Managed Service für Apache Flink Entwicklerleitfaden

Managed Service für Apache Flink

Managed Service für Apache Flink: Managed Service für Apache Flink Entwicklerleitfaden

Table of Contents

... xvi
Was ist Managed Service für Apache Flink? ... 1
Auswahl von Managed Service für Apache Flink oder Managed Service für Apache Flink Studio ... 1
Auswählen, welche Apache-Flink-APIs in Managed Service für Apache Flink verwendet werden sollen ... 3
Auswählen einer Flink-API ... 3
Erste Schritte ... 4
So funktioniert's ... 6
Programmieren Ihrer Apache-Flink-Anwendung ... 6
DataStream-API ... 6
Tabellen-API ... 7
Erstellen Ihrer Anwendung mit Managed Service für Apache Flink ... 8
Erstellen von Anwendungen ... 8
Erstellen Sie Ihren Managed Service für Apache Flink-Anwendungscode ... 8
Erstellen der Anwendung Managed Service für Apache Flink ... 10
Starten Ihrer Anwendung Managed Service für Apache Flink ... 11
Verifizieren Ihrer Anwendung Managed Service für Apache Flink ... 12
Ausführen von Anwendungen ... 12
Anwendung und Auftragsstatus ... 13
Batch-Workloads ... 14
Anwendungsressourcen ... 15
Managed Service für Apache Flink Anwendungsressourcen ... 15
Apache Flink Anwendungsressourcen ... 15
DataStream-API ... 16
DataStream API-Konnektoren ... 17
DataStream-API Operatoren ... 32
Zeitstempel der DataStream-API ... 33
Tabellen-API ... 34
Tabellen-API Konnektoren ... 34
Tabellen-API Zeitattribute ... 36
Verwenden von Python ... 37
Programmieren einer Anwendung ... 37
Erstellen einer Anwendung ... 41

Überwachen ... 42
Laufzeiteigenschaften ... 43
Arbeiten mit Laufzeiteigenschaften in der Konsole ... 44
Arbeiten mit Laufzeiteigenschaften in der CLI ... 44
Zugriff auf Laufzeiteigenschaften in einer Anwendung, die Managed Service für Apache Flink nutzt ... 47
Fehlertoleranz ... 48
Konfigurieren der Prüfpunktprüfung ... 49
Beispiele für Prüfpunktprüfungs-APIs ... 50
Snapshots ... 52
Skalierung ... 58
Konfigurieren von Anwendungsparallelität und ParallelismPerKPU ... 58
Zuweisen von Kinesis Processing Units ... 59
Aktualisieren der Parallelität Ihrer Anwendung ... 60
Automatische Skalierung ... 61
Tagging ... 64
Hinzufügen von Tags, wenn eine Anwendung erstellt wird ... 64
Hinzufügen oder Aktualisieren von Tags für eine vorhandene Anwendung ... 65
Auflisten von Tags für eine Anwendung ... 65
Entfernen von Tags aus einer Anwendung ... 66
Verwenden von CloudFormation mit Managed Service für Apache Flink ... 66
Bevor Sie beginnen ... 66
Schreiben einer Lambda-Funktion ... 66
Erstellen einer Lambda-Rolle ... 68
Aufrufen der Lambda-Funktion ... 69
Aufrufen der Lambda-Funktion ... 70
Apache Flink-Dashboard ... 76
Zugriff auf das Apache Flink-Dashboard Ihrer Anwendung ... 77
Release-Versionen ... 79
Amazon Managed Service für Apache Flink Release 1.15.2 ... 79
Änderungen in Amazon Managed Service für Apache Flink mit Apache Flink 1.15 ... 81
Komponenten ... 81
Studio-Notebooks ... 83
Erstellen eines Studio-Notebooks ... 84
Interaktive Analyse von Streaming-Daten ... 85
Flink-Interpreter ... 86

Tabellenumgebungsvariablen von Apache Zeppelin 87
Bereitstellen als Anwendung mit dauerhaftem Zustand 87
Scala/Python-Kriterien 89
SQL-Kriterien 89
IAM-Berechtigungen 90
Konnektoren und Abhängigkeiten 90
Standardkonnektoren 91
Abhängigkeiten und benutzerdefinierte Konnektoren 92
Benutzerdefinierte Funktionen 93
Überlegungen zu benutzerdefinierten Funktionen 94
Aktivieren von Checkpointing 96
Einstellen des Checkpointing-Intervalls 96
Einstellen des Checkpointing-Typs 97
Arbeiten mit AWS Glue 97
Tabelleneigenschaften 97
Beispiele und Tutorials 100
Tutorial zum Erstellen eines Studio-Notebooks 100
Tutorial zur Bereitstellung als Anwendung mit Durable State 120
Beispiele 124
Fehlerbehebung 136
Anhalten einer hängengebliebenen Anwendung 136
Bereitstellen als Anwendung mit dauerhaftem Zustand in einer VPC ohne Internetzugang ... 136
Reduzierung der D-eploy-as-app Größe und der Build-Zeit 137
Abbrechen von Aufträgen 139
Neustarten des Apache-Flink-Interpreters 140
Anhang: Erstellen benutzerdefinierter IAM-Richtlinien 141
AWS Glue 141
CloudWatch Protokolle 142
Kinesis-Streams 143
Amazon-MSK-Cluster 145
Erste Schritte (DataStream API) 146
Komponenten der Anwendung 146
Voraussetzungen 147
Schritt 1: Einrichten eines Kontos 147
So melden Sie sich für ein AWS-Konto an 147
Erstellen eines Administratorbenutzers 148

Erteilen programmgesteuerten Zugriffs 149
Nächster Schritt 151
Schritt 2: Einrichten von AWS CLI 151
Nächster Schritt 153
Schritt 3: Erstellen einer Anwendung 153
Erstellen von zwei Amazon Kinesis Datenstroms 154
Schreiben Sie Beispieldatensätze in den Eingabe-Stream 154
Herunterladen und Überprüfen des Apache Flink-Streaming-Java-Codes 156
Kompilieren des Anwendungscodes 157
Hochladen des Apache Flink-Streaming-Java-Codes 157
Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus 158
Nächster Schritt 171
Schritt 4: Bereinigen 171
Löschen Sie die Anwendung Managed Service für Apache Flink 171
Löschen Sie Ihre Kinesis Data Streams 172
Löschen von Amazon-S3-Objekten und -Buckets 172
Löschen Sie Ihre IAM-Ressourcen 172
Löschen Ihrer CloudWatch Ressourcen 172
Nächster Schritt 173
Schritt 5: Nächste Schritte 173
Erste Schritte (Tabellen-API) 175
Komponenten der Anwendung 175
Voraussetzungen 176
Erstellen einer Anwendung 176
Erstellen Sie abhängige Ressourcen 177
Schreiben Sie Beispieldatensätze in den Eingabe-Stream 178
Herunterladen und Überprüfen des Apache Flink-Streaming-Java-Codes 179
Kompilieren des Anwendungscodes 181
Hochladen des Apache Flink-Streaming-Java-Codes 182
Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus 183
Nächster Schritt 187
Bereinigen 187
Löschen Sie die Anwendung Managed Service für Apache Flink 188
Löschen Sie Ihren Amazon-MSK-Cluster 188
Löschen der VPC 188
Löschen Sie Ihre Amazon-S3-Objekten und -Buckets 188

Löschen Sie Ihre IAM-Ressourcen ... 189
Löschen Ihrer CloudWatch Ressourcen ... 189
Nächster Schritt ... 189
Nächste Schritte ... 190
Erste Schritte (Python) ... 191
Erste Schritte mit Pyflink – Der Python-Interpreter für Apache | Amazon Web Services ... 191
Komponenten der Anwendung ... 192
Voraussetzungen ... 192
Erstellen einer Anwendung ... 193
Erstellen Sie abhängige Ressourcen ... 193
Schreiben Sie Beispieldatensätze in den Eingabe-Stream ... 195
Erstellen und Überprüfen des Apache Flink-Streaming-Python-Codes ... 196
Hinzufügen von Abhängigkeiten von Drittanbietern zu Python-Apps ... 198
Hochladen des Apache Flink-Streaming-Python-Codes ... 199
Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus ... 201
Nächster Schritt ... 206
Bereinigen ... 206
Löschen Sie die Anwendung Managed Service für Apache Flink ... 206
Löschen Sie die Kinesis Data Streams ... 206
Löschen Sie Ihre Amazon-S3-Objekten und -Buckets ... 207
Löschen Sie Ihre IAM-Ressourcen ... 207
Löschen Ihrer CloudWatch Ressourcen ... 207
Erste Schritte (Scala) ... 208
Erstellen Sie abhängige Ressourcen ... 208
Schreiben von Beispieldatensätze in den Eingabe-Stream ... 209
Laden Sie den Anwendungscode herunter und überprüfen Sie ihn ... 211
Kompilieren Sie den Anwendungscode und laden Sie ihn hoch ... 212
Erstellen und Ausführen der Anwendung (Konsole) ... 213
Erstellen der -Anwendung ... 214
Konfigurieren der Anwendung ... 214
Bearbeiten der IAM-Richtlinie ... 216
Ausführen der Anwendung ... 218
Stoppen der Anwendung ... 218
Erstellen und Ausführen der Anwendung (CLI) ... 218
Erstellen einer Berechtigungsrichtlinie ... 218
Erstellen Sie eine IAM-Richtlinie ... 220

Erstellen der Anwendung ... 222
Starten der Anwendung ... 223
Stoppen der Anwendung ... 224
Eine CloudWatch-Protokollierungs-Option hinzufügen ... 224
Umgebungseigenschaften aktualisieren ... 225
Aktualisieren des Anwendungscodes ... 226
Bereinigen ... 227
Löschen Sie die Anwendung Managed Service für Apache Flink ... 227
Löschen Sie Ihre Kinesis Data Streams ... 227
Löschen von Amazon-S3-Objekten und -Buckets ... 227
Löschen Sie Ihre IAM-Ressourcen ... 228
Löschen Sie Ihre CloudWatch-Ressourcen ... 228
Verwendung von Apache Beam ... 229
Verwendung von Apache Beam mit Managed Service für Apache Flink ... 229
Beam-Fähigkeiten ... 229
Erstellen einer Anwendung mit Apache Beam ... 230
Erstellen Sie abhängige Ressourcen ... 230
Schreiben Sie Beispieldatensätze in den Eingabe-Stream ... 231
Laden Sie den Anwendungscode herunter und überprüfen Sie ihn ... 232
Kompilieren des Anwendungscodes ... 233
Hochladen des Apache Flink-Streaming-Java-Codes ... 234
Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus ... 234
Bereinigen ... 238
Nächste Schritte ... 240
Schulungsworkshops, Labore und Lösungsimplementierungen ... 241
Lokales Entwickeln von Apache-Flink-Anwendungen vor der Bereitstellung mit Managed Service für Apache Flink ... 241
Ereigniserkennung mit Managed Service für Apache Flink Studio ... 241
AWS-Streaming-Datenlösung ... 242
Clickstream-Labor ... 242
Benutzerdefiniertes Skalieren ... 242
CloudWatch Dashboard ... 243
Amazon MSK ... 243
Weitere Managed Service für Apache Flink-Lösungen in GitHub ... 243
Dienstprogramme ... 244
Snapshot-Manager ... 244

Benchmarking ... 244
Beispiele ... 245
DataStream API-Beispiele ... 245
Rollierendes Fenster ... 246
Gleitendes Fenster ... 255
S3-Senke ... 265
MSK-Replikation ... 280
EFO-Verbraucher ... 287
Kinesis Data Firehose Senke ... 299
Kontoübergreifend ... 316
Benutzerdefinierter Truststore ... 325
Python-Beispiele ... 335
Rollierendes Fenster ... 335
Gleitendes Fenster ... 346
S3-Senke ... 357
Scala-Beispiele ... 368
Rollierendes Fenster ... 369
Gleitendes Fenster ... 387
S3-Senke ... 405
Sicherheit ... 423
Datenschutz ... 424
Datenverschlüsselung ... 424
Identitäts- und Zugriffsverwaltung ... 425
Zielgruppe ... 425
Authentifizierung mit Identitäten ... 426
Verwalten des Zugriffs mit Richtlinien ... 430
So funktioniert Amazon Managed Service für Apache Flink mit IAM ... 433
Beispiele für identitätsbasierte Richtlinien ... 441
Fehlerbehebung ... 444
Serviceübergreifende Confused-Deputy-Prävention ... 446
Überwachen ... 448
Compliance-Validierung ... 448
FedRAMP ... 449
Ausfallsicherheit ... 450
Notfallwiederherstellung ... 450
Versionsverwaltung ... 451

Sicherheit der Infrastruktur ... 451
Bewährte Methoden für die Sicherheit ... 452
Implementieren des Zugriffs mit geringsten Berechtigungen ... 452
Verwenden von IAM-Rollen zum Zugriff auf andere Amazon-Services ... 452
Implementieren einer serverseitigen Verschlüsselung in abhängigen Ressourcen ... 453
Verwenden von CloudTrail zur Überwachung von API-Aufrufen ... 453
Protokollieren und Überwachen ... 454
Protokollierung ... 455
Abfragen von Protokollen mit CloudWatch Logs Insights ... 455
Überwachen ... 455
Einrichten der Protokollierung ... 457
Einrichten der CloudWatch Protokollierung mit der Konsole ... 458
Einrichten der CloudWatch Protokollierung mit der CLI ... 458
Anwendungsüberwachungsebenen ... 464
Bewährte Methoden der Protokollierung ... 465
Protokollierung von Problemlösungen ... 465
Nächster Schritt ... 466
Analysieren von Protokollen ... 466
Ausführen einer Beispielabfrage ... 466
Beispielabfragen ... 467
Metriken und Dimensionen in Managed Service für Apache Flink ... 470
Anwendungsmetriken ... 471
Kinesis Data Streams-Konnektormetriken ... 500
Amazon MSK-Konnektor-Metriken ... 502
Apache Zeppelin-Metriken ... 504
Anzeigen von CloudWatch Metriken ... 505
Metriken ... 506
Benutzerdefinierte Metriken ... 507
Alarme ... 511
Schreiben benutzerdefinierter Nachrichten ... 523
Schreiben in CloudWatch Protokolle mit Log4J ... 523
Schreiben in CloudWatch Protokolle mit SLF4J ... 524
Verwenden von AWS CloudTrail ... 525
Informationen zu Managed Service für Apache Flink in CloudTrail ... 525
Wissenswertes zu Managed Service für Apache Flink Protokolldateieinträgen ... 527
Leistung ... 529

Behebung von Leistungsproblemen ... 529
Der Datenpfad ... 529
Lösungen zur Behebung von Leistungsproblemen ... 530
Best Practices zur Leistungsoptimierung ... 532
Richtiges Verwalten der Skalierung ... 532
Überwachen der Nutzung externer Abhängigkeitsressourcen ... 535
Lokales Ausführen Ihrer Apache-Flink-Anwendung ... 535
Leistung überwachen ... 535
Leistung überwachen mit CloudWatch-Metriken ... 535
Leistung überwachen mit CloudWatch-Protokollen und -Alarmen ... 535
Kontingent ... 537
Wartung ... 539
Festlegen einer UUID für alle Operatoren ... 541
Produktionsbereitschaft ... 542
Anwendungen für Lasttests ... 542
Maximale Parallelität ... 542
Festlegen einer UUID für alle Operatoren ... 543
Bewährte Methoden ... 544
Fehlertoleranz: Prüfpunkte und Savepoints ... 544
Nicht unterstützte Konnektor-Versionen ... 545
Leistung und Parallelität ... 545
Parallelität pro Operator festlegen ... 546
Protokollierung ... 547
Codierung ... 547
Verwalten von Anmeldeinformationen. ... 548
Lesen aus Quellen mit wenigen Shards/Partitionen ... 548
Aktualisierungsintervall für Studio-Notebooks ... 549
Optimale Leistung des Studio-Notebooks ... 549
Wie sich Strategien mit Wasserzeichen und ungenutzte Shards auf Zeitfenster auswirken ... 549
Übersicht ... 551
Beispiel ... 551
Festlegen einer UUID für alle Operatoren ... 561
Dem ServiceResourceTransformer Maven Shade Plugin hinzufügen ... 561
Apache Flink Stateful Functions ... 563
Apache Flink Anwendungsvorlage ... 563
Ort der Modulkonfiguration ... 564

Frühere Versionen 565
Verwenden des Apache Flink Kinesis Streams-Konnektor mit früheren Apache Flink-Versionen 566
Anwendungen mit Apache Flink 1.8.2 erstellen 567
Erstellen von Anwendungen mit Apache Flink 1.6.2 568
Aktualisieren von Anwendungen 569
Verfügbare Konnektoren in Apache Flink 1.6.2 und 1.8.2 569
Erste Schritte: Flink 1.13.2 570
Komponenten der Anwendung 570
Voraussetzungen 571
Schritt 1: Einrichten eines Kontos 571
Nächster Schritt 575
Schritt 2: Einrichten von AWS CLI 576
Schritt 3: Erstellen einer Anwendung 577
Schritt 4: Bereinigen 595
Schritt 5: Nächste Schritte 597
Erste Schritte: Flink 1.11.1 598
Komponenten der Anwendung 598
Voraussetzungen 599
Schritt 1: Einrichten eines Kontos 600
Schritt 2: Einrichten von AWS CLI 603
Schritt 3: Erstellen einer Anwendung 605
Schritt 4: Bereinigen 623
Schritt 5: Nächste Schritte 625
Erste Schritte: Flink 1.8.2 626
Komponenten der Anwendung 146
Voraussetzungen 627
Schritt 1: Einrichten eines Kontos 628
Schritt 2: Einrichten von AWS CLI 631
Schritt 3: Erstellen einer Anwendung 633
Schritt 4: Bereinigen 651
Erste Schritte: Flink 1.6.2 653
Komponenten der Anwendung 653
Voraussetzungen 654
Schritt 1: Einrichten eines Kontos 654
Schritt 2: Einrichten von AWS CLI 658

Schritt 3: Erstellen einer Anwendung 659
Schritt 4: Bereinigen 677
Flink-Einstellungen 680
Apache-Flink-Konfiguration 680
Zustands-Backend 681
Checkpointing 681
Savepointing 683
Heap-Größen 683
Puffer-Entlastung 684
Anpassbare Flink-Konfigurationseigenschaften 684
Fehlertoleranz 684
Checkpoints und Zustands-Backends 684
Checkpointing 684
Native RocksDB-Metriken 684
Erweiterte Zustands-Backend-Optionen 686
Vollständige Aufgabenmanager-Optionen 686
Arbeitsspeicherkonfiguration 687
RPC//Akka 687
Client 687
Erweiterte Cluster-Optionen 687
Dateisystemkonfigurationen 688
Erweiterte Fehlertoleranz-Optionen 688
Arbeitsspeicherkonfiguration 687
Metriken 688
Erweiterte Optionen für den REST-Endpunkt und Client 688
Erweiterte SSL-Sicherheitsoptionen 688
Erweiterte Planungsoptionen 688
Erweiterte Optionen für die Flink-Web-UI 688
Konfigurierte Flink-Eigenschaften anzeigen 689
Verwenden einer Amazon VPC 690
Amazon VPC – Konzepte 690
VPC-Anwendungsberechtigungen 691
Berechtigungsrichtlinie für den Zugriff auf eine Amazon VPC 692
Internet- und Servicezugriff 693
Verwandte Informationen 694
VPC API 694

CreateApplication ... 694
AddApplicationVpcConfiguration ... 695
DeleteApplicationVpcConfiguration ... 696
UpdateApplication ... 696
Beispiel: Verwenden einer VPC ... 697
Fehlerbehebung ... 698
Fehlerbehebung während der Entwicklung ... 698
Apache Flink Flame-Diagramme ... 698
Anmeldeinformationsanbieter-Problem mit EFO-Konnektor 1.15.2 ... 699
Anwendungen mit nicht unterstützten Kinesis-Konnektoren ... 699
Kompilierungsfehler: „Abhängigkeiten für das Projekt konnten nicht aufgelöst werden“ ... 702
Ungültige Auswahl: „kinesisanalyticsv2“ ... 702
UpdateApplication Aktion lädt Anwendungscode nicht neu ... 702
S3 StreamingFileSink FileNotFoundExceptions ... 703
FlinkKafkaConsumer Problem mit Stopp mit Savepoint ... 705
Flink 1.15 Async Sink Deadlock ... 705
Die Quellverarbeitung von Amazon Kinesis Data Streams ist beim Re-Sharding nicht in der richtigen Reihenfolge ... 715
Fehlerbehebung bei Laufzeitproblemen ... 716
Tools zur Fehlerbehebung ... 717
Anwendungsprobleme ... 717
Anwendung wird neu gestartet ... 722
Durchsatz ist zu langsam ... 725
Unbegrenztes Zustandswachstum ... 726
E/A-gebundene Operatoren ... 728
Upstream- oder Quelldrosselung aus einem Kinesis-Datenstrom ... 728
Checkpoints ... 729
Beim Checkpointing kommt es zu einer Zeitüberschreitung ... 736
Checkpoint-Fehler (Beam) ... 737
Gegendruck ... 739
Verzerrte Datenverteilung ... 741
Zustandsverzerrung ... 741
Integration mit Ressourcen in verschiedenen Regionen ... 742
Dokumentverlauf ... 743
API-Codebeispiele ... 750
AddApplicationCloudWatchLoggingOption ... 751

AddApplicationInput ... 751
AddApplicationInputProcessingConfiguration ... 752
AddApplicationOutput ... 753
AddApplicationReferenceDataSource ... 753
AddApplicationVpcConfiguration ... 754
CreateApplication ... 754
CreateApplicationSnapshot ... 756
DeleteApplication ... 756
DeleteApplicationCloudWatchLoggingOption ... 756
DeleteApplicationInputProcessingConfiguration ... 757
DeleteApplicationOutput ... 757
DeleteApplicationReferenceDataSource ... 757
DeleteApplicationSnapshot ... 758
DeleteApplicationVpcConfiguration ... 758
DescribeApplication ... 758
DescribeApplicationSnapshot ... 758
DiscoverInputSchema ... 759
ListApplications ... 759
ListApplicationSnapshots ... 760
StartApplication ... 760
StopApplication ... 760
UpdateApplication ... 761
API-Referenz ... 762

Amazon Managed Service für Apache Flink war zuvor als Amazon Kinesis Data Analytics für Apache Flink bekannt.

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.

Was ist Amazon Managed Service für Apache Flink?

Mit Amazon Managed Service für Apache Flink können Sie Java, Scala, Python oder SQL verwenden, um Streaming-Daten zu verarbeiten und zu analysieren. Der Service ermöglicht es Ihnen, Code für Streaming-Quellen und statische Quellen zu erstellen und auszuführen, um Zeitreihenanalysen durchzuführen, Echtzeit-Dashboards und Metriken zu speist.

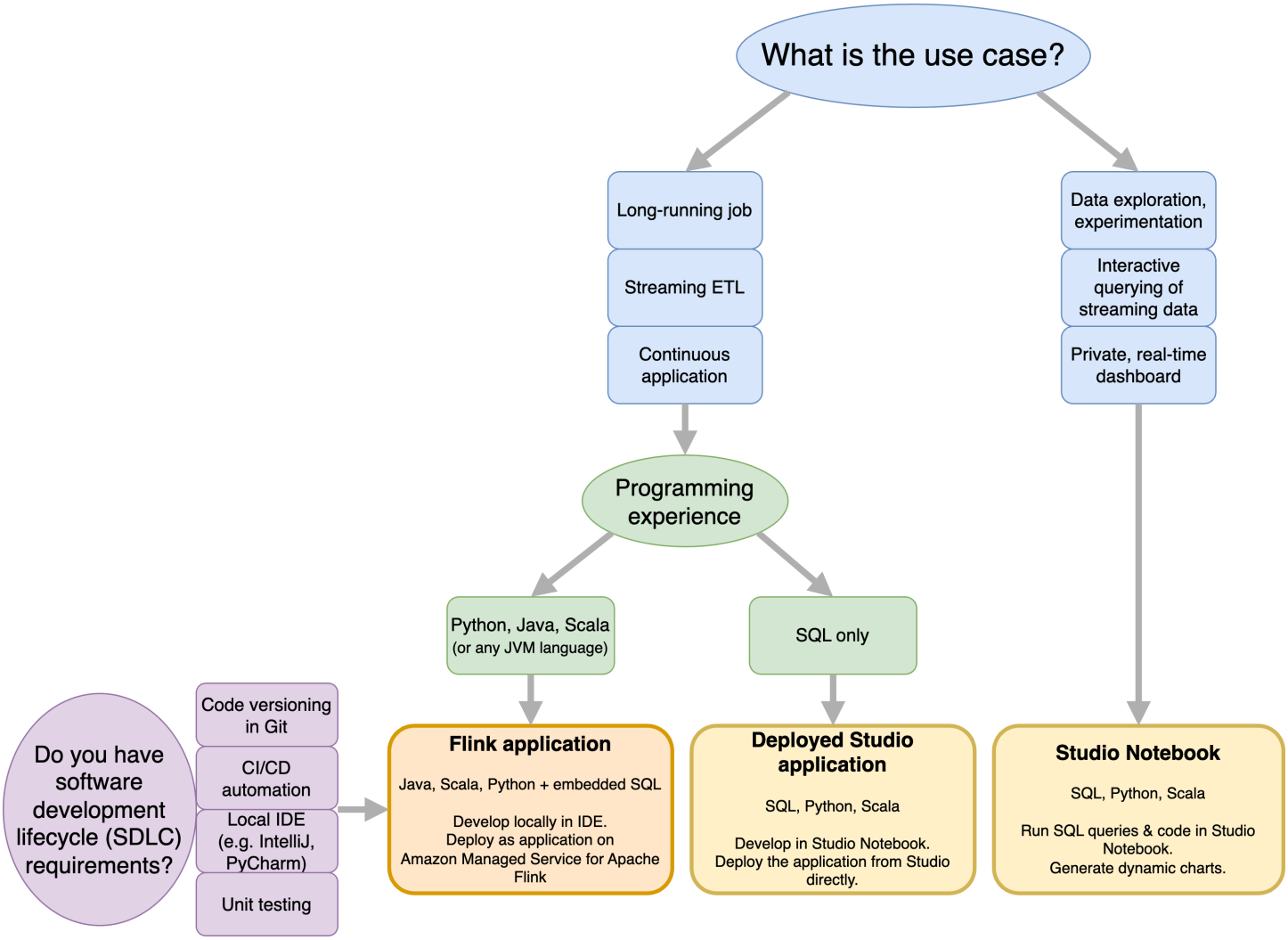
Sie können Anwendungen mit der Sprache Ihrer Wahl in Managed Service für Apache Flink mithilfe von Open-Source-Bibliotheken erstellen, die auf Apache Flink basieren. Apache Flink ist ein beliebtes Framework und eine verteilte Engine zum Verarbeiten von Datenströmen.

Managed Service für Apache Flink stellt die zugrunde liegende Infrastruktur für Ihre Apache-Flink-Anwendungen bereit. Es kümmert sich um Kernfunktionen wie die Bereitstellung von Rechenressourcen, AZ-Failover-Resilienz, parallele Berechnungen, automatische Skalierung und Anwendungs-Backups (implementiert als Checkpoints und Snapshots). Sie können die allgemeinen Flink-Programmierfunktionen (wie Operatoren, Funktionen, Quellen und Senken) genauso verwenden, wie Sie sie verwenden, wenn Sie die Flink-Infrastruktur selbst hosten.

Auswahl von Managed Service für Apache Flink oder Managed Service für Apache Flink Studio

Sie haben zwei Möglichkeiten, Ihre Flink-Aufträge mit Amazon Managed Service für Apache Flink auszuführen. Mit Managed Service für Apache Flink erstellen Sie Flink-Anwendungen in Java, Scala oder Python (und eingebettetem SQL) mithilfe einer IDE Ihrer Wahl und der Apache Flink Datastream- oder Tabellen-APIs . Mit Managed Service für Apache Flink Studio können Sie Datenströme interaktiv in Echtzeit abfragen und Stream-Verarbeitungsanwendungen einfach mit Standard-SQL, Python und Scala erstellen und ausführen.

Sie können auswählen, welche Methode am besten zu Ihrem Anwendungsfall passt. Wenn Sie sich nicht sicher sind, bietet dieser Abschnitt allgemeine Anleitungen, die Ihnen helfen.



Bevor Sie sich entscheiden, ob Sie Amazon Managed Service für Apache Flink oder Amazon Managed Service für Apache Flink Studio verwenden möchten, sollten Sie Ihren Anwendungsfall berücksichtigen.

Wenn Sie planen, eine langlebige Anwendung zu betreiben, die Workloads wie Streaming ETL oder kontinuierliche Anwendungen verursacht, sollten Sie die Verwendung von Managed Service für Apache Flink in Betracht ziehen. Dies liegt daran, dass Sie Ihre Flink-Anwendung mithilfe der Flink-APIs direkt in der IDE Ihrer Wahl erstellen können. Durch die lokale Entwicklung mit Ihrer IDE können Sie auch gängige Prozesse und Tools des Software Development Lifecycle (SDLC) wie Code-Versioning in Git, CI/CD-Automatisierung oder Einheitentests nutzen.

Wenn Sie an der Ad-hoc-Datenexploration interessiert sind, Streaming-Daten interaktiv abfragen oder private Echtzeit-Dashboards erstellen möchten, hilft Ihnen Managed Service für Apache Flink Studio mit nur wenigen Klicks dabei, diese Ziele zu erreichen. Benutzer, die mit SQL vertraut sind, können erwägen, eine lang laufende Anwendung direkt in Studio bereitzustellen.

> **Note**
>
> Sie können Ihr Studio-Notebook zu einer lang andauernden Anwendung hochstufen. Wenn Sie jedoch in Ihre SDLC-Tools wie Code-Versioning in Git und CI/CD-Automatisierung oder Techniken wie Einheitentests integrieren möchten, empfehlen wir Managed Service für Apache Flink mithilfe der IDE Ihrer Wahl.

Auswählen, welche Apache-Flink-APIs in Managed Service für Apache Flink verwendet werden sollen

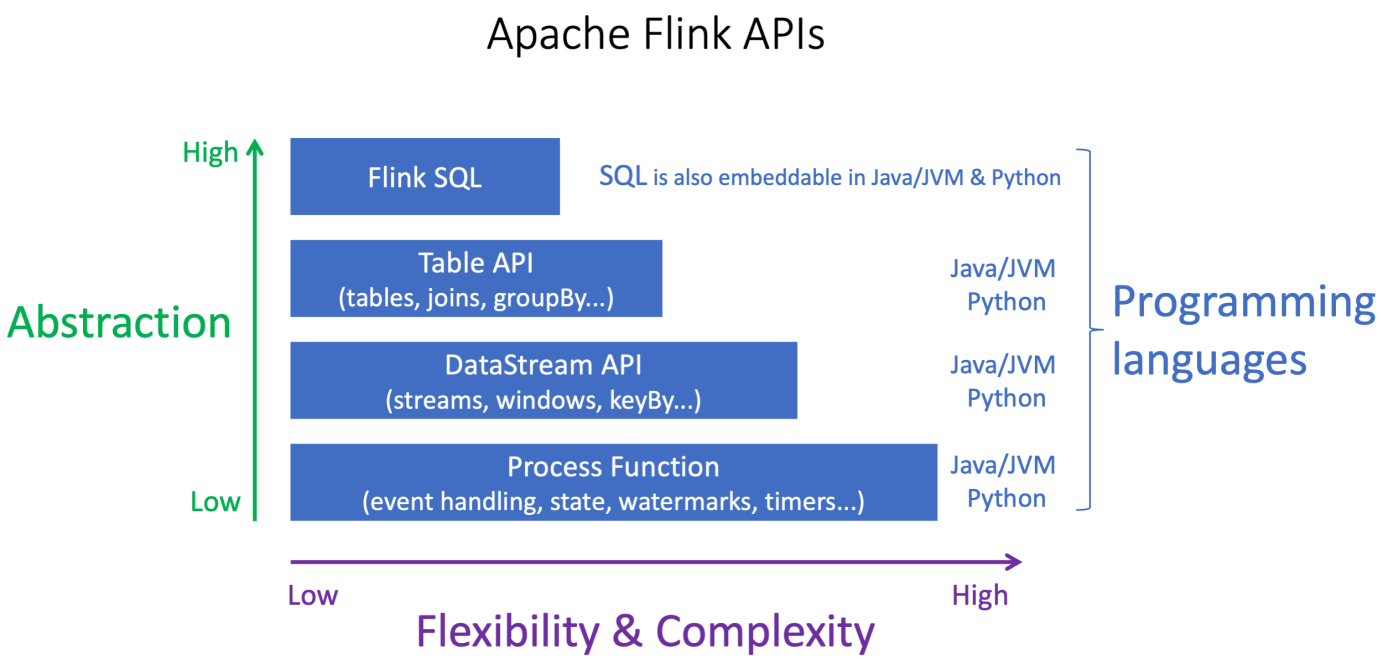
Sie können Anwendungen mit Java, Python und Scala in Managed Service für Apache Flink mithilfe von Apache-Flink-APIs in einer IDE Ihrer Wahl erstellen. Anleitungen zum Erstellen von Anwendungen mit der Flink Datastream und der Tabellen-API finden Sie in der -Dokumentation. Sie können die Sprache auswählen, in der Sie Ihre Flink-Anwendung erstellen, und die APIs, die Sie verwenden, um die Anforderungen Ihrer Anwendung und Ihres Betriebs am besten zu erfüllen. Wenn Sie sich nicht sicher sind, finden Sie in diesem Abschnitt allgemeine Anleitungen, die Ihnen helfen.

Auswählen einer Flink-API

Die Apache-Flink-APIs haben unterschiedliche Abstraktionsstufen, die sich darauf auswirken können, wie Sie Ihre Anwendung erstellen. Sie sind ausdrucksstark und flexibel und können zusammen verwendet werden, um Ihre Anwendung zu erstellen. Sie müssen nicht nur eine Flink-API verwenden. Weitere Informationen zu den Flink-APIs finden Sie in der Apache-Flink-Dokumentation.

Flink bietet vier Ebenen der API-Abstraktion: Flink SQL, Table API, DataStream API und Process Function, die in Verbindung mit der DataStream API verwendet wird. Diese werden alle in Amazon Managed Service für Apache Flink unterstützt. Es ist ratsam, nach Möglichkeit mit einer höheren Abstraktionsstufe zu beginnen. Einige Flink-Funktionen sind jedoch nur mit der Datastream-API verfügbar, mit der Sie Ihre Anwendung in Java, Python oder Scala erstellen können. Sie sollten die Verwendung der Datastream-API in Betracht ziehen, wenn:

- Sie benötigen eine differenzierte Kontrolle über den Status
- Sie möchten die Fähigkeit nutzen, eine externe Datenbank oder einen externen Endpunkt asynchron aufzurufen (z. B. für Inferenz)
- Sie möchten benutzerdefinierte Timer verwenden



> **Note**
>
> Auswählen einer Sprache mit der Datastream-API:
>
> - SQL kann unabhängig von der gewählten Programmiersprache in jede Flink-Anwendung eingebettet werden.
> - Wenn Sie planen, die DataStream API zu verwenden, werden nicht alle Connectors in Python unterstützt.
> - Wenn Sie eine niedrige Latenz/einen hohen Durchsatz benötigen, sollten Sie Java/Scala unabhängig von der API berücksichtigen.
> - Wenn Sie Async IO in der Prozessfunktionen-API verwenden möchten, müssen Sie Java verwenden.

Erste Schritte

Sie können damit beginnen, eine Anwendung, die Managed Service für Apache Flink nutzt, zu erstellen, die kontinuierlich Streaming-Daten liest und verarbeitet. Verfassen Sie dann Ihren Code mit der IDE Ihrer Wahl und testen Sie ihn mit Live-Streaming-Daten. Sie können auch Ziele konfigurieren, an die Managed Service für Apache Flink die Ergebnisse senden soll.

Um loszulegen, empfehlen wir Ihnen, die folgenden Abschnitte zu lesen:

- Managed Service für Apache Flink: So funktioniert's
- Erste Schritte mit Amazon Managed Service für Apache Flink (DataStream API)

Altenativ können Sie mit der Erstellung eines Managed Service für Apache Flink Studio-Notebooks beginnen, mit dem Sie Datenströme interaktiv in Echtzeit abfragen und Stream-Verarbeitungsanwendungen einfach mit Standard-SQL, Python und Scala erstellen und ausführen können. Mit ein paar Klicks in der AWS Management Console können Sie ein Serverless Notebook starten, um Datenströme abzufragen und innerhalb von Sekunden Ergebnisse zu erhalten. Um loszulegen, empfehlen wir Ihnen, die folgenden Abschnitte zu lesen:

- Verwenden eines Studio-Notebooks mit Managed Service für Apache Flink
- Erstellen eines Studio-Notebooks

Managed Service für Apache Flink: So funktioniert's

Managed Service für Apache Flink ist ein vollständig verwalteter Amazon-Service, mit dem Sie Apache-Flink-Anwendungen zur Verarbeitung von Streaming-Daten entwickeln und verwalten können.

Programmieren Ihrer Apache-Flink-Anwendung

Eine Apache-Flink-Anwendung ist eine Java- oder Scala-Anwendung, die mit dem Apache-Flink-Framework erstellt wurde. Sie entwickeln und erstellen Ihre Apache-Flink-Anwendung lokal.

Anwendungen verwenden hauptsächlich entweder die DataStream-API oder die Tabellen-API. Die anderen Apache-Flink-APIs stehen Ihnen ebenfalls zur Verfügung, werden jedoch weniger häufig beim Erstellen von Streaming-Anwendungen verwendet.

Die beiden APIs bieten die folgenden Funktionen:

DataStream-API

Das Programmiermodell der DataStream-API von Apache Flink basiert auf zwei Komponenten:

- Datenstrom: Die strukturierte Darstellung eines kontinuierlichen Flusses von Datensätzen.
- Transformationsoperator: Nimmt einen oder mehrere Datenströme als Eingabe und erzeugt einen oder mehrere Datenströme als Ausgabe.

Anwendungen, die mit der DataStream-API erstellt wurden, haben folgende Funktionen:

- Daten aus einer Datenquelle (z. B. einem Kinesis-Strom oder einem Amazon-MSK-Thema).
- Transformationen auf die Daten anwenden, z. B. Filterung, Aggregation oder Anreicherung.
- Transformierte Daten in eine Datensenke schreiben.

Anwendungen, die die DataStream-API verwenden, können in Java oder Scala geschrieben werden und können aus einem Kinesis-Datenstrom, einem Amazon-MSK-Thema oder einer benutzerdefinierten Quelle lesen.

Ihre Anwendung verarbeitet Daten mithilfe eines Konnektors. Apache Flink verwendet die folgenden Arten von Konnektoren:

- Quelle: Ein Konnektor, der zum Lesen externer Daten verwendet wird.
- Senke: Ein Konnektor, der zum Schreiben an externe Standorte verwendet wird.
- Operator: Ein Konnektor, der zur Verarbeitung von Daten innerhalb der Anwendung verwendet wird.

Eine typische Anwendung besteht aus mindestens einem Datenstrom mit einer Quelle, einem Datenstrom mit einem oder mehreren Operatoren und mindestens einer Datensenke.

Weitere Informationen zur Verwendung der API finden Sie unter DataStream-API.

Tabellen-API

Das Programmiermodell der Tabellen-API von Apache Flink basiert auf den folgenden Komponenten:

- Tabellenumgebung: Eine Schnittstelle zu zugrunde liegenden Daten, die Sie verwenden, um eine oder mehrere Tabellen zu erstellen und zu hosten.
- Tabelle: Ein Objekt, das den Zugriff auf eine SQL-Tabelle oder -Ansicht ermöglicht.
- Tabellenquelle: Wird verwendet, um Daten aus einer externen Quelle zu lesen, z. B. aus einem Amazon-MSK-Thema.
- Tabellenfunktion: Eine SQL-Abfrage oder ein API-Aufruf, der zur Transformation von Daten verwendet wird.
- Tabellensenke: Wird verwendet, um Daten an einen externen Speicherort zu schreiben, z. B. in einen Amazon-S3-Bucket.

Anwendungen, die mit der Tabellen-API erstellt werden, haben folgende Funktionen:

- Erstellen einer `TableEnvironment` durch Herstellen einer Verbindung zu einer `Table Source`.
- Erstellen einer Tabelle in der `TableEnvironment` durch entweder SQL-Abfragen oder Tabellen-API-Funktionen.
- Ausführen einer Tabellenabfrage über die Tabellen-API oder SQL
- Anwenden von Transformationen auf die Abfrageergebnisse über Tabellenfunktionen oder SQL-Abfragen.
- Schreiben der Abfrage- oder Funktionsergebnisse in eine `Table Sink`.

Anwendungen, die die Tabellen-API verwenden, können in Java oder Scala geschrieben werden und Daten entweder mittels API-Aufrufen oder SQL-Abfragen abfragen.

Weitere Informationen zur Verwendung der Tabellen-API finden Sie unter Tabellen-API.

Erstellen Ihrer Anwendung mit Managed Service für Apache Flink

Managed Service für Apache Flink ist ein AWS-Service, der eine Umgebung für das Hosten Ihrer Apache-Flink-Anwendung erstellt und ihr die folgenden Einstellungen zur Verfügung stellt:

- Laufzeiteigenschaften: Parameter, die Sie Ihrer Anwendung zur Verfügung stellen können. Sie können diese Parameter ändern, ohne Ihren Anwendungscode neu kompilieren zu müssen.
- Fehlertoleranz: Wie sich Ihre Anwendung nach Unterbrechungen und Neustarts wiederherstellt.
- Protokollieren und Überwachen: Wie Ihre Anwendung Ereignisse in CloudWatch-Protokollen protokolliert.
- Skalierung: Wie Ihre Anwendung Datenverarbeitungsressourcen bereitstellt.

Sie können die Anwendung mit Managed Service für Apache Flink entweder über die Konsole oder die AWS CLI erstellen und ausführen. Erste Schritte zum Erstellen einer Anwendung mit Managed Service für Apache Flink finden Sie unter Erste Schritte (DataStream API).

Erstellen der Anwendung Managed Service für Apache Flink

Dieses Thema enthält Informationen zum Erstellen eines Managed Service für Apache Flink.

Dieses Thema enthält die folgenden Abschnitte:

- Erstellen Sie Ihren Managed Service für Apache Flink-Anwendungscode
- Erstellen der Anwendung Managed Service für Apache Flink
- Starten Ihrer Anwendung Managed Service für Apache Flink
- Verifizieren Ihrer Anwendung Managed Service für Apache Flink

Erstellen Sie Ihren Managed Service für Apache Flink-Anwendungscode

In diesem Abschnitt werden die Komponenten beschrieben, die Sie verwenden, um den Anwendungscode für die Anwendung Managed Service für Apache Flink zu erstellen.

Es wird empfohlen, die neueste unterstützte Version von Apache Flink für Ihren Anwendungscode zu verwenden. Die neueste Version von Apache Flink, die Managed Service für Apache Flink unterstützt, ist 1.15.2. Informationen zur Aktualisierung von Managed Service für Apache Flink-Anwendungen finden Sie unter Aktualisieren von Anwendungen.

Sie erstellen Ihren Anwendungscode mit Apache Maven. Ein Apache Maven-Projekt verwendet eine `pom.xml`-Datei, um die Versionen der verwendeten Komponenten anzugeben.

> **Note**
>
> Managed Service für Apache Flink unterstützt JAR-Dateien mit einer Größe von bis zu 512 MB. Wenn Sie eine größere JAR-Datei verwenden, kann Ihre Anwendung nicht gestartet werden.

Verwenden Sie die folgenden Komponentenversionen für Managed Service für Apache Flink-Anwendungen:

Komponente	Version
Java	11 (empfohlen)
Scala	Siehe den Hinweis zur Entkopplung von Scala weiter unten
Managed Service für Apache Flink Laufzeit (aws-kinesisanalytics-runtime)	1.2.0
AWS Kinesis Connector (flink-connector-kinesis)	1.15.2
Apache Beam (nur Beam-Anwendungen)	2.33.0, mit Jackson-Version 2.12.2

Ab Version 1.15 ist Flink Scala-frei. Anwendungen können jetzt die Java-API von jeder Scala-Version aus verwenden. Sie müssen die Scala-Standardbibliothek Ihrer Wahl in Ihre Scala-Anwendungen integrieren.

Ein Beispiel für eine `pom.xml`-Datei für eine Managed Service für Apache Flink-Anwendung, die Apache Flink Version 1.15.2 verwendet, finden Sie in der Managed Service für Apache Flink Erste Schritte Anwendung.

Informationen zum Erstellen einer Managed Service für Apache Flink-Anwendung, die Apache Beam verwendet, finden Sie unter Verwendung von Apache Beam.

Angabe der Apache Flink-Version Ihrer Anwendung

Wenn Sie Managed Service für Apache Flink Laufzeit Version 1.1.0 und höher verwenden, geben Sie die Version von Apache Flink an, die Ihre Anwendung verwendet, wenn Sie Ihre Anwendung kompilieren. Sie geben die Version von Apache Flink mit dem `-Dflink.version`-Parameter wie folgt an:

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

Informationen zum Erstellen von Anwendungen mit älteren Versionen von Apache Flink finden Sie unter Frühere Versionen.

Erstellen der Anwendung Managed Service für Apache Flink

Nachdem Sie Ihren Anwendungscode erstellt haben, gehen Sie wie folgt vor, um Ihre Managed Service für Apache Flink-Anwendung zu erstellen:

- Laden Sie Ihren Anwendungscode hoch: Laden Sie Ihren Anwendungscode in einen Amazon-S3-Bucket hoch. Geben Sie beim Erstellen Ihrer Anwendung den S3-Bucket-Namen und den Objektnamen Ihres Anwendungscodes an. Ein Tutorial, das zeigt, wie Sie Ihren Anwendungscode hochladen, finden Sie in the section called “Hochladen des Apache Flink-Streaming-Java-Codes” im Erste Schritte (DataStream API)-Tutorial.
- Erstellen Sie Ihre Managed Service für Apache Flink-Anwendung: Verwenden Sie eine der folgenden Methoden, um Ihre Managed Service für Apache Flink-Anwendung zu erstellen:
  - Erstellen Sie Ihre Managed Service für Apache Flink-Anwendung mithilfe der AWS-Konsole: Sie können Ihre Anwendung mithilfe der AWS-Konsole erstellen und konfigurieren.

    Wenn Sie Ihre Anwendung über die Konsole erstellen, werden die abhängigen Ressourcen Ihrer Anwendung (wie CloudWatch Protokollstreams, IAM-Rollen und IAM-Richtlinien) für Sie erstellt.

Wenn Sie die Anwendung mithilfe der Konsole erstellen, geben Sie an, welche Version von Apache Flink Ihre Anwendung verwendet, indem Sie sie aus dem Pulldown-Menü auf der Seite Managed– Service für Apache Flink Anwendung erstellen auswählen.

Ein Tutorial zur Verwendung der Konsole zum Erstellen einer Anwendung finden Sie in the section called “Erstellen und Ausführen der Anwendung (Konsole)” im Erste Schritte (DataStream API)-Tutorial.

- Erstellen Sie Ihre Managed Service für Apache Flink-Anwendung mit der AWS-CLI: Sie können Ihre Anwendung mit der AWS-CLI erstellen und konfigurieren.

  Wenn Sie Ihre Anwendung mit der CLI erstellen, müssen Sie auch die abhängigen Ressourcen Ihrer Anwendung (wie CloudWatch Protokollstreams, IAM-Rollen und IAM-Richtlinien) manuell erstellen.

  Wenn Sie Ihre Anwendung mit der CLI erstellen, geben Sie mithilfe des `RuntimeEnvironment`-Aktionsparameters der `CreateApplication`-Aktion an, welche Version von Apache Flink Ihre Anwendung verwendet.

  Ein Tutorial zur Verwendung der CLI zum Erstellen einer Anwendung finden Sie in the section called “Eine Anwendung mit der CLI erstellen und ausführen” im Erste Schritte (DataStream API)-Tutorial.

> **Note**
>
> Sie können die `RuntimeEnvironment` einer vorhandenen Anwendung nicht ändern. Wenn Sie die `RuntimeEnvironment` einer vorhandenen Anwendung ändern müssen, müssen Sie die Anwendung löschen und erneut erstellen.

Starten Ihrer Anwendung Managed Service für Apache Flink

Nachdem Sie Ihren Anwendungscode erstellt, in S3 hochgeladen und Ihre Managed Service für Apache Flink-Anwendung erstellt haben, starten Sie die Anwendung. Das Starten einer Anwendung Managed Service für Apache Flink dauert in der Regel mehrere Minuten.

Verwenden Sie eine der folgenden Methoden, um die Anwendung zu starten:

- Starten Sie Ihre Managed Service für Apache Flink-Anwendung über die AWS-Konsole: Sie können Ihre Anwendung ausführen, indem Sie auf der Seite Ihrer Anwendung in der AWS-Konsole auf Ausführen klicken.
- Starten Sie Ihre Managed Service für Apache Flink-Anwendung mit der AWS -API: Sie können Ihre Anwendung mit der -StartApplicationAktion ausführen.

Verifizieren Ihrer Anwendung Managed Service für Apache Flink

Sie können auf folgende Arten überprüfen, ob die Anwendung funktioniert:

- Verwenden von - CloudWatch Protokollen: Sie können CloudWatch Logs und CloudWatch Logs Insights verwenden, um zu überprüfen, ob Ihre Anwendung ordnungsgemäß ausgeführt wird. Informationen zur Verwendung von CloudWatch Protokollen mit Ihrer Anwendung von Managed Service für Apache Flink finden Sie unter Protokollieren und Überwachen.
- Verwenden von CloudWatch Metriken: Sie können CloudWatch Metriken verwenden, um die Aktivität Ihrer Anwendung oder die Aktivität in den Ressourcen zu überwachen, die Ihre Anwendung für Eingabe oder Ausgabe verwendet (z. B. Kinesis-Streams, Kinesis-Data-Firehose-Streams oder Amazon S3-Buckets). Weitere Informationen zu CloudWatch Metriken finden Sie unter Arbeiten mit Metriken im Amazon CloudWatch -Benutzerhandbuch.
- Überwachung der Ausgabespeicherorte: Wenn Ihre Anwendung die Ausgabe an einen Speicherort schreibt (z. B. einen Amazon S3-Bucket oder eine Datenbank), können Sie diesen Speicherort auf geschriebene Daten überwachen.

Ausführen einer Anwendung, die Managed Service für Apache Flink nutzt

Dieser Abschnitt enthält Informationen zum Ausführen eines Managed Service für Apache Flink.

Wenn Sie Ihre Anwendung, die Managed Service für Apache Flink nutzt, ausführen, erstellt der Service einen Apache-Flink-Auftrag. Ein Apache-Flink-Auftrag ist der Ausführungszyklus Ihrer Anwendung, die Managed Service für Apache Flink nutzt. Die Ausführung des Auftrags und die verwendeten Ressourcen werden vom Auftragsmanager verwaltet. Der Auftragsmanager unterteilt die Ausführung der Anwendung in Aufgaben. Jede Aufgabe wird von einem Aufgabenmanager verwaltet. Wenn Sie die Leistung Ihrer Anwendung überwachen, können Sie die Leistung jedes Aufgabenmanagers oder des Auftragsmanagers als Ganzes untersuchen.

Informationen zu Apache-Flink-Aufträgen finden Sie unter Aufträge und Planung in der Apache-Flink-Dokumentation.

Anwendung und Auftragsstatus

Sowohl Ihre Anwendung als auch der Auftrag der Anwendung haben einen aktuellen Ausführungsstatus:

- Anwendungsstatus: Ihre Anwendung hat einen aktuellen Status, der die Ausführungsphase beschreibt. Die folgenden Anwendungsstatus sind möglich:
  - Stabile Anwendungsstatus: Ihre Anwendung bleibt in der Regel so lange in diesem Status, bis Sie einen Statuswechsel vornehmen:
    - BEREIT: Eine neue oder angehaltene Anwendung befindet sich im BEREIT-Status, bis Sie sie ausführen.
    - LÄUFT: Eine Anwendung, die erfolgreich gestartet wurde, befindet sich im LÄUFT-Status.
  - Vorübergehende Anwendungsstatus: Eine Anwendung mit einem solchen Status ist in der Regel dabei, in einen anderen Status überzugehen. Wenn sich eine Anwendung längere Zeit in einem vorübergehenden Status befindet, können Sie die Anwendung mit der StopApplication-Aktion anhalten, wobei der `Force`Parameter auf `true` eingestellt ist. Diese umfassen u. a. folgende:
    - `STARTING`: Tritt nach der StartApplication-Aktion auf. Die Anwendung wechselt vom Status `READY` in den Status `RUNNING`.
    - `STOPPING`: Tritt nach der StopApplication-Aktion auf. Die Anwendung wechselt vom Status `RUNNING` in den Status `READY`.
    - `DELETING`: Tritt nach der DeleteApplication-Aktion auf. Die Anwendung wird gerade gelöscht.
    - `UPDATING`: Tritt nach der UpdateApplication-Aktion auf. Die Anwendung wird gerade aktualisiert und kehrt in den Status `RUNNING` oder `READY` zurück.
    - `AUTOSCALING`: Für die Anwendung ist die `AutoScalingEnabled`-Eigenschaft der ParallelismConfiguration auf `true` gesetzt und der Service erhöht die Parallelität der Anwendung. Wenn sich die Anwendung in diesem Status befindet, ist die einzige gültige API-Aktion, die Sie verwenden können, die Aktion StopApplication, deren `Force`-Parameter auf `true` gesetzt ist. Weitere Informationen zum Auto Scaling finden Sie unter Automatische Skalierung.
    - `FORCE_STOPPING`: Tritt auf, nachdem die StopApplication-Aktion aufgerufen wurde, wobei der `Force`-Parameter auf `true` gesetzt ist. Die Anwendung wird gerade zwangsweise

angehalten. Die Anwendung wechselt vom Status `STARTING`, `UPDATING`, `STOPPING` oder `AUTOSCALING` in den Status `READY`.

- `ROLLING_BACK`: Tritt auf, nachdem die RollbackApplication-Aktion aufgerufen wurde. Die Anwendung wird gerade auf eine frühere Version zurückgesetzt. Die Anwendung wechselt vom Status `UPDATING` oder `AUTOSCALING` in den Status `RUNNING`.
- `ROLLED_BACK`: Wenn Sie eine Anwendung erfolgreich zurückgesetzt haben, wird dies zum Status der Version, von der aus Sie das Rollback durchgeführt haben. Weitere Informationen zum Rollback einer Anwendung finden Sie unter RollbackApplication.
- `MAINTENANCE`: Tritt auf, während Managed Service für Apache Flink Patches auf Ihre Anwendung einspielt. Weitere Informationen finden Sie unter Wartung.

Sie können den Status Ihrer Anwendung in der Konsole oder mithilfe der Aktion DescribeApplication überprüfen.

- Auftragsstatus: Wenn sich Ihre Anwendung im `RUNNING`-Status befindet, hat Ihr Auftrag einen Status, der die aktuelle Ausführungsphase beschreibt. Ein Auftrag beginnt im `CREATED`-Status und geht dann in den `RUNNING`-Satus über, wenn er gestartet wurde. Wenn Fehlerzustände auftreten, wechselt Ihre Anwendung in den folgenden Status:
  - Bei Anwendungen, die Apache Flink 1.11 und höher verwenden, wechselt Ihre Anwendung in den `RESTARTING`-Status.
  - Bei Anwendungen, die Apache Flink 1.8 und niedriger verwenden, wechselt Ihre Anwendung in den `FAILING`-Status.

  Die Anwendung wechselt dann entweder zum Status `RESTARTING` oder `FAILED`, je nachdem, ob der Auftrag neu gestartet werden kann.

  Sie können den Status des Auftrags überprüfen, indem Sie das CloudWatch-Protokoll Ihrer Anwendung auf Statusänderungen überprüfen.

Batch-Workloads

Managed Service für Apache Flink unterstützt die Ausführung von Apache-Flink-Batch-Workloads. Wenn in einem Batch-Auftrag ein Apache-Flink-Auftrag den Status ABGESCHLOSSEN erreicht, wird der Anwendungsstatus von Managed Service für Apache Flink auf BEREIT gesetzt. Weitere Informationen zum Status von Flink-Aufträgen finden Sie unter Aufträge und Planung.

Anwendungsressourcen

In diesem Abschnitt werden die Systemressourcen beschrieben, die Ihre Anwendung verwendet. Wenn Sie verstehen, wie Managed Service für Apache Flink Ressourcen bereitstellt und verwendet, können Sie eine leistungsstarke und stabile Anwendung mit Managed Service für die Apache Flink entwerfen, erstellen und verwalten.

Managed Service für Apache Flink Anwendungsressourcen

Managed Service für Apache Flink ist ein AWS-Service, der eine Umgebung für das Hosten Ihrer Apache-Flink-Anwendung erstellt. Der Service von Managed Service für Apache Flink stellt Ressourcen mithilfe von Einheiten bereit, die als Kinesis Processing Units (KPUs) bezeichnet werden.

Eine KPU steht für die folgenden Systemressourcen:

- Ein CPU-Kern
- 4 GB Arbeitsspeicher, davon 1 GB systemeigener Speicher und 3 GB Heap-Speicher
- 50 GB freier Festplattenspeicher

KPUs führen Anwendungen in separaten Ausführungseinheiten aus, die als Aufgaben und Unteraufgaben bezeichnet werden. Sie können sich eine Unteraufgabe als das Äquivalent eines Threads vorstellen.

Die Anzahl der für eine Anwendung verfügbaren KPUs entspricht der Anwendungseinstellung für `Parallelism` geteilt durch die Anwendungseinstellung für `ParallelismPerKPU`.

Informationen zur Anwendungsparallelität finden Sie unter Skalierung.

Apache Flink Anwendungsressourcen

Die Apache-Flink-Umgebung weist Ressourcen für Ihre Anwendung mithilfe von Einheiten zu, die als Aufgabenslots bezeichnet werden. Wenn Managed Service für Apache Flink Ressourcen für Ihre Anwendung zuweist, weist es einer einzelnen KPU einen oder mehrere Apache-Flink-Aufgabenslots zu. Die Anzahl der Slots, die einer einzelnen KPU zugewiesen sind, entspricht der Einstellung `ParallelismPerKPU` Ihrer Anwendung. Weitere Informationen über Aufgabenslots finden Sie unter Auftragsplanung in der Apache-Flink-Dokumentation.

Operatorenparallelität

Sie können die maximale Anzahl von Unteraufgaben festlegen, die ein Operator verwenden kann. Dieser Wert wird als Operatorenparallelität bezeichnet. Standardmäßig entspricht die Parallelität der einzelnen Operatoren in Ihrer Anwendung der Parallelität der Anwendung. Das bedeutet, dass standardmäßig jeder Operator in Ihrer Anwendung bei Bedarf alle verfügbaren Unteraufgaben in der Anwendung verwenden kann.

Sie können die Parallelität der Operatoren in Ihrer Anwendung mithilfe der `setParallelism`-Methode festlegen. Mit dieser Methode können Sie die Anzahl der Unteraufgaben steuern, die jeder Operator gleichzeitig verwenden kann.

Weitere Informationen zur Operatorverkettung finden Sie unter Aufgabenverkettung und Ressourcengruppen in der Apache-Flink-Dokumentation.

Operatorverkettung

Normalerweise verwendet jeder Operator eine separate Unteraufgabe für die Ausführung, aber wenn mehrere Operatoren immer nacheinander ausgeführt werden, kann die Laufzeit sie alle derselben Aufgabe zuweisen. Dieser Vorgang wird Operatorverkettung genannt.

Mehrere sequenzielle Operatoren können zu einer einzigen Aufgabe verkettet werden, wenn sie alle mit denselben Daten arbeiten. Dies ist eine Auswahl der Kriterien, die erforderlich sind, damit dies zutrifft:

- Die Operatoren führen eine einfache 1:1-Weiterleitung durch.
- Die Operatoren haben alle dieselbe Operatorenparallelität.

Wenn Ihre Anwendung Operatoren zu einer einzigen Unteraufgabe zusammenfasst, werden Systemressourcen geschont, da der Service keine Netzwerkoperationen durchführen und jedem Operator Unteraufgaben zuweisen muss. Um festzustellen, ob Ihre Anwendung Operatorverkettung verwendet, sehen Sie sich das Auftragsdiagramm in der Konsole von Managed Service für Apache Flink an. Jeder Scheitelpunkt in der Anwendung steht für einen oder mehrere Operatoren. Das Diagramm zeigt Operatoren, die zu einem einzigen Scheitelpunkt verkettet wurden.

DataStream-API

Ihre Apache Flink-Anwendung verwendet die Apache Flink DataStream-API, um Daten in einen Datenstrom umzuwandeln.

Dieser Abschnitt enthält die folgenden Themen:

- Verwenden von Connectors zum Verschieben von Daten in Managed Service für Apache Flink mit der DataStream API: Diese Komponenten verschieben Daten zwischen Ihrer Anwendung und externen Datenquellen und Zielen.
- Transformieren von Daten mithilfe von Operatoren in Managed Service für Apache Flink mit der DataStream-API: Diese Komponenten transformieren oder gruppieren Datenelemente innerhalb Ihrer Anwendung.
- Verfolgen von Ereignissen in Managed Service für Apache Flink mithilfe der DataStream-API: In diesem Thema wird beschrieben, wie Managed Service für Apache Flink Ereignisse verfolgt, wenn die DataStream-API verwendet wird.

Verwenden von Connectors zum Verschieben von Daten in Managed Service für Apache Flink mit der DataStream API

In der DataStream API von Amazon Managed Service für Apache Flink sind Konnektoren Softwarekomponenten, die Daten in und aus einer Anwendung von Managed Service für Apache Flink verschieben. Konnektoren sind flexible Integrationen, mit denen Sie aus Dateien und Verzeichnissen lesen können. Konnektoren bestehen aus kompletten Modulen für die Interaktion mit Amazon-Services und Systemen von Drittanbietern.

Zu den Konnektoren gehören die folgenden:

- Quellen: Stellen Ihrer Anwendung Daten aus einem Kinesis Data Stream, einer Datei oder einer anderen Datenquelle zur Verfügung.
- Senken: Senden Daten aus Ihrer Anwendung an einen Kinesis Data Stream, Kinesis Data Firehose-Stream oder ein anderes Datenziel.
- Asynchrone I/O: Ermöglicht asynchronen Zugriff auf eine Datenquelle (z. B. eine Datenbank), um Stream-Ereignisse zu bereichern.

Verfügbare Konnektoren

Das Apache Flink-Framework enthält Konnektoren für den Zugriff auf Daten aus verschiedenen Quellen. Informationen zu den im Apache Flink-Framework verfügbaren Konnektoren finden Sie unter Konnektoren in der Apache Flink-Dokumentation.

Warning

Wenn Sie Anwendungen haben, die auf Flink 1.6, 1.8, 1.11 oder 1.13 laufen und diese in den Regionen Nahen Osten (VAE), in Asien-Pazifik (Hyderabad), in Israel (Tel Aviv), in Europa (Zürich), im Nahen Osten (VAE) und in Asien-Pazifik (Melbourne) oder Asien-Pazifik (Jakarta) ausführen möchten, müssen Sie möglicherweise Ihr Anwendungsarchiv mit einem aktualisierten Konnektor neu erstellen oder auf Flink 1.15 aktualisieren. Im Folgenden finden Sie empfohlene Richtlinien:

Konnektor-Upgrades

Fl Vi si	Verwendeter Konnektor	Auflösung
1. – 1.	Firehose	Ihre Anwendun hängt von einer veraltete n Version des Firehose-Konnektor s ab, die keine neueren AWS-Regio nen kennt. Erstellen Sie Ihr Anwendun sarchiv mit Firehose Konnector Version

Fl Vi si	Verwendeter Konnektor	Auflösung
		2.1.0 neu. v2.1.0

Fl Vi si	Verwendeter Konnektor	Auflösung
1.	Kinesis	Ihre Anwendun hängt von einer veraltete n Version des Flink Kinesis-K onnektors ab, die keine neueren AWS-Regio nen kennt. Erstellen Sie Ihr Anwendun sarchiv mit Flink Kinesis-K onnector Version

Fl Vi si	Verwendeter Konnektor	Auflösung
		1.6.1 neu. https://github.com/awslabs/amazon-kinesis-connector-flink/Baum/1.6.1

Fl Vi si	Verwendeter Konnektor	Auflösung
1.	Kinesis	Ihre Anwendun hängt von einer veraltete n Version des Flink Kinesis-K onnektors ab, die keine neueren AWS-Regio nen kennt. Erstellen Sie Ihr Anwendun sarchiv mit Flink Kinesis-K onnector Version

Fl Vi si	Verwendeter Konnektor	Auflösung
		2.4.1 neu. https://github.com/awslabs/amazon-kinesis-connector-flink/Baum/2.4.1

Fl V si	Verwendeter Konnektor	Auflösung
1. ur 1.	Kinesis	Ihre Anwendun hängt von einer veraltete n Version des Flink Kinesis-K onnektors ab, die keine neueren AWS-Regio nen kennt. Leider veröffent licht Flink keine Patches oder Bugfixes mehr für 1.6/1.13-

Fl Vi si	Verwendeter Konnektor	Auflösung
		Konnektoren. Wir empfehlen, auf Flink 1.15 zu aktualisieren, indem Sie Ihr Anwendungsarchiv mit Flink 1.15 neu erstellen.

Streaming-Datenquellen zu Managed Service für Apache Flink hinzufügen

Apache Flink bietet Konnektoren zum Lesen aus Dateien, Sockets, Sammlungen und benutzerdefinierten Quellen. In Ihrem Anwendungscode verwenden Sie eine Apache Flink-Quelle, um Daten aus einem Stream zu empfangen. In diesem Abschnitt werden die Quellen beschrieben, die für Amazon-Services verfügbar sind.

Kinesis Data Streams

Die `FlinkKinesisConsumer`-Quelle stellt Streaming-Daten aus einem Amazon Kinesis Data Stream für Ihre Anwendung bereit.

Erstellen einer **`FlinkKinesisConsumer`**

Das folgende Code-Beispiel zeigt das Erstellen eines `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));
```

Weitere Informationen zur Verwendung von `FlinkKinesisConsumer` finden Sie unter Herunterladen und Überprüfen des Apache Flink-Streaming-Java-Codes.

Erstellen eines **`FlinkKinesisConsumer`**, der einen EFO Verbraucher verwendet

unterstützt FlinkKinesisConsumer jetzt Enhanced Fan-Out (EFO).

Wenn ein Kinesis-Verbraucher EFO verwendet, stellt ihm der Kinesis Data Streams-Service seine eigene dedizierte Bandbreite zur Verfügung, anstatt dass der Verbraucher die feste Bandbreite des Streams mit den anderen Verbrauchern teilt, die aus dem Stream lesen.

Weitere Informationen zur Verwendung von EFO mit dem Kinesis Consumer finden Sie unter FLIP-128: Verbesserte Verteilung für AWS-Kinesis-Verbraucher.

Sie aktivieren den EFO-Consumer, indem Sie die folgenden Parameter für den Kinesis-Consumer festlegen:

- RECORD_PUBLISHER_TYPE: Setzen Sie diesen Parameter auf EFO, damit Ihre Anwendung einen EFO-Consumer für den Zugriff auf die Kinesis Data Stream-Daten verwendet.
- EFO_CONSUMER_NAME: Setzen Sie diesen Parameter auf einen Zeichenfolgenwert, der unter den Verbrauchern dieses Streams eindeutig ist. Die Wiederverwendung eines Verbrauchernamens in demselben Kinesis Data Stream führt dazu, dass der vorherige Verbraucher, der diesen Namen verwendet hat, beendet wird.

Um einen `FlinkKinesisConsumer` für die Verwendung von EFO zu konfigurieren, fügen Sie dem Verbraucher die folgenden Parameter hinzu:

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

Ein Beispiel für eine Managed Service für Apache Flink-Anwendung, die einen EFO-Consumer verwendet, finden Sie unter EFO-Verbraucher.

Amazon MSK

Die `KafkaSource`-Quelle stellt Streaming-Daten aus einem Amazon MSK-Thema für Ihre Anwendung bereit.

Erstellen einer **`KafkaSource`**

Das folgende Code-Beispiel zeigt das Erstellen eines `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");
```

Weitere Informationen zur Verwendung von `KafkaSource` finden Sie unter MSK-Replikation.

Schreiben von Daten mithilfe von Senken in Managed Service für Apache Flink

In Ihrem Anwendungscode verwenden Sie eine Apache Flink-Senke, um Daten aus einem Apache Flink-Stream in einen AWS-Service wie Kinesis Data Streams zu schreiben.

Apache Flink bietet Senken für Dateien, Sockets und benutzerdefinierte Senken. Die folgenden Senken sind für AWS verfügbar:

Kinesis Data Streams

Apache Flink bietet Informationen zum Kinesis Data Streams-Konnektor in der Apache Flink-Dokumentation.

Ein Beispiel für eine Anwendung, die einen Kinesis Data Stream für Eingabe und Ausgabe verwendet, finden Sie unter Erste Schritte (DataStream API).

Amazon S3

Sie können Apache Flink `StreamingFileSink` verwenden, um Objekte in einen Amazon S3-Bucket zu schreiben.

Ein Beispiel dafür, wie man Objekte in S3 schreibt, finden Sie unter the section called “S3-Senke”.

Kinesis Data Firehose

Der `FlinkKinesisFirehoseProducer` ist eine zuverlässige, skalierbare Apache Flink-Senke zum Speichern von Anwendungsausgaben mithilfe des Kinesis Data Firehose-Service. In diesem Abschnitt wird die Einrichtung eines Maven-Projekts beschrieben, um einen `FlinkKinesisFirehoseProducer` zu erstellen und zu verwenden.

Themen

- Erstellen einer FlinkKinesisFirehoseProducer
- FlinkKinesisFirehoseProducer-Codebeispiel

Erstellen einer **`FlinkKinesisFirehoseProducer`**

Das folgende Code-Beispiel zeigt das Erstellen eines `FlinkKinesisFirehoseProducer`:

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

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

`FlinkKinesisFirehoseProducer`-Codebeispiel

Das folgende Codebeispiel zeigt, wie Sie einen `FlinkKinesisFirehoseProducer` erstellen und konfigurieren und Daten aus einem Apache Flink Data Stream an den Kinesis Data Firehose-Service senden.

```
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");
  }
}
```

Ein vollständiges Tutorial zur Verwendung der Kinesis Data Firehose-Senke finden Sie unter the section called “Kinesis Data Firehose Senke”.

Verwendung von asynchroner I/O in Managed Service für Apache Flink

Ein asynchroner I/O-Operator reichert Stream-Daten mithilfe einer externen Datenquelle wie einer Datenbank an. Managed Service für Apache Flink reichert die Stream-Ereignisse asynchron an, sodass Anfragen zur Steigerung der Effizienz gebündelt werden können.

Weitere Informationen finden Sie unter Asynchrone I/O in der Apache Flink-Dokumentation.

Transformieren von Daten mithilfe von Operatoren in Managed Service für Apache Flink mit der DataStream-API

Um eingehende Daten in einem Managed Service für Apache Flink umzuwandeln, verwenden Sie einen Apache-Flink-Operator. Ein Apache-Flink-Operator wandelt einen oder mehrere Datenströme in einen neuen Datenstrom um. Der neue Datenstrom enthält modifizierte Daten aus dem ursprünglichen Datenstrom. Apache Flink bietet mehr als 25 vorgefertigte Operatoren zur Stream-Verarbeitung. Weitere Informationen finden Sie unter Operatoren in der Apache-Flink-Dokumentation.

Dieses Thema enthält die folgenden Abschnitte:

- Transformationsoperatoren
- Aggregationsoperator

Transformationsoperatoren

Im Folgenden finden Sie ein Beispiel für eine einfache Texttransformation in einem der Felder eines JSON-Datenstroms.

Dieser Code erstellt einen transformierten Datenstrom. Der neue Datenstrom enthält dieselben Daten wie der ursprüngliche Stream, wobei die Zeichenfolge „ `Company`“ an den Inhalt des `TICKER`-Felds angehängt wird.

```
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");
        }
    }
);
```

Aggregationsoperator

Es folgt ein Beispiel für einen Aggregationsoperator. Der Code erstellt einen aggregierten Datenstrom. Der Operator erstellt ein 5-sekündiges rollierendes Fenster und gibt die Summe der `PRICE`-Werte für die Datensätze im Fenster mit demselben `TICKER`-Wert zurück.

```
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;
});
```

Ein vollständiges Codebeispiel, das Operatoren verwendet, finden Sie unter Erste Schritte (DataStream API). Der Quellcode für die Erste-Schritte-Anwendung ist unter Getting Started im GitHub-Repository Managed Service for Apache Flink Java Examples verfügbar.

Verfolgen von Ereignissen in Managed Service für Apache Flink mithilfe der DataStream-API

Managed Service für Apache Flink verfolgt Ereignisse mit den folgenden Zeitstempeln:

- Verarbeitungszeit: Bezieht sich auf die Systemzeit der Maschine, die den jeweiligen Vorgang ausführt.
- Ereigniszeit: Bezieht sich auf die Zeit, zu der jedes einzelne Ereignis auf seinem produzierenden Gerät eingetreten ist.
- Erfassungszeit: Bezieht sich auf den Zeitpunkt, zu dem Ereignisse in den Service von Managed Service für Apache Flink eingehen.

Sie legen die von der Streaming-Umgebung verwendete Zeit mittels `setStreamTimeCharacteristic` fest:

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

Weitere Informationen über Zeitstempel finden Sie unter Ereigniszeit in der Apache-Flink-Dokumentation.

Tabellen-API

Ihre Apache-Flink-Anwendung verwendet die Apache Flink Table API, um mithilfe eines relationalen Modells mit Daten in einem Stream zu interagieren. Sie verwenden die Tabellen-API, um mithilfe von Tabellenquellen auf Daten zuzugreifen, und verwenden dann Tabellenfunktionen, um Tabellendaten zu transformieren und zu filtern. Sie können Tabellendaten entweder mithilfe von API-Funktionen oder SQL-Befehlen transformieren und filtern.

In diesem Abschnitt werden folgende Themen behandelt:

- Tabellen-API Konnektoren: Diese Komponenten verschieben Daten zwischen Ihrer Anwendung und externen Datenquellen und -zielen.
- Tabellen-API Zeitattribute: In diesem Thema wird beschrieben, wie Managed Service for Apache Flink Ereignisse verfolgt, wenn die Tabellen-API verwendet wird.

Tabellen-API Konnektoren

Im Apache-Flink-Programmiermodell sind Konnektoren Komponenten, die Ihre Anwendung verwendet, um Daten aus externen Quellen, z. B. anderen AWS-Services, zu lesen oder zu schreiben.

Mit der Apache Flink Table API können Sie die folgenden Arten von Konnektoren verwenden:

- Tabellen-API Quellen: Sie verwenden Tabellen-API-Quellkonnektoren, um Tabellen innerhalb Ihrer `TableEnvironment` zu erstellen, indem Sie entweder API-Aufrufe oder SQL-Abfragen verwenden.
- Tabellen-API Senken: Sie verwenden SQL-Befehle, um Tabellendaten in externe Quellen wie ein Amazon-MSK-Thema oder einen Amazon-S3-Bucket zu schreiben.

Tabellen-API Quellen

Sie erstellen eine Tabellenquelle aus einem Datenstrom. Der folgende Code erstellt eine Tabelle aus einem Amazon-MSK-Thema:

```
//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);
```

Weitere Informationen zu Tabellenquellen finden Sie unter Tabelle und Konnektoren in der Apache-Flink-Dokumentation.

Tabellen-API Senken

Um Tabellendaten in eine Senke zu schreiben, erstellen Sie die Senke in SQL und führen dann die SQL-basierte Senke für das `StreamTableEnvironment`-Objekt aus.

Das folgende Codebeispiel zeigt, wie Tabellendaten in eine Amazon-S3-Senke geschrieben werden:

```
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");
```

Sie können den `format`-Parameter verwenden, um zu steuern, welches Format Managed Service für Apache Flink verwendet, um die Ausgabe in die Senke zu schreiben. Informationen über Formate finden Sie unter Formate in der Apache-Flink-Dokumentation.

Weitere Informationen zu Tabellensenken finden Sie unter Tabelle und Konnektoren in der Apache-Flink-Dokumentation.

Benutzerdefinierte Quellen und Senken

Sie können vorhandene Apache-Kafka-Konnektoren verwenden, um Daten zu und von anderen AWS-Services wie Amazon MSK und Amazon S3 zu senden. Für die Interaktion mit anderen Datenquellen und -zielen können Sie Ihre eigenen Quellen und Senken definieren. Weitere Informationen finden Sie unter Benutzerdefinierte Quellen und Senken in der Apache-Flink-Dokumentation.

Tabellen-API Zeitattribute

Jeder Datensatz in einem Datenstrom hat mehrere Zeitstempel, die definieren, wann Ereignisse im Zusammenhang mit dem Datensatz aufgetreten sind:

- Ereigniszeit: Ein benutzerdefinierter Zeitstempel, der definiert, wann das Ereignis eingetreten ist, durch das der Datensatz erstellt wurde.
- Erfassungszeit: Der Zeitpunkt, zu dem Ihre Anwendung den Datensatz aus dem Datenstrom abgerufen hat.
- Verarbeitungszeit: Der Zeitpunkt, zu dem Ihre Anwendung den Datensatz verarbeitet hat.

Wenn die Apache Flink Table API Fenster auf der Grundlage von Datensatzzeiten erstellt, definieren Sie mithilfe der Methode setStreamTimeCharacteristic, welche dieser Zeitstempel verwendet werden.

Weitere Informationen zur Verwendung von Zeitstempeln mit der Tabellen-API finden Sie unter Zeitattribute in der Apache-Flink-Dokumentation.

Verwenden von Python mit Managed Service für Apache Flink

> **Note**
>
> Wenn Sie eine Python-Flink-Anwendung auf einem neuen Mac mit Apple-Silicon-Chip entwickeln, können einige bekannte Probleme mit den Python-Abhängigkeiten von PyFlink 1.15 auftreten. In diesem Fall empfehlen wir, den Python-Interpreter in Docker auszuführen. Schrittweise Anleitungen hierzu finden Sie unter PyFlink 1.15 Entwicklung auf Apple Silicon Mac.

Apache Flink Version 1.15.2 bietet Unterstützung für die Erstellung von Anwendungen mit Python Version 3.8 unter Verwendung der PyFlink-Bibliothek. Gehen Sie wie folgt vor, um mithilfe von Python eine Anwendung zu erstellen, die Managed Service für Apache Flink nutzt:

- Erstellen Sie Ihren Python-Anwendungscode als Textdatei mit einer `main`-Methode.
- Bündeln Sie Ihre Anwendungscodedatei und alle Python- oder Java-Abhängigkeiten in einer ZIP-Datei und laden Sie sie in einen Amazon-S3-Bucket hoch.
- Erstellen Sie Ihre Anwendung, die Managed Service für Apache Flink nutzt, und geben Sie dabei Ihren Amazon-S3-Codespeicherort sowie die Anwendungseigenschaften und die Anwendungseinstellungen an.

Auf einer hohen Ebene ist die Python Table API ein Wrapper rund um die Java Table API. Informationen zur Python Table API finden Sie unter Einführung in die Python Table API in der Apache-Flink-Dokumentation.

Programmieren Ihrer Python-Anwendung, die Managed Service für Apache Flink nutzt

Sie programmieren Ihre Python-Anwendung, die Managed Service für Apache Flink nutzt, mithilfe der Apache Flink Python Table API. Die Apache-Flink-Engine übersetzt Python-Table-API-Anweisungen (die in der Python-VM ausgeführt werden) in Java-Table-API-Anweisungen (die in der Java-VM ausgeführt werden).

Sie verwenden die Python Table API folgendermaßen:

- Erstellen Sie eine Referenz auf die `StreamTableEnvironment`.
- Erstellen Sie `table`-Objekte aus Ihren Quell-Streaming-Daten, indem Sie Abfragen für die `StreamTableEnvironment`-Referenz ausführen.
- Führen Sie Abfragen an Ihren `table`-Objekten aus, um Ausgabetabellen zu erstellen.
- Schreiben Sie Ihre Ausgabetabellen mit einem `StatementSet` an Ihre Ziele.

Informationen zu den ersten Schritten mit der Python Table API in Managed Service für Apache Flink finden Sie unter Erste Schritte mit Amazon Managed Service für Apache Flink für Python.

Lesen und schreiben von Streaming-Daten

Um Streaming-Daten zu lesen und zu schreiben, führen Sie SQL-Abfragen in der Tabellenumgebung aus.

Erstellen einer Tabelle

Das folgende Codebeispiel demonstriert eine benutzerdefinierte Funktion, die eine SQL-Abfrage erstellt. Die SQL-Abfrage erstellt eine Tabelle, die mit einem Kinesis-Stream interagiert:

```
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)
```

Lesen von Streaming-Daten

Das folgende Codebeispiel zeigt, wie die vorherige `CreateTable`-SQL-Abfrage für eine Tabellenumgebungsreferenz zum Lesen von Daten verwendet wird:

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

Schreiben von Streaming-Daten

Das folgende Codebeispiel zeigt, wie Sie die SQL-Abfrage aus dem `CreateTable`-Beispiel verwenden, um eine Ausgabetabellenreferenz zu erstellen, und wie Sie ein `StatementSet` verwenden, um mit den Tabellen zu interagieren und Daten in einen Kinesis-Ziel-Stream zu schreiben:

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

Lesen von Laufzeiteigenschaften

Sie können Laufzeiteigenschaften verwenden, um Ihre Anwendung zu konfigurieren, ohne Ihren Anwendungscode zu ändern.

Sie geben Anwendungseigenschaften für Ihre Anwendung auf die gleiche Weise an wie bei einer Java-Anwendung, die Managed Service für Apache Flink nutzt. Sie können Laufzeiteigenschaften auf folgende Weise angeben:

- Verwenden der Aktion CreateApplication.
- Verwenden der Aktion CreateApplication.
- Konfigurieren Ihrer Anwendung mit Hilfe der Konsole.

Sie rufen Anwendungseigenschaften im Code ab, indem Sie eine JSON-Datei mit dem Namen `application_properties.json` auslesen, die von der Laufzeit von Managed Service for Apache Flink erstellt wird.

Das folgende Codebeispiel zeigt das Lesen von Anwendungseigenschaften aus der Datei `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)
```

Das folgende Beispiel für einen benutzerdefinierten Funktionscode demonstriert das Lesen einer Eigenschaftsgruppe aus dem Anwendungseigenschaftenobjekt: ruft ab:

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

Das folgende Codebeispiel zeigt das Lesen einer Eigenschaft namens INPUT_STREAM_KEY aus einer Eigenschaftsgruppe, die im vorherigen Beispiel zurückgegeben wurde:

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

Erstellen des Codepakets Ihrer Anwendung

Sobald Sie Ihre Python-Anwendung erstellt haben, bündeln Sie Ihre Codedatei und Abhängigkeiten in einer ZIP-Datei.

Ihre ZIP-Datei muss ein Python-Skript mit einer `main`-Methode enthalten und kann optional Folgendes enthalten:

- Zusätzliche Python-Codedateien
- Benutzerdefinierter Java-Code in JAR-Dateien
- Java-Bibliotheken in JAR-Dateien

> **Note**
>
> Ihre Anwendungs-ZIP-Datei muss alle Abhängigkeiten für Ihre Anwendung enthalten. Sie können für Ihre Anwendung nicht auf Bibliotheken aus anderen Quellen verweisen.

Erstellen Ihrer Python-Anwendung, die Managed Service für Apache Flink nutzt

Angeben Ihrer Codedateien

Sobald Sie das Codepaket für Ihre Anwendung erstellt haben, laden Sie es in einen Amazon-S3-Bucket hoch. Anschließend erstellen Sie Ihre Anwendung entweder über die Konsole oder die Aktion CreateApplication.

Wenn Sie Ihre Anwendung mit der Aktion CreateApplication erstellen, geben Sie die Codedateien und Archive in Ihrer ZIP-Datei mithilfe einer speziellen Anwendungseigenschaftengruppe namens `kinesis.analytics.flink.run.options` an. Sie können die folgenden Dateitypen definieren:

- python: Eine Textdatei, die eine Python-main-Methode enthält.
- jarfile: Eine Java-JAR-Datei, die benutzerdefinierte Java-Funktionen enthält.
- pyFiles: Eine Python-Ressourcendatei, die Ressourcen enthält, die von der Anwendung verwendet werden sollen.
- PyArchives: Eine ZIP-Datei mit Ressourcendateien für die Anwendung.

Weitere Informationen zu Python-Codedateitypen in Apache Flink finden Sie unter Befehlszeilenverwendung in der Apache-Flink-Dokumentation.

> **Note**
>
> Managed Service für Apache Flink unterstützt nicht die Dateitypen `pyModule`, `pyExecutable` oder `pyRequirements`. Der gesamte Code, die Anforderungen und Abhängigkeiten müssen sich in Ihrer ZIP-Datei befinden. Sie können keine Abhängigkeiten angeben, die mit pip installiert werden sollen.

Der folgende JSON-Beispielausschnitt zeigt, wie Sie Dateispeicherorte in der ZIP-Datei Ihrer Anwendung angeben:

```
"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"
        }
      },
```

Überwachen Ihrer Python-Anwendung, die Managed Service für Apache Flink nutzt

Sie verwenden das CloudWatch-Protokoll Ihrer Anwendung, um Ihre Python-Anwendung, die Managed Service für Apache Flink nutzt, zu überwachen.

Managed Service für Apache Flink protokolliert die folgenden Meldungen für Python-Anwendungen:

- Nachrichten, die mithilfe von `print()` in der `main`-Methode der Anwendung in die Konsole geschrieben wurden.
- Nachrichten, die mithilfe des `logging`-Pakets in benutzerdefinierten Funktionen gesendet werden. Das folgende Codebeispiel zeigt, wie eine benutzerdefinierte Funktion in das Anwendungsprotokoll schreibt:

  ```
  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
  ```

- Von der Anwendung ausgelöste Fehlermeldungen.

  Wenn die Anwendung in der `main`-Funktion eine Ausnahme auslöst, wird diese in den Protokollen Ihrer Anwendung angezeigt.

  Das folgende Beispiel zeigt einen Protokolleintrag für eine Ausnahme, die aus dem Python-Code ausgelöst wurde:

  ```
  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**
>
> Aufgrund von Leistungsproblemen empfehlen wir, während der Anwendungsentwicklung nur benutzerdefinierte Protokollnachrichten zu verwenden.

Abfragen von Protokollen mit CloudWatch Insights

Die folgende Abfrage in CloudWatch Insights sucht nach Protokollen, die vom Python-Einstiegspunkt erstellt wurden, während die Hauptfunktion Ihrer Anwendung ausgeführt wird:

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

Laufzeiteigenschaften in Managed Service für Apache Flink

Sie können Laufzeiteigenschaften verwenden, um Ihre Anwendung zu konfigurieren, ohne Ihren Anwendungscode neu zu kompilieren.

Dieses Thema enthält die folgenden Abschnitte:

- Arbeiten mit Laufzeiteigenschaften in der Konsole

- Arbeiten mit Laufzeiteigenschaften in der CLI
- Zugriff auf Laufzeiteigenschaften in einer Anwendung, die Managed Service für Apache Flink nutzt

Arbeiten mit Laufzeiteigenschaften in der Konsole

Mithilfe der Konsole können Sie Laufzeiteigenschaften zu Ihrer Anwendung, die Managed Service for Apache Flink nutzt, hinzufügen, aktualisieren oder daraus entfernen.

> **Note**
>
> Sie können keine Laufzeiteigenschaften hinzufügen, wenn Sie eine Anwendung in der Konsole von Managed Service für Apache Flink erstellen.

Aktualisieren von Laufzeiteigenschaften für eine Anwendung, die Managed Service für Apache Flink nutzt

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie Ihre Anwendung, die Managed Service für Apache Flink nutzt. Wählen Sie Anwendungsdetails aus.
3. Wählen Sie auf der Seite für Ihre Anwendung Konfigurieren aus.
4. Erweitern Sie den Bereich Eigenschaften.
5. Verwenden Sie die Steuerelemente im Abschnitt Eigenschaften, um eine Eigenschaftsgruppe mit Schlüssel-Wert-Paaren zu definieren. Verwenden Sie diese Steuerelemente, um Eigenschaftsgruppen und Laufzeiteigenschaften hinzuzufügen, zu aktualisieren oder zu entfernen.
6. Wählen Sie Aktualisieren aus.

Arbeiten mit Laufzeiteigenschaften in der CLI

Sie können Laufzeiteigenschaften mit der AWS CLI hinzufügen, aktualisieren oder entfernen.

Dieser Abschnitt enthält Beispielanfragen für API-Aktionen zur Konfiguration von Laufzeiteigenschaften für eine Anwendung. Weitere Informationen zur Verwendung einer JSON-Datei für die Eingabe einer API-Aktion finden Sie unter Beispielcode für Managed Service für Apache Flink.

> **Note**
>
> Ersetzen Sie die beispielhafte Konto-ID (*012345678901*) im folgenden Beispiel durch Ihre tatsächliche Konto-ID.

Hinzufügen von Laufzeiteigenschaften beim Erstellen einer Anwendung

Die folgende Beispielanfrage für die `CreateApplication`-Aktion fügt zwei Laufzeiteigenschaftsgruppen (`ProducerConfigProperties` und `ConsumerConfigProperties`) hinzu, wenn Sie eine Anwendung erstellen:

```
{
    "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"
               }
```

```
            }
         ]
      }
   }
}
```

Hinzufügen und Aktualisieren von Laufzeiteigenschaften in einer vorhandenen Anwendung

Mit der folgenden Beispielanfrage für die `UpdateApplication`-Aktion werden Laufzeiteigenschaften für eine bestehende Anwendung hinzugefügt oder aktualisiert:

```
{
  "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**
>
> Wenn Sie in einer Eigenschaftsgruppe einen Schlüssel verwenden, für den keine entsprechende Laufzeiteigenschaft vorhanden ist, fügt Managed Service für Apache Flink das Schlüssel-Wert-Paar als neue Eigenschaft hinzu. Wenn Sie einen Schlüssel für eine

vorhandene Laufzeiteigenschaft in einer Eigenschaftsgruppe verwenden, aktualisiert Managed Service für Apache Flink den Eigenschaftswert.

Entfernen von Laufzeiteigenschaften

Mit der folgenden Beispielanfrage für die `UpdateApplication`-Aktion werden alle Laufzeiteigenschaften und Eigenschaftsgruppen aus einer bestehenden Anwendung entfernt:

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

Important

Wenn Sie eine vorhandene Eigenschaftsgruppe oder einen vorhandenen Eigenschaftsschlüssel in einer Eigenschaftsgruppe weglassen, wird diese Eigenschaftsgruppe oder Eigenschaft entfernt.

Zugriff auf Laufzeiteigenschaften in einer Anwendung, die Managed Service für Apache Flink nutzt

Sie rufen Laufzeiteigenschaften in Ihrem Java-Anwendungscode mithilfe der statischen `KinesisAnalyticsRuntime.getApplicationProperties()`-Methode ab, die ein `Map<String, Properties>`-Objekt zurückgibt.

Im folgenden Java-Codebeispiel werden Laufzeiteigenschaften für Ihre Anwendung abgerufen:

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

Sie rufen eine Eigenschaftsgruppe (als `Java.Util.Properties`-Objekt) wie folgt ab:

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

Normalerweise konfigurieren Sie eine Apache-Flink-Quelle oder -Senke, indem Sie das `Properties`-Objekt übergeben, ohne die einzelnen Eigenschaften abrufen zu müssen. Das folgende Codebeispiel zeigt, wie Sie eine Flink-Quelle erstellen, indem Sie ein `Properties`-Objekt übergeben, das aus Laufzeiteigenschaften abgerufen wurde:

```
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;
}
```

Ein vollständiges Codebeispiel, das Laufzeiteigenschaften verwendet, finden Sie unter Erste Schritte (DataStream API). Der Quellcode für die Erste-Schritte-Anwendung ist unter Getting Started im GitHub-Repository Managed Service for Apache Flink Java Examples verfügbar.

Implementierung von Fehlertoleranz in Managed Service für Apache Flink

Prüfpunktprüfung ist die Methode, die zur Implementierung von Fehlertoleranz in Amazon Managed Service für Apache Flink verwendet wird. Ein Prüfpunkt ist ein aktuelles Backup einer laufenden Anwendung, das zur sofortigen Wiederherstellung nach einer unerwarteten Anwendungsunterbrechung oder einem Failover verwendet wird.

Einzelheiten zur Prüfpunktprüfung in Apache Flink-Anwendungen finden Sie unter Prüfpunkte in der Apache Flink-Dokumentation.

Ein Snapshot ist ein manuell erstelltes und verwaltetes Backup des Anwendungsstatus. Mit Snapshots können Sie Ihre Anwendung durch einen Aufruf von `UpdateApplication`in einen früheren Zustand zurückversetzen. Weitere Informationen finden Sie unter Verwaltung von Anwendungs-Backups mithilfe von Snapshots.

Wenn Prüfpunktprüfung für Ihre Anwendung aktiviert ist, bietet der Dienst Fehlertoleranz, indem er bei unerwarteten Anwendungsneustarts Backups der Anwendungsdaten erstellt und lädt. Diese unerwarteten Anwendungsneustarts können durch unerwartete Jobneustarts, Instancefehler usw. verursacht werden. Dadurch erhält die Anwendung dieselbe Semantik wie eine fehlerfreie Ausführung bei diesen Neustarts.

Wenn Snapshots für die Anwendung aktiviert und mithilfe der ApplicationRestoreConfiguration der Anwendung konfiguriert sind, stellt der Service bei Anwendungsupdates oder während der servicebezogenen Skalierung oder Wartung die Semantik für die Verarbeitung exakt einmal bereit.

Konfigurieren von Prüfpunktprüfung in Managed Service für Apache Flink

Sie können das Prüfpunktprüfungs-Verhalten Ihrer Anwendung konfigurieren. Sie können festlegen, ob sie den Prüfpunktprüfungs-Status beibehält, wie oft sie ihren Status an Prüfpunkten speichert und das Mindestintervall zwischen dem Ende einer Prüfpunkt-Operation und dem Beginn einer anderen.

Sie konfigurieren die folgenden Einstellungen mithilfe der `CreateApplication`- oder `UpdateApplication`-API-Operationen:

- `CheckpointingEnabled` – Gibt an, ob Prüfpunktprüfung in der Anwendung aktiviert ist.
- `CheckpointInterval` – Enthält die Zeit in Millisekunden zwischen Prüfpunkt-Vorgängen (Persistenzoperationen).
- `ConfigurationType` – Setzen Sie diesen Wert auf `DEFAULT`, um das standardmäßige Prüfpunktprüf-Verhalten zu verwenden. Setzen Sie diesen Wert auf `CUSTOM`, um andere Werte zu konfigurieren.

> **Note**
>
> Das Standardverhalten von Prüfpunkten ist wie folgt:
>
> - CheckpointingEnabled: richtig
> - CheckpointInterval: 60000
> - MinPauseBetweenCheckpoints: 5000
>
> Wenn ConfigurationType auf `DEFAULT` festgelegt ist, werden die vorherigen Werte verwendet, auch wenn sie mit der AWS Command Line Interface oder durch Festlegen der Werte im Anwendungscode auf andere Werte festgelegt sind.

> **Note**
>
> Ab Flink 1.15 verwendet Managed Service für Apache Flink `stop-with-savepoint` während der automatischen Snapshot-Erstellung, d. h. beim Aktualisieren, Skalieren oder Stoppen von Anwendungen.

- `MinPauseBetweenCheckpoints` – Die Mindestzeit in Millisekunden zwischen dem Ende einer Prüfpunkt-Operation und dem Beginn einer anderen. Wenn dieser Wert festgelegt ist, verhindert dies, dass die Anwendung fortlaufende Prüfpunktprüfung durchführt, wenn eine Prüfpunkt-Operation länger dauert als das `CheckpointInterval`.

Beispiele für Prüfpunktprüfungs-APIs

Dieser Abschnitt enthält Beispielanfragen für API-Aktionen zur Konfiguration von Prüfpunktprüfung für eine Anwendung. Weitere Informationen zur Verwendung einer JSON-Datei als Eingabe für API-Aktionen finden Sie unter Beispielcode für Managed Service für Apache Flink.

Konfigurieren Sie Prüfpunktprüfung für eine neue Anwendung

In der folgenden Beispielanforderung für die `CreateApplication`-Aktion wird Prüfpunktprüfung konfiguriert, wenn Sie eine Anwendung erstellen:

```
{
   "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
         }
      }
}
```

Deaktivieren Sie Prüfpunktprüfung für eine neue Anwendung

Die folgende Beispielanforderung für die `CreateApplication`-Aktion deaktiviert Prüfpunktprüfung, wenn Sie eine Anwendung erstellen:

```
{
   "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"
         }
      }
}
```

Konfigurieren Sie Prüfpunktprüfung für eine bestehende Anwendung

Die folgende Beispielanforderung für die `UpdateApplication`-Aktion konfiguriert Prüfpunktprüfung für eine bestehende Anwendung:

```
{
   "ApplicationName": "MyApplication",
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "CheckpointConfigurationUpdate": {
            "CheckpointingEnabledUpdate": true,
```

```
            "CheckpointIntervalUpdate": 20000,
            "ConfigurationTypeUpdate": "CUSTOM",
            "MinPauseBetweenCheckpointsUpdate": 10000
         }
      }
   }
}
```

Deaktivieren Sie Prüfpunktprüfung für eine bestehende Anwendung

Die folgende Beispielanforderung für die `UpdateApplication`-Aktion deaktiviert Prüfpunktprüfung für eine bestehende Anwendung:

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

Verwaltung von Anwendungs-Backups mithilfe von Snapshots

Ein Snapshot ist die Managed Service für Apache Flink-Implementierung eines Apache Flink Savepoint. Ein Snapshot ist ein vom Benutzer oder Service ausgelöstes, erstelltes und verwaltetes Backup des Anwendungsstatus. Informationen zu Apache Flink Savepoints finden Sie unter Savepoints in der Apache Flink-Dokumentation. Mithilfe von Snapshots können Sie eine Anwendung von einem bestimmten Snapshot des Anwendungsstatus aus neu starten.

> **Note**
>
> Wir empfehlen, dass Ihre Anwendung mehrmals täglich einen Snapshot erstellt, um einen ordnungsgemäßen Neustart mit den korrekten Statusdaten zu gewährleisten. Die richtige Häufigkeit für Ihre Snapshots hängt von der Geschäftslogik Ihrer Anwendung ab. Durch

häufiges Erstellen von Snapshots können Sie neuere Daten wiederherstellen. Dies erhöht jedoch die Kosten und erfordert mehr Systemressourcen.

In Managed Service für Apache Flink verwalten Sie Snapshots mit den folgenden API-Aktionen:

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

Informationen zur Beschränkung der Anzahl von Snapshots pro Anwendung finden Sie unter Kontingent. Wenn Ihre Anwendung das Limit für Snapshots erreicht, schlägt das manuelle Erstellen eines Snapshots mit einer `LimitExceededException` fehl.

Managed Service für Apache Flink löscht Snapshots niemals. Sie müssen die Snapshots mit der Aktion `DeleteApplicationSnapshot` manuell löschen.

Um beim Starten einer Anwendung einen gespeicherten Snapshot des Anwendungsstatus zu laden, verwenden Sie den `ApplicationRestoreConfiguration`-Parameter der `StartApplication`- oder `UpdateApplication`-Aktion.

Dieses Thema enthält die folgenden Abschnitte:

- Automatische Snapshotserstellung
- Wiederherstellung aus einem Snapshot, der inkompatible Statusdaten enthält
- Snapshots API-Beispiele

Automatische Snapshotserstellung

Wenn `SnapshotsEnabled` in der ApplicationSnapshotConfiguration für die Anwendung auf `true` gesetzt ist, erstellt und verwendet Managed Service für Apache Flink automatisch Snapshots, wenn die Anwendung aktualisiert, skaliert oder gestoppt wird, um eine Semantik für die Verarbeitung genau einmal bereitzustellen.

> **Note**
>
> Die Einstellung von `ApplicationSnapshotConfiguration::SnapshotsEnabled` auf `false` führt bei Anwendungsupdates zu Datenverlust.

> **Note**
>
> Managed Service für Apache Flink löst Zwischen-Savepoints während der Snapshoterstellung aus. Bei Flink Version 1.15 oder höher haben Zwischen-Savepoints keine Nebenwirkungen mehr. Siehe Savepoints auslösen

Automatisch erstellte Snapshots haben die folgenden Eigenschaften:

- Der Snapshot wird vom Service verwaltet, Sie können den Snapshot jedoch mithilfe der Aktion ListApplicationSnapshots anzeigen. Automatisch erstellte Snapshots werden auf Ihr Snapshot-Limit angerechnet.
- Wenn Ihre Anwendung das Snapshot-Limit überschreitet, schlagen manuell erstellte Snapshots fehl, aber der Managed Service für Apache Flink-Service erstellt weiterhin erfolgreich Snapshots, wenn die Anwendung aktualisiert, skaliert oder gestoppt wird. Sie müssen Snapshots mithilfe der Aktion DeleteApplicationSnapshot manuell löschen, bevor Sie weitere Snapshots manuell erstellen können.

Wiederherstellung aus einem Snapshot, der inkompatible Statusdaten enthält

Da Snapshots Informationen über Operatoren enthalten, kann das Wiederherstellen von Zustandsdaten aus einem Snapshot für einen Operator, der sich seit der vorherigen Anwendungsversion geändert hat, zu unerwarteten Ergebnissen führen. Eine Anwendung schlägt fehl, wenn sie versucht, Zustandsdaten aus einem Snapshot wiederherzustellen, der nicht dem aktuellen Operator entspricht. Die fehlerhafte Anwendung bleibt entweder im `UPDATING`- oder `STOPPING`-Status hängen.

Um einer Anwendung die Wiederherstellung von einem Snapshot zu ermöglichen, der inkompatible Statusdaten enthält, setzen Sie den `AllowNonRestoredState`-Parameter der FlinkRunConfiguration mit der Aktion UpdateApplication auf `true`.

Wenn eine Anwendung aus einem veralteten Snapshot wiederhergestellt wird, tritt das folgende Verhalten auf:

- Operator hinzugefügt: Wenn ein neuer Operator hinzugefügt wird, hat der Savepoint keine Statusdaten für den neuen Operator. Es tritt kein Fehler auf und es ist nicht nötig, `AllowNonRestoredState` festzulegen.
- Operator gelöscht: Wenn ein vorhandener Operator gelöscht wird, enthält der Savepoint Statusdaten für den fehlenden Operator. Es tritt ein Fehler auf, sofern `AllowNonRestoredState` nicht auf `true` festgelegt ist.
- Operator geändert: Wenn kompatible Änderungen vorgenommen werden, z. B. wenn der Typ eines Parameters in einen kompatiblen Typ geändert wird, kann die Anwendung die Daten aus dem veralteten Snapshot wiederherstellen. Weitere Informationen zur Wiederherstellung aus Snapshots finden Sie unter Savepoints in der Apache Flink-Dokumentation. Eine Anwendung, die Apache Flink Version 1.8 oder höher verwendet, kann möglicherweise aus einem Snapshot mit einem anderen Schema wiederhergestellt werden. Eine Anwendung, die Apache Flink Version 1.6 verwendet, kann nicht wiederhergestellt werden. Für zweiphasige Commit-Senken empfehlen wir, einen System-Snapshot (SwS) anstelle eines vom Benutzer erstellten Snapshots (CreateApplicationSnapshot) zu verwenden.

  Für Flink löst Managed Service für Apache Flink während der Snapshot-Erstellung Zwischen-Savepoints aus. Ab Flink 1.15 haben Zwischen-Savepoints keine Nebenwirkungen mehr. Siehe Savepoints auslösen.

Wenn Sie eine Anwendung fortsetzen müssen, die mit vorhandenen Savepoint-Daten nicht kompatibel ist, empfehlen wir, die Wiederherstellung aus dem Snapshot zu überspringen, indem Sie den `ApplicationRestoreType`-Parameter der StartApplication-Aktion auf `SKIP_RESTORE_FROM_SNAPSHOT` festlegen.

Weitere Informationen darüber, wie Apache Flink mit inkompatiblen Statusdaten umgeht, finden Sie unter State Schema Evolution in der Apache Flink-Dokumentation.

Snapshots API-Beispiele

Dieser Abschnitt enthält Beispielanfragen für API-Aktionen zur Verwendung von Snapshots mit einer Anwendung. Weitere Informationen zur Verwendung einer JSON-Datei als Eingabe für API-Aktionen finden Sie unter Beispielcode für Managed Service für Apache Flink.

Snapshots für eine Anwendung aktivieren

In der folgenden Beispiel-Anfrage für die Aktion `UpdateApplication` werden Tags für eine Anwendung aktiviert:

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

Snapshot erstellen

Die folgende Beispielanforderung für die `CreateApplicationSnapshot`-Aktion erstellt einen Snapshot des aktuellen Anwendungsstatus:

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

Listet die Snapshots für eine Anwendung auf

In der folgenden Beispielanforderung für die `ListApplicationSnapshots`-Aktion werden die ersten 50 Snapshots für den aktuellen Anwendungsstatus aufgeführt:

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

Details für einen Snapshot für eine Anwendung aufführen

In der folgenden Beispiel-Anfrage für die Aktion `DescribeApplicationSnapshot` werden Details für eines bestimmten Anwendungssnapshot aufgelistet:

```
{
```

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

Löschen eines Snapshots

Die folgende Beispielanforderung für die DeleteApplicationSnapshot-Aktion löscht einen zuvor gespeicherten Snapshot. Sie können den SnapshotCreationTimestamp-Wert entweder mit ListApplicationSnapshots oder DeleteApplicationSnapshot abrufen:

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

Starten Sie eine Anwendung mithilfe eines benannten Snapshots neu

Mit der folgenden Beispielanforderung für die StartApplication-Aktion wird die Anwendung mit dem gespeicherten Status eines bestimmten Snapshots gestartet:

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

Starten Sie eine Anwendung mit dem neuesten Snapshot neu

Mit der folgenden Beispielanforderung für die StartApplication-Aktion wird die Anwendung mit dem neuesten Snapshot gestartet:

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

```
    }
}
```

Starten Sie eine Anwendung ohne Snapshot neu

Mit der folgenden Beispielanforderung für die `StartApplication`-Aktion wird die Anwendung gestartet, ohne den Anwendungsstatus zu laden, auch wenn ein Snapshot vorhanden ist:

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

Anwendungsskalierung in Managed Service für Apache Flink

Sie können die parallele Ausführung von Aufgaben und die Zuweisung von Ressourcen für Amazon Managed Service für Apache Flink konfigurieren, um die Skalierung zu implementieren. Weitere Informationen darüber, wie Apache Flink parallele Instances verwaltet, finden Sie unter Parallele Ausführung in der Dokumentation von Apache Flink.

Themen

- Konfigurieren von Anwendungsparallelität und ParallelismPerKPU
- Zuweisen von Kinesis Processing Units
- Aktualisieren der Parallelität Ihrer Anwendung
- Automatische Skalierung

Konfigurieren von Anwendungsparallelität und ParallelismPerKPU

Sie konfigurieren die parallele Ausführung der Aufgaben Ihrer mit Managed Service für Apache Flink erstellten Anwendung (wie das Lesen aus einer Quelle oder das Ausführen eines Operators) mithilfe der folgenden `ParallelismConfiguration`-Eigenschaften:

- `Parallelism` – Verwenden Sie diese Eigenschaft, um die Standardparallelität der Apache-Flink-Anwendung festzulegen. Alle Operatoren, Quellen und Senken werden mit dieser Parallelität

ausgeführt, sofern sie nicht im Anwendungscode überschrieben werden. Der Standardwert beträgt 1; der voreingestellte Maximalwert beträgt 256.

- `ParallelismPerKPU` – Verwenden Sie diese Eigenschaft, um die Anzahl der parallelen Aufgaben festzulegen, die pro Kinesis Processing Unit (KPU) Ihrer Anwendung geplant werden können. Der Standardwert ist 1 und der Maximalwert ist 8. Bei Anwendungen mit blockierenden Vorgängen (z. B. E/A) führt ein höherer `ParallelismPerKPU`-Wert zu einer Vollauslastung der KPU-Ressourcen.

> **Note**
>
> Der Grenzwert für `Parallelism` entspricht dem Wert von `ParallelismPerKPU` multipliziert mit dem Grenzwert für KPUs (der standardmäßig 64 ist). Der KPU-Grenzwert kann erhöht werden, indem eine Erhöhung des Grenzwerts angefordert wird. Anweisungen zum Anfordern einer Erhöhung dieses Grenzwerts finden Sie unter „So fordern Sie eine Erhöhung des Grenzwerts an“ unter Service Quotas.

Informationen zur Einstellung der Aufgabenparallelität für einen bestimmten Operator finden Sie unter Einstellen der Parallelität: Operator in der Dokumentation von Apache Flink.

Zuweisen von Kinesis Processing Units

Managed Service für Apache Flink stellt Kapazität als KPUs bereit. Eine einzelne KPU bietet Ihnen 1 vCPU und 4 GB Arbeitsspeicher. Für jede zugewiesene KPU werden außerdem 50 GB Speicher für laufende Anwendungen bereitgestellt.

Managed Service für Apache Flink berechnet die KPUs, die zum Ausführen Ihrer Anwendung benötigt werden, mithilfe der Eigenschaften von `Parallelism` und `ParallelismPerKPU` wie folgt:

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

Managed Service für Apache Flink stellt Ihren Anwendungen schnell Ressourcen zur Verfügung, um auf Spitzen im Durchsatz oder bei der Verarbeitungsaktivität zu reagieren. Es entfernt Ressourcen schrittweise aus Ihrer Anwendung, nachdem die Aktivitätsspitze vorüber ist. Um die automatische Zuweisung von Ressourcen zu deaktivieren, setzen Sie den Wert von `AutoScalingEnabled` auf `false`, wie weiter unten unter Aktualisieren der Parallelität Ihrer Anwendung beschrieben.

Der Standardgrenzwert für KPUs für Ihre Anwendung ist 64. Anweisungen zum Anfordern einer Erhöhung dieses Grenzwerts finden Sie unter „So fordern Sie eine Erhöhung des Grenzwerts an“ unter Service Quotas.

> **Note**
>
> Für Orchestrierungszwecke wird eine zusätzliche KPU berechnet. Weitere Informationen finden Sie unter Managed Service für Apache Flink – Preise.

Aktualisieren der Parallelität Ihrer Anwendung

Dieser Abschnitt enthält Beispielanfragen für API-Aktionen, die die Parallelität einer Anwendung festlegen. Weitere Beispiele und Anweisungen zur Verwendung von Anforderungsblöcken mit API-Aktionen finden Sie unter Beispielcode für Managed Service für Apache Flink.

Die folgende Beispielanforderung für die `CreateApplication`-Aktion legt die Parallelität fest, wenn Sie eine Anwendung erstellen:

```
{
   "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
         }
      }
```

```
   }
}
```

Die folgende Beispielanforderung für die UpdateApplication-Aktion legt die Parallelität für eine bestehende Anwendung fest:

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

Die folgende Beispielanforderung für die UpdateApplication-Aktion deaktiviert die Parallelität für eine bestehende Anwendung:

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

Automatische Skalierung

Managed Service für Apache Flink skaliert die Parallelität Ihrer Anwendung elastisch, um dem Datendurchsatz Ihrer Quelle und der Komplexität Ihres Operators in den meisten Szenarien Rechnung zu tragen. Managed Service für Apache Flink überwacht die Ressourcenauslastung (CPU-

Auslastung) Ihrer Anwendung und skaliert die Parallelität Ihrer Anwendung entsprechend elastisch nach oben oder unten:

- Ihre Anwendung skaliert hoch (erhöht die Parallelität), wenn die CloudWatch Metrik 15 Minuten lang größer oder höher `containerCPUUtilization` ist. Das bedeutet, dass die `ScaleUp`-Aktion ausgelöst wird, wenn es 15 aufeinanderfolgende Datenpunkte mit einem Zeitraum von 1 Minute gibt, der 75 Prozent oder mehr entspricht.
- Ihre Anwendung wird herunterskaliert (Parallelität wird verringert), wenn die CPU-Auslastung sechs Stunden lang unter 10 Prozent bleibt. Das bedeutet, dass die `ScaleDown`-Aktion ausgelöst wird, wenn es 360 aufeinanderfolgende Datenpunkte mit einem Zeitraum von 1 Minute gibt, der weniger als 10 Prozent beträgt.

> **Note**
>
> Es kann ein Maximum von `containerCPUUtilization` über 1 Minute referenziert werden, um die Korrelation mit einem Datenpunkt zu ermitteln, der für die Skalierungsaktion verwendet wird. Es ist jedoch nicht erforderlich, den genauen Zeitpunkt wiederzugeben, zu dem die Aktion ausgelöst wird.

Managed Service für Apache Flink reduziert den `CurrentParallelism`-Wert Ihrer Anwendung nicht auf weniger als die `Parallelism`-Einstellung Ihrer Anwendung.

Wenn der Service von Managed Service für Apache Flink Ihre Anwendung skaliert, befindet er sich im Status `AUTOSCALING`. Sie können Ihren aktuellen Anwendungsstatus mit den ListApplications Aktionen DescribeApplication oder überprüfen. Während der Service Ihre Anwendung skaliert, können Sie die einzige gültige API-Aktion verwenden, StopApplication wenn der `Force` Parameter auf festgelegt ist`true`.

Sie können die Eigenschaft `AutoScalingEnabled` (Teil von `FlinkApplicationConfiguration`) verwenden, um das Auto-Scaling-Verhalten zu aktivieren oder zu deaktivieren. Ihr AWS-Konto wird für die von Managed Service für Apache Flink bereitgestellten KPUs belastet. Dies hängt von den Einstellungen für `parallelism` und `parallelismPerKPU` Ihrer Anwendung ab. Eine Aktivitätsspitze erhöht Ihre Kosten für Managed Service für Apache Flink.

Weitere Informationen finden Sie unter Amazon Managed Service für Apache Flink – Preise.

Beachten Sie Folgendes im Zusammenhang mit der Anwendungsskalierung:

- Die automatische Skalierung ist standardmäßig aktiviert.
- Skalierung gilt nicht für Studio-Notebooks. Wenn Sie ein Studio-Notebook jedoch als Anwendung mit dauerhaftem Zustand bereitstellen, gilt die Skalierung für die bereitgestellte Anwendung.
- Ihre Anwendung hat einen Standardgrenzwert von 64 KPUs. Weitere Informationen finden Sie unter Kontingent.
- Wenn Auto Scaling die Anwendungsparallelität aktualisiert, kommt es bei der Anwendung zu Ausfallzeiten. Gehen Sie wie folgt vor, um diese Ausfallzeit zu vermeiden:
  - Deaktivieren der automatischen Skalierung
  - Konfigurieren Sie die Ihrer Anwendung `parallelism` und `parallelismPerKPU` mit der UpdateApplication Aktion . Weitere Informationen über die Einstellung der Parallelitätseinstellungen Ihrer Anwendung finden Sie nachfolgend unter the section called “Aktualisieren der Parallelität Ihrer Anwendung”.
  - Überwachen Sie regelmäßig den Ressourcenverbrauch Ihrer Anwendung, um sicherzustellen, dass Ihre Anwendung über die richtigen Parallelitätseinstellungen für ihren Workload verfügt. Weitere Informationen über die Überwachung des Ressourcenverbrauchs finden Sie unter the section called “Metriken und Dimensionen in Managed Service für Apache Flink”.

Überlegungen zu maxParallelism

- Durch die Auto-Scaling-Logik wird verhindert, dass ein Flink-Auftrag zur Parallelität skaliert wird, was zu Interferenzen mit dem Auftrag und dem Operator `maxParallelism` führen würde. Wenn es sich beispielsweise um einen einfachen Auftrag mit nur einer Quelle und einer Senke handelt, bei der die Quelle `maxParallelism` 16 hat und die `sink` 8 hat, werden wir den Auftrag nicht automatisch auf über 8 skalieren.
- Wenn für einen Job nicht `maxParallelism` eingestellt ist, wird Flink standardmäßig 128 verwenden. Wenn Sie also der Meinung sind, dass ein Auftrag mit einer höheren Parallelität als 128 ausgeführt werden muss, müssen Sie diese Zahl für Ihre Anwendung festlegen.
- Wenn Sie erwarten, dass Ihr Auftrag automatisch skaliert wird, dies aber nicht der Fall ist, stellen Sie sicher, dass Ihre `maxParallelism`-Werte dies zulassen.

Weitere Informationen finden Sie unter Erweiterte Überwachung und automatische Skalierung für Apache Flink

Ein Beispiel finden Sie unter kda-flink-app-autoscaling.

Verwenden von Tagging

In diesem Abschnitt wird beschrieben, wie Sie Schlüssel-Wert-Metadaten-Tags zu Anwendungen, die Managed Service für Apache Flink nutzen, hinzufügen. Diese Tags können für die folgenden Zwecke verwendet werden:

- Festlegen der Abrechnung für einzelne Anwendungen, die Managed Service für Apache Flink nutzen. Weitere Informationen finden Sie unter Verwendung von Kostenzuordnungs-Tags im Benutzerhandbuch Fakturierungs- und Kostenverwaltung.
- Steuern des Zugriffs auf Anwendungsressourcen basierend auf Tags. Weitere Informationen finden Sie unter Zugriffssteuerung mit Tags im AWS Identity and Access Management Benutzerhandbuch.
- Benutzerdefinierte Zwecke. Sie können die Anwendungsfunktionalität basierend auf dem Vorhandensein von Benutzer-Tags definieren.

Bitte beachten Sie die folgenden Informationen über Tagging:

- Die maximale Anzahl an Anwendungs-Tags enthält System-Tags. Die maximale Anzahl an benutzerdefinierten Anwendungs-Tags ist 50.
- Wenn eine Aktion eine Tag-Liste beinhaltet, die doppelte `Key`-Werte enthält, löst der Service eine `InvalidArgumentException` aus.

Dieses Thema enthält die folgenden Abschnitte:

- Hinzufügen von Tags, wenn eine Anwendung erstellt wird
- Hinzufügen oder Aktualisieren von Tags für eine vorhandene Anwendung
- Auflisten von Tags für eine Anwendung
- Entfernen von Tags aus einer Anwendung

Hinzufügen von Tags, wenn eine Anwendung erstellt wird

Fügen Sie Tags beim Erstellen einer Anwendung mit dem `tags`-Parameter der Aktion CreateApplication hinzu.

Das folgende Beispiel zeigt den `Tags`-Knoten für eine `CreateApplication`-Anforderung:

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

Hinzufügen oder Aktualisieren von Tags für eine vorhandene Anwendung

Fügen Sie Tags zu einer Anwendung mithilfe der Aktion TagResource hinzu. Sie können keine Tags zu einer Anwendung mithilfe der Aktion UpdateApplication hinzufügen.

Fügen Sie zum Aktualisieren eines vorhandenen Tags ein Tag mit demselben Schlüssel wie das vorhandene Tag hinzu.

In der folgenden Beispiel-Anfrage für die Aktion`TagResource` werden neue Tags hinzugefügt oder vorhandene Tags aktualisiert:

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

Auflisten von Tags für eine Anwendung

Verwenden Sie zum Auflisten vorhandener Tags die Aktion ListTagsForResource.

In der folgenden Beispiel-Anfrage für die Aktion `ListTagsForResource` werden Tags für eine Anwendung aufgelistet:

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

Entfernen von Tags aus einer Anwendung

Verwenden Sie zum Entfernen von Tags aus einer Anwendung die Aktion UntagResource.

In der folgenden Beispiel-Anfrage für die Aktion `UntagResource` werden Tags aus einer Anwendung entfernt:

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

Verwenden von CloudFormation mit Managed Service für Apache Flink

Die folgenden Übungen zeigen, wie Sie eine Flink-Anwendung starten, die mittels AWS CloudFormation über eine Lambda-Funktion im selben Stack erstellt wurde.

Bevor Sie beginnen

Bevor Sie mit dieser Übung beginnen, folgen Sie den Schritten zum Erstellen einer Flink-Anwendung mithilfe von AWS CloudFormation unter AWS::KinesisAnalytics::Application.

Schreiben einer Lambda-Funktion

Um eine Flink-Anwendung nach der Erstellung oder Aktualisierung zu starten, verwenden wir die kinesisanalyticsv2 start-application API. Der Aufruf wird durch ein AWS CloudFormation-Ereignis nach der Erstellung der Flink-Anwendung ausgelöst. Wir werden später in dieser Übung besprechen, wie der Stack so eingerichtet wird, dass er die Lambda-Funktion auslöst, aber zuerst konzentrieren wir uns auf die Lambda-Funktionsdeklaration und ihren Code. In diesem Beispiel verwenden wir die `Python3.8`-Laufzeit.

```
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)})
```

Im vorherigen Code verarbeitet Lambda eingehende AWS CloudFormation-Ereignisse, filtert alles andere außer `Create` und `Update` heraus, ruft den Anwendungszustand ab und startet sie, wenn der Zustand `READY` ist. Um den Anwendungszustand zu ermitteln, müssen Sie die Lambda-Rolle wie folgt erstellen:

Erstellen einer Lambda-Rolle

Sie erstellen eine Rolle für Lambda, um erfolgreich mit der Anwendung zu kommunizieren und Protokolle zu schreiben. Für diese Rolle werden standardmäßig verwaltete Richtlinien verwendet, aber Sie sollten sie ggf. mithilfe benutzerdefinierter Richtlinien eingrenzen.

```
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: /
```

Beachten Sie, dass die Lambda-Ressourcen nach der Erstellung der Flink-Anwendung im selben Stack erstellt werden, da sie davon abhängen.

Aufrufen der Lambda-Funktion

Jetzt müssen Sie nur noch die Lambda-Funktion aufrufen. Dies erfolgt mithilfe einer benutzerdefinierten Ressource.

```
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
```

Das ist alles, was Sie benötigen, um Ihre Flink-Anwendung mit Lambda zu starten. Sie sind jetzt bereit, Ihren eigenen Stack zu erstellen oder anhand des vollständigen Beispiels unten zu sehen, wie all diese Schritte in der Praxis funktionieren.

Vollständiges Beispiel

Das folgende Beispiel ist eine leicht erweiterte Version der obigen Schritte mit einer zusätzlichen `RunConfiguration`-Optimierung über Vorlagenparameter. Dies ist ein funktionierender Stack, den Sie ausprobieren können. Lesen Sie unbedingt die beigefügten Hinweise:

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
```

Auch hier sollten Sie ggf. die Rollen für Lambda sowie eine Anwendung selbst anpassen.

Vergessen Sie nicht, Ihre Parameter anzugeben, bevor Sie den obigen Stack erstellen.

parameters.json

```
[
```

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

Ersetzen Sie `YOUR_BUCKET_ARN` und `YOUR_JAR` durch Ihre spezifischen Anforderungen. Sie können dieser Anleitung folgen, um einen Amazon-S3-Bucket und ein Anwendungs-Jar zu erstellen.

Erstellen Sie nun den Stack (ersetzen Sie YOUR_REGION durch eine Region Ihrer Wahl, z. B. 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
```

Sie können nun zu https://console.aws.amazon.com/cloudformation navigieren und sich den Fortschritt ansehen. Nach der Erstellung sollte Ihre Flink-Anwendung im `Starting`-Zustand angezeigt werden. Es kann einige Minuten dauern, bis es `Running` startet.

Weitere Informationen finden Sie hier:

- Vier Möglichkeiten zum Abrufen beliebiger AWS-Service-Eigenschaften mithilfe von AWS CloudFormation (Teil 1 von 3).
- Walkthrough: Nachschlagen von Amazon Machine Image-IDs.

Verwenden des Apache Flink-Dashboards mit Managed Service für Apache Flink

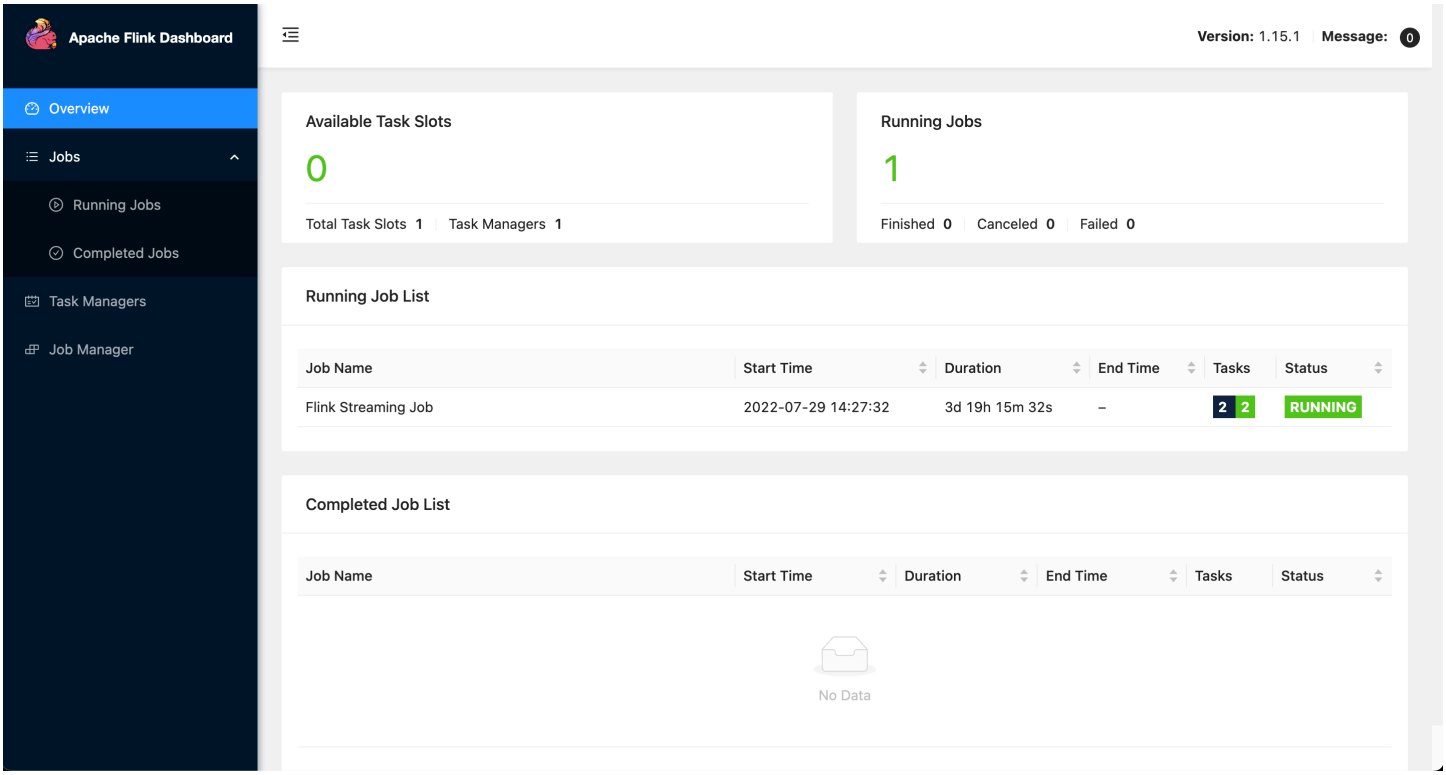
Sie können das Apache Flink-Dashboard Ihrer Anwendung verwenden, um den Zustand Ihres Managed Service für Apache Flink-Anwendung zu überwachen. Das Dashboard Ihrer Anwendung zeigt die folgenden Informationen an:

- Verwendete Ressourcen, einschließlich Task-Managern und Task-Slots.
- Informationen zu Jobs, einschließlich laufender, abgeschlossener, abgebrochener und fehlgeschlagener Jobs.

Informationen zu Apache Flink Task Managern, Task Slots und Jobs finden Sie unter Apache Flink Architektur auf der Apache Flink-Website.

Beachten Sie Folgendes zur Verwendung des Apache Flink-Dashboards mit Managed Service für Apache Flink-Anwendungen:

- Das Apache Flink Dashboard für Managed Service für Apache Flink-Anwendungen ist schreibgeschützt. Sie können mit dem Apache Flink Dashboard keine Änderungen an Ihrem Managed Service für Apache Flink-Anwendung vornehmen.
- Das Apache Flink Dashboard ist nicht mit Microsoft Internet Explorer kompatibel.



Zugriff auf das Apache Flink-Dashboard Ihrer Anwendung

Sie können auf das Apache Flink-Dashboard Ihrer Anwendung entweder über den Managed Service für Apache Flink-Konsole zugreifen oder indem Sie über die CLI einen sicheren URL-Endpunkt anfordern.

Zugriff auf das Apache Flink Dashboard Ihrer Anwendung mit Managed Service für Apache Flink Konsole

Um von der Konsole aus auf das Apache Flink Dashboard Ihrer Anwendung zuzugreifen, wählen Sie Apache Flink Dashboard auf der Seite Ihrer Anwendung.

> **Note**
>
> Wenn Sie das Dashboard von dem Managed Service für Apache Flink-Konsole aus öffnen, ist die von der Konsole generierte URL 12 Stunden lang gültig.

Zugriff auf das Apache Flink Dashboard Ihrer Anwendung mit Managed Service für Apache Flink CLI

Sie können den Managed Services für Apache Flink CLI verwenden, um eine URL für den Zugriff auf Ihr Anwendungs-Dashboard zu generieren. Die URL, die Sie generieren, ist für eine bestimmte Zeit gültig.

> **Note**
>
> Wenn Sie nicht innerhalb von drei Minuten auf die generierte URL zugreifen, ist sie nicht mehr gültig.

Sie generieren Ihre Dashboard-URL mithilfe der Aktion CreateApplicationPresignedUrl. Sie können die folgenden Werte für die Aktion angeben:

- Den Anwendungsnamen
- Die Zeit in Sekunden, über die hinweg wird die URL gültig sein
- Sie geben `FLINK_DASHBOARD_URL` als URL-Typ an.

Release-Versionen

Dieses Thema enthält Informationen zu den unterstützten Features und den empfohlenen Komponentenversionen für die einzelnen Versionen von Managed Service für Apache Flink.

Amazon Managed Service für Apache Flink Release 1.15.2

Managed Service für Apache Flink unterstützt die folgenden neuen Features in Apache 1.15.2

Feature	Beschreibung	Apache-FLIP-Referenz
Asynchrone Senke	Ein von AWS bereitgestelltes Framework für die Erstellung asynchroner Ziele, das es Entwicklern ermöglicht, benutzerdefinierte AWS-Konnektoren mit weniger als der Hälfte des bisherigen Aufwands zu erstellen. Weitere Informationen finden Sie unter Die generische asynchrone Basissenke.	FLIP-171: Asynchrone Senke.
Kinesis Data Firehose Senke	AWS hat mithilfe des Async-Frameworks eine neue Amazon Kinesis Firehose Senke bereitgestellt.	Amazon Kinesis Data Firehose Senke
Anhalten mit Savepoint	Anhalten mit Savepoint sorgt für einen sauberen Anhaltevorgang und unterstützt vor allem die Exakt-einmal-Semantik für Kunden, die sich darauf verlassen.	FLIP-34: Auftrag mit Savepoint beenden/aussetzen.

Feature	Beschreibung	Apache-FLIP-Referenz
Scala-Entkopplung	Benutzer können die Java-API jetzt von jeder Scala-Version aus nutzen, einschließlich Scala 3. Kunden müssen die Scala-Standardbibliothek ihrer Wahl in ihren Scala-Anwendungen bündeln.	FLIP-28: Langfristiges Ziel, flink-table Scala-frei zu machen.
Scala	Siehe Scala-Entkopplung oben	FLIP-28: Langfristiges Ziel, flink-table Scala-frei zu machen.
Einheitliche Konnektor-Metriken	Flink hat Standardmetriken für Aufträge, Aufgaben und Operatoren definiert. Managed Service für Apache Flink wird weiterhin Senken- und Quell-Metriken unterstützen und in 1.15 `numRestarts` parallel zu `fullRestarts` für Verfügbarkeitsmetriken einführen.	FLIP-33: Konnektor-Metriken standardisieren und FLIP-179: Standardisierte Operator-Metriken verfügbar machen.
Checkpointing von abgeschlossenen Aufgaben	Dieses Feature ist in Flink 1.15 standardmäßig aktiviert und ermöglicht es, weiterhin Checkpoints durchzuführen, auch wenn Teile des Auftragsdiagramms die Verarbeitung aller Daten abgeschlossen haben, was passieren kann, wenn er gebundene (Batch-)Quellen enthält.	FLIP-147: Checkpoints nach Abschluss von Aufgaben unterstützen.

Änderungen in Amazon Managed Service für Apache Flink mit Apache Flink 1.15

Studio-Notebooks

Managed Service für Apache Flink Studio unterstützt jetzt Apache Flink 1.15. Managed Service für Apache Flink Studio nutzt Apache-Zeppelin-Notebooks, um eine zentrale Benutzeroberfläche für die Entwicklung, das Debuggen von Code und die Ausführung von Apache-Flink-Streamverarbeitungsanwendungen bereitzustellen. Weitere Informationen über Managed Service für Apache Flink Studio und die ersten Schritte finden Sie unter Verwenden eines Studio-Notebooks mit Managed Service für Apache Flink.

EFO-Konnektor

Stellen Sie beim Upgrade auf Managed Service für Apache Flink Version 1.15 sicher, dass Sie den neuesten EFO-Konnektor verwenden, d. h. eine beliebige Version 1.15.3 oder neuer. Weitere Informationen zu den Gründen finden Sie unter FLINK-29324.

Scala-Entkopplung

Ab Flink 1.15.2 müssen Sie die Scala-Standardbibliothek Ihrer Wahl in Ihren Scala-Anwendungen bündeln.

Kinesis Data Firehose Senke

Stellen Sie beim Upgrade auf Managed Service für Apache Flink Version 1.15 sicher, dass Sie die neueste Amazon Kinesis Data Firehose Senke verwenden.

Kafka-Konnektoren

Stellen Sie beim Upgrade auf Amazon Managed Service für Apache Flink Version 1.15 sicher, dass Sie die neuesten Kafka-Konnektor-APIs verwenden. Apache Flink hat FlinkKafkaConsumer und FlinkKafkaProducer als veraltet markiert. Diese APIs für die Kafka-Senke können nicht zu Kafka für Flink 1.15 übergeben. Stellen Sie sicher, dass Sie KafkaSource und KafkaSink verwenden.

Komponenten

Komponente	Version
Java	11 (empfohlen)

Komponente	Version
Scala	2.12
Managed Service for Apache Flink Laufzeit (aws-kinesisanalytics-runtime)	1.2.0
AWS Kinesis-Konnektor (flink-connector-kinesis)	1.15.4
Apache Beam (nur Beam-Anwendungen)	2.33.0, mit Jackson-Version 2.12.2

Verwenden eines Studio-Notebooks mit Managed Service für Apache Flink

Studio-Notebooks für Managed Service für Apache Flink ermöglichen Ihnen die interaktive Abfrage von Datenströmen in Echtzeit und die einfache Erstellung und Ausführung von Stream-Verarbeitungsanwendungen mit Standard-SQL, Python und Scala. Mit ein paar Klicks in der AWS-Managementkonsole können Sie ein Serverless Notebook starten, um Datenströme abzufragen und innerhalb von Sekunden Ergebnisse zu erhalten.

Ein Notebook ist eine webbasierte Entwicklungsumgebung. Notebooks bieten ein einfaches interaktives Entwicklungserlebnis in Kombination mit den fortschrittlichen Funktionen von Apache Flink. Studio-Notebooks verwenden Notebooks, die auf Apache Zeppelin basieren, und Apache Flink als Engine für die Streamverarbeitung. Studio-Notebooks kombinieren diese Technologien nahtlos, um Entwicklern aller Qualifikationsstufen erweiterte Analysen von Datenströmen zugänglich zu machen.

Apache Zeppelin bietet für Ihre Studio-Notebooks eine komplette Suite von Analysetools, darunter die folgenden:

- Datenvisualisierung
- Exportieren der Daten in Dateien
- Kontrolle über das Ausgabeformat zur Erleichterung von Analysen

Hinweise zu den ersten Schritten mit Managed Service für Apache Flink und Apache Zeppelin finden Sie unter Tutorial zum Erstellen eines Studio-Notebooks. Weitere Informationen zu Apache Zeppelin finden Sie in der Apache-Zeppelin-Dokumentation.

Mit einem Notebook modellieren Sie Abfragen mithilfe der Apache-Flink-Tabellen-API und SQL in SQL, Python oder Scala oder DataStream API in Scala. Mit wenigen Klicks können Sie das Studio-Notebook dann zu einer kontinuierlich laufenden, nicht interaktiven Stream-Verarbeitungsanwendung, die Managed Service für Apache Flink nutzt, für Ihre Produktions-Workloads heraufstufen.

Dieses Thema enthält die folgenden Abschnitte:

- Erstellen eines Studio-Notebooks
- Interaktive Analyse von Streaming-Daten

- Bereitstellen als Anwendung mit dauerhaftem Zustand
- IAM-Berechtigungen für Studio-Notebooks
- Konnektoren und Abhängigkeiten
- Benutzerdefinierte Funktionen
- Aktivieren von Checkpointing
- Arbeiten mit AWS Glue
- Beispiele und Tutorials
- Fehlerbehebung
- Anhang: Erstellen benutzerdefinierter IAM-Richtlinien

Erstellen eines Studio-Notebooks

Ein Studio-Notebook enthält in SQL, Python oder Scala geschriebene Abfragen oder Programme, die auf Streaming-Daten ausgeführt werden und Analyseergebnisse zurückgeben. Sie erstellen Ihre Anwendung entweder mit der Konsole oder der CLI und stellen Abfragen zur Analyse der Daten aus Ihrer Datenquelle bereit.

Die Anwendung besteht aus folgenden Komponenten:

- Einer Datenquelle, z. B. einem Amazon-MSK-Cluster, einem Kinesis-Datenstrom oder einem Amazon-S3-Bucket.
- Einer AWS Glue-Datenbank. Diese Datenbank enthält Tabellen, in denen Ihre Datenquellen- und Zielschemas und Endpunkte gespeichert sind. Weitere Informationen finden Sie unter Arbeiten mit AWS Glue.
- Ihr Anwendungscode. Ihr Code implementiert Ihre Analyseabfrage oder Ihr Analyseprogramm.
- Ihre Anwendungseinstellungen und Laufzeiteigenschaften. Informationen zu Anwendungseinstellungen und Laufzeiteigenschaften finden Sie unter den folgenden Themen im Entwicklerhandbuch für Apache-Flink-Anwendungen:
  - Anwendungsparallelität und -skalierung: Sie verwenden die Parallelitätseinstellung Ihrer Anwendung, um die Anzahl der Abfragen zu steuern, die Ihre Anwendung gleichzeitig ausführen kann. Ihre Abfragen können auch von der erhöhten Parallelität profitieren, wenn sie mehrere Ausführungspfade haben, z. B. unter den folgenden Umständen:
    - Bei der Verarbeitung mehrerer Shards eines Kinesis-Datenstroms
    - Bei der Partitionierung von Daten mit dem `KeyBy`-Operator.

- Bei Verwendung mehrerer Fensteroperatoren

  Weitere Informationen zur Anwendungsskalierung finden Sie unter Anwendungsskalierung in Managed Service für Apache Flink.

- Protokollierung und Überwachung: Informationen zur Anwendungsprotokollierung und -überwachung finden Sie unter Protokollierung und Überwachung in Amazon Managed Service für Apache Flink.
- Ihre Anwendung verwendet Checkpoints und Savepoints aus Gründen der Fehlertoleranz. Checkpoints und Savepoints sind für Studio-Notebooks standardmäßig nicht aktiviert.

Sie können Ihr Studio-Notebook entweder über die AWS Management Console oder die AWS CLI erstellen.

Beim Erstellen der Anwendung über die Konsole stehen Ihnen die folgenden Optionen zur Verfügung:

- Wählen Sie in der Amazon-MSK-Konsole Ihren Cluster aus und wählen Sie dann Daten in Echtzeit verarbeiten.
- Wählen Sie in der Konsole von Kinesis Data Streams Ihren Datenstrom und dann auf der Registerkarte Anwendungen die Option Daten in Echtzeit verarbeiten aus.
- Wählen Sie in der Konsole von Managed Service für Apache Flink die Registerkarte Studio und dann Studio-Notebook erstellen aus.

Ein Tutorial finden Sie unter Ereigniserkennung mit Managed Service für Apache Flink.

Ein Beispiel für eine erweiterte Studio-Notebook-Lösung finden Sie unter Apache Flink in Amazon Managed Service für Apache Flink Studio.

Interaktive Analyse von Streaming-Daten

Sie verwenden ein Serverless Notebook mit Apache Zeppelin, um mit Ihren Streaming-Daten zu interagieren. Ihr Notebook kann mehrere Notizen enthalten, und jede Notiz kann einen oder mehrere Absätze enthalten, in die Sie Ihren Code schreiben können.

Die folgende beispielhafte SQL-Abfrage zeigt, wie Daten aus einer Datenquelle abgerufen werden:

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

```
select * from stock;
```

Weitere Beispiele für Flink-Streaming-SQL-Abfragen finden Sie nachfolgend unter Beispiele und Tutorials und unter Abfragen in der Apache-Flink-Dokumentation.

Sie können Flink-SQL-Abfragen im Studio-Notebook verwenden, um Streaming-Daten abzufragen. Sie können auch Python (Table-API) und Scala (Table- und Datastream-APIs) verwenden, um Programme zu schreiben, mit denen Sie Ihre Streaming-Daten interaktiv abfragen können. Sie können die Ergebnisse Ihrer Abfragen oder Programme anzeigen, sie innerhalb von Sekunden aktualisieren und erneut ausführen, um aktualisierte Ergebnisse anzuzeigen.

Flink-Interpreter

Sie geben mithilfe eines Interpreters an, in welcher Sprache Managed Service für Apache Flink Ihre Anwendung ausführt. Sie können die folgenden Interpreter mit Managed Service für Apache Flink verwenden:

Name	Klasse	Beschreibung
%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

Weitere Informationen zu Flink-Interpretern finden Sie unter Flink-Interpreter für Apache Zeppelin.

Wenn Sie `%flink.pyflink` oder `%flink.ipyflink` als Interpreter verwenden, müssen Sie den `ZeppelinContext` verwenden, um die Ergebnisse im Notebook zu visualisieren.

PyFlink Spezifischere Beispiele finden Sie unter Interaktives Abfragen Ihrer Datenströme mit Managed Service für Apache Flink Studio und Python.

Tabellenumgebungsvariablen von Apache Zeppelin

Apache Zeppelin bietet mithilfe von Umgebungsvariablen Zugriff auf Tabellenumgebungsressourcen.

Sie greifen mit den folgenden Variablen auf Ressourcen der Scala-Tabellenumgebung zu:

Variable	Ressource
`senv`	`StreamExecutionEnvironment`
`stenv`	`StreamTableEnvironment für Bol-Planer`

Sie greifen mit den folgenden Variablen auf Ressourcen der Python-Tabellenumgebung zu:

Variable	Ressource
`s_env`	`StreamExecutionEnvironment`
`st_env`	`StreamTableEnvironment für Bol-Planer`

Weitere Informationen zur Verwendung von Tabellenumgebungen finden Sie unter Erstellen eines TableEnvironment in der Apache-Flink-Dokumentation.

Bereitstellen als Anwendung mit dauerhaftem Zustand

Sie können Ihren Code erstellen und zu Amazon S3 exportieren. Sie können den Code, den Sie in Ihrer Notiz geschrieben haben, in eine kontinuierlich laufende Stream-Verarbeitungsanwendung umwandeln. Es gibt zwei Arten, eine Apache-Flink-Anwendung auf Managed Service für Apache Flink auszuführen: Mit einem Studio-Notebook haben Sie die Möglichkeit, Ihren Code interaktiv zu entwickeln, die Ergebnisse Ihres Codes in Echtzeit anzuzeigen und ihn in Ihrer Notiz zu visualisieren.

Nachdem Sie eine Notiz für die Ausführung im Streaming-Modus bereitgestellt haben, erstellt Managed Service für Apache Flink eine Anwendung für Sie, die kontinuierlich ausgeführt wird, Daten aus Ihren Quellen liest, in Ihre Ziele schreibt, den Status lang laufender Anwendungen beibehält und automatisch auf der Grundlage des Durchsatzes Ihrer Quellströme skaliert.

> **Note**
>
> Der S3-Bucket, in den Sie den Anwendungscode exportieren, muss sich in derselben Region wie Ihr Studio-Notebook befinden.

Sie können eine Notiz aus Ihrem Studio-Notebook nur bereitstellen, wenn sie die folgenden Kriterien erfüllt:

- Die Absätze müssen der Reihe nach angeordnet werden. Wenn Sie Ihre Anwendung bereitstellen, werden alle Absätze innerhalb einer Notiz sequenziell (left-to-right, top-to-bottom) ausgeführt, wie sie in Ihrer Notiz erscheinen. Sie können diese Reihenfolge überprüfen, indem Sie in Ihrer Notiz Alle Absätze ausführen wählen.
- Ihr Code ist eine Kombination aus Python und SQL oder Scala und SQL. Python und Scala werden derzeit nicht zusammen für unterstützt deploy-as-application.
- Ihre Notiz darf nur die folgenden Interpreter enthalten: `%flink`, `%flink.ssql`, `%flink.pyflink`, `%flink.ipyflink`, `%md`.
- Die Verwendung des Zeppelin-Kontext-Objekts z wird nicht unterstützt. Methoden, die nichts zurückgeben, tun nichts, außer eine Warnung zu protokollieren. Andere Methoden lösen Python-Ausnahmen aus oder können nicht in Scala kompiliert werden.
- Eine Notiz muss zu einem einzigen Apache-Flink-Auftrag führen.
- Notizen mit dynamischen Formularen werden für die Bereitstellung als Anwendung nicht unterstützt.
- %md-Absätze (Markdown) werden bei der Bereitstellung als Anwendung übersprungen, da davon ausgegangen wird, dass sie menschenlesbare Dokumentation enthalten, die für die Ausführung als Teil der resultierenden Anwendung nicht geeignet ist.
- Absätze, die für die Ausführung in Zeppelin deaktiviert sind, werden bei der Bereitstellung als Anwendung übersprungen. Selbst wenn ein deaktivierter Absatz einen inkompatiblen Interpreter verwendet, z. B. `%flink.ipyflink` in einer Notiz mit `%flink- and %flink.ssql`-Interpretern, wird er bei der Bereitstellung der Notiz als Anwendung übersprungen und führt nicht zu einem Fehler.

- Es muss mindestens einen Absatz mit Quellcode (Flink SQL PyFlink oder Flink Scala) vorhanden sein, der für die Ausführung der Anwendungsbereitstellung aktiviert ist, damit die Anwendungsbereitstellung erfolgreich ist.
- Das Einstellen von Parallelität in der Interpreter-Direktive innerhalb eines Absatzes (z. B. `%flink.ssql(parallelism=32)`) wird in Anwendungen, die über eine Notiz bereitgestellt werden, ignoriert. Stattdessen können Sie die bereitgestellte Anwendung über die AWS Command Line Interface oder die AWS API aktualisierenAWS Management Console, um die Parallelitäts- und/oder ParallelismPerKPU-Einstellungen entsprechend der Parallelitätsstufe zu ändern, die Ihre Anwendung benötigt, oder Sie können die automatische Skalierung für Ihre bereitgestellte Anwendung aktivieren.
- Wenn Sie als Anwendung mit einem dauerhaften Zustand bereitstellen, muss Ihre VPC über Internetzugang verfügen. Wenn Ihre VPC keinen Internetzugang hat, finden Sie weitere Informationen unter Bereitstellen als Anwendung mit dauerhaftem Zustand in einer VPC ohne Internetzugang.

Scala/Python-Kriterien

- Verwenden Sie in Ihrem Scala- oder Python-Code den Blink-Planer (`senv`, `stenv` für Scala; `s_env`, `st_env` für Python) und nicht den älteren „Flink“-Planer (`stenv_2` für Scala, `st_env_2` für Python). Das Apache-Flink-Projekt empfiehlt die Verwendung des Blink-Planers für Produktionsanwendungen. Dies ist der Standardplaner in Zeppelin und Flink.
- Ihre Python-Absätze dürfen keine Shell-Aufrufe/Zuweisungen mit `!` oder magische IPython-Befehle wie `%timeit` oder `%conda` in Notizen verwenden, die als Anwendungen bereitgestellt werden sollen.
- Sie können Scala-Fallklassen nicht als Parameter von Funktionen verwenden, die an Datenflussoperatoren höherer Ordnung wie `map` und `filter` übergeben werden. Informationen zu Scala-Fallklassen finden Sie unter FALLKLASSEN in der Scala-Dokumentation.

SQL-Kriterien

- Einfache SELECT-Anweisungen sind nicht zulässig, da es kein Äquivalent zum Ausgabeabschnitt eines Absatzes gibt, in den die Daten übermittelt werden können.
- In jedem Absatz müssen DDL-Anweisungen (`USE`, `CREATE`, `ALTER`, `DROP`, `SET`, `RESET`) den DML-(`INSERT`)-Anweisungen vorangehen. Dies liegt daran, dass DML-Anweisungen in einem Absatz zusammen als ein einziger Flink-Auftrag eingereicht werden müssen.

- Es darf höchstens einen Absatz geben, der DML-Anweisungen enthält. Das liegt daran, dass wir für die deploy-as-application Funktion nur das Senden eines einzelnen Auftrags an Flink unterstützen.

Weitere Informationen und ein Beispiel finden Sie unter Übersetzen, Redigieren und Analysieren von Streaming-Daten mithilfe von SQL-Funktionen mit Amazon Managed Service für Apache Flink, Amazon Translate und Amazon Comprehend.

IAM-Berechtigungen für Studio-Notebooks

Managed Service für Apache Flink erstellt eine IAM-Rolle für Sie, wenn Sie ein Studio-Notebook über die AWS Management Console erstellen. Außerdem wird dieser Rolle eine Richtlinie zugeordnet, die den folgenden Zugriff ermöglicht:

Service	Zugriff
CloudWatch Protokolle	Auflisten
Amazon EC2	Auflisten
AWS Glue	Lesen, Schreiben
Managed Service für Apache Flink	Lesen
Managed Service für Apache Flink V2	Lesen
Amazon S3	Lesen, Schreiben

Konnektoren und Abhängigkeiten

Konnektoren ermöglichen es Ihnen, Daten über verschiedene Technologien hinweg zu lesen und zu schreiben. Managed Service für Apache Flink bündelt drei Standard-Konnektoren mit Ihrem Studio-Notebook. Sie können auch benutzerdefinierte Konnektoren verwenden. Weitere Informationen zu Konnektoren finden Sie unter Tabellen- und SQL-Konnektoren in der Apache-Flink-Dokumentation.

Standardkonnektoren

Wenn Sie die AWS Management Console verwenden, um Ihr Studio-Notebook zu erstellen, enthält Managed Service für Apache Flink standardmäßig die folgenden benutzerdefinierten Konnektoren: `flink-sql-connector-flink`, `flink-connector-kafka_2.12` und `aws-msk-iam-auth`. Um über die Konsole ein Studio-Notebook ohne diese benutzerdefinierten Konnektoren zu erstellen, wählen Sie die Option Mit benutzerdefinierten Einstellungen erstellen. Wenn Sie dann zur Seite Konfigurationen gelangen, deaktivieren Sie die Kontrollkästchen neben den beiden Konnektoren.

Wenn Sie die CreateApplication-API verwenden, um Ihr Studio-Notebook zu erstellen, sind die `flink-connector-kafka`Konnektoren `flink-sql-connector-flink` und nicht standardmäßig enthalten. Um sie hinzuzufügen, geben Sie sie als eine `MavenRefernce` im `CustomArtifactsConfiguration`-Datentyp an, wie in den folgenden Beispielen gezeigt.

Der Konnektor `aws-msk-iam-auth` ist der Konnektor, der mit Amazon MSK verwendet werden soll und das Feature zur automatischen Authentifizierung bei IAM enthält.

> **Note**
>
> Die im folgenden Beispiel gezeigten Konnektor-Versionen sind die einzigen Versionen, die wir unterstützen.

```
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"

   }
}]
```

Um diese Connectors zu einem vorhandenen Notebook hinzuzufügen, verwenden Sie die UpdateApplication -API-Operation und geben Sie sie als `MavenReference` im `CustomArtifactsConfigurationUpdate` Datentyp an.

> **Note**
>
> Sie können `failOnError` für den Konnektor `flink-sql-connector-kinesis` in der Tabellen-API auf true setzen.

Abhängigkeiten und benutzerdefinierte Konnektoren

Gehen Sie folgendermaßen vor, um Ihrem Studio-Notebook über die AWS Management Console eine Abhängigkeit oder einen benutzerdefinierten Konnektor hinzuzufügen:

1. Laden Sie die Datei Ihres benutzerdefinierten Konnektors in Amazon S3 hoch.
2. Wählen Sie in der AWS Management Console die Option Benutzerdefiniert erstellen, um Ihr Studio-Notebook zu erstellen.
3. Folgen Sie dem Workflow zur Erstellung eines Studio-Notebooks, bis Sie zum Schritt Konfigurationen gelangen.

4. Wählen Sie im Abschnitt Benutzerdefinierte Konnektoren die Option Benutzerdefinierten Konnektor hinzufügen aus.
5. Geben Sie den Amazon-S3-Speicherort der Abhängigkeit oder des benutzerdefinierten Konnektors an.
6. Wählen Sie Änderungen speichern.

Um ein Abhängigkeits-JAR oder einen benutzerdefinierten Connector hinzuzufügen, wenn Sie ein neues Studio-Notebook mithilfe der CreateApplication API erstellen, geben Sie den Amazon S3-Speicherort des Abhängigkeits-JAR oder des benutzerdefinierten Connectors im `CustomArtifactsConfiguration` Datentyp an. Um einem vorhandenen Studio-Notebook eine Abhängigkeit oder einen benutzerdefinierten Connector hinzuzufügen, rufen Sie die UpdateApplication -API-Operation auf und geben Sie den Amazon S3-Speicherort des Abhängigkeits-JAR oder des benutzerdefinierten Connectors im `CustomArtifactsConfigurationUpdate` Datentyp an.

> **Note**
>
> Wenn Sie eine Abhängigkeit oder einen benutzerdefinierten Konnektor einbeziehen, müssen Sie auch alle zugehörigen transitiven Abhängigkeiten einbeziehen, die nicht darin gebündelt sind.

Benutzerdefinierte Funktionen

Benutzerdefinierte Funktionen (User-Defined Functions, UDFs) sind Erweiterungspunkte, mit denen Sie häufig verwendete Logik oder benutzerdefinierte Logik aufrufen können, die in Abfragen nicht anders ausgedrückt werden kann. Sie können Python oder eine JVM-Sprache wie Java oder Scala verwenden, um Ihre UDFs in Absätzen in Ihrem Studio-Notebook zu implementieren. Sie können Ihrem Studio-Notebook auch externe JAR-Dateien hinzufügen, die in einer JVM-Sprache implementierte UDFs enthalten.

Verwenden Sie bei der Implementierung von JARs, die abstrakte Klassen dieser Unterklasse `UserDefinedFunction` (oder Ihre eigenen abstrakten Klassen) registrieren, den bereitgestellten Bereich in Apache Maven, `compileOnly`-Abhängigkeitsdeklarationen in Gradle, den bereitgestellten Scope in SBT oder eine entsprechende Direktive in Ihrer UDF-Projekt-Build-Konfiguration. Dadurch kann der UDF-Quellcode anhand der Flink-APIs kompiliert werden, aber die Flink-API-Klassen sind

selbst nicht in den Build-Artefakten enthalten. Beziehen Sie sich auf dieses POM aus dem UDF-JAR-Beispiel, das diese Voraussetzung für ein Maven-Projekt erfüllt.

> **Note**
>
> Weitere Informationen und ein Beispiel finden Sie unter Übersetzen, Redigieren und Analysieren von Streaming-Daten mithilfe von SQL-Funktionen mit Amazon Managed Service für Apache Flink, Amazon Translate und Amazon Comprehend im AWS Machine Learning Blog.

Gehen Sie folgendermaßen vor, um die Konsole zum Hinzufügen von UDF-JAR-Dateien zu Ihrem Studio-Notebook zu verwenden:

1. Laden Sie Ihre UDF-JAR-Datei in Amazon S3 hoch.
2. Wählen Sie in AWS Management Console die Option Benutzerdefiniert erstellen aus, um Ihr Studio-Notebook zu erstellen.
3. Folgen Sie dem Workflow zur Erstellung eines Studio-Notebooks, bis Sie zum Schritt Konfigurationen gelangen.
4. Wählen Sie im Abschnitt Benutzerdefinierte Funktionen die Option Benutzerdefinierte Funktion hinzufügen aus.
5. Geben Sie den Amazon-S3-Speicherort der JAR- oder ZIP-Datei an, in der Ihre UDF implementiert ist.
6. Wählen Sie Änderungen speichern aus.

Um ein UDF-JAR hinzuzufügen, wenn Sie ein neues Studio-Notebook mithilfe der CreateApplication API erstellen, geben Sie den JAR-Speicherort im `CustomArtifactConfiguration` Datentyp an. Um einem vorhandenen Studio-Notebook ein UDF-JAR hinzuzufügen, rufen Sie die UpdateApplication API-Operation auf und geben Sie den JAR-Speicherort im `CustomArtifactsConfigurationUpdate` Datentyp an. Alternativ können Sie die AWS Management Console zum Hinzufügen von UDF-JAR-Dateien zu Ihrem Studio-Notebook verwenden.

Überlegungen zu benutzerdefinierten Funktionen

- Managed Service für Apache Flink Studio verwendet die Apache-Zeppelin-Terminologie, wobei ein Notebook eine Zeppelin-Instance ist, die mehrere Notizen enthalten kann. Jede Notiz kann dann wiederum mehrere Absätze enthalten. Mit Managed Service für Apache Flink Studio wird

der Interpreter-Prozess von allen Notizen im Notebook gemeinsam genutzt. Wenn Sie also eine explizite Funktionsregistrierung mit createTemporarySystemFunktion in einer Notiz durchführen, kann auf diese unverändert in einer anderen Notiz desselben Notebooks verwiesen werden.

Der Vorgang Als Anwendung bereitstellen bezieht sich jedoch auf eine einzelne Notiz und nicht auf alle Notizen im Notebook. Wenn Sie Als Anwendung bereitstellen ausführen, werden nur die Inhalte der aktiven Notiz zur Generierung der Anwendung verwendet. Jede explizite Funktionsregistrierung, die in anderen Notebooks durchgeführt wird, ist nicht Teil der generierten Anwendungsabhängigkeiten. Darüber hinaus erfolgt bei der Option „Als Anwendung bereitstellen“ eine implizite Funktionsregistrierung, indem der Hauptklassenname von JAR in eine Zeichenfolge in Kleinbuchstaben umgewandelt wird.

Wenn `TextAnalyticsUDF` beispielsweise die Hauptklasse für UDF-JAR ist, führt eine implizite Registrierung zum Funktionsnamen `textanalyticsudf`. Wenn also eine explizite Funktionsregistrierung in Notiz 1 von Studio wie folgt erfolgt, dann können alle anderen Notizen in diesem Notebook (z. B. Notiz 2) aufgrund des gemeinsamen Interpreters mit dem Namen `myNewFuncNameForClass` auf die Funktion verweisen:

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

Bei der Bereitstellung als Anwendung in Notiz 2 ist diese explizite Registrierung jedoch nicht in den Abhängigkeiten enthalten, sodass die bereitgestellte Anwendung nicht wie erwartet funktioniert. Aufgrund der impliziten Registrierung wird standardmäßig erwartet, dass alle Verweise auf diese Funktion mit `textanalyticsudf` und nicht `myNewFuncNameForClass` erfolgen.

Falls eine Registrierung von benutzerdefinierten Funktionsnamen erforderlich ist, wird davon ausgegangen, dass Notiz 2 selbst einen weiteren Absatz enthält, in dem eine weitere explizite Registrierung wie folgt durchgeführt wird:

```
%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
;
```

- Wenn Ihr UDF-JAR Flink-SDKs enthält, konfigurieren Sie Ihr Java-Projekt so, dass der UDF-Quellcode mit den Flink-SDKs kompiliert werden kann, die Flink-SDK-Klassen selbst jedoch nicht im Build-Artefakt enthalten sind, z. B. in der JAR.

  Sie können den `provided`-Scope in Apache Maven, `compileOnly`-Abhängigkeitsdeklarationen in Gradle, `provided`-Scope in SBT oder eine gleichwertige Direktive in der Build-Konfiguration ihres UDF-Projekts verwenden. Beziehen Sie sich auf dieses POM aus dem UDF-JAR-Beispiel, das diese Voraussetzung für ein Maven-Projekt erfüllt. Ein vollständiges step-by-step Tutorial finden Sie in diesem Tutorial: Streaming-Daten mithilfe von SQL-Funktionen mit Amazon Managed Service für Apache Flink, Amazon Translate und Amazon Comprehend umwandeln, redigieren und analysieren.

Aktivieren von Checkpointing

Sie aktivieren Checkpointing mithilfe der Umgebungseinstellungen. Informationen zum Checkpointing finden Sie unter Fehlertoleranz im Managed Service für Apache Flink Entwicklerhandbuch.

Einstellen des Checkpointing-Intervalls

Im folgenden Scala-Codebeispiel wird das Checkpointing-Intervall Ihrer Anwendung auf eine Minute festgelegt:

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

Im folgenden Python-Codebeispiel wird das Checkpointing-Intervall Ihrer Anwendung auf eine Minute festgelegt:

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

Einstellen des Checkpointing-Typs

Das folgende Scala-Codebeispiel setzt den Checkpoint-Modus Ihrer Anwendung auf `EXACTLY_ONCE` (Standard):

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

Das folgende Python-Codebeispiel setzt den Checkpoint-Modus Ihrer Anwendung auf `EXACTLY_ONCE` (Standard):

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

Arbeiten mit AWS Glue

Ihr Studio-Notebook speichert und ruft Informationen über seine Datenquellen und Senken aus AWS Glue ab. Wenn Sie Ihr Studio-Notebook erstellen, geben Sie die AWS Glue-Datenbank an, die Ihre Verbindungsinformationen enthält. Wenn Sie auf Ihre Datenquellen und Senken zugreifen, geben Sie die in der Datenbank enthaltenen AWS Glue-Tabellen an. Ihre AWS Glue-Tabellen bieten Zugriff auf die AWS Glue-Verbindungen, die die Speicherorte, Schemas und Parameter Ihrer Datenquellen und Ziele definieren.

Studio-Notebooks verwenden Tabelleneigenschaften, um anwendungsspezifische Daten zu speichern. Weitere Informationen finden Sie unter Tabelleneigenschaften.

Ein Beispiel für die Einrichtung einer AWS Glue-Verbindung, -Datenbank und -Tabelle für die Verwendung mit Studio-Notebooks finden Sie unter Erstellen einer AWS Glue-Datenbank im Tutorial zum Erstellen eines Studio-Notebooks-Tutorial.

Tabelleneigenschaften

Zusätzlich zu den Datenfeldern stellen Ihre AWS Glue-Tabellen mithilfe von Tabelleneigenschaften weitere Informationen für Ihr Studio-Notebook bereit. Managed Service für Apache Flink verwendet die folgenden AWS Glue-Tabelleneigenschaften:

- Verwenden von Apache-Flink-Zeitwerten: Diese Eigenschaften definieren, wie Managed Service für Apache Flink interne Datenverarbeitungszeitwerte von Apache Flink ausgibt.

- Verwenden von Flink-Konnektor und Formateigenschaften: Diese Eigenschaften liefern Informationen über Ihre Datenströme.

So fügen Sie einer AWS Glue-Tabelle eine Eigenschaft hinzu:

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS Glue-Konsole unter https://console.aws.amazon.com/glue/.
2. Wählen Sie aus der Tabellenliste die Tabelle aus, die Ihre Anwendung zum Speichern von Datenverbindungsinformationen verwendet. Wählen Sie Aktion, Tabellendetails bearbeiten aus.
3. Geben Sie unter Tabelleneigenschaften den Wert **`managed-flink.proctime`** für Schlüssel und **`user_action_time`** für Wert ein.

Verwenden von Apache-Flink-Zeitwerten

Apache Flink stellt Zeitwerte bereit, die beschreiben, wann Ereignisse bei der Stream-Verarbeitung aufgetreten sind, z. B. Verarbeitungszeit und Ereigniszeit. Um diese Werte in Ihre Anwendungsausgabe aufzunehmen, definieren Sie Eigenschaften in Ihrer AWS Glue-Tabelle, die die Laufzeit von Managed Service für Apache Flink anweisen, diese Werte in die angegebenen Felder auszugeben.

Die Schlüssel und Werte, die Sie in Ihren Tabelleneigenschaften verwenden, lauten wie folgt:

Zeitstempeltyp	Schlüssel	Wert
Verarbeitungszeit	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.
Ereigniszeit	managed-flink.rowtime	The column name that AWS Glue will use to expose the value. This column name corresponds to an existing table column.

Zeitstempeltyp	Schlüssel	Wert
	managed-flink.watermark.*column_name*.milliseconds	The watermark interval in milliseconds

Verwenden von Flink-Konnektor und Formateigenschaften

Mithilfe von AWS Glue-Tabelleneigenschaften stellen Sie den Flink-Konnektoren Ihrer Anwendung Informationen über Ihre Datenquellen zur Verfügung. Im Folgenden einige Beispiele für die Eigenschaften, die Managed Service für Apache Flink für Konnektoren verwendet:

Konnektortyp	Schlüssel	Wert
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. `earliest-offset` or `Zeitstempel`.
Kinesis	`Format`	The format used to deserialize and serialize Kinesis data stream records, e.g. `json` or `csv`.
	`aws.region`	The AWS region where the stream is defined.
S3 (Dateisystem)	format	The format used to deserialize and serialize files, e.g. `json` or `csv`.
	`Pfad`	The Amazon S3 path, e.g. `s3://mybucket/`.

Weitere Informationen zu anderen Konnektoren neben Kinesis und Apache Kafka finden Sie in der Dokumentation Ihres Konnektors.

Beispiele und Tutorials

Themen

- Tutorial: Erstellen eines Studio-Notebooks in Managed Service für Apache Flink
- Tutorial: Bereitstellung als Anwendung mit dauerhaftem Zustand
- Beispiele

Tutorial: Erstellen eines Studio-Notebooks in Managed Service für Apache Flink

Das folgende Tutorial zeigt, wie Sie ein Studio-Notebook erstellen, das Daten aus einem Kinesis Data Stream oder einem Amazon MSK-Cluster liest.

Dieses Tutorial enthält die folgenden Abschnitte:

- Setup
- Erstellen einer AWS Glue-Datenbank
- Nächste Schritte
- Erstellen eines Studio-Notebooks mit Kinesis Data Streams
- Erstellen eines Studio-Notebooks mit Amazon MSK
- Ihre Anwendung und die abhängigen Ressourcen bereinigen

Setup

Stellen Sie sicher, dass Die AWS CLI Version 2 oder höher ist. Informationen zur Installation der neuesten AWS CLI-Version finden Sie unter Installieren, Aktualisieren und Deinstallieren der AWS CLI Version 2.

Erstellen einer AWS Glue-Datenbank

Ihr Studio-Notebook verwendet eine AWS Glue-Datenbank für Metadaten zu Ihrer Amazon MSK-Datenquelle.

Erstellen einer AWS Glue-Datenbank

1. Öffnen Sie die AWS Glue-Konsole unter https://console.aws.amazon.com/glue/.
2. Wählen Sie Add database (Datenbank hinzufügen). Geben Sie im Fenster Datenbank hinzufügen **`default`** als Namen der Datenbank ein. Wählen Sie Erstellen aus.

Nächste Schritte

Mit diesem Tutorial können Sie ein Studio-Notebook erstellen, das entweder Kinesis Data Streams oder Amazon MSK verwendet:

- Kinesis Data Streams: Mit Kinesis Data Streams können Sie schnell eine Anwendung erstellen, die einen Kinesis Data Stream als Quelle verwendet. Sie müssen nur einen Kinesis Data Stream als abhängige Ressource erstellen.
- Amazon MSK: Mit Amazon MSK erstellen Sie eine Anwendung, die einen Amazon MSK-Cluster als Quelle verwendet. Sie müssen eine Amazon VPC, eine Amazon EC2-Client-Instance und einen Amazon MSK-Cluster als abhängige Ressourcen erstellen.

Erstellen eines Studio-Notebooks mit Kinesis Data Streams

In diesem Tutorial wird beschrieben, wie Sie ein Studio-Notebook erstellen, das einen Kinesis Data Stream als Quelle verwendet.

Dieses Tutorial enthält die folgenden Abschnitte:

- Aufstellen
- Erstellen einer AWS GlueTabelle
- Erstellen Sie ein Studio-Notebook mit Kinesis Data Streams
- Senden von Daten an den Kinesis Data Stream
- Testen Sie Ihr Studio-Notebook

Aufstellen

Bevor Sie ein Studio-Notebook erstellen, erstellen Sie einen Kinesis Data Stream (`ExampleInputStream`). Ihre Anwendung verwendet diesen Stream als Anwendungsquelle.

Sie können diesen Stream mithilfe der Amazon Kinesis-Konsole oder des folgenden AWS CLI-Befehls erstellen. Anweisungen für die Konsole finden Sie unter Erstellen und Aktualisieren von

Datenströmen im Amazon Kinesis Data Streams Entwicklerhandbuch. Benennen Sie den Stream **`ExampleInputStream`** und legen Sie die Anzahl der offenen Shards auf **`1`** fest.

Verwenden Sie den folgenden Amazon Kinesis `create-stream` AWS CLI-Befehl mit der AWS CLI, um den Stream (`ExampleInputStream`) zu erstellen.

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

Erstellen einer AWS GlueTabelle

Ihr Studio-Notebook verwendet eine AWS Glue-Datenbank für Metadaten zu Ihrer Kinesis Data Streams-Datenquelle.

> **Note**
>
> Sie können die Datenbank entweder zuerst manuell erstellen oder sie beim Erstellen des Notebooks von Managed Service für Apache Flink für Sie erstellen lassen. Ebenso können Sie die Tabelle entweder manuell erstellen, wie im folgenden Abschnitt beschrieben, oder Sie können den Konnektorcode zum Erstellen einer Tabelle für Managed Service für Apache Flink in Ihrem Notebook innerhalb von Apache Zeppelin verwenden, um Ihre Tabelle über eine DDL-Anweisung zu erstellen. Sie können dann in AWS Glue überprüfen, um sicherzustellen, dass die Tabelle korrekt erstellt wurde.

Erstellen einer Tabelle

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS Glue-Konsole unter https://console.aws.amazon.com/glue/.
2. Wenn Sie noch keine AWS Glue-Datenbank haben, wählen Sie in der linken Navigationsleiste Datenbanken aus. Wählen Sie Datenbank hinzufügen. Geben Sie im Fenster Datenbank hinzufügen **`default`** als Namen der Datenbank ein. Wählen Sie Erstellen.
3. Wählen Sie in der linken Navigationsleiste die Option Tabellen. Wählen Sie auf der Seite Tabellen die Optionen Tabellen hinzufügen, Tabelle manuell hinzufügen aus.

4. Geben Sie auf der Seite Eigenschaften Ihrer Tabelle einrichten **`stock`** als Tabellennamen ein. Stellen Sie sicher, dass Sie die Datenbank auswählen, die Sie zuvor erstellt haben. Wählen Sie Weiter aus.
5. Wählen Sie auf der Seite Datenstore hinzufügen die Option Kinesis aus. Geben Sie als Streamnamen **`ExampleInputStream`** ein. Wählen Sie für Kinesis-Quell-URL die Eingabetaste **`https://kinesis.us-east-1.amazonaws.com`**. Wenn Sie die Kinesis-Quell-URL kopieren und einfügen, achten Sie darauf, alle führenden oder nachfolgenden Leerzeichen zu löschen. Wählen Sie Weiter aus.
6. Wählen Sie auf der Seite Klassifikation die Option JSON aus. Wählen Sie Weiter aus.
7. Wählen Sie auf der Seite Schema definieren die Option „Spalte hinzufügen“, um eine Spalte hinzuzufügen. Fügen Sie Spalten mit den folgenden Eigenschaften hinzu:

Spaltenname	Datentyp
`Ticker`	**`Zeichenfolge`**
`price`	**`double`**

   Wählen Sie Weiter aus.
8. Überprüfen Sie auf der nächsten Seite Ihre Einstellungen und wählen Sie Fertigstellen.
9. Wählen Sie die neu erstellte Tabelle aus der Liste der Tabellen aus.
10. Wählen Sie Tabelle bearbeiten und fügen Sie eine Eigenschaft mit dem Schlüssel `managed-flink.proctime` und dem Wert `proctime` hinzu.
11. Wählen Sie Apply (Anwenden) aus.

Erstellen Sie ein Studio-Notebook mit Kinesis Data Streams

Nachdem Sie die Ressourcen erstellt haben, die Ihre Anwendung verwendet, erstellen Sie Ihr Studio-Notebook.

Sie können Ihre Anwendung entweder mit der AWS Management Console oder mit der AWS CLI erstellen.

- Erstellen Sie ein Studio-Notebook mit der AWS Management Console
- Erstellen Sie ein Studio-Notebook mit der AWS CLI

Erstellen Sie ein Studio-Notebook mit der AWS Management Console

1. Öffnen Sie die Konsole Managed Service für Apache Flink unter https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard.
2. Wählen Sie auf der Seite Managed Service für Apache Flink-Anwendungen die Registerkarte Studio aus. Wählen Sie Studio-Notebook erstellen.

   > **Note**
   >
   > Sie können ein Studio-Notebook auch über die Amazon MSK- oder Kinesis Data Streams-Konsolen erstellen, indem Sie Ihren Amazon MSK-Eingabe-Cluster oder Kinesis Data Stream auswählen und dann Daten in Echtzeit verarbeiten auswählen.

3. Geben Sie auf der Seite Notebook-Instance erstellen folgende Informationen ein:

   - Geben Sie **`MyNotebook`** als Namen des Notebooks ein.
   - Wählen Sie Standard für die AWS-Glue-Datenbank.

   Wählen Sie Studio-Notebook erstellen.
4. Wählen Sie auf der MyNotebook Seite Ausführen aus. Warten Sie, bis der Status Wird ausgeführt angezeigt wird. Es fallen Gebühren an, wenn das Notebook läuft.

Erstellen Sie ein Studio-Notebook mit der AWS CLI

Gehen Sie wie folgt vor, um Ihr Studio-Notebook mit der AWS CLI zu erstellen:

1. Überprüfen Sie die Konto-ID. Sie benötigen diesen Wert, um die Anwendung zu erstellen.
2. Erstellen Sie die Rolle `arn:aws:iam::AccountID:role/ZeppelinRole` und fügen Sie der automatisch erstellten Rolle über die Konsole die folgenden Berechtigungen hinzu.

   ```
   "kinesis:GetShardIterator",

   "kinesis:GetRecords",

   "kinesis:ListShards"
   ```
3. Erstellen Sie eine Datei mit dem Namen `create.json` und den folgenden Inhalten. Ersetzen Sie die Platzhalterwerte durch Ihre Informationen.

```
{
    "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. Um Ihre Anwendung zu erstellen, führen Sie den folgenden Befehl aus.

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

5. Wenn der Befehl abgeschlossen ist, sehen Sie eine Ausgabe, die die Details für Ihr neues Studio-Notebook enthält. Es folgt ein Beispiel für die Ausgabe.

```
{
    "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. Um Ihre Anwendung zu starten, führen Sie den folgenden Befehl aus. Ersetzen Sie die Beispielwerte durch Ihre Konto-ID.

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

Senden von Daten an den Kinesis Data Stream

Gehen Sie wie folgt vor, um Testdaten an Ihren Kinesis Data Stream zu senden:

1. Öffnen Sie den Kinesis Data Generator.
2. Wählen Sie Cognito-Benutzer mit erstellen aus CloudFormation.
3. Die AWS CloudFormation-Konsole wird mit der Kinesis Data Generator-Vorlage geöffnet. Wählen Sie Weiter aus.
4. Auf der Seite Festlegen von Komponentendetails geben Sie den Benutzernamen und das Passwort für Ihren Cognito-Benutzer ein. Wählen Sie Weiter aus.
5. Wählen Sie auf der Seite Stack-Optionen konfigurieren Weiter aus.
6. Wählen Sie auf der Seite Kinesis-Data-Generator-Cognito-User überprüfen das Kontrollkästchen Ich bestätige, dass AWS CloudFormation möglicherweise IAM-Ressourcen erstellt. Wählen Sie Stapel erstellen aus.
7. Warten Sie, bis der AWS CloudFormation-Stapel erstellt wurde. Wenn der Stapel abgeschlossen ist, öffnen Sie den Kinesis-Data-Generator-Cognito-User-Stapel in der AWS CloudFormation-Konsole und wählen Sie die Registerkarte Outputs. Öffnen Sie die URL, die für den KinesisDataGeneratorUrl Ausgabewert aufgeführt ist.
8. Melden Sie sich auf der Amazon Kinesis Data Generator-Seite mit den Anmeldeinformationen an, die Sie in Schritt 4 erstellt haben.
9. Geben Sie auf der nächsten Seite die folgenden Werte an:

Region	**us-east-1**
Stream/Kinesis Data Firehose stream	**ExampleInputStream**
Datensätze pro Sekunde	**1**

Fügen Sie für Datensatzvorlage den folgenden Code ein:

```
{
```

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

10. Wählen Sie Daten senden aus.
11. Der Generator sendet Daten an den Kinesis Data Stream.

    Lassen Sie den Generator laufen, während Sie den nächsten Abschnitt abschließen.

Testen Sie Ihr Studio-Notebook

In diesem Abschnitt verwenden Sie Ihr Studio-Notebook, um Daten aus Ihrem Kinesis Data Stream abzufragen.

1. Öffnen Sie die Konsole Managed Service für Apache Flink unter https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard.
2. Wählen Sie auf der Seite Managed Service für Apache Flink-Anwendungen die Registerkarte Studio-Notebook aus. Wählen Sie MyNotebook.
3. Wählen Sie auf der MyNotebook Seite In Apache Zeppelin öffnen aus.

   Die Oberfläche von Apache Zeppelin wird in einer neuen Registerkarte geöffnet.
4. Auf der Seite Willkommen bei Zeppelin! wählen Sie Zeppelin Notiz aus.
5. Geben Sie auf der Seite Zeppelin Notiz die folgende Abfrage in eine neue Notiz ein:

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

   Wählen Sie das Ausführungssymbol.

   Nach kurzer Zeit werden in der Notiz Daten aus dem Kinesis Data Stream angezeigt.

Um das Apache Flink-Dashboard für Ihre Anwendung zu öffnen und betriebliche Aspekte zu sehen, wählen Sie FLINK JOB. Weitere Informationen zum Flink-Dashboard finden Sie unter Apache Flink-Dashboard im Managed Service für Apache Flink Entwicklerhandbuch.

Weitere Beispiele für Flink-Streaming-SQL-Abfragen finden Sie unter Abfragen in der Apache Flink-Dokumentation.

Erstellen eines Studio-Notebooks mit Amazon MSK

In diesem Tutorial wird beschrieben, wie Sie ein Studio-Notebook erstellen, das einen Amazon-MSK-Cluster als Quelle verwendet.

Dieses Tutorial enthält die folgenden Abschnitte:

- Setup
- Fügen Sie Ihrer VPC ein NAT-Gateway hinzu
- Erstellen Sie eine AWS Glue-Verbindung und eine Tabelle
- Erstellen Sie ein Studio-Notebook mit Amazon MSK
- Senden Sie Daten an Ihren Amazon MSK-Cluster
- Testen Sie Ihr Studio-Notebook

Setup

Für dieses Tutorial benötigen Sie einen Amazon MSK-Cluster, der Klartextzugriff ermöglicht. Wenn Sie noch keinen Amazon MSK-Cluster eingerichtet haben, folgen Sie dem Tutorial Erste Schritte mit Amazon MSK, um eine Amazon VPC, einen Amazon MSK-Cluster, ein Thema und eine Amazon EC2-Client-Instance zu erstellen.

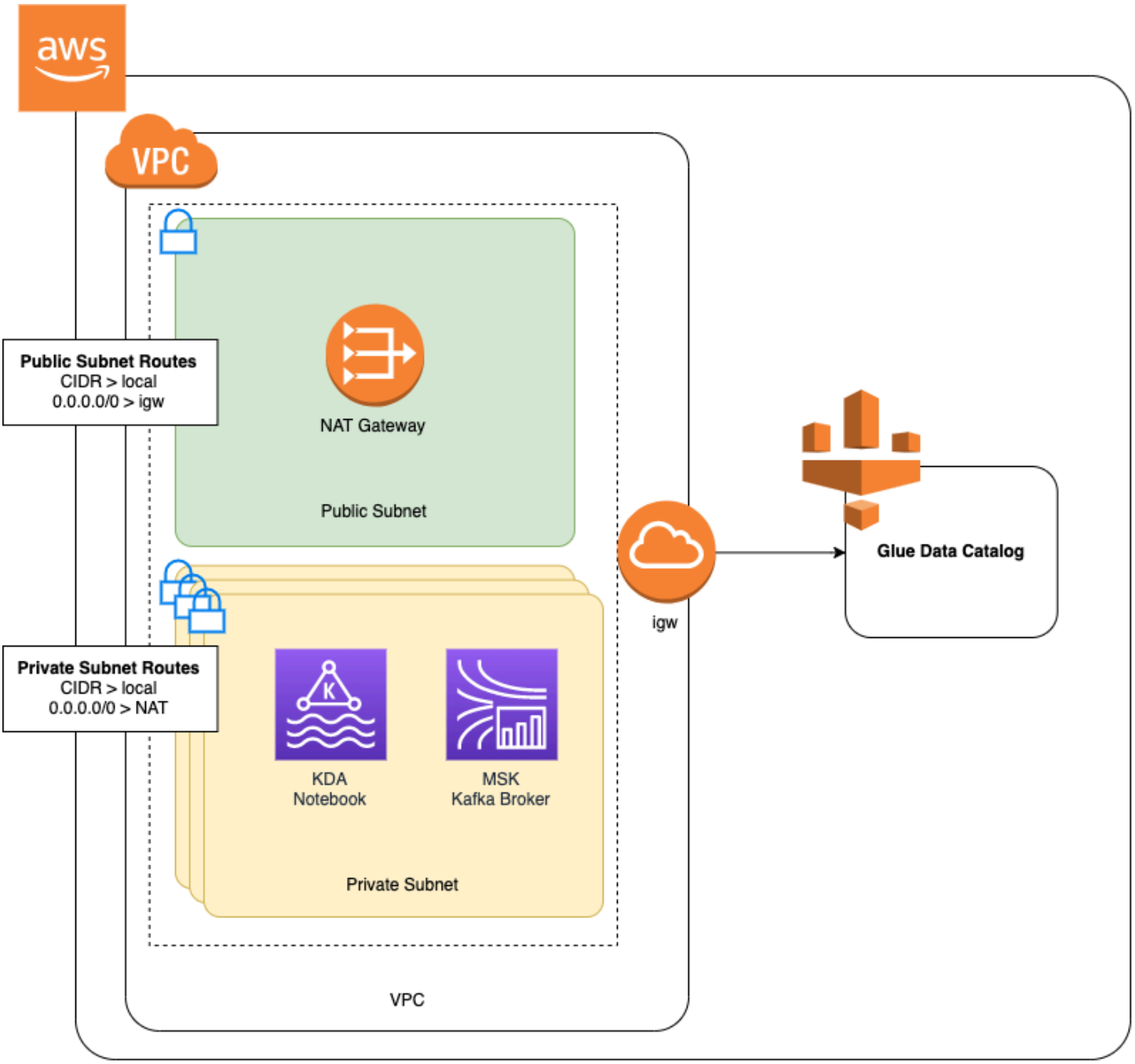
Gehen Sie beim Befolgen des Tutorials wie folgt vor:

- Ändern Sie in Schritt 3: Amazon MSK-Cluster erstellen bei Schritt 4 den `ClientBroker`-Wert von `TLS` auf **`PLAINTEXT`**.

Fügen Sie Ihrer VPC ein NAT-Gateway hinzu

Wenn Sie einen Amazon MSK-Cluster erstellt haben, indem Sie dem Tutorial Erste Schritte mit Amazon MSK gefolgt sind, oder wenn Ihre bestehende Amazon VPC noch kein NAT-Gateway für ihre

privaten Subnetze hat, müssen Sie Ihrer Amazon VPC ein NAT-Gateway hinzufügen. Das folgende Diagramm zeigt die Architektur.



Gehen Sie wie folgt vor, um ein NAT-Gateway für Ihre Amazon VPC zu erstellen:

1. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc/.
2. Wählen Sie in der linken Navigationsleiste NAT-Gateway aus.
3. Wählen Sie auf der Seite NAT-Gateways die Option NAT-Gateway erstellen aus.

4. Geben Sie auf der Seite NAT-Gateway erstellen die folgenden Werte an:

Name – optional	**`ZeppelinGateway`**
Subnetz	AWSKafkaTutorialSubnet1
Elastic-IP-Zuweisungs-ID	Choose an available Elastic IP. If there are no Elastic IPs available, choose Elastic-IP zuweisen, and then choose the Elasic IP that the console creates.

   Wählen Sie NAT-Gateway erstellen aus.
5. Wählen Sie in der linken Navigationsleiste Routing-Tabellen aus.
6. Klicken Sie auf Create Route Table (Routing-Tabelle erstellen).
7. Geben Sie auf der Seite Routing-Tabelle erstellen folgende Informationen ein:

   - Name-Tag: **`ZeppelinRouteTable`**
   - VPC: Wählen Sie Ihre VPC (z. B. AWSKafkaTutorialVPC).

   Wählen Sie Erstellen aus.
8. Wählen Sie in der Liste der Routing-Tabellen ZeppelinRouteTable aus. Klicken Sie auf der Registerkarte Routen auf Routen bearbeiten.
9. Wählen Sie auf der Seite Routen bearbeiten die Option Route hinzufügen aus.
10. Geben Sie im Für-Ziel **`0.0.0.0/0`** ein. Wählen Sie als Ziel NAT-Gateway, ZeppelinGateway. Wählen Sie Routen speichern aus. Klicken Sie auf Close (Schließen).
11. Wählen Sie auf der Seite Routing-Tabellen, wenn ZeppelinRouteTable ausgewählt ist, die Registerkarte Subnetzzuordnungen aus. Wählen Sie Subnetzzuordnungen bearbeiten aus.
12. Wählen Sie auf der Seite Subnetz-Zuordnungen bearbeiten die Optionen AWSKafkaTutorialSubnet2 und AWSKafkaTutorialSubnet3 aus. Wählen Sie Save (Speichern).

Erstellen Sie eine AWS Glue-Verbindung und eine Tabelle

Ihr Studio-Notebook verwendet eine AWS Glue-Datenbank für Metadaten zu Ihrer Amazon MSK-Datenquelle. In diesem Abschnitt erstellen Sie eine AWS Glue-Verbindung, die beschreibt, wie Sie

auf Ihren Amazon MSK-Cluster zugreifen, und eine AWS Glue-Tabelle, in der beschrieben wird, wie Sie die Daten in Ihrer Datenquelle für Clients wie Ihr Studio-Notebook präsentieren.

Eine Verbindung erstellen

1. Melden Sie sich bei der AWS Management Console an und öffnen Sie die AWS Glue-Konsole unter https://console.aws.amazon.com/glue/.
2. Wenn Sie noch keine AWS Glue-Datenbank haben, wählen Sie in der linken Navigationsleiste Datenbanken aus. Wählen Sie Datenbank hinzufügen. Geben Sie im Fenster Datenbank hinzufügen **`default`** als Namen der Datenbank ein. Wählen Sie Erstellen aus.
3. Wählen Sie in der linken Navigationsleiste Verbindungen aus. Wählen Sie Verbindung hinzufügen aus.
4. Geben Sie im Fenster Verbindung hinzufügen die folgenden Werte ein:

   - Geben Sie für Verbindungsname **`ZeppelinConnection`** ein.
   - Wählen Sie für Verbindungstyp den Eintrag Kafka.
   - Geben Sie für Kafka-Bootstrap-Server-URLs die Bootstrap-Broker-Zeichenfolge für Ihren Cluster an. Sie können die Bootstrap-Broker entweder über die MSK-Konsole oder durch Eingabe des folgenden CLI-Befehls abrufen:

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

   - Deaktivieren Sie das Kontrollkästchen SSL-Verbindung erforderlich.

   Wählen Sie Weiter aus.
5. Geben Sie auf der VPC-Seite die folgenden Werte an:

   - Wählen Sie für VPC den Namen Ihrer VPC (z. B. AWSKafkaTutorialVPC).
   - Wählen Sie für Subnetz die Option AWSKafkaTutorialSubnet2 aus.
   - Wählen Sie für Sicherheitsgruppen alle verfügbaren Gruppen aus.

   Wählen Sie Weiter aus.
6. Wählen Sie auf der Seite Verbindungseigenschaften / Verbindungszugriff die Option Fertigstellen aus.

Erstellen einer Tabelle

> **Note**
>
> Sie können die Tabelle entweder manuell erstellen, wie in den folgenden Schritten beschrieben, oder Sie können den Konnektorcode zum Erstellen einer Tabelle für Managed Service für Apache Flink in Ihrem Notebook innerhalb von Apache Zeppelin verwenden, um Ihre Tabelle über eine DDL-Anweisung zu erstellen. Sie können dann in AWS Glue überprüfen, um sicherzustellen, dass die Tabelle korrekt erstellt wurde.

1. Wählen Sie in der linken Navigationsleiste die Option Tabellen. Wählen Sie auf der Seite Tabellen die Optionen Tabellen hinzufügen, Tabelle manuell hinzufügen aus.
2. Geben Sie auf der Seite Eigenschaften Ihrer Tabelle einrichten **`stock`** als Tabellennamen ein. Stellen Sie sicher, dass Sie die Datenbank auswählen, die Sie zuvor erstellt haben. Wählen Sie Weiter aus.
3. Wählen Sie auf der Seite Datenspeicher hinzufügen die Option Kafka aus. Geben Sie als Themennamen Ihren Themennamen ein (z. B. AWSKafkaTutorialTopic). Wählen Sie für Verbindung die Option ZeppelinConnection aus.
4. Wählen Sie auf der Seite Klassifikation die Option JSON aus. Wählen Sie Weiter aus.
5. Wählen Sie auf der Seite Schema definieren die Option „Spalte hinzufügen“, um eine Spalte hinzuzufügen. Fügen Sie Spalten mit den folgenden Eigenschaften hinzu:

Spaltenname	Datentyp
`Ticker`	**`Zeichenfolge`**
`price`	**`double`**

   Wählen Sie Weiter aus.
6. Überprüfen Sie auf der nächsten Seite Ihre Einstellungen und wählen Sie Fertigstellen.
7. Wählen Sie Ihre neu erstellte Tabelle aus der Liste der Tabellen aus.
8. Wählen Sie Tabelle bearbeiten und fügen Sie eine Eigenschaft mit dem Schlüssel `managed-flink.proctime` und dem Wert `proctime` hinzu.
9. Wählen Sie Anwenden aus.

Erstellen Sie ein Studio-Notebook mit Amazon MSK

Nachdem Sie die Ressourcen erstellt haben, die Ihre Anwendung verwendet, erstellen Sie Ihr Studio-Notebook.

Sie können Ihre Anwendung entweder mit der AWS Management Console oder mit der AWS CLI erstellen.

- Erstellen Sie ein Studio-Notebook mit der AWS Management Console
- Erstellen Sie ein Studio-Notebook mit der AWS CLI

> **Note**
>
> Sie können ein Studio-Notebook auch von der Amazon MSK-Konsole aus erstellen, indem Sie einen vorhandenen Cluster auswählen und dann Daten in Echtzeit verarbeiten wählen.

Erstellen Sie ein Studio-Notebook mit der AWS Management Console

1. Öffnen Sie die Konsole Managed Service für Apache Flink unter https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard.
2. Wählen Sie auf der Seite Managed Service für Apache Flink-Anwendungen die Registerkarte Studio aus. Wählen Sie Studio-Notebook erstellen.

   > **Note**
   >
   > Um ein Studio-Notebook über die Amazon MSK- oder Kinesis Data Streams-Konsolen zu erstellen, wählen Sie Ihren Amazon MSK-Eingabe-Cluster oder Kinesis Data Stream aus und wählen Sie dann Daten in Echtzeit verarbeiten aus.

3. Geben Sie auf der Seite Notebook-Instance erstellen folgende Informationen ein:

   - Geben Sie **`MyNotebook`** als Studio-Notebookname.
   - Wählen Sie Standard für die AWS-Glue-Datenbank.

   Wählen Sie Studio-Notebook erstellen.
4. Wählen Sie auf der Seite MyNotebook die Registerkarte Konfiguration aus. Wählen Sie im Abschnitt Netzwerk die Option Bearbeiten.

5. Wählen Sie auf der Seite Netzwerk für MyNotebook bearbeiten die VPC-Konfiguration basierend auf dem Amazon MSK-Cluster aus. Wählen Sie Ihren Amazon MSK-Cluster für den Amazon MSK-Cluster aus. Wählen Sie Änderungen speichern.
6. Wählen Sie auf der Seite MyNotebook die Option Ausführen aus. Warten Sie, bis der Status Wird ausgeführt angezeigt wird.

Erstellen Sie ein Studio-Notebook mit der AWS CLI

Gehen Sie wie folgt vor, um Ihr Studio-Notebook mit der AWS CLI zu erstellen:

1. Stellen Sie sicher, dass Sie über die folgenden Informationen verfügen: Sie benötigen diese Werte, um Ihre Anwendung zu erstellen.
   - Ihre Konto-ID.
   - Die Subnetz-IDs und Sicherheitsgruppen-ID für die Amazon-VPC, die Ihren Amazon-MSK-Cluster enthält.
2. Erstellen Sie eine Datei mit dem Namen `create.json` und den folgenden Inhalten. Ersetzen Sie die Platzhalterwerte durch Ihre Informationen.

```
{
    "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. Um Ihre Anwendung zu erstellen, führen Sie den folgenden Befehl aus.

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

4. Wenn der Befehl abgeschlossen ist, sollte eine Ausgabe wie die folgende angezeigt werden, die die Details für Ihr neues Studio-Notebook enthält:

```
{
    "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. Um Ihre Anwendung zu starten, führen Sie den folgenden Befehl aus. Ersetzen Sie die Beispielwerte durch Ihre Konto-ID.

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

Senden Sie Daten an Ihren Amazon MSK-Cluster

In diesem Abschnitt führen Sie ein Python-Skript in Ihrem Amazon EC2-Client aus, um Daten an Ihre Amazon MSK-Datenquelle zu senden.

1. Stellen Sie eine Verbindung zu Ihrem Amazon EC2-Client her.

2. Führen Sie die folgenden Befehle aus, um Python Version 3, Pip und das Kafka für Python-Paket zu installieren, und bestätigen Sie die Aktionen:

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

3. Konfigurieren Sie die AWS CLI auf Ihrem Client-Computer, indem Sie den folgenden Befehl eingeben:

```
aws configure
```

   Geben Sie Ihre Kontoanmeldeinformationen ein, und **`us-east-1`** für die `region`.

4. Erstellen Sie eine Datei mit dem Namen `stock.py` und den folgenden Inhalten. Ersetzen Sie den Beispielwert durch die Bootstrap Brokers-Zeichenfolge Ihres Amazon MSK-Clusters und aktualisieren Sie den Themennamen, wenn Ihr Thema nicht AWSKafkaTutorialTopic ist:

```
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. Führen Sie das Skript mit dem folgenden Befehl aus:

```
$ python3 stock.py
```

6. Lassen Sie das Skript laufen, während Sie den folgenden Abschnitt abschließen.

Testen Sie Ihr Studio-Notebook

In diesem Abschnitt verwenden Sie Ihr Studio-Notebook, um Daten aus Ihrem Amazon MSK-Cluster abzufragen.

1. Öffnen Sie die Konsole Managed Service für Apache Flink unter https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard.
2. Wählen Sie auf der Seite Managed Service für Apache Flink-Anwendungen die Registerkarte Studio-Notebook aus. Wählen Sie MyNotebook.
3. Wählen Sie auf der Seite MyNotebook die Option In Apache Zeppelin öffnen aus.

   Die Oberfläche von Apache Zeppelin wird in einer neuen Registerkarte geöffnet.
4. Auf der Seite Willkommen bei Zeppelin! wählen Sie Zeppelin neue Notiz aus.
5. Geben Sie auf der Seite Zeppelin Notiz die folgende Abfrage in eine neue Notiz ein:

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

   Wählen Sie das Ausführungssymbol.

Die Anwendung zeigt Daten aus dem Amazon MSK-Cluster an.

Um das Apache Flink-Dashboard für Ihre Anwendung zu öffnen und betriebliche Aspekte zu sehen, wählen Sie FLINK JOB. Weitere Informationen zum Flink-Dashboard finden Sie unter Apache Flink-Dashboard im Managed Service für Apache Flink Entwicklerhandbuch.

Weitere Beispiele für Flink Streaming SQL-Abfragen finden Sie unter Abfragen in der Apache Flink-Dokumentation.

Ihre Anwendung und die abhängigen Ressourcen bereinigen

Löschen Sie Ihr Studio-Notebook

1. Öffnen Sie die Managed Service für Apache Flink-Konsole.
2. Wählen Sie MyNotebook.
3. Wählen Sie Aktionen und dann Löschen aus.

Löschen Sie Ihre AWS Glue-Datenbank und Verbindung

1. Öffnen Sie die AWS Glue-Konsole unter https://console.aws.amazon.com/glue/.
2. Wählen Sie in der linken Navigationsleiste Datenbanken aus. Markieren Sie das Kontrollkästchen neben Standard, um es auszuwählen. Wählen Sie Aktion, Datenbank löschen. Bestätigen Sie Ihre Auswahl.
3. Wählen Sie in der linken Navigationsleiste Verbindungen aus. Markieren Sie das Kontrollkästchen neben ZeppelinConnection, um es auszuwählen. Wählen Sie Aktion, Verbindung löschen. Bestätigen Sie Ihre Auswahl.

So löschen Sie die IAM-Richtlinie und -Richtlinie

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Klicken Sie in der linken Navigationsleiste auf Rollen.
3. Verwenden Sie die Suchleiste, um nach der ZeppelinRole-Rolle zu suchen.
4. Wählen Sie die ZeppelinRole-Rolle aus. Wählen Sie Rolle löschen aus. Bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch-Protokollgruppe

Die Konsole erstellt eine CloudWatch-Protokollgruppe und einen Protokollstream für Sie, wenn Sie Ihre Anwendung mit der Konsole erstellen. Sie haben keine Protokollgruppe und keinen Stream, wenn Sie Ihre Anwendung mit der AWS CLI erstellt haben.

1. Öffnen Sie die CloudWatch-Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der linken Navigationsleiste Protokollgruppen aus.
3. Wählen Sie die Protokollgruppe /AWS/KinesisAnalytics/MyNotebook.
4. Wählen Sie Actions (Aktionen), Delete log group(s) (Protokollgruppe(n) löschen) aus. Bestätigen Sie das Löschen.

Kinesis Data Streams-Ressourcen bereinigen

Um Ihren Kinesis Stream zu löschen, öffnen Sie die Kinesis Data Streams-Konsole, wählen Sie Ihren Kinesis Stream aus und wählen Sie Aktionen, Löschen.

Bereinigen von MSK-Ressourcen

Führen Sie die Schritte in diesem Abschnitt aus, wenn Sie für dieses Tutorial einen Amazon MSK-Cluster erstellt haben. Dieser Abschnitt enthält Anweisungen zur Bereinigung Ihrer Amazon EC2-Client-Instance, Amazon VPC und Amazon MSK-Cluster.

Löschen Sie Ihren Amazon-MSK-Cluster

Gehen Sie wie folgt vor, wenn Sie für dieses Tutorial einen Amazon MSK-Cluster erstellt haben.

1. Öffnen Sie die Amazon-MSK-Konsole unter https://console.aws.amazon.com/msk/home?region=us-east-1#/home/.
2. Wählen Sie AWSKafkaTutorialCluster. Wählen Sie Delete (Löschen). Geben Sie **`delete`** in das angezeigte Fenster ein und bestätigen Sie Ihre Auswahl.

Beenden Ihrer Client-Instance

Gehen Sie wie folgt vor, wenn Sie für dieses Tutorial eine Amazon EC2-Client-Instance erstellt haben.

1. Öffnen Sie die Amazon EC2-Konsole unter https://console.aws.amazon.com/ec2/.

2. Wählen Sie in der linken Navigationsleiste Instances aus.
3. Aktivieren Sie das Kontrollkästchen neben ZeppelinClient, um es auszuwählen.
4. Wählen Sie Instance-Status, Instance beenden.

Löschen Ihrer Amazon VPC

Gehen Sie wie folgt vor, wenn Sie für dieses Tutorial eine Amazon VPC erstellt haben.

1. Öffnen Sie die Amazon EC2-Konsole unter https://console.aws.amazon.com/ec2/.
2. Wählen Sie in der linken Navigationsleiste Netzwerkschnittstellen aus.
3. Geben Sie Ihre VPC-ID in das Suchfeld ein und drücken Sie die Eingabetaste.
4. Aktivieren Sie das Kontrollkästchen in der Kopfzeile der Tabelle, um alle angezeigten Netzwerkschnittstellen auszuwählen.
5. Wählen Sie Actions (Aktionen), Loslösen (Detach). Wählen Sie in dem daraufhin angezeigten Fenster unter Trennung erzwingen die Option Aktivieren aus. Wählen Sie Trennen und warten Sie, bis alle Netzwerkschnittstellen den Status Verfügbar erreichen.
6. Aktivieren Sie das Kontrollkästchen in der Kopfzeile der Tabelle, um alle angezeigten Netzwerkschnittstellen erneut auszuwählen.
7. Wählen Sie Aktionen, Löschen aus. Bestätigen Sie die Aktion.
8. Öffnen Sie die Amazon-VPC-Konsole unter https://console.aws.amazon.com/vpc/.
9. Wählen Sie AWSKafkaTutorialVPC aus. Wählen Sie Aktionen, Löschen aus. Geben Sie **`delete`** ein und bestätigen Sie den Löschvorgang.

Tutorial: Bereitstellung als Anwendung mit dauerhaftem Zustand

Das folgende Tutorial zeigt, wie Sie ein Studio-Notebook als Managed-Service für Apache Flink-Anwendung mit einem dauerhaften Status bereitstellen.

Dieses Thema enthält die folgenden Abschnitte:

- Aufstellen
- Stellen Sie eine Anwendung mit einem dauerhaften Zustand bereit, indem Sie AWS Management Console
- Stellen Sie eine Anwendung mit dem dauerhaften Status bereit, indem Sie AWS CLI

Aufstellen

Erstellen Sie ein neues Studio-Notizbuch, indem Sie den Anweisungen folgenTutorial zum Erstellen eines Studio-Notebooks und entweder Kinesis Datenstrom oder Amazon MSK verwenden. Name des Studio-Notizbuchs `ExampleTestDeploy`.

Stellen Sie eine Anwendung mit einem dauerhaften Zustand bereit, indem Sie AWS Management Console

1. Fügen Sie einen S3-Bucket-Speicherort hinzu, an dem der gepackte Code unter Speicherort des Anwendungscodes gespeichert werden soll - optional in der Konsole. Dies ermöglicht die Schritte zum Bereitstellen und Ausführen Ihrer Anwendung direkt vom Notebook aus.
2. Fügen Sie der Anwendungsrolle die erforderlichen Berechtigungen hinzu, um die von Ihnen verwendete Rolle zum Lesen und Schreiben in einen Amazon S3-Bucket zu aktivieren und um eine Managed Service for Apache Flink-Anwendung zu starten:
   - AmazonS3FullAccess
   - Amazon Managed — Vollzugriff auf Flink
   - Zugriff auf Ihre Quellen, Ziele und VPCs, sofern zutreffend. Weitere Informationen finden Sie unter IAM-Berechtigungen für Studio-Notebooks.
3. Verwenden Sie den folgenden Beispielcode:

```
%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. Mit der Einführung dieses Feature sehen Sie in der rechten oberen Ecke jeder Notiz in Ihrem Notizbuch ein neues Dropdown-Menü mit dem Namen des Notizbuchs. Sie haben die folgenden Möglichkeiten:

   - Sehen Sie sich die Studio-Notizbucheinstellungen in der AWS Management Console an.
   - Erstellen Sie Ihren Zeppelin Note und exportieren Sie ihn zu Amazon S3. Geben Sie an dieser Stelle einen Namen für Ihre Anwendung ein und wählen Sie Erstellen und Exportieren. Sie erhalten eine Benachrichtigung, wenn der Export abgeschlossen ist.
   - Bei Bedarf können Sie alle zusätzlichen Tests der ausführbaren Datei in Amazon S3 anzeigen und ausführen.
   - Sobald der Build abgeschlossen ist, können Sie Ihren Code als Kinesis-Streaming-Anwendung mit dauerhaftem Zustand und automatischer Skalierung bereitstellen.
   - Verwenden Sie das Drop-down-Menü und wählen Sie Zeppelin Note als Kinesis-Streaming-Anwendung bereitstellen. Überprüfen Sie den Anwendungsnamen und wählen Sie Bereitstellen über AWS Konsole.
   - Dies führt Sie zu der AWS Management Console Seite zur Erstellung von Verwaltetem Dienst für Apache Flink-Anwendung. Beachten Sie, dass Anwendungsname, Parallelität, Codespeicherort, Standard-Glue-DB, VPC (falls zutreffend) und IAM-Rollen vorab ausgefüllt wurden. Stellen Sie sicher, dass die IAM-Rollen über die erforderlichen Berechtigungen für Ihre Quellen und Ziele verfügen. Snapshots sind standardmäßig aktiviert, um eine dauerhafte Verwaltung des Anwendungsstatus zu gewährleisten.
   - Wählen Sie Erstellen der Anwendung.
   - Sie können alle Einstellungen konfigurieren und ändern und anschließend Ausführen wählen, um Ihre Streaming-Anwendung zu starten.

Stellen Sie eine Anwendung mit dem dauerhaften Status bereit, indem Sie AWS CLI

Um eine Anwendung mithilfe von bereitzustellenAWS CLI, müssen Sie Ihr AWS CLI aktualisieren, damit es das mit Ihren Beta 2-Informationen bereitgestellte Servicemodell verwenden kann. Weitere Informationen zur Verwendung des aktualisierten Servicemodells finden Sie unter Setup.

Der folgende Beispielcode erstellt ein neues Studio-Notizbuch:

```
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
```

Das folgende Codebeispiel startet ein Studio-Notebook:

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

Der folgende Code gibt die URL für die Apache Zeppelin-Notizbuch-Seite einer Anwendung zurück:

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

    --region us-east-1 \
    --no-verify-ssl
```

Beispiele

Die folgenden Beispielabfragen zeigen, wie Daten mithilfe von Fensterabfragen in einem Studio-Notebook analysiert werden.

- Erstellen von Tabellen mit Amazon MSK/Apache Kafka
- Erstellen von Tabellen mit Kinesis
- Rollierendes Fenster
- Gleitendes Fenster
- Interaktives SQL
- BlackHole SQL-Konnektor
- Datengenerator
- Interaktives Scala
- Interaktives Python
- Interaktives Python, SQL und Scala
- Kontenübergreifender Kinesis-Datenstrom

Informationen zu den SQL-Abfrageeinstellungen von Apache Flink finden Sie unter Flink auf Zeppelin-Notebooks für interaktive Datenanalyse.

Um Ihre Anwendung im Apache-Flink-Dashboard anzuzeigen, wählen Sie FLINK-AUFTRAG auf der Seite Zeppelin Notiz Ihrer Anwendung.

Weitere Informationen zu Fensterabfragen finden Sie unter Windows in der Apache-Flink-Dokumentation.

Weitere Beispiele für Streaming-SQL-Abfragen in Apache Flink finden Sie unter Abfragen in der Apache-Flink-Dokumentation.

Erstellen von Tabellen mit Amazon MSK/Apache Kafka

Sie können den Amazon-MSK-Flink-Konnektor mit Managed Service für Apache Flink Studio verwenden, um Ihre Verbindung mit Klartext-, SSL- oder IAM-Authentifizierung zu authentifizieren. Erstellen Sie Ihre Tabellen mit den spezifischen Eigenschaften gemäß Ihren Anforderungen.

```
-- 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'
);
```

Sie können diese mit anderen Eigenschaften im Apache-Kafka-SQL-Konnektor kombinieren.

Erstellen von Tabellen mit Kinesis

Im folgenden Beispiel erstellen Sie eine Tabelle mit 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'
);
```

Weitere Informationen zu anderen Eigenschaften, die Sie verwenden können, finden Sie unter Amazon Kinesis Data Streams SQL-Konnektor.

Rollierendes Fenster

Die folgende Flink-Streaming-SQL-Abfrage wählt den höchsten Preis in jedem fünfsekündigen rollierenden Fenster aus der `ZeppelinTopic`-Tabelle aus:

```
%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)
```

Gleitendes Fenster

Die folgende Flink-Streaming-SQL-Abfrage wählt den höchsten Preis in jedem fünfsekündigen rollierenden Fenster aus der `ZeppelinTopic`-Tabelle aus:

```
%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)
```

Interaktives SQL

In diesem Beispiel wird der Höchstwert der Ereignis- und Verarbeitungszeit sowie die Summe der Werte aus der Schlüssel-Wert-Tabelle ausgegeben. Stellen Sie sicher, dass Sie das Beispielskript zur Datengenerierung aus dem laufenden the section called “Datengenerator” haben. Informationen zum Ausprobieren anderer SQL-Abfragen wie Filtern und Joins in Ihrem Studio-Notebook finden Sie in der Apache-Flink-Dokumentation: Abfragen in der Apache-Flink-Dokumentation.

```
%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 SQL-Konnektor

Der BlackHole SQL-Konnektor erfordert nicht, dass Sie einen Kinesis-Datenstrom oder einen Amazon-MSK-Cluster erstellen, um Ihre Abfragen zu testen. Informationen zum BlackHole SQL-Konnektor finden Sie unter BlackHole SQL Connector in der Apache-Flink-Dokumentation. In diesem Beispiel ist der Standardkatalog ein speicherinterner Katalog.

```
%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
```

Datengenerator

In diesem Beispiel wird Scala verwendet, um Beispieldaten zu generieren. Sie können diese Beispieldaten verwenden, um verschiedene Abfragen zu testen. Verwenden Sie die Anweisung create table, um die Schlüssel-Wert-Tabelle zu erstellen.

```
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`
```

Interaktives Scala

Dies ist die Scala-Übersetzung von the section called “Interaktives SQL”. Weitere Scala-Beispiele finden Sie unter Tabellen-API in der Apache-Flink-Dokumentation.

```
%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`
```

Interaktives Python

Dies ist die Python-Übersetzung von the section called “Interaktives SQL”. Weitere Python-Beispiele finden Sie unter Tabellen-API in der Apache-Flink-Dokumentation.

```
%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`
```

Interaktives Python, SQL und Scala

Sie können eine beliebige Kombination aus SQL, Python und Scala in Ihrem Notebook für interaktive Analysen verwenden. In einem Studio-Notebook, das Sie als dauerhafte Anwendung bereitstellen möchten, können Sie eine Kombination aus SQL und Scala verwenden. Dieses Beispiel zeigt Ihnen die Abschnitte, die ignoriert werden, und diejenigen, die in der Anwendung mit dem dauerhaften Zustand bereitgestellt werden.

```
%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!")
```

Kontenübergreifender Kinesis-Datenstrom

Um einen Kinesis-Datenstrom zu verwenden, der sich in einem anderen Konto als dem Konto befindet, das Ihr Studio-Notebook enthält, erstellen Sie eine Serviceausführungsrolle in dem Konto, in dem Ihr Studio-Notebook ausgeführt wird, und eine Rollenvertrauensrichtlinie für das Konto, das den Datenstrom enthält. Verwenden Sie `aws.credentials.provider`, `aws.credentials.role.arn` und `aws.credentials.role.sessionName` im Kinesis-Konnektor in Ihrer DDL-Anweisung create table, um eine Tabelle anhand des Datenstroms zu erstellen.

Verwenden Sie die folgende Serviceausführungsrolle für das Studio-Notebook-Konto.

```
{
 "Sid": "AllowNotebookToAssumeRole",
 "Effect": "Allow",
 "Action": "sts:AssumeRole"
 "Resource": "*"
}
```

Verwenden Sie die `AmazonKinesisFullAccess`-Richtlinie und die folgende Rollenvertrauensrichtlinie für das Datenstrom-Konto.

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "AWS": "arn:aws:iam::<accountID>:root"
 },
 "Action": "sts:AssumeRole",
 "Condition": {}
 }
 ]
}
```

Verwenden Sie den folgenden Absatz für die create-table-Anweisung.

```
%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'
)
```

Fehlerbehebung

Dieser Abschnitt enthält Informationen zur Fehlerbehebung für Studio-Notebooks.

Anhalten einer hängengebliebenen Anwendung

Um eine Anwendung anzuhalten, die in einem vorübergehenden Zustand hängen bleibt, rufen Sie die StopApplication Aktion auf, wobei der `Force` Parameter auf gesetzt ist`true`. Weitere Informationen finden Sie unter Ausführen von Anwendungen im Managed Service für Apache Flink Entwicklerhandbuch.

Bereitstellen als Anwendung mit dauerhaftem Zustand in einer VPC ohne Internetzugang

Die deploy-as-application Funktion Managed Service für Apache Flink Studio unterstützt keine VPC-Anwendungen ohne Internetzugang. Wir empfehlen, dass Sie Ihre Anwendung in Studio erstellen und dann Managed Service für Apache Flink verwenden, um manuell eine Flink-Anwendung zu erstellen und die ZIP-Datei auszuwählen, die Sie in Ihrem Notebook erstellt haben.

Die folgenden Schritte beschreiben, wie Sie dies tun:

1. Erstellen und exportieren Sie Ihre Studio-Anwendung in Amazon S3. Dies sollte eine ZIP-Datei sein.
2. Erstellen Sie manuell eine Anwendung, die Managed Service für Apache Flink nutzt, mit einem Codepfad, der auf den Speicherort der ZIP-Datei in Amazon S3 verweist. Darüber hinaus

müssen Sie die Anwendung mit den folgenden env-Variablen (2 `groupID`, 3 `var` insgesamt) konfigurieren:

3. kinesis.analytics.flink.run.options

   a. python: source/note.py

   b. jarfile: lib/PythonApplicationDependencies.jar

4. managed.deploy_as_app.options

   - DatabaseARN: *<glue database ARN (Amazon Resource Name)>*

5. Möglicherweise müssen Sie Managed Service für Apache Flink Studio und Managed Service für Apache Flink IAM-Rollen für die Services, die Ihre Anwendung verwendet, Berechtigungen erteilen. Sie können dieselbe IAM-Rolle für beide Apps verwenden.

Reduzierung der D-eploy-as-app Größe und der Build-Zeit

Studio deploy-as-app -für-Python-Anwendungen packen alles, was in der Python-Umgebung verfügbar ist, da wir nicht ermitteln können, welche Bibliotheken Sie benötigen. Dies kann zu einer Größe führen, die größer als nötig deploy-as-app ist. Das folgende Verfahren zeigt, wie Sie die Größe der deploy-as-app Python-Anwendung reduzieren können, indem Sie Abhängigkeiten deinstallieren.

Wenn Sie eine Python-Anwendung mit - deploy-as-app Funktion aus Studio erstellen, können Sie erwägen, vorinstallierte Python-Pakete aus dem System zu entfernen, wenn Ihre Anwendungen nicht davon abhängig sind. Dies trägt nicht nur dazu bei, die endgültige Artefaktgröße zu reduzieren, um eine Überschreitung des Servicelimits für die Anwendungsgröße zu vermeiden, sondern auch die Build-Zeit von Anwendungen mit der deploy-as-app Funktion zu verbessern.

Sie können den folgenden Befehl ausführen, um alle installierten Python-Pakete mit ihrer jeweiligen installierten Größe aufzulisten und Pakete mit signifikanter Größe selektiv zu entfernen.

```
%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` wird von Flink Python zum Betrieb benötigt. Sie sollten dieses Paket und seine Abhängigkeiten niemals entfernen.

Im Folgenden finden Sie eine Liste der vorinstallierten Python-Pakete in Studio V2, deren Entfernung in Betracht gezogen werden kann:

```
scipy
statsmodels
plotnine
seaborn
llvmlite
bokeh
pandas
matplotlib
botocore
boto3
numba
```

So entfernen Sie ein Python-Paket aus dem Zeppelin-Notebook:

1. Prüfen Sie, ob Ihre Anwendung von dem Paket oder einem seiner konsumierenden Pakete abhängt, bevor Sie es entfernen. Mit pipdeptree können Sie die Abhängigkeiten eines Pakets identifizieren.
2. Führen Sie den folgenden Befehl aus, um ein Paket zu entfernen:

   ```
   %flink.pyflink
   !pip uninstall -y <package-to-remove>
   ```

3. Wenn Sie ein Paket abrufen müssen, das Sie versehentlich entfernt haben, führen Sie den folgenden Befehl aus:

   ```
   %flink.pyflink
   !pip install <package-to-install>
   ```

Example Beispiel: Entfernen Sie das -**`scipy`**Paket, bevor Sie Ihre Python-Anwendung mit - deploy-as-app Funktion bereitstellen.

1. Verwenden Sie `pipdeptree`, um alle `scipy`-Verbraucher zu ermitteln und zu überprüfen, ob Sie `scipy` sicher entfernen können.

   - Installieren Sie das Tool über das Notebook:

     ```
     %flink.pyflink
     !pip install pipdeptree
     ```

   - Rufen Sie den umgekehrten Abhängigkeitsbaum von `scipy` ab, indem Sie Folgendes ausführen:

     ```
     %flink.pyflink
     !pip -r -p scipy
     ```

     Sie sollten eine ähnliche Ausgabe wie die folgende sehen (aus Platzgründen gekürzt):

     ```
     ...
     ------------------------------------------------------------------------
     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. Prüfen Sie sorgfältig die Verwendung von `seaborn`, `statsmodels` und `plotnine` in Ihren Anwendungen. Wenn Ihre Anwendungen nicht von `scipy`, `seaborn`, `statemodels` oder `plotnine` abhängig sind, können Sie alle diese Pakete oder nur diejenigen entfernen, die Ihre Anwendungen nicht benötigen.
3. Entfernen Sie das Paket, indem Sie Folgendes ausführen:

   ```
   !pip uninstall -y scipy plotnine seaborn statemodels
   ```

Abbrechen von Aufträgen

In diesem Abschnitt erfahren Sie, wie Sie Apache-Flink-Aufträge abbrechen, auf die Sie von Apache Zeppelin aus nicht zugreifen können. Wenn Sie einen solchen Auftrag abbrechen möchten, rufen Sie

das Apache-Flink-Dashboard auf, kopieren Sie die Auftrags-ID und verwenden Sie sie dann in einem der folgenden Beispiele.

Um einen einzelnen Auftrag abzubrechen:

```
%flink.pyflink
import requests

requests.patch("https://zeppelin-flink:8082/jobs/[job_id]", verify=False)
```

Um alle laufenden Aufträge abzubrechen:

```
%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))
```

Um alle Aufträge abzubrechen:

```
%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)
```

Neustarten des Apache-Flink-Interpreters

Um den Apache-Flink-Interpreter in Ihrem Studio-Notebook neu zu starten

1. Wählen Sie Konfiguration in der oberen rechten Ecke des Bildschirms.
2. Wählen Sie Interpreter.

3. Wählen Sie Neustart und dann OK.

Anhang: Erstellen benutzerdefinierter IAM-Richtlinien

Normalerweise verwenden Sie verwaltete IAM-Richtlinien, um Ihrer Anwendung den Zugriff auf abhängige Ressourcen zu ermöglichen. Wenn Sie eine genauere Kontrolle über die Berechtigungen Ihrer Anwendung benötigen, können Sie eine benutzerdefinierte IAM-Richtlinie verwenden. Dieser Abschnitt enthält Beispiele für benutzerdefinierte IAM-Richtlinien.

> **Note**
>
> Ersetzen Sie in den folgenden Richtlinienbeispielen den Platzhaltertext durch die Werte Ihrer Anwendung.

Dieses Thema enthält die folgenden Abschnitte:

- AWS Glue
- CloudWatch Protokolle
- Kinesis-Streams
- Amazon-MSK-Cluster

AWS Glue

Die folgende Beispielrichtlinie gewährt die Berechtigung zum Zugriff auf eine AWS Glue-Datenbank.

```
{
    "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 Protokolle

Die folgende Richtlinie gewährt Berechtigungen für den Zugriff auf CloudWatch Protokolle:

```
{
      "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**
>
> Wenn Sie Ihre Anwendung mit der Konsole erstellen, fügt die Konsole die erforderlichen Richtlinien für den Zugriff auf CloudWatch Protokolle zu Ihrer Anwendungsrolle hinzu.

Kinesis-Streams

Ihre Anwendung kann einen Kinesis-Stream für eine Quelle oder ein Ziel verwenden. Ihre Anwendung benötigt Leseberechtigungen, um aus einem Quell-Stream zu lesen, und Schreibberechtigungen, um in einen Ziel-Stream zu schreiben.

Die folgende Richtlinie gewährt Berechtigungen zum Lesen aus einem Kinesis-Stream, der als Quelle verwendet wird:

```
{
  "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/*"
    }
  ]
}
```

Die folgende Richtlinie gewährt Berechtigungen zum Schreiben in einen Kinesis-Stream, der als Ziel verwendet wird:

```
{
    "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>"
        }
    ]
}
```

Wenn Ihre Anwendung auf einen verschlüsselten Kinesis-Stream zugreift, müssen Sie zusätzliche Berechtigungen für den Zugriff auf den Stream und den Verschlüsselungsschlüssel des Streams gewähren.

Die folgende Richtlinie gewährt Berechtigungen für den Zugriff auf einen verschlüsselten Quell-Stream und den Verschlüsselungsschlüssel des Streams:

```
{
      "Sid": "ReadEncryptedKinesisStreamSource",
      "Effect": "Allow",
      "Action": [
        "kms:Decrypt"
      ],
      "Resource": [
        "<inputStreamKeyArn>"
      ]
    }
    ,
```

Die folgende Richtlinie gewährt Berechtigungen für den Zugriff auf einen verschlüsselten Ziel-Stream und den Verschlüsselungsschlüssel des Streams:

```
{
      "Sid": "WriteEncryptedKinesisStreamSink",
      "Effect": "Allow",
      "Action": [
        "kms:GenerateDataKey"
      ],
      "Resource": [
        "<outputStreamKeyArn>"
      ]
    }
```

Amazon-MSK-Cluster

Um Zugriff auf einen Amazon-MSK-Cluster zu gewähren, gewähren Sie Zugriff auf die VPC des Clusters. Richtlinienbeispiele für den Zugriff auf eine Amazon VPC finden Sie unter VPC-Anwendungsberechtigungen.

Erste Schritte mit Amazon Managed Service für Apache Flink (DataStream API)

In diesem Abschnitt werden Ihnen die grundlegenden Konzepte von Managed Service für Apache Flink und der DataStream -API vorgestellt. Es werden die verfügbaren Optionen für die Erstellung und das Testen von Anwendungen beschrieben. Er enthält auch Anweisungen zur Installation der Tools, die Sie benötigen, um die Tutorials in diesem Handbuch abzuschließen und Ihre erste Anwendung zu erstellen.

Themen

- Komponenten der Anwendung Managed Service für Apache Flink
- Voraussetzungen für das Fertigstellen der Übungen
- Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Administratorbenutzers
- Schritt 2: Einrichten der AWS Command Line Interface (AWS CLI)
- Schritt 3: Erstellen und Ausführen eines Managed Service für Apache Flink-Anwendung
- Schritt 4: Bereinigen von AWS-Ressourcen
- Schritt 5: Nächste Schritte

Komponenten der Anwendung Managed Service für Apache Flink

Zur Verarbeitung von Daten verwendet Ihre Managed Service für Apache Flink-Anwendung eine Java/Apache Maven- oder Scala-Anwendung, die mithilfe der Apache Flink-Laufzeit Eingaben verarbeitet und Ausgaben erzeugt.

Eine Anwendung Managed Service für Apache Flink besteht aus folgenden Komponenten:

- Runtime-Eigenschaften: Sie können Runtime-Eigenschaften verwenden, um Ihre Anwendung zu konfigurieren, ohne Ihren Anwendungscode neu kompilieren zu müssen.
- Quelle: Die Anwendung verwendet Daten mithilfe einer Quelle. Ein Quell-Connector liest Daten aus einem Kinesis Data Stream, einem Amazon S3-Bucket usw. Weitere Informationen finden Sie unter Quellen.
- Operatoren: Die Anwendung verarbeitet Daten mithilfe eines oder mehrerer Operatoren. Ein Operator kann Daten transformieren, anreichern oder aggregieren. Weitere Informationen finden Sie unter DataStream-API Operatoren.

- Senke: Die Anwendung erzeugt Daten für externe Quellen mithilfe von Senken. Ein Senken-Connector schreibt Daten in einen Kinesis Data Stream, einen Kinesis Data Firehose-Stream, einen Amazon S3-Bucket usw. Weitere Informationen finden Sie unter Senken.

Nachdem Sie Ihren Anwendungscode erstellt, kompiliert und verpackt haben, laden Sie das Codepaket in einen Amazon Simple Storage Service (Amazon S3)-Bucket hoch. Anschließend erstellen Sie eine Managed Service für Apache Flink-Anwendung. Sie übergeben den Speicherort des Codepakets, einen Kinesis Data Stream als Streaming-Datenquelle und in der Regel einen Streaming- oder Dateispeicherort, der die verarbeiteten Daten der Anwendung empfängt.

Voraussetzungen für das Fertigstellen der Übungen

Zur Durchführung der Schritte in dieser Anleitung benötigen Sie Folgendes:

- Java Development Kit (JDK), Version 11. Legen Sie die `JAVA_HOME` Umgebungsvariable so fest, dass sie auf Ihren JDK-Installationsspeicherort weist.
- Wir empfehlen die Verwendung einer Entwicklungsumgebung (wie Eclipse Java Neon oder IntelliJ Idea), um Ihre Anwendung zu entwickeln und zu kompilieren.
- Git-Client. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben.
- Apache Maven-Compiler-Plugin. Maven muss sich in Ihrem Arbeitspfad befinden. Zum Testen Ihrer Apache Maven-Installation geben Sie Folgendes ein:

```
$ mvn -version
```

Um zu beginnen, gehen Sie zu Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Administratorbenutzers.

Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Administratorbenutzers

Führen Sie die folgenden Aufgaben aus, bevor Sie Managed Service für Apache Flink zum ersten Mal verwenden:

So melden Sie sich für ein AWS-Konto an

Wenn Sie kein AWS-Konto haben, führen Sie die folgenden Schritte zum Erstellen durch.

Anmeldung für ein AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für ein AWS-Konto anmelden, wird ein Root-Benutzer des AWS-Kontos erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen eine Bestätigungs-E-Mail, sobald die Anmeldung abgeschlossen ist. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Erstellen eines Administratorbenutzers

Nachdem Sie sich für ein AWS-Kontoangemeldet haben, sichern Sie Ihr Root-Benutzer des AWS-Kontos, aktivieren Sie AWS IAM Identity Centerund erstellen Sie einen administrativen Benutzer, damit Sie nicht den Root-Benutzer für alltägliche Aufgaben verwenden.

Schützen Ihres Root-Benutzer des AWS-Kontos

1. Melden Sie sich bei der AWS Management Console als Kontobesitzer an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto-E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-AnmeldungBenutzerhandbuch zu .
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen dazu finden Sie unter Aktivieren eines virtuellen MFA-Geräts für den Root-Benutzer Ihres AWS-Konto (Konsole) im IAM-Benutzerhandbuch.

Erstellen eines Administratorbenutzers

1. Aktivieren von IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.
2. Im IAM Identity Center gewähren Sie einem administrativen Benutzer administrativen Zugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit dem standardmäßigen IAM-Identity-Center-Verzeichnis konfigurieren im AWS IAM Identity Center-Benutzerhandbuch.

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM-Identity-Center-Benutzer finden Sie unter Anmelden beim AWS-Zugangsportal im AWS-Anmeldung Benutzerhandbuch zu.

Erteilen programmgesteuerten Zugriffs

Benutzer benötigen programmgesteuerten Zugriff, wenn sie außerhalb der AWS Management Console mit AWS interagieren möchten. Die Vorgehensweise, um programmgesteuerten Zugriff zu gewähren, hängt davon ab, welcher Benutzertyp auf zugreift AWS.

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zur AWS CLI finden Sie unter Konfiguri

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
		eren der AWS CLI für die Verwendung von AWS IAM Identity Center im AWS Command Line Interface-Benutzerhandbuch. • Informationen zu AWS-SDKs, Tools und AWS-APIs finden Sie unter IAM-Identity-Center-Authentifizierung im Referenzhandbuch zu AWS-SDKs und Tools.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Folgen Sie den Anweisungen unter Verwenden temporärer Anmeldeinformationen mit AWS-Ressourcen im IAM-Benutzerhandbuch.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zur AWS CLI finden Sie unter Authentifizierung mit IAM-Benutzer-Anmeldeinformationen im AWS Command Line Interface-Benutzerhandbuch. • Informationen zu AWS-SDKs und Tools finden Sie unter Authentifizierung mit langfristigen Anmeldeinformationen im Referenzhandbuch zu AWS-SDKs und Tools. • Informationen zu AWS-APIs finden Sie unter Verwalten von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch.

Nächster Schritt

Schritt 2: Einrichten der AWS Command Line Interface (AWS CLI)

Schritt 2: Einrichten der AWS Command Line Interface (AWS CLI)

In diesem Schritt laden Sie die AWS CLI herunter und konfigurieren sie für die Verwendung mit Amazon Managed Service für Apache Flink.

> **Note**
>
> Bei allen Erste-Schritte-Übungen in diesem Handbuch wird davon ausgegangen, dass Sie in Ihrem Konto Administrator-Anmeldeinformationen (`adminuser`) verwenden, um die Operationen auszuführen.

> **Note**
>
> Wenn Sie die AWS CLI bereits installiert haben, müssen Sie möglicherweise ein Upgrade durchführen, um die neueste Funktionalität zu erhalten. Weitere Informationen finden Sie unter Installieren der AWS Command Line Interface im AWS Command Line Interface-Benutzerhandbuch. Zum Überprüfen der Version der AWS CLI führen Sie den folgenden Befehl aus:
>
> ```
> aws --version
> ```
>
> Die Übungen in diesem Tutorial erfordern die folgende AWS CLI-Version oder höher:
>
> ```
> aws-cli/1.16.63
> ```

Um das AWS CLI einzurichten

1. Herunterladen und Konfigurieren von AWS CLI. Eine Anleitung finden Sie unter den folgenden Themen im AWS Command Line Interface-Benutzerhandbuch:
   - Installieren des AWS Command Line Interface
   - Konfigurieren von AWS CLI
2. Fügen Sie ein benanntes Profil für den Administratorbenutzer in der AWS CLI `config`-Datei hinzu. Verwenden Sie dieses Profil beim Ausführen von AWS CLI-Befehlen. Weitere Informationen zu benannten Profilen finden Sie unter Benannte Profile im AWS Command Line Interface Benutzerhandbuch.

   ```
   [profile adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   ```

```
region = aws-region
```

Eine Liste der verfügbaren AWS-Regionen finden Sie unter Regionen und Endpunkte im Allgemeine Amazon Web Services-Referenz

> **Note**
>
> Der Beispielcode und die Befehle in diesem Tutorial verwenden die Region USA West (Oregon). Um eine andere Region zu verwenden, ändern Sie die Region im Code und in den Befehlen für dieses Tutorial in die Region, die Sie verwenden möchten.

3. Überprüfen Sie die Einrichtung, indem Sie die folgenden Hilfebefehle in die Befehlszeile eingeben:

```
aws help
```

Nachdem Sie ein AWS Konto und die eingerichtet habenAWS CLI, können Sie die nächste Übung ausprobieren, in der Sie eine Beispielanwendung konfigurieren und die end-to-end Einrichtung testen.

Nächster Schritt

Schritt 3: Erstellen und Ausführen eines Managed Service für Apache Flink-Anwendung

Schritt 3: Erstellen und Ausführen eines Managed Service für Apache Flink-Anwendung

In dieser Übung erstellen Sie eine Anwendung von Managed Service für Apache Flink mit Datenströmen als Quelle und Senke.

Dieser Abschnitt enthält die folgenden Schritte:

- Erstellen von zwei Amazon Kinesis Datenstroms
- Schreiben Sie Beispieldatensätze in den Eingabe-Stream
- Herunterladen und Überprüfen des Apache Flink-Streaming-Java-Codes
- Kompilieren des Anwendungscodes
- Hochladen des Apache Flink-Streaming-Java-Codes

- Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus
- Nächster Schritt

Erstellen von zwei Amazon Kinesis Datenstroms

Bevor Sie für diese Übung eine Anwendung von Managed Service für Apache Flink erstellen, erstellen Sie zwei Kinesis Data Streams (`ExampleInputStream` und `ExampleOutputStream`). Ihre Anwendung verwendet diese Streams für die Quell- und Ziel-Streams der Anwendung.

Sie können diese Streams mithilfe der Amazon-Kinesis-Konsole oder des folgenden AWS CLI-Befehls erstellen. Anweisungen für die Konsole finden Sie unter Erstellen und Aktualisieren von Datenströmen im Amazon Kinesis Data Streams Entwicklerhandbuch.

So erstellen Sie die Daten-Streams (AWS CLI)

1. Verwenden Sie den Befehl Amazon Kinesis `create-stream` AWS CLI, um den ersten Stream (`ExampleInputStream`) zu erstellen.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Um den zweiten Stream zu erstellen, den die Anwendung zum Schreiben der Ausgabe verwendet, führen Sie denselben Befehl aus und ändern den Stream-Namen in `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Abschnitt verwenden Sie ein Python-Skript zum Schreiben von Datensätzen in den Stream für die zu verarbeitende Anwendung.

> **Note**
>
> Dieser Abschnitt erfordert AWS SDK for Python (Boto).

1. Erstellen Sie eine Datei `stock.py` mit dem folgenden Inhalt:

```
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. Im weiteren Verlauf des Tutorials führen Sie das `stock.py`-Skript zum Senden von Daten an die Anwendung aus.

```
$ python stock.py
```

Herunterladen und Überprüfen des Apache Flink-Streaming-Java-Codes

Der Java-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Klonen Sie das Remote-Repository, indem Sie den folgenden Befehl verwenden:

   ```
   git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
   ```

2. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/GettingStarted` Verzeichnis .

Beachten Sie Folgendes zum Anwendungscode:

- Eine Project Object Model (pom.xml) Datei enthält Informationen über die Konfiguration und Abhängigkeiten der Anwendung, einschließlich der Bibliotheken des Managed Service für Apache Flink.
- Die `BasicStreamingJob.java`-Datei enthält die `main`-Methode, die die Funktionalität der Anwendung definiert.
- Die Anwendung verwendet eine Kinesis-Quelle zum Lesen aus dem Quell-Stream. Der folgende Codeausschnitt erstellt die Kinesis-Quelle:

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

- Ihre Anwendung erstellt Quell- und Senkenkonnektoren für den Zugriff auf externe Ressourcen, indem ein `StreamExecutionEnvironment`-Objekt verwendet wird.
- Die Anwendung erstellt Quell- und Senkenkonnektoren mit statischen Eigenschaften. Zum Verwenden dynamischer Anwendungseigenschaften verwenden Sie die Methoden `createSourceFromApplicationProperties` und `createSinkFromApplicationProperties`, um die Konnektoren zu erstellen. Diese Methoden lesen die Eigenschaften der Anwendung zum Konfigurieren der Konnektoren.

  Weitere Informationen zu den Laufzeiteigenschaften finden Sie unter Laufzeiteigenschaften.

Kompilieren des Anwendungscodes

In diesem Abschnitt verwenden Sie den Apache Maven-Compiler zum Erstellen des Java-Codes für die Anwendung. Weitere Informationen zum Installieren von Apache Maven und des Java Development Kit (JDK) finden Sie unter Voraussetzungen für das Fertigstellen der Übungen.

Kompilieren des Anwendungscodes

1. Zum Verwenden Ihres Anwendungscodes kompilieren und packen Sie ihn in eine JAR-Datei. Sie können Ihren Code auf zwei Arten kompilieren und packen:

   - Verwenden Sie das Befehlszeilen-Maven-Tool. Erstellen Sie Ihre JAR-Datei, indem Sie den folgenden Befehl in dem Verzeichnis ausführen, das die `pom.xml`-Datei enthält:

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

   - Verwenden Sie Ihre Entwicklungsumgebung. Weitere Informationen finden Sie in der Dokumentation Ihrer Entwicklungsumgebung.

     > **Note**
     >
     > Der bereitgestellte Quellcode basiert auf Bibliotheken aus Java 11.

   Sie können Ihr Paket als JAR-Datei hochladen oder komprimieren und als ZIP-Datei hochladen. Wenn Sie Ihre Anwendung mit der AWS CLI erstellen, geben Sie Ihren Codeinhaltstyp (JAR oder ZIP) an.

2. Wenn während der Erstellung Fehler aufgetreten sind, überprüfen Sie, ob Ihre `JAVA_HOME`-Umgebungsvariable richtig eingestellt ist.

Wenn die Anwendung erfolgreich kompiliert wurde, wird die folgende Datei erstellt:

```
target/aws-kinesis-analytics-java-apps-1.0.jar
```

Hochladen des Apache Flink-Streaming-Java-Codes

In diesem Abschnitt erstellen Sie einen Amazon Simple Storage Service (Amazon S3)-Bucket und laden Ihren Anwendungscode hoch.

So laden Sie den Anwendungscode hoch

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie Bucket erstellen aus.
3. Geben Sie **`ka-app-code-<username>`** im Feld Bucket-Name ein. Fügen Sie dem Bucket-Namen ein Suffix hinzu, wie z. B. Ihren Benutzernamen, damit er global eindeutig ist. Wählen Sie Weiter aus.
4. Lassen Sie im Schritt Optionen konfigurieren die Einstellungen unverändert und klicken Sie auf Weiter.
5. Lassen Sie im Schritt Berechtigungen festlegen die Einstellungen unverändert und klicken Sie auf Weiter.
6. Wählen Sie Bucket erstellen aus.
7. Wählen Sie in der Amazon S3-Konsole den Bucket ka-app-code-**`<username>`** und dann Hochladen aus.
8. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `aws-kinesis-analytics-java-apps-1.0.jar`Datei, die Sie im vorherigen Schritt erstellt haben. Wählen Sie Weiter aus.
9. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus

Sie können eine Anwendung von Managed Service für Apache Flink entweder über die Konsole oder die AWS CLI erstellen und ausführen.

> **Note**
>
> Wenn Sie die Anwendung mit der Konsole erstellen, werden Ihre AWS Identity and Access Management (IAM)- und Amazon- CloudWatch Logs-Ressourcen für Sie erstellt. Wenn Sie die Anwendung mit der AWS CLI erstellen, erstellen Sie diese Ressourcen separat.

Themen

- Erstellen und Ausführen der Anwendung (Konsole)
- Erstellen und Ausführen der Anwendung (AWS CLI)

Erstellen und Ausführen der Anwendung (Konsole)

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Geben Sie für Beschreibung den Text **`My java test app`** ein.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.
   - Belassen Sie den Versions-Pulldown bei Apache Flink Version 1.15.2 (empfohlene Version).
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rolle: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf die Kinesis-Datenströme.

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **`kinesis-analytics-service-MyApplication-us-west-2`**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*012345678901*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):

- Geben Sie für Amazon-S3-Bucket **ka-app-code-*<username>*** ein.
- Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **aws-kinesis-analytics-java-apps-1.0.jar** ein.

3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **kinesis-analytics-MyApplication-us-west-2** erstellen/aktualisieren aus.
4. Geben Sie unter Eigenschaften für Gruppen-ID den Text **ProducerConfigProperties** ein.
5. Geben Sie die folgenden Eigenschaften und Werte der Anwendung ein:

Gruppen-ID	Schlüssel	Wert
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

6. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
7. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
8. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die Amazon- CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`

Ausführen der Anwendung

Das Flink-Jobdiagramm kann angezeigt werden, indem Sie die Anwendung ausführen, das Apache Flink-Dashboard öffnen und den gewünschten Flink-Job auswählen.

Stoppen der Anwendung

Wählen Sie auf der MyApplication Seite Stoppen aus. Bestätigen Sie die Aktion.

Aktualisieren der Anwendung

Mithilfe der Konsole können Sie Anwendungseinstellungen wie beispielsweise Anwendungseigenschaften, Überwachungseinstellungen und den Speicherort oder den Dateinamen der JAR-Anwendungsdatei aktualisieren. Außerdem können Sie die JAR-Anwendungsdatei erneut aus dem Amazon-S3-Bucket laden, wenn Sie den Anwendungscode aktualisieren müssen.

Wählen Sie auf der MyApplication Seite Konfigurieren aus. Aktualisieren Sie die Anwendungseinstellungen und klicken Sie auf Aktualisieren.

Erstellen und Ausführen der Anwendung (AWS CLI)

In diesem Abschnitt verwenden Sie AWS CLI, um die Anwendung Managed Service für Apache Flink zu erstellen und auszuführen. Managed Service für Apache Flink verwendet den Befehl `kinesisanalyticsv2` AWS CLI, um Managed-Service-für-Apache-Flink-Anwendungen zu erstellen und mit diesen zu interagieren.

Erstellen einer Berechtigungsrichtlinie

> **Note**
>
> Sie müssen eine Berechtigungsrichtlinie und eine Rolle für Ihre Anwendung erstellen. Wenn Sie diese IAM-Ressourcen nicht erstellen, kann Ihre Anwendung nicht auf ihre Daten und Protokollstreams zugreifen.

Zuerst erstellen Sie eine Berechtigungsrichtlinie mit zwei Anweisungen: eine, die Berechtigungen für die `read`-Aktion auf den Quell-Stream zulässt, und eine andere, die Berechtigungen für die `write`-Aktionen auf den Senken-Stream zulässt. Anschließend fügen Sie die Richtlinie an eine IAM-Rolle (die Sie im nächsten Abschnitt erstellen) an. Wenn Managed Service für Apache Flink also die Rolle

übernimmt, verfügt der Service über die erforderlichen Berechtigungen zum Lesen aus dem Quell-Stream und zum Schreiben in den Senken-Stream.

Verwenden Sie den folgenden Code zum Erstellen der `AKReadSourceStreamWriteSinkStream`-Berechtigungsrichtlinie. Ersetzen Sie *`username`* durch den Benutzernamen, den Sie verwendet haben, um den Amazon-S3-Bucket zum Speichern des Anwendungscodes zu erstellen. Ersetzen Sie die Konto-ID in den Amazon-Ressourcennamen (ARNs) (*`012345678901`*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

step-by-step Anweisungen zum Erstellen einer Berechtigungsrichtlinie finden Sie unter Tutorial: Erstellen und Anfügen Ihrer ersten vom Kunden verwalteten Richtlinie im IAM-Benutzerhandbuch.

> **Note**
>
> Um auf andere Amazon-Services zuzugreifen, können Sie AWS SDK for Java verwenden. Managed Service für Apache Flink setzt die vom SDK benötigten Anmeldeinformationen automatisch auf die der IAM-Rolle für die Dienstausführung, die mit Ihrer Anwendung verknüpft ist. Es sind keine weiteren Schritte erforderlich.

Erstellen einer IAM-Rolle

In diesem Abschnitt erstellen Sie eine IAM-Rolle, die die Anwendung von Managed Service für Apache Flink annehmen kann, um einen Quell-Stream zu lesen und in den Senken-Stream zu schreiben.

Managed Service für Apache Flink kann ohne Berechtigungen nicht auf Ihren Stream zugreifen. Sie erteilen diese Berechtigungen über eine IAM-Rolle. Jeder IAM-Rolle sind zwei Richtlinien angefügt. Die Vertrauensrichtlinie erteilt Managed Service für Apache Flink die Berechtigung zum Übernehmen der Rolle und die Berechtigungsrichtlinie bestimmt, was Managed Service für Apache Flink nach Annahme der Rolle tun kann.

Sie können die Berechtigungsrichtlinie, die Sie im vorherigen Abschnitt erstellt haben, dieser Rolle anfügen.

So erstellen Sie eine IAM-Rolle

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Roles (Rollen) und Create Role (Rolle erstellen) aus.
3. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option AWS-Service aus. Wählen Sie unter Choose the service that will use this role (Wählen Sie den Service aus, der diese Rolle verwendet) die Option Kinesis aus. Wählen Sie unter Select your use case (Wählen Sie Ihren Anwendungsfall aus) die Option Kinesis Analytics aus.

   Wählen Sie Weiter: Berechtigungen aus.
4. Wählen Sie auf der Seite Attach permissions policies (Berechtigungsrichtlinien hinzufügen) Next: Review (Weiter: Überprüfen) aus. Sie fügen Berechtigungsrichtlinien an, nachdem Sie die Rolle erstellt haben.
5. Geben Sie auf der Seite Create role (Rolle erstellen) den Text **`MF-stream-rw-role`** für Role name (Rollenname) ein. Wählen Sie Rolle erstellen aus.

Jetzt haben Sie eine neue IAM-Rolle mit dem Namen `MF-stream-rw-role` erstellt. Im nächsten Schritt aktualisieren Sie die Vertrauens- und Berechtigungsrichtlinien für die Rolle.

6. Fügen Sie die Berechtigungsrichtlinie der Rolle an.

> **Note**
>
> Für diese Übung übernimmt Managed Service für Apache Flink diese Rolle sowohl für das Lesen von Daten aus einem Kinesis-Datenstrom (Quelle) als auch zum Schreiben der Ausgabedaten in einen anderen Kinesis-Datenstrom. Daher fügen Sie die Richtlinie an, die Sie im vorherigen Schritt erstellt haben, the section called “Erstellen einer Berechtigungsrichtlinie”.

   a. Wählen Sie auf der Seite Summary (Übersicht) die Registerkarte Permissions (Berechtigungen) aus.
   b. Wählen Sie Attach Policies (Richtlinien anfügen) aus.
   c. Geben Sie im Suchfeld **`AKReadSourceStreamWriteSinkStream`** (die Richtlinie, die Sie im vorhergehenden Abschnitt erstellt haben) ein.
   d. Wählen Sie die AKReadSourceStreamWriteSinkStream-Richtlinie und dann Richtlinie anfügen aus.

Sie haben nun die Service-Ausführungsrolle erstellt, die Ihre Anwendung für den Zugriff auf Ressourcen verwendet. Notieren Sie sich den ARN der neuen Rolle.

step-by-step Anweisungen zum Erstellen einer Rolle finden Sie unter Erstellen einer IAM-Rolle (Konsole) im IAM-Benutzerhandbuch.

Erstellen Sie die Anwendung Managed Service für Apache Flink

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `create_request.json`. Ersetzen Sie den Beispiel-Rollen-ARN durch den ARN für die Rolle, die Sie zuvor erstellt haben. Ersetzen Sie das Bucket-ARN-Suffix (*`username`*) mit dem Suffix, das Sie im vorherigen Abschnitt gewählt haben. Ersetzen Sie die beispielhafte Konto-ID (*`012345678901`*) in der Service-Ausführungsrolle mit Ihrer Konto-ID.

```
{
    "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. Führen Sie die `CreateApplication`-Aktion mit der vorherigen Anforderung zum Erstellen der Anwendung aus:

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

Die Anwendung wird nun erstellt. Sie starten die Anwendung im nächsten Schritt.

Starten der Anwendung

In diesem Abschnitt verwenden Sie die StartApplication-Aktion, um die Anwendung zu starten.

So starten Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `start_request.json`.

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

2. Führen Sie die StartApplication-Aktion mit der vorherigen Anforderung zum Starten der Anwendung aus:

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

Die Anwendung wird jetzt ausgeführt. Sie können die Metriken von Managed Service für Apache Flink in der Amazon- CloudWatch Konsole überprüfen, um zu überprüfen, ob die Anwendung funktioniert.

Stoppen der Anwendung

In diesem Abschnitt verwenden Sie die StopApplication-Aktion, um die Anwendung zu stoppen.

So stoppen Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `stop_request.json`.

```
{
   "ApplicationName": "test"
}
```

2. Führen Sie die StopApplication-Aktion mit der folgenden Anforderung zum Stoppen der Anwendung aus:

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

Die Anwendung wird nun gestoppt.

Hinzufügen einer CloudWatch Protokollierungsoption

Sie können die verwendenAWS CLI, um Ihrer Anwendung einen Amazon- CloudWatch Protokollstream hinzuzufügen. Informationen zur Verwendung von - CloudWatch Protokollen mit Ihrer Anwendung finden Sie unter the section called “Einrichten der Protokollierung”.

Umgebungseigenschaften aktualisieren

In diesem Abschnitt verwenden Sie die `UpdateApplication`-Aktion, um die Umgebungseigenschaften für die Anwendung zu ändern, ohne den Anwendungscode neu kompilieren zu müssen. In diesem Beispiel ändern Sie die Region der Quell- und Ziel-Streams.

So aktualisieren Sie die Umgebungseigenschaften für die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `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. Führen Sie die UpdateApplication-Aktion mit der vorherigen Anforderung aus, um die Umgebungseigenschaften zu aktualisieren:

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

Aktualisieren Sie den Anwendungscode

Wenn Sie Ihren Anwendungscode mit einer neuen Version Ihres Codepakets aktualisieren müssen, verwenden Sie die UpdateApplication AWS CLI-Aktion.

> **Note**
>
> Um eine neue Version des Anwendungscodes mit demselben Dateinamen zu laden, müssen Sie die neue Objektversion angeben. Weitere Informationen zur Verwendung von Amazon S3-Objektversionen finden Sie unter Versionierung aktivieren oder deaktivieren.

Um die AWS CLI zu verwenden, löschen Sie Ihr vorheriges Codepaket aus Ihrem Amazon S3-Bucket, laden Sie die neue Version hoch und rufen Sie `UpdateApplication` auf. Geben Sie dabei denselben Amazon S3-Bucket und Objektnamen sowie die neue Objektversion an. Die Anwendung wird mit dem neuen Codepaket neu gestartet.

Die folgende Beispielanforderung für die `UpdateApplication`-Aktion lädt den Anwendungscode neu und startet die Anwendung neu. Aktualisieren Sie die `CurrentApplicationVersionId` auf die aktuelle Anwendungsversion. Sie können die aktuelle Anwendungsversion mithilfe der Aktionen `ListApplications` oder `DescribeApplication` überprüfen. Aktualisieren Sie das Bucket-Namenssuffix (*<username>*) mit dem Suffix, das Sie im the section called “Erstellen von zwei Amazon Kinesis Datenstroms”-Abschnitt ausgewählt haben.

```
{
    "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"
                }
            }
        }
    }
}
```

Nächster Schritt

Schritt 4: Bereinigen von AWS-Ressourcen

Schritt 4: Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zum Bereinigen von AWS-Ressourcen, die im Tutorial Erste Schritte erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte:

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Sie Ihre Kinesis Data Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen
- Nächster Schritt

Löschen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen Sie Ihre Kinesis Data Streams

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Bereich Kinesis Data Streams die Option ausExampleInputStream.
3. Wählen Sie auf der ExampleInputStream Seite Kinesis Stream löschen aus und bestätigen Sie dann den Löschvorgang.
4. Wählen Sie auf der Seite Kinesis Streams die , ExampleOutputStreamwählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den ka-app-codeBucket -***<username>*** aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle kinesis-analytics-MyApplication--us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.

4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Nächster Schritt

Schritt 5: Nächste Schritte

Schritt 5: Nächste Schritte

Nachdem Sie nun eine grundlegende Managed Service für Apache Flink-Anwendung erstellt und ausgeführt haben, finden Sie in den folgenden Ressourcen erweiterte Managed Service für Apache Flink-Lösungen.

- Die AWS Streaming Data Solution for Amazon Kinesis : Die AWS Streaming Data Solution for Amazon Kinesis konfiguriert automatisch die AWS-Services, die zum einfachen Erfassen, Speichern, Verarbeiten und Bereitstellen von Streaming-Daten erforderlich sind. Die Lösung bietet mehrere Optionen zur Lösung von Anwendungsfällen mit Streaming-Daten. Die Option Managed Service für Apache Flink bietet ein end-to-end -Streaming-ETL-Beispiel, das eine reale Anwendung demonstriert, die analytische Operationen für simulierte New York-Taxisdaten ausführt. Die Lösung richtet alle erforderlichen AWS Ressourcen wie IAM-Rollen und -Richtlinien, ein CloudWatch Dashboard und CloudWatch Alarme ein.
- AWS Streaming Data Solution for Amazon MSK: Die AWS Streaming Data Solution for Amazon MSK bietet AWS CloudFormation-Vorlagen, in denen Daten durch Produzenten, Streaming-Speicher, Verbraucher und Ziele fließen.
- Clickstream Lab mit Apache Flink und Apache Kafka: Ein End-to-End-Lab für Clickstream-Anwendungsfälle mit Amazon Managed Streaming for Apache Kafka als Streaming-Speicher und Managed Service für Apache Flink für Apache Flink-Anwendungen zur Stream-Verarbeitung.
- Workshop für Amazon Managed Service für Apache Flink: In diesem Workshop erstellen Sie eine - end-to-end Streaming-Architektur, um Streaming-Daten nahezu in Echtzeit aufzunehmen, zu analysieren und zu visualisieren. Sie haben sich vorgenommen, den Betrieb eines Taxiunternehmens in New York City zu verbessern. Sie analysieren die Telemetriedaten einer Taxiflotte in New York City nahezu in Echtzeit, um deren Flottenbetrieb zu optimieren.
- Managed Service für Apache Flink: Beispiele: Dieser Abschnitt dieses Entwicklerhandbuchs enthält Beispiele für die Erstellung von und die Arbeit mit Anwendungen in Managed Service für Apache Flink. Sie enthalten Beispielcode und step-by-step Anweisungen, mit denen Sie Managed Service für Apache Flink-Anwendungen erstellen und Ihre Ergebnisse testen können.

- Lernen Sie Flink kennen: Praktisches Training: Offizielle Apache Flink-Einführungsschulung, die Ihnen den Einstieg in die Entwicklung skalierbarer Streaming-ETL-, Analyse- und ereignisgesteuerter Anwendungen ermöglicht.

> **Note**
>
> Beachten Sie, dass Managed Service für Apache Flink die in dieser Schulung verwendete Apache Flink-Version (1.12) nicht unterstützt. Sie können Flink 1.15.2 in Flink Managed Service für Apache Flink verwenden.

Erste Schritte mit Amazon Managed Service für Apache Flink (Tabellen-API)

In diesem Abschnitt werden Ihnen die grundlegenden Konzepte von Managed Service for Apache Flink und der Tabellen-API vorgestellt. Es werden die verfügbaren Optionen für die Erstellung und das Testen von Anwendungen beschrieben. Er enthält auch Anweisungen zur Installation der Tools, die Sie benötigen, um die Tutorials in diesem Handbuch abzuschließen und Ihre erste Anwendung zu erstellen.

Themen

- Komponenten der Anwendung Managed Service für Apache Flink
- Voraussetzungen
- Erstellen Sie einen Managed Service für Apache Flink-Anwendung und führen Sie ihn aus
- Bereinigen von AWS-Ressourcen
- Nächste Schritte

Komponenten der Anwendung Managed Service für Apache Flink

Zur Verarbeitung von Daten verwendet Ihre Managed Service für Apache Flink-Anwendung eine Java/Apache Maven- oder Scala-Anwendung, die mithilfe der Apache Flink-Laufzeit Eingaben verarbeitet und Ausgaben erzeugt.

Die Anwendung Managed Service für Apache Flink besteht aus folgenden Komponenten:

- Runtime-Eigenschaften: Sie können Runtime-Eigenschaften verwenden, um Ihre Anwendung zu konfigurieren, ohne Ihren Anwendungscode neu kompilieren zu müssen.
- Tabellenquelle: Die Anwendung verwendet Daten mithilfe einer Quelle. Ein Quell-Konnektor liest Daten aus einem Kinesis Data Stream, einem Amazon MSK-Thema oder ähnlichem. Weitere Informationen finden Sie unter Tabellen-API Quellen.
- Funktionen: Die Anwendung verarbeitet Daten mithilfe einer oder mehrerer Funktionen. Eine Funktion kann Daten transformieren, anreichern oder aggregieren.
- Senke: Die Anwendung erzeugt Daten für externe Quellen mithilfe von Senken. Ein Senken-Konnektor schreibt Daten in einen Kinesis Data Stream, einen Kinesis Data Firehose-Stream

von Kinesis Data Firehose, ein Amazon MSK-Thema, einen Amazon S3-Bucket usw. Weitere Informationen finden Sie unter Tabellen-API Senken.

Nachdem Sie den Anwendungscode erstellt, kompiliert und verpackt haben, laden Sie das Codepaket in einen Amazon-S3-Bucket hoch. Anschließend erstellen Sie eine Managed Service für Apache Flink-Anwendung. Sie übergeben den Speicherort des Codepakets, ein Amazon MSK-Thema als Streaming-Datenquelle und in der Regel einen Streaming- oder Dateispeicherort, der die verarbeiteten Daten der Anwendung empfängt.

Voraussetzungen

Bevor Sie mit diesem Tutorial beginnen, führen Sie die ersten beiden Schritte von Erste Schritte mit Amazon Managed Service für Apache Flink (DataStream API) aus:

- Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Administratorbenutzers
- Schritt 2: Einrichten der AWS Command Line Interface (AWS CLI)

Um zu beginnen, sehen Sie sich Erstellen einer Anwendung an.

Erstellen Sie einen Managed Service für Apache Flink-Anwendung und führen Sie ihn aus

In dieser Übung erstellen Sie eine Anwendung von Managed Service für Apache Flink mit einem Amazon MSK-Thema als Quelle und einem Amazon S3-Bucket als Senke.

Dieser Abschnitt enthält die folgenden Schritte.

- Erstellen Sie abhängige Ressourcen
- Schreiben Sie Beispieldatensätze in den Eingabe-Stream
- Herunterladen und Überprüfen des Apache Flink-Streaming-Java-Codes
- Kompilieren des Anwendungscodes
- Hochladen des Apache Flink-Streaming-Java-Codes
- Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus
- Nächster Schritt

Erstellen Sie abhängige Ressourcen

Bevor Sie für diese Übung einen Managed Service für Apache Flink erstellen, erstellen Sie die folgenden abhängigen Ressourcen:

- Eine Virtual Private Cloud (VPC) basierend auf Amazon VPC und einem Amazon-MSK-Cluster
- Ein Amazon S3-Bucket zum Speichern des Codes und der Ausgabe der Anwendung (`ka-app-code-<username>`)

Erstellen Sie eine VPC und einen Amazon-MSK-Cluster

Um einen VPC- und Amazon MSK-Cluster zu erstellen, auf den Sie von Ihrer Managed Service für Apache Flink-Anwendung aus zugreifen können, folgen Sie dem Tutorial Erste Schritte mit Amazon MSK.

Beachten Sie beim Abschluss des Tutorials Folgendes:

- Notieren Sie sich die Bootstrap-Serverliste für Ihren Cluster. Sie können die Liste der Bootstrap-Server mit dem folgenden Befehl abrufen und *ClusterArn* durch den Amazon-Ressourcennamen (ARN) für Ihren MSK-Cluster ersetzen:

```
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"
}
```

- Wenn Sie den Schritten in den Tutorials folgen, achten Sie darauf, dass Sie Ihre gewählte AWS-Region in Ihrem Code, Ihren Befehlen und Konsoleneinträgen verwenden.

Erstellen eines Amazon-S3-Buckets

Sie können ein Amazon-S3-Bucket mithilfe der Konsole erstellen. Anweisungen zum Erstellen dieser Ressource finden Sie in den folgenden Themen:

- Wie erstelle ich einen S3-Bucket? im Amazon Simple Storage Service-Benutzerhandbuch. Geben Sie dem Amazon S3-Bucket einen global eindeutigen Namen, indem Sie Ihren Anmeldenamen anhängen, z. B. **ka-app-code-*<username>***.

Sonstige Ressourcen

Wenn Sie Ihre Anwendung erstellen, erstellt Managed Service für Apache Flink die folgenden Amazon- CloudWatch Ressourcen, sofern sie noch nicht vorhanden sind:

- Eine Protokollgruppe namens `/AWS/KinesisAnalytics-java/MyApplication`.
- Einen Protokollstream mit dem Namen `kinesis-analytics-log-stream`.

Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Abschnitt verwenden Sie ein Python-Skript zum Schreiben von Beispiel-Datensätzen in das Amazon MSK-Thema für die zu verarbeitende Anwendung.

1. Stellen Sie eine Verbindung zu der Client-Instance her, die Sie in Schritt 4: Einen Client-Computer erstellen des Tutorials Erste Schritte mit Amazon MSK erstellt haben.
2. Installieren Sie Python3, Pip und die Kafka-Python-Bibliothek:

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

3. Erstellen Sie eine Datei mit dem Namen `stock.py` und dem folgenden Inhalt. Ersetzen Sie den `BROKERS`-Wert durch Ihre Bootstrap-Broker-Liste, die Sie zuvor aufgezeichnet haben.

```
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. Im weiteren Verlauf des Tutorials führen Sie das `stock.py`-Skript zum Senden von Daten an die Anwendung aus.

```
$ python3 stock.py
```

Herunterladen und Überprüfen des Apache Flink-Streaming-Java-Codes

Der Java-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub.

So laden Sie den Java-Anwendungscode herunter

1. Klonen Sie das Remote-Repository, indem Sie den folgenden Befehl verwenden:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/GettingStartedTable` Verzeichnis .

Beachten Sie Folgendes zum Anwendungscode:

- Eine Project Object Model (pom.xml) Datei enthält Informationen über die Konfiguration und Abhängigkeiten der Anwendung, einschließlich der Bibliotheken des Managed Service für Apache Flink.
- Die `StreamingJob.java`-Datei enthält die `main`-Methode, die die Funktionalität der Anwendung definiert.
- Die Anwendung verwendet `FlinkKafkaConsumer` zum Lesen aus dem Amazon-MSK-Thema. Der folgende Codeausschnitt erstellt ein `FlinkKafkaConsumer`-Objekt:

```
final FlinkKafkaConsumer<StockRecord> consumer = new
 FlinkKafkaConsumer<StockRecord>(kafkaTopic, new KafkaEventDeserializationSchema(),
 kafkaProps);
```

- Ihre Anwendung erstellt Quell- und Senken-Konnektoren, um mithilfe von `StreamExecutionEnvironment`- und `TableEnvironment`-Objekten auf externe Ressourcen zuzugreifen.
- Die Anwendung erstellt Quell- und Senken-Konnektoren mithilfe dynamischer Anwendungseigenschaften, sodass Sie Ihre Anwendungsparameter (z. B. Ihren S3-Bucket) angeben können, ohne den Code neu kompilieren zu müssen.

```
//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();
}
```

Weitere Informationen zu Laufzeiteigenschaften finden Sie unter Laufzeiteigenschaften.

> **Note**
>
> Beim Erstellen Ihrer Anwendung empfehlen wir dringend, die Anwendung Managed Service für Apache Flink in derselben Region wie den Amazon MSK-Cluster zu erstellen und auszuführen. Das liegt daran, dass der Flink Kafka-Konnektor standardmäßig für Umgebungen mit niedriger Latenz optimiert ist. Wenn Sie von einem regionsübergreifenden Kafka-Cluster aus konsumieren müssen, sollten Sie erwägen, den Konfigurationswert für `receive.buffer.byte` zu erhöhen, z. B. 2097152.
> Weitere Informationen finden Sie unter Benutzerdefinierte MSK-Konfigurationen.

Kompilieren des Anwendungscodes

In diesem Abschnitt verwenden Sie den Apache Maven-Compiler zum Erstellen des Java-Codes für die Anwendung. Weitere Informationen zum Installieren von Apache Maven und des Java Development Kit (JDK) finden Sie unter Voraussetzungen für das Fertigstellen der Übungen.

Kompilieren des Anwendungscodes

1. Zum Verwenden Ihres Anwendungscodes kompilieren und packen Sie ihn in eine JAR-Datei. Sie können Ihren Code auf zwei Arten kompilieren und packen:

   - Verwenden Sie das Befehlszeilen-Maven-Tool. Erstellen Sie Ihre JAR-Datei, indem Sie den folgenden Befehl in dem Verzeichnis ausführen, das die `pom.xml`-Datei enthält:

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

   - Verwenden Sie Ihre Entwicklungsumgebung. Weitere Informationen finden Sie in der Dokumentation Ihrer Entwicklungsumgebung.

     > **Note**
     >
     > Der bereitgestellte Quellcode basiert auf Bibliotheken aus Java 11.

   Sie können Ihr Paket als JAR-Datei hochladen oder komprimieren und als ZIP-Datei hochladen. Wenn Sie Ihre Anwendung mit der AWS CLI erstellen, geben Sie Ihren Codeinhaltstyp (JAR oder ZIP) an.

2. Wenn während der Erstellung Fehler aufgetreten sind, überprüfen Sie, ob Ihre `JAVA_HOME`-Umgebungsvariable richtig eingestellt ist.

Wenn die Anwendung erfolgreich kompiliert wurde, wird die folgende Datei erstellt:

`target/aws-kinesis-analytics-java-apps-1.0.jar`

Hochladen des Apache Flink-Streaming-Java-Codes

In diesem Abschnitt erstellen Sie einen Amazon S3-Bucket und laden Ihren Anwendungscode hoch.

So laden Sie den Anwendungscode hoch

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie Bucket erstellen aus.
3. Geben Sie **`ka-app-code-<username>`** im Feld Bucket-Name ein. Fügen Sie dem Bucket-Namen ein Suffix hinzu, wie z. B. Ihren Benutzernamen, damit er global eindeutig ist. Wählen Sie Weiter aus.
4. Lassen Sie im Schritt Optionen konfigurieren die Einstellungen unverändert und klicken Sie auf Weiter.
5. Lassen Sie im Schritt Berechtigungen festlegen die Einstellungen unverändert und klicken Sie auf Weiter.
6. Wählen Sie Bucket erstellen aus.
7. Wählen Sie in der Amazon S3-Konsole den Bucket ka-app-code-**`<username>`** und dann Hochladen aus.
8. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `aws-kinesis-analytics-java-apps-1.0.jar`Datei, die Sie im vorherigen Schritt erstellt haben. Wählen Sie Weiter aus.
9. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Geben Sie für Beschreibung den Text **`My java test app`** ein.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.
   - Belassen Sie die Version als Apache Flink Version 1.15.2 (empfohlene Version).
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rolle: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf den Amazon S3-Bucket.

Um die IAM-Richtlinie zu bearbeiten, um S3-Bucket-Berechtigungen hinzuzufügen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **`kinesis-analytics-service-MyApplication-us-west-2`**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu.

```
{
    "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"
            ]
        }
    ]
}
```

Konfigurieren der Anwendung

Verwenden Sie das folgende Verfahren, um die Anwendung zu konfigurieren.

Zum Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):
   - Geben Sie für Amazon-S3-Bucket **ka-app-code-*<username>*** ein.

- Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **aws-kinesis-analytics-java-apps-1.0.jar** ein.

3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **kinesis-analytics-MyApplication-us-west-2** erstellen/aktualisieren aus.
4. Wählen Sie unter Eigenschaften die Option Gruppe erstellen aus.
5. Geben Sie Folgendes ein:

Gruppen-ID	Schlüssel	Wert
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/lib/security/cacerts**
FlinkApplicationProperties	**ssl.truststore.password**	**changeit**

6. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
7. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
8. Wählen Sie im Abschnitt Virtual Private Cloud (VPC) VPC-Konfiguration basierend auf Amazon MSK-Cluster aus. Wählen Sie AWSKafkaTutorialCluster.
9. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die Amazon- CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`

Ausführen der Anwendung

Gehen Sie wie folgt vor, um die Anwendung auszuführen.

Ausführen der Anwendung

1. Wählen Sie auf der MyApplication Seite Ausführen aus. Bestätigen Sie die Aktion.
2. Wenn die Anwendung ausgeführt wird, aktualisieren Sie die Seite. Die Konsole zeigt den Application graph (Anwendungs-Graph) an.
3. Führen Sie von Ihrem Amazon EC2-Client aus das Python-Skript aus, das Sie zuvor erstellt haben, um Datensätze in den Amazon MSK-Cluster zu schreiben, damit Ihre Anwendung sie verarbeiten kann:

   ```
   $ python3 stock.py
   ```

Stoppen der Anwendung

Um die Anwendung zu beenden, wählen Sie auf der MyApplication Seite Stoppen aus. Bestätigen Sie die Aktion.

Nächster Schritt

Bereinigen von AWS-Ressourcen

Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zum Bereinigen von AWS-Ressourcen, die im Tutorial Erste Schritte (Tabellen-API) erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte.

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Sie Ihren Amazon-MSK-Cluster
- Löschen der VPC
- Löschen Sie Ihre Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen
- Nächster Schritt

Löschen Sie die Anwendung Managed Service für Apache Flink

Gehen Sie wie folgt vor, um die Anwendung zu löschen.

So löschen Sie die Anwendung:

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie auf der Anwendungsseite Löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen Sie Ihren Amazon-MSK-Cluster

Um Ihren Amazon MSK-Cluster zu löschen, folgen Sie Schritt 8: Löschen des Amazon MSK-Clusters im Amazon Managed Streaming for Apache Kafka Developer Guide.

Löschen der VPC

Um Ihre Amazon-VPC zu löschen, führen Sie die folgenden Schritte aus:

- Öffnen Sie die Amazon VPC-Konsole.
- Wählen Sie Ihre VPCs aus.
- Wählen Sie unter Actions (Aktionen) die Option Delete VPC (VPC löschen) aus.

Löschen Sie Ihre Amazon-S3-Objekten und -Buckets

Gehen Sie wie folgt vor, um die S3-Objekte und den Bucket zu löschen.

Löschen Ihrer Objekte und Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den Bucket ka-app-code-***<username>*** aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

Führen Sie die folgenden Schritte aus, um Ihre IAM-Ressourcen zu löschen.

Um Ihre IAM-Ressourcen zu löschen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle kinesis-analytics--MyApplication-us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch Ressourcen

Gehen Sie wie folgt vor, um Ihre - CloudWatch Ressourcen zu löschen.

So löschen Sie Ihre CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Nächster Schritt

Nächste Schritte

Nächste Schritte

Nachdem Sie nun eine Managed Service für Apache Flink-Anwendung erstellt und ausgeführt haben, die die Tabellen-API verwendet, finden Sie Schritt 5: Nächste Schritte in Erste Schritte mit Amazon Managed Service für Apache Flink (DataStream API).

Erste Schritte mit Amazon Managed Service für Apache Flink für Python

In diesem Abschnitt werden Sie mit den grundlegenden Konzepten eines Managed Service für Apache Flink unter Verwendung von Python und der Tabellen-API vertraut gemacht. Es werden die verfügbaren Optionen für die Erstellung und das Testen von Anwendungen beschrieben. Er enthält auch Anweisungen zur Installation der Tools, die Sie benötigen, um die Tutorials in diesem Handbuch abzuschließen und Ihre erste Anwendung zu erstellen.

Themen

- Erste Schritte mit Pyflink – Der Python-Interpreter für Apache | Amazon Web Services
- Komponenten der Anwendung Managed Service für Apache Flink
- Voraussetzungen
- Erstellen und Ausführen eines Managed Service für Apache Flink für die Python-Anwendung
- Bereinigen von AWS-Ressourcen

> **Note**
>
> Wenn Sie Python Flink-Anwendung auf einem neuen Mac mit Apple Silicon-Chip entwickeln, können einige bekannte Probleme mit Python-Abhängigkeiten von PyFlink 1.15 auftreten. In diesem Fall empfehlen wir, den Python-Interpreter in Docker auszuführen. step-by-step Anweisungen finden Sie unter PyFlink 1.15-Entwicklung auf Apple Silicon Mac.

Erste Schritte mit Pyflink – Der Python-Interpreter für Apache | Amazon Web Services

Bevor Sie beginnen, empfehlen wir Ihnen, sich das folgende Video anzusehen:

Erste Schritte mit Pyflink – Der Python-Interpreter für Apache | Amazon Web Services

Komponenten der Anwendung Managed Service für Apache Flink

Um Daten zu verarbeiten, verwendet Ihre Managed Service for Apache Flink-Anwendung eine Python-Anwendung, die mithilfe der Apache Flink-Laufzeit Eingaben verarbeitet und Ausgaben erzeugt.

Die Anwendung Managed Service für Apache Flink besteht aus den folgenden Komponenten:

- Runtime-Eigenschaften: Sie können Runtime-Eigenschaften verwenden, um Ihre Anwendung zu konfigurieren, ohne Ihren Anwendungscode neu kompilieren zu müssen.
- Tabellenquelle: Die Anwendung verwendet Daten mithilfe einer Quelle. Ein Quell-Konnektor liest Daten aus einem Kinesis Data Stream, einem Amazon MSK-Thema oder ähnlichem. Weitere Informationen finden Sie unter Tabellen-API Quellen.
- Funktionen: Die Anwendung verarbeitet Daten mithilfe einer oder mehrerer Funktionen. Eine Funktion kann Daten transformieren, anreichern oder aggregieren.
- Senke: Die Anwendung erzeugt Daten für externe Quellen mithilfe von Senken. Ein Senken-Konnektor schreibt Daten in einen Kinesis Data Stream, einen Kinesis Data Firehose-Stream von Kinesis Data Firehose, ein Amazon MSK-Thema, einen Amazon S3-Bucket usw. Weitere Informationen finden Sie unter Tabellen-API Senken.

Nachdem Sie den Anwendungscode erstellt und verpackt haben, laden Sie das Codepaket in einen Amazon-S3-Bucket hoch. Anschließend erstellen Sie eine Managed Service für Apache Flink-Anwendung. Sie übergeben den Speicherort des Codepakets, eine Streaming-Datenquelle und in der Regel einen Streaming- oder Dateispeicherort, der die verarbeiteten Daten der Anwendung empfängt.

Voraussetzungen

Bevor Sie dieses Tutorial starten, führen Sie die ersten zwei Schritte von Erste Schritte mit Amazon Managed Service für Apache Flink (DataStream API) aus.

- Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Administratorbenutzers
- Schritt 2: Einrichten der AWS Command Line Interface (AWS CLI)

Informationen zum Einstieg finden Sie unter Erstellen einer Anwendung.

Erstellen und Ausführen eines Managed Service für Apache Flink für die Python-Anwendung

In dieser Übung erstellen Sie eine Anwendung von Managed Service für Apache Flink für Python-Anwendung mit einem Kinesis Stream als Quelle und Senke.

Dieser Abschnitt enthält die folgenden Schritte.

- Erstellen Sie abhängige Ressourcen
- Schreiben Sie Beispieldatensätze in den Eingabe-Stream
- Erstellen und Überprüfen des Apache Flink-Streaming-Python-Codes
- Hinzufügen von Abhängigkeiten von Drittanbietern zu Python-Apps
- Hochladen des Apache Flink-Streaming-Python-Codes
- Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus
- Nächster Schritt

Erstellen Sie abhängige Ressourcen

Bevor Sie für diese Übung einen Managed Service für Apache Flink erstellen, erstellen Sie die folgenden abhängigen Ressourcen:

- Zwei Kinesis Streams für Eingaben und Ausgaben.
- Einen Amazon S3-Bucket zum Speichern des Codes und der Ausgabe der Anwendung (`ka-app-code-<username>`)

Zwei Kinesis Streams erstellen

Bevor Sie für diese Übung eine Anwendung von Managed Service für Apache Flink erstellen, erstellen Sie zwei Kinesis Data Streams (`ExampleInputStream` und `ExampleOutputStream`). Ihre Anwendung verwendet diese Streams für die Quell- und Ziel-Streams der Anwendung.

Sie können diese Streams mithilfe der Amazon-Kinesis-Konsole oder des folgenden AWS CLI-Befehls erstellen. Anweisungen für die Konsole finden Sie unter Erstellen und Aktualisieren von Datenströmen im Amazon Kinesis Data Streams Entwicklerhandbuch.

So erstellen Sie die Daten-Streams (AWS CLI)

1. Verwenden Sie den Befehl Amazon Kinesis `create-stream` AWS CLI, um den ersten Stream (`ExampleInputStream`) zu erstellen.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Um den zweiten Stream zu erstellen, den die Anwendung zum Schreiben der Ausgabe verwendet, führen Sie denselben Befehl aus und ändern den Stream-Namen in `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Erstellen eines Amazon-S3-Buckets

Sie können ein Amazon-S3-Bucket mithilfe der Konsole erstellen. Anweisungen zum Erstellen dieser Ressource finden Sie in den folgenden Themen:

- Wie erstelle ich einen S3-Bucket? im Amazon Simple Storage Service-Benutzerhandbuch. Geben Sie dem Amazon S3-Bucket einen global eindeutigen Namen, indem Sie Ihren Anmeldenamen anhängen, z. B. **`ka-app-code-<username>`**.

Sonstige Ressourcen

Wenn Sie Ihre Anwendung erstellen, erstellt Managed Service für Apache Flink die folgenden Amazon- CloudWatch Ressourcen, sofern sie noch nicht vorhanden sind:

- Eine Protokollgruppe namens `/AWS/KinesisAnalytics-java/MyApplication`.
- Einen Protokollstream mit dem Namen `kinesis-analytics-log-stream`.

Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Abschnitt verwenden Sie ein Python-Skript zum Schreiben von Datensätzen in den Stream für die zu verarbeitende Anwendung.

> **Note**
>
> Dieser Abschnitt erfordert AWS SDK for Python (Boto).

> **Note**
>
> Das Python-Skript in diesem Abschnitt verwendet die AWS CLI. Sie müssen Ihre AWS CLI so konfigurieren, dass sie Ihre Kontoanmeldeinformationen und Ihre Standardregion verwendet. Geben Sie zum Konfigurieren der AWS CLI Folgendes ein:
>
> ```
> aws configure
> ```

1. Erstellen Sie eine Datei `stock.py` mit dem folgenden Inhalt:

```
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. Führen Sie das `stock.py`Skript aus:

```
$ python stock.py
```

   Lassen Sie das Skript laufen, während Sie den Rest des Tutorials abschließen.

Erstellen und Überprüfen des Apache Flink-Streaming-Python-Codes

Der Python-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben. Weitere Informationen finden Sie unter Git installieren.
2. Klonen Sie das Remote-Repository mit dem folgenden Befehl:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/python/GettingStarted` Verzeichnis .

Der Anwendungscode befindet sich in der `getting_started.py`-Datei. Beachten Sie Folgendes zum Anwendungscode:

- Die Anwendung verwendet eine Kinesis-Tabellenquelle zum Lesen aus dem Quell-Stream. Der folgende Ausschnitt ruft die `create_table`-Funktion zum Erstellen der Kinesis-Tabellenquelle auf:

```
table_env.execute_sql(
        create_table(output_table_name, output_stream, output_region)
```

Die `create_table`-Funktion verwendet einen SQL-Befehl, um eine Tabelle zu erstellen, die von der Streaming-Quelle unterstützt wird:

```
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)
 }
```

- Die Anwendung erstellt zwei Tabellen und schreibt dann den Inhalt einer Tabelle in die andere.

```
    # 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))
```

- Die Anwendung verwendet den Flink-Konnektor aus der Datei flink- sql-connector-kinesis_2.12/1.15.2.

Hinzufügen von Abhängigkeiten von Drittanbietern zu Python-Apps

Wenn Sie Python-Pakete von Drittanbietern (wie boto3) verwenden, müssen Sie deren transitive Abhängigkeiten und die Eigenschaften hinzufügen, die für diese Abhängigkeiten erforderlich sind. Allgemein können Sie für PyPi Abhängigkeiten die Dateien und Ordner kopieren, die sich in Ihrem Python-`site-packages`Umgebungsordner befinden, um eine Verzeichnisstruktur wie die folgende zu erstellen:

```
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
#    # ...
...
```

Um den boto3 als Drittanbieter-Abhängigkeit hinzuzufügen:

1. Erstellen Sie eine eigenständige Python-Umgebung (conda oder ähnliches) auf Ihrem lokalen Computer mit den erforderlichen Abhängigkeiten.
2. Notieren Sie sich die anfängliche Liste der Pakete im `site_packages`-Ordner dieser Umgebung.
3. `pip-install` alle erforderlichen Abhängigkeiten für Ihre Anwendung.
4. Notieren Sie sich die Pakete, die dem `site_packages`-Ordner nach Schritt 3 oben hinzugefügt wurden. Dies sind die Ordner, die Sie in Ihr Paket (unter dem `my_deps`-Ordner) aufnehmen

müssen. Sie sind wie oben dargestellt organisiert. Auf diese Weise können Sie einen Unterschied zwischen den Paketen zwischen den Schritten 2 und 3 erfassen, um die richtigen Paketabhängigkeiten für Ihre Anwendung zu ermitteln.

5. Geben Sie `my_deps/` als Argument für die `pyFiles`-Eigenschaft in der `kinesis.analytics.flink.run.options`-Eigenschaftengruppe an, wie unten für die `jarfiles`-Eigenschaft beschrieben. Mit Flink können Sie auch Python-Abhängigkeiten mit der Funktion add_python_file angeben. Beachten Sie jedoch, dass Sie nur die eine oder die andere angeben müssen – nicht beide.

> **Note**
>
> Sie müssen den Ordner `my_deps` nicht benennen. Der wichtige Teil besteht darin, die Abhängigkeiten entweder mit `pyFiles` oder `add_python_file` zu registrieren. Ein Beispiel finden Sie unter So verwenden Sie boto3 in pyFlink.

Hochladen des Apache Flink-Streaming-Python-Codes

In diesem Abschnitt erstellen Sie einen Amazon S3-Bucket und laden Ihren Anwendungscode hoch.

Um den Anwendungscode über die Konsole hochzuladen:

1. Verwenden Sie Ihre bevorzugte Komprimierungsanwendung, um die Dateien `getting-started.py` und https://mvnrepository.com/artifact/org.apache.flink/flink- sql-connector-kinesis_2.12/1.15.2 zu komprimieren. Benennen Sie das Archiv `myapp.zip`. Wenn Sie den äußeren Ordner in Ihr Archiv aufnehmen, müssen Sie dies in den Pfad mit dem Code in Ihrer/Ihren Konfigurationsdatei(en) aufnehmen: `GettingStarted/getting-started.py`.
2. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
3. Wählen Sie Bucket erstellen aus.
4. Geben Sie **`ka-app-code-<username>`** im Feld Bucket-Name ein. Fügen Sie dem Bucket-Namen ein Suffix hinzu, wie z. B. Ihren Benutzernamen, damit er global eindeutig ist. Wählen Sie Weiter aus.
5. Lassen Sie im Schritt Optionen konfigurieren die Einstellungen unverändert und klicken Sie auf Weiter.
6. Lassen Sie im Schritt Berechtigungen festlegen die Einstellungen unverändert und klicken Sie auf Weiter.

7. Wählen Sie Bucket erstellen aus.
8. Wählen Sie in der Amazon S3-Konsole den Bucket ka-app-code-*<username>* und dann Upload aus.
9. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `myapp.zip`Datei, die Sie im vorherigen Schritt erstellt haben. Wählen Sie Weiter aus.
10. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

So laden Sie den Anwendungscode hoch mit AWS CLI:

> **Note**
>
> Verwenden Sie nicht die Komprimierungsfunktionen im Finder (macOS) oder Windows Explorer (Windows), um das `myapp.zip`-Archiv zu erstellen. Dies kann zu einem ungültigen Anwendungscode führen.

1. Verwenden Sie Ihre bevorzugte Komprimierungsanwendung, um die Dateien `streaming-file-sink.py` und https://mvnrepository.com/artifact/org.apache.flink/flink- sql-connector-kinesis_2.12/1.15.2 zu komprimieren.

   > **Note**
   >
   > Verwenden Sie nicht die Komprimierungsfunktionen im Finder (macOS) oder Windows Explorer (Windows), um das Archiv myapp.zip zu erstellen. Dies kann zu einem ungültigen Anwendungscode führen.

2. Verwenden Sie Ihre bevorzugte Komprimierungsanwendung, um die Dateien `getting-started.py` und https://mvnrepository.com/artifact/org.apache.flink/flink-sql-connector-kinesis/1.15.2 zu komprimieren. Benennen Sie das Archiv `myapp.zip`. Wenn Sie den äußeren Ordner in Ihr Archiv aufnehmen, müssen Sie dies in den Pfad mit dem Code in Ihrer/Ihren Konfigurationsdatei(en) aufnehmen: `GettingStarted/getting-started.py`.
3. Führen Sie den folgenden Befehl aus:

   ```
   $ aws s3 --region aws region cp myapp.zip s3://ka-app-code-<username>
   ```

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Geben Sie für Beschreibung den Text **`My java test app`** ein.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.
   - Belassen Sie die Version als Apache Flink Version 1.15.2 (empfohlene Version).
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rolle: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Konfigurieren der Anwendung

Verwenden Sie das folgende Verfahren, um die Anwendung zu konfigurieren.

Zum Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):
   - Geben Sie für Amazon-S3-Bucket **ka-app-code-*<username>*** ein.
   - Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **myapp.zip** ein.
3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **kinesis-analytics-MyApplication-us-west-2** erstellen/aktualisieren aus.
4. Wählen Sie unter Eigenschaften die Option Gruppe hinzufügen aus.
5. Geben Sie Folgendes ein:

Gruppen-ID	Schlüssel	Wert
consumer.config.0	**input.stream.name**	**ExampleInputStream**
consumer.config.0	**aws.region**	**us-west-2**
consumer.config.0	**scan.stream.initpos**	**LATEST**

   Wählen Sie Speichern.
6. Wählen Sie unter Eigenschaften erneut Gruppe hinzufügen aus.
7. Geben Sie Folgendes ein:

Gruppen-ID	Schlüssel	Wert
producer.config.0	**output.stream.name**	**ExampleOutputStream**
producer.config.0	**aws.region**	**us-west-2**
producer.config.0	**shard.count**	**1**

8. Wählen Sie unter Eigenschaften erneut Gruppe hinzufügen aus. Geben Sie für Gruppen-ID **`kinesis.analytics.flink.run.options`** ein. Diese spezielle Eigenschaftsgruppe teilt Ihrer Anwendung mit, wo sich ihre Coderessourcen befinden. Weitere Informationen finden Sie unter Angeben Ihrer Codedateien.
9. Geben Sie Folgendes ein:

Gruppen-ID	Schlüssel	Wert
`kinesis.analytics.flink.run.options`	**`python`**	**`getting-started.py`**
`kinesis.analytics.flink.run.options`	**`jarfile`**	**`flink-sql-connector-kinesis-1.15.2.jar`**

10. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
11. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
12. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die Amazon- CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf den Amazon S3-Bucket.

Um die IAM-Richtlinie zu bearbeiten, um S3-Bucket-Berechtigungen hinzuzufügen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **`kinesis-analytics-service-MyApplication-us-west-2`**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*012345678901*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

Ausführen der Anwendung

Das Flink-Jobdiagramm kann angezeigt werden, indem Sie die Anwendung ausführen, das Apache Flink-Dashboard öffnen und den gewünschten Flink-Job auswählen.

Stoppen der Anwendung

Um die Anwendung zu beenden, wählen Sie auf der MyApplication Seite Stoppen aus. Bestätigen Sie die Aktion.

Nächster Schritt

Bereinigen von AWS-Ressourcen

Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zum Bereinigen von AWS-Ressourcen, die im Tutorial Erste Schritte (Python) erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte.

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Sie die Kinesis Data Streams
- Löschen Sie Ihre Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen

Löschen Sie die Anwendung Managed Service für Apache Flink

Gehen Sie wie folgt vor, um die Anwendung zu löschen.

So löschen Sie die Anwendung:

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie auf der Anwendungsseite Löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen Sie die Kinesis Data Streams

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Bereich Kinesis Data Streams die Option ausExampleInputStream.
3. Wählen Sie auf der ExampleInputStream Seite Kinesis Stream löschen aus und bestätigen Sie dann den Löschvorgang.
4. Wählen Sie auf der Seite Kinesis Streams die , ExampleOutputStreamwählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann den Löschvorgang.

Löschen Sie Ihre Amazon-S3-Objekten und -Buckets

Gehen Sie wie folgt vor, um die S3-Objekte und den Bucket zu löschen.

Löschen Ihrer Objekte und Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den Bucket ka-app-code-***<username>*** aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

Führen Sie die folgenden Schritte aus, um Ihre IAM-Ressourcen zu löschen.

Um Ihre IAM-Ressourcen zu löschen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle kinesis-analytics--MyApplication-us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch Ressourcen

Gehen Sie wie folgt vor, um Ihre - CloudWatch Ressourcen zu löschen.

So löschen Sie Ihre CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Erste Schritte (Scala)

> **Note**
>
> Ab Version 1.15 ist Flink Scala-frei. Anwendungen können jetzt die Java-API von jeder Scala-Version aus verwenden. Flink verwendet Scala intern immer noch in einigen Schlüsselkomponenten, macht Scala jedoch nicht im Benutzercode-Classloader verfügbar. Aus diesem Grund müssen Benutzer Scala-Abhängigkeiten zu ihren Jar-Archiven hinzufügen. Weitere Informationen zu den Scala-Änderungen in Flink 1.15 finden Sie unter Scala kostenlos in One Fifteen.

In dieser Übung erstellen Sie eine Anwendung von Managed Service für Apache Flink-Anwendung für Scala mit einem Kinesis Stream als Quelle und Senke.

Dieses Thema enthält die folgenden Abschnitte:

- Erstellen Sie abhängige Ressourcen
- Schreiben von Beispieldatensätze in den Eingabe-Stream
- Laden Sie den Anwendungscode herunter und überprüfen Sie ihn
- Kompilieren Sie den Anwendungscode und laden Sie ihn hoch
- Erstellen und Ausführen der Anwendung (Konsole)
- Erstellen und Ausführen der Anwendung (CLI)
- Bereinigen von AWS-Ressourcen

Erstellen Sie abhängige Ressourcen

Bevor Sie für diese Übung eine Anwendung von Managed Service für Apache Flink erstellen, erstellen Sie die folgenden abhängigen Ressourcen:

- Zwei Kinesis Streams für Eingaben und Ausgaben.
- Einen Amazon S3-Bucket zum Speichern des Anwendungscodes (`ka-app-code-<username>`)

Sie können die Kinesis Streams und Amazon S3-Bucket mithilfe der Konsole erstellen. Anweisungen zum Erstellen dieser Ressourcen finden Sie in den folgenden Themen:

- Erstellen und Aktualisieren von Datenströmen im Amazon Kinesis Data Streams Developer Guide. Benennen Sie Ihre Data Streams **`ExampleInputStream`** und **`ExampleOutputStream`**.

  So erstellen Sie die Daten-Streams (AWS CLI)

  - Verwenden Sie den folgenden Amazon Kinesis create-stream AWS CLI-Befehl, um den ersten Stream (`ExampleInputStream`) zu erstellen.

    ```
    aws kinesis create-stream \
        --stream-name ExampleInputStream \
        --shard-count 1 \
        --region us-west-2 \
        --profile adminuser
    ```

  - Um den zweiten Stream zu erstellen, den die Anwendung zum Schreiben der Ausgabe verwendet, führen Sie denselben Befehl aus und ändern den Stream-Namen in `ExampleOutputStream`.

    ```
    aws kinesis create-stream \
        --stream-name ExampleOutputStream \
        --shard-count 1 \
        --region us-west-2 \
        --profile adminuser
    ```

- Wie erstelle ich einen S3-Bucket? im Amazon Simple Storage Service Benutzerhandbuch. Geben Sie dem Amazon S3-Bucket einen global eindeutigen Namen, indem Sie Ihren Anmeldenamen anhängen, z. B. **`ka-app-code-<username>`**.

Sonstige Ressourcen

Wenn Sie Ihre Anwendung erstellen, erstellt Managed Service für Apache Flink die folgenden Amazon CloudWatch-Ressourcen, falls sie noch nicht vorhanden sind:

- Eine Protokollgruppe mit dem Namen `/AWS/KinesisAnalytics-java/MyApplication`
- Einen Protokollstream mit dem Namen `kinesis-analytics-log-stream`

Schreiben von Beispieldatensätze in den Eingabe-Stream

In diesem Abschnitt verwenden Sie ein Python-Skript zum Schreiben von Datensätzen in den Stream für die zu verarbeitende Anwendung.

> **Note**
>
> Dieser Abschnitt erfordert AWS SDK for Python (Boto).

> **Note**
>
> Das Python-Skript in diesem Abschnitt verwendet die AWS CLI. Sie müssen Ihren AWS CLI so konfigurieren, dass er Ihre Kontoanmeldeinformationen und Ihre Standardregion verwendet. Geben Sie zum Konfigurieren der AWS CLI Folgendes ein:
>
> ```
> aws configure
> ```

1. Erstellen Sie eine Datei mit dem Namen `stock.py` und dem folgenden Inhalt:

```
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. Führen Sie das `stock.py`-Skript aus:

```
$ python stock.py
```

Lassen Sie das Skript laufen, während Sie den Rest des Tutorials abschließen.

Laden Sie den Anwendungscode herunter und überprüfen Sie ihn

Der Python-Anwendungscode für dieses Beispiel ist auf GitHub verfügbar. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben. Weitere Informationen finden Sie unter Git installieren.
2. Klonen Sie das Remote-Repository, indem Sie den folgenden Befehl verwenden:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navigieren Sie zum Verzeichnis `amazon-kinesis-data-analytics-java-examples/scala/GettingStarted`.

Beachten Sie Folgendes im Zusammenhang mit dem Anwendungscode:

- Eine `build.sbt`-Datei enthält Informationen über die Konfiguration und Abhängigkeiten der Anwendung, einschließlich der Bibliotheken des Managed Service für Apache Flink.
- Die `BasicStreamingJob.scala`-Datei enthält die Hauptmethode, die die Funktionalität der Anwendung definiert.
- Die Anwendung verwendet eine Kinesis-Quelle zum Lesen aus dem Quell-Stream. Der folgende Codeausschnitt erstellt die Kinesis-Quelle:

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
 defaultInputStreamName),
```

```
    new SimpleStringSchema, inputProperties)
}
```

Die Anwendung verwendet auch eine Kinesis-Senke, um in den Ergebnisstream zu schreiben. Der folgende Codeausschnitt erstellt die Kinesis-Senke:

```
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
}
```

- Die Anwendung erstellt Quell- und Senkenkonnektoren für den Zugriff auf externe Ressourcen, indem ein StreamExecutionEnvironment-Objekt verwendet wird.
- Die Anwendung erstellt Quell- und Senkenkonnektoren mit dynamischen Anwendungseigenschaften. Die Laufzeiteigenschaften der Anwendung werden gelesen, um die Konnektoren zu konfigurieren. Weitere Informationen zu Laufzeiteigenschaften finden Sie unter Laufzeiteigenschaften.

Kompilieren Sie den Anwendungscode und laden Sie ihn hoch

In diesem Abschnitt kompilieren Sie Ihren Anwendungscode und laden ihn in den Amazon S3-Bucket hoch, den Sie im Abschnitt Erstellen Sie abhängige Ressourcen erstellt haben.

Kompilieren des Anwendungscodes

In diesem Abschnitt verwenden Sie das SBT-Build-Tool, um den Scala-Code für die Anwendung zu erstellen. Informationen zur Installation von SBT finden Sie unter Installieren von SBT mit CS-Setup. Sie müssen auch das Java Development Kits (JDK) installieren. Siehe Voraussetzungen für die Absolvierung der Übungen.

1. Zum Verwenden Ihres Anwendungscodes kompilieren und packen Sie ihn in eine JAR-Datei. Sie können Ihren Code mit SBT kompilieren und verpacken:

```
sbt assembly
```

2. Wenn die Anwendung erfolgreich kompiliert wurde, wird die folgende Datei erstellt:

```
target/scala-3.2.0/getting-started-scala-1.0.jar
```

Hochladen des Apache Flink-Streaming-Java-Codes

In diesem Abschnitt erstellen Sie einen Amazon S3-Bucket und laden Ihren Anwendungscode hoch.

1. Öffnen Sie die Amazon S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie Bucket erstellen aus
3. Geben Sie `ka-app-code-<username>` im Feld Bucket-Name ein. Fügen Sie dem Bucket-Namen ein Suffix hinzu, wie z. B. Ihren Benutzernamen, damit er global eindeutig ist. Wählen Sie Next (Weiter).
4. Lassen Sie im Schritt Optionen konfigurieren die Einstellungen unverändert und klicken Sie auf Weiter.
5. Lassen Sie im Schritt Berechtigungen festlegen die Einstellungen unverändert und klicken Sie auf Weiter.
6. Wählen Sie Create Bucket (Bucket erstellen) aus.
7. Öffnen Sie den `ka-app-code-<username>`-Bucket und wählen Sie Hochladen aus.
8. Klicken Sie im Schritt Dateien auswählen auf Dateien hinzufügen. Navigieren Sie zu der `getting-started-scala-1.0.jar`-Datei, die Sie im vorherigen Schritt erstellt haben.
9. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und Ausführen der Anwendung (Konsole)

Führen Sie die folgenden Schritte aus, um die Anwendung mit der Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen der -Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard von Managed Service für Apache Flink die Option Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie für Application name (Anwendungsname) den Text **`MyApplication`** ein.
   - Geben Sie für Beschreibung den Text **`My scala test app`** ein.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.
   - Belassen Sie die Version als Apache Flink Version 1.15.2 (empfohlene Version).
4. Wählen Sie für Access permissions (Zugriffsberechtigungen) die Option IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** erstellen/aktualisieren aus.
5. Wählen Sie Create application aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rolle: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Konfigurieren der Anwendung

Verwenden Sie das folgende Verfahren, um die Anwendung zu konfigurieren.

Konfigurieren der Anwendung

1. Klicken Sie auf der Seite MyApplication auf die Option Configure (Konfigurieren).

2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):
   - Geben Sie für Amazon-S3-Bucket **ka-app-code-*<username>*** ein.
   - Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **getting-started-scala-1.0.jar.** ein.
3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **kinesis-analytics-MyApplication-us-west-2** erstellen/aktualisieren aus.
4. Wählen Sie unter Eigenschaften die Option Gruppe hinzufügen aus.
5. Geben Sie Folgendes ein:

Gruppen-ID	Schlüssel	Value (Wert)
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

   Wählen Sie Save (Speichern).
6. Wählen Sie unter Eigenschaften erneut Gruppe hinzufügen aus.
7. Geben Sie Folgendes ein:

Gruppen-ID	Schlüssel	Value (Wert)
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**
ProducerConfigProperties	**aws.region**	**us-west-2**

8. Stellen Sie unter Monitoring (Überwachung) sicher, dass die Option Monitoring metrics level (Überwachung der Metrikebene) auf Application (Anwendung) festgelegt ist.

9. Aktivieren Sie für CloudWatch-Protokollierung das Kontrollkästchen Aktivieren.
10. Wählen Sie Aktualisieren aus.

> **Note**
>
> Wenn Sie die Amazon CloudWatch-Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf den Amazon S3-Bucket.

Um die IAM-Richtlinie zu bearbeiten, um S3-Bucket-Berechtigungen hinzuzufügen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **`kinesis-analytics-service-MyApplication-us-west-2`**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*`012345678901`*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

Ausführen der Anwendung

Das Flink-Jobdiagramm kann angezeigt werden, indem Sie die Anwendung ausführen, das Apache Flink-Dashboard öffnen und den gewünschten Flink-Job auswählen.

Stoppen der Anwendung

Um die Anwendung zu beenden, wählen Sie auf der Seite MyApplication die Option Stopp. Bestätigen Sie die Aktion.

Erstellen und Ausführen der Anwendung (CLI)

In diesem Abschnitt verwenden Sie AWS Command Line Interface, um die Anwendung Managed Service für Apache Flink zu erstellen und auszuführen. Verwenden Sie den AWS CLI-Befehl kinesisanalyticsv2, um Managed-Service-für-Apache-Flink-Anwendungen zu erstellen und mit diesen zu interagieren.

Erstellen einer Berechtigungsrichtlinie

> **Note**
>
> Sie müssen eine Berechtigungsrichtlinie und eine Rolle für Ihre Anwendung erstellen. Wenn Sie diese IAM-Ressourcen nicht erstellen, kann Ihre Anwendung nicht auf ihre Daten- und Protokollstreams zugreifen.

Zuerst erstellen Sie eine Berechtigungsrichtlinie mit zwei Anweisungen: eine, die Berechtigungen für die Leseaktion auf den Quell-Stream zulässt, und eine andere, die Berechtigungen für Schreibaktionen auf den Senken-Stream zulässt. Anschließend fügen Sie die Richtlinie an eine IAM-

Rolle (die Sie im nächsten Abschnitt erstellen) an. Wenn Managed Service für Apache Flink also die Rolle übernimmt, verfügt der Service über die erforderlichen Berechtigungen zum Lesen aus dem Quell-Stream und zum Schreiben in den Senken-Stream.

Verwenden Sie den folgenden Code zum Erstellen der `AKReadSourceStreamWriteSinkStream`-Berechtigungsrichtlinie. Ersetzen Sie **`username`** durch den Benutzernamen, den Sie verwendet haben, um den Amazon-S3-Bucket zum Speichern des Anwendungscodes zu erstellen. Ersetzen Sie die Konto-ID in den Amazon-Ressourcennamen (ARNs) **`(012345678901)`** mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

Schritt-für-Schritt-Anweisungen zum Erstellen einer Berechtigungsrichtlinie finden Sie unter Praktische Anleitung: Erstellen und Anfügen Ihrer ersten vom Kunden verwalteten Richtlinie im IAM-Benutzerhandbuch.

Erstellen Sie eine IAM-Richtlinie

In diesem Abschnitt erstellen Sie eine IAM-Rolle, die die Anwendung von Managed Service für Apache Flink annehmen kann, um einen Quell-Stream zu lesen und in den Senken-Stream zu schreiben.

Managed Service für Apache Flink kann ohne Berechtigungen nicht auf Ihren Stream zugreifen. Sie erteilen diese Berechtigungen über eine IAM-Rolle. Jeder IAM-Rolle sind zwei Richtlinien angefügt. Die Vertrauensrichtlinie erteilt Managed Service für Apache Flink die Berechtigung zum Übernehmen

der Rolle und die Berechtigungsrichtlinie bestimmt, was Managed Service für Apache Flink nach Annahme der Rolle tun kann.

Sie können die Berechtigungsrichtlinie, die Sie im vorherigen Abschnitt erstellt haben, dieser Rolle anfügen.

So erstellen Sie eine IAM-Rolle

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Rollen und dann Rolle erstellen.
3. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option AWS-Service aus
4. Wählen Sie unter Choose the service that will use this role (Wählen Sie den Service aus, der diese Rolle verwendet) die Option Kinesis aus.
5. Wählen Sie unter Wählen Sie Ihren Anwendungsfall die Option Managed Service for Apache Flink aus.
6. Wählen Sie Next: Permissions (Weiter: Berechtigungen) aus.
7. Wählen Sie auf der Seite Attach permissions policies (Berechtigungsrichtlinien hinzufügen) Next: Review (Weiter: Überprüfen) aus. Sie fügen Berechtigungsrichtlinien an, nachdem Sie die Rolle erstellt haben.
8. Geben Sie auf der Seite Create role (Rolle erstellen) den Text **`MF-stream-rw-role`** für Role name (Rollenname) ein. Wählen Sie Create role (Rolle erstellen) aus.

   Jetzt haben Sie eine neue IAM-Rolle mit dem Namen `MF-stream-rw-role` erstellt. Im nächsten Schritt aktualisieren Sie die Vertrauens- und Berechtigungsrichtlinien für die Rolle
9. Fügen Sie die Berechtigungsrichtlinie der Rolle an.

   > **Note**
   >
   > Für diese Übung übernimmt Managed Service für Apache Flink diese Rolle sowohl für das Lesen von Daten aus einem Kinesis-Datenstrom (Quelle) als auch zum Schreiben der Ausgabedaten in einen anderen Kinesis-Datenstrom. Daher fügen Sie die Richtlinie an, die Sie im vorherigen Schritt, Erstellen einer Berechtigungsrichtlinie, erstellt haben.

   a. Wählen Sie auf der Seite Summary (Übersicht) die Registerkarte Permissions (Berechtigungen) aus.

b. Wählen Sie Attach Policies (Richtlinien anfügen) aus.

c. Geben Sie im Suchfeld **`AKReadSourceStreamWriteSinkStream`** (die Richtlinie, die Sie im vorhergehenden Abschnitt erstellt haben) ein.

d. Wählen Sie die `AKReadSourceStreamWriteSinkStream`-Richtlinie und wählen Sie Richtlinie anhängen aus.

Sie haben nun die Service-Ausführungsrolle erstellt, die Ihre Anwendung für den Zugriff auf Ressourcen verwendet. Notieren Sie sich den ARN der neuen Rolle.

Schritt-für-Schritt-Anweisungen zum Erstellen einer Rolle finden Sie unter Erstellen einer IAM-Rolle (Konsole) im IAM-Benutzerhandbuch.

Erstellen der Anwendung

Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `create_request.json`. Ersetzen Sie den Beispiel-Rollen-ARN durch den ARN für die Rolle, die Sie zuvor erstellt haben. Ersetzen Sie das Bucket-ARN-Suffix (username) mit dem Suffix, das Sie im vorherigen Abschnitt gewählt haben. Ersetzen Sie die beispielhafte Konto-ID (012345678901) in der Service-Ausführungsrolle mit Ihrer Konto-ID.

```
{
    "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"
      }
   ]
}
```

Führen Sie CreateApplication mit der folgenden Anforderung aus, um die Anwendung zu erstellen:

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

Die Anwendung wird nun erstellt. Sie starten die Anwendung im nächsten Schritt.

Starten der Anwendung

In diesem Abschnitt verwenden Sie die StartApplication-Aktion, um die Anwendung zu starten.

So starten Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `start_request.json`.

   ```
   {
       "ApplicationName": "getting_started",
       "RunConfiguration": {
           "ApplicationRestoreConfiguration": {
   ```

```
        "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
        }
    }
}
```

2. Führen Sie die `StartApplication`-Aktion mit der vorherigen Anforderung zum Starten der Anwendung aus:

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

Die Anwendung wird jetzt ausgeführt. Sie können die Metriken des Managed Service für Apache Flink in der Amazon-CloudWatch-Konsole überprüfen, um zu sehen, ob die Anwendung funktioniert.

Stoppen der Anwendung

In diesem Abschnitt verwenden Sie die StopApplication-Aktion, um die Anwendung zu stoppen.

So stoppen Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `stop_request.json`.

```
{
   "ApplicationName": "s3_sink"
}
```

2. Führen Sie die `StopApplication`-Aktion mit der vorherigen Anforderung zum Stoppen der Anwendung aus:

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

Die Anwendung wird nun gestoppt.

Eine CloudWatch-Protokollierungs-Option hinzufügen

Sie können die AWS CLI verwenden, um Ihrer Anwendung einen Amazon CloudWatch-Protokolldatenstrom hinzuzufügen. Informationen zur Verwendung von CloudWatch Logs mit Ihrer Anwendung finden Sie unter Anwendungsprotokollierung einrichten.

Umgebungseigenschaften aktualisieren

In diesem Abschnitt verwenden Sie die Aktion UpdateApplication, um die Umgebungseigenschaften für die Anwendung zu ändern, ohne den Anwendungscode neu kompilieren zu müssen. In diesem Beispiel ändern Sie die Region der Quell- und Ziel-Streams.

So aktualisieren Sie die Umgebungseigenschaften für die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `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. Führen Sie die `UpdateApplication`-Aktion mit der vorherigen Anforderung aus, um die Umgebungseigenschaften zu aktualisieren:

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

Aktualisieren des Anwendungscodes

Wenn Sie Ihren Anwendungscode mit einer neuen Version Ihres Codepakets aktualisieren müssen, verwenden Sie die CLI-Aktion UpdateApplication.

> **Note**
>
> Um eine neue Version des Anwendungscodes mit demselben Dateinamen zu laden, müssen Sie die neue Objektversion angeben. Weitere Informationen zur Verwendung von Amazon S3-Objektversionen finden Sie unter Versionierung aktivieren oder deaktivieren.

Um die AWS CLI zu verwenden, löschen Sie Ihr vorheriges Codepaket aus Ihrem Amazon S3-Bucket, laden Sie die neue Version hoch und rufen Sie `UpdateApplication` auf. Geben Sie dabei denselben Amazon S3-Bucket und Objektnamen sowie die neue Objektversion an. Die Anwendung wird mit dem neuen Codepaket neu gestartet.

Die folgende Beispielanforderung für die `UpdateApplication`-Aktion lädt den Anwendungscode neu und startet die Anwendung neu. Aktualisieren Sie die `CurrentApplicationVersionId` auf die aktuelle Anwendungsversion. Sie können die aktuelle Anwendungsversion mithilfe der Aktionen `ListApplications` oder `DescribeApplication` überprüfen. Aktualisieren Sie das Bucket-Namenssuffix (<username>) mit dem Suffix, das Sie im Erstellen Sie abhängige Ressourcen-Abschnitt ausgewählt haben.

```
{{
    "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"
                }
            }
        }
    }
}
```

Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zur Bereinigung von AWS-Ressourcen, die im Rollierendes Fenster-Tutorial erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte:

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Sie Ihre Kinesis Data Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Sie Ihre CloudWatch-Ressourcen

Löschen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option MyApplication aus.
3. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen Sie Ihre Kinesis Data Streams

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Kinesis Data Streams die Option ExampleInputStream aus.
3. Wählen Sie auf der Seite ExampleInputStream Kinesis-Stream löschen aus und bestätigen Sie dann den Löschvorgang.
4. Wählen Sie auf der Seite Kinesis Streams den ExampleOutputStream aus, wählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den ka-app-code-***<username>***-Bucket aus.

3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle kinesis-analytics-MyApplication-us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Sie Ihre CloudWatch-Ressourcen

1. Öffnen Sie die CloudWatch-Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Erstellen von Managed Service für Apache Flink-Anwendungen mit Apache Beam

Sie können das Apache Beam-Framework mit Ihrer Managed Service for Apache Flink-Anwendung verwenden, um Streaming-Daten zu verarbeiten. Managed Service für Apache Flink-Anwendungen, die Apache Beam verwenden, verwendet Apache Flink Runner zur Ausführung von Beam-Pipelines.

Ein Tutorial zur Verwendung von Apache Beam in einer Managed Service for Apache Flink-Anwendung finden Sie unterVerwenden von CloudFormation mit Managed Service für Apache Flink.

Dieses Thema enthält die folgenden Abschnitte:

- Verwendung von Apache Beam mit Managed Service für Apache Flink
- Beam-Fähigkeiten
- Erstellen einer Anwendung mit Apache Beam

Verwendung von Apache Beam mit Managed Service für Apache Flink

Beachten Sie Folgendes zur Verwendung des Apache Flink Runners mit Managed Service für Apache Flink:

- Apache Beam-Metriken sind in der Managed Service for Apache Flink-Konsole nicht sichtbar.
- Apache Beam wird nur mit Managed Service für Apache Flink-Anwendungen unterstützt, die Apache Flink Version 1.8 und höher verwenden. Apache Beam wird mit Managed Service für Apache Flink-Anwendungen, die Apache Flink Version 1.6 verwenden, nicht unterstützt.

Beam-Fähigkeiten

Managed Service für Apache Flink unterstützt dieselben Apache Beam-Funktionen wie der Apache Flink Runner. Informationen darüber, welche Feature vom Apache Flink Runner unterstützt werden, finden Sie in der Beam-Kompatibilitätsmatrix.

Wir empfehlen Ihnen, Ihre Apache Flink-Anwendung im Managed Service for Apache Flink-Dienst zu testen, um sicherzustellen, dass wir alle Feature unterstützen, die Ihre Anwendung benötigt.

Erstellen einer Anwendung mit Apache Beam

In dieser Übung erstellen Sie eine Managed Service for Apache Flink-Anwendung, die Daten mithilfe von Apache Beam transformiert. Apache Beam ist ein Programmiermodell für die Verarbeitung von Streaming-Daten. Informationen zur Verwendung von Apache Beam mit Managed Service für Apache Flink finden Sie unterVerwendung von Apache Beam.

> **Note**
>
> Um die erforderlichen Voraussetzungen für diese Übung festzulegen, schließen Sie zunächst die Erste Schritte (DataStream API) Übung ab.

Dieses Thema enthält die folgenden Abschnitte:

- Erstellen Sie abhängige Ressourcen
- Schreiben Sie Beispieldatensätze in den Eingabe-Stream
- Laden Sie den Anwendungscode herunter und überprüfen Sie ihn
- Kompilieren des Anwendungscodes
- Hochladen des Apache Flink-Streaming-Java-Codes
- Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus
- Bereinigen von Ressourcen
- Nächste Schritte

Erstellen Sie abhängige Ressourcen

Bevor Sie für diese Übung eine Anwendung von Managed Service für Apache Flink erstellen, erstellen Sie die folgenden abhängigen Ressourcen:

- Zwei Kinesis-Datenströme (`ExampleInputStream`und`ExampleOutputStream`)
- Einen Amazon S3-Bucket zum Speichern des Anwendungscodes (`ka-app-code-<username>`)

Sie können die Kinesis-Streams und den Amazon S3 S3-Bucket mithilfe der Konsole erstellen. Anweisungen zum Erstellen dieser Ressourcen finden Sie in den folgenden Themen:

- Erstellen und Aktualisieren von Datenströmen im Amazon Kinesis Data Streams Entwicklerhandbuch. Benennen Sie Ihre Data Streams **`ExampleInputStream`** und **`ExampleOutputStream`**.
- Wie erstelle ich einen S3-Bucket? im Amazon Simple Storage Service-Benutzerhandbuch Geben Sie dem Amazon S3-Bucket einen weltweit eindeutigen Namen, indem Sie Ihren Anmeldenamen anhängen, z. B. **`ka-app-code-<username>`**

Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Abschnitt verwenden Sie ein Python-Skript zum Schreiben von Datensätzen in den Stream für die zu verarbeitende Anwendung.

> **Note**
>
> Dieser Abschnitt erfordert AWS SDK for Python (Boto).

1. Erstellen Sie eine Datei `ping.py` mit dem folgenden Inhalt:

```
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. Führen Sie das `ping.py`Skript aus:

```
$ python ping.py
```

   Lassen Sie das Skript laufen, während Sie den Rest des Tutorials abschließen.

Laden Sie den Anwendungscode herunter und überprüfen Sie ihn

Der Java-Anwendungscode für diese Beispiele ist auf GitHub verfügbar. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben. Weitere Informationen finden Sie unter Installieren Git.
2. Klonen Sie das Remote-Repository mit dem folgenden Befehl:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/Beam` Verzeichnis .

Der Anwendungscode befindet sich in der `BasicBeamStreamingJob.java` Datei. Beachten Sie Folgendes zum Anwendungscode:

- Die Anwendung verwendet Apache Beam ParDo, um eingehende Datensätze zu verarbeiten, indem sie eine benutzerdefinierte Transformationsfunktion namens aufruft`PingPongFn`.

  Der Code zum Aufrufen der `PingPongFn` Funktion lautet wie folgt:

```
.apply("Pong transform",
    ParDo.of(new PingPongFn())
```

- Managed Service für Apache Flink-Anwendungen, die Apache Beam verwenden, erfordert die folgenden Komponenten. Wenn Sie diese Komponenten und Versionen nicht in Ihre `pom.xml` aufnehmen, lädt Ihre Anwendung die falschen Versionen aus den Umgebungsabhängigkeiten, und da die Versionen nicht übereinstimmen, stürzt Ihre Anwendung zur Laufzeit ab.

```
<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>
```

- Die `PingPongFn` Transformationsfunktion übergibt die Eingabedaten an den Ausgabestrom, sofern es sich bei den Eingabedaten nicht um einen Ping-Wert handelt. In diesem Fall gibt sie die Zeichenfolge pong\nan den Ausgabestrom aus.

  Der Code der Transformationsfunktion lautet wie folgt:

```
    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());
        }
    }
}
```

Kompilieren des Anwendungscodes

Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Installieren Sie Java und Maven, wenn das noch nicht geschehen ist. Weitere Informationen finden Sie unter Voraussetzungen im Erste Schritte (DataStream API) Tutorial.
2. Kompilieren Sie die Anwendung mit dem folgenden Befehl:

```
mvn package -Dflink.version=1.15.3 -Dflink.version.minor=1.8
```

> **Note**
>
> Der bereitgestellte Quellcode basiert auf Bibliotheken von Java 11.

Beim Kompilieren der Anwendung wird die JAR-Datei der Anwendung (`target/basic-beam-app-1.0.jar`) erstellt.

Hochladen des Apache Flink-Streaming-Java-Codes

In diesem Abschnitt erstellen Sie Ihre und laden sie in den -Bucket hoch, den Sie im Erstellen Sie abhängige Ressourcen Abschnitt erstellt haben.

1. Wählen Sie in der Amazon-S3-Konsole den ka-app-code-***<username>***-Bucket aus und klicken Sie auf Hochladen.
2. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `basic-beam-app-1.0.jar`Datei, die Sie im vorherigen Schritt erstellt haben.
3. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Managed Service für Apache Flink-Dashboard die Option Erstellen Sie Analyseanwendung aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Wählen Sie als Laufzeit Apache Flink aus.

> **Note**
>
> Managed Service für Apache Flink verwendet Apache Flink Version 1.15.2.

- Belassen Sie den Versions-Pulldown bei Apache Flink Version 1.15.2 (empfohlene Version).

4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink für Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rolle: `kinesis-analytics-MyApplication-`*`us-west-2`*

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf die Kinesis-Datenströme.

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **`kinesis-analytics-service-MyApplication-us-west-2`**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*`012345678901`*) mit Ihrer Konto-ID.

```
{
```

```
    "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"
        }
    ]
}
```

Konfigurieren der Anwendung

1. Klicken Sie auf der Seite MyApplication auf die Option Configure (Konfigurieren).
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):
   - Geben Sie für Amazon-S3-Bucket **ka-app-code-*<username>*** ein.
   - Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **basic-beam-app-1.0.jar** ein.
3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **kinesis-analytics-MyApplication-us-west-2** erstellen/aktualisieren aus.
4. Geben Sie Folgendes ein:

Gruppen-ID	Schlüssel	Value (Wert)
BeamApplicationProperties	**InputStreamName**	**ExampleInputStream**
BeamApplicationProperties	**OutputStreamName**	**ExampleOutputStream**
BeamApplicationProperties	**AwsRegion**	**us-west-2**

5. Stellen Sie unter Monitoring (Überwachung) sicher, dass die Option Monitoring metrics level (Überwachung der Metrikebene) auf Application (Anwendung) festgelegt ist.

6. Aktivieren Sie für CloudWatch-Protokollierung das Kontrollkästchen Aktivieren.
7. Wählen Sie Aktualisieren aus.

> **Note**
>
> Wenn Sie die CloudWatch-Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`
>
> Dieser Protokollstream wird zur Überwachung der Anwendung verwendet. Dies ist nicht derselbe Protokollstream, den die Anwendung zum Senden von Ergebnissen verwendet.

Ausführen der Anwendung

Das Flink-Jobdiagramm kann angezeigt werden, indem Sie die Anwendung ausführen, das Apache Flink-Dashboard öffnen und den gewünschten Flink-Job auswählen.

Sie können die Metriken des Managed Service für Apache Flink in der Amazon-CloudWatch-Konsole überprüfen, um zu sehen, ob die Anwendung funktioniert.

Bereinigen von Ressourcen

Dieser Abschnitt enthält Verfahren zur Bereinigung von AWS-Ressourcen, die im Rollierendes Fenster-Tutorial erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte:

- Erstellen Sie die Anwendung Managed Service für Apache Flink
- Löschen Sie Ihre Kinesis Data Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Sie Ihre CloudWatch-Ressourcen

Erstellen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. wählen Sie im Bereich Managed Service für Apache Flink MyApplication.
3. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie den Löschvorgang.

Löschen Sie Ihre Kinesis Data Streams

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Kinesis Data Streams die Option ExampleInputStream aus.
3. Wählen Sie auf der Seite ExampleInputStream Kinesis-Stream löschen aus und bestätigen Sie dann den Löschvorgang.
4. Wählen Sie auf der Seite Kinesis-Streams den ExampleOutputStream aus, wählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon S3-Konsole unter https://console.aws.amazon.com/s3/.
2. <username>Wählen Sie den ka-app-code-Bucket aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie Kinesis-Analytics-Service-MyApplication-US-West-2.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle Kinesis-Analytics-MyApplication-US-West-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Sie Ihre CloudWatch-Ressourcen

1. Öffnen Sie die CloudWatch-Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/Kinesis-Analytics/MyApplication aus.
4. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Nächste Schritte

Nachdem Sie nun eine grundlegende Managed Service for Apache Flink-Anwendung erstellt und ausgeführt haben, die Daten mithilfe von Apache Beam transformiert, finden Sie in der folgenden Anwendung ein Beispiel für eine erweiterte Managed Service für Apache Flink-Lösung.

- Workshop „Beam on Managed Service for Apache Flink Streaming“: In diesem Workshop untersuchen wir ein durchgängiges Beispiel, das Batch- und Streaming-Aspekte in einer einheitlichen Apache Beam-Pipeline kombiniert.

Schulungsworkshops, Labore und Lösungsimplementierungen

Die folgenden end-to-end Beispiele zeigen erweiterte Managed Service für Apache Flink-Lösungen.

Themen

- Lokales Entwickeln von Apache-Flink-Anwendungen vor der Bereitstellung mit Managed Service für Apache Flink
- Ereigniserkennung mit Managed Service für Apache Flink Studio
- AWS-Streaming-Datenlösung für Amazon Kinesis
- Clickstream Lab mit Apache Flink und Apache Kafka
- Benutzerdefiniertes Skalieren mit Auto Scaling von Anwendungen
- Amazon CloudWatch -Dashboard
- AWS-Streaming-Datenlösung für Amazon MSK
- Weitere Managed Service für Apache Flink-Lösungen in GitHub

Lokales Entwickeln von Apache-Flink-Anwendungen vor der Bereitstellung mit Managed Service für Apache Flink

Dieser Workshop vermittelt Ihnen die Grundlagen für den Einstieg in die lokale Entwicklung von Apache-Flink-Anwendungen mit dem langfristigen Ziel, sie mit Managed Service für Apache Flink zu implementieren.

Die Lösung finden Sie hier: Einstiegsleitfaden zur lokalen Entwicklung mit Apache Flink

Ereigniserkennung mit Managed Service für Apache Flink Studio

Dieser Workshop beschreibt die Ereigniserkennung mit Managed Service für Apache Flink Studio und die Bereitstellung als Anwendung, die Managed Service für Apache Flink nutzt

Die Lösung finden Sie hier: Ereigniserkennung mit Managed Service für Apache Flink

AWS-Streaming-Datenlösung für Amazon Kinesis

Die AWS-Streaming-Datenlösung für Amazon Kinesis konfiguriert automatisch die AWS-Services, die für die einfache Erfassung, Speicherung, Verarbeitung und Bereitstellung von Streaming-Daten erforderlich sind. Die Lösung bietet mehrere Optionen zur Lösung von Anwendungsfällen mit Streaming-Daten. Die Option Managed Service für Apache Flink bietet ein - end-to-end Streaming-ETL-Beispiel, das eine reale Anwendung demonstriert, die analytische Operationen für simulierte New York-Taxisdaten ausführt.

Jede Lösung enthält die folgenden Komponenten:

- Ein AWS CloudFormation-Paket zur Bereitstellung des vollständigen Beispiels.
- Ein CloudWatch Dashboard zum Anzeigen von Anwendungsmetriken.
- CloudWatch -Alarme für die relevantesten Anwendungsmetriken.
- Alle erforderlichen IAM-Rollen und -Richtlinien.

Die Lösung finden Sie hier: Streaming-Datenlösung für Amazon Kinesis

Clickstream Lab mit Apache Flink und Apache Kafka

Ein durchgängiges Labor für Clickstream-Anwendungsfälle mit Amazon Managed Streaming für Apache Kafka als Streaming-Speicher und Managed Service für Apache Flink für Apache-Flink-Anwendungen für die Stream-Verarbeitung.

Die Lösung finden Sie hier: Clickstream Lab

Benutzerdefiniertes Skalieren mit Auto Scaling von Anwendungen

Ein Beispiel, das Benutzern hilft, ihre Managed Service für Apache Flink-Anwendungen mithilfe von Application Auto Scaling automatisch zu skalieren. Auf diese Weise können Benutzer benutzerdefinierte Skalierungsrichtlinien und benutzerdefinierte Skalierungsattribute einrichten.

Die Lösungen finden Sie hier:

- Managed Service für Apache Flink App Autoscaling
- Geplante Skalierung

Weitere Informationen dazu, wie Sie eine benutzerdefinierte Skalierung durchführen können, finden Sie unter Aktivieren der metrikbasierten und geplanten Skalierung für Amazon Managed Service für Apache Flink.

Amazon CloudWatch -Dashboard

Ein Beispiel CloudWatch -Dashboard für die Überwachung von Managed Service für Apache Flink-Anwendungen. Das Beispiel-Dashboard enthält auch eine Demo-Anwendung, mit der Sie die Funktionalität des Dashboards demonstrieren können.

Die Lösung finden Sie hier: Managed Service für Apache Flink Metrik-Dashboard

AWS-Streaming-Datenlösung für Amazon MSK

Die neueste AWS-Streaming-Datenlösung für Amazon MSK stellt AWS CloudFormation-Vorlagen bereit, bei denen Daten durch Produzenten, Streaming-Speicher, Verbraucher und Ziele fließen.

Die Lösung finden Sie hier: AWS-Streaming-Datenlösung für Amazon MSK

Weitere Managed Service für Apache Flink-Lösungen in GitHub

Die folgenden end-to-end Beispiele zeigen erweiterte Managed Service für Apache Flink-Lösungen und sind auf verfügbar GitHub:

- Amazon Managed Service für Apache Flink – Benchmarking-Programm
- Snapshot Manager – Amazon Managed Service für Apache Flink
- Streamen von ETL mit Apache Flink und Amazon Managed Service für Apache Flink
- Stimmungsanalyse von Kundenfeedback in Echtzeit

Dienstprogramme

Die folgenden Dienstprogramme können die Verwendung des Services von Managed Service für Apache Flink vereinfachen:

Themen

- Snapshot-Manager
- Benchmarking

Snapshot-Manager

Es ist Best Practice für Flink-Anwendungen, regelmäßig Savepoints/Snapshots auszulösen, um eine reibungslosere Wiederherstellung nach einer Störung zu ermöglichen. Der Snapshot-Manager automatisiert diese Aufgabe und bietet folgende Vorteile:

- erstellt einen neuen Snapshot einer laufenden Anwendung, die Managed Service für Apache Flink nutzt
- ruft eine Anzahl von Anwendungs-Snapshots ab
- prüft, ob die Anzahl die erforderliche Anzahl von Snapshots übersteigt
- löscht ältere Snapshots, die älter als die erforderliche Anzahl sind

Ein Beispiel finden Sie unter Snapshot-Manager.

Benchmarking

Das Benchmarking-Dienstprogramm von Managed Service für Apache Flink hilft bei der Kapazitätsplanung, bei Integrationstests und beim Benchmarking von Anwendungen, die Managed Service für Apache Flink nutzen.

Ein Beispiel finden Sie unter Benchmarking

Managed Service für Apache Flink: Beispiele

Dieser Abschnitt enthält Beispiele für das Erstellen und Arbeiten mit Anwendungen im Managed Service für Apache Flink. Sie enthalten Beispielcode und step-by-step Anweisungen, mit denen Sie Managed Service für Apache Flink-Anwendungen erstellen und Ihre Ergebnisse testen können.

Bevor Sie sich mit diesen Beispielen befassen, empfehlen wir Ihnen, zunächst Folgendes zu lesen:

- So funktioniert's
- Erste Schritte (DataStream API)

> **Note**
>
> In diesen Beispielen wird vorausgesetzt, dass Sie die Region USA West (Oregon) (`us-west-2`) verwenden. Wenn Sie eine andere Region verwenden, aktualisieren Sie Ihren Anwendungscode, Ihre Befehle und IAM-Rollen entsprechend.

Themen

- DataStream API-Beispiele
- Python-Beispiele
- Scala-Beispiele

DataStream API-Beispiele

Die folgenden Beispiele zeigen, wie Sie Anwendungen mit der Apache Flink DataStream API erstellen.

Themen

- Beispiel: Rollierendes Fenster
- Beispiel: Gleitendes Fenster
- Beispiel: In einen Amazon-S3-Bucket schreiben
- Tutorial: Verwenden einer Managed Service für Apache Flink-Anwendung zum Replizieren von Daten von einem Thema in einem MSK-Cluster zu einem anderen in einer VPC

- Beispiel: Verwenden Sie einen EFO-Verbraucher mit einem Kinesis Data Stream
- Beispiel: Schreiben in Kinesis Data Firehose
- Beispiel: Aus einem Kinesis Stream in einem anderen Konto lesen
- Tutorial: Einen benutzerdefinierten Truststore mit Amazon MSK verwenden

Beispiel: Rollierendes Fenster

In dieser Übung erstellen Sie eine Anwendung von Managed Service für Apache Flink, die Daten in einem rollierenden Fenster aggregiert. Die Aggregation ist in Flink standardmäßig aktiviert. Um sie deaktivieren, verwenden Sie Folgendes:

```
sink.producer.aggregation-enabled' = 'false'
```

> **Note**
>
> Um die erforderlichen Voraussetzungen für diese Übung einzurichten, schließen Sie zunächst die Erste Schritte (DataStream API)-Übung ab.

Dieses Thema enthält die folgenden Abschnitte:

- Erstellen Sie abhängige Ressourcen
- Schreiben Sie Beispieldatensätze in den Eingabe-Stream
- Laden Sie den Anwendungscode herunter und überprüfen Sie ihn
- Kompilieren des Anwendungscodes
- Hochladen des Apache Flink-Streaming-Java-Codes
- Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus
- Bereinigen von AWS-Ressourcen

Erstellen Sie abhängige Ressourcen

Bevor Sie für diese Übung eine Anwendung von Managed Service für Apache Flink erstellen, erstellen Sie die folgenden abhängigen Ressourcen:

- Zwei Kinesis Data Streams (`ExampleInputStream` und `ExampleOutputStream`)
- Einen Amazon S3-Bucket zum Speichern des Anwendungscodes (`ka-app-code-<username>`)

Sie können die Kinesis Streams und den Amazon-S3-Bucket mithilfe der Konsole erstellen. Anweisungen zum Erstellen dieser Ressourcen finden Sie in den folgenden Themen:

- Data Streams erstellen und aktualisieren im Amazon Kinesis Data Streams Entwicklerleitfaden. Benennen Sie Ihren Data Stream **`ExampleInputStream`** und **`ExampleOutputStream`**.
- Wie erstelle ich einen S3-Bucket? im Amazon Simple Storage Service Benutzerhandbuch. Geben Sie dem Amazon S3-Bucket einen global eindeutigen Namen, indem Sie Ihren Anmeldenamen anhängen, z. B. **`ka-app-code-<username>`**.

Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Abschnitt verwenden Sie ein Python-Skript zum Schreiben von Datensätzen in den Stream für die zu verarbeitende Anwendung.

> **Note**
>
> Dieser Abschnitt erfordert AWS SDK for Python (Boto).

1. Erstellen Sie eine Datei `stock.py` mit dem folgenden Inhalt:

```
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. Führen Sie das `stock.py`Skript aus:

```
$ python stock.py
```

   Lassen Sie das Skript laufen, während Sie den Rest des Tutorials abschließen.

Laden Sie den Anwendungscode herunter und überprüfen Sie ihn

Der Java-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben. Weitere Informationen finden Sie unter Git installieren.
2. Klonen Sie das Remote-Repository mit dem folgenden Befehl:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/TumblingWindow` Verzeichnis .

Der Anwendungscode befindet sich in der `TumblingWindowStreamingJob.java`-Datei. Beachten Sie Folgendes zum Anwendungscode:

- Die Anwendung verwendet eine Kinesis-Quelle zum Lesen aus dem Quell-Stream. Der folgende Codeausschnitt erstellt die Kinesis-Quelle:

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

- Fügen Sie die folgende Importanweisung hinzu:

```
import
 org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows; //
flink 1.13 onward
```

- Die Anwendung verwendet den `timeWindow`-Operator, um die Anzahl der Werte für jedes Aktionssymbol über ein rollierendes Fenster von 5 Sekunden zu ermitteln. Der folgende Code erstellt den Operator und sendet die aggregierten Daten an eine neue Kinesis Data Streams Senke:

```
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());
```

Kompilieren des Anwendungscodes

Zum Kompilieren der Anwendung gehen Sie wie folgt vor:

1. Installieren Sie Java und Maven, wenn das noch nicht geschehen ist. Weitere Informationen finden Sie unter Voraussetzungen im Erste Schritte (DataStream API) Tutorial.
2. Kompilieren Sie die Anwendung mit dem folgenden Befehl:

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

> **Note**
>
> Der bereitgestellte Quellcode basiert auf Bibliotheken von Java 11.

Beim Kompilieren der Anwendung wird die JAR-Datei der Anwendung (`target/aws-kinesis-analytics-java-apps-1.0.jar`) erstellt.

Hochladen des Apache Flink-Streaming-Java-Codes

In diesem Abschnitt laden Sie Ihren Anwendungscode in den Amazon S3-Bucket hoch, den Sie im Erstellen Sie abhängige Ressourcen-Abschnitt erstellt haben.

1. Wählen Sie in der Amazon S3-Konsole den Bucket ka-app-code-***<username>*** und dann Upload aus.
2. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `aws-kinesis-analytics-java-apps-1.0.jar`Datei, die Sie im vorherigen Schritt erstellt haben.
3. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.

     > **Note**
     >
     > Managed Service für Apache Flink verwendet Apache Flink Version 1.15.2.

   - Belassen Sie den Versions-Pulldown bei Apache Flink Version 1.15.2 (empfohlene Version).
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.

5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-MyApplication-us-west-2`
> - Rolle: `kinesisanalytics-MyApplication-us-west-2`

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf die Kinesis-Datenströme.

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **`kinesis-analytics-service-MyApplication-us-west-2`**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*012345678901*) mit Ihrer Konto-ID.

```
{
    "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"
```

```
        }
    ]
}
```

Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):
   - Geben Sie für Amazon-S3-Bucket **ka-app-code-*<username>*** ein.
   - Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **aws-kinesis-analytics-java-apps-1.0.jar** ein.
3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **kinesis-analytics-MyApplication-us-west-2** erstellen/aktualisieren aus.
4. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
5. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
6. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`
>
> Dieser Protokollstream wird zur Überwachung der Anwendung verwendet. Dies ist nicht derselbe Protokollstream, den die Anwendung zum Senden von Ergebnissen verwendet.

Ausführen der Anwendung

1. Wählen Sie auf der MyApplication Seite Ausführen aus. Lassen Sie die Option Ohne Snapshot ausführen aktiviert und bestätigen Sie die Aktion.
2. Wenn die Anwendung ausgeführt wird, aktualisieren Sie die Seite. Die Konsole zeigt den Application graph (Anwendungs-Graph) an.

Sie können die Metriken von Managed Service für Apache Flink in der CloudWatch Konsole überprüfen, um zu überprüfen, ob die Anwendung funktioniert.

Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zur Bereinigung von AWS-Ressourcen, die im Rollierendes Fenster-Tutorial erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte:

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Ihrer Kinesis Data Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen

Löschen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen Ihrer Kinesis Data Streams

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Kinesis Data Streams die Option ausExampleInputStream.
3. Wählen Sie auf der ExampleInputStream Seite Kinesis Stream löschen aus und bestätigen Sie dann den Löschvorgang.

4. Wählen Sie auf der Seite Kinesis Streams die , ExampleOutputStreamwählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den ka-app-codeBucket -***<username>*** aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle kinesis-analytics-MyApplication--us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Beispiel: Gleitendes Fenster

> **Note**
>
> Um die erforderlichen Voraussetzungen für diese Übung einzurichten, schließen Sie zunächst die Erste Schritte (DataStream API)-Übung ab.

Dieses Thema enthält die folgenden Abschnitte:

- Erstellen Sie abhängige Ressourcen
- Schreiben Sie Beispieldatensätze in den Eingabe-Stream
- Laden Sie den Anwendungscode herunter und überprüfen Sie ihn
- Kompilieren des Anwendungscodes
- Hochladen des Apache Flink-Streaming-Java-Codes
- Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus
- Bereinigen von AWS-Ressourcen

Erstellen Sie abhängige Ressourcen

Bevor Sie für diese Übung eine Anwendung von Managed Service für Apache Flink erstellen, erstellen Sie die folgenden abhängigen Ressourcen:

- Zwei Kinesis-Datenströme (`ExampleInputStream` und `ExampleOutputStream`).
- Einen Amazon S3-Bucket zum Speichern des Anwendungscodes (`ka-app-code-<username>`)

Sie können die Kinesis Streams und den Amazon-S3-Bucket mithilfe der Konsole erstellen. Anweisungen zum Erstellen dieser Ressourcen finden Sie in den folgenden Themen:

- Data Streams erstellen und aktualisieren im Amazon Kinesis Data Streams Entwicklerleitfaden. Benennen Sie Ihre Data Streams **`ExampleInputStream`** und **`ExampleOutputStream`**.
- Wie erstelle ich einen S3-Bucket? im Amazon Simple Storage Service Benutzerhandbuch. Geben Sie dem Amazon S3-Bucket einen global eindeutigen Namen, indem Sie Ihren Anmeldenamen anhängen, z. B. **`ka-app-code-<username>`**.

Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Abschnitt verwenden Sie ein Python-Skript zum Schreiben von Datensätzen in den Stream für die zu verarbeitende Anwendung.

> **Note**
>
> Dieser Abschnitt erfordert AWS SDK for Python (Boto).

1. Erstellen Sie eine Datei `stock.py` mit dem folgenden Inhalt:

```
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. Führen Sie das `stock.py`Skript aus:

```
$ python stock.py
```

   Lassen Sie das Skript laufen, während Sie den Rest des Tutorials abschließen.

Laden Sie den Anwendungscode herunter und überprüfen Sie ihn

Der Java-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben. Weitere Informationen finden Sie unter Git installieren.
2. Klonen Sie das Remote-Repository mit dem folgenden Befehl:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/SlidingWindow` Verzeichnis .

Der Anwendungscode befindet sich in der `SlidingWindowStreamingJobWithParallelism.java`-Datei. Beachten Sie Folgendes zum Anwendungscode:

- Die Anwendung verwendet eine Kinesis-Quelle zum Lesen aus dem Quell-Stream. Der folgende Codeausschnitt erstellt die Kinesis-Quelle:

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

- Die Anwendung verwendet den `timeWindow`-Operator, um in einem 10-Sekunden-Fenster, das um 5 Sekunden verschoben wird, den Mindestwert für jedes Aktionssymbol zu ermitteln. Der folgende Code erstellt den Operator und sendet die aggregierten Daten an eine neue Kinesis Data Streams Senke:
- Fügen Sie die folgende Importanweisung hinzu:

```
import
 org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows; //
flink 1.13 onward
```

- Die Anwendung verwendet den `timeWindow`-Operator, um die Anzahl der Werte für jedes Aktionssymbol über ein rollierendes Fenster von 5 Sekunden zu ermitteln. Der folgende Code erstellt den Operator und sendet die aggregierten Daten an eine neue Kinesis Data Streams Senke:

```
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());
```

Kompilieren des Anwendungscodes

Zum Kompilieren der Anwendung gehen Sie wie folgt vor:

1. Installieren Sie Java und Maven, wenn das noch nicht geschehen ist. Weitere Informationen finden Sie unter Voraussetzungen im Erste Schritte (DataStream API) Tutorial.
2. Kompilieren Sie die Anwendung mit dem folgenden Befehl:

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

> **Note**
>
> Der bereitgestellte Quellcode basiert auf Bibliotheken von Java 11.

Beim Kompilieren der Anwendung wird die JAR-Datei der Anwendung (`target/aws-kinesis-analytics-java-apps-1.0.jar`) erstellt.

Hochladen des Apache Flink-Streaming-Java-Codes

In diesem Abschnitt laden Sie Ihren Anwendungscode in den Amazon S3-Bucket hoch, den Sie im Abschnitt Erstellen Sie abhängige Ressourcen erstellt haben.

1. Wählen Sie in der Amazon S3-Konsole den Bucket ka-app-code-***<username>*** und dann Hochladen aus.
2. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `aws-kinesis-analytics-java-apps-1.0.jar`Datei, die Sie im vorherigen Schritt erstellt haben.
3. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.
   - Belassen Sie den Versions-Pulldown bei Apache Flink Version 1.15.2 (empfohlene Version).
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rolle: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf die Kinesis-Datenströme.

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **`kinesis-analytics-service-MyApplication-us-west-2`**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*012345678901*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):
   - Geben Sie für Amazon-S3-Bucket **ka-app-code-*<username>*** ein.
   - Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **aws-kinesis-analytics-java-apps-1.0.jar** ein.
3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **kinesis-analytics-MyApplication-us-west-2** erstellen/aktualisieren aus.
4. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.

5. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
6. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die Amazon- CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`
>
> Dieser Protokollstream wird zur Überwachung der Anwendung verwendet. Dies ist nicht derselbe Protokollstream, den die Anwendung zum Senden von Ergebnissen verwendet.

Konfigurieren Sie die Anwendungsparallelität

Dieses Anwendungsbeispiel verwendet die parallele Ausführung von Aufgaben. Der folgende Anwendungscode legt die Parallelität des Operators `min` fest:

```
.setParallelism(3) // Set parallelism for the min operator
```

Die Anwendungsparallelität kann nicht größer sein als die bereitgestellte Parallelität, die den Standardwert 1 hat. Verwenden Sie die folgende AWS CLI-Aktion, um die Parallelität Ihrer Anwendung zu erhöhen:

```
aws kinesisanalyticsv2 update-application
      --application-name MyApplication
      --current-application-version-id <VersionId>
      --application-configuration-update "{\"FlinkApplicationConfigurationUpdate
\": { \"ParallelismConfigurationUpdate\": {\"ParallelismUpdate\": 5,
 \"ConfigurationTypeUpdate\": \"CUSTOM\" }}}"
```

Sie können die aktuelle Anwendungsversions-ID mit den ListApplications Aktionen DescribeApplication oder abrufen.

Ausführen der Anwendung

Das Flink-Jobdiagramm kann angezeigt werden, indem Sie die Anwendung ausführen, das Apache Flink-Dashboard öffnen und den gewünschten Flink-Job auswählen.

Sie können die Metriken von Managed Service für Apache Flink in der CloudWatch Konsole überprüfen, um zu überprüfen, ob die Anwendung funktioniert.

Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zum Bereinigen von AWS-Ressourcen, die im Tutorial Gleitendes Fenster erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte:

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Ihrer Kinesis Data Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen

Löschen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen Ihrer Kinesis Data Streams

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Kinesis Data Streams die Option ausExampleInputStream.
3. Wählen Sie auf der ExampleInputStream Seite Kinesis Stream löschen aus und bestätigen Sie dann den Löschvorgang.
4. Wählen Sie auf der Seite Kinesis Streams die , ExampleOutputStreamwählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den ka-app-codeBucket -*<username>* aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle kinesis-analytics--MyApplication-us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Beispiel: In einen Amazon-S3-Bucket schreiben

In dieser Übung erstellen Sie einen Managed Service für Apache Flink, der einen Kinesis Data Stream als Quelle und einen Amazon S3-Bucket als Senke hat. Mithilfe der Senke können Sie die Ausgabe der Anwendung in der Amazon S3-Konsole überprüfen.

> **Note**
>
> Um die erforderlichen Voraussetzungen für diese Übung einzurichten, schließen Sie zunächst die Erste Schritte (DataStream API)-Übung ab.

Dieses Thema enthält die folgenden Abschnitte:

- Erstellen Sie abhängige Ressourcen
- Schreiben Sie Beispieldatensätze in den Eingabe-Stream
- Laden Sie den Anwendungscode herunter und überprüfen Sie ihn
- Modifizieren Sie den Anwendungscode
- Kompilieren des Anwendungscodes
- Hochladen des Apache Flink-Streaming-Java-Codes
- Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus
- Überprüfen der Anwendungsausgabe
- Optional: Passen Sie Quelle und Senke an
- Bereinigen von AWS-Ressourcen

Erstellen Sie abhängige Ressourcen

Bevor Sie für diese Übung einen Managed Service für Apache Flink erstellen, erstellen Sie die folgenden abhängigen Ressourcen:

- Einen Kinesis Data Stream (`ExampleInputStream`).
- Einen Amazon S3-Bucket zum Speichern des Codes und der Ausgabe der Anwendung (`ka-app-code-<username>`)

> **Note**
>
> Managed Service für Apache Flink kann keine Daten auf Amazon S3 schreiben, wenn die serverseitige Verschlüsselung auf Managed Service für Apache Flink aktiviert ist.

Sie können den Kinesis-Stream und den Amazon-S3-Bucket mithilfe der Konsole erstellen. Anweisungen zum Erstellen dieser Ressourcen finden Sie in den folgenden Themen:

- Data Streams erstellen und aktualisieren im Amazon Kinesis Data Streams Entwicklerleitfaden. Benennen Sie Ihren Data Stream **`ExampleInputStream`**.
- Wie erstelle ich einen S3-Bucket? im Amazon Simple Storage Service Benutzerhandbuch. Geben Sie dem Amazon S3-Bucket einen global eindeutigen Namen, indem Sie Ihren Anmeldenamen

anhängen, z. B. **ka-app-code-*<username>***. Erstellen Sie zwei Ordner (**code** und **data**) im Amazon S3-Bucket.

Die Anwendung erstellt die folgenden CloudWatch Ressourcen, wenn sie noch nicht vorhanden sind:

- Eine Protokollgruppe namens `/AWS/KinesisAnalytics-java/MyApplication`.
- Einen Protokollstream mit dem Namen `kinesis-analytics-log-stream`.

Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Abschnitt verwenden Sie ein Python-Skript zum Schreiben von Datensätzen in den Stream für die zu verarbeitende Anwendung.

> **Note**
>
> Dieser Abschnitt erfordert AWS SDK for Python (Boto).

1. Erstellen Sie eine Datei `stock.py` mit dem folgenden Inhalt:

```
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. Führen Sie das `stock.py`Skript aus:

```
$ python stock.py
```

   Lassen Sie das Skript laufen, während Sie den Rest des Tutorials abschließen.

Laden Sie den Anwendungscode herunter und überprüfen Sie ihn

Der Java-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben. Weitere Informationen finden Sie unter Git installieren.
2. Klonen Sie das Remote-Repository mit dem folgenden Befehl:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/S3Sink` Verzeichnis .

Der Anwendungscode befindet sich in der `S3StreamingSinkJob.java`-Datei. Beachten Sie Folgendes zum Anwendungscode:

- Die Anwendung verwendet eine Kinesis-Quelle zum Lesen aus dem Quell-Stream. Der folgende Codeausschnitt erstellt die Kinesis-Quelle:

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

- Sie müssen die folgende Import-Anweisung hinzufügen.

```
import
 org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows;
```

- Die Anwendung verwendet eine Apache Flink S3-Senke, um auf Amazon S3 zu schreiben.

  Die Senke liest Nachrichten in einem rollierenden Fenster, kodiert Nachrichten in S3-Bucket-Objekte und sendet die codierten Objekte an die S3-Senke. Der folgende Code kodiert Objekte für das Senden an 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**
>
> Die Anwendung verwendet ein `StreamingFileSink`-Flink-Objekt, um in Amazon S3 zu schreiben. Weitere Informationen zu finden Sie `StreamingFileSink`unter StreamingFileSink in der Apache-Flink-Dokumentation.

Modifizieren Sie den Anwendungscode

In diesem Abschnitt ändern Sie den Anwendungscode, um die Ausgabe in Ihren Amazon S3-Bucket zu schreiben.

Aktualisieren Sie die folgende Zeile mit Ihrem Benutzernamen, um den Ausgabespeicherort der Anwendung anzugeben:

```
private static final String s3SinkPath = "s3a://ka-app-code-<username>/data";
```

Kompilieren des Anwendungscodes

Zum Kompilieren der Anwendung gehen Sie wie folgt vor:

1. Installieren Sie Java und Maven, wenn das noch nicht geschehen ist. Weitere Informationen finden Sie unter Voraussetzungen im Erste Schritte (DataStream API) Tutorial.
2. Kompilieren Sie die Anwendung mit dem folgenden Befehl:

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

Beim Kompilieren der Anwendung wird die JAR-Datei der Anwendung (`target/aws-kinesis-analytics-java-apps-1.0.jar`) erstellt.

> **Note**
>
> Der bereitgestellte Quellcode basiert auf Bibliotheken von Java 11.

Hochladen des Apache Flink-Streaming-Java-Codes

In diesem Abschnitt laden Sie Ihren Anwendungscode in den Amazon S3-Bucket hoch, den Sie im Erstellen Sie abhängige Ressourcen-Abschnitt erstellt haben.

1. Wählen Sie in der Amazon S3-Konsole den Bucket ka-app-code-***<username>*** aus, navigieren Sie zum Codeordner und wählen Sie Hochladen aus.
2. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `aws-kinesis-analytics-java-apps-1.0.jar`Datei, die Sie im vorherigen Schritt erstellt haben.
3. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink

2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.
   - Belassen Sie den Versions-Pulldown bei Apache Flink Version 1.15.2 (empfohlene Version).
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Geben Sie als Anwendungsname ein **`MyApplication`**.
> - Wählen Sie für Laufzeit die Option Apache Flink aus.
> - Belassen Sie die Version als Apache Flink Version 1.15.2 (empfohlene Version).

6. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
7. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen eines Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*

- Rolle: `kinesisanalytics-MyApplication-us-west-2`

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf den Kinesis Data Stream.

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **`kinesis-analytics-service-MyApplication-us-west-2`**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*012345678901*) mit Ihrer Konto-ID. Ersetzen Sie <username> durch Ihren Benutzernamen.

```
{
            "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"
        },

    ]
}
```

Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.

2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):
   - Geben Sie für Amazon-S3-Bucket **`ka-app-code-<username>`** ein.
   - Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **`code/aws-kinesis-analytics-java-apps-1.0.jar`** ein.
3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** erstellen/aktualisieren aus.
4. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
5. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
6. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`
>
> Dieser Protokollstream wird zur Überwachung der Anwendung verwendet. Dies ist nicht derselbe Protokollstream, den die Anwendung zum Senden von Ergebnissen verwendet.

Ausführen der Anwendung

1. Wählen Sie auf der MyApplication Seite Ausführen aus. Lassen Sie die Option Ohne Snapshot ausführen aktiviert und bestätigen Sie die Aktion.
2. Wenn die Anwendung ausgeführt wird, aktualisieren Sie die Seite. Die Konsole zeigt den Application graph (Anwendungs-Graph) an.

Überprüfen der Anwendungsausgabe

Öffnen Sie in der Amazon S3-Konsole den Ordner Daten in Ihrem S3-Bucket.

Nach einigen Minuten werden Objekte angezeigt, die aggregierte Daten aus der Anwendung enthalten.

> **Note**
>
> Die Aggregation ist in Flink standardmäßig aktiviert. Um sie deaktivieren, verwenden Sie Folgendes:
>
> ```
> sink.producer.aggregation-enabled' = 'false'
> ```

Optional: Passen Sie Quelle und Senke an

In diesem Abschnitt passen Sie die Einstellungen für die Quell- und Senkenobjekte an.

> **Note**
>
> Nachdem Sie die in den folgenden Abschnitten beschriebenen Codeabschnitte geändert haben, gehen Sie wie folgt vor, um den Anwendungscode neu zu laden:
>
> - Wiederholen Sie die Schritte im the section called “Kompilieren des Anwendungscodes”-Abschnitt, um den aktualisierten Anwendungscode zu kompilieren.
> - Wiederholen Sie die Schritte im the section called “Hochladen des Apache Flink-Streaming-Java-Codes”-Abschnitt, um den aktualisierten Anwendungscode hochzuladen.
> - Wählen Sie auf der Seite der Anwendung in der Konsole Konfigurieren und anschließend Aktualisieren aus, um den aktualisierten Anwendungscode erneut in Ihre Anwendung zu laden.

Dieser Abschnitt umfasst die folgenden Abschnitte:

- Konfigurieren der Datenpartitionierung
- Konfigurieren der Lesehäufigkeit
- Konfigurieren der Schreibpufferung

Konfigurieren der Datenpartitionierung

In diesem Abschnitt konfigurieren Sie die Namen der Ordner, die die Streaming-Dateisenke im S3-Bucket erstellt. Dies geschieht, indem Sie der Streaming-Dateisenke einen Bucket-Assigner hinzufügen.

Gehen Sie wie folgt vor, um die im S3-Bucket erstellten Ordnernamen anzupassen:

1. Fügen Sie am Anfang der `S3StreamingSinkJob.java`-Datei die folgenden Importanweisungen hinzu:

```
import
 org.apache.flink.streaming.api.functions.sink.filesystem.rollingpolicies.DefaultRollingPol
import
 org.apache.flink.streaming.api.functions.sink.filesystem.bucketassigners.DateTimeBucketAss
```

2. Aktualisieren Sie die `createS3SinkFromStaticConfig()`-Methode im Code so, dass sie wie folgt aussieht:

```
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;
}
```

Im vorherigen Codebeispiel wird `DateTimeBucketAssigner` mit einem benutzerdefinierten Datumsformat verwendet, um Ordner im S3-Bucket zu erstellen. Der `DateTimeBucketAssigner` verwendet die aktuelle Systemzeit, um Bucket-Namen zu erstellen. Wenn Sie einen benutzerdefinierten Bucket-Zuweiser erstellen möchten, um die erstellten Ordnernamen weiter anzupassen, können Sie eine Klasse erstellen, die implementiertBucketAssigner. Sie implementieren Ihre benutzerdefinierte Logik mithilfe der Methode `getBucketId`.

Eine benutzerdefinierte Implementierung von `BucketAssigner` kann den Kontext-Parameter verwenden, um weitere Informationen zu einem Datensatz abzurufen und seinen Zielordner zu bestimmen.

Konfigurieren der Lesehäufigkeit

In diesem Abschnitt konfigurieren Sie die Häufigkeit von Lesevorgängen im Quellstream.

Der Kinesis Streams-Consumer liest standardmäßig fünfmal pro Sekunde aus dem Quell-Stream. Diese Häufigkeit führt zu Problemen, wenn mehr als ein Client aus dem Stream liest oder wenn die Anwendung erneut versuchen muss, einen Datensatz zu lesen. Sie können diese Probleme vermeiden, indem Sie die Lesehäufigkeit des Benutzers festlegen.

Um die Lesehäufigkeit des Kinesis-Consumers festzulegen, legen Sie die Einstellung `SHARD_GETRECORDS_INTERVAL_MILLIS` fest.

Im folgenden Codebeispiel wird die `SHARD_GETRECORDS_INTERVAL_MILLIS`-Einstellung auf eine Sekunde festgelegt:

```
kinesisConsumerConfig.setProperty(ConsumerConfigConstants.SHARD_GETRECORDS_INTERVAL_MILLIS,
 "1000");
```

Konfigurieren der Schreibpufferung

In diesem Abschnitt konfigurieren Sie die Schreibfrequenz und andere Einstellungen der Senke.

Standardmäßig schreibt die Anwendung jede Minute in den Ziel-Bucket. Sie können dieses Intervall und andere Einstellungen ändern, indem Sie das `DefaultRollingPolicy`-Objekt konfigurieren.

> **Note**
>
> Die Apache Flink-Streaming-Dateisenke schreibt jedes Mal in ihren Ausgabe-Bucket, wenn die Anwendung einen Prüfpunkt erstellt. Die Anwendung erstellt standardmäßig jede Minute einen Prüfpunkt. Um das Schreibintervall der S3-Senke zu erhöhen, müssen Sie auch das Prüfpunkt-Intervall erhöhen.

Führen Sie zur Konfiguration des `DefaultRollingPolicy`-Objekts folgende Schritte aus:

1. Erhöhen Sie die `CheckpointInterval`-Einstellung der Anwendung. Die folgende Eingabe für die UpdateApplication Aktion legt das Checkpoint-Intervall auf 10 Minuten fest:

```
{
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "CheckpointConfigurationUpdate": {
            "ConfigurationTypeUpdate" : "CUSTOM",
            "CheckpointIntervalUpdate": 600000
         }
      }
   },
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 5
}
```

Geben Sie die aktuelle Anwendungsversion an, um den vorherigen Code zu verwenden. Sie können die Anwendungsversion mithilfe der -ListApplicationsAktion abrufen.

2. Fügen Sie am Anfang der `S3StreamingSinkJob.java`-Datei die folgende Importanweisung hinzu:

```
import java.util.concurrent.TimeUnit;
```

3. Aktualisieren Sie die `createS3SinkFromStaticConfig`-Methode in der `S3StreamingSinkJob.java`-Datei so, dass sie wie folgt aussieht:

```
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;
    }
```

Im vorherigen Codebeispiel wird die Häufigkeit von Schreibvorgängen in den Amazon S3-Bucket auf 8 Minuten festgelegt.

Weitere Informationen zur Konfiguration der Apache Flink-Streaming-Dateisenke finden Sie unter Reihencodierte Formate in der Apache Flink-Dokumentation.

Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zum Bereinigen von AWS-Ressourcen, die Sie im Amazon S3-Tutorial erstellt haben.

Dieses Thema enthält die folgenden Abschnitte:

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Ihres Kinesis Data Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen

Löschen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen Ihres Kinesis Data Streams

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Kinesis Data Streams die Option ausExampleInputStream.
3. Wählen Sie auf der ExampleInputStream Seite Kinesis Stream löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den ka-app-codeBucket -***<username>*** aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Richtlinien aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Rollen aus.
7. Wählen Sie die Rolle kinesis-analytics-MyApplication--us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie auf der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Tutorial: Verwenden einer Managed Service für Apache Flink-Anwendung zum Replizieren von Daten von einem Thema in einem MSK-Cluster zu einem anderen in einer VPC

Das folgende Tutorial zeigt, wie Sie eine Amazon VPC mit einem Amazon MSK-Cluster und zwei Themen erstellen und wie Sie eine Managed Service für Apache Flink-Anwendung erstellen, die aus einem Amazon MSK-Thema liest und in ein anderes schreibt.

> **Note**
>
> Um die erforderlichen Voraussetzungen für diese Übung einzurichten, schließen Sie zunächst die Erste Schritte (DataStream API)-Übung ab.

Dieses Tutorial enthält die folgenden Abschnitte:

- Erstellen einer Amazon VPC mit einem Amazon-MSK-Cluster
- Den Anwendungscode erstellen:
- Hochladen des Apache Flink-Streaming-Java-Codes
- Erstellen Sie die Anwendung
- Konfigurieren der Anwendung
- Ausführen der Anwendung
- Testen der Anwendung

Erstellen einer Amazon VPC mit einem Amazon-MSK-Cluster

Folgen Sie dem Tutorial Erste Schritte mit Amazon MSK, um eine Beispiel-VPC und Amazon MSK-Cluster für den Zugriff über eine Managed Service für Apache Flink-Anwendung zu erstellen.

Beachten Sie beim Abschluss des Tutorials Folgendes:

- Wiederholen Sie in Schritt 3: Thema erstellen den Befehl `kafka-topics.sh --create`, um ein Zielthema mit dem Namen `AWSKafkaTutorialTopicDestination` zu erstellen:

```
bin/kafka-topics.sh --create --zookeeper ZooKeeperConnectionString --replication-
factor 3 --partitions 1 --topic AWSKafkaTutorialTopicDestination
```

- Notieren Sie sich die Bootstrap-Serverliste für Ihren Cluster. Sie können die Liste der Bootstrap-Server mit dem folgenden Befehl abrufen (ersetzen Sie *ClusterArn* durch den ARN Ihres MSK-Clusters):

```
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"
```

```
}
```

- Wenn Sie den Schritten in den Tutorials folgen, achten Sie darauf, dass Sie die von Ihnen gewählte AWS-Region in Ihrem Code, Ihren Befehlen und Ihren Konsoleneinträgen verwenden.

Den Anwendungscode erstellen:

In diesem Abschnitt laden Sie die Anwendungs-JAR-Datei herunter und kompilieren sie. Wir empfehlen die Verwendung von Java 11.

Der Java-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben. Weitere Informationen finden Sie unter Git installieren.
2. Klonen Sie das Remote-Repository mit dem folgenden Befehl:

   ```
   git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
   ```

3. Der Anwendungscode befindet sich in der `amazon-kinesis-data-analytics-java-examples/KafkaConnectors/KafkaGettingStartedJob.java`-Datei. Sie können den Code untersuchen, um sich mit der Struktur des Anwendungscodes von Managed Service für Apache Flink vertraut zu machen.
4. Verwenden Sie entweder das Befehlszeilentool Maven oder Ihre bevorzugte Entwicklungsumgebung, um die JAR-Datei zu erstellen. Um die JAR-Datei mit dem Maven-Befehlszeilentool zu kompilieren, geben Sie Folgendes ein:

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

   Wenn der Build erfolgreich ist, wird die folgende Datei erstellt:

   ```
   target/KafkaGettingStartedJob-1.0.jar
   ```

   > **Note**
   >
   > Der bereitgestellte Quellcode basiert auf Bibliotheken von Java 11. Wenn Sie eine Entwicklungsumgebung verwenden,

Hochladen des Apache Flink-Streaming-Java-Codes

In diesem Abschnitt laden Sie Ihren Anwendungscode in den Amazon S3-Bucket hoch, den Sie im Erste Schritte (DataStream API) Tutorial erstellt haben.

> **Note**
>
> Wenn Sie den Amazon S3-Bucket aus dem Tutorial Erste Schritte gelöscht haben, führen Sie den Schritt the section called “Hochladen des Apache Flink-Streaming-Java-Codes” erneut aus.

1. Wählen Sie in der Amazon S3-Konsole den Bucket ka-app-code-***<username>*** und dann Hochladen aus.
2. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `KafkaGettingStartedJob-1.0.jar`Datei, die Sie im vorherigen Schritt erstellt haben.
3. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Wählen Sie für Laufzeit die Option Apache Flink 1.15.2 aus.
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-MyApplication-us-west-2`
> - Rolle: `kinesisanalytics-MyApplication-us-west-2`

Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):
   - Geben Sie für Amazon-S3-Bucket **`ka-app-code-<username>`** ein.
   - Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **`KafkaGettingStartedJob-1.0.jar`** ein.
3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** erstellen/aktualisieren aus.

   > **Note**
   >
   > Wenn Sie Anwendungsressourcen über die Konsole angeben (z. B. CloudWatch Protokolle oder eine Amazon VPC), ändert die Konsole Ihre Anwendungsausführungsrolle, um die Berechtigung für den Zugriff auf diese Ressourcen zu erteilen.

4. Wählen Sie unter Eigenschaften die Option Gruppe hinzufügen aus. Geben Sie die folgenden Eigenschaften ein:

Gruppen-ID	Schlüssel	Wert
`KafkaSource`	Thema	AWSKafkaTutorialTopic

Gruppen-ID	Schlüssel	Wert
KafkaSource	bootstrap.servers	*Die Bootstrap-Serverliste, die Sie zuvor gespeichert haben*
KafkaSource	security.protocol	SSL
KafkaSource	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSource	ssl.truststore.passwort	changeit

> **Note**
>
> Das ssl.truststore.password für das Standardzertifikat ist „changeit“. Sie müssen diesen Wert nicht ändern, wenn Sie das Standardzertifikat verwenden.

Wählen Sie erneut Gruppe hinzufügen. Geben Sie die folgenden Eigenschaften ein:

Gruppen-ID	Schlüssel	Wert
KafkaSink	Thema	AWSKafkaTutorialTopicDestination
KafkaSink	bootstrap.servers	*Die Bootstrap-Serverliste, die Sie zuvor gespeichert haben*
KafkaSink	security.protocol	SSL
KafkaSink	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts

Gruppen-ID	Schlüssel	Wert
KafkaSink	ssl.truststore.passwort	changeit
KafkaSink	transaction.timeout.ms	1000

Der Anwendungscode liest die oben genannten Anwendungseigenschaften, um die Quelle und Senke zu konfigurieren, die für die Interaktion mit Ihrer VPC und Ihrem Amazon MSK-Cluster verwendet werden. Weitere Informationen zur Verwendung von Eigenschaften finden Sie unter Laufzeiteigenschaften.

5. Wählen Sie unter Snapshots die Option Deaktivieren aus. Dadurch wird es einfacher, die Anwendung zu aktualisieren, ohne ungültige Anwendungsstatusdaten zu laden.
6. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
7. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
8. Wählen Sie im Abschnitt Virtual Private Cloud (VPC) die VPC aus, die mit Ihrer Anwendung verknüpft werden soll. Wählen Sie die mit Ihrer VPC verknüpften Subnetze und Sicherheitsgruppe aus, die die Anwendung für den Zugriff auf VPC-Ressourcen verwenden soll.
9. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`
>
> Dieser Protokollstream wird zur Überwachung der Anwendung verwendet.

Ausführen der Anwendung

Das Flink-Jobdiagramm kann angezeigt werden, indem Sie die Anwendung ausführen, das Apache Flink-Dashboard öffnen und den gewünschten Flink-Job auswählen.

Testen der Anwendung

In diesem Abschnitt schreiben Sie Datensätze zum Quellthema. Die Anwendung liest Datensätze aus dem Quellthema und schreibt sie in das Zielthema. Sie überprüfen, ob die Anwendung funktioniert, indem Sie Datensätze in das Quellthema schreiben und Datensätze aus dem Zielthema lesen.

Um Datensätze aus den Themen zu schreiben und zu lesen, folgen Sie den Schritten in Schritt 6: Daten produzieren und verwenden im Tutorial Erste Schritte mit Amazon MSK.

Um aus dem Zielthema zu lesen, verwenden Sie in Ihrer zweiten Verbindung zum Cluster den Namen des Zielthemas anstelle des Quellthemas:

```
bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString --
consumer.config client.properties --topic AWSKafkaTutorialTopicDestination --from-
beginning
```

Wenn im Zielthema keine Datensätze angezeigt werden, lesen Sie den Abschnitt Kein Zugriff auf Ressourcen in einer VPC möglich im Thema Fehlerbehebung.

Beispiel: Verwenden Sie einen EFO-Verbraucher mit einem Kinesis Data Stream

In dieser Übung erstellen Sie eine Anwendung von Managed Service für Apache Flink, die mithilfe eines Enhanced Fan-Out-(EFO)-Verbrauchers aus einem Kinesis Data Stream liest. Wenn ein Kinesis-Verbraucher EFO verwendet, stellt ihm der Kinesis Data Streams-Service seine eigene dedizierte Bandbreite zur Verfügung, anstatt dass der Verbraucher die feste Bandbreite des Streams mit den anderen Verbrauchern teilt, die aus dem Stream lesen.

Weitere Informationen zur Verwendung von EFO mit dem Kinesis Consumer finden Sie unter FLIP-128: Verbesserte Verteilung für Kinesis-Verbraucher.

Die Anwendung, die Sie in diesem Beispiel erstellen, verwendet AWS Kinesis Connector (flink-connector-kinesis) 1.15.3.

> **Note**
>
> Um die erforderlichen Voraussetzungen für diese Übung einzurichten, schließen Sie zunächst die Erste Schritte (DataStream API)-Übung ab.

Dieses Thema enthält die folgenden Abschnitte:

- Erstellen Sie abhängige Ressourcen
- Schreiben Sie Beispieldatensätze in den Eingabe-Stream
- Laden Sie den Anwendungscode herunter und überprüfen Sie ihn
- Kompilieren des Anwendungscodes
- Hochladen des Apache Flink-Streaming-Java-Codes
- Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus
- Bereinigen von AWS-Ressourcen

Erstellen Sie abhängige Ressourcen

Bevor Sie für diese Übung eine Anwendung von Managed Service für Apache Flink erstellen, erstellen Sie die folgenden abhängigen Ressourcen:

- Zwei Kinesis Data Streams (`ExampleInputStream` und `ExampleOutputStream`)
- Einen Amazon S3-Bucket zum Speichern des Anwendungscodes (`ka-app-code-<username>`)

Sie können die Kinesis Streams und den Amazon-S3-Bucket mithilfe der Konsole erstellen. Anweisungen zum Erstellen dieser Ressourcen finden Sie in den folgenden Themen:

- Data Streams erstellen und aktualisieren im Amazon Kinesis Data Streams Entwicklerleitfaden. Benennen Sie Ihren Data Stream **`ExampleInputStream`** und **`ExampleOutputStream`**.
- Wie erstelle ich einen S3-Bucket? im Amazon Simple Storage Service Benutzerhandbuch. Geben Sie dem Amazon S3-Bucket einen global eindeutigen Namen, indem Sie Ihren Anmeldenamen anhängen, z. B. **`ka-app-code-<username>`**.

Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Abschnitt verwenden Sie ein Python-Skript zum Schreiben von Datensätzen in den Stream für die zu verarbeitende Anwendung.

> **Note**
>
> Dieser Abschnitt erfordert AWS SDK for Python (Boto).

1. Erstellen Sie eine Datei `stock.py` mit dem folgenden Inhalt:

```
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. Führen Sie das `stock.py`Skript aus:

```
$ python stock.py
```

Lassen Sie das Skript laufen, während Sie den Rest des Tutorials abschließen.

Laden Sie den Anwendungscode herunter und überprüfen Sie ihn

Der Java-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben. Weitere Informationen finden Sie unter Git installieren.
2. Klonen Sie das Remote-Repository mit dem folgenden Befehl:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/EfoConsumer` Verzeichnis .

Der Anwendungscode befindet sich in der `EfoApplication.java`-Datei. Beachten Sie Folgendes zum Anwendungscode:

- Sie aktivieren den EFO-Consumer, indem Sie die folgenden Parameter für den Kinesis-Consumer festlegen:
  - RECORD_PUBLISHER_TYPE: Setzen Sie diesen Parameter auf EFO, damit Ihre Anwendung einen EFO-Consumer für den Zugriff auf die Kinesis Data Stream-Daten verwendet.
  - EFO_CONSUMER_NAME: Setzen Sie diesen Parameter auf einen Zeichenfolgenwert, der unter den Verbrauchern dieses Streams eindeutig ist. Die Wiederverwendung eines Verbrauchernamens in demselben Kinesis Data Stream führt dazu, dass der vorherige Verbraucher, der diesen Namen verwendet hat, beendet wird.
- Das folgende Codebeispiel zeigt, wie den Consumer-Konfigurationseigenschaften Werte zugewiesen werden, um einen EFO-Consumer zum Lesen aus dem Quell-Stream zu verwenden:

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

Kompilieren des Anwendungscodes

Zum Kompilieren der Anwendung gehen Sie wie folgt vor:

1. Installieren Sie Java und Maven, wenn das noch nicht geschehen ist. Weitere Informationen finden Sie unter Voraussetzungen im Erste Schritte (DataStream API) Tutorial.
2. Kompilieren Sie die Anwendung mit dem folgenden Befehl:

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

> **Note**
>
> Der bereitgestellte Quellcode basiert auf Bibliotheken von Java 11.

Beim Kompilieren der Anwendung wird die JAR-Datei der Anwendung (`target/aws-kinesis-analytics-java-apps-1.0.jar`) erstellt.

Hochladen des Apache Flink-Streaming-Java-Codes

In diesem Abschnitt laden Sie Ihren Anwendungscode in den Amazon S3-Bucket hoch, den Sie im Erstellen Sie abhängige Ressourcen-Abschnitt erstellt haben.

1. Wählen Sie in der Amazon S3-Konsole den Bucket ka-app-code-***<username>*** und dann Upload aus.
2. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `aws-kinesis-analytics-java-apps-1.0.jar`Datei, die Sie im vorherigen Schritt erstellt haben.
3. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.

     > **Note**
     >
     > Managed Service für Apache Flink verwendet Apache Flink Version 1.15.2.

   - Belassen Sie den Versions-Pulldown bei Apache Flink Version 1.15.2 (empfohlene Version).
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rolle: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf die Kinesis-Datenströme.

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **`kinesis-analytics-service-MyApplication-us-west-2`**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*012345678901*) mit Ihrer Konto-ID.

> **Note**
>
> Diese Berechtigungen gewähren der Anwendung die Möglichkeit, auf den EFO-Consumer zuzugreifen.

```
{
    "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:*"
            ]
        }
    ]
}
```

Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):
   - Geben Sie für Amazon-S3-Bucket **ka-app-code-*<username>*** ein.
   - Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **aws-kinesis-analytics-java-apps-1.0.jar** ein.
3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **kinesis-analytics-MyApplication-us-west-2** erstellen/aktualisieren aus.
4. Wählen Sie unter Eigenschaften die Option Gruppe erstellen aus.
5. Geben Sie die folgenden Eigenschaften und Werte der Anwendung ein:

Gruppen-ID	Schlüssel	Wert
ConsumerConfigProperties	**flink.stream.recordpublisher**	**EFO**
ConsumerConfigProperties	**flink.stream.efo.consumername**	**basic-efo-flink-app**

Gruppen-ID	Schlüssel	Wert
ConsumerConfigProperties	**INPUT_STREAM**	**ExampleInputStream**
ConsumerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ConsumerConfigProperties	**AWS_REGION**	**us-west-2**

6. Wählen Sie unter Eigenschaften die Option Gruppe erstellen aus.
7. Geben Sie die folgenden Eigenschaften und Werte der Anwendung ein:

Gruppen-ID	Schlüssel	Wert
ProducerConfigProperties	**OUTPUT_STREAM**	**ExampleOutputStream**
ProducerConfigProperties	**AWS_REGION**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

8. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
9. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
10. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`

Dieser Protokollstream wird zur Überwachung der Anwendung verwendet. Dies ist nicht derselbe Protokollstream, den die Anwendung zum Senden von Ergebnissen verwendet.

Ausführen der Anwendung

Das Flink-Jobdiagramm kann angezeigt werden, indem Sie die Anwendung ausführen, das Apache Flink-Dashboard öffnen und den gewünschten Flink-Job auswählen.

Sie können die Metriken von Managed Service für Apache Flink in der CloudWatch Konsole überprüfen, um zu überprüfen, ob die Anwendung funktioniert.

Sie können auch die Konsole von Kinesis Data Streams auf der Registerkarte Erweitertes Rundsenden des Datenstroms nach dem Namen Ihres Verbrauchers () suchenbasic-efo-flink-app.

Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zur Bereinigung von AWS-Ressourcen, die im efo-Fenster-Tutorial erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte:

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Ihrer Kinesis Data Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen

Löschen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen Ihrer Kinesis Data Streams

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Kinesis Data Streams die Option ausExampleInputStream.
3. Wählen Sie auf der ExampleInputStream Seite Kinesis Stream löschen aus und bestätigen Sie dann den Löschvorgang.
4. Wählen Sie auf der Seite Kinesis Streams die , ExampleOutputStreamwählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den ka-app-codeBucket -***<username>*** aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle kinesis-analytics--MyApplication-us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Beispiel: Schreiben in Kinesis Data Firehose

In dieser Übung erstellen Sie eine Managed Service für Apache Flink-Anwendung, die einen Kinesis Data Stream als Quelle und einen Kinesis Data Firehose-Bereitstellungs-Stream als Senke hat. Mithilfe der Senke können Sie die Ausgabe der Anwendung in einem Amazon-S3-Bucket überprüfen.

> **Note**
>
> Um die erforderlichen Voraussetzungen für diese Übung einzurichten, schließen Sie zunächst die Erste Schritte (DataStream API)-Übung ab.

Dieser Abschnitt enthält die folgenden Schritte:

- Erstellen Sie abhängige Ressourcen
- Schreiben Sie Beispieldatensätze in den Eingabe-Stream
- Herunterladen und Überprüfen des Apache Flink-Streaming-Java-Codes
- Kompilieren des Anwendungscodes
- Hochladen des Apache Flink-Streaming-Java-Codes
- Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus
- Bereinigen von AWS-Ressourcen

Erstellen Sie abhängige Ressourcen

Bevor Sie für diese Übung einen Managed Service für Apache Flink erstellen, erstellen Sie die folgenden abhängigen Ressourcen:

- Einen Kinesis Data Stream (`ExampleInputStream`)
- Einen Kinesis Data Firehose-Stream, in den die Anwendung die Ausgabe schreibt (`ExampleDeliveryStream`).
- Einen Amazon S3-Bucket zum Speichern des Codes der Anwendung (`ka-app-code-<username>`)

Sie können den Kinesis-Stream, die Amazon S3-Buckets und den Kinesis Data Firehose-Stream mithilfe der Konsole erstellen. Anweisungen zum Erstellen dieser Ressourcen finden Sie in den folgenden Themen:

- Data Streams erstellen und aktualisieren im Amazon Kinesis Data Streams Entwicklerleitfaden. Benennen Sie Ihren Data Stream **`ExampleInputStream`**.
- Erstellen eines Amazon Kinesis Data Firehose-Bereitstellungs-Streams im Amazon Kinesis Data Firehose Entwicklerhandbuch. Benennen Sie Ihren Kinesis Data Firehose-Stream **`ExampleDeliveryStream`**. Wenn Sie den Kinesis Data Firehose-Stream erstellen, erstellen Sie auch das S3-Ziel und die IAM-Rolle des Kinesis Data Firehose-Streams.
- Wie erstelle ich einen S3-Bucket? im Amazon Simple Storage Service Benutzerhandbuch. Geben Sie dem Amazon S3-Bucket einen global eindeutigen Namen, indem Sie Ihren Anmeldenamen anhängen, z. B. **`ka-app-code-<username>`**.

Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Abschnitt verwenden Sie ein Python-Skript zum Schreiben von Datensätzen in den Stream für die zu verarbeitende Anwendung.

> **Note**
>
> Dieser Abschnitt erfordert AWS SDK for Python (Boto).

1. Erstellen Sie eine Datei `stock.py` mit dem folgenden Inhalt:

```
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. Führen Sie das `stock.py`Skript aus:

```
$ python stock.py
```

   Lassen Sie das Skript laufen, während Sie den Rest des Tutorials abschließen.

Herunterladen und Überprüfen des Apache Flink-Streaming-Java-Codes

Der Java-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Klonen Sie das Remote-Repository mit dem folgenden Befehl:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/FirehoseSink` Verzeichnis .

Der Anwendungscode befindet sich in der `FirehoseSinkStreamingJob.java`-Datei. Beachten Sie Folgendes zum Anwendungscode:

- Die Anwendung verwendet eine Kinesis-Quelle zum Lesen aus dem Quell-Stream. Der folgende Codeausschnitt erstellt die Kinesis-Quelle:

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

- Die Anwendung verwendet eine Kinesis Data Firehose-Senke, um Daten in einen Kinesis Data Firehose-Stream zu schreiben. Der folgende Codeausschnitt erstellt die Kinesis Date Firehose-Senke:

```
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();
    }
```

Kompilieren des Anwendungscodes

Zum Kompilieren der Anwendung gehen Sie wie folgt vor:

1. Installieren Sie Java und Maven, wenn das noch nicht geschehen ist. Weitere Informationen finden Sie unter Voraussetzungen im Erste Schritte (DataStream API) Tutorial.
2. Um den Kinesis-Konnektor für die folgende Anwendung verwenden zu können, müssen Sie Apache Maven herunterladen, erstellen und installieren. Weitere Informationen finden Sie unter the section called “Verwenden des Apache Flink Kinesis Streams-Konnektor mit früheren Apache Flink-Versionen”.
3. Kompilieren Sie die Anwendung mit dem folgenden Befehl:

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

> **Note**
>
> Der bereitgestellte Quellcode basiert auf Bibliotheken von Java 11.

Beim Kompilieren der Anwendung wird die JAR-Datei der Anwendung (`target/aws-kinesis-analytics-java-apps-1.0.jar`) erstellt.

Hochladen des Apache Flink-Streaming-Java-Codes

In diesem Abschnitt laden Sie Ihren Anwendungscode in den Amazon S3-Bucket hoch, den Sie im Abschnitt Erstellen Sie abhängige Ressourcen erstellt haben.

So laden Sie den Anwendungscode hoch

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie in der -Konsole den Bucket ka-app-code-***<username>*** und dann Hochladen aus.
3. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `java-getting-started-1.0.jar`Datei, die Sie im vorherigen Schritt erstellt haben.
4. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus

Sie können eine Anwendung von Managed Service für Apache Flink entweder über die Konsole oder die AWS CLI erstellen und ausführen.

> **Note**
>
> Wenn Sie die Anwendung mit der Konsole erstellen, werden Ihre AWS Identity and Access Management (IAM)- und Amazon- CloudWatch Logs-Ressourcen für Sie erstellt. Wenn Sie die Anwendung mit der AWS CLI erstellen, erstellen Sie diese Ressourcen separat.

Themen

- Erstellen und Ausführen der Anwendung (Konsole)
- Erstellen und Ausführen der Anwendung (AWS CLI)

Erstellen und Ausführen der Anwendung (Konsole)

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink

2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:

   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Geben Sie für Beschreibung den Text **`My java test app`** ein.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.

   > **Note**
   >
   > Managed Service für Apache Flink verwendet Apache Flink Version 1.15.2.

   - Belassen Sie den Versions-Pulldown bei Apache Flink Version 1.15.2 (empfohlene Version).
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen der Anwendung mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Die Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rolle: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf den Kinesis Data Stream und den Kinesis Data Firehose-Stream.

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **`kinesis-analytics-service-MyApplication-us-west-2`**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.

3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie alle Instances der beispielhaften Konto-IDs (*012345678901*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):
   - Geben Sie für Amazon-S3-Bucket **ka-app-code-*<username>*** ein.
   - Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **java-getting-started-1.0.jar** ein.
3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **kinesis-analytics-MyApplication-us-west-2** erstellen/aktualisieren aus.
4. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
5. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.

6. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`

Ausführen der Anwendung

Das Flink-Jobdiagramm kann angezeigt werden, indem Sie die Anwendung ausführen, das Apache Flink-Dashboard öffnen und den gewünschten Flink-Job auswählen.

Stoppen der Anwendung

Wählen Sie auf der MyApplication Seite Stoppen aus. Bestätigen Sie die Aktion.

Aktualisieren der Anwendung

Mithilfe der Konsole können Sie Anwendungseinstellungen wie beispielsweise Anwendungseigenschaften, Überwachungseinstellungen und den Speicherort oder den Dateinamen der JAR-Anwendungsdatei aktualisieren.

Wählen Sie auf der MyApplication Seite Konfigurieren aus. Aktualisieren Sie die Anwendungseinstellungen und klicken Sie auf Aktualisieren.

> **Note**
>
> Um den Code der Anwendung auf der Konsole zu aktualisieren, müssen Sie entweder den Objektnamen der JAR ändern, einen anderen S3-Bucket verwenden oder den AWS CLI wie im Abschnitt beschrieben verwenden. the section called “Aktualisieren Sie den Anwendungscode” Wenn sich der Dateiname oder der Bucket nicht ändert, wird der Anwendungscode nicht neu geladen, wenn Sie auf der Seite „Konfigurieren“ die Option „Aktualisieren“ wählen.

Erstellen und Ausführen der Anwendung (AWS CLI)

In diesem Abschnitt verwenden Sie AWS CLI, um die Anwendung Managed Service für Apache Flink zu erstellen und auszuführen.

Erstellen einer Berechtigungsrichtlinie

Zuerst erstellen Sie eine Berechtigungsrichtlinie mit zwei Anweisungen: eine, die Berechtigungen für die `read`-Aktion auf den Quell-Stream zulässt, und eine andere, die Berechtigungen für die `write`-Aktionen auf den Senken-Stream zulässt. Anschließend fügen Sie die Richtlinie an eine IAM-Rolle (die Sie im nächsten Abschnitt erstellen) an. Wenn Managed Service für Apache Flink also die Rolle übernimmt, verfügt der Service über die erforderlichen Berechtigungen zum Lesen aus dem Quell-Stream und zum Schreiben in den Senken-Stream.

Verwenden Sie den folgenden Code zum Erstellen der `AKReadSourceStreamWriteSinkStream`-Berechtigungsrichtlinie. Ersetzen Sie *`username`* durch den Benutzernamen, den Sie verwenden werden, um den Amazon-S3-Bucket zum Speichern des Anwendungscodes zu erstellen. Ersetzen Sie die Konto-ID in den Amazon-Ressourcennamen (ARNs) (*`012345678901`*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

step-by-step Anweisungen zum Erstellen einer Berechtigungsrichtlinie finden Sie unter Tutorial: Erstellen und Anfügen Ihrer ersten vom Kunden verwalteten Richtlinie im IAM-Benutzerhandbuch.

> **Note**
>
> Um auf andere Amazon-Services zuzugreifen, können Sie AWS SDK for Java verwenden. Managed Service für Apache Flink setzt die vom SDK benötigten Anmeldeinformationen automatisch auf die der IAM-Rolle für die Dienstausführung, die mit Ihrer Anwendung verknüpft ist. Es sind keine weiteren Schritte erforderlich.

Erstellen einer IAM-Rolle

In diesem Abschnitt erstellen Sie eine IAM-Rolle, die die Anwendung von Managed Service für Apache Flink annehmen kann, um einen Quell-Stream zu lesen und in den Senken-Stream zu schreiben.

Managed Service für Apache Flink kann ohne Berechtigungen nicht auf Ihren Stream zugreifen. Sie erteilen diese Berechtigungen über eine IAM-Rolle. Jeder IAM-Rolle sind zwei Richtlinien angefügt. Die Vertrauensrichtlinie erteilt Managed Service für Apache Flink die Berechtigung zum Übernehmen der Rolle. Die Berechtigungsrichtlinie bestimmt, was Managed Service für Apache Flink nach Annahme der Rolle tun kann.

Sie können die Berechtigungsrichtlinie, die Sie im vorherigen Abschnitt erstellt haben, dieser Rolle anfügen.

So erstellen Sie eine IAM-Rolle

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Roles (Rollen) und Create Role (Rolle erstellen) aus.
3. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option AWS-Service aus. Wählen Sie unter Choose the service that will use this role (Wählen Sie den Service aus, der

diese Rolle verwendet) die Option Kinesis aus. Wählen Sie unter Select your use case (Wählen Sie Ihren Anwendungsfall aus) die Option Kinesis Analytics aus.

Wählen Sie Weiter: Berechtigungen aus.

4. Wählen Sie auf der Seite Attach permissions policies (Berechtigungsrichtlinien hinzufügen) Next: Review (Weiter: Überprüfen) aus. Sie fügen Berechtigungsrichtlinien an, nachdem Sie die Rolle erstellt haben.
5. Geben Sie auf der Seite Create role (Rolle erstellen) den Text **`MF-stream-rw-role`** für Role name (Rollenname) ein. Wählen Sie Rolle erstellen aus.

   Jetzt haben Sie eine neue IAM-Rolle mit dem Namen `MF-stream-rw-role` erstellt. Im nächsten Schritt aktualisieren Sie die Vertrauens- und Berechtigungsrichtlinien für die Rolle.
6. Fügen Sie die Berechtigungsrichtlinie der Rolle an.

   > **Note**
   >
   > Für diese Übung übernimmt Managed Service für Apache Flink diese Rolle sowohl für das Lesen von Daten aus einem Kinesis-Datenstrom (Quelle) als auch zum Schreiben der Ausgabedaten in einen anderen Kinesis-Datenstrom. Daher fügen Sie die Richtlinie an, die Sie im vorherigen Schritt erstellt haben, the section called “Erstellen einer Berechtigungsrichtlinie”.

   a. Wählen Sie auf der Seite Summary (Übersicht) die Registerkarte Permissions (Berechtigungen) aus.
   b. Wählen Sie Attach Policies (Richtlinien anfügen) aus.
   c. Geben Sie im Suchfeld **`AKReadSourceStreamWriteSinkStream`** (die Richtlinie, die Sie im vorhergehenden Abschnitt erstellt haben) ein.
   d. Wählen Sie die AKReadSourceStreamWriteSinkStream-Richtlinie und dann Richtlinie anfügen aus.

Sie haben nun die Service-Ausführungsrolle erstellt, die Ihre Anwendung für den Zugriff auf Ressourcen verwenden wird. Notieren Sie sich den ARN der neuen Rolle.

step-by-step Anweisungen zum Erstellen einer Rolle finden Sie unter Erstellen einer IAM-Rolle (Konsole) im IAM-Benutzerhandbuch.

Erstellen Sie die Anwendung Managed Service für Apache Flink

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `create_request.json`. Ersetzen Sie den Beispiel-Rollen-ARN durch den ARN für die Rolle, die Sie zuvor erstellt haben. Ersetzen Sie das Bucket-ARN-Suffix mit dem Suffix, das Sie im the section called “Erstellen Sie abhängige Ressourcen”-Abschnitt (`ka-app-code-<username>`) gewählt haben. Ersetzen Sie die beispielhafte Konto-ID (*012345678901*) in der Service-Ausführungsrolle mit Ihrer Konto-ID.

```
{
    "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. Führen Sie die `CreateApplication`-Aktion mit der vorherigen Anforderung zum Erstellen der Anwendung aus:

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

Die Anwendung wird nun erstellt. Sie starten die Anwendung im nächsten Schritt.

Starten der Anwendung

In diesem Abschnitt verwenden Sie die `StartApplication`-Aktion, um die Anwendung zu starten.

So starten Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `start_request.json`.

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

2. Führen Sie die `StartApplication`-Aktion mit der vorherigen Anforderung zum Starten der Anwendung aus:

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

Die Anwendung wird jetzt ausgeführt. Sie können die Metriken von Managed Service für Apache Flink in der Amazon- CloudWatch Konsole überprüfen, um zu überprüfen, ob die Anwendung funktioniert.

Stoppen der Anwendung

In diesem Abschnitt verwenden Sie die `StopApplication`-Aktion, um die Anwendung zu stoppen.

So stoppen Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `stop_request.json`.

```
{
    "ApplicationName": "test"
}
```

2. Führen Sie die `StopApplication`-Aktion mit der folgenden Anforderung zum Stoppen der Anwendung aus:

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

Die Anwendung wird nun gestoppt.

Hinzufügen einer CloudWatch Protokollierungsoption

Sie können die verwendenAWS CLI, um Ihrer Anwendung einen Amazon- CloudWatch Protokollstream hinzuzufügen. Informationen zur Verwendung von - CloudWatch Protokollen mit Ihrer Anwendung finden Sie unter the section called “Einrichten der Protokollierung”.

Aktualisieren Sie den Anwendungscode

Wenn Sie Ihren Anwendungscode mit einer neuen Version Ihres Codepakets aktualisieren müssen, verwenden Sie die `UpdateApplication` AWS CLI-Aktion.

Um die AWS CLI zu verwenden, löschen Sie Ihr vorheriges Codepaket aus Ihrem Amazon S3-Bucket, laden Sie die neue Version hoch und rufen Sie `UpdateApplication` auf, wobei Sie denselben Amazon S3-Bucket und Objektnamen angeben.

Die folgende Beispielanforderung für die `UpdateApplication`-Aktion lädt den Anwendungscode neu und startet die Anwendung neu. Aktualisieren Sie die `CurrentApplicationVersionId` auf die aktuelle Anwendungsversion. Sie können die aktuelle Anwendungsversion mithilfe der Aktionen `ListApplications` oder `DescribeApplication` überprüfen. Aktualisieren Sie das Bucket-Namenssuffix (**<*username*>**) mit dem Suffix, das Sie im Abschnitt the section called “Erstellen Sie abhängige Ressourcen” ausgewählt haben.

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "java-getting-started-1.0.jar"
                }
            }
        }
    }
}
```

Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zur Bereinigung von AWS-Ressourcen, die im Erste Schritte-Tutorial erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte:

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Ihres Kinesis Data Streams
- Löschen Ihres Kinesis Data Firehose-Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen

Löschen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie Konfigurieren aus.
4. Wählen Sie im Abschnitt Snapshots die Option Deaktivieren und anschließend Aktualisieren.
5. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie den Löschvorgang.

Löschen Ihres Kinesis Data Streams

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Kinesis Data Streams die Option ausExampleInputStream.
3. Wählen Sie auf der ExampleInputStream Seite Kinesis Stream löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen Ihres Kinesis Data Firehose-Streams

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Kinesis Data Firehose die Option ausExampleDeliveryStream.

3. Wählen Sie auf der ExampleDeliveryStream Seite Kinesis-Data-Firehose-Stream löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den ka-app-codeBucket -*<username>* aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.
4. Wenn Sie einen Amazon S3-Bucket für das Ziel Ihres Kinesis Data Firehose-Streams erstellt haben, löschen Sie auch diesen Bucket.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wenn Sie eine neue Richtlinie für Ihren Kinesis Data Firehose-Stream erstellt haben, löschen Sie auch diese Richtlinie.
7. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
8. Wählen Sie die Rolle kinesis-analytics-MyApplication--us-west-2 aus.
9. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.
10. Wenn Sie eine neue Rolle für Ihren Kinesis Data Firehose-Stream erstellt haben, löschen Sie auch diese Rolle.

Löschen Ihrer CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Beispiel: Aus einem Kinesis Stream in einem anderen Konto lesen

Dieses Beispiel zeigt, wie Sie eine Managed Service für Apache Flink-Anwendung erstellen, die Daten aus einem Kinesis Stream in einem anderen Konto liest. In diesem Beispiel verwenden Sie ein Konto für den Kinesis-Quellstream und ein zweites Konto für die Anwendung Managed Service für Apache Flink und den Senk-Kinesis-Stream.

Dieses Thema enthält die folgenden Abschnitte:

- Voraussetzungen
- Aufstellen
- Kinesis-Quellstream erstellen
- IAM-Rollen und -Richtlinien erstellen und aktualisieren
- Aktualisieren Sie das Python-Skript
- Aktualisieren der Java-Anwendung
- Erstellen Sie die Anwendung, laden Sie sie hoch und führen Sie sie aus.

Voraussetzungen

- In diesem Tutorial ändern Sie das Erste Schritte-Beispiel, um Daten aus einem Kinesis Stream in einem anderen Konto zu lesen. Schließen Sie das Erste Schritte (DataStream API)-Tutorial ab, bevor Sie fortfahren.
- Sie benötigen zwei AWS-Konten, um dieses Tutorial abzuschließen: eines für den Quellstream und eines für die Anwendung und den Senken-Stream. Verwenden Sie das AWS-Konto, das Sie für das Tutorial Erste Schritte verwendet haben, für die Anwendung und den Senken-Stream. Verwenden Sie ein anderes AWS-Konto für den Quellstream.

Aufstellen

Sie greifen mithilfe von benannten Profilen auf Ihre beiden AWS-Konten zu. Ändern Sie Ihre AWS-Anmeldeinformationen und Konfigurationsdateien so, dass sie zwei Profile enthalten, die die Regions- und Verbindungsinformationen für Ihre beiden Konten enthalten.

Die folgende Beispieldatei mit Anmeldeinformationen enthält zwei benannte Profile, `ka-source-stream-account-profile` und `ka-sink-stream-account-profile`. Verwenden Sie das Konto, das Sie für das Tutorial Erste Schritte verwendet haben, für das Senken-Stream-Konto.

```
[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
```

Die folgende Beispielkonfigurationsdatei enthält die gleichen benannten Profile mit Regions- und Ausgabeformatinformationen.

```
[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**
>
> In diesem Tutorial wird das `ka-sink-stream-account-profile` nicht verwendet. Es ist als Beispiel für den Zugriff auf zwei verschiedene AWS-Konten mit Profilen enthalten.

Weitere Informationen zur Verwendung von benannten Profilen mit der AWS CLI finden Sie unter Benannte Profile im der AWS Command Line Interface-Dokumentation.

Kinesis-Quellstream erstellen

In diesem Abschnitt erstellen Sie den Kinesis Stream im Quellkonto.

Geben Sie den folgenden Befehl ein, um den Kinesis Stream zu erstellen, den die Anwendung für die Eingabe verwendet. Beachten Sie, dass der `--profile`-Parameter angibt, welches Kontoprofil verwendet werden soll.

```
$ aws kinesis create-stream \
--stream-name SourceAccountExampleInputStream \
--shard-count 1 \
```

```
--profile ka-source-stream-account-profile
```

IAM-Rollen und -Richtlinien erstellen und aktualisieren

Um den AWS-kontenübergreifenden Zugriff auf Objekte zu ermöglichen, erstellen Sie eine IAM-Rolle und -Richtlinie im Quellkonto. Anschließend ändern Sie die IAM-Richtlinie im Senkenkonto. Weitere Informationen zum Erstellen und Verwalten von IAM-Rollen und -Richtlinien finden Sie in den folgenden Themen im AWS Identity and Access Management-Benutzerhandbuch:

- Erstellen von IAM-Rollen
- Erstellen von IAM-Richtlinien

Rollen und Richtlinien für Senken-Konten

1. Bearbeiten Sie die Richtlinie `kinesis-analytics-service-MyApplication-us-west-2` aus dem Tutorial Erste Schritte. Mit dieser Richtlinie kann die Rolle im Quellkonto übernommen werden, um den Quellstream zu lesen.

   > **Note**
   >
   > Wenn Sie die Konsole verwenden, um Ihre Anwendung zu erstellen, erstellt die Konsole eine Richtlinie mit dem Namen `kinesis-analytics-service-<application name>-<application region>` und eine Rolle namens `kinesisanalytics-<application name>-<application region>`.

   Fügen Sie der Richtlinie den unten hervorgehobenen Abschnitt hinzu. Ersetzen Sie die Beispielkonto-ID (*SOURCE01234567*) durch die ID des Kontos, das Sie für den Quellstream verwenden werden.

```
{
    "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. Öffnen Sie die Rolle `kinesis-analytics-MyApplication-us-west-2` und notieren Sie sich ihren Amazon-Ressourcennamen (ARN). Sie brauchen diesen im nächsten Abschnitt. Der Rollen-ARN sieht wie folgt aus.

```
arn:aws:iam::SINK012345678:role/service-role/kinesis-analytics-MyApplication-us-
west-2
```

Rollen und Richtlinien für Quell-Konten

1. Erstellen Sie eine Richtlinie im Quellkonto mit dem Namen `KA-Source-Stream-Policy`. Verwenden Sie die folgende JSON-Datei für die Richtlinie. Ersetzen Sie die Beispielkontonummer durch die Kontonummer des Quellkontos.

```
{
    "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. Erstellen Sie eine Rolle in dem Quellkonto namens `MF-Source-Stream-Role`. Gehen Sie wie folgt vor, um die Rolle mithilfe des Managed Flink-Anwendungsfalls zu erstellen:

    1. Wählen Sie in der IAM-Managementkonsole die Option Rolle erstellen aus.

2. Wählen Sie auf der Seite Rolle erstellen AWS-Service aus. Wählen Sie in der Serviceliste Kinesis aus.
3. Wählen Sie im Abschnitt Wählen Sie Ihren Anwendungsfall aus die Option Managed Service für Apache Flink aus.
4. Wählen Sie Weiter: Berechtigungen aus.
5. Fügen Sie die `KA-Source-Stream-Policy`-Berechtigungsrichtlinie hinzu, die Sie im vorherigen Schritt erstellt haben. Wählen Sie Weiter: Tags aus.
6. Wählen Sie Weiter: Prüfen aus.
7. Benennen Sie die Rolle `KA-Source-Stream-Role`. Ihre Anwendung verwendet diese Rolle für den Zugriff auf den Quellstream.

3. Fügen Sie die ARN `kinesis-analytics-MyApplication-us-west-2` aus dem Senken-Konto zur Vertrauensstellung der `KA-Source-Stream-Role`-Rolle im Quellkonto hinzu:

   1. Öffnen Sie die `KA-Source-Stream-Role` in der IAM-Konsole.
   2. Wählen Sie die Registerkarte Trust Relationships.
   3. Wählen Sie Edit Trust Relationship (Vertrauensstellungen bearbeiten).
   4. Verwenden Sie den folgenden Code für die Vertrauensstellung. Ersetzen Sie die beispielhafte Konto-ID (*SINK012345678901*) mit Ihrer Senken-Konto-ID.

```
{
    "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"
      }
    ]
}
```

Aktualisieren Sie das Python-Skript

In diesem Abschnitt aktualisieren Sie das Python-Skript, das Beispieldaten generiert, um das Quellkontoprofil zu verwenden.

Aktualisieren Sie das `stock.py`-Skript mit den folgenden hervorgehobenen Änderungen.

```
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")
```

Aktualisieren der Java-Anwendung

In diesem Abschnitt aktualisieren Sie den Java-Anwendungscode, sodass er beim Lesen aus dem Quellstream die Rolle des Quellkontos übernimmt.

Führen Sie die folgenden Änderungen an der Datei `BasicStreamingJob.java` durch. Ersetzen Sie die Beispiel-Quellkontonummer (*SOURCE01234567*) durch Ihre Quellkontonummer.

```
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");
    }
}
```

Erstellen Sie die Anwendung, laden Sie sie hoch und führen Sie sie aus.

Um die Anwendung zu aktualisieren und auszuführen, führen Sie Folgendes aus:

1. Erstellen Sie die Anwendung erneut, indem Sie folgenden Befehl im Verzeichnis mit der Datei `pom.xml` ausführen.

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

2. Löschen Sie die vorherige JAR-Datei aus Ihrem Amazon Simple Storage Service (Amazon S3)-Bucket und laden Sie dann die neue `aws-kinesis-analytics-java-apps-1.0.jar` Datei in den S3-Bucket hoch.
3. Wählen Sie auf der Seite der Anwendung in der Managed Service für Apache Flink-Konsole die Optionen Konfigurieren, Aktualisieren aus, um die JAR-Datei der Anwendung neu zu laden.

4. Führen Sie das `stock.py`-Skript aus, um Daten an den Quellstream zu senden.

```
python stock.py
```

Die Anwendung liest jetzt Daten aus dem Kinesis Stream in dem anderen Konto.

Sie können überprüfen, ob die Anwendung funktioniert, indem Sie die `PutRecords.Bytes`-Metrik des `ExampleOutputStream`-Streams überprüfen. Wenn im Ausgabestrom Aktivität vorhanden ist, funktioniert die Anwendung ordnungsgemäß.

Tutorial: Einen benutzerdefinierten Truststore mit Amazon MSK verwenden

Aktuelle Datenquellen-APIs

Wenn Sie die aktuellen Datenquellen-APIs verwenden, kann Ihre Anwendung das hier beschriebene Serviceprogramm MSK Config Providers nutzen. Auf diese Weise kann Ihre KafkaSource Funktion auf Ihren Keystore und Truststore für gegenseitige TLS in Amazon S3 zugreifen.

```
...
// 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 + "}");
...
```

Weitere Informationen und einen Walkthrough finden Sie hier.

Legacy SourceFunction APIs

Wenn Sie die Legacy- SourceFunction APIs verwenden, verwendet Ihre Anwendung benutzerdefinierte Serialisierungs- und Deserialisierungsschemata, die die open Methode zum Laden des benutzerdefinierten Truststores überschreiben. Dadurch steht der Truststore der Anwendung zur Verfügung, nachdem die Anwendung Threads neu gestartet oder ersetzt hat.

Der benutzerdefinierte Truststore wird mithilfe des folgenden Codes abgerufen und gespeichert:

```
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**
>
> Für Apache Flink muss der Truststore imJKS-Format sein.

> **Note**
>
> Um die erforderlichen Voraussetzungen für diese Übung einzurichten, schließen Sie zunächst die Erste Schritte (DataStream API)-Übung ab.

Das folgende Tutorial zeigt, wie eine sichere Verbindung (Verschlüsselung bei der Übertragung) zu einem Kafka-Cluster hergestellt wird, der Serverzertifikate verwendet, die von einer benutzerdefinierten, privaten oder sogar selbst gehosteten Zertifizierungsstelle (CA) ausgestellt wurden.

Um einen Kafka-Client sicher über TLS mit einem Kafka-Cluster zu verbinden, muss der Kafka-Client (wie die Flink-Beispielanwendung) der gesamten Vertrauenskette vertrauen, die durch die Serverzertifikate des Kafka-Clusters dargestellt wird (von der ausstellenden CA bis zur Root-Level-CA). Als Beispiel für einen benutzerdefinierten Truststore verwenden wir einen Amazon MSK-Cluster mit aktivierter gegenseitiger TLS (MTLS)-Authentifizierung. Dies bedeutet, dass die MSK-Clusterknoten Serverzertifikate verwenden, die von einer AWS-Certificate Manager Private Certificate Authority (ACM Private CA) ausgestellt wurden, die für Ihr Konto und Ihre Region privat ist und der daher vom Standard-Truststore der Java Virtual Machine (JVM), die die Flink-Anwendung ausführt, nicht vertraut wird.

> **Note**
>
> - Ein Keystore wird verwendet, um private Schlüssel und Identitätszertifikate zu speichern, die eine Anwendung sowohl dem Server als auch dem Client zur Überprüfung vorlegen sollte.
> - Ein Truststore wird verwendet, um Zertifikate von zertifizierten Stellen (CA) zu speichern, die das vom Server in einer SSL-Verbindung vorgelegte Zertifikat verifizieren.

Sie können die in diesem Tutorial beschriebene Technik auch für Interaktionen zwischen einer Managed Service für Apache Flink-Anwendung und anderen Apache Kafka-Quellen verwenden, z. B.:

- Einem benutzerdefinierten Apache Kafka-Cluster, der in AWS (Amazon EC2 oder Amazon EKS) gehostet wird
- Einem Confluent-Kafka-Cluster, gehostet in AWS

- Einem lokalen Kafka-Cluster, auf den über AWS Direct Connect oder VPN zugegriffen wird

Dieses Tutorial enthält die folgenden Abschnitte:

- Erstellen einer VPC mit einem Amazon-MSK-Cluster
- Erstellen Sie einen benutzerdefinierten Truststore und wenden Sie ihn auf Ihren Cluster an
- Den Anwendungscode erstellen:
- Hochladen des Apache Flink-Streaming-Java-Codes
- Erstellen Sie die Anwendung
- Konfigurieren der Anwendung
- Ausführen der Anwendung
- Testen der Anwendung

Erstellen einer VPC mit einem Amazon-MSK-Cluster

Folgen Sie dem Tutorial Erste Schritte mit Amazon MSK, um eine Beispiel-VPC und Amazon MSK-Cluster für den Zugriff über eine Managed Service für Apache Flink-Anwendung zu erstellen.

Wenn Sie das Tutorial abgeschlossen haben, gehen Sie auch folgendermaßen vor:

- Wiederholen Sie in Schritt 3: Thema erstellen den Befehl `kafka-topics.sh --create`, um ein Zielthema mit dem Namen `AWSKafkaTutorialTopicDestination` zu erstellen:

```
bin/kafka-topics.sh --create --bootstrap-server ZooKeeperConnectionString --
replication-factor 3 --partitions 1 --topic AWSKafkaTutorialTopicDestination
```

> **Note**
>
> Wenn der `kafka-topics.sh`-Befehl eine `ZooKeeperClientTimeoutException` zurückgibt, stellen Sie sicher, dass die Sicherheitsgruppe des Kafka-Clusters über eine Regel für eingehenden Datenverkehr verfügt, die den gesamten Datenverkehr von der privaten IP-Adresse der Client-Instance zulässt.

- Notieren Sie sich die Bootstrap-Serverliste für Ihren Cluster. Sie können die Liste der Bootstrap-Server mit dem folgenden Befehl abrufen (ersetzen Sie *`ClusterArn`* durch den ARN Ihres MSK-Clusters):

```
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"
}
```

- Wenn Sie den Schritten in diesem Tutorial und den Voraussetzungs-Tutorials folgen, achten Sie darauf, dass Sie die von Ihnen gewählte AWS-Region in Ihrem Code, Ihren Befehlen und Ihren Konsoleneinträgen verwenden.

Erstellen Sie einen benutzerdefinierten Truststore und wenden Sie ihn auf Ihren Cluster an

In diesem Abschnitt erstellen Sie eine benutzerdefinierte Zertifizierungsstelle (CA), verwenden sie, um einen benutzerdefinierten Truststore zu generieren, und wenden sie auf Ihren MSK-Cluster an.

Folgen Sie dem Tutorial zur Client-Authentifizierung im Amazon Managed Streaming für Apache Kafka Entwicklerhandbuch, um Ihren benutzerdefinierten Truststore zu erstellen und anzuwenden.

Den Anwendungscode erstellen:

In diesem Abschnitt laden Sie die JAR-Datei der Anwendung herunter und kompilieren sie.

Der Java-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben. Weitere Informationen finden Sie unter Git installieren.
2. Klonen Sie das Remote-Repository mit dem folgenden Befehl:

   ```
   git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
   ```

3. Der Anwendungscode befindet sich in `amazon-kinesis-data-analytics-java-examples/CustomKeystore`. Sie können den Code untersuchen, um sich mit der Struktur des Managed Service für Apache Flink-Codes vertraut zu machen.

4. Verwenden Sie entweder das Befehlszeilen-Maven-Tool oder Ihre bevorzugte Entwicklungsumgebung, um die JAR-Datei zu erstellen. Geben Sie Folgendes ein, um die JAR-Datei mit dem Befehlszeilen-Maven-Tool zu kompilieren:

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

Wenn der Build erfolgreich ist, wird die folgende Datei erstellt:

```
target/flink-app-1.0-SNAPSHOT.jar
```

> **Note**
>
> Der bereitgestellte Quellcode basiert auf Bibliotheken von Java 11.

Hochladen des Apache Flink-Streaming-Java-Codes

In diesem Abschnitt laden Sie Ihren Anwendungscode in den Amazon S3-Bucket hoch, den Sie im Erste Schritte (DataStream API)-Tutorial erstellt haben.

> **Note**
>
> Wenn Sie den Amazon S3-Bucket aus dem Tutorial Erste Schritte gelöscht haben, führen Sie den Schritt the section called “Hochladen des Apache Flink-Streaming-Java-Codes” erneut aus.

1. Wählen Sie in der Amazon S3-Konsole den Bucket ka-app-code-***<username>*** und dann Hochladen aus.
2. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `flink-app-1.0-SNAPSHOT.jar`Datei, die Sie im vorherigen Schritt erstellt haben.
3. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Wählen Sie für Laufzeit die Option Apache Flink 1.15.2 aus.
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen eines Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rolle: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):
   - Geben Sie für Amazon-S3-Bucket **`ka-app-code-`***`<username>`* ein.
   - Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **`flink-app-1.0-SNAPSHOT.jar`** ein.
3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** erstellen/aktualisieren aus.

> **Note**
>
> Wenn Sie Anwendungsressourcen mithilfe der Konsole angeben (z. B. Protokolle oder eine VPC), ändert die Konsole Ihre Anwendungsausführungsrolle, um die Berechtigung für den Zugriff auf diese Ressourcen zu gewähren.

4. Wählen Sie unter Eigenschaften die Option Gruppe hinzufügen aus. Geben Sie die folgenden Eigenschaften ein:

Gruppen-ID	Schlüssel	Wert
KafkaSource	Thema	AWSKafkaTutorialTopic
KafkaSource	bootstrap.servers	*Die Bootstrap-Serverliste, die Sie zuvor gespeichert haben*
KafkaSource	security.protocol	SSL
KafkaSource	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSource	ssl.truststore.passwort	changeit

> **Note**
>
> Das ssl.truststore.password für das Standardzertifikat ist „changeit“. Sie müssen diesen Wert nicht ändern, wenn Sie das Standardzertifikat verwenden.

Wählen Sie erneut Gruppe hinzufügen. Geben Sie die folgenden Eigenschaften ein:

Gruppen-ID	Schlüssel	Wert
KafkaSink	Thema	AWSKafkaTutorialTopicDestination
KafkaSink	bootstrap.servers	*Die Bootstrap-Serverliste, die Sie zuvor gespeichert haben*
KafkaSink	security.protocol	SSL
KafkaSink	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSink	ssl.truststore.passwort	changeit
KafkaSink	transaction.timeout.ms	1000

Der Anwendungscode liest die oben genannten Anwendungseigenschaften, um die Quelle und Senke zu konfigurieren, die für die Interaktion mit Ihrer VPC und Ihrem Amazon MSK-Cluster verwendet werden. Weitere Informationen zur Verwendung von Eigenschaften finden Sie unter Laufzeiteigenschaften.

5. Wählen Sie unter Snapshots die Option Deaktivieren aus. Dadurch wird es einfacher, die Anwendung zu aktualisieren, ohne ungültige Anwendungsstatusdaten zu laden.
6. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
7. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
8. Wählen Sie im Abschnitt Virtual Private Cloud (VPC) die VPC aus, die mit Ihrer Anwendung verknüpft werden soll. Wählen Sie die mit Ihrer VPC verknüpften Subnetze und Sicherheitsgruppe aus, die die Anwendung für den Zugriff auf VPC-Ressourcen verwenden soll.
9. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`
>
> Dieser Protokollstream wird zur Überwachung der Anwendung verwendet.

Ausführen der Anwendung

Das Flink-Jobdiagramm kann angezeigt werden, indem Sie die Anwendung ausführen, das Apache Flink-Dashboard öffnen und den gewünschten Flink-Job auswählen.

Testen der Anwendung

In diesem Abschnitt schreiben Sie Datensätze zum Quellthema. Die Anwendung liest Datensätze aus dem Quellthema und schreibt sie in das Zielthema. Sie überprüfen, ob die Anwendung funktioniert, indem Sie Datensätze in das Quellthema schreiben und Datensätze aus dem Zielthema lesen.

Um Datensätze aus den Themen zu schreiben und zu lesen, folgen Sie den Schritten in Schritt 6: Daten produzieren und verwenden im Tutorial Erste Schritte mit Amazon MSK.

Um aus dem Zielthema zu lesen, verwenden Sie in Ihrer zweiten Verbindung zum Cluster den Namen des Zielthemas anstelle des Quellthemas:

```
bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString --
consumer.config client.properties --topic AWSKafkaTutorialTopicDestination --from-
beginning
```

Wenn im Zielthema keine Datensätze angezeigt werden, lesen Sie den Abschnitt Kein Zugriff auf Ressourcen in einer VPC möglich im Thema Fehlerbehebung.

Python-Beispiele

In den folgenden Beispielen wird die Erstellung von Anwendungen über Python mit der Apache Flink Tabellen-API gezeigt.

Themen

- Beispiel: Ein rollierendes Fenster in Python erstellen
- Beispiel: Ein gleitendes Fenster in Python erstellen
- Beispiel: Streaming-Daten in Python an Amazon S3 senden

Beispiel: Ein rollierendes Fenster in Python erstellen

In dieser Übung erstellen Sie eine Anwendung von Python Managed Service für Apache Flink, die Daten mithilfe eines rollierenden Fensters aggregiert.

> **Note**
>
> Um die erforderlichen Voraussetzungen für diese Übung einzurichten, schließen Sie zunächst die Erste Schritte (Python)-Übung ab.

Dieses Thema enthält die folgenden Abschnitte:

- Erstellen Sie abhängige Ressourcen
- Schreiben Sie Beispieldatensätze in den Eingabe-Stream
- Laden Sie den Anwendungscode herunter und überprüfen Sie ihn
- Komprimieren Sie den Apache Flink Streaming Python-Code und laden Sie ihn hoch
- Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus
- Bereinigen von AWS-Ressourcen

Erstellen Sie abhängige Ressourcen

Bevor Sie für diese Übung eine Anwendung von Managed Service für Apache Flink erstellen, erstellen Sie die folgenden abhängigen Ressourcen:

- Zwei Kinesis Data Streams (`ExampleInputStream` und `ExampleOutputStream`)

- Einen Amazon S3-Bucket zum Speichern des Anwendungscodes (`ka-app-code-<username>`)

Sie können die Kinesis Streams und den Amazon-S3-Bucket mithilfe der Konsole erstellen. Anweisungen zum Erstellen dieser Ressourcen finden Sie in den folgenden Themen:

- Data Streams erstellen und aktualisieren im Amazon Kinesis Data Streams Entwicklerleitfaden. Benennen Sie Ihre Data Streams **`ExampleInputStream`** und **`ExampleOutputStream`**.
- Wie erstelle ich einen S3-Bucket? im Amazon Simple Storage Service Benutzerhandbuch. Geben Sie dem Amazon S3-Bucket einen global eindeutigen Namen, indem Sie Ihren Anmeldenamen anhängen, z. B. **`ka-app-code-<username>`**.

Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Abschnitt verwenden Sie ein Python-Skript zum Schreiben von Datensätzen in den Stream für die zu verarbeitende Anwendung.

> **Note**
>
> Dieser Abschnitt erfordert AWS SDK for Python (Boto).

> **Note**
>
> Das Python-Skript in diesem Abschnitt verwendet die AWS CLI. Sie müssen Ihre AWS CLI so konfigurieren, dass sie Ihre Kontoanmeldeinformationen und Ihre Standardregion verwendet. Geben Sie zum Konfigurieren der AWS CLI Folgendes ein:
>
> ```
> aws configure
> ```

1. Erstellen Sie eine Datei `stock.py` mit dem folgenden Inhalt:

```
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. Führen Sie das `stock.py`Skript aus:

```
$ python stock.py
```

   Lassen Sie das Skript laufen, während Sie den Rest des Tutorials abschließen.

Laden Sie den Anwendungscode herunter und überprüfen Sie ihn

Der Python-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben. Weitere Informationen finden Sie unter Git installieren.
2. Klonen Sie das Remote-Repository mit dem folgenden Befehl:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/python/TumblingWindow` Verzeichnis .

Der Anwendungscode befindet sich in der `tumbling-windows.py`-Datei. Beachten Sie Folgendes zum Anwendungscode:

- Die Anwendung verwendet eine Kinesis-Tabellenquelle zum Lesen aus dem Quell-Stream. Der folgende Ausschnitt ruft die `create_table`-Funktion zum Erstellen der Kinesis-Tabellenquelle auf:

```
table_env.execute_sql(
        create_input_table(input_table_name, input_stream, input_region,
 stream_initpos)
    )
```

  Die `create_table`-Funktion verwendet einen SQL-Befehl, um eine Tabelle zu erstellen, die von der Streaming-Quelle unterstützt wird:

```
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)
```

- Die Anwendung verwendet den `Tumble`-Operator, um Datensätze innerhalb eines bestimmten rollierenden Fensters zu aggregieren und die aggregierten Datensätze als Tabellenobjekt zurückzugeben:

```
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")
```

- Die Anwendung verwendet den Kinesis Flink-Konnektor aus flink-sql-connector-kinesis-1.15.2.jar.

Komprimieren Sie den Apache Flink Streaming Python-Code und laden Sie ihn hoch

In diesem Abschnitt laden Sie Ihren Anwendungscode in den Amazon S3-Bucket hoch, den Sie im Erstellen Sie abhängige Ressourcen-Abschnitt erstellt haben.

1. Verwenden Sie Ihre bevorzugte Komprimierungsanwendung, um die Dateien `tumbling-windows.py` und `flink-sql-connector-kinesis-1.15.2.jar` zu komprimieren. Benennen Sie das Archiv `myapp.zip`.
2. Wählen Sie in der Amazon S3-Konsole den Bucket ka-app-code-***<username>*** und dann Upload aus.
3. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `myapp.zip`Datei, die Sie im vorherigen Schritt erstellt haben.
4. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.

3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:

   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.

   > **Note**
   >
   > Managed Service für Apache Flink verwendet Apache Flink Version 1.15.2.

   - Belassen Sie den Versions-Pulldown bei Apache Flink Version 1.15.2 (empfohlene Version).
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rolle: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):

   - Geben Sie für Amazon-S3-Bucket **`ka-app-code-`***`<username>`* ein.
   - Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **`myapp.zip`** ein.
3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** erstellen/aktualisieren aus.

4. Wählen Sie unter Eigenschaften die Option Gruppe hinzufügen aus.
5. Geben Sie Folgendes ein:

Gruppen-ID	Schlüssel	Wert
consumer.config.0	**input.stream.name**	**ExampleInputStream**
consumer.config.0	**aws.region**	**us-west-2**
consumer.config.0	**scan.stream.initpos**	**LATEST**

   Wählen Sie Speichern.
6. Wählen Sie unter Eigenschaften erneut Gruppe hinzufügen aus.
7. Geben Sie Folgendes ein:

Gruppen-ID	Schlüssel	Wert
producer.config.0	**output.stream.name**	**ExampleOutputStream**
producer.config.0	**aws.region**	**us-west-2**
producer.config.0	**shard.count**	**1**

8. Wählen Sie unter Eigenschaften erneut Gruppe hinzufügen aus. Geben Sie für Gruppen-ID **kinesis.analytics.flink.run.options** ein. Diese spezielle Eigenschaftsgruppe teilt Ihrer Anwendung mit, wo sich ihre Coderessourcen befinden. Weitere Informationen finden Sie unter Angeben Ihrer Codedateien.
9. Geben Sie Folgendes ein:

Gruppen-ID	Schlüssel	Wert
kinesis.analytics.flink.run.options	**python**	**tumbling-windows.py**

Gruppen-ID	Schlüssel	Wert
kinesis.analytics.flink.run.options	**jarfile**	**flink-sql-connector-kinesis-1.15.2.jar**

10. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
11. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
12. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`
>
> Dieser Protokollstream wird zur Überwachung der Anwendung verwendet. Dies ist nicht derselbe Protokollstream, den die Anwendung zum Senden von Ergebnissen verwendet.

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf die Kinesis-Datenströme.

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **kinesis-analytics-service-MyApplication-us-west-2**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.

4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*012345678901*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

Ausführen der Anwendung

Das Flink-Jobdiagramm kann angezeigt werden, indem Sie die Anwendung ausführen, das Apache Flink-Dashboard öffnen und den gewünschten Flink-Job auswählen.

Sie können die Metriken von Managed Service für Apache Flink in der CloudWatch Konsole überprüfen, um zu überprüfen, ob die Anwendung funktioniert.

Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zur Bereinigung von AWS-Ressourcen, die im Rollierendes Fenster-Tutorial erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte:

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Ihrer Kinesis Data Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen

Löschen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen Ihrer Kinesis Data Streams

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Kinesis Data Streams die Option ausExampleInputStream.
3. Wählen Sie auf der ExampleInputStream Seite Kinesis Stream löschen aus und bestätigen Sie dann den Löschvorgang.
4. Wählen Sie auf der Seite Kinesis Streams die , ExampleOutputStreamwählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den ka-app-codeBucket -***<username>*** aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle kinesis-analytics-MyApplication--us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Beispiel: Ein gleitendes Fenster in Python erstellen

> **Note**
>
> Um die erforderlichen Voraussetzungen für diese Übung einzurichten, schließen Sie zunächst die Erste Schritte (Python)-Übung ab.

Dieses Thema enthält die folgenden Abschnitte:

- Erstellen Sie abhängige Ressourcen
- Schreiben Sie Beispieldatensätze in den Eingabe-Stream
- Laden Sie den Anwendungscode herunter und überprüfen Sie ihn
- Komprimieren Sie den Apache Flink Streaming Python-Code und laden Sie ihn hoch
- Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus
- Bereinigen von AWS-Ressourcen

Erstellen Sie abhängige Ressourcen

Bevor Sie für diese Übung eine Anwendung von Managed Service für Apache Flink erstellen, erstellen Sie die folgenden abhängigen Ressourcen:

- Zwei Kinesis Data Streams (`ExampleInputStream` und `ExampleOutputStream`)
- Einen Amazon S3-Bucket zum Speichern des Anwendungscodes (`ka-app-code-<username>`)

Sie können die Kinesis Streams und den Amazon-S3-Bucket mithilfe der Konsole erstellen. Anweisungen zum Erstellen dieser Ressourcen finden Sie in den folgenden Themen:

- Data Streams erstellen und aktualisieren im Amazon Kinesis Data Streams Entwicklerleitfaden. Benennen Sie Ihre Data Streams **ExampleInputStream** und **ExampleOutputStream**.
- Wie erstelle ich einen S3-Bucket? im Amazon Simple Storage Service Benutzerhandbuch. Geben Sie dem Amazon S3-Bucket einen global eindeutigen Namen, indem Sie Ihren Anmeldenamen anhängen, z. B. **ka-app-code-*<username>***.

Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Abschnitt verwenden Sie ein Python-Skript zum Schreiben von Datensätzen in den Stream für die zu verarbeitende Anwendung.

> **Note**
>
> Dieser Abschnitt erfordert AWS SDK for Python (Boto).

> **Note**
>
> Das Python-Skript in diesem Abschnitt verwendet die AWS CLI. Sie müssen Ihre AWS CLI so konfigurieren, dass sie Ihre Kontoanmeldeinformationen und Ihre Standardregion verwendet. Geben Sie zum Konfigurieren der AWS CLI Folgendes ein:
>
> ```
> aws configure
> ```

1. Erstellen Sie eine Datei `stock.py` mit dem folgenden Inhalt:

```
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. Führen Sie das `stock.py`Skript aus:

```
$ python stock.py
```

   Lassen Sie das Skript laufen, während Sie den Rest des Tutorials abschließen.

Laden Sie den Anwendungscode herunter und überprüfen Sie ihn

Der Python-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben. Weitere Informationen finden Sie unter Git installieren.
2. Klonen Sie das Remote-Repository mit dem folgenden Befehl:

```
git clone https://github.com/aws-samples/>amazon-kinesis-data-analytics-java-
examples
```

3. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/python/SlidingWindow` Verzeichnis .

Der Anwendungscode befindet sich in der `sliding-windows.py`-Datei. Beachten Sie Folgendes zum Anwendungscode:

- Die Anwendung verwendet eine Kinesis-Tabellenquelle zum Lesen aus dem Quell-Stream. Der folgende Ausschnitt ruft die `create_input_table`-Funktion zum Erstellen der Kinesis-Tabellenquelle auf:

```
table_env.execute_sql(
        create_input_table(input_table_name, input_stream, input_region,
 stream_initpos)
    )
```

  Die `create_input_table`-Funktion verwendet einen SQL-Befehl, um eine Tabelle zu erstellen, die von der Streaming-Quelle unterstützt wird:

```
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)
 }
```

- Die Anwendung verwendet den `Slide`-Operator, um Datensätze innerhalb eines bestimmten gleitenden Fensters zu aggregieren und die aggregierten Datensätze als Tabellenobjekt zurückzugeben:

```
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")
    )
```

- Die Anwendung verwendet den Kinesis-Flink-Konnektor aus der flink-sql-connector-kinesis-1.15.2.jar-Datei.

Komprimieren Sie den Apache Flink Streaming Python-Code und laden Sie ihn hoch

In diesem Abschnitt laden Sie Ihren Anwendungscode in den Amazon S3-Bucket hoch, den Sie im Erstellen Sie abhängige Ressourcen-Abschnitt erstellt haben.

In diesem Abschnitt wird beschrieben, wie Sie Ihre Python-Anwendung verpacken.

1. Verwenden Sie Ihre bevorzugte Komprimierungsanwendung, um die Dateien `sliding-windows.py` und `flink-sql-connector-kinesis-1.15.2.jar` zu komprimieren. Benennen Sie das Archiv `myapp.zip`.
2. Wählen Sie in der Amazon S3-Konsole den Bucket ka-app-code-***<username>*** und dann Upload aus.
3. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `myapp.zip`Datei, die Sie im vorherigen Schritt erstellt haben.
4. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.

3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:

   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.

   > **Note**
   >
   > Managed Service für Apache Flink verwendet Apache Flink Version 1.15.2.

   - Belassen Sie den Versions-Pulldown bei Apache Flink Version 1.15.2 (empfohlene Version).
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rolle: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):

   - Geben Sie für Amazon-S3-Bucket **`ka-app-code-`***`<username>`* ein.
   - Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **`myapp.zip`** ein.
3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** erstellen/aktualisieren aus.

4. Wählen Sie unter Eigenschaften die Option Gruppe hinzufügen aus.
5. Geben Sie die folgenden Eigenschaften und Werte der Anwendung ein:

Gruppen-ID	Schlüssel	Wert
consumer.config.0	**input.stream.name**	**ExampleInputStream**
consumer.config.0	**aws.region**	**us-west-2**
consumer.config.0	**scan.stream.initpos**	**LATEST**

   Wählen Sie Speichern.
6. Wählen Sie unter Eigenschaften erneut Gruppe hinzufügen aus.
7. Geben Sie die folgenden Eigenschaften und Werte der Anwendung ein:

Gruppen-ID	Schlüssel	Wert
producer.config.0	**output.stream.name**	**ExampleOutputStream**
producer.config.0	**aws.region**	**us-west-2**
producer.config.0	**shard.count**	**1**

8. Wählen Sie unter Eigenschaften erneut Gruppe hinzufügen aus. Geben Sie für Gruppen-ID **kinesis.analytics.flink.run.options** ein. Diese spezielle Eigenschaftsgruppe teilt Ihrer Anwendung mit, wo sich ihre Coderessourcen befinden. Weitere Informationen finden Sie unter Angeben Ihrer Codedateien.
9. Geben Sie die folgenden Eigenschaften und Werte der Anwendung ein:

Gruppen-ID	Schlüssel	Wert
kinesis.analytics.flink.run.options	**python**	**sliding-windows.py**

Gruppen-ID	Schlüssel	Wert
kinesis.analytics.flink.run.options	**jarfile**	**flink-sql-connector-kinesis_1.15.2.jar**

10. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
11. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
12. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`
>
> Dieser Protokollstream wird zur Überwachung der Anwendung verwendet. Dies ist nicht derselbe Protokollstream, den die Anwendung zum Senden von Ergebnissen verwendet.

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf die Kinesis-Datenströme.

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **kinesis-analytics-service-MyApplication-us-west-2**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.

4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*012345678901*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

Ausführen der Anwendung

Das Flink-Jobdiagramm kann angezeigt werden, indem Sie die Anwendung ausführen, das Apache Flink-Dashboard öffnen und den gewünschten Flink-Job auswählen.

Sie können die Metriken von Managed Service für Apache Flink in der CloudWatch Konsole überprüfen, um zu überprüfen, ob die Anwendung funktioniert.

Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zum Bereinigen von AWS-Ressourcen, die im Tutorial Gleitendes Fenster erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte:

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Ihrer Kinesis Data Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen

Löschen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen Ihrer Kinesis Data Streams

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Kinesis Data Streams die Option ausExampleInputStream.
3. Wählen Sie auf der ExampleInputStream Seite Kinesis Stream löschen aus und bestätigen Sie dann den Löschvorgang.
4. Wählen Sie auf der Seite Kinesis Streams die , ExampleOutputStreamwählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den ka-app-codeBucket -***<username>*** aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle kinesis-analytics--MyApplication-us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Beispiel: Streaming-Daten in Python an Amazon S3 senden

In dieser Übung erstellen Sie eine Anwendung von Python Managed Service für Apache Flink, die Daten an eine Amazon Simple Storage Service-Senke streamt.

> **Note**
>
> Um die erforderlichen Voraussetzungen für diese Übung einzurichten, schließen Sie zunächst die Erste Schritte (Python)-Übung ab.

Dieses Thema enthält die folgenden Abschnitte:

- Erstellen Sie abhängige Ressourcen
- Schreiben Sie Beispieldatensätze in den Eingabe-Stream
- Laden Sie den Anwendungscode herunter und überprüfen Sie ihn
- Komprimieren Sie den Apache Flink Streaming Python-Code und laden Sie ihn hoch
- Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus
- Bereinigen von AWS-Ressourcen

Erstellen Sie abhängige Ressourcen

Bevor Sie für diese Übung eine Anwendung von Managed Service für Apache Flink erstellen, erstellen Sie die folgenden abhängigen Ressourcen:

- Einen Kinesis Data Stream (`ExampleInputStream`)
- Einen Amazon S3-Bucket zum Speichern des Codes und der Ausgabe der Anwendung (`ka-app-code-<username>`)

> **Note**
>
> Managed Service für Apache Flink kann keine Daten auf Amazon S3 schreiben, wenn die serverseitige Verschlüsselung auf Managed Service für Apache Flink aktiviert ist.

Sie können den Kinesis-Stream und den Amazon-S3-Bucket mithilfe der Konsole erstellen. Anweisungen zum Erstellen dieser Ressourcen finden Sie in den folgenden Themen:

- Data Streams erstellen und aktualisieren im Amazon Kinesis Data Streams Entwicklerleitfaden. Benennen Sie Ihren Data Stream **`ExampleInputStream`**.
- Wie erstelle ich einen S3-Bucket? im Amazon Simple Storage Service Benutzerhandbuch. Geben Sie dem Amazon S3-Bucket einen global eindeutigen Namen, indem Sie Ihren Anmeldenamen anhängen, z. B. **`ka-app-code-`*`<username>`***.

Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Abschnitt verwenden Sie ein Python-Skript zum Schreiben von Datensätzen in den Stream für die zu verarbeitende Anwendung.

> **Note**
>
> Dieser Abschnitt erfordert AWS SDK for Python (Boto).

> **Note**
>
> Das Python-Skript in diesem Abschnitt verwendet die AWS CLI. Sie müssen Ihre AWS CLI so konfigurieren, dass sie Ihre Kontoanmeldeinformationen und Ihre Standardregion verwendet. Geben Sie zum Konfigurieren der AWS CLI Folgendes ein:
>
> ```
> aws configure
> ```

1. Erstellen Sie eine Datei `stock.py` mit dem folgenden Inhalt:

```
    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. Führen Sie das `stock.py`Skript aus:

```
$ python stock.py
```

   Lassen Sie das Skript laufen, während Sie den Rest des Tutorials abschließen.

Laden Sie den Anwendungscode herunter und überprüfen Sie ihn

Der Python-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben. Weitere Informationen finden Sie unter Git installieren.

2. Klonen Sie das Remote-Repository mit dem folgenden Befehl:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/python/S3Sink` Verzeichnis .

Der Anwendungscode befindet sich in der `streaming-file-sink.py`-Datei. Beachten Sie Folgendes zum Anwendungscode:

- Die Anwendung verwendet eine Kinesis-Tabellenquelle zum Lesen aus dem Quell-Stream. Der folgende Ausschnitt ruft die `create_source_table`-Funktion zum Erstellen der Kinesis-Tabellenquelle auf:

```
table_env.execute_sql(
        create_source_table(input_table_name, input_stream, input_region,
 stream_initpos)
    )
```

  Die `create_source_table`-Funktion verwendet einen SQL-Befehl, um eine Tabelle zu erstellen, die von der Streaming-Quelle unterstützt wird

```
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'))
```

- Die Anwendung verwendet den `filesystem`-Konnektor zum Senden von Datensätzen an einen Amazon-S3-Bucket:

```
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)
```

- Die Anwendung verwendet den Kinesis-Flink-Konnektor aus der flink-sql-connector-kinesis-1.15.2.jar-Datei.

Komprimieren Sie den Apache Flink Streaming Python-Code und laden Sie ihn hoch

In diesem Abschnitt laden Sie Ihren Anwendungscode in den Amazon S3-Bucket hoch, den Sie im Erstellen Sie abhängige Ressourcen-Abschnitt erstellt haben.

1. Verwenden Sie Ihre bevorzugte Komprimierungsanwendung, um die Dateien `streaming-file-sink.py` und flink-sql-connector-kinesis-1.15.2.jar zu komprimieren. Benennen Sie das Archiv `myapp.zip`.
2. Wählen Sie in der Amazon S3-Konsole den Bucket ka-app-code-***<username>*** und dann Hochladen aus.

3. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `myapp.zip`Datei, die Sie im vorherigen Schritt erstellt haben.
4. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.

     > **Note**
     >
     > Managed Service für Apache Flink verwendet Apache Flink Version 1.15.2.

   - Belassen Sie den Versions-Pulldown bei Apache Flink Version 1.15.2 (empfohlene Version).
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf

ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:

- Richtlinie: `kinesis-analytics-service-MyApplication-us-west-2`
- Rolle: `kinesisanalytics-MyApplication-us-west-2`

Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):
   - Geben Sie für Amazon-S3-Bucket **`ka-app-code-<username>`** ein.
   - Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **`myapp.zip`** ein.
3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** erstellen/aktualisieren aus.
4. Wählen Sie unter Eigenschaften die Option Gruppe hinzufügen aus.
5. Geben Sie die folgenden Eigenschaften und Werte der Anwendung ein:

Gruppen-ID	Schlüssel	Wert
`consumer.config.0`	**`input.stream.name`**	**`ExampleInputStream`**
`consumer.config.0`	**`aws.region`**	**`us-west-2`**
`consumer.config.0`	**`scan.stream.initpos`**	**`LATEST`**

   Wählen Sie Speichern.
6. Wählen Sie unter Eigenschaften erneut Gruppe hinzufügen aus. Geben Sie für Gruppen-ID **`kinesis.analytics.flink.run.options`** ein. Diese spezielle Eigenschaftsgruppe teilt Ihrer Anwendung mit, wo sich ihre Coderessourcen befinden. Weitere Informationen finden Sie unter Angeben Ihrer Codedateien.
7. Geben Sie die folgenden Eigenschaften und Werte der Anwendung ein:

Gruppen-ID	Schlüssel	Wert
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. Wählen Sie unter Eigenschaften erneut Gruppe hinzufügen aus. Geben Sie für Gruppen-ID **sink.config.0** ein. Diese spezielle Eigenschaftsgruppe teilt Ihrer Anwendung mit, wo sich ihre Coderessourcen befinden. Weitere Informationen finden Sie unter Angeben Ihrer Codedateien.
9. Geben Sie die folgenden Anwendungseigenschaften und -Werte ein: (Ersetzen Sie *bucket-name* durch den tatsächlichen Namen Ihres Amazon S3-Buckets.)

Gruppen-ID	Schlüssel	Wert
sink.config.0	**output.bucket.name**	***bucket-name***

10. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
11. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
12. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`
>
> Dieser Protokollstream wird zur Überwachung der Anwendung verwendet. Dies ist nicht derselbe Protokollstream, den die Anwendung zum Senden von Ergebnissen verwendet.

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf die Kinesis-Datenströme.

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **`kinesis-analytics-service-MyApplication-us-west-2`**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*012345678901*) mit Ihrer Konto-ID.

```
{
    "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>/*"
            ]
        }
    ]
}
```

Ausführen der Anwendung

Das Flink-Jobdiagramm kann angezeigt werden, indem Sie die Anwendung ausführen, das Apache Flink-Dashboard öffnen und den gewünschten Flink-Job auswählen.

Sie können die Metriken von Managed Service für Apache Flink in der CloudWatch Konsole überprüfen, um zu überprüfen, ob die Anwendung funktioniert.

Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zum Bereinigen von AWS-Ressourcen, die im Tutorial Gleitendes Fenster erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte:

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Ihres Kinesis Data Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen

Löschen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen Ihres Kinesis Data Streams

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Kinesis Data Streams die Option ausExampleInputStream.
3. Wählen Sie auf der ExampleInputStream Seite Kinesis Stream löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den ka-app-codeBucket -***<username>*** aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle kinesis-analytics--MyApplication-us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Scala-Beispiele

In den folgenden Beispielen wird die Erstellung von Anwendungen über Scala mit Apache Flink gezeigt.

Themen

- Beispiel: Ein rollierendes Fenster in Scala erstellen
- Beispiel: Ein gleitendes Fenster in Scala erstellen
- Beispiel: Streaming-Daten in Scala an Amazon S3 senden

Beispiel: Ein rollierendes Fenster in Scala erstellen

> **Note**
>
> Ab Version 1.15 ist Flink Scala-frei. Anwendungen können jetzt die Java-API von jeder Scala-Version aus verwenden. Flink verwendet Scala intern immer noch in einigen Schlüsselkomponenten, macht Scala jedoch nicht im Benutzercode-Classloader verfügbar. Aus diesem Grund müssen Benutzer Scala-Abhängigkeiten zu ihren Jar-Archiven hinzufügen. Weitere Informationen zu den Scala-Änderungen in Flink 1.15 finden Sie unter Scalafrei in One Fifteen.

In dieser Übung erstellen Sie eine einfache Streaming-Anwendung, die Scala 3.2.0 und die Java DataStream -API von Flink verwendet. Die Anwendung liest Daten aus dem Kinesis Stream, aggregiert sie mithilfe von gleitenden Fenstern und schreibt die Ergebnisse in den Kinesis-Ausgabestream.

> **Note**
>
> Um die erforderlichen Voraussetzungen für diese Übung einzurichten, schließen Sie zunächst die Übung Erste Schritte (Scala) ab.

Dieses Thema enthält die folgenden Abschnitte:

- Laden Sie den Anwendungscode herunter und überprüfen Sie ihn
- Kompilieren Sie den Anwendungscode und laden Sie ihn hoch
- Erstellen und Ausführen der Anwendung (Konsole)
- Erstellen und Ausführen der Anwendung (CLI)
- Den Anwendungscode aktualisieren
- Bereinigen von AWS-Ressourcen

Laden Sie den Anwendungscode herunter und überprüfen Sie ihn

Der Python-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben. Weitere Informationen finden Sie unter Git installieren.
2. Klonen Sie das Remote-Repository mit dem folgenden Befehl:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/scala/TumblingWindow` Verzeichnis .

Beachten Sie Folgendes zum Anwendungscode:

- Eine `build.sbt`-Datei enthält Informationen über die Konfiguration und Abhängigkeiten der Anwendung, einschließlich der Bibliotheken des Managed Service für Apache Flink.
- Die `BasicStreamingJob.scala`-Datei enthält die Hauptmethode, die die Funktionalität der Anwendung definiert.
- Die Anwendung verwendet eine Kinesis-Quelle zum Lesen aus dem Quell-Stream. Der folgende Codeausschnitt erstellt die Kinesis-Quelle:

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
 defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

  Die Anwendung verwendet auch eine Kinesis-Senke, um in den Ergebnisstream zu schreiben. Der folgende Codeausschnitt erstellt die Kinesis-Senke:

```
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
}
```

- Die Anwendung verwendet den Fensteroperator, um die Anzahl der Werte für jedes Aktionssymbol über ein rollierendes Fenster von 5 Sekunden zu ermitteln. Der folgende Code erstellt den Operator und sendet die aggregierten Daten an eine neue Kinesis Data Streams Senke:

```
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)
```

- Die Anwendung erstellt Quell- und Senken-Konnektoren, um über ein - StreamExecutionEnvironment Objekt auf externe Ressourcen zuzugreifen.
- Die Anwendung erstellt Quell- und Senkenkonnektoren mit dynamischen Anwendungseigenschaften. Die Laufzeiteigenschaften der Anwendung werden gelesen, um die Konnektoren zu konfigurieren. Weitere Informationen zu Laufzeiteigenschaften finden Sie unter Laufzeiteigenschaften.

Kompilieren Sie den Anwendungscode und laden Sie ihn hoch

In diesem Abschnitt kompilieren Sie Ihren Anwendungscode und laden ihn in einen Amazon-S3-Bucket hoch.

Kompilieren des Anwendungscodes

Verwenden Sie das SBT-Build-Tool, um den Scala-Code für die Anwendung zu erstellen. Informationen zur Installation von SBT finden Sie unter Installieren von SBT mit CS-Setup. Sie müssen auch das Java Development Kits (JDK) installieren. Siehe Voraussetzungen für das Fertigstellen der Übungen

1. Zum Verwenden Ihres Anwendungscodes kompilieren und packen Sie ihn in eine JAR-Datei. Sie können Ihren Code mit SBT kompilieren und verpacken:

```
sbt assembly
```

2. Wenn die Anwendung erfolgreich kompiliert wurde, wird die folgende Datei erstellt:

```
target/scala-3.2.0/tumbling-window-scala-1.0.jar
```

Hochladen des Apache Flink-Streaming-Scala-Codes

In diesem Abschnitt erstellen Sie einen Amazon S3-Bucket und laden Ihren Anwendungscode hoch.

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie Bucket erstellen aus
3. Geben Sie `ka-app-code-<username>` im Feld Bucket-Name ein. Fügen Sie dem Bucket-Namen ein Suffix hinzu, wie z. B. Ihren Benutzernamen, damit er global eindeutig ist. Wählen Sie Weiter aus.
4. Lassen Sie im Schritt Optionen konfigurieren die Einstellungen unverändert und klicken Sie auf Weiter.
5. Lassen Sie im Schritt Berechtigungen festlegen die Einstellungen unverändert und klicken Sie auf Weiter.
6. Wählen Sie Bucket erstellen aus.
7. Wählen Sie den Bucket `ka-app-code-<username>` und dann Hochladen aus.
8. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `tumbling-window-scala-1.0.jar`Datei, die Sie im vorherigen Schritt erstellt haben.
9. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und Ausführen der Anwendung (Konsole)

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Geben Sie für Beschreibung den Text **`My Scala test app`** ein.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.
   - Belassen Sie die Version als Apache Flink Version 1.15.2 (empfohlene Version).
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rolle: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Konfigurieren der Anwendung

Verwenden Sie das folgende Verfahren, um die Anwendung zu konfigurieren.

Zum Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):

- Geben Sie für Amazon-S3-Bucket **ka-app-code-*<username>*** ein.
- Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **tumbling-window-scala-1.0.jar** ein.

3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **kinesis-analytics-MyApplication-us-west-2** erstellen/aktualisieren aus.
4. Wählen Sie unter Eigenschaften die Option Gruppe hinzufügen aus.
5. Geben Sie Folgendes ein:

Gruppen-ID	Schlüssel	Wert
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

Wählen Sie Speichern.

6. Wählen Sie unter Eigenschaften erneut Gruppe hinzufügen aus.
7. Geben Sie Folgendes ein:

Gruppen-ID	Schlüssel	Wert
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**
ProducerConfigProperties	**aws.region**	**us-west-2**

8. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
9. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
10. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die Amazon- CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf den Amazon S3-Bucket.

Um die IAM-Richtlinie zu bearbeiten, um S3-Bucket-Berechtigungen hinzuzufügen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **`kinesis-analytics-service-MyApplication-us-west-2`**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*012345678901*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

Ausführen der Anwendung

Das Flink-Jobdiagramm kann angezeigt werden, indem Sie die Anwendung ausführen, das Apache Flink-Dashboard öffnen und den gewünschten Flink-Job auswählen.

Stoppen der Anwendung

Um die Anwendung zu beenden, wählen Sie auf der MyApplication Seite Stoppen aus. Bestätigen Sie die Aktion.

Erstellen und Ausführen der Anwendung (CLI)

In diesem Abschnitt verwenden Sie AWS Command Line Interface, um die Anwendung Managed Service für Apache Flink zu erstellen und auszuführen. Verwenden Sie den AWS CLI-Befehl kinesisanalyticsv2, um Managed Service für Apache Flink-Anwendungen zu erstellen und mit diesen zu interagieren.

Erstellen einer Berechtigungsrichtlinie

> **Note**
>
> Sie müssen eine Berechtigungsrichtlinie und eine Rolle für Ihre Anwendung erstellen. Wenn Sie diese IAM-Ressourcen nicht erstellen, kann Ihre Anwendung nicht auf ihre Daten und Protokollstreams zugreifen.

Zuerst erstellen Sie eine Berechtigungsrichtlinie mit zwei Anweisungen: eine, die Berechtigungen für die Lese-Aktion auf den Quell-Stream zulässt, und eine andere, die Berechtigungen für die Schreib-Aktionen auf den Senken-Stream zulässt. Anschließend fügen Sie die Richtlinie an eine IAM-Rolle (die Sie im nächsten Abschnitt erstellen) an. Wenn Managed Service für Apache Flink also die Rolle übernimmt, verfügt der Service über die erforderlichen Berechtigungen zum Lesen aus dem Quell-Stream und zum Schreiben in den Senken-Stream.

Verwenden Sie den folgenden Code zum Erstellen der AKReadSourceStreamWriteSinkStream-Berechtigungsrichtlinie. Ersetzen Sie **username** durch den Benutzernamen, den Sie verwendet haben, um den Amazon-S3-Bucket zum Speichern des Anwendungscodes zu erstellen. Ersetzen Sie die Konto-ID in den Amazon-Ressourcennamen (ARNs) **(012345678901)** mit Ihrer Konto-ID. Die **MF-stream-rw-role**-Serviceausführungsrolle sollte auf die kundenspezifische Rolle zugeschnitten sein.

```
{
    "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"
      }
   ]
}
```

step-by-step Anweisungen zum Erstellen einer Berechtigungsrichtlinie finden Sie unter Tutorial: Erstellen und Anfügen Ihrer ersten vom Kunden verwalteten Richtlinie im IAM-Benutzerhandbuch.

Erstellen einer IAM-Rolle

In diesem Abschnitt erstellen Sie eine IAM-Rolle, die die Anwendung von Managed Service für Apache Flink annehmen kann, um einen Quell-Stream zu lesen und in den Senken-Stream zu schreiben.

Managed Service für Apache Flink kann ohne Berechtigungen nicht auf Ihren Stream zugreifen. Sie erteilen diese Berechtigungen über eine IAM-Rolle. Jeder IAM-Rolle sind zwei Richtlinien angefügt. Die Vertrauensrichtlinie erteilt Managed Service für Apache Flink die Berechtigung zum Übernehmen der Rolle und die Berechtigungsrichtlinie bestimmt, was Managed Service für Apache Flink nach Annahme der Rolle tun kann.

Sie können die Berechtigungsrichtlinie, die Sie im vorherigen Abschnitt erstellt haben, dieser Rolle anfügen.

So erstellen Sie eine IAM-Rolle

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Rollen und dann Rolle erstellen.
3. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option AWS-Service aus
4. Wählen Sie unter Choose the service that will use this role (Wählen Sie den Service aus, der diese Rolle verwendet) die Option Kinesis aus.
5. Wählen Sie unter Wählen Sie Ihren Anwendungsfall aus die Option Managed Service für Apache Flink aus.
6. Wählen Sie Weiter: Berechtigungen aus.
7. Wählen Sie auf der Seite Attach permissions policies (Berechtigungsrichtlinien hinzufügen) Next: Review (Weiter: Überprüfen) aus. Sie fügen Berechtigungsrichtlinien an, nachdem Sie die Rolle erstellt haben.

8. Geben Sie auf der Seite Create role (Rolle erstellen) den Text **`MF-stream-rw-role`** für Role name (Rollenname) ein. Wählen Sie Rolle erstellen aus.

   Jetzt haben Sie eine neue IAM-Rolle mit dem Namen `MF-stream-rw-role` erstellt. Im nächsten Schritt aktualisieren Sie die Vertrauens- und Berechtigungsrichtlinien für die Rolle

9. Fügen Sie die Berechtigungsrichtlinie der Rolle an.

   > **Note**
   >
   > Für diese Übung übernimmt Managed Service für Apache Flink diese Rolle sowohl für das Lesen von Daten aus einem Kinesis-Datenstrom (Quelle) als auch zum Schreiben der Ausgabedaten in einen anderen Kinesis-Datenstrom. Daher fügen Sie die Richtlinie an, die Sie im vorherigen Schritt, Erstellen einer Berechtigungsrichtlinie, erstellt haben.

   a. Wählen Sie auf der Seite Summary (Übersicht) die Registerkarte Permissions (Berechtigungen) aus.

   b. Wählen Sie Attach Policies (Richtlinien anfügen) aus.

   c. Geben Sie im Suchfeld **`AKReadSourceStreamWriteSinkStream`** (die Richtlinie, die Sie im vorhergehenden Abschnitt erstellt haben) ein.

   d. Wählen Sie die `AKReadSourceStreamWriteSinkStream`-Richtlinie und wählen Sie Richtlinie anhängen aus.

Sie haben nun die Service-Ausführungsrolle erstellt, die Ihre Anwendung für den Zugriff auf Ressourcen verwendet. Notieren Sie sich den ARN der neuen Rolle.

step-by-step Anweisungen zum Erstellen einer Rolle finden Sie unter Erstellen einer IAM-Rolle (Konsole) im IAM-Benutzerhandbuch.

Erstellen der Anwendung

Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `create_request.json`. Ersetzen Sie den Beispiel-Rollen-ARN durch den ARN für die Rolle, die Sie zuvor erstellt haben. Ersetzen Sie das Bucket-ARN-Suffix (username) mit dem Suffix, das Sie im vorherigen Abschnitt gewählt haben. Ersetzen Sie die beispielhafte Konto-ID (012345678901) in der Service-Ausführungsrolle mit Ihrer Konto-ID. Die `ServiceExecutionRole` sollte die IAM-Benutzerrolle enthalten, die Sie im vorherigen Abschnitt erstellt haben.

```
"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"
      }
   ]
}
```

Führen Sie die CreateApplication mit der folgenden Anforderung aus, um die Anwendung zu erstellen:

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

Die Anwendung wird nun erstellt. Sie starten die Anwendung im nächsten Schritt.

Starten der Anwendung

In diesem Abschnitt verwenden Sie die StartApplication-Aktion, um die Anwendung zu starten.

So starten Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `start_request.json`.

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

2. Führen Sie die `StartApplication`-Aktion mit der vorherigen Anforderung zum Starten der Anwendung aus:

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

Die Anwendung wird jetzt ausgeführt. Sie können die Metriken von Managed Service für Apache Flink in der Amazon- CloudWatch Konsole überprüfen, um zu überprüfen, ob die Anwendung funktioniert.

Stoppen der Anwendung

In diesem Abschnitt verwenden Sie die StopApplication-Aktion, um die Anwendung zu stoppen.

So stoppen Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `stop_request.json`.

```
{
   "ApplicationName": "tumbling_window"
}
```

2. Führen Sie die `StopApplication`-Aktion mit der vorherigen Anforderung zum Stoppen der Anwendung aus:

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

Die Anwendung wird nun gestoppt.

Hinzufügen einer CloudWatch Protokollierungsoption

Sie können die verwendenAWS CLI, um Ihrer Anwendung einen Amazon- CloudWatch Protokollstream hinzuzufügen. Informationen zur Verwendung von CloudWatch Protokollen mit Ihrer Anwendung finden Sie unter Einrichten der Anwendungsprotokollierung.

Umgebungseigenschaften aktualisieren

In diesem Abschnitt verwenden Sie die UpdateApplication-Aktion, um die Umgebungseigenschaften für die Anwendung zu ändern, ohne den Anwendungscode neu kompilieren zu müssen. In diesem Beispiel ändern Sie die Region der Quell- und Ziel-Streams.

So aktualisieren Sie die Umgebungseigenschaften für die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `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. Führen Sie die `UpdateApplication`-Aktion mit der vorherigen Anforderung aus, um die Umgebungseigenschaften zu aktualisieren:

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

Den Anwendungscode aktualisieren

Wenn Sie Ihren Anwendungscode mit einer neuen Version Ihres Codepakets aktualisieren müssen, verwenden Sie die UpdateApplication CLI-Aktion .

> **Note**
>
> Um eine neue Version des Anwendungscodes mit demselben Dateinamen zu laden, müssen Sie die neue Objektversion angeben. Weitere Informationen zur Verwendung von Amazon S3-Objektversionen finden Sie unter Versionierung aktivieren oder deaktivieren.

Um die AWS CLI zu verwenden, löschen Sie Ihr vorheriges Codepaket aus Ihrem Amazon S3-Bucket, laden Sie die neue Version hoch und rufen Sie `UpdateApplication` auf. Geben Sie dabei denselben Amazon S3-Bucket und Objektnamen sowie die neue Objektversion an. Die Anwendung wird mit dem neuen Codepaket neu gestartet.

Die folgende Beispielanforderung für die `UpdateApplication`-Aktion lädt den Anwendungscode neu und startet die Anwendung neu. Aktualisieren Sie die `CurrentApplicationVersionId` auf die aktuelle Anwendungsversion. Sie können die aktuelle Anwendungsversion mithilfe der Aktionen `ListApplications` oder `DescribeApplication` überprüfen. Aktualisieren Sie das

Bucket-Namenssuffix (<username>) mit dem Suffix, das Sie im Abschnitt Erstellen Sie abhängige Ressourcen ausgewählt haben.

```
{
    "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"
                }
            }
        }
    }
}
```

Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zur Bereinigung von AWS-Ressourcen, die im Rollierendes Fenster Tutorial erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte:

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Ihrer Kinesis Data Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen

Löschen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen Ihrer Kinesis Data Streams

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Kinesis Data Streams die Option ausExampleInputStream.
3. Wählen Sie auf der ExampleInputStream Seite Kinesis Stream löschen aus und bestätigen Sie dann den Löschvorgang.
4. Wählen Sie auf der Seite Kinesis Streams die , ExampleOutputStreamwählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den ka-app-codeBucket -***<username>*** aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle kinesis-analytics--MyApplication-us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Beispiel: Ein gleitendes Fenster in Scala erstellen

> **Note**
>
> Ab Version 1.15 ist Flink Scala-frei. Anwendungen können jetzt die Java-API von jeder Scala-Version aus verwenden. Flink verwendet Scala intern immer noch in einigen Schlüsselkomponenten, macht Scala jedoch nicht im Benutzercode-Classloader verfügbar. Aus diesem Grund müssen Benutzer Scala-Abhängigkeiten zu ihren Jar-Archiven hinzufügen. Weitere Informationen zu den Scala-Änderungen in Flink 1.15 finden Sie unter Scalafrei in One Fifteen.

In dieser Übung erstellen Sie eine einfache Streaming-Anwendung, die Scala 3.2.0 und die Java DataStream -API von Flink verwendet. Die Anwendung liest Daten aus dem Kinesis Stream, aggregiert sie mithilfe von gleitenden Fenstern und schreibt die Ergebnisse in den Kinesis-Ausgabestream.

> **Note**
>
> Um die erforderlichen Voraussetzungen für diese Übung einzurichten, schließen Sie zunächst die Übung Erste Schritte (Scala) ab.

Dieses Thema enthält die folgenden Abschnitte:

- Laden Sie den Anwendungscode herunter und überprüfen Sie ihn
- Kompilieren Sie den Anwendungscode und laden Sie ihn hoch
- Erstellen und Ausführen der Anwendung (Konsole)
- Erstellen und Ausführen der Anwendung (CLI)
- Den Anwendungscode aktualisieren
- Bereinigen von AWS-Ressourcen

Laden Sie den Anwendungscode herunter und überprüfen Sie ihn

Der Python-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben. Weitere Informationen finden Sie unter Git installieren.
2. Klonen Sie das Remote-Repository mit dem folgenden Befehl:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/scala/SlidingWindow` Verzeichnis .

Beachten Sie Folgendes zum Anwendungscode:

- Eine `build.sbt`-Datei enthält Informationen über die Konfiguration und Abhängigkeiten der Anwendung, einschließlich der Bibliotheken des Managed Service für Apache Flink.
- Die `BasicStreamingJob.scala`-Datei enthält die Hauptmethode, die die Funktionalität der Anwendung definiert.
- Die Anwendung verwendet eine Kinesis-Quelle zum Lesen aus dem Quell-Stream. Der folgende Codeausschnitt erstellt die Kinesis-Quelle:

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
 defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

  Die Anwendung verwendet auch eine Kinesis-Senke, um in den Ergebnisstream zu schreiben. Der folgende Codeausschnitt erstellt die Kinesis-Senke:

```
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
}
```

- Die Anwendung verwendet den Fensteroperator, um die Anzahl der Werte für jedes Aktionssymbol über ein 10-Sekunden-Fenster, das um 5 Sekunden gleitet, zu ermitteln. Der folgende Code erstellt den Operator und sendet die aggregierten Daten an eine neue Kinesis Data Streams Senke:

```
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)
```

- Die Anwendung erstellt Quell- und Senken-Konnektoren, um über ein -StreamExecutionEnvironment Objekt auf externe Ressourcen zuzugreifen.
- Die Anwendung erstellt Quell- und Senkenkonnektoren mit dynamischen Anwendungseigenschaften. Die Laufzeiteigenschaften der Anwendung werden gelesen, um die Konnektoren zu konfigurieren. Weitere Informationen zu Laufzeiteigenschaften finden Sie unter Laufzeiteigenschaften.

Kompilieren Sie den Anwendungscode und laden Sie ihn hoch

In diesem Abschnitt kompilieren Sie Ihren Anwendungscode und laden ihn in einen Amazon-S3-Bucket hoch.

Kompilieren des Anwendungscodes

Verwenden Sie das SBT-Build-Tool, um den Scala-Code für die Anwendung zu erstellen. Informationen zur Installation von SBT finden Sie unter Installieren von SBT mit CS-Setup. Sie müssen auch das Java Development Kits (JDK) installieren. Siehe Voraussetzungen für das Fertigstellen der Übungen

1. Zum Verwenden Ihres Anwendungscodes kompilieren und packen Sie ihn in eine JAR-Datei. Sie können Ihren Code mit SBT kompilieren und verpacken:

```
sbt assembly
```

2. Wenn die Anwendung erfolgreich kompiliert wurde, wird die folgende Datei erstellt:

```
target/scala-3.2.0/sliding-window-scala-1.0.jar
```

Hochladen des Apache Flink-Streaming-Scala-Codes

In diesem Abschnitt erstellen Sie einen Amazon S3-Bucket und laden Ihren Anwendungscode hoch.

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie Bucket erstellen aus
3. Geben Sie `ka-app-code-<username>` im Feld Bucket-Name ein. Fügen Sie dem Bucket-Namen ein Suffix hinzu, wie z. B. Ihren Benutzernamen, damit er global eindeutig ist. Wählen Sie Weiter aus.
4. Lassen Sie im Schritt Optionen konfigurieren die Einstellungen unverändert und klicken Sie auf Weiter.
5. Lassen Sie im Schritt Berechtigungen festlegen die Einstellungen unverändert und klicken Sie auf Weiter.
6. Wählen Sie Bucket erstellen aus.
7. Wählen Sie den Bucket `ka-app-code-<username>` und dann Hochladen aus.
8. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `sliding-window-scala-1.0.jar`Datei, die Sie im vorherigen Schritt erstellt haben.
9. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und Ausführen der Anwendung (Konsole)

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Geben Sie für Beschreibung den Text **`My Scala test app`** ein.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.
   - Belassen Sie die Version als Apache Flink Version 1.15.2 (empfohlene Version).
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rolle: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Konfigurieren der Anwendung

Verwenden Sie das folgende Verfahren, um die Anwendung zu konfigurieren.

Zum Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):

- Geben Sie für Amazon-S3-Bucket **ka-app-code-*<username>*** ein.
- Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **sliding-window-scala-1.0.jar.** ein.

3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **kinesis-analytics-MyApplication-us-west-2** erstellen/aktualisieren aus.
4. Wählen Sie unter Eigenschaften die Option Gruppe hinzufügen aus.
5. Geben Sie Folgendes ein:

Gruppen-ID	Schlüssel	Wert
ConsumerConfigProperties	**input.stream.name**	**ExampleInputStream**
ConsumerConfigProperties	**aws.region**	**us-west-2**
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

   Wählen Sie Speichern.
6. Wählen Sie unter Eigenschaften erneut Gruppe hinzufügen aus.
7. Geben Sie Folgendes ein:

Gruppen-ID	Schlüssel	Wert
ProducerConfigProperties	**output.stream.name**	**ExampleOutputStream**
ProducerConfigProperties	**aws.region**	**us-west-2**

8. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
9. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
10. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die Amazon- CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf den Amazon S3-Bucket.

Um die IAM-Richtlinie zu bearbeiten, um S3-Bucket-Berechtigungen hinzuzufügen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **`kinesis-analytics-service-MyApplication-us-west-2`**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*012345678901*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

Ausführen der Anwendung

Das Flink-Jobdiagramm kann angezeigt werden, indem Sie die Anwendung ausführen, das Apache Flink-Dashboard öffnen und den gewünschten Flink-Job auswählen.

Stoppen der Anwendung

Um die Anwendung zu beenden, wählen Sie auf der MyApplication Seite Stoppen aus. Bestätigen Sie die Aktion.

Erstellen und Ausführen der Anwendung (CLI)

In diesem Abschnitt verwenden Sie AWS Command Line Interface, um die Anwendung Managed Service für Apache Flink zu erstellen und auszuführen. Verwenden Sie den AWS CLI-Befehl kinesisanalyticsv2, um Managed Service für Apache Flink-Anwendungen zu erstellen und mit diesen zu interagieren.

Erstellen einer Berechtigungsrichtlinie

> **Note**
>
> Sie müssen eine Berechtigungsrichtlinie und eine Rolle für Ihre Anwendung erstellen. Wenn Sie diese IAM-Ressourcen nicht erstellen, kann Ihre Anwendung nicht auf ihre Daten und Protokollstreams zugreifen.

Zuerst erstellen Sie eine Berechtigungsrichtlinie mit zwei Anweisungen: eine, die Berechtigungen für die Lese-Aktion auf den Quell-Stream zulässt, und eine andere, die Berechtigungen für die Schreib-Aktionen auf den Senken-Stream zulässt. Anschließend fügen Sie die Richtlinie an eine IAM-Rolle (die Sie im nächsten Abschnitt erstellen) an. Wenn Managed Service für Apache Flink also die Rolle übernimmt, verfügt der Service über die erforderlichen Berechtigungen zum Lesen aus dem Quell-Stream und zum Schreiben in den Senken-Stream.

Verwenden Sie den folgenden Code zum Erstellen der `AKReadSourceStreamWriteSinkStream`-Berechtigungsrichtlinie. Ersetzen Sie **`username`** durch den Benutzernamen, den Sie verwendet haben, um den Amazon-S3-Bucket zum Speichern des Anwendungscodes zu erstellen. Ersetzen Sie die Konto-ID in den Amazon-Ressourcennamen (ARNs) **`(012345678901)`** mit Ihrer Konto-ID.

```
{
    "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"
      }
   ]
}
```

step-by-step Anweisungen zum Erstellen einer Berechtigungsrichtlinie finden Sie unter Tutorial: Erstellen und Anfügen Ihrer ersten vom Kunden verwalteten Richtlinie im IAM-Benutzerhandbuch.

Erstellen einer IAM-Rolle

In diesem Abschnitt erstellen Sie eine IAM-Rolle, die die Anwendung von Managed Service für Apache Flink annehmen kann, um einen Quell-Stream zu lesen und in den Senken-Stream zu schreiben.

Managed Service für Apache Flink kann ohne Berechtigungen nicht auf Ihren Stream zugreifen. Sie erteilen diese Berechtigungen über eine IAM-Rolle. Jeder IAM-Rolle sind zwei Richtlinien angefügt. Die Vertrauensrichtlinie erteilt Managed Service für Apache Flink die Berechtigung zum Übernehmen der Rolle und die Berechtigungsrichtlinie bestimmt, was Managed Service für Apache Flink nach Annahme der Rolle tun kann.

Sie können die Berechtigungsrichtlinie, die Sie im vorherigen Abschnitt erstellt haben, dieser Rolle anfügen.

So erstellen Sie eine IAM-Rolle

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Rollen und dann Rolle erstellen.
3. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option AWS-Service aus
4. Wählen Sie unter Choose the service that will use this role (Wählen Sie den Service aus, der diese Rolle verwendet) die Option Kinesis aus.
5. Wählen Sie unter Wählen Sie Ihren Anwendungsfall aus die Option Managed Service für Apache Flink aus.
6. Wählen Sie Weiter: Berechtigungen aus.
7. Wählen Sie auf der Seite Attach permissions policies (Berechtigungsrichtlinien hinzufügen) Next: Review (Weiter: Überprüfen) aus. Sie fügen Berechtigungsrichtlinien an, nachdem Sie die Rolle erstellt haben.

8. Geben Sie auf der Seite Create role (Rolle erstellen) den Text **`MF-stream-rw-role`** für Role name (Rollenname) ein. Wählen Sie Rolle erstellen aus.

   Jetzt haben Sie eine neue IAM-Rolle mit dem Namen `MF-stream-rw-role` erstellt. Im nächsten Schritt aktualisieren Sie die Vertrauens- und Berechtigungsrichtlinien für die Rolle
9. Fügen Sie die Berechtigungsrichtlinie der Rolle an.

   > **Note**
   >
   > Für diese Übung übernimmt Managed Service für Apache Flink diese Rolle sowohl für das Lesen von Daten aus einem Kinesis-Datenstrom (Quelle) als auch zum Schreiben der Ausgabedaten in einen anderen Kinesis-Datenstrom. Daher fügen Sie die Richtlinie an, die Sie im vorherigen Schritt, Erstellen einer Berechtigungsrichtlinie, erstellt haben.

   a. Wählen Sie auf der Seite Summary (Übersicht) die Registerkarte Permissions (Berechtigungen) aus.
   b. Wählen Sie Attach Policies (Richtlinien anfügen) aus.
   c. Geben Sie im Suchfeld **`AKReadSourceStreamWriteSinkStream`** (die Richtlinie, die Sie im vorhergehenden Abschnitt erstellt haben) ein.
   d. Wählen Sie die `AKReadSourceStreamWriteSinkStream`-Richtlinie und wählen Sie Richtlinie anhängen aus.

Sie haben nun die Service-Ausführungsrolle erstellt, die Ihre Anwendung für den Zugriff auf Ressourcen verwendet. Notieren Sie sich den ARN der neuen Rolle.

step-by-step Anweisungen zum Erstellen einer Rolle finden Sie unter Erstellen einer IAM-Rolle (Konsole) im IAM-Benutzerhandbuch.

Erstellen der Anwendung

Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `create_request.json`. Ersetzen Sie den Beispiel-Rollen-ARN durch den ARN für die Rolle, die Sie zuvor erstellt haben. Ersetzen Sie das Bucket-ARN-Suffix (username) mit dem Suffix, das Sie im vorherigen Abschnitt gewählt haben. Ersetzen Sie die beispielhafte Konto-ID (012345678901) in der Service-Ausführungsrolle mit Ihrer Konto-ID.

```
{
```

```
    "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"
      }
   ]
}
```

Führen Sie die CreateApplication mit der folgenden Anforderung aus, um die Anwendung zu erstellen:

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

Die Anwendung wird nun erstellt. Sie starten die Anwendung im nächsten Schritt.

Starten der Anwendung

In diesem Abschnitt verwenden Sie die StartApplication-Aktion, um die Anwendung zu starten.

So starten Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `start_request.json`.

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

2. Führen Sie die `StartApplication`-Aktion mit der vorherigen Anforderung zum Starten der Anwendung aus:

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

Die Anwendung wird jetzt ausgeführt. Sie können die Metriken von Managed Service für Apache Flink in der Amazon- CloudWatch Konsole überprüfen, um zu überprüfen, ob die Anwendung funktioniert.

Stoppen der Anwendung

In diesem Abschnitt verwenden Sie die StopApplication-Aktion, um die Anwendung zu stoppen.

So stoppen Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `stop_request.json`.

```
{
   "ApplicationName": "sliding_window"
}
```

2. Führen Sie die `StopApplication`-Aktion mit der vorherigen Anforderung zum Stoppen der Anwendung aus:

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

Die Anwendung wird nun gestoppt.

Hinzufügen einer CloudWatch Protokollierungsoption

Sie können die verwendenAWS CLI, um Ihrer Anwendung einen Amazon- CloudWatch Protokollstream hinzuzufügen. Informationen zur Verwendung von CloudWatch Protokollen mit Ihrer Anwendung finden Sie unter Einrichten der Anwendungsprotokollierung.

Umgebungseigenschaften aktualisieren

In diesem Abschnitt verwenden Sie die UpdateApplication-Aktion, um die Umgebungseigenschaften für die Anwendung zu ändern, ohne den Anwendungscode neu kompilieren zu müssen. In diesem Beispiel ändern Sie die Region der Quell- und Ziel-Streams.

So aktualisieren Sie die Umgebungseigenschaften für die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `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. Führen Sie die `UpdateApplication`-Aktion mit der vorherigen Anforderung aus, um die Umgebungseigenschaften zu aktualisieren:

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

Den Anwendungscode aktualisieren

Wenn Sie Ihren Anwendungscode mit einer neuen Version Ihres Codepakets aktualisieren müssen, verwenden Sie die UpdateApplication CLI-Aktion .

> **Note**
>
> Um eine neue Version des Anwendungscodes mit demselben Dateinamen zu laden, müssen Sie die neue Objektversion angeben. Weitere Informationen zur Verwendung von Amazon S3-Objektversionen finden Sie unter Versionierung aktivieren oder deaktivieren.

Um die AWS CLI zu verwenden, löschen Sie Ihr vorheriges Codepaket aus Ihrem Amazon S3-Bucket, laden Sie die neue Version hoch und rufen Sie `UpdateApplication` auf. Geben Sie dabei denselben Amazon S3-Bucket und Objektnamen sowie die neue Objektversion an. Die Anwendung wird mit dem neuen Codepaket neu gestartet.

Die folgende Beispielanforderung für die `UpdateApplication`-Aktion lädt den Anwendungscode neu und startet die Anwendung neu. Aktualisieren Sie die `CurrentApplicationVersionId` auf die aktuelle Anwendungsversion. Sie können die aktuelle Anwendungsversion mithilfe der Aktionen `ListApplications` oder `DescribeApplication` überprüfen. Aktualisieren Sie das

Bucket-Namenssuffix (<username>) mit dem Suffix, das Sie im Abschnitt Erstellen Sie abhängige Ressourcen ausgewählt haben.

```
{
    "ApplicationName": "sliding_window",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "-1.0.jar",
                    "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
                }
            }
        }
    }
}
```

Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zum Bereinigen von AWS-Ressourcen, die im Tutorial Gleitendes Fenster erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte:

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Ihrer Kinesis Data Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen

Löschen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen Ihrer Kinesis Data Streams

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Kinesis Data Streams die Option ausExampleInputStream.
3. Wählen Sie auf der ExampleInputStream Seite Kinesis Stream löschen aus und bestätigen Sie dann den Löschvorgang.
4. Wählen Sie auf der Seite Kinesis Streams die , ExampleOutputStreamwählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den ka-app-codeBucket -***<username>*** aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle kinesis-analytics--MyApplication-us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Beispiel: Streaming-Daten in Scala an Amazon S3 senden

> **Note**
>
> Ab Version 1.15 ist Flink Scala-frei. Anwendungen können jetzt die Java-API von jeder Scala-Version aus verwenden. Flink verwendet Scala intern immer noch in einigen Schlüsselkomponenten, macht Scala jedoch nicht im Benutzercode-Classloader verfügbar. Aus diesem Grund müssen Benutzer Scala-Abhängigkeiten zu ihren Jar-Archiven hinzufügen. Weitere Informationen zu den Scala-Änderungen in Flink 1.15 finden Sie unter Scalafrei in One Fifteen.

In dieser Übung erstellen Sie eine einfache Streaming-Anwendung, die Scala 3.2.0 und die Java DataStream -API von Flink verwendet. Die Anwendung liest Daten aus dem Kinesis Stream, aggregiert sie mithilfe von gleitenden Fenstern und schreibt die Ergebnisse in S3.

> **Note**
>
> Um die erforderlichen Voraussetzungen für diese Übung einzurichten, schließen Sie zunächst die Übung Erste Schritte (Scala) ab. Sie müssen nur einen zusätzlichen Ordner **`data/`** im Amazon S3ka-app-code-Bucket erstellen –<username>.

Dieses Thema enthält die folgenden Abschnitte:

- Laden Sie den Anwendungscode herunter und überprüfen Sie ihn
- Kompilieren Sie den Anwendungscode und laden Sie ihn hoch
- Erstellen und Ausführen der Anwendung (Konsole)
- Erstellen und Ausführen der Anwendung (CLI)
- Den Anwendungscode aktualisieren
- Bereinigen von AWS-Ressourcen

Laden Sie den Anwendungscode herunter und überprüfen Sie ihn

Der Python-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben. Weitere Informationen finden Sie unter Git installieren.
2. Klonen Sie das Remote-Repository mit dem folgenden Befehl:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

3. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/scala/S3Sink` Verzeichnis .

Beachten Sie Folgendes zum Anwendungscode:

- Eine `build.sbt`-Datei enthält Informationen über die Konfiguration und Abhängigkeiten der Anwendung, einschließlich der Bibliotheken des Managed Service für Apache Flink.
- Die `BasicStreamingJob.scala`-Datei enthält die Hauptmethode, die die Funktionalität der Anwendung definiert.
- Die Anwendung verwendet eine Kinesis-Quelle zum Lesen aus dem Quell-Stream. Der folgende Codeausschnitt erstellt die Kinesis-Quelle:

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
 defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

Die Anwendung verwendet auch eine StreamingFileSink , um in einen Amazon S3-Bucket zu schreiben:`

```
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()
}
```

- Die Anwendung erstellt Quell- und Senken-Konnektoren, um über ein -StreamExecutionEnvironment Objekt auf externe Ressourcen zuzugreifen.
- Die Anwendung erstellt Quell- und Senkenkonnektoren mit dynamischen Anwendungseigenschaften. Die Laufzeiteigenschaften der Anwendung werden gelesen, um die Konnektoren zu konfigurieren. Weitere Informationen zu Laufzeiteigenschaften finden Sie unter Laufzeiteigenschaften.

Kompilieren Sie den Anwendungscode und laden Sie ihn hoch

In diesem Abschnitt kompilieren Sie Ihren Anwendungscode und laden ihn in einen Amazon-S3-Bucket hoch.

Kompilieren des Anwendungscodes

Verwenden Sie das SBT-Build-Tool, um den Scala-Code für die Anwendung zu erstellen. Informationen zur Installation von SBT finden Sie unter Installieren von SBT mit CS-Setup. Sie müssen auch das Java Development Kits (JDK) installieren. Siehe Voraussetzungen für das Fertigstellen der Übungen

1. Zum Verwenden Ihres Anwendungscodes kompilieren und packen Sie ihn in eine JAR-Datei. Sie können Ihren Code mit SBT kompilieren und verpacken:

```
sbt assembly
```

2. Wenn die Anwendung erfolgreich kompiliert wurde, wird die folgende Datei erstellt:

```
target/scala-3.2.0/s3-sink-scala-1.0.jar
```

Hochladen des Apache Flink-Streaming-Scala-Codes

In diesem Abschnitt erstellen Sie einen Amazon S3-Bucket und laden Ihren Anwendungscode hoch.

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie Bucket erstellen aus
3. Geben Sie `ka-app-code-<username>` im Feld Bucket-Name ein. Fügen Sie dem Bucket-Namen ein Suffix hinzu, wie z. B. Ihren Benutzernamen, damit er global eindeutig ist. Wählen Sie Weiter aus.

4. Lassen Sie im Schritt Optionen konfigurieren die Einstellungen unverändert und klicken Sie auf Weiter.
5. Lassen Sie im Schritt Berechtigungen festlegen die Einstellungen unverändert und klicken Sie auf Weiter.
6. Wählen Sie Bucket erstellen aus.
7. Wählen Sie den Bucket `ka-app-code-<username>` und dann Hochladen aus.
8. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `s3-sink-scala-1.0.jar`Datei, die Sie im vorherigen Schritt erstellt haben.
9. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und Ausführen der Anwendung (Konsole)

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Geben Sie für Beschreibung den Text **`My java test app`** ein.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.
   - Belassen Sie die Version als Apache Flink Version 1.15.2 (empfohlene Version).
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-MyApplication-us-west-2`
> - Rolle: `kinesisanalytics-MyApplication-us-west-2`

Konfigurieren der Anwendung

Verwenden Sie das folgende Verfahren, um die Anwendung zu konfigurieren.

Zum Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):
   - Geben Sie für Amazon-S3-Bucket **`ka-app-code-<username>`** ein.
   - Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **`s3-sink-scala-1.0.jar`** ein.
3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** erstellen/aktualisieren aus.
4. Wählen Sie unter Eigenschaften die Option Gruppe hinzufügen aus.
5. Geben Sie Folgendes ein:

Gruppen-ID	Schlüssel	Wert
`ConsumerConfigProperties`	**`input.stream.name`**	**`ExampleInputStream`**
`ConsumerConfigProperties`	**`aws.region`**	**`us-west-2`**

Gruppen-ID	Schlüssel	Wert
ConsumerConfigProperties	**flink.stream.initpos**	**LATEST**

Wählen Sie Speichern.

6. Wählen Sie unter Eigenschaften die Option Gruppe hinzufügen aus.
7. Geben Sie Folgendes ein:

Gruppen-ID	Schlüssel	Wert
ProducerConfigProperties	**s3.sink.path**	**s3a://ka-app-code-*<user-name>* /data**

8. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
9. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
10. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die Amazon- CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf den Amazon S3-Bucket.

Um die IAM-Richtlinie zu bearbeiten, um S3-Bucket-Berechtigungen hinzuzufügen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **`kinesis-analytics-service-MyApplication-us-west-2`**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*012345678901*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

Ausführen der Anwendung

Das Flink-Jobdiagramm kann angezeigt werden, indem Sie die Anwendung ausführen, das Apache Flink-Dashboard öffnen und den gewünschten Flink-Job auswählen.

Stoppen der Anwendung

Um die Anwendung zu beenden, wählen Sie auf der MyApplication Seite Stoppen aus. Bestätigen Sie die Aktion.

Erstellen und Ausführen der Anwendung (CLI)

In diesem Abschnitt verwenden Sie AWS Command Line Interface, um die Anwendung Managed Service für Apache Flink zu erstellen und auszuführen. Verwenden Sie den AWS CLI-Befehl kinesisanalyticsv2, um Managed Service für Apache Flink-Anwendungen zu erstellen und mit diesen zu interagieren.

Erstellen einer Berechtigungsrichtlinie

> **Note**
>
> Sie müssen eine Berechtigungsrichtlinie und eine Rolle für Ihre Anwendung erstellen. Wenn Sie diese IAM-Ressourcen nicht erstellen, kann Ihre Anwendung nicht auf ihre Daten und Protokollstreams zugreifen.

Zuerst erstellen Sie eine Berechtigungsrichtlinie mit zwei Anweisungen: eine, die Berechtigungen für die Lese-Aktion auf den Quell-Stream zulässt, und eine andere, die Berechtigungen für die Schreib-Aktionen auf den Senken-Stream zulässt. Anschließend fügen Sie die Richtlinie an eine IAM-Rolle (die Sie im nächsten Abschnitt erstellen) an. Wenn Managed Service für Apache Flink also die Rolle übernimmt, verfügt der Service über die erforderlichen Berechtigungen zum Lesen aus dem Quell-Stream und zum Schreiben in den Senken-Stream.

Verwenden Sie den folgenden Code zum Erstellen der `AKReadSourceStreamWriteSinkStream`-Berechtigungsrichtlinie. Ersetzen Sie **`username`** durch den Benutzernamen, den Sie verwendet haben, um den Amazon-S3-Bucket zum Speichern des Anwendungscodes zu erstellen. Ersetzen Sie die Konto-ID in den Amazon-Ressourcennamen (ARNs) **`(012345678901)`** mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

step-by-step Anweisungen zum Erstellen einer Berechtigungsrichtlinie finden Sie unter Tutorial: Erstellen und Anfügen Ihrer ersten vom Kunden verwalteten Richtlinie im IAM-Benutzerhandbuch.

Erstellen einer IAM-Rolle

In diesem Abschnitt erstellen Sie eine IAM-Rolle, die die Anwendung von Managed Service für Apache Flink annehmen kann, um einen Quell-Stream zu lesen und in den Senken-Stream zu schreiben.

Managed Service für Apache Flink kann ohne Berechtigungen nicht auf Ihren Stream zugreifen. Sie erteilen diese Berechtigungen über eine IAM-Rolle. Jeder IAM-Rolle sind zwei Richtlinien angefügt. Die Vertrauensrichtlinie erteilt Managed Service für Apache Flink die Berechtigung zum Übernehmen der Rolle und die Berechtigungsrichtlinie bestimmt, was Managed Service für Apache Flink nach Annahme der Rolle tun kann.

Sie können die Berechtigungsrichtlinie, die Sie im vorherigen Abschnitt erstellt haben, dieser Rolle anfügen.

So erstellen Sie eine IAM-Rolle

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Rollen und dann Rolle erstellen.
3. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option AWS-Service aus
4. Wählen Sie unter Choose the service that will use this role (Wählen Sie den Service aus, der diese Rolle verwendet) die Option Kinesis aus.
5. Wählen Sie unter Wählen Sie Ihren Anwendungsfall aus die Option Managed Service für Apache Flink aus.
6. Wählen Sie Weiter: Berechtigungen aus.
7. Wählen Sie auf der Seite Attach permissions policies (Berechtigungsrichtlinien hinzufügen) Next: Review (Weiter: Überprüfen) aus. Sie fügen Berechtigungsrichtlinien an, nachdem Sie die Rolle erstellt haben.

8. Geben Sie auf der Seite Create role (Rolle erstellen) den Text **`MF-stream-rw-role`** für Role name (Rollenname) ein. Wählen Sie Rolle erstellen aus.

   Jetzt haben Sie eine neue IAM-Rolle mit dem Namen `MF-stream-rw-role` erstellt. Im nächsten Schritt aktualisieren Sie die Vertrauens- und Berechtigungsrichtlinien für die Rolle
9. Fügen Sie die Berechtigungsrichtlinie der Rolle an.

   > **Note**
   >
   > Für diese Übung übernimmt Managed Service für Apache Flink diese Rolle sowohl für das Lesen von Daten aus einem Kinesis-Datenstrom (Quelle) als auch zum Schreiben der Ausgabedaten in einen anderen Kinesis-Datenstrom. Daher fügen Sie die Richtlinie an, die Sie im vorherigen Schritt, Erstellen einer Berechtigungsrichtlinie, erstellt haben.

   a. Wählen Sie auf der Seite Summary (Übersicht) die Registerkarte Permissions (Berechtigungen) aus.
   b. Wählen Sie Attach Policies (Richtlinien anfügen) aus.
   c. Geben Sie im Suchfeld **`AKReadSourceStreamWriteSinkStream`** (die Richtlinie, die Sie im vorhergehenden Abschnitt erstellt haben) ein.
   d. Wählen Sie die `AKReadSourceStreamWriteSinkStream`-Richtlinie und wählen Sie Richtlinie anhängen aus.

Sie haben nun die Service-Ausführungsrolle erstellt, die Ihre Anwendung für den Zugriff auf Ressourcen verwendet. Notieren Sie sich den ARN der neuen Rolle.

step-by-step Anweisungen zum Erstellen einer Rolle finden Sie unter Erstellen einer IAM-Rolle (Konsole) im IAM-Benutzerhandbuch.

Erstellen der Anwendung

Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `create_request.json`. Ersetzen Sie den Beispiel-Rollen-ARN durch den ARN für die Rolle, die Sie zuvor erstellt haben. Ersetzen Sie das Bucket-ARN-Suffix (username) mit dem Suffix, das Sie im vorherigen Abschnitt gewählt haben. Ersetzen Sie die beispielhafte Konto-ID (012345678901) in der Service-Ausführungsrolle mit Ihrer Konto-ID.

```
{
```

```
    "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"
      }
   ]
}
```

Führen Sie die CreateApplication mit der folgenden Anforderung aus, um die Anwendung zu erstellen:

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

Die Anwendung wird nun erstellt. Sie starten die Anwendung im nächsten Schritt.

Starten der Anwendung

In diesem Abschnitt verwenden Sie die StartApplication-Aktion, um die Anwendung zu starten.

So starten Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `start_request.json`.

```
{{
    "ApplicationName": "s3_sink",
    "RunConfiguration": {
        "ApplicationRestoreConfiguration": {
         "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
         }
    }
}
```

2. Führen Sie die `StartApplication`-Aktion mit der vorherigen Anforderung zum Starten der Anwendung aus:

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

Die Anwendung wird jetzt ausgeführt. Sie können die Metriken von Managed Service für Apache Flink in der Amazon- CloudWatch Konsole überprüfen, um zu überprüfen, ob die Anwendung funktioniert.

Stoppen der Anwendung

In diesem Abschnitt verwenden Sie die StopApplication-Aktion, um die Anwendung zu stoppen.

So stoppen Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `stop_request.json`.

```
{
   "ApplicationName": "s3_sink"
```

```
}
```

2. Führen Sie die `StopApplication`-Aktion mit der vorherigen Anforderung zum Stoppen der Anwendung aus:

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

Die Anwendung wird nun gestoppt.

Hinzufügen einer CloudWatch Protokollierungsoption

Sie können die verwendenAWS CLI, um Ihrer Anwendung einen Amazon- CloudWatch Protokollstream hinzuzufügen. Informationen zur Verwendung von CloudWatch Protokollen mit Ihrer Anwendung finden Sie unter Einrichten der Anwendungsprotokollierung.

Umgebungseigenschaften aktualisieren

In diesem Abschnitt verwenden Sie die UpdateApplication-Aktion, um die Umgebungseigenschaften für die Anwendung zu ändern, ohne den Anwendungscode neu kompilieren zu müssen. In diesem Beispiel ändern Sie die Region der Quell- und Ziel-Streams.

So aktualisieren Sie die Umgebungseigenschaften für die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `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. Führen Sie die `UpdateApplication`-Aktion mit der vorherigen Anforderung aus, um die Umgebungseigenschaften zu aktualisieren:

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

Den Anwendungscode aktualisieren

Wenn Sie Ihren Anwendungscode mit einer neuen Version Ihres Codepakets aktualisieren müssen, verwenden Sie die UpdateApplication CLI-Aktion .

> **Note**
>
> Um eine neue Version des Anwendungscodes mit demselben Dateinamen zu laden, müssen Sie die neue Objektversion angeben. Weitere Informationen zur Verwendung von Amazon S3-Objektversionen finden Sie unter Versionierung aktivieren oder deaktivieren.

Um die AWS CLI zu verwenden, löschen Sie Ihr vorheriges Codepaket aus Ihrem Amazon S3-Bucket, laden Sie die neue Version hoch und rufen Sie `UpdateApplication` auf. Geben Sie dabei denselben Amazon S3-Bucket und Objektnamen sowie die neue Objektversion an. Die Anwendung wird mit dem neuen Codepaket neu gestartet.

Die folgende Beispielanforderung für die `UpdateApplication`-Aktion lädt den Anwendungscode neu und startet die Anwendung neu. Aktualisieren Sie die `CurrentApplicationVersionId` auf die aktuelle Anwendungsversion. Sie können die aktuelle Anwendungsversion mithilfe der Aktionen `ListApplications` oder `DescribeApplication` überprüfen. Aktualisieren Sie das Bucket-Namenssuffix (<username>) mit dem Suffix, das Sie im Abschnitt Erstellen Sie abhängige Ressourcen ausgewählt haben.

```
{
```

```
    "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"
                }
            }
        }
    }
}
```

Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zur Bereinigung von AWS-Ressourcen, die im Rollierendes Fenster-Tutorial erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte:

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Ihrer Kinesis Data Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen

Löschen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen Ihrer Kinesis Data Streams

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.

2. Wählen Sie im Bereich Kinesis Data Streams die Option ausExampleInputStream.
3. Wählen Sie auf der ExampleInputStream Seite Kinesis Stream löschen aus und bestätigen Sie dann den Löschvorgang.
4. Wählen Sie auf der Seite Kinesis Streams die , ExampleOutputStreamwählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den ka-app-codeBucket -***<username>*** aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle kinesis-analytics-MyApplication--us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Amazon Managed Service für Apache Flink verwenden

Die Sicherheit in der Cloud hat für AWS höchste Priorität. Als AWS-Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die eingerichtet wurde, um die Anforderungen der anspruchsvollsten Organisationen in puncto Sicherheit zu erfüllen.

Sicherheit gilt zwischen AWS und Ihnen eine geteilte Verantwortung. Das Modell der geteilten Verantwortung beschreibt dies als Sicherheit der Cloud und Sicherheit in der Cloud:

- Sicherheit der Cloud: AWS ist dafür verantwortlich, die Infrastruktur zu schützen, mit der AWS-Services in der AWS-Cloud ausgeführt werden. AWS stellt Ihnen außerdem Services bereit, die Sie sicher nutzen können. Die Wirksamkeit unserer Sicherheitsfunktionen wird regelmäßig von externen Prüfern im Rahmen des AWS-Compliance-Programms getestet und überprüft. Weitere Informationen zu den für Managed Service für Apache Flink geltenden Compliance-Programmen finden Sie unter Im Rahmen des Compliance-Programms zugelassene AWS-Services.
- Sicherheit in der Cloud – Ihr Verantwortungsumfang wird durch den AWS-Service bestimmt, den Sie verwenden. In Ihre Verantwortung fallen außerdem weitere Faktoren, wie z. B. die Vertraulichkeit der Daten, die Anforderungen Ihrer Organisation sowie geltende Gesetze und Vorschriften.

Diese Dokumentation hilft Ihnen zu verstehen, wie Sie das Modell der geteilten Verantwortung bei der Verwendung von Managed Service für Apache Flink einsetzen können. Die folgenden Themen veranschaulichen, wie Sie Managed Service für Apache Flink zur Erfüllung Ihrer Sicherheits- und Compliance-Ziele konfigurieren können. Sie erfahren außerdem, wie Sie andere Amazon-Services verwenden können, die Ihnen beim Überwachen und Sichern Ihrer Managed Service für Apache Flink-Ressourcen helfen.

Themen

- Datenschutz in Amazon Managed Service für Apache Flink
- Identity and Access Management für Amazon Managed Service für Apache Flink
- Überwachung von Managed Service für Apache Flink
- Compliance-Validierung für Amazon Managed Service für Apache Flink
- Ausfallsicherheit in Amazon Managed Service für Apache Flink
- Infrastruktursicherheit in Managed Service für Apache Flink
- Bewährte Methoden für die Sicherheit für Managed Service für Apache Flink

Datenschutz in Amazon Managed Service für Apache Flink

Sie können Ihre Daten mithilfe von Tools schützen, die von AWS bereitgestellt werden. Managed Service für Apache Flink kann mit Services zusammenarbeiten, die die Verschlüsselung von Daten unterstützen, einschließlich Kinesis Data Firehose und Amazon S3.

Datenverschlüsselung im Managed Service für Apache Flink

Verschlüsselung im Ruhezustand

Beachten Sie die folgenden Informationen zur Verschlüsselung von Daten im Ruhezustand mit Managed Service für Apache Flink:

- Sie können Daten im eingehenden Kinesis-Datenstrom mit verschlüsselnStartStreamEncryption. Weitere Informationen finden Sie unter Was bedeutet eine serverseitige Verschlüsselung in Kinesis-Daten-Streams?.
- Ausgabedaten können mithilfe von Kinesis Data Firehose zum Speichern von Daten im Ruhezustand verschlüsselt werden, um Daten in einem verschlüsselten Amazon S3-Bucket zu speichern. Sie können den Verschlüsselungsschlüssel angeben, der von Ihrem Amazon S3-Bucket verwendet wird. Weitere Informationen hierzu finden Sie unter Daten schützen durch serverseitige Verschlüsselung mit KMS-verwalteten Schlüsseln (SSE-KMS).
- Managed Service für Apache Flink kann von jeder Streaming-Quelle lesen und in jedes Streaming- oder Datenbankziel schreiben. Stellen Sie sicher, dass Ihre Quellen und Ziele alle Daten während der Übertragung und alle Daten im Ruhezustand verschlüsseln.
- Der Code Ihrer Anwendung wird im Ruhezustand verschlüsselt.
- Der langlebige Anwendungsspeicher wird im Ruhezustand verschlüsselt.
- Der laufende Anwendungsspeicher ist im Ruhezustand verschlüsselt.

Verschlüsselung während der Übertragung

Managed Service für Apache Flink verschlüsselt alle Daten während der Übertragung. Die Verschlüsselung während der Übertragung ist für alle Managed Service für Apache Flink-Anwendungen aktiviert und kann nicht deaktiviert werden.

Managed Service für Apache Flink verschlüsselt Daten während der Übertragung in den folgenden Szenarien:

- Daten werden von Kinesis Data Streams an Managed Service für Apache Flink übertragen.
- Daten werden zwischen internen Komponenten innerhalb von Managed Service für Apache Flink übertragen.
- Daten werden zwischen Managed Service für Apache Flink und Kinesis Data Firehose übertragen.

Schlüsselverwaltung

Die Datenverschlüsselung in Managed Service für Apache Flink verwendet vom Service verwaltete Schlüssel. Vom Kunden verwaltete Schlüssel werden nicht unterstützt.

Identity and Access Management für Amazon Managed Service für Apache Flink

AWS Identity and Access Management (IAM) ist ein AWS-Service, mit dem Administratoren den Zugriff auf AWS-Ressourcen sicher steuern können. IAM-Administratoren steuern, wer für die Verwendung von Managed Service für Apache Flink-Ressourcen authentifiziert (angemeldet) und autorisiert (Berechtigungen haben) sein kann. IAM ist ein AWS-Service, den Sie ohne zusätzliche Kosten verwenden können.

Themen

- Zielgruppe
- Authentifizierung mit Identitäten
- Verwalten des Zugriffs mit Richtlinien
- So funktioniert Amazon Managed Service für Apache Flink mit IAM
- Beispiele für identitätsbasierte Richtlinien für Amazon Managed Service für Apache Flink
- Problembehandlung bei Identität und Zugriff auf Amazon Managed Service für Apache Flink
- Serviceübergreifende Confused-Deputy-Prävention

Zielgruppe

Wie Sie AWS Identity and Access Management (IAM) verwenden, unterscheidet sich je nach Ihrer Arbeit in Managed Service für Apache Flink.

Service-Benutzer – Wenn Sie den Managed Service für Apache Flink-Service zur Ausführung von Aufgaben verwenden, stellt Ihnen Ihr Administrator die Anmeldeinformationen und Berechtigungen bereit, die Sie benötigen. Wenn Sie für Ihre Arbeit weitere Funktionen von Managed Service für Apache Flink verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Featuresweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anzufordern müssen. Wenn Sie nicht auf eine Funktion in Managed Service für Apache Flink zugreifen können, informieren Sie sich unter Problembehandlung bei Identität und Zugriff auf Amazon Managed Service für Apache Flink.

Service-Administrator – Wenn Sie in Ihrem Unternehmen für Managed Service für Apache Flink-Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollständigen Zugriff auf Managed Service für Apache Flink. Es ist Ihre Aufgabe, zu bestimmen, auf welche Funktionen und Ressourcen von Managed Service für Apache Flink Ihre Service-Benutzer zugreifen sollen. Sie müssen dann Anträge an Ihren IAM-Administrator stellen, um die Berechtigungen Ihrer Servicenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen dazu, wie Ihr Unternehmen IAM mit Managed Service für Apache Flink verwenden kann, finden Sie unter So funktioniert Amazon Managed Service für Apache Flink mit IAM.

IAM-Administrator – Wenn Sie als IAM-Administrator fungieren, sollten Sie Einzelheiten dazu kennen, wie Sie Richtlinien zur Verwaltung des Zugriffs auf Managed Service für Apache Flink verfassen können. Beispiele für identitätsbasierte Managed Service für Apache Flink-Richtlinien, die Sie in IAM verwenden können, finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon Managed Service für Apache Flink.

Authentifizierung mit Identitäten

Authentifizierung ist die Art, wie Sie sich mit Ihren Anmeldeinformationen bei AWS anmelden. Die Authentifizierung (Anmeldung bei AWS) muss als Root-Benutzer des AWS-Kontos, als IAM-Benutzer oder durch Übernahme einer IAM-Rolle erfolgen.

Sie können sich bei AWS als Verbundidentität mit Anmeldeinformationen anmelden, die über eine Identitätsquelle bereitgestellt werden. Benutzer von AWS IAM Identity Center (IAM Identity Center), die Single-Sign-on-Authentifizierung Ihres Unternehmens und Anmeldeinformationen für Google oder Facebook sind Beispiele für Verbundidentitäten. Wenn Sie sich als Verbundidentität anmelden, hat der Administrator vorher mithilfe von IAM-Rollen einen Identitätsverbund eingerichtet. Wenn Sie auf AWS mithilfe des Verbunds zugreifen, übernehmen Sie indirekt eine Rolle.

Je nachdem, welcher Benutzertyp Sie sind, können Sie sich bei der AWS Management Console oder beim AWS-Zugriffsportal anmelden. Weitere Informationen zum Anmelden bei AWS finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch von AWS-Anmeldung.

Bei programmgesteuertem Zugriff auf AWS bietet AWS ein Software Development Kit (SDK) und eine Command Line Interface (CLI, Befehlszeilenschnittstelle) zum kryptographischen Signieren Ihrer Anfragen mit Ihren Anmeldeinformationen. Wenn Sie keine AWS-Tools verwenden, müssen Sie Anforderungen selbst signieren. Weitere Informationen zur Verwendung der empfohlenen Methode zum eigenen Signieren von Anforderungen finden Sie unter Signieren von AWS-API-Anforderungen im IAM-Benutzerhandbuch.

Unabhängig von der verwendeten Authentifizierungsmethode müssen Sie möglicherweise zusätzliche Sicherheitsinformationen angeben. AWS empfiehlt beispielsweise die Verwendung von Multi-Faktor Authentifizierung (MFA), um die Sicherheit Ihres Kontos zu verbessern. Weitere Informationen finden Sie unter Multi-Faktor-Authentifizierung im AWS IAM Identity Center-Benutzerhandbuch und Verwenden der Multi-Faktor-Authentifizierung (MFA) in AWS im IAM-Benutzerhandbuch.

AWS-Konto-Root-Benutzer

Wenn Sie ein AWS-Konto neu erstellen, beginnen Sie mit einer Anmeldeidentität, die vollständigen Zugriff auf alle AWS-Services und Ressourcen des Kontos hat. Diese Identität wird als AWS-Konto-Root-Benutzer bezeichnet. Für den Zugriff auf den Root-Benutzer müssen Sie sich mit der E-Mail-Adresse und dem Passwort anmelden, die zur Erstellung des Kontos verwendet wurden. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Schützen Sie Ihre Root-Benutzer-Anmeldeinformationen und verwenden Sie diese, um die Aufgaben auszuführen, die nur der Root-Benutzer ausführen kann. Eine vollständige Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Aufgaben, die Root-Benutzer-Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Verbundidentität

Als bewährte Methode empfiehlt es sich, menschliche Benutzer, einschließlich Benutzer, die Administratorzugriff benötigen, aufzufordern, den Verbund mit einem Identitätsanbieter zu verwenden, um auf AWS-Services mit temporären Anmeldeinformationen zuzugreifen.

Eine Verbundidentität ist ein Benutzer aus dem Benutzerverzeichnis Ihres Unternehmens, ein Web Identity Provider, AWS Directory Service, das Identity-Center-Verzeichnis oder jeder Benutzer, der mit Anmeldeinformationen, die über eine Identitätsquelle bereitgestellt werden, auf AWS-Services

zugreift. Wenn Verbundidentitäten auf AWS-Konten zugreifen, übernehmen sie Rollen und die Rollen stellen temporäre Anmeldeinformationen bereit.

Für die zentrale Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center zu verwenden. Sie können Benutzer und Gruppen im IAM Identity Center erstellen oder Sie können eine Verbindung mit einer Gruppe von Benutzern und Gruppen in Ihrer eigenen Identitätsquelle herstellen und synchronisieren, um sie in allen AWS-Konten und Anwendungen zu verwenden. Informationen zu IAM Identity Center finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center-Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität in Ihrem AWS-Konto mit bestimmten Berechtigungen für eine einzelne Person oder eine einzelne Anwendung. Wenn möglich, empfehlen wir, temporäre Anmeldeinformationen zu verwenden, anstatt IAM-Benutzer zu erstellen, die langfristige Anmeldeinformationen wie Passwörter und Zugriffsschlüssel haben. Bei speziellen Anwendungsfällen, die langfristige Anmeldeinformationen mit IAM-Benutzern erfordern, empfehlen wir jedoch, die Zugriffsschlüssel zu rotieren. Weitere Informationen finden Sie unter Regelmäßiges Rotieren von Zugriffsschlüsseln für Anwendungsfälle, die langfristige Anmeldeinformationen erfordern im IAM-Benutzerhandbuch.

Eine IAM-Gruppe ist eine Identität, die eine Sammlung von IAM-Benutzern angibt. Sie können sich nicht als Gruppe anmelden. Mithilfe von Gruppen können Sie Berechtigungen für mehrere Benutzer gleichzeitig angeben. Gruppen vereinfachen die Verwaltung von Berechtigungen, wenn es zahlreiche Benutzer gibt. Sie könnten beispielsweise einer Gruppe mit dem Namen IAMAdmins Berechtigungen zum Verwalten von IAM-Ressourcen erteilen.

Benutzer unterscheiden sich von Rollen. Ein Benutzer ist einer einzigen Person oder Anwendung eindeutig zugeordnet. Eine Rolle kann von allen Personen angenommen werden, die sie benötigen. Benutzer besitzen dauerhafte Anmeldeinformationen. Rollen stellen temporäre Anmeldeinformationen bereit. Weitere Informationen finden Sie unter Erstellen eines IAM-Benutzers (anstatt einer Rolle) im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität in Ihrem AWS-Konto mit spezifischen Berechtigungen. Sie ist einem IAM-Benutzer vergleichbar, ist aber nicht mit einer bestimmten Person verknüpft. Sie können vorübergehend eine IAM-Rolle in der AWS Management Console übernehmen, indem Sie Rollen wechseln. Sie können eine Rolle annehmen, indem Sie eine AWS CLI oder AWS-API-Operation

aufrufen oder eine benutzerdefinierte URL verwenden. Weitere Informationen zu Methoden für die Verwendung von Rollen finden Sie unter Verwenden von IAM-Rollen im IAM-Benutzerhandbuch.

IAM-Rollen mit temporären Anmeldeinformationen sind in folgenden Situationen hilfreich:

- Verbundbenutzerzugriff: Um einer Verbundidentität Berechtigungen zuzuweisen, erstellen Sie eine Rolle und definieren Berechtigungen für die Rolle. Wird eine Verbundidentität authentifiziert, so wird die Identität der Rolle zugeordnet und erhält die von der Rolle definierten Berechtigungen. Informationen zu Rollen für den Verbund finden Sie unter Erstellen von Rollen für externe Identitätsanbieter im IAM-Benutzerhandbuch. Wenn Sie IAM Identity Center verwenden, konfigurieren Sie einen Berechtigungssatz. Wenn Sie steuern möchten, worauf Ihre Identitäten nach der Authentifizierung zugreifen können, korreliert IAM Identity Center den Berechtigungssatz mit einer Rolle in IAM. Informationen zu Berechtigungssätzen finden Sie unter Berechtigungssätze im AWS IAM Identity Center-Benutzerhandbuch.
- Temporäre IAM-Benutzerberechtigungen: Ein IAM-Benutzer oder eine -Rolle kann eine IAM-Rolle übernehmen, um vorübergehend andere Berechtigungen für eine bestimmte Aufgabe zu erhalten.
- Kontoübergreifender Zugriff – Sie können eine IAM-Rolle verwenden, um einem vertrauenswürdigen Prinzipal in einem anderen Konto den Zugriff auf Ressourcen in Ihrem Konto zu ermöglichen. Rollen stellen die primäre Möglichkeit dar, um kontoübergreifendem Zugriff zu gewähren. In einigen AWS-Services können Sie jedoch eine Richtlinie direkt an eine Ressource anfügen (anstatt eine Rolle als Proxy zu verwenden). Informationen zu den Unterschieden zwischen Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.
- Serviceübergreifender Zugriff: Einige AWS-Services verwenden Features in anderen AWS-Services. Wenn Sie beispielsweise einen Aufruf in einem Service tätigen, führt dieser Service häufig Anwendungen in Amazon EC2 aus oder speichert Objekte in Amazon S3. Ein Dienst kann dies mit den Berechtigungen des aufrufenden Prinzipals mit einer Servicerolle oder mit einer serviceverknüpften Rolle tun.
  - Forward access sessions (FAS) – Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle zum Ausführen von Aktionen in AWS verwenden, gelten Sie als Prinzipal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service auslösen. FAS verwendet die Berechtigungen des Prinzipals, der einen AWS-Service aufruft, in Kombination mit der Anforderung an den AWS-Service, Anforderungen an nachgelagerte Services zu stellen. FAS-Anfragen werden nur dann gestellt, wenn ein Service eine Anfrage erhält, die eine Interaktion mit anderen AWS-Services oder -Ressourcen erfordert. In diesem Fall müssen Sie

über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

- Servicerolle: Eine Servicerolle ist eine IAM-Rolle, die ein Service übernimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.
- Serviceverknüpfte Rolle: Eine serviceverknüpfte Rolle ist ein Typ von Servicerolle, die mit einem AWS-Service verknüpft ist. Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem AWS-Konto angezeigt und gehören zum Service. Ein IAM-Administrator kann die Berechtigungen für serviceverbundene Rollen anzeigen, aber nicht bearbeiten.

- Anwendungen in Amazon EC2: Sie können eine IAM-Rolle verwenden, um temporäre Anmeldeinformationen für Anwendungen zu verwalten, die auf einer EC2-Instance ausgeführt werden und AWS CLI- oder AWS-API-Anforderungen durchführen. Das ist eher zu empfehlen, als Zugriffsschlüssel innerhalb der EC2-Instance zu speichern. Erstellen Sie ein Instance-Profil, das an die Instance angefügt ist, um eine AWS-Rolle einer EC2-Instance zuzuweisen und die Rolle für sämtliche Anwendungen der Instance bereitzustellen. Ein Instance-Profil enthält die Rolle und ermöglicht, dass Programme, die in der EC2-Instance ausgeführt werden, temporäre Anmeldeinformationen erhalten. Weitere Informationen finden Sie unter Verwenden einer IAM-Rolle zum Erteilen von Berechtigungen für Anwendungen, die auf Amazon EC2-Instances ausgeführt werden im IAM-Benutzerhandbuch.

Informationen dazu, wann Sie IAM-Rollen oder IAM-Benutzer verwenden sollten, finden Sie unter Erstellen einer IAM-Rolle (anstatt eines Benutzers) im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Für die Zugriffssteuerung in AWS erstellen Sie Richtlinien und weisen diese den AWS-Identitäten oder -Ressourcen zu. Eine Richtlinie ist ein Objekt in AWS, das, wenn es einer Identität oder Ressource zugeordnet wird, deren Berechtigungen definiert. AWS wertet diese Richtlinien aus, wenn ein Prinzipal (Benutzer, Root-Benutzer oder Rollensitzung) eine Anforderung stellt. Berechtigungen in den Richtlinien bestimmen, ob die Anforderung zugelassen oder abgelehnt wird. Die meisten Richtlinien werden in AWS als JSON-Dokumente gespeichert. Weitere Informationen zu Struktur und Inhalten von JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das bedeutet, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

IAM-Richtlinien definieren Berechtigungen für eine Aktion unabhängig von der Methode, die Sie zur Ausführung der Aktion verwenden. Angenommen, es gibt eine Richtlinie, die Berechtigungen für die `iam:GetRole`-Aktion erteilt. Ein Benutzer mit dieser Richtlinie kann Benutzerinformationen über die AWS Management Console, die AWS CLI oder die AWS -API abrufen.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können weiter als Inline-Richtlinien oder verwaltete Richtlinien kategorisiert werden. Inline-Richtlinien sind direkt in einen einzelnen Benutzer, eine einzelne Gruppe oder eine einzelne Rolle eingebettet. Verwaltete Richtlinien sind eigenständige Richtlinien, die Sie mehreren Benutzern, Gruppen und Rollen in Ihrem AWS-Konto anfügen können. Verwaltete Richtlinien umfassen von AWS verwaltete und von Kunden verwaltete Richtlinien. Informationen dazu, wie Sie zwischen einer verwalteten Richtlinie und einer eingebundenen Richtlinie wählen, finden Sie unter Auswahl zwischen verwalteten und eingebundenen Richtlinien im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen

in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Prinzipale können Konten, Benutzer, Rollen, Verbundbenutzer oder AWS-Services umfassen.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können verwaltete AWS-Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffssteuerungslisten (ACLs)

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3, AWS WAF und Amazon VPC sind Beispiele für Dienste, die ACLs unterstützen. Weitere Informationen“ zu ACLs finden Sie unter Zugriffskontrollliste (ACL) – Übersicht (Access Control List) im Amazon-Simple-Storage-Service-Entwicklerhandbuch.

Weitere Richtlinientypen

AWS unterstützt zusätzliche, weniger häufig verwendete Richtlinientypen. Diese Richtlinientypen können die maximalen Berechtigungen festlegen, die Ihnen von den häufiger verwendeten Richtlinientypen erteilt werden können.

- Berechtigungsgrenzen:Eine Berechtigungsgrenze ist ein erweitertes Feature, mit der Sie die maximalen Berechtigungen festlegen können, die eine identitätsbasierte Richtlinie einer IAM-Entität (IAM-Benutzer oder -Rolle) erteilen kann. Sie können eine Berechtigungsgrenze für eine Entität festlegen. Die daraus resultierenden Berechtigungen sind der Schnittpunkt der identitätsbasierten Richtlinien einer Entität und ihrer Berechtigungsgrenzen. Ressourcenbasierte Richtlinien, die den Benutzer oder die Rolle im Feld `Principal` angeben, werden nicht durch Berechtigungsgrenzen eingeschränkt. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen über Berechtigungsgrenzen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im IAM-Benutzerhandbuch.
- Service-Kontrollrichtlinien (SCPs) – SCPs sind JSON-Richtlinien, die die maximalen Berechtigungen für eine Organisation oder Organisationseinheit (OE) in AWS Organizations angeben. AWS Organizations ist ein Dienst für die Gruppierung und zentrale Verwaltung mehrerer AWS-Konten Ihres Unternehmens. Wenn Sie innerhalb einer Organisation alle Features aktivieren, können Sie Service-Kontrollrichtlinien (SCPs) auf alle oder einzelne Ihrer Konten anwenden. SCPs schränken Berechtigungen für Entitäten in Mitgliedskonten einschließlich des jeweiligen Root-Benutzer des AWS-Kontos ein. Weitere Informationen zu Organizations und SCPs finden Sie unter Funktionsweise von SCPs im AWS Organizations-Benutzerhandbuch.

- Sitzungsrichtlinien:Sitzungsrichtlinien sind erweiterte Richtlinien, die Sie als Parameter übergeben, wenn Sie eine temporäre Sitzung für eine Rolle oder einen verbundenen Benutzer programmgesteuert erstellen. Die resultierenden Sitzungsberechtigungen sind eine Schnittmenge der auf der Identität des Benutzers oder der Rolle basierenden Richtlinien und der Sitzungsrichtlinien. Berechtigungen können auch aus einer ressourcenbasierten Richtlinie stammen. Eine explizite Zugriffsverweigerung in einer dieser Richtlinien setzt eine Zugriffserlaubnis außer Kraft. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn mehrere auf eine Anforderung mehrere Richtlinientypen angewendet werden können, sind die entsprechenden Berechtigungen komplizierter. Informationen dazu, wie AWS die Zulässigkeit einer Anforderung ermittelt, wenn mehrere Richtlinientypen beteiligt sind, finden Sie unter Logik für die Richtlinienauswertung im IAM-Benutzerhandbuch.

So funktioniert Amazon Managed Service für Apache Flink mit IAM

Bevor Sie IAM zum Verwalten des Zugriffs auf Managed Service für Apache Flink verwenden, erfahren Sie, welche IAM-Funktionen Sie mit Managed Service for Apache Flink verwenden können.

IAM-Funktionen, die Sie mit Amazon Managed Service für Apache Flink verwenden können

IAM-Feature	Support für Managed Service für Apache Flink
Identitätsbasierte Richtlinien	Ja
Ressourcenbasierte Richtlinien	Nein
Richtlinienaktionen	Ja
Richtlinienressourcen	Ja
Bedingungsschlüssel für die Richtlinie	Nein
ACLs	Nein
ABAC (Tags in Richtlinien)	Ja
Temporäre Anmeldeinformationen	Ja

IAM-Feature	Support für Managed Service für Apache Flink
Hauptberechtigungen	Ja
Servicerollen	Nein
Serviceverknüpfte Rollen	Nein

Einen Überblick über das Zusammenwirken von Managed Service für Apache Flink und anderen AWS-Services mit den meisten IAM-Features finden Sie unter AWS-Services, die mit IAM funktionieren im Handbuch für IAM-Benutzer.

Identitätsbasierte Richtlinien für Managed Service für Apache Flink

Unterstützt Richtlinien auf Identitätsbasis.	Ja

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität anfügen können, wie z. B. IAM-Benutzern, -Benutzergruppen oder -Rollen. Diese Richtlinien steuern, welche Aktionen die Benutzer und Rollen für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Mit identitätsbasierten IAM-Richtlinien können Sie angeben, welche Aktionen und Ressourcen zugelassen oder abgelehnt werden. Darüber hinaus können Sie die Bedingungen festlegen, unter denen Aktionen zugelassen oder abgelehnt werden. Sie können den Prinzipal nicht in einer identitätsbasierten Richtlinie angeben, da er für den Benutzer oder die Rolle gilt, dem er zugeordnet ist. Informationen zu sämtlichen Elementen, die Sie in einer JSON-Richtlinie verwenden, finden Sie in der IAM-Referenz für JSON-Richtlinienelemente im IAM-Benutzerhandbuch.

Beispiele für identitätsbasierte Richtlinien für Managed Service für Apache Flink

Beispiele für identitätsbasierte Managed Service für Apache Flink-Richtlinien, finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon Managed Service für Apache Flink.

Ressourcenbasierte Richtlinien in Managed Service für Apache Flink

Unterstützt ressourcenbasierte Richtlinien	Ja

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele für ressourcenbasierte Richtlinien sind IAM-Rollen-Vertrauensrichtlinien und Amazon-S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Für die Ressource, an welche die Richtlinie angehängt ist, legt die Richtlinie fest, welche Aktionen ein bestimmter Prinzipal unter welchen Bedingungen für diese Ressource ausführen kann. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben. Prinzipale können Konten, Benutzer, Rollen, Verbundbenutzer oder AWS-Services umfassen.

Um kontoübergreifenden Zugriff zu ermöglichen, können Sie ein gesamtes Konto oder IAM-Entitäten in einem anderen Konto als Prinzipal in einer ressourcenbasierten Richtlinie angeben. Durch das Hinzufügen eines kontoübergreifenden Auftraggebers zu einer ressourcenbasierten Richtlinie ist nur die halbe Vertrauensbeziehung eingerichtet. Wenn sich der Prinzipal und die Ressource in unterschiedlichen AWS-Konten befinden, muss ein IAM-Administrator im vertrauenswürdigen Konto auch der Prinzipalentität (Benutzer oder Rolle) die Berechtigung zum Zugriff auf die Ressource erteilen. Sie erteilen Berechtigungen, indem Sie der juristischen Stelle eine identitätsbasierte Richtlinie anfügen. Wenn jedoch eine ressourcenbasierte Richtlinie Zugriff auf einen Prinzipal in demselben Konto gewährt, ist keine zusätzliche identitätsbasierte Richtlinie erforderlich. Weitere Informationen finden Sie unter Wie sich IAM-Rollen von ressourcenbasierten Richtlinien unterscheiden im IAM-Benutzerhandbuch.

Richtlinienaktionen für Managed Service für Apache Flink

Unterstützt Richtlinienaktionen	Ja

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Action` einer JSON-Richtlinie beschreibt die Aktionen, mit denen Sie den Zugriff in einer Richtlinie zulassen oder verweigern können. Richtlinienaktionen haben normalerweise denselben Namen wie die zugehörige AWS-API-Operation. Es gibt einige Ausnahmen, z. B. Aktionen, die nur mit Genehmigung durchgeführt werden können und für die es keine passende API-Operation gibt. Es gibt auch einige Operationen, die mehrere Aktionen in einer Richtlinie erfordern. Diese zusätzlichen Aktionen werden als abhängige Aktionen bezeichnet.

Schließen Sie Aktionen in eine Richtlinie ein, um Berechtigungen zur Durchführung der zugeordneten Operation zu erteilen.

Eine Liste der Managed Service für Apache Flink-Aktionen finden Sie unter Von Amazon Managed Service für Apache Flink definierte Aktionen in der Referenz für die Service-Autorisierung.

Richtlinienaktionen in Managed Service für Apache Flink verwenden das folgende Präfix vor der Aktion:

```
Kinesis Analytics
```

Um mehrere Aktionen in einer einzigen Anweisung anzugeben, trennen Sie sie mit Kommata:

```
"Action": [
      "Kinesis Analytics:action1",
      "Kinesis Analytics:action2"
         ]
```

Sie können auch Platzhalter verwenden, um mehrere Aktionen anzugeben. Beispielsweise können Sie alle Aktionen festlegen, die mit dem Wort `Describe` beginnen, einschließlich der folgenden Aktion:

```
"Action": "Kinesis Analytics:Describe*"
```

Beispiele für identitätsbasierte Managed Service für Apache Flink-Richtlinien, finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon Managed Service für Apache Flink.

Richtlinienressourcen für Managed Service für Apache Flink

Unterstützt Richtlinienressourcen	Ja

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das bedeutet die Festlegung, welcher Prinzipal Aktionen für welche Ressourcen unter welchen Bedingungen ausführen kann.

Das JSON-Richtlinienelement `Resource` gibt die Objekte an, auf welche die Aktion angewendet wird. Anweisungen müssen entweder ein – `Resource`oder ein `NotResource`-Element enthalten.

Als bewährte Methode geben Sie eine Ressource mit dem zugehörigen Amazon-Ressourcennamen (ARN) an. Sie können dies für Aktionen tun, die einen bestimmten Ressourcentyp unterstützen, der als Berechtigungen auf Ressourcenebene bezeichnet wird.

Verwenden Sie für Aktionen, die keine Berechtigungen auf Ressourcenebene unterstützen, z. B. Auflistungsoperationen, einen Platzhalter (*), um anzugeben, dass die Anweisung für alle Ressourcen gilt.

```
"Resource": "*"
```

Eine Liste der Managed Service für Apache Flink-Ressourcentypen und ihrer ARNs finden Sie unter Von Amazon Managed Service für Apache Flink definierte Ressourcen in der Referenz für die Service-Autorisierung. Informationen zu den Aktionen, mit denen Sie den ARN einzelner Ressourcen angeben können, finden Sie unter Von Amazon Managed Service für Apache Flink definierte Aktionen.

Beispiele für identitätsbasierte Managed Service für Apache Flink-Richtlinien, finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon Managed Service für Apache Flink.

Richtlinienbedingungsschlüssel für Managed Service für Apache Flink

Unterstützt servicespezifische Richtlinienbedingungsschlüssel	Ja

Administratoren können mithilfe von AWS-JSON-Richtlinien festlegen, wer zum Zugriff auf was berechtigt ist. Das heißt, welcher Prinzipal kann Aktionen für welche Ressourcen und unter welchen Bedingungen ausführen.

Das Element `Condition` (oder `Condition` block) ermöglicht Ihnen die Angabe der Bedingungen, unter denen eine Anweisung wirksam ist. Das Element `Condition` ist optional. Sie können bedingte Ausdrücke erstellen, die Bedingungsoperatoren verwenden, z. B. ist gleich oder kleiner als, damit die Bedingung in der Richtlinie mit Werten in der Anforderung übereinstimmt.

Wenn Sie mehrere `Condition`-Elemente in einer Anweisung oder mehrere Schlüssel in einem einzelnen `Condition`-Element angeben, wertet AWS diese mittels einer logischen `AND`-Operation aus. Wenn Sie mehrere Werte für einen einzelnen Bedingungsschlüssel angeben, wertet AWS die

Bedingung mittels einer logischen OR-Operation aus. Alle Bedingungen müssen erfüllt werden, bevor die Berechtigungen der Anweisung gewährt werden.

Sie können auch Platzhaltervariablen verwenden, wenn Sie Bedingungen angeben. Beispielsweise können Sie einem IAM-Benutzer die Berechtigung für den Zugriff auf eine Ressource nur dann gewähren, wenn sie mit dessen IAM-Benutzernamen gekennzeichnet ist. Weitere Informationen finden Sie unter IAM-Richtlinienelemente: Variablen und Tags im IAM-Benutzerhandbuch.

AWS unterstützt globale Bedingungsschlüssel und servicespezifische Bedingungsschlüssel. Eine Liste aller globalen AWS-Bedingungsschlüssel finden Sie unter Globale AWS-Bedingungskontextschlüssel im IAM-Benutzerhandbuch.

Eine Liste der Managed Service für Apache Flink-Bedingungsschlüssel finden Sie unter Bedingungsschlüssel für Amazon Managed Service für Apache Flink in der Referenz für die Service-Autorisierung. Informationen dazu, mit welchen Aktionen und Ressourcen Sie einen Bedingungsschlüssel verwenden können, finden Sie unter Von Amazon Managed Service für Apache Flink definierte Aktionen.

Beispiele für identitätsbasierte Managed Service für Apache Flink-Richtlinien, finden Sie unter Beispiele für identitätsbasierte Richtlinien für Amazon Managed Service für Apache Flink.

Zugriffssteuerungslisten (ACLs) in Managed Service für Apache Flink

Unterstützt ACLs	Nein

Zugriffssteuerungslisten (ACLs) steuern, welche Prinzipale (Kontomitglieder, Benutzer oder Rollen) auf eine Ressource zugreifen können. ACLs sind ähnlich wie ressourcenbasierte Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Attributbasierte Zugriffskontrolle (ABAC) mit Managed Service für Apache Flink

Unterstützt ABAC (Tags in Richtlinien)	Ja

Die attributbasierte Zugriffskontrolle (ABAC) ist eine Autorisierungsstrategie, bei der Berechtigungen basierend auf Attributen definiert werden. In AWS werden diese Attribute als Tags bezeichnet. Sie können Tags an IAM-Entitäten (Benutzer oder Rollen) und mehrere AWS-Ressourcen anfügen. Das Markieren von Entitäten und Ressourcen ist der erste Schritt von ABAC. Anschließend entwerfen

Sie ABAC-Richtlinien, um Operationen zuzulassen, wenn das Tag des Prinzipals mit dem Tag der Ressource übereinstimmt, auf die sie zugreifen möchten.

ABAC ist in Umgebungen hilfreich, die schnell wachsen, und unterstützt Sie in Situationen, in denen die Richtlinienverwaltung mühsam wird.

Um den Zugriff auf der Grundlage von Tags zu steuern, geben Sie im Bedingungselement einer Richtlinie Tag-Informationen an, indem Sie die Schlüssel `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, oder Bedingung `aws:TagKeys` verwenden.

Wenn ein Service alle drei Bedingungsschlüssel für jeden Ressourcentyp unterstützt, lautet der Wert für den Service Ja. Wenn ein Service alle drei Bedingungsschlüssel für nur einige Ressourcentypen unterstützt, lautet der Wert Teilweise.

Weitere Informationen zu ABAC finden Sie unter Was ist ABAC? im IAM-Benutzerhandbuch. Um ein Tutorial mit Schritten zur Einstellung von ABAC anzuzeigen, siehe Attributbasierte Zugriffskontrolle (ABAC) verwenden im IAM-Benutzerhandbuch.

Temporäre Anmeldeinformationen mit Managed Service für Apache Flink verwenden

Unterstützt temporäre Anmeldeinformationen	Ja

Einige AWS-Services Featureieren nicht, wenn Sie sich mit temporären Anmeldeinformationen anmelden. Weitere Informationen, unter anderem darüber, welche AWS-Services mit temporären Anmeldeinformationen arbeiten, finden Sie unter AWS-Services, die mit IAM arbeiten im IAM-Benutzerhandbuch.

Sie verwenden temporäre Anmeldeinformationen, wenn Sie sich mit einer anderen Methode als einem Benutzernamen und einem Passwort bei der AWS Management Console anmelden. Wenn Sie beispielsweise über den Single Sign-On (SSO)-Link Ihres Unternehmens auf AWS zugreifen, erstellt dieser Prozess automatisch temporäre Anmeldeinformationen. Sie erstellen auch automatisch temporäre Anmeldeinformationen, wenn Sie sich als Benutzer bei der Konsole anmelden und dann die Rollen wechseln. Weitere Informationen zum Wechseln von Rollen finden Sie unter Wechseln zu einer Rolle (Konsole) im IAM-Benutzerhandbuch.

Sie können mithilfe der AWS CLI- oder AWS-API manuell temporäre Anmeldeinformationen erstellen. Sie können dann diese temporären Anmeldeinformationen verwenden, um auf AWS zuzugreifen. AWS empfiehlt, dass Sie temporäre Anmeldeinformationen dynamisch generieren, anstatt

langfristige Zugriffsschlüssel zu verwenden. Weitere Informationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen in IAM.

Serviceübergreifende Prinzipal-Berechtigungen für Managed Service für Apache Flink

Unterstützt Forward Access Sessions (FAS)	Ja

Wenn Sie einen IAM-Benutzer oder eine IAM-Rolle zum Ausführen von Aktionen in AWS verwenden, gelten Sie als Prinzipal. Bei einigen Services könnte es Aktionen geben, die dann eine andere Aktion in einem anderen Service auslösen. FAS verwendet die Berechtigungen des Prinzipals, der einen AWS-Service aufruft, in Kombination mit der Anforderung an den AWS-Service, Anforderungen an nachgelagerte Services zu stellen. FAS-Anfragen werden nur dann gestellt, wenn ein Service eine Anfrage erhält, die eine Interaktion mit anderen AWS-Services oder -Ressourcen erfordert. In diesem Fall müssen Sie über Berechtigungen zum Ausführen beider Aktionen verfügen. Einzelheiten zu den Richtlinien für FAS-Anfragen finden Sie unter Zugriffssitzungen weiterleiten.

Servicerollen für Managed Service für Apache Flink

Unterstützt Servicerollen	Ja

Eine Servicerolle ist eine IAM-Rolle, die ein Service annimmt, um Aktionen in Ihrem Namen auszuführen. Ein IAM-Administrator kann eine Servicerolle innerhalb von IAM erstellen, ändern und löschen. Weitere Informationen finden Sie unter Erstellen einer Rolle zum Delegieren von Berechtigungen an einen AWS-Service im IAM-Benutzerhandbuch.

> **Warning**
>
> Das Ändern der Berechtigungen für eine Servicerolle könnte die Funktionalität von Managed Service für Apache Flink beeinträchtigen. Bearbeiten Sie Servicerollen nur, wenn Managed Service für Apache Flink dazu Anleitungen gibt.

Serviceverknüpfte Rollen für Managed Service für Apache Flink

Unterstützt serviceverknüpfte Rollen	Ja

Eine serviceverknüpfte Rolle ist eine Art von Servicerolle, die mit einem AWS-Service verknüpft ist. Der Service kann die Rolle übernehmen, um eine Aktion in Ihrem Namen auszuführen. Serviceverknüpfte Rollen werden in Ihrem AWS-Konto angezeigt und gehören zum Service. Ein IAM-Administrator kann die Berechtigungen für Service-verknüpfte Rollen anzeigen, aber nicht bearbeiten.

Details zum Erstellen oder Verwalten von serviceverknüpften Rollen finden Sie unter AWS-Services, die mit IAM funktionieren. Suchen Sie in der Tabelle nach einem Service mit einem `Yes` in der Spalte Service-linked role (Serviceverknüpfte Rolle). Wählen Sie den Link Yes (Ja) aus, um die Dokumentation für die serviceverknüpfte Rolle für diesen Service anzuzeigen.

Beispiele für identitätsbasierte Richtlinien für Amazon Managed Service für Apache Flink

Benutzer und Rollen besitzen standardmäßig keine Berechtigungen zum Erstellen oder Ändern von Managed Service für Apache Flink-Ressourcen. Sie können auch keine Aufgaben über die AWS Management Console, die AWS Command Line Interface (AWS CLI) oder die AWS-API ausführen. Ein IAM-Administrator muss IAM-Richtlinien erstellen, die Benutzern die Berechtigung erteilen, Aktionen für die Ressourcen auszuführen, die sie benötigen. Der Administrator kann dann die IAM-Richtlinien zu Rollen hinzufügen, und Benutzer können die Rollen annehmen.

Informationen dazu, wie Sie unter Verwendung dieser beispielhaften JSON-Richtliniendokumente eine identitätsbasierte IAM-Richtlinie erstellen, finden Sie unter Erstellen von IAM-Richtlinien im IAM-Benutzerhandbuch.

Einzelheiten zu den von Managed Service für Apache Flink definierten Aktionen und Ressourcentypen, einschließlich des Formats der ARNs für die einzelnen Ressourcentypen, finden Sie unter Aktionen, Ressourcen und Bedingungsschlüssel für Amazon Managed Service für Apache Flink in der Referenz für die Service-Autorisierung.

Themen

- Bewährte Methoden für Richtlinien
- Nutzung der Managed Service für Apache Flink-Konsole
- Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

Bewährte Methoden für Richtlinien

Identitätsbasierte Richtlinien legen fest, ob jemand Managed Service für Apache Flink-Ressourcen in Ihrem Konto erstellen, darauf zugreifen oder sie löschen kann. Dies kann zusätzliche Kosten für Ihr

verursachen AWS-Konto. Befolgen Sie beim Erstellen oder Bearbeiten identitätsbasierter Richtlinien die folgenden Anleitungen und Empfehlungen:

- Erste Schritte mit AWS-verwaltete Richtlinien und Umstellung auf Berechtigungen mit den geringsten Berechtigungen:Um Ihren Benutzern und Workloads Berechtigungen zu gewähren, verwenden Sie die AWS-verwaltete Richtlinien die Berechtigungen für viele allgemeine Anwendungsfälle gewähren. Sie sind in Ihrem AWS-Konto verfügbar. Wir empfehlen Ihnen, die Berechtigungen weiter zu reduzieren, indem Sie vom Kunden verwaltete AWS-Richtlinien definieren, die speziell auf Ihre Anwendungsfälle zugeschnitten sind. Weitere Informationen finden Sie unter AWS-verwaltete Richtlinien oder AWS-verwaltete Richtlinien für Auftragsfunktionen im IAM-Benutzerhandbuch.
- Anwendung von Berechtigungen mit den geringsten Rechten:Wenn Sie mit IAM-Richtlinien Berechtigungen festlegen, gewähren Sie nur die Berechtigungen, die für die Durchführung einer Aufgabe erforderlich sind. Sie tun dies, indem Sie die Aktionen definieren, die für bestimmte Ressourcen unter bestimmten Bedingungen durchgeführt werden können, auch bekannt als die geringsten Berechtigungen. Weitere Informationen zur Verwendung von IAM zum Anwenden von Berechtigungen finden Sie unter Richtlinien und Berechtigungen in IAM im IAM-Benutzerhandbuch.
- Verwenden von Bedingungen in IAM-Richtlinien zur weiteren Einschränkung des Zugriffs:Sie können Ihren Richtlinien eine Bedingung hinzufügen, um den Zugriff auf Aktionen und Ressourcen zu beschränken. Sie können beispielsweise eine Richtlinienbedingung schreiben, um festzulegen, dass alle Anforderungen mithilfe von SSL gesendet werden müssen. Sie können auch Bedingungen verwenden, um Zugriff auf Service-Aktionen zu gewähren, wenn diese durch ein bestimmtes AWS-Service, wie beispielsweise AWS CloudFormation, verwendet werden. Weitere Informationen finden Sie unter IAM-JSON-Richtlinienelemente: Bedingung im IAM-Benutzerhandbuch.
- Verwenden von IAM Access Analyzer zur Validierung Ihrer IAM-Richtlinien, um sichere und funktionale Berechtigungen zu gewährleisten:IAM Access Analyzer validiert neue und vorhandene Richtlinien, damit die Richtlinien der IAM-Richtliniensprache (JSON) und den bewährten IAM-Methoden entsprechen. IAM Access Analyzer stellt mehr als 100 Richtlinienprüfungen und umsetzbare Empfehlungen zur Verfügung, damit Sie sichere und funktionale Richtlinien erstellen können. Weitere Informationen finden Sie unter Richtlinienvalidierung zum IAM Access Analyzer im IAM-Benutzerhandbuch.
- Bedarf einer Multi-Faktor-Authentifizierung (MFA):Wenn Sie ein Szenario haben, das IAM-Benutzer oder Root-Benutzer in Ihrem AWS-Konto erfordert, aktivieren Sie MFA für zusätzliche Sicherheit. Um MFA beim Aufrufen von API-Vorgängen anzufordern, fügen Sie Ihren Richtlinien MFA-

Bedingungen hinzu. Weitere Informationen finden Sie unter Konfigurieren eines MFA-geschützten API-Zugriffs im IAM-Benutzerhandbuch.

Weitere Informationen zu bewährten Methoden in IAM finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.

Nutzung der Managed Service für Apache Flink-Konsole

Um auf die Amazon Managed Service für Apache Flink-Konsole zugreifen zu können, müssen Sie über einen Mindestsatz von Berechtigungen verfügen. Diese Berechtigungen müssen Ihnen das Auflisten und Anzeigen von Details zu den Managed Service für Apache Flink-Ressourcen in Ihrem AWS-Konto gestatten. Wenn Sie eine identitätsbasierte Richtlinie erstellen, die strenger ist als die mindestens erforderlichen Berechtigungen, funktioniert die Konsole nicht wie vorgesehen für Entitäten (Benutzer oder Rollen) mit dieser Richtlinie.

Für Benutzer, die nur Aufrufe an die AWS CLI oder AWS-API durchführen, müssen Sie keine Mindestberechtigungen in der Konsole erteilen. Stattdessen sollten Sie nur Zugriff auf die Aktionen zulassen, die der API-Operation entsprechen, die die Benutzer ausführen möchten.

Um sicherzustellen, dass Benutzer und Rollen weiterhin die Managed Service für Apache Flink-Konsole verwenden können, fügen Sie den Entitäten auch die von AWS verwaltete `ConsoleAccess`- oder `ReadOnly`-Managed Service für Apache Flink-Richtlinie hinzu. Weitere Informationen finden Sie unter Hinzufügen von Berechtigungen zu einem Benutzer im IAM-Benutzerhandbuch.

Gewähren der Berechtigung zur Anzeige der eigenen Berechtigungen für Benutzer

In diesem Beispiel wird gezeigt, wie Sie eine Richtlinie erstellen, die IAM-Benutzern die Berechtigung zum Anzeigen der eingebundenen Richtlinien und verwalteten Richtlinien gewährt, die ihrer Benutzeridentität angefügt sind. Diese Richtlinie enthält Berechtigungen für die Ausführung dieser Aktion auf der Konsole oder für die programmgesteuerte Ausführung über die AWS CLI oder die AWS-API.

```
{
    "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": "*"
        }
    ]
}
```

Problembehandlung bei Identität und Zugriff auf Amazon Managed Service für Apache Flink

Verwenden Sie die folgenden Informationen, um häufige Probleme zu diagnostizieren und zu beheben, die beim Arbeiten mit Managed Service für Apache Flink und IAM auftreten könnten.

Themen

- Ich bin nicht berechtigt, eine Aktion in Managed Service für Apache Flink durchzuführen
- Ich bin nicht autorisiert, iam durchzuführen:PassRole
- Ich möchte Personen außerhalb meines AWS-Kontos Zugriff auf meine Managed Service für Apache Flink Ressourcen erteilen

Ich bin nicht berechtigt, eine Aktion in Managed Service für Apache Flink durchzuführen

Wenn die AWS Management Console Ihnen mitteilt, dass Sie nicht zur Ausführung einer Aktion autorisiert sind, müssen Sie sich an Ihren Administrator wenden, um Unterstützung zu erhalten. Ihr Administrator ist die Person, die Ihnen Ihren Benutzernamen und Ihr Passwort bereitgestellt hat.

Der folgende Beispielfehler tritt auf, wenn der `mateojackson`-Benutzer versucht, die Konsole zum Anzeigen von Details zu einer fiktiven *`my-example-widget`*-Ressource zu verwenden, jedoch nicht über `Kinesis Analytics:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: Kinesis
 Analytics:GetWidget on resource: my-example-widget
```

In diesem Fall bittet Mateo seinen Administrator um die Aktualisierung seiner Richtlinien, um unter Verwendung der Aktion *`my-example-widget`* auf die Ressource `Kinesis Analytics:`*`GetWidget`* zugreifen zu können.

Ich bin nicht autorisiert, iam durchzuführen:PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zum Durchführen der `iam:PassRole`-Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an Managed Service für Apache Flink übergeben zu können.

Einige AWS-Services erlauben die Übergabe einer vorhandenen Rolle an diesen Dienst, sodass keine neue Servicerolle oder serviceverknüpfte Rolle erstellt werden muss. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in Managed Service für Apache Flink auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenden Sie sich an Ihren AWS-Administrator, falls Sie weitere Unterstützung benötigen. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb meines AWS-Kontos Zugriff auf meine Managed Service für Apache Flink Ressourcen erteilen

Sie können eine Rolle erstellen, die Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation für den Zugriff auf Ihre Ressourcen verwenden können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Im Fall von Diensten, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (Access Control Lists, ACLs) verwenden, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen dazu, ob Managed Service für Apache Flink diese Features unterstützt, finden Sie unter So funktioniert Amazon Managed Service für Apache Flink mit IAM.
- Informationen zum Gewähren des Zugriffs auf Ihre Ressourcen für alle Ihre AWS-Konten finden Sie unter Gewähren des Zugriffs für einen IAM-Benutzer in einem anderen Ihrer AWS-Konto im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie AWS-Konten-Drittanbieter Zugriff auf Ihre Ressourcen bereitstellen, finden Sie unter Gewähren des Zugriffs auf AWS-Konten von externen Benutzern im IAM-Benutzerhandbuch.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter So unterscheiden sich IAM-Rollen von ressourcenbasierten Richtlinien im IAM-Benutzerhandbuch.

Serviceübergreifende Confused-Deputy-Prävention

In AWS kann ein servicebergreifender Identitätswechsel auftreten, wenn ein Service (der aufrufende Service) einen anderen Service aufruft (den aufgerufenen Service). Der aufrufende Dienst kann so manipuliert werden, dass er auf die Ressourcen eines anderen Kunden zugreift, obwohl er nicht die entsprechenden Berechtigungen haben sollte, was zu dem Problem des confused Deputy führt.

Um confused Deputys zu verhindern, bietet AWS Tools, mit denen Sie Ihre Daten für alle Services mit Serviceprinzipalen schützen können, die Zugriff auf Ressourcen in Ihrem Konto erhalten haben. In diesem Abschnitt wird speziell für Managed Service für Apache Flink die Vermeidung von serviceübergreifenden Problemen mit confused Deputys behandelt. Weitere Informationen zu diesem Thema finden Sie jedoch im IAM-Benutzerhandbuch im Abschnitt Das Problem mit confused Deputys.

Im Kontext von Managed Service für Apache Flink empfehlen wir die Verwendung der globalen Bedingungskontextschlüssel aws:SourceArn und aws:SourceAccount in Ihrer Rollenvertrauensrichtlinie, um den Zugriff auf die Rolle auf die Anforderungen zu beschränken, die von erwarteten Ressourcen generiert werden.

Verwenden Sie `aws:SourceArn`, wenn Sie nur eine Ressource mit dem betriebsübergreifenden Zugriff verknüpfen möchten. Verwenden Sie `aws:SourceAccount`, wenn Sie zulassen möchten, dass Ressourcen in diesem Konto mit der betriebsübergreifenden Verwendung verknüpft werden.

Der Wert von `aws:SourceArn` muss die ARN der von Managed Service für Apache Flink verwendeten Ressource sein, die im folgenden Format angegeben wird: `arn:aws:kinesisanalytics:region:account:resource`.

Der effektivste Weg, um sich vor dem Confused-Deputy-Problem zu schützen, ist die Verwendung des globalen `aws:SourceArn`-Bedingungskontextschlüssels mit dem vollständigen ARN der Ressource.

Wenn Sie die vollständige ARN der Ressource nicht kennen oder wenn Sie mehrere Ressourcen angeben, verwenden Sie den `aws:SourceArn`-Schlüssel mit Platzhalterzeichen (*) für die unbekannten Teile der ARN. Zum Beispiel: `arn:aws:kinesisanalytics::111122223333:*`.

Richtlinien für Rollen, die Sie Managed Service für Apache Flink zur Verfügung stellen, sowie Vertrauensrichtlinien für Rollen, die für Sie generiert wurden, können diese Schlüssel verwenden.

Um sich vor dem Confused-Deputy-Problem zu schützen, führen Sie die folgenden Schritte durch:

Schutz vor dem Confused-Deputy-Problem

1. Melden Sie sich bei der AWS-Managementkonsole an und öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Rollen und dann die Tabelle aus, die Sie ändern möchten.
3. Wählen Sie Vertrauensrichtlinie bearbeiten aus.

4. Ersetzen Sie auf der Seite Vertrauensrichtlinie bearbeiten die Standard-JSON-Richtlinie durch eine Richtlinie, die einen oder beide der globalen `aws:SourceArn`- und `aws:SourceAccount`-Bedingungskontextschlüssel verwendet. Sehen Sie sich die folgende Beispielrichtlinie an:
5. Wählen Sie Richtlinie aktualisieren.

```
{
   "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"
            }
         }
      }
   ]
}
```

Überwachung von Managed Service für Apache Flink

Managed Service für Apache Flink bietet Überwachungsfunktionen für Ihre Anwendungen. Weitere Informationen finden Sie unter Protokollieren und Überwachen.

Compliance-Validierung für Amazon Managed Service für Apache Flink

Externe Prüfer bewerten im Rahmen verschiedener AWS-Compliance-Programme die Sicherheit und Compliance von Amazon Managed Service für Apache Flink. Zu diesen Programmen gehören SOC, PCI, HIPAA und andere.

Eine Liste der AWS-Services, die in den Geltungsbereich bestimmter Compliance-Programme fallen, finden Sie unter . Allgemeine Informationen finden Sie unter AWS-Compliance-Programme.

Die Auditberichte von Drittanbietern lassen sich mit herunterlade AWS Artifact. Weitere Informationen finden Sie unter Herunterladen von Berichten in AWS Artifact.

Ihre Compliance-Verantwortung bei Verwendung von Managed Service für Apache Flink hängt von der Vertraulichkeit der Daten, den Compliance-Zielen des Unternehmens und den geltenden Gesetzen und Vorschriften ab. Wenn Ihre Nutzung von Managed Service für Apache Flink von der Einhaltung von Standards wie HIPAA oder PCI abhängig ist, stellt AWS Ressourcen zur Unterstützung bereit:

- Kurzanleitungen für Sicherheit und Compliance – In diesen Bereitstellungsleitfäden finden Sie wichtige Überlegungen zur Architektur sowie die einzelnen Schritte zur Bereitstellung von sicherheits- und Compliance-orientierten Basisumgebungen in AWS.
- Erstellen von Architekturen für HIPAA-Sicherheit und -Compliance in Amazon Web Services. Dieses Whitepaper beschreibt, wie Unternehmen mithilfe von AWS HIPAA-konforme Anwendungen erstellen.
- AWS-Compliance-Ressourcen – Diese Arbeitsbücher und Leitfäden könnten für Ihre Branche und Ihren Standort interessant sein.
- AWS Config – Dieser AWS-Service bewertet, zu welchem Grad die Konfiguration Ihrer Ressourcen den internen Vorgehensweisen, Branchenrichtlinien und Vorschriften entspricht.
- AWS Security Hub – Dieser AWS-Service liefert einen umfassenden Überblick über den Sicherheitsstatus in AWS. So können Sie die Compliance mit den Sicherheitsstandards in der Branche und den bewährten Methoden abgleichen.

FedRAMP

Das AWS-FedRAMP-Compliance-Programm umfasst Managed Service für Apache Flink als FedRAMP-konformen Service. Wenn Sie Bundes- oder Geschäftskunde sind, können Sie den Service verwenden, um sensible Workloads in der Autorisierungsgrenze der Region AWS GovCloud (USA) mit Daten bis zur hohen Auswirkungsstufe sowie in den Regionen USA Ost (Nord-Virginia), USA Ost (Ohio), USA West (Nordkalifornien) und USA West (Oregon) mit Daten bis zu einer mittleren Stufe zu verarbeiten und zu speichern.

Sie können den Zugang zu den AWS-FedRAMP-Sicherheitspaketen über das FedRAMP PMO oder Ihren AWS-Sales-Account-Manager beantragen oder sie können über AWS-Artifact unter AWS-Artifact heruntergeladen werden.

Weitere Informationen finden Sie unter FedRAMP.

Ausfallsicherheit in Amazon Managed Service für Apache Flink

Im Zentrum der globalen AWS Infrastruktur stehen die AWS-Regionen und Availability Zones (Verfügbarkeitszonen, AZs). AWS Regionen stellen mehrere physisch getrennte und isolierte Availability Zones bereit, die mit Netzwerken mit geringer Latenz, hohem Durchsatz und hochredundanten Vernetzungen verbunden sind. Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Availability Zones ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser hoch verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen über AWS-Regionen und -Availability Zones finden Sie unter AWS Globale Infrastruktur.

Zusätzlich zur globalen AWS-Infrastruktur stellt Managed Service für Apache Flink verschiedene Features bereit, um Ihren Anforderungen in Bezug auf Daten-Ausfallsicherheit und Datensicherung zu erfüllen.

Notfallwiederherstellung

Managed Service für Apache Flink wird in einem Serverless-Modus ausgeführt und bietet dank der automatischen Migration Unterstützung bei Host-Leistungsverschlechterungen, Availability-Zone-Verfügbarkeit und anderen infrastrukturbezogenen Problemen. Managed Service für Apache Flink erreicht dies mithilfe mehrerer redundanter Mechanismen. Jede Managed Service für Apache Flink-Anwendung wird in einem Apache Flink-Cluster mit einem Mandanten ausgeführt. Der Apache-Flink-Cluster wird mit im JobMananger Hochverfügbarkeitsmodus unter Verwendung von Zookeeper über mehrere Availability Zones hinweg ausgeführt. Managed Service für Apache Flink stellt Apache Flink mithilfe von Amazon EKS bereit. In Amazon EKS werden für jede AWS-Region in allen Verfügbarkeitszonen mehrere Kubernetes-Pods verwendet. Im Falle eines Fehlers versucht Managed Service für Apache Flink zunächst, die Anwendung innerhalb des laufenden Apache Flink-Clusters mithilfe der Prüfpunkte Ihrer Anwendung, sofern verfügbar, wiederherzustellen.

Managed Service für Apache Flink sichert den Anwendungsstatus mithilfe von Prüfpunkten und Snapshots:

- Prüfpunkte sind Backups des Anwendungsstatus, die Managed Service für Apache Flink automatisch in regelmäßigen Abständen erstellt und zur Wiederherstellung nach Fehlern verwendet.
- Snapshots sind Backups des Anwendungsstatus, die Sie manuell erstellen und anhand derer Sie manuell wiederherstellen.

Weitere Informationen zu Prüfpunkten und Snapshots finden Sie unter Fehlertoleranz.

Versionsverwaltung

Gespeicherte Versionen des Anwendungsstatus werden wie folgt versioniert:

- Prüfpunkte werden vom Service automatisch versioniert. Wenn der Service einen Prüfpunkt verwendet, um die Anwendung neu zu starten, wird der neueste Prüfpunkt verwendet.
- Savepoints werden mit dem -SnapshotNameParameter der -CreateApplicationSnapshotAktion versioniert.

Managed Service für Apache Flink verschlüsselt Daten, die in Prüfpunkten und Savepoints gespeichert sind.

Infrastruktursicherheit in Managed Service für Apache Flink

Als verwalteter Service ist Amazon Managed Service für Apache Flink durch die globalen AWS-Netzwerksicherheit-Verfahren geschützt, die im Whitepaper Amazon Web Services: Überblick der Sicherheitsverfahren beschrieben sind.

Sie verwenden durch AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf Managed Service für Apache Flink zuzugreifen. Alle API-Aufrufe an Managed Service für Apache Flink werden über Transport Layer Security (TLS) gesichert und über IAM authentifiziert. Kunden müssen TLS 1.2 oder höher unterstützen. Clients müssen außerdem Verschlüsselungssammlungen mit PFS (Perfect Forward Secrecy) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman) unterstützen. Die meisten modernen Systemen wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS Security Token Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Bewährte Methoden für die Sicherheit für Managed Service für Apache Flink

Amazon Managed Service für Apache Flink enthält eine Reihe von Sicherheitsfeatures, die Sie bei der Entwicklung und Implementierung Ihrer eigenen Sicherheitsrichtlinien berücksichtigen sollten. Die folgenden bewährten Methoden sind allgemeine Richtlinien und keine vollständige Sicherheitslösung. Da diese bewährten Methoden für Ihre Umgebung möglicherweise nicht angemessen oder ausreichend sind, sollten Sie sie als hilfreiche Überlegungen und nicht als bindend ansehen.

Implementieren des Zugriffs mit geringsten Berechtigungen

Beim Erteilen von Berechtigungen entscheiden Sie, wer welche Berechtigungen für welche Managed Service für Apache Flink-Ressourcen erhält. Sie aktivieren die spezifischen Aktionen, die daraufhin für die betreffenden Ressourcen erlaubt sein sollen. Aus diesem Grund sollten Sie nur Berechtigungen gewähren, die zum Ausführen einer Aufgabe erforderlich sind. Die Implementierung der geringstmöglichen Zugriffsrechte ist eine grundlegende Voraussetzung zum Reduzieren des Sicherheitsrisikos und der Auswirkungen, die aufgrund von Fehlern oder böswilligen Absichten entstehen könnten.

Verwenden von IAM-Rollen zum Zugriff auf andere Amazon-Services

Ihre Managed Service für Apache Flink-Anwendung muss über gültige Anmeldeinformationen verfügen, um auf Ressourcen in anderen Services wie Kinesis Data Streams, Kinesis Data Firehose-Streams oder Amazon S3-Buckets zugreifen zu können. Sie sollten AWS-Anmeldeinformationen nicht direkt in der Anwendung oder in einem Amazon S3-Bucket speichern. Dabei handelt es sich um langfristige Anmeldeinformationen, die nicht automatisch rotiert werden und bedeutende geschäftliche Auswirkungen haben könnten, wenn sie kompromittiert werden.

Stattdessen sollten Sie mithilfe einer IAM-Rolle temporäre Anmeldeinformationen für Ihre Anwendung für den Zugriff auf andere Ressourcen verwalten. Wenn Sie eine Rolle verwenden, müssen Sie keine langfristigen Anmeldeinformationen für den Zugriff auf andere Ressourcen verwenden.

Weitere Informationen finden Sie unter folgenden Themen im IAM-Benutzerhandbuch:

- IAM-Rollen
- Gängige Szenarien für Rollen: Benutzer, Anwendungen und Services

Implementieren einer serverseitigen Verschlüsselung in abhängigen Ressourcen

Daten im Ruhezustand und Daten während der Übertragung werden in Managed Service für Apache Flink verschlüsselt, und diese Verschlüsselung kann nicht deaktiviert werden. Sie sollten die serverseitige Verschlüsselung bei abhängigen Ressourcen, wie z. B. Kinesis Data Streams, Kinesis Data Firehose-Streams und Amazon S3-Buckets implementieren. Weitere Informationen zum Implementieren der serverseitigen Verschlüsselung bei abhängigen Ressourcen finden Sie unter Datenschutz.

Verwenden von CloudTrail zur Überwachung von API-Aufrufen

Managed Service für Apache Flink ist in AWS CloudTrail integriert, einen Service, der die Aktionen eines Benutzers, einer Rolle oder eines Amazon-Services in Managed Service für Apache Flink aufzeichnet.

Anhand der von CloudTrailgesammelten Informationen können Sie die an Managed Service für Apache Flink gestellte Anfrage, die IP-Adresse, von der die Anfrage gestellt wurde, den Initiator der Anfrage, den Zeitpunkt der Anfrage und zusätzliche Details bestimmen.

Weitere Informationen finden Sie unter the section called “Verwenden von AWS CloudTrail”.

Protokollierung und Überwachung in Amazon Managed Service für Apache Flink

Die Überwachung ist wichtig, um Zuverlässigkeit, Verfügbarkeit und Leistung Ihrer Managed Service für Apache Flink-Anwendungen aufrechtzuerhalten. Sie sollten von allen Teilen der AWS-Lösung Überwachungsdaten sammeln, damit Sie Ausfälle, die sich über mehrere Punkte erstrecken, leichter debuggen können.

Bevor Sie mit der Überwachung von Managed Service für Apache Flink beginnen, sollten Sie einen Überwachungsplan mit Antworten auf die folgenden Fragen erstellen:

- Was sind Ihre Ziele bei der Überwachung?
- Welche Ressourcen werden überwacht?
- Wie oft werden diese Ressourcen überwacht?
- Welche Überwachungstools werden verwendet?
- Wer soll die Überwachungsaufgaben ausführen?
- Wer soll benachrichtigt werden, wenn Fehler auftreten?

Der nächste Schritt besteht darin, eine Baseline für normale Managed Service für Apache Flink-Performance in Ihrer Umgebung aufzustellen. Dies geschieht, indem Sie die Leistung zu verschiedenen Zeiten und unter verschiedenen Lastbedingungen messen. Sie können bei der Überwachung von Managed Service für Apache Flink historische Überwachungsdaten speichern. Sie können diese dann mit aktuellen Leistungsdaten vergleichen, normale Leistungsmuster und Leistungsanomalien identifizieren sowie Verfahren für den Umgang mit Problemen entwickeln.

Themen

- Protokollierung
- Überwachen
- Anwendungsprotokollierung einrichten
- Analysieren von Protokollen mit CloudWatch Logs Insights
- Anzeigen von Metriken und Dimensionen in Managed Service für Apache Flink
- Schreiben von benutzerdefinierten Nachrichten in CloudWatch Protokolle
- Protokollieren von Managed Service für Apache Flink API-Aufrufen mit AWS CloudTrail

Protokollierung

Die Protokollierung ist wichtig für Produktionsanwendungen, um Fehler und Ausfälle zu verstehen. Das Protokollierungssubsystem muss jedoch Protokolleinträge erfassen und an - CloudWatch Protokolle weiterleiten. Einige Protokollierungen sind zwar in Ordnung und erwünscht, eine umfangreiche Protokollierung kann jedoch den Service überlasten und dazu führen, dass die Flink-Anwendung zurückfällt. Das Protokollieren von Ausnahmen und Warnungen ist sicherlich eine gute Idee. Sie können jedoch nicht für jede einzelne Nachricht, die von der Flink-Anwendung verarbeitet wird, eine Protokollnachricht generieren. Flink ist für hohen Durchsatz und geringe Latenz optimiert, das Protokollierungs-Subsystem nicht. Falls es wirklich erforderlich ist, die Protokollausgabe für jede verarbeitete Nachricht zu generieren, verwenden Sie eine zusätzliche DataStream innerhalb der Flink-Anwendung und eine richtige Senke, um die Daten an Amazon S3 oder zu senden CloudWatch. Verwenden Sie das Java-Protokollierungs-System nicht für diesen Zweck. Darüber hinaus generiert die `Debug Monitoring Log Level`-Einstellung von Managed Service für Apache Flink eine große Menge an Datenverkehr, was zu Gegendruck führen kann. Sie sollten sie nur verwenden, wenn Sie aktiv Probleme mit der Anwendung untersuchen.

Abfragen von Protokollen mit CloudWatch Logs Insights

CloudWatch Logs Insights ist ein leistungsstarker Service zur skalierbaren Abfrage von Protokollen. Kunden sollten seine Funktionen nutzen, um Protokolle schnell zu durchsuchen, um Fehler bei Betriebsereignissen zu identifizieren und zu beheben.

Die folgende Abfrage sucht in allen Task-Manager-Protokollen nach Ausnahmen und ordnet sie nach dem Zeitpunkt, zu dem sie aufgetreten sind.

```
fields @timestamp, @message
| filter isPresent(throwableInformation.0) or isPresent(throwableInformation) or
 @message like /(Error|Exception)/
| sort @timestamp desc
```

Weitere nützliche Abfragen finden Sie unter Beispielabfragen.

Überwachen

Wenn Sie Streaming-Anwendungen in der Produktion ausführen, möchten Sie die Anwendung kontinuierlich und unbegrenzt ausführen. Es ist wichtig, die Überwachung und korrekte Alarmierung

aller Komponenten zu implementieren, nicht nur der Flink-Anwendung. Andernfalls riskieren Sie, aufkommende Probleme frühzeitig zu übersehen und ein operatives Ereignis erst dann zu erkennen, wenn es vollständig aufgelöst und viel schwieriger zu beheben ist. Zu den allgemeinen Dingen, die es zu überwachen gilt, gehören:

- Nimmt die Quelle Daten auf?
- Werden Daten aus der Quelle gelesen (aus der Perspektive der Quelle)?
- Empfängt die Flink-Anwendung Daten?
- Kann die Flink-Anwendung Schritt halten oder gerät sie ins Hintertreffen?
- Speichert die Flink-Anwendung Daten dauerhaft in der Senke (aus Sicht der Anwendung)?
- Empfängt die Senke Daten?

Spezifischere Metriken sollten dann für die Flink-Anwendung in Betracht gezogen werden. Dieses CloudWatch Dashboard bietet einen guten Ausgangspunkt. Weitere Informationen darüber, welche Metriken für Produktionsanwendungen überwacht werden sollten, finden Sie unter Verwenden von CloudWatch Alarmen mit Amazon Managed Service für Apache Flink. Zu diesen Metriken gehören:

- records_lag_max und millisbehindLatest – Wenn die Anwendung von Kinesis oder Kafka konsumiert, geben diese Metriken an, ob die Anwendung hinterherhinkt und geringer skaliert werden muss, um mit der aktuellen Auslastung Schritt zu halten. Dies ist eine gute generische Metrik, die für alle Arten von Anwendungen leicht nachzuverfolgen ist. Sie kann jedoch nur für reaktive Skalierung verwendet werden, d. h. wenn die Anwendung bereits ins Hintertreffen geraten ist.
- cpuUtilization und heapMemoryUtilization – Diese Metriken geben einen guten Hinweis auf die Gesamtressourcenauslastung der Anwendung und können für die proaktive Skalierung verwendet werden, es sei denn, die Anwendung ist E/A-gebunden.
- downtime – Eine Ausfallzeit von mehr als Null bedeutet, dass die Anwendung ausgefallen ist. Wenn der Wert größer als 0 ist, verarbeitet die Anwendung keine Daten.
- lastCheckpointSize und lastCheckpointDuration – Diese Metriken überwachen, wie viele Daten im -Status gespeichert sind und wie lange es dauert, bis ein Checkpoint ausgeführt wird. Wenn die Anzahl der Prüfpunkte zunimmt oder lange dauert, verbringt die Anwendung kontinuierlich Zeit mit Prüfpunkten und hat weniger Zyklen für die eigentliche Verarbeitung. An manchen Stellen können Prüfpunkte zu groß werden oder so lange dauern, dass sie ausfallen. Neben der Überwachung absoluter Werte sollten Kunden auch erwägen, die Änderungsrate mit `RATE(lastCheckpointSize)` und `RATE(lastCheckpointDuration)` zu überwachen.

- numberOfFailedCheckpoints – Diese Metrik zählt die Anzahl der fehlgeschlagenen Checkpoints seit dem Start der Anwendung. Je nach Anwendung kann es toleriert werden, dass Prüfpunkte gelegentlich fehlschlagen. Wenn Prüfpunkte jedoch regelmäßig ausfallen, ist die Anwendung wahrscheinlich fehlerhaft und benötigt weitere Aufmerksamkeit. Wir empfehlen die Überwachung von `RATE(numberOfFailedCheckpoints)`, dass der Alarm anhand des Gefälles und nicht anhand absoluter Werte ausgelöst wird.

Anwendungsprotokollierung einrichten

Indem Sie Ihrer Anwendung Managed Service für Apache Flink eine Amazon- CloudWatch Protokollierungsoption hinzufügen, können Sie auf Anwendungsereignisse oder Konfigurationsprobleme überwachen.

In diesem Thema wird beschrieben, wie Sie Ihre Anwendung so konfigurieren, dass Anwendungsereignisse in einen CloudWatch Logs-Stream geschrieben werden. Eine CloudWatch Protokollierungsoption ist eine Sammlung von Anwendungseinstellungen und Berechtigungen, die Ihre Anwendung verwendet, um die Art und Weise zu konfigurieren, wie sie Anwendungsereignisse in CloudWatch Protokolle schreibt. Sie können eine CloudWatch Protokollierungsoption entweder über die AWS Management Console oder die AWS Command Line Interface () hinzufügen und konfigurierenAWS CLI.

Beachten Sie Folgendes zum Hinzufügen einer CloudWatch Protokollierungsoption zu Ihrer Anwendung:

- Wenn Sie eine CloudWatch Protokollierungsoption über die Konsole hinzufügen, erstellt Managed Service für Apache Flink die CloudWatch Protokollgruppe und den Protokollstream für Sie und fügt die Berechtigungen hinzu, die Ihre Anwendung zum Schreiben in den Protokollstream benötigt.
- Wenn Sie eine CloudWatch Protokollierungsoption mithilfe der API hinzufügen, müssen Sie auch die Protokollgruppe und den Protokollstream der Anwendung erstellen und die Berechtigungen hinzufügen, die Ihre Anwendung zum Schreiben in den Protokollstream benötigt.

Dieses Thema enthält die folgenden Abschnitte:

- Einrichten der CloudWatch Protokollierung mit der Konsole
- Einrichten der CloudWatch Protokollierung mit der CLI
- Anwendungsüberwachungsebenen
- Bewährte Methoden der Protokollierung

- Protokollierung von Problemlösungen
- Nächster Schritt

Einrichten der CloudWatch Protokollierung mit der Konsole

Wenn Sie die CloudWatch Protokollierung für Ihre Anwendung in der Konsole aktivieren, werden eine CloudWatch Protokollgruppe und ein Protokollstream für Sie erstellt. Außerdem wird die Berechtigungsrichtlinie Ihrer Anwendung mit den Berechtigungen zum Schreiben in den Stream aktualisiert.

Managed Service für Apache Flink erstellt eine Protokollgruppe mit dem Namen unter Verwendung der folgenden Konvention, wobei der Name Ihrer Anwendung *ApplicationName* ist.

```
/AWS/KinesisAnalytics/ApplicationName
```

Managed Service für Apache Flink erstellt einen Protokollstream in der neuen Protokollgruppe mit dem folgenden Namen.

```
kinesis-analytics-log-stream
```

Sie legen die Ebene der Anwendungsüberwachungsmetriken und die Protokollebene mit dem Abschnitt Überwachung der Protokollebene der Seite Anwendung konfigurieren fest. Hinweise zu den Protokollebenen von Anwendungen finden Sie unter the section called “Anwendungsüberwachungsebenen”.

Einrichten der CloudWatch Protokollierung mit der CLI

Gehen Sie wie folgt vorAWS CLI, um eine CloudWatch Protokollierungsoption mithilfe der hinzuzufügen:

- Erstellen Sie eine CloudWatch Protokollgruppe und einen Protokollstream.
- Fügen Sie eine Protokollierungsoption hinzu, wenn Sie eine Anwendung mithilfe der -CreateApplicationAktion erstellen, oder fügen Sie einer vorhandenen Anwendung mithilfe der -AddApplicationCloudWatchLoggingOptionAktion eine Protokollierungsoption hinzu.
- Fügen Sie der Richtlinie Ihrer Anwendung Berechtigungen zum Schreiben in die Protokolle hinzu.

In diesem Abschnitt werden folgende Themen behandelt:

- Erstellen einer CloudWatch Protokollgruppe und eines Protokollstreams
- Arbeiten mit CloudWatch Anwendungsprotokollierungsoptionen
- Hinzufügen von Berechtigungen zum Schreiben in den CloudWatch Protokollstream

Erstellen einer CloudWatch Protokollgruppe und eines Protokollstreams

Sie erstellen eine CloudWatch Protokollgruppe und streamen entweder über die CloudWatch Logs-Konsole oder die API. Informationen zum Erstellen einer CloudWatch Protokollgruppe und eines Protokollstreams finden Sie unter Arbeiten mit Protokollgruppen und Protokollstreams.

Arbeiten mit CloudWatch Anwendungsprotokollierungsoptionen

Verwenden Sie die folgenden API-Aktionen, um einer neuen oder vorhandenen Anwendung eine CloudWatch Protokolloption hinzuzufügen oder eine Protokolloption für eine vorhandene Anwendung zu ändern. Weitere Informationen zur Verwendung einer JSON-Datei für die Eingabe einer API-Aktion finden Sie unter Beispielcode für Managed Service für Apache Flink.

Hinzufügen einer CloudWatch Protokolloption beim Erstellen einer Anwendung

Das folgende Beispiel zeigt, wie Sie die -`CreateApplication`Aktion verwenden, um eine CloudWatch Protokolloption hinzuzufügen, wenn Sie eine Anwendung erstellen. Ersetzen Sie im Beispiel den *`Amazon-Ressourcennamen (ARN) des CloudWatch Protokollstreams, um ihn der neuen Anwendung durch Ihre eigenen Informationen hinzuzufügen`*. Weitere Informationen zur Aktion finden Sie unter `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>"
 }]
}
```

Hinzufügen einer CloudWatch Protokolloption zu einer vorhandenen Anwendung

Das folgende Beispiel zeigt, wie Sie mit der -AddApplicationCloudWatchLoggingOptionAktion eine CloudWatch Protokolloption zu einer vorhandenen Anwendung hinzufügen. Ersetzen Sie im Beispiel jeden *Platzhalter für Benutzereingaben* durch Ihre eigenen Informationen. Weitere Informationen zur Aktion finden Sie unter 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>
}
```

Aktualisieren einer vorhandenen CloudWatch Protokolloption

Das folgende Beispiel zeigt, wie Sie die -UpdateApplicationAktion verwenden, um eine vorhandene CloudWatch Protokolloption zu ändern. Ersetzen Sie im Beispiel jeden *Platzhalter für Benutzereingaben* durch Ihre eigenen Informationen. Weitere Informationen zur Aktion finden Sie unter 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>
}
```

Löschen einer CloudWatch Protokolloption aus einer Anwendung

Das folgende Beispiel zeigt, wie Sie die -`DeleteApplicationCloudWatchLoggingOption`Aktion verwenden, um eine vorhandene CloudWatch Protokolloption zu löschen. Ersetzen Sie im Beispiel jeden *`Platzhalter für Benutzereingaben`* durch Ihre eigenen Informationen. Weitere Informationen zur Aktion finden Sie unter `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>
}
```

Einstellung der Anwendungsprotokollierungsebene

Um die Ebene der Anwendungsprotokollierung festzulegen, verwenden Sie den `MonitoringConfiguration`-Parameter der `CreateApplication`-Aktion oder den `MonitoringConfigurationUpdate`-Parameter der `UpdateApplication`-Aktion.

Hinweise zu den Protokollebenen von Anwendungen finden Sie unter the section called “Anwendungsüberwachungsebenen”.

Legen Sie die Anwendungsprotokollierungsebene fest, wenn Sie eine Anwendung erstellen

In der folgenden Beispielanforderung für die `CreateApplication`-Aktion wird die Protokollebene der Anwendung auf `INFO` festgelegt.

```
{
   "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"
}
```

Aktualisieren der Anwendungsprotokollierungsebene

In der folgenden Beispielanforderung für die `UpdateApplication`-Aktion wird die Protokollebene der Anwendung auf `INFO` festgelegt.

```
{
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "MonitoringConfigurationUpdate": {
            "ConfigurationTypeUpdate": "CUSTOM",
            "LogLevelUpdate": "INFO"
         }
      }
   }
}
```

Hinzufügen von Berechtigungen zum Schreiben in den CloudWatch Protokollstream

Managed Service für Apache Flink benötigt Berechtigungen, um Fehlkonfigurationsfehler in zu schreiben CloudWatch. Sie können diese Berechtigungen der AWS Identity and Access Management (IAM)-Rolle hinzufügen, die Managed Service für Apache Flink annimmt.

Weitere Informationen zur Verwendung einer IAM-Rolle für Managed Service für Apache Flink finden Sie unter. Identity and Access Management für Amazon Managed Service für Apache Flink

Vertrauensrichtlinie

Zum Erteilen von Managed Service für Apache Flink-Berechtigungen, um eine IAM-Rolle anzunehmen, können Sie an die Serviceausführungs-Rolle die folgende Vertrauensrichtlinie anhängen.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanlaytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Berechtigungsrichtlinie

Um einer Anwendung Berechtigungen zum Schreiben von Protokollereignissen in CloudWatch von einer Managed Service für Apache Flink-Ressource zu erteilen, können Sie die folgende IAM-Berechtigungsrichtlinie verwenden. Geben Sie die richtigen Amazon-Ressourcennamen (ARNs) für Ihre Protokollgruppe und Ihren Stream an.

```
{
    "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:*",
```

```
            ]
        }
    ]
}
```

Anwendungsüberwachungsebenen

Sie steuern die Generierung von Anwendungsprotokollmeldungen mithilfe von Überwachung der Metrikebene und Überwachung der Protokollebene der Anwendung.

Die Überwachung der Metrikebene der Anwendung steuert die Granularität der Protokollnachrichten. Die Überwachung der Metrikebenen sind wie folgt definiert:

- Anwendung: Metriken beziehen sich auf die gesamte Anwendung.
- Aufgabe: Metriken beziehen sich auf jede Aufgabe. Weitere Informationen zu Aufgaben finden Sie unter the section called “Skalierung”.
- Operator: Metriken sind auf jeden Operator beschränkt. Weitere Informationen zu Operatoren finden Sie unter the section called “DataStream-API Operatoren”.
- Parallelität: Metriken sind auf Anwendungsparallelität beschränkt. Sie können diese Metrikebene nur mit dem - MonitoringConfigurationUpdateParameter der UpdateApplication-API festlegen. Diese Metrikebene kann nicht mithilfe der Konsole festgelegt werden. Informationen zur Parallelität finden Sie unter the section called “Skalierung”.

Die Überwachung der Protokollebene der Anwendung steuert die Ausführlichkeit des Anwendungsprotokolls. Die Überwachung der Protokollebene ist wie folgt definiert:

- Fehler: Mögliche katastrophale Ereignisse der Anwendung.
- Warnung: Potenziell schädliche Situationen der Anwendung.
- Info: Informative und vorübergehende Ausfälle der Anwendung. Wir empfehlen die Verwendung dieser Protokollierungsebene.
- Debug: Detaillierte Informationsereignisse, die für das Debuggen einer Anwendung am nützlichsten sind. Hinweis: Verwenden Sie diese Ebene nur für temporäre Debugging-Zwecke.

Bewährte Methoden der Protokollierung

Wir empfehlen, dass Ihre Anwendung die Protokollierungsebene Info verwendet. Wir empfehlen diese Stufe, um sicherzustellen, dass Sie Apache Flink-Fehler sehen, die auf der Informations-Ebene und nicht auf der Fehler-Ebene protokolliert werden.

Wir empfehlen, die Debug-Ebene nur vorübergehend zu verwenden, um Anwendungsprobleme zu untersuchen. Wechseln Sie zurück zur Informations-Ebene, wenn das Problem behoben ist. Die Verwendung der Debug-Protokollierungsebene wirkt sich erheblich auf die Leistung Ihrer Anwendung aus.

Eine übermäßige Protokollierung kann sich auch erheblich auf die Anwendungsleistung auswirken. Wir empfehlen beispielsweise, nicht für jeden verarbeiteten Datensatz einen Protokolleintrag zu schreiben. Eine übermäßige Protokollierung kann zu schwerwiegenden Engpässen bei der Datenverarbeitung und zu einem Gegendruck beim Lesen von Daten aus den Quellen führen.

Protokollierung von Problemlösungen

Wenn Anwendungsprotokolle nicht in den Protokollstream geschrieben werden, überprüfen Sie Folgendes:

- Stellen Sie sicher, dass die IAM-Rolle und die Richtlinien Ihrer Anwendung korrekt sind. Die Richtlinie Ihrer Anwendung benötigt die folgenden Berechtigungen, um auf Ihren Protokollstream zugreifen zu können:
  - `logs:PutLogEvents`
  - `logs:DescribeLogGroups`
  - `logs:DescribeLogStreams`

  Weitere Informationen finden Sie unter the section called “Hinzufügen von Berechtigungen zum Schreiben in den CloudWatch Protokollstream”.
- Überprüfen Sie, dass Ihre Anwendung ausgeführt wird. Um den Status Ihrer Anwendung zu überprüfen, zeigen Sie die Seite Ihrer Anwendung in der Konsole an oder verwenden Sie die ListApplications Aktionen DescribeApplication oder .
- Überwachen Sie CloudWatch Metriken wie `downtime`, um andere Anwendungsprobleme zu diagnostizieren. Weitere Informationen zum Lesen von CloudWatch Metriken finden Sie unter Metriken und Dimensionen in Managed Service für Apache Flink.

Nächster Schritt

Nachdem Sie die CloudWatch Protokollierung in Ihrer Anwendung aktiviert haben, können Sie CloudWatch Logs Insights verwenden, um Ihre Anwendungsprotokolle zu analysieren. Weitere Informationen finden Sie unter the section called “Analysieren von Protokollen”.

Analysieren von Protokollen mit CloudWatch Logs Insights

Nachdem Sie Ihrer Anwendung wie im vorherigen Abschnitt beschrieben eine CloudWatch Protokollierungsoption hinzugefügt haben, können Sie CloudWatch Logs Insights verwenden, um Ihre Protokollstreams nach bestimmten Ereignissen oder Fehlern abzufragen.

CloudWatch Mit Logs Insights können Sie interaktiv Ihre Protokolldaten in - CloudWatch Protokollen durchsuchen und analysieren.

Informationen zu den ersten Schritten mit CloudWatch Logs Insights finden Sie unter Analysieren von Protokolldaten mit CloudWatch Logs Insights.

Ausführen einer Beispielabfrage

In diesem Abschnitt wird beschrieben, wie Sie eine CloudWatch Logs-Insights-Beispielabfrage ausführen.

Voraussetzungen

- Vorhandene Protokollgruppen und Protokollstreams, die in - CloudWatch Protokollen eingerichtet wurden.
- In - CloudWatch Protokollen gespeicherte Protokolle.

Wenn Sie Services wie AWS CloudTrail, Amazon Route 53 oder Amazon VPC verwenden, haben Sie wahrscheinlich bereits Protokolle von diesen Services eingerichtet, um zu - CloudWatch Protokolle zu gelangen. Weitere Informationen zum Senden von Protokollen an CloudWatch Logs finden Sie unter Erste Schritte mit CloudWatch Protokollen.

Abfragen in CloudWatch Logs Insights geben entweder eine Reihe von Feldern aus Protokollereignissen oder das Ergebnis einer mathematischen Aggregation oder einer anderen Operation zurück, die für Protokollereignisse ausgeführt wird. Dieser Abschnitt demonstriert eine Abfrage, die eine Liste von Protokollereignissen zurückgibt.

So führen Sie eine CloudWatch Logs-Insights-Beispielabfrage aus

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im Navigationsbereich Insights aus.
3. Der Abfrage-Editor fast am oberen Ende des Bildschirms enthält eine Standardabfrage, die die 20 letzten Protokollereignisse zurückgibt. Wählen Sie oberhalb des Abfrage-Editors eine Protokollgruppe zur Abfrage aus.

   Wenn Sie eine Protokollgruppe auswählen, erkennt CloudWatch Logs Insights automatisch Felder in den Daten in der Protokollgruppe und zeigt sie unter Erkannte Felder im rechten Bereich an. Dort finden Sie auch ein Balkendiagramm der Protokollereignisse in dieser Protokollgruppe im Zeitverlauf. Dieses Balkendiagramm zeigt die Verteilung der Ereignisse in der Protokollgruppe, die Ihrer Abfrage und Ihrem Zeitraum entspricht, nicht nur die in der Tabelle angezeigten Ereignisse.
4. Wählen Sie Abfrage ausführen.

   Die Ergebnisse der Abfrage werden angezeigt. In diesem Beispiel sind die Ergebnisse die letzten 20 Protokollereignisse aller Art.
5. Um alle Felder eines der zurückgegebenen Protokollereignisse anzuzeigen, wählen Sie den Pfeil links neben diesem Protokollereignis aus.

Weitere Informationen zum Ausführen und Ändern von CloudWatch Logs-Insights-Abfragen finden Sie unter Ausführen und Ändern einer Beispielabfrage.

Beispielabfragen

Dieser Abschnitt enthält Beispielabfragen von CloudWatch Logs Insights für die Analyse von Anwendungsprotokollen von Managed Service für Apache Flink. Diese Abfragen suchen nach mehreren Beispielfehlerbedingungen und dienen als Vorlagen für das Schreiben von Abfragen, die andere Fehlerbedingungen finden.

> **Note**
>
> Ersetzen Sie die Region (*us-west-2*), die Konto-ID (*012345678901*) und den Anwendungsnamen (*YourApplication*) in den folgenden Abfragebeispielen durch die Region Ihrer Anwendung und Ihre Konto-ID.

Dieses Thema enthält die folgenden Abschnitte:

- Betriebsabläufe analysieren: Verteilung der Aufgaben
- Betriebsabläufe analysieren: Änderung der Parallelität
- Fehler analysieren: Zugriff verweigert
- Fehler analysieren: Quelle oder Senke nicht gefunden
- Fehler analysieren: Fehler im Zusammenhang mit Anwendungsaufgaben

Betriebsabläufe analysieren: Verteilung der Aufgaben

Die folgende CloudWatch Logs-Insights-Abfrage gibt die Anzahl der Aufgaben zurück, die der Apache Flink Job Manager auf die Aufgabenmanager verteilt. Sie müssen den Zeitrahmen der Abfrage so einstellen, dass er einer Jobausführung entspricht, sodass die Abfrage keine Aufgaben aus früheren Jobs zurückgibt. Informationen zur Parallelität finden Sie unter Skalierung.

```
fields @timestamp, message
| filter message like /Deploying/
| parse message " to flink-taskmanager-*" as @tmid
| stats count(*) by @tmid
| sort @timestamp desc
| limit 2000
```

Die folgende CloudWatch Logs-Insights-Abfrage gibt die jedem Task Manager zugewiesenen Unteraufgaben zurück. Die Gesamtzahl der Unteraufgaben ist die Summe der Parallelität jeder Aufgabe. Die Aufgabenparallelität wird aus der Operatorparallelität abgeleitet und entspricht standardmäßig der Parallelität der Anwendung, sofern Sie sie nicht im Code durch Angabe von `setParallelism` ändern. Informationen zur Einstellung der Operatorparallelität finden Sie unter Einstellen der Parallelität: Operatorebene in der Dokumentation von Apache Flink.

```
fields @timestamp, @tmid, @subtask
```

```
| filter message like /Deploying/
| parse message "Deploying * to flink-taskmanager-*" as @subtask, @tmid
| sort @timestamp desc
| limit 2000
```

Informationen zu Aufgabenplanung finden Sie unter Aufträge und Planung in der Apache-Flink-Dokumentation.

Betriebsabläufe analysieren: Änderung der Parallelität

Die folgende CloudWatch Logs-Insights-Abfrage gibt Änderungen an der Parallelität einer Anwendung zurück (z. B. aufgrund der automatischen Skalierung). Diese Abfrage gibt auch manuelle Änderungen an der Parallelität der Anwendung zurück. Weitere Informationen zum Auto Scaling finden Sie unter the section called “Automatische Skalierung”.

```
fields @timestamp, @parallelism
| filter message like /property: parallelism.default, /
| parse message "default, *" as @parallelism
| sort @timestamp asc
```

Fehler analysieren: Zugriff verweigert

Die folgende CloudWatch Logs-Insights-Abfrage gibt `Access Denied` Protokolle zurück.

```
fields @timestamp, @message, @messageType
| filter applicationARN like /arn:aws:kinesisanalytics<us-west-2>:<012345678901>:application\/<YourApplication>/
| filter @message like /AccessDenied/
| sort @timestamp desc
```

Fehler analysieren: Quelle oder Senke nicht gefunden

Die folgende CloudWatch Logs-Insights-Abfrage gibt `ResourceNotFound` Protokolle zurück. -`ResourceNotFound`Protokolle ergeben sich, wenn eine Kinesis-Quelle oder -Senke nicht gefunden wird.

```
fields @timestamp,@message
| filter applicationARN like /arn:aws:kinesisanalytics<us-west-2>:<012345678901>:application\/<YourApplication>/
| filter @message like /ResourceNotFoundException/
```

```
| sort @timestamp desc
```

Fehler analysieren: Fehler im Zusammenhang mit Anwendungsaufgaben

Die folgende CloudWatch Logs-Insights-Abfrage gibt die aufgabenbezogenen Fehlerprotokolle einer Anwendung zurück. Diese Protokolle entstehen, wenn der Status einer Anwendung von `RUNNING` zu wechselt `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
```

Bei Anwendungen, die Apache Flink Version 1.8.2 und früher verwenden, führen aufgabenbezogene Fehler dazu, dass der Anwendungsstatus stattdessen von `RUNNING` zu wechselt `FAILED`. Wenn Sie Apache Flink 1.8.2 und frühere Versionen verwenden, verwenden Sie die folgende Abfrage, um nach Fehlern im Zusammenhang mit der Anwendungsaufgabe zu suchen:

```
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
```

Anzeigen von Metriken und Dimensionen in Managed Service für Apache Flink

Dieses Thema enthält die folgenden Abschnitte:

- Anwendungsmetriken
- Kinesis Data Streams-Konnektormetriken
- Amazon MSK-Konnektor-Metriken
- Apache Zeppelin-Metriken
- Anzeigen von CloudWatch Metriken
- Festlegen von Berichtsebenen für CloudWatch Metriken
- Benutzerdefinierte Metriken mit Amazon Managed Service für Apache Flink verwenden

- Verwenden von CloudWatch Alarmen mit Amazon Managed Service für Apache Flink

Wenn Ihr Managed Service für Apache Flink eine Datenquelle verarbeitet, meldet Managed Service für Apache Flink die folgenden Metriken und Dimensionen an Amazon CloudWatch.

Anwendungsmetriken

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
`backPressuredTimeMsPerSecond*`	Millisekunden	Die Zeit (in Millisekunden), in der diese Aufgabe oder dieser Operator pro Sekunde unter Gegendruck gesetzt wird.	Aufgabe, Operator, Parallelität	*Nur für Managed Service für Apache Flink-Anwendungen verfügbar, auf denen Flink Version 1.13 ausgeführt wird. Diese Metriken können nützlich sein, um Engpässe in einer Anwendung zu identifizieren.
`busyTimeMsPerSecond*`	Millisekunden	Die Zeit (in Millisekunden), in der diese Aufgabe oder dieser Operator pro Sekunde beschäftigt	Aufgabe, Operator, Parallelität	*Nur für Managed Service für Apache Flink-Anwendungen verfügbar, auf denen Flink Version 1.13

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
		(weder inaktiv noch unter Gegendruck gesetzt) ist. Kann NaN sein, wenn der Wert nicht berechnet werden konnte.		ausgeführt wird. Diese Metriken können nützlich sein, um Engpässe in einer Anwendung zu identifizieren.
`cpuUtilization`	Prozentsatz	Prozentsatz der CPU-Auslastung in allen Task-Managern. Wenn es beispielsweise fünf Taskmanager gibt, veröffentlicht Managed Service für Apache Flink pro Berichtsintervall fünf Beispiele dieser Metrik.	Anwendung	Sie können diese Metrik verwenden, um die minimale, durchschnittliche und maximale CPU-Auslastung in Ihrer Anwendung zu überwachen. Die `CPUUtilization` Metrik berücksichtigt nur die CPU-Auslastung des TaskManager JVM-Prozesses, der im Container ausgeführt wird.

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise	
`containerCPUUtilization`	Prozentsatz	Gesamtprozentsatz der CPU-Auslastung in allen Task-Manager-Containern im Flink-Anwendungscluster. Wenn es beispielsweise fünf Aufgabenmanager gibt, gibt es entsprechend fünf TaskManager Container und Managed Service für Apache Flink veröffentlicht 2 x fünf Stichproben dieser Metrik pro Berichtsintervall von einer Minute.	Anwendung	Sie wird pro Container wie folgt berechnet: Gesamt-CPU-Zeit (in Sekunden), die vom Container verbraucht wird * 100/Container-CPU-Limit (in CPUs/Sekunden) Die `CPUUtilization` Metrik berücksichtigt nur die CPU-Auslastung des TaskManager JVM-Prozesses, der im Container ausgeführt wird. Es gibt andere Komponenten, die außerhalb der JVM innerhalb desselben Containers ausgeführt	

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise	
				werden. Die `containerCPUUtilization` -Metrik gibt Ihnen ein vollständigeres Bild, einschließlich aller Prozesse im Hinblick auf die CPU-Auslastung im Container und die daraus resultierenden Ausfälle.	

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
`containerMemoryUtilization`	Prozentsatz	Gesamtprozentsatz der Speicherauslastung in allen Task-Manager-Containern im Flink-Anwendungscluster. Wenn es beispielsweise fünf Aufgabenmanager gibt, gibt es entsprechend fünf TaskManager Container und Managed Service für Apache Flink veröffentlicht 2 x fünf Stichproben dieser Metrik pro Berichtsintervall von einer Minute.	Anwendung	Sie wird pro Container wie folgt berechnet: Speichernutzung des Containers (Byte) * 100/ Container-Speicherlimit gemäß der Pod-Bereitstellungsspezifikation (in Byte) Die `ManagedMemoryUtilzations` Metriken `HeapMemoryUtilization` und berücksichtigen nur bestimmte Speichermetriken wie Heap Memory Usage of TaskManager JVM oder Managed

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise	
				Memory (Speichernutzung außerhalb von JVM für native Prozesse wie RocksDB State Backend). Die `containerMemoryUtilization` - Metrik gibt Ihnen ein vollständigeres Bild, da sie den festgelegten Arbeitsspeicher mit einbezieht, wodurch die gesamte Speicherschöpfung besser erfasst werden kann. Nach seiner Ausschöpfung führt dies zu `Out of Memory Error` für den TaskManager Pod.	

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
`containerDiskUtilization`	Prozentsatz	Gesamtprozentsatz der Festplattenauslastung in allen Task-Manager-Containern im Flink-Anwendungscluster. Wenn es beispielsweise fünf Aufgabenmanager gibt, gibt es entsprechend fünf TaskManager Container und Managed Service für Apache Flink veröffentlicht 2 x fünf Stichproben dieser Metrik pro Berichtsintervall von einer Minute.	Anwendung	Sie wird pro Container wie folgt berechnet: Festplattennutzung in Byte * 100/ Festplattenlimit für Container in Byte Bei Containern steht dies für die Nutzung des Dateisystems, auf dem das Root-Volume des Containers eingerichtet ist.

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise	
currentInputWatermark	Millisekunden	Das letzte Wasserzeichen, das diese Anwendung/dieser Operator/Aufgabe/dieser Thread erhalten hat	Anwendung, Operator, Aufgabe, Parallelität	Dieser Datensatz wird nur für Dimensionen mit zwei Eingaben ausgegeben. Dies ist der Mindestwert der zuletzt empfangenen Wasserzeichen.	
currentOutputWatermark	Millisekunden	Das letzte Wasserzeichen, das diese Anwendung/dieser Operator/Aufgabe/dieser Thread ausgegeben hat	Anwendung, Operator, Aufgabe, Parallelität		

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
`downtime`	Millisekunden	Bei Aufträgen, die sich derzeit in einer Situation befinden, in der ein Fehler aufgetreten ist oder der wiederhergestellt wird, ist dies die Zeit, die während dieses Ausfalls verstrichen ist.	Anwendung	Diese Kennzahl misst die Zeit, die verstrichen ist, während ein Job ausfällt oder wiederhergestellt wird. Diese Metrik gibt 0 für laufende Jobs und -1 für abgeschlossene Jobs zurück. Wenn diese Metrik nicht 0 oder -1 ist, bedeutet dies, dass der Apache Flink-Job für die Anwendung nicht ausgeführt werden konnte.

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise	
`fullRestarts`	Anzahl	Gibt an, wie oft dieser Job seit seiner Übermittlung vollständig neu gestartet wurde. Mit dieser Metrik werden keine detaillierten Neustarts gemessen.	Anwendung	Sie können diese Metrik verwenden, um den allgemeinen Zustand von Anwendungen zu bewerten. Neustarts können während der internen Wartung durch Managed Service für Apache Flink erfolgen. Neustarts, die höher als normal sind, können auf ein Problem mit der Anwendung hinweisen.	

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
`heapMemoryUtilization`	Prozentsatz	Gesamtauslastung des Heap-Speichers in allen Task-Managern. Wenn es beispielsweise fünf Taskmanager gibt, veröffentlicht Managed Service für Apache Flink pro Berichtsintervall fünf Beispiele dieser Metrik.	Anwendung	Sie können diese Metrik verwenden, um die minimale, durchschnittliche und maximale Heap-Speicherauslastung in Ihrer Anwendung zu überwachen. Der berücksichtigt `HeapMemoryUtilization` nur bestimmte Speichermetriken wie die Heap-Speichernutzung von TaskManager JVM.

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
`idleTimeMsPerSecond*`	Millisekunden	Die Zeit (in Millisekunden), in der sich diese Task oder dieser Operator pro Sekunde im Leerlauf befindet (keine zu verarbeitenden Daten hat). Bei der Leerlaufzeit wird die Zeit nicht berücksichtigt, in der Gegendruck ausgeübt wird, wenn also die Aufgabe unter Gegendruck steht, handelt es sich nicht um Inaktivität.	Aufgabe, Operator, Parallelität	*Nur für Managed Service für Apache Flink-Anwendungen verfügbar, auf denen Flink Version 1.13 ausgeführt wird. Diese Metriken können nützlich sein, um Engpässe in einer Anwendung zu identifizieren.

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
`lastCheckpointSize`	Bytes	Die Gesamtgröße des letzten Prüfpunkts	Anwendung	Sie können diese Metrik verwenden, um die Speicherauslastung laufender Anwendungen zu ermitteln. Wenn der Wert dieser Metrik steigt, kann dies darauf hindeuten, dass ein Problem mit Ihrer Anwendung vorliegt, z. B. ein Speicherleck oder ein Engpass.

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
`lastCheckpointDuration`	Millisekunden	Die Zeit, die benötigt wurde, um den letzten Prüfpunkt abzuschließen	Anwendung	Diese Kennzahl misst die Zeit, die benötigt wurde, um den letzten Prüfpunkt abzuschließen. Wenn der Wert dieser Metrik steigt, kann dies darauf hindeuten, dass ein Problem mit Ihrer Anwendung vorliegt, z. B. ein Speicherleck oder ein Engpass. In einigen Fällen können Sie dieses Problem beheben, indem Sie die Prüfpunktprüfung deaktivieren.

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
managedMemoryUsed*	Bytes	Die derzeit verwendete verwaltete Speicherm enge.	Anwendung, Operator, Aufgabe, Parallelität	*Nur für Managed Service für Apache Flink-Anwendungen verfügbar, auf denen Flink Version 1.13 ausgeführt wird. Dies bezieht sich auf Speicher, der von Flink außerhalb des Java-Heaps verwaltet wird. Es wird für das RocksDB-State-Backend verwendet und ist auch für Anwendungen verfügbar.

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
`managedMemoryTotal*`	Bytes	Die Gesamtgröße des verwalteten Speichers.	Anwendung, Operator, Aufgabe, Parallelität	*Nur für Managed Service für Apache Flink-Anwendungen verfügbar, auf denen Flink Version 1.13 ausgeführt wird. Dies bezieht sich auf Speicher, der von Flink außerhalb des Java-Heaps verwaltet wird. Es wird für das RocksDB-State-Backend verwendet und ist auch für Anwendungen verfügbar. Die `ManagedMemoryUtilizations` -Metrik berücksichtigt nur bestimmte Speichermetriken wie Managed

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise	
				Memory (Speichernutzung außerhalb von JVM für native Prozesse wie RocksDB State Backend)	
`managedMemoryUtilization*`	Prozentsatz	Abgeleitet von managedMemoryUsed/managedMemoryTotal	Anwendung, Operator, Aufgabe, Parallelität	*Nur für Managed Service für Apache Flink-Anwendungen verfügbar, auf denen Flink Version 1.13 ausgeführt wird. Dies bezieht sich auf Speicher, der von Flink außerhalb des Java-Heaps verwaltet wird. Es wird für das RocksDB-State-Backend verwendet und ist auch für Anwendungen verfügbar.	

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise	
`numberOfFailedCheckpoints`	Anzahl	Gibt an, wie oft die Prüfpunktüberprüfung fehlgeschlagen ist.	Anwendung	Sie können diese Metrik verwenden, um den Zustand und den Fortschritt von Anwendungen zu überwachen. Prüfpunkte können aufgrund von Anwendungsproblemen wie Durchsatz- oder Berechtigungsprobleme fehlschlagen.	

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
`numRecordsIn*`	Anzahl	Die Gesamtzahl der Datensätze, die diese Anwendung, dieser Operator oder diese Aufgabe erhalten hat.	Anwendung, Operator, Aufgabe, Parallelität	*Um die SUM-Statistik über einen bestimmten Zeitraum (Sekunde/Minute) anzuwenden: • Wählen Sie die Metrik auf der richtigen Ebene aus. Wenn Sie die Metrik für einen Operator verfolgen, müssen Sie die entsprechenden Operator-Metriken auswählen. • Da Managed Service für Apache Flink 4 Metrik-Snapshots pro Minute erstellt, sollte die folgende

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
				Metrikmathematik verwendet werden: m1/4, wobei m1 die SUM-Statistik über einen Zeitraum (Sekunde/Minute) ist Die Ebene der Metrik gibt an, ob diese Metrik die Gesamtzahl der Datensätze misst, die die gesamte Anwendung, ein bestimmter Operator oder eine bestimmte Aufgabe empfangen hat.

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise	
numRecordsInPerSecond*	Anzahl/Sekunde	Die Gesamtzahl der Datensätze, die diese Anwendung, dieser Operator oder diese Aufgabe pro Sekunde erhalten hat.	Anwendung, Operator, Aufgabe, Parallelität	*Um die SUM-Statistik über einen bestimmten Zeitraum (Sekunde/Minute) anzuwenden: • Wählen Sie die Metrik auf der richtigen Ebene aus. Wenn Sie die Metrik für einen Operator verfolgen, müssen Sie die entsprechenden Operator-Metriken auswählen. • Da Managed Service für Apache Flink 4 Metrik-Snapshots pro Minute erstellt, sollte die folgende	

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise	
				Metrikmathematik verwendet werden: m1/4, wobei m1 die SUM-Statistik über einen Zeitraum (Sekunde/Minute) ist Die Ebene der Metrik gibt an, ob diese Metrik die Gesamtzahl der Datensätze misst, die die gesamte Anwendung, ein bestimmter Operator oder eine bestimmte Aufgabe pro Sekunde empfangen hat.	

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
`numRecordsOut*`	Anzahl	Die Gesamtzahl der Datensätze, die diese Anwendung, dieser Operator oder diese Aufgabe ausgegeben hat.	Anwendung, Operator, Aufgabe, Parallelität	*Um die SUM-Statistik über einen bestimmten Zeitraum (Sekunde/Minute) anzuwenden: • Wählen Sie die Metrik auf der richtigen Ebene aus. Wenn Sie die Metrik für einen Operator verfolgen, müssen Sie die entsprechenden Operator-Metriken auswählen. • Da Managed Service für Apache Flink 4 Metrik-Snapshots pro Minute erstellt, sollte die folgende

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise	
				Metrikmathematik verwendet werden: m1/4, wobei m1 die SUM-Statistik über einen Zeitraum (Sekunde/Minute) ist Die Ebene der Metrik gibt an, ob diese Metrik die Gesamtzahl der Datensätze misst, die die gesamte Anwendung, ein bestimmter Operator oder eine bestimmte Aufgabe ausgegeben hat.	

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise	
`numLateRecordsDropped*`	Anzahl	Anwendung, Operator, Aufgabe, Parallelität		*Um die SUM-Statistik über einen bestimmten Zeitraum (Sekunde/Minute) anzuwenden: • Wählen Sie die Metrik auf der richtigen Ebene aus. Wenn Sie die Metrik für einen Operator verfolgen, müssen Sie die entsprechenden Operator-Metriken auswählen. • Da Managed Service für Apache Flink 4 Metrik-Snapshots pro Minute erstellt, sollte die folgende	

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise	
				Metrikmathematik verwendet werden: m1/4, wobei m1 die SUM-Statistik über einen Zeitraum (Sekunde/Minute) ist Die Anzahl der Datensätze, die dieser Operator oder diese Aufgabe aufgrund einer verspätetenen Ankunft gelöscht hat.	

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
`numRecordsOutPerSecond*`	Anzahl/Sekunde	Die Gesamtzahl der Datensätze, die diese Anwendung, dieser Operator oder diese Aufgabe pro Sekunde ausgegeben hat.	Anwendung, Operator, Aufgabe, Parallelität	*Um die SUM-Statistik über einen bestimmten Zeitraum (Sekunde/Minute) anzuwenden: • Wählen Sie die Metrik auf der richtigen Ebene aus. Wenn Sie die Metrik für einen Operator verfolgen, müssen Sie die entsprechenden Operator-Metriken auswählen. • Da Managed Service für Apache Flink 4 Metrik-Snapshots pro Minute erstellt, sollte die folgende

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
				Metrikmathematik verwendet werden: m1/4, wobei m1 die SUM-Statistik über einen Zeitraum (Sekunde/Minute) ist Die Ebene der Metrik gibt an, ob diese Metrik die Gesamtzahl der Datensätze misst, die die gesamte Anwendung, ein bestimmter Operator oder eine bestimmte Aufgabe pro Sekunde ausgegeben hat.

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
`oldGenerationGCCount`	Anzahl	Die Gesamtzahl der alten Garbage-Collection-Vorgänge, die in allen Task-Managern stattgefunden haben.	Anwendung	
`oldGenerationGCTime`	Millisekunden	Die Gesamtzeit, die für die Durchführung alter Garbage-Collection-Vorgänge aufgewendet wurde.	Anwendung	Sie können diese Metrik verwenden, um die Summe, den Durchschnitt und die maximale Zeit für die Garbage Collection zu überwachen.

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
`threadCount`	Anzahl	Die Gesamtzahl der von der Anwendung verwendeten Live-Threads.	Anwendung	Diese Metrik misst die Anzahl der Threads, die vom Anwendungscode verwendet werden. Dies ist nicht dasselbe wie Anwendungsparallelität.
`uptime`	Millisekunden	Die Zeit, zu der der Job ohne Unterbrechung ausgeführt wurde.	Anwendung	Sie können diese Metrik verwenden, um festzustellen, ob ein Job erfolgreich ausgeführt wird. Diese Metrik gibt -1 für abgeschlossene Jobs zurück.

Kinesis Data Streams-Konnektormetriken

AWS gibt alle Datensätze für Kinesis Data Streams zusätzlich zu den folgenden aus:

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
`millisbehindLatest`	Millisekunden	Die Anzahl der Millisekunden, die der Verbraucher hinter der Spitze des Streams zurückliegt. Dies zeigt an, wie weit der Verbraucher hinter der aktuellen Zeit zurückliegt.	Anwendung (für Stream), Parallelität (für ShardId)	• Der Wert 0 gibt an, dass die Datenverarbeitung aktuell ist und dass zurzeit keine neuen zu verarbeitenden Datensätze vorhanden sind. Die Metrik eines bestimmten Shards kann durch den Stream-Namen und die Shard-ID angegeben werden. • Ein Wert von -1 gibt an, dass der Service noch keinen Wert für die Metrik gemeldet hat.
`bytesRequestedPerFetch`	Bytes	Die in einem einzigen Aufruf an `getRecord`	Anwendung (für Stream), Parallelität (für ShardId)	

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
		s angeforderten Bytes.		

Amazon MSK-Konnektor-Metriken

AWS gibt alle Datensätze für Amazon MSK zusätzlich zu den folgenden aus:

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
currentoffsets	N/A	Der aktuelle Lese-Offset des Verbrauchers für jede Partition. Die Metrik einer bestimmten Partition kann anhand des Themennamens und der Partitions-ID angegeben werden.	Anwendung (für Thema), Parallelität (für Partition Id)	
commitsFailed	N/A	Die Gesamtzahl der Fehler beim Offset-Commit an Kafka, wenn Offset-Commit und Prüfpunktprüfung aktiviert sind.	Anwendung, Operator, Aufgabe, Parallelität	Das Zurückschreiben von Offsets an Kafka ist nur ein Mittel, um den Verbrauchserfortschritt aufzudecken. Ein Commit-Fehler beeinträchtigt also nicht

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
				die Integrität der Prüfpunkt-Partitions-Offsets von Flink.
`commitsSucceeded`	N/A	Die Gesamtzahl erfolgreichen Offset-Commits an Kafka, wenn Offset-Commit und Prüfpunkt prüfung aktiviert sind.	Anwendung, Operator, Aufgabe, Parallelität	
`committed offsets`	N/A	Die letzten erfolgreich an Kafka übergebenen Offsets für jede Partition. Die Metrik einer bestimmten Partition kann anhand des Themennamens und der Partitions-ID angegeben werden.	Anwendung (für Thema), Parallelität (für Partition Id)	

Metrik	Einheit	Beschreibung	Level	Nutzungshinweise
`records_lag_max`	Anzahl	Die maximale Verzögerung in Bezug auf die Anzahl der Datensätze für jede Partition in diesem Fenster	Anwendung, Operator, Aufgabe, Parallelität	
`bytes_consumed_rate`	Bytes	Die durchschnittliche Anzahl von Bytes, die pro Sekunde für ein Thema verbraucht werden	Anwendung, Operator, Aufgabe, Parallelität	

Apache Zeppelin-Metriken

Für Studio-Notebooks gibt AWS die folgenden Metriken auf Anwendungsebene aus: `KPUs`, `cpuUtilization`, `heapMemoryUtilization`, `oldGenerationGCTime`, `oldGenerationGCCount`, und `threadCount`. Darüber hinaus werden die in der folgenden Tabelle aufgeführten Metriken auch auf Anwendungsebene ausgegeben.

Metrik	Einheit	Beschreibung	Prometheus-Name
`zeppelinCpuUtilization`	Prozentsatz	Gesamtprozentsatz der CPU-Auslastung auf dem Apache Zeppelin-Server.	`process_cpu_usage`
`zeppelinHeapMemoryUtilization`	Prozentsatz	Gesamtprozentsatz der Heap-Speicherauslastung für	`jvm_memory_used_bytes`

Metrik	Einheit	Beschreibung	Prometheus-Name
		den Apache Zeppelin-Server.	
`zeppelinThreadCount`	Anzahl	Die Gesamtzahl der vom Apache Zeppelin-Server verwendeten Live-Threads.	`jvm_threads_live_threads`
`zeppelinWaitingJobs`	Anzahl	Die Anzahl der Apache Zeppelin-Jobs in der Warteschlange, die auf einen Thread warten.	`jetty_threads_jobs`
`zeppelinServerUptime`	Sekunden	Die Gesamtzeit, in der der Server betriebsbereit war.	`process_uptime_seconds`

Anzeigen von CloudWatch Metriken

Sie können CloudWatch Metriken für Ihre Anwendung mithilfe der Amazon- CloudWatch Konsole oder der anzeigenAWS CLI.

So zeigen Sie Metriken mit der CloudWatch Konsole an

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie im Navigationsbereich Metrics (Metriken) aus.
3. Wählen Sie im Bereich CloudWatch Metriken nach Kategorie für Managed Service für Apache Flink eine Metrikkategorie aus.
4. Führen Sie im oberen Bereich einen Bildlauf durch, um die vollständige Liste der Metriken anzuzeigen.

So zeigen Sie Metriken mit der AWS CLI

- Geben Sie als Eingabeaufforderung den folgenden Befehl ein.

```
aws cloudwatch list-metrics --namespace "AWS/KinesisAnalytics" --region region
```

Festlegen von Berichtsebenen für CloudWatch Metriken

Sie können die Ebene der Anwendungsmetriken steuern, die Ihre Anwendung erstellt. Managed Service für Apache Flink unterstützt die folgenden Metrikebenen:

- Anwendung: Die Anwendung meldet für jede Anwendung nur die höchste Stufe an Metriken. Die Metriken von Managed Service für Apache Flink werden standardmäßig auf Anwendungsebene veröffentlicht.
- Aufgabe: Die Anwendung meldet aufgabenspezifische Metrikdimensionen für Metriken, die mit der Berichtsebene Aufgaben-Metrik definiert wurden, wie z. B. die Anzahl der Datensätze pro Sekunde, die in die Anwendung ein- und von ihr ausgehen.
- Operator: Die Anwendung meldet operatorspezifische Metrikdimensionen für Metriken, die mit der Berichtsebene Operator-Metrik definiert wurden, wie z. B. Metriken für jeden Filter- oder Zuordnungsvorgang.
- Parallelität: Die Anwendung erstellt `Task`- und `Operator`–Ebenen-Metriken für jeden Ausführungsthread. Diese Berichtsebene wird wegen übermäßiger Kosten nicht für Anwendungen mit Parallelitätseinstellung über 64 empfohlen.

> **Note**
>
> Aufgrund der Menge an Metrikdaten, die der Service generiert, sollten Sie diese Metrikebene nur zur Fehlerbehebung verwenden. Sie können diese Metrikebene nur mit der CLI festlegen. Diese Metrikebene ist in der Konsole nicht verfügbar.

Die Standardebene ist Anwendung. Die Anwendung meldet Metriken auf der aktuellen Ebene und allen höheren Ebenen. Wenn die Berichtsebene beispielsweise auf Operator gesetzt ist, meldet die Anwendung Anwendungs-, Aufgaben-, and Operator-Metriken.

Sie legen die Berichtsebene für CloudWatch Metriken mithilfe des -`MonitoringConfiguration`Parameters der -`CreateApplication`Aktion oder des -`MonitoringConfigurationUpdate`Parameters der -`UpdateApplication`Aktion fest. Die folgende Beispielanforderung für die `UpdateApplication` Aktion legt die Berichtsebene für CloudWatch Metriken auf Aufgabe fest:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "ApplicationConfigurationUpdate": {
      "FlinkApplicationConfigurationUpdate": {
         "MonitoringConfigurationUpdate": {
            "ConfigurationTypeUpdate": "CUSTOM",
            "MetricsLevelUpdate": "TASK"
         }
      }
   }
}
```

Sie können die Protokollierungsebene auch mithilfe des `LogLevel`-Parameters der Aktion `CreateApplication` oder des `LogLevelUpdate`-Parameters der Aktion `UpdateApplication` konfigurieren. Sie können die folgenden Protokollebenen verwenden:

- `ERROR`: Protokolliert potenziell behebbare Fehlerereignisse.
- `WARN`: Protokolliert Warnereignisse, die zu einem Fehler führen könnten.
- `INFO`: Protokolliert Informationsereignisse.
- `DEBUG`: Protokolliert allgemeine Debugging-Ereignisse.

Weitere Informationen zu Log4j-Protokollierungsebenen finden Sie unter Benutzerdefinierte Protokollebenen in der Apache Log4j-Dokumentation.

Benutzerdefinierte Metriken mit Amazon Managed Service für Apache Flink verwenden

Managed Service für Apache Flink stellt 19 Metriken für bereit CloudWatch, einschließlich Metriken für Ressourcennutzung und Durchsatz. Darüber hinaus können Sie Ihre eigenen Metriken erstellen, um anwendungsspezifische Daten zu verfolgen, z. B. Verarbeitungsereignisse oder den Zugriff auf externe Ressourcen.

Dieses Thema enthält die folgenden Abschnitte:

- So funktioniert's
- Beispiele
- Benutzerdefinierte Metriken anzeigen

So funktioniert's

Benutzerdefinierte Metriken in Managed Service für Apache Flink verwenden das Apache Flink-Metriksystem. Apache Flink-Metriken haben die folgenden Attribute:

- Typ: Der Typ einer Metrik beschreibt, wie Daten gemessen und gemeldet werden. Zu den verfügbaren Apache Flink-Metriktypen gehören Anzahl, Diagramm, Histogramm und Messung. Weitere Informationen zu den Metriktypen von Apache Flink finden Sie unter Metriktypen.

  > **Note**
  >
  > AWS CloudWatch Metriken unterstützen den Metriktyp Histogramm Apache Flink nicht. CloudWatch kann nur Apache Flink-Metriken der Typen Count, Gauge und Meter anzeigen.

- Umfang: Der Umfang einer Metrik besteht aus ihrer Kennung und einer Reihe von Schlüssel-Wert-Paaren, die angeben, wie die Metrik an gemeldet wird CloudWatch. Die Kennung einer Metrik enthält die folgenden Elemente:
  - Einen Systembereich, der die Ebene angibt, auf der die Metrik gemeldet wird (z. B. Operator).
  - Einen Benutzerbereich, der Attribute wie Benutzervariablen oder Metrikgruppennamen definiert. Diese Attribute werden mit `MetricGroup.addGroup(key, value)` oder `MetricGroup.addGroup(name)` definiert.

  Weitere Informationen zu Metrikbereichen finden Sie unter Bereich.

Weitere Informationen zu Apache Flink-Metriken finden Sie unter Metriken in der Apache Flink-Dokumentation.

Um eine benutzerdefinierte Metrik in Ihrem Managed Service für Apache Flink zu erstellen, können Sie von jeder Benutzerfunktion aus, die `RichFunction` erweitert, durch Aufrufen von `GetMetricGroup` auf das Apache Flink-Metriksystem zugreifen. Diese Methode gibt ein MetricGroup Objekt zurück, mit dem Sie benutzerdefinierte Metriken erstellen und registrieren können. Managed Service für Apache Flink meldet alle Metriken, die mit dem Gruppenschlüssel

erstellt wurdenKinesisAnalytics, an CloudWatch. Benutzerdefinierte Metriken, die Sie definieren, weisen folgende Merkmale auf:

- Ihre benutzerdefinierte Metrik hat einen Metriknamen und einen Gruppennamen. Diese Namen müssen aus alphanumerischen Zeichen bestehen.
- Attribute, die Sie im Benutzerbereich definieren (mit Ausnahme der KinesisAnalytics Metrikgruppe), werden als CloudWatch Dimensionen veröffentlicht.
- Benutzerdefinierte Metriken werden standardmäßig auf der Application-Ebene veröffentlicht.
- Dimensionen (Aufgabe/Operator/Parallelismus) werden der Metrik auf der Grundlage der Überwachungsebene der Anwendung hinzugefügt. Sie legen die Überwachungsebene der Anwendung mithilfe des -MonitoringConfigurationParameters der -CreateApplicationAktion oder des - oder -MonitoringConfigurationUpdateParameters der -UpdateApplicationAktion fest.

Beispiele

Die folgenden Codebeispiele zeigen, wie Sie eine Mapping-Klasse erstellen, die eine benutzerdefinierte Metrik erstellt und inkrementiert, und wie Sie die Mapping-Klasse in Ihrer Anwendung implementieren, indem Sie sie einem DataStream-Objekt hinzufügen.

Benutzerdefinierte Metrik für die Datensatzanzahl

Das folgende Codebeispiel zeigt, wie eine Mapping-Klasse erstellt wird, die eine Metrik erstellt, die Datensätze in einem Datenstrom zählt (dieselbe Funktionalität wie die numRecordsIn-Metrik):

```
    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;
        }
    }
```

Im vorherigen Beispiel wird die `valueToExpose`-Variable für jeden Datensatz, den die Anwendung verarbeitet, inkrementiert.

Nachdem Sie Ihre Mapping-Klasse definiert haben, erstellen Sie einen anwendungsinternen Stream, der die Map implementiert:

```
DataStream<String> noopMapperFunctionAfterFilter =
    kinesisProcessed.map(new NoOpMapperFunction("FilteredRecords"));
```

Den vollständigen Code für diese Anwendung finden Sie unter Datensatzanzahl benutzerdefinierte Metrikanwendung.

Benutzerdefinierte Metrik für die Wortanzahl

Das folgende Codebeispiel zeigt, wie eine Mapping-Klasse erstellt wird, die eine Metrik erstellt, die Wörter in einem Datenstrom zählt:

```
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));
                    }
                }
            }
        }
```

Im vorherigen Beispiel wird die `counter`-Variable für jedes Wort, das die Anwendung verarbeitet, inkrementiert.

Nachdem Sie Ihre Mapping-Klasse definiert haben, erstellen Sie einen anwendungsinternen Stream, der die Map implementiert:

```
// 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());
```

Den vollständigen Code für diese Anwendung finden Sie unter Wortanzahl benutzerdefinierte Metrikanwendung.

Benutzerdefinierte Metriken anzeigen

Benutzerdefinierte Metriken für Ihre Anwendung werden in der CloudWatch Metrikkonsole im AWS/KinesisAnalytics Dashboard unter der Metrikgruppe Anwendung angezeigt.

Verwenden von CloudWatch Alarmen mit Amazon Managed Service für Apache Flink

Mithilfe von Amazon- CloudWatch Metrikalarmen überwachen Sie eine CloudWatch Metrik über einen von Ihnen angegebenen Zeitraum. Der Alarm führt eine oder mehrere Aktionen durch, die vom Wert der Metrik oder des Ausdrucks im Vergleich zu einem Schwellenwert in einer Reihe von Zeiträumen abhängt. Eine Aktion könnte beispielsweise der Versand einer Benachrichtigung an ein Amazon Simple Notification Service (Amazon SNS)-Thema sein.

Weitere Informationen zu CloudWatch Alarmen finden Sie unter Verwenden von Amazon CloudWatch Alarms.

Empfohlene -Alarme

Dieser Abschnitt enthält die empfohlenen Alarme für die Überwachung von Managed Service für Apache Flink-Anwendungen.

Die Tabelle beschreibt die empfohlenen Alarme und enthält die folgenden Spalten:

- Metrikausdruck: Die Metrik oder der Metrikausdruck, der anhand des Schwellenwerts getestet werden soll.
- Statistik: Die Statistik, die zur Überprüfung der Metrik verwendet wird, z. B. Durchschnitt.
- Schwellenwert: Für die Verwendung dieses Alarms müssen Sie einen Schwellenwert festlegen, der die Grenze der erwarteten Anwendungsleistung definiert. Sie müssen diesen Schwellenwert ermitteln, indem Sie Ihre Anwendung unter normalen Bedingungen überwachen.
- Beschreibung: Ursachen, die diesen Alarm auslösen könnten, und mögliche Lösungen für diesen Zustand.

Metrikausdruck	Statistik	Threshold	Beschreibung
`Ausfallzeit > 0`	Average	0	A downtime greater than zero indicates that the applicati on has failed. If the value is larger than 0, the application is not processing any data. Recommended for all applications. The `Ausfallzeit` metric measures the duration of an outage. A downtime greater than zero indicates that the applicati on has failed. For

Metrikausdruck	Statistik	Threshold	Beschreibung
			troubleshooting, see Anwendung wird neu gestartet.

Metrikausdruck	Statistik	Threshold	Beschreibung
`RATE (numberOf FailedPrüfpunkte) > 0`	Average	0	This metric counts the number of failed checkpoints since the application started. Depending on the application, it can be 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

Metrikausdruck	Statistik	Threshold	Beschreibung
			data. For troublesh ooting, see Beim Checkpointing kommt es zu einer Zeitübers chreitung.
`Operator. numRecord sOutPerSecond` < threshold	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 Durchsatz ist zu langsam.

Metrikausdruck	Statistik	Threshold	Beschreibung
`records_lag_max|millisbehindLatest` > threshold	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 application has already fallen behind. Recommended for all applications. Use the `records_lag_max` metric for a Kafka source, or the `millisbehindLatest` for a Kinesis stream source. Rising above this threshold can indicate that the application isn't making expected progress on the input data. For troubleshooting, see Durchsatz ist zu langsam.

Metrikausdruck	Statistik	Threshold	Beschreibung
`lastCheckpointDuration` > threshold	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 `RATE(lastCheckpointSize)` and `RATE(lastCheckpointDuration)` . If the `lastCheckpointDuration` continuously increases, rising above this threshold can indicate that the application isn't making expected

Metrikausdruck	Statistik	Threshold	Beschreibung
			progress on the input data, or that there are problems with application health such as backpressure. For troubleshooting, see Unbegrenztes Zustandswachstum.

Metrikausdruck	Statistik	Threshold	Beschreibung
`lastCheckpointSize` > threshold	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 `RATE(lastCheckpointSize)` and `RATE(lastCheckpointDuration)` . If the `lastCheckpointSize` continuously increases, rising above this threshold can indicate that the application is accumulating state

Metrikausdruck	Statistik	Threshold	Beschreibung
			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 troubleshooting, see Unbegrenztes Zustandswachstum.
`heapMemoryUtilization` > threshold	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 `heapMemoryUtilization` 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 increasing the application parallelism. For more information about increasing resources, see Skalierung.

Metrikausdruck	Statistik	Threshold	Beschreibung
`cpuUtilization` > threshold	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 Skalierung.
`threadsCount` > threshold	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.

Metrikausdruck	Statistik	Threshold	Beschreibung
`(oldGarbageCollectionZeit * 100)/60_000` über einen Zeitraum von 1 `Minute)` > threshold	Maximum	The maximum expected `oldGarbageCollectionZeit` 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.
`RATE (oldGarbageCollectionAnzahl)` > threshold	Maximum	The maximum expected `oldGarbageCollectionAnzahl` 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.
`Operator.currentOutputWatermark - Operator.currentInputWatermark` > threshold	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.

Schreiben von benutzerdefinierten Nachrichten in CloudWatch Protokolle

Sie können benutzerdefinierte Nachrichten in das CloudWatch Protokoll Ihrer Anwendung von Managed Service für Apache Flink schreiben. Sie tun dies, indem Sie die Apache log4j-Bibliothek oder die Simple Logging Facade for Java (SLF4J)-Bibliothek verwenden.

Themen

- Schreiben in CloudWatch Protokolle mit Log4J
- Schreiben in CloudWatch Protokolle mit SLF4J

Schreiben in CloudWatch Protokolle mit Log4J

1. Fügen Sie der `pom.xml`-Datei Ihrer Anwendung die folgenden Abhängigkeiten hinzu:

```
<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. Fügen Sie das Objekt aus der Bibliothek hinzu:

```
import org.apache.logging.log4j.Logger;
```

3. Instanziieren Sie das `Logger`-Objekt und übergeben Sie Ihre Anwendungsklasse:

```
private static final Logger log =
 LogManager.getLogger.getLogger(YourApplicationClass.class);
```

4. Schreiben Sie mit `log.info` in das Protokoll. Eine große Anzahl von Nachrichten wird in das Anwendungsprotokoll geschrieben. Verwenden Sie die `INFO`-Anwendungsprotokollebene, damit Ihre benutzerdefinierten Nachrichten einfacher gefiltert werden können.

```
log.info("This message will be written to the application's CloudWatch log");
```

Die Anwendung schreibt einen Datensatz in das Protokoll mit einer Meldung wie der folgenden:

```
{
  "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"
}
```

Schreiben in CloudWatch Protokolle mit SLF4J

1. Fügen Sie der `pom.xml`-Datei Ihrer Anwendung die folgende Abhängigkeit hinzu:

   ```
   <dependency>
       <groupId>org.slf4j</groupId>
       <artifactId>slf4j-log4j12</artifactId>
       <version>1.7.7</version>
       <scope>runtime</scope>
   </dependency>
   ```

2. Fügen Sie die Objekte aus der Bibliothek hinzu:

   ```
   import org.slf4j.Logger;
   import org.slf4j.LoggerFactory;
   ```

3. Instanziieren Sie das `Logger`-Objekt und übergeben Sie Ihre Anwendungsklasse:

   ```
   private static final Logger log =
    LoggerFactory.getLogger(YourApplicationClass.class);
   ```

4. Schreiben Sie mit `log.info` in das Protokoll. Eine große Anzahl von Nachrichten wird in das Anwendungsprotokoll geschrieben. Verwenden Sie die `INFO`-Anwendungsprotokollebene, damit Ihre benutzerdefinierten Nachrichten einfacher gefiltert werden können.

```
log.info("This message will be written to the application's CloudWatch log");
```

Die Anwendung schreibt einen Datensatz in das Protokoll mit einer Meldung wie der folgenden:

```
{
  "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"
}
```

Protokollieren von Managed Service für Apache Flink API-Aufrufen mit AWS CloudTrail

Managed Service für Apache Flink ist in integriert, einen ServiceAWS CloudTrail, der die Aktionen eines Benutzers, einer Rolle oder eines -AWSServices in Managed Service für Apache Flink aufzeichnet. CloudTrail erfasst alle API-Aufrufe für Managed Service für Apache Flink als Ereignisse. Zu den erfassten Aufrufen gehören Aufrufe von der Konsole Managed Service für Apache Flink und Codeaufträge an Managed Service für Apache-Flink-API-Operationen. Wenn Sie einen Trail erstellen, können Sie die kontinuierliche Bereitstellung von CloudTrail Ereignissen an einen Amazon S3-Bucket aktivieren, einschließlich Ereignissen für Managed Service für Apache Flink. Wenn Sie keinen Trail konfigurieren, können Sie trotzdem die neuesten Ereignisse in der CloudTrail Konsole unter Ereignisverlauf anzeigen. Anhand der von CloudTrailgesammelten Informationen können Sie die an Managed Service für Apache Flink gestellte Anfrage, die IP-Adresse, von der die Anfrage gestellt wurde, den Initiator der Anfrage, den Zeitpunkt der Anfrage und zusätzliche Details bestimmen.

Weitere Informationen zu CloudTrailfinden Sie im AWS CloudTrail -Benutzerhandbuch.

Informationen zu Managed Service für Apache Flink in CloudTrail

CloudTrail wird beim Erstellen des AWS Kontos in Ihrem Konto aktiviert. Wenn eine Aktivität in Managed Service für Apache Flink auftritt, wird diese Aktivität in einem - CloudTrail Ereignis

zusammen mit anderen -AWSServiceereignissen im Ereignisverlauf aufgezeichnet. Sie können die neusten Ereignisse in Ihr AWS-Konto herunterladen und dort suchen und anzeigen. Weitere Informationen finden Sie unter Anzeigen von Ereignissen mit dem CloudTrail Ereignisverlauf.

Erstellen Sie einen Trail für eine fortlaufende Aufzeichnung der Ereignisse in Ihrem AWS-Konto, darunter Ereignisse für Managed Service für Apache Flink. Ein Trail ermöglicht CloudTrail die Bereitstellung von Protokolldateien an einen Amazon S3-Bucket. Wenn Sie einen Trail in der Konsole anlegen, gilt dieser für alle AWS-Regionen. Der Trail protokolliert Ereignisse aus allen Regionen in der AWS-Partition und stellt die Protokolldateien in dem von Ihnen angegebenen Amazon-S3-Bucket bereit. Darüber hinaus können Sie andere -AWSServices konfigurieren, um die in den CloudTrail Protokollen erfassten Ereignisdaten weiter zu analysieren und entsprechend zu agieren. Weitere Informationen finden Sie hier:

- Übersicht zum Erstellen eines Trails
- CloudTrail Unterstützte Services und Integrationen
- Konfigurieren von Amazon SNS-Benachrichtigungen für CloudTrail
- Empfangen von CloudTrail Protokolldateien aus mehreren Regionen und Empfangen von CloudTrail Protokolldateien aus mehreren Konten

Alle Aktionen von Managed Service für Apache Flink werden von protokolliert CloudTrail und sind in der API-Referenz von Managed Service für Apache Flink dokumentiert. Aufrufe der `UpdateApplication` Aktionen `CreateApplication` und erzeugen beispielsweise Einträge in den CloudTrail Protokolldateien.

Jeder Ereignis- oder Protokolleintrag enthält Informationen zu dem Benutzer, der die Anforderung generiert hat. Anhand der Identitätsinformationen zur Benutzeridentität können Sie Folgendes bestimmen:

- Ob die Anfrage mit Stammbenutzer- oder AWS Identity and Access Management (IAM)-Benutzeranmeldeinformationen ausgeführt wurde.
- Ob die Anforderung mit temporären Sicherheitsanmeldeinformationen für eine Rolle oder einen Verbundbenutzer ausgeführt wurde.
- Ob die Anforderung aus einem anderen AWS-Service gesendet wurde

Weitere Informationen finden Sie unter CloudTrail userIdentity-Element.

Wissenswertes zu Managed Service für Apache Flink Protokolldateieinträgen

Ein Trail ist eine Konfiguration, die die Bereitstellung von Ereignissen als Protokolldateien an einen von Ihnen angegebenen Amazon S3-Bucket ermöglicht. CloudTrail Protokolldateien enthalten einen oder mehrere Protokolleinträge. Ein Ereignis stellt eine einzelne Anforderung aus einer beliebigen Quelle dar und enthält Informationen über die angeforderte Aktion, das Datum und die Uhrzeit der Aktion, Anforderungsparameter usw. CloudTrail Protokolldateien sind kein geordnetes Stacktrace der öffentlichen API-Aufrufe und erscheinen daher nicht in einer bestimmten Reihenfolge.

Das folgende Beispiel zeigt einen - CloudTrail Protokolleintrag, der die DescribeApplication Aktionen AddApplicationCloudWatchLoggingOption und demonstriert.

```
{
    "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"
        }
    ]
}
```

Leistung optimieren in Amazon Managed Service für Apache Flink

Dieser Abschnitt beschreibt Techniken zur Überwachung und Verbesserung der Leistung Ihrer Anwendung, die Managed Service für Apache Flink nutzt.

Themen

- Behebung von Leistungsproblemen
- Best Practices zur Leistungsoptimierung
- Leistung überwachen

Behebung von Leistungsproblemen

Dieser Abschnitt enthält eine Liste von Symptomen, anhand derer Sie Leistungsprobleme diagnostizieren und beheben können.

Wenn es sich bei Ihrer Datenquelle um einen Kinesis-Stream handelt, zeigen sich Leistungsprobleme in der Regel als hohe oder ansteigende `millisbehindLatest`-Metrik. Bei anderen Quellen können Sie eine ähnliche Metrik überprüfen, die die Verzögerung beim Lesen aus der Quelle darstellt.

Der Datenpfad

Wenn Sie ein Leistungsproblem mit Ihrer Anwendung untersuchen, sollten Sie den gesamten Pfad berücksichtigen, den Ihre Daten zurücklegen. Die folgenden Anwendungskomponenten können zu Leistungsengpässen und Gegendruck führen, wenn sie nicht richtig konzipiert oder bereitgestellt werden:

- Datenquellen und -ziele: Stellen Sie sicher, dass die externen Ressourcen, mit denen Ihre Anwendung interagiert, entsprechend dem Durchsatz bereitgestellt werden, den Ihre Anwendung erfährt.
- Zustandsdaten: Stellen Sie sicher, dass Ihre Anwendung nicht zu häufig mit dem Zustandsspeicher interagiert.

  Sie können den Serializer optimieren, den Ihre Anwendung verwendet. Der standardmäßige Kryo-Serializer kann jeden serialisierbaren Typ verarbeiten, aber Sie können einen leistungsfähigeren

Serializer verwenden, wenn Ihre Anwendung nur Daten in POJO-Typen speichert. Informationen zu Apache-Flink-Serializern finden Sie unter Datentypen und Serialisierung in der Apache-Flink-Dokumentation.

- Operatoren: Stellen Sie sicher, dass die von Ihren Operatoren implementierte Geschäftslogik nicht zu kompliziert ist oder dass Sie nicht für jeden verarbeiteten Datensatz Ressourcen erstellen oder verwenden. Stellen Sie außerdem sicher, dass Ihre Anwendung nicht zu häufig gleitende oder rollierende Fenster erzeugt.

Lösungen zur Behebung von Leistungsproblemen

Dieser Abschnitt enthält mögliche Lösungen für Leistungsprobleme.

Themen

- CloudWatch-Überwachungsebenen
- CPU-Metrik der Anwendung
- Anwendungsparallelität
- Anwendungsprotokollierung
- Operatorenparallelität
- Anwendungslogik
- Anwendungsspeicher

CloudWatch-Überwachungsebenen

Stellen Sie sicher, dass die CloudWatch-Überwachungsebenen nicht zu ausführlich eingestellt sind.

Die Einstellung `Debug` der Überwachungsprotokollebene generiert eine große Menge an Datenverkehr, was zu Gegendruck führen kann. Sie sollten sie nur verwenden, wenn Sie aktiv Probleme mit der Anwendung untersuchen.

Wenn Ihre Anwendung eine hohe `Parallelism`-Einstellung hat, erzeugt die Verwendung der Ebene `Parallelism` der Überwachungsmetriken ebenfalls eine große Menge an Datenverkehr, was zu Gegendruck führen kann. Verwenden Sie diese Metrikebene nur, wenn `Parallelism` für Ihre Anwendung niedrig ist oder wenn Sie Probleme mit der Anwendung untersuchen.

Weitere Informationen finden Sie unter Anwendungsüberwachungsebenen.

CPU-Metrik der Anwendung

Überprüfen Sie die CPU-Metrik der Anwendung. Wenn diese Metrik über 75 Prozent liegt, können Sie der Anwendung erlauben, mehr Ressourcen für sich selbst zuzuweisen, indem Sie Auto Scaling aktivieren.

Wenn Auto Scaling aktiviert ist, weist die Anwendung mehr Ressourcen zu, wenn die CPU-Auslastung 15 Minuten lang über 75 Prozent liegt. Weitere Informationen zur Skalierung finden Sie im folgenden Abschnitt Richtiges Verwalten der Skalierung und unter Skalierung.

> **Note**
>
> Eine Anwendung wird nur als Reaktion auf die CPU-Auslastung automatisch skaliert. Die Anwendung skaliert nicht automatisch als Reaktion auf andere Systemmetriken, wie z. B. `heapMemoryUtilization`. Wenn Ihre Anwendung häufig andere Metriken verwendet, erhöhen Sie die Parallelität Ihrer Anwendung manuell.

Anwendungsparallelität

Erhöhen Sie die Parallelität der Anwendung. Sie aktualisieren die Parallelität der Anwendung mithilfe des Parameters `ParallelismConfigurationUpdate` der Aktion UpdateApplication.

Die maximale Anzahl von KPUs für eine Anwendung beträgt standardmäßig 64 und kann erhöht werden, indem eine Erhöhung des Grenzwerts angefordert wird.

Es ist wichtig, jedem Operator auch auf der Grundlage seines Workloads Parallelität zuzuweisen, anstatt nur die Anwendungsparallelität allein zu erhöhen. Weitere Informationen finden Sie im Nachfolgenden unter Operatorenparallelität.

Anwendungsprotokollierung

Prüfen Sie, ob die Anwendung für jeden Datensatz, der verarbeitet wird, einen Eintrag protokolliert. Das Schreiben eines Protokolleintrags für jeden Datensatz in Zeiten, in denen die Anwendung einen hohen Durchsatz hat, kann zu schwerwiegenden Engpässen bei der Datenverarbeitung führen. Um diesen Zustand zu überprüfen, fragen Sie Ihre Protokolle auf Protokolleinträge ab, die Ihre Anwendung bei jedem verarbeiteten Datensatz schreibt. Weitere Informationen zum Auslesen von Anwendungsprotokollen finden Sie unter the section called “Analysieren von Protokollen”.

Operatorenparallelität

Stellen Sie sicher, dass der Workload Ihrer Anwendung gleichmäßig auf die Worker-Prozesse verteilt ist.

Informationen zur Optimierung des Workloads der Operatoren Ihrer Anwendung finden Sie unter Operatorenskalierung.

Anwendungslogik

Untersuchen Sie Ihre Anwendungslogik auf ineffiziente oder leistungsschwache Operationen, wie z. B. den Zugriff auf eine externe Abhängigkeit (z. B. eine Datenbank oder einen Webservice), den Zugriff auf den Anwendungszustand usw. Eine externe Abhängigkeit kann auch die Leistung beeinträchtigen, wenn sie nicht performant ist oder nicht zuverlässig zugänglich ist, was dazu führen kann, dass die externe Abhängigkeit `HTTP 500`-Fehler zurückgibt.

Wenn Ihre Anwendung eine externe Abhängigkeit verwendet, um eingehende Daten anzureichern oder anderweitig zu verarbeiten, sollten Sie stattdessen asynchrone E/A verwenden. Weitere Informationen finden Sie unter Async E/A in der Apache-Flink-Dokumentation.

Anwendungsspeicher

Überprüfen Sie Ihre Anwendung auf Ressourcenlecks. Wenn Ihre Anwendung Threads oder Speicher nicht ordnungsgemäß entsorgt, kann es sein, dass die Metriken `millisbehindLatest`, `CheckpointSize` und `CheckpointDuration` steil ansteigen oder allmählich zunehmen. Dieser Zustand kann auch zu Fehlern im Aufgaben- oder Auftragsmanager führen.

Best Practices zur Leistungsoptimierung

In diesem Abschnitt werden besondere Überlegungen zum Entwerfen einer Anwendung im Hinblick auf die Leistung beschrieben.

Richtiges Verwalten der Skalierung

Dieser Abschnitt enthält Informationen zur Verwaltung der Skalierung auf Anwendungs- und Operatorenebene.

In diesem Abschnitt werden folgende Themen behandelt:

- Richtiges Verwalten der Anwendungsskalierung
- Richtiges Verwalten der Operatorenskalierung

Richtiges Verwalten der Anwendungsskalierung

Sie können Auto Scaling verwenden, um unerwartete Spitzen bei der Anwendungsaktivität zu bewältigen. Die KPUs Ihrer Anwendung werden automatisch erhöht, wenn die folgenden Kriterien erfüllt sind:

- Auto Scaling ist für die Anwendung aktiviert.
- Die CPU-Auslastung bleibt 15 Minuten lang über 75 Prozent.

Wenn Auto Scaling aktiviert ist, die CPU-Auslastung aber nicht bei diesem Schwellenwert bleibt, skaliert die Anwendung keine KPUs hoch. Wenn Sie einen Anstieg der CPU-Auslastung feststellen, der diesen Schwellenwert nicht erreicht, oder einen Anstieg bei einer anderen Auslastungsmetrik, z. B. `heapMemoryUtilization`, erhöhen Sie die Skalierung manuell, damit Ihre Anwendung Aktivitätsspitzen bewältigen kann.

> **Note**
>
> Wenn die Anwendung durch Auto Scaling automatisch mehr Ressourcen hinzugefügt hat, gibt die Anwendung die neuen Ressourcen nach einer gewissen Zeit der Inaktivität frei. Das Herunterskalieren von Ressourcen wirkt sich vorübergehend auf die Leistung aus.

Weitere Informationen zur Skalierung finden Sie unter Skalierung.

Richtiges Verwalten der Operatorenskalierung

Sie können die Leistung Ihrer Anwendung verbessern, indem Sie sicherstellen, dass der Workload Ihrer Anwendung gleichmäßig auf die Worker-Prozesse verteilt ist und dass die Operatoren in Ihrer Anwendung über die Systemressourcen verfügen, die sie für einen stabilen und performanten Betrieb benötigen.

Mithilfe der `parallelism`-Einstellung können Sie die Parallelität für jeden Operator im Code Ihrer Anwendung festlegen. Wenn Sie die Parallelität für einen Operator nicht festlegen, wird die Parallelitätseinstellung auf Anwendungsebene verwendet. Operatoren, die die Parallelitätseinstellung auf Anwendungsebene verwenden, können potenziell alle für die Anwendung verfügbaren Systemressourcen nutzen, wodurch die Anwendung instabil wird.

Um die Parallelität für jeden Operator optimal zu bestimmen, sollten die relativen Ressourcenanforderungen des Operators im Vergleich zu anderen Operatoren in der Anwendung

berücksichtigt werden. Stellen Sie für ressourcenintensivere Operatoren eine höhere Einstellung für die Operatorenparallelität ein als für weniger ressourcenintensive Operatoren.

Die gesamte Operatorenparallelität für die Anwendung ist die Summe der Parallelität für alle Operatoren in der Anwendung. Sie optimieren die gesamte Operatorenparallelität für Ihre Anwendung, indem Sie das beste Verhältnis zwischen ihr und der Gesamtzahl der für Ihre Anwendung verfügbaren Aufgabenslots ermitteln. Ein typisches stabiles Verhältnis zwischen der gesamten Operatorenparallelität und den Aufgabenslots ist 4:1, d. h. in der Anwendung steht für jeweils vier verfügbare Operator-Unteraufgaben ein Aufgabenslot zur Verfügung. Eine Anwendung mit ressourcenintensiveren Operatoren benötigt möglicherweise ein Verhältnis von 3:1 oder 2:1, während eine Anwendung mit weniger ressourcenintensiven Operatoren mit einem Verhältnis von 10:1 stabil sein kann.

Sie können das Verhältnis für den verwendeten Operator mittels Laufzeiteigenschaften festlegen, sodass Sie die Parallelität des Operators anpassen können, ohne Ihren Anwendungscode kompilieren und hochladen zu müssen.

Das folgende Beispiel zeigt, wie Sie die Operatorenparallelität als einstellbares Verhältnis zur aktuellen Anwendungsparallelität festlegen:

```
Map<String, Properties> applicationProperties =
 KinesisAnalyticsRuntime.getApplicationProperties();
operatorParallelism =
    StreamExecutionEnvironment.getParallelism() /
    Integer.getInteger(

 applicationProperties.get("OperatorProperties").getProperty("MyOperatorParallelismRatio")
    );
```

Informationen zu Unteraufgaben, Aufgabenslots und anderen Anwendungsressourcen finden Sie unter Anwendungsressourcen.

Verwenden Sie die `Parallelism`-Einstellung und die `KeyBy`-Partitionsmethode, um die Verteilung des Workloads auf die Worker-Prozesse Ihrer Anwendung zu steuern. Weitere Informationen finden Sie in den folgenden Themen in der Apache-Flink-Dokumentation:

- Parallele Ausführung
- DataStream-Transformationen

Überwachen der Nutzung externer Abhängigkeitsressourcen

Wenn bei einem Ziel (wie Kinesis Streams, Kinesis Data Firehose, DynamoDB oder OpenSearch Service) ein Leistungsengpass auftritt, kommt es bei Ihrer Anwendung zu Gegendruck. Stellen Sie sicher, dass Ihre externen Abhängigkeiten für Ihren Anwendungsdurchsatz ordnungsgemäß bereitgestellt wurden.

> **Note**
>
> Fehler in anderen Services können zu Fehlern in Ihrer Anwendung führen. Wenn Sie Fehler in Ihrer Anwendung feststellen, überprüfen Sie die CloudWatch-Protokolle für Ihre Ziel-Services auf Fehler.

Lokales Ausführen Ihrer Apache-Flink-Anwendung

Um Speicherprobleme zu beheben, können Sie Ihre Anwendung in einer lokalen Flink-Installation ausführen. Dadurch erhalten Sie Zugriff auf Debugging-Tools wie Stack-Trace und Heap-Dumps, die nicht verfügbar sind, wenn Sie Ihre Anwendung in Managed Service für Apache Flink ausführen.

Informationen zum Erstellen einer lokalen Flink-Installation finden Sie im Tutorial zur lokalen Einrichtung in der Apache-Flink-Dokumentation.

Leistung überwachen

In diesem Abschnitt werden Tools zur Überwachung der Leistung einer Anwendung beschrieben.

Leistung überwachen mit CloudWatch-Metriken

Mithilfe von CloudWatch-Metriken überwachen Sie die Ressourcennutzung, den Durchsatz, das Checkpointing und die Ausfallzeiten Ihrer Anwendung. Informationen zur Verwendung von CloudWatch-Metriken mit Ihrer Anwendung, die Managed Service für Apache Flink nutzt, finden Sie unter Metriken und Dimensionen in Managed Service für Apache Flink.

Leistung überwachen mit CloudWatch-Protokollen und -Alarmen

Mithilfe von CloudWatch-Protokollen überwachen Sie Fehlerbedingungen, die möglicherweise zu Leistungsproblemen führen könnten.

Fehlerbedingungen werden in Protokolleinträgen angezeigt, wenn der Status des Apache-Flink-Auftrags vom Zustand `RUNNING` zum Zustand `FAILED` wechselt.

Sie verwenden CloudWatch-Alarme, um Benachrichtigungen für Leistungsprobleme zu erstellen, z. B. bei Ressourcennutzung oder Checkpoint-Metriken, die einen sicheren Schwellenwert überschreiten, oder bei unerwarteten Änderungen des Anwendungszustands.

Informationen zum Erstellen von CloudWatch-Alarmen für eine Anwendung, die Managed Service für Apache Flink nutzt, finden Sie unter Alarme.

Managed Service für Apache Flink und Studio-Notebook-Kontingent

Beachten Sie bei der Arbeit mit Amazon Managed Service für Apache Flink das folgende Kontingent:

- Sie können pro Region bis zu 50 Anwendungen, die Managed Service für Apache Flink nutzen, in Ihrem Konto erstellen. Sie können einen Fall erstellen, um weitere Anwendungen über das Formular zur Erhöhung Ihres Servicekontingents anzufordern. Weitere Informationen erhalten Sie im AWS Support Center.

  Eine Liste der Regionen, die Managed Service für Apache Flink unterstützen, finden Sie unter Managed Service für Apache Flink Regionen und Endpunkte.

- Die Anzahl der Kinesis Processing Units (KPU) ist standardmäßig auf 64 begrenzt. Anweisungen zum Anfordern einer Erhöhung dieses Kontingents finden Sie unter So fordern Sie eine Kontingenterhöhung an in Service Quotas. Stellen Sie sicher, dass Sie das Anwendungspräfix angeben, auf das das neue KPU-Limit angewendet werden muss.

  Bei Managed Service für Apache Flink werden Ihrem AWS-Konto die zugewiesenen Ressourcen in Rechnung gestellt und nicht die Ressourcen, die Ihre Anwendung verwendet. Sie zahlen einen Stundenpreis auf der Basis der maximalen Zahl von KPUs, die zum Ausführen Ihrer Anwendung für die Stream-Verarbeitung genutzt werden. Eine einzelne KPU bietet Ihnen 1 vCPU und 4 GB Arbeitsspeicher. Für jede KPU stellt der Dienst außerdem 50 GB Speicher für laufende Anwendungen bereit.

- Sie können pro Anwendung bis zu 1.000 Managed Service für Apache Flink Snapshots erstellen.

- Sie können bis zu 50 Tags pro Anwendung zuweisen.

- Die maximale Größe für eine Anwendungs-JAR-Datei beträgt 512 MB. Wenn Sie dieses Kontingent überschreiten, kann Ihre Anwendung nicht gestartet werden.

Für Studio-Notebooks gelten die folgenden Kontingente. Um ein höheres Kontingent anzufordern, erstellen Sie einen Support-Fall.

- `websocketMessageSize` = 5 MB
- `noteSize` = 5 MB
- `noteCount` = 1000
- `Max cumulative UDF size` = 100 MB
- `Max cumulative dependency jar size` = 300 MB

Wartung von Managed Service für Apache Flink

Managed Service für Apache Flink aktualisiert Ihre Anwendungen regelmäßig mit Betriebssystem- und Container-Image-Sicherheitsupdates, um die Einhaltung der Vorschriften zu gewährleisten und die AWS-Sicherheitsziele zu erreichen. In der folgenden Tabelle ist das Standardzeitfenster aufgeführt, in dem Managed Service für Apache Flink diese Art von Wartung durchführt. Die Wartung Ihrer Anwendung kann jederzeit während des Zeitfensters erfolgen, das Ihrer Region entspricht. Während dieses Wartungsvorgangs kann es bei Ihrer Anwendung zu einer Ausfallzeit von 10 bis 30 Sekunden kommen. Die tatsächliche Dauer der Ausfallzeit hängt jedoch vom Anwendungszustand ab. Informationen darüber, wie Sie die Auswirkungen dieser Ausfallzeit minimieren können, finden Sie unter the section called “Fehlertoleranz: Prüfpunkte und Savepoints”.

Um das Zeitfenster zu ändern, in dem Managed Service für Apache Flink Wartungsarbeiten an Ihrer Anwendung durchführt, verwenden Sie die UpdateApplicationMaintenanceConfiguration-API.

Region	Wartungszeitfenster
AWS GovCloud (USA-West)	06:00 - 14:00 UTC
AWS GovCloud (USA-Ost)	03:00 - 11:00 UTC
USA Ost (Nord-Virginia)	03:00 - 11:00 UTC
USA Ost (Ohio)	03:00 - 11:00 UTC
USA West (Nordkalifornien)	06:00 - 14:00 UTC
USA West (Oregon)	06:00 - 14:00 UTC
Asien-Pazifik (Hongkong)	13:00 - 21:00 UTC
Asien-Pazifik (Mumbai)	16:30 - 00:30 UTC
Asien-Pazifik (Hyderabad)	16:30 - 00:30 UTC
Asien-Pazifik (Seoul)	13:00 - 21:00 UTC
Asien-Pazifik (Singapur)	14:00 - 22:00 UTC

Region	Wartungszeitfenster
Asien-Pazifik (Sydney)	12:00 - 20:00 UTC
Asien-Pazifik (Jakarta)	15:00 - 23:00 UTC
Asien-Pazifik (Tokio)	13:00 - 21:00 UTC
Kanada (Zentral)	03:00 - 11:00 UTC
China (Peking)	13:00 - 21:00 UTC
China (Ningxia)	13:00 - 21:00 UTC
Europa (Frankfurt)	06:00 - 14:00 UTC
Europa (Zürich)	20:00 - 04:00 UTC
Europa (Irland)	22:00 - 06:00 UTC
Europa (London)	22:00 - 06:00 UTC
Europa (Stockholm)	23:00 - 07:00 UTC
Europa (Mailand)	21:00 - 05:00 UTC
Europa (Spanien)	21:00 - 05:00 UTC
Afrika (Kapstadt)	20:00 - 04:00 UTC
Europa (Irland)	22:00 - 06:00 UTC
Europa (London)	23:00 - 07:00 UTC
Europa (Paris)	23:00 - 07:00 UTC
Europa (Stockholm)	23:00 - 07:00 UTC
Naher Osten (Bahrain)	13:00 - 21:00 UTC
Naher Osten (VAE)	18:00 - 02:00 UTC

Region	Wartungszeitfenster
Südamerika (São Paulo)	19:00 - 03:00 UTC
Israel (Tel Aviv)	20:00 - 04:00 UTC

Festlegen einer UUID für alle Operatoren

Wenn Managed Service für Apache Flink einen Flink-Auftrag für eine Anwendung mit einem Snapshot startet, kann der Flink-Auftrag aufgrund bestimmter ggf. Probleme nicht gestartet werden. Eines davon ist die Nichtübereinstimmung der Operator-ID. Flink erwartet explizite, konsistente Operator-IDs für Flink-Auftragsdiagramm-Operatoren. Wenn nicht explizit gesetzt, generiert Flink automatisch eine ID für die Operatoren. Das liegt daran, dass Flink diese Operator-IDs verwendet, um die Operatoren in einem Auftragsdiagramm eindeutig zu identifizieren, und sie verwendet, um den Zustand jedes Operators in einem Savepoint zu speichern.

Das Problem der Nichtübereinstimmung der Operator-ID tritt auf, wenn Flink keine 1:1-Zuordnung zwischen den Operator-IDs eines Auftragsdiagramms und den in einem Savepoint definierten Operator-IDs findet. Dies passiert, wenn keine expliziten konsistenten Operator-IDs gesetzt sind und Flink automatisch Operator-IDs generiert, die möglicherweise nicht bei jeder Auftragsdiagramm-Erstellung konsistent sind. Die Wahrscheinlichkeit, dass Anwendungen bei Wartungsarbeiten auf dieses Problem stoßen, ist hoch. Um dies zu vermeiden, empfehlen wir Kunden, die UUID für alle Operatoren im Flink-Code festzulegen. Weitere Informationen finden Sie im Abschnitt Festlegen einer UUID für alle Operatoren unter Produktionsbereitschaft.

Produktionsbereitschaft

Dies ist eine Sammlung wichtiger Aspekte der Ausführung von Produktionsanwendungen auf Managed Service für Apache Flink. Es handelt sich nicht um eine vollständige Liste, sondern um das absolute Minimum dessen, worauf Sie achten sollten, bevor Sie eine Anwendung in Produktion nehmen.

Anwendungen für Lasttests

Einige Probleme mit Anwendungen treten nur bei hoher Auslastung auf. Wir haben Fälle gesehen, in denen Anwendungen funktionstüchtig zu sein schienen und ein Betriebsereignis die Last der Anwendung erheblich erhöhte. Dies kann völlig unabhängig von der Anwendung selbst geschehen: Wenn die Datenquelle oder die Datensenke für ein paar Stunden nicht verfügbar ist, kann die Flink-Anwendung keine Fortschritte machen. Sobald dieses Problem behoben ist, hat sich ein Rückstand an unverarbeiteten Daten angesammelt, der die verfügbaren Ressourcen vollständig erschöpfen kann. Die Last kann dann Fehler oder Leistungsprobleme verstärken, die zuvor nicht aufgetreten sind.

Es ist daher wichtig, angemessene Lasttests für Produktionsanwendungen durchzuführen. Zu den Fragen, die bei diesen Lasttests beantwortet werden sollten, gehören:

- Ist die Anwendung bei anhaltend hoher Last stabil?
- Kann die Anwendung bei Spitzenlast immer noch einen Savepoint verwenden?
- Wie lange braucht die Verarbeitung eines Rückstands von 1 Stunde? Und wie lange von 24 Stunden (abhängig von der maximalen Aufbewahrung der Daten im Stream)?
- Steigt der Durchsatz der Anwendung, wenn die Anwendung skaliert wird?

Beim Verbrauch aus einem Datenstrom können diese Szenarien simuliert werden, indem sie für einige Zeit in den Stream übertragen werden. Starten Sie dann die Anwendung und lassen Sie sie Daten vom Beginn der Zeit an verbrauchen, verwenden Sie z. B. bei einem Kinesis Data Stream die Startposition `TRIM_HORIZON`.

Maximale Parallelität

Die maximale Parallelität definiert die maximale Parallelität, auf die eine zustandsbehaftete Anwendung skalieren kann. Dies wird definiert, wenn der Zustand erstmalig erstellt wird, und es gibt

keine Möglichkeit, den Operator über dieses Maximum hinaus zu skalieren, ohne den Zustand zu verwerfen.

Die maximale Parallelität wird festgelegt, wenn der Zustand erstmalig erstellt wird.

Standardmäßig ist Maximale Parallelität festgelegt auf:

- 128, wenn Parallelität <= 128
- `MIN(nextPowerOfTwo(parallelism + (parallelism / 2)), 2^15)`: wenn Parallelität > 128

Wenn Sie planen, Ihre Anwendung auf > 128 Parallelität zu skalieren, sollten Sie die maximale Parallelität explizit definieren.

Die maximale Parallelität kann auf Anwendungsebene mit `env.setMaxParallelism(x)` oder mit einem einzelnen Operator definiert werden. Sofern nicht anders angegeben, erben alle Operatoren die Maximale Parallelität der Anwendung.

Weitere Informationen finden Sie unter Festlegen einer expliziten Maximalen Parallelität in der Flink-Dokumentation.

Festlegen einer UUID für alle Operatoren

Eine UUID wird bei dem Vorgang verwendet, bei dem Flink einen Savepoint auf einen einzelnen Operator abbildet. Wenn Sie für jeden Operator eine bestimmte UUID festlegen, erhalten Sie eine stabile Zuordnung für den wiederherzustellenden Savepoint-Prozess.

```
.map(...).uid("my-map-function")
```

Weitere Informationen finden Sie unter Checkliste zur Produktionsbereitschaft.

Bewährte Methoden für Managed Service für Apache Flink

Dieser Abschnitt enthält Informationen und Empfehlungen für die Entwicklung eines stabilen, performanten Managed Service für Apache Flink-Anwendungen.

Themen

- Fehlertoleranz: Prüfpunkte und Savepoints
- Nicht unterstützte Konnektor-Versionen
- Leistung und Parallelität
- Parallelität pro Operator festlegen
- Protokollierung
- Codierung
- Verwalten von Anmeldeinformationen.
- Lesen aus Quellen mit wenigen Shards/Partitionen
- Aktualisierungsintervall für Studio-Notebooks
- Optimale Leistung des Studio-Notebooks
- Wie sich Strategien mit Wasserzeichen und ungenutzte Shards auf Zeitfenster auswirken
- Festlegen einer UUID für alle Operatoren
- Dem ServiceResourceTransformer Maven Shade Plugin hinzufügen

Fehlertoleranz: Prüfpunkte und Savepoints

Verwenden Sie Prüfpunkte und Savepoints, um Fehlertoleranz in Ihrer Anwendung von Managed Service für Apache Flink zu implementieren. Berücksichtigen Sie bei Entwicklung und Wartung Ihrer Anwendung Folgendes:

- Sie sollten Prüfupunktprüfung für Ihre Anwendung aktiviert lassen. Prüfpunktprüfung bietet Fehlertoleranz für Ihre Anwendung bei geplanten Wartungsarbeiten sowie bei unerwarteten Ausfällen aufgrund von Serviceproblemen, Fehlern bei der Anwendungsabhängigkeit und anderen Problemen. Weitere Informationen zur geplanten Wartung finden Sie unter Wartung.
- Setzen Sie ApplicationSnapshotConfiguration::SnapshotsEnabled `false` während der Anwendungsentwicklung oder Fehlerbehebung auf . Bei jedem Anwendungsstopp wird ein Snapshot erstellt. Dies kann zu Problemen führen, wenn sich die Anwendung in einem fehlerhaften

Zustand befindet oder nicht leistungsfähig ist. Setzen Sie `SnapshotsEnabled` auf `true`, wenn die Anwendung in Produktion und stabil ist.

> **Note**
>
> Wir empfehlen, dass Ihre Anwendung mehrmals täglich einen Snapshot erstellt, um einen ordnungsgemäßen Neustart mit den korrekten Statusdaten zu gewährleisten. Die richtige Häufigkeit für Ihre Snapshots hängt von der Geschäftslogik Ihrer Anwendung ab. Durch häufiges Erstellen von Snapshots können Sie neuere Daten wiederherstellen. Dies erhöht jedoch die Kosten und erfordert mehr Systemressourcen.

Informationen zur Überwachung von Anwendungsausfällen finden Sie unter Metriken und Dimensionen in Managed Service für Apache Flink.

Weitere Informationen zur Implementierung der Fehlertoleranz finden Sie unter Fehlertoleranz.

Nicht unterstützte Konnektor-Versionen

Managed Service für Apache Flink Version 1.15 verhindert automatisch den Start oder die Aktualisierung von Anwendungen, wenn sie nicht unterstützte Kinesis-Konnektor-Versionen (in Anwendungs-JARs gebündelt) verwenden. Stellen Sie beim Upgrade auf Managed Service für Apache Flink Version 1.15 sicher, dass Sie den neuesten Kinesis-Konnektor verwende. Dies ist jede Version, die Version 1.15.2 entspricht oder neuer ist. Alle anderen Versionen werden von Managed Service fü Apache Flink nicht unterstützt, da sie zu Konsistenzproblemen oder Ausfällen führen können, da die Funktion Mit Savepoint beenden verhindert, dass saubere Stop-/Aktualisierungsvorgänge durchgeführt werden.

Leistung und Parallelität

Ihre Anwendung kann so skaliert werden, dass sie jedes Durchsatzniveau erreicht, indem Sie die Parallelität Ihrer Anwendung optimieren und Leistungsprobleme vermeiden. Berücksichtigen Sie bei Entwicklung und Wartung Ihrer Anwendung Folgendes:

- Stellen Sie sicher, dass alle Ihre Anwendungsquellen und -senken ausreichend bereitgestellt sind und nicht gedrosselt werden. Wenn es sich bei den Quellen und Senken um andere -AWSServices handelt, überwachen Sie diese Services mit CloudWatch.

- Prüfen Sie bei Anwendungen mit sehr hoher Parallelität, ob die hohen Parallelitätsebenen auf alle Operatoren in der Anwendung angewendet werden. Standardmäßig wendet Apache Flink dieselbe Anwendungsparallelität für alle Operatoren im Anwendungsdiagramm an. Dies kann entweder zu Problemen bei der Bereitstellung auf Quellen oder Senken oder zu Engpässen bei der Datenverarbeitung durch die Operatoren führen. Sie können die Parallelität der einzelnen Operatoren im Code mit setParallelism ändern.
- Machen Sie sich mit der Bedeutung der Parallelitätseinstellungen für die Operatoren in Ihrer Anwendung vertraut. Wenn Sie die Parallelität für einen Operator ändern, können Sie die Anwendung möglicherweise nicht aus einem Snapshot wiederherstellen, der erstellt wurde, als der Operator eine Parallelität hatte, die mit den aktuellen Einstellungen nicht kompatibel ist. Weitere Informationen zum Einstellen der Operatorparallelität finden Sie unter Explizite Festlegung der maximalen Parallelität für Operatoren.

Weitere Informationen über die Implementierung von Skalierung finden Sie unter Skalierung.

Parallelität pro Operator festlegen

Standardmäßig ist die Parallelität für alle Operatoren auf Anwendungsebene festgelegt. Sie können die Parallelität eines einzelnen Operators mithilfe der DataStream API mit überschreiben `.setParallelism(x)`. Sie können eine Operatorparallelität auf eine beliebige Parallelität festlegen, die gleich oder niedriger als die Anwendungsparallelität ist.

Wenn möglich, definieren Sie die Operatorparallelität als Funktion der Anwendungsparallelität. Auf diese Weise variiert die Operatorparallelität mit der Anwendungsparallelität. Wenn Sie beispielsweise automatische Skalierung verwenden, variieren alle Operatoren ihre Parallelität im gleichen Verhältnis:

```
int appParallelism = env.getParallelism();
...
...ops.setParalleism(appParallelism/2);
```

In einigen Fällen möchten Sie möglicherweise die Operatorparallelität auf eine Konstante setzen. Stellen Sie beispielsweise die Parallelität einer Kinesis Stream-Quelle auf die Anzahl der Shards ein. In diesen Fällen sollten Sie erwägen, die Operator-Parallelität als Anwendungskonfigurationsparameter zu übergeben, um sie zu ändern, ohne den Code zu ändern, wenn Sie beispielsweise den Quellstream resharden müssen.

Protokollierung

Sie können die Leistung und Fehlerbedingungen Ihrer Anwendung mithilfe von - CloudWatch Protokollen überwachen. Berücksichtigen Sie bei der Konfiguration der Protokollierung für Ihre Anwendung Folgendes:

- Aktivieren Sie die CloudWatch Protokollierung für die Anwendung, damit alle Laufzeitprobleme debuggt werden können.
- Erstellen Sie nicht für jeden Datensatz, der in der Anwendung verarbeitet wird, einen Protokolleintrag. Dies führt zu schwerwiegenden Engpässen bei der Verarbeitung und kann zu Gegendruck bei der Datenverarbeitung führen.
- Erstellen Sie CloudWatch Alarme, um Sie zu benachrichtigen, wenn Ihre Anwendung nicht ordnungsgemäß ausgeführt wird. Weitere Informationen finden Sie unter Alarme.

Weitere Informationen über die Implementierung von Protokollierung finden Sie unter Protokollieren und Überwachen.

Codierung

Sie können Ihre Anwendung leistungsfähig und stabil machen, indem Sie empfohlene Programmierpraktiken anwenden. Berücksichtigen Sie beim Schreiben von Anwendungscode Folgendes:

- Verwenden Sie `system.exit()` in Ihrem Anwendungscode nicht, weder in der `main`-Methode Ihrer Anwendung noch in benutzerdefinierten Funktionen. Wenn Sie Ihre Anwendung aus dem Code heraus beenden möchten, lösen Sie eine von `Exception` oder `RuntimeException` abgeleitete Ausnahme aus, die eine Meldung darüber enthält, was bei der Anwendung schiefgelaufen ist.

  Beachten Sie Folgendes darüber, wie der Service mit dieser Ausnahme umgeht:

  - Wenn die Ausnahme von der `main`-Methode Ihrer Anwendung ausgelöst wird, wird sie vom Service beim Übergang der Anwendung in den `RUNNING`-Status in eine `ProgramInvocationException` eingeschlossen, und der Job-Manager kann den Job nicht weiterleiten.

  - Wenn die Ausnahme von einer benutzerdefinierten Funktion ausgelöst wird, schlägt der Job-Manager den Job fehl und startet ihn neu. Die Details der Ausnahme werden in das Ausnahmeprotokoll geschrieben.
- Erwägen Sie, die JAR-Datei Ihrer Anwendung und die darin enthaltenen Abhängigkeiten zu schattieren. Das Schattieren wird empfohlen, wenn es potenzielle Konflikte bei den Paketnamen zwischen Ihrer Anwendung und der Apache Flink-Laufzeit gibt. Wenn ein Konflikt auftritt, können Ihre Anwendungsprotokolle eine Ausnahme des Typs `java.util.concurrent.ExecutionException` enthalten. Weitere Informationen zum Schattieren Ihrer Anwendungs-JAR-Datei finden Sie unter Apache Maven Shade-Plugin.

Verwalten von Anmeldeinformationen.

Sie sollten keine langfristigen Anmeldeinformationen in Produktionsanwendungen (oder andere Anwendungen) integrieren. Langfristige Anmeldeinformationen werden wahrscheinlich in ein Versionskontrollsystem eingecheckt und können leicht verloren gehen. Stattdessen können Sie der Anwendung Managed Service für Apache Flink eine Rolle zuordnen und dieser Rolle Berechtigungen gewähren. Die laufende Flink-Anwendung kann dann temporäre Anmeldeinformationen mit den entsprechenden Rechten aus der Umgebung abrufen. Falls eine Authentifizierung für einen Service erforderlich ist, der nicht nativ in IAM integriert ist, z. B. für eine Datenbank, die einen Benutzernamen und ein Passwort für die Authentifizierung benötigt, sollten Sie erwägen, Secrets in AWS-Secrets Manager zu speichern.

Viele AWS-native Services unterstützen die Authentifizierung:

- Kinesis Data Streams – ProcessTaxiStream.java
- Amazon MSK – https://github.com/aws/aws-msk-iam-auth/#using-the-amazon-msk-library-for-iam-authentication
- Amazon Elasticsearch Service – AmazonElasticsearchSink.java
- Amazon S3 – funktioniert direkt mit Managed Service für Apache Flink

Lesen aus Quellen mit wenigen Shards/Partitionen

Beim Lesen aus Apache Kafka oder einem Kinesis Data Stream kann es zu einer Diskrepanz zwischen der Parallelität des Streams (d. h. der Anzahl der Partitionen für Kafka und der Anzahl der Shards für Kinesis) und der Parallelität der Anwendung kommen. Bei einem naiven Design kann die

Parallelität einer Anwendung nicht über die Parallelität eines Streams skalieren: Jede Unteraufgabe eines Quelloperators kann nur aus 1 oder mehreren Shards/Partitionen lesen. Das bedeutet für einen Stream mit nur 2 Shards und eine Anwendung mit einer Parallelität von 8, dass nur zwei Unteraufgaben den Stream tatsächlich verbrauchen und 6 Unteraufgaben inaktiv bleiben. Dies kann den Durchsatz der Anwendung erheblich einschränken, insbesondere wenn die Deserialisierung teuer ist und von der Quelle durchgeführt wird (was die Standardeinstellung ist).

Um diesen Effekt zu mildern, können Sie den Stream entweder skalieren. Dies ist jedoch möglicherweise nicht immer wünschenswert oder möglich. Alternativ können Sie die Quelle so umstrukturieren, dass sie keine Serialisierung durchführt und nur die `byte[]` weitergibt. Anschließend können Sie die Daten umverteilen, um sie gleichmäßig auf alle Aufgaben zu verteilen, und die Daten dann dort deserialisieren. Auf diese Weise können Sie alle Unteraufgaben für die Deserialisierung nutzen und dieser potenziell teure Vorgang ist nicht mehr an die Anzahl der Shards/Partitionen des Streams gebunden.

Aktualisierungsintervall für Studio-Notebooks

Wenn Sie das Aktualisierungsintervall für die Absatzergebnisse ändern, setzen Sie es auf einen Wert, der mindestens 1000 Millisekunden beträgt.

Optimale Leistung des Studio-Notebooks

Wir haben mit der folgenden Aussage getestet und die beste Leistung erhalten, wenn `events-per-second` mit `number-of-keys` multipliziert unter 25.000.000 ergab. Das war bei `events-per-second` weniger als 150.000.

```
SELECT key, sum(value) FROM key-values GROUP BY key
```

Wie sich Strategien mit Wasserzeichen und ungenutzte Shards auf Zeitfenster auswirken

Beim Lesen von Ereignissen aus Apache Kafka und Kinesis Data Streams kann die Quelle die Ereigniszeit basierend auf den Attributen des Streams festlegen. Im Fall von Kinesis entspricht die Ereigniszeit der ungefähren Ankunftszeit von Ereignissen. Die Festlegung der Ereigniszeit an der Quelle für Ereignisse reicht jedoch nicht aus, damit eine Flink-Anwendung die Ereigniszeit verwenden

kann. Die Quelle muss außerdem Wasserzeichen erzeugen, die Informationen über die Ereigniszeit von der Quelle an alle anderen Operatoren weitergeben. Die Flink-Dokumentation bietet einen guten Überblick darüber, wie dieser Prozess funktioniert.

Standardmäßig ist der Zeitstempel eines aus Kinesis gelesenen Ereignisses auf die ungefähre Ankunftszeit gesetzt, die von Kinesis bestimmt wird. Eine zusätzliche Voraussetzung dafür, dass die Ereigniszeit in der Anwendung funktioniert, ist eine Wasserzeichen-Strategie.

```
WatermarkStrategy<String> s = WatermarkStrategy
    .<String>forMonotonousTimestamps()
    .withIdleness(Duration.ofSeconds(...));
```

Die Wasserzeichenstrategie wird dann auf einen `DataStream` mit der Methode `assignTimestampsAndWatermarks` angewendet. Es gibt einige nützliche integrierte Strategien:

- `forMonotonousTimestamps()` verwendet einfach die Uhrzeit des Ereignisses (ungefähre Ankunftszeit) und gibt in regelmäßigen Abständen den Maximalwert als Wasserzeichen aus (für jede spezifische Unteraufgabe)
- `forBoundedOutOfOrderness(Duration.ofSeconds(...))` ähnelt der vorherigen Strategie, verwendet jedoch die Ereigniszeit – Dauer für die Generierung von Wasserzeichen.

Das funktioniert, aber es gibt ein paar Vorbehalte, die Sie beachten sollten. Wasserzeichen werden auf Unteraufgabenebene generiert und fließen durch das Operatordiagramm.

Aus der Flink-Dokumentation:

Jede parallel Unteraufgabe einer Quellfunktion generiert ihre Wasserzeichen normalerweise unabhängig. Diese Wasserzeichen definieren die Ereigniszeit an dieser bestimmten parallelen Quelle.

Während die Wasserzeichen durch das Streaming-Programm fließen, verschieben sie die Ereigniszeit bei den Operatoren, wo sie ankommen. Immer wenn ein Betreiber seine Ereigniszeit vorverlegt, generiert er nachgelagert ein neues Wasserzeichen für seine nachfolgenden Operatoren.

Manche Operatoren verbrauchen mehrere Eingabestreams; zum Beispiel eine Union oder Operatoren, die einer keyBy(...)- oder partition(...)-Funktion folgen. Die aktuelle Ereigniszeit eines solchen Operators ist das Minimum der Ereigniszeiten seiner Eingabestreams. Wenn seine Eingabestreams ihre Ereigniszeiten aktualisieren, tut dies auch der Operator.

Das heißt, wenn eine Quell-Unteraufgabe von einem inaktiven Shard aus konsumiert, erhalten nachgeschaltete Operatoren keine neuen Wasserzeichen von dieser Unteraufgabe, sodass die Verarbeitung für alle nachgeschalteten Operatoren, die Zeitfenster verwenden, zum Stillstand kommt. Um dies zu vermeiden, können Kunden die `withIdleness`-Option zur Wasserzeichenstrategie hinzufügen. Bei dieser Option schließt ein Operator bei der Berechnung der Ereigniszeit des Operators die Wasserzeichen aus ungenutzten Upsteam-Unteraufgaben aus. Unteraufgaben im Leerlauf blockieren somit nicht mehr die Weiterleitung der Ereigniszeit bei nachgeschalteten Operatoren.

Die Idleness-Option mit den integrierten Wasserzeichen-Strategien verschiebt die Ereigniszeit jedoch nicht nach vorn, wenn keine Unteraufgabe ein Ereignis liest, d. h. es keine Ereignisse im Stream gibt. Dies wird besonders bei Testfällen sichtbar, in denen eine begrenzte Menge von Ereignissen aus dem Stream gelesen wird. Da die Ereigniszeit nach dem Lesen des letzten Ereignisses nicht vorverlegt wird, wird das letzte Fenster (das das letzte Ereignis enthält) niemals geschlossen.

Übersicht

- Mit `withIdleness`-Einstellung werden keine neuen Wasserzeichen generiert, falls ein Shard inaktiv ist. Sie schließt lediglich das letzte Wasserzeichen aus, das von Unteraufgaben im Leerlauf gesendet wurde, bei der Berechnung des Mindestwasserzeichens in nachgeschalteten Operatoren
- Bei den integrierten Wasserzeichen-Strategien wird das zuletzt geöffnete Fenster nie geschlossen (es sei denn, es werden neue Ereignisse gesendet, die das Wasserzeichen weiterleiten, aber dadurch wird ein neues Fenster erstellt, das dann geöffnet bleibt)
- Selbst wenn die Zeit durch den Kinesis-Stream festgelegt wird, können spät eintreffende Ereignisse immer noch auftreten, wenn ein Shard schneller verbraucht wird als andere (z. B. während der App-Initialisierung oder bei der Verwendung von `TRIM_HORIZON`, wo alle vorhandenen Shards parallel konsumiert werden, wobei ihre Eltern/Kind-Beziehung ignoriert wird)
- Die `withIdleness`-Einstellungen der Wasserzeichenstrategie scheinen die quellspezifischen Kinesis-Einstellungen für inaktive Shards als veraltet zu markieren `(ConsumerConfigConstants.SHARD_IDLE_INTERVAL_MILLIS`

Beispiel

Die folgende Anwendung liest aus einem Stream und erstellt Sitzungsfenster auf der Grundlage der Ereigniszeit.

```
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);
            }
        }
    });
```

Im folgenden Beispiel werden 8 Ereignisse in einen Stream mit 16 Shards geschrieben (die ersten 2 und das letzte Ereignis landen zufällig im selben Shard).

```
$ 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
```

Diese Eingabe sollte zu 5 Sitzungsfenstern führen: Ereignis 1,2,3; Ereignis 4,5; Ereignis 6; Ereignis 7; Ereignis 8. Das Programm liefert jedoch nur die ersten 4 Fenster.

```
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:
```

```
 4962789433850315183450815791266608495756527394990168485O,}}'}, 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:
 4962789433857005407010374978209069211238321903441962614O,}}'}, 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:
 4962789433850315183450815791266608495756527394990168485O,}}'} 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:
 4962789433845855034411109666638301352101997722688972398O,}}'}, 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:
 4962789433845855034411109666638301352101997722688972398O,}}'} 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:
 850705917302346158658436518579420528O4,EndingHashKey:
 106338239662793269832304564822427566079},SequenceNumberRange: {StartingSequenceNumber:
 4962789433843624959891256604324147780274732886538374355O,}}'}, 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:
 638029437976759618993827388934565396480,EndingHashKey:
 850705917302346158658436518579420528O3},SequenceNumberRange: {StartingSequenceNumber:
```

```
 496278943384139488537140354200999420844746805038777631 22,}}'}, 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:
 496278943386815577960964028977983707037464608419495283 06,}}'}, 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:
 297747071055821155530452781502797185024,EndingHashKey:
 319014718988379809496913694467282698239},SequenceNumberRange: {StartingSequenceNumber:
 496278943386592570508978722746568349854738124804435478 74,}}'}, 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:
 850705917302346158658436518579420528 64,EndingHashKey:
 106338239662793269832304564822427566079},SequenceNumberRange: {StartingSequenceNumber:
 496278943384362495989125660432414778027473288653837435 54,}}'} 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:
 638029437976759618993827387893456539648,EndingHashKey:
 850705917302346158658436518579420528 63},SequenceNumberRange: {StartingSequenceNumber:
 496278943384139488537140354200999420844746805038777631 22,}}'} 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:
 212676479325586539664609129644855132 16,EndingHashKey:
 425352958651173079329218259289710264 31},SequenceNumberRange: {StartingSequenceNumber:
 496278943386934736331697417381687064792938378086580225 8,}}'}, 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:
 191408831393027885698148216680369618944,EndingHashKey:
 212676479325586539664609129644855132159},SequenceNumberRange: {StartingSequenceNumber:
```

```
 496278943385477533249052191589491563941105706729136457l4,}}'}, 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:
 212676479325586539664609129644855132l5},SequenceNumberRange: {StartingSequenceNumber:
 496278943383470466181184435506753349296567354193598218l6,}}'}, 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:
 255211775190703847597530955573826158592,EndingHashKey:
 276479423123262501563991868538311671807},SequenceNumberRange: {StartingSequenceNumber:
 496278943386146555605008110283737635489285157574315870l0,}}'}, 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:
 170141183460469231731687303715884105728,EndingHashKey:
 191408831393027885698148216680369618943},SequenceNumberRange: {StartingSequenceNumber:
 496278943385254525797066885358076206758379223114076652l2,}}'}, 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:
 212676479325586539664609129644855132l6,EndingHashKey:
 425352958651173079329218259289710264l1},SequenceNumberRange: {StartingSequenceNumber:
 496278943383693473633169741738168706479293837808658022581,}}'} 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:
 233944127258145193631070042609340645376,EndingHashKey:
 255211775190703847597530955573826158591},SequenceNumberRange: {StartingSequenceNumber:
 496278943385923548153022804052322278306558673959256065781,}}'}, 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:
 212676479325586539664609129644855132l5},SequenceNumberRange: {StartingSequenceNumber:
 496278943383470466181184435506753349296567354193598218l6,}}'} 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:
 27647942312326250156399186853831167180 8,EndingHashKey:
 29774707105582115553045278150279718502 3},SequenceNumberRange: {StartingSequenceNumber:
 49627894338636956305699341651515299267201164118937567442,}}'}, 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:
 49627894338391648108515504796958406366202032142371782690,}}'} 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:
 19140883139302788569814821668036961894 4,EndingHashKey:
 21267647932558653966460912964485513215 9},SequenceNumberRange: {StartingSequenceNumber:
 49627894338547753324905219158949156394110570672913645714,}}'} 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:
 21267647932558653966460912964485513216 0,EndingHashKey:
 23394412725814519363107004260934064537 5},SequenceNumberRange: {StartingSequenceNumber:
 49627894338570054070103749782090692112383219034419626146,}}'} 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:
 31901471898837980949691369446728269824 0,EndingHashKey:
```

```
 340282366920938463463374607431768211455},SequenceNumberRange: {StartingSequenceNumber:
 49627894338681557796096402897798370703746460841949528306,}}'} 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:
 49627894338659257050897872274656834985473812480443547874,}}'} 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:
 49627894338525452579706688535807620675837922311407665282,}}'} 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:
 49627894338636956305699341651515299267201164118937567442,}}'} 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:
 49627894338614655560500811028373763548928515757431587010,}}'} 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:
```

```
 4962789433859235481530228040523222783065586739592560657 8,}}'} 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
```

Die Ausgabe zeigt nur 4 Fenster (es fehlt das letzte Fenster, das Ereignis 8 enthält). Das liegt an der Ereigniszeit und der Wasserzeichen-Strategie. Das letzte Fenster kann nicht geschlossen werden, da bei den vorgefertigten Wasserzeichen-Strategie die Zeit nie über den Zeitpunkt des letzten Ereignisses hinausgeht, das aus dem Stream gelesen wurde. Damit das Fenster geschlossen werden kann, muss die Zeit jedoch um mehr als 10 Sekunden über das letzte Ereignis hinausgehen. In diesem Fall ist das letzte Wasserzeichen 2022-03-23T10:21:27.170Z, aber damit das Sitzungsfenster geschlossen werden kann, ist ein Wasserzeichen 10s und 1ms später erforderlich.

Wenn die `withIdleness`-Option aus der Wasserzeichenstrategie entfernt wird, wird kein Sitzungsfenster jemals geschlossen, da das „globale Wasserzeichen“ des Fensteroperators nicht vorrücken kann.

Beachten Sie, dass beim Start der Flink-Anwendung (oder bei Datenverzerrungen) einige Shards möglicherweise schneller verbraucht werden als andere. Dies kann dazu führen, dass einige Wasserzeichen zu früh von einer Unteraufgabe ausgegeben werden (die Unteraufgabe kann das Wasserzeichen basierend auf dem Inhalt eines Shards ausgeben, ohne die anderen Shards, die sie abonniert hat, verbraucht zu haben). Abhilfe schaffen verschiedene Strategien mit Wasserzeichen, die einen Sicherheitspuffer hinzufügen (`forBoundedOutOfOrderness(Duration.ofSeconds(30))` oder verspätet eintreffende Ereignisse ausdrücklich zulassen (`allowedLateness(Time.minutes(5))`.

Festlegen einer UUID für alle Operatoren

Wenn Managed Service für Apache Flink einen Flink-Auftrag für eine Anwendung mit einem Snapshot startet, kann der Flink-Auftrag aufgrund bestimmter Probleme nicht gestartet werden. Eines davon ist die Nichtübereinstimmung der Operator-ID. Flink erwartet explizite, konsistente Operator-IDs für Flink-Auftragsdiagramm-Operatoren. Wenn nicht explizit gesetzt, generiert Flink automatisch eine ID für die Operatoren. Das liegt daran, dass Flink diese Operator-IDs verwendet, um die Operatoren in einem Auftragsdiagramm eindeutig zu identifizieren, und sie verwendet, um den Zustand jedes Operators in einem Savepoint zu speichern.

Das Problem der Nichtübereinstimmung der Operator-ID tritt auf, wenn Flink keine 1:1-Zuordnung zwischen den Operator-IDs eines Auftragsdiagramms und den in einem Savepoint definierten Operator-IDs findet. Dies passiert, wenn keine expliziten konsistenten Operator-IDs gesetzt sind und Flink automatisch Operator-IDs generiert, die möglicherweise nicht bei jeder Auftragsdiagramm-Erstellung konsistent sind. Die Wahrscheinlichkeit, dass Anwendungen bei Wartungsarbeiten auf dieses Problem stoßen, ist hoch. Um dies zu vermeiden, empfehlen wir Kunden, die UUID für alle Operatoren im Flink-Code festzulegen. Weitere Informationen finden Sie im Abschnitt Festlegen einer UUID für alle Operatoren unter Produktionsbereitschaft.

Dem ServiceResourceTransformer Maven Shade Plugin hinzufügen

Flink verwendet die Service Provider Interfaces (SPI) von Java, um Komponenten wie Konnektoren und Formate zu laden. Mehrere Flink-Abhängigkeiten, die SPI verwenden, können zu Konflikten im uber-jar und zu unerwartetem Anwendungsverhalten führen. Es wird empfohlen, den ServiceResourceTransformer des Maven-Shade-Plugins hinzuzufügen, der in pom.xml definiert ist

```
<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>
```

Apache Flink Stateful Functions

Stateful Functions ist eine API, die die Erstellung verteilter zustandsbehafteter Anwendungen vereinfacht. Sie basiert auf Funktionen mit persistentem Status, die dynamisch mit starken Konsistenzgarantien interagieren können.

Eine Stateful-Functions-Anwendung ist im Grunde lediglich eine Apache-Flink-Anwendung und kann daher in Managed Service für Apache Flink bereitgestellt werden. Es gibt jedoch einige Unterschiede zwischen der Paketierung von Stateful Functions für einen Kubernetes-Cluster und für Managed Service für Apache Flink. Der wichtigste Aspekt einer Stateful-Functions-Anwendung ist, dass die Modulkonfiguration alle erforderlichen Laufzeitinformationen zur Konfiguration der Stateful-Functions-Laufzeit enthält. Diese Konfiguration wird normalerweise in einen Stateful-Functions-spezifischen Container gepackt und auf Kubernetes bereitgestellt. Aber das ist mit Managed Service für Apache Flink nicht möglich.

Es folgt eine Anpassung des StateFun-Python-Beispiels für Managed Service für Apache Flink:

Apache Flink Anwendungsvorlage

Anstatt einen Kundencontainer für die Stateful-Functions-Laufzeit zu verwenden, können Kunden eine Flink-Anwendungs-JAR-Datei kompilieren, die lediglich die Stateful-Functions-Laufzeit aufruft und die erforderlichen Abhängigkeiten enthält. Für Flink 1.13 sehen die erforderlichen Abhängigkeiten etwa wie folgt aus:

```
<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>
```

Und die Hauptmethode der Flink-Anwendung zum Aufrufen der Stateful-Function-Laufzeit sieht so aus:

```
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);
}
```

Beachten Sie, dass diese Komponenten generisch und unabhängig von der Logik sind, die in der Stateful Function implementiert ist.

Ort der Modulkonfiguration

Die Konfiguration des Stateful-Functions-Moduls muss im Klassenpfad enthalten sein, damit sie für die Stateful-Functions-Laufzeit auffindbar ist. Am besten fügen Sie es in den Ressourcenordner der Flink-Anwendung ein und packen es in die JAR-Datei.

Ähnlich wie bei einer gewöhnlichen Apache-Flink-Anwendung können Sie dann Maven verwenden, um eine Uber-JAR-Datei zu erstellen und diese auf Managed Service für Apache Flink bereitzustellen.

Informationen zu früheren Versionen für Managed Service für Apache Flink

Dieses Thema enthält Informationen zur Verwendung von Managed Service für Apache Flink mit älteren Versionen von Apache Flink. Die Versionen von Apache Flink, die Managed Service für Apache Flink unterstützt, sind 1.15.2 (empfohlen), 1.13.2, 1.11.1, 1.8.2 und 1.6.2.

Wir empfehlen, dass Sie die neueste unterstützte Version von Apache Flink mit Ihrer Anwendung Managed Service für Apache Flink verwenden. Apache Flink Version 1.15.2 hat die folgenden Funktionen:

- Support für Apache Flink Tabellen-API und SQL
- Support für Python-Anwendungen.
- Support für Java Version 11 und jede Scala-Version
- Ein verbessertes Speichermodell
- RocksDB-Optimierungen für erhöhte Anwendungsstabilität
- Support für Taskmanager und Stack-Traces im Apache Flink-Dashboard.

Dieses Thema enthält die folgenden Abschnitte:

- Verwenden des Apache Flink Kinesis Streams-Konnektor mit früheren Apache Flink-Versionen
- Anwendungen mit Apache Flink 1.8.2 erstellen
- Erstellen von Anwendungen mit Apache Flink 1.6.2
- Aktualisieren von Anwendungen
- Verfügbare Konnektoren in Apache Flink 1.6.2 und 1.8.2
- Erste Schritte: Flink 1.13.2
- Erste Schritte: Flink 1.11.1
- Erste Schritte: Flink 1.8.2
- Erste Schritte: Flink 1.6.2

Verwenden des Apache Flink Kinesis Streams-Konnektor mit früheren Apache Flink-Versionen

Der Apache Flink Kinesis Streams-Konnektor war vor Version 1.11 nicht in Apache Flink enthalten. Damit Ihre Anwendung den Apache Flink Kinesis-Konnektor mit früheren Versionen von Apache Flink verwenden kann, müssen Sie die Version von Apache Flink, die Ihre Anwendung verwendet, herunterladen, kompilieren und installieren. Dieser Konnektor wird verwendet, um Daten aus einem Kinesis Stream zu konsumieren, der als Anwendungsquelle verwendet wird, oder um Daten in einen Kinesis Stream zu schreiben, der für die Anwendungsausgabe verwendet wird.

> **Note**
>
> Stellen Sie sicher, dass Sie den Konnektor mit KPL-Version 0.14.0 oder höher erstellen.

Gehen Sie wie folgt vor, um den Quellcode von Apache Flink Version 1.8.2 herunterzuladen und zu installieren:

1. Stellen Sie sicher, dass Sie Apache Maven installiert haben und dass Ihre `JAVA_HOME` Umgebungsvariable auf ein JDK und nicht auf eine JRE verweist. Sie können Ihre Apache Maven-Installation mit dem folgenden Befehl testen:

   ```
   mvn -version
   ```

2. Laden Sie den Quellcode von Apache Flink Version 1.8.2 herunter:

   ```
   wget https://archive.apache.org/dist/flink/flink-1.8.2/flink-1.8.2-src.tgz
   ```

3. Dekomprimieren Sie den Apache Flink-Quellcode:

   ```
   tar -xvf flink-1.8.2-src.tgz
   ```

4. Wechseln Sie in das Apache Flink-Quellcodeverzeichnis:

   ```
   cd flink-1.8.2
   ```

5. Kompilieren und installieren Sie Apache Flink:

   ```
   mvn clean install -Pinclude-kinesis -DskipTests
   ```

> **Note**
>
> Wenn Sie Flink unter Microsoft Windows kompilieren, müssen Sie den `-Drat.skip=true` Parameter hinzufügen.

Anwendungen mit Apache Flink 1.8.2 erstellen

Dieser Abschnitt enthält Informationen über Komponenten, die Sie für die Erstellung von Managed Service für Apache Flink-Anwendungen verwenden, die mit Apache Flink 1.8.2 funktionieren.

Verwenden Sie die folgenden Komponentenversionen für Managed Service für Apache Flink-Anwendungen:

Komponente	Version
Java	1.8 (empfohlen)
Apache Flink	1.8.2
Managed Service für Apache Flink für Flink Laufzeit (aws-kinesisanalytics-runtime)	1.0.1
Managed Service für Apache Flink Flink-Konnektoren (aws-kinesisanalytics-runtime)	1.0.1
Apache Maven	3.1

Um eine Anwendung mit Apache Flink 1.8.2 zu kompilieren, führen Sie Maven mit dem folgenden Parameter aus:

```
mvn package -Dflink.version=1.8.2
```

Ein Beispiel für eine `pom.xml`-Datei für eine Managed Service für Apache Flink-Anwendung, die Apache Flink Version 1.8.2 verwendet, finden Sie in der Managed Service für Apache Flink 1.8.2 Erste Schritte-Anwendung.

Informationen zum Erstellen und Verwenden von Anwendungscode für eine Managed Service für Apache Flink-Anwendung finden Sie unter Erstellen von Anwendungen.

Erstellen von Anwendungen mit Apache Flink 1.6.2

Dieser Abschnitt enthält Informationen über Komponenten, die Sie für die Erstellung von Managed Service für Apache Flink-Anwendungen verwenden, die mit Apache Flink 1.6.2 funktionieren.

Verwenden Sie die folgenden Komponentenversionen für Managed Service für Apache Flink-Anwendungen:

Komponente	Version
Java	1.8 (empfohlen)
AWS Java-SDK	1.11.379
Apache Flink	1.6.2
Managed Service für Apache Flink für Flink Laufzeit (aws-kinesisanalytics-runtime)	1.0.1
Managed Service für Apache Flink Flink-Konnektoren (aws-kinesisanalytics-runtime)	1.0.1
Apache Maven	3.1
Apache Beam	Wird mit Apache Flink 1.6.2 nicht unterstützt.

> **Note**
>
> Wenn Sie Managed Service für Apache Flink Laufzeit Version 1.0.1 verwenden, geben Sie die Version von Apache Flink in Ihrer `pom.xml`-Datei an, anstatt den `-Dflink.version`-Parameter beim Kompilieren Ihres Anwendungscodes zu verwenden.

Ein Beispiel für eine `pom.xml`-Datei für eine Managed Service für Apache Flink-Anwendung, die Apache Flink Version 1.6.2 verwendet, finden Sie in der Managed Service für Apache Flink 1.6.2 Erste Schritte-Anwendung.

Informationen zum Erstellen und Verwenden von Anwendungscode für eine Managed Service für Apache Flink-Anwendung finden Sie unter Erstellen von Anwendungen.

Aktualisieren von Anwendungen

Um die Version einer Managed Service für Apache Flink-Anwendung zu aktualisieren, müssen Sie Ihren Anwendungscode aktualisieren, die vorherige Anwendung löschen und eine neue Anwendung mit dem aktualisierten Code erstellen. Gehen Sie dazu wie folgt vor:

- Ändern Sie die Versionen von Managed Service für Apache Flink Laufzeit und Managed Service für Apache Flink Flink-Konnektoren (aws-kinesisanalytics-flink) in der `pom.xml`-Datei Ihrer Anwendung auf 1.1.0.
- Entfernen Sie die `flink.version`-Eigenschaft aus der `pom.xml`-Datei Ihrer Anwendung. Sie geben diesen Parameter an, wenn Sie den Anwendungscode im nächsten Schritt kompilieren.
- Kompilieren Sie Ihren Anwendungscode mit dem folgenden Befehl neu:

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

- Löschen Sie Ihre bestehende Anwendung. Erstellen Sie Ihre Anwendung erneut und wählen Sie Apache Flink Version 1.15.2 (empfohlene Version) für die Laufzeit der Anwendung.

> **Note**
>
> Sie können keine Snapshots aus Ihren früheren Anwendungsversionen verwenden.

Verfügbare Konnektoren in Apache Flink 1.6.2 und 1.8.2

Das Apache Flink Framework enthält Konnektoren für den Zugriff auf Daten aus verschiedenen Quellen.

- Informationen zu den im Apache Flink 1.6.2-Framework verfügbaren Konnektoren finden Sie unter Konnektoren (1.6.2) in der Apache Flink-Dokumentation (1.6.2).
- Informationen zu den im Apache Flink 1.8.2-Framework verfügbaren Konnektoren finden Sie unter Konnektoren (1.8.2) in der Apache Flink-Dokumentation (1.8.2).

Erste Schritte: Flink 1.13.2

In diesem Abschnitt werden Ihnen die grundlegenden Konzepte von Managed Service für Apache Flink und der DataStream -API vorgestellt. Es werden die verfügbaren Optionen für die Erstellung und das Testen von Anwendungen beschrieben. Er enthält auch Anweisungen zur Installation der Tools, die Sie benötigen, um die Tutorials in diesem Handbuch abzuschließen und Ihre erste Anwendung zu erstellen.

Themen

- Komponenten der Anwendung Managed Service für Apache Flink
- Voraussetzungen für das Fertigstellen der Übungen
- Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Administratorbenutzers
- Nächster Schritt
- Schritt 2: Einrichten der AWS Command Line Interface (AWS CLI)
- Schritt 3: Erstellen und Ausführen eines Managed Service für Apache Flink-Anwendung
- Schritt 4: Bereinigen von AWS-Ressourcen
- Schritt 5: Nächste Schritte

Komponenten der Anwendung Managed Service für Apache Flink

Zur Verarbeitung von Daten verwendet Ihre Managed Service für Apache Flink-Anwendung eine Java/Apache Maven- oder Scala-Anwendung, die mithilfe der Apache Flink-Laufzeit Eingaben verarbeitet und Ausgaben erzeugt.

Die Anwendung Managed Service für Apache Flink besteht aus folgenden Komponenten:

- Runtime-Eigenschaften: Sie können Runtime-Eigenschaften verwenden, um Ihre Anwendung zu konfigurieren, ohne Ihren Anwendungscode neu kompilieren zu müssen.
- Quelle: Die Anwendung verwendet Daten mithilfe einer Quelle. Ein Quell-Connector liest Daten aus einem Kinesis Data Stream, einem Amazon S3-Bucket usw. Weitere Informationen finden Sie unter Quellen.
- Operatoren: Die Anwendung verarbeitet Daten mithilfe eines oder mehrerer Operatoren. Ein Operator kann Daten transformieren, anreichern oder aggregieren. Weitere Informationen finden Sie unter DataStream-API Operatoren.

- Senke: Die Anwendung erzeugt Daten für externe Quellen mithilfe von Senken. Ein Senken-Connector schreibt Daten in einen Kinesis Data Stream, einen Kinesis Data Firehose-Stream, einen Amazon S3-Bucket usw. Weitere Informationen finden Sie unter Senken.

Nachdem Sie Ihren Anwendungscode erstellt, kompiliert und verpackt haben, laden Sie das Codepaket in einen Amazon Simple Storage Service (Amazon S3)-Bucket hoch. Anschließend erstellen Sie eine Managed Service für Apache Flink-Anwendung. Sie übergeben den Speicherort des Codepakets, einen Kinesis Data Stream als Streaming-Datenquelle und in der Regel einen Streaming- oder Dateispeicherort, der die verarbeiteten Daten der Anwendung empfängt.

Voraussetzungen für das Fertigstellen der Übungen

Zur Durchführung der Schritte in dieser Anleitung benötigen Sie Folgendes:

- Java Development Kit (JDK), Version 11. Legen Sie die `JAVA_HOME` Umgebungsvariable so fest, dass sie auf Ihren JDK-Installationsspeicherort weist.
- Wir empfehlen die Verwendung einer Entwicklungsumgebung (wie Eclipse Java Neon oder IntelliJ Idea), um Ihre Anwendung zu entwickeln und zu kompilieren.
- Git-Client. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben.
- Apache Maven-Compiler-Plugin. Maven muss sich in Ihrem Arbeitspfad befinden. Zum Testen Ihrer Apache Maven-Installation geben Sie Folgendes ein:

```
$ mvn -version
```

Um zu beginnen, gehen Sie zu Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Administratorbenutzers.

Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Administratorbenutzers

So melden Sie sich für ein AWS-Konto an

Wenn Sie kein AWS-Konto haben, führen Sie die folgenden Schritte zum Erstellen durch.

Anmeldung für ein AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.

2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für ein AWS-Konto anmelden, wird ein Root-Benutzer des AWS-Kontos erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen eine Bestätigungs-E-Mail, sobald die Anmeldung abgeschlossen ist. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Erstellen eines Administratorbenutzers

Nachdem Sie sich für ein AWS-Kontoangemeldet haben, sichern Sie Ihr Root-Benutzer des AWS-Kontos, aktivieren Sie AWS IAM Identity Centerund erstellen Sie einen administrativen Benutzer, damit Sie nicht den Root-Benutzer für alltägliche Aufgaben verwenden.

Schützen Ihres Root-Benutzer des AWS-Kontos

1. Melden Sie sich bei der AWS Management Console als Kontobesitzer an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto-E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-AnmeldungBenutzerhandbuch zu .

2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen dazu finden Sie unter Aktivieren eines virtuellen MFA-Geräts für den Root-Benutzer Ihres AWS-Konto (Konsole) im IAM-Benutzerhandbuch.

Erstellen eines Administratorbenutzers

1. Aktivieren von IAM Identity Center.

Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.

2. Im IAM Identity Center gewähren Sie einem administrativen Benutzer administrativen Zugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit dem standardmäßigen IAM-Identity-Center-Verzeichnis konfigurieren im AWS IAM Identity Center-Benutzerhandbuch.

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM-Identity-Center-Benutzer finden Sie unter Anmelden beim AWS-Zugangsportal im AWS-Anmeldung Benutzerhandbuch zu.

Erteilen programmgesteuerten Zugriffs

Benutzer benötigen programmgesteuerten Zugriff, wenn sie außerhalb der AWS Management Console mit AWS interagieren möchten. Die Vorgehensweise, um programmgesteuerten Zugriff zu gewähren, hängt davon ab, welcher Benutzertyp auf zugreift AWS.

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zur AWS CLI finden Sie unter Konfigurieren der AWS CLI für die Verwendung von AWS IAM Identity Center im AWS

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
		Command Line Interface-Benutzerhandbuch. • Informationen zu AWS-SDKs, Tools und AWS-APIs finden Sie unter IAM-Identity-Center-Authentifizierung im Referenzhandbuch zu AWS-SDKs und Tools.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Folgen Sie den Anweisungen unter Verwenden temporärer Anmeldeinformationen mit AWS-Ressourcen im IAM-Benutzerhandbuch.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zur AWS CLI finden Sie unter Authentifizierung mit IAM-Benutzer-Anmeldeinformationen im AWS Command Line Interface-Benutzerhandbuch. • Informationen zu AWS-SDKs und Tools finden Sie unter Authentifizierung mit langfristigen Anmeldeinformationen im Referenzhandbuch zu AWS-SDKs und Tools. • Informationen zu AWS-APIs finden Sie unter Verwalten von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch.

Nächster Schritt

Schritt 2: Einrichten der AWS Command Line Interface (AWS CLI)

Nächster Schritt

Schritt 2: Einrichten der AWS Command Line Interface (AWS CLI)

Schritt 2: Einrichten der AWS Command Line Interface (AWS CLI)

In diesem Schritt laden Sie die AWS CLI herunter und konfigurieren sie für die Verwendung mit Amazon Managed Service für Apache Flink.

> **Note**
>
> Bei allen Erste-Schritte-Übungen in diesem Handbuch wird davon ausgegangen, dass Sie in Ihrem Konto Administrator-Anmeldeinformationen (`adminuser`) verwenden, um die Operationen auszuführen.

> **Note**
>
> Wenn Sie die AWS CLI bereits installiert haben, müssen Sie möglicherweise ein Upgrade durchführen, um die neueste Funktionalität zu erhalten. Weitere Informationen finden Sie unter Installieren der AWS Command Line Interface im AWS Command Line Interface-Benutzerhandbuch. Zum Überprüfen der Version der AWS CLI führen Sie den folgenden Befehl aus:
>
> ```
> aws --version
> ```
>
> Die Übungen in diesem Tutorial erfordern die folgende AWS CLI-Version oder höher:
>
> ```
> aws-cli/1.16.63
> ```

Um das AWS CLI einzurichten

1. Herunterladen und Konfigurieren von AWS CLI. Eine Anleitung finden Sie unter den folgenden Themen im AWS Command Line Interface-Benutzerhandbuch:
   - Installieren des AWS Command Line Interface
   - Konfigurieren von AWS CLI
2. Fügen Sie ein benanntes Profil für den Administratorbenutzer in der AWS CLI `config`-Datei hinzu. Verwenden Sie dieses Profil beim Ausführen von AWS CLI-Befehlen. Weitere

Informationen zu benannten Profilen finden Sie unter Benannte Profile im AWS Command Line Interface Benutzerhandbuch.

```
[profile adminuser]
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
```

Eine Liste der verfügbaren AWS-Regionen finden Sie unter Regionen und Endpunkte im Allgemeine Amazon Web Services-Referenz

> **Note**
>
> Der Beispielcode und die Befehle in diesem Tutorial verwenden die Region USA West (Oregon). Um eine andere Region zu verwenden, ändern Sie die Region im Code und in den Befehlen für dieses Tutorial in die Region, die Sie verwenden möchten.

3. Überprüfen Sie die Einrichtung, indem Sie die folgenden Hilfebefehle in die Befehlszeile eingeben:

```
aws help
```

Nachdem Sie ein -AWSKonto und die eingerichtet habenAWS CLI, können Sie die nächste Übung ausprobieren, in der Sie eine Beispielanwendung konfigurieren und die end-to-end Einrichtung testen.

Nächster Schritt

Schritt 3: Erstellen und Ausführen eines Managed Service für Apache Flink-Anwendung

Schritt 3: Erstellen und Ausführen eines Managed Service für Apache Flink-Anwendung

In dieser Übung erstellen Sie eine Anwendung von Managed Service für Apache Flink mit Datenströmen als Quelle und Senke.

Dieser Abschnitt enthält die folgenden Schritte:

- Erstellen von zwei Amazon Kinesis Datenstroms

- Schreiben Sie Beispieldatensätze in den Eingabe-Stream
- Herunterladen und Überprüfen des Apache Flink-Streaming-Java-Codes
- Kompilieren des Anwendungscodes
- Hochladen des Apache Flink-Streaming-Java-Codes
- Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus
- Nächster Schritt

Erstellen von zwei Amazon Kinesis Datenstroms

Bevor Sie für diese Übung eine Anwendung von Managed Service für Apache Flink erstellen, erstellen Sie zwei Kinesis Data Streams (`ExampleInputStream` und `ExampleOutputStream`). Ihre Anwendung verwendet diese Streams für die Quell- und Ziel-Streams der Anwendung.

Sie können diese Streams mithilfe der Amazon-Kinesis-Konsole oder des folgenden AWS CLI-Befehls erstellen. Anweisungen für die Konsole finden Sie unter Erstellen und Aktualisieren von Datenströmen im Amazon Kinesis Data Streams Entwicklerhandbuch.

So erstellen Sie die Daten-Streams (AWS CLI)

1. Verwenden Sie den Befehl Amazon Kinesis `create-stream` AWS CLI, um den ersten Stream (`ExampleInputStream`) zu erstellen.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Um den zweiten Stream zu erstellen, den die Anwendung zum Schreiben der Ausgabe verwendet, führen Sie denselben Befehl aus und ändern den Stream-Namen in `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Abschnitt verwenden Sie ein Python-Skript zum Schreiben von Datensätzen in den Stream für die zu verarbeitende Anwendung.

> **Note**
>
> Dieser Abschnitt erfordert AWS SDK for Python (Boto).

1. Erstellen Sie eine Datei `stock.py` mit dem folgenden Inhalt:

```
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. Im weiteren Verlauf des Tutorials führen Sie das `stock.py`-Skript zum Senden von Daten an die Anwendung aus.

```
$ python stock.py
```

Herunterladen und Überprüfen des Apache Flink-Streaming-Java-Codes

Der Java-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Klonen Sie das Remote-Repository, indem Sie den folgenden Befehl verwenden:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/GettingStarted` Verzeichnis .

Beachten Sie Folgendes zum Anwendungscode:

- Eine Project Object Model (pom.xml) Datei enthält Informationen über die Konfiguration und Abhängigkeiten der Anwendung, einschließlich der Bibliotheken des Managed Service für Apache Flink.
- Die `BasicStreamingJob.java`-Datei enthält die `main`-Methode, die die Funktionalität der Anwendung definiert.
- Die Anwendung verwendet eine Kinesis-Quelle zum Lesen aus dem Quell-Stream. Der folgende Codeausschnitt erstellt die Kinesis-Quelle:

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

- Ihre Anwendung erstellt Quell- und Senkenkonnektoren für den Zugriff auf externe Ressourcen, indem ein `StreamExecutionEnvironment`-Objekt verwendet wird.
- Die Anwendung erstellt Quell- und Senkenkonnektoren mit statischen Eigenschaften. Zum Verwenden dynamischer Anwendungseigenschaften verwenden Sie die Methoden `createSourceFromApplicationProperties` und `createSinkFromApplicationProperties`, um die Konnektoren zu erstellen. Diese Methoden lesen die Eigenschaften der Anwendung zum Konfigurieren der Konnektoren.

  Weitere Informationen zu den Laufzeiteigenschaften finden Sie unter Laufzeiteigenschaften.

Kompilieren des Anwendungscodes

In diesem Abschnitt verwenden Sie den Apache Maven-Compiler zum Erstellen des Java-Codes für die Anwendung. Weitere Informationen zum Installieren von Apache Maven und des Java Development Kit (JDK) finden Sie unter Voraussetzungen für das Fertigstellen der Übungen.

Kompilieren des Anwendungscodes

1. Zum Verwenden Ihres Anwendungscodes kompilieren und packen Sie ihn in eine JAR-Datei. Sie können Ihren Code auf zwei Arten kompilieren und packen:

   - Verwenden Sie das Befehlszeilen-Maven-Tool. Erstellen Sie Ihre JAR-Datei, indem Sie den folgenden Befehl in dem Verzeichnis ausführen, das die `pom.xml`-Datei enthält:

     ```
     mvn package -Dflink.version=1.13.2
     ```

   - Verwenden Sie Ihre Entwicklungsumgebung. Weitere Informationen finden Sie in der Dokumentation Ihrer Entwicklungsumgebung.

     > **Note**
     >
     > Der bereitgestellte Quellcode basiert auf Bibliotheken aus Java 11.

   Sie können Ihr Paket als JAR-Datei hochladen oder komprimieren und als ZIP-Datei hochladen. Wenn Sie Ihre Anwendung mit der AWS CLI erstellen, geben Sie Ihren Codeinhaltstyp (JAR oder ZIP) an.

2. Wenn während der Erstellung Fehler aufgetreten sind, überprüfen Sie, ob Ihre `JAVA_HOME`-Umgebungsvariable richtig eingestellt ist.

Wenn die Anwendung erfolgreich kompiliert wurde, wird die folgende Datei erstellt:

`target/aws-kinesis-analytics-java-apps-1.0.jar`

Hochladen des Apache Flink-Streaming-Java-Codes

In diesem Abschnitt erstellen Sie einen Amazon Simple Storage Service (Amazon S3)-Bucket und laden Ihren Anwendungscode hoch.

So laden Sie den Anwendungscode hoch

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie Bucket erstellen aus.
3. Geben Sie **ka-app-code-*<username>*** im Feld Bucket-Name ein. Fügen Sie dem Bucket-Namen ein Suffix hinzu, wie z. B. Ihren Benutzernamen, damit er global eindeutig ist. Wählen Sie Weiter aus.
4. Lassen Sie im Schritt Optionen konfigurieren die Einstellungen unverändert und klicken Sie auf Weiter.
5. Lassen Sie im Schritt Berechtigungen festlegen die Einstellungen unverändert und klicken Sie auf Weiter.
6. Wählen Sie Bucket erstellen aus.
7. Wählen Sie in der Amazon S3-Konsole den Bucket ka-app-code-***<username>*** und dann Hochladen aus.
8. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `aws-kinesis-analytics-java-apps-1.0.jar`Datei, die Sie im vorherigen Schritt erstellt haben. Wählen Sie Weiter aus.
9. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus

Sie können eine Anwendung von Managed Service für Apache Flink entweder über die Konsole oder die AWS CLI erstellen und ausführen.

> **Note**
>
> Wenn Sie die Anwendung mit der Konsole erstellen, werden Ihre AWS Identity and Access Management (IAM)- und Amazon- CloudWatch Logs-Ressourcen für Sie erstellt. Wenn Sie die Anwendung mit der AWS CLI erstellen, erstellen Sie diese Ressourcen separat.

Themen

- Erstellen und Ausführen der Anwendung (Konsole)
- Erstellen und Ausführen der Anwendung (AWS CLI)

Erstellen und Ausführen der Anwendung (Konsole)

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Geben Sie für Beschreibung den Text **`My java test app`** ein.
   - Wählen Sie für Runtime (Laufzeit) die Option Apache Flink aus.
   - Behalten Sie im Pulldown-Menü die Version Apache Flink Version 1.13 bei.
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rolle: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf die Kinesis-Datenströme.

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **`kinesis-analytics-service-MyApplication-us-west-2`**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*012345678901*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):

- Geben Sie für Amazon-S3-Bucket **ka-app-code-*<username>*** ein.
- Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **aws-kinesis-analytics-java-apps-1.0.jar** ein.

3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **kinesis-analytics-MyApplication-us-west-2** erstellen/aktualisieren aus.
4. Geben Sie Folgendes ein:

Gruppen-ID	Schlüssel	Wert
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

5. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
6. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
7. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die Amazon- CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`

Ausführen der Anwendung

Das Flink-Jobdiagramm kann angezeigt werden, indem Sie die Anwendung ausführen, das Apache Flink-Dashboard öffnen und den gewünschten Flink-Job auswählen.

Stoppen der Anwendung

Wählen Sie auf der MyApplication Seite Stoppen aus. Bestätigen Sie die Aktion.

Aktualisieren der Anwendung

Mithilfe der Konsole können Sie Anwendungseinstellungen wie beispielsweise Anwendungseigenschaften, Überwachungseinstellungen und den Speicherort oder den Dateinamen der JAR-Anwendungsdatei aktualisieren. Außerdem können Sie die JAR-Anwendungsdatei erneut aus dem Amazon-S3-Bucket laden, wenn Sie den Anwendungscode aktualisieren müssen.

Wählen Sie auf der MyApplication Seite Konfigurieren aus. Aktualisieren Sie die Anwendungseinstellungen und klicken Sie auf Aktualisieren.

Erstellen und Ausführen der Anwendung (AWS CLI)

In diesem Abschnitt verwenden Sie AWS CLI, um die Anwendung Managed Service für Apache Flink zu erstellen und auszuführen. Managed Service für Apache Flink verwendet den Befehl `kinesisanalyticsv2` AWS CLI, um Managed-Service-für-Apache-Flink-Anwendungen zu erstellen und mit diesen zu interagieren.

Erstellen einer Berechtigungsrichtlinie

> **Note**
>
> Sie müssen eine Berechtigungsrichtlinie und eine Rolle für Ihre Anwendung erstellen. Wenn Sie diese IAM-Ressourcen nicht erstellen, kann Ihre Anwendung nicht auf ihre Daten und Protokollstreams zugreifen.

Zuerst erstellen Sie eine Berechtigungsrichtlinie mit zwei Anweisungen: eine, die Berechtigungen für die `read`-Aktion auf den Quell-Stream zulässt, und eine andere, die Berechtigungen für die `write`-Aktionen auf den Senken-Stream zulässt. Anschließend fügen Sie die Richtlinie an eine IAM-Rolle (die Sie im nächsten Abschnitt erstellen) an. Wenn Managed Service für Apache Flink also die Rolle

übernimmt, verfügt der Service über die erforderlichen Berechtigungen zum Lesen aus dem Quell-Stream und zum Schreiben in den Senken-Stream.

Verwenden Sie den folgenden Code zum Erstellen der `AKReadSourceStreamWriteSinkStream`-Berechtigungsrichtlinie. Ersetzen Sie *`username`* durch den Benutzernamen, den Sie verwendet haben, um den Amazon-S3-Bucket zum Speichern des Anwendungscodes zu erstellen. Ersetzen Sie die Konto-ID in den Amazon-Ressourcennamen (ARNs) (*`012345678901`*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

step-by-step Anweisungen zum Erstellen einer Berechtigungsrichtlinie finden Sie unter Tutorial: Erstellen und Anfügen Ihrer ersten vom Kunden verwalteten Richtlinie im IAM-Benutzerhandbuch.

> **Note**
>
> Um auf andere Amazon-Services zuzugreifen, können Sie AWS SDK for Java verwenden. Managed Service für Apache Flink setzt die vom SDK benötigten Anmeldeinformationen automatisch auf die der IAM-Rolle für die Dienstausführung, die mit Ihrer Anwendung verknüpft ist. Es sind keine weiteren Schritte erforderlich.

Erstellen einer IAM-Rolle

In diesem Abschnitt erstellen Sie eine IAM-Rolle, die die Anwendung von Managed Service für Apache Flink annehmen kann, um einen Quell-Stream zu lesen und in den Senken-Stream zu schreiben.

Managed Service für Apache Flink kann ohne Berechtigungen nicht auf Ihren Stream zugreifen. Sie erteilen diese Berechtigungen über eine IAM-Rolle. Jeder IAM-Rolle sind zwei Richtlinien angefügt. Die Vertrauensrichtlinie erteilt Managed Service für Apache Flink die Berechtigung zum Übernehmen der Rolle und die Berechtigungsrichtlinie bestimmt, was Managed Service für Apache Flink nach Annahme der Rolle tun kann.

Sie können die Berechtigungsrichtlinie, die Sie im vorherigen Abschnitt erstellt haben, dieser Rolle anfügen.

So erstellen Sie eine IAM-Rolle

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Roles (Rollen) und Create Role (Rolle erstellen) aus.
3. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option AWS-Service aus. Wählen Sie unter Choose the service that will use this role (Wählen Sie den Service aus, der diese Rolle verwendet) die Option Kinesis aus. Wählen Sie unter Select your use case (Wählen Sie Ihren Anwendungsfall aus) die Option Kinesis Analytics aus.

   Wählen Sie Weiter: Berechtigungen aus.
4. Wählen Sie auf der Seite Attach permissions policies (Berechtigungsrichtlinien hinzufügen) Next: Review (Weiter: Überprüfen) aus. Sie fügen Berechtigungsrichtlinien an, nachdem Sie die Rolle erstellt haben.
5. Geben Sie auf der Seite Create role (Rolle erstellen) den Text **`MF-stream-rw-role`** für Role name (Rollenname) ein. Wählen Sie Rolle erstellen aus.

Jetzt haben Sie eine neue IAM-Rolle mit dem Namen `MF-stream-rw-role` erstellt. Im nächsten Schritt aktualisieren Sie die Vertrauens- und Berechtigungsrichtlinien für die Rolle.

6. Fügen Sie die Berechtigungsrichtlinie der Rolle an.

> **Note**
>
> Für diese Übung übernimmt Managed Service für Apache Flink diese Rolle sowohl für das Lesen von Daten aus einem Kinesis-Datenstrom (Quelle) als auch zum Schreiben der Ausgabedaten in einen anderen Kinesis-Datenstrom. Daher fügen Sie die Richtlinie an, die Sie im vorherigen Schritt erstellt haben, the section called “Erstellen einer Berechtigungsrichtlinie”.

   a. Wählen Sie auf der Seite Summary (Übersicht) die Registerkarte Permissions (Berechtigungen) aus.
   b. Wählen Sie Attach Policies (Richtlinien anfügen) aus.
   c. Geben Sie im Suchfeld **`AKReadSourceStreamWriteSinkStream`** (die Richtlinie, die Sie im vorhergehenden Abschnitt erstellt haben) ein.
   d. Wählen Sie die AKReadSourceStreamWriteSinkStream-Richtlinie und dann Richtlinie anfügen aus.

Sie haben nun die Service-Ausführungsrolle erstellt, die Ihre Anwendung für den Zugriff auf Ressourcen verwendet. Notieren Sie sich den ARN der neuen Rolle.

step-by-step Anweisungen zum Erstellen einer Rolle finden Sie unter Erstellen einer IAM-Rolle (Konsole) im IAM-Benutzerhandbuch.

Erstellen Sie die Anwendung Managed Service für Apache Flink

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `create_request.json`. Ersetzen Sie den Beispiel-Rollen-ARN durch den ARN für die Rolle, die Sie zuvor erstellt haben. Ersetzen Sie das Bucket-ARN-Suffix (*`username`*) mit dem Suffix, das Sie im vorherigen Abschnitt gewählt haben. Ersetzen Sie die beispielhafte Konto-ID (*`012345678901`*) in der Service-Ausführungsrolle mit Ihrer Konto-ID.

```
{
    "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. Führen Sie die `CreateApplication`-Aktion mit der vorherigen Anforderung zum Erstellen der Anwendung aus:

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

Die Anwendung wird nun erstellt. Sie starten die Anwendung im nächsten Schritt.

Starten der Anwendung

In diesem Abschnitt verwenden Sie die `StartApplication`-Aktion, um die Anwendung zu starten.

So starten Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `start_request.json`.

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

2. Führen Sie die `StartApplication`-Aktion mit der vorherigen Anforderung zum Starten der Anwendung aus:

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

Die Anwendung wird jetzt ausgeführt. Sie können die Metriken von Managed Service für Apache Flink in der Amazon- CloudWatch Konsole überprüfen, um zu überprüfen, ob die Anwendung funktioniert.

Stoppen der Anwendung

In diesem Abschnitt verwenden Sie die `StopApplication`-Aktion, um die Anwendung zu stoppen.

So stoppen Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `stop_request.json`.

```
{
   "ApplicationName": "test"
}
```

2. Führen Sie die `StopApplication`-Aktion mit der folgenden Anforderung zum Stoppen der Anwendung aus:

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

Die Anwendung wird nun gestoppt.

Hinzufügen einer CloudWatch Protokollierungsoption

Sie können die verwendenAWS CLI, um Ihrer Anwendung einen Amazon- CloudWatch Protokollstream hinzuzufügen. Informationen zur Verwendung von - CloudWatch Protokollen mit Ihrer Anwendung finden Sie unter the section called “Einrichten der Protokollierung”.

Umgebungseigenschaften aktualisieren

In diesem Abschnitt verwenden Sie die `UpdateApplication`-Aktion, um die Umgebungseigenschaften für die Anwendung zu ändern, ohne den Anwendungscode neu kompilieren zu müssen. In diesem Beispiel ändern Sie die Region der Quell- und Ziel-Streams.

So aktualisieren Sie die Umgebungseigenschaften für die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `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. Führen Sie die UpdateApplication-Aktion mit der vorherigen Anforderung aus, um die Umgebungseigenschaften zu aktualisieren:

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

Aktualisieren Sie den Anwendungscode

Wenn Sie Ihren Anwendungscode mit einer neuen Version Ihres Codepakets aktualisieren müssen, verwenden Sie die UpdateApplication AWS CLI-Aktion.

> **Note**
>
> Um eine neue Version des Anwendungscodes mit demselben Dateinamen zu laden, müssen Sie die neue Objektversion angeben. Weitere Informationen zur Verwendung von Amazon S3-Objektversionen finden Sie unter Versionierung aktivieren oder deaktivieren.

Um die AWS CLI zu verwenden, löschen Sie Ihr vorheriges Codepaket aus Ihrem Amazon S3-Bucket, laden Sie die neue Version hoch und rufen Sie `UpdateApplication` auf. Geben Sie dabei denselben Amazon S3-Bucket und Objektnamen sowie die neue Objektversion an. Die Anwendung wird mit dem neuen Codepaket neu gestartet.

Die folgende Beispielanforderung für die `UpdateApplication`-Aktion lädt den Anwendungscode neu und startet die Anwendung neu. Aktualisieren Sie die `CurrentApplicationVersionId` auf die aktuelle Anwendungsversion. Sie können die aktuelle Anwendungsversion mithilfe der Aktionen `ListApplications` oder `DescribeApplication` überprüfen. Aktualisieren Sie das Bucket-Namenssuffix (*<username>*) mit dem Suffix, das Sie im the section called “Erstellen von zwei Amazon Kinesis Datenstroms”-Abschnitt ausgewählt haben.

```
{
    "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"
                }
            }
        }
    }
}
```

Nächster Schritt

[Schritt 4: Bereinigen von AWS-Ressourcen]

Schritt 4: Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zum Bereinigen von AWS-Ressourcen, die im Tutorial Erste Schritte erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte:

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Sie Ihre Kinesis Data Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen
- Nächster Schritt

Löschen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen Sie Ihre Kinesis Data Streams

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Bereich Kinesis Data Streams die Option ausExampleInputStream.
3. Wählen Sie auf der ExampleInputStream Seite Kinesis Stream löschen aus und bestätigen Sie dann den Löschvorgang.
4. Wählen Sie auf der Seite Kinesis Streams die , ExampleOutputStreamwählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den ka-app-codeBucket -***<username>*** aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle kinesis-analytics-MyApplication--us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Nächster Schritt

Schritt 5: Nächste Schritte

Schritt 5: Nächste Schritte

Nachdem Sie nun eine grundlegende Managed Service für Apache Flink-Anwendung erstellt und ausgeführt haben, finden Sie in den folgenden Ressourcen erweiterte Managed Service für Apache Flink-Lösungen.

- Die AWS Streaming Data Solution for Amazon Kinesis : Die AWS Streaming Data Solution for Amazon Kinesis konfiguriert automatisch die AWS-Services, die zum einfachen Erfassen, Speichern, Verarbeiten und Bereitstellen von Streaming-Daten erforderlich sind. Die Lösung bietet mehrere Optionen zur Lösung von Anwendungsfällen mit Streaming-Daten. Die Option Managed Service für Apache Flink bietet ein end-to-end -Streaming-ETL-Beispiel, das eine reale Anwendung demonstriert, die analytische Operationen für simulierte New York-Taxisdaten ausführt. Die Lösung richtet alle erforderlichen AWS Ressourcen wie IAM-Rollen und -Richtlinien, ein CloudWatch Dashboard und CloudWatch Alarme ein.
- AWS Streaming Data Solution for Amazon MSK: Die AWS Streaming Data Solution for Amazon MSK bietet AWS CloudFormation-Vorlagen, in denen Daten durch Produzenten, Streaming-Speicher, Verbraucher und Ziele fließen.
- Clickstream Lab mit Apache Flink und Apache Kafka: Ein End-to-End-Lab für Clickstream-Anwendungsfälle mit Amazon Managed Streaming for Apache Kafka als Streaming-Speicher und Managed Service für Apache Flink für Apache Flink-Anwendungen zur Stream-Verarbeitung.
- Workshop für Amazon Managed Service für Apache Flink: In diesem Workshop erstellen Sie eine - end-to-end Streaming-Architektur, um Streaming-Daten nahezu in Echtzeit aufzunehmen, zu analysieren und zu visualisieren. Sie haben sich vorgenommen, den Betrieb eines Taxiunternehmens in New York City zu verbessern. Sie analysieren die Telemetriedaten einer Taxiflotte in New York City nahezu in Echtzeit, um deren Flottenbetrieb zu optimieren.
- Managed Service für Apache Flink: Beispiele: Dieser Abschnitt dieses Entwicklerhandbuchs enthält Beispiele für die Erstellung von und die Arbeit mit Anwendungen in Managed Service für Apache Flink. Sie enthalten Beispielcode und step-by-step Anweisungen, mit denen Sie Managed Service für Apache Flink-Anwendungen erstellen und Ihre Ergebnisse testen können.
- Lernen Sie Flink kennen: Praktisches Training: Offizielle Apache Flink-Einführungsschulung, die Ihnen den Einstieg in die Entwicklung skalierbarer Streaming-ETL-, Analyse- und ereignisgesteuerter Anwendungen ermöglicht.

> **Note**
>
> Beachten Sie, dass Managed Service für Apache Flink die in dieser Schulung verwendete Apache Flink-Version (1.12) nicht unterstützt. Sie können Flink 1.15.2 in Flink Managed Service für Apache Flink verwenden.

Erste Schritte: Flink 1.11.1

Dieses Thema enthält eine Version des Erste Schritte (DataStream API)-Tutorials, die Apache Flink 1.11.1 verwendet.

In diesem Abschnitt werden Ihnen die grundlegenden Konzepte von Managed Service für Apache Flink und der DataStream -API vorgestellt. Es werden die verfügbaren Optionen für die Erstellung und das Testen von Anwendungen beschrieben. Er enthält auch Anweisungen zur Installation der Tools, die Sie benötigen, um die Tutorials in diesem Handbuch abzuschließen und Ihre erste Anwendung zu erstellen.

Themen

- Komponenten der Anwendung Managed Service für Apache Flink
- Voraussetzungen für das Fertigstellen der Übungen
- Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Administratorbenutzers
- Schritt 2: Einrichten der AWS Command Line Interface (AWS CLI)
- Schritt 3: Erstellen und Ausführen eines Managed Service für Apache Flink-Anwendung
- Schritt 4: Bereinigen von AWS-Ressourcen
- Schritt 5: Nächste Schritte

Komponenten der Anwendung Managed Service für Apache Flink

Zur Verarbeitung von Daten verwendet Ihre Managed Service für Apache Flink-Anwendung eine Java/Apache Maven- oder Scala-Anwendung, die mithilfe der Apache Flink-Laufzeit Eingaben verarbeitet und Ausgaben erzeugt.

Eine Anwendung Managed Service für Apache Flink besteht aus folgenden Komponenten:

- Runtime-Eigenschaften: Sie können Runtime-Eigenschaften verwenden, um Ihre Anwendung zu konfigurieren, ohne Ihren Anwendungscode neu kompilieren zu müssen.
- Quelle: Die Anwendung verwendet Daten mithilfe einer Quelle. Ein Quell-Connector liest Daten aus einem Kinesis Data Stream, einem Amazon S3-Bucket usw. Weitere Informationen finden Sie unter Quellen.
- Operatoren: Die Anwendung verarbeitet Daten mithilfe eines oder mehrerer Operatoren. Ein Operator kann Daten transformieren, anreichern oder aggregieren. Weitere Informationen finden Sie unter DataStream-API Operatoren.
- Senke: Die Anwendung erzeugt Daten für externe Quellen mithilfe von Senken. Ein Senken-Connector schreibt Daten in einen Kinesis Data Stream, einen Kinesis Data Firehose-Stream, einen Amazon S3-Bucket usw. Weitere Informationen finden Sie unter Senken.

Nachdem Sie Ihren Anwendungscode erstellt, kompiliert und verpackt haben, laden Sie das Codepaket in einen Amazon Simple Storage Service (Amazon S3)-Bucket hoch. Anschließend erstellen Sie eine Managed Service für Apache Flink-Anwendung. Sie übergeben den Speicherort des Codepakets, einen Kinesis Data Stream als Streaming-Datenquelle und in der Regel einen Streaming- oder Dateispeicherort, der die verarbeiteten Daten der Anwendung empfängt.

Voraussetzungen für das Fertigstellen der Übungen

Zur Durchführung der Schritte in dieser Anleitung benötigen Sie Folgendes:

- Java Development Kit (JDK), Version 11. Legen Sie die `JAVA_HOME` Umgebungsvariable so fest, dass sie auf Ihren JDK-Installationsspeicherort weist.
- Wir empfehlen die Verwendung einer Entwicklungsumgebung (wie Eclipse Java Neon oder IntelliJ Idea), um Ihre Anwendung zu entwickeln und zu kompilieren.
- Git-Client. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben.
- Apache Maven-Compiler-Plugin. Maven muss sich in Ihrem Arbeitspfad befinden. Zum Testen Ihrer Apache Maven-Installation geben Sie Folgendes ein:

```
$ mvn -version
```

Um zu beginnen, gehen Sie zu Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Administratorbenutzers.

Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Administratorbenutzers

So melden Sie sich für ein AWS-Konto an

Wenn Sie kein AWS-Konto haben, führen Sie die folgenden Schritte zum Erstellen durch.

Anmeldung für ein AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für ein AWS-Konto anmelden, wird ein Root-Benutzer des AWS-Kontos erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen eine Bestätigungs-E-Mail, sobald die Anmeldung abgeschlossen ist. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Erstellen eines Administratorbenutzers

Nachdem Sie sich für ein AWS-Kontoangemeldet haben, sichern Sie Ihr Root-Benutzer des AWS-Kontos, aktivieren Sie AWS IAM Identity Centerund erstellen Sie einen administrativen Benutzer, damit Sie nicht den Root-Benutzer für alltägliche Aufgaben verwenden.

Schützen Ihres Root-Benutzer des AWS-Kontos

1. Melden Sie sich bei der AWS Management Console als Kontobesitzer an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto-E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-AnmeldungBenutzerhandbuch zu .

2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen dazu finden Sie unter Aktivieren eines virtuellen MFA-Geräts für den Root-Benutzer Ihres AWS-Konto (Konsole) im IAM-Benutzerhandbuch.

Erstellen eines Administratorbenutzers

1. Aktivieren von IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.

2. Im IAM Identity Center gewähren Sie einem administrativen Benutzer administrativen Zugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit dem standardmäßigen IAM-Identity-Center-Verzeichnis konfigurieren im AWS IAM Identity Center-Benutzerhandbuch.

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM-Identity-Center-Benutzer finden Sie unter Anmelden beim AWS-Zugangsportal im AWS-Anmeldung Benutzerhandbuch zu.

Erteilen programmgesteuerten Zugriffs

Benutzer benötigen programmgesteuerten Zugriff, wenn sie außerhalb der AWS Management Console mit AWS interagieren möchten. Die Vorgehensweise, um programmgesteuerten Zugriff zu gewähren, hängt davon ab, welcher Benutzertyp auf zugreift AWS.

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zur AWS CLI finden Sie unter Konfigurieren der AWS CLI für die Verwendung von AWS IAM Identity Center im AWS Command Line Interface-Benutzerhandbuch. • Informationen zu AWS-SDKs, Tools und AWS-APIs finden Sie unter IAM-Identity-Center-Authentifizierung im Referenzhandbuch zu AWS-SDKs und Tools.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Folgen Sie den Anweisungen unter Verwenden temporärer Anmeldeinformationen mit AWS-Ressourcen im IAM-Benutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zur AWS CLI finden Sie unter Authentifizierung mit IAM-Benutzer-Anmeldeinformationen im AWS Command

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
		Line Interface-Benutzer handbuch. • Informationen zu AWS-SDKs und Tools finden Sie unter Authentifizierung mit langfristigen Anmeldein formationen im Referenzh andbuch zu AWS-SDKs und Tools. • Informationen zu AWS-APIs finden Sie unter Verwalten von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benut zerhandbuch.

Nächster Schritt

Schritt 2: Einrichten der AWS Command Line Interface (AWS CLI)

Schritt 2: Einrichten der AWS Command Line Interface (AWS CLI)

In diesem Schritt laden Sie die AWS CLI herunter und konfigurieren sie für die Verwendung mit Amazon Managed Service für Apache Flink.

> **Note**
>
> Bei allen Erste-Schritte-Übungen in diesem Handbuch wird davon ausgegangen, dass Sie in Ihrem Konto Administrator-Anmeldeinformationen (`adminuser`) verwenden, um die Operationen auszuführen.

> **Note**
>
> Wenn Sie die AWS CLI bereits installiert haben, müssen Sie möglicherweise ein Upgrade durchführen, um die neueste Funktionalität zu erhalten. Weitere Informationen finden Sie unter Installieren der AWS Command Line Interface im AWS Command Line Interface-Benutzerhandbuch. Zum Überprüfen der Version der AWS CLI führen Sie den folgenden Befehl aus:
>
> ```
> aws --version
> ```
>
> Die Übungen in diesem Tutorial erfordern die folgende AWS CLI-Version oder höher:
>
> ```
> aws-cli/1.16.63
> ```

Um das AWS CLI einzurichten

1. Herunterladen und Konfigurieren von AWS CLI. Eine Anleitung finden Sie unter den folgenden Themen im AWS Command Line Interface-Benutzerhandbuch:
   - Installieren des AWS Command Line Interface
   - Konfigurieren von AWS CLI
2. Fügen Sie ein benanntes Profil für den Administratorbenutzer in der AWS CLI `config`-Datei hinzu. Verwenden Sie dieses Profil beim Ausführen von AWS CLI-Befehlen. Weitere Informationen zu benannten Profilen finden Sie unter Benannte Profile im AWS Command Line Interface Benutzerhandbuch.

   ```
   [profile adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   region = aws-region
   ```

   Eine Liste der verfügbaren AWS-Regionen finden Sie unter Regionen und Endpunkte im Allgemeine Amazon Web Services-Referenz

> **Note**
>
> Der Beispielcode und die Befehle in diesem Tutorial verwenden die Region USA West (Oregon). Um eine andere Region zu verwenden, ändern Sie die Region im Code und in den Befehlen für dieses Tutorial in die Region, die Sie verwenden möchten.

3. Überprüfen Sie die Einrichtung, indem Sie die folgenden Hilfebefehle in die Befehlszeile eingeben:

```
aws help
```

Nachdem Sie ein -AWSKonto und die eingerichtet habenAWS CLI, können Sie die nächste Übung ausprobieren, in der Sie eine Beispielanwendung konfigurieren und die end-to-end Einrichtung testen.

Nächster Schritt

Schritt 3: Erstellen und Ausführen eines Managed Service für Apache Flink-Anwendung

Schritt 3: Erstellen und Ausführen eines Managed Service für Apache Flink-Anwendung

In dieser Übung erstellen Sie eine Anwendung von Managed Service für Apache Flink mit Datenströmen als Quelle und Senke.

Dieser Abschnitt enthält die folgenden Schritte:

- Erstellen von zwei Amazon Kinesis Datenstroms
- Schreiben Sie Beispieldatensätze in den Eingabe-Stream
- Herunterladen und Überprüfen des Apache Flink-Streaming-Java-Codes
- Kompilieren des Anwendungscodes
- Hochladen des Apache Flink-Streaming-Java-Codes
- Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus
- Nächster Schritt

Erstellen von zwei Amazon Kinesis Datenstroms

Bevor Sie für diese Übung eine Anwendung von Managed Service für Apache Flink erstellen, erstellen Sie zwei Kinesis Data Streams (`ExampleInputStream` und `ExampleOutputStream`). Ihre Anwendung verwendet diese Streams für die Quell- und Ziel-Streams der Anwendung.

Sie können diese Streams mithilfe der Amazon-Kinesis-Konsole oder des folgenden AWS CLI-Befehls erstellen. Anweisungen für die Konsole finden Sie unter Erstellen und Aktualisieren von Datenströmen im Amazon Kinesis Data Streams Entwicklerhandbuch.

So erstellen Sie die Daten-Streams (AWS CLI)

1. Verwenden Sie den Befehl Amazon Kinesis `create-stream` AWS CLI, um den ersten Stream (`ExampleInputStream`) zu erstellen.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Um den zweiten Stream zu erstellen, den die Anwendung zum Schreiben der Ausgabe verwendet, führen Sie denselben Befehl aus und ändern den Stream-Namen in `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Abschnitt verwenden Sie ein Python-Skript zum Schreiben von Datensätzen in den Stream für die zu verarbeitende Anwendung.

> **Note**
>
> Dieser Abschnitt erfordert AWS SDK for Python (Boto).

1. Erstellen Sie eine Datei `stock.py` mit dem folgenden Inhalt:

```
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. Im weiteren Verlauf des Tutorials führen Sie das `stock.py`-Skript zum Senden von Daten an die Anwendung aus.

```
$ python stock.py
```

Herunterladen und Überprüfen des Apache Flink-Streaming-Java-Codes

Der Java-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Klonen Sie das Remote-Repository, indem Sie den folgenden Befehl verwenden:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/GettingStarted` Verzeichnis .

Beachten Sie Folgendes zum Anwendungscode:

- Eine Project Object Model (pom.xml) Datei enthält Informationen über die Konfiguration und Abhängigkeiten der Anwendung, einschließlich der Bibliotheken des Managed Service für Apache Flink.
- Die `BasicStreamingJob.java`-Datei enthält die `main`-Methode, die die Funktionalität der Anwendung definiert.
- Die Anwendung verwendet eine Kinesis-Quelle zum Lesen aus dem Quell-Stream. Der folgende Codeausschnitt erstellt die Kinesis-Quelle:

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

- Ihre Anwendung erstellt Quell- und Senkenkonnektoren für den Zugriff auf externe Ressourcen, indem ein `StreamExecutionEnvironment`-Objekt verwendet wird.
- Die Anwendung erstellt Quell- und Senkenkonnektoren mit statischen Eigenschaften. Zum Verwenden dynamischer Anwendungseigenschaften verwenden Sie die Methoden `createSourceFromApplicationProperties` und `createSinkFromApplicationProperties`, um die Konnektoren zu erstellen. Diese Methoden lesen die Eigenschaften der Anwendung zum Konfigurieren der Konnektoren.

  Weitere Informationen zu den Laufzeiteigenschaften finden Sie unter Laufzeiteigenschaften.

Kompilieren des Anwendungscodes

In diesem Abschnitt verwenden Sie den Apache Maven-Compiler zum Erstellen des Java-Codes für die Anwendung. Weitere Informationen zum Installieren von Apache Maven und des Java Development Kit (JDK) finden Sie unter Voraussetzungen für das Fertigstellen der Übungen.

Kompilieren des Anwendungscodes

1. Zum Verwenden Ihres Anwendungscodes kompilieren und packen Sie ihn in eine JAR-Datei. Sie können Ihren Code auf zwei Arten kompilieren und packen:

   - Verwenden Sie das Befehlszeilen-Maven-Tool. Erstellen Sie Ihre JAR-Datei, indem Sie den folgenden Befehl in dem Verzeichnis ausführen, das die `pom.xml`-Datei enthält:

     ```
     mvn package -Dflink.version=1.11.3
     ```

   - Verwenden Sie Ihre Entwicklungsumgebung. Weitere Informationen finden Sie in der Dokumentation Ihrer Entwicklungsumgebung.

     > **Note**
     >
     > Der bereitgestellte Quellcode basiert auf Bibliotheken aus Java 11. Stellen Sie sicher, dass die Java-Version Ihres Projekts 11 ist.

   Sie können Ihr Paket als JAR-Datei hochladen oder komprimieren und als ZIP-Datei hochladen. Wenn Sie Ihre Anwendung mit der AWS CLI erstellen, geben Sie Ihren Codeinhaltstyp (JAR oder ZIP) an.

2. Wenn während der Erstellung Fehler aufgetreten sind, überprüfen Sie, ob Ihre `JAVA_HOME`-Umgebungsvariable richtig eingestellt ist.

Wenn die Anwendung erfolgreich kompiliert wurde, wird die folgende Datei erstellt:

`target/aws-kinesis-analytics-java-apps-1.0.jar`

Hochladen des Apache Flink-Streaming-Java-Codes

In diesem Abschnitt erstellen Sie einen Amazon Simple Storage Service (Amazon S3)-Bucket und laden Ihren Anwendungscode hoch.

So laden Sie den Anwendungscode hoch

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie Bucket erstellen aus.

3. Geben Sie **`ka-app-code-<username>`** im Feld Bucket-Name ein. Fügen Sie dem Bucket-Namen ein Suffix hinzu, wie z. B. Ihren Benutzernamen, damit er global eindeutig ist. Wählen Sie Weiter aus.
4. Lassen Sie im Schritt Optionen konfigurieren die Einstellungen unverändert und klicken Sie auf Weiter.
5. Lassen Sie im Schritt Berechtigungen festlegen die Einstellungen unverändert und klicken Sie auf Weiter.
6. Wählen Sie Bucket erstellen aus.
7. Wählen Sie in der Amazon S3-Konsole den Bucket ka-app-code-***<username>*** und dann Hochladen aus.
8. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `aws-kinesis-analytics-java-apps-1.0.jar`Datei, die Sie im vorherigen Schritt erstellt haben. Wählen Sie Weiter aus.
9. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus

Sie können eine Anwendung von Managed Service für Apache Flink entweder über die Konsole oder die AWS CLI erstellen und ausführen.

> **Note**
>
> Wenn Sie die Anwendung mit der Konsole erstellen, werden Ihre AWS Identity and Access Management (IAM)- und Amazon- CloudWatch Logs-Ressourcen für Sie erstellt. Wenn Sie die Anwendung mit der AWS CLI erstellen, erstellen Sie diese Ressourcen separat.

Themen

- Erstellen und Ausführen der Anwendung (Konsole)
- Erstellen und Ausführen der Anwendung (AWS CLI)

Erstellen und Ausführen der Anwendung (Konsole)

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Geben Sie für Beschreibung den Text **`My java test app`** ein.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.
   - Behalten Sie im Pulldown-Menü für die Version Apache Flink Version 1.11 (empfohlene Version) bei.
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rolle: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf die Kinesis-Datenströme.

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **`kinesis-analytics-service-MyApplication-us-west-2`**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*012345678901*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):

- Geben Sie für Amazon-S3-Bucket **ka-app-code-*<username>*** ein.
- Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **aws-kinesis-analytics-java-apps-1.0.jar** ein.

3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **kinesis-analytics-MyApplication-us-west-2** erstellen/aktualisieren aus.
4. Geben Sie unter Eigenschaften für Gruppen-ID den Text **ProducerConfigProperties** ein.
5. Geben Sie die folgenden Eigenschaften und Werte der Anwendung ein:

Gruppen-ID	Schlüssel	Wert
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**
ProducerConfigProperties	**AggregationEnabled**	**false**

6. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
7. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
8. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die Amazon- CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`

Ausführen der Anwendung

Das Flink-Jobdiagramm kann angezeigt werden, indem Sie die Anwendung ausführen, das Apache Flink-Dashboard öffnen und den gewünschten Flink-Job auswählen.

Stoppen der Anwendung

Wählen Sie auf der MyApplication Seite Stoppen aus. Bestätigen Sie die Aktion.

Aktualisieren der Anwendung

Mithilfe der Konsole können Sie Anwendungseinstellungen wie beispielsweise Anwendungseigenschaften, Überwachungseinstellungen und den Speicherort oder den Dateinamen der JAR-Anwendungsdatei aktualisieren. Außerdem können Sie die JAR-Anwendungsdatei erneut aus dem Amazon-S3-Bucket laden, wenn Sie den Anwendungscode aktualisieren müssen.

Wählen Sie auf der MyApplication Seite Konfigurieren aus. Aktualisieren Sie die Anwendungseinstellungen und klicken Sie auf Aktualisieren.

Erstellen und Ausführen der Anwendung (AWS CLI)

In diesem Abschnitt verwenden Sie AWS CLI zum Erstellen und Ausführen der Anwendung Managed Service für Apache Flink. Ein Managed Service für apache Flink verwendet den Befehl `kinesisanalyticsv2` AWS CLI, um Managed-Service-für-Apache-Flink-Anwendungen zu erstellen und mit diesen zu interagieren.

Erstellen einer Berechtigungsrichtlinie

> **Note**
>
> Sie müssen eine Berechtigungsrichtlinie und eine Rolle für Ihre Anwendung erstellen. Wenn Sie diese IAM-Ressourcen nicht erstellen, kann Ihre Anwendung nicht auf ihre Daten und Protokollstreams zugreifen.

Zuerst erstellen Sie eine Berechtigungsrichtlinie mit zwei Anweisungen: eine, die Berechtigungen für die `read`-Aktion auf den Quell-Stream zulässt, und eine andere, die Berechtigungen für die `write`-Aktionen auf den Senken-Stream zulässt. Anschließend fügen Sie die Richtlinie an eine IAM-Rolle (die Sie im nächsten Abschnitt erstellen) an. Wenn Managed Service für Apache Flink also die Rolle

übernimmt, verfügt der Service über die erforderlichen Berechtigungen zum Lesen aus dem Quell-Stream und zum Schreiben in den Senken-Stream.

Verwenden Sie den folgenden Code zum Erstellen der `AKReadSourceStreamWriteSinkStream`-Berechtigungsrichtlinie. Ersetzen Sie *`username`* durch den Benutzernamen, den Sie verwendet haben, um den Amazon-S3-Bucket zum Speichern des Anwendungscodes zu erstellen. Ersetzen Sie die Konto-ID in den Amazon-Ressourcennamen (ARNs) (*`012345678901`*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

step-by-step Anweisungen zum Erstellen einer Berechtigungsrichtlinie finden Sie unter Tutorial: Erstellen und Anfügen Ihrer ersten vom Kunden verwalteten Richtlinie im IAM-Benutzerhandbuch.

> **Note**
>
> Um auf andere Amazon-Services zuzugreifen, können Sie AWS SDK for Java verwenden. Managed Service für Apache Flink setzt die vom SDK benötigten Anmeldeinformationen automatisch auf die der IAM-Rolle für die Dienstausführung, die mit Ihrer Anwendung verknüpft ist. Es sind keine weiteren Schritte erforderlich.

Erstellen einer IAM-Rolle

In diesem Abschnitt erstellen Sie eine IAM-Rolle, die die Anwendung von Managed Service für Apache Flink annehmen kann, um einen Quell-Stream zu lesen und in den Senken-Stream zu schreiben.

Managed Service für Apache Flink kann ohne Berechtigungen nicht auf Ihren Stream zugreifen. Sie erteilen diese Berechtigungen über eine IAM-Rolle. Jeder IAM-Rolle sind zwei Richtlinien angefügt. Die Vertrauensrichtlinie erteilt Managed Service für Apache Flink die Berechtigung zum Übernehmen der Rolle und die Berechtigungsrichtlinie bestimmt, was Managed Service für Apache Flink nach Annahme der Rolle tun kann.

Sie können die Berechtigungsrichtlinie, die Sie im vorherigen Abschnitt erstellt haben, dieser Rolle anfügen.

So erstellen Sie eine IAM-Rolle

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Roles (Rollen) und Create Role (Rolle erstellen) aus.
3. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option AWS-Service aus. Wählen Sie unter Choose the service that will use this role (Wählen Sie den Service aus, der diese Rolle verwendet) die Option Kinesis aus. Wählen Sie unter Select your use case (Wählen Sie Ihren Anwendungsfall aus) die Option Kinesis Analytics aus.

   Wählen Sie Weiter: Berechtigungen aus.
4. Wählen Sie auf der Seite Attach permissions policies (Berechtigungsrichtlinien hinzufügen) Next: Review (Weiter: Überprüfen) aus. Sie fügen Berechtigungsrichtlinien an, nachdem Sie die Rolle erstellt haben.
5. Geben Sie auf der Seite Create role (Rolle erstellen) den Text **`MF-stream-rw-role`** für Role name (Rollenname) ein. Wählen Sie Rolle erstellen aus.

Jetzt haben Sie eine neue IAM-Rolle mit dem Namen `MF-stream-rw-role` erstellt. Im nächsten Schritt aktualisieren Sie die Vertrauens- und Berechtigungsrichtlinien für die Rolle.

6. Fügen Sie die Berechtigungsrichtlinie der Rolle an.

> **Note**
>
> Für diese Übung übernimmt Managed Service für Apache Flink diese Rolle sowohl für das Lesen von Daten aus einem Kinesis-Datenstrom (Quelle) als auch zum Schreiben der Ausgabedaten in einen anderen Kinesis-Datenstrom. Daher fügen Sie die Richtlinie an, die Sie im vorherigen Schritt erstellt haben, the section called “Erstellen einer Berechtigungsrichtlinie”.

   a. Wählen Sie auf der Seite Summary (Übersicht) die Registerkarte Permissions (Berechtigungen) aus.
   b. Wählen Sie Attach Policies (Richtlinien anfügen) aus.
   c. Geben Sie im Suchfeld **`AKReadSourceStreamWriteSinkStream`** (die Richtlinie, die Sie im vorhergehenden Abschnitt erstellt haben) ein.
   d. Wählen Sie die AKReadSourceStreamWriteSinkStream-Richtlinie und dann Richtlinie anfügen aus.

Sie haben nun die Service-Ausführungsrolle erstellt, die Ihre Anwendung für den Zugriff auf Ressourcen verwendet. Notieren Sie sich den ARN der neuen Rolle.

step-by-step Anweisungen zum Erstellen einer Rolle finden Sie unter Erstellen einer IAM-Rolle (Konsole) im IAM-Benutzerhandbuch.

Erstellen Sie die Anwendung Managed Service für Apache Flink

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `create_request.json`. Ersetzen Sie den Beispiel-Rollen-ARN durch den ARN für die Rolle, die Sie zuvor erstellt haben. Ersetzen Sie das Bucket-ARN-Suffix (*`username`*) mit dem Suffix, das Sie im vorherigen Abschnitt gewählt haben. Ersetzen Sie die beispielhafte Konto-ID (*`012345678901`*) in der Service-Ausführungsrolle mit Ihrer Konto-ID.

```
{
    "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. Führen Sie die `CreateApplication`-Aktion mit der vorherigen Anforderung zum Erstellen der Anwendung aus:

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

Die Anwendung wird nun erstellt. Sie starten die Anwendung im nächsten Schritt.

Starten der Anwendung

In diesem Abschnitt verwenden Sie die `StartApplication`-Aktion, um die Anwendung zu starten.

So starten Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `start_request.json`.

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

2. Führen Sie die `StartApplication`-Aktion mit der vorherigen Anforderung zum Starten der Anwendung aus:

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

Die Anwendung wird jetzt ausgeführt. Sie können die Metriken von Managed Service für Apache Flink in der Amazon- CloudWatch Konsole überprüfen, um zu überprüfen, ob die Anwendung funktioniert.

Stoppen der Anwendung

In diesem Abschnitt verwenden Sie die `StopApplication`-Aktion, um die Anwendung zu stoppen.

So stoppen Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `stop_request.json`.

```
{
    "ApplicationName": "test"
}
```

2. Führen Sie die `StopApplication`-Aktion mit der folgenden Anforderung zum Stoppen der Anwendung aus:

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

Die Anwendung wird nun gestoppt.

Hinzufügen einer CloudWatch Protokollierungsoption

Sie können die verwendenAWS CLI, um Ihrer Anwendung einen Amazon- CloudWatch Protokollstream hinzuzufügen. Informationen zur Verwendung von - CloudWatch Protokollen mit Ihrer Anwendung finden Sie unter the section called “Einrichten der Protokollierung”.

Umgebungseigenschaften aktualisieren

In diesem Abschnitt verwenden Sie die `UpdateApplication`-Aktion, um die Umgebungseigenschaften für die Anwendung zu ändern, ohne den Anwendungscode neu kompilieren zu müssen. In diesem Beispiel ändern Sie die Region der Quell- und Ziel-Streams.

So aktualisieren Sie die Umgebungseigenschaften für die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `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. Führen Sie die UpdateApplication-Aktion mit der vorherigen Anforderung aus, um die Umgebungseigenschaften zu aktualisieren:

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

Aktualisieren Sie den Anwendungscode

Wenn Sie Ihren Anwendungscode mit einer neuen Version Ihres Codepakets aktualisieren müssen, verwenden Sie die UpdateApplication AWS CLI-Aktion.

> **Note**
>
> Um eine neue Version des Anwendungscodes mit demselben Dateinamen zu laden, müssen Sie die neue Objektversion angeben. Weitere Informationen zur Verwendung von Amazon S3-Objektversionen finden Sie unter Versionierung aktivieren oder deaktivieren.

Um die AWS CLI zu verwenden, löschen Sie Ihr vorheriges Codepaket aus Ihrem Amazon S3-Bucket, laden Sie die neue Version hoch und rufen Sie `UpdateApplication` auf. Geben Sie dabei denselben Amazon S3-Bucket und Objektnamen sowie die neue Objektversion an. Die Anwendung wird mit dem neuen Codepaket neu gestartet.

Die folgende Beispielanforderung für die `UpdateApplication`-Aktion lädt den Anwendungscode neu und startet die Anwendung neu. Aktualisieren Sie die `CurrentApplicationVersionId` auf die aktuelle Anwendungsversion. Sie können die aktuelle Anwendungsversion mithilfe der Aktionen `ListApplications` oder `DescribeApplication` überprüfen. Aktualisieren Sie das Bucket-Namenssuffix (*<username>*) mit dem Suffix, das Sie im the section called “Erstellen von zwei Amazon Kinesis Datenstroms”-Abschnitt ausgewählt haben.

```
{
    "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"
                }
            }
        }
    }
}
```

Nächster Schritt

[Schritt 4: Bereinigen von AWS-Ressourcen]

Schritt 4: Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zum Bereinigen von AWS-Ressourcen, die im Tutorial Erste Schritte erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte:

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Sie Ihre Kinesis Data Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen
- Nächster Schritt

Löschen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie dann den Löschvorgang.

Löschen Sie Ihre Kinesis Data Streams

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Bereich Kinesis Data Streams die Option ausExampleInputStream.
3. Wählen Sie auf der ExampleInputStream Seite Kinesis Stream löschen aus und bestätigen Sie dann den Löschvorgang.
4. Wählen Sie auf der Seite Kinesis Streams die , ExampleOutputStreamwählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den ka-app-codeBucket -***<username>*** aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle kinesis-analytics--MyApplication-us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Nächster Schritt

Schritt 5: Nächste Schritte

Schritt 5: Nächste Schritte

Nachdem Sie nun eine grundlegende Managed Service für Apache Flink-Anwendung erstellt und ausgeführt haben, finden Sie in den folgenden Ressourcen erweiterte Managed Service für Apache Flink-Lösungen.

- Die AWS Streaming Data Solution for Amazon Kinesis : Die AWS Streaming Data Solution for Amazon Kinesis konfiguriert automatisch die AWS-Services, die zum einfachen Erfassen, Speichern, Verarbeiten und Bereitstellen von Streaming-Daten erforderlich sind. Die Lösung bietet mehrere Optionen zur Lösung von Anwendungsfällen mit Streaming-Daten. Die Option Managed Service für Apache Flink bietet ein end-to-end -Streaming-ETL-Beispiel, das eine reale Anwendung demonstriert, die analytische Operationen für simulierte New York-Taxisdaten ausführt. Die Lösung richtet alle erforderlichen AWS Ressourcen wie IAM-Rollen und -Richtlinien, ein CloudWatch Dashboard und CloudWatch Alarme ein.
- AWS Streaming Data Solution for Amazon MSK: Die AWS Streaming Data Solution for Amazon MSK bietet AWS CloudFormation-Vorlagen, in denen Daten durch Produzenten, Streaming-Speicher, Verbraucher und Ziele fließen.
- Clickstream Lab mit Apache Flink und Apache Kafka: Ein End-to-End-Lab für Clickstream-Anwendungsfälle mit Amazon Managed Streaming for Apache Kafka als Streaming-Speicher und Managed Service für Apache Flink für Apache Flink-Anwendungen zur Stream-Verarbeitung.
- Workshop für Amazon Managed Service für Apache Flink: In diesem Workshop erstellen Sie eine - end-to-end Streaming-Architektur, um Streaming-Daten nahezu in Echtzeit aufzunehmen, zu analysieren und zu visualisieren. Sie haben sich vorgenommen, den Betrieb eines Taxiunternehmens in New York City zu verbessern. Sie analysieren die Telemetriedaten einer Taxiflotte in New York City nahezu in Echtzeit, um deren Flottenbetrieb zu optimieren.
- Managed Service für Apache Flink: Beispiele: Dieser Abschnitt dieses Entwicklerhandbuchs enthält Beispiele für die Erstellung von und die Arbeit mit Anwendungen in Managed Service für Apache Flink. Sie enthalten Beispielcode und step-by-step Anweisungen, mit denen Sie Managed Service für Apache Flink-Anwendungen erstellen und Ihre Ergebnisse testen können.
- Lernen Sie Flink kennen: Praktisches Training: Offizielle Apache Flink-Einführungsschulung, die Ihnen den Einstieg in die Entwicklung skalierbarer Streaming-ETL-, Analyse- und ereignisgesteuerter Anwendungen ermöglicht.

> **Note**
>
> Beachten Sie, dass Managed Service für Apache Flink die in dieser Schulung verwendete Apache Flink-Version (1.12) nicht unterstützt. Sie können Flink 1.15.2 in Flink Managed Service für Apache Flink verwenden.

- Apache-Flink-Codebeispiele: Ein GitHub Repository mit einer Vielzahl von Apache-Flink-Anwendungsbeispielen.

Erste Schritte: Flink 1.8.2

Dieses Thema enthält eine Version des Erste Schritte (DataStream API)-Tutorials, die Apache Flink 1.8.2 verwendet.

Themen

- Komponenten der Anwendung Managed Service für Apache Flink
- Voraussetzungen für das Fertigstellen der Übungen
- Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Administratorbenutzers
- Schritt 2: Einrichten der AWS Command Line Interface (AWS CLI)
- Schritt 3: Erstellen und Ausführen eines Managed Service für Apache Flink-Anwendung
- Schritt 4: Bereinigen von AWS-Ressourcen

Komponenten der Anwendung Managed Service für Apache Flink

Zur Verarbeitung von Daten verwendet Ihre Managed Service für Apache Flink-Anwendung eine Java/Apache Maven- oder Scala-Anwendung, die mithilfe der Apache Flink-Laufzeit Eingaben verarbeitet und Ausgaben erzeugt.

Eine Anwendung Managed Service für Apache Flink besteht aus folgenden Komponenten:

- Runtime-Eigenschaften: Sie können Runtime-Eigenschaften verwenden, um Ihre Anwendung zu konfigurieren, ohne Ihren Anwendungscode neu kompilieren zu müssen.
- Quelle: Die Anwendung verwendet Daten mithilfe einer Quelle. Ein Quell-Connector liest Daten aus einem Kinesis Data Stream, einem Amazon S3-Bucket usw. Weitere Informationen finden Sie unter Quellen.

- Operatoren: Die Anwendung verarbeitet Daten mithilfe eines oder mehrerer Operatoren. Ein Operator kann Daten transformieren, anreichern oder aggregieren. Weitere Informationen finden Sie unter DataStream-API Operatoren.
- Senke: Die Anwendung erzeugt Daten für externe Quellen mithilfe von Senken. Ein Senken-Connector schreibt Daten in einen Kinesis Data Stream, einen Kinesis Data Firehose-Stream, einen Amazon S3-Bucket usw. Weitere Informationen finden Sie unter Senken.

Nachdem Sie Ihren Anwendungscode erstellt, kompiliert und verpackt haben, laden Sie das Codepaket in einen Amazon Simple Storage Service (Amazon S3)-Bucket hoch. Anschließend erstellen Sie eine Managed Service für Apache Flink-Anwendung. Sie übergeben den Speicherort des Codepakets, einen Kinesis Data Stream als Streaming-Datenquelle und in der Regel einen Streaming- oder Dateispeicherort, der die verarbeiteten Daten der Anwendung empfängt.

Voraussetzungen für das Fertigstellen der Übungen

Zur Durchführung der Schritte in dieser Anleitung benötigen Sie Folgendes:

- Java Development Kit (JDK), Version 8. Legen Sie die `JAVA_HOME` Umgebungsvariable so fest, dass sie auf Ihren JDK-Installationsspeicherort weist.
- Um den Apache Flink Kinesis Connector in diesem Tutorial verwenden zu können, müssen Sie Apache Flink herunterladen und installieren. Details hierzu finden Sie unter Verwenden des Apache Flink Kinesis Streams-Konnektor mit früheren Apache Flink-Versionen.
- Wir empfehlen die Verwendung einer Entwicklungsumgebung (wie Eclipse Java Neon oder IntelliJ Idea), um Ihre Anwendung zu entwickeln und zu kompilieren.
- Git-Client. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben.
- Apache Maven-Compiler-Plugin. Maven muss sich in Ihrem Arbeitspfad befinden. Zum Testen Ihrer Apache Maven-Installation geben Sie Folgendes ein:

```
$ mvn -version
```

Um zu beginnen, gehen Sie zu Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Administratorbenutzers.

Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Administratorbenutzers

So melden Sie sich für ein AWS-Konto an

Wenn Sie kein AWS-Konto haben, führen Sie die folgenden Schritte zum Erstellen durch.

Anmeldung für ein AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für ein AWS-Konto anmelden, wird ein Root-Benutzer des AWS-Kontos erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen eine Bestätigungs-E-Mail, sobald die Anmeldung abgeschlossen ist. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Erstellen eines Administratorbenutzers

Nachdem Sie sich für ein AWS-Kontoangemeldet haben, sichern Sie Ihr Root-Benutzer des AWS-Kontos, aktivieren Sie AWS IAM Identity Centerund erstellen Sie einen administrativen Benutzer, damit Sie nicht den Root-Benutzer für alltägliche Aufgaben verwenden.

Schützen Ihres Root-Benutzer des AWS-Kontos

1. Melden Sie sich bei der AWS Management Console als Kontobesitzer an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto-E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-AnmeldungBenutzerhandbuch zu .

2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen dazu finden Sie unter Aktivieren eines virtuellen MFA-Geräts für den Root-Benutzer Ihres AWS-Konto (Konsole) im IAM-Benutzerhandbuch.

Erstellen eines Administratorbenutzers

1. Aktivieren von IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.
2. Im IAM Identity Center gewähren Sie einem administrativen Benutzer administrativen Zugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit dem standardmäßigen IAM-Identity-Center-Verzeichnis konfigurieren im AWS IAM Identity Center-Benutzerhandbuch.

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM-Identity-Center-Benutzer finden Sie unter Anmelden beim AWS-Zugangsportal im AWS-Anmeldung Benutzerhandbuch zu.

Erteilen programmgesteuerten Zugriffs

Benutzer benötigen programmgesteuerten Zugriff, wenn sie außerhalb der AWS Management Console mit AWS interagieren möchten. Die Vorgehensweise, um programmgesteuerten Zugriff zu gewähren, hängt davon ab, welcher Benutzertyp auf zugreift AWS.

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zur AWS CLI finden Sie unter Konfigurieren der AWS CLI für die Verwendung von AWS IAM Identity Center im AWS Command Line Interface-Benutzerhandbuch. • Informationen zu AWS-SDKs, Tools und AWS-APIs finden Sie unter IAM-Identity-Center-Authentifizierung im Referenzhandbuch zu AWS-SDKs und Tools.
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Folgen Sie den Anweisungen unter Verwenden temporärer Anmeldeinformationen mit AWS-Ressourcen im IAM-Benutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zur AWS CLI finden Sie unter Authentifizierung mit IAM-Benutzer-Anmeldeinformationen im AWS Command

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
		Line Interface-Benutzer handbuch. • Informationen zu AWS-SDKs und Tools finden Sie unter Authentifizierung mit langfristigen Anmeldeinformationen im Referenzhandbuch zu AWS-SDKs und Tools. • Informationen zu AWS-APIs finden Sie unter Verwalten von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch.

Schritt 2: Einrichten der AWS Command Line Interface (AWS CLI)

In diesem Schritt laden Sie die AWS CLI herunter und konfigurieren sie für die Verwendung mit Amazon Managed Service für Apache Flink.

> **Note**
>
> Bei allen Erste-Schritte-Übungen in diesem Handbuch wird davon ausgegangen, dass Sie in Ihrem Konto Administrator-Anmeldeinformationen (`adminuser`) verwenden, um die Operationen auszuführen.

> **Note**
>
> Wenn Sie die AWS CLI bereits installiert haben, müssen Sie möglicherweise ein Upgrade durchführen, um die neueste Funktionalität zu erhalten. Weitere Informationen finden Sie unter Installieren der AWS Command Line Interface im AWS Command Line Interface-

Benutzerhandbuch. Zum Überprüfen der Version der AWS CLI führen Sie den folgenden Befehl aus:

```
aws --version
```

Die Übungen in diesem Tutorial erfordern die folgende AWS CLI-Version oder höher:

```
aws-cli/1.16.63
```

Um das AWS CLI einzurichten

1. Herunterladen und Konfigurieren von AWS CLI. Eine Anleitung finden Sie unter den folgenden Themen im AWS Command Line Interface-Benutzerhandbuch:
   - Installieren des AWS Command Line Interface
   - Konfigurieren von AWS CLI
2. Fügen Sie ein benanntes Profil für den Administratorbenutzer in der AWS CLI `config`-Datei hinzu. Verwenden Sie dieses Profil beim Ausführen von AWS CLI-Befehlen. Weitere Informationen zu benannten Profilen finden Sie unter Benannte Profile im AWS Command Line Interface Benutzerhandbuch.

   ```
   [profile adminuser]
   aws_access_key_id = adminuser access key ID
   aws_secret_access_key = adminuser secret access key
   region = aws-region
   ```

   Eine Liste der verfügbaren Regionen finden Sie unter Regionen und Endpunkte in Allgemeine Amazon Web Services-Referenz.

   > **Note**
   >
   > Der Beispielcode und die Befehle in diesem Tutorial verwenden die Region USA West (Oregon). Um eine andere AWS-Region zu verwenden, ändern Sie die Region im Code und in den Befehlen für dieses Tutorial in die Region, die Sie verwenden möchten.

3. Überprüfen Sie die Einrichtung, indem Sie die folgenden Hilfebefehle in die Befehlszeile eingeben:

```
aws help
```

Nachdem Sie ein AWS Konto und die eingerichtet habenAWS CLI, können Sie die nächste Übung ausprobieren, in der Sie eine Beispielanwendung konfigurieren und die end-to-end Einrichtung testen.

Nächster Schritt

Schritt 3: Erstellen und Ausführen eines Managed Service für Apache Flink-Anwendung

Schritt 3: Erstellen und Ausführen eines Managed Service für Apache Flink-Anwendung

In dieser Übung erstellen Sie eine Anwendung von Managed Service für Apache Flink mit Datenströmen als Quelle und Senke.

Dieser Abschnitt enthält die folgenden Schritte:

- Erstellen von zwei Amazon Kinesis Datenstroms
- Schreiben Sie Beispieldatensätze in den Eingabe-Stream
- Herunterladen und Überprüfen des Apache Flink-Streaming-Java-Codes
- Kompilieren des Anwendungscodes
- Hochladen des Apache Flink-Streaming-Java-Codes
- Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus
- Nächster Schritt

Erstellen von zwei Amazon Kinesis Datenstroms

Bevor Sie für diese Übung eine Anwendung von Managed Service für Apache Flink erstellen, erstellen Sie zwei Kinesis Data Streams (`ExampleInputStream` und `ExampleOutputStream`). Ihre Anwendung verwendet diese Streams für die Quell- und Ziel-Streams der Anwendung.

Sie können diese Streams mithilfe der Amazon-Kinesis-Konsole oder des folgenden AWS CLI-Befehls erstellen. Anweisungen für die Konsole finden Sie unter Erstellen und Aktualisieren von Datenströmen im Amazon Kinesis Data Streams Entwicklerhandbuch.

So erstellen Sie die Daten-Streams (AWS CLI)

1. Verwenden Sie den Befehl Amazon Kinesis `create-stream` AWS CLI, um den ersten Stream (`ExampleInputStream`) zu erstellen.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Um den zweiten Stream zu erstellen, den die Anwendung zum Schreiben der Ausgabe verwendet, führen Sie denselben Befehl aus und ändern den Stream-Namen in `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Abschnitt verwenden Sie ein Python-Skript zum Schreiben von Datensätzen in den Stream für die zu verarbeitende Anwendung.

> **Note**
>
> Dieser Abschnitt erfordert AWS SDK for Python (Boto).

1. Erstellen Sie eine Datei `stock.py` mit dem folgenden Inhalt:

```
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. Im weiteren Verlauf des Tutorials führen Sie das `stock.py`-Skript zum Senden von Daten an die Anwendung aus.

```
$ python stock.py
```

Herunterladen und Überprüfen des Apache Flink-Streaming-Java-Codes

Der Java-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Klonen Sie das Remote-Repository, indem Sie den folgenden Befehl verwenden:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/GettingStarted_1_8` Verzeichnis .

Beachten Sie Folgendes zum Anwendungscode:

- Eine Project Object Model (pom.xml) Datei enthält Informationen über die Konfiguration und Abhängigkeiten der Anwendung, einschließlich der Bibliotheken des Managed Service für Apache Flink.
- Die `BasicStreamingJob.java`-Datei enthält die `main`-Methode, die die Funktionalität der Anwendung definiert.
- Die Anwendung verwendet eine Kinesis-Quelle zum Lesen aus dem Quell-Stream. Der folgende Codeausschnitt erstellt die Kinesis-Quelle:

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

- Ihre Anwendung erstellt Quell- und Senkenkonnektoren für den Zugriff auf externe Ressourcen, indem ein `StreamExecutionEnvironment`-Objekt verwendet wird.
- Die Anwendung erstellt Quell- und Senkenkonnektoren mit statischen Eigenschaften. Zum Verwenden dynamischer Anwendungseigenschaften verwenden Sie die Methoden `createSourceFromApplicationProperties` und `createSinkFromApplicationProperties`, um die Konnektoren zu erstellen. Diese Methoden lesen die Eigenschaften der Anwendung zum Konfigurieren der Konnektoren.

  Weitere Informationen zu den Laufzeiteigenschaften finden Sie unter Laufzeiteigenschaften.

Kompilieren des Anwendungscodes

In diesem Abschnitt verwenden Sie den Apache Maven-Compiler zum Erstellen des Java-Codes für die Anwendung. Weitere Informationen zum Installieren von Apache Maven und des Java Development Kit (JDK) finden Sie unter Voraussetzungen für das Fertigstellen der Übungen.

> **Note**
>
> Um den Kinesis-Connector mit Versionen von Apache Flink vor 1.11 verwenden zu können, müssen Sie Apache Maven herunterladen, erstellen und installieren. Weitere Informationen finden Sie unter the section called “Verwenden des Apache Flink Kinesis Streams-Konnektor mit früheren Apache Flink-Versionen”.

Kompilieren des Anwendungscodes

1. Zum Verwenden Ihres Anwendungscodes kompilieren und packen Sie ihn in eine JAR-Datei. Sie können Ihren Code auf zwei Arten kompilieren und packen:

   - Verwenden Sie das Befehlszeilen-Maven-Tool. Erstellen Sie Ihre JAR-Datei, indem Sie den folgenden Befehl in dem Verzeichnis ausführen, das die `pom.xml`-Datei enthält:

     ```
     mvn package -Dflink.version=1.8.2
     ```

   - Verwenden Sie Ihre Entwicklungsumgebung. Weitere Informationen finden Sie in der Dokumentation Ihrer Entwicklungsumgebung.

     > **Note**
     >
     > Der bereitgestellte Quellcode basiert auf Bibliotheken aus Java 1.8. Stellen Sie sicher, dass die Java-Version Ihres Projekts 1.8 ist.

   Sie können Ihr Paket als JAR-Datei hochladen oder komprimieren und als ZIP-Datei hochladen. Wenn Sie Ihre Anwendung mit der AWS CLI erstellen, geben Sie Ihren Codeinhaltstyp (JAR oder ZIP) an.

2. Wenn während der Erstellung Fehler aufgetreten sind, überprüfen Sie, ob Ihre `JAVA_HOME`-Umgebungsvariable richtig eingestellt ist.

Wenn die Anwendung erfolgreich kompiliert wurde, wird die folgende Datei erstellt:

`target/aws-kinesis-analytics-java-apps-1.0.jar`

Hochladen des Apache Flink-Streaming-Java-Codes

In diesem Abschnitt erstellen Sie einen Amazon Simple Storage Service (Amazon S3)-Bucket und laden Ihren Anwendungscode hoch.

So laden Sie den Anwendungscode hoch

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie Bucket erstellen aus.

3. Geben Sie **`ka-app-code-<username>`** im Feld Bucket-Name ein. Fügen Sie dem Bucket-Namen ein Suffix hinzu, wie z. B. Ihren Benutzernamen, damit er global eindeutig ist. Wählen Sie Weiter aus.
4. Lassen Sie im Schritt Optionen konfigurieren die Einstellungen unverändert und klicken Sie auf Weiter.
5. Lassen Sie im Schritt Berechtigungen festlegen die Einstellungen unverändert und klicken Sie auf Weiter.
6. Wählen Sie Bucket erstellen aus.
7. Wählen Sie in der Amazon S3-Konsole den Bucket ka-app-code-*<username>* und dann Upload aus.
8. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `aws-kinesis-analytics-java-apps-1.0.jar`Datei, die Sie im vorherigen Schritt erstellt haben. Wählen Sie Weiter aus.
9. Sie müssen keine der Einstellungen für das Objekt ändern. Wählen Sie daher Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus

Sie können eine Anwendung von Managed Service für Apache Flink entweder über die Konsole oder die AWS CLI erstellen und ausführen.

> **Note**
>
> Wenn Sie die Anwendung mit der Konsole erstellen, werden Ihre AWS Identity and Access Management (IAM)- und Amazon- CloudWatch Logs-Ressourcen für Sie erstellt. Wenn Sie die Anwendung mit der AWS CLI erstellen, erstellen Sie diese Ressourcen separat.

Themen

- Erstellen und Ausführen der Anwendung (Konsole)
- Erstellen und Ausführen der Anwendung (AWS CLI)

Erstellen und Ausführen der Anwendung (Konsole)

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Geben Sie für Beschreibung den Text **`My java test app`** ein.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.
   - Behalten Sie im Pulldown-Menü die Option Apache Flink 1.8 (empfohlene Version) bei.
4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rolle: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf die Kinesis-Datenströme.

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **`kinesis-analytics-service-MyApplication-us-west-2`**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*012345678901*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):
   - Geben Sie für Amazon-S3-Bucket **ka-app-code-*<username>*** ein.
   - Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **aws-kinesis-analytics-java-apps-1.0.jar** ein.

3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **`kinesis-analytics-MyApplication-us-west-2`** erstellen/aktualisieren aus.
4. Geben Sie die folgenden Eigenschaften und Werte der Anwendung ein:

Gruppen-ID	Schlüssel	Wert
`ProducerConfigProperties`	**`flink.inputstream.initpos`**	**`LATEST`**
`ProducerConfigProperties`	**`aws.region`**	**`us-west-2`**
`ProducerConfigProperties`	**`AggregationEnabled`**	**`false`**

5. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
6. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
7. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die Amazon- CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`

Ausführen der Anwendung

1. Wählen Sie auf der MyApplication Seite Ausführen aus. Bestätigen Sie die Aktion.
2. Wenn die Anwendung ausgeführt wird, aktualisieren Sie die Seite. Die Konsole zeigt den Application graph (Anwendungs-Graph) an.

Stoppen der Anwendung

Wählen Sie auf der MyApplication Seite Stoppen aus. Bestätigen Sie die Aktion.

Aktualisieren der Anwendung

Mithilfe der Konsole können Sie Anwendungseinstellungen wie beispielsweise Anwendungseigenschaften, Überwachungseinstellungen und den Speicherort oder den Dateinamen der JAR-Anwendungsdatei aktualisieren. Außerdem können Sie die JAR-Anwendungsdatei erneut aus dem Amazon-S3-Bucket laden, wenn Sie den Anwendungscode aktualisieren müssen.

Wählen Sie auf der MyApplication Seite Konfigurieren aus. Aktualisieren Sie die Anwendungseinstellungen und klicken Sie auf Aktualisieren.

Erstellen und Ausführen der Anwendung (AWS CLI)

In diesem Abschnitt verwenden Sie AWS CLI, um die Anwendung Managed Service für Apache Flink zu erstellen und auszuführen. Managed Service für Apache Flink verwendet den Befehl `kinesisanalyticsv2` AWS CLI, um Managed-Service-für-Apache-Flink-Anwendungen zu erstellen und mit diesen zu interagieren.

Erstellen einer Berechtigungsrichtlinie

> **Note**
>
> Sie müssen eine Berechtigungsrichtlinie und eine Rolle für Ihre Anwendung erstellen. Wenn Sie diese IAM-Ressourcen nicht erstellen, kann Ihre Anwendung nicht auf ihre Daten und Protokollstreams zugreifen.

Zuerst erstellen Sie eine Berechtigungsrichtlinie mit zwei Anweisungen: eine, die Berechtigungen für die `read`-Aktion auf den Quell-Stream zulässt, und eine andere, die Berechtigungen für die `write`-Aktionen auf den Senken-Stream zulässt. Anschließend fügen Sie die Richtlinie an eine IAM-Rolle (die Sie im nächsten Abschnitt erstellen) an. Wenn Managed Service für Apache Flink also die Rolle übernimmt, verfügt der Service über die erforderlichen Berechtigungen zum Lesen aus dem Quell-Stream und zum Schreiben in den Senken-Stream.

Verwenden Sie den folgenden Code zum Erstellen der `AKReadSourceStreamWriteSinkStream`-Berechtigungsrichtlinie. Ersetzen Sie *`username`* durch den Benutzernamen, den Sie verwendet

haben, um den Amazon-S3-Bucket zum Speichern des Anwendungscodes zu erstellen. Ersetzen Sie die Konto-ID in den Amazon-Ressourcennamen (ARNs) (*012345678901*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

step-by-step Anweisungen zum Erstellen einer Berechtigungsrichtlinie finden Sie unter Tutorial: Erstellen und Anfügen Ihrer ersten vom Kunden verwalteten Richtlinie im IAM-Benutzerhandbuch.

> **Note**
>
> Um auf andere Amazon-Services zuzugreifen, können Sie AWS SDK for Java verwenden. Managed Service für Apache Flink setzt die vom SDK benötigten Anmeldeinformationen

automatisch auf die der IAM-Rolle für die Dienstausführung, die mit Ihrer Anwendung verknüpft ist. Es sind keine weiteren Schritte erforderlich.

Erstellen einer IAM-Rolle

In diesem Abschnitt erstellen Sie eine IAM-Rolle, die die Anwendung von Managed Service für Apache Flink annehmen kann, um einen Quell-Stream zu lesen und in den Senken-Stream zu schreiben.

Managed Service für Apache Flink kann ohne Berechtigungen nicht auf Ihren Stream zugreifen. Sie erteilen diese Berechtigungen über eine IAM-Rolle. Jeder IAM-Rolle sind zwei Richtlinien angefügt. Die Vertrauensrichtlinie erteilt Managed Service für Apache Flink die Berechtigung zum Übernehmen der Rolle und die Berechtigungsrichtlinie bestimmt, was Managed Service für Apache Flink nach Annahme der Rolle tun kann.

Sie können die Berechtigungsrichtlinie, die Sie im vorherigen Abschnitt erstellt haben, dieser Rolle anfügen.

So erstellen Sie eine IAM-Rolle

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Roles (Rollen) und Create Role (Rolle erstellen) aus.
3. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option AWS-Service aus. Wählen Sie unter Choose the service that will use this role (Wählen Sie den Service aus, der diese Rolle verwendet) die Option Kinesis aus. Wählen Sie unter Select your use case (Wählen Sie Ihren Anwendungsfall aus) die Option Kinesis Analytics aus.

   Wählen Sie Weiter: Berechtigungen aus.
4. Wählen Sie auf der Seite Attach permissions policies (Berechtigungsrichtlinien hinzufügen) Next: Review (Weiter: Überprüfen) aus. Sie fügen Berechtigungsrichtlinien an, nachdem Sie die Rolle erstellt haben.
5. Geben Sie auf der Seite Create role (Rolle erstellen) den Text **`MF-stream-rw-role`** für Role name (Rollenname) ein. Wählen Sie Rolle erstellen aus.

   Jetzt haben Sie eine neue IAM-Rolle mit dem Namen `MF-stream-rw-role` erstellt. Im nächsten Schritt aktualisieren Sie die Vertrauens- und Berechtigungsrichtlinien für die Rolle.
6. Fügen Sie die Berechtigungsrichtlinie der Rolle an.

> **Note**
>
> Für diese Übung übernimmt Managed Service für Apache Flink diese Rolle sowohl für das Lesen von Daten aus einem Kinesis-Datenstrom (Quelle) als auch zum Schreiben der Ausgabedaten in einen anderen Kinesis-Datenstrom. Daher fügen Sie die Richtlinie an, die Sie im vorherigen Schritt erstellt haben, the section called “Erstellen einer Berechtigungsrichtlinie”.

a. Wählen Sie auf der Seite Summary (Übersicht) die Registerkarte Permissions (Berechtigungen) aus.

b. Wählen Sie Attach Policies (Richtlinien anfügen) aus.

c. Geben Sie im Suchfeld **`AKReadSourceStreamWriteSinkStream`** (die Richtlinie, die Sie im vorhergehenden Abschnitt erstellt haben) ein.

d. Wählen Sie die AKReadSourceStreamWriteSinkStream-Richtlinie und dann Richtlinie anfügen aus.

Sie haben nun die Service-Ausführungsrolle erstellt, die Ihre Anwendung für den Zugriff auf Ressourcen verwendet. Notieren Sie sich den ARN der neuen Rolle.

step-by-step Anweisungen zum Erstellen einer Rolle finden Sie unter Erstellen einer IAM-Rolle (Konsole) im IAM-Benutzerhandbuch.

Erstellen Sie die Anwendung Managed Service für Apache Flink

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `create_request.json`. Ersetzen Sie den Beispiel-Rollen-ARN durch den ARN für die Rolle, die Sie zuvor erstellt haben. Ersetzen Sie das Bucket-ARN-Suffix (*`username`*) mit dem Suffix, das Sie im vorherigen Abschnitt gewählt haben. Ersetzen Sie die beispielhafte Konto-ID (*`012345678901`*) in der Service-Ausführungsrolle mit Ihrer Konto-ID.

```
{
    "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. Führen Sie die `CreateApplication`-Aktion mit der vorherigen Anforderung zum Erstellen der Anwendung aus:

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

Die Anwendung wird nun erstellt. Sie starten die Anwendung im nächsten Schritt.

Starten der Anwendung

In diesem Abschnitt verwenden Sie die `StartApplication`-Aktion, um die Anwendung zu starten.

So starten Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `start_request.json`.

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

2. Führen Sie die `StartApplication`-Aktion mit der vorherigen Anforderung zum Starten der Anwendung aus:

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

Die Anwendung wird jetzt ausgeführt. Sie können die Metriken von Managed Service für Apache Flink in der Amazon- CloudWatch Konsole überprüfen, um zu überprüfen, ob die Anwendung funktioniert.

Stoppen der Anwendung

In diesem Abschnitt verwenden Sie die `StopApplication`-Aktion, um die Anwendung zu stoppen.

So stoppen Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `stop_request.json`.

```
{
   "ApplicationName": "test"
}
```

2. Führen Sie die `StopApplication`-Aktion mit der folgenden Anforderung zum Stoppen der Anwendung aus:

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

Die Anwendung wird nun gestoppt.

Hinzufügen einer CloudWatch Protokollierungsoption

Sie können die verwendenAWS CLI, um Ihrer Anwendung einen Amazon- CloudWatch Protokollstream hinzuzufügen. Informationen zur Verwendung von - CloudWatch Protokollen mit Ihrer Anwendung finden Sie unter the section called “Einrichten der Protokollierung”.

Umgebungseigenschaften aktualisieren

In diesem Abschnitt verwenden Sie die `UpdateApplication`-Aktion, um die Umgebungseigenschaften für die Anwendung zu ändern, ohne den Anwendungscode neu kompilieren zu müssen. In diesem Beispiel ändern Sie die Region der Quell- und Ziel-Streams.

So aktualisieren Sie die Umgebungseigenschaften für die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `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. Führen Sie die `UpdateApplication`-Aktion mit der vorherigen Anforderung aus, um die Umgebungseigenschaften zu aktualisieren:

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

Aktualisieren Sie den Anwendungscode

Wenn Sie Ihren Anwendungscode mit einer neuen Version Ihres Codepakets aktualisieren müssen, verwenden Sie die `UpdateApplication` AWS CLI-Aktion.

> **Note**
>
> Um eine neue Version des Anwendungscodes mit demselben Dateinamen zu laden, müssen Sie die neue Objektversion angeben. Weitere Informationen zur Verwendung von Amazon S3-Objektversionen finden Sie unter Versionierung aktivieren oder deaktivieren.

Um die AWS CLI zu verwenden, löschen Sie Ihr vorheriges Codepaket aus Ihrem Amazon S3-Bucket, laden Sie die neue Version hoch und rufen Sie `UpdateApplication` auf. Geben Sie dabei denselben Amazon S3-Bucket und Objektnamen sowie die neue Objektversion an. Die Anwendung wird mit dem neuen Codepaket neu gestartet.

Die folgende Beispielanforderung für die `UpdateApplication`-Aktion lädt den Anwendungscode neu und startet die Anwendung neu. Aktualisieren Sie die `CurrentApplicationVersionId` auf die aktuelle Anwendungsversion. Sie können die aktuelle Anwendungsversion mithilfe der Aktionen `ListApplications` oder `DescribeApplication` überprüfen. Aktualisieren Sie das Bucket-Namenssuffix (*<username>*) mit dem Suffix, das Sie im the section called “Erstellen von zwei Amazon Kinesis Datenstroms”-Abschnitt ausgewählt haben.

```
{
    "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"
                }
            }
        }
    }
}
```

Nächster Schritt

Schritt 4: Bereinigen von AWS-Ressourcen

Schritt 4: Bereinigen von AWS-Ressourcen

Dieser Abschnitt enthält Verfahren zum Bereinigen von AWS-Ressourcen, die im Tutorial Erste Schritte erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte:

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Sie Ihre Kinesis Data Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen

Löschen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie Konfigurieren aus.
4. Wählen Sie im Abschnitt Snapshots die Option Deaktivieren und anschließend Aktualisieren.
5. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie den Löschvorgang.

Löschen Sie Ihre Kinesis Data Streams

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink

2. Wählen Sie im Bereich Kinesis Data Streams die Option ausExampleInputStream.
3. Wählen Sie auf der ExampleInputStream Seite Kinesis Stream löschen aus und bestätigen Sie dann den Löschvorgang.
4. Wählen Sie auf der Seite Kinesis Streams die , ExampleOutputStreamwählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den ka-app-codeBucket -***<username>*** aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle kinesis-analytics--MyApplication-us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.
2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Erste Schritte: Flink 1.6.2

Dieses Thema enthält eine Version des Erste Schritte (DataStream API)-Tutorials, die Apache Flink 1.6.2 verwendet.

Themen

- Komponenten eines Managed Service für Apache Flink
- Voraussetzungen für das Fertigstellen der Übungen
- Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Administratorbenutzers
- Schritt 2: Einrichten der AWS Command Line Interface (AWS CLI)
- Schritt 3: Erstellen und Ausführen eines Managed Service für Apache Flink-Anwendung
- Schritt 4: Bereinigen vonAWS-Ressourcen

Komponenten eines Managed Service für Apache Flink

Zur Verarbeitung von Daten verwendet Ihre Managed Service für Apache Flink-Anwendung eine Java/Apache Maven- oder Scala-Anwendung, die mithilfe der Apache Flink-Laufzeit Eingaben verarbeitet und Ausgaben erzeugt.

Ein Managed Service für Apache Flink besteht aus den folgenden Komponenten:

- Runtime-Eigenschaften: Sie können Runtime-Eigenschaften verwenden, um Ihre Anwendung zu konfigurieren, ohne Ihren Anwendungscode neu kompilieren zu müssen.
- Quelle: Die Anwendung verwendet Daten mithilfe einer Quelle. Ein Quell-Connector liest Daten aus einem Kinesis Data Stream, einem Amazon S3-Bucket usw. Weitere Informationen finden Sie unter Quellen.
- Operatoren: Die Anwendung verarbeitet Daten mithilfe eines oder mehrerer Operatoren. Ein Operator kann Daten transformieren, anreichern oder aggregieren. Weitere Informationen finden Sie unter DataStream-API Operatoren.
- Senke: Die Anwendung erzeugt Daten für externe Quellen mithilfe von Senken. Ein Senken-Connector schreibt Daten in einen Kinesis Data Stream, einen Kinesis Data Firehose-Stream, einen Amazon S3-Bucket usw. Weitere Informationen finden Sie unter Senken.

Nachdem Sie Ihre Anwendung erstellt, kompiliert und verpackt haben, laden Sie das Codepaket in einen Amazon Simple Storage Service (Amazon S3)-Bucket hoch. Anschließend erstellen Sie eine

Managed Service für Apache Flink-Anwendung. Sie übergeben den Speicherort des Codepakets, einen Kinesis Data Stream als Streaming-Datenquelle und in der Regel einen Streaming- oder Dateispeicherort, der die verarbeiteten Daten der Anwendung empfängt.

Voraussetzungen für das Fertigstellen der Übungen

Zur Durchführung der Schritte in dieser Anleitung benötigen Sie Folgendes:

- Java Development Kit (JDK), Version 8. Legen Sie die `JAVA_HOME` Umgebungsvariable so fest, dass sie auf Ihren JDK-Installationsspeicherort weist.
- Wir empfehlen die Verwendung einer Entwicklungsumgebung (wie Eclipse Java Neon oder IntelliJ Idea), um Ihre Anwendung zu entwickeln und zu kompilieren.
- Git-Client. Installieren Sie den Git-Client, wenn Sie dies noch nicht getan haben.
- Apache Maven-Compiler-Plugin. Maven muss sich in Ihrem Arbeitspfad befinden. Zum Testen Ihrer Apache Maven-Installation geben Sie Folgendes ein:

```
$ mvn -version
```

Um zu beginnen, gehen Sie zu Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Administratorbenutzers.

Schritt 1: Einrichten eines AWS-Kontos und Erstellen eines Administratorbenutzers

So melden Sie sich für ein AWS-Konto an

Wenn Sie kein AWS-Konto haben, führen Sie die folgenden Schritte zum Erstellen durch.

Anmeldung für ein AWS-Konto

1. Öffnen Sie https://portal.aws.amazon.com/billing/signup.
2. Folgen Sie den Online-Anweisungen.

   Bei der Anmeldung müssen Sie auch einen Telefonanruf entgegennehmen und einen Verifizierungscode über die Telefontasten eingeben.

   Wenn Sie sich für ein AWS-Konto anmelden, wird ein Root-Benutzer des AWS-Kontos erstellt. Der Root-Benutzer hat Zugriff auf alle AWS-Services und Ressourcen des Kontos. Als bewährte

Sicherheitsmethode weisen Sie einem Administratorbenutzer Administratorzugriff zu und verwenden Sie nur den Root-Benutzer, um Aufgaben auszuführen, die Root-Benutzerzugriff erfordern.

AWS sendet Ihnen eine Bestätigungs-E-Mail, sobald die Anmeldung abgeschlossen ist. Sie können jederzeit Ihre aktuelle Kontoaktivität anzeigen und Ihr Konto verwalten. Rufen Sie dazu https://aws.amazon.com/ auf und klicken Sie auf Mein Konto.

Erstellen eines Administratorbenutzers

Nachdem Sie sich für ein AWS-Kontoangemeldet haben, sichern Sie Ihr Root-Benutzer des AWS-Kontos, aktivieren Sie AWS IAM Identity Centerund erstellen Sie einen administrativen Benutzer, damit Sie nicht den Root-Benutzer für alltägliche Aufgaben verwenden.

Schützen Ihres Root-Benutzer des AWS-Kontos

1. Melden Sie sich bei der AWS Management Console als Kontobesitzer an, indem Sie Root-Benutzer auswählen und Ihre AWS-Konto-E-Mail-Adresse eingeben. Geben Sie auf der nächsten Seite Ihr Passwort ein.

   Hilfe bei der Anmeldung mit dem Root-Benutzer finden Sie unter Anmelden als Root-Benutzer im AWS-AnmeldungBenutzerhandbuch zu .
2. Aktivieren Sie die Multi-Faktor-Authentifizierung (MFA) für den Root-Benutzer.

   Anweisungen dazu finden Sie unter Aktivieren eines virtuellen MFA-Geräts für den Root-Benutzer Ihres AWS-Konto (Konsole) im IAM-Benutzerhandbuch.

Erstellen eines Administratorbenutzers

1. Aktivieren von IAM Identity Center.

   Anweisungen finden Sie unter Aktivieren AWS IAM Identity Center im AWS IAM Identity Center Benutzerhandbuch.
2. Im IAM Identity Center gewähren Sie einem administrativen Benutzer administrativen Zugriff.

   Ein Tutorial zur Verwendung von IAM-Identity-Center-Verzeichnis als Identitätsquelle finden Sie unter Benutzerzugriff mit dem standardmäßigen IAM-Identity-Center-Verzeichnis konfigurieren im AWS IAM Identity Center-Benutzerhandbuch.

Anmelden als Administratorbenutzer

- Um sich mit Ihrem IAM-Identity-Center-Benutzer anzumelden, verwenden Sie die Anmelde-URL, die an Ihre E-Mail-Adresse gesendet wurde, als Sie den IAM-Identity-Center-Benutzer erstellt haben.

  Hilfe bei der Anmeldung mit einem IAM-Identity-Center-Benutzer finden Sie unter Anmelden beim AWS-Zugangsportal im AWS-Anmeldung Benutzerhandbuch zu.

Erteilen programmgesteuerten Zugriffs

Benutzer benötigen programmgesteuerten Zugriff, wenn sie außerhalb der AWS Management Console mit AWS interagieren möchten. Die Vorgehensweise, um programmgesteuerten Zugriff zu gewähren, hängt davon ab, welcher Benutzertyp auf zugreift AWS.

Um Benutzern programmgesteuerten Zugriff zu gewähren, wählen Sie eine der folgenden Optionen.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
Mitarbeiteridentität (Benutzer, die in IAM Identity Center verwaltet werden)	Verwenden Sie temporäre Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zur AWS CLI finden Sie unter Konfigurieren der AWS CLI für die Verwendung von AWS IAM Identity Center im AWS Command Line Interface-Benutzerhandbuch. • Informationen zu AWS-SDKs, Tools und AWS-APIs finden Sie unter IAM-Identity-Center-Authentifizierung im Referenzhandbuch zu AWS-SDKs und Tools.

Welcher Benutzer benötigt programmgesteuerten Zugriff?	Bis	Von
IAM	Verwenden Sie temporäre Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Folgen Sie den Anweisungen unter Verwenden temporärer Anmeldeinformationen mit AWS-Ressourcen im IAM-Benutzerhandbuch.
IAM	(Nicht empfohlen) Verwenden Sie langfristige Anmeldeinformationen, um programmgesteuerte Anforderungen an die AWS CLI, AWS-SDKs oder AWS-APIs zu signieren.	Befolgen Sie die Anweisungen für die Schnittstelle, die Sie verwenden möchten. • Informationen zur AWS CLI finden Sie unter Authentifizierung mit IAM-Benutzer-Anmeldeinformationen im AWS Command Line Interface-Benutzerhandbuch. • Informationen zu AWS-SDKs und Tools finden Sie unter Authentifizierung mit langfristigen Anmeldeinformationen im Referenzhandbuch zu AWS-SDKs und Tools. • Informationen zu AWS-APIs finden Sie unter Verwalten von Zugriffsschlüsseln für IAM-Benutzer im IAM-Benutzerhandbuch.

Schritt 2: Einrichten der AWS Command Line Interface (AWS CLI)

In diesem Schritt laden Sie die AWS CLI herunter und konfigurieren sie für die Verwendung mit einem Managed Service für Apache Flink.

> **Note**
>
> Bei allen Erste-Schritte-Übungen in diesem Handbuch wird davon ausgegangen, dass Sie in Ihrem Konto Administrator-Anmeldeinformationen (`adminuser`) verwenden, um die Operationen auszuführen.

> **Note**
>
> Wenn Sie die AWS CLI bereits installiert haben, müssen Sie möglicherweise ein Upgrade durchführen, um die neueste Funktionalität zu erhalten. Weitere Informationen finden Sie unter Installieren der AWS Command Line Interface im AWS Command Line Interface-Benutzerhandbuch. Zum Überprüfen der Version der AWS CLI führen Sie den folgenden Befehl aus:
>
> ```
> aws --version
> ```
>
> Die Übungen in diesem Tutorial erfordern die folgende AWS CLI-Version oder höher:
>
> ```
> aws-cli/1.16.63
> ```

Um das AWS CLI einzurichten

1. Herunterladen und Konfigurieren von AWS CLI. Eine Anleitung finden Sie unter den folgenden Themen im AWS Command Line Interface-Benutzerhandbuch:
   - Installieren des AWS Command Line Interface
   - Konfigurieren von AWS CLI
2. Fügen Sie ein benanntes Profil für den Administratorbenutzer in der AWS CLI `config`-Datei hinzu. Verwenden Sie dieses Profil beim Ausführen von AWS CLI-Befehlen. Weitere

Informationen zu benannten Profilen finden Sie unter Benannte Profile im AWS Command Line Interface Benutzerhandbuch.

```
[profile adminuser]
aws_access_key_id = adminuser access key ID
aws_secret_access_key = adminuser secret access key
region = aws-region
```

Eine Liste der verfügbaren AWS-Regionen finden Sie unter Regionen und Endpunkte im Allgemeine Amazon Web Services-Referenz

> **Note**
>
> Der Beispielcode und die Befehle in diesem Tutorial verwenden die Region USA West (Oregon). Um eine andere Region zu verwenden, ändern Sie die Region im Code und in den Befehlen für dieses Tutorial in die Region, die Sie verwenden möchten.

3. Überprüfen Sie die Einrichtung, indem Sie die folgenden Hilfebefehle in die Befehlszeile eingeben:

```
aws help
```

Nachdem Sie ein -AWSKonto und die eingerichtet habenAWS CLI, können Sie die nächste Übung ausprobieren, in der Sie eine Beispielanwendung konfigurieren und die end-to-end Einrichtung testen.

Nächster Schritt

Schritt 3: Erstellen und Ausführen eines Managed Service für Apache Flink-Anwendung

Schritt 3: Erstellen und Ausführen eines Managed Service für Apache Flink-Anwendung

In dieser Übung erstellen Sie eine Anwendung von Managed Service für Apache Flink mit Datenströmen als Quelle und Senke.

Dieser Abschnitt enthält die folgenden Schritte:

- Erstellen von zwei Amazon Kinesis Datenstroms

- Schreiben Sie Beispieldatensätze in den Eingabe-Stream
- Herunterladen und Überprüfen des Apache Flink-Streaming-Java-Codes
- Kompilieren des Anwendungscodes
- Hochladen des Apache Flink-Streaming-Java-Codes
- Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus

Erstellen von zwei Amazon Kinesis Datenstroms

Bevor Sie für diese Übung eine Anwendung von Managed Service für Apache Flink erstellen, erstellen Sie zwei Kinesis Data Streams (`ExampleInputStream` und `ExampleOutputStream`). Ihre Anwendung verwendet diese Streams für die Quell- und Ziel-Streams der Anwendung.

Sie können diese Streams mithilfe der Amazon-Kinesis-Konsole oder des folgenden AWS CLI-Befehls erstellen. Anweisungen für die Konsole finden Sie unter Erstellen und Aktualisieren von Datenströmen im Amazon Kinesis Data Streams Entwicklerhandbuch.

So erstellen Sie die Daten-Streams (AWS CLI)

1. Verwenden Sie den Befehl Amazon Kinesis `create-stream` AWS CLI, um den ersten Stream (`ExampleInputStream`) zu erstellen.

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

2. Um den zweiten Stream zu erstellen, den die Anwendung zum Schreiben der Ausgabe verwendet, führen Sie denselben Befehl aus und ändern den Stream-Namen in `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-west-2 \
--profile adminuser
```

Schreiben Sie Beispieldatensätze in den Eingabe-Stream

In diesem Abschnitt verwenden Sie ein Python-Skript zum Schreiben von Datensätzen in den Stream für die zu verarbeitende Anwendung.

> **Note**
>
> Dieser Abschnitt erfordert AWS SDK for Python (Boto).

1. Erstellen Sie eine Datei `stock.py` mit dem folgenden Inhalt:

```
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. Im weiteren Verlauf des Tutorials führen Sie das `stock.py`-Skript zum Senden von Daten an die Anwendung aus.

```
$ python stock.py
```

Herunterladen und Überprüfen des Apache Flink-Streaming-Java-Codes

Der Java-Anwendungscode für dieses Beispiel ist unter verfügbar GitHub. Zum Herunterladen des Anwendungscodes gehen Sie wie folgt vor:

1. Klonen Sie das Remote-Repository, indem Sie den folgenden Befehl verwenden:

```
git clone https://github.com/aws-samples/amazon-kinesis-data-analytics-examples.git
```

2. Navigieren Sie zum `amazon-kinesis-data-analytics-java-examples/GettingStarted_1_6` Verzeichnis .

Beachten Sie Folgendes zum Anwendungscode:

- Eine Project Object Model (pom.xml) Datei enthält Informationen über die Konfiguration und Abhängigkeiten der Anwendung, einschließlich der Bibliotheken des Managed Service für Apache Flink .
- Die `BasicStreamingJob.java`-Datei enthält die`main` -Methode, die die Funktionalität der Anwendung definiert.
- Die Anwendung verwendet eine Kinesis-Quelle zum Lesen aus dem Quell-Stream. Der folgende Codeausschnitt erstellt die Kinesis-Quelle:

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

- Ihre Anwendung erstellt Quell- und Senkenkonnektoren für den Zugriff auf externe Ressourcen, indem ein `StreamExecutionEnvironment`-Objekt verwendet wird.
- Die Anwendung erstellt Quell- und Senkenkonnektoren mit statischen Eigenschaften. Zum Verwenden dynamischer Anwendungseigenschaften verwenden Sie die Methoden `createSourceFromApplicationProperties` und `createSinkFromApplicationProperties`, um die Konnektoren zu erstellen. Diese Methoden lesen die Eigenschaften der Anwendung zum Konfigurieren der Konnektoren.

Weitere Informationen zu den Laufzeiteigenschaften finden Sie unter Laufzeiteigenschaften.

Kompilieren des Anwendungscodes

In diesem Abschnitt verwenden Sie den Apache Maven-Compiler zum Erstellen des Java-Codes für die Anwendung. Weitere Informationen zum Installieren von Apache Maven und des Java Development Kit (JDK) finden Sie unter Voraussetzungen für das Fertigstellen der Übungen.

> **Note**
>
> Zur Nutzung des Kinesis-Konnektors für Versionen von Apache Flink vor 1.11, müssen Sie den Quellcode für den Konnektor herunterladen und ihn erstellen. Einzelheiten dazu finden Sie in der Apache-Flink-Dokumentation.

Kompilieren des Anwendungscodes

1. Zum Verwenden Ihres Anwendungscodes kompilieren und packen Sie ihn in eine JAR-Datei. Sie können Ihren Code auf zwei Arten kompilieren und packen:

   - Verwenden Sie das Befehlszeilen-Maven-Tool. Erstellen Sie Ihre JAR-Datei, indem Sie den folgenden Befehl in dem Verzeichnis ausführen, das die `pom.xml`-Datei enthält:

     ```
     mvn package
     ```

     > **Note**
     >
     > Der Parameter -Dflink.version ist für Managed Service für Apache Flink Laufzeit Version 1.0.1 nicht erforderlich; er ist nur für Version 1.1.0 und höher erforderlich. Weitere Informationen finden Sie unter the section called “Angabe der Apache Flink-Version Ihrer Anwendung”.

   - Verwenden Sie Ihre Entwicklungsumgebung. Weitere Informationen finden Sie in der Dokumentation Ihrer Entwicklungsumgebung.

Sie können Ihr Paket als JAR-Datei hochladen oder komprimieren und als ZIP-Datei hochladen. Wenn Sie Ihre Anwendung mit der AWS CLI erstellen, geben Sie Ihren Codeinhaltstyp (JAR oder ZIP) an.

2. Wenn während der Erstellung Fehler aufgetreten sind, überprüfen Sie, ob Ihre `JAVA_HOME`-Umgebungsvariable richtig eingestellt ist.

Wenn die Anwendung erfolgreich kompiliert wurde, wird die folgende Datei erstellt:

```
target/aws-kinesis-analytics-java-apps-1.0.jar
```

Hochladen des Apache Flink-Streaming-Java-Codes

In diesem Abschnitt erstellen Sie einen Amazon Simple Storage Service (Amazon S3)-Bucket und laden Ihren Anwendungscode hoch.

So laden Sie den Anwendungscode hoch

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie Bucket erstellen aus.
3. Geben Sie **`ka-app-code-<username>`** im Feld Bucket-Name ein. Fügen Sie dem Bucket-Namen ein Suffix hinzu, wie z. B. Ihren Benutzernamen, damit er global eindeutig ist. Wählen Sie Weiter aus.
4. Lassen Sie im Schritt Optionen konfigurieren die Einstellungen unverändert und klicken Sie auf Weiter.
5. Lassen Sie im Schritt Berechtigungen festlegen die Einstellungen unverändert und klicken Sie auf Weiter.
6. Wählen Sie Bucket erstellen aus.
7. Wählen Sie in der Amazon S3-Konsole den Bucket ka-app-code-***<username>*** und dann Hochladen aus.
8. Klicken Sie im Schritt Auswählen von Dateien auf Hinzufügen von Dateien. Navigieren Sie zu der `aws-kinesis-analytics-java-apps-1.0.jar`Datei, die Sie im vorherigen Schritt erstellt haben. Wählen Sie Weiter aus.
9. Lassen Sie im Schritt Berechtigungen festlegen die Einstellungen unverändert. Wählen Sie Weiter aus.

10. Lassen Sie im Schritt Eigenschaften festlegen die Einstellungen unverändert. Klicken Sie auf Hochladen.

Ihr Anwendungscode ist jetzt in einem Amazon-S3-Bucket gespeichert, in dem Ihre Anwendung darauf zugreifen kann.

Erstellen und führen Sie die Anwendung Managed Service für Apache Flink aus

Sie können eine Anwendung von Managed Service für Apache Flink entweder über die Konsole oder die AWS CLI erstellen und ausführen.

> **Note**
>
> Wenn Sie die Anwendung mit der Konsole erstellen, werden Ihre AWS Identity and Access Management (IAM)- und Amazon- CloudWatch Logs-Ressourcen für Sie erstellt. Wenn Sie die Anwendung mit der AWS CLI erstellen, erstellen Sie diese Ressourcen separat.

Themen

- Erstellen und Ausführen der Anwendung (Konsole)
- Erstellen und Ausführen der Anwendung (AWS CLI)

Erstellen und Ausführen der Anwendung (Konsole)

Befolgen Sie diese Schritte, um die Anwendung über die Konsole zu erstellen, zu konfigurieren, zu aktualisieren und auszuführen.

Erstellen Sie die Anwendung

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Dashboard Managed Service für Apache Flink Analyseanwendung erstellen aus.
3. Geben Sie auf der Seite Managed Service für Apache Flink – Anwendung erstellen die Anwendungsdetails wie folgt ein:
   - Geben Sie als Anwendungsname ein **`MyApplication`**.
   - Geben Sie für Beschreibung den Text **`My java test app`** ein.
   - Wählen Sie für Laufzeit die Option Apache Flink aus.

> **Note**
>
> Managed Service für Apache Flink verwendet Apache Flink Version 1.8.2 oder 1.6.2.

- Ändern Sie den Versions-Pulldown auf Apache Flink 1.6.

4. Wählen Sie für Zugriffsberechtigungen die Option Erstellen / Aktualisieren Sie IAM-Rolle **kinesis-analytics-MyApplication-us-west-2** aus.
5. Wählen Sie Erstellen Sie Anwendung aus.

> **Note**
>
> Beim Erstellen einer Anwendung von Managed Service für Apache Flink mit der Konsole haben Sie die Möglichkeit, eine IAM-Rolle und -Richtlinie für Ihre Anwendung erstellen zu lassen. Ihre Anwendung verwendet diese Rolle und Richtlinie für den Zugriff auf ihre abhängigen Ressourcen. Diese IAM-Ressourcen werden unter Verwendung Ihres Anwendungsnamens und der Region wie folgt benannt:
>
> - Richtlinie: `kinesis-analytics-service-`*`MyApplication`*`-`*`us-west-2`*
> - Rolle: `kinesisanalytics-`*`MyApplication`*`-`*`us-west-2`*

Bearbeiten der IAM-Richtlinie

Bearbeiten Sie die IAM-Richtlinie zum Hinzufügen von Berechtigungen für den Zugriff auf die Kinesis-Datenströme.

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie Policies (Richtlinien). Wählen Sie die **kinesis-analytics-service-MyApplication-us-west-2**-Richtlinie aus, die die Konsole im vorherigen Abschnitt für Sie erstellt hat.
3. Wählen Sie auf der Seite Summary (Übersicht) die Option Edit policy (Richtlinie bearbeiten) aus. Wählen Sie den Tab JSON.
4. Fügen Sie den markierten Abschnitt der folgenden Beispielrichtlinie der Richtlinie hinzu. Ersetzen Sie die beispielhaften Konto-IDs (*012345678901*) mit Ihrer Konto-ID.

```
{
```

```
    "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"
        }
    ]
}
```

Konfigurieren der Anwendung

1. Wählen Sie auf der MyApplication Seite Konfigurieren aus.
2. Klicken Sie auf der Seite Configure application (Anwendung konfigurieren) auf die Option Code location (Codespeicherort):
   - Geben Sie für Amazon-S3-Bucket **ka-app-code-*<username>*** ein.
   - Geben Sie als Pfad zum Amazon-S3-Objekt den Wert **java-getting-started-1.0.jar** ein.
3. Wählen Sie unter Zugriff auf Anwendungsressourcen für Zugriffsberechtigungen die Option IAM-Rolle **kinesis-analytics-MyApplication-us-west-2** erstellen/aktualisieren aus.
4. Geben Sie die folgenden Eigenschaften und Werte der Anwendung ein:

Gruppen-ID	Schlüssel	Wert
ProducerConfigProperties	**flink.inputstream.initpos**	**LATEST**
ProducerConfigProperties	**aws.region**	**us-west-2**

Gruppen-ID	Schlüssel	Wert
ProducerConfigProperties	**AggregationEnabled**	**false**

5. Stellen Sie unter Überwachung sicher, dass die Ebene der Überwachungsmetriken auf Anwendung eingestellt ist.
6. Aktivieren Sie für die CloudWatch Protokollierung das Kontrollkästchen Aktivieren.
7. Wählen Sie Aktualisieren.

> **Note**
>
> Wenn Sie die Amazon- CloudWatch Protokollierung aktivieren, erstellt Managed Service für Apache Flink eine Protokollgruppe und einen Protokollstream für Sie. Die Namen dieser Ressourcen lauten wie folgt:
>
> - Protokollgruppe: `/aws/kinesis-analytics/MyApplication`
> - Protokollstream: `kinesis-analytics-log-stream`

Ausführen der Anwendung

1. Wählen Sie auf der MyApplication Seite Ausführen aus. Bestätigen Sie die Aktion.
2. Wenn die Anwendung ausgeführt wird, aktualisieren Sie die Seite. Die Konsole zeigt den Application graph (Anwendungs-Graph) an.

Stoppen der Anwendung

Wählen Sie auf der MyApplication Seite Stoppen aus. Bestätigen Sie die Aktion.

Aktualisieren der Anwendung

Mithilfe der Konsole können Sie Anwendungseinstellungen wie beispielsweise Anwendungseigenschaften, Überwachungseinstellungen und den Speicherort oder den Dateinamen der JAR-Anwendungsdatei aktualisieren. Außerdem können Sie die JAR-Anwendungsdatei erneut aus dem Amazon-S3-Bucket laden, wenn Sie den Anwendungscode aktualisieren müssen.

Wählen Sie auf der MyApplication Seite Konfigurieren aus. Aktualisieren Sie die Anwendungseinstellungen und klicken Sie auf Aktualisieren.

Erstellen und Ausführen der Anwendung (AWS CLI)

In diesem Abschnitt verwenden Sie AWS CLI, um die Anwendung Managed Service für Apache Flink zu erstellen und auszuführen. Managed Service für Apache Flink verwendet den Befehl `kinesisanalyticsv2` AWS CLI, um Managed-Service-für-Apache-Flink-Anwendungen zu erstellen und mit diesen zu interagieren.

Erstellen einer Berechtigungsrichtlinie

Zuerst erstellen Sie eine Berechtigungsrichtlinie mit zwei Anweisungen: eine, die Berechtigungen für die `read`-Aktion auf den Quell-Stream zulässt, und eine andere, die Berechtigungen für die `write`-Aktionen auf den Senken-Stream zulässt. Anschließend fügen Sie die Richtlinie an eine IAM-Rolle (die Sie im nächsten Abschnitt erstellen) an. Wenn Managed Service für Apache Flink also die Rolle übernimmt, verfügt der Service über die erforderlichen Berechtigungen zum Lesen aus dem Quell-Stream und zum Schreiben in den Senken-Stream.

Verwenden Sie den folgenden Code zum Erstellen der `AKReadSourceStreamWriteSinkStream`-Berechtigungsrichtlinie. Ersetzen Sie *`username`* durch den Benutzernamen, den Sie verwendet haben, um den Amazon-S3-Bucket zum Speichern des Anwendungscodes zu erstellen. Ersetzen Sie die Konto-ID in den Amazon-Ressourcennamen (ARNs) (*`012345678901`*) mit Ihrer Konto-ID.

```
{
    "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"
        }
    ]
}
```

step-by-step Anweisungen zum Erstellen einer Berechtigungsrichtlinie finden Sie unter Tutorial: Erstellen und Anfügen Ihrer ersten vom Kunden verwalteten Richtlinie im IAM-Benutzerhandbuch.

> **Note**
>
> Um auf andere Amazon-Services zuzugreifen, können Sie AWS SDK for Java verwenden. Managed Service für Apache Flink setzt die vom SDK benötigten Anmeldeinformationen automatisch auf die der IAM-Rolle für die Dienstausführung, die mit Ihrer Anwendung verknüpft ist. Es sind keine weiteren Schritte erforderlich.

Erstellen einer IAM-Rolle

In diesem Abschnitt erstellen Sie eine IAM-Rolle, die die Anwendung von Managed Service für Apache Flink annehmen kann, um einen Quell-Stream zu lesen und in den Senken-Stream zu schreiben.

Managed Service für Apache Flink kann ohne Berechtigungen nicht auf Ihren Stream zugreifen. Sie erteilen diese Berechtigungen über eine IAM-Rolle. Jeder IAM-Rolle sind zwei Richtlinien angefügt. Die Vertrauensrichtlinie erteilt Managed Service für Apache Flink die Berechtigung zum Übernehmen der Rolle und die Berechtigungsrichtlinie bestimmt, was Managed Service für Apache Flink nach Annahme der Rolle tun kann.

Sie können die Berechtigungsrichtlinie, die Sie im vorherigen Abschnitt erstellt haben, dieser Rolle anfügen.

So erstellen Sie eine IAM-Rolle

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie im Navigationsbereich Roles (Rollen) und Create Role (Rolle erstellen) aus.
3. Wählen Sie unter Typ der vertrauenswürdigen Entität auswählen die Option AWS-Service aus. Wählen Sie unter Choose the service that will use this role (Wählen Sie den Service aus, der diese Rolle verwendet) die Option Kinesis aus. Wählen Sie unter Select your use case (Wählen Sie Ihren Anwendungsfall aus) die Option Kinesis Analytics aus.

   Wählen Sie Weiter: Berechtigungen aus.
4. Wählen Sie auf der Seite Attach permissions policies (Berechtigungsrichtlinien hinzufügen) Next: Review (Weiter: Überprüfen) aus. Sie fügen Berechtigungsrichtlinien an, nachdem Sie die Rolle erstellt haben.
5. Geben Sie auf der Seite Create role (Rolle erstellen) den Text **`MF-stream-rw-role`** für Role name (Rollenname) ein. Wählen Sie Rolle erstellen aus.

   Jetzt haben Sie eine neue IAM-Rolle mit dem Namen `MF-stream-rw-role` erstellt. Im nächsten Schritt aktualisieren Sie die Vertrauens- und Berechtigungsrichtlinien für die Rolle.
6. Fügen Sie die Berechtigungsrichtlinie der Rolle an.

   > **Note**
   >
   > Für diese Übung übernimmt Managed Service für Apache Flink diese Rolle sowohl für das Lesen von Daten aus einem Kinesis-Datenstrom (Quelle) als auch zum Schreiben der Ausgabedaten in einen anderen Kinesis-Datenstrom. Daher fügen Sie die Richtlinie an, die Sie im vorherigen Schritt erstellt haben, the section called “Erstellen einer Berechtigungsrichtlinie”.

   a. Wählen Sie auf der Seite Summary (Übersicht) die Registerkarte Permissions (Berechtigungen) aus.
   b. Wählen Sie Attach Policies (Richtlinien anfügen) aus.
   c. Geben Sie im Suchfeld **`AKReadSourceStreamWriteSinkStream`** (die Richtlinie, die Sie im vorhergehenden Abschnitt erstellt haben) ein.
   d. Wählen Sie die AKReadSourceStreamWriteSinkStream-Richtlinie und dann Richtlinie anfügen aus.

Sie haben nun die Service-Ausführungsrolle erstellt, die Ihre Anwendung für den Zugriff auf Ressourcen verwendet. Notieren Sie sich den ARN der neuen Rolle.

step-by-step Anweisungen zum Erstellen einer Rolle finden Sie unter Erstellen einer IAM-Rolle (Konsole) im IAM-Benutzerhandbuch.

Erstellen Sie die Anwendung Managed Service für Apache Flink

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `create_request.json`. Ersetzen Sie den Beispiel-Rollen-ARN durch den ARN für die Rolle, die Sie zuvor erstellt haben. Ersetzen Sie das Bucket-ARN-Suffix (*username*) mit dem Suffix, das Sie im vorherigen Abschnitt gewählt haben. Ersetzen Sie die beispielhafte Konto-ID (*012345678901*) in der Service-Ausführungsrolle mit Ihrer Konto-ID.

```
{
    "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. Führen Sie die `CreateApplication`-Aktion mit der vorherigen Anforderung zum Erstellen der Anwendung aus:

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

Die Anwendung wird nun erstellt. Sie starten die Anwendung im nächsten Schritt.

Starten der Anwendung

In diesem Abschnitt verwenden Sie die `StartApplication`-Aktion, um die Anwendung zu starten.

So starten Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `start_request.json`.

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

2. Führen Sie die `StartApplication`-Aktion mit der vorherigen Anforderung zum Starten der Anwendung aus:

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

Die Anwendung wird jetzt ausgeführt. Sie können die Metriken von Managed Service für Apache Flink in der Amazon- CloudWatch Konsole überprüfen, um zu überprüfen, ob die Anwendung funktioniert.

Stoppen der Anwendung

In diesem Abschnitt verwenden Sie die `StopApplication`-Aktion, um die Anwendung zu stoppen.

So stoppen Sie die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `stop_request.json`.

```
{
   "ApplicationName": "test"
}
```

2. Führen Sie die `StopApplication`-Aktion mit der folgenden Anforderung zum Stoppen der Anwendung aus:

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

Die Anwendung wird nun gestoppt.

Hinzufügen einer CloudWatch Protokollierungsoption

Sie können die verwendenAWS CLI, um Ihrer Anwendung einen Amazon- CloudWatch Protokollstream hinzuzufügen. Informationen zur Verwendung von CloudWatch Protokollen mit Ihrer Anwendung finden Sie unter the section called “Einrichten der Protokollierung”.

Umgebungseigenschaften aktualisieren

In diesem Abschnitt verwenden Sie die `UpdateApplication`-Aktion, um die Umgebungseigenschaften für die Anwendung zu ändern, ohne den Anwendungscode neu kompilieren zu müssen. In diesem Beispiel ändern Sie die Region der Quell- und Ziel-Streams.

So aktualisieren Sie die Umgebungseigenschaften für die Anwendung

1. Speichern Sie den folgenden JSON-Code in eine Datei mit dem Namen `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. Führen Sie die `UpdateApplication`-Aktion mit der vorherigen Anforderung aus, um die Umgebungseigenschaften zu aktualisieren:

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

Aktualisieren Sie den Anwendungscode

Wenn Sie Ihren Anwendungscode mit einer neuen Version Ihres Codepakets aktualisieren müssen, verwenden Sie die `UpdateApplication` AWS CLI-Aktion.

Um AWS CLI zu verwenden, löschen Sie Ihr vorheriges Codepaket aus Ihrem Amazon S3-Bucket, laden Sie die neue Version hoch und rufen Sie `UpdateApplication` auf, wobei Sie denselben Amazon S3-Bucket und Objektnamen angeben. Die Anwendung wird mit dem neuen Codepaket neu gestartet.

Die folgende Beispielanforderung für die `UpdateApplication`-Aktion lädt den Anwendungscode neu und startet die Anwendung neu. Aktualisieren Sie die `CurrentApplicationVersionId` auf die aktuelle Anwendungsversion. Sie können die aktuelle Anwendungsversion mithilfe der Aktionen

`ListApplications` oder `DescribeApplication` überprüfen. Aktualisieren Sie das Bucket-Namenssuffix (*<username>*) mit dem Suffix, das Sie im the section called “Erstellen von zwei Amazon Kinesis Datenstroms”-Abschnitt ausgewählt haben.

```
{
    "ApplicationName": "test",
    "CurrentApplicationVersionId": 1,
    "ApplicationConfigurationUpdate": {
        "ApplicationCodeConfigurationUpdate": {
            "CodeContentUpdate": {
                "S3ContentLocationUpdate": {
                    "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
                    "FileKeyUpdate": "java-getting-started-1.0.jar"
                }
            }
        }
    }
}
```

Schritt 4: Bereinigen vonAWS-Ressourcen

Dieser Abschnitt enthält Verfahren zum Bereinigen von AWS-Ressourcen, die im Tutorial Erste Schritte erstellt wurden.

Dieses Thema enthält die folgenden Abschnitte:

- Löschen Sie die Anwendung Managed Service für Apache Flink
- Löschen Sie Ihre Kinesis Data Streams
- Löschen von Amazon-S3-Objekten und -Buckets
- Löschen Sie Ihre IAM-Ressourcen
- Löschen Ihrer CloudWatch Ressourcen

Löschen Sie die Anwendung Managed Service für Apache Flink

1. Öffnen Sie die Kinesis-Konsole unter.https://console.aws.amazon.com/kinesis.
2. Wählen Sie im Bereich Managed Service für Apache Flink die Option ausMyApplication.
3. Wählen Sie Konfigurieren aus.
4. Wählen Sie im Abschnitt Snapshots die Option Deaktivieren und anschließend Aktualisieren.

5. Wählen Sie auf der Seite der Anwendung die Option Löschen aus und bestätigen Sie den Löschvorgang.

Löschen Sie Ihre Kinesis Data Streams

1. Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
2. Wählen Sie im Bereich Kinesis Data Streams die Option ausExampleInputStream.
3. Wählen Sie auf der ExampleInputStream Seite Kinesis Stream löschen aus und bestätigen Sie dann den Löschvorgang.
4. Wählen Sie auf der Seite Kinesis Streams die , ExampleOutputStreamwählen Sie Aktionen, wählen Sie Löschen und bestätigen Sie dann den Löschvorgang.

Löschen von Amazon-S3-Objekten und -Buckets

1. Öffnen Sie die Amazon-S3-Konsole unter https://console.aws.amazon.com/s3/.
2. Wählen Sie den ka-app-codeBucket -***<username>*** aus.
3. Wählen Sie Löschen und geben Sie dann den Bucketnamen ein, um das Löschen zu bestätigen.

Löschen Sie Ihre IAM-Ressourcen

1. Öffnen Sie die IAM-Konsole unter https://console.aws.amazon.com/iam/.
2. Wählen Sie in der Navigationsleiste Policies aus.
3. Geben Sie in der Filtersteuerung Kinesis ein.
4. Wählen Sie die Richtlinie kinesis-analytics-service-MyApplication-us-west-2 aus.
5. Klicken Sie auf Richtlinienaktionen und anschließend auf Löschen.
6. Wählen Sie in der Navigationsleiste Roles (Rollen) aus.
7. Wählen Sie die Rolle kinesis-analytics--MyApplication-us-west-2 aus.
8. Wählen Sie dann Rolle löschen und bestätigen Sie das Löschen.

Löschen Ihrer CloudWatch Ressourcen

1. Öffnen Sie die - CloudWatch Konsole unter https://console.aws.amazon.com/cloudwatch/.

2. Wählen Sie in der Navigationsleiste Protokolle aus.
3. Wählen Sie die Protokollgruppe /aws/kinesis-analytics/MyApplication aus.
4. Wählen Sie dann Protokollgruppe löschen und bestätigen Sie das Löschen.

Apache-Flink-Einstellungen

Managed Service für Apache Flink ist eine Implementierung des Apache-Flink-Frameworks. Managed Service für Apache Flink verwendet die in diesem Abschnitt beschriebenen Standardwerte. Einige dieser Werte können von Anwendungen, die Managed Service für Apache Flink nutzen, im Code festgelegt werden, andere können nicht geändert werden.

Dieses Thema enthält die folgenden Abschnitte:

- Apache-Flink-Konfiguration
- Zustands-Backend
- Checkpointing
- Savepointing
- Heap-Größen
- Puffer-Entlastung
- Anpassbare Flink-Konfigurationseigenschaften
- Konfigurierte Flink-Eigenschaften anzeigen

Apache-Flink-Konfiguration

Managed Service für Apache Flink bietet eine Standard-Flink-Konfiguration, die aus von Apache Flink empfohlenen Werten für die meisten Eigenschaften und einigen wenigen, die auf gängigen Anwendungsprofilen basieren, besteht. Weitere Informationen zur Flink-Konfiguration finden Sie unter Konfiguration. Die vom Service bereitgestellte Standardkonfiguration funktioniert für die meisten Anwendungen. Wenn Sie jedoch die Flink-Konfigurationseigenschaften anpassen müssen, um die Leistung bestimmter Anwendungen mit hoher Parallelität, hoher Speicher- und Zustandsauslastung zu verbessern oder neue Debugging-Funktionen in Apache Flink zu aktivieren, können Sie bestimmte Eigenschaften ändern, indem Sie eine Support-Anfrage stellen. Weitere Informationen finden Sie unter AWS Support-Center. Sie können die aktuelle Konfiguration für Ihre Anwendung mithilfe des Apache-Flink-Dashboards überprüfen.

Zustands-Backend

Managed Service für Apache Flink speichert transiente Daten in einem Zustands-Backend. Managed Service für Apache Flink verwendet das RocksDBStateBackend. Der Aufruf von `setStateBackend`, um ein anderes Backend festzulegen, hat keine Auswirkung.

Wir aktivieren die folgenden Funktionen im Zustands-Backend:

- Inkrementelle Zustands-Backend-Snapshots
- Asynchrone Zustands-Backend-Snapshots
- Lokale Wiederherstellung von Checkpoints

In Managed Service für Apache Flink ist die `state.backend.rocksdb.ttl.compaction.filter.enabled`-Konfiguration standardmäßig aktiviert. Mithilfe dieses Filters können Sie Ihren Anwendungscode aktualisieren, um die Strategie zur Verdichtungsbereinigung zu aktivieren. Weitere Informationen finden Sie unter Zustands-TTL in Flink 1.8.0 in der Apache-Flink-Dokumentation.

Weitere Informationen zu Zustands-Backends finden Sie unter Zustands-Backends in der Apache-Flink-Dokumentation.

Checkpointing

Managed Service für Apache Flink verwendet eine Checkpoint-Standardkonfiguration mit den folgenden Werten. Einige dieser Werte können geändert werden. Sie müssen CheckpointConfiguration.ConfigurationType auf `CUSTOM` setzen, damit Managed Service für Apache Flink angepasste Checkpointing-Werte verwendet.

Einstellung	Kann angepasst werden?	Wie	Standardwert
CheckpointingEnabled	Anpassbar	Create Application Update Application AWS CloudFormation	True
CheckpointInterval	Anpassbar	Create Application	60000

Einstellung	Kann angepasst werden?	Wie	Standardwert
		Update Application AWS CloudFormation	
MinPauseBetweenCheckpoints	Anpassbar	Create Application Update Application AWS CloudFormation	5000
Nicht ausgerichtete Checkpoints	Anpassbar	Support-Fall	False
Anzahl gleichzeitiger Checkpoints	Nicht anpassbar	N/A	1
Checkpointing-Modus	Nicht anpassbar	N/A	Genau einmal
Checkpoint-Aufbewahrungsrichtlinie	Nicht anpassbar	N/A	Bei Fehlschlag
Checkpoint-Timeout	Nicht anpassbar	N/A	60 Minuten
Maximale Anzahl aufbewahrter Checkpoints	Nicht anpassbar	N/A	1
Neustart-Strategie	Nicht anpassbar	N/A	Feste Verzögerung, mit unendlichen Wiederholungen alle 10 Sekunden.

Einstellung	Kann angepasst werden?	Wie	Standardwert
Checkpoint- und Savepoint-Speicherort	Nicht anpassbar	N/A	Wir speichern dauerhafte Checkpoint- und Savepoint-Daten in einem serviceeigenen S3-Bucket.
Schwellenwert für den Zustands-Backend-Speicher	Nicht anpassbar	N/A	1048576

Savepointing

Bei der Wiederherstellung von einem Savepoint aus versucht der Wiederaufnahmevorgang standardmäßig, den gesamten Zustand des Savepoints dem Programm zuzuordnen, mit dem Sie die Wiederherstellung durchführen. Wenn Sie einen Operator gelöscht haben, schlägt die Wiederherstellung von einem Savepoint, der Daten enthält, die dem fehlenden Operator entsprechen, standardmäßig fehl. Sie können sicherstellen, dass der Vorgang erfolgreich ist, indem Sie den Parameter AllowNonRestoredState der FlinkRunConfiguration der Anwendung auf `true` setzen. Dadurch können beim Wiederaufnahmevorgang Zustandsdaten übersprungen werden, die dem neuen Programm nicht zugeordnet werden können.

Weitere Informationen finden Sie unter Nicht wiederhergestellten Status zulassen in der Dokumentation zu Apache Flink.

Heap-Größen

Managed Service für Apache Flink weist jeder KPU 3 GB JVM-Heap zu und reserviert 1 GB für native Code-Zuweisungen. Informationen zur Erhöhung der Anwendungskapazität finden Sie unter the section called “Skalierung”.

Weitere Informationen über JVM-Heap-Größen finden Sie unter Konfiguration in der Apache-Flink-Dokumentation.

Puffer-Entlastung

Die Puffer-Entlastung kann Anwendungen mit hohem Gegendruck helfen. Wenn in Ihrer Anwendung fehlgeschlagene Checkpoints/Savepoints auftreten, kann es hilfreich sein, dieses Feature zu aktivieren. Reichen Sie dazu einen Support-Fall ein.

Weitere Informationen finden Sie unter Die Puffer-Entlastung in der Apache-Flink-Dokumentation.

Anpassbare Flink-Konfigurationseigenschaften

Im Folgenden finden Sie die Flink-Konfigurationseinstellungen, die Sie mithilfe eines Support-Falls anpassen können. Sie können mehr als eine Eigenschaft gleichzeitig und für mehrere Anwendungen gleichzeitig ändern, indem Sie das Anwendungspräfix angeben. Wenn es andere Eigenschaften der Flink-Konfiguration außerhalb dieser Liste gibt, die Sie anpassen möchten, geben Sie bitte die genaue Eigenschaft in Ihrem Fall an.

Fehlertoleranz

```
restart-strategy:
```

```
restart-strategy.fixed-delay.delay:
```

Checkpoints und Zustands-Backends

```
state.backend:
```

```
state.backend.fs.memory-threshold:
```

```
state.backend.incremental:
```

Checkpointing

```
execution.checkpointing.unaligned:
```

Native RocksDB-Metriken

Native RocksDB-Metriken werden nicht an CloudWatch gesendet. Nach der Aktivierung können diese Metriken entweder über das Flink-Dashboard oder die Flink-REST-API mit benutzerdefinierten Tools abgerufen werden.

Managed Service für Apache Flink ermöglicht Kunden den Zugriff auf die neueste Flink REST-API (oder die unterstützte Version, die Sie verwenden) im schreibgeschützten Modus unter Verwendung der CreateApplicationPresignedUrl-API. Diese API wird vom eigenen Dashboard von Flink verwendet, kann aber auch von benutzerdefinierten Überwachungstools verwendet werden.

```
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:
```

Erweiterte Zustands-Backend-Optionen

```
state.storage.fs.memory-threshold:
```

Vollständige Aufgabenmanager-Optionen

```
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:

Arbeitsspeicherkonfiguration

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:

Erweiterte Cluster-Optionen

cluster.intercept-user-system-exit:

cluster.processes.halt-on-fatal-error:

Dateisystemkonfigurationen

`fs.s3.connection.maximum:`

`fs.s3a.connection.maximum:`

`fs.s3a.threads.max:`

`s3.upload.max.concurrent.uploads:`

Erweiterte Fehlertoleranz-Optionen

`heartbeat.timeout:`

`jobmanager.execution.failover-strategy:`

Arbeitsspeicherkonfiguration

`jobmanager.memory.heap.size:`

Metriken

`metrics.latency.interval:`

Erweiterte Optionen für den REST-Endpunkt und Client

`rest.flamegraph.enabled:`

`rest.server.numThreads:`

Erweiterte SSL-Sicherheitsoptionen

`security.ssl.internal.handshake-timeout:`

Erweiterte Planungsoptionen

`slot.request.timeout:`

Erweiterte Optionen für die Flink-Web-UI

`web.timeout:`

Konfigurierte Flink-Eigenschaften anzeigen

Sie können die Apache-Flink-Eigenschaften, die Sie selbst konfiguriert haben oder deren Änderung Sie in einem Support-Fall angefordert haben, über das Apache-Flink-Dashboard einsehen. Gehen Sie dazu wie folgt vor:

1. Gehen Sie zum Flink-Dashboard
2. Wählen Sie im linken Navigationsbereich Auftragsmanager aus.
3. Wählen Sie Konfiguration, um die Liste der Flink-Eigenschaften anzuzeigen.

Konfiguration von Managed Service für Apache Flink für den Zugriff auf Ressourcen in einer Amazon VPC

Sie können eine Anwendung, die Managed Service for Apache Flink nutzt, so konfigurieren, dass sie sich mit privaten Subnetzen in einer virtuellen privaten Cloud (VPC) in Ihrem Konto verbindet. Verwenden Sie Amazon Virtual Private Cloud (Amazon VPC) zum Erstellen eines privaten Netzwerks für Ressourcen wie z. B. Datenbanken, Cache-Instances oder interne Services. Verbinden Sie Ihre Anwendung mit der VPC, um während der Ausführung auf private Ressourcen zuzugreifen.

Dieses Thema enthält die folgenden Abschnitte:

- Amazon VPC – Konzepte
- VPC-Anwendungsberechtigungen
- Internet- und Servicezugriff für eine VPC-verbundene Anwendung, die Managed Service für Apache Flink nutzt.
- Managed Service für Apache Flink VPC API
- Beispiel: Verwenden einer VPC für den Zugriff auf Daten in einem Amazon-MSK-Cluster

Amazon VPC – Konzepte

Amazon VPC ist die Netzwerkschicht von Amazon EC2. Wenn Sie neu bei Amazon EC2 sind, erhalten Sie unter Was ist Amazon EC2? im Amazon EC2-Benutzerhandbuch für Linux-Instances einen kurzen Überblick.

Die wichtigsten Komponenten für VPCs sind folgende:

- Eine Virtual Private Cloud (VPC) ist ein virtuelles Netzwerk für Ihr AWS-Konto.
- Ein Subnetz ist ein Bereich an IP-Adressen in Ihrer VPC.
- Eine Routing-Tabelle enthält eine Reihe von Regeln, so genannte Routen, die festlegen, wohin der Netzwerkdatenverkehr gelenkt wird.
- Ein Internet-Gateway ist eine horizontal skalierte, redundante und hochverfügbare VPC-Komponente, die die Kommunikation zwischen Instances in Ihrer VPC und dem Internet ermöglicht. Sie verursacht daher keine Verfügbarkeitsrisiken oder Bandbreitenbeschränkungen in Ihrem Netzwerkverkehr.

- Ein VPC-Endpunkt ermöglicht Ihnen, eine private Verbindung zwischen Ihrer VPC und unterstützten AWS-Services und VPC-Endpunktservices mit PrivateLink einzurichten, ohne dass ein Internet-Gateway, ein NAT-Gerät, eine VPN-Verbindung oder eine AWS Direct Connect-Verbindung erforderlich sind. Instances in Ihrer VPC benötigen keine öffentlichen IP-Adressen, um mit den Ressourcen in dem Service zu kommunizieren. Der Datenverkehr zwischen Ihrer VPC und dem anderen Service verlässt das Amazon-Netzwerk nicht.

Weitere Informationen zu Amazon-VPC-Service finden Sie im Benutzerhandbuch von Amazon Virtual Private Cloud.

Managed Service für Apache Flink erstellt elastische Netzwerkschnittstellen in einem der Subnetze, die in Ihrer VPC-Konfiguration für die Anwendung bereitgestellt werden. Die Anzahl der in Ihren VPC-Subnetzen erstellten elastischen Netzwerkschnittstellen kann je nach Parallelität und Parallelität pro KPU der Anwendung variieren. Weitere Informationen zur Anwendungsskalierung finden Sie unter Skalierung.

> **Note**
>
> VPC-Konfigurationen werden für SQL-Anwendungen nicht unterstützt.

> **Note**
>
> Der Service von Managed Service für Apache Flink verwaltet den Checkpoint- und Snapshot-Status für Anwendungen, die über eine VPC-Konfiguration verfügen.

VPC-Anwendungsberechtigungen

In diesem Abschnitt werden die Berechtigungsrichtlinien beschrieben, die Ihre Anwendung benötigt, um mit Ihrer VPC zu funktionieren. Weitere Informationen zur Verwendung von Berechtigungsrichtlinien finden Sie unter Identity and Access Management für Amazon Managed Service für Apache Flink.

Die folgende Berechtigungsrichtlinie gewährt Ihrer Anwendung die erforderlichen Berechtigungen für die Interaktion mit einer VPC. Um diese Berechtigungsrichtlinie zu verwenden, fügen Sie sie der Ausführungsrolle Ihrer Anwendung hinzu.

Berechtigungsrichtlinie für den Zugriff auf eine Amazon VPC

```
{
  "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**
>
> Wenn Sie Anwendungsressourcen mithilfe der Konsole angeben (z. B. CloudWatch-Protokolle oder eine Amazon VPC), ändert die Konsole Ihre Anwendungsausführungsrolle, um die Berechtigung für den Zugriff auf diese Ressourcen zu gewähren. Sie müssen die Ausführungsrolle Ihrer Anwendung nur manuell ändern, wenn Sie Ihre Anwendung erstellen, ohne die Konsole zu verwenden.

Internet- und Servicezugriff für eine VPC-verbundene Anwendung, die Managed Service für Apache Flink nutzt.

Wenn Sie eine Funktion mit einer VPC in Ihrem Konto verbinden, hat sie keinen Zugriff auf das Internet, es sei denn, die VPC bietet Zugriff. Wenn die Anwendung Internetzugriff benötigt, muss Folgendes zutreffen:

- Die Anwendung, die Managed Service für Apache Flink nutzt, darf nur mit privaten Subnetzen konfiguriert werden.
- Die VPC muss ein NAT-Gateway oder eine -Instance in einem öffentlichen Subnetz enthalten.
- In einem öffentlichen Subnetz muss eine Route für ausgehenden Datenverkehr aus den privaten Subnetzen zum NAT-Gateway vorhanden sein.

> **Note**
>
> Verschiedene Services bieten VPC-Endpunkte. Sie können VPC-Endpunkte verwenden, um Verbindungen zu Amazon-Services aus einer VPC ohne Internetzugriff herzustellen.

Ob ein Subnetz öffentlich oder privat ist, hängt von seiner Routing-Tabelle ab. Jede Routing-Tabelle hat eine Standardroute, die den nächsten Hop für Pakete bestimmt, die ein öffentliches Ziel haben.

- Für ein privates Subnetz: Die Standardroute zeigt auf ein NAT-Gateway (nat-...) oder eine NAT-Instance (eni-...).
- Für ein öffentliches Subnetz: Die Standardroute zeigt auf ein Internet-Gateway (igw-...).

Nachdem Sie Ihre VPC mit einem öffentlichen Subnetz (mit einem NAT) und einem oder mehreren privaten Subnetzen konfiguriert haben, gehen Sie wie folgt vor, um Ihre privaten und öffentlichen Subnetze zu identifizieren:

- Wählen Sie im Navigationsbereich der VPC-Konsole Subnetze.
- Wählen Sie die Subnetze aus und klicken Sie dann auf die Registerkarte Routing-Tabelle. Überprüfen Sie die Standardroute:
  - Öffentliches Subnetz: Destination: 0.0.0.0/0, Ziel: igw-…
  - Privates Subnetz: Destination: 0.0.0.0/0, Ziel: nat-… oder eni-…

So verknüpfen Sie die Anwendung, die Managed Service für Apache Flink nutzt, mit privaten Subnetzen:

- Öffnen Sie die Konsole von Managed Service für Apache unter https://console.aws.amazon.com/flink
- Wählen Sie auf der Seite Anwendungen, die Managed Service für Apache Flink nutzen Ihre Anwendung und anschließend Anwendungsdetails aus.
- Wählen Sie auf der Seite für Ihre Anwendung Konfigurieren aus.
- Wählen Sie im Abschnitt VPC-Konnektivität die VPC, die Ihrer Anwendung zugeordnet werden soll. Wählen Sie die mit Ihrer VPC verknüpften Subnetze und Sicherheitsgruppe aus, die die Anwendung für den Zugriff auf VPC-Ressourcen verwenden soll.
- Wählen Sie Aktualisieren aus.

Verwandte Informationen

Erstellen einer VPC mit öffentlichen und privaten Subnetzen

Grundlagen zu NAT-Gateways

Managed Service für Apache Flink VPC API

Verwenden Sie die folgenden API-Operationen von Managed Service für Apache Flink, um VPCs für Ihre Anwendung zu verwalten. Weitere Informationen zur Verwendung der API von Managed Service für Apache Flink finden Sie unter API-Codebeispiele.

CreateApplication

Verwenden Sie die Aktion CreateApplication, um Ihrer Anwendung während der Erstellung eine VPC-Konfiguration hinzuzufügen.

Der folgende Beispiel-Anforderungscode für die `CreateApplication`-Aktion schließt eine VPC-Konfiguration ein, wenn die Anwendung erstellt wird:

```
{
  "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

Verwenden Sie die Aktion AddApplicationVpcConfiguration, um Ihrer Anwendung nach der Erstellung eine VPC-Konfiguration hinzuzufügen.

Der folgende Beispiel-Anforderungscode für die `AddApplicationVpcConfiguration`-Aktion fügt einer bestehenden Anwendung eine VPC-Konfiguration hinzu:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfiguration": {
      "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
      "SubnetIds": [ "subnet-0123456789abcdef0" ]
```

```
   }
}
```

DeleteApplicationVpcConfiguration

Verwenden Sie die Aktion DeleteApplicationVpcConfiguration, um eine VPC-Konfiguration aus Ihrer Anwendung zu entfernen.

Der folgende Beispiel-Anforderungscode für die `AddApplicationVpcConfiguration`-Aktion entfernt eine bestehende VPC-Konfiguration aus einer Anwendung:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfigurationId": "1.1"
}
```

UpdateApplication

Verwenden Sie die Aktion UpdateApplication, um alle VPC-Konfigurationen einer Anwendung gleichzeitig zu aktualisieren.

Der folgende Beispiel-Anforderungscode für die `UpdateApplication`-Aktion aktualisiert alle VPC-Konfigurationen einer Anwendung:

```
{
   "ApplicationConfigurationUpdate": {
      "VpcConfigurationUpdates": [
         {
            "SecurityGroupIdUpdates": [ "sg-0123456789abcdef0" ],
            "SubnetIdUpdates": [ "subnet-0123456789abcdef0" ],
            "VpcConfigurationId": "2.1"
         }
      ]
   },
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9
}
```

Beispiel: Verwenden einer VPC für den Zugriff auf Daten in einem Amazon-MSK-Cluster

Ein vollständiges Tutorial zum Zugriff auf Daten aus einem Amazon-MSK-Cluster in einer VPC finden Sie unter MSK-Replikation.

Fehlerbehebung für Managed Service für Apache Flink

Die folgenden Hinweise können Ihnen bei der Behebung von Problemen helfen, die bei Amazon Managed Service für Apache Flink auftreten können.

Themen

- Fehlerbehebung während der Entwicklung
- Fehlerbehebung bei Laufzeitproblemen

Fehlerbehebung während der Entwicklung

Themen

- Apache Flink Flame-Diagramme
- Anmeldeinformationsanbieter-Problem mit EFO-Konnektor 1.15.2
- Anwendungen mit nicht unterstützten Kinesis-Konnektoren
- Kompilierungsfehler: „Abhängigkeiten für das Projekt konnten nicht aufgelöst werden“
- Ungültige Auswahl: „kinesisanalyticsv2“
- UpdateApplication Aktion lädt Anwendungscode nicht neu
- S3 StreamingFileSink FileNotFoundExceptions
- FlinkKafkaConsumer Problem mit Stopp mit Savepoint
- Flink 1.15 Async Sink Deadlock
- Die Quellverarbeitung von Amazon Kinesis Data Streams ist beim Re-Sharding nicht in der richtigen Reihenfolge

Apache Flink Flame-Diagramme

Flame-Diagramme sind für Anwendungen in Versionen von Managed Service für Apache Flink, die Flame-Diagramme unterstützen, standardmäßig aktiviert. Flame-Diagramme können die Leistung der Anwendung beeinträchtigen, wenn Sie das Diagramm geöffnet lassen, wie in der Flink-Dokumentation beschrieben.

Wenn Sie Flame-Diagramme für Ihre Anwendung deaktivieren möchten, erstellen Sie einen Fall, um die Deaktivierung für Ihren Anwendungs-ARN anzufordern. Weitere Informationen finden Sie im AWS Support-Center.

Anmeldeinformationsanbieter-Problem mit EFO-Konnektor 1.15.2

Es gibt ein bekanntes Problem mit EFO-Konnektor-Versionen bis 1.15.2 von Kinesis Data Streams, bei dem der `FlinkKinesisConsumer` die `Credential Provider`-Konfiguration nicht berücksichtigt. Gültige Konfigurationen werden aufgrund des Problems ignoriert, was dazu führt, dass der AUTO-Anmeldeinformationsanbieter verwendet wird. Dies kann zu Problemen beim kontoübergreifenden Zugriff auf Kinesis mithilfe des EFO-Konnektors führen.

Um diesen Fehler zu beheben, verwenden Sie bitte die EFO-Konnektor-Version 1.15.3 oder höher.

Anwendungen mit nicht unterstützten Kinesis-Konnektoren

Managed Service für Apache Flink Version 1.15 lehnt automatisch den Start oder die Aktualisierung von Anwendungen ab, wenn sie nicht unterstützte Kinesis-Konnektor-Versionen (vor Version 1.15.2) verwenden, die in Anwendungs-JARs oder -Archiven (ZIP) gebündelt sind.

Ablehnungsfehler

Sie erhalten folgende Fehlermeldung, wenn Sie Aufrufe zum Erstellen und Aktualisieren von Anwendungen übermitteln:

```
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
```

Schritte zur Behebung

- Aktualisieren Sie die Abhängigkeit der Anwendung von `flink-connector-kinesis`. Wenn Sie Maven als Build-Tool für Ihr Projekt verwenden, folgen Sie Aktualisieren einer Maven-Abhängigkeit . Wenn Sie Gradle verwenden, folgen Sie Aktualisieren einer Gradle-Abhängigkeit .
- Verpacken Sie die Anwendung erneut.
- Laden Sie sie in einen Amazon-S3-Bucket hoch.
- Reichen Sie die Anfrage zum Erstellen/Aktualisieren der Anwendung erneut mit der überarbeiteten Anwendung ein, die gerade in den Amazon-S3-Bucket hochgeladen wurde.

- Wenn Sie weiterhin dieselbe Fehlermeldung erhalten, überprüfen Sie Ihre Anwendungsabhängigkeiten erneut. Wenn das Problem weiterhin besteht, erstellen Sie bitte ein Support-Ticket.

Aktualisieren einer Maven-Abhängigkeit

1. Öffnen Sie die `pom.xml` des Projekts.
2. Finden Sie die Abhängigkeiten des Projekts. Sie sehen etwa so aus:

```
<project>

    ...

    <dependencies>

        ...

        <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-connector-kinesis</artifactId>
        </dependency>

        ...

    </dependencies>

    ...

</project>
```

3. Aktualisieren Sie `flink-connector-kinesis` auf Version 1.15.2 oder neuer. Zum Beispiel:

```
<project>

    ...

    <dependencies>

        ...

        <dependency>
            <groupId>org.apache.flink</groupId>
```

```
            <artifactId>flink-connector-kinesis</artifactId>
            <version>1.15.2</version>
        </dependency>

        ...

    </dependencies>

    ...

</project>
```

Aktualisieren einer Gradle-Abhängigkeit

1. Öffnen Sie die `build.gradle` des Projekts (oder `build.gradle.kts` für Kotlin-Anwendungen).
2. Finden Sie die Abhängigkeiten des Projekts. Sie sehen etwa so aus:

```
...

dependencies {

    ...

    implementation("org.apache.flink:flink-connector-kinesis")

    ...

}

...
```

3. Aktualisieren Sie `flink-connector-kinesis` auf Version 1.15.2 oder neuer. Zum Beispiel:

```
...

dependencies {

    ...

    implementation("org.apache.flink:flink-connector-kinesis:1.15.2")
```

```
    ...

}

...
```

Kompilierungsfehler: „Abhängigkeiten für das Projekt konnten nicht aufgelöst werden“

Um die Beispielanwendungen von Managed Service für Apache Flink zu kompilieren, müssen Sie zuerst den Apache-Flink-Kinesis-Konnektor herunterladen, kompilieren und zu Ihrem lokalen Maven-Repository hinzufügen. Wenn der Konnektor nicht zu Ihrem Repository hinzugefügt wurde, wird ein Kompilierungsfehler ähnlich dem folgenden angezeigt:

```
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
```

Um diesen Fehler zu beheben, müssen Sie den Apache-Flink-Quellcode (Version 1.8.2 von https://flink.apache.org/downloads.html) für den Konnektor herunterladen. Anweisungen zum Herunterladen, Kompilieren und Installieren des Apache-Flink-Quellcodes finden Sie unter the section called “Verwenden des Apache Flink Kinesis Streams-Konnektor mit früheren Apache Flink-Versionen”.

Ungültige Auswahl: „kinesisanalyticsv2“

Um Version 2 der Managed Service für Apache Flink API zu verwenden, benötigen Sie die neueste Version von AWS Command Line Interface (AWS CLI).

Informationen über das Einrichten von AWS CLI finden Sie unter Installieren von AWS Command Line Interface im AWS Command Line Interface-Benutzerhandbuch.

UpdateApplication Aktion lädt Anwendungscode nicht neu

Die UpdateApplication Aktion lädt den Anwendungscode nicht mit demselben Dateinamen neu, wenn keine S3-Objektversion angegeben ist. Um den Anwendungscode mit demselben Dateinamen neu zu laden, aktivieren Sie die Versionsverwaltung in Ihrem S3-Bucket und geben Sie die neue

Objektversion mithilfe des Parameters `ObjectVersionUpdate` an. Weitere Informationen zur Aktivierung der Objektversionsverwaltung in einem S3-Bucket finden Sie unter Aktivieren oder Deaktivieren der Versionsverwaltung.

S3 StreamingFileSink FileNotFoundExceptions

Anwendungen, die Managed Service für Apache Flink nutzen, können beim Starten von Snapshots auf den Fehler `FileNotFoundException` einer Teildatei in Bearbeitung stoßen, wenn eine in Bearbeitung befindliche Teildatei fehlt, auf die ihr Savepoint verweist. Wenn dieser Fehlermodus eintritt, ist der Operatorstatus der Anwendung, die Managed Service für Apache Flink nutzt, normalerweise nicht wiederherstellbar und muss ohne Verwendung eines Snapshots mittels `SKIP_RESTORE_FROM_SNAPSHOT` neu gestartet werden. Sehen Sie sich den folgenden beispielhaften Stacktrace an:

```
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`schreibt Datensätze in Dateisysteme, die von der Flink- FileSystem Abstraktion unterstützt werden. Da die eingehenden Streams unbegrenzt sein können, werden die Daten in Teildateien endlicher Größe organisiert, wobei beim Schreiben der Daten neue Dateien hinzugefügt werden. Der Teil-Lebenszyklus und die Rollover-Richtlinie bestimmen den Zeitpunkt, die Größe und die Benennung der Teildateien.

> **Note**
>
> Weitere Informationen finden Sie unter Teildatei-Lebenszylus .

Beim Checkpointing und Savepointing (Snapshotting) werden alle ausstehenden Dateien umbenannt und festgeschrieben. Teildateien, die sich in Bearbeitung befinden, werden jedoch nicht festgeschrieben, sondern umbenannt, und ihr Verweis wird in den Checkpoint- oder Savepoint-Metadaten gespeichert, die bei der Wiederherstellung von Aufträgen verwendet werden. Diese Teildateien, die sich in Bearbeitung befinden, werden irgendwann in den Status Ausstehend verschoben, umbenannt und von einem nachfolgenden Checkpoint oder Savepoint festgeschrieben.

Im Folgenden sind die Hauptursachen und Abhilfemaßnahmen für fehlende Teildateien in Bearbeitung aufgeführt:

- Veralteter Snapshot, der zum Starten der Anwendung von Managed Service für Apache Flink verwendet wird – nur der neueste System-Snapshot, der erstellt wird, wenn eine Anwendung gestoppt oder aktualisiert wird, kann verwendet werden, um eine Anwendung von Managed Service für Apache Flink mit Amazon S3 StreamingFileSink zu starten. Um diese Art von Fehlern zu vermeiden, verwenden Sie den neuesten System-Snapshot.
  - Dies ist beispielsweise der Fall, wenn Sie während des Stopps oder der Aktualisierung einen Snapshot auswählen, der mit `CreateSnapshot` statt mit einem vom System ausgelösten Snapshot erstellt wurde. Der Savepoint des älteren Snapshots enthält einen out-of-date Verweis auf die in Bearbeitung befindliche Teildatei, die umbenannt und von nachfolgendem Checkpoint oder Savepoint bestätigt wurde.
  - Dies kann auch passieren, wenn ein vom System ausgelöster Snapshot ausgewählt wird, der nicht beim letzten Anhalten/Aktualisieren-Ereignis ausgelöst wurde. Ein Beispiel ist eine Anwendung, bei der der System-Snapshot deaktiviert, aber `RESTORE_FROM_LATEST_SNAPSHOT` konfiguriert wurde. Im Allgemeinen ist bei Managed Service für Apache Flink-Anwendungen mit Amazon S3 StreamingFileSink -Abfrage immer der System-Snapshot aktiviert und `RESTORE_FROM_LATEST_SNAPSHOT` konfiguriert.
- Teildatei in Bearbeitung entfernt – Da sich die in Bearbeitung befindliche Teildatei in einem S3-Bucket befindet, kann sie von anderen Komponenten oder Akteuren, die Zugriff auf den Bucket haben, entfernt werden.
  - Dies kann passieren, wenn Sie Ihre App zu lange angehalten haben und die in Bearbeitung befindliche Teildatei, auf die der Savepoint Ihrer App verweist, von der S3-Bucket MultiPartUpload-Lebenszyklusrichtlinie entfernt wurde. Um diese Art von Fehlern zu vermeiden, stellen Sie sicher, dass Ihre S3-Bucket-MPU-Lebenszyklusrichtlinie einen ausreichend langen Zeitraum für Ihren Anwendungsfall abdeckt.
  - Dies kann auch passieren, wenn die Teildatei in Bearbeitung manuell oder durch eine andere Komponente Ihres Systems entfernt wurde. Um diese Art von Fehlern zu vermeiden, stellen Sie

bitte sicher, dass die in Bearbeitung befindlichen Teildateien nicht von anderen Akteuren oder Komponenten entfernt werden.

- Wettlaufsituation, bei der nach dem Savepoint ein automatisierter Checkpoint ausgelöst wird – dies betrifft Versionen von Managed Service für Apache Flink bis einschließlich 1.13. Dieses Problem wurde in Managed Service für Apache Flink Version 1.15 behoben. Migrieren Sie Ihre Anwendung auf Managed Service für Apache Flink Version 1.15, um ein erneutes Auftreten zu verhindern. Wir empfehlen auch, von StreamingFileSink zu zu migrierenFileSink.
  - Wenn Anwendungen angehalten oder aktualisiert werden, löst Managed Service für Apache Flink einen Savepoint aus und hält die Anwendung in zwei Schritten an. Wenn zwischen den beiden Schritten ein automatisierter Checkpoint ausgelöst wird, ist der Savepoint unbrauchbar, da seine in Bearbeitung befindliche Teildatei umbenannt und möglicherweise festgeschrieben würde.

FlinkKafkaConsumer Problem mit Stopp mit Savepoint

Wenn Sie das Legacy-System verwenden, besteht die Möglichkeit FlinkKafkaConsumer , dass Ihre Anwendung in UPDATING, STOPPING oder SCALING hängen bleibt, wenn Sie System-Snapshots aktiviert haben. Für dieses Problem ist kein veröffentlichter Fix verfügbar. Daher empfehlen wir Ihnen, ein Upgrade auf den neuen durchzuführenKafkaSource, um dieses Problem zu beheben.

Wenn Sie den `FlinkKafkaConsumer` mit aktivierten Snapshots verwenden, besteht die Möglichkeit, dass, wenn der Flink-Auftrag eine API-Anfrage vom Typ Anhalten mit Savepoint verarbeitet, der `FlinkKafkaConsumer` mit einem Laufzeitfehler fehlschlägt und eine `ClosedException` meldet. Unter diesen Bedingungen bleibt die Flink-Anwendung hängen, was sich als Fehlgeschlagene Checkpoints äußert.

Flink 1.15 Async Sink Deadlock

Es gibt ein bekanntes Problem mit AWS Connectors für die Apache-Flink- AsyncSink Implementierungsschnittstelle. Dies betrifft Anwendungen, die Flink 1.15 mit den folgenden Konnektoren verwenden:

- Für Java-Anwendungen:
  - 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`

- Flink SQL/TableAPI/Python-Anwendungen:
  - 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`

Bei betroffenen Anwendungen treten die folgenden Symptome auf:

- Der Flink-Auftrag befindet sich im `RUNNING`-Status, verarbeitet aber keine Daten;
- Es gibt keine Auftragsneustarts;
- Bei Checkpoints kommt es zu Zeitüberschreitungen.

Das Problem wird durch einen Fehler im AWS SDK verursacht, der dazu führt, dass dem Aufrufer bestimmte Fehler nicht angezeigt werden, wenn der asynchrone HTTP-Client verwendet wird. Dies führt dazu, dass die Senke während eines Checkpoint-Flush-Vorgangs auf unbestimmte Zeit darauf wartet, dass eine in Übertragung befindliche Anfrage abgeschlossen wird.

Dieses Problem wurde im AWS SDK ab Version 2.20.144 behoben.

Im Folgenden finden Sie Anweisungen zum Aktualisieren der betroffenen Konnektoren, um die neue Version des AWS SDK in Ihren Anwendungen zu verwenden:

Themen

- Aktualisieren der Java-Anwendungen
- Aktualisieren von Python-Anwendungen

Aktualisieren der Java-Anwendungen

Gehen Sie wie folgt vor, um Java-Anwendungen zu aktualisieren:

flink-connector-kinesis

Wenn die Anwendung `flink-connector-kinesis` verwendet:

Der Kinesis-Konnektor verwendet Shading, um einige Abhängigkeiten, einschließlich des AWS SDK, in das Konnektor-JAR zu verpacken. Gehen Sie wie folgt vor, um die AWS-SDK-Version zu aktualisieren, um diese schattierten Klassen zu ersetzen:

Maven

1. Fügen Sie den Kinesis-Konnektor und die erforderlichen AWS-SDK-Module als Projektabhängigkeiten hinzu.
2. Konfigurieren von `maven-shade-plugin`:

   a. Fügen Sie einen Filter hinzu, um schattierte AWS-SDK-Klassen auszuschließen, wenn der Inhalt der Kinesis-Konnektor-Jar kopiert wird.

   b. Fügen Sie eine Verschiebungsregel hinzu, um aktualisierte AWS-SDK-Klassen in das Paket zu verschieben, wie es vom Kinesis-Konnektor erwartet wird.

   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. Fügen Sie den Kinesis-Konnektor und die erforderlichen AWS-SDK-Module als Projektabhängigkeiten hinzu.
2. Anpassen der ShadowJar-Konfiguration:

   a. Schließen Sie schattierte AWS-SDK-Klassen aus, wenn der Inhalt der Kinesis-Konnektor-Jar kopiert wird.

b. Verschieben Sie aktualisierte AWS-SDK-Klassen in ein Paket, das vom Kinesis-Konnektor erwartet wird.

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")
}
...
```

Andere betroffene Konnektoren

Wenn die Anwendung einen anderen betroffenen Konnektor verwendet:

Um die AWS-SDK-Version zu aktualisieren, sollte die SDK-Version in der Build-Konfiguration des Projekts durchgesetzt werden.

Maven

Fügen Sie die AWS-SDK-Stückliste (BOM) zum Abschnitt der Abhängigkeitsverwaltung der Datei `pom.xml` hinzu, um die SDK-Version für das Projekt durchzusetzen.

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

Fügen Sie der AWS-SDK-Stückliste (BOM) eine Plattformabhängigkeit hinzu, um die SDK-Version für das Projekt durchzusetzen. Dies erfordert Gradle 5.0 oder neuer:

build.gradle

```
...
dependencies {
    ...
    flinkShadowJar(platform("software.amazon.awssdk:bom:2.20.144"))
    ...
}
...
```

Aktualisieren von Python-Anwendungen

Python-Anwendungen können Konnektoren auf zwei verschiedene Arten verwenden: Konnektoren und andere Java-Abhängigkeiten als Teil eines einzelnen uber-Jars verpacken oder Konnektor-JAR direkt verwenden. Beheben von Anwendungsproblemen, die durch Async Sink-Deadlock verursacht werden:

- Wenn die Anwendung ein uber-Jar verwendet, folgen Sie den Anweisungen für Aktualisieren der Java-Anwendungen .
- Gehen Sie wie folgt vor, um Konnektor-Jars aus dem Quellcode neu zu erstellen:

Erstellen von Konnektoren aus dem Quellcode:

Voraussetzungen, ähnlich den Build-Anforderungen von Flink:

- Java 11
- Maven 3.2.5

flink-sql-connector-kinesis

1. Herunterladen des Quellcodes für Flink 1.15.4:

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Entpacken des Quellcodes:

```
tar -xvf flink-1.15.4-src.tgz
```

3. Navigieren Sie zum Kinesis-Konnektor-Verzeichnis

```
cd flink-1.15.4/flink-connectors/flink-connector-kinesis/
```

4. Kompilieren und installieren Sie Konnektor-JAR unter Angabe der erforderlichen AWS-SDK-Version. Um den Build zu beschleunigen, verwenden Sie `-DskipTests`, um die Testausführung zu überspringen, und `-Dfast`, um zusätzliche Quellcodeprüfungen zu überspringen:

```
mvn clean install -DskipTests -Dfast -Daws.sdkv2.version=2.20.144
```

5. Navigieren Sie zum Kinesis-Konnektor-Verzeichnis

```
cd ../flink-sql-connector-kinesis
```

6. Kompilieren und installieren Sie SQL-Konnektor-JAR:

```
mvn clean install -DskipTests -Dfast
```

7. Das resultierende JAR wird verfügbar sein unter:

```
target/flink-sql-connector-kinesis-1.15.4.jar
```

flink-sql-connector-aws-kinesis-streams

1. Herunterladen des Quellcodes für Flink 1.15.4:

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Entpacken des Quellcodes:

```
tar -xvf flink-1.15.4-src.tgz
```

3. Navigieren Sie zum Kinesis-Konnektor-Verzeichnis

```
cd flink-1.15.4/flink-connectors/flink-connector-aws-kinesis-streams/
```

4. Kompilieren und installieren Sie Konnektor-JAR unter Angabe der erforderlichen AWS-SDK-Version. Um den Build zu beschleunigen, verwenden Sie `-DskipTests`, um die Testausführung zu überspringen, und `-Dfast`, um zusätzliche Quellcodeprüfungen zu überspringen:

```
mvn clean install -DskipTests -Dfast -Daws.sdk.version=2.20.144
```

5. Navigieren Sie zum Kinesis-Konnektor-Verzeichnis

```
cd ../flink-sql-connector-aws-kinesis-streams
```

6. Kompilieren und installieren Sie SQL-Konnektor-JAR:

```
mvn clean install -DskipTests -Dfast
```

7. Das resultierende JAR wird verfügbar sein unter:

```
target/flink-sql-connector-aws-kinesis-streams-1.15.4.jar
```

flink-sql-connector-aws-kinesis-firehose

1. Herunterladen des Quellcodes für Flink 1.15.4:

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Entpacken des Quellcodes:

```
tar -xvf flink-1.15.4-src.tgz
```

3. Navigieren Sie zum Konnektor-Verzeichnis

```
cd flink-1.15.4/flink-connectors/flink-connector-aws-kinesis-firehose/
```

4. Kompilieren und installieren Sie Konnektor-JAR unter Angabe der erforderlichen AWS-SDK-Version. Um den Build zu beschleunigen, verwenden Sie `-DskipTests`, um die Testausführung zu überspringen, und `-Dfast`, um zusätzliche Quellcodeprüfungen zu überspringen:

```
mvn clean install -DskipTests -Dfast -Daws.sdk.version=2.20.144
```

5. Navigieren Sie zum SQL-Konnektor-Verzeichnis

```
cd ../flink-sql-connector-aws-kinesis-firehose
```

6. Kompilieren und installieren Sie SQL-Konnektor-JAR:

```
mvn clean install -DskipTests -Dfast
```

7. Das resultierende JAR wird verfügbar sein unter:

```
target/flink-sql-connector-aws-kinesis-firehose-1.15.4.jar
```

flink-sql-connector-dynamodb

1. Herunterladen des Quellcodes für 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. Entpacken des Quellcodes:

```
tar -xvf flink-connector-aws-3.0.0-src.tgz
```

3. Navigieren Sie zum Konnektor-Verzeichnis

```
cd flink-connector-aws-3.0.0
```

4. Kompilieren und installieren Sie Konnektor-JAR unter Angabe der erforderlichen AWS-SDK-Version. Um den Build zu beschleunigen, verwenden Sie `-DskipTests`, um die Testausführung zu überspringen, und `-Dfast`, um zusätzliche Quellcodeprüfungen zu überspringen:

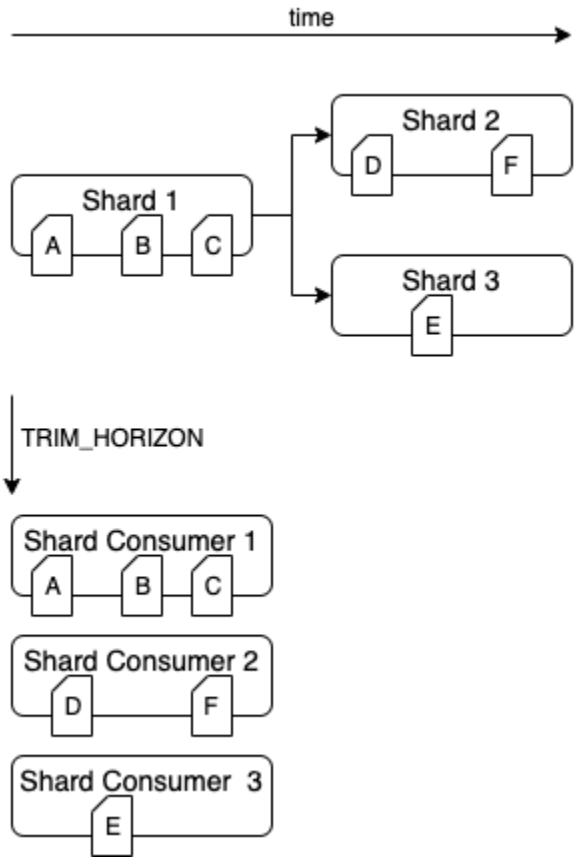
```
mvn clean install -DskipTests -Dfast -Dflink.version=1.15.4 -
Daws.sdk.version=2.20.144
```

5. Das resultierende JAR wird verfügbar sein unter:

```
flink-sql-connector-dynamodb/target/flink-sql-connector-dynamodb-3.0.0.jar
```

Die Quellverarbeitung von Amazon Kinesis Data Streams ist beim Re-Sharding nicht in der richtigen Reihenfolge

Die aktuelle FlinkKinesisConsumer Implementierung bietet keine starken Bestellgarantien zwischen Kinesis-Shards. Dies kann zur out-of-order Verarbeitung während des Re-Shardings von Kinesis Stream führen, insbesondere für Flink-Anwendungen, bei denen Verarbeitungsverzögerungen auftreten. Unter bestimmten Umständen, z. B. bei Windows-Operatoren, die auf Ereigniszeiten basieren, können Ereignisse aufgrund der daraus resultierenden Verspätung verworfen werden.



Dies ist ein bekanntes Problem in Open Source Flink. Stellen Sie sicher, dass Ihre Flink-Anwendungen bei der Neupartitionierung nicht hinter Kinesis Data Streams zurückfallen, bis die Fehlerbehebung für den Konnektor verfügbar ist. Indem Sie sicherstellen, dass die Verarbeitungsverzögerung von Ihren Flink-Apps toleriert wird, können Sie die Auswirkungen der out-of-order Verarbeitung und das Risiko eines Datenverlusts minimieren.

Fehlerbehebung bei Laufzeitproblemen

Dieser Abschnitt enthält Informationen zum Diagnostizieren und Beheben von Laufzeitproblemen mit Ihrer Anwendung, die Managed Service für Apache Flink nutzt.

Themen

- Tools zur Fehlerbehebung
- Anwendungsprobleme
- Anwendung wird neu gestartet
- Durchsatz ist zu langsam
- Unbegrenztes Zustandswachstum

- E/A-gebundene Operatoren
- Upstream- oder Quelldrosselung aus einem Kinesis-Datenstrom
- Checkpoints
- Beim Checkpointing kommt es zu einer Zeitüberschreitung
- Checkpoint-Fehler für Apache-Beam-Anwendung
- Gegendruck
- Verzerrte Datenverteilung
- Zustandsverzerrung
- Integration mit Ressourcen in verschiedenen Regionen

Tools zur Fehlerbehebung

Das wichtigste Tool zur Erkennung von Anwendungsproblemen sind CloudWatch-Alarme. Mithilfe von CloudWatch-Alarmen können Sie Schwellenwerte für CloudWatch-Metriken festlegen, die auf Fehler oder Engpässe in Ihrer Anwendung hinweisen. Weitere Informationen zu empfohlenen CloudWatch-Alarmen finden Sie unter Verwenden von CloudWatch Alarmen mit Amazon Managed Service für Apache Flink.

Anwendungsprobleme

Dieser Abschnitt enthält Lösungen für Fehlerbedingungen, die bei Ihrer Anwendung, die Managed Service für Apache Flink nutzt, auftreten können.

Themen

- Anwendung hängt in einem vorübergehenden Status fest
- Snapshot-Erstellung schlägt fehl
- Kein Zugriff auf Ressourcen in einer VPC möglich
- Daten gehen beim Schreiben in einen Amazon-S3-Bucket verloren
- Die Anwendung befindet sich im RUNNING-Status, verarbeitet aber keine Daten
- Fehler beim Snapshot, beim Anwendungsupdate oder beim Beenden der Anwendung: InvalidApplicationConfigurationException
- java.nio.file.NoSuchFileException: /usr/local/openjdk-8/lib/security/cacerts

Anwendung hängt in einem vorübergehenden Status fest

Wenn Ihre Anwendung in einem vorübergehenden Status (`STARTING`, `UPDATING`, `STOPPING`, oder `AUTOSCALING`) hängen bleibt, können Sie Ihre Anwendung beenden, indem Sie die Aktion StopApplication verwenden, wobei der `Force`-Parameter auf `true` gesetzt ist. Sie können das Beenden einer Anwendung im Status `DELETING` nicht erzwingen. Wenn sich die Anwendung im Status `UPDATING` oder `AUTOSCALING` befindet, können Sie sie alternativ auf die vorherige laufende Version zurücksetzen. Wenn Sie ein Rollback einer Anwendung durchführen, werden Statusdaten aus dem letzten erfolgreichen Snapshot geladen. Wenn die Anwendung keine Snapshots hat, lehnt Managed Service für Apache Flink die Rollback-Anfrage ab. Weitere Informationen zum Rollback einer Anwendung finden Sie unter der Aktion RollbackApplication.

> **Note**
>
> Das erzwungene Beenden Ihrer Anwendung kann zu Datenverlust oder -duplizierung führen. Um Datenverlust oder doppelte Verarbeitung von Daten bei Anwendungsneustarts zu verhindern, empfehlen wir Ihnen, regelmäßig Snapshots Ihrer Anwendung zu erstellen.

Ursachen für hängengebliebene Anwendungen sind unter anderem:

- Anwendungszustand ist zu groß: Ein zu großer oder zu persistenter Anwendungszustand kann dazu führen, dass die Anwendung während eines Checkpoint- oder Snapshot-Vorgangs hängen bleibt. Überprüfen Sie die Metriken `lastCheckpointDuration` und `lastCheckpointSize` Ihrer Anwendung auf stetig steigende Werte oder ungewöhnlich hohe Werte.
- Anwendungscode ist zu groß: Stellen Sie sicher, dass die JAR-Datei Ihrer Anwendung kleiner als 512 MB ist. JAR-Dateien, die größer als 512 MB sind, werden nicht unterstützt.
- Erstellung eines Anwendungs-Snapshots schlägt fehl: Managed Service für Apache Flink erstellt während einer `UpdateApplication`- oder `StopApplication`-Anfrage einen Snapshot der Anwendung. Der Service verwendet dann diesen Snapshot-Status und stellt die Anwendung mithilfe der aktualisierten Anwendungskonfiguration wieder her, um exakt einmal eine Verarbeitungssemantik bereitzustellen. Falls die automatische Snapshot-Erstellung fehlschlägt, finden Sie im Folgenden unter Snapshot-Erstellung schlägt fehl weitere Informationen.
- Wiederherstellung aus einem Snapshot schlägt fehl: Wenn Sie einen Operator in einem Anwendungsupdate entfernen oder ändern und versuchen, eine Wiederherstellung aus einem Snapshot durchzuführen, schlägt die Wiederherstellung standardmäßig fehl, wenn der Snapshot Zustandsdaten für den fehlenden Operator enthält. Darüber hinaus bleibt die Anwendung entweder

im Status `STOPPED` oder `UPDATING` hängen. Um dieses Verhalten zu ändern und sicherzustellen, dass die Wiederherstellung erfolgreich ist, ändern Sie den Parameter AllowNonRestoredState der FlinkRunConfiguration der Anwendung auf `true`. Dadurch können beim Wiederaufnahmevorgang Zustandsdaten übersprungen werden, die dem neuen Programm nicht zugeordnet werden können.

- Anwendungsinitialisierung dauert länger: Managed Service für Apache Flink verwendet ein internes Timeout von 5 Minuten (Soft-Einstellung), während auf den Start eines Flink-Auftrags gewartet wird. Wenn Ihr Auftrag innerhalb dieses Timeouts nicht gestartet werden kann, wird Ihnen ein CloudWatch-Protokoll wie folgt angezeigt:

  ```
  Flink job did not start within a total timeout of 5 minutes for application: %s under
   account: %s
  ```

  Wenn Sie auf den oben genannten Fehler stoßen, bedeutet dies, dass Ihre mit der `main`-Methode des Flink-Auftrags definierten Operationen mehr als 5 Minuten dauern, was zu einem Timeout bei der Erstellung des Flink-Auftrags auf der Seite von Managed Service für Apache Flink führt. Wir empfehlen Ihnen, die JobManager-Protokolle von Flink sowie Ihren Anwendungscode zu überprüfen, um festzustellen, ob diese Verzögerung der `main`-Methode zu erwarten ist. Wenn nicht, müssen Sie Maßnahmen ergreifen, um das Problem zu beheben, damit sie in weniger als 5 Minuten abgeschlossen wird.

Sie können den Status Ihrer Anwendung mithilfe der `ListApplications`- oder der `DescribeApplication`-Aktion überprüfen.

Snapshot-Erstellung schlägt fehl

Der Service von Managed Service für Apache Flink kann unter den folgenden Umständen keinen Snapshot erstellen:

- Die Anwendung hat das Snapshot-Limit überschritten. Das Limit für Snapshots ist 1 000. Weitere Informationen finden Sie unter Snapshots.
- Die Anwendung ist nicht berechtigt, auf ihre Quelle oder Senke zuzugreifen.
- Der Anwendungscode funktioniert nicht richtig.
- Bei der Anwendung treten andere Konfigurationsprobleme auf.

Wenn beim Erstellen eines Snapshots während eines Anwendungsupdates oder beim Beenden der Anwendung eine Ausnahme auftritt, setzen Sie die `SnapshotsEnabled`-Eigenschaft der

`ApplicationSnapshotConfiguration` Ihrer Anwendung auf `false` und wiederholen Sie die Anfrage.

Snapshots können fehlschlagen, wenn die Operatoren Ihrer Anwendung nicht ordnungsgemäß bereitgestellt werden. Informationen zur Optimierung der Operatorleistung finden Sie unter Operatorenskalierung.

Wenn die Anwendung wieder in einen fehlerfreien Zustand zurückkehrt, empfehlen wir, die `SnapshotsEnabled`-Eigenschaft Ihrer Anwendung auf `true` zu setzen.

Kein Zugriff auf Ressourcen in einer VPC möglich

Wenn Ihre Anwendung eine VPC verwendet, die auf Amazon VPC läuft, gehen Sie wie folgt vor, um zu überprüfen, ob Ihre Anwendung Zugriff auf ihre Ressourcen hat:

- Überprüfen Sie Ihre CloudWatch-Protokolle auf den folgenden Fehler. Dieser Fehler weist darauf hin, dass Ihre Anwendung nicht auf Ressourcen in Ihrer VPC zugreifen kann:

  ```
  org.apache.kafka.common.errors.TimeoutException: Failed to update metadata after
   60000 ms.
  ```

  Wenn Sie diesen Fehler sehen, überprüfen Sie, ob Ihre Routing-Tabellen korrekt eingerichtet sind und ob Ihre Konnektoren die richtigen Verbindungseinstellungen haben.

  Informationen zum Einrichten und Analysieren von CloudWatch-Protokollen finden Sie unter Protokollieren und Überwachen.

Daten gehen beim Schreiben in einen Amazon-S3-Bucket verloren

Beim Schreiben von Ausgaben in einen Amazon-S3-Bucket mit Apache Flink Version 1.6.2 kann es zu Datenverlusten kommen. Wir empfehlen, die neueste unterstützte Version von Apache Flink zu verwenden, wenn Sie Amazon S3 direkt für die Ausgabe verwenden. Um mit Apache Flink 1.6.2 in einen Amazon-S3-Bucket zu schreiben, empfehlen wir die Verwendung von Kinesis Data Firehose. Weitere Informationen zum Verwenden von Kinesis Data Firehose mit Managed Service für Apache Flink finden Sie unter Kinesis Data Firehose Senke.

Die Anwendung befindet sich im RUNNING-Status, verarbeitet aber keine Daten

Sie können den Status Ihrer Anwendung mithilfe der `ListApplications`- oder der `DescribeApplication`-Aktion überprüfen. Wenn Ihre Anwendung den `RUNNING`-Status annimmt,

aber keine Daten in Ihre Senke schreibt, können Sie das Problem beheben, indem Sie Ihrer Anwendung einen Amazon-CloudWatch-Protokollstream hinzufügen. Weitere Informationen finden Sie unter Arbeiten mit CloudWatch Anwendungsprotokollierungsoptionen. Der Protokollstream enthält Meldungen, mit denen Sie Anwendungsprobleme beheben können.

Fehler beim Snapshot, beim Anwendungsupdate oder beim Beenden der Anwendung: InvalidApplicationConfigurationException

Während eines Snapshot-Vorgangs oder während eines Vorgangs, der einen Snapshot erstellt, z. B. beim Aktualisieren oder Beenden einer Anwendung, kann ein Fehler auftreten, der dem folgenden ähnelt:

```
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
```

Dieser Fehler tritt auf, wenn die Anwendung keinen Snapshot erstellen kann.

Wenn dieser Fehler während eines Snapshot-Vorgangs oder eines Vorgangs, der einen Snapshot erstellt, auftritt, gehen Sie wie folgt vor:

- Deaktivieren Sie Snapshots für Ihre Anwendung. Sie können dies entweder in der Konsole von Managed Service für Apache Flink oder mithilfe des `SnapshotsEnabledUpdate`-Parameters der UpdateApplication-Aktion tun.
- Untersuchen Sie, warum keine Snapshots erstellt werden können. Weitere Informationen finden Sie unter Anwendung hängt in einem vorübergehenden Status fest.
- Aktivieren Sie Snapshots erneut, wenn die Anwendung wieder fehlerfrei ist.

java.nio.file.NoSuchFileException: /usr/local/openjdk-8/lib/security/cacerts

Der Speicherort des SSL-Truststores wurde in einer früheren Bereitstellung aktualisiert. Verwenden Sie stattdessen den folgenden Wert für den `ssl.truststore.location`-Parameter:

```
/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
```

Anwendung wird neu gestartet

Wenn Ihre Anwendung nicht fehlerfrei ist, schlägt ihr Apache-Flink-Auftrag ständig fehl und wird neu gestartet. In diesem Abschnitt werden Symptome und Schritte zur Behebung dieses Problems beschrieben.

Symptome

Dieses Problem kann folgende Symptome aufweisen:

- Die `FullRestarts`-Metrik ist nicht Null. Diese Metrik gibt an, wie oft der Auftrag der Anwendung neu gestartet wurde, seit Sie die Anwendung gestartet haben.
- Die `Downtime`-Metrik ist nicht Null. Diese Metrik stellt die Anzahl der Millisekunden dar, für die sich die Anwendung im Status `FAILING` oder `RESTARTING` befindet.
- Das Anwendungsprotokoll enthält Statusänderungen zu `RESTARTING` oder `FAILED`. Sie können Ihr Anwendungsprotokoll mit der folgenden CloudWatch-Logs-Insights-Abfrage nach diesen Statusänderungen abfragen: Fehler analysieren: Fehler im Zusammenhang mit Anwendungsaufgaben.

Ursachen und Lösungen

Die folgenden Bedingungen können dazu führen, dass Ihre Anwendung instabil wird und wiederholt neu gestartet wird:

- Der Operator löst eine Ausnahme aus: Wenn eine Ausnahme in einem Operator in Ihrer Anwendung nicht behandelt wird, führt die Anwendung einen Failover durch (indem interpretiert wird, dass der Fehler nicht vom Operator behandelt werden kann). Die Anwendung wird vom letzten Checkpoint aus neu gestartet, um die Semantik der Exakt-einmal-Verarbeitung beizubehalten. Daher ist während dieser Neustartphasen `Downtime` nicht Null. Um dies zu verhindern, empfehlen wir Ihnen, alle wiederholbaren Ausnahmen im Anwendungscode zu behandeln.

  Sie können die Ursachen für dieses Problem untersuchen, indem Sie Ihre Anwendungsprotokolle nach Änderungen des Zustands Ihrer Anwendung von `RUNNING` auf `FAILED` abfragen. Weitere Informationen finden Sie unter the section called “Fehler analysieren: Fehler im Zusammenhang mit Anwendungsaufgaben”.

- Kinesis Data Streams werden nicht ordnungsgemäß bereitgestellt: Wenn eine Quelle oder Senke für Ihre Anwendung ein Kinesis-Datenstrom ist, überprüfen Sie die Metriken für den Stream auf die Fehler `ReadProvisionedThroughputExceeded` oder `WriteProvisionedThroughputExceeded`.

  Wenn Sie diese Fehler sehen, können Sie den verfügbaren Durchsatz für den Kinesis-Stream erhöhen, indem Sie die Anzahl der Shards des Streams erhöhen. Weitere Informationen finden Sie unter Wie kann ich die Anzahl der offenen Shards in Kinesis Data Streams ändern?.

- Andere Quellen oder Senken werden nicht ordnungsgemäß bereitgestellt oder sind nicht verfügbar: Stellen Sie sicher, dass Ihre Anwendung Quellen und Senken korrekt bereitstellt. Vergewissern Sie sich, dass alle in der Anwendung verwendeten Quellen oder Senken (z. B. andere AWS-Services oder externe Quellen oder Ziele) ordnungsgemäß bereitgestellt wurden, dass keine Lese- oder Schreibdrosselung auftritt oder dass sie regelmäßig nicht verfügbar sind.

  Wenn Sie Probleme mit dem Durchsatz Ihrer abhängigen Services haben, erhöhen Sie entweder die für diese Services verfügbaren Ressourcen oder untersuchen Sie die Ursache für Fehler oder Nichtverfügbarkeit.

- Operatoren werden nicht ordnungsgemäß bereitgestellt: Wenn der Workload auf den Threads für einen der Operatoren in Ihrer Anwendung nicht richtig verteilt ist, kann der Operator überlastet werden und die Anwendung kann abstürzen. Informationen zur Optimierung der Operatorparallelität finden Sie unter Richtiges Verwalten der Operatorenskalierung.

- Anwendung schlägt mit DaemonException fehl: Dieser Fehler erscheint in Ihrem Anwendungsprotokoll, wenn Sie eine Version von Apache Flink vor 1.11 verwenden. Möglicherweise müssen Sie auf eine neuere Version von Apache Flink aktualisieren, damit eine KPL-Version von 0.14 oder höher verwendet wird.

- Anwendung schlägt mit TimeoutException, FlinkException oder RemoteTransportException fehl: Diese Fehler können in Ihrem Anwendungsprotokoll erscheinen, wenn Ihre Aufgabenmanager abstürzen. Wenn Ihre Anwendung überlastet ist, kann es bei Ihren Aufgabenmanagern zu einer Überlastung der CPU- oder Speicherressourcen kommen, wodurch sie ausfallen.

  Diese Fehler können wie folgt aussehen:

  - `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`

Um dieses Problem zu beheben, überprüfen Sie Folgendes:

- Überprüfen Sie Ihre CloudWatch-Metriken auf ungewöhnliche Spitzen bei der CPU- oder Speicherauslastung.
- Überprüfen Sie Ihre Anwendung auf Durchsatzprobleme. Weitere Informationen finden Sie unter Behebung von Leistungsproblemen.
- Untersuchen Sie Ihr Anwendungsprotokoll auf unbehandelte Ausnahmen, die Ihr Anwendungscode auslöst.

- Anwendung schlägt mit dem Fehler JaxBannotationModule Not Found fehl: Dieser Fehler tritt auf, wenn Ihre Anwendung Apache Beam verwendet, aber nicht über die richtigen Abhängigkeiten oder Abhängigkeitsversionen verfügt. Anwendungen, die Managed Service für Apache Flink nutzen und Apache Beam verwenden, müssen die folgenden Versionen von Abhängigkeiten verwenden:

```
<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>
```

  Wenn Sie nicht die richtige Version von `jackson-module-jaxb-annotations` als explizite Abhängigkeit angeben, lädt Ihre Anwendung sie aus den Umgebungsabhängigkeiten, und da die Versionen nicht übereinstimmen, stürzt die Anwendung zur Laufzeit ab.

  Weitere Informationen zum Verwenden von Apache Beam mit Managed Service für Apache Flink finden Sie unter Verwenden von CloudFormation mit Managed Service für Apache Flink.

- Anwendung schlägt mit java.io.IOException fehl: Unzureichende Anzahl von Netzwerkpuffern

  Dies passiert, wenn einer Anwendung nicht genügend Speicher für Netzwerkpuffer zugewiesen ist. Netzwerkpuffer erleichtern die Kommunikation zwischen Unteraufgaben. Sie werden verwendet, um Datensätze vor der Übertragung über ein Netzwerk zu speichern und eingehende Daten zu speichern, bevor sie in Datensätze zerlegt und an Unteraufgaben übergeben werden. Die Anzahl der benötigten Netzwerkpuffer hängt direkt von der Parallelität und Komplexität Ihres Auftragsdiagramms ab. Es gibt eine Reihe von Ansätzen, um dieses Problem zu beheben:

  - Sie können einen niedrigeren Wert für `parallelismPerKpu` konfigurieren, sodass pro Unteraufgabe und Netzwerkpuffer mehr Speicher zugewiesen wird. Beachten Sie, dass eine

Senkung des Werts für `parallelismPerKpu` die KPU und damit die Kosten erhöht. Um dies zu vermeiden, können Sie die gleiche Menge an KPU beibehalten, indem Sie die Parallelität um denselben Faktor verringern.

- Sie können Ihr Auftragsdiagramm vereinfachen, indem Sie die Anzahl der Operatoren reduzieren oder sie so verketten, dass weniger Puffer benötigt werden.
- Andernfalls können Sie sich an https://aws.amazon.com/premiumsupport/ wenden, um eine benutzerdefinierte Netzwerkpufferkonfiguration zu erhalten.

Durchsatz ist zu langsam

Wenn Ihre Anwendung eingehende Streaming-Daten nicht schnell genug verarbeitet, funktioniert sie schlecht und wird instabil. In diesem Abschnitt werden Symptome und Schritte zur Behebung dieses Problems beschrieben.

Symptome

Dieser Zustand kann folgende Symptome aufweisen:

- Wenn die Datenquelle für Ihre Anwendung ein Kinesis-Stream ist, nimmt die `millisbehindLatest`-Metrik des Streams kontinuierlich zu.
- Wenn es sich bei der Datenquelle für Ihre Anwendung um einen Amazon-MSK-Cluster handelt, nehmen die Verbraucherverzögerungsmetriken des Clusters kontinuierlich zu. Weitere Informationen finden Sie unter Verbraucherverzögerungsüberwachung im Amazon MSK Entwicklerhandbuch.
- Wenn es sich bei der Datenquelle für Ihre Anwendung um einen anderen Service oder eine andere Quelle handelt, überprüfen Sie alle verfügbaren Verbraucherverzögerungsmetriken oder –daten.

Ursachen und Lösungen

Es kann viele Ursachen für einen langsamen Anwendungsdurchsatz geben. Wenn Ihre Anwendung mit den Eingaben nicht Schritt hält, überprüfen Sie Folgendes:

- Wenn die Durchsatzverzögerung stark ansteigt und dann abnimmt, überprüfen Sie, ob die Anwendung neu gestartet wird. Ihre Anwendung stoppt die Verarbeitung von Eingaben, während sie neu gestartet wird, was zu einem Anstieg der Verzögerung führt. Weitere Informationen über Anwendungsausfälle finden Sie unter Anwendung wird neu gestartet.

- Wenn die Durchsatzverzögerung konstant ist, überprüfen Sie, ob Ihre Anwendung leistungsoptimiert ist. Informationen zur Optimierung der Leistung Ihrer Anwendung finden Sie unter Behebung von Leistungsproblemen.
- Wenn die Durchsatzverzögerung nicht in die Höhe schnellt, sondern kontinuierlich zunimmt und Ihre Anwendung leistungsoptimiert ist, müssen Sie Ihre Anwendungsressourcen erhöhen. Informationen zur Erhöhung der Anwendungsressourcen finden Sie unter Skalierung.
- Wenn Ihre Anwendung aus einem Kafka-Cluster in einer anderen Region liest und `FlinkKafkaConsumer` oder `KafkaSource` sich trotz hoher Verbraucherverzögerung größtenteils im Leerlauf befindet (hohe `idleTimeMsPerSecond` oder niedrige `CPUUtilization`), können Sie den Wert für `receive.buffer.byte` erhöhen, z. B. auf 2097152. Weitere Informationen finden Sie im Abschnitt zu einer Umgebung mit hoher Latenz unter Benutzerdefinierte MSK-Konfigurationen.

Schritte zur Problembehebung bei langsamem Durchsatz oder zunehmender Verbraucherverzögerung in der Anwendungsquelle finden Sie unter Behebung von Leistungsproblemen.

Unbegrenztes Zustandswachstum

Wenn Ihre Anwendung veraltete Zustandsinformationen nicht ordnungsgemäß löscht, sammeln sich diese kontinuierlich an und führen zu Leistungs- oder Stabilitätsproblemen der Anwendung. In diesem Abschnitt werden Symptome und Schritte zur Behebung dieses Problems beschrieben.

Symptome

Dieses Problem kann folgende Symptome aufweisen:

- Die `lastCheckpointDuration`-Metrik nimmt allmählich zu oder steigt sprunghaft an.
- Die `lastCheckpointSize`-Metrik nimmt allmählich zu oder steigt sprunghaft an.

Ursachen und Lösungen

Die folgenden Bedingungen können dazu führen, dass Ihre Anwendung Zustandsdaten ansammelt:

- In Ihrer Anwendung werden Zustandsdaten länger aufbewahrt, als sie benötigt werden.
- Ihre Anwendung verwendet Fensterabfragen mit einer zu langen Dauer.

- Sie haben TTL für Ihre Zustandsdaten nicht festgelegt. Weitere Informationen finden Sie unter Zustand Time-To-Live (TTL) in der Apache-Flink-Dokumentation.
- Sie führen eine Anwendung aus, die von Apache Beam Version 2.25.0 oder neuer abhängt. Sie können die neue Version der Lesetransformation deaktivieren, indem Sie Ihre BeamApplicationProperties um die wichtigsten Experimente und den Wert `use_deprecated_read` erweitern. Weitere Informationen finden Sie in der Apache-Beam-Dokumentation.

Manchmal sind Anwendungen mit einem stetigen Zuwachs der Zustandsgröße konfrontiert, was auf lange Sicht nicht nachhaltig ist (eine Flink-Anwendung läuft schließlich unbegrenzt). Manchmal kann dies darauf zurückgeführt werden, dass Anwendungen Daten im Zustand speichern und alte Informationen nicht ordnungsgemäß veralten lassen. Aber manchmal werden einfach unangemessene Erwartungen an das gestellt, was Flink bieten kann. Anwendungen können Aggregationen über große Zeitfenster hinweg verwenden, die sich über Tage oder sogar Wochen erstrecken. Sofern AggregateFunctions nicht verwendet werden, die inkrementelle Aggregationen ermöglichen, muss Flink die Ereignisse des gesamten Fensters im Zustand halten.

Darüber hinaus muss die Anwendung bei der Verwendung von Prozessfunktionen zur Implementierung benutzerdefinierter Operatoren Daten aus dem Zustand entfernen, die für die Geschäftslogik nicht mehr benötigt werden. In diesem Fall kann Zustand Time-to-Live verwendet werden, um Daten auf der Grundlage der Verarbeitungszeit automatisch veralten zu lassen. Managed Service für Apache Flink verwendet inkrementelle Checkpoints, weshalb Zustand TTL auf der RocksDB-Verdichtung basiert. Sie können eine tatsächliche Verringerung der Zustandsgröße (angezeigt durch die Checkpoint-Größe) erst beobachten, nachdem ein Verdichtungsvorgang durchgeführt wurde. Insbesondere bei Checkpoint-Größen unter 200 MB ist es unwahrscheinlich, dass Sie aufgrund des Ablaufs des Zustands eine Verringerung der Checkpoint-Größe feststellen. Savepoints basieren jedoch auf einer sauberen Kopie des Zustands, die keine alten Daten enthält. Sie können also in Managed Service für Apache Flink einen Snapshot auslösen, um die Entfernung eines veralteten Zustands zu erzwingen.

Zu Debugging-Zwecken kann es sinnvoll sein, inkrementelle Checkpoints zu deaktivieren, um schneller zu überprüfen, ob die Checkpoint-Größe tatsächlich abnimmt oder sich stabilisiert (und um den Effekt der Verdichtung in RockSBS zu vermeiden). Dafür ist allerdings ein Ticket an das Serviceteam erforderlich.

E/A-gebundene Operatoren

Es empfiehlt sich, Abhängigkeiten von externen Systemen auf dem Datenpfad zu vermeiden. Es ist oft viel leistungsfähiger, einen Referenzdatensatz im Zustand zu halten, als ein externes System abzufragen, um einzelne Ereignisse anzureichern. Manchmal gibt es jedoch Abhängigkeiten, die nicht einfach in den Zustand versetzt werden können, z. B. wenn Sie Ereignisse mit einem auf Amazon Sagemaker gehosteten Machine-Learning-Modell anreichern möchten.

Operatoren, die über das Netzwerk Schnittstellen zu externen Systemen herstellen, können zu einem Engpass werden und zu Gegendruck führen. Es wird dringend empfohlen, AsyncIO zur Implementierung der Funktionalität zu verwenden, um die Wartezeit für einzelne Anrufe zu reduzieren und zu verhindern, dass die gesamte Anwendung langsamer wird.

Darüber hinaus kann es für Anwendungen mit E/A-gebundenen Operatoren auch sinnvoll sein, die ParallelismPerKPU-Einstellung der Anwendung, die Managed Service für Apache Flink nutzt, zu erhöhen. Diese Konfiguration beschreibt die Anzahl der parallelen Unteraufgaben, die eine Anwendung pro Kinesis Processing Unit (KPU) ausführen kann. Wenn der Standardwert von 1 auf beispielsweise 4 erhöht wird, nutzt die Anwendung dieselben Ressourcen (und hat dieselben Kosten), kann aber auf das Vierfache der Parallelität skaliert werden. Dies funktioniert gut für E/A-gebundene Anwendungen, verursacht jedoch zusätzlichen Overhead für Anwendungen, die nicht E/A-gebunden sind.

Upstream- oder Quelldrosselung aus einem Kinesis-Datenstrom

Symptom: Die Anwendung stößt auf `LimitExceededExceptions` aus ihrem Upstream-Kinesis-Datenstrom.

Mögliche Ursache: Die Standardeinstellung für den Kinesis-Konnektor der Apache-Flink-Bibliothek ist so eingestellt, dass er aus der Kinesis-Datenstromquelle liest, wobei eine sehr aggressive Standardeinstellung für die maximale Anzahl von Datensätzen gilt, die pro `GetRecords`-Aufruf abgerufen werden. Apache Flink ist standardmäßig so konfiguriert, dass 10 000 Datensätze pro GetRecords-Aufruf abgerufen werden (dieser Aufruf erfolgt standardmäßig alle 200 ms), obwohl das Limit pro Shard nur 1 000 Datensätze beträgt.

Dieses Standardverhalten kann zu Drosselung führen, wenn versucht wird, Daten aus dem Kinesis-Datenstrom zu verbrauchen, was sich auf die Leistung und Stabilität der Anwendung auswirkt.

Dies kann bestätigt werden, indem die CloudWatch-Metrik `ReadProvisionedThroughputExceeded` überprüft und längere oder anhaltende Zeiträume ermittelt werden, in denen diese Metrik größer als Null ist.

Der Kunde kann dies auch in den CloudWatch-Protokollen für seine Anwendung, die Managed Service für Apache Flink nutzt, sehen, da er weiterhin `LimitExceededException`-Fehler sieht.

Lösung: Der Kunde kann eines von zwei Dingen tun, um dieses Szenario zu lösen:

- Senken Sie das Standardlimit für die Anzahl der pro GetRecords-Aufruf abgerufenen Datensätze
- Der Kunde kann Adaptive Reads in seiner Anwendung, die Managed Service für Apache Flink nutzt, aktivieren. Weitere Informationen zum Feature Adaptive Reads finden Sie unter SHARD_USE_ADAPTIVE_READS

Checkpoints

Checkpoints sind der Mechanismus von Flink, der sicherstellt, dass der Zustand einer Anwendung fehlertolerant ist. Dieser Mechanismus ermöglicht es Flink, den Status der Operatoren wiederherzustellen, falls der Auftrag fehlschlägt, und verleiht der Anwendung dieselbe Semantik wie bei einer fehlerfreien Ausführung. Mit Managed Service für Apache Flink wird der Zustand einer Anwendung in RocksDB gespeichert, einem eingebetteten Schlüssel-Wert-Speicher, der seinen Betriebszustand auf der Festplatte speichert. Wenn ein Checkpoint erreicht wird, wird der Zustand auch auf Amazon S3 hochgeladen. Selbst wenn die Festplatte verloren geht, kann der Checkpoint verwendet werden, um den Zustand der Anwendung wiederherzustellen.

Weitere Informationen finden Sie unter Wie funktionieren Zustand-Snapshots?.

Checkpointing-Phasen

Für eine Checkpointing-Operator-Unteraufgabe in Flink gibt es 5 Hauptphasen:

- Warten [Startverzögerung] – Flink verwendet Checkpoint-Barrieren, die in den Stream eingefügt werden. Die Zeit in dieser Phase ist also die Zeit, in der der Operator darauf wartet, dass die Checkpoint-Barriere sie erreicht.
- Ausrichtung [Ausrichtungsdauer] – In dieser Phase hat die Unteraufgabe eine Barriere erreicht, wartet aber auf Barrieren aus anderen Eingabeströmen.
- Sync-Checkpointing [Sync-Dauer] – In dieser Phase nimmt die Unteraufgabe tatsächlich einen Snapshot des Zustands des Operators auf und blockiert alle anderen Aktivitäten in der Unteraufgabe.
- Async-Checkpointing [Async-Dauer] – Der Großteil dieser Phase besteht aus der Unteraufgabe, den Zustand auf Amazon S3 hochzuladen. Während dieser Phase ist die Unteraufgabe nicht mehr blockiert und kann Datensätze verarbeiten.

- Bestätigung – Dies ist normalerweise eine kurze Phase und besteht einfach aus der Unteraufgabe, die eine Bestätigung an den JobManager sendet und auch alle Commit-Nachrichten ausführt (z. B. mit Kafka-Senken).

Jede dieser Phasen (außer Bestätigung) ist einer Dauermetrik für Checkpoints zugeordnet, die über die Flink-WebUI verfügbar ist und die dazu beitragen kann, die Ursache für den langen Checkpoint zu isolieren.

Eine genaue Definition der einzelnen für Checkpoints verfügbaren Metriken finden Sie auf der Registerkarte Verlauf.

Untersuchen

Bei der Untersuchung einer langen Checkpoint-Dauer ist es am wichtigsten, den Engpass für den Checkpoint zu ermitteln, d. h. welcher Operator und welche Unteraufgabe am längsten bis zum Checkpoint benötigt und welche Phase dieser Unteraufgabe länger dauert. Dies kann mithilfe der Flink-WebUI unter der Aufgabe Aufträge Checkpoint ermittelt werden. Die Weboberfläche von Flink bietet Daten und Informationen, die bei der Untersuchung von Checkpoint-Problemen helfen. Eine vollständige Aufschlüsselung finden Sie unter Überwachen des Checkpointing.

Als Erstes sollten Sie sich die Gesamtdauer jedes Operators im Auftragsdiagramm ansehen, um festzustellen, welcher Operator lange braucht, um den Checkpoint zu erreichen, und wo weitere Untersuchungen erforderlich sind. Gemäß der Flink-Dokumentation lautet die Definition der Dauer wie folgt:

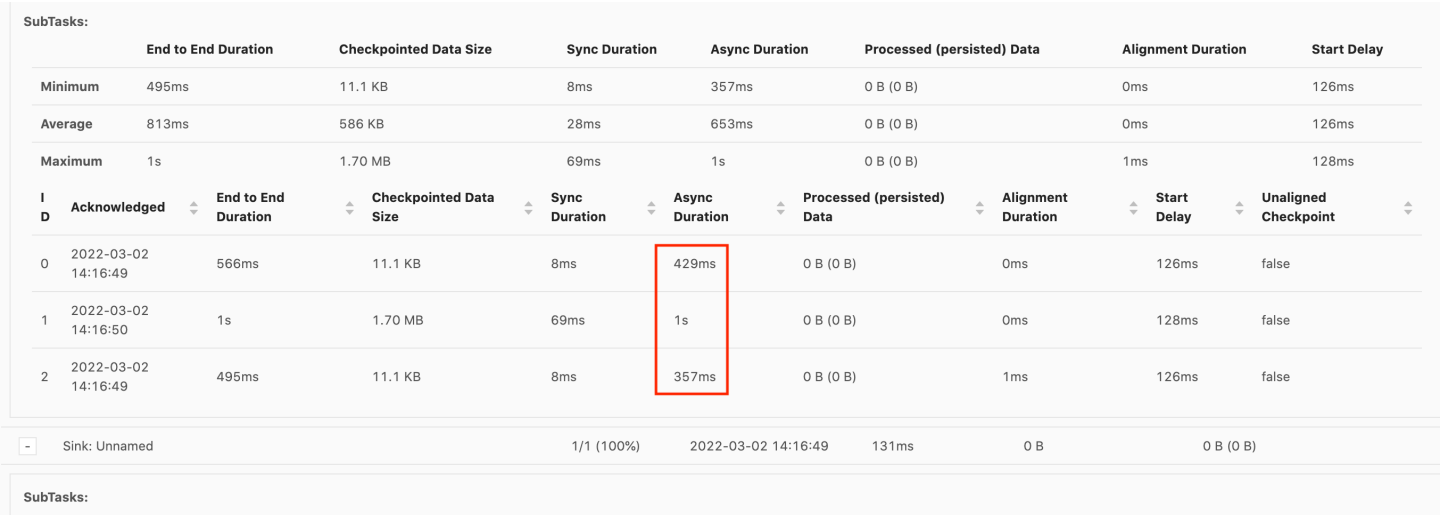
Die Dauer vom Trigger-Zeitstempel bis zur letzten Bestätigung (oder n/a, wenn noch keine Bestätigung eingegangen ist). Diese Gesamtdauer für einen vollständigen Checkpoint wird durch die letzte Unteraufgabe bestimmt, die den Checkpoint bestätigt. Diese Zeit ist normalerweise länger, als einzelne Teilaufgaben benötigen, um tatsächlich einen Checkpoint des Zustands zu erstellen.

Die anderen Zeitdauerangaben für den Checkpoint geben auch detailliertere Informationen darüber, wo die Zeit verbracht wird.

Wenn die Sync-Dauer hoch ist, deutet dies darauf hin, dass während der Snapshot-Erstellung etwas passiert. In dieser Phase wird `snapshotState()` für Klassen aufgerufen, die die SnapshotState-Schnittstelle implementieren. Dabei kann es sich um Benutzercode handeln, sodass Thread-Dumps nützlich sein können, um dies zu untersuchen.

Eine lange Async-Dauer würde darauf hindeuten, dass viel Zeit für das Hochladen des Zustands auf Amazon S3 aufgewendet wird. Dies kann der Fall sein, wenn der Zustand groß ist oder wenn

viele Zustandsdateien hochgeladen werden. Wenn dies der Fall ist, lohnt es sich zu untersuchen, wie der Zustand von der Anwendung verwendet wird, und sicherzustellen, dass die systemeigenen Flink-Datenstrukturen verwendet werden, wo immer dies möglich ist (Gekennzeichneten Zustand verwenden). Managed Service für Apache Flink konfiguriert Flink so, dass die Anzahl der Amazon-S3-Aufrufe minimiert wird, um sicherzustellen, dass diese nicht zu lang werden. Im Folgenden finden Sie ein Beispiel für die Checkpoint-Statistiken eines Operators. Es zeigt, dass die Async-Dauer im Vergleich zu den vorherigen Checkpoint-Statistiken für Operatoren relativ lang ist.



SubTasks:

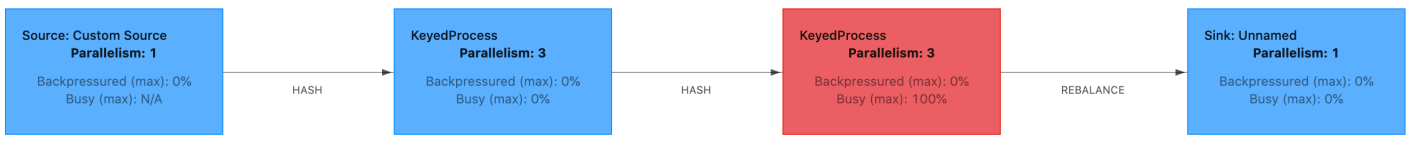
	End to End Duration	Checkpointed Data Size	Sync Duration	Async Duration	Processed (persisted) Data	Alignment Duration	Start Delay
Minimum	495ms	11.1 KB	8ms	357ms	0 B (0 B)	0ms	126ms
Average	813ms	586 KB	28ms	653ms	0 B (0 B)	0ms	126ms
Maximum	1s	1.70 MB	69ms	1s	0 B (0 B)	1ms	128ms

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:

Eine hohe Startverzögerung würde bedeuten, dass die meiste Zeit damit verbracht wird, darauf zu warten, dass die Checkpoint-Barriere den Operator erreicht. Dies deutet darauf hin, dass die Anwendung eine Weile benötigt, um Datensätze zu verarbeiten, was bedeutet, dass die Barriere langsam durch das Auftragsdiagramm fließt. Dies ist normalerweise der Fall, wenn der Auftrag Gegendruck erhält oder wenn ein oder mehrere Operatoren ständig beschäftigt sind. Es folgt ein Beispiel für einen JobGraph, bei dem der zweite KeyedProcess-Operator beschäftigt ist.



Sie können untersuchen, was so lange dauert, indem Sie entweder Flink-Flame-Diagramme oder TaskManager-Thread-Dumps verwenden. Sobald der Engpass identifiziert wurde, kann er mithilfe von Flame-Diagrammen oder Thread-Dumps weiter untersucht werden.

Thread-Dumps

Thread-Dumps sind ein weiteres Debugging-Tool auf einer etwas niedrigeren Ebene als Flame-Diagramme. Ein Thread-Dump gibt den Ausführungszustand aller Threads zu einem bestimmten Zeitpunkt aus. Flink nimmt einen JVM-Thread-Dump, der den Ausführungszustand aller Threads innerhalb des Flink-Prozesses darstellt. Der Zustand eines Threads wird durch einen Stack-Trace des Threads sowie durch einige zusätzliche Informationen dargestellt. Flame-Diagramme werden tatsächlich aus mehreren Stack-Traces erstellt, die schnell hintereinander aufgenommen wurden. Das Diagramm ist eine aus diesen Traces erstellte Visualisierung, die es einfach macht, die gemeinsamen Codepfade zu identifizieren.

```
"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
    ...
```

Oben sehen Sie einen Ausschnitt eines Thread-Dumps aus der Flink-Benutzeroberfläche für einen einzelnen Thread. Die erste Zeile enthält einige allgemeine Informationen zu diesem Thread, darunter:

- Der Thread-Name KeyedProcess (1/3) #0
- Priorität des Threads prio=5
- Eine eindeutige Thread-ID Id=1423
- Thread-Status RUNNABLE

Der Name eines Threads gibt normalerweise Auskunft über den allgemeinen Zweck des Threads. Operator-Threads können anhand ihres Namens identifiziert werden, da Operator-Threads denselben Namen wie der Operator haben und außerdem angeben, zu welcher Unteraufgabe sie gehören, z. B. stammt der KeyedProcess (1/3) #0-Thread vom KeyedProcess-Operator und ist von der ersten (von 3) Unteraufgabe.

Threads können sich in einem von wenigen Zuständen befinden:

- NEW – Der Thread wurde erstellt, aber noch nicht verarbeitet
- RUNNABLE – Der Thread wird auf der CPU ausgeführt
- BLOCKED – Der Thread wartet darauf, dass ein anderer Thread seine Sperre freigibt
- WAITING – Der Thread wartet mit einer `wait()`-, `join()`-, oder `park()`-Methode
- TIMED_WAITING – Der Thread wartet mit einer Sleep-, Wait-, Join- oder Park-Methode, jedoch mit einer maximalen Wartezeit.

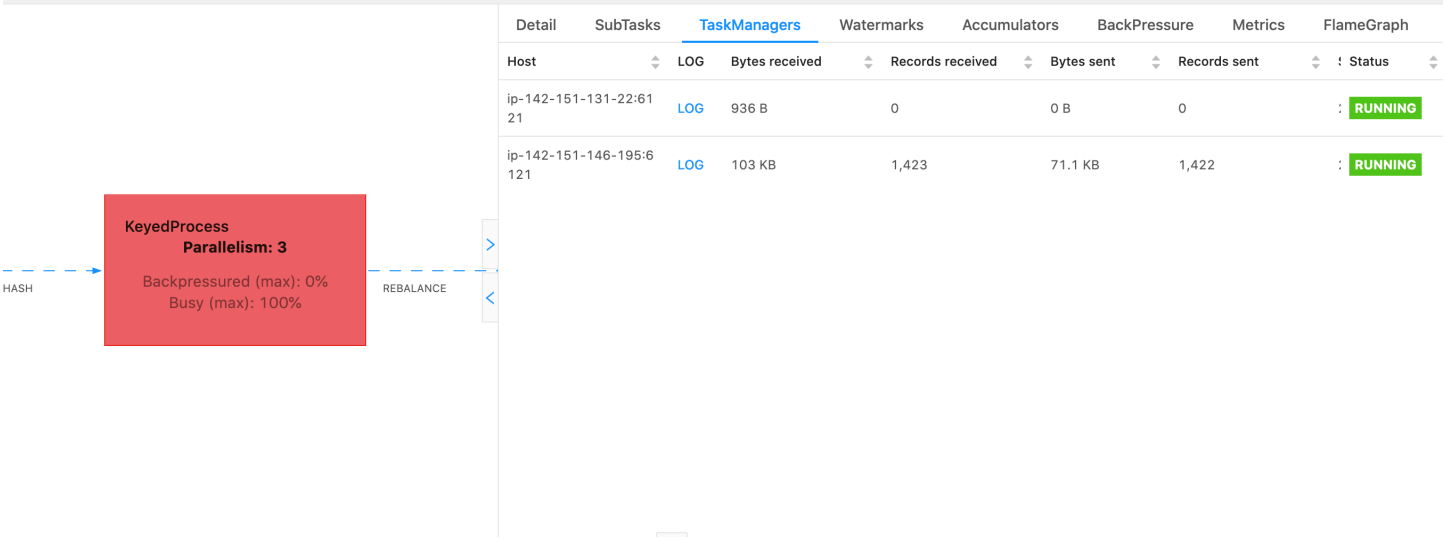
> **Note**
>
> In Flink 1.13 ist die maximale Tiefe eines einzelnen Stack-Trace im Thread-Dump auf 8 begrenzt.

> **Note**
>
> Thread-Dumps sollten das letzte Mittel zum Debuggen von Leistungsproblemen in einer Flink-Anwendung sein, da sie schwierig zu lesen sein können und die Entnahme mehrerer Stichproben und deren manuelle Analyse erfordern. Wenn möglich, ist es vorzuziehen, Flame-Diagramme zu verwenden.

Thread-Dumps in Flink

In Flink kann ein Thread-Dump erstellt werden, indem Sie in der linken Navigationsleiste der Flink-Benutzeroberfläche die Option Aufgabenmanager auswählen, einen bestimmten Aufgabenmanager auswählen und dann zur Registerkarte Thread-Dump navigieren. Der Thread-Dump kann heruntergeladen, in Ihren bevorzugten Texteditor (oder Thread-Dump-Analysator) kopiert oder direkt in der Textansicht in der Flink-Web-UI analysiert werden (diese letzte Option kann jedoch etwas umständlich sein).

Um zu bestimmen, von welchem Aufgabenmanager ein Thread-Dump erstellt werden soll, kann die Registerkarte Aufgabenmanager verwendet werden, wenn ein bestimmter Operator ausgewählt wird. Dies zeigt, dass der Operator für verschiedene Unteraufgaben eines Operators ausgeführt wird und auf verschiedenen Aufgabenmanagern ausgeführt werden kann.



Der Dump wird aus mehreren Stack-Traces bestehen. Bei der Untersuchung des Dumps sind jedoch diejenigen am wichtigsten, die sich auf einen Operator beziehen. Diese können leicht gefunden werden, da Operator-Threads denselben Namen wie der Operator haben und auch angeben, auf welche Unteraufgabe sie sich beziehen. Der folgende Stack-Trace stammt beispielsweise vom KeyedProcess-Operator und ist die erste Unteraufgabe.

```
"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
    ...
```

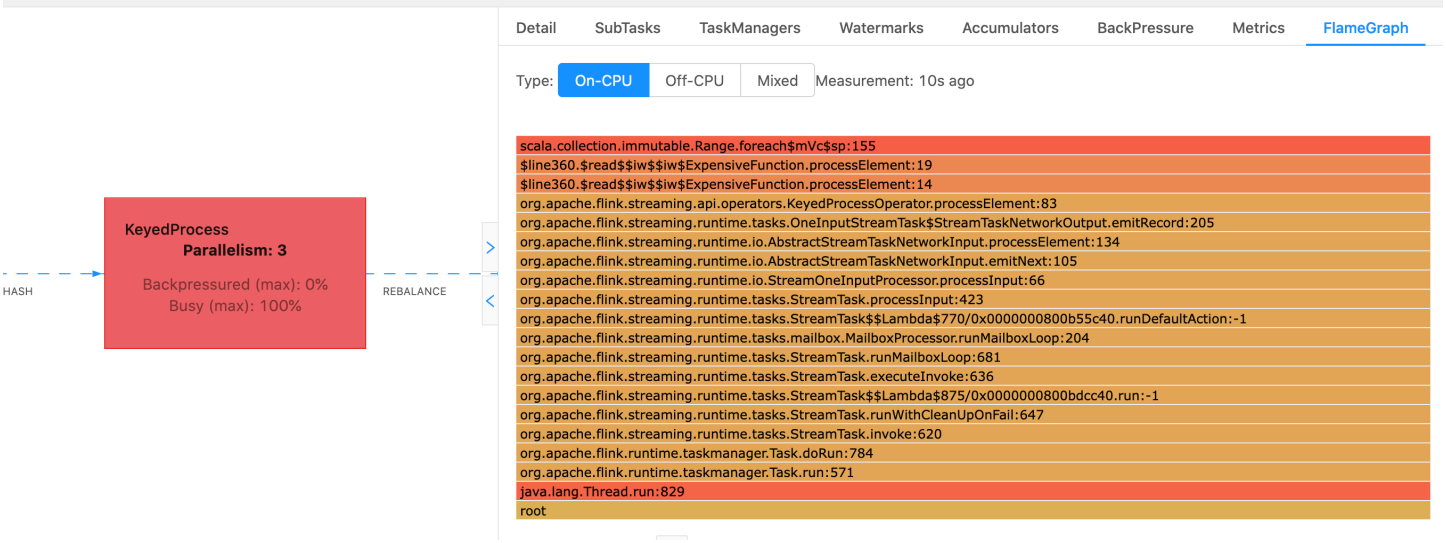
Dies kann verwirrend werden, wenn es mehrere Operatoren mit demselben Namen gibt, aber wir können Operatoren benennen, um dies zu umgehen. Beispiel:

```
....
.process(new ExpensiveFunction).name("Expensive function")
```

Flame-Diagramme

Flame-Diagramme sind ein nützliches Debugging-Tool, das die Stack-Traces des Zielcodes visualisiert und so die Identifizierung der häufigsten Codepfade ermöglicht. Sie werden erstellt, indem Stack-Traces mehrmals abgetastet werden. Die X-Achse eines Flame-Diagramms zeigt die verschiedenen Stack-Profile, während die Y-Achse die Stack-Tiefe und die Aufrufe des Stack-Trace zeigt. Ein einzelnes Rechteck in einem Flame-Diagramm steht für einen Stack-Frame, und die Breite eines Frames gibt an, wie häufig es in den Stacks vorkommt. Weitere Informationen über Flame-Diagramme und deren Nutzung finden Sie unter Flame-Diagramme.

In Flink kann das Flame-Diagramm für einen Operator über die Weboberfläche aufgerufen werden, indem Sie einen Operator auswählen und dann die Registerkarte FlameGraph wählen. Sobald genügend Proben gesammelt wurden, wird das Flame-Diagramm angezeigt. Im Folgenden finden Sie den FlameGraph für die ProcessFunction, deren Checkpoint viel Zeit in Anspruch genommen hat.



Dies ist ein sehr einfaches Flame-Diagramm, das zeigt, dass die gesamte CPU-Zeit innerhalb einer foreach-Schleife innerhalb des `processElement` des ExpensiveFunction-Operators aufgewendet wird. Sie erhalten auch die Zeilennummer, anhand derer Sie feststellen können, wo die Codeausführung stattfindet.

Beim Checkpointing kommt es zu einer Zeitüberschreitung

Wenn Ihre Anwendung nicht optimiert oder ordnungsgemäß bereitgestellt ist, können Checkpoints fehlschlagen. In diesem Abschnitt werden Symptome und Schritte zur Behebung dieses Problems beschrieben.

Symptome

Wenn Checkpoints für Ihre Anwendung fehlschlagen, ist der Wert von `numberOfFailedCheckpoints` größer als Null.

Checkpoints können entweder aufgrund direkter Fehler, wie Anwendungsfehler, oder aufgrund vorübergehender Fehler, z. B. aufgrund unzureichender Anwendungsressourcen, fehlschlagen. Überprüfen Sie Ihre Anwendungsprotokolle und Metriken auf die folgenden Symptome:

- Fehler in Ihrem Code.
- Fehler beim Zugriff auf die abhängigen Services Ihrer Anwendung.
- Fehler beim Serialisieren von Daten. Wenn der Standard-Serializer Ihre Anwendungsdaten nicht serialisieren kann, schlägt die Anwendung fehl. Informationen zur Verwendung eines benutzerdefinierten Serializers in Ihrer Anwendung finden Sie unter Benutzerdefinierte Serializer in der Apache-Flink-Dokumentation.
- Fehler wegen Speichermangel.
- Spitzen oder stetiger Anstieg der folgenden Metriken:
  - `heapMemoryUtilization`
  - `oldGenerationGCTime`
  - `oldGenerationGCCount`
  - `lastCheckpointSize`
  - `lastCheckpointDuration`

Weitere Informationen finden Sie unter Überwachen des Checkpointings in der Apache-Flink-Dokumentation.

Ursachen und Lösungen

Die Fehlermeldungen im Anwendungsprotokoll zeigen die Ursache für direkte Fehler. Vorübergehende Fehler können folgende Ursachen haben:

- Ihre Anwendung verfügt über eine unzureichende KPU-Bereitstellung. Informationen zur Erhöhung der Anwendungsbereitstellung finden Sie unter Skalierung.
- Die Größe des Anwendungszustands ist zu hoch. Sie können die Größe Ihres Anwendungszustands anhand der `lastCheckpointSize`-Metrik überwachen.
- Die Zustandsdaten Ihrer Anwendung sind ungleich auf die Schlüssel verteilt. Wenn Ihre Anwendung den `KeyBy`-Operator verwendet, stellen Sie sicher, dass Ihre eingehenden Daten gleichmäßig auf die Schlüssel aufgeteilt werden. Wenn die meisten Daten einem einzigen Schlüssel zugewiesen werden, entsteht ein Engpass, der zu Fehlern führt.
- Ihre Anwendung leidet unter einem Gegendruck im Speicher oder bei der Garbage Collection. Überwachen Sie `heapMemoryUtilization`, `oldGenerationGCTime` und `oldGenerationGCCount` Ihrer Anwendung auf Spitzen oder stetig steigende Werte.

Checkpoint-Fehler für Apache-Beam-Anwendung

Wenn Ihre Beam-Anwendung so konfiguriert ist, dass shutdownSourcesAfterIdleMs auf 0 ms gesetzt ist, können Checkpoints möglicherweise nicht ausgelöst werden, da sich die Aufgaben im Zustand „Abgeschlossen“ befinden. In diesem Abschnitt werden Symptome und Lösungen für diesen Zustand beschrieben.

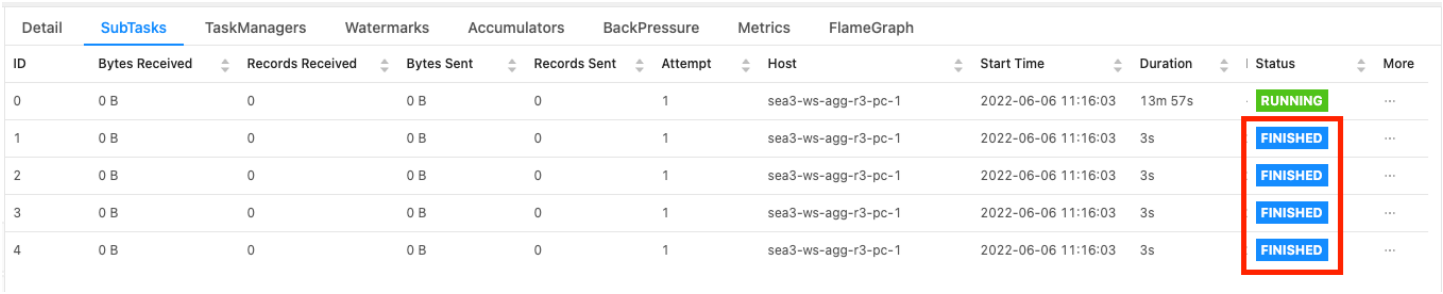
Symptom

Gehen Sie zu den CloudWatch-Protokollen Ihrer Anwendung, die Managed Service für Apache Flink nutzt, und überprüfen Sie, ob die folgende Protokollnachricht protokolliert wurde. Die folgende Protokollmeldung weist darauf hin, dass der Checkpoint nicht ausgelöst werden konnte, da einige Aufgaben abgeschlossen wurden.

```
                {
                "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"
```

```
        }
```

Dies kann auch im Flink-Dashboard gefunden werden, wo einige Aufgaben den Zustand „ABGESCHLOSSEN“ erreicht haben und Checkpointing nicht mehr möglich ist.



Ursache

shutdownSourcesAfterIdleMs ist eine Beam-Konfigurationsvariable, die Quellen herunterfährt, die für die konfigurierte Zeit von Millisekunden inaktiv waren. Sobald eine Quelle heruntergefahren wurde, ist Checkpointing nicht mehr möglich. Dies könnte zu einem Checkpoint-Fehler führen.

Einer der Gründe dafür, dass Aufgaben in den Zustand „ABGESCHLOSSEN“ wechseln, ist, wenn shutdownSourcesAfterIdleMs auf 0 ms gesetzt ist, was bedeutet, dass Aufgaben, die sich im Leerlauf befinden, sofort heruntergefahren werden.

Lösung

Um zu verhindern, dass Aufgaben sofort in den Zustand „ABGESCHLOSSEN“ wechseln, setzen Sie shutdownSourcesAfterIdleMs auf Long.MAX_VALUE. Es gibt zwei Methoden dafür:

- Option 1: Wenn Ihre Beam-Konfiguration auf der Konfigurationsseite Ihrer Anwendung, die Managed Service für Apache Flink nutzt, festgelegt ist, können Sie ein neues Schlüssel-Wert-Paar hinzufügen, um shutdownSourcesAfterIdleMs wie folgt festzulegen:

- Option 2: Wenn Ihre Beam-Konfiguration in Ihrer JAR-Datei festgelegt ist, können Sie shutdownSourcesAfterIdleMs wie folgt festlegen:

```
                        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
```

Gegendruck

Flink nutzt Gegendruck, um die Verarbeitungsgeschwindigkeit einzelner Operatoren anzupassen.

Der Operator kann aus vielen Gründen Schwierigkeiten haben, das Nachrichtenvolumen, das er empfängt, weiter zu verarbeiten. Der Vorgang benötigt möglicherweise mehr CPU-Ressourcen, als der Operator zur Verfügung hat. Der Operator wartet möglicherweise, bis die E/A-Operationen abgeschlossen sind. Wenn ein Operator Ereignisse nicht schnell genug verarbeiten kann, entsteht ein Gegendruck bei den vorgeschalteten Operatoren, die in den langsamen Operator einspeisen. Dies führt dazu, dass die vorgelagerten Operatoren langsamer werden, wodurch sich der Gegendruck zur Quelle weiter ausbreiten kann und die Quelle sich an den Gesamtdurchsatz der Anwendung anpasst, indem sie ebenfalls langsamer wird. Eine ausführlichere Beschreibung von Gegendruck und seiner Funktionsweise finden Sie unter So handhabt Apache Flink™ Gegendruck.

Wenn Sie wissen, welche Operatoren in einer Anwendung langsam sind, erhalten Sie wichtige Informationen, um die Ursache von Leistungsproblemen in der Anwendung zu verstehen. Informationen zum Gegendruck werden über das Flink-Dashboard angezeigt. Um den langsamen Operator zu identifizieren, suchen Sie nach dem Operator mit einem hohen Gegendruckwert, der einer Senke am nächsten ist (im folgenden Beispiel Operator B). Der Operator, der die Langsamkeit verursacht, ist dann einer der nachgeschalteten Operatoren (im Beispiel Operator C). B könnte Ereignisse schneller verarbeiten, gerät jedoch unter Druck, da er die Ausgabe nicht an den langsamen Operator C weiterleiten kann.

```
A (backpressured 93%) -> B (backpressured 85%) -> C (backpressured 11%) -> D
 (backpressured 0%)
```

Sobald Sie den langsamen Operator identifiziert haben, versuchen Sie zu verstehen, warum er langsam ist. Es kann viele Gründe dafür geben, und manchmal ist nicht klar, was das Problem ist. Es kann Tage des Debuggens und Profilings erfordern, um das Problem zu lösen. Im Folgenden sind einige offensichtliche und häufigere Gründe aufgeführt, von denen einige im Folgenden näher erläutert werden:

- Der Operator führt langsame E/A durch, z. B. Netzwerkaufrufe (erwägen Sie stattdessen die Verwendung von AsyncIO).
- Die Daten sind verzerrt und ein Operator empfängt mehr Ereignisse als andere (überprüfen Sie dies, indem Sie sich die Anzahl der eingehenden/ausgehenden Nachrichten einzelner Unteraufgaben (d. h. Instanzen desselben Operators) im Flink-Dashboard ansehen.
- Es handelt sich um einen ressourcenintensiven Vorgang (wenn es keine verzerrte Datenverteilung gibt, sollten Sie erwägen, bei CPU-/speichergebundener Arbeit horizontal zu skalieren oder bei E/A-gebundener Arbeit den Wert von `ParallelismPerKPU` zu erhöhen).
- Umfangreiche Protokollierung durch den Operator (reduzieren Sie die Protokollierung auf ein Minimum für Produktionsanwendungen oder erwägen Sie, stattdessen die Debug-Ausgabe an einen Datenstrom zu senden).

Testen des Durchsatzes mit der Verwurfsenke

Die Verwurfsenke ignoriert einfach alle Ereignisse, die sie empfängt, während die Anwendung weiterhin ausgeführt wird (eine Anwendung ohne Senke kann nicht ausgeführt werden). Dies ist sehr nützlich für Durchsatztests, zum Profiling und um zu überprüfen, ob die Anwendung ordnungsgemäß skaliert. Es ist außerdem eine sehr pragmatische Plausibilitätsprüfung, um zu überprüfen, ob die Senken einen Gegendruck erzeugen oder die Anwendung (aber die bloße Überprüfung der Gegendruckmetriken ist oft einfacher und unkomplizierter).

Indem Sie alle Senken einer Anwendung durch eine Verwurfsenke ersetzen und eine Mock-Quelle erstellen, die Daten generiert, die Produktionsdaten ähneln, können Sie den maximalen Durchsatz der Anwendung für eine bestimmte Parallelitätseinstellung messen. Sie können dann auch die Parallelität erhöhen, um sicherzustellen, dass die Anwendung ordnungsgemäß skaliert und keinen Engpass aufweist, der erst bei höherem Durchsatz auftritt (z. B. aufgrund einer verzerrten Datenverteilung).

Verzerrte Datenverteilung

Eine Flink-Anwendung wird auf einem Cluster verteilt ausgeführt. Um horizontal auf mehrere Knoten zu skalieren, verwendet Flink das Konzept der verschlüsselten Streams, was im Wesentlichen bedeutet, dass die Ereignisse eines Streams nach einem bestimmten Schlüssel, z. B. einer Kunden-ID, partitioniert werden und Flink dann verschiedene Partitionen auf verschiedenen Knoten verarbeiten kann. Viele der Flink-Operatoren werden dann auf der Grundlage dieser Partitionen ausgewertet, z. B. Keyed Windows, Process Functions und Async I/O.

Die Auswahl eines Partitionsschlüssels hängt häufig von der Geschäftslogik ab. Gleichzeitig gelten viele der Best Practices für z. B. DynamoDB und Spark auch für Flink, darunter:

- Sicherstellung einer hohen Kardinalität der Partitionsschlüssel
- Vermeidung von Verzerrungen im Ereignisvolumen zwischen Partitionen

Sie können Verzerrungen in den Partitionen erkennen, indem Sie die empfangenen/gesendeten Datensätze von Unteraufgaben (d. h. Instances desselben Operators) im Flink-Dashboard vergleichen. Darüber hinaus kann die Überwachung von Managed Service für Apache Flink so konfiguriert werden, dass Metriken auch für `numRecordsIn/Out` und `numRecordsInPerSecond/OutPerSecond` auf Unteraufgabenebene verfügbar gemacht werden.

Zustandsverzerrung

Bei zustandsbehafteten Operatoren, d. h. Operatoren, die den Zustand für ihre Geschäftslogik verwerten, wie z. B. Fenster, führt eine Verzerrung der Datenverteilung immer zu einer Zustandsverzerrung. Einige Unteraufgaben empfangen aufgrund der Verzerrung der Datenverteilung mehr Ereignisse als andere und behalten daher auch mehr Daten im Zustand. Aber selbst bei einer Anwendung mit gleichmäßig ausgeglichenen Partitionen kann es zu Abweichungen bei der Menge der im Zustand gespeicherten Daten kommen. Beispielsweise können bei Sitzungsfenstern einige Benutzer bzw. Sitzungen viel länger sein als andere. Wenn die längeren Sitzungen zufällig Teil derselben Partition sind, kann dies zu einem Ungleichgewicht der Zustandsgröße führen, die von verschiedenen Unteraufgaben desselben Operators verwaltet wird.

Zustandsverzerrungen erhöhen nicht nur die Arbeitsspeicher- und Festplattenressourcen, die für einzelne Unteraufgaben benötigt werden, sondern sie können auch die Gesamtleistung der Anwendung verringern. Wenn eine Anwendung einen Checkpoint oder Savepoint verwendet, wird der Operatorzustand in Amazon S3 gespeichert, um den Zustand vor Knoten- oder Cluster-Fehlern zu schützen. Während dieses Vorgangs (insbesondere bei der Genau-einmal-Semantik,

die standardmäßig in Managed Service für Apache Flink aktiviert ist) kommt die Verarbeitung aus externer Sicht zum Stillstand, bis der Checkpoint/Savepoint abgeschlossen ist. Bei einer verzerrten Datenverteilung kann die Zeit bis zum Abschluss des Vorgangs an eine einzelne Unteraufgabe gebunden sein, die eine besonders hohe Zustandsmenge angesammelt hat. In extremen Fällen kann das Erstellen von Checkpoints/Savepoints fehlschlagen, weil eine einzelne Unteraufgabe den Zustand nicht beibehalten kann.

Ähnlich wie bei einer verzerrten Datenverteilung kann auch eine Zustandsverzerrung eine Anwendung erheblich verlangsamen.

Um Zustandsverzerrungen zu identifizieren, können Sie das Flink-Dashboard nutzen. Suchen Sie in den Details nach einem aktuellen Checkpoint oder Savepoint und vergleichen Sie die Datenmenge, die für einzelne Unteraufgaben gespeichert wurde.

Integration mit Ressourcen in verschiedenen Regionen

Sie können `StreamingFileSink` für das Schreiben in einen Amazon-S3-Bucket in einer von Ihrer Anwendung, die Managed Service für Apache Flink nutzt, abweichenden Region über eine Einstellung aktivieren, die für die regionsübergreifende Replikation in der Flink-Konfiguration erforderlich ist. Reichen Sie dazu ein Support-Ticket im AWS Support Center ein.

Dokumentverlauf für Amazon Managed Service für Apache Flink

In der folgenden Tabelle werden die wichtigen Änderungen an der Dokumentation seit der letzten Version von Managed Service für Apache Flink beschrieben.

- API-Version: 2018-05-23
- Letzte Aktualisierung der Dokumentation: 30. August 2023

Änderung	Beschreibung	Datum
Kinesis Data Analytics ist jetzt als Managed Service für Apache Flink bekannt	Es gibt keine Änderungen an den Service-Endpunkten, APIs, der -Befehlszeilenschnittstelle, IAM-Zugriffsrichtlinien, CloudWatch Metriken oder den AWS Fakturierungs-Dashboards. Ihre vorhandenen Anwendungen funktionieren weiterhin wie zuvor. Weitere Informationen finden Sie unter Was ist Managed Service für Apache Flink?	30. August 2023
Support für Apache Flink Version 1.15.2	Managed Service für Apache Flink unterstützt jetzt Anwendungen, die Apache Flink Version 1.15.2 verwenden. Erstellen Sie Kinesis Data Analytics-Anwendungen mithilfe der Apache Flink Tabellen-API. Weitere Informationen finden Sie unter Erstellen von Anwendungen.	22. November 2022

Änderung	Beschreibung	Datum
Support für Apache Flink Version 1.13.2	Managed Service für Apache Flink unterstützt jetzt Anwendungen, die Apache Flink Version 1.13.2 verwenden. Erstellen Sie Kinesis Data Analytics-Anwendungen mithilfe der Apache Flink Tabellen-API. Weitere Informationen finden Sie unter Erste Schritte: Flink 1.13.2.	13. Oktober 2021
Unterstützung für Python	Managed Service für Apache Flink unterstützt jetzt Anwendungen, die Python mit der Apache Flink Tabellen-API und SQL verwenden. Weitere Informationen finden Sie unter Verwenden von Python.	25. März 2021
Unterstützung für Apache Flink 1.11.1	Managed Service für Apache Flink unterstützt jetzt Anwendungen, die Apache Flink 1.11.1 verwenden. Erstellen Sie Kinesis Data Analytics-Anwendungen mithilfe der Apache Flink Tabellen-API. Weitere Informationen finden Sie unter Erstellen von Anwendungen.	19. November 2020

Änderung	Beschreibung	Datum
Apache Flink-Dashboard	Verwenden Sie das Apache Flink-Dashboard, um den Zustand und die Leistung von Anwendungen zu überwache n. Weitere Informationen finden Sie unter Apache Flink-Dashboard.	19. November 2020
EFO Verbraucher	Erstellen Sie Anwendung en, die einen Enhanced Fan-Out (EFO)-Verbraucher verwenden, um aus einem Kinesis Stream zu lesen. Weitere Informationen finden Sie unter EFO-Verbraucher.	6. Oktober 2020
Apache Beam	Erstellen Sie Anwendung en, die Apache Beam zur Verarbeitung von Streaming -Daten verwenden. Weitere Informationen finden Sie unter Verwenden von CloudForm ation mit Managed Service für Apache Flink.	15. September 2020
Leistung	Wie man Probleme mit der Anwendungsleistung behebt und wie man eine performan te Anwendung erstellt. Weitere Informationen finden Sie unter Leistung.	21. Juli 2020

Änderung	Beschreibung	Datum
Benutzerdefinierter Keystore	Wie Sie auf einen Amazon MSK-Cluster zugreifen, der einen benutzerdefinierten Keystore für die Verschlüs selung bei der Übertragu ng verwendet. Weitere Informationen finden Sie unter Benutzerdefinierter Truststore.	10. Juni 2020
CloudWatch Alarme	Empfehlungen zum Erstellen von CloudWatch Alarmen mit Managed Service für Apache Flink. Weitere Informationen finden Sie unter Alarme.	5. Juni 2020
Neue CloudWatch Metriken	Managed Service für Apache Flink gibt jetzt 22 Metriken an Amazon CloudWatch Metrics aus. Weitere Informationen finden Sie unter Metriken und Dimensionen in Managed Service für Apache Flink.	12. Mai 2020
Benutzerdefinierte CloudWatc h Metriken	Definieren Sie anwendung sspezifische Metriken und geben Sie sie an Amazon CloudWatch Metrics aus. Weitere Informationen finden Sie unter Benutzerdefinierte Metriken.	12. Mai 2020

Änderung	Beschreibung	Datum
Beispiel: Aus einem Kinesis Stream in einem anderen Konto lesen	Erfahren Sie, wie Sie in Ihrer Managed Service für Apache Flink-Anwendung auf einen Kinesis Stream in einem anderen AWS-Konto zugreifen können. Weitere Informationen finden Sie unter Kontoüber greifend.	30. März 2020
Unterstützung für Apache Flink 1.8.2	Managed Service für Apache Flink unterstützt jetzt Anwendungen, die Apache Flink 1.8.2 verwenden . Verwenden Sie den Flink-StreamingFileSink Konnektor , um die Ausgabe direkt in S3 zu schreiben. Weitere Informationen finden Sie unter Erstellen von Anwendungen.	17. Dezember 2019
Managed Service für Apache Flink VPC	Konfigurieren Sie eine Anwendung, die Managed Service für Apache Flink nutzt, auf solche Weise, dass sie eine Verbindung zu einer virtuellen privaten Cloud (VPC) herstellt. Weitere Informationen finden Sie unter Verwenden einer Amazon VPC.	25. November 2019

Änderung	Beschreibung	Datum
Bewährte Methoden für Managed Service für Apache Flink	Bewährte Methoden für die Erstellung und Verwaltung von Managed Service für Apache Flink-Anwendungen. Weitere Informationen finden Sie unter Bewährte Methoden.	14. Oktober 2019
Analysieren von Managed Service für Apache Flink-Anwendungsprotokollen	Verwenden Sie CloudWatch Logs Insights, um Ihre Anwendung von Managed Service für Apache Flink zu überwachen. Weitere Informationen finden Sie unter Analysieren von Protokollen.	26. Juni 2019
Managed Service für Apache Flink-Anwendung Laufzeit-Eigenschaften	Arbeiten Sie mit Laufzeit-Eigenschaften in Managed Service für Apache Flink. Weitere Informationen finden Sie unter Laufzeiteigenschaften.	24. Juni 2019
Markieren von Managed Service für Apache Flink-Anwendungen	Verwenden Sie Anwendungs-Tagging zum Bestimmen der Kosten pro Anwendung, zur Kontrolle des Zugriffs oder für benutzerdefinierte Zwecke. Weitere Informationen finden Sie unter Verwenden von Tagging.	8. Mai 2019

Änderung	Beschreibung	Datum
Managed Service für Apache Flink-Anwendungen	Beispielanwendungen für Managed Service für Apache Flink, die Fensterop eratoren demonstrieren und Ausgaben in - CloudWatch Protokolle schreiben. Weitere Informationen finden Sie unter Beispiele.	1. Mai 2019
Protokollieren von Managed Service für Apache Flink API-Aufrufen mit AWS CloudTrail	Managed Service für Apache Flink ist in AWS CloudTrail integriert, einen Service, der die Aktionen eines Benutzers , einer Rolle oder eines AWS-Services in Managed Service für Apache Flink aufzeichnet. Weitere Informationen finden Sie unter Verwenden von AWS CloudTrail.	22. März 2019
Anwendung erstellen (Kinesis Data Firehose Sink)	Übung zur Erstellung eines Managed Service für Apache Flink mit einem Amazon Kinesis Data Stream als Quelle und einem Stream von Amazon Kinesis Data Firehose als Senke. Weitere Informationen finden Sie unter Kinesis Data Firehose Senke.	13. Dezember 2018
Öffentliche Freigabe	Dies ist die erste Version des Entwicklerhandbuchs für Managed Service für Apache Flink für Java-Anwendungen.	27. November 2018

Beispielcode für Managed Service für Apache Flink

Dieses Thema enthält Beispielanforderungsblöcke für Managed Service für Apache Flink-Aktionen.

Um JSON als Eingabe für eine Aktion mit dem AWS Command Line Interface (AWS CLI) zu verwenden, speichern Sie die Anfrage in einer JSON-Datei. Übergeben Sie dann den Dateinamen mithilfe des Parameters `--cli-input-json` an die Aktion.

Das folgende Beispiel zeigt, wie Sie eine JSON-Datei mit einer Aktion verwenden.

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

Weitere Informationen zur Verwendung von JSON mit dem AWS CLI finden Sie unter Generieren der CLI-Skeleton- und CLI-Input-JSON-Parameter im AWS Command Line InterfaceBenutzerhandbuch.

Themen

- AddApplicationCloudWatchLoggingOption
- AddApplicationInput
- AddApplicationInputProcessingConfiguration
- AddApplicationOutput
- AddApplicationReferenceDataSource
- AddApplicationVpcConfiguration
- CreateApplication
- CreateApplicationSnapshot
- DeleteApplication
- DeleteApplicationCloudWatchLoggingOption
- DeleteApplicationInputProcessingConfiguration
- DeleteApplicationOutput
- DeleteApplicationReferenceDataSource
- DeleteApplicationSnapshot
- DeleteApplicationVpcConfiguration
- DescribeApplication
- DescribeApplicationSnapshot
- DiscoverInputSchema

- ListApplications
- ListApplicationSnapshots
- StartApplication
- StopApplication
- UpdateApplication

AddApplicationCloudWatchLoggingOption

Der folgende Beispiel-Anforderungscode für die Aktion AddApplicationCloudWatchLoggingOption fügt einer Managed Service für Apache Flink-Anwendung eine Amazon CloudWatch-Protokollierungsoption hinzu:

```
{
    "ApplicationName": "MyApplication",
    "CloudWatchLoggingOption": {
        "LogStreamARN": "arn:aws:logs:us-east-1:123456789123:log-group:my-log-
group:log-stream:My-LogStream"
    },
    "CurrentApplicationVersionId": 2
}
```

AddApplicationInput

Der folgende Beispiel-Anforderungscode für die Aktion AddApplicationInput fügt einer Managed Service für Apache Flink-Anwendung eine Anwendungseingabe hinzu:

```
{
   "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

Der folgende Beispiel-Anforderungscode für die Aktion AddApplicationInputProcessingConfiguration fügt einer Managed Service für Apache Flink-Anwendung eine Konfiguration für die Verarbeitung von Anwendungseingaben hinzu:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 2,
   "InputId": "2.1",
   "InputProcessingConfiguration": {
      "InputLambdaProcessor": {
         "ResourceARN": "arn:aws:lambda:us-
east-1:012345678901:function:MyLambdaFunction"
      }
   }
}
```

AddApplicationOutput

Der folgende Beispiel-Anforderungscode für die Aktion AddApplicationOutput fügt einen Kinesis Data Stream als Anwendungsausgabe zu einer Managed Service für Apache Flink-Anwendung hinzu:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 2,
   "Output": {
      "DestinationSchema": {
         "RecordFormatType": "JSON"
      },
      "KinesisStreamsOutput": {
         "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleOutputStream"
      },
      "Name": "DESTINATION_SQL_STREAM"
   }
}
```

AddApplicationReferenceDataSource

Der folgende Beispiel-Anforderungscode für die Aktion AddApplicationReferenceDataSource fügt einer Managed Service für Apache Flink-Anwendung eine CSV-Anwendungsreferenzdatenquelle hinzu:

```
{
   "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

Der folgende Beispiel-Anforderungscode für die Aktion AddApplicationVpcConfiguration fügt einer vorhandenen Anwendung eine VPC-Konfiguration hinzu:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfiguration": {
      "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
      "SubnetIds": [ "subnet-0123456789abcdef0" ]
   }
}
```

CreateApplication

Der folgende Beispiel-Anforderungscode für die Aktion CreateApplication erstellt eine Managed Service für Apache Flink-Anwendung:

```
{
  "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

Der folgende Beispiel-Anforderungscode für die Aktion CreateApplicationSnapshot erstellt eine Momentaufnahme des Anwendungsstatus:

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

DeleteApplication

Der folgende Beispiel-Anforderungscode für die Aktion DeleteApplication löscht eine Managed Service für Apache Flink-Anwendung:

```
{"ApplicationName": "MyApplication",
"CreateTimestamp": 12345678912}
```

DeleteApplicationCloudWatchLoggingOption

Der folgende Beispiel-Anforderungscode für die Aktion DeleteApplicationCloudWatchLoggingOption löscht eine Amazon CloudWatch Protokollierungsoption aus einer Managed Service für Apache Flink-Anwendung:

```
{
    "ApplicationName": "MyApplication",
    "CloudWatchLoggingOptionId": "3.1"
    "CurrentApplicationVersionId": 3
}
```

DeleteApplicationInputProcessingConfiguration

Der folgende Beispiel-Anforderungscode für die Aktion DeleteApplicationInputProcessingConfiguration entfernt eine Konfiguration für die Eingabeverarbeitung aus einer Managed Service für Apache Flink-Anwendung:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "InputId": "2.1"
}
```

DeleteApplicationOutput

Der folgende Beispiel-Anforderungscode für die Aktion DeleteApplicationOutput entfernt eine Anwendungsausgabe aus einer Managed Service für Apache Flink-Anwendung:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 4,
   "OutputId": "4.1"
}
```

DeleteApplicationReferenceDataSource

Der folgende Beispiel-Anforderungscode für die Aktion DeleteApplicationReferenceDataSource entfernt eine Anwendungsreferenz-Datenquelle aus einer Managed Service für Apache Flink-Anwendung:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 5,
   "ReferenceId": "5.1"
}
```

DeleteApplicationSnapshot

Der folgende Beispiel-Anforderungscode für die Aktion DeleteApplicationSnapshot löscht einen Snapshot des Anwendungsstatus:

```
{
   "ApplicationName": "MyApplication",
   "SnapshotCreationTimestamp": 12345678912,
   "SnapshotName": "MySnapshot"
}
```

DeleteApplicationVpcConfiguration

Der folgende Beispiel-Anforderungscode für die Aktion DeleteApplicationVpcConfiguration entfernt eine vorhandene VPC-Konfiguration aus einer Anwendung:

```
{
   "ApplicationName": "MyApplication",
   "CurrentApplicationVersionId": 9,
   "VpcConfigurationId": "1.1"
}
```

DescribeApplication

Der folgende Beispiel-Anforderungscode für die Aktion DescribeApplication gibt Details zu einer Managed Service für Apache Flink-Anwendung zurück:

```
{"ApplicationName": "MyApplication"}
```

DescribeApplicationSnapshot

Der folgende Beispiel-Anforderungscode für die Aktion DescribeApplicationSnapshot gibt Details zu einer Momentaufnahme des Anwendungsstatus zurück:

```
{
   "ApplicationName": "MyApplication",
   "SnapshotName": "MySnapshot"
```

```
}
```

DiscoverInputSchema

Der folgende Beispiel-Anforderungscode für die Aktion DiscoverInputSchema generiert ein Schema aus einer Streaming-Quelle:

```
{
   "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"
}
```

Der folgende Beispiel-Anforderungscode für die Aktion DiscoverInputSchema generiert ein Schema aus einer Referenzquelle:

```
{
   "S3Configuration": {
      "BucketARN": "arn:aws:s3:::mybucket",
      "FileKey": "TickerReference.csv"
   },
   "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole"
}
```

ListApplications

Der folgende Beispiel-Anforderungscode für die Aktion ListApplications gibt eine Liste der Managed Service für Apache Flink-Anwendungen in Ihrem Konto zurück:

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

ListApplicationSnapshots

Der folgende Beispiel-Anforderungscode für die Aktion ListApplicationSnapshots gibt eine Liste von Snapshots des Anwendungsstatus zurück:

```
{"ApplicationName": "MyApplication",
   "Limit": 50,
   "NextToken": "aBcDeFgHiJkLmNoPqRsTuVwXyZ0123"
}
```

StartApplication

Der folgende Beispiel-Anforderungscode für die Aktion StartApplication startet eine Managed Service für Apache Flink-Anwendung und lädt den Anwendungsstatus aus dem letzten Snapshot (falls vorhanden):

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

StopApplication

Der folgende Beispiel-Anforderungscode für die Aktion API_StopApplication beendet eine Managed Service für Apache Flink-Anwendung:

```
{"ApplicationName": "MyApplication"}
```

UpdateApplication

Der folgende Beispiel-Anforderungscode für die Aktion UpdateApplication aktualisiert eine Managed Service for Apache Flink-Anwendung, um den Speicherort des Anwendungscodes zu ändern:

```
{"ApplicationName": "MyApplication",
"CurrentApplicationVersionId": 1,
"ApplicationConfigurationUpdate": {
      "ApplicationCodeConfigurationUpdate": {
         "CodeContentTypeUpdate": "ZIPFILE",
         "CodeContentUpdate": {
            "S3ContentLocationUpdate": {
               "BucketARNUpdate": "arn:aws:s3:::my_new_bucket",
               "FileKeyUpdate": "my_new_code.zip",
               "ObjectVersionUpdate": "2"
         }
      }
   }
}
```

Managed Service für Apache Flink API-Referenz

Informationen zu den APIs, die Managed Service für Apache Flink bereitstellt, finden Sie unter Managed Service für Apache Flink API-Referenz.