprès de l'environnement. Si l'authentification est nécessaire pour un service qui n'est pas intégré nativement à IAM, par exemple une base de données qui nécessite un nom d'utilisateur et un mot de passe pour l'authentification, vous devez envisager de stocker des secrets dans AWS Secrets Manager.

De nombreux services AWS natifs prennent en charge l'authentification :

- Kinesis Data Streams — .java ProcessTaxiStream
- Amazon MSK — https://github.com/aws/ aws-msk-iam-auth using-the-amazon-msk /# - library-for-iam-authentication
- Amazon Elasticsearch Service — .java AmazonElasticsearchSink
- Amazon S3 : fonctionne immédiatement sur le service géré pour Apache Flink

Lecture à partir de sources contenant peu de partitions

Lors de la lecture depuis Apache Kafka ou un flux de données Kinesis, il peut y avoir un décalage entre le parallélisme du flux (c'est-à-dire le nombre de partitions pour Kafka et le nombre de partitions pour Kinesis) et le parallélisme de l'application. Avec une conception naïve, le parallélisme d'une application ne peut pas dépasser le parallélisme d'un flux : chaque sous-tâche d'un opérateur source ne peut lire qu'à partir d'une ou plusieurs partitions. Cela signifie que pour un flux contenant seulement 2 partitions et une application avec un parallélisme de 8, seules deux sous-tâches consomment réellement du flux et 6 sous-tâches restent inactives. Cela peut limiter considérablement

le débit de l'application, en particulier si la désérialisation est coûteuse et réalisée par la source (ce qui est le cas par défaut).

Pour atténuer cet effet, vous pouvez redimensionner le flux. Toutefois, cela n'est pas toujours souhaitable ou possible. Vous pouvez également restructurer la source afin qu'elle n'effectue aucune sérialisation et qu'elle transmette simplement le `byte[]`. Vous pouvez ensuite rééquilibrer les données pour les répartir uniformément entre toutes les tâches, puis les désérialiser. De cette façon, vous pouvez tirer parti de toutes les sous-tâches pour la désérialisation et cette opération potentiellement coûteuse n'est plus limitée par le nombre de partitions du flux.

Intervalle d'actualisation des blocs-notes Studio

Si vous modifiez l'intervalle d'actualisation des résultats des paragraphes, définissez-le sur une valeur d'au moins 1 000 millisecondes.

Performances optimales du bloc-notes Studio

Nous avons testé l'énoncé suivant et avons obtenu la meilleure performance lorsque `events-per-second` multiplié par `number-of-keys` était inférieur à 25 000 000. C'était pour `events-per-second` inférieur à 150 000.

```
SELECT key, sum(value) FROM key-values GROUP BY key
```

Comment les stratégies de filigrane et les partitions inactives affectent les fenêtres temporelles

Lors de la lecture d'événements provenant d'Apache Kafka et de Kinesis Data Streams, la source peut définir l'heure de l'événement en fonction des attributs du flux. Dans le cas de Kinesis, l'heure de l'événement est égale à l'heure approximative d'arrivée des événements. Mais il ne suffit pas de définir l'heure de l'événement à la source pour qu'une application Flink utilise l'heure de l'événement. La source doit également générer des filigranes qui propagent les informations sur l'heure de l'événement de la source à tous les autres opérateurs. La documentation de Flink donne un bon aperçu du fonctionnement de ce processus.

Par défaut, l'horodatage d'un événement lu par Kinesis est défini sur l'heure d'arrivée approximative déterminée par Kinesis. Une autre condition préalable pour que le temps consacré aux événements fonctionne dans l'application est une stratégie de filigrane.

```
WatermarkStrategy<String> s = WatermarkStrategy
    .<String>forMonotonousTimestamps()
    .withIdleness(Duration.ofSeconds(...));
```

La stratégie de filigrane est ensuite appliquée à un `DataStream` avec la méthode `assignTimestampsAndWatermarks`. Il existe quelques stratégies intégrées utiles :

- `forMonotonousTimestamps()` utilisera simplement l'heure de l'événement (heure d'arrivée approximative) et émettra périodiquement la valeur maximale sous forme de filigrane (pour chaque sous-tâche spécifique)
- `forBoundedOutOfOrderness(Duration.ofSeconds(...))` est similaire à la stratégie précédente, mais utilisera l'heure et la durée de l'événement pour la génération du filigrane.

Cela fonctionne, mais il y a quelques mises en garde à prendre en compte. Les filigranes sont générés au niveau d'une sous-tâche et circulent dans le graphique de l'opérateur.

Extrait de la documentation de Flink :

Chaque sous-tâche parallèle d'une fonction source génère généralement ses filigranes indépendamment. Ces filigranes définissent l'heure de l'événement sur cette source parallèle particulière.

Au fur et à mesure que les filigranes circulent dans le programme de streaming, ils font avancer l'heure de l'événement chez les opérateurs où ils arrivent. Chaque fois qu'un opérateur avance l'heure de son événement, il génère un nouveau filigrane en aval pour les opérateurs qui lui succèdent.

Certains opérateurs consomment plusieurs flux d'entrée ; une union, par exemple, ou des opérateurs suivant une fonction KeyBy (...) ou partition (...). La durée actuelle des événements d'un tel opérateur est la durée minimale des événements de ses flux d'entrée. Au fur et à mesure que ses flux d'entrée mettent à jour l'heure de leurs événements, l'opérateur le fait également.

Cela signifie que si une sous-tâche source consomme du contenu à partir d'une partition inactive, les opérateurs en aval ne reçoivent pas de nouveaux filigranes provenant de cette sous-tâche, ce qui bloque le traitement pour tous les opérateurs en aval utilisant des fenêtres temporelles. Pour éviter cela, les clients peuvent ajouter l'option `withIdleness` à la stratégie de filigrane. Avec cette option, un opérateur exclut les filigranes des sous-tâches inactives en amont lorsqu'il calcule l'heure de l'événement de l'opérateur. Les sous-tâches inactives ne bloquent donc plus l'avancement de l'heure des événements chez les opérateurs en aval.

Cependant, l'option d'inactivité avec les stratégies de filigrane intégrées n'avancera pas l'heure de l'événement si aucune sous-tâche ne lit un événement, c'est-à-dire s'il n'y a aucun événement dans le flux. Cela devient particulièrement visible dans les cas de test où un ensemble fini d'événements est lu à partir du flux. Comme l'heure de l'événement n'avance pas après la lecture du dernier événement, la dernière fenêtre (contenant le dernier événement) ne se fermera jamais.

Récapitulatif

- le paramètre `withIdleness` ne générera pas de nouveaux filigranes dans le cas où une partition est inactive, il exclura simplement le dernier filigrane envoyé par les sous-tâches inactives du calcul du filigrane minimum pour les opérateurs en aval
- avec les stratégies de filigrane intégrées, la dernière fenêtre ouverte ne se fermera jamais (à moins que de nouveaux événements faisant avancer le filigrane ne soient envoyés, mais cela crée une nouvelle fenêtre qui reste ensuite ouverte)
- même lorsque l'heure est définie par le flux Kinesis, des événements d'arrivée tardive peuvent toujours se produire si une partition est consommée plus rapidement que les autres (par exemple, lors de l'initialisation de l'application ou lors de l'utilisation de `TRIM_HORIZON` lorsque toutes les partitions existantes sont consommées en parallèle sans tenir compte de leur relation parent/enfant)
- les paramètres `withIdleness` de la stratégie de filigrane semblent déprécier les paramètres spécifiques à la source Kinesis pour les partitions inactives (`ConsumerConfigConstants.SHARD_IDLE_INTERVAL_MILLIS`

Exemple

L'application suivante lit un flux et crée des fenêtres de session en fonction de l'heure de l'événement.

```
Properties consumerConfig = new Properties();
consumerConfig.put(AWSConfigConstants.AWS_REGION, "eu-west-1");
consumerConfig.put(ConsumerConfigConstants.STREAM_INITIAL_POSITION, "TRIM_HORIZON");

FlinkKinesisConsumer<String> consumer = new FlinkKinesisConsumer<>("...", new
 SimpleStringSchema(), consumerConfig);

WatermarkStrategy<String> s = WatermarkStrategy
    .<String>forMonotonousTimestamps()
    .withIdleness(Duration.ofSeconds(15));
```

```
env.addSource(consumer)
    .assignTimestampsAndWatermarks(s)
    .map(new MapFunction<String, Long>() {
        @Override
        public Long map(String s) throws Exception {
            return Long.parseLong(s);
        }
    })
    .keyBy(l -> 0l)
    .window(EventTimeSessionWindows.withGap(Time.seconds(10)))
    .process(new ProcessWindowFunction<Long, Object, Long, TimeWindow>() {
        @Override
        public void process(Long aLong, ProcessWindowFunction<Long, Object, Long,
 TimeWindow>.Context context, Iterable<Long>iterable, Collector<Object> collector)
 throws Exception {
            long count = StreamSupport.stream(iterable.spliterator(), false).count();
            long timestamp = context.currentWatermark();

            System.out.print("XXXXXXXXXXXXXX Window with " + count + " events");
            System.out.println("; Watermark: " + timestamp + ", " +
 Instant.ofEpochMilli(timestamp));


            for (Long l : iterable) {
                System.out.println(l);
            }
        }
    });
```

Dans l'exemple suivant, 8 événements sont écrits dans un flux de 16 partitions (les 2 premiers et le dernier événement se retrouvent dans la même partition).

```
$ aws kinesis put-record --stream-name hp-16 --partition-key 1 --data MQ==
$ aws kinesis put-record --stream-name hp-16 --partition-key 2 --data Mg==
$ aws kinesis put-record --stream-name hp-16 --partition-key 3 --data Mw==
$ date

{
    "ShardId": "shardId-000000000012",
    "SequenceNumber": "49627894338614655560500811028721934184977530127978070210"
}
{
```

```
    "ShardId": "shardId-000000000012",
    "SequenceNumber": "49627894338614655560500811028795678659974022576354623682"
}
{
    "ShardId": "shardId-000000000014",
    "SequenceNumber": "49627894338659257050897872275134360684221592378842022114"
}
Wed Mar 23 11:19:57 CET 2022

$ sleep 10
$ aws kinesis put-record --stream-name hp-16 --partition-key 4 --data NA==
$ aws kinesis put-record --stream-name hp-16 --partition-key 5 --data NQ==
$ date

{
    "ShardId": "shardId-000000000010",
    "SequenceNumber": "49627894338570054070103749783042116732419934393936642210"
}
{
    "ShardId": "shardId-000000000014",
    "SequenceNumber": "49627894338659257050897872275659034489934342334017700066"
}
Wed Mar 23 11:20:10 CET 2022

$ sleep 10
$ aws kinesis put-record --stream-name hp-16 --partition-key 6 --data Ng==
$ date

{
    "ShardId": "shardId-000000000001",
    "SequenceNumber": "49627894338369347363316974173886988345467035365375213586"
}
Wed Mar 23 11:20:22 CET 2022

$ sleep 10
$ aws kinesis put-record --stream-name hp-16 --partition-key 7 --data Nw==
$ date

{
    "ShardId": "shardId-000000000008",
    "SequenceNumber": "49627894338525452579706688535878947299195189349725503618"
}
Wed Mar 23 11:20:34 CET 2022
```

```
$ sleep 60
$ aws kinesis put-record --stream-name hp-16 --partition-key 8 --data OA==
$ date

{
    "ShardId": "shardId-000000000012",
    "SequenceNumber": "49627894338614655560500811029600823255837371928900796610"
}
Wed Mar 23 11:21:27 CET 2022
```

Cette entrée doit donner lieu à 5 fenêtres de session : événement 1, 2, 3 ; événement 4, 5 ; événement 6 ; événement 7 ; événement 8. Cependant, le programme ne fournit que les 4 premières fenêtres.

```
11:59:21,529 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 5 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000006,HashKeyRange: {StartingHashKey:
 127605887595351923798765477786913079296,EndingHashKey:
 148873535527910577765226390751398592511},SequenceNumberRange: {StartingSequenceNumber:
 49627894338480851089309627289524549239292625588395704418,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,530 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 5 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000006,HashKeyRange: {StartingHashKey:
 127605887595351923798765477786913079296,EndingHashKey:
 148873535527910577765226390751398592511},SequenceNumberRange: {StartingSequenceNumber:
 49627894338480851089309627289524549239292625588395704418,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,530 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 6 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000007,HashKeyRange: {StartingHashKey:
 148873535527910577765226390751398592512,EndingHashKey:
 170141183460469231731687303715884105727},SequenceNumberRange: {StartingSequenceNumber:
 49627894338503151834508157912666084957565273949901684850,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,530 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 6 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000010,HashKeyRange: {StartingHashKey:
 212676479325586539664609129644855132160,EndingHashKey:
 233944127258145193631070042609340645375},SequenceNumberRange: {StartingSequenceNumber:
 49627894338570054070103749782090692112383219034419626146,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
```

```
11:59:21,530 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 6 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000007,HashKeyRange: {StartingHashKey:
 148873535527910577765226390751398592512,EndingHashKey:
 170141183460469231731687303715884105727},SequenceNumberRange: {StartingSequenceNumber:
 49627894338503151834508157912666084957565273949901684850,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,531 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 4 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000005,HashKeyRange: {StartingHashKey:
 106338239662793269832304564822427566080,EndingHashKey:
 127605887595351923798765477786913079295},SequenceNumberRange: {StartingSequenceNumber:
 49627894338458550344111096666383013521019977226889723986,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 4 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000005,HashKeyRange: {StartingHashKey:
 106338239662793269832304564822427566080,EndingHashKey:
 127605887595351923798765477786913079295},SequenceNumberRange: {StartingSequenceNumber:
 49627894338458550344111096666383013521019977226889723986,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 3 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000004,HashKeyRange: {StartingHashKey:
 850705917302346158658436518579420528 64,EndingHashKey:
 106338239662793269832304564822427566079},SequenceNumberRange: {StartingSequenceNumber:
 49627894338436249598912566043241477802747328865383743554,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 2 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000003,HashKeyRange: {StartingHashKey:
 638029437976759618993827388934565396 48,EndingHashKey:
 850705917302346158658436518579420528 63},SequenceNumberRange: {StartingSequenceNumber:
 49627894338413948853714035420099942084474680503877763122,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 3 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000015,HashKeyRange: {StartingHashKey:
 319014718988379809496913694467282698240,EndingHashKey:
 340282366920938463463374607431768211455},SequenceNumberRange: {StartingSequenceNumber:
 49627894338681557796096402897798370703746460841949528306,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
```

```
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 2 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000014,HashKeyRange: {StartingHashKey:
 29774707105582115553045278150279718502 4,EndingHashKey:
 31901471898837980949691369446728269 8239},SequenceNumberRange: {StartingSequenceNumber:
 49627894338659257050897872274656834985473812480443547874,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 3 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000004,HashKeyRange: {StartingHashKey:
 85070591730234615865843651857942052864,EndingHashKey:
 10633823966279326983230456482242756607 9},SequenceNumberRange: {StartingSequenceNumber:
 49627894338436249598912566043241477802747328865383743554,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,532 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 2 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000003,HashKeyRange: {StartingHashKey:
 63802943797675961899382738893456539648,EndingHashKey:
 85070591730234615865843651857942052863},SequenceNumberRange: {StartingSequenceNumber:
 49627894338413948853714035420099942084474680503877763122,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000001,HashKeyRange: {StartingHashKey:
 21267647932558653966460912964485513216,EndingHashKey:
 42535295865117307932921825928971026431},SequenceNumberRange: {StartingSequenceNumber:
 49627894338369347363316974173816870647929383780865802258,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000009,HashKeyRange: {StartingHashKey:
 19140883139302788569814821668036961 8944,EndingHashKey:
 21267647932558653966460912964485513215 9},SequenceNumberRange: {StartingSequenceNumber:
 49627894338547753324905219158949156394110570672913645714,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey:
 21267647932558653966460912964485513215},SequenceNumberRange: {StartingSequenceNumber:
 49627894338347046618118443550675334929656735419359821826,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
```

```
11:59:21,533 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000012,HashKeyRange: {StartingHashKey:
 25521177519070384759753095557382615859 2,EndingHashKey:
 27647942312326250156399186853831167 1807},SequenceNumberRange: {StartingSequenceNumber:
 49627894338614655560500811028373763548928515757431587010,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000008,HashKeyRange: {StartingHashKey:
 17014118346046923173168730371588410 5728,EndingHashKey:
 19140883139302788569814821668036961 8943},SequenceNumberRange: {StartingSequenceNumber:
 49627894338525452579706688535807620675837922311407665282,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000001,HashKeyRange: {StartingHashKey:
 21267647932558653966460912964485513216,EndingHashKey:
 42535295865117307932921825928971026431},SequenceNumberRange: {StartingSequenceNumber:
 49627894338369347363316974173816870647929383780865802258,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,533 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000011,HashKeyRange: {StartingHashKey:
 23394412725814519363107004260934064537 6,EndingHashKey:
 25521177519070384759753095557382615859 1},SequenceNumberRange: {StartingSequenceNumber:
 49627894338592354815302280405232227830655867395925606578,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey:
 21267647932558653966460912964485513215},SequenceNumberRange: {StartingSequenceNumber:
 49627894338347046618118443550675334929656735419359821826,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,568 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 1 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000002,HashKeyRange: {StartingHashKey:
 42535295865117307932921825928971026432,EndingHashKey:
 63802943797675961899382738893456539647},SequenceNumberRange: {StartingSequenceNumber:
 49627894338391648108515504796958406366202032142371782690,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
```

```
11:59:21,568 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
 [] - Subtask 1 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000013,HashKeyRange: {StartingHashKey:
 276479423123262501563991868538311671808,EndingHashKey:
 297747071055821155530452781502797185023},SequenceNumberRange: {StartingSequenceNumber:
 496278943386369563056993416515152992672011641l8937567442,}}'}, starting state set as
 sequence number EARLIEST_SEQUENCE_NUM
11:59:21,568 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 1 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000002,HashKeyRange: {StartingHashKey:
 42535295865117307932921825928971026432,EndingHashKey:
 63802943797675961899382738893456539647},SequenceNumberRange: {StartingSequenceNumber:
 496278943383916481085155047969584063662020321423717826 90,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:23,209 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000009,HashKeyRange: {StartingHashKey:
 191408831393027885698148216680369618944,EndingHashKey:
 212676479325586539664609129644855132159},SequenceNumberRange: {StartingSequenceNumber:
 496278943385477533249052191589491563941105706729136457l4,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,244 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 6 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000010,HashKeyRange: {StartingHashKey:
 212676479325586539664609129644855132160,EndingHashKey:
 233944127258145193631070042609340645375},SequenceNumberRange: {StartingSequenceNumber:
 496278943385700540701037497820906921123832l9034419626146,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
event: 6; timestamp: 1648030822428, 2022-03-23T10:20:22.428Z
11:59:23,377 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 3 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000015,HashKeyRange: {StartingHashKey:
 319014718988379809496913694467282698240,EndingHashKey:
 340282366920938463463374607431768211455},SequenceNumberRange: {StartingSequenceNumber:
 496278943386815577960964028977983707037464608419495283 06,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,405 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 2 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000014,HashKeyRange: {StartingHashKey:
```

```
 297747071055821155530452781502797185024,EndingHashKey:
 319014718988379809496913694467282698239},SequenceNumberRange: {StartingSequenceNumber:
 496278943386592570508978722746568349854738124804435478 74,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,581 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000008,HashKeyRange: {StartingHashKey:
 170141183460469231731687303715884105728,EndingHashKey:
 191408831393027885698148216680369618943},SequenceNumberRange: {StartingSequenceNumber:
 496278943385254525797066885358076206758379223114076652 82,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,586 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 1 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000013,HashKeyRange: {StartingHashKey:
 276479423123262501563991868538311671808,EndingHashKey:
 297747071055821155530452781502797185023},SequenceNumberRange: {StartingSequenceNumber:
 496278943386369563056993416515152992672011641189375674 42,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:24,790 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000012,HashKeyRange: {StartingHashKey:
 255211775190703847597530955573826158592,EndingHashKey:
 276479423123262501563991868538311671807},SequenceNumberRange: {StartingSequenceNumber:
 496278943386146555605008110283737635489285157574315870 10,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 2
event: 4; timestamp: 1648030809282, 2022-03-23T10:20:09.282Z
event: 3; timestamp: 1648030797697, 2022-03-23T10:19:57.697Z
event: 5; timestamp: 1648030810871, 2022-03-23T10:20:10.871Z
11:59:24,907 INFO
 org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
 Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
 shard='{ShardId: shardId-000000000011,HashKeyRange: {StartingHashKey:
 233944127258145193631070042609340645376,EndingHashKey:
 255211775190703847597530955573826158591},SequenceNumberRange: {StartingSequenceNumber:
 496278943385923548153022804052322278306558673959256065 78,}}'} from sequence number
 EARLIEST_SEQUENCE_NUM with ShardConsumer 2
event: 7; timestamp: 1648030834105, 2022-03-23T10:20:34.105Z
event: 1; timestamp: 1648030794441, 2022-03-23T10:19:54.441Z
event: 2; timestamp: 1648030796122, 2022-03-23T10:19:56.122Z
event: 8; timestamp: 1648030887171, 2022-03-23T10:21:27.171Z
XXXXXXXXXXXXXX Window with 3 events; Watermark: 1648030809281, 2022-03-23T10:20:09.281Z
```

```
3
1
2
XXXXXXXXXXXXXX Window with 2 events; Watermark: 1648030834104, 2022-03-23T10:20:34.104Z
4
5
XXXXXXXXXXXXXX Window with 1 events; Watermark: 1648030834104, 2022-03-23T10:20:34.104Z
6
XXXXXXXXXXXXXX Window with 1 events; Watermark: 1648030887170, 2022-03-23T10:21:27.170Z
7
```

La sortie n'affiche que 4 fenêtres (il manque la dernière fenêtre contenant l'événement 8). Cela est dû à l'heure de l'événement et à la stratégie de filigrane. La dernière fenêtre ne peut pas se fermer, car avec la stratégie de filigrane prédéfinie, le temps n'avance jamais au-delà de l'heure du dernier événement lu depuis le flux. Mais pour que la fenêtre se ferme, le temps doit avancer de plus de 10 secondes après le dernier événement. Dans ce cas, le dernier filigrane est 2022-03-23T10:21:27.170Z, mais pour que la fenêtre de session se ferme, un filigrane à 10 s et 1 ms plus tard est requis.

Si l'option `withIdleness` est supprimée de la stratégie de filigrane, aucune fenêtre de session ne se fermera, car le « filigrane global » de l'opérateur de fenêtre ne peut pas avancer.

Notez que lorsque l'application Flink démarre (ou en cas d’asymétrie des données), certaines partitions peuvent être consommées plus rapidement que d'autres. Cela peut entraîner l'émission de certains filigranes trop tôt à partir d'une sous-tâche (la sous-tâche peut émettre le filigrane en fonction du contenu d'une partition sans avoir consommé les autres partitions auxquelles elle est abonnée). Les moyens d'atténuer les risques sont d'adopter différentes stratégies de filigrane qui ajoutent un tampon de sécurité (`forBoundedOutOfOrderness(Duration.ofSeconds(30))` ou qui autorisent explicitement les arrivées tardives (`allowedLateness(Time.minutes(5))`.

Définir un UUID pour tous les opérateurs

Lorsque le service géré pour Apache Flink lance une tâche Flink pour une application avec un instantané, la tâche Flink peut ne pas démarrer en raison de certains problèmes. L'un d'eux est la non-concordance des identifiants d'opérateur. Flink attend des identifiants d'opérateur explicites et cohérents pour les opérateurs de graphiques de tâches Flink. S'il n'est pas défini explicitement, Flink génère automatiquement un identifiant pour les opérateurs. Cela est dû au fait que Flink utilise ces identifiants d'opérateur pour identifier de manière unique les opérateurs dans un graphe de tâches et les utilise pour stocker l'état de chaque opérateur dans un point de sauvegarde.

Le problème de non-concordance des identifiants d'opérateur se produit lorsque Flink ne trouve pas de correspondance 1:1 entre les identifiants d'opérateur d'un graphique de tâches et les identifiants d'opérateur définis dans un point de sauvegarde. Cela se produit lorsque des identifiants d'opérateur cohérents et explicites ne sont pas définis et que Flink génère automatiquement des identifiants d'opérateur qui peuvent ne pas être cohérents à chaque création de graphique de tâches. La probabilité que les applications rencontrent ce problème est élevée lors des opérations de maintenance. Pour éviter cela, nous recommandons aux clients de définir l'UUID pour tous les opérateurs en code Flink. Pour plus d'informations, consultez la rubrique Définir un UUID pour tous les opérateurs dans la section Préparation à la production.

Ajouter ServiceResourceTransformer au plugin Maven Shade

Flink utilise les interfaces Service Provider Interfaces (SPI) de Java pour charger des composants tels que des connecteurs et des formats. Plusieurs dépendances Flink utilisant les SPI peuvent provoquer des conflits dans le fichier JAR uber et des comportements inattendus des applications. Il est recommandé d'ajouter le ServiceResourceTransformerplugin maven shade, défini dans le fichier pom.xml

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-shade-plugin</artifactId>
                <executions>
                    <execution>
                        <id>shade</id>
                        <phase>package</phase>
                        <goals>
                            <goal>shade</goal>
                        </goals>
                        <configuration>
                            <transformers combine.children="append">
                                <!-- The service transformer is needed to merge META-
INF/services files -->
                                <transformer
 implementation="org.apache.maven.plugins.shade.resource.ServicesResourceTransformer"/>
                                <!-- ... -->
                            </transformers>
                        </configuration>
                    </execution>
                </executions>
```

```
        </plugin>
```

Stateful Functions pour Apache Flink

Stateful Functions est une API qui simplifie la création d'applications avec état distribuées. Elle repose sur des fonctions avec état persistant qui peuvent interagir de manière dynamique avec des garanties de cohérence élevée.

Une application Stateful Functions est essentiellement une application Apache Flink et peut donc être déployée dans le service géré pour Apache Flink. Cependant, il existe quelques différences entre l'empaquetage de Stateful Functions pour un cluster Kubernetes et pour le service géré pour Apache Flink. L'aspect le plus important d'une application Stateful Functions est que la configuration du module contient toutes les informations d'exécution nécessaires pour configurer l'exécution de Stateful Functions. Cette configuration est généralement empaquetée dans un conteneur spécifique à Stateful Functions et déployée sur Kubernetes. Mais cela n'est pas possible avec le service géré pour Apache Flink.

Voici une adaptation de l'exemple StateFun Python pour le service géré pour Apache Flink :

Modèle d'application Apache Flink

Au lieu d'utiliser un conteneur client pour l'exécution de Stateful Functions, les clients peuvent compiler un fichier JAR d'application Flink qui invoque simplement l'exécution de Stateful Functions et contient les dépendances requises. Pour Flink 1.13, les dépendances requises ressemblent à ceci :

```
<dependency>
 <groupId>org.apache.flink</groupId>
 <artifactId>statefun-flink-distribution</artifactId>
 <version>3.1.0</version>
 <exclusions>
  <exclusion>
   <groupId>org.slf4j</groupId>
   <artifactId>slf4j-log4j12</artifactId>
  </exclusion>
  <exclusion>
   <groupId>log4j</groupId>
   <artifactId>log4j</artifactId>
  </exclusion>
 </exclusions>
</dependency>
```

Et la méthode principale de l'application Flink pour invoquer l’exécution de Stateful Function ressemble à ceci :

```
public static void main(String[] args) throws Exception {
 final StreamExecutionEnvironment env =
 StreamExecutionEnvironment.getExecutionEnvironment();

 StatefulFunctionsConfig stateFunConfig = StatefulFunctionsConfig.fromEnvironment(env);

 stateFunConfig.setProvider((StatefulFunctionsUniverseProvider) (classLoader,
 statefulFunctionsConfig) -> {
  Modules modules = Modules.loadFromClassPath();
  return modules.createStatefulFunctionsUniverse(stateFunConfig);
 });

 StatefulFunctionsJob.main(env, stateFunConfig);
}
```

Notez que ces composants sont génériques et indépendants de la logique implémentée dans Stateful Function.

Emplacement de la configuration du module

La configuration du module Stateful Functions doit être incluse dans le chemin de classe pour être détectable par l’exécution de Stateful Functions. Il est préférable de l'inclure dans le dossier des ressources de l'application Flink et de l'empaqueter dans le fichier JAR.

À l'instar d'une application Apache Flink courante, vous pouvez ensuite utiliser Maven pour créer un fichier JAR uber et le déployer dans le service géré pour Apache Flink.

Informations sur les versions antérieures du service géré pour Apache Flink

Cette rubrique contient des informations sur l'utilisation du service géré pour Apache Flink avec les versions antérieures d'Apache Flink. Les versions d’Apache Flink prises en charge par le service géré pour Apache Flink sont 1.15.2 (recommandée), 1.13.2, 1.11.1, 1.8.2 et 1.6.2.

Nous vous recommandons d'utiliser la dernière version prise en charge d'Apache Flink avec votre service géré pour Apache Flink. La version 1.15.2 d'Apache Flink possède les fonctionnalités suivantes :

- Prise en charge de l'API de table Apache Flink et SQL
- Prise en charge des applications Python.
- Prise en charge de Java version 11 et de toutes les versions de Scala
- Un modèle de mémoire amélioré
- Optimisations de RocksDB pour une stabilité accrue des applications
- Prise en charge du gestionnaire de tâches et des traces de pile dans le tableau de bord Apache Flink.

Cette rubrique contient les sections suivantes :

- Utilisation du connecteur Kinesis Streams d’Apache Flink avec les versions précédentes d'Apache Flink
- Création d'applications avec Apache Flink 1.8.2
- Création d'applications avec Apache Flink 1.6.2
- Mise à niveau des applications
- Connecteurs disponibles dans Apache Flink 1.6.2 et 1.8.2
- Mise en route : Flink 1.13.2
- Mise en route : Flink 1.11.1
- Mise en route : Flink 1.8.2
- Mise en route : Flink 1.6.2

Utilisation du connecteur Kinesis Streams d'Apache Flink avec les versions précédentes d'Apache Flink

Le connecteur Kinesis Streams d'Apache Flink n'était pas inclus dans Apache Flink avant la version 1.11. Pour que votre application puisse utiliser le connecteur Kinesis d'Apache Flink avec les versions précédentes d'Apache Flink, vous devez télécharger, compiler et installer la version d'Apache Flink utilisée par votre application. Ce connecteur est utilisé pour consommer les données d'un flux Kinesis utilisé comme source d'application ou pour écrire des données dans un flux Kinesis utilisé pour la sortie de l'application.

> **Note**
>
> Assurez-vous que vous créez le connecteur avec la version KPL 0.14.0 ou ultérieure.

Pour télécharger et installer le code source d'Apache Flink version 1.8.2, procédez comme suit :

1. Assurez-vous qu'Apache Maven est installé et que votre variable d'environnement `JAVA_HOME` pointe vers un JDK plutôt qu'un JRE. Vous pouvez tester votre installation Apache Maven à l'aide de la commande suivante :

   ```
   mvn -version
   ```

2. Téléchargez le code source d'Apache Flink version 1.8.2 :

   ```
   wget https://archive.apache.org/dist/flink/flink-1.8.2/flink-1.8.2-src.tgz
   ```

3. Décompressez le code source d'Apache Flink :

   ```
   tar -xvf flink-1.8.2-src.tgz
   ```

4. Accédez au répertoire du code source d'Apache Flink :

   ```
   cd flink-1.8.2
   ```

5. Compilez et installez Apache Flink :

   ```
   mvn clean install -Pinclude-kinesis -DskipTests
   ```

> **Note**
>
> Si vous compilez Flink sous Microsoft Windows, vous devez ajouter le paramètre `-Drat.skip=true`.

Création d'applications avec Apache Flink 1.8.2

Cette section contient des informations sur les composants que vous utilisez pour créer des applications de service géré Apache Flink qui fonctionnent avec Apache Flink 1.8.2.

Utilisez les versons de composants suivants pour les applications de service géré pour Apache Flink :

Composant	Version
Java	1.8 (recommandée)
Apache Flink	1.8.2
Service géré pour Apache Flink pour l'éxécution Flink (aws-kinesisanalytics-runtime)	1.0.1
Connecteurs Flink pour le service géré pour Apache Flink (aws-kinesisanalytics-flink)	1.0.1
Apache Maven	3.1

Pour compiler une application à l'aide d'Apache Flink 1.8.2, exécutez Maven avec le paramètre suivant :

```
mvn package -Dflink.version=1.8.2
```

Pour un exemple de fichier `pom.xml` pour une application de service géré pour Apache Flink utilisant Apache Flink version 1.8.2, consultez Managed Service for Apache Flink 1.8.2 Getting Started Application.

Pour plus d'informations sur la création et l'utilisation du code d'application pour une application de service géré pour Apache Flink, consultez Création d'applications.

Création d'applications avec Apache Flink 1.6.2

Cette section contient des informations sur les composants que vous utilisez pour créer des applications de service géré Apache Flink qui fonctionnent avec Apache Flink 1.6.2.

Utilisez les versons de composants suivants pour les applications de service géré pour Apache Flink :

Composant	Version
Java	1.8 (recommandée)
Kit SDK Java d'AWS	1.11.379
Apache Flink	1.6.2
Service géré pour Apache Flink pour l'éxécution Flink (aws-kinesisanalytics-runtime)	1.0.1
Connecteurs Flink pour le service géré pour Apache Flink (aws-kinesisanalytics-flink)	1.0.1
Apache Maven	3.1
Apache Beam	Non pris en charge avec Apache Flink 1.6.2.

> **Note**
>
> Lorsque vous utilisez l'exécution de service géré pour Apache Flink version 1.0.1, vous spécifiez la version d'Apache Flink dans votre fichier `pom.xml` plutôt que d'utiliser le paramètre `-Dflink.version` lors de la compilation du code de votre application.

Pour un exemple de fichier `pom.xml` pour une application de service géré pour Apache Flink utilisant Apache Flink version 1.6.2, consultez Managed Service for Apache Flink 1.6.2 Getting Started Application.

Pour plus d'informations sur la création et l'utilisation du code d'application pour une application de service géré pour Apache Flink, consultez Création d'applications.

Mise à niveau des applications

Pour mettre à niveau la version d'une application de service géré pour Apache Flink, vous devez mettre à jour le code de votre application, supprimer l'application précédente et créer une nouvelle application avec le code mis à jour. Pour ce faire, procédez comme suit :

- Modifiez les versions de l'exécution du service géré pour Apache Flink et les connecteurs Flink du service géré pour Apache Flink (aws-kinesisanalytics-flink) dans le fichier `pom.xml` de votre application pour 1.1.0.
- Supprimez la propriété `flink.version` du fichier `pom.xml` de votre application. Vous fournirez ce paramètre lors de la compilation du code de l'application à l'étape suivante.
- Recompilez le code de votre application à l'aide de la commande suivante :

```
mvn package -Dflink.version=1.15.3
```

- Supprimez votre application existante. Créez à nouveau votre application et choisissez Apache Flink version 1.15.2 (version recommandée) pour l'exécution de l'application.

> **Note**
>
> Vous ne pouvez pas utiliser les instantanés des versions précédentes de votre application.

Connecteurs disponibles dans Apache Flink 1.6.2 et 1.8.2

L'environnement Apache Flink contient des connecteurs permettant d'accéder aux données provenant de diverses sources.

- Pour plus d'informations sur les connecteurs disponibles dans l'environnement Apache Flink 1.6.2, consultez Connectors (1.6.2) dans la documentation Apache Flink (1.6.2).
- Pour plus d'informations sur les connecteurs disponibles dans l'environnement Apache Flink 1.8.2, consultez Connectors (1.8.2) dans la documentation Apache Flink (1.8.2).

Mise en route : Flink 1.13.2

Cette section présente les concepts fondamentaux du service géré pour Apache Flink et de l' DataStream API. Elle décrit les options disponibles pour créer et tester vos applications. Elle fournit également des instructions pour installer les outils nécessaires pour suivre les didacticiels de ce guide et pour créer votre première application.

Rubriques

- Composants d’une application de service géré pour Apache Flink pour Flink
- Prérequis pour effectuer les exercices
- Étape 1 : configuration d'un compte AWS et création d'un administrateur
- Étape suivante
- Étape 2 : configuration de AWS Command Line Interface (AWS CLI)
- Étape 3 : création et exécution d'une application de service géré pour Apache Flink.
- Etape 4 : nettoyage des ressources AWS
- Étape 5 : étapes suivantes

Composants d’une application de service géré pour Apache Flink pour Flink

Pour traiter les données, votre application de service géré pour Apache Flink utilise une application Java/Apache Maven ou Scala qui traite les entrées et produit des sorties à l'aide de l’exécution Apache Flink.

L’application de service géré pour Apache Flink comprend les composants suivants :

- Propriétés d'exécution : vous pouvez utiliser les propriétés d'exécution pour configurer votre application sans recompiler le code de votre application.
- Source : l'application consomme des données en utilisant une source. Un connecteur source lit les données d'un flux de données Kinesis, d'un compartiment Amazon S3, etc. Pour plus d’informations, consultez Sources.
- Opérateurs : l'application traite les données à l'aide d'un ou de plusieurs opérateurs. Un opérateur peut transformer, enrichir ou agréger des données. Pour plus d’informations, consultez Opérateurs d'API DataStream.

- Récepteur : l'application produit des données vers des sources externes à l'aide de récepteurs. Un connecteur récepteur écrit des données dans un flux de données Kinesis, un flux Kinesis Data Firehose, un compartiment Amazon S3, etc. Pour plus d'informations, consultez Récepteurs.

Après avoir créé, compilé et empaqueté votre code d'application, vous chargez le package de code dans un compartiment Amazon Simple Storage Service (Amazon S3). Vous créez ensuite une application de service géré pour Apache Flink. Vous transmettez l'emplacement du package de code, un flux de données Kinesis comme source de données de streaming et généralement un emplacement de streaming ou de fichier qui reçoit les données traitées par l'application.

Prérequis pour effectuer les exercices

Pour exécuter la procédure indiquée dans ce guide, vous devez disposer des éléments suivants :

- Kit de développement Java (JDK) version 11. Définissez la variable d'environnement `JAVA_HOME` pour qu'elle pointe vers l'emplacement d'installation de votre JDK.
- Nous vous recommandons d'utiliser un environnement de développement (par exemple Eclipse Java Neon ou IntelliJ Idea) pour développer et compiler votre application.
- Client Git. Installez le client Git si vous ne l'avez pas déjà fait.
- Apache Maven Compiler Plugin. Maven doit être installé dans votre chemin de travail. Pour tester votre installation Apache Maven, saisissez les informations suivantes :

```
$ mvn -version
```

Pour démarrer, accédez à Étape 1 : configuration d'un compte AWS et création d'un administrateur.

Étape 1 : configuration d'un compte AWS et création d'un administrateur

S'inscrire à un Compte AWS

Si vous n'avez pas de compte Compte AWS, procédez comme suit pour en créer un.

Pour s'inscrire à un Compte AWS

1. Ouvrez https://portal.aws.amazon.com/billing/signup.
2. Suivez les instructions en ligne.

Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

Lorsque vous souscrivez à un Compte AWS, un Utilisateur racine d'un compte AWS est créé. Par défaut, seul l'utilisateur root a accès à l'ensemble des Services AWS et des ressources de ce compte. La meilleure pratique de sécurité consiste à attribuer un accès administratif à un utilisateur administratif, et à uniquement utiliser l'utilisateur root pour effectuer tâches nécessitant un accès utilisateur root.

AWS vous envoie un e-mail de confirmation lorsque le processus d'inscription est terminé. Vous pouvez afficher l'activité en cours de votre compte et gérer votre compte à tout moment en accédant à https://aws.amazon.com/ et en cliquant sur Mon compte.

Création d'un utilisateur administratif

Après vous être inscrit à un Compte AWS, sécurisez votre Utilisateur racine d'un compte AWS, activez AWS IAM Identity Center, puis créez un utilisateur administratif afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisation de votre Utilisateur racine d'un compte AWS

1. Connectez-vous à la AWS Management Console en tant que propriétaire du compte en sélectionnant Root user (utilisateur root) et en saisissant l'adresse e-mail de Compte AWS. Sur la page suivante, saisissez votre mot de passe.

   Pour obtenir de l'aide pour vous connecter en utilisant l'utilisateur root, consultez Connexion en tant qu'utilisateur root dans le Guide de l'utilisateur Connexion à AWS.
2. Activez l'authentification multifactorielle (MFA) pour votre utilisateur root.

   Pour obtenir des instructions, consultez Activation d'un dispositif MFA virtuel pour l'utilisateur root de votre Compte AWS (console) dans le Guide de l'utilisateur IAM.

Création d'un utilisateur administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Configuration d'AWS IAM Identity Center dans le guide de l'utilisateur AWS IAM Identity Center.

2. Dans IAM Identity Center, octroyez un accès administratif à un utilisateur administratif.

   Pour profiter d'un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, consultez Configuration de l'accès utilisateur avec le répertoire Répertoire IAM Identity Center par défaut dans le Guide de l'utilisateur AWS IAM Identity Center.

Connexion en tant qu'utilisateur administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l'URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l'utilisateur IAM Identity Center.

  Pour obtenir de l'aide pour vous connecter à l'aide d'un utilisateur IAM Identity Center, consultez Connexion au portail d'accès AWS dans le Guide de l'utilisateur Connexion à AWS.

Octroi d'un accès par programmation

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS en dehors de la AWS Management Console. La manière d'octroyer un accès par programmation dépend du type d'utilisateur qui accède à AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l'une des options suivantes.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
Identité de la main-d'œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour l'AWS CLI, veuillez consulter la rubrique Configuration de l'AWS CLI pour l'utilisation d'AWS IAM Identity Center dans le Guide de l'utilisateur AWS Command Line Interface. • Pour les kits AWS SDK, les outils et les API AWS,

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
		consultez Authentification IAM Identity Center dans le Guide de référence des kits SDK et des outils AWS.
IAM	Utilisez des informations d'identification temporaires pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec des ressources AWS dans le Guide de l'utilisateur IAM.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour l'AWS CLI, veuillez consulter la rubrique Authentification à l'aide des informations d'identification d'utilisateur IAM dans le Guide de l'utilisateur AWS Command Line Interface. • Pour les kits SDK et les outils AWS, veuillez consulter la rubrique Authentification à l'aide d'informations d'identification à long terme dans le Guide de référence des kits SDK et des outils AWS. • Pour les API AWS, veuillez consulter la rubrique Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Étape suivante

Étape 2 : configuration de AWS Command Line Interface (AWS CLI)

Étape suivante

Étape 2 : configuration de AWS Command Line Interface (AWS CLI)

Étape 2 : configuration de AWS Command Line Interface (AWS CLI)

Dans cette étape, vous allez télécharger et configurer la AWS CLI pour l'utiliser avec le service géré pour Apache Flink.

> **Note**
>
> Les exercices de mise en route de ce guide supposent que vous utilisez les informations d'identification d'administrateur (`adminuser`) de votre compte pour effectuer les opérations.

> **Note**
>
> Si l'interface AWS CLI est déjà installée, vous pouvez avoir besoin de procéder à une mise à niveau pour obtenir les dernières fonctionnalités. Pour plus d'informations, consultez Installation d'AWS Command Line Interface dans le Guide de l'utilisateur AWS Command Line Interface. Pour vérifier la version de l'interface AWS CLI, exécutez la commande suivante :
>
> ```
> aws --version
> ```
>
> Les exercices de ce didacticiel nécessitent la version AWS CLI suivante ou une version ultérieure :
>
> ```
> aws-cli/1.16.63
> ```

Pour configurer l’interface AWS CLI

1. Téléchargez et configurez l’interface AWS CLI. Pour obtenir des instructions, consultez les rubriques suivantes dans le Guide de l'utilisateur de l'interface AWS Command Line Interface :
   - Installation de AWS Command Line Interface
   - Configuration de l'interface AWS CLI (français non garanti)
2. Ajoutez un profil désigné pour l'utilisateur administrateur dans le fichier `config` de l’interface AWS CLI. Vous utiliserez ce profil lorsque vous exécuterez les commandes AWS CLI. Pour plus

d'informations sur les profils nommés, consultez la rubrique Profils nommés dans le Guide de l'utilisateur AWS Command Line Interface.

```
[profile adminuser]
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
```

Pour connaître la liste des régions AWS disponibles, consultez Régions et points de terminaison du manuel Référence générale d'Amazon Web Services.

> **Note**
>
> Les exemples de code et de commandes présentés dans ce didacticiel utilisent la région USA Ouest (Oregon). Pour utiliser une autre région, remplacez la région dans le code et les commandes de ce didacticiel par la région que vous souhaitez utiliser.

3. Vérifiez la configuration en saisissant la commande d'aide suivante à l'invite de commande :

```
aws help
```

Après avoir configuré un AWS compteAWS CLI, vous pouvez passer à l'exercice suivant, dans lequel vous configurez un exemple d'application et testez la end-to-end configuration.

Étape suivante

Étape 3 : création et exécution d'une application de service géré pour Apache Flink.

Étape 3 : création et exécution d'une application de service géré pour Apache Flink.

Dans cet exercice, vous allez créer une application de service géré pour Apache Flink avec des flux de données comme source et comme récepteur.

Cette section contient les étapes suivantes :

- Création de deux flux de données Amazon Kinesis Data Streams
- Écriture d'exemples d'enregistrements dans le flux d'entrée

- Téléchargement et examen du code Java Apache Flink
- Compilation du code d'application
- Chargement du code Java Apache Flink
- Création et exécution de l'application de service géré pour Apache Flink
- Étape suivante

Création de deux flux de données Amazon Kinesis Data Streams

Avant de créer une application de service géré pour Apache Flink dans le cadre de cet exercice, commencez par créer deux flux de données Kinesis (`ExampleInputStream` et `ExampleOutputStream`). Votre application utilise ces flux pour les flux source et de destination de l'application.

Vous pouvez créer ces flux à l'aide de la console Amazon Kinesis ou de la commande AWS CLI suivante. Pour obtenir des instructions sur la console, consultez Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams.

Pour créer les flux de données (AWS CLI)

1. Pour créer le premier flux (`ExampleInputStream`) utilisez la commande AWS CLI `create-stream` Amazon Kinesis suivante.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Pour créer le second flux utilisé par l'application pour écrire la sortie, exécutez la même commande en remplaçant le nom du flux par `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Écriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette section, vous utilisez un script Python pour écrire les exemples d'enregistrements dans le flux pour que l'application les traite.

> **Note**
>
> Cette section nécessite le kit AWS SDK for Python (Boto).

1. Créez un fichier nommé `stock.py` avec le contenu suivant :

```
import datetime
import json
import random
import boto3
STREAM_NAME = "ExampleInputStream"
def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}
def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
            Data=json.dumps(data),
            PartitionKey="partitionkey")
if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

2. Plus loin dans ce didacticiel, vous exécutez le script `stock.py` pour envoyer des données à l'application.

```
$ python stock.py
```

Téléchargement et examen du code Java Apache Flink

Le code de l'application Java pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/GettingStarted`.

Notez les informations suivantes à propos du code d'application :

- Un fichier de modèle d'objet du projet (pom.xml) contient des informations sur la configuration et les dépendances de l'application, y compris les bibliothèques du service géré pour Apache Flink.
- Le fichier `BasicStreamingJob.java` contient la méthode `main` qui définit la fonctionnalité de l'application.
- L'application utilise une source Kinesis pour lire à partir du flux source. L'extrait de code suivant crée la source Kinesis :

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- Votre application crée les connecteurs source et récepteur pour accéder aux ressources externes à l'aide d'un objet `StreamExecutionEnvironment`.
- L'application crée les connecteurs source et récepteur à l'aide de propriétés statiques. Pour utiliser les propriétés de l'application dynamique, utilisez les méthodes `createSourceFromApplicationProperties` et `createSinkFromApplicationProperties` pour créer les connecteurs. Ces méthodes lisent les propriétés de l'application pour configurer les connecteurs.

  Pour de plus amples informations sur les propriétés d'exécution, veuillez consulter Propriétés d'exécution.

Compilation du code d'application

Dans cette section, vous allez utiliser le compilateur Apache Maven pour créer le code Java pour l'application. Pour de plus amples informations sur l'installation d'Apache Maven et sur le kit de développement Java (JDK), veuillez consulter Prérequis pour effectuer les exercices.

Pour compiler le code d'application

1. Pour utiliser votre code d'application, vous le compilez et l'intégrez dans un fichier JAR. Vous pouvez compiler et intégrer votre code de deux manières :

   - À l'aide de l'outil de ligne de commande Maven. Créez votre fichier JAR en exécutant la commande suivante dans le répertoire qui contient le fichier `pom.xml` :

     ```
     mvn package -Dflink.version=1.13.2
     ```

   - À l'aide de votre environnement de développement. Consultez la documentation de votre environnement de développement pour plus de détails.

     > **Note**
     >
     > Le code source fourni repose sur les bibliothèques de Java 11.

   Vous pouvez charger votre package en tant que fichier JAR, ou compresser le package et le charger en tant que fichier ZIP. Si vous créez votre application à l'aide de l'interface AWS CLI, vous spécifiez le type de contenu de votre code (JAR ou ZIP).

2. En cas d'erreur lors de la compilation, vérifiez que votre variable d'environnement `JAVA_HOME` est correctement définie.

Si la compilation de l'application aboutit, le fichier suivant est créé :

`target/aws-kinesis-analytics-java-apps-1.0.jar`

Chargement du code Java Apache Flink

Dans cette section, vous allez créer un compartiment Amazon Simple Storage Service (Amazon S3) et charger votre code d'application.

Pour charger le code d'application

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez Créer un compartiment.
3. Saisissez **ka-app-code-*<username>*** dans le champ Nom du compartiment. Ajoutez un suffixe au nom du compartiment, par exemple votre nom d'utilisateur, pour qu'il soit unique. Choisissez Suivant.
4. À l'étape Configurer les options, conservez les paramètres, puis choisissez Suivant.
5. À l'étape Définir des autorisations, conservez les paramètres, puis choisissez Suivant.
6. Choisissez Créer un compartiment.
7. Dans la console Amazon S3, choisissez le <username>compartiment ka-app-code-, puis Upload.
8. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `aws-kinesis-analytics-java-apps-1.0.jar` que vous avez créé à l'étape précédente. Choisissez Suivant.
9. Vous n'avez pas besoin de modifier les paramètres de l'objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l'application de service géré pour Apache Flink

Vous pouvez créer et exécuter une application de service géré pour Apache Flink à l'aide de la console ou de l'interface AWS CLI.

> **Note**
>
> Lorsque vous créez l'application à l'aide de la console, vos ressources AWS Identity and Access Management (IAM) et Amazon CloudWatch Logs sont créées pour vous. Lorsque vous créez l'application à l'aide de l'interface AWS CLI, vous créez ces ressources séparément.

Rubriques

- Création et exécution de l'application (console)
- Création et exécution de l'application (AWS CLI)

Création et exécution de l'application (console)

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Description, saisissez **`My java test app`**.
   - Pour Exécution, choisissez Apache Flink.
   - Laissez le menu déroulant de la version sur Apache Flink version 1.13 .
4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rôle : `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder aux flux de données Kinesis.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **`kinesis-analytics-service-MyApplication-us-west-2`** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*012345678901*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
```

```
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Configuration de l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **ka-app-code-*<username>***.
   - Pour le chemin de l'objet Amazon S3, saisissez **aws-kinesis-analytics-java-apps-1.0.jar**.
3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Saisissez :

ID du groupe	Clé	Valeur
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

5. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
6. Pour la CloudWatch journalisation, cochez la case Activer.
7. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation Amazon, Managed Service for Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`

Exécution de l'application

Le graphique des tâches Flink peut être visualisé en exécutant l'application, en ouvrant le tableau de bord Apache Flink et en choisissant la tâche Flink souhaitée.

Arrêt de l'application

Sur la MyApplicationpage, choisissez Stop. Confirmez l'action.

Mise à jour de l'application

À l'aide de la console, vous pouvez mettre à jour les paramètres d'application tels que les paramètres de surveillance, les propriétés d'application et l'emplacement ou le nom du fichier JAR de l'application. Vous pouvez également recharger le fichier JAR de l'application à partir du compartiment Amazon S3 si vous avez besoin de mettre à jour le code de l'application.

Sur la MyApplicationpage, choisissez Configurer. Mettez à jour les paramètres de l'application, puis choisissez Mettre à jour.

Création et exécution de l'application (AWS CLI)

Dans cette section, vous allez utiliser l'interface AWS CLI pour créer et exécuter l'application de service géré pour Apache Flink. Le service géré pour Apache Flink utilise la commande AWS CLI `kinesisanalyticsv2` pour créer et interagir avec les applications de service géré pour Apache Flink.

Créer une stratégie d'autorisations

> **Note**
>
> Vous devez créer une stratégie d'autorisations et un rôle pour votre application. Si vous ne créez pas ces ressources IAM, votre application ne peut pas accéder à ses flux de données et de journaux.

Vous commencez par créer une stratégie d'autorisations avec deux instructions : une qui accorde des autorisations pour l'action `read` sur le flux source et une autre qui accorde des autorisations pour les actions `write` sur le flux récepteur. Vous attachez ensuite la politique à un rôle IAM (que vous allez créer dans la section suivante). Ainsi, lorsque le service géré pour Apache Flink assume le rôle, le service dispose des autorisations nécessaires pour lire à partir du flux source et écrire dans le flux récepteur.

Utilisez le code suivant pour créer la stratégie d'autorisations `AKReadSourceStreamWriteSinkStream`. Remplacez *`username`* par le nom d'utilisateur que vous avez utilisé pour créer le compartiment Amazon S3 pour stocker le code d'application. Remplacez l'ID de compte dans Amazon Resource Names (ARN) (*`012345678901`*) par votre ID de compte.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": ["arn:aws:s3:::ka-app-code-username",
                "arn:aws:s3:::ka-app-code-username/*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Pour step-by-step obtenir des instructions sur la création d'une politique d'autorisations, voir Tutoriel : créer et joindre votre première politique gérée par le client dans le guide de l'utilisateur IAM.

> **Note**
>
> Pour accéder à d'autres services Amazon, vous pouvez utiliser le AWS SDK for Java. Le service géré pour Apache Flink définit automatiquement les informations d'identification requises par le kit SDK en fonction du rôle IAM d'exécution du service associé à votre application. Aucune étape supplémentaire n'est nécessaire.

Créer un rôle IAM

Dans cette section, vous créez un rôle IAM que l'application de service géré pour Apache Flink peut assumer pour lire un flux source et écrire dans le flux récepteur.

Le service géré pour Apache Flink ne peut pas accéder à votre flux sans autorisation. Vous utilisez un rôle IAM pour accorder ces autorisations. Deux politiques sont attachées à chaque rôle IAM. La politique d'approbation accorde au service géré pour Apache Flink l'autorisation d'assumer le rôle, et la politique d'autorisation détermine ce que le service géré pour Apache Flink peut faire après avoir assumé le rôle.

Vous attachez la stratégie d'autorisations que vous avez créée dans la section précédente à ce rôle.

Pour créer un rôle IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, choisissez Rôles, puis Créer un rôle.
3. Sous Sélectionner le type d'identité approuvée, choisissez Service AWS. Sous Choisir le service qui utilisera ce rôle, choisissez EC2. Sous Sélectionner votre cas d'utilisation, choisissez Kinesis Analytics.

   Sélectionnez Next: Permissions (Étape suivante : autorisations).
4. Dans la page Attacher des stratégies d'autorisations, choisissez Suivant : vérification. Vous attachez des stratégies d'autorisations après avoir créé le rôle.
5. Sur la page Créer un rôle, saisissez **`MF-stream-rw-role`** pour le Nom du rôle. Sélectionnez Créer un rôle.

   Vous venez de créer un nouveau rôle IAM appelé `MF-stream-rw-role`. Ensuite, vous mettez à jour les stratégies d'approbation et d'autorisation pour le rôle.
6. Attachez la politique d'autorisation au rôle.

> **Note**
>
> Dans le cadre de cet exercice, le service géré pour Apache Flink assume ce rôle à la fois pour la lecture des données à partir d'un flux de données Kinesis (source) et pour l'écriture des résultats dans un autre flux de données Kinesis. Vous attachez donc la stratégie que vous avez créée à l'étape précédente, the section called “Créer une stratégie d'autorisations”.

a. Sur la page Récapitulatif, choisissez l'onglet Autorisations.

b. Choisissez Attacher des stratégies.

c. Dans la zone de recherche, saisissez **`AKReadSourceStreamWriteSinkStream`** (la stratégie que vous avez créée dans la section précédente).

d. Choisissez la ReadSourceStreamWriteSinkStream politique AK, puis choisissez Attach policy.

Vous avez maintenant créé le rôle d'exécution de service que votre application utilise pour accéder aux ressources. Notez l'ARN du nouveau rôle.

Pour step-by-step obtenir des instructions sur la création d'un rôle, consultez la section Création d'un rôle IAM (console) dans le guide de l'utilisateur IAM.

Création de l’application de service géré pour Apache Flink

1. Copiez le code JSON suivant dans un fichier nommé `create_request.json`. Remplacez l'exemple d'ARN du rôle par l'ARN du rôle que vous avez créé précédemment. Remplacez le suffixe de l'ARN du compartiment (*`username`*) par le suffixe que vous avez choisi dans la section précédente. Remplacez l'exemple d'ID de compte (*`012345678901`*) dans le rôle d'exécution de service par votre ID de compte.

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "aws-kinesis-analytics-java-apps-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
```

```
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
   }
}
```

2. Exécutez l'action [CreateApplication](#) avec la demande précédente pour créer l'application :

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

L'application est maintenant créée. Vous démarrez l'application dans l'étape suivante.

Démarrage de l'application

Dans cette section, vous utilisez l'action [StartApplication](#) pour démarrer l'application.

Pour démarrer l'application

1. Copiez le code JSON suivant dans un fichier nommé `start_request.json`.

```
{
    "ApplicationName": "test",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
```

```
}
```

2. Exécutez l'action StartApplication avec la demande précédente pour démarrer l'application :

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

L'application est maintenant en cours d'exécution. Vous pouvez consulter les métriques du service géré pour Apache Flink sur la CloudWatch console Amazon pour vérifier que l'application fonctionne.

Arrêt de l'application

Dans cette section, vous allez utiliser l'action StopApplication pour arrêter l'application.

Pour arrêter l'application

1. Copiez le code JSON suivant dans un fichier nommé `stop_request.json`.

```
{
   "ApplicationName": "test"
}
```

2. Exécutez l'action StopApplication avec la demande suivante pour arrêter l'application :

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

L'application est maintenant arrêtée.

Ajouter une option de CloudWatch journalisation

Vous pouvez utiliser le AWS CLI pour ajouter un flux de CloudWatch journal Amazon à votre application. Pour plus d'informations sur l'utilisation de CloudWatch Logs avec votre application, consultezthe section called “Configuration de la journalisation”.

Mettre à jour des propriétés d'environnement

Dans cette section, vous utilisez l'action UpdateApplication pour modifier les propriétés d'environnement de l'application sans recompiler le code de l'application. Dans cet exemple, vous modifiez la région des flux source et de destination.

Pour mettre à jour des propriétés d'environnement pour l'application

1. Copiez le code JSON suivant dans un fichier nommé `update_properties_request.json`.

```
{"ApplicationName": "test",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
   }
}
```

2. Exécutez l'action [UpdateApplication] avec la demande précédente pour mettre à jour les propriétés de l'environnement :

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Mise à jour du code de l’application

Lorsque vous devez mettre à jour le code de votre application avec une nouvelle version de votre package de code, vous utilisez l'action[UpdateApplication] de l’interface AWS CLI.

> **Note**
>
> Pour charger une nouvelle version du code de l'application portant le même nom de fichier, vous devez spécifier la nouvelle version de l'objet. Pour de plus amples informations sur l'utilisation des versions d'objet Amazon S3, veuillez consulter Activation et désactivation de la gestion des versions.

Pour utiliser l'interface AWS CLI, supprimez votre ancien package de code de votre compartiment Amazon S3, téléchargez la nouvelle version et appelez `UpdateApplication` en spécifiant le même compartiment Amazon S3 et le même nom d'objet, ainsi que la nouvelle version de l'objet. L'application redémarrera avec le nouveau package de code.

L'exemple de demande d'action `UpdateApplication` suivant recharge le code de l'application et redémarre l'application. Mettez à jour l'`CurrentApplicationVersionId` à la version actuelle de l'application. Vous pouvez vérifier la version actuelle de l'application à l'aide des actions `ListApplications` ou `DescribeApplication`. Mettez à jour le suffixe du nom du compartiment (<username>) avec le suffixe que vous avez choisi dans la section the section called "Création de deux flux de données Amazon Kinesis Data Streams".

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "aws-kinesis-analytics-java-apps-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Étape suivante

Etape 4 : nettoyage des ressources AWS

Etape 4 : nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Mise en route.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer vos flux de données Kinesis
- Supprimer votre objet et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources
- Étape suivante

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Managed Service for Apache Flink, choisissez MyApplication.
3. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer vos flux de données Kinesis

1. Ouvrez la console du service géré pour Apache Flink à l’adresse https://console.aws.amazon.com/flink
2. Dans le panneau Kinesis Data Streams, ExampleInputStreamsélectionnez.
3. Sur la ExampleInputStreampage, choisissez Supprimer Kinesis Stream, puis confirmez la suppression.
4. Sur la page Kinesis Streams, choisissez le ExampleOutputStream, choisissez Actions, choisissez Supprimer, puis confirmez la suppression.

Supprimer votre objet et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code -. <username>

3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.
8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Étape suivante

Étape 5 : étapes suivantes

Étape 5 : étapes suivantes

Maintenant que vous avez créé et exécuté une application de service géré de base pour Apache Flink, consultez les ressources suivantes pour des solutions de service géré plus avancées pour Apache Flink.

- Solution de streaming de données AWS pour Amazon Kinesis : la solution de streaming de données AWS pour Amazon Kinesis configure automatiquement les services AWS nécessaires pour capturer, stocker, traiter et diffuser facilement des données de streaming. La solution propose plusieurs options pour résoudre les problèmes d'utilisation de données en streaming. L'option Managed Service for Apache Flink fournit un exemple de end-to-end streaming ETL illustrant une

application réelle qui exécute des opérations analytiques sur des données de taxis simulées à New York. La solution met en place toutes les AWS ressources nécessaires telles que les rôles et les politiques IAM, un CloudWatch tableau de bord et des CloudWatch alarmes.

- Solution de données de streaming AWS pour Amazon MSK :la solution de données de streaming AWS pour Amazon MSK fournit des modèles AWS CloudFormation dans lesquels les données circulent entre les producteurs, le stockage en streaming, les consommateurs et les destinations.
- Clickstream Lab avec Apache Flink et Apache Kafka : un laboratoire de bout en bout pour les cas d'utilisation d'Amazon Managed Streaming for Apache Kafka pour le stockage de streaming et le service géré pour Apache Flink pour les applications Apache Flink pour le traitement des flux.
- Amazon Managed Service for Apache Flink Workshop : dans cet atelier, vous allez créer une architecture de end-to-end streaming pour ingérer, analyser et visualiser les données de streaming en temps quasi réel. Vous avez décidé d'améliorer les opérations d'une compagnie de taxi à New York. Vous analysez les données de télémétrie d'une flotte de taxis à New York en temps quasi réel afin d'optimiser le fonctionnement de la flotte.
- Service géré Amazon pour Apache Flink : exemples : cette section de ce guide du développeur fournit des exemples de création et d'utilisation d'applications dans le service géré pour Apache Flink. Ils incluent des exemples de code et des step-by-step instructions pour vous aider à créer un service géré pour les applications Apache Flink et à tester vos résultats.
- Learn Flink : Hands On Training : formation d'introduction officielle à Apache Flink qui vous permet de commencer à écrire des applications ETL, analytiques et axées sur les événements évolutives pour le streaming.

> **Note**
>
> Sachez que le service géré pour Apache Flink ne prend pas en charge la version Apache Flink (1.12) utilisée dans cette formation. Vous pouvez utiliser Flink 1.15.2 dans le service géré Flink pour Apache Flink.

Mise en route : Flink 1.11.1

Cette rubrique contient une version du didacticiel Mise en route (DataStream API) qui utilise Apache Flink 1.11.1.

Cette section présente les concepts fondamentaux du service géré pour Apache Flink et de l' DataStream API. Elle décrit les options disponibles pour créer et tester vos applications. Elle fournit

également des instructions pour installer les outils nécessaires pour suivre les didacticiels de ce guide et pour créer votre première application.

Rubriques

- Composants d’une application de service géré pour Apache Flink pour Flink
- Prérequis pour effectuer les exercices
- Étape 1 : configuration d'un compte AWS et création d'un administrateur
- Étape 2 : configuration de AWS Command Line Interface (AWS CLI)
- Étape 3 : création et exécution d'une application de service géré pour Apache Flink.
- Etape 4 : nettoyage des ressources AWS
- Étape 5 : étapes suivantes

Composants d’une application de service géré pour Apache Flink pour Flink

Pour traiter les données, votre application de service géré pour Apache Flink utilise une application Java/Apache Maven ou Scala qui traite les entrées et produit des sorties à l'aide de l’exécution Apache Flink.

Une application de service géré for Apache Flink comprend les composants suivants :

- Propriétés d'exécution : vous pouvez utiliser les propriétés d'exécution pour configurer votre application sans recompiler le code de votre application.
- Source : l'application consomme des données en utilisant une source. Un connecteur source lit les données d'un flux de données Kinesis, d'un compartiment Amazon S3, etc. Pour plus d’informations, consultez Sources.
- Opérateurs : l'application traite les données à l'aide d'un ou de plusieurs opérateurs. Un opérateur peut transformer, enrichir ou agréger des données. Pour plus d’informations, consultez Opérateurs d'API DataStream.
- Récepteur : l'application produit des données vers des sources externes à l'aide de récepteurs. Un connecteur récepteur écrit des données dans un flux de données Kinesis, un flux Kinesis Data Firehose, un compartiment Amazon S3, etc. Pour plus d’informations, consultez Récepteurs.

Après avoir créé, compilé et empaqueté votre code d'application, vous chargez le package de code dans un compartiment Amazon Simple Storage Service (Amazon S3). Vous créez ensuite une application de service géré pour Apache Flink. Vous transmettez l'emplacement du package de

code, un flux de données Kinesis comme source de données de streaming et généralement un emplacement de streaming ou de fichier qui reçoit les données traitées par l'application.

Prérequis pour effectuer les exercices

Pour exécuter la procédure indiquée dans ce guide, vous devez disposer des éléments suivants :

- Kit de développement Java (JDK) version 11. Définissez la variable d'environnement `JAVA_HOME` pour qu'elle pointe vers l'emplacement d'installation de votre JDK.
- Nous vous recommandons d'utiliser un environnement de développement (par exemple Eclipse Java Neon ou IntelliJ Idea) pour développer et compiler votre application.
- Client Git. Installez le client Git si vous ne l'avez pas déjà fait.
- Apache Maven Compiler Plugin. Maven doit être installé dans votre chemin de travail. Pour tester votre installation Apache Maven, saisissez les informations suivantes :

```
$ mvn -version
```

Pour démarrer, accédez à Étape 1 : configuration d'un compte AWS et création d'un administrateur.

Étape 1 : configuration d'un compte AWS et création d'un administrateur

S’inscrire à un Compte AWS

Si vous n’avez pas de compte Compte AWS, procédez comme suit pour en créer un.

Pour s’inscrire à un Compte AWS

1. Ouvrez https://portal.aws.amazon.com/billing/signup.
2. Suivez les instructions en ligne.

   Dans le cadre de la procédure d’inscription, vous recevrez un appel téléphonique et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

   Lorsque vous souscrivez à un Compte AWS, un Utilisateur racine d'un compte AWS est créé. Par défaut, seul l’utilisateur root a accès à l’ensemble des Services AWS et des ressources de ce compte. La meilleure pratique de sécurité consiste à attribuer un accès administratif à un utilisateur administratif, et à uniquement utiliser l’utilisateur root pour effectuer tâches nécessitant un accès utilisateur root.

AWS vous envoie un e-mail de confirmation lorsque le processus d'inscription est terminé. Vous pouvez afficher l'activité en cours de votre compte et gérer votre compte à tout moment en accédant à https://aws.amazon.com/ et en cliquant sur Mon compte.

Création d'un utilisateur administratif

Après vous être inscrit à un Compte AWS, sécurisez votre Utilisateur racine d'un compte AWS, activez AWS IAM Identity Center, puis créez un utilisateur administratif afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisation de votre Utilisateur racine d'un compte AWS

1. Connectez-vous à la AWS Management Console en tant que propriétaire du compte en sélectionnant Root user (utilisateur root) et en saisissant l'adresse e-mail de Compte AWS. Sur la page suivante, saisissez votre mot de passe.

   Pour obtenir de l'aide pour vous connecter en utilisant l'utilisateur root, consultez Connexion en tant qu'utilisateur root dans le Guide de l'utilisateur Connexion à AWS.
2. Activez l'authentification multifactorielle (MFA) pour votre utilisateur root.

   Pour obtenir des instructions, consultez Activation d'un dispositif MFA virtuel pour l'utilisateur root de votre Compte AWS (console) dans le Guide de l'utilisateur IAM.

Création d'un utilisateur administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Configuration d'AWS IAM Identity Center dans le guide de l'utilisateur AWS IAM Identity Center.
2. Dans IAM Identity Center, octroyez un accès administratif à un utilisateur administratif.

   Pour profiter d'un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, consultez Configuration de l'accès utilisateur avec le répertoire Répertoire IAM Identity Center par défaut dans le Guide de l'utilisateur AWS IAM Identity Center.

Connexion en tant qu'utilisateur administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l'URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l'utilisateur IAM Identity Center.

Pour obtenir de l'aide pour vous connecter à l'aide d'un utilisateur IAM Identity Center, consultez Connexion au portail d'accès AWS dans le Guide de l'utilisateur Connexion à AWS.

Octroi d'un accès par programmation

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS en dehors de la AWS Management Console. La manière d'octroyer un accès par programmation dépend du type d'utilisateur qui accède à AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l'une des options suivantes.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
Identité de la main-d'œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d'identification temporaires pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour l'AWS CLI, veuillez consulter la rubrique Configuration de l'AWS CLI pour l'utilisation d'AWS IAM Identity Center dans le Guide de l'utilisateur AWS Command Line Interface. • Pour les kits AWS SDK, les outils et les API AWS, consultez Authentification IAM Identity Center dans le Guide de référence des kits SDK et des outils AWS.
IAM	Utilisez des informations d'identification temporaires pour signer des demandes par programmation destinées à	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec des ressources AWS

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
	l'AWS CLI, aux kits SDK AWS ou aux API AWS.	dans le Guide de l'utilisateur IAM.
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour l'AWS CLI, veuillez consulter la rubrique Authentification à l'aide des informations d'identification d'utilisateur IAM dans le Guide de l'utilisateur AWS Command Line Interface. • Pour les kits SDK et les outils AWS, veuillez consulter la rubrique Authentification à l'aide d'informations d'identification à long terme dans le Guide de référence des kits SDK et des outils AWS. • Pour les API AWS, veuillez consulter la rubrique Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Étape suivante

Étape 2 : configuration de AWS Command Line Interface (AWS CLI)

Étape 2 : configuration de AWS Command Line Interface (AWS CLI)

Dans cette étape, vous allez télécharger et configurer la AWS CLI pour l'utiliser avec le service géré pour Apache Flink.

> **Note**
>
> Les exercices de mise en route de ce guide supposent que vous utilisez les informations d'identification d'administrateur (`adminuser`) de votre compte pour effectuer les opérations.

> **Note**
>
> Si l'interface AWS CLI est déjà installée, vous pouvez avoir besoin de procéder à une mise à niveau pour obtenir les dernières fonctionnalités. Pour plus d'informations, consultez Installation d'AWS Command Line Interface dans le Guide de l'utilisateur AWS Command Line Interface. Pour vérifier la version de l'interface AWS CLI, exécutez la commande suivante :
>
> ```
> aws --version
> ```
>
> Les exercices de ce didacticiel nécessitent la version AWS CLI suivante ou une version ultérieure :
>
> ```
> aws-cli/1.16.63
> ```

Pour configurer l’interface AWS CLI

1. Téléchargez et configurez l’interface AWS CLI. Pour obtenir des instructions, consultez les rubriques suivantes dans le Guide de l'utilisateur de l'interface AWS Command Line Interface :
   - Installation de AWS Command Line Interface
   - Configuration de l'interface AWS CLI (français non garanti)
2. Ajoutez un profil désigné pour l'utilisateur administrateur dans le fichier `config` de l’interface AWS CLI. Vous utiliserez ce profil lorsque vous exécuterez les commandes AWS CLI. Pour plus

d'informations sur les profils nommés, consultez la rubrique Profils nommés dans le Guide de l'utilisateur AWS Command Line Interface.

```
[profile adminuser]
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
```

Pour connaître la liste des régions AWS disponibles, consultez Régions et points de terminaison du manuel Référence générale d'Amazon Web Services.

> **Note**
>
> Les exemples de code et de commandes présentés dans ce didacticiel utilisent la région USA Ouest (Oregon). Pour utiliser une autre région, remplacez la région dans le code et les commandes de ce didacticiel par la région que vous souhaitez utiliser.

3. Vérifiez la configuration en saisissant la commande d'aide suivante à l'invite de commande :

```
aws help
```

Après avoir configuré un AWS compteAWS CLI, vous pouvez passer à l'exercice suivant, dans lequel vous configurez un exemple d'application et testez la end-to-end configuration.

Étape suivante

Étape 3 : création et exécution d'une application de service géré pour Apache Flink.

Étape 3 : création et exécution d'une application de service géré pour Apache Flink.

Dans cet exercice, vous allez créer une application de service géré pour Apache Flink avec des flux de données comme source et comme récepteur.

Cette section contient les étapes suivantes :

- Création de deux flux de données Amazon Kinesis Data Streams
- Écriture d'exemples d'enregistrements dans le flux d'entrée

- Téléchargement et examen du code Java Apache Flink
- Compilation du code d'application
- Chargement du code Java Apache Flink
- Création et exécution de l'application de service géré pour Apache Flink
- Étape suivante

Création de deux flux de données Amazon Kinesis Data Streams

Avant de créer une application de service géré pour Apache Flink dans le cadre de cet exercice, commencez par créer deux flux de données Kinesis (`ExampleInputStream` et `ExampleOutputStream`). Votre application utilise ces flux pour les flux source et de destination de l'application.

Vous pouvez créer ces flux à l'aide de la console Amazon Kinesis ou de la commande AWS CLI suivante. Pour obtenir des instructions sur la console, consultez Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams.

Pour créer les flux de données (AWS CLI)

1. Pour créer le premier flux (`ExampleInputStream`) utilisez la commande AWS CLI `create-stream` Amazon Kinesis suivante.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Pour créer le second flux utilisé par l'application pour écrire la sortie, exécutez la même commande en remplaçant le nom du flux par `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Écriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette section, vous utilisez un script Python pour écrire les exemples d'enregistrements dans le flux pour que l'application les traite.

> **Note**
>
> Cette section nécessite le kit AWS SDK for Python (Boto).

1. Créez un fichier nommé `stock.py` avec le contenu suivant :

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

2. Plus loin dans ce didacticiel, vous exécutez le script `stock.py` pour envoyer des données à l'application.

```
$ python stock.py
```

Téléchargement et examen du code Java Apache Flink

Le code de l'application Java pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/GettingStarted`.

Notez les informations suivantes à propos du code d'application :

- Un fichier de modèle d'objet du projet (pom.xml) contient des informations sur la configuration et les dépendances de l'application, y compris les bibliothèques du service géré pour Apache Flink.
- Le fichier `BasicStreamingJob.java` contient la méthode `main` qui définit la fonctionnalité de l'application.
- L'application utilise une source Kinesis pour lire à partir du flux source. L'extrait de code suivant crée la source Kinesis :

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- Votre application crée les connecteurs source et récepteur pour accéder aux ressources externes à l'aide d'un objet `StreamExecutionEnvironment`.
- L'application crée les connecteurs source et récepteur à l'aide de propriétés statiques. Pour utiliser les propriétés de l'application dynamique, utilisez les méthodes `createSourceFromApplicationProperties` et `createSinkFromApplicationProperties` pour créer les connecteurs. Ces méthodes lisent les propriétés de l'application pour configurer les connecteurs.

Pour de plus amples informations sur les propriétés d'exécution, veuillez consulter Propriétés d'exécution.

Compilation du code d'application

Dans cette section, vous allez utiliser le compilateur Apache Maven pour créer le code Java pour l'application. Pour de plus amples informations sur l'installation d'Apache Maven et sur le kit de développement Java (JDK), veuillez consulter Prérequis pour effectuer les exercices.

Pour compiler le code d'application

1. Pour utiliser votre code d'application, vous le compilez et l'intégrez dans un fichier JAR. Vous pouvez compiler et intégrer votre code de deux manières :

   - À l'aide de l'outil de ligne de commande Maven. Créez votre fichier JAR en exécutant la commande suivante dans le répertoire qui contient le fichier `pom.xml` :

     ```
     mvn package -Dflink.version=1.11.3
     ```

   - À l'aide de votre environnement de développement. Consultez la documentation de votre environnement de développement pour plus de détails.

     > **Note**
     >
     > Le code source fourni repose sur les bibliothèques de Java 11. Assurez-vous que la version Java de votre projet est la version 11.

   Vous pouvez charger votre package en tant que fichier JAR, ou compresser le package et le charger en tant que fichier ZIP. Si vous créez votre application à l'aide de l'interface AWS CLI, vous spécifiez le type de contenu de votre code (JAR ou ZIP).

2. En cas d'erreur lors de la compilation, vérifiez que votre variable d'environnement `JAVA_HOME` est correctement définie.

Si la compilation de l'application aboutit, le fichier suivant est créé :

```
target/aws-kinesis-analytics-java-apps-1.0.jar
```

Chargement du code Java Apache Flink

Dans cette section, vous allez créer un compartiment Amazon Simple Storage Service (Amazon S3) et charger votre code d'application.

Pour charger le code d'application

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez Créer un compartiment.
3. Saisissez **`ka-app-code-<username>`** dans le champ Nom du compartiment. Ajoutez un suffixe au nom du compartiment, par exemple votre nom d'utilisateur, pour qu'il soit unique. Choisissez Suivant.
4. À l'étape Configurer les options, conservez les paramètres, puis choisissez Suivant.
5. À l'étape Définir des autorisations, conservez les paramètres, puis choisissez Suivant.
6. Choisissez Créer un compartiment.
7. Dans la console Amazon S3, choisissez le <username>compartiment ka-app-code-, puis Upload.
8. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `aws-kinesis-analytics-java-apps-1.0.jar` que vous avez créé à l'étape précédente. Choisissez Suivant.
9. Vous n'avez pas besoin de modifier les paramètres de l’objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l’application de service géré pour Apache Flink

Vous pouvez créer et exécuter une application de service géré pour Apache Flink à l'aide de la console ou de l’interface AWS CLI.

> **Note**
>
> Lorsque vous créez l'application à l'aide de la console, vos ressources AWS Identity and Access Management (IAM) et Amazon CloudWatch Logs sont créées pour vous. Lorsque vous créez l'application à l'aide de l'interface AWS CLI, vous créez ces ressources séparément.

Rubriques

- Création et exécution de l'application (console)
- Création et exécution de l'application (AWS CLI)

Création et exécution de l'application (console)

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Description, saisissez **`My java test app`**.
   - Pour Exécution, choisissez Apache Flink.
   - Laissez le menu déroulant de la version sur Apache Flink version 1.11 (version recommandée).
4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*

- Rôle : `kinesisanalytics-MyApplication-us-west-2`

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder aux flux de données Kinesis.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **`kinesis-analytics-service-MyApplication-us-west-2`** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*012345678901*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
```

```
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Configuration de l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:

- Pour le compartiment Amazon S3, saisissez **ka-app-code-*<username>***.
- Pour le chemin de l'objet Amazon S3, saisissez **aws-kinesis-analytics-java-apps-1.0.jar**.

3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Sous Propriétés, pour ID de groupe, saisissez **ProducerConfigProperties**.
5. Entrez les valeurs et propriétés d'application suivantes :

ID du groupe	Clé	Valeur
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

6. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
7. Pour la CloudWatch journalisation, cochez la case Activer.
8. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation Amazon, Managed Service for Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`

Exécution de l'application

Le graphique des tâches Flink peut être visualisé en exécutant l'application, en ouvrant le tableau de bord Apache Flink et en choisissant la tâche Flink souhaitée.

Arrêt de l'application

Sur la MyApplicationpage, choisissez Stop. Confirmez l'action.

Mise à jour de l'application

À l'aide de la console, vous pouvez mettre à jour les paramètres d'application tels que les paramètres de surveillance, les propriétés d'application et l'emplacement ou le nom du fichier JAR de l'application. Vous pouvez également recharger le fichier JAR de l'application à partir du compartiment Amazon S3 si vous avez besoin de mettre à jour le code de l'application.

Sur la MyApplicationpage, choisissez Configurer. Mettez à jour les paramètres de l'application, puis choisissez Mettre à jour.

Création et exécution de l'application (AWS CLI)

Dans cette section, vous utilisez l'interface AWS CLI pour créer et exécuter l'application de service géré pour Apache Flink. Un service géré pour Apache Flink utilise la commande AWS CLI `kinesisanalyticsv2` pour créer et interagir avec les applications de service géré pour Apache Flink.

Créer une stratégie d'autorisations

> **Note**
>
> Vous devez créer une stratégie d'autorisations et un rôle pour votre application. Si vous ne créez pas ces ressources IAM, votre application ne peut pas accéder à ses flux de données et de journaux.

Vous commencez par créer une stratégie d'autorisations avec deux instructions : une qui accorde des autorisations pour l'action `read` sur le flux source et une autre qui accorde des autorisations pour les actions `write` sur le flux récepteur. Vous attachez ensuite la politique à un rôle IAM (que vous allez créer dans la section suivante). Ainsi, lorsque le service géré pour Apache Flink assume le rôle, le

service dispose des autorisations nécessaires pour lire à partir du flux source et écrire dans le flux récepteur.

Utilisez le code suivant pour créer la stratégie d'autorisations `AKReadSourceStreamWriteSinkStream`. Remplacez *`username`* par le nom d'utilisateur que vous avez utilisé pour créer le compartiment Amazon S3 pour stocker le code d'application. Remplacez l'ID de compte dans Amazon Resource Names (ARN) (*`012345678901`*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": ["arn:aws:s3:::ka-app-code-username",
                "arn:aws:s3:::ka-app-code-username/*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Pour step-by-step obtenir des instructions sur la création d'une politique d'autorisations, voir Tutoriel : créer et joindre votre première politique gérée par le client dans le guide de l'utilisateur IAM.

> **Note**
>
> Pour accéder à d'autres services Amazon, vous pouvez utiliser le AWS SDK for Java. Le service géré pour Apache Flink définit automatiquement les informations d'identification requises par le kit SDK en fonction du rôle IAM d'exécution du service associé à votre application. Aucune étape supplémentaire n'est nécessaire.

Créer un rôle IAM

Dans cette section, vous créez un rôle IAM que l'application de service géré pour Apache Flink peut assumer pour lire un flux source et écrire dans le flux récepteur.

Le service géré pour Apache Flink ne peut pas accéder à votre flux sans autorisation. Vous utilisez un rôle IAM pour accorder ces autorisations. Deux politiques sont attachées à chaque rôle IAM. La politique d'approbation accorde au service géré pour Apache Flink l'autorisation d'assumer le rôle, et la politique d'autorisation détermine ce que le service géré pour Apache Flink peut faire après avoir assumé le rôle.

Vous attachez la stratégie d'autorisations que vous avez créée dans la section précédente à ce rôle.

Pour créer un rôle IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, choisissez Rôles, puis Créer un rôle.
3. Sous Sélectionner le type d'identité approuvée, choisissez Service AWS. Sous Choisir le service qui utilisera ce rôle, choisissez EC2. Sous Sélectionner votre cas d'utilisation, choisissez Kinesis Analytics.

   Sélectionnez Next: Permissions (Étape suivante : autorisations).
4. Dans la page Attacher des stratégies d'autorisations, choisissez Suivant : vérification. Vous attachez des stratégies d'autorisations après avoir créé le rôle.
5. Sur la page Créer un rôle, saisissez **`MF-stream-rw-role`** pour le Nom du rôle. Sélectionnez Créer un rôle.

   Vous venez de créer un nouveau rôle IAM appelé `MF-stream-rw-role`. Ensuite, vous mettez à jour les stratégies d'approbation et d'autorisation pour le rôle.
6. Attachez la politique d'autorisation au rôle.

> **Note**
>
> Dans le cadre de cet exercice, le service géré pour Apache Flink assume ce rôle à la fois pour la lecture des données à partir d'un flux de données Kinesis (source) et pour l'écriture des résultats dans un autre flux de données Kinesis. Vous attachez donc la stratégie que vous avez créée à l'étape précédente, the section called “Créer une stratégie d'autorisations”.

a. Sur la page Récapitulatif, choisissez l'onglet Autorisations.

b. Choisissez Attacher des stratégies.

c. Dans la zone de recherche, saisissez **`AKReadSourceStreamWriteSinkStream`** (la stratégie que vous avez créée dans la section précédente).

d. Choisissez la ReadSourceStreamWriteSinkStream politique AK, puis choisissez Attach policy.

Vous avez maintenant créé le rôle d'exécution de service que votre application utilise pour accéder aux ressources. Notez l'ARN du nouveau rôle.

Pour step-by-step obtenir des instructions sur la création d'un rôle, consultez la section Création d'un rôle IAM (console) dans le guide de l'utilisateur IAM.

Création de l’application de service géré pour Apache Flink

1. Copiez le code JSON suivant dans un fichier nommé `create_request.json`. Remplacez l'exemple d'ARN du rôle par l'ARN du rôle que vous avez créé précédemment. Remplacez le suffixe de l'ARN du compartiment (*`username`*) par le suffixe que vous avez choisi dans la section précédente. Remplacez l'exemple d'ID de compte (*`012345678901`*) dans le rôle d'exécution de service par votre ID de compte.

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_11",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
```

```
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "aws-kinesis-analytics-java-apps-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
    }
}
```

2. Exécutez l'action [CreateApplication](#) avec la demande précédente pour créer l'application :

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

L'application est maintenant créée. Vous démarrez l'application dans l'étape suivante.

Démarrage de l'application

Dans cette section, vous utilisez l'action [StartApplication](#) pour démarrer l'application.

Pour démarrer l'application

1. Copiez le code JSON suivant dans un fichier nommé `start_request.json`.

```
{
    "ApplicationName": "test",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. Exécutez l'action `StartApplication` avec la demande précédente pour démarrer l'application :

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

L'application est maintenant en cours d'exécution. Vous pouvez consulter les métriques du service géré pour Apache Flink sur la CloudWatch console Amazon pour vérifier que l'application fonctionne.

Arrêt de l'application

Dans cette section, vous allez utiliser l'action `StopApplication` pour arrêter l'application.

Pour arrêter l'application

1. Copiez le code JSON suivant dans un fichier nommé `stop_request.json`.

```
{
    "ApplicationName": "test"
}
```

2. Exécutez l'action `StopApplication` avec la demande suivante pour arrêter l'application :

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

L'application est maintenant arrêtée.

Ajouter une option de CloudWatch journalisation

Vous pouvez utiliser le AWS CLI pour ajouter un flux de CloudWatch journal Amazon à votre application. Pour plus d'informations sur l'utilisation de CloudWatch Logs avec votre application, consultezthe section called “Configuration de la journalisation”.

Mettre à jour des propriétés d'environnement

Dans cette section, vous utilisez l'action `UpdateApplication` pour modifier les propriétés d'environnement de l'application sans recompiler le code de l'application. Dans cet exemple, vous modifiez la région des flux source et de destination.

Pour mettre à jour des propriétés d'environnement pour l'application

1. Copiez le code JSON suivant dans un fichier nommé `update_properties_request.json`.

```
{"ApplicationName": "test",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
   }
}
```

2. Exécutez l'action `UpdateApplication` avec la demande précédente pour mettre à jour les propriétés de l'environnement :

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Mise à jour du code de l'application

Lorsque vous devez mettre à jour le code de votre application avec une nouvelle version de votre package de code, vous utilisez l'action`UpdateApplication` de l'interface AWS CLI.

> **Note**
>
> Pour charger une nouvelle version du code de l'application portant le même nom de fichier, vous devez spécifier la nouvelle version de l'objet. Pour de plus amples informations sur l'utilisation des versions d'objet Amazon S3, veuillez consulter Activation et désactivation de la gestion des versions.

Pour utiliser l'interface AWS CLI, supprimez votre ancien package de code de votre compartiment Amazon S3, téléchargez la nouvelle version et appelez `UpdateApplication` en spécifiant le même compartiment Amazon S3 et le même nom d'objet, ainsi que la nouvelle version de l'objet. L'application redémarrera avec le nouveau package de code.

L'exemple de demande d'action `UpdateApplication` suivant recharge le code de l'application et redémarre l'application. Mettez à jour l'`CurrentApplicationVersionId` à la version actuelle de l'application. Vous pouvez vérifier la version actuelle de l'application à l'aide des actions `ListApplications` ou `DescribeApplication`. Mettez à jour le suffixe du nom du compartiment (<username>) avec le suffixe que vous avez choisi dans la section the section called "Création de deux flux de données Amazon Kinesis Data Streams".

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "aws-kinesis-analytics-java-apps-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
```

```
                }
            }
        }
    }
}
```

Étape suivante

Etape 4 : nettoyage des ressources AWS

Etape 4 : nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Mise en route.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer vos flux de données Kinesis
- Supprimer votre objet et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources
- Étape suivante

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Managed Service for Apache Flink, choisissez MyApplication.
3. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer vos flux de données Kinesis

1. Ouvrez la console du service géré pour Apache Flink à l’adresse https://console.aws.amazon.com/flink
2. Dans le panneau Kinesis Data Streams, ExampleInputStreamsélectionnez.
3. Sur la ExampleInputStreampage, choisissez Supprimer Kinesis Stream, puis confirmez la suppression.

4. Sur la page Kinesis Streams, choisissez le ExampleOutputStream, choisissez Actions, choisissez Supprimer, puis confirmez la suppression.

Supprimer votre objet et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code -. <username>
3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.
8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Étape suivante

Étape 5 : étapes suivantes

Étape 5 : étapes suivantes

Maintenant que vous avez créé et exécuté une application de service géré de base pour Apache Flink, consultez les ressources suivantes pour des solutions de service géré plus avancées pour Apache Flink.

- Solution de streaming de données AWS pour Amazon Kinesis : la solution de streaming de données AWS pour Amazon Kinesis configure automatiquement les services AWS nécessaires pour capturer, stocker, traiter et diffuser facilement des données de streaming. La solution propose plusieurs options pour résoudre les problèmes d'utilisation de données en streaming. L'option Managed Service for Apache Flink fournit un exemple de end-to-end streaming ETL illustrant une application réelle qui exécute des opérations analytiques sur des données de taxis simulées à New York. La solution met en place toutes les AWS ressources nécessaires, telles que les rôles et les politiques IAM, un CloudWatch tableau de bord et des CloudWatch alarmes.
- Solution de données de streaming AWS pour Amazon MSK :la solution de données de streaming AWS pour Amazon MSK fournit des modèles AWS CloudFormation dans lesquels les données circulent entre les producteurs, le stockage en streaming, les consommateurs et les destinations.
- Clickstream Lab avec Apache Flink et Apache Kafka : un laboratoire de bout en bout pour les cas d'utilisation d’Amazon Managed Streaming for Apache Kafka pour le stockage de streaming et le service géré pour Apache Flink pour les applications Apache Flink pour le traitement des flux.
- Amazon Managed Service for Apache Flink Workshop : dans cet atelier, vous allez créer une architecture de end-to-end streaming pour ingérer, analyser et visualiser les données de streaming en temps quasi réel. Vous avez décidé d'améliorer les opérations d'une compagnie de taxi à New York. Vous analysez les données de télémétrie d'une flotte de taxis à New York en temps quasi réel afin d'optimiser le fonctionnement de la flotte.
- Service géré Amazon pour Apache Flink : exemples : cette section de ce guide du développeur fournit des exemples de création et d'utilisation d'applications dans le service géré pour Apache Flink. Ils incluent des exemples de code et des step-by-step instructions pour vous aider à créer un service géré pour les applications Apache Flink et à tester vos résultats.
- Learn Flink : Hands On Training : formation d'introduction officielle à Apache Flink qui vous permet de commencer à écrire des applications ETL, analytiques et axées sur les événements évolutives pour le streaming.

> **Note**
>
> Sachez que le service géré pour Apache Flink ne prend pas en charge la version Apache Flink (1.12) utilisée dans cette formation. Vous pouvez utiliser Flink 1.15.2 dans le service géré Flink pour Apache Flink.

- Exemples de code Apache Flink : GitHub référentiel contenant une grande variété d'exemples d'applications Apache Flink.

Mise en route : Flink 1.8.2

Cette rubrique contient une version du didacticiel Mise en route (DataStream API) qui utilise Apache Flink 1.8.2.

Rubriques

- Composants d’une application de service géré pour Apache Flink
- Prérequis pour effectuer les exercices
- Étape 1 : configuration d'un compte AWS et création d'un administrateur
- Étape 2 : configuration de AWS Command Line Interface (AWS CLI)
- Étape 3 : création et exécution d'une application de service géré pour Apache Flink.
- Etape 4 : nettoyage des ressources AWS

Composants d’une application de service géré pour Apache Flink

Pour traiter les données, votre application de service géré pour Apache Flink utilise une application Java/Apache Maven ou Scala qui traite les entrées et produit des sorties à l'aide de l’exécution Apache Flink.

Une application de service géré for Apache Flink comprend les composants suivants :

- Propriétés d'exécution : vous pouvez utiliser les propriétés d'exécution pour configurer votre application sans recompiler le code de votre application.
- Source : l'application consomme des données en utilisant une source. Un connecteur source lit les données d'un flux de données Kinesis, d'un compartiment Amazon S3, etc. Pour plus d’informations, consultez Sources.

- Opérateurs : l'application traite les données à l'aide d'un ou de plusieurs opérateurs. Un opérateur peut transformer, enrichir ou agréger des données. Pour plus d'informations, consultez Opérateurs d'API DataStream.
- Récepteur : l'application produit des données vers des sources externes à l'aide de récepteurs. Un connecteur récepteur écrit des données dans un flux de données Kinesis, un flux Kinesis Data Firehose, un compartiment Amazon S3, etc. Pour plus d'informations, consultez Récepteurs.

Après avoir créé, compilé et empaqueté votre code d'application, vous chargez le package de code dans un compartiment Amazon Simple Storage Service (Amazon S3). Vous créez ensuite une application de service géré pour Apache Flink. Vous transmettez l'emplacement du package de code, un flux de données Kinesis comme source de données de streaming et généralement un emplacement de streaming ou de fichier qui reçoit les données traitées par l'application.

Prérequis pour effectuer les exercices

Pour exécuter la procédure indiquée dans ce guide, vous devez disposer des éléments suivants :

- Kit de développement Java (JDK) version 8. Définissez la variable d'environnement `JAVA_HOME` pour qu'elle pointe vers l'emplacement d'installation de votre JDK.
- Pour utiliser le connecteur Kinesis Apache Flink dans ce didacticiel, vous devez télécharger et installer Apache Flink. Pour plus de détails, consultez Utilisation du connecteur Kinesis Streams d'Apache Flink avec les versions précédentes d'Apache Flink.
- Nous vous recommandons d'utiliser un environnement de développement (par exemple Eclipse Java Neon ou IntelliJ Idea) pour développer et compiler votre application.
- Client Git. Installez le client Git si vous ne l'avez pas déjà fait.
- Apache Maven Compiler Plugin. Maven doit être installé dans votre chemin de travail. Pour tester votre installation Apache Maven, saisissez les informations suivantes :

```
$ mvn -version
```

Pour démarrer, accédez à Étape 1 : configuration d'un compte AWS et création d'un administrateur.

Étape 1 : configuration d'un compte AWS et création d'un administrateur

S’inscrire à un Compte AWS

Si vous n’avez pas de compte Compte AWS, procédez comme suit pour en créer un.

Pour s’inscrire à un Compte AWS

1. Ouvrez https://portal.aws.amazon.com/billing/signup.
2. Suivez les instructions en ligne.

   Dans le cadre de la procédure d’inscription, vous recevrez un appel téléphonique et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

   Lorsque vous souscrivez à un Compte AWS, un Utilisateur racine d'un compte AWS est créé. Par défaut, seul l’utilisateur root a accès à l’ensemble des Services AWS et des ressources de ce compte. La meilleure pratique de sécurité consiste à attribuer un accès administratif à un utilisateur administratif, et à uniquement utiliser l’utilisateur root pour effectuer tâches nécessitant un accès utilisateur root.

AWS vous envoie un e-mail de confirmation lorsque le processus d’inscription est terminé. Vous pouvez afficher l’activité en cours de votre compte et gérer votre compte à tout moment en accédant à https://aws.amazon.com/ et en cliquant sur Mon compte.

Création d’un utilisateur administratif

Après vous être inscrit à un Compte AWS, sécurisez votre Utilisateur racine d'un compte AWS, activez AWS IAM Identity Center, puis créez un utilisateur administratif afin de ne pas utiliser l’utilisateur root pour les tâches quotidiennes.

Sécurisation de votre Utilisateur racine d'un compte AWS

1. Connectez-vous à la AWS Management Console en tant que propriétaire du compte en sélectionnant Root user (utilisateur root) et en saisissant l’adresse e-mail de Compte AWS. Sur la page suivante, saisissez votre mot de passe.

   Pour obtenir de l’aide pour vous connecter en utilisant l’utilisateur root, consultez Connexion en tant qu’utilisateur root dans le Guide de l’utilisateur Connexion à AWS.
2. Activez l’authentification multifactorielle (MFA) pour votre utilisateur root.

Pour obtenir des instructions, consultez Activation d'un dispositif MFA virtuel pour l'utilisateur root de votre Compte AWS (console) dans le Guide de l'utilisateur IAM.

Création d'un utilisateur administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Configuration d'AWS IAM Identity Center dans le guide de l'utilisateur AWS IAM Identity Center.
2. Dans IAM Identity Center, octroyez un accès administratif à un utilisateur administratif.

   Pour profiter d'un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, consultez Configuration de l'accès utilisateur avec le répertoire Répertoire IAM Identity Center par défaut dans le Guide de l'utilisateur AWS IAM Identity Center.

Connexion en tant qu'utilisateur administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l'URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l'utilisateur IAM Identity Center.

  Pour obtenir de l'aide pour vous connecter à l'aide d'un utilisateur IAM Identity Center, consultez Connexion au portail d'accès AWS dans le Guide de l'utilisateur Connexion à AWS.

Octroi d'un accès par programmation

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS en dehors de la AWS Management Console. La manière d'octroyer un accès par programmation dépend du type d'utilisateur qui accède à AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l'une des options suivantes.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
Identité de la main-d'œuvre	Utilisez des informations d'identification temporaires pour signer des demandes par	Suivez les instructions de l'interface que vous souhaitez utiliser.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
(Utilisateurs gérés dans IAM Identity Center)	programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	• Pour l'AWS CLI, veuillez consulter la rubrique Configuration de l'AWS CLI pour l'utilisation d'AWS IAM Identity Center dans le Guide de l'utilisateur AWS Command Line Interface. • Pour les kits AWS SDK, les outils et les API AWS, consultez Authentification IAM Identity Center dans le Guide de référence des kits SDK et des outils AWS.
IAM	Utilisez des informations d'identification temporaires pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de la section Utilisation d'informations d'identification temporaires avec des ressources AWS dans le Guide de l'utilisateur IAM.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour l'AWS CLI, veuillez consulter la rubrique Authentification à l'aide des informations d'identification d'utilisateur IAM dans le Guide de l'utilisateur AWS Command Line Interface. • Pour les kits SDK et les outils AWS, veuillez consulter la rubrique Authentification à l'aide d'informations d'identification à long terme dans le Guide de référence des kits SDK et des outils AWS. • Pour les API AWS, veuillez consulter la rubrique Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Étape 2 : configuration de AWS Command Line Interface (AWS CLI)

Dans cette étape, vous allez télécharger et configurer la AWS CLI pour l'utiliser avec le service géré pour Apache Flink.

> **Note**
>
> Les exercices de mise en route de ce guide supposent que vous utilisez les informations d'identification d'administrateur (`adminuser`) de votre compte pour effectuer les opérations.

> **Note**
>
> Si l'interface AWS CLI est déjà installée, vous pouvez avoir besoin de procéder à une mise à niveau pour obtenir les dernières fonctionnalités. Pour plus d'informations, consultez Installation d'AWS Command Line Interface dans le Guide de l'utilisateur AWS Command Line Interface. Pour vérifier la version de l'interface AWS CLI, exécutez la commande suivante :
>
> ```
> aws --version
> ```
>
> Les exercices de ce didacticiel nécessitent la version AWS CLI suivante ou une version ultérieure :
>
> ```
> aws-cli/1.16.63
> ```

Pour configurer l’interface AWS CLI

1. Téléchargez et configurez l’interface AWS CLI. Pour obtenir des instructions, consultez les rubriques suivantes dans le Guide de l'utilisateur de l'interface AWS Command Line Interface :
   - Installation de AWS Command Line Interface
   - Configuration de l'interface AWS CLI (français non garanti)
2. Ajoutez un profil désigné pour l'utilisateur administrateur dans le fichier `config` de l’interface AWS CLI. Vous utiliserez ce profil lorsque vous exécuterez les commandes AWS CLI. Pour plus d'informations sur les profils nommés, consultez la rubrique Profils nommés dans le Guide de l'utilisateur AWS Command Line Interface.

   ```
   [profile adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   ```

```
region = aws-region
```

Pour obtenir la liste des régions disponibles, consultez Régions et points de terminaison dans la documentation Référence générale d'Amazon Web Services.

> **Note**
>
> Les exemples de code et de commandes présentés dans ce didacticiel utilisent la région USA Ouest (Oregon). Pour utiliser une autre région AWS, remplacez la région dans le code et les commandes de ce didacticiel par la région que vous souhaitez utiliser.

3. Vérifiez la configuration en saisissant la commande d'aide suivante à l'invite de commande :

```
aws help
```

Après avoir configuré un AWS compteAWS CLI, vous pouvez passer à l'exercice suivant, dans lequel vous configurez un exemple d'application et testez la end-to-end configuration.

Étape suivante

Étape 3 : création et exécution d'une application de service géré pour Apache Flink.

Étape 3 : création et exécution d'une application de service géré pour Apache Flink.

Dans cet exercice, vous allez créer une application de service géré pour Apache Flink avec des flux de données comme source et comme récepteur.

Cette section contient les étapes suivantes :

- Création de deux flux de données Amazon Kinesis Data Streams
- Écriture d'exemples d'enregistrements dans le flux d'entrée
- Téléchargement et examen du code Java Apache Flink
- Compilation du code d'application
- Chargement du code Java Apache Flink
- Création et exécution de l’application de service géré pour Apache Flink
- Étape suivante

Création de deux flux de données Amazon Kinesis Data Streams

Avant de créer une application de service géré pour Apache Flink dans le cadre de cet exercice, commencez par créer deux flux de données Kinesis (`ExampleInputStream` et `ExampleOutputStream`). Votre application utilise ces flux pour les flux source et de destination de l'application.

Vous pouvez créer ces flux à l'aide de la console Amazon Kinesis ou de la commande AWS CLI suivante. Pour obtenir des instructions sur la console, consultez Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams.

Pour créer les flux de données (AWS CLI)

1. Pour créer le premier flux (`ExampleInputStream`) utilisez la commande AWS CLI `create-stream` Amazon Kinesis suivante.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Pour créer le second flux utilisé par l'application pour écrire la sortie, exécutez la même commande en remplaçant le nom du flux par `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Écriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette section, vous utilisez un script Python pour écrire les exemples d'enregistrements dans le flux pour que l'application les traite.

> **Note**
>
> Cette section nécessite le kit AWS SDK for Python (Boto).

1. Créez un fichier nommé `stock.py` avec le contenu suivant :

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

2. Plus loin dans ce didacticiel, vous exécutez le script `stock.py` pour envoyer des données à l'application.

```
$ python stock.py
```

Téléchargement et examen du code Java Apache Flink

Le code de l'application Java pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/GettingStarted_1_8`.

Notez les informations suivantes à propos du code d'application :

- Un fichier de modèle d'objet du projet (pom.xml) contient des informations sur la configuration et les dépendances de l'application, y compris les bibliothèques du service géré pour Apache Flink.
- Le fichier `BasicStreamingJob.java` contient la méthode `main` qui définit la fonctionnalité de l'application.
- L'application utilise une source Kinesis pour lire à partir du flux source. L'extrait de code suivant crée la source Kinesis :

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- Votre application crée les connecteurs source et récepteur pour accéder aux ressources externes à l'aide d'un objet `StreamExecutionEnvironment`.
- L'application crée les connecteurs source et récepteur à l'aide de propriétés statiques. Pour utiliser les propriétés de l'application dynamique, utilisez les méthodes `createSourceFromApplicationProperties` et `createSinkFromApplicationProperties` pour créer les connecteurs. Ces méthodes lisent les propriétés de l'application pour configurer les connecteurs.

  Pour de plus amples informations sur les propriétés d'exécution, veuillez consulter Propriétés d'exécution.

Compilation du code d'application

Dans cette section, vous allez utiliser le compilateur Apache Maven pour créer le code Java pour l'application. Pour de plus amples informations sur l'installation d'Apache Maven et sur le kit de développement Java (JDK), veuillez consulter Prérequis pour effectuer les exercices.

> **Note**
>
> Pour utiliser le connecteur Kinesis avec les versions d'Apache Flink antérieures à la version 1.11, vous devez télécharger, compiler et installer Apache Maven. Pour plus d'informations, consultez the section called “Utilisation du connecteur Kinesis Streams d’Apache Flink avec les versions précédentes d'Apache Flink”.

Pour compiler le code d'application

1. Pour utiliser votre code d'application, vous le compilez et l'intégrez dans un fichier JAR. Vous pouvez compiler et intégrer votre code de deux manières :

   - À l'aide de l'outil de ligne de commande Maven. Créez votre fichier JAR en exécutant la commande suivante dans le répertoire qui contient le fichier `pom.xml` :

     ```
     mvn package -Dflink.version=1.8.2
     ```

   - À l'aide de votre environnement de développement. Consultez la documentation de votre environnement de développement pour plus de détails.

     > **Note**
     >
     > Le code source fourni repose sur les bibliothèques de Java 1.8. Assurez-vous que la version Java de votre projet est la version 1.8.

   Vous pouvez charger votre package en tant que fichier JAR, ou compresser le package et le charger en tant que fichier ZIP. Si vous créez votre application à l'aide de l'interface AWS CLI, vous spécifiez le type de contenu de votre code (JAR ou ZIP).

2. En cas d'erreur lors de la compilation, vérifiez que votre variable d'environnement `JAVA_HOME` est correctement définie.

Si la compilation de l'application aboutit, le fichier suivant est créé :

`target/aws-kinesis-analytics-java-apps-1.0.jar`

Chargement du code Java Apache Flink

Dans cette section, vous allez créer un compartiment Amazon Simple Storage Service (Amazon S3) et charger votre code d'application.

Pour charger le code d'application

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez Créer un compartiment.
3. Saisissez **`ka-app-code-<username>`** dans le champ Nom du compartiment. Ajoutez un suffixe au nom du compartiment, par exemple votre nom d'utilisateur, pour qu'il soit unique. Choisissez Suivant.
4. À l'étape Configurer les options, conservez les paramètres, puis choisissez Suivant.
5. À l'étape Définir des autorisations, conservez les paramètres, puis choisissez Suivant.
6. Choisissez Créer un compartiment.
7. Dans la console Amazon S3, choisissez le <username>compartiment ka-app-code-, puis Upload.
8. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `aws-kinesis-analytics-java-apps-1.0.jar` que vous avez créé à l'étape précédente. Choisissez Suivant.
9. Vous n'avez pas besoin de modifier les paramètres de l'objet, donc choisissez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l'application de service géré pour Apache Flink

Vous pouvez créer et exécuter une application de service géré pour Apache Flink à l'aide de la console ou de l'interface AWS CLI.

> **Note**
>
> Lorsque vous créez l'application à l'aide de la console, vos ressources AWS Identity and Access Management (IAM) et Amazon CloudWatch Logs sont créées pour vous. Lorsque vous créez l'application à l'aide de l'interface AWS CLI, vous créez ces ressources séparément.

Rubriques

- Création et exécution de l'application (console)
- Création et exécution de l'application (AWS CLI)

Création et exécution de l'application (console)

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Description, saisissez **`My java test app`**.
   - Pour Exécution, choisissez Apache Flink.
   - Laissez le menu déroulant de la version sur Apache Flink version 1.8 (version recommandée).
4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :
>
> - Stratégie : `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rôle : `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder aux flux de données Kinesis.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **`kinesis-analytics-service-MyApplication-us-west-2`** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*012345678901*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/aws-kinesis-analytics-java-
apps-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
```

```
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Configuration de l'application

1. Sur la MyApplicationpage, choisissez Configurer.
2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **ka-app-code-*<username>***.
   - Pour le chemin de l'objet Amazon S3, saisissez **aws-kinesis-analytics-java-apps-1.0.jar**.

3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **`kinesis-analytics-MyApplication-us-west-2`**.
4. Entrez les valeurs et propriétés d'application suivantes :

ID du groupe	Clé	Valeur
`ProducerConfigProperties`	**`flink.inputstream.initpos`**	**`LATEST`**
`ProducerConfigProperties`	**`aws.region`**	**`us-west-2`**
`ProducerConfigProperties`	**`AggregationEnabled`**	**`false`**

5. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
6. Pour la CloudWatch journalisation, cochez la case Activer.
7. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation Amazon, Managed Service for Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`

Exécution de l'application

1. Sur la MyApplicationpage, choisissez Exécuter. Confirmez l'action.
2. Lorsque l'application est en cours d'exécution, actualisez la page. La console affiche le graphique de l'application.

Arrêt de l'application

Sur la MyApplicationpage, choisissez Stop. Confirmez l'action.

Mise à jour de l'application

À l'aide de la console, vous pouvez mettre à jour les paramètres d'application tels que les paramètres de surveillance, les propriétés d'application et l'emplacement ou le nom du fichier JAR de l'application. Vous pouvez également recharger le fichier JAR de l'application à partir du compartiment Amazon S3 si vous avez besoin de mettre à jour le code de l'application.

Sur la MyApplicationpage, choisissez Configurer. Mettez à jour les paramètres de l'application, puis choisissez Mettre à jour.

Création et exécution de l'application (AWS CLI)

Dans cette section, vous allez utiliser l'interface AWS CLI pour créer et exécuter l'application de service géré pour Apache Flink. Le service géré pour Apache Flink utilise la commande AWS CLI `kinesisanalyticsv2` pour créer et interagir avec les applications de service géré pour Apache Flink.

Créer une stratégie d'autorisations

> **Note**
>
> Vous devez créer une stratégie d'autorisations et un rôle pour votre application. Si vous ne créez pas ces ressources IAM, votre application ne peut pas accéder à ses flux de données et de journaux.

Vous commencez par créer une stratégie d'autorisations avec deux instructions : une qui accorde des autorisations pour l'action `read` sur le flux source et une autre qui accorde des autorisations pour les actions `write` sur le flux récepteur. Vous attachez ensuite la politique à un rôle IAM (que vous allez créer dans la section suivante). Ainsi, lorsque le service géré pour Apache Flink assume le rôle, le service dispose des autorisations nécessaires pour lire à partir du flux source et écrire dans le flux récepteur.

Utilisez le code suivant pour créer la stratégie d'autorisations `AKReadSourceStreamWriteSinkStream`. Remplacez *`username`* par le nom d'utilisateur

que vous avez utilisé pour créer le compartiment Amazon S3 pour stocker le code d'application. Remplacez l'ID de compte dans Amazon Resource Names (ARN) (*012345678901*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": ["arn:aws:s3:::ka-app-code-username",
                "arn:aws:s3:::ka-app-code-username/*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Pour step-by-step obtenir des instructions sur la création d'une politique d'autorisations, voir Tutoriel : créer et joindre votre première politique gérée par le client dans le guide de l'utilisateur IAM.

> **Note**
>
> Pour accéder à d'autres services Amazon, vous pouvez utiliser le AWS SDK for Java. Le service géré pour Apache Flink définit automatiquement les informations d'identification

requises par le kit SDK en fonction du rôle IAM d'exécution du service associé à votre application. Aucune étape supplémentaire n'est nécessaire.

Créer un rôle IAM

Dans cette section, vous créez un rôle IAM que l'application de service géré pour Apache Flink peut assumer pour lire un flux source et écrire dans le flux récepteur.

Le service géré pour Apache Flink ne peut pas accéder à votre flux sans autorisation. Vous utilisez un rôle IAM pour accorder ces autorisations. Deux politiques sont attachées à chaque rôle IAM. La politique d'approbation accorde au service géré pour Apache Flink l'autorisation d'assumer le rôle, et la politique d'autorisation détermine ce que le service géré pour Apache Flink peut faire après avoir assumé le rôle.

Vous attachez la stratégie d'autorisations que vous avez créée dans la section précédente à ce rôle.

Pour créer un rôle IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, choisissez Rôles, puis Créer un rôle.
3. Sous Sélectionner le type d'identité approuvée, choisissez Service AWS. Sous Choisir le service qui utilisera ce rôle, choisissez EC2. Sous Sélectionner votre cas d'utilisation, choisissez Kinesis Analytics.

   Sélectionnez Next: Permissions (Étape suivante : autorisations).
4. Dans la page Attacher des stratégies d'autorisations, choisissez Suivant : vérification. Vous attachez des stratégies d'autorisations après avoir créé le rôle.
5. Sur la page Créer un rôle, saisissez **`MF-stream-rw-role`** pour le Nom du rôle. Sélectionnez Créer un rôle.

   Vous venez de créer un nouveau rôle IAM appelé `MF-stream-rw-role`. Ensuite, vous mettez à jour les stratégies d'approbation et d'autorisation pour le rôle.
6. Attachez la politique d'autorisation au rôle.

> **Note**
>
> Dans le cadre de cet exercice, le service géré pour Apache Flink assume ce rôle à la fois pour la lecture des données à partir d'un flux de données Kinesis (source) et pour

l'écriture des résultats dans un autre flux de données Kinesis. Vous attachez donc la stratégie que vous avez créée à l'étape précédente, the section called “Créer une stratégie d'autorisations”.

a. Sur la page Récapitulatif, choisissez l'onglet Autorisations.
b. Choisissez Attacher des stratégies.
c. Dans la zone de recherche, saisissez **`AKReadSourceStreamWriteSinkStream`** (la stratégie que vous avez créée dans la section précédente).
d. Choisissez la ReadSourceStreamWriteSinkStream politique AK, puis choisissez Attach policy.

Vous avez maintenant créé le rôle d'exécution de service que votre application utilise pour accéder aux ressources. Notez l'ARN du nouveau rôle.

Pour step-by-step obtenir des instructions sur la création d'un rôle, consultez la section Création d'un rôle IAM (console) dans le guide de l'utilisateur IAM.

Création de l’application de service géré pour Apache Flink

1. Copiez le code JSON suivant dans un fichier nommé `create_request.json`. Remplacez l'exemple d'ARN du rôle par l'ARN du rôle que vous avez créé précédemment. Remplacez le suffixe de l'ARN du compartiment (*`username`*) par le suffixe que vous avez choisi dans la section précédente. Remplacez l'exemple d'ID de compte (*`012345678901`*) dans le rôle d'exécution de service par votre ID de compte.

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_8",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "aws-kinesis-analytics-java-apps-1.0.jar"
                }
            },
```

```
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
    }
}
```

2. Exécutez l'action [CreateApplication](#) avec la demande précédente pour créer l'application :

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

L'application est maintenant créée. Vous démarrez l'application dans l'étape suivante.

Démarrage de l'application

Dans cette section, vous utilisez l'action [StartApplication](#) pour démarrer l'application.

Pour démarrer l'application

1. Copiez le code JSON suivant dans un fichier nommé `start_request.json`.

```
{
    "ApplicationName": "test",
    "RunConfiguration": {
```

```
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. Exécutez l'action StartApplication avec la demande précédente pour démarrer l'application :

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

L'application est maintenant en cours d'exécution. Vous pouvez consulter les métriques du service géré pour Apache Flink sur la CloudWatch console Amazon pour vérifier que l'application fonctionne.

Arrêt de l'application

Dans cette section, vous allez utiliser l'action StopApplication pour arrêter l'application.

Pour arrêter l'application

1. Copiez le code JSON suivant dans un fichier nommé `stop_request.json`.

```
{
   "ApplicationName": "test"
}
```

2. Exécutez l'action StopApplication avec la demande suivante pour arrêter l'application :

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

L'application est maintenant arrêtée.

Ajouter une option de CloudWatch journalisation

Vous pouvez utiliser le AWS CLI pour ajouter un flux de CloudWatch journal Amazon à votre application. Pour plus d'informations sur l'utilisation de CloudWatch Logs avec votre application, consultezthe section called “Configuration de la journalisation”.

Mettre à jour des propriétés d'environnement

Dans cette section, vous utilisez l'action `UpdateApplication` pour modifier les propriétés d'environnement de l'application sans recompiler le code de l'application. Dans cet exemple, vous modifiez la région des flux source et de destination.

Pour mettre à jour des propriétés d'environnement pour l'application

1. Copiez le code JSON suivant dans un fichier nommé `update_properties_request.json`.

```
{"ApplicationName": "test",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
   }
}
```

2. Exécutez l'action `UpdateApplication` avec la demande précédente pour mettre à jour les propriétés de l'environnement :

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Mise à jour du code de l'application

Lorsque vous devez mettre à jour le code de votre application avec une nouvelle version de votre package de code, vous utilisez l'action`UpdateApplication` de l'interface AWS CLI.

> **Note**
>
> Pour charger une nouvelle version du code de l'application portant le même nom de fichier, vous devez spécifier la nouvelle version de l'objet. Pour de plus amples informations sur l'utilisation des versions d'objet Amazon S3, veuillez consulter Activation et désactivation de la gestion des versions.

Pour utiliser l'interface AWS CLI, supprimez votre ancien package de code de votre compartiment Amazon S3, téléchargez la nouvelle version et appelez `UpdateApplication` en spécifiant le même compartiment Amazon S3 et le même nom d'objet, ainsi que la nouvelle version de l'objet. L'application redémarrera avec le nouveau package de code.

L'exemple de demande d'action `UpdateApplication` suivant recharge le code de l'application et redémarre l'application. Mettez à jour l'`CurrentApplicationVersionId` à la version actuelle de l'application. Vous pouvez vérifier la version actuelle de l'application à l'aide des actions `ListApplications` ou `DescribeApplication`. Mettez à jour le suffixe du nom du compartiment (<username>) avec le suffixe que vous avez choisi dans la section the section called "Création de deux flux de données Amazon Kinesis Data Streams".

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "aws-kinesis-analytics-java-apps-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Étape suivante

Etape 4 : nettoyage des ressources AWS

Etape 4 : nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Mise en route.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer vos flux de données Kinesis
- Supprimer votre objet et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Managed Service for Apache Flink, choisissez MyApplication.
3. Choisissez Configurer.
4. Dans la section Instantanés, choisissez Désactiver, puis sélectionnez Mettre à jour.
5. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer vos flux de données Kinesis

1. Ouvrez la console du service géré pour Apache Flink à l’adresse https://console.aws.amazon.com/flink
2. Dans le panneau Kinesis Data Streams, ExampleInputStreamsélectionnez.
3. Sur la ExampleInputStreampage, choisissez Supprimer Kinesis Stream, puis confirmez la suppression.
4. Sur la page Kinesis Streams, choisissez le ExampleOutputStream, choisissez Actions, choisissez Supprimer, puis confirmez la suppression.

Supprimer votre objet et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code -. <username>
3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.
8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Mise en route : Flink 1.6.2

Cette rubrique contient une version du didacticiel Mise en route (DataStream API) qui utilise Apache Flink 1.6.2.

Rubriques

- Composants d'un service géré pour Apache Flink
- Prérequis pour effectuer les exercices
- Étape 1 : configuration d'un compte AWS et création d'un administrateur

- Étape 2 : configuration de AWS Command Line Interface (AWS CLI)
- Étape 3 : création et exécution d'une application de service géré pour Apache Flink.
- Etape 4 : nettoyage des ressources AWS

Composants d'un service géré pour Apache Flink

Pour traiter les données, votre application de service géré pour Apache Flink utilise une application Java/Apache Maven ou Scala qui traite les entrées et produit des sorties à l'aide de l'exécution Apache Flink.

un service géré pour Apache Flink comporte les composants suivants :

- Propriétés d'exécution : vous pouvez utiliser les propriétés d'exécution pour configurer votre application sans recompiler le code de votre application.
- Source : l'application consomme des données en utilisant une source. Un connecteur source lit les données d'un flux de données Kinesis, d'un compartiment Amazon S3, etc. Pour plus d'informations, consultez Sources.
- Opérateurs : l'application traite les données à l'aide d'un ou de plusieurs opérateurs. Un opérateur peut transformer, enrichir ou agréger des données. Pour plus d'informations, consultez Opérateurs d'API DataStream.
- Récepteur : l'application produit des données vers des sources externes à l'aide de récepteurs. Un connecteur récepteur écrit des données dans un flux de données Kinesis, un flux Kinesis Data Firehose, un compartiment Amazon S3, etc. Pour plus d'informations, consultez Récepteurs.

Après avoir créé, compilé et empaqueté votre application, vous chargez le package de code dans un compartiment Amazon Simple Storage Service (Amazon S3). Vous créez ensuite une application de service géré pour Apache Flink. Vous transmettez l'emplacement du package de code, un flux de données Kinesis comme source de données de streaming et généralement un emplacement de streaming ou de fichier qui reçoit les données traitées par l'application.

Prérequis pour effectuer les exercices

Pour exécuter la procédure indiquée dans ce guide, vous devez disposer des éléments suivants :

- Kit de développement Java (JDK) version 8. Définissez la variable d'environnement `JAVA_HOME` pour qu'elle pointe vers l'emplacement d'installation de votre JDK.

- Nous vous recommandons d'utiliser un environnement de développement (par exemple Eclipse Java Neon ou IntelliJ Idea) pour développer et compiler votre application.
- Client Git. Installez le client Git si vous ne l'avez pas déjà fait.
- Apache Maven Compiler Plugin. Maven doit être installé dans votre chemin de travail. Pour tester votre installation Apache Maven, saisissez les informations suivantes :

```
$ mvn -version
```

Pour démarrer, accédez à Étape 1 : configuration d'un compte AWS et création d'un administrateur.

Étape 1 : configuration d'un compte AWS et création d'un administrateur

S’inscrire à un Compte AWS

Si vous n’avez pas de compte Compte AWS, procédez comme suit pour en créer un.

Pour s’inscrire à un Compte AWS

1. Ouvrez https://portal.aws.amazon.com/billing/signup.
2. Suivez les instructions en ligne.

   Dans le cadre de la procédure d’inscription, vous recevrez un appel téléphonique et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

   Lorsque vous souscrivez à un Compte AWS, un Utilisateur racine d'un compte AWS est créé. Par défaut, seul l’utilisateur root a accès à l’ensemble des Services AWS et des ressources de ce compte. La meilleure pratique de sécurité consiste à attribuer un accès administratif à un utilisateur administratif, et à uniquement utiliser l’utilisateur root pour effectuer tâches nécessitant un accès utilisateur root.

AWS vous envoie un e-mail de confirmation lorsque le processus d’inscription est terminé. Vous pouvez afficher l’activité en cours de votre compte et gérer votre compte à tout moment en accédant à https://aws.amazon.com/ et en cliquant sur Mon compte.

Création d'un utilisateur administratif

Après vous être inscrit à un Compte AWS, sécurisez votre Utilisateur racine d'un compte AWS, activez AWS IAM Identity Center, puis créez un utilisateur administratif afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisation de votre Utilisateur racine d'un compte AWS

1. Connectez-vous à la AWS Management Console en tant que propriétaire du compte en sélectionnant Root user (utilisateur root) et en saisissant l'adresse e-mail de Compte AWS. Sur la page suivante, saisissez votre mot de passe.

   Pour obtenir de l'aide pour vous connecter en utilisant l'utilisateur root, consultez Connexion en tant qu'utilisateur root dans le Guide de l'utilisateur Connexion à AWS.
2. Activez l'authentification multifactorielle (MFA) pour votre utilisateur root.

   Pour obtenir des instructions, consultez Activation d'un dispositif MFA virtuel pour l'utilisateur root de votre Compte AWS (console) dans le Guide de l'utilisateur IAM.

Création d'un utilisateur administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Configuration d'AWS IAM Identity Center dans le guide de l'utilisateur AWS IAM Identity Center.
2. Dans IAM Identity Center, octroyez un accès administratif à un utilisateur administratif.

   Pour profiter d'un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, consultez Configuration de l'accès utilisateur avec le répertoire Répertoire IAM Identity Center par défaut dans le Guide de l'utilisateur AWS IAM Identity Center.

Connexion en tant qu'utilisateur administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l'URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l'utilisateur IAM Identity Center.

  Pour obtenir de l'aide pour vous connecter à l'aide d'un utilisateur IAM Identity Center, consultez Connexion au portail d'accès AWS dans le Guide de l'utilisateur Connexion à AWS.

Octroi d’un accès par programmation

Les utilisateurs ont besoin d'un accès programmatique s'ils souhaitent interagir avec AWS en dehors de la AWS Management Console. La manière d’octroyer un accès par programmation dépend du type d’utilisateur qui accède à AWS.

Pour accorder aux utilisateurs un accès programmatique, choisissez l’une des options suivantes.

Quel utilisateur a besoin d’un accès programmatique ?	Pour	Par
Identité de la main-d’œuvre (Utilisateurs gérés dans IAM Identity Center)	Utilisez des informations d’identification temporaires pour signer des demandes par programmation destinées à l’AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de l’interface que vous souhaitez utiliser. • Pour l’AWS CLI, veuillez consulter la rubrique Configuration de l’AWS CLI pour l’utilisation d’AWS IAM Identity Center dans le Guide de l’utilisateur AWS Command Line Interface. • Pour les kits AWS SDK, les outils et les API AWS, consultez Authentification IAM Identity Center dans le Guide de référence des kits SDK et des outils AWS.
IAM	Utilisez des informations d’identification temporaires pour signer des demandes par programmation destinées à l’AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de la section Utilisation d’informations d’identification temporaires avec des ressources AWS dans le Guide de l’utilisateur IAM.

Quel utilisateur a besoin d'un accès programmatique ?	Pour	Par
IAM	(Non recommandé) Utilisez des informations d'identification à long terme pour signer des demandes par programmation destinées à l'AWS CLI, aux kits SDK AWS ou aux API AWS.	Suivez les instructions de l'interface que vous souhaitez utiliser. • Pour l'AWS CLI, veuillez consulter la rubrique Authentification à l'aide des informations d'identification d'utilisateur IAM dans le Guide de l'utilisateur AWS Command Line Interface. • Pour les kits SDK et les outils AWS, veuillez consulter la rubrique Authentification à l'aide d'informations d'identification à long terme dans le Guide de référence des kits SDK et des outils AWS. • Pour les API AWS, veuillez consulter la rubrique Gestion des clés d'accès pour les utilisateurs IAM dans le Guide de l'utilisateur IAM.

Étape 2 : configuration de AWS Command Line Interface (AWS CLI)

Dans cette étape, vous allez télécharger et configurer l'interface AWS CLI pour l'utiliser avec le service géré pour Apache Flink.

> **Note**
>
> Les exercices de mise en route de ce guide supposent que vous utilisez les informations d'identification d'administrateur (`adminuser`) de votre compte pour effectuer les opérations.

> **Note**
>
> Si l'interface AWS CLI est déjà installée, vous pouvez avoir besoin de procéder à une mise à niveau pour obtenir les dernières fonctionnalités. Pour plus d'informations, consultez Installation d'AWS Command Line Interface dans le Guide de l'utilisateur AWS Command Line Interface. Pour vérifier la version de l'interface AWS CLI, exécutez la commande suivante :
>
> ```
> aws --version
> ```
>
> Les exercices de ce didacticiel nécessitent la version AWS CLI suivante ou une version ultérieure :
>
> ```
> aws-cli/1.16.63
> ```

Pour configurer l’interface AWS CLI

1. Téléchargez et configurez l’interface AWS CLI. Pour obtenir des instructions, consultez les rubriques suivantes dans le Guide de l'utilisateur de l'interface AWS Command Line Interface :
   - Installation de AWS Command Line Interface
   - Configuration de l'interface AWS CLI (français non garanti)
2. Ajoutez un profil désigné pour l'utilisateur administrateur dans le fichier `config` de l’interface AWS CLI. Vous utiliserez ce profil lorsque vous exécuterez les commandes AWS CLI. Pour plus d'informations sur les profils nommés, consultez la rubrique Profils nommés dans le Guide de l'utilisateur AWS Command Line Interface.

   ```
   [profile adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   ```

```
region = aws-region
```

Pour connaître la liste des régions AWS disponibles, consultez Régions et points de terminaison du manuel Référence générale d'Amazon Web Services.

> **Note**
>
> Les exemples de code et de commandes présentés dans ce didacticiel utilisent la région USA Ouest (Oregon). Pour utiliser une autre région, remplacez la région dans le code et les commandes de ce didacticiel par la région que vous souhaitez utiliser.

3. Vérifiez la configuration en saisissant la commande d'aide suivante à l'invite de commande :

```
aws help
```

Après avoir configuré un AWS compteAWS CLI, vous pouvez passer à l'exercice suivant, dans lequel vous configurez un exemple d'application et testez la end-to-end configuration.

Étape suivante

Étape 3 : création et exécution d'une application de service géré pour Apache Flink.

Étape 3 : création et exécution d'une application de service géré pour Apache Flink.

Dans cet exercice, vous allez créer une application de service géré pour Apache Flink avec des flux de données comme source et comme récepteur.

Cette section contient les étapes suivantes :

- Création de deux flux de données Amazon Kinesis Data Streams
- Écriture d'exemples d'enregistrements dans le flux d'entrée
- Téléchargement et examen du code Java Apache Flink
- Compilation du code d'application
- Chargement du code Java Apache Flink
- Création et exécution de l’application de service géré pour Apache Flink

Création de deux flux de données Amazon Kinesis Data Streams

Avant de créer une application de service géré pour Apache Flink dans le cadre de cet exercice, commencez par créer deux flux de données Kinesis (`ExampleInputStream` et `ExampleOutputStream`). Votre application utilise ces flux pour les flux source et de destination de l'application.

Vous pouvez créer ces flux à l'aide de la console Amazon Kinesis ou de la commande AWS CLI suivante. Pour obtenir des instructions sur la console, consultez Création et mise à jour de flux de données dans le Guide du développeur Amazon Kinesis Data Streams.

Pour créer les flux de données (AWS CLI)

1. Pour créer le premier flux (`ExampleInputStream`) utilisez la commande AWS CLI `create-stream` Amazon Kinesis suivante.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Pour créer le second flux utilisé par l'application pour écrire la sortie, exécutez la même commande en remplaçant le nom du flux par `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Écriture d'exemples d'enregistrements dans le flux d'entrée

Dans cette section, vous utilisez un script Python pour écrire les exemples d'enregistrements dans le flux pour que l'application les traite.

> **Note**
>
> Cette section nécessite le kit AWS SDK for Python (Boto).

1. Créez un fichier nommé `stock.py` avec le contenu suivant :

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"


def get_data():
    return {
        "EVENT_TIME": datetime.datetime.now().isoformat(),
        "TICKER": random.choice(["AAPL", "AMZN", "MSFT", "INTC", "TBV"]),
        "PRICE": round(random.random() * 100, 2),
    }


def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name, Data=json.dumps(data),
 PartitionKey="partitionkey"
        )


if __name__ == "__main__":
    generate(STREAM_NAME, boto3.client("kinesis"))
```

2. Plus loin dans ce didacticiel, vous exécutez le script `stock.py` pour envoyer des données à l'application.

```
$ python stock.py
```

Téléchargement et examen du code Java Apache Flink

Le code de l'application Java pour cet exemple est disponible sur GitHub. Pour télécharger le code d'application, procédez comme suit :

1. Cloner le référentiel distant à l'aide de la commande suivante :

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Accédez au répertoire `amazon-kinesis-data-analytics-java-examples/GettingStarted_1_6`.

Notez les informations suivantes à propos du code d'application :

- Un fichier de modèle d'objet du projet (pom.xml) contient des informations sur la configuration et les dépendances de l'application, y compris les bibliothèques du service géré pour Apache Flink.
- Le fichier `BasicStreamingJob.java` contient la méthode `main` qui définit la fonctionnalité de l'application.
- L'application utilise une source Kinesis pour lire à partir du flux source. L'extrait de code suivant crée la source Kinesis :

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
                new SimpleStringSchema(), inputProperties));
```

- Votre application crée les connecteurs source et récepteur pour accéder aux ressources externes à l'aide d'un objet `StreamExecutionEnvironment`.
- L'application crée les connecteurs source et récepteur à l'aide de propriétés statiques. Pour utiliser les propriétés de l'application dynamique, utilisez les méthodes `createSourceFromApplicationProperties` et `createSinkFromApplicationProperties` pour créer les connecteurs. Ces méthodes lisent les propriétés de l'application pour configurer les connecteurs.

  Pour de plus amples informations sur les propriétés d'exécution, veuillez consulter Propriétés d'exécution.

Compilation du code d'application

Dans cette section, vous allez utiliser le compilateur Apache Maven pour créer le code Java pour l'application. Pour de plus amples informations sur l'installation d'Apache Maven et sur le kit de développement Java (JDK), veuillez consulter Prérequis pour effectuer les exercices.

> **Note**
>
> Afin d'utiliser le connecteur Kinesis avec les versions d'Apache Flink antérieures à la version 1.11, vous devez télécharger le code source pour le connecteur et le construire comme décrit dans la documentation Apache Flink.

Pour compiler le code d'application

1. Pour utiliser votre code d'application, vous le compilez et l'intégrez dans un fichier JAR. Vous pouvez compiler et intégrer votre code de deux manières :

   - À l'aide de l'outil de ligne de commande Maven. Créez votre fichier JAR en exécutant la commande suivante dans le répertoire qui contient le fichier `pom.xml` :

     ```
     mvn package
     ```

     > **Note**
     >
     > Le paramètre -DFLink.version n'est pas obligatoire pour l'environnement d'exécution du service géré pour Apache Flink version 1.0.1 ; il n'est requis que pour les versions 1.1.0 et ultérieures. Pour plus d'informations, consultez the section called "Spécification de la version d'Apache Flink de votre application".

   - À l'aide de votre environnement de développement. Consultez la documentation de votre environnement de développement pour plus de détails.

   Vous pouvez charger votre package en tant que fichier JAR, ou compresser le package et le charger en tant que fichier ZIP. Si vous créez votre application à l'aide de l'interface AWS CLI, vous spécifiez le type de contenu de votre code (JAR ou ZIP).

2. En cas d'erreur lors de la compilation, vérifiez que votre variable d'environnement `JAVA_HOME` est correctement définie.

Si la compilation de l'application aboutit, le fichier suivant est créé :

```
target/aws-kinesis-analytics-java-apps-1.0.jar
```

Chargement du code Java Apache Flink

Dans cette section, vous allez créer un compartiment Amazon Simple Storage Service (Amazon S3) et charger votre code d'application.

Pour charger le code d'application

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez Créer un compartiment.
3. Saisissez **`ka-app-code-<username>`** dans le champ Nom du compartiment. Ajoutez un suffixe au nom du compartiment, par exemple votre nom d'utilisateur, pour qu'il soit unique. Choisissez Suivant.
4. À l'étape Configurer les options, conservez les paramètres, puis choisissez Suivant.
5. À l'étape Définir des autorisations, conservez les paramètres, puis choisissez Suivant.
6. Choisissez Créer un compartiment.
7. Dans la console Amazon S3, choisissez le <username>compartiment ka-app-code-, puis Upload.
8. À l'étape Sélectionner les fichiers, choisissez Ajouter des fichiers. Accédez au fichier `aws-kinesis-analytics-java-apps-1.0.jar` que vous avez créé à l'étape précédente. Choisissez Suivant.
9. À l'étape Définir des autorisations, conservez les paramètres. Choisissez Suivant.
10. À l'étape Définir les propriétés, conservez les paramètres. Sélectionnez Charger.

Votre code d'application est désormais stocké dans un compartiment Amazon S3 auquel votre application peut accéder.

Création et exécution de l’application de service géré pour Apache Flink

Vous pouvez créer et exécuter une application de service géré pour Apache Flink à l'aide de la console ou de l’interface AWS CLI.

> **Note**
>
> Lorsque vous créez l'application à l'aide de la console, vos ressources AWS Identity and Access Management (IAM) et Amazon CloudWatch Logs sont créées pour vous. Lorsque vous créez l'application à l'aide de l'interface AWS CLI, vous créez ces ressources séparément.

Rubriques

- Création et exécution de l'application (console)
- Création et exécution de l'application (AWS CLI)

Création et exécution de l'application (console)

Suivez ces étapes pour créer, configurer, mettre à jour et exécuter l'application à l'aide de la console.

Pour créer l'application

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le tableau de bord du service géré pour Apache Flink, choisissez Créer une application d'analyse.
3. Sur la page Service géré pour Apache Flink - Créer une application, fournissez les détails de l'application comme suit :
   - Pour Nom de l'application, saisissez **`MyApplication`**.
   - Pour Description, saisissez **`My java test app`**.
   - Pour Exécution, choisissez Apache Flink.

     > **Note**
     >
     > Le service géré pour Apache Flink utilise Apache Flink version 1.8.2 ou 1.6.2.

   - Modifiez le menu déroulant de la version sur Apache Flink 1.6.
4. Pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM) **`kinesis-analytics-MyApplication-us-west-2`**.
5. Choisissez Créer une application.

> **Note**
>
> Lorsque vous créez une application de service géré pour Apache Flink à l'aide de la console, vous avez la possibilité de créer un rôle et une politique IAM pour votre application. Votre application utilise ce rôle et cette politique pour accéder à ses ressources dépendantes. Ces

ressources IAM sont nommées en utilisant le nom de votre application et la région, comme suit :

- Stratégie : `kinesis-analytics-service-MyApplication-us-west-2`
- Rôle : `kinesisanalytics-MyApplication-us-west-2`

Modification de la stratégie IAM

Modifiez la politique IAM pour ajouter des autorisations afin d'accéder aux flux de données Kinesis.

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Choisissez Stratégies. Choisissez la stratégie **kinesis-analytics-service-MyApplication-us-west-2** créée pour vous par la console dans la section précédente.
3. Sur la page Récapitulatif, choisissez Modifier la stratégie. Sélectionnez l'onglet JSON.
4. Ajoutez la section mise en surbrillance dans l'exemple de stratégie suivant à la stratégie. Remplacez l'exemple d'ID de compte (*012345678901*) par votre ID de compte.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ReadCode",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": [
                "arn:aws:s3:::ka-app-code-username/java-getting-started-1.0.jar"
            ]
        },
        {
            "Sid": "DescribeLogGroups",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:*"
            ]
```

```
        },
        {
            "Sid": "DescribeLogStreams",
            "Effect": "Allow",
            "Action": [
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
            ]
        },
        {
            "Sid": "PutLogEvents",
            "Effect": "Allow",
            "Action": [
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Configuration de l'application

1. Sur la MyApplicationpage, choisissez Configurer.

2. Sur la page Configurer l'application, indiquez l'emplacement du code:
   - Pour le compartiment Amazon S3, saisissez **ka-app-code-*<username>***.
   - Pour le chemin de l'objet Amazon S3, saisissez **java-getting-started-1.0.jar**.
3. Sous Accéder aux ressources de l'application, pour Autorisations d'accès, choisissez Créer/mettre à jour un rôle IAM **kinesis-analytics-MyApplication-us-west-2**.
4. Entrez les valeurs et propriétés d'application suivantes :

ID du groupe	Clé	Valeur
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

5. Sous Surveillance, assurez-vous que Surveillance du niveau des métriques est défini sur Application.
6. Pour la CloudWatch journalisation, cochez la case Activer.
7. Choisissez Mettre à jour.

> **Note**
>
> Lorsque vous choisissez d'activer la CloudWatch journalisation Amazon, Managed Service for Apache Flink crée un groupe de journaux et un flux de journaux pour vous. Les noms de ces ressources sont les suivants :
>
> - Groupe de journaux : `/aws/kinesis-analytics/MyApplication`
> - Flux de journaux : `kinesis-analytics-log-stream`

Exécution de l'application

1. Sur la MyApplicationpage, choisissez Exécuter. Confirmez l'action.

2. Lorsque l'application est en cours d'exécution, actualisez la page. La console affiche le graphique de l'application.

Arrêt de l'application

Sur la MyApplicationpage, choisissez Stop. Confirmez l'action.

Mise à jour de l'application

À l'aide de la console, vous pouvez mettre à jour les paramètres d'application tels que les paramètres de surveillance, les propriétés d'application et l'emplacement ou le nom du fichier JAR de l'application. Vous pouvez également recharger le fichier JAR de l'application à partir du compartiment Amazon S3 si vous avez besoin de mettre à jour le code de l'application.

Sur la MyApplicationpage, choisissez Configurer. Mettez à jour les paramètres de l'application, puis choisissez Mettre à jour.

Création et exécution de l'application (AWS CLI)

Dans cette section, vous allez utiliser l'interface AWS CLI pour créer et exécuter l'application de service géré pour Apache Flink. Le service géré pour Apache Flink utilise la commande AWS CLI `kinesisanalyticsv2` pour créer et interagir avec les applications de service géré pour Apache Flink.

Créer une stratégie d'autorisations

Vous commencez par créer une stratégie d'autorisations avec deux instructions : une qui accorde des autorisations pour l'action `read` sur le flux source et une autre qui accorde des autorisations pour les actions `write` sur le flux récepteur. Vous attachez ensuite la politique à un rôle IAM (que vous allez créer dans la section suivante). Ainsi, lorsque le service géré pour Apache Flink assume le rôle, le service dispose des autorisations nécessaires pour lire à partir du flux source et écrire dans le flux récepteur.

Utilisez le code suivant pour créer la stratégie d'autorisations `AKReadSourceStreamWriteSinkStream`. Remplacez *`username`* par le nom d'utilisateur que vous avez utilisé pour créer le compartiment Amazon S3 pour stocker le code d'application. Remplacez l'ID de compte dans Amazon Resource Names (ARN) (*`012345678901`*) par votre ID de compte.

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:GetObjectVersion"
            ],
            "Resource": ["arn:aws:s3:::ka-app-code-username",
                "arn:aws:s3:::ka-app-code-username/*"
            ]
        },
        {
            "Sid": "ReadInputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
        },
        {
            "Sid": "WriteOutputStream",
            "Effect": "Allow",
            "Action": "kinesis:*",
            "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
        }
    ]
}
```

Pour step-by-step obtenir des instructions sur la création d'une politique d'autorisations, voir Tutoriel : créer et joindre votre première politique gérée par le client dans le guide de l'utilisateur IAM.

> **Note**
>
> Pour accéder à d'autres services Amazon, vous pouvez utiliser le AWS SDK for Java. Le service géré pour Apache Flink définit automatiquement les informations d'identification requises par le kit SDK en fonction du rôle IAM d'exécution du service associé à votre application. Aucune étape supplémentaire n'est nécessaire.

Créer un rôle IAM

Dans cette section, vous créez un rôle IAM que l'application de service géré pour Apache Flink peut assumer pour lire un flux source et écrire dans le flux récepteur.

Le service géré pour Apache Flink ne peut pas accéder à votre flux sans autorisation. Vous utilisez un rôle IAM pour accorder ces autorisations. Deux politiques sont attachées à chaque rôle IAM. La politique d'approbation accorde au service géré pour Apache Flink l'autorisation d'assumer le rôle, et la politique d'autorisation détermine ce que le service géré pour Apache Flink peut faire après avoir assumé le rôle.

Vous attachez la stratégie d'autorisations que vous avez créée dans la section précédente à ce rôle.

Pour créer un rôle IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans le volet de navigation, choisissez Rôles, puis Créer un rôle.
3. Sous Sélectionner le type d'identité approuvée, choisissez Service AWS. Sous Choisir le service qui utilisera ce rôle, choisissez EC2. Sous Sélectionner votre cas d'utilisation, choisissez Kinesis Analytics.

   Sélectionnez Next: Permissions (Étape suivante : autorisations).
4. Dans la page Attacher des stratégies d'autorisations, choisissez Suivant : vérification. Vous attachez des stratégies d'autorisations après avoir créé le rôle.
5. Sur la page Créer un rôle, saisissez **`MF-stream-rw-role`** pour le Nom du rôle. Sélectionnez Créer un rôle.

   Vous venez de créer un nouveau rôle IAM appelé `MF-stream-rw-role`. Ensuite, vous mettez à jour les stratégies d'approbation et d'autorisation pour le rôle.
6. Attachez la politique d'autorisation au rôle.

> **Note**
>
> Dans le cadre de cet exercice, le service géré pour Apache Flink assume ce rôle à la fois pour la lecture des données à partir d'un flux de données Kinesis (source) et pour l'écriture des résultats dans un autre flux de données Kinesis. Vous attachez donc la stratégie que vous avez créée à l'étape précédente, the section called “Créer une stratégie d'autorisations”.

a. Sur la page Récapitulatif, choisissez l'onglet Autorisations.

b. Choisissez Attacher des stratégies.

c. Dans la zone de recherche, saisissez **`AKReadSourceStreamWriteSinkStream`** (la stratégie que vous avez créée dans la section précédente).

d. Choisissez la ReadSourceStreamWriteSinkStream politique AK, puis choisissez Attach policy.

Vous avez maintenant créé le rôle d'exécution de service que votre application utilise pour accéder aux ressources. Notez l'ARN du nouveau rôle.

Pour step-by-step obtenir des instructions sur la création d'un rôle, consultez la section Création d'un rôle IAM (console) dans le guide de l'utilisateur IAM.

Création de l’application de service géré pour Apache Flink

1. Copiez le code JSON suivant dans un fichier nommé `create_request.json`. Remplacez l'exemple d'ARN du rôle par l'ARN du rôle que vous avez créé précédemment. Remplacez le suffixe de l'ARN du compartiment (*`username`*) par le suffixe que vous avez choisi dans la section précédente. Remplacez l'exemple d'ID de compte (*`012345678901`*) dans le rôle d'exécution de service par votre ID de compte.

```
{
    "ApplicationName": "test",
    "ApplicationDescription": "my java test app",
    "RuntimeEnvironment": "FLINK-1_6",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
    "ApplicationConfiguration": {
        "ApplicationCodeConfiguration": {
            "CodeContent": {
                "S3ContentLocation": {
                    "BucketARN": "arn:aws:s3:::ka-app-code-username",
                    "FileKey": "java-getting-started-1.0.jar"
                }
            },
            "CodeContentType": "ZIPFILE"
        },
        "EnvironmentProperties":  {
         "PropertyGroups": [
```

```
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
   }
}
```

2. Exécutez l'action [CreateApplication](#) avec la demande précédente pour créer l'application :

```
aws kinesisanalyticsv2 create-application --cli-input-json file://
create_request.json
```

L'application est maintenant créée. Vous démarrez l'application dans l'étape suivante.

Démarrage de l'application

Dans cette section, vous utilisez l'action [StartApplication](#) pour démarrer l'application.

Pour démarrer l'application

1. Copiez le code JSON suivant dans un fichier nommé `start_request.json`.

```
{
    "ApplicationName": "test",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
```

```
}
```

2. Exécutez l'action StartApplication avec la demande précédente pour démarrer l'application :

```
aws kinesisanalyticsv2 start-application --cli-input-json file://start_request.json
```

L'application est maintenant en cours d'exécution. Vous pouvez consulter les métriques du service géré pour Apache Flink sur la CloudWatch console Amazon pour vérifier que l'application fonctionne.

Arrêt de l'application

Dans cette section, vous allez utiliser l'action StopApplication pour arrêter l'application.

Pour arrêter l'application

1. Copiez le code JSON suivant dans un fichier nommé `stop_request.json`.

```
{
   "ApplicationName": "test"
}
```

2. Exécutez l'action StopApplication avec la demande suivante pour arrêter l'application :

```
aws kinesisanalyticsv2 stop-application --cli-input-json file://stop_request.json
```

L'application est maintenant arrêtée.

Ajouter une option de CloudWatch journalisation

Vous pouvez utiliser le AWS CLI pour ajouter un flux de CloudWatch journal Amazon à votre application. Pour plus d'informations sur l'utilisation de CloudWatch Logs avec votre application, consultezthe section called “Configuration de la journalisation”.

Mettre à jour des propriétés d'environnement

Dans cette section, vous utilisez l'action UpdateApplication pour modifier les propriétés d'environnement de l'application sans recompiler le code de l'application. Dans cet exemple, vous modifiez la région des flux source et de destination.

Pour mettre à jour des propriétés d'environnement pour l'application

1. Copiez le code JSON suivant dans un fichier nommé `update_properties_request.json`.

```
{"ApplicationName": "test",
   "CurrentApplicationVersionId": 1,
   "ApplicationConfigurationUpdate": {
      "EnvironmentPropertyUpdates": {
         "PropertyGroups": [
            {
               "PropertyGroupId": "ProducerConfigProperties",
               "PropertyMap" : {
                    "flink.stream.initpos" : "LATEST",
                    "aws.region" : "us-west-2",
                    "AggregationEnabled" : "false"
               }
            },
            {
               "PropertyGroupId": "ConsumerConfigProperties",
               "PropertyMap" : {
                    "aws.region" : "us-west-2"
               }
            }
         ]
      }
   }
}
```

2. Exécutez l'action [UpdateApplication](#) avec la demande précédente pour mettre à jour les propriétés de l'environnement :

```
aws kinesisanalyticsv2 update-application --cli-input-json file://
update_properties_request.json
```

Mise à jour du code de l’application

Lorsque vous devez mettre à jour le code de votre application avec une nouvelle version de votre package de code, vous utilisez l'action[UpdateApplication](#) de l’interface AWS CLI.

Pour utiliser l’interface AWS CLI, supprimez votre ancien package de code de votre compartiment Amazon S3, téléchargez la nouvelle version et appelez `UpdateApplication` en spécifiant le

même compartiment Amazon S3 et le même nom d'objet. L'application redémarrera avec le nouveau package de code.

L'exemple de demande d'action `UpdateApplication` suivant recharge le code de l'application et redémarre l'application. Mettez à jour l'`CurrentApplicationVersionId` à la version actuelle de l'application. Vous pouvez vérifier la version actuelle de l'application à l'aide des actions `ListApplications` ou `DescribeApplication`. Mettez à jour le suffixe du nom du compartiment (<username>) avec le suffixe que vous avez choisi dans la section the section called “Création de deux flux de données Amazon Kinesis Data Streams”.

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "java-getting-started-1.0.jar"
                }
            }
        }
    }
}
```

Etape 4 : nettoyage des ressources AWS

Cette section inclut les procédures de nettoyage des ressources AWScréées dans le didacticiel Mise en route.

Cette rubrique contient les sections suivantes :

- Suppression de votre application de service géré pour Apache Flink
- Supprimer vos flux de données Kinesis
- Supprimer votre objet et votre compartiment Amazon S3
- Supprimer vos ressources IAM
- Supprimer vos CloudWatch ressources

Suppression de votre application de service géré pour Apache Flink

1. Ouvrez la console Kinesis à l'adresse https://console.aws.amazon.com/kinesis.
2. Dans le panneau Managed Service for Apache Flink, choisissez MyApplication.
3. Choisissez Configurer.
4. Dans la section Instantanés, choisissez Désactiver, puis sélectionnez Mettre à jour.
5. Sur la page de l'application, choisissez Supprimer, puis confirmez la suppression.

Supprimer vos flux de données Kinesis

1. Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
2. Dans le panneau Kinesis Data Streams, ExampleInputStreamsélectionnez.
3. Sur la ExampleInputStreampage, choisissez Supprimer Kinesis Stream, puis confirmez la suppression.
4. Sur la page Kinesis Streams, choisissez le ExampleOutputStream, choisissez Actions, choisissez Supprimer, puis confirmez la suppression.

Supprimer votre objet et votre compartiment Amazon S3

1. Ouvrez la console Amazon S3 sur https://console.aws.amazon.com/s3/.
2. Choisissez le compartiment ka-app-code -. <username>
3. Choisissez Supprimer, puis saisissez le nombre du compartiment pour confirmer la suppression.

Supprimer vos ressources IAM

1. Ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/.
2. Dans la barre de navigation, choisissez Stratégies.
3. Dans le contrôle du filtre, saisissez kinesis.
4. Choisissez la politique kinesis-analytics-service- MyApplication -us-west-2.
5. Choisissez Actions de stratégie, puis Supprimer.
6. Dans la barre de navigation, choisissez Rôles.
7. Choisissez le rôle kinesis-analytics- MyApplication -us-west-2.

8. Choisissez Supprimer le rôle, puis confirmez la suppression.

Supprimer vos CloudWatch ressources

1. Ouvrez la CloudWatch console à l'adresse https://console.aws.amazon.com/cloudwatch/.
2. Dans la barre de navigation, choisissez Journaux.
3. Choisissez le groupe de journaux MyApplication/aws/kinesis-analytics/.
4. Choisissez Supprimer le groupe de journaux, puis confirmez la suppression.

Paramètres Apache Flink

Le service géré pour Apache Flink est une implémentation de l'environnement Apache Flink. Le service géré pour Apache Flink utilise les valeurs par défaut décrites dans cette section. Certaines de ces valeurs peuvent être définies par les applications de service géré pour Apache Flink dans le code, tandis que d'autres ne peuvent pas être modifiées.

Cette rubrique contient les sections suivantes :

- Configuration d'Apache Flink
- Backend d'État
- Point de contrôle
- Point de sauvegarde
- Tailles des tas
- Dégonflement de la mémoire tampon
- Propriétés de configuration Flink modifiables
- Affichage des propriétés Flink configurées

Configuration d'Apache Flink

Le service géré pour Apache Flink fournit une configuration Flink par défaut composée de valeurs recommandées par Apache Flink pour la plupart des propriétés et de quelques-unes basées sur des profils d'application courants. Pour plus d'informations sur la configuration de Flink, consultez Configuration. La configuration par défaut fournie par le service fonctionne pour la plupart des applications. Toutefois, si vous devez modifier les propriétés de configuration de Flink pour améliorer les performances de certaines applications présentant un parallélisme élevé, une utilisation élevée de la mémoire et de l'état, ou si vous souhaitez activer de nouvelles fonctionnalités de débogage dans Apache Flink, vous pouvez modifier certaines propriétés en demandant un dossier de support. Pour plus d'informations, consultez Centre de support AWS. Vous pouvez vérifier la configuration actuelle de votre application à l'aide du tableau de bord Apache Flink.

Backend d'État

Le service géré pour Apache Flink stocke les données transitoires dans un backend d'état. Le service géré pour Apache Flink utilise le RocksDBStateBackend. L'appel `setStateBackend` pour définir un backend différent n'a aucun effet.

Nous activons les fonctionnalités suivantes sur le backend d'état :

- Instantanés du backend d'état incrémentiel
- Instantanés du backend d'état asynchrone
- Restauration locale des points de contrôle

Dans le service géré pour Apache Flink, la configuration `state.backend.rocksdb.ttl.compaction.filter.enabled` est activée par défaut. À l'aide de ce filtre, vous pouvez mettre à jour le code de votre application pour activer la stratégie de nettoyage par compactage. Pour plus d'informations, voir State TTL in Flink 1.8.0 dans la documentation Apache Flink.

Pour plus d'informations sur les backends d'état, consultez la section State Backends dans la documentation d'Apache Flink.

Point de contrôle

Le service géré pour Apache Flink utilise une configuration de point de contrôle par défaut avec les valeurs suivantes. Certaines de ces valeurs peuvent être modifiées. Vous devez définir CheckpointConfiguration.ConfigurationType sur `CUSTOM` pour que le service géré pour Apache Flink utilise des valeurs de point de contrôle modifiées.

Paramètre	Peut être modifié ?	Comment ?	Valeur par défaut
CheckpointingEnabled	Adaptabilité	Créer une application Mettre à jour une application AWS CloudFormation	True
CheckpointInterval	Adaptabilité	Créer une application	60 000

Paramètre	Peut être modifié ?	Comment ?	Valeur par défaut
		Mettre à jour une application AWS CloudFormation	
MinPauseBetweenCheckpoints	Adaptabilité	Créer une application Mettre à jour une application AWS CloudFormation	5 000
Points de contrôle non alignés	Adaptabilité	Cas de support	False
Nombre de points de contrôle simultanés	Non modifiable	N/A	1
Mode de point de contrôle	Non modifiable	N/A	Exactement une fois
Politique de rétention des points de contrôle	Non modifiable	N/A	En échec
Délai d'expiration du point de contrôle	Non modifiable	N/A	60 minutes
Nombre maximum de points de contrôle conservés	Non modifiable	N/A	1
Stratégie de redémarrage	Non modifiable	N/A	Délai fixe, avec un nombre infini de tentatives toutes les 10 secondes.

Paramètre	Peut être modifié ?	Comment ?	Valeur par défaut
Emplacement du point de contrôle et du point de sauvegarde	Non modifiable	N/A	Nous stockons des données de point de contrôle et de point de sauvegarde durables dans un compartim ent S3 appartenant au service.
Seuil de mémoire du backend d'état	Non modifiable	N/A	1048576

Point de sauvegarde

Par défaut, lors de la restauration à partir d'un point de sauvegarde, l'opération de reprise essaie de faire correspondre l'ensemble de l'état du point de sauvegarde au programme avec lequel vous effectuez la restauration. Si vous supprimez un opérateur, par défaut, la restauration à partir d'un point de sauvegarde contenant des données correspondant à l'opérateur manquant échouera. Vous pouvez autoriser le succès de l'opération en définissant le paramètre AllowNonRestoredState de la configuration FlinkRunConfiguration de l'application sur `true`. Cela permettra à l'opération de reprise d'ignorer l'état qui ne peut pas être mis en correspondance avec le nouveau programme.

Pour plus d'informations, voir Allowing Non-Restored State dans la documentation Apache Flink.

Tailles des tas

Le service géré pour Apache Flink alloue 3 GiO de mémoire JVM à chaque KPU et réserve 1 GiO pour les allocations de code natif. Pour plus d'informations sur l'augmentation de la capacité de votre application, consultez the section called “Mise à l'échelle”.

Pour plus d'informations sur les tailles de tas JVM, consultez Configuration dans la documentation Apache Flink.

Dégonflement de la mémoire tampon

Le dégonflement de la mémoire tampon peut aider les applications soumises à une contre-pression élevée. Si les points de contrôle ou de sauvegarde de votre application échouent, il peut être utile d'activer cette fonctionnalité. Pour ce faire, demandez un dossier de support.

Pour plus d'informations, consultez The Buffer Debloating Mechanism dans la documentation Apache Flink.

Propriétés de configuration Flink modifiables

Vous trouverez ci-dessous les paramètres de configuration de Flink que vous pouvez modifier à l'aide d'un dossier de support. Vous pouvez modifier plusieurs propriétés à la fois et pour plusieurs applications en même temps en spécifiant le préfixe de l'application. S'il existe d'autres propriétés de configuration de Flink que vous souhaitez modifier en dehors de cette liste, veuillez spécifier la propriété exacte dans votre cas.

Tolérance aux pannes

`restart-strategy:`

`restart-strategy.fixed-delay.delay:`

Points de contrôle et backends d'État

`state.backend:`

`state.backend.fs.memory-threshold:`

`state.backend.incremental:`

Point de contrôle

`execution.checkpointing.unaligned:`

Métriques natives de RockSDB

Les métriques natives de RockSDB ne sont pas envoyées à CloudWatch. Une fois activées, ces métriques sont accessibles à partir du tableau de bord Flink ou de l'API REST de Flink avec des outils personnalisés.

Le service géré pour Apache Flink permet aux clients d'accéder à la dernière API REST de Flink (ou à la version prise en charge que vous utilisez) en mode lecture seule à l'aide de l'API CreateApplicationPresignedURL. Cette API est utilisée par le tableau de bord Flink, mais elle peut également être utilisée par des outils de surveillance personnalisés.

```
state.backend.rocksdb.compaction.style:

state.backend.rocksdb.memory.partitioned-index-filters:

state.backend.rocksdb.metrics.actual-delayed-write-rate:

state.backend.rocksdb.metrics.background-errors:

state.backend.rocksdb.metrics.block-cache-capacity:

state.backend.rocksdb.metrics.block-cache-pinned-usage:

state.backend.rocksdb.metrics.block-cache-usage:

state.backend.rocksdb.metrics.column-family-as-variable:

state.backend.rocksdb.metrics.compaction-pending:

state.backend.rocksdb.metrics.cur-size-active-mem-table:

state.backend.rocksdb.metrics.cur-size-all-mem-tables:

state.backend.rocksdb.metrics.estimate-live-data-size:

state.backend.rocksdb.metrics.estimate-num-keys:

state.backend.rocksdb.metrics.estimate-pending-compaction-bytes:

state.backend.rocksdb.metrics.estimate-table-readers-mem:

state.backend.rocksdb.metrics.is-write-stopped:

state.backend.rocksdb.metrics.mem-table-flush-pending:

state.backend.rocksdb.metrics.num-deletes-active-mem-table:

state.backend.rocksdb.metrics.num-deletes-imm-mem-tables:

state.backend.rocksdb.metrics.num-entries-active-mem-table:
```

```
state.backend.rocksdb.metrics.num-entries-imm-mem-tables:

state.backend.rocksdb.metrics.num-immutable-mem-table:

state.backend.rocksdb.metrics.num-live-versions:

state.backend.rocksdb.metrics.num-running-compactions:

state.backend.rocksdb.metrics.num-running-flushes:

state.backend.rocksdb.metrics.num-snapshots:

state.backend.rocksdb.metrics.size-all-mem-tables:

state.backend.rocksdb.thread.num:
```

Options avancées pour les backends d'état

```
state.storage.fs.memory-threshold:
```

Options complètes du gestionnaire de tâches

```
task.cancellation.timeout:

taskmanager.jvm-exit-on-oom:

taskmanager.numberOfTaskSlots:

taskmanager.slot.timeout:

taskmanager.network.memory.fraction:

taskmanager.network.memory.max:

taskmanager.network.request-backoff.initial:

taskmanager.network.request-backoff.max:

taskmanager.network.memory.buffer-debloat.enabled:

taskmanager.network.memory.buffer-debloat.period:

taskmanager.network.memory.buffer-debloat.samples:
```

taskmanager.network.memory.buffer-debloat.threshold-percentages:

Configuration de mémoire

taskmanager.memory.jvm-metaspace.size:

taskmanager.memory.jvm-overhead.fraction:

taskmanager.memory.jvm-overhead.max:

taskmanager.memory.managed.consumer-weights:

taskmanager.memory.managed.fraction:

taskmanager.memory.network.fraction:

taskmanager.memory.network.max:

taskmanager.memory.segment-size:

taskmanager.memory.task.off-heap.size:

RPC/Akka

akka.ask.timeout:

akka.client.timeout:

akka.framesize:

akka.lookup.timeout:

akka.tcp.timeout:

Client

client.timeout:

Options avancées du cluster

cluster.intercept-user-system-exit:

cluster.processes.halt-on-fatal-error:

Configurations du système de fichiers

`fs.s3.connection.maximum:`

`fs.s3a.connection.maximum:`

`fs.s3a.threads.max:`

`s3.upload.max.concurrent.uploads:`

Options avancées de tolérance aux pannes

`heartbeat.timeout:`

`jobmanager.execution.failover-strategy:`

Configuration de mémoire

`jobmanager.memory.heap.size:`

Métriques

`metrics.latency.interval:`

Options avancées pour le point de terminaison et le client REST

`rest.flamegraph.enabled:`

`rest.server.numThreads:`

Options de sécurité SSL avancées

`security.ssl.internal.handshake-timeout:`

Options de planification avancées

`slot.request.timeout:`

Options avancées pour l'interface utilisateur Web de Flink

`web.timeout:`

Affichage des propriétés Flink configurées

Vous pouvez consulter les propriétés Apache Flink que vous avez configurées vous-même ou dont vous avez demandé la modification par le biais d'un dossier de support via le tableau de bord Apache Flink et en suivant ces étapes :

1. Accédez au tableau de bord Flink
2. Choisissez Gestionnaire de tâches dans le volet de navigation de gauche.
3. Choisissez Configuration pour afficher la liste des propriétés de Flink.

Configuration du service géré pour Apache Flink pour accéder aux ressources dans un VPC Amazon

Vous pouvez configurer une application de service géré pour Apache Flink pour qu'elle se connecte aux sous-réseaux privés d'un Virtual Private Cloud (VPC) de votre compte. Utilisez Amazon Virtual Private Cloud (Amazon VPC) afin de créer un réseau privé pour des ressources telles que des bases de données, des instances de mémoire cache ou des services internes. Connectez votre application au VPC pour accéder à des ressources privées pendant l'exécution.

Cette rubrique contient les sections suivantes :

- Concepts Amazon VPC
- Autorisations d'application VPC
- Accès à Internet et aux services pour un service géré connecté à un VPC pour l'application Apache Flink
- API VPC pour le service géré pour Apache Flink
- Exemple : utilisation d'un VPC pour accéder aux données d'un cluster Amazon MSK

Concepts Amazon VPC

Amazon VPC est la couche réseau pour Amazon EC2. Si vous êtes nouveau sur Amazon EC2, veuillez consulter Qu'est-ce qu'Amazon EC2 ? dans le Guide de l’utilisateur Amazon EC2 pour les instances Linux pour obtenir une brève présentation.

Les concepts clés liés aux VPC sont les suivants :

- Un cloud privé virtuel (VPC) est un réseau virtuel dédié à votre compte AWS.
- Un sous-réseau est une plage d'adresses IP dans votre VPC.
- Une table de routage contient un ensemble de règles, appelées routes, qui permettent de déterminer où diriger le trafic réseau.
- Une passerelle Internet est un composant de VPC dimensionné horizontalement, redondant et hautement disponible qui permet la communication entre des instances dans votre VPC et sur Internet. Elle n'impose par conséquent aucun risque de disponibilité ni de contraintes de bande passante sur votre trafic réseau.

- Un point de terminaison de VPC permet une connexion privée entre votre VPC et les services AWS pris en charge ou les services de point de terminaison de VPC alimentés par PrivateLink sans nécessiter une passerelle Internet, un périphérique NAT, une connexion VPN ni une connexion AWS Direct Connect. Les instances de votre VPC ne requièrent pas d'adresses IP publiques pour communiquer avec les ressources du service. Le trafic entre votre VPC et les autres services ne quitte pas le réseau Amazon.

Pour plus d'informations sur le VPC Amazon, consultez le Guide de l'utilisateur Amazon Virtual Private Cloud.

Le service géré pour Apache Flink crée des interfaces réseau élastiques dans l'un des sous-réseaux fournis dans la configuration de votre VPC pour l'application. Le nombre d'interfaces réseau élastiques créées dans vos sous-réseaux VPC peut varier en fonction du parallélisme et du parallélisme par KPU de l'application. Pour en savoir plus sur l’autoscaling d’application, consultez Mise à l'échelle.

> **Note**
>
> Les configurations VPC ne sont pas prises en charge pour les applications SQL.

> **Note**
>
> Le service géré pour Apache Flink gère l'état du point de contrôle et de l’instantané pour les applications dotées d'une configuration VPC.

Autorisations d'application VPC

Cette section décrit les politiques d'autorisation dont votre application aura besoin pour fonctionner avec votre VPC. Pour plus d'informations sur l’utilisation de stratégies d'autorisations, consultez Gestion de l'identité et des accès dans le service géré Amazon pour Apache Flink.

La politique d'autorisation suivante accorde à votre application les autorisations nécessaires pour interagir avec un VPC. Pour utiliser cette stratégie d'autorisation, ajoutez-la au rôle d'exécution de votre application.

Stratégie d'autorisations pour accéder à un VPC Amazon

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VPCReadOnlyPermissions",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeVpcs",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeDhcpOptions"
      ],
      "Resource": "*"
    },
    {
      "Sid": "ENIReadWritePermissions",
      "Effect": "Allow",
      "Action": [
        "ec2:CreateNetworkInterface",
        "ec2:CreateNetworkInterfacePermission",
        "ec2:DescribeNetworkInterfaces",
        "ec2:DeleteNetworkInterface"
      ],
      "Resource": "*"
    }

    ]
}
```

> **Note**
>
> Lorsque vous spécifiez des ressources d'application à l'aide de la console (comme CloudWatch Logs ou Amazon VPC), la console modifie le rôle d'exécution de votre application pour autoriser l'accès à ces ressources. Vous ne devez modifier manuellement le rôle d'exécution de votre application que si vous créez votre application sans utiliser la console.

Accès à Internet et aux services pour un service géré connecté à un VPC pour l'application Apache Flink

PAr défaut, lorsque vous connectez une application de service géré pour Apache Flink à un VPC de votre compte, elle n'a pas accès à Internet, sauf si le VPC le lui fournit. Si l'application nécessite un accès à Internet, les conditions suivantes doivent être remplies :

- L'application de service géré pour Apache Flink ne doit être configurée qu'avec des sous-réseaux privés.
- Le VPC doit contenir une passerelle ou une instance NAT dans un sous-réseau public.
- Une route doit exister pour le trafic sortant depuis les sous-réseaux privés vers la passerelle NAT dans un sous-réseau public.

> **Note**
>
> Plusieurs services offrent des points de terminaison de VPC. Vous pouvez utiliser les points de terminaison d'un VPC pour vous connecter aux services Amazon à partir d'un VPC sans accès à Internet.

Le caractère public ou privé d'un sous-réseau dépend de sa table de routage. Chaque table de routage possède une route par défaut, qui détermine le saut suivant pour les paquets ayant une destination publique.

- Pour un sous-réseau privé : la route par défaut pointe vers une passerelle NAT (nat-...) ou une instance NAT (eni-...).
- Pour un sous-réseau public : la route par défaut pointe vers une passerelle Internet (igw-...).

Une fois que vous avez configuré votre VPC avec un sous-réseau public (avec un NAT) et un ou plusieurs sous-réseaux privés, procédez comme suit pour identifier vos sous-réseaux privés et publics :

- Dans le panneau de navigation de la console VPC, choisissez Sous-réseaux.
- Choisissez un sous-réseau, puis choisissez l'onglet Table de routage. Vérifiez la route par défaut :
  - Sous-réseau public : destination : 0.0.0.0/0, cible : igw-...

- Sous-réseau privé : destination : 0.0.0.0/0, cible : nat-... ou eni-...

Pour associer l'application de service géré pour Apache Flink à des sous-réseaux privés :

- Ouvrez la console du service géré pour Apache Flink à l'adresse https://console.aws.amazon.com/flink
- Sur la page Applications de service géré pour Apache Flink, choisissez votre application, puis sélectionnez Détails de l'application.
- Sur la page de votre application, choisissez Configurer.
- Dans la section Connectivité VPC, choisissez le VPC à associer à votre application. Choisissez les sous-réseaux et le groupe de sécurité associés à votre VPC que vous souhaitez que l'application utilise pour accéder aux ressources du VPC.
- Choisissez Mettre à jour.

Informations connexes

Création d'un VPC avec sous-réseaux publics et privés

Principes de base d'une passerelle NAT

API VPC pour le service géré pour Apache Flink

Utilisez le service géré suivant pour les opérations d'API Apache Flink afin de gérer les VPC de votre application. Pour des informations sur l'utilisation de l'API de service géré pour Apache Flink, consultez Exemple de code d'API.

CreateApplication

Utilisez l'action CreateApplication pour ajouter une configuration VPC à votre application lors de sa création.

L'exemple de code de demande suivant pour l'action `CreateApplication` inclut une configuration VPC lors de la création de l'application :

```
{
  "ApplicationName":"MyApplication",
  "ApplicationDescription":"My-Application-Description",
```

```
  "RuntimeEnvironment":"FLINK-1_15",
  "ServiceExecutionRole":"arn:aws:iam::123456789123:role/myrole",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration":{
      "CodeContent":{
        "S3ContentLocation":{
          "BucketARN":"arn:aws:s3:::mybucket",
          "FileKey":"myflink.jar",
          "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "CodeContentType":"ZIPFILE"
    },
      "FlinkApplicationConfiguration":{
      "ParallelismConfiguration":{
        "ConfigurationType":"CUSTOM",
        "Parallelism":2,
        "ParallelismPerKPU":1,
        "AutoScalingEnabled":true
      }
    },
  "VpcConfigurations": [
        {
           "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
           "SubnetIds": [ "subnet-0123456789abcdef0" ]
        }
      ]
  }
}
```

AddApplicationVpcConfiguration

Utilisez l'action AddApplicationVpcConfiguration pour ajouter une configuration VPC à votre application après sa création.

L'exemple de code de demande suivant pour l'action `AddApplicationVpcConfiguration` ajoute une configuration VPC à une application existante :

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfiguration": {
      "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
```

```
      "SubnetIds": [ "subnet-0123456789abcdef0" ]
   }
}
```

DeleteApplicationVpcConfiguration

Utilisez l'action DeleteApplicationVPCConfiguration pour supprimer une configuration VPC de votre application.

L'exemple de code de demande suivant pour l'action `AddApplicationVpcConfiguration` supprime une configuration VPC existante d'une application :

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfigurationId": "1.1"
}
```

UpdateApplication

Utilisez l'action UpdateApplication pour mettre à jour toutes les configurations VPC d'une application en une seule fois.

L'exemple de code de demande suivant pour l'action `UpdateApplication` met à jour toutes les configurations VPC d'une application :

```
{
   "ApplicationConfigurationUpdate": {
      "VpcConfigurationUpdates": [
         {
            "SecurityGroupIdUpdates": [ "sg-0123456789abcdef0" ],
            "SubnetIdUpdates": [ "subnet-0123456789abcdef0" ],
            "VpcConfigurationId": "2.1"
         }
      ]
   },
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9
}
```

Exemple : utilisation d'un VPC pour accéder aux données d'un cluster Amazon MSK

Pour un didacticiel complet sur l'accès aux données d'un cluster Amazon MSK dans un VPC, consultez Réplication MSK.

Résolution des problèmes liés au service géré pour Apache Flink

La documentation suivante peut vous aider à résoudre les problèmes que vous pouvez rencontrer avec le service géré Amazon pour Apache Flink.

Rubriques

- Résolutions des problèmes de développement
- Résolution des problèmes d'exécution

Résolutions des problèmes de développement

Rubriques

- Graphiques en flamme pour Apache Flink
- Problème lié au fournisseur d'informations d'identification avec le connecteur EFO 1.15.2
- Applications dotées de connecteurs Kinesis non pris en charge
- Erreur de compilation : « Impossible de résoudre les dépendances du projet »
- Choix non valide : « kinesisanalyticsv2 »
- UpdateApplication L'action ne consiste pas à recharger le code de l'application
- S3 StreamingFileSink FileNotFoundExceptions
- FlinkKafkaConsumer problème avec l'arrêt avec le point de sauvegarde
- Flink 1.15 Async Sink Deadlock
- Traitement de la source Amazon Kinesis Data Streams désordonné lors du repartitionnement

Graphiques en flamme pour Apache Flink

Les graphiques en flamme sont activés par défaut sur les applications de service géré pour Apache Flink pour les versions qui les prennent en charge. Les graphiques en flamme peuvent affecter les performances de l'application si vous maintenez le graphique ouvert, comme indiqué dans la documentation de Flink.

Si vous souhaitez désactiver les graphiques en flamme pour votre application, créez une demande pour demander qu'il soit désactivé pour l'ARN de votre application. Pour plus d'informations, consultez le AWSCentre de support.

Problème lié au fournisseur d'informations d'identification avec le connecteur EFO 1.15.2

Il existe un problème connu lié aux versions du connecteur EFO de Kinesis Data Streams antérieures à la version 1.15.2, dans lesquelles le `FlinkKinesisConsumer` ne respecte pas la configuration du `Credential Provider`. Les configurations valides ne sont pas prises en compte en raison de ce problème, ce qui entraîne l'utilisation du fournisseur d'informations d'identification `AUTO`. Cela peut entraîner un problème lors de l'accès intercompte à Kinesis à l'aide du connecteur EFO.

Pour résoudre cette erreur, utilisez la version 1.15.3 ou ultérieure du connecteur EFO.

Applications dotées de connecteurs Kinesis non pris en charge

La version 1.15 du service géré pour Apache Flink empêchera automatiquement le démarrage ou la mise à jour des applications si elles utilisent des versions de Kinesis Connector non prises en charge (antérieures à la version 1.15.2) regroupées dans des fichiers JAR ou des archives (ZIP) d'applications.

Erreur de rejet

L'erreur suivante s'affichera lorsque vous soumettrez des appels de création/mise à jour d'application via :

```
An error occurred (InvalidArgumentException) when calling the CreateApplication
 operation: An unsupported Kinesis connector version has been detected in the
 application. Please update flink-connector-kinesis to any version equal to or newer
 than 1.15.2.
For more information refer to connector fix: https://issues.apache.org/jira/browse/
FLINK-23528
```

Étapes de correction

- Mettez à jour la dépendance de l'application sur `flink-connector-kinesis`. Si vous utilisez Maven comme outil de création de projet, suivez Mettre à jour une dépendance Maven . Si vous utilisez Gradle, suivez Mettre à jour une dépendance Gradle .

- Ré-empaquetez l'application.
- Chargez-la sur un compartiment Amazon S3
- Soumettez à nouveau la demande de création/mise à jour d'application avec l'application révisée qui vient d'être chargée sur le compartiment Amazon S3.
- Si le même message d'erreur persiste, vérifiez à nouveau les dépendances de votre application. Si le problème persiste, veuillez créer un ticket d'assistance.

Mettre à jour une dépendance Maven

1. Ouvrez le fichier `pom.xml` du projet.
2. Cherchez les dépendances du projet. Elles se présentent comme suit :

```
<project>

    ...

    <dependencies>

        ...

        <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-connector-kinesis</artifactId>
        </dependency>

        ...

    </dependencies>

    ...

</project>
```

3. Effectuez une mise à jour de `flink-connector-kinesis` vers la version 1.15.2 ou une version ultérieure. Par exemple :

```
<project>

    ...
```

```
    <dependencies>

        ...

        <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-connector-kinesis</artifactId>
            <version>1.15.2</version>
        </dependency>

        ...

    </dependencies>

    ...

</project>
```

Mettre à jour une dépendance Gradle

1. Ouvrez le fichier `build.gradle` (ou `build.gradle.kts` pour les applications Kotlin) du projet.
2. Cherchez les dépendances du projet. Elles se présentent comme suit :

```
...

dependencies {

    ...

    implementation("org.apache.flink:flink-connector-kinesis")

    ...

}

...
```

3. Effectuez une mise à jour de `flink-connector-kinesis` vers la version 1.15.2 ou une version ultérieure. Par exemple :

```
...

dependencies {

    ...

    implementation("org.apache.flink:flink-connector-kinesis:1.15.2")

    ...

}

...
```

Erreur de compilation : « Impossible de résoudre les dépendances du projet »

Pour compiler les exemples d'applications de service géré pour Apache Flink, vous devez d'abord télécharger et compiler le connecteur Apache Flink Kinesis et l'ajouter à votre référentiel Maven local. Si le connecteur n'a pas été ajouté à votre référentiel, une erreur de compilation similaire à la suivante apparaît :

```
Could not resolve dependencies for project your project name: Failure to
 find org.apache.flink:flink-connector-kinesis_2.11:jar:1.8.2 in https://
repo.maven.apache.org/maven2 was cached in the local repository, resolution will not be
 reattempted until the update interval of central has elapsed or updates are forced
```

Pour résoudre cette erreur, vous devez télécharger le code source d'Apache Flink (version 1.8.2 sur https://flink.apache.org/downloads.html) pour le connecteur. Pour obtenir des instructions sur le téléchargement, la compilation et l'installation du code source d'Apache Flink, consultez the section called “Utilisation du connecteur Kinesis Streams d’Apache Flink avec les versions précédentes d'Apache Flink”.

Choix non valide : « kinesisanalyticsv2 »

Pour utiliser la version 2 de l'API Managed Service for Apache Flink, vous avez besoin de la dernière version de AWS Command Line Interface (AWS CLI).

Pour plus d'informations sur la mise à niveau de AWS CLI, consultez Installation de AWS Command Line Interface dans le Guide de l'utilisateur AWS Command Line Interface.

UpdateApplication L'action ne consiste pas à recharger le code de l'application

L'UpdateApplicationaction ne rechargera pas le code de l'application avec le même nom de fichier si aucune version d'objet S3 n'est spécifiée. Pour recharger le code d'application avec le même nom de fichier, activez la gestion des versions sur votre compartiment S3 et spécifiez la nouvelle version de l'objet à l'aide du paramètre `ObjectVersionUpdate`. Pour de plus amples informations sur l'activation de la gestion des versions sur un compartiment S3, veuillez consulter Activation et désactivation de la gestion des versions.

S3 StreamingFileSink FileNotFoundExceptions

Les applications de service géré pour Apache Flink peuvent rencontrer une erreur `FileNotFoundException` de fichier partiels En cours lors du démarrage à partir d'instantanés si un fichier partiel En cours référencé par son point de sauvegarde est manquant. Lorsque ce mode d'échec se produit, l'état d'opérateur de l'application de service géré pour Apache Flink est généralement irrécupérable et doit être redémarré sans instantané, en utilisant `SKIP_RESTORE_FROM_SNAPSHOT`. Consultez l'exemple de trace de pile suivant :

```
java.io.FileNotFoundException: No such file or directory: s3://your-s3-bucket/pathj/
INSERT/2023/4/19/7/_part-2-1234_tmp_12345678-1234-1234-1234-123456789012
        at
 org.apache.hadoop.fs.s3a.S3AFileSystem.s3GetFileStatus(S3AFileSystem.java:2231)
        at
 org.apache.hadoop.fs.s3a.S3AFileSystem.innerGetFileStatus(S3AFileSystem.java:2149)
        at
 org.apache.hadoop.fs.s3a.S3AFileSystem.getFileStatus(S3AFileSystem.java:2088)
        at org.apache.hadoop.fs.s3a.S3AFileSystem.open(S3AFileSystem.java:699)
        at org.apache.hadoop.fs.FileSystem.open(FileSystem.java:950)
        at
 org.apache.flink.fs.s3hadoop.HadoopS3AccessHelper.getObject(HadoopS3AccessHelper.java:98)
        at
 org.apache.flink.fs.s3.common.writer.S3RecoverableMultipartUploadFactory.recoverInProgressPart
...
```

Flink `StreamingFileSink` écrit des enregistrements dans les systèmes de fichiers supportés par l'abstraction Flink. FileSystem Étant donné que les flux entrants peuvent être illimités, les données

sont organisées en fichiers partiels de taille limitée et de nouveaux fichiers sont ajoutés au fur et à mesure de l'écriture des données. Le cycle de vie des fichiers partiels et la stratégie de substitution déterminent le calendrier, la taille et le nom des fichiers partiels.

> **Note**
>
> Pour plus d'informations, consultez Cycle de vie des fichiers partiels.

Lors de la la création de points de contrôle et de points de sauvegarde (création d'instantané), tous les fichiers en attente sont renommés et validés. Cependant, les fichiers partiels En cours ne sont pas validés, mais sont renommés, et leur référence est conservée dans les métadonnées du point de contrôle ou du point de sauvegarde à utiliser lors de la restauration des tâches. Ces fichiers partiels En cours finiront par passer à l'état En suspens, puis seront renommés et validés à un point de contrôle ou de sauvegarde ultérieur.

Voici les causes premières et les mesures à prendre pour les fichiers partiels En cours manquants :

- Instantané périmé utilisé pour démarrer l'application Managed Service for Apache Flink : seul le dernier instantané du système pris lors de l'arrêt ou de la mise à jour d'une application peut être utilisé pour démarrer une application Managed Service for Apache Flink avec Amazon S3. StreamingFileSink Pour éviter ce type d'échec, utilisez le dernier instantané du système.
  - Cela se produit par exemple lorsque vous choisissez un instantané créé à l'aide de `CreateSnapshot` au lieu d'un instantané déclenché par le système lors d'un arrêt ou d'une mise à jour. Le point de sauvegarde de l'ancien instantané conserve une out-of-date référence au fichier de pièce en cours qui a été renommé et validé par le point de contrôle ou le point de sauvegarde suivant.
  - Cela peut également se produire lorsqu'un instantané déclenché par le système à la suite d'un événement d'arrêt/de mise à jour non récent est sélectionné. Par exemple, une application dont la capture d'instantané par le système est désactivée, mais où l'option `RESTORE_FROM_LATEST_SNAPSHOT` est configurée. En règle générale, le service géré pour les applications Apache Flink avec Amazon S3 StreamingFileSink doit toujours avoir la capture instantanée du système activée et `RESTORE_FROM_LATEST_SNAPSHOT` configurée.
- Fichier partiel En cours supprimé : comme le fichier partiel En cours se trouve dans un compartiment S3, il peut être supprimé par d'autres composants ou acteurs ayant accès au compartiment.

  - Cela peut se produire lorsque vous avez arrêté votre application trop longtemps et que le fichier de partie en cours référencé par le point de sauvegarde de votre application a été supprimé conformément à la politique de MultiPartUpload cycle de vie des compartiments S3. Pour éviter ce type de panne, assurez-vous que la stratégie de cycle de vie MPU du compartiment S3 couvre une période suffisamment longue pour votre cas d'utilisation.
  - Cela peut également se produire lorsque le fichier partiel En cours a été supprimé manuellement ou par un autre composant de votre système. Pour éviter ce type d'échec, assurez-vous que les fichiers partiels En cours ne sont pas supprimés par d'autres acteurs ou composants.
- Condition de course dans laquelle un point de contrôle automatique est déclenché après le point de sauvegarde : cela affecte les versions du service géré pour Apache Flink jusqu'à la version 1.13 incluse. Ce problème est résolu dans la version 1.15 du service géré pour Apache Flink ; migrez votre application vers cette version pour éviter qu'il ne se reproduise. Nous vous suggérons également de migrer de StreamingFileSink vers. FileSink
  - Lorsque des applications sont arrêtées ou mises à jour, le service géré pour Apache Flink déclenche un point de sauvegarde et arrête l'application en deux étapes. Si un point de contrôle automatique se déclenche entre les deux étapes, le point de sauvegarde sera inutilisable, car son fichier partiel En cours sera renommé et potentiellement validé.

FlinkKafkaConsumer problème avec l'arrêt avec le point de sauvegarde

Lorsque vous utilisez l'ancienne version, FlinkKafkaConsumer il est possible que votre application soit bloquée dans UPDATING, STOPPING ou SCALING, si les instantanés du système sont activés. Aucun correctif publié n'est disponible pour ce problème. Nous vous recommandons donc de passer à la nouvelle version afin KafkaSourced'atténuer ce problème.

Si vous utilisez `FlinkKafkaConsumer` avec les instantanés activés, il est possible que lorsque la tâche Flink traite une demande d'API d'arrêt avec point de sauvegarde, `FlinkKafkaConsumer`échoue avec une erreur d'exécution signalant `ClosedException`. Dans ces conditions, l'application Flink se bloque, indiquant des points de contrôle défaillants.

Flink 1.15 Async Sink Deadlock

Il existe un problème connu lié aux AWS connecteurs de l' AsyncSink interface d'implémentation d'Apache Flink. Cela concerne les applications utilisant Flink 1.15 avec les connecteurs suivants :

- Pour les applications Java :
  - KinesisStreamsSink – `org.apache.flink:flink-connector-kinesis`

  - KinesisStreamsSink – `org.apache.flink:flink-connector-aws-kinesis-streams`
  - KinesisFirehoseSink – `org.apache.flink:flink-connector-aws-kinesis-firehose`
  - DynamoDbSink – `org.apache.flink:flink-connector-dynamodb`

- Applications Flink SQL/TableAPI/Python :
  - kinesis – `org.apache.flink:flink-sql-connector-kinesis`
  - kinesis – `org.apache.flink:flink-sql-connector-aws-kinesis-streams`
  - firehose – `org.apache.flink:flink-sql-connector-aws-kinesis-firehose`
  - dynamodb – `org.apache.flink:flink-sql-connector-dynamodb`

Les applications concernées présenteront les symptômes suivants :

- la tâche Flink est à l'état `RUNNING`, mais ne traite pas les données ;
- il n'y a aucun redémarrage de tâche ;
- les points de contrôle arrivent à expiration.

Le problème est dû à un bogue dans le kit SDK AWS qui l'empêche de signaler certaines erreurs à l'appelant lors de l'utilisation du client HTTP asynchrone. Le puits attend donc indéfiniment qu'une « demande en cours » soit traitée pendant une opération de vidage au point de contrôle.

Ce problème a été résolu dans le kit SDK AWS à partir de la version 2.20.144.

Vous trouverez ci-dessous des instructions sur la façon de mettre à jour les connecteurs concernés afin d'utiliser la nouvelle version du kit SDK AWS dans vos applications :

Rubriques

- Mettre à jour les applications Java
- Mise à jour des applications Python

Mettre à jour les applications Java

Suivez les procédures ci-dessous pour mettre à jour les applications Java :

flink-connector-kinesis

Si l'application utilise `flink-connector-kinesis` :

Le connecteur Kinesis utilise le grisage pour intégrer certaines dépendances, notamment le kit SDK AWS, dans le fichier JAR du connecteur. Pour mettre à jour la version du kit SDK AWS, suivez la procédure suivante pour remplacer ces classes grisées :

Maven

1. Ajoutez le connecteur Kinesis et les modules du kit SDK AWS requis en tant que dépendances du projet.
2. Configuration de `maven-shade-plugin` :

    a. Ajoutez un filtre pour exclure les classes du kit SDK AWS grisées lors de la copie du contenu du fichier JAR du connecteur Kinesis.

    b. Ajoutez une règle de relocalisation pour déplacer les classes du kit SDK AWS mises à jour vers le package, comme prévu par le connecteur Kinesis.

    pom.xml

```
<project>
    ...
    <dependencies>
        ...
        <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-connector-kinesis</artifactId>
            <version>1.15.4</version>
        </dependency>

        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>kinesis</artifactId>
            <version>2.20.144</version>
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>netty-nio-client</artifactId>
            <version>2.20.144</version>
        </dependency>
        <dependency>
            <groupId>software.amazon.awssdk</groupId>
            <artifactId>sts</artifactId>
            <version>2.20.144</version>
```

```
        </dependency>
        ...
    </dependencies>
    ...
    <build>
        ...
        <plugins>
            ...
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-shade-plugin</artifactId>
                <version>3.1.1</version>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>shade</goal>
                        </goals>
                        <configuration>
                            ...
                            <filters>
                                ...
                                <filter>
                                    <artifact>org.apache.flink:flink-connector-
kinesis</artifact>
                                    <excludes>
                                        <exclude>org/apache/flink/kinesis/
shaded/software/amazon/awssdk/**</exclude>
                                        <exclude>org/apache/flink/kinesis/
shaded/org/reactivestreams/**</exclude>
                                        <exclude>org/apache/flink/kinesis/
shaded/io/netty/**</exclude>
                                        <exclude>org/apache/flink/kinesis/
shaded/com/typesafe/netty/**</exclude>
                                    </excludes>
                                </filter>
                                ...
                            </filters>
                            <relocations>
                                ...
                                <relocation>
                                    <pattern>software.amazon.awssdk</pattern>
```

```
 <shadedPattern>org.apache.flink.kinesis.shaded.software.amazon.awssdk</
shadedPattern>
                                </relocation>
                                <relocation>
                                    <pattern>org.reactivestreams</pattern>

 <shadedPattern>org.apache.flink.kinesis.shaded.org.reactivestreams</
shadedPattern>
                                </relocation>
                                <relocation>
                                    <pattern>io.netty</pattern>

 <shadedPattern>org.apache.flink.kinesis.shaded.io.netty</shadedPattern>
                                </relocation>
                                <relocation>
                                    <pattern>com.typesafe.netty</pattern>

 <shadedPattern>org.apache.flink.kinesis.shaded.com.typesafe.netty</
shadedPattern>
                                </relocation>
                                ...
                            </relocations>
                           ...
                        </configuration>
                    </execution>
                </executions>
            </plugin>
            ...
        </plugins>
        ...
    </build>
</project>
```

Gradle

1. Ajoutez le connecteur Kinesis et les modules du kit SDK AWS requis en tant que dépendances du projet.
2. Ajuster la configuration de ShadowJar :

    a. Excluez les classes du kit SDK AWS grisées lors de la copie du contenu du fichier JAR du connecteur Kinesis.

b. Relocalisez les classes du kit SDK AWS mises à jour vers le package, comme prévu par le connecteur Kinesis.

build.gradle

```
...
dependencies {
    ...
    flinkShadowJar("org.apache.flink:flink-connector-kinesis:1.15.4")

    flinkShadowJar("software.amazon.awssdk:kinesis:2.20.144")
    flinkShadowJar("software.amazon.awssdk:sts:2.20.144")
    flinkShadowJar("software.amazon.awssdk:netty-nio-client:2.20.144")
    ...
}
...
shadowJar {
    configurations = [project.configurations.flinkShadowJar]

    exclude("org/apache/flink/kinesis/shaded/software/amazon/awssdk/**/*.class")
    exclude("org/apache/flink/kinesis/shaded/org/reactivestreams/**/*.class")
    exclude("org/apache/flink/kinesis/shaded/io/netty/**/*.class")
    exclude("org/apache/flink/kinesis/shaded/com/typesafe/netty/**/*.class")

    relocate("software.amazon.awssdk",
 "org.apache.flink.kinesis.shaded.software.amazon.awssdk")
    relocate("org.reactivestreams",
 "org.apache.flink.kinesis.shaded.org.reactivestreams")
    relocate("io.netty", "org.apache.flink.kinesis.shaded.io.netty")
    relocate("com.typesafe.netty",
 "org.apache.flink.kinesis.shaded.com.typesafe.netty")
}
...
```

Autres connecteurs concernés

Si l'application utilise un autre connecteur concerné :

Afin de mettre à jour la version du kit SDK AWS, la version du kit SDK doit être appliquée dans la configuration de construction du projet.

Maven

Ajoutez la nomenclature (BOM) du kit SDK AWS à la section de gestion des dépendances du fichier `pom.xml` pour appliquer la version du kit SDK au projet.

pom.xml

```
<project>
    ...
    <dependencyManagement>
        <dependencies>
            ...
            <dependency>
                <groupId>software.amazon.awssdk</groupId>
                <artifactId>bom</artifactId>
                <version>2.20.144</version>
                <scope>import</scope>
                <type>pom</type>
            </dependency>
            ...
        </dependencies>
    </dependencyManagement>
    ...
</project>
```

Gradle

Ajoutez la dépendance de la plateforme à la nomenclature (BOM) du kit SDK AWS pour appliquer la version du kit SDK au projet. Cela nécessite Gradle 5.0 ou une version ultérieure :

build.gradle

```
...
dependencies {
    ...
    flinkShadowJar(platform("software.amazon.awssdk:bom:2.20.144"))
    ...
}
...
```

Mise à jour des applications Python

Les applications Python peuvent utiliser les connecteurs de deux manières différentes : empaqueter des connecteurs et autres dépendances Java dans le cadre d'un fichier JAR uber unique ou utiliser directement le connecteur JAR. Pour corriger les applications affectées par le blocage d'Async Sink :

- Si l'application utilise un fichier JAR uber, suivez les instructions de Mettre à jour les applications Java .
- Pour reconstruire les connecteurs JAR à partir de la source, procédez comme suit :

Création de connecteurs à partir de la source :

Prérequis, similaires aux exigences de compilation Flink :

- Java 11
- Maven 3.2.5

flink-sql-connector-kinesis

1. Téléchargez le code source pour Flink 1.15.4 :

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Décompressez le code source :

```
tar -xvf flink-1.15.4-src.tgz
```

3. Accédez au répertoire du connecteur Kinesis

```
cd flink-1.15.4/flink-connectors/flink-connector-kinesis/
```

4. Compilez et installez le JAR du connecteur, en spécifiant la version du kit SDK AWS requise. Pour accélérer la compilation, utilisez `-DskipTests` pour ignorer l'exécution des tests et `-Dfast` pour ignorer les vérifications supplémentaires du code source :

```
mvn clean install -DskipTests -Dfast -Daws.sdkv2.version=2.20.144
```

5. Accédez au répertoire du connecteur Kinesis

```
cd ../flink-sql-connector-kinesis
```

6. Compilez et installez le JAR du connecteur SQL :

```
mvn clean install -DskipTests -Dfast
```

7. Le fichier JAR obtenu sera disponible à l'adresse suivante :

```
target/flink-sql-connector-kinesis-1.15.4.jar
```

flink-sql-connector-aws-kinesis-streams

1. Téléchargez le code source pour Flink 1.15.4 :

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Décompressez le code source :

```
tar -xvf flink-1.15.4-src.tgz
```

3. Accédez au répertoire du connecteur Kinesis

```
cd flink-1.15.4/flink-connectors/flink-connector-aws-kinesis-streams/
```

4. Compilez et installez le JAR du connecteur, en spécifiant la version du kit SDK AWS requise. Pour accélérer la compilation, utilisez `-DskipTests` pour ignorer l'exécution des tests et `-Dfast` pour ignorer les vérifications supplémentaires du code source :

```
mvn clean install -DskipTests -Dfast -Daws.sdk.version=2.20.144
```

5. Accédez au répertoire du connecteur Kinesis

```
cd ../flink-sql-connector-aws-kinesis-streams
```

6. Compilez et installez le JAR du connecteur SQL :

```
mvn clean install -DskipTests -Dfast
```

7. Le fichier JAR obtenu sera disponible à l'adresse suivante :

```
target/flink-sql-connector-aws-kinesis-streams-1.15.4.jar
```

flink-sql-connector-aws-kinesis-firehose

1. Téléchargez le code source pour Flink 1.15.4 :

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Décompressez le code source :

```
tar -xvf flink-1.15.4-src.tgz
```

3. Accédez au répertoire du connecteur

```
cd flink-1.15.4/flink-connectors/flink-connector-aws-kinesis-firehose/
```

4. Compilez et installez le JAR du connecteur, en spécifiant la version du kit SDK AWS requise. Pour accélérer la compilation, utilisez `-DskipTests` pour ignorer l'exécution des tests et `-Dfast` pour ignorer les vérifications supplémentaires du code source :

```
mvn clean install -DskipTests -Dfast -Daws.sdk.version=2.20.144
```

5. Accédez au répertoire du connecteur SQL

```
cd ../flink-sql-connector-aws-kinesis-firehose
```

6. Compilez et installez le JAR du connecteur SQL :

```
mvn clean install -DskipTests -Dfast
```

7. Le fichier JAR obtenu sera disponible à l'adresse suivante :

```
target/flink-sql-connector-aws-kinesis-firehose-1.15.4.jar
```

flink-sql-connector-dynamodb

1. Téléchargez le code source pour Flink 1.15.4 :

```
wget https://archive.apache.org/dist/flink/flink-connector-aws-3.0.0/flink-
connector-aws-3.0.0-src.tgz
```

2. Décompressez le code source :

```
tar -xvf flink-connector-aws-3.0.0-src.tgz
```

3. Accédez au répertoire du connecteur

```
cd flink-connector-aws-3.0.0
```

4. Compilez et installez le JAR du connecteur, en spécifiant la version du kit SDK AWS requise. Pour accélérer la compilation, utilisez `-DskipTests` pour ignorer l'exécution des tests et `-Dfast` pour ignorer les vérifications supplémentaires du code source :

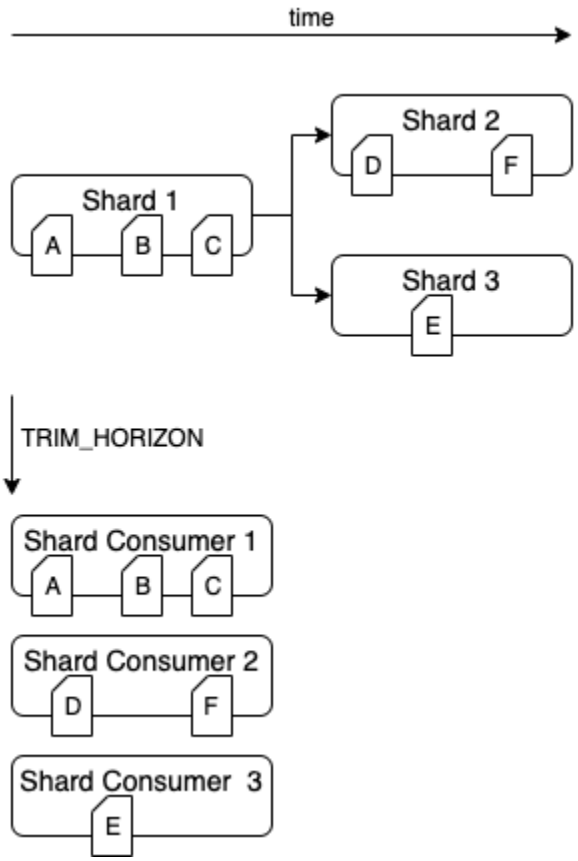
```
mvn clean install -DskipTests -Dfast -Dflink.version=1.15.4 -
Daws.sdk.version=2.20.144
```

5. Le fichier JAR obtenu sera disponible à l'adresse suivante :

```
flink-sql-connector-dynamodb/target/flink-sql-connector-dynamodb-3.0.0.jar
```

Traitement de la source Amazon Kinesis Data Streams désordonné lors du repartitionnement

L' FlinkKinesisConsumer implémentation actuelle ne fournit pas de solides garanties d'ordre entre les partitions Kinesis. Cela peut entraîner un out-of-order traitement lors du resharding de Kinesis Stream, en particulier pour les applications Flink présentant un retard de traitement. Dans certaines circonstances, par exemple lorsque les opérateurs Windows sont basés sur l'heure des événements, les événements peuvent être ignorés en raison du retard qui en résulte.



Il s'agit d'un problème connu dans Open Source Flink. Jusqu'à ce que le correctif du connecteur soit disponible, assurez-vous que vos applications Flink ne prennent pas de retard sur Kinesis Data Streams lors du repartitionnement. En vous assurant que le délai de traitement est toléré par vos applications Flink, vous pouvez minimiser l'impact du out-of-order traitement et le risque de perte de données.

Résolution des problèmes d'exécution

Cette section contient des informations sur le diagnostic et la résolution des problèmes d'exécution de votre application de service géré pour Apache Flink.

Rubriques

- Outils de dépannage
- Problèmes d'applications
- L'application est en train de redémarrer
- Le débit est trop lent
- Croissance illimitée de l'état

- Opérateurs liés aux E/S
- Limitation en amont ou à la source à partir d'un flux de données Kinesis
- Points de contrôle
- Le délai du point de contrôle est expiré
- Échec de point de contrôle pour l'application Apache Beam
- Contre-pression
- Asymétrie de données
- Asymétrie d'état
- Intégration de ressources de différentes régions

Outils de dépannage

Les alarmes CloudWatch constituent le principal outil de détection des problèmes liés aux applications. À l'aide des alarmes CloudWatch, vous pouvez définir des seuils pour les métriques CloudWatch qui indiquent des erreurs ou des goulots d'étranglement dans votre application. Pour des informations sur les alarmes CloudWatch recommandées, consultez Utilisation des CloudWatch alarmes avec Amazon Managed Service pour Apache Flink.

Problèmes d'applications

Cette section contient des solutions aux erreurs que vous pourriez rencontrer avec votre application de service géré pour for Apache Flink.

Rubriques

- L'application est bloquée dans un état transitoire
- La création d'un instantané échoue
- Impossible d’accéder aux ressources d'un VPC
- Des données sont perdues lors de l'écriture dans un compartiment Amazon S3
- L'application affiche l’état EN COURS D’EXÉCUTION mais ne traite pas les données
- Erreur d’instantané, de mise à jour de l'application ou d'arrêt de l'application : InvalidApplicationConfigurationException
- java.nio.file.NoSuchFileException : /usr/local/openjdk-8/lib/security/cacerts

L'application est bloquée dans un état transitoire

Si votre application reste dans un état transitoire (`STARTING`,`UPDATING`, `STOPPING` ou `AUTOSCALING`), vous pouvez l'arrêter en utilisant l'action StopApplication avec le paramètre `Force`défini sur `true`. Vous ne pouvez pas forcer l'arrêt d'une application lorsqu'elle est à l'état `DELETING`. Sinon, si l'application a l'état `UPDATING` ou `AUTOSCALING`, vous pouvez revenir à la version précédente en cours d'exécution. Lorsque vous annulez une application, elle charge les données d'état du dernier instantané réussi. Si l'application ne possède aucun instantané, le service géré pour Apache Flink rejette la demande de restauration. Pour plus d'informations sur l'annulation d'une application, consultez l'action RollbackApplication.

> **Note**
>
> L'arrêt forcé de votre application peut entraîner une perte ou une duplication des données. Pour éviter la perte de données ou le double traitement des données lors du redémarrage de l'application, nous vous recommandons d'enregistrer fréquemment des instantanés de votre application.

Les causes de blocage des applications peuvent être les suivantes :

- L'état de l'application est trop important : si l'état de l'application est trop important ou trop persistant, l'application peut se bloquer lors d'une opération de point de contrôle ou d'instantané. Vérifiez les métriques `lastCheckpointDuration` et `lastCheckpointSize` de votre application pour détecter des valeurs en augmentation constante ou anormalement élevées.
- Le code de l'application est trop volumineux : vérifiez que le fichier JAR de votre application est inférieur à 512 Mo. Les fichiers JAR de plus de 512 Mo ne sont pas pris en charge.
- La création d'un instantané de l'application échoue : le service géré pour Apache Flink crée un instantané de l'application lors d'une demande `UpdateApplication` ou `StopApplication`. Le service utilise ensuite cet état d'instantané et restaure l'application à l'aide de la configuration mise à jour de l'application afin de fournir une sémantique de traitement unique. Si la création automatique d'instantané échoue, consultez La création d'un instantané échoue.
- La restauration à partir d'un instantané échoue : si vous supprimez ou modifiez un opérateur dans une mise à jour d'application et que vous tentez de restaurer à partir d'un instantané, la restauration échouera par défaut si l'instantané contient les données d'état de l'opérateur manquant. De plus, l'application sera bloquée à l'état `STOPPED` ou `UPDATING`. Pour modifier ce comportement et permettre à la restauration de réussir, remplacez le paramètre

AllowNonRestoredState de FlinkRunConfiguration de l'application par `true`. Cela permettra à l'opération de reprise d'ignorer les données d'état qui ne peuvent pas être mises en correspondance avec le nouveau programme.

- L'initialisation de l'application prend trop de temps : le service géré pour Apache Flink utilise un délai d'attente interne de 5 minutes (réglage modifiable) pour le démarrage d'une tâche Flink. Si votre tâche ne démarre pas dans ce délai, vous verrez un journal CloudWatch comme suit :

```
Flink job did not start within a total timeout of 5 minutes for application: %s under
 account: %s
```

  Si vous rencontrez l'erreur ci-dessus, cela signifie que vos opérations définies selon la méthode `main` de la tâche Flink prennent plus de 5 minutes, entraînant l'expiration du délai de création de la tâche Flink du côté du service géré pour Apache Flink. Nous vous suggérons de consulter les journaux JobManager de Flink ainsi que le code de votre application pour voir si ce retard dans la méthode `main` est normal. Si ce n'est pas le cas, vous devez prendre des mesures pour résoudre le problème afin que l'opération prenne moins de 5 minutes.

Vous pouvez vérifier l'état de votre application à l'aide des actions `ListApplications` ou `DescribeApplication`.

La création d'un instantané échoue

Le service géré pour Apache Flink ne peut pas prendre d'instantané dans les circonstances suivantes :

- L'application a dépassé la limite d'instantané. La limite est de 1 000 instantanés. Pour de plus amples informations, veuillez consulter Instantanés.
- L'application n'est pas autorisée à accéder à sa source ou à son récepteur.
- Le code de l'application ne fonctionne pas correctement.
- L'application rencontre d'autres problèmes de configuration.

Si vous obtenez une exception lors de la création d'un instantané pendant la mise à jour ou l'arrêt de l'application, définissez la propriété `SnapshotsEnabled` de la `ApplicationSnapshotConfiguration` de votre application sur `false` et réessayez la demande.

Les instantanés peuvent échouer si les opérateurs de votre application ne sont pas correctement configurés. Pour plus d'informations sur le réglage des performances des opérateurs, consultez Mise à l'échelle des opérateurs.

Une fois que l'application est revenue à un état sain, nous vous recommandons de définir la propriété `SnapshotsEnabled` de l'application sur `true`.

Impossible d'accéder aux ressources d'un VPC

Si votre application utilise un VPC exécuté sur Amazon VPC, procédez comme suit pour vérifier que votre application a accès à ses ressources :

- Vérifiez que vos journaux CloudWatch ne contiennent pas l'erreur suivante. Cette erreur indique que votre application ne peut pas accéder aux ressources de votre VPC :

  ```
  org.apache.kafka.common.errors.TimeoutException: Failed to update metadata after
   60000 ms.
  ```

  Si cette erreur s'affiche, vérifiez que vos tables de routage sont correctement configurées et que les paramètres de connexion de vos connecteurs sont corrects.

  Pour plus d'informations sur la configuration et l'analyse des journaux de CloudWatch, consultez Journalisation et surveillance.

Des données sont perdues lors de l'écriture dans un compartiment Amazon S3

Certaines pertes de données peuvent se produire lors de l'écriture d'un fichier de sortie dans un compartiment Amazon S3 à l'aide d'Apache Flink version 1.6.2. Nous vous recommandons d'utiliser la dernière version prise en charge d'Apache Flink lorsque vous utilisez Amazon S3 pour écrire un fichier de sortie directement. Pour écrire dans un compartiment Amazon S3 à l'aide d'Apache Flink 1.6.2, nous vous recommandons d'utiliser Kinesis Data Firehose. Pour plus d'informations sur l'utilisation de Kinesis Data Firehose avec le service géré pour Apache Flink, consultez Récepteur Kinesis Data Firehose.

L'application affiche l'état EN COURS D'EXÉCUTION mais ne traite pas les données

Vous pouvez vérifier l'état de votre application à l'aide des actions `ListApplications` ou `DescribeApplication`. Si votre application entre dans l'état `RUNNING`, mais n'écrit pas de données dans votre récepteur, vous pouvez résoudre le problème en ajoutant un flux de journaux

Amazon CloudWatch à votre application. Pour de plus amples informations, veuillez consulter Utilisation des options de CloudWatch journalisation des applications. Le flux de journaux contient des messages que vous pouvez utiliser pour résoudre les problèmes de l'application.

Erreur d'instantané, de mise à jour de l'application ou d'arrêt de l'application : InvalidApplicationConfigurationException

Une erreur similaire à la suivante peut se produire lors d'une opération d'instantané ou lors d'une opération de création d'un instantané, telle que la mise à jour ou l'arrêt d'une application :

```
An error occurred (InvalidApplicationConfigurationException) when calling the
 UpdateApplication operation:

Failed to take snapshot for the application xxxx at this moment. The application is
 currently experiencing downtime.
Please check the application's CloudWatch metrics or CloudWatch logs for any possible
 errors and retry the request.
You can also retry the request after disabling the snapshots in the Managed Service for
 Apache Flink console or by updating
the ApplicationSnapshotConfiguration through the AWS SDK
```

Cette erreur se produit lorsque l'application ne parvient pas à créer un instantané.

Si vous rencontrez cette erreur lors d'une opération d'instantané ou d'une opération de création d'un instantané, procédez comme suit :

- Désactivez les instantanés pour votre application. Vous pouvez le faire soit dans la console du service géré pour Apache Flink, soit en utilisant le paramètre `SnapshotsEnabledUpdate` de l'action UpdateApplication.
- Découvrez pourquoi les instantanés ne peuvent pas être créés. Pour de plus amples informations, veuillez consulter L'application est bloquée dans un état transitoire.
- Réactivez les instantanés lorsque l'application revient à un état sain.

java.nio.file.NoSuchFileException : /usr/local/openjdk-8/lib/security/cacerts

L'emplacement du truststore SSL a été mis à jour lors d'un déploiement précédent. Utilisez la valeur suivante pour le paramètre `ssl.truststore.location` à la place :

```
/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
```

L'application est en train de redémarrer

Si votre application n'est pas saine, sa tâche Apache Flink échoue et redémarre continuellement. Cette section décrit les symptômes et les étapes de résolution de ce problème.

Symptômes

Ce problème peut présenter les symptômes suivants :

- La métrique `FullRestarts` n'est pas nulle. Cette métrique représente le nombre de fois où la tâche de l'application a été redémarrée depuis que vous l'avez lancée.
- La métrique `Downtime` n'est pas nulle. Cette métrique représente le nombre de millisecondes pendant lesquelles l'application est à l'état `FAILING` ou `RESTARTING`.
- Le journal des applications contient les passages à l'état `RESTARTING` ou `FAILED`. Vous pouvez interroger le journal de votre application pour connaître ces changements d'état à l'aide de la requête CloudWatch Logs Insights suivante : Analyser les erreurs : Application Task-Related Failures.

Causes et solutions

Les conditions suivantes peuvent rendre votre application instable et provoquer des redémarrages répétés :

- L'opérateur lève une exception : si une exception d'un opérateur de votre application n'est pas gérée, l'application bloque (interprétant que l'échec ne peut pas être géré par l'opérateur). L'application redémarre à partir du dernier point de contrôle afin de conserver la sémantique de traitement unique. Par conséquent, la valeur `Downtime` n'est pas nulle pendant ces périodes de redémarrage. Pour éviter que cela ne se produise, nous vous recommandons de gérer toutes les exceptions réessayables dans le code de l'application.

  Vous pouvez rechercher les causes de cette situation en interrogeant les journaux de votre application pour vérifier si l'état de votre application est passé de `RUNNING` à `FAILED`. Pour de plus amples informations, veuillez consulter the section called “Analyser les erreurs : Application Task-Related Failures”.
- Les flux de données Kinesis ne sont pas correctement provisionnés : si la source ou le récepteur de votre application est un flux de données Kinesis, vérifiez si les métriques du flux ne contiennent pas d'erreurs `ReadProvisionedThroughputExceeded` ou `WriteProvisionedThroughputExceeded`.

Si ces erreurs s'affichent, vous pouvez augmenter le débit disponible pour le flux Kinesis en augmentant le nombre de partitions du flux. Pour plus d'informations, consultez la rubrique Comment modifier le nombre de partitions ouvertes dans Kinesis Data Streams ?

- Les autres sources ou récepteurs ne sont pas correctement provisionnés ou disponibles : vérifiez que votre application provisionne correctement les sources et les récepteurs. Vérifiez que les sources ou les récepteurs utilisés dans l'application (tels que les autres services AWS ou les sources et destinations externes) sont bien provisionnés et ne sont pas limités en lecture ou en écriture ou périodiquement indisponibles.

  Si vous rencontrez des problèmes de débit avec vos services dépendants, augmentez les ressources disponibles pour ces services ou recherchez la cause des erreurs ou indisponibilités.
- Les opérateurs ne sont pas correctement provisionnés : si la charge de travail sur les threads de l'un des opérateurs de votre application n'est pas correctement répartie, l'opérateur peut être surchargé et l'application peut se bloquer. Pour plus d'informations sur le réglage du parallélisme des opérateurs, consultez Gérez correctement la mise à l'échelle des opérateurs.
- L'application échoue avec DaemonException : cette erreur apparaît dans le journal de votre application si vous utilisez une version d'Apache Flink antérieure à la version 1.11. Vous devrez peut-être effectuer une mise à niveau vers une version ultérieure d'Apache Flink afin d'utiliser une version KPL 0.14 ou ultérieure.
- L'application échoue avec TimeoutException, FlinkException ou RemoteTransportException : ces erreurs peuvent apparaître dans le journal de votre application en cas de panne de vos gestionnaires de tâches. Si votre application est surchargée, vos gestionnaires de tâches peuvent être soumis à une pression sur les ressources du processeur ou de la mémoire, ce qui peut entraîner leur échec.

  Ces erreurs peuvent se présenter comme suit :

  - `java.util.concurrent.TimeoutException: The heartbeat of JobManager with id xxx timed out`
  - `org.apache.flink.util.FlinkException: The assigned slot xxx was removed`
  - `org.apache.flink.runtime.io.network.netty.exception.RemoteTransportException: Connection unexpectedly closed by remote task manager`

  Pour résoudre ce problème, vérifiez les points suivants :

  - Vérifiez vos métriques CloudWatch pour détecter des pics inhabituels d'utilisation du processeur ou de la mémoire.

- Vérifiez que votre application ne présente aucun problème de débit. Pour de plus amples informations, veuillez consulter Dépannage des performances.
- Examinez le journal de votre application pour détecter les exceptions non gérées générées par le code de votre application.

- L'application échoue avec l'erreur JaxBannotationModule Not Found : cette erreur se produit si votre application utilise Apache Beam, mais ne possède pas les dépendances ou les versions de dépendances correctes. Les applications du service géré pour Apache Flink qui utilisent Apache Beam doivent utiliser les versions de dépendances suivantes :

```
<jackson.version>2.10.2</jackson.version>
...
<dependency>
    <groupId>com.fasterxml.jackson.module</groupId>
    <artifactId>jackson-module-jaxb-annotations</artifactId>
    <version>2.10.2</version>
</dependency>
```

  Si vous ne fournissez pas la bonne version de `jackson-module-jaxb-annotations` en tant que dépendance explicite, votre application la charge à partir des dépendances de l'environnement, et comme les versions ne correspondent pas, l'application se bloque au moment de l'exécution.

  Pour plus d'informations sur l'utilisation d'Apache Beam avec le service géré pour Apache Flink, consultez Utilisation de CloudFormation avec le service géré pour Apache Flink.

- L'application échoue avec java.io.IOException : nombre insuffisant de tampons réseau

  Cela se produit lorsqu'une application ne dispose pas de mémoire suffisante pour les tampons réseau. Les tampons réseau facilitent la communication entre les sous-tâches. Ils sont utilisés pour stocker des enregistrements avant leur transmission sur un réseau et pour stocker les données entrantes avant de les disséquer en enregistrements et de les transmettre à des sous-tâches. Le nombre de tampons réseau requis varie directement en fonction du parallélisme et de la complexité de votre graphique de tâches. Il existe plusieurs approches pour atténuer ce problème :

  - Vous pouvez configurer une valeur `parallelismPerKpu` inférieure pour augmenter la quantité de mémoire allouée par sous-tâche et par mémoire tampon réseau. Notez que la réduction de la valeur `parallelismPerKpu` augmentera le nombre d'unités KPU et donc le coût. Pour éviter cela, vous pouvez conserver la même quantité d'unités KPU en réduisant le parallélisme du même facteur.

- Vous pouvez simplifier votre graphe de tâches en réduisant le nombre d'opérateurs ou en créant des chaînes afin de réduire le nombre de tampons nécessaires.
- Sinon, vous pouvez vous rendre sur https://aws.amazon.com/premiumsupport/ pour une configuration personnalisée de la mémoire tampon réseau.

Le débit est trop lent

Si votre application ne traite pas assez rapidement les données de streaming entrantes, elle fonctionnera mal et deviendra instable. Cette section décrit les symptômes et les étapes de résolution de ce problème.

Symptômes

Ce problème peut présenter les symptômes suivants :

- Si la source de données de votre application est un flux Kinesis, la métrique `millisbehindLatest` du flux augmente continuellement.
- Si la source de données de votre application est un cluster Amazon MSK, les métriques de retard des consommateurs pour le cluster augmentent continuellement. Pour plus d'informations, consultez la section Surveillance du retard des consommateurs du guide du développeur Amazon MSK.
- Si la source de données de votre application est un autre service ou une autre source, vérifiez les métriques de retard des consommateurs ou les données disponibles.

Causes et solutions

Le ralentissement du débit des applications peut avoir de nombreuses causes. Si votre application ne suit pas le rythme des entrées, vérifiez les points suivants :

- Si le retard de débit augmente puis diminue, vérifiez si l'application redémarre. Votre application arrêtera de traiter les entrées pendant le redémarrage, ce qui provoquera un pic de retard. Pour plus d'informations sur les échecs d'application, consultez L'application est en train de redémarrer.
- Si le retard de débit est constant, vérifiez si les performances de votre application sont optimisées. Pour plus d'informations sur l'optimisation des performances de votre application, consultez Dépannage des performances.

- Si le retard de débit n'augmente pas de manière excessive, mais augmente continuellement et que les performances de votre application sont optimisées, vous devez augmenter les ressources de votre application. Pour plus d'informations sur l'augmentation des ressources d'application, consultezMise à l'échelle.
- Si votre application lit à partir d'un cluster Kafka situé dans une autre région et que `FlinkKafkaConsumer` ou `KafkaSource` sont principalement inactifs (`idleTimeMsPerSecond` élevé ou `CPUUtilization` faible) malgré un retard important pour les consommateurs, vous pouvez augmenter la valeur pour `receive.buffer.byte`, par exemple 2097152. Pour de plus amples informations, veuillez consulter la section relative à l'environnement à retard élevé de Configurations MSK personnalisées.

Pour les étapes de résolution des problèmes liés au ralentissement du débit ou à l'augmentation du retard des consommateurs dans la source de l'application, consultez Dépannage des performances.

Croissance illimitée de l'état

Si votre application ne supprime pas correctement les informations d'état obsolètes, elles s'accumuleront continuellement et entraîneront des problèmes de performance ou de stabilité de l'application. Cette section décrit les symptômes et les étapes de résolution de ce problème.

Symptômes

Ce problème peut présenter les symptômes suivants :

- La métrique `lastCheckpointDuration` augmente progressivement ou connaît des pics.
- La métrique `lastCheckpointSize` augmente progressivement ou connaît des pics.

Causes et solutions

Les conditions suivantes peuvent amener votre application à accumuler des données d'état :

- Votre application conserve les données d'état plus longtemps que nécessaire.
- Votre application utilise des requêtes de fenêtre d'une durée trop longue.
- Vous n'avez pas défini le TTL pour vos données d'état. Pour plus d'informations, consultez State Time-To-Live (TTL) dans la documentation d'Apache Flink.
- Vous exécutez une application qui dépend de la version 2.25.0 ou ultérieure d'Apache Beam. Vous pouvez désactiver la nouvelle version de la transformation de lecture en étendant vos

BeamApplicationProperties avec les expériences clés et la valeur `use_deprecated_read`. Pour plus d'informations, consultez la documentation Apache Beam.

Parfois, les applications sont confrontées à une augmentation constante de la taille de l'état, qu'elles ne peuvent supporter à long terme (une application Flink fonctionne indéfiniment, après tout). Parfois, cela peut venir du fait que les applications stockent les données dans leur état, sans permettre le vieillissement correct des anciennes informations. Mais parfois, les attentes quant à ce que Flink peut offrir sont tout simplement déraisonnables. Les applications peuvent utiliser des agrégations sur de longues périodes s'étendant sur des jours, voire des semaines. À moins que des fonctions d'agrégation ne soient utilisées, qui permettent des agrégations incrémentielles, Flink doit conserver les événements de l'ensemble de la fenêtre dans l'état.

En outre, lors de l'utilisation de fonctions de processus pour implémenter des opérateurs personnalisés, l'application doit supprimer les données d'état qui ne sont plus nécessaires à la logique métier. Dans ce cas, la durée de vie de l'état peut être utilisée pour éliminer automatiquement les données en fonction du temps de traitement. Le service géré pour Apache Flink utilise des points de contrôle incrémentiels, ainsi l'état TTL est basé sur le compactage RockSDB. Vous ne pouvez observer une réduction réelle de la taille de l'état (indiquée par la taille du point de contrôle) qu'après une opération de compactage. En particulier pour les points de contrôle de taille inférieure à 200 Mo, il est peu probable que vous observiez une réduction de la taille des points de contrôle à la suite de l'expiration de l'état. Cependant, les points de sauvegarde sont basés sur une copie propre de l'état, qui ne contient pas d'anciennes données. Vous pouvez donc déclencher un instantané dans le service géré pour Apache Flink afin de forcer la suppression de l'état obsolète.

À des fins de débogage, il peut être judicieux de désactiver les points de contrôle incrémentiels pour vérifier plus rapidement que la taille du point de contrôle diminue ou se stabilise réellement (et éviter l'effet de compactage dans RocksBS). Pour cela, il faut cependant envoyer un ticket à l'équipe du service.

Opérateurs liés aux E/S

Il est préférable d'éviter toute dépendance à des systèmes externes sur le chemin des données. Il est souvent beaucoup plus performant de conserver un ensemble de données de référence dans son état plutôt que de faire appel à un système externe pour enrichir des événements individuels. Cependant, certaines dépendances ne peuvent pas être facilement déplacées vers un état, par exemple si vous souhaitez enrichir les événements avec un modèle d'apprentissage automatique hébergé sur Amazon Sagemaker.

Les opérateurs qui s'interfacent avec des systèmes externes via le réseau peuvent devenir un goulot d'étranglement et provoquer une contre-pression. Il est fortement recommandé d'utiliser AsyncIO pour implémenter la fonctionnalité, afin de réduire le temps d'attente pour les appels individuels et d'éviter le ralentissement de l'ensemble de l'application.

En outre, pour les applications avec des opérateurs liés aux E/S, il peut également être judicieux d'augmenter le paramètre ParallelismPerKPU de l'application de service géré pour Apache Flink. Cette configuration décrit le nombre de sous-tâches parallèles qu'une application peut effectuer par unité de traitement Kinesis (KPU). En augmentant la valeur par défaut de 1 à 4, par exemple, l'application utilise les mêmes ressources (et a le même coût) mais peut augmenter le parallélisme jusqu'à 4 fois. Cela fonctionne bien pour les applications liées aux E/S, mais entraîne une surcharge supplémentaire pour les applications qui ne sont pas liées aux E/S.

Limitation en amont ou à la source à partir d'un flux de données Kinesis

Symptôme : l'application rencontre une erreur `LimitExceededExceptions` provenant du flux de données Kinesis source en amont.

Cause potentielle : le paramètre par défaut du connecteur Kinesis de la bibliothèque Apache Flink est défini pour lire à partir du flux de données Kinesis source, avec un paramètre par défaut très agressif pour le nombre maximum d'enregistrements extraits par appel `GetRecords`. Apache Flink est configuré par défaut pour récupérer 10 000 enregistrements par appel GetRecords (cet appel est effectué par défaut toutes les 200 ms), bien que la limite par partition ne soit que de 1 000 enregistrements.

Ce comportement par défaut peut entraîner une limitation lors de la tentative de consommation à partir du flux de données Kinesis, ce qui affectera les performances et la stabilité des applications.

Cela peut être confirmé en vérifiant la métrique `ReadProvisionedThroughputExceeded` CloudWatch et en observant les périodes prolongées pendant lesquelles cette métrique est supérieure à zéro.

Le client pourra également voir cela dans les journaux CloudWatch de son application de service géré pour Apache Flink en affichant les erreurs `LimitExceededException` récurrentes.

Résolution : le client peut effectuer l'une des deux actions suivantes pour résoudre ce problème :

- Réduire la limite par défaut du nombre d'enregistrements récupérés par appel GetRecords

- Le client peut activer les lectures adaptatives dans son application de service géré pour Apache Flink. Pour plus d'informations sur la fonctionnalité de lecture adaptative, consultez SHARD_USE_ADAPTIVE_READS

Points de contrôle

Les points de contrôle sont le mécanisme utilisé par Flink pour garantir que l'état d'une application est tolérant aux pannes. Ce mécanisme permet à Flink de récupérer l'état des opérateurs en cas d'échec de la tâche et donne à l'application la même sémantique qu'une exécution sans échec. Avec le service géré pour Apache Flink, l'état d'une application est stocké dans RockSDB, un magasin de clés/valeurs intégré qui conserve son état de fonctionnement sur le disque. Lorsqu'un point de contrôle est pris, l'état est également transféré sur Amazon S3. Ainsi, même en cas de perte du disque, le point de contrôle peut être utilisé pour restaurer l'état de l'application.

Pour de plus amples informations, veuillez consulter Fonctionnement des instantanés d'état.

Étapes du point de contrôle

Pour une sous-tâche d'opérateur de point de contrôle dans Flink, il y a 5 étapes principales :

- En attente [Délai de démarrage] : Flink utilise des barrières de point de contrôle qui sont insérées dans le flux. Ainsi, le délai à cette étape correspond au temps pendant lequel l'opérateur attend que la barrière de contrôle l'atteigne.
- Alignement [Durée de l'alignement] : à cette étape, la sous-tâche a atteint une barrière, mais elle attend les barrières provenant d'autres flux d'entrée.
- Point de contrôle synchrone [Durée de synchronisation] : c’est à cette étape que la sous-tâche crée réellement un instantané de l'état de l'opérateur et bloque toutes les autres activités de la sous-tâche.
- Point de contrôle asynchrone [Durée asynchrone] : pendant la majeure partie de cette étape, la sous-tâche télécharge l'état vers Amazon S3. Au cours de cette étape, la sous-tâche n'est plus bloquée et peut traiter des enregistrements.
- Accusé de réception : cette étape est généralement courte et consiste simplement à envoyer un accusé de réception au gestionnaire de tâches et à exécuter les messages de validation (par exemple, avec des récepteurs Kafka).

Chacune de ces étapes (à l'exception de la reconnaissance) correspond à une métrique de durée pour les points de contrôle, disponible dans l'interface utilisateur Web de Flink, qui peut aider à isoler la cause d’un long point de contrôle.

Pour voir une définition exacte de chacune des métriques disponibles sur les points de contrôle, rendez-vous dans l'onglet Historique.

Enquête

Lorsque vous étudiez la durée prolongée d'un point de contrôle, la chose la plus importante à déterminer est le goulot d'étranglement du point de contrôle, c'est-à-dire quel opérateur et quelle sous-tâche mettent le plus de temps à atteindre le point de contrôle et quelle étape de cette sous-tâche prend le plus de temps. Cela peut être déterminé à l'aide de l'interface utilisateur Web de Flink, dans la tâche de contrôle des tâches. L'interface Web de Flink fournit des données et des informations qui aident à étudier les problèmes liés aux points de contrôle. Pour une analyse complète, consultez Surveillance des points de contrôle.

La première chose à examiner est la durée de bout en bout de chaque opérateur dans le graphique des tâches afin de déterminer quel opérateur met beaucoup de temps à effectuer un point de contrôle et mérite une enquête plus approfondie. Selon la documentation de Flink, la durée est définie comme suit :

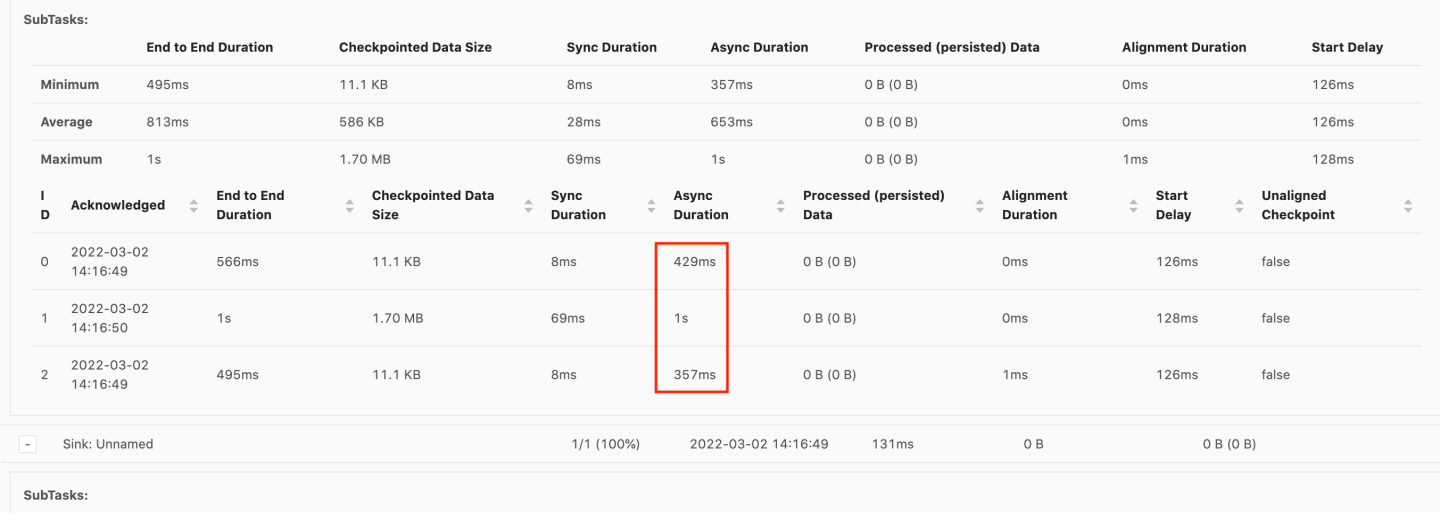
Durée comprise entre l'horodatage du déclencheur et le dernier accusé de réception (ou n/a si aucun accusé de réception n'a encore été reçu). Cette durée de bout en bout pour un point de contrôle complet est déterminée par la dernière sous-tâche qui accuse réception du point de contrôle. Ce délai est généralement supérieur à celui dont ont besoin les sous-tâches individuelles pour effectuer un point de contrôle de l'état.

Les autres durées du point de contrôle fournissent également des informations plus précises sur l'endroit où le temps est passé.

Si la durée de synchronisation est élevée, cela indique que quelque chose se passe pendant la création de l’instantané. Au cours de cette étape, `snapshotState()` est appelé pour les classes qui implémentent l'interface SnapshotState ; il peut s'agir de code utilisateur, donc les vidages de threads peuvent être utiles pour étudier cela.

Une longue durée asynchrone suggère que beaucoup de temps est consacré au chargement de l'état sur Amazon S3. Cela peut se produire si l'état est volumineux ou si de nombreux fichiers d'état sont en cours de chargement. Si tel est le cas, cela vaut la peine d'étudier la manière dont l'état est

utilisé par l'application et de s'assurer que les structures de données natives de Flink sont utilisées dans la mesure du possible (Using Keyed State). Le service géré pour Apache Flink configure Flink de manière à minimiser le nombre d'appels Amazon S3 pour garantir que cela ne soit pas trop long. Voici un exemple des statistiques de point de contrôle d'un opérateur. Cela montre que la durée asynchrone est relativement longue par rapport aux statistiques de point de contrôle des opérateurs précédentes.



SubTasks:

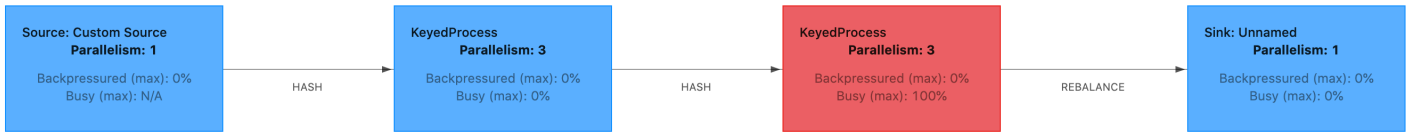
	End to End Duration	Checkpointed Data Size	Sync Duration	Async Duration	Processed (persisted) Data	Alignment Duration	Start Delay
Minimum	495ms	11.1 KB	8ms	357ms	0 B (0 B)	0ms	126ms
Average	813ms	586 KB	28ms	653ms	0 B (0 B)	0ms	126ms
Maximum	1s	1.70 MB	69ms	1s	0 B (0 B)	1ms	128ms

ID	Acknowledged	End to End Duration	Checkpointed Data Size	Sync Duration	Async Duration	Processed (persisted) Data	Alignment Duration	Start Delay	Unaligned Checkpoint
0	2022-03-02 14:16:49	566ms	11.1 KB	8ms	429ms	0 B (0 B)	0ms	126ms	false
1	2022-03-02 14:16:50	1s	1.70 MB	69ms	1s	0 B (0 B)	0ms	128ms	false
2	2022-03-02 14:16:49	495ms	11.1 KB	8ms	357ms	0 B (0 B)	1ms	126ms	false

-	Sink: Unnamed	1/1 (100%)	2022-03-02 14:16:49	131ms	0 B	0 B (0 B)

SubTasks:

Si le délai de démarrage est élevé, cela signifie que la majeure partie du temps est consacrée à attendre que la barrière du point de contrôle atteigne l'opérateur. Cela indique que l'application met un certain temps à traiter les enregistrements, ce qui signifie que l'obstacle traverse lentement le graphique des tâches. C'est généralement le cas si le la tâche est soumise à une contre-pression ou si un ou plusieurs opérateurs sont constamment occupés. Voici un exemple de graphique des tâches où le deuxième opérateur KeyedProcess est occupé.



Vous pouvez étudier ce qui prend autant de temps en utilisant les graphiques en flamme de Flink ou en utilisant les vidages de threads du gestionnaire de tâches. Une fois que le goulot d'étranglement a été identifié, il peut être étudié plus en détail à l'aide des graphiques en flamme ou des vidages de thread.

Vidages de thread

Les vidages de thread sont un autre outil de débogage légèrement inférieur à celui des graphiques en flamme. Un vidage de thread affiche l'état d'exécution de tous les threads à un moment donné. Flink effectue un vidage de thread JVM, qui est un état d'exécution de tous les threads du processus Flink. L'état d'un thread est présenté par une trace de pile du thread ainsi que par des informations supplémentaires. Les graphiques en flamme sont en fait construits à partir de plusieurs traces de pile prises en succession rapide. Le graphique est une visualisation réalisée à partir de ces traces qui permet d'identifier facilement les chemins de code courants.

```
"KeyedProcess (1/3)#0" prio=5 Id=1423 RUNNABLE
    at app//scala.collection.immutable.Range.foreach$mVc$sp(Range.scala:154)
    at $line33.$read$$iw$$iw$ExpensiveFunction.processElement(<console>>19)
    at $line33.$read$$iw$$iw$ExpensiveFunction.processElement(<console>:14)
    at app//
org.apache.flink.streaming.api.operators.KeyedProcessOperator.processElement(KeyedProcessOperat
    at app//org.apache.flink.streaming.runtime.tasks.OneInputStreamTask
$StreamTaskNetworkOutput.emitRecord(OneInputStreamTask.java:205)
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.processElement(AbstractStr
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.emitNext(AbstractStreamTas
    at app//
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.processInput(StreamOneInputProces
    ...
```

Vous trouverez ci-dessus un extrait d'un vidage de thread issu de l'interface utilisateur de Flink pour un seul thread. La première ligne contient des informations générales sur ce thread, notamment :

- Le nom du thread KeyedProcess (1/3) #0
- La priorité du thread prio=5
- Un identifiant de thread unique Id=1423
- L’état du thread : EXÉCUTABLE

Le nom d'un thread donne généralement des informations sur son usage général. Les threads d'opérateur peuvent être identifiés par leur nom, puisqu'ils portent le même nom que l'opérateur, ainsi qu'une indication de la sous-tâche à laquelle ils sont liés. Par exemple, le thread KeyedProcess (1/3) #0 provient de l'opérateur KeyedProcess et de la première sous-tâche (sur 3).

Les threads peuvent avoir l'un des états suivants :

- NOUVEAU : le thread a été créé mais n'a pas encore été traité
- EXÉCUTABLE : le thread est en cours d'exécution sur le processeur
- BLOQUÉ : le thread attend qu'un autre thread lève son verrou
- EN ATTENTE : le thread attend à l'aide d'une méthode `wait()`, `join()` ou `park()`
- EN ATTENTE PROGRAMMÉE : le thread attend en utilisant une méthode veille, attendre, rejoindre ou parquer, mais avec un temps d'attente maximal.

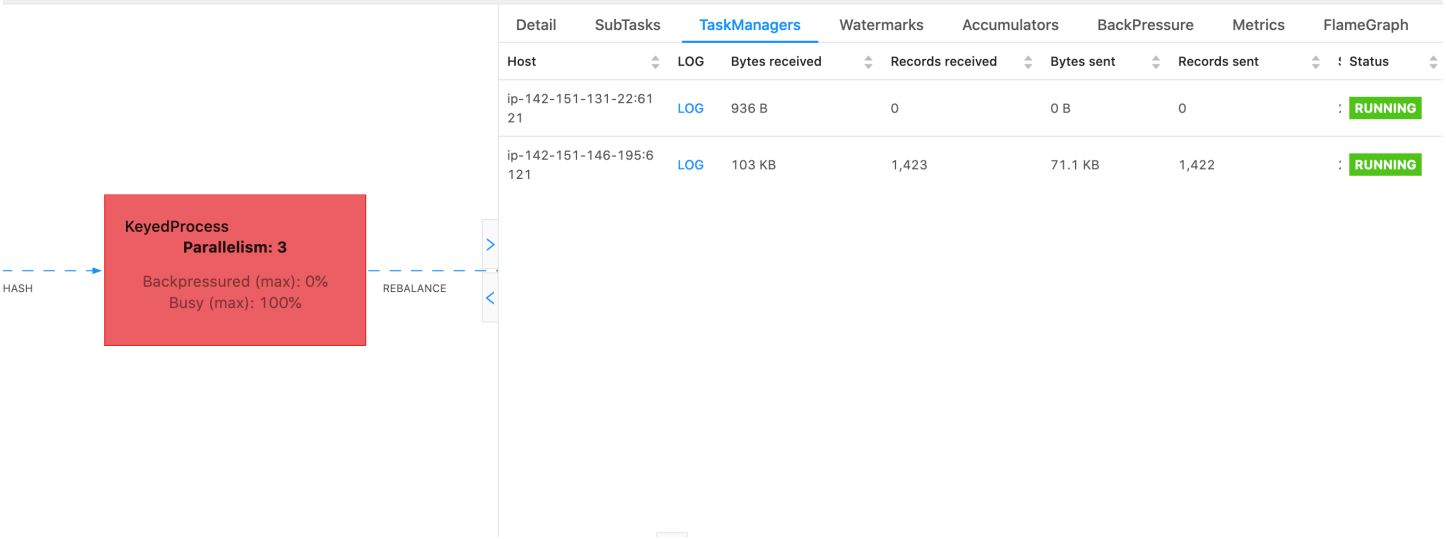
> **Note**
>
> Dans Flink 1.13, la profondeur maximale d'une trace de pile dans le vidage de thread est limitée à 8.

> **Note**
>
> Les vidages de thread doivent être le dernier recours pour résoudre les problèmes de performances dans une application Flink, car ils peuvent être difficiles à lire et nécessiter le prélèvement de plusieurs échantillons et leur analyse manuelle. Dans la mesure du possible, il est préférable d'utiliser des graphiques en flamme.

Vidages de thread dans Flink

Dans Flink, un vidage de thread peut être effectué en choisissant l'option Gestionnaires de tâches dans la barre de navigation à gauche de l'interface utilisateur de Flink, en sélectionnant un gestionnaire de tâches spécifique, puis en accédant à l'onglet Thread Dump. Le vidage de thread peut être téléchargé, copié dans votre éditeur de texte préféré (ou analyseur de vidage de thread) ou analysé directement dans l'affichage en texte de l'interface utilisateur Web de Flink (cette dernière option peut toutefois s'avérer un peu compliquée).

Pour déterminer dans quel gestionnaire de tâches utiliser un vidage de thread, il est possible d'utiliser l'onglet Gestionnaires de tâches lorsqu'un opérateur particulier est sélectionné. Cela montre que l'opérateur exécute différentes sous-tâches d'un opérateur et qu'il peut s'exécuter sur différents gestionnaires de tâches.



Le vidage sera composé de plusieurs traces de pile. Cependant, lors de l'enquête sur le vidage, les informations relatives à un opérateur sont les plus importantes. Elles sont faciles à trouver, puisque les threads d'opérateurs portent le même nom que l'opérateur et indiquent à quelle sous-tâche ils sont liés. Par exemple, la trace de pile suivante provient de l'opérateur KeyedProcess et constitue la première sous-tâche.

```
"KeyedProcess (1/3)#0" prio=5 Id=595 RUNNABLE
    at app//scala.collection.immutable.Range.foreach$mVc$sp(Range.scala:155)
    at $line360.$read$$iw$$iw$ExpensiveFunction.processElement(<console>:19)
    at $line360.$read$$iw$$iw$ExpensiveFunction.processElement(<console>:14)
    at app//
org.apache.flink.streaming.api.operators.KeyedProcessOperator.processElement(KeyedProcessOperat
    at app//org.apache.flink.streaming.runtime.tasks.OneInputStreamTask
$StreamTaskNetworkOutput.emitRecord(OneInputStreamTask.java:205)
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.processElement(AbstractStr
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.emitNext(AbstractStreamTas
    at app//
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.processInput(StreamOneInputProces
    ...
```

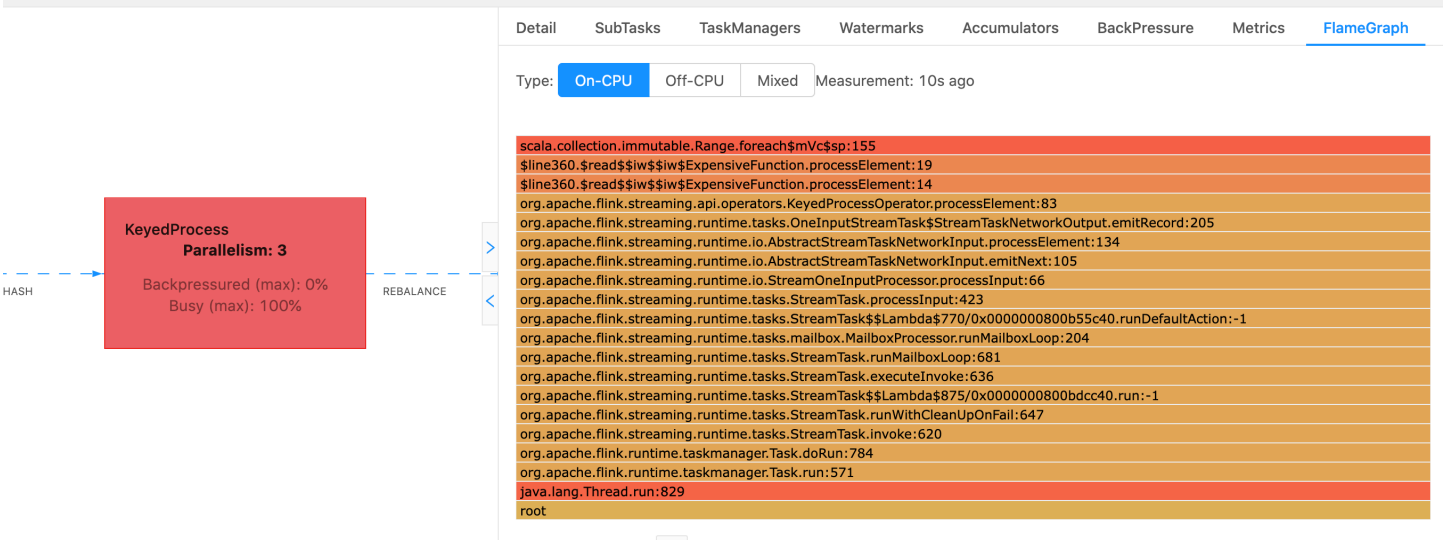
Cela peut prêter à confusion s'il existe plusieurs opérateurs portant le même nom, mais nous pouvons nommer les opérateurs pour contourner ce problème. Par exemple :

```
....
.process(new ExpensiveFunction).name("Expensive function")
```

Graphiques en flamme

Les graphiques en flamme constituent un outil de débogage utile qui permet de visualiser les traces du code ciblé, ce qui permet d'identifier les chemins de code les plus fréquents. Ils sont créés en échantillonnant plusieurs fois des traces de pile. L'axe X d'un graphique en flamme montre les différents profils de pile, tandis que l'axe Y indique la profondeur de la pile et les appels dans la trace de pile. Un seul rectangle dans un graphique en flamme représente un cadre de pile, et la largeur d'un cadre indique la fréquence à laquelle il apparaît dans les piles. Pour plus de détails sur les graphiques en flamme et sur leur utilisation, consultez Graphiques en flamme.

Dans Flink, le graphique en flamme d'un opérateur est accessible via l'interface utilisateur Web, en sélectionnant un opérateur, puis en choisissant l'onglet FlameGraph. Une fois que suffisamment d'échantillons ont été prélevés, le graphique en flamme s'affiche. Voici le graphique en flamme de la fonction du processus dont le point de contrôle prenait beaucoup de temps.



Il s'agit d'un graphique en flamme très simple qui montre que tout le temps du processeur est consacré à un examen foreach dans le cadre du `processElement` de l'opérateur ExpensiveFunction. Vous obtenez également le numéro de ligne qui aide à déterminer l'endroit où l'exécution du code a lieu.

Le délai du point de contrôle est expiré

Si votre application n'est pas optimisée ou correctement provisionnée, les points de contrôle peuvent échouer. Cette section décrit les symptômes et les étapes de résolution de ce problème.

Symptômes

Si les points de contrôle échouent pour votre application, le `numberOfFailedCheckpoints` sera supérieur à zéro.

Les points de contrôle peuvent échouer soit en raison de d'échecs directs, telles que des erreurs d'application, soit en raison d'échecs temporaires, tels que l'épuisement des ressources de l'application. Vérifiez les journaux et les métriques de votre application pour détecter les symptômes suivants :

- Des erreurs dans votre code.
- Des erreurs lors de l'accès aux services dépendants de votre application.
- Des erreurs lors de la sérialisation des données. Si le sérialiseur par défaut ne parvient pas à sérialiser les données de votre application, celle-ci échouera. Pour plus d'informations sur l'utilisation d'un sérialiseur personnalisé dans votre application, consultez la section Custom Serializers dans la documentation d'Apache Flink.
- Erreurs de mémoire insuffisante.
- Des pics ou des augmentations constantes des métriques suivantes :
  - `heapMemoryUtilization`
  - `oldGenerationGCTime`
  - `oldGenerationGCCount`
  - `lastCheckpointSize`
  - `lastCheckpointDuration`

Pour de plus amples informations sur la surveillance des points de contrôle, consultez Monitoring Checkpointing dans la documentation Apache Flink.

Causes et solutions

Les messages d'erreur du journal de votre application indiquent la cause des échecs directs. Les échecs temporaires peuvent avoir les causes suivantes :

- Le provisionnement en KPU de votre application est insuffisant. Pour plus d'informations sur l'augmentation du provisionnement de l'application, consultez Mise à l'échelle.
- La taille de l'état de votre application est trop grande. Vous pouvez surveiller la taille de l'état de votre application à l'aide de la métrique `lastCheckpointSize`.
- Les données d'état de votre application sont réparties de manière inégale entre les clés. Si votre application utilise l'opérateur `KeyBy`, assurez-vous que les données entrantes sont réparties de manière égale entre les clés. Si la plupart des données sont attribuées à une seule clé, cela crée un goulot d'étranglement, qui entraînera des échecs.
- Votre application est confrontée à une contre-pression liée à la mémoire ou au récupérateur de mémoire. Vérifiez s'il y a des pics ou des augmentations constantes des valeurs `heapMemoryUtilization`, `oldGenerationGCTime` et `oldGenerationGCCount`.

Échec de point de contrôle pour l'application Apache Beam

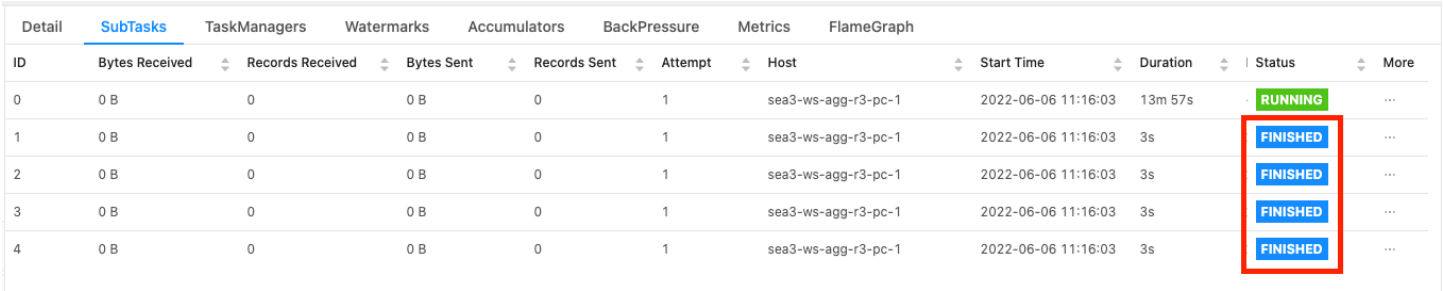
Si votre application Beam est configurée avec shutdownSourcesAfterIdleMs défini sur 0 ms, les points de contrôle peuvent ne pas se déclencher car les tâches sont à l'état « TERMINÉ ». Cette section décrit les symptômes et la résolution de ce problème.

Symptôme

Accédez aux journaux CloudWatch de votre application de service géré pour Apache Flink et vérifiez si le message de journal suivant a été enregistré. Le message de journal suivant indique que le point de contrôle n'a pas pu être déclenché, car certaines tâches sont terminées.

```
               {
               "locationInformation":
"org.apache.flink.runtime.checkpoint.CheckpointCoordinator.onTriggerFailure(CheckpointCoordina
               "logger": "org.apache.flink.runtime.checkpoint.CheckpointCoordinator",
               "message": "Failed to trigger checkpoint for job your job ID since some
tasks of job your job ID has been finished, abort the checkpoint Failure reason: Not
all required tasks are currently running.",
               "threadName": "Checkpoint Timer",
               "applicationARN": your application ARN,
               "applicationVersionId": "5",
               "messageSchemaVersion": "1",
               "messageType": "INFO"
               }
```

Vous pouvez également le voir sur le tableau de bord de Flink, où certaines tâches sont passées à l'état « TERMINÉ » et où le point de contrôle n'est plus possible.



Cause

shutdownSourcesAfterIdleMs est une variable de configuration Beam qui arrête les sources inactives pendant la durée configurée de millisecondes. Une fois qu'une source a été fermée, le point de contrôle n'est plus possible. Cela pourrait entraîner un échec du point de contrôle.

L'une des raisons pour lesquelles les tâches passent à l'état « TERMINÉ » est lorsque shutdownSourcesAfterIdleMs est défini sur 0 ms, ce qui signifie que les tâches inactives seront immédiatement arrêtées.

Solution

Pour empêcher les tâches de passer immédiatement à l'état « TERMINÉ », définissez shutdownSourcesAfterIdleMs sur Long.MAX_VALUE. Vous pouvez effectuer cette opération de deux façons :

- Option 1 : si votre configuration Beam est définie sur la page de configuration de votre application de service géré pour Apache Flink, vous pouvez ajouter une nouvelle paire clé/valeur pour définir shutdPwnSourcesAfterIdleMs comme suit :

- Option 2 : si votre configuration Beam est définie dans votre fichier JAR, vous pouvez définir shutdownSourcesAfterIdleMs comme suit :

```
                        FlinkPipelineOptions options =
PipelineOptionsFactory.create().as(FlinkPipelineOptions.class); // Initialize Beam
Options object

                        options.setShutdownSourcesAfterIdleMs(Long.MAX_VALUE); // set
shutdownSourcesAfterIdleMs to Long.MAX_VALUE
                        options.setRunner(FlinkRunner.class);

                        Pipeline p = Pipeline.create(options); // attach specified
options to Beam pipeline
```

Contre-pression

Flink utilise la contre-pression pour adapter la vitesse de traitement de chaque opérateur.

L'opérateur peut avoir du mal à traiter le volume de messages qu'il reçoit pour de nombreuses raisons. L'opération peut nécessiter plus de ressources du processeur que celles dont dispose l'opérateur. L'opérateur peut attendre la fin des opérations d'I/O. Si un opérateur ne parvient pas à traiter les événements assez rapidement, cela crée une contre-pression chez les opérateurs en amont qui alimentent l'opérateur lent. Cela ralentit les opérateurs en amont, ce qui peut propager davantage la contre-pression vers la source et amener la source à s'adapter au débit global de l'application en ralentissant également. Vous trouverez une description plus détaillée de la contre-pression et de son fonctionnement dans How Apache Flink™ handles backpressure.

Le fait de savoir quels opérateurs d'une application sont lents vous fournit des informations cruciales pour comprendre la cause première des problèmes de performances de l'application. Les informations sur la contre-pression sont affichées via le tableau de bord Flink. Pour identifier l'opérateur lent, recherchez l'opérateur présentant une valeur de contre-pression élevée qui est le plus proche d'un récepteur (l'opérateur B dans l'exemple suivant). L'opérateur à l'origine de la lenteur est alors l'un des opérateurs en aval (l'opérateur C dans l'exemple). L'opérateur B peut traiter les événements plus rapidement, mais il est soumis à une contre-pression, car il ne peut pas transmettre le résultat à l'opérateur C lent.

```
A (backpressured 93%) -> B (backpressured 85%) -> C (backpressured 11%) -> D
 (backpressured 0%)
```

Une fois que vous avez identifié l'opérateur lent, essayez de comprendre pourquoi il est lent. Il peut y avoir une multitude de raisons. Parfois, le problème n'est pas évident et peut nécessiter plusieurs

jours de débogage et de profilage pour être résolu. Voici quelques raisons évidentes et courantes, dont certaines sont expliquées plus en détail ci-dessous :

- L'opérateur effectue des opérations d'I/O lentes, par exemple des appels réseau (pensez plutôt à utiliser AsyncIO à la place).
- Il y a une asymétrie dans les données et un opérateur reçoit plus d'événements que les autres (vérifiez en regardant le nombre de messages entrée/sortie de sous-tâches individuelles (c'est-à-dire les instances du même opérateur) dans le tableau de bord Flink.
- Il s'agit d'une opération gourmande en ressources (s'il n'y a pas d'asymétrie des données, envisagez de monter la puissance du processeur/de la mémoire ou d'augmenter le `ParallelismPerKPU` pour le travail lié aux I/O)
- Journalisation étendue dans l'opérateur (réduisez la journalisation au minimum pour les applications de production ou envisagez plutôt d'envoyer les résultats de débogage vers un flux de données).

Test de débit avec Discarding Sink

Discarding Sink ignore simplement tous les événements qu'il reçoit pendant l'exécution de l'application (une application sans récepteur ne s'exécute pas). Cela est très utile pour les tests de débit, le profilage et pour vérifier si l'application est correctement mise à l'échelle. Il s'agit également d'un contrôle de santé très pragmatique pour vérifier si les récepteurs sont à l'origine de la contre-pression ou cela vient de l'application (mais il est souvent plus facile et plus simple de vérifier les métriques de contre-pression).

En remplaçant tous les récepteurs d'une application par un récepteur de rejet et en créant une source fictive qui génère des données similaires aux données de production, vous pouvez mesurer le débit maximal de l'application pour un certain paramètre de parallélisme. Vous pouvez également augmenter le parallélisme pour vérifier que l'application évolue correctement et qu'elle ne présente pas de goulot d'étranglement qui n'apparaît qu'à un débit plus élevé (par exemple, en raison d'une asymétrie de données).

Asymétrie de données

Une application Flink est exécutée sur un cluster de manière distribuée. Pour s'étendre à plusieurs nœuds, Flink utilise le concept de flux liés à une clé, ce qui signifie essentiellement que les événements d'un flux sont partitionnés en fonction d'une clé spécifique, par exemple l'identifiant du client, et Flink peut ensuite traiter différentes partitions sur différents nœuds. De nombreux opérateurs

Flink sont ensuite évalués en fonction de ces partitions, par exemple Keyed Windows, Process Functions et Async I/O.

Le choix d'une clé de partition dépend souvent de la logique métier. Dans le même temps, bon nombre des bonnes pratiques, par exemple pour DynamoDB et Spark, s'appliquent également à Flink, notamment :

- garantir une cardinalité élevée des clés de partition ;
- éviter toute asymétrie du volume d'événements entre les partitions.

Vous pouvez identifier l'asymétrie dans les partitions en comparant les enregistrements reçus/envoyés des sous-tâches (c'est-à-dire les instances du même opérateur) dans le tableau de bord Flink. En outre, la surveillance du service géré pour Apache Flink peut également être configurée pour exposer des métriques au niveau de `numRecordsIn/Out` et `numRecordsInPerSecond/OutPerSecond` au niveau des sous-tâches.

Asymétrie d'état

Pour les opérateurs dynamiques, c'est-à-dire les opérateurs qui maintiennent l'état de leur logique métier, comme les fenêtres, l’asymétrie de données entraîne toujours une asymétrie d'état. Certaines sous-tâches reçoivent plus d'événements que d'autres en raison de l'asymétrie des données et conservent donc un plus grand nombre de données dans leur état. Cependant, même pour une application dont les partitions sont équilibrées, il peut y avoir une asymétrie dans la quantité de données conservées dans leur état. Par exemple, pour les fenêtres de session, certains utilisateurs et certaines sessions peuvent respectivement être beaucoup plus longs que d'autres. Si les sessions les plus longues font partie de la même partition, cela peut entraîner un déséquilibre de la taille des états conservés par les différentes sous-tâches du même opérateur.

L'asymétrie d'état augmente non seulement la mémoire et les ressources disque requises par les sous-tâches individuelles, mais elle peut également diminuer les performances globales de l'application. Lorsqu'une application atteint un point de contrôle ou un point de sauvegarde, l'état de l'opérateur est conservé dans Amazon S3, afin de protéger l'état contre l’échec d'un nœud ou d'un cluster. Au cours de ce processus (en particulier avec la sémantique unique activée par défaut sur le service géré pour Apache Flink), le traitement s'arrête d'un point de vue externe jusqu'à ce que le point de contrôle/de sauvegarde soit terminé. En cas d’asymétrie des données, le temps nécessaire pour terminer l'opération peut être limité par une seule sous-tâche ayant accumulé un nombre d'états particulièrement élevé. Dans les cas extrêmes, la prise de points de contrôle/de sauvegarde peut échouer parce qu'une seule sous-tâche ne parvient pas à conserver son état.

Tout comme l’asymétrie de données, l’asymétrie d'état peut considérablement ralentir une application.

Pour identifier une asymétrie d'état, vous pouvez utiliser le tableau de bord Flink. Trouvez un point de contrôle ou de sauvegarde récent et comparez la quantité de données stockées pour chaque sous-tâche dans les détails.

Intégration de ressources de différentes régions

Vous pouvez activer l'utilisation de `StreamingFileSink` pour écrire dans un compartiment Amazon S3 situé dans une région différente de celle de votre application de service géré pour Apache Flink via un paramètre requis pour la réplication entre régions dans la configuration Flink. Pour ce faire, envoyez un ticket d'assistance au Centre AWS Support.

Historique du document pour le service géré Amazon pour Apache Flink

Le tableau suivant décrit les modifications importantes apportées à la documentation depuis la dernière version du service géré pour Apache Flink.

- Version de l'API : 2018-05-23
- Dernière mise à jour de la documentation : 30 août 2023

Modification	Description	Date
Kinesis Data Analytics est désormais connu sous le nom de service géré pour Apache Flink	Aucune modification n'est apportée aux points de terminaison du service, aux API, à l'interface de ligne de commande, aux politiques d'accès IAM, aux CloudWatch métriques ou aux tableaux de bord de AWS facturation. Vos applications existantes continueront de fonctionner comme auparavant. Pour plus d'informations, voir Qu'est-ce qu'un service géré pour Apache Flink ?	30 août 2023
Prise en charge d’Apache Flink version 1.15.2.	Le service géré pour Apache Flink prend désormais en charge les applications qui utilisent Apache Flink version 1.15.2. Créez des applications Kinesis Data Analytics à l'aide de l'API de table Apache Flink. Pour	22 novembre 2022

Modification	Description	Date
	plus d'informations, consultez Création d'applications.	
Prise en charge d'Apache Flink version 1.13.2.	Le service géré pour Apache Flink prend désormais en charge les applications qui utilisent Apache Flink version 1.13.2. Créez des applications Kinesis Data Analytics à l'aide de l'API de table Apache Flink. Pour plus d'informations, consultez Mise en route : Flink 1.13.2.	13 octobre 2021
Prise en charge de Python	Le service géré pour Apache Flink prend désormais en charge les applications qui utilisent Python avec l'API de table Apache Flink et SQL. Pour plus d'informations, consultez Utilisation de Python.	25 mars 2021
Prise en charge d'Apache Flink 1.11.1	Le service géré pour Apache Flink prend désormais en charge les applications qui utilisent Apache Flink version 1.11.1. Créez des applications Kinesis Data Analytics à l'aide de l'API de table Apache Flink. Pour plus d'informations, consultez Création d'applications.	19 novembre 2020

Modification	Description	Date
Tableau de bord Apache Flink	Utilisez le tableau de bord Apache Flink pour surveille r l'état et les performances des applications. Pour plus d'informations, consultez Tableau de bord Apache Flink.	19 novembre 2020
Consommateur EFO	Créez des applications qui utilisent un consommateur Enhanced Fan-Out (EFO) pour lire à partir d'un flux de données Kinesis. Pour plus d'informations, consultez Consommateur EFO.	6 octobre 2020
Apache Beam	Créez des applications qui utilisent Apache Beam pour traiter les données de streaming. Pour plus d'informa tions, consultez Utilisation de CloudFormation avec le service géré pour Apache Flink.	15 septembre 2020
Performance	Comment résoudre les problèmes de performan ce des applications et comment créer une applicati on performante. Pour plus d'informations, consultez Performances.	21 juillet 2020

Modification	Description	Date
Keystore personnalisé	Comment accéder à un cluster Amazon MSK qui utilise un keystore personnalisé pour le chiffrement en transit. Pour plus d'informations, consultez Truststore personnalisé.	10 juin 2020
CloudWatch Alarmes	Recommandations pour créer des CloudWatch alarmes avec le service géré pour Apache Flink. Pour plus d'informations, consultez Alertes.	5 juin 2020
Nouvelles CloudWatch métriques	Le service géré pour Apache Flink envoie désormais 22 métriques à Amazon CloudWatch Metrics. Pour plus d'informations, consultez Métriques et dimensions dans le service géré pour Apache Flink.	12 mai 2020
CloudWatch Métriques personnalisées	Définissez des métriques spécifiques à l'application et transmettez-les à Amazon CloudWatch Metrics. Pour plus d'informations, consultez Métriques personnalisées.	12 mai 2020

Modification	Description	Date
Exemple : lire à partir d'un flux Kinesis dans un autre compte	Découvrez comment accéder à un flux Kinesis depuis un autre compte AWS dans votre application de service géré pour Apache Flink. Pour plus d'informations, consultez Intercompte.	30 mars 2020
Prise en charge d'Apache Flink 1.8.2	Le service géré pour Apache Flink prend désormais en charge les applications qui utilisent Apache Flink version 1.8.2. Utilisez le StreamingFileSink connecteu r Flink pour écrire la sortie directement dans S3. Pour plus d'informations, consultez Création d'applications.	17 décembre 2019
VPC pour le service géré pour Apache Flink	Configurez un service géré pour l'application Apache Flink afin de se connecter à un cloud privé virtuel. Pour plus d'informations, consultez Utilisation d'un VPC Amazon.	25 novembre 2019
Bonnes pratiques pour le service géré pour Apache Flink	Bonnes pratiques pour la création et l'administration des applications de service géré pour Apache Flink. Pour plus d'informations, consultez Bonnes pratiques.	14 octobre 2019

Modification	Description	Date
Analyse des journaux d’application dans le service géré pour Apache Flink.	Utilisez CloudWatch Logs Insights pour surveiller votre service géré pour l'application Apache Flink. Pour plus d’informations, consultez Analyse des journaux.	26 juin 2019
Propriétés d’exécution du service géré pour l'application Apache Flink	Utilisez les propriétés d’exécution dans le service géré pour Apache Flink Pour plus d’informations, consultez Propriétés d'exécution.	24 juin 2019
Balisage des applications de service géré pour Apache Flink	Utilisez le balisage d'application pour déterminer les coûts par application, pour contrôler les accès, ou à des fins définis par l'utilisateur. Pour plus d’informations, consultez Utilisation du balisage.	8 mai 2019
Exemples d’applications de service géré pour Apache Flink	Exemples d'applications pour Managed Service for Apache Flink illustrant des opérateurs de fenêtres et écrivant des résultats dans des CloudWatch journaux. Pour plus d’informations, consultez Exemples.	1er mai 2019

Modification	Description	Date
Appels d'API avec AWS CloudTrail pour la journalisation du service géré pour Apache Flink	Le service géré pour Apache Flink est intégré à AWS CloudTrail, un service qui enregistre les actions effectuées par un utilisateur, un rôle ou un service AWS dans le service géré pour Apache Flink. Pour plus d'informations, consultez Utiliser AWS CloudTrail.	22 mars 2019
Création d'une application (récepteur Kinesis Data Firehose)	Exercice de création d'un service géré pour Apache Flink qui a un flux de données Amazon Kinesis comme source et un flux de diffusion Amazon Kinesis Data Firehose comme récepteur. Pour plus d'informations, consultez Récepteur Kinesis Data Firehose.	13 décembre 2018
Publication	Il s'agit de la première version du guide du développeur du service géré pour Apache Flink pour les applications Flink.	27 novembre 2018

Exemple de code pour d’API pour le service géré pour Apache Flink

Cette rubrique contient un exemple de blocs de requêtes pour les actions du service géré pour Apache Flink.

Pour utiliser JSON comme entrée pour une action avec AWS Command Line Interface (AWS CLI), enregistrez la demande dans un fichier JSON. Transmettez ensuite le nom du fichier à l'action à l'aide du paramètre `--cli-input-json`.

L'exemple suivant montre comment utiliser un fichier JSON avec une action.

```
$ aws kinesisanalyticsv2 start-application --cli-input-json file://start.json
```

Pour plus d'informations sur l'utilisation de JSON avec l’interface AWS CLI, consultez Génération des paramètres de squelette de CLI et JSON en entrée de la CLI dans le guide de l'utilisateur AWS Command Line Interface.

Rubriques

- AddApplicationCloudWatchLoggingOption
- AddApplicationInput
- AddApplicationInputProcessingConfiguration
- AddApplicationOutput
- AddApplicationReferenceDataSource
- AddApplicationVpcConfiguration
- CreateApplication
- CreateApplicationSnapshot
- DeleteApplication
- DeleteApplicationCloudWatchLoggingOption
- DeleteApplicationInputProcessingConfiguration
- DeleteApplicationOutput
- DeleteApplicationReferenceDataSource
- DeleteApplicationSnapshot

- DeleteApplicationVpcConfiguration
- DescribeApplication
- DescribeApplicationSnapshot
- DiscoverInputSchema
- ListApplications
- ListApplicationSnapshots
- StartApplication
- StopApplication
- UpdateApplication

AddApplicationCloudWatchLoggingOption

L'exemple de code de demande suivant pour l'action AddApplicationCloudWatchLoggingOption ajoute une option de journalisation Amazon CloudWatch à une application de service géré pour Apache Flink :

```
{
    "ApplicationName": "MyApplication",
    "CloudWatchLoggingOption": {
        "LogStreamARN": "arn:aws:logs:us-east-1:123456789123:log-group:my-log-
group:log-stream:My-LogStream"
    },
    "CurrentApplicationVersionId": 2
}
```

AddApplicationInput

L'exemple de code de demande suivant pour l'action AddApplicationInput ajoute une entrée d'application à une application de service géré pour Apache Flink :

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 2,
   "Input": {
      "InputParallelism": {
         "Count": 2
```

```
      },
      "InputSchema": {
         "RecordColumns": [
            {
               "Mapping": "$.TICKER",
               "Name": "TICKER_SYMBOL",
               "SqlType": "VARCHAR(50)"
            },
            {
                "SqlType": "REAL",
                "Name": "PRICE",
                "Mapping": "$.PRICE"
            }
         ],
         "RecordEncoding": "UTF-8",
         "RecordFormat": {
            "MappingParameters": {
               "JSONMappingParameters": {
                  "RecordRowPath": "$"
               }
            },
            "RecordFormatType": "JSON"
         }
      },
      "KinesisStreamsInput": {
         "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleInputStream"
      }
   }
}
```

AddApplicationInputProcessingConfiguration

L'exemple de code de demande suivant pour l'action AddApplicationInputProcessingConfiguration ajoute une configuration de traitement des entrées d'application à une application de service géré pour Apache Flink :

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 2,
   "InputId": "2.1",
   "InputProcessingConfiguration": {
```

```
      "InputLambdaProcessor": {
         "ResourceARN": "arn:aws:lambda:us-
east-1:012345678901:function:MyLambdaFunction"
      }
   }
}
```

AddApplicationOutput

L'exemple de code de demande suivant pour l'action AddApplicationOutput ajoute un flux de données Kinesis en tant que sortie d'application à une application de service géré pour Apache Flink :

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 2,
   "Output": {
      "DestinationSchema": {
         "RecordFormatType": "JSON"
      },
      "KinesisStreamsOutput": {
         "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleOutputStream"
      },
      "Name": "DESTINATION_SQL_STREAM"
   }
}
```

AddApplicationReferenceDataSource

L'exemple de code de demande suivant pour l'action AddApplicationReferenceDataSource ajoute une source de données de référence d'application à une application de service géré pour Apache Flink :

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 5,
   "ReferenceDataSource": {
      "ReferenceSchema": {
         "RecordColumns": [
            {
```

```
                "Mapping": "$.TICKER",
                "Name": "TICKER",
                "SqlType": "VARCHAR(4)"
             },
             {
                "Mapping": "$.COMPANYNAME",
                "Name": "COMPANY_NAME",
                "SqlType": "VARCHAR(40)"
             },
          ],
          "RecordEncoding": "UTF-8",
          "RecordFormat": {
             "MappingParameters": {
                "CSVMappingParameters": {
                   "RecordColumnDelimiter": " ",
                   "RecordRowDelimiter": "\r\n"
                }
             },
             "RecordFormatType": "CSV"
          }
       },
       "S3ReferenceDataSource": {
          "BucketARN": "arn:aws:s3:::MyS3Bucket",
          "FileKey": "TickerReference.csv"
       },
       "TableName": "string"
    }
}
```

AddApplicationVpcConfiguration

L'exemple de code de demande suivant pour l'action AddApplicationVpcConfiguration ajoute une configuration de VPC à une application existante :

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfiguration": {
      "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
      "SubnetIds": [ "subnet-0123456789abcdef0" ]
   }
}
```

CreateApplication

L'exemple de code de demande suivant pour l'action CreateApplication crée une application de service géré pour Apache Flink :

```
{
  "ApplicationName":"MyApplication",
  "ApplicationDescription":"My-Application-Description",
  "RuntimeEnvironment":"FLINK-1_15",
  "ServiceExecutionRole":"arn:aws:iam::123456789123:role/myrole",
  "CloudWatchLoggingOptions":[
    {
      "LogStreamARN":"arn:aws:logs:us-east-1:123456789123:log-group:my-log-group:log-
stream:My-LogStream"
    }
  ],
  "ApplicationConfiguration": {
    "EnvironmentProperties":
      {"PropertyGroups":
        [
          {"PropertyGroupId": "ConsumerConfigProperties",
            "PropertyMap":
              {"aws.region": "us-east-1",
              "flink.stream.initpos": "LATEST"}
          },
          {"PropertyGroupId": "ProducerConfigProperties",
            "PropertyMap":
              {"aws.region": "us-east-1"}
          },
        ]
      },
    "ApplicationCodeConfiguration":{
      "CodeContent":{
        "S3ContentLocation":{
          "BucketARN":"arn:aws:s3:::mybucket",
          "FileKey":"myflink.jar",
          "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "CodeContentType":"ZIPFILE"
```

```
    },
      "FlinkApplicationConfiguration":{
      "ParallelismConfiguration":{
        "ConfigurationType":"CUSTOM",
        "Parallelism":2,
        "ParallelismPerKPU":1,
        "AutoScalingEnabled":true
      }
    }
  }
}
```

CreateApplicationSnapshot

L'exemple de code de demande suivant pour l'action CreateApplicationSnapshot crée un instantané de l'état de l'application :

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MySnapshot"
}
```

DeleteApplication

L'exemple de code de demande suivant pour l'action DeleteApplication supprime une application de service géré pour Apache Flink :

```
{"ApplicationName": "MyApplication",
"CreateTimestamp": 12345678912}
```

DeleteApplicationCloudWatchLoggingOption

L'exemple de code de demande suivant pour l'action DeleteApplicationCloudWatchLoggingOption supprime une option de journalisation Amazon CloudWatch pour une application de service géré pour Apache Flink :

```
{
    "ApplicationName": "MyApplication",
```

```
    "CloudWatchLoggingOptionId": "3.1"
    "CurrentApplicationVersionId": 3
}
```

DeleteApplicationInputProcessingConfiguration

L'exemple de code de demande suivant pour l'action DeleteApplicationInputProcessingConfiguration supprime une configuration de traitement des entrées d'une application de service géré pour Apache Flink :

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "InputId": "2.1"
}
```

DeleteApplicationOutput

L'exemple de code de demande suivant pour l'action DeleteApplicationOutput supprime une sortie d'application d'une application de service géré pour Apache Flink :

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "OutputId": "4.1"
}
```

DeleteApplicationReferenceDataSource

L'exemple de code de demande suivant pour l'action DeleteApplicationReferenceDataSource supprime une source de données de référence d'application d'une application de service géré pour Apache Flink :

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 5,
   "ReferenceId": "5.1"
}
```

DeleteApplicationSnapshot

L'exemple de code de demande suivant pour l'action DeleteApplicationSnapshot supprime un instantané d'état d'application :

```
{
   "ApplicationName": "MyApplication",
   "SnapshotCreationTimestamp": 12345678912,
   "SnapshotName": "MySnapshot"
}
```

DeleteApplicationVpcConfiguration

L'exemple de code de demande suivant pour l'action DeleteApplicationVpcConfiguration supprime une configuration VPC existante d'une application :

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfigurationId": "1.1"
}
```

DescribeApplication

L'exemple de code de demande suivant pour l'action DescribeApplication renvoie des informations sur une application de service géré pour Apache Flink :

```
{"ApplicationName": "MyApplication"}
```

DescribeApplicationSnapshot

L'exemple de code de demande suivant pour l'action DescribeApplicationSnapshot renvoie des informations sur instantané de l'état d'une application :

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MySnapshot"
```

```
}
```

DiscoverInputSchema

L'exemple de code de demande suivant pour l'action DiscoverInputSchema génère un schéma à partir d'une source de streaming :

```
{
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "arn:aws:lambda:us-
east-1:012345678901:function:MyLambdaFunction"
      }
   },
   "InputStartingPositionConfiguration": {
      "InputStartingPosition": "NOW"
   },
   "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/ExampleInputStream",
   "S3Configuration": {
      "BucketARN": "string",
      "FileKey": "string"
   },
   "ServiceExecutionRole": "string"
}
```

L'exemple de code de demande suivant pour l'action DiscoverInputSchema génère un schéma à partir d'une source de référence :

```
{
   "S3Configuration": {
      "BucketARN": "arn:aws:s3:::mybucket",
      "FileKey": "TickerReference.csv"
   },
   "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole"
}
```

ListApplications

L'exemple de code de requête suivant pour l'action ListApplications renvoie une liste des applications de service géré pour Apache Flink dans votre compte :

```
{
   "ExclusiveStartApplicationName": "MyApplication",
   "Limit": 50
}
```

ListApplicationSnapshots

L'exemple de code de demande suivant pour l'action ListApplicationSnapshots renvoie une liste d'instantanés de l'état de l'application :

```
{"ApplicationName": "MyApplication",
   "Limit": 50,
   "NextToken": "aBcDeFgHiJkLmNoPqRsTuVwXyZ0123"
}
```

StartApplication

L'exemple de code de demande suivant pour l'action StartApplication démarre une application Managed Service for Apache Flink et charge l'état de l'application à partir du dernier instantané (le cas échéant) :

```
{
    "ApplicationName": "MyApplication",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

StopApplication

L'exemple de code de demande suivant pour l'action API_StopApplication arrête une application de service géré pour Apache Flink :

```
{"ApplicationName": "MyApplication"}
```

UpdateApplication

L'exemple de code de demande suivant pour l'action UpdateApplication met à jour une application de service de géré pour Apache Flink afin de modifier l'emplacement du code de l'application :

```
{"ApplicationName": "MyApplication",
"CurrentApplicationVersionId": 1,
"ApplicationConfigurationUpdate": {
      "ApplicationCodeConfigurationUpdate": {
         "CodeContentTypeUpdate": "ZIPFILE",
         "CodeContentUpdate": {
            "S3ContentLocationUpdate": {
               "BucketARNUpdate": "arn:aws:s3:::my_new_bucket",
               "FileKeyUpdate": "my_new_code.zip",
               "ObjectVersionUpdate": "2"
         }
      }
   }
}
```

Référence API pour le service géré pour Apache Flink

Pour plus d'informations sur les API fournies par le service géré par Apache Flink, voir Référence API pour le service géré pour Apache Flink.