



Guida per gli sviluppatori del servizio gestito per Apache Flink

Servizio gestito per Apache Flink



Servizio gestito per Apache Flink: Guida per gli sviluppatori del servizio gestito per Apache Flink

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

I marchi e l'immagine commerciale di Amazon non possono essere utilizzati in relazione a prodotti o servizi che non siano di Amazon, in una qualsiasi modalità che possa causare confusione tra i clienti o in una qualsiasi modalità che denigri o discrediti Amazon. Tutti gli altri marchi non di proprietà di Amazon sono di proprietà delle rispettive aziende, che possono o meno essere associate, collegate o sponsorizzate da Amazon.

Table of Contents

.....	xvii
Cos'è il servizio gestito per Apache Flink?	1
Decidi se utilizzare Managed Service per Apache Flink o Managed Service per Apache Flink Studio	1
Scegliete quale Apache Flink utilizzare in Managed Service for Apache Flink APIs	3
Scegli un'API Flink	3
Inizia a usare le applicazioni di streaming di dati	5
Come funziona	6
Programma la tua applicazione Apache Flink	6
DataStream API	6
API Table	7
Crea il tuo servizio gestito per l'applicazione Apache Flink	8
Creazione di un'applicazione	8
Crea il codice dell'applicazione Managed Service for Apache Flink	8
Crea la tua applicazione Managed Service for Apache Flink	9
Avvia l'applicazione Managed Service for Apache Flink	10
Verifica l'applicazione Managed Service for Apache Flink	11
Abilita i rollback del sistema	11
Esecuzione di un'applicazione	14
Identifica lo stato della candidatura e del lavoro	14
Esegui carichi di lavoro in batch	16
Risorse dell'applicazione	16
Managed Service per le risorse delle applicazioni Apache Flink	16
Risorse dell'applicazione Apache Flink	17
Prezzi	18
Come funziona	16
Regione AWS disponibilità	19
Esempi di prezzi	20
Esamina i componenti DataStream dell'API	24
Connectors (Connettori)	24
Operatori	34
Monitoraggio degli eventi	36
Componenti dell'API Table	36
Connettori API per tabelle	37

Attributi temporali dell'API della tabella	38
Usa Python	39
Programma la tua applicazione Python	39
Crea la tua applicazione Python	42
Monitora la tua applicazione Python	43
Usa le proprietà di runtime	45
Gestisci le proprietà di runtime utilizzando la console	45
Gestisci le proprietà di runtime utilizzando la CLI	46
Accedi alle proprietà di runtime in un'applicazione Managed Service for Apache Flink	49
Usa i connettori Apache Flink	50
Problemi noti	53
Implementa la tolleranza agli	53
Configura il checkpoint in Managed Service for Apache Flink	53
Consulta gli esempi di API di checkpointing	54
Gestisci i backup delle applicazioni utilizzando le istantanee	57
Gestisci la creazione automatica di istantanee	58
Esegui il ripristino da un'istananea che contiene dati di stato incompatibili	59
Consulta gli esempi di API snapshot	60
Usa gli aggiornamenti di versione in loco per Apache Flink	62
Aggiorna le applicazioni	63
Effettua l'aggiornamento a una nuova versione	64
Ripristina gli aggiornamenti delle applicazioni	70
Best practice	71
Problemi noti	71
Implementa la scalabilità delle applicazioni	73
Configura il parallelismo dell'applicazione e la KPU ParallelismPer	73
Allocazione delle unità di elaborazione Kinesis	74
Aggiorna il parallelismo dell'applicazione	75
Usa il ridimensionamento automatico	76
Considerazioni su maxParallelism	79
Aggiungi tag alle applicazioni	80
Aggiungi tag quando viene creata un'applicazione	81
Aggiungi o aggiorna i tag per un'applicazione esistente	81
Elenca i tag di un'applicazione	82
Rimuovere i tag da un'applicazione	82
Usa CloudFormation	82

Prima di iniziare	82
Scrivere una funzione Lambda	82
Creare un ruolo Lambda	84
Richiama la funzione Lambda	85
Rivedi un esempio esteso	86
Usa la dashboard Apache Flink	92
Accedi alla dashboard Apache Flink della tua applicazione	92
Versioni di rilascio	94
Servizio gestito Amazon per Apache Flink 1.20	96
Funzionalità supportate	96
Componenti	97
Problemi noti	98
Servizio gestito Amazon per Apache Flink 1.19	98
Funzionalità supportate	99
Modifiche ad Amazon Managed Service per Apache Flink 1.19.1	101
Componenti	103
Problemi noti	104
Servizio gestito Amazon per Apache Flink 1.18	104
Modifiche al servizio gestito da Amazon per Apache Flink con Apache Flink 1.15	106
Componenti	108
Problemi noti	108
Servizio gestito Amazon per Apache Flink 1.15	109
Modifiche al servizio gestito da Amazon per Apache Flink con Apache Flink 1.15	111
Componenti	108
Problemi noti	112
Versioni precedenti	112
Utilizzo del connettore di flussi Apache Flink Kinesis con versioni precedenti di Apache Flink	114
Creazione di applicazioni con Apache Flink 1.8.2	115
Creazione di applicazioni con Apache Flink 1.6.2	116
Aggiornamento delle applicazioni	117
Connettori disponibili in Apache Flink 1.6.2 e 1.8.2	117
Nozioni di base: Flink 1.13.2	117
Nozioni di base: Flink 1.11.1	143
Nozioni di base: Flink 1.8.2 - obsoleto	171
Nozioni di base: Flink 1.6.2 - deprecazione	197

Esempi legacy	222
Usa i notebook Studio con Managed Service for Apache Flink	396
Usa la versione corretta di Studio Notebook Runtime	397
Crea un taccuino da Studio	398
Esegui un'analisi interattiva dei dati in streaming	399
Interpreti Flink	400
Variabili dell'ambiente tabellare Apache Flink	400
Implementa come applicazione con uno stato durevole	401
Criteri Scala/Python	403
Criteri SQL	403
Autorizzazioni IAM	403
Usa connettori e dipendenze	404
Connettori predefiniti	404
Aggiungi dipendenze e connettori personalizzati	406
Funzioni definite dall'utente	407
Considerazioni sulle funzioni definite dall'utente	408
Abilita il checkpointing	409
Imposta l'intervallo di checkpoint	409
Imposta il tipo di checkpoint	410
Aggiorna Studio Runtime	410
Aggiorna il tuo notebook a un nuovo Studio Runtime	410
Lavora con AWS Glue	415
Proprietà tabella	415
Esempi e tutorial per notebook Studio in Managed Service for Apache Flink	417
Tutorial: crea un notebook Studio in Managed Service per Apache Flink	418
Tutorial: Implementa un notebook Studio come servizio gestito per un'applicazione Apache Flink con uno stato durevole	438
Visualizza esempi di query per analizzare i dati in un notebook Studio	442
Risolvi i problemi relativi ai notebook Studio per Managed Service for Apache Flink	454
Arresta un'applicazione bloccata	454
Implementa come applicazione con uno stato durevole in un VPC senza accesso a Internet	454
Deploy-as-app riduzione delle dimensioni e dei tempi di costruzione	455
Annullare i lavori	457
Riavvia l'interprete Apache Flink	458
Crea policy IAM personalizzate per i notebook Managed Service for Apache Flink Studio	458

AWS Glue	459
CloudWatch Registri	460
Flussi Kinesis	461
Cluster Amazon MSK	463
Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink	464
Esamina i componenti dell'applicazione	172
Completa i prerequisiti richiesti	465
Configurazione di un account	466
Registrati per un Account AWS	119
Crea un utente con accesso amministrativo	119
Concessione dell'accesso programmatico	468
Fase successiva	469
Configura il AWS CLI	470
Approfondimenti	471
Creazione di un'applicazione	471
Crea risorse dipendenti	472
Configurazione dell'ambiente di sviluppo locale	473
Scarica ed esamina il codice Java per lo streaming di Apache Flink	474
Scrivi record di esempio nel flusso di input	479
Esegui l'applicazione localmente	480
Osserva i dati di input e output nei flussi Kinesis	484
Interrompi l'esecuzione locale dell'applicazione	484
Compila e impacchetta il codice dell'applicazione	484
Carica il file JAR del codice dell'applicazione	485
Crea e configura l'applicazione Managed Service for Apache Flink	485
Approfondimenti	493
Pulizia delle risorse	493
Eliminare l'applicazione Managed Service for Apache Flink	493
Eliminare i flussi di dati Kinesis	494
Elimina oggetti e bucket Amazon S3	494
Elimina le tue risorse IAM	494
CloudWatch Elimina le tue risorse	495
Esplora risorse aggiuntive per Apache Flink	495
Esplora risorse aggiuntive	495
Tutorial: Nozioni di base sull'utilizzo della TableAPI nel servizio gestito per Apache Flink	497
Esamina i componenti dell'applicazione	497

Completa i prerequisiti obbligatori	498
Creazione di un'applicazione	499
Creazione delle risorse dipendenti	499
Configurazione dell'ambiente di sviluppo locale	500
Scarica ed esamina il codice Java per lo streaming di Apache Flink	501
Esegui l'applicazione localmente	507
Osservare l'applicazione che scrive i dati in un bucket S3	510
Interruzione dell'applicazione in esecuzione locale	511
Compila e impacchetta il codice dell'applicazione	511
Caricare il file JAR del codice dell'applicazione	511
Creazione e configurazione dell'applicazione del servizio gestito per Apache Flink	512
Approfondimenti	518
Pulizia delle risorse	518
Eliminazione dell'applicazione del servizio gestito per Apache Flink	519
Eliminazione di oggetti e di bucket Amazon S3	519
Eliminazione delle risorse IAM	520
CloudWatch Elimina le tue risorse	520
Approfondimenti	520
Esplora altre risorse	521
Tutorial: Nozioni di base sull'utilizzo del servizio gestito per Apache Flink	522
Esamina i componenti dell'applicazione	522
Soddisfare i prerequisiti	523
Creazione di un'applicazione	525
Creazione delle risorse dipendenti	525
Configurazione dell'ambiente di sviluppo locale	527
Scarica ed esamina il codice Python per lo streaming di Apache Flink	528
Gestione delle dipendenze JAR	531
Scrivi record di esempio nel flusso di input	533
Esegui l'applicazione localmente	534
Osserva i dati di input e output nei flussi Kinesis	537
Arresta l'esecuzione dell'applicazione in locale	537
Package del codice dell'applicazione	537
Caricare il pacchetto dell'applicazione in un bucket Amazon S3	538
Creazione e configurazione dell'applicazione del servizio gestito per Apache Flink	538
Approfondimenti	545
Pulizia delle risorse	545

Eliminazione dell'applicazione del servizio gestito per Apache Flink	545
Eliminazione dei flussi di dati Kinesis	546
Eliminazione di oggetti e bucket Amazon S3	546
Eliminazione delle risorse IAM	547
CloudWatch Elimina le tue risorse	547
Tutorial: Nozioni di base sull'utilizzo di Scala nel servizio gestito per Apache Flink	548
Creazione delle risorse dipendenti	548
Scrivi record di esempio nel flusso di input	549
Scarica ed esamina il codice dell'applicazione	551
Per creare e compilare il codice dell'applicazione	552
Creazione ed esecuzione dell'applicazione (console)	553
Creazione dell'applicazione	553
Configurazione dell'applicazione	554
Modifica la policy IAM	556
Esecuzione dell'applicazione.	557
Interruzione dell'applicazione	558
Creazione ed esecuzione dell'applicazione (CLI)	558
Creazione di una policy di autorizzazione	558
Creazione di una policy IAM	560
Creazione dell'applicazione	561
Avvia l'applicazione	562
Interruzione dell'applicazione	391
Aggiungi un'opzione CloudWatch di registrazione	391
Aggiornamento delle proprietà dell'ambiente	391
Aggiornamento del codice dell'applicazione	392
Eliminazione AWS delle risorse	566
Eliminazione dell'applicazione del servizio gestito per Apache Flink	566
Eliminazione dei flussi di dati Kinesis	566
Eliminazione dell'oggetto Amazon S3 e del bucket	567
Eliminazione delle risorse IAM	567
CloudWatch Elimina le tue risorse	567
Utilizzo di Apache Beam con il servizio gestito per Apache Flink	568
Limitazioni del runner Apache Flink con il servizio gestito per Apache Flink	568
Funzionalità di Apache Beam con il servizio gestito per Apache Flink	569
Creazione di un'applicazione utilizzando Apache Beam	569
Creazione delle risorse dipendenti	570

Scrivi record di esempio nel flusso di input	570
Scarica ed esamina il codice dell'applicazione	571
Per creare il codice dell'applicazione	572
Caricamento del codice Java di streaming di Apache Flink	573
Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink	573
Pulizia	577
Passaggi successivi	579
Workshop di formazione, laboratori e implementazioni di soluzioni	580
Workshop Managed Service per Apache Flink	580
Sviluppa le applicazioni Apache Flink localmente prima di distribuirle su Managed Service for Apache Flink	580
Rilevamento di eventi con il servizio gestito per Apache Flink Studio	581
AWS Soluzione per lo streaming di dati	581
Esercitati a usare un laboratorio Clickstream con Apache Flink e Apache Kafka	581
Configurare il ridimensionamento personalizzato utilizzando Application Auto Scaling	582
Visualizza un esempio di CloudWatch dashboard Amazon	582
Usa modelli per la soluzione di AWS streaming di dati per Amazon MSK	582
Esplora altre soluzioni Managed Service for Apache Flink su GitHub	582
Usa utilità pratiche per Managed Service for Apache Flink	584
Snapshot manager	584
Analisi comparativa	584
Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink	585
Esempi di Java per Managed Service for Apache Flink	585
Esempi in Python per Managed Service for Apache Flink	589
.....	589
Esempi di Scala per Managed Service for Apache Flink	591
Sicurezza nel servizio gestito per Apache Flink	592
Protezione dei dati	593
Crittografia dei dati	593
Identity and Access Management per il servizio gestito per Apache Flink	594
Destinatari	594
Autenticazione con identità	595
Gestione dell'accesso con policy	599
Funzionamento del servizio gestito da Amazon per Apache Flink con IAM	601
Esempi di policy basate su identità	609
Risoluzione dei problemi	612

Prevenzione del problema "confused deputy" tra servizi	614
Convalida della conformità per il servizio gestito per Apache Flink	616
FedRAMP	617
Resilienza e disaster recovery nel servizio gestito per Apache Flink	617
Ripristino di emergenza	618
Controllo delle versioni	618
Sicurezza dell'infrastruttura nel servizio gestito per Apache Flink	619
Best practice di sicurezza per il servizio gestito per Apache Flink	619
Implementazione dell'accesso con privilegi minimi	619
Uso di ruoli IAM per accedere ad altri servizi Amazon	619
Implementazione della crittografia lato server	620
Utilizzalo per monitorare le chiamate API CloudTrail	620
Registrazione e monitoraggio in Amazon Managed Service per Apache Flink	621
Accesso al servizio gestito per Apache Flink	622
Interrogazione dei log con Logs Insights CloudWatch	622
Monitoraggio in Managed Service per Apache Flink	622
Configurare la registrazione delle applicazioni in Managed Service per Apache Flink	624
Configura la CloudWatch registrazione utilizzando la console	624
Configurare la CloudWatch registrazione utilizzando la CLI	625
Controlla i livelli di monitoraggio delle applicazioni	630
Applica le migliori pratiche di registrazione	631
Eseguire la risoluzione dei problemi di registrazione	631
Usa CloudWatch Logs Insights	632
Analizza i log con CloudWatch Logs Insights	632
Esecuzione di una query di esempio	632
Esamina le query di esempio	633
Metriche e dimensioni in Managed Service for Apache Flink	636
Parametri di applicazione	636
Metriche del connettore Kinesis Data Streams	667
Metriche del connettore Amazon MSK	669
Metriche di Apache Zeppelin	671
CloudWatch Visualizza le metriche	672
Imposta i livelli di CloudWatch report sulle metriche	672
Usa metriche personalizzate con Amazon Managed Service per Apache Flink	674
Usa gli CloudWatch allarmi con Amazon Managed Service per Apache Flink	678
Scrivi messaggi personalizzati in Logs CloudWatch	691

Scrivi nei CloudWatch log usando Log4J	692
Scrivi nei log usando CloudWatch J SLF4	693
Registra il servizio gestito per le chiamate API Apache Flink con AWS CloudTrail	694
Informazioni su Managed Service for Apache Flink in CloudTrail	694
Comprendi le voci dei file di registro di Managed Service for Apache Flink	695
Ottimizza le prestazioni	698
Risolvi i problemi relativi alle prestazioni	698
Comprendi il percorso dei dati	698
Soluzioni per la risoluzione dei problemi	699
Utilizza le migliori pratiche in materia di prestazioni	701
Gestire correttamente il dimensionamento	701
Monitoraggio dell'utilizzo delle risorse di dipendenza esterne	704
Avviamento dell'applicazione Apache Flink in locale	704
Monitora le prestazioni	704
Monitora le prestazioni utilizzando le CloudWatch metriche	704
Monitora le prestazioni utilizzando CloudWatch registri e allarmi	704
Servizio gestito per Apache Flink e quota di notebook Studio	706
Gestisci le attività di manutenzione per Managed Service for Apache Flink	708
Scegli una finestra di manutenzione	710
Identifica le istanze di manutenzione	710
Raggiungi la prontezza di produzione per il tuo servizio gestito per le applicazioni Apache Flink ...	712
Effettua il test di caricamento delle tue applicazioni	712
Definisci il parallelismo massimo	713
Impostare un UUID per tutti gli operatori	713
Best practice	714
Riduci al minimo le dimensioni di uber JAR	714
Tolleranza agli errori: checkpoint e savepoint	717
Versioni di connettori non supportate	717
Prestazioni e parallelismo	718
Impostazione del parallelismo per operatore	718
Registrazione	719
Codifica	720
Credenziali root.	720
Lettura da fonti con pochi shard/partizioni	721
Intervallo di aggiornamento del notebook Studio	721
Prestazioni ottimali del notebook Studio	721

Come le strategie di watermark e le partizioni inattive influiscono sulle finestre temporali	722
Riepilogo	723
Esempio	724
Impostare un UUID per tutti gli operatori	733
Aggiungi ServiceResourceTransformer al plugin Maven shade	733
Funzioni statiche di Apache Flink	735
Modello di applicazione Apache Flink	735
Ubicazione della configurazione del modulo	736
Scopri le impostazioni di Apache Flink	737
Configurazione di Apache Flink	737
Backend statale	738
Checkpoint	738
Savepoint	739
Dimensioni del mucchio	740
Debloating del buffer	740
Proprietà di configurazione modificabili di Flink	740
strategia di riavvio	740
Checkpoint e backend di stato	741
Checkpoint	741
Metriche native di RocksDB	741
Opzioni RockSDB	742
Opzioni avanzate di backend di stato	743
Opzioni complete TaskManager	743
Configurazione della memoria	743
RPC/Akka	744
Client	744
Opzioni di cluster avanzate	744
Configurazioni del file system	744
Opzioni avanzate di tolleranza ai guasti	744
Configurazione della memoria	743
Metriche	745
Opzioni avanzate per l'endpoint e il client REST	745
Opzioni di sicurezza SSL avanzate	745
Opzioni di pianificazione avanzate	745
Opzioni avanzate per l'interfaccia utente web di Flink	745
Visualizza le proprietà Flink configurate	745

Configura MSF per accedere alle risorse in un Amazon VPC	746
Concetti di Amazon VPC	746
Autorizzazioni per applicazioni VPC	747
Aggiungi una politica di autorizzazioni per l'accesso a un Amazon VPC	747
Stabilisci l'accesso a Internet e ai servizi per un'applicazione Managed Service for Apache Flink connessa a VPC	748
Informazioni correlate	750
Usa l'API VPC Managed Service for Apache Flink	750
Crea applicazione	750
AddApplicationVpcConfiguration	751
DeleteApplicationVpcConfiguration	751
Aggiorna l'applicazione	752
Esempio: utilizzare un VPC	752
Risolvi i problemi relativi al servizio gestito per Apache Flink	753
Risoluzione dei problemi di sviluppo	753
Le migliori pratiche per il rollback del sistema	754
Le migliori pratiche di configurazione di Hudi	755
Grafici a candela di Apache Flink	755
Problema con il provider di credenziali con il connettore EFO 1.15.2	755
Applicazioni con connettori Kinesis non supportati	756
Errore di compilazione: «Impossibile risolvere le dipendenze per il progetto»	759
Scelta non valida: «kinesisanalyticsv2»	759
UpdateApplication l'azione non consiste nel ricaricare il codice dell'applicazione	759
S3 StreamingFileSink FileNotFoundExceptions	760
FlinkKafkaConsumer problema con stop with savepoint	762
Blocco di Async sink per Flink 1.15	762
L'elaborazione del codice sorgente dei flussi di dati di Amazon Kinesis non funziona correttamente durante il re-sharding	772
Domande frequenti e risoluzione dei problemi relativi ai progetti di incorporamento vettoriale in tempo reale	773
Risoluzione dei problemi di runtime	785
Strumenti per la risoluzione dei problemi	785
Problemi relativi all'applicazione	786
L'applicazione si sta riavviando	790
La velocità effettiva è troppo lenta	793
Crescita statale illimitata	794

Operatori vincolati all'I/O	796
Limitazione della larghezza di banda della rete in upstream o all'origine da un flusso di dati	
Kinesis	796
Checkpoint	797
Il checkpoint è in fase di interruzione	804
Errore al checkpoint per Apache Beam	805
Contropressione	807
Disallineamento dei dati	809
Disallineamento dello stato	809
Integrazione con risorse di diverse regioni	810
Cronologia dei documenti	811
Codice di esempio di API	817
AddApplicationCloudWatchLoggingOption	818
AddApplicationInput	818
AddApplicationInputProcessingConfiguration	819
AddApplicationOutput	820
AddApplicationReferenceDataSource	820
AddApplicationVpcConfiguration	821
CreateApplication	822
CreateApplicationSnapshot	823
DeleteApplication	823
DeleteApplicationCloudWatchLoggingOption	823
DeleteApplicationInputProcessingConfiguration	824
DeleteApplicationOutput	824
DeleteApplicationReferenceDataSource	824
DeleteApplicationSnapshot	825
DeleteApplicationVpcConfiguration	825
DescribeApplication	825
DescribeApplicationSnapshot	825
DiscoverInputSchema	826
ListApplications	826
ListApplicationSnapshots	827
StartApplication	827
StopApplication	827
UpdateApplication	828
Documentazione di riferimento API	829

..... 830

Il servizio gestito da Amazon per Apache Flink era precedentemente noto come Analisi dei dati Amazon Kinesis per Apache Flink.

Le traduzioni sono generate tramite traduzione automatica. In caso di conflitto tra il contenuto di una traduzione e la versione originale in Inglese, quest'ultima prevarrà.

Cos'è Amazon Managed Service per Apache Flink?

Con Amazon Managed Service for Apache Flink, puoi usare Java, Scala, Python o SQL per elaborare e analizzare i dati in streaming. Il servizio consente di creare ed eseguire codice su sorgenti di streaming e fonti statiche per eseguire analisi di serie temporali, alimentare dashboard e metriche in tempo reale.

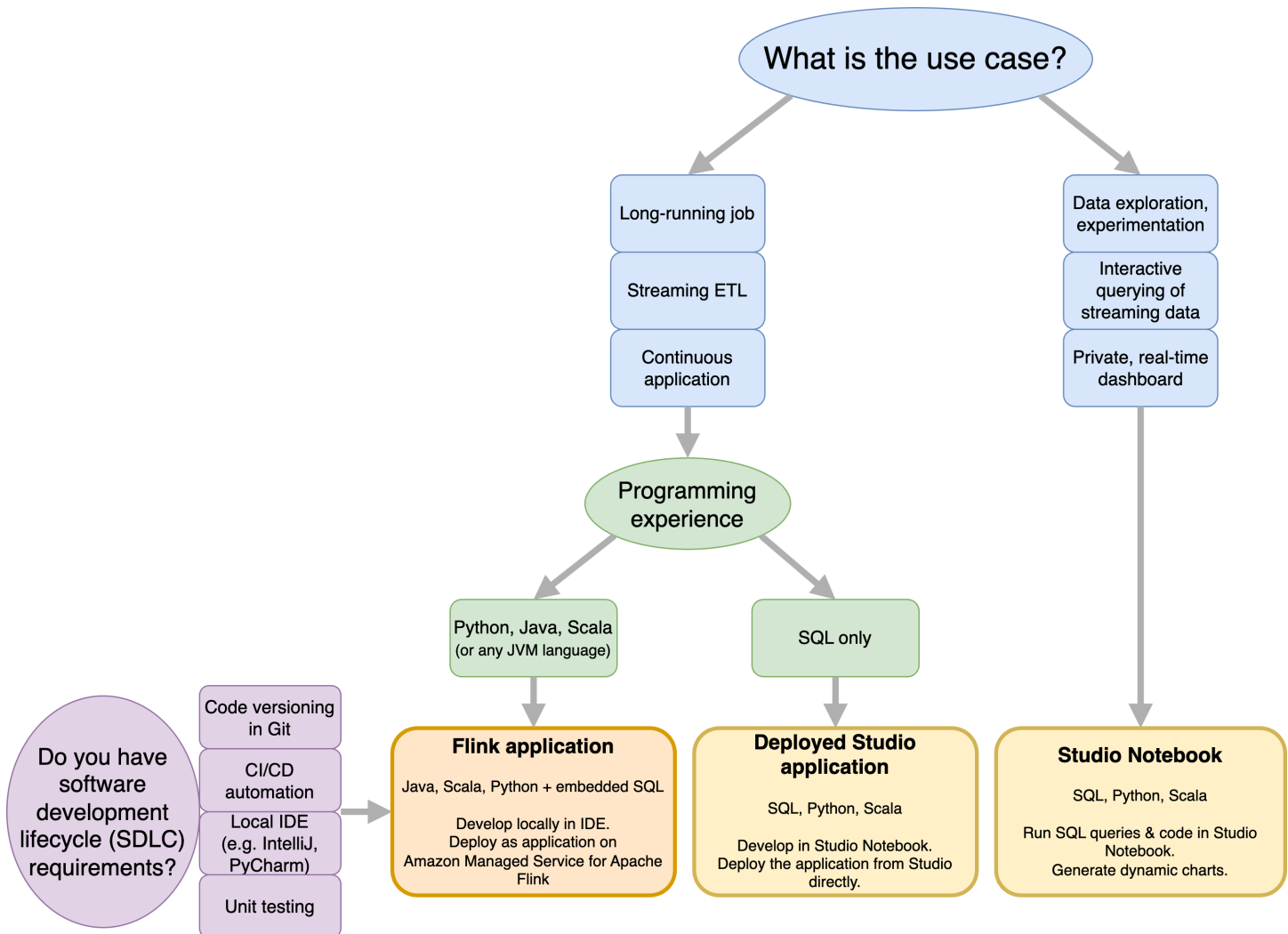
[Puoi creare applicazioni con il linguaggio che preferisci in Managed Service for Apache Flink utilizzando librerie open source basate su Apache Flink.](#) Apache Flink è un celebre framework e motore per l'elaborazione di flussi di dati.

Il servizio gestito per Apache Flink provvede alla creazione della struttura di base per le applicazioni Apache Flink. Gestisce funzionalità di base come il provisioning delle risorse di calcolo, la resilienza del failover AZ, il calcolo parallelo, la scalabilità automatica e i backup delle applicazioni (implementati come checkpoint e snapshot). È possibile utilizzare le funzionalità di programmazione di alto livello di Flink (che includono operatori, funzioni, origini e sink) nello stesso modo in cui le si utilizza quando si ospita l'infrastruttura Flink direttamente.

Decidi se utilizzare Managed Service per Apache Flink o Managed Service per Apache Flink Studio

Hai due opzioni per eseguire i tuoi job Flink con Amazon Managed Service for Apache Flink. Con [Managed Service for Apache Flink](#), puoi creare applicazioni Flink in Java, Scala o Python (e SQL integrato) utilizzando un IDE di tua scelta e Apache Flink Datastream o Table. APIs Con [Managed Service for Apache Flink Studio](#), puoi interrogare in modo interattivo i flussi di dati in tempo reale e creare ed eseguire facilmente applicazioni di elaborazione dei flussi utilizzando SQL, Python e Scala standard.

Puoi selezionare il metodo più adatto al tuo caso d'uso. Se non sei sicuro, questa sezione ti offrirà una guida di alto livello per aiutarti.



Prima di decidere se utilizzare Amazon Managed Service per Apache Flink o Amazon Managed Service per Apache Flink Studio, dovresti considerare il tuo caso d'uso.

[Se prevedi di utilizzare un'applicazione a lunga durata che eseguirà carichi di lavoro come Streaming ETL o Continuous Applications, dovresti prendere in considerazione l'utilizzo di Managed Service for Apache Flink.](#) Questo perché potete creare la vostra applicazione Flink utilizzando Flink APIs direttamente nell'IDE di vostra scelta. Lo sviluppo locale con il tuo IDE ti consente inoltre di sfruttare processi e strumenti comuni del ciclo di vita dello sviluppo del software (SDLC), come il controllo delle versioni del codice in Git, l'automazione CI/CD o il test delle unità.

Se sei interessato all'esplorazione dei dati ad hoc, desideri interrogare i dati di streaming in modo interattivo o creare dashboard private in tempo reale, [Managed Service for Apache Flink Studio](#) ti aiuterà a raggiungere questi obiettivi in pochi clic. Gli utenti che hanno familiarità con SQL possono prendere in considerazione la distribuzione di un'applicazione a lunga durata direttamente da Studio.

 Note

Puoi promuovere il tuo notebook Studio a un'applicazione a lunga durata. Tuttavia, se desideri integrarti con i tuoi strumenti SDLC come il controllo delle versioni del codice su Git e l'automazione CI/CD o tecniche come il test delle unità, ti consigliamo Managed Service for Apache Flink utilizzando l'IDE di tua scelta.

Scegliete quale Apache Flink utilizzare in Managed Service for Apache Flink APIs

Puoi creare applicazioni utilizzando Java, Python e Scala in Managed Service for Apache Flink usando Apache Flink APIs in un IDE di tua scelta. [Puoi trovare indicazioni su come creare applicazioni utilizzando l'API Flink Datastream and Table nella documentazione.](#) È possibile selezionare la lingua in cui creare l'applicazione Flink e quella da APIs utilizzare per soddisfare al meglio le esigenze dell'applicazione e delle operazioni. Se non siete sicuri, questa sezione fornisce una guida di alto livello per aiutarvi.

Scegli un'API Flink

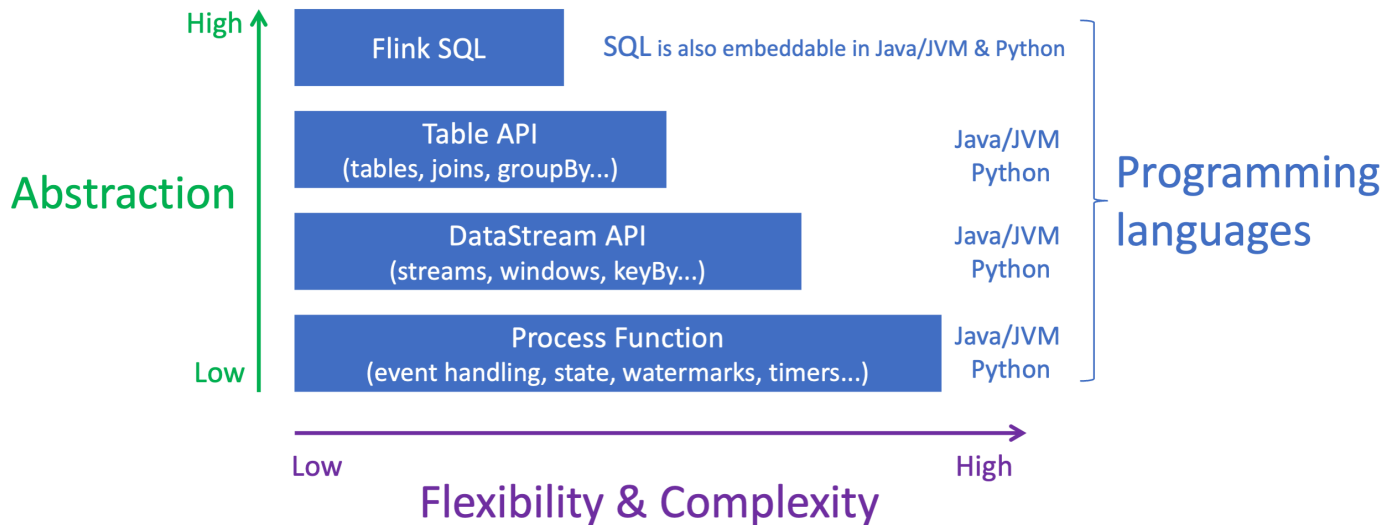
Gli Apache Flink APIs hanno diversi livelli di astrazione che possono influire sul modo in cui decidi di creare la tua applicazione. Sono espressivi e flessibili e possono essere usati insieme per creare un'applicazione. Non è necessario utilizzare una sola API Flink. Puoi saperne di più su Flink APIs nella documentazione di [Apache](#) Flink.

Flink offre quattro livelli di astrazione delle API: Flink SQL, Table API, DataStream API e Process Function, che viene utilizzata insieme all'API. DataStream Sono tutte supportate in Amazon Managed Service for Apache Flink. È consigliabile iniziare con un livello di astrazione più elevato ove possibile, tuttavia alcune funzionalità di Flink sono disponibili solo con l'[API Datastream](#) in cui è possibile creare l'applicazione in Java, Python o Scala. Dovresti prendere in considerazione l'utilizzo dell'API Datastream se:

- È necessario un controllo granulare sullo stato
- Desiderate sfruttare la possibilità di chiamare un database o un endpoint esterno in modo asincrono (ad esempio per l'inferenza)
- Desiderate utilizzare timer personalizzati (ad esempio per implementare finestre personalizzate o la gestione degli eventi tardivi)

- Vuoi essere in grado di modificare il flusso della tua applicazione senza reimpostare lo stato

Apache Flink APIs



Note

Sceita di una lingua con l'`DataStreamAPI`:

- SQL può essere incorporato in qualsiasi applicazione Flink, indipendentemente dal linguaggio di programmazione scelto.
- Se hai intenzione di utilizzare l' `DataStream API`, non tutti i connettori sono supportati in Python.
- Se hai bisogno di un valore basso, latency/high-throughput you should consider Java/Scala indipendentemente dall'API.
- Se prevedi di utilizzare Async IO nell'API Process Functions, dovrai usare Java.

La scelta dell'API può influire anche sulla capacità di far evolvere la logica dell'applicazione senza dover reimpostare lo stato. Ciò dipende da una funzionalità specifica, la possibilità di impostare l'UID sugli operatori, disponibile solo nell'`DataStreamAPI` per Java e Python. Per ulteriori informazioni, consulta [Set UUIDs For All Operators nella documentazione](#) di Apache Flink.

Inizia a usare le applicazioni di streaming di dati

Innanzitutto, è consigliabile creare un'applicazione del servizio gestito per Apache Flink che legga ed elabori in maniera continua i dati in streaming. In seguito, consigliamo di scrivere il codice utilizzando l'ambiente di sviluppo integrato (IDE) prescelto ed effettuare test con i dati di streaming live. È inoltre possibile configurare le destinazioni a cui si desidera che il servizio gestito per Apache Flink invii i risultati.

Per iniziare, consigliamo di consultare le sezioni seguenti:

- [Servizio gestito per Apache Flink: come funziona](#)
- [Inizia a usare Amazon Managed Service per Apache Flink \(DataStream API\)](#)

In alternativa, puoi iniziare creando un notebook Managed Service for Apache Flink Studio che ti permetta di interrogare interattivamente i flussi di dati in tempo reale e di creare ed eseguire facilmente applicazioni di elaborazione dei flussi utilizzando SQL, Python e Scala standard. Con pochi clic AWS Management Console, puoi avviare un notebook serverless per interrogare i flussi di dati e ottenere risultati in pochi secondi. Per iniziare, consigliamo di consultare le sezioni seguenti:

- [Usa un notebook Studio con Managed Service per Apache Flink](#)
- [Crea un taccuino da Studio](#)

Servizio gestito per Apache Flink: come funziona

Managed Service for Apache Flink è un servizio Amazon completamente gestito che consente di utilizzare un'applicazione Apache Flink per elaborare dati in streaming. Innanzitutto, si programma l'applicazione Apache Flink, quindi si crea l'applicazione Managed Service for Apache Flink.

Programma la tua applicazione Apache Flink

Un'applicazione Apache Flink è un'applicazione Java o Scala creata con il framework Apache Flink. Puoi creare la tua applicazione Apache Flink in locale.

Le applicazioni utilizzano principalmente l'[DataStream API](#) o l'[API Table](#). APIs Sono disponibili anche gli altri Apache Flink, ma sono usati meno comunemente nella creazione di applicazioni di streaming.

Le caratteristiche dei due APIs sono le seguenti:

DataStream API

Il modello di programmazione dell' DataStream API Apache Flink si basa su due componenti:

- Flusso di dati: la rappresentazione strutturata di un flusso continuo di record di dati.
- Operatore di trasformazione: accetta uno o più flussi di dati come input e produce uno o più flussi di dati come output.

Le applicazioni create con l' DataStream API eseguono le seguenti operazioni:

- Lettura dei dati da un'origine dati (ad esempio un flusso Kinesis o un argomento Amazon MSK).
- Trasformazione di dati, ad esempio filtraggio, aggregazione o arricchimento.
- Scrittura dei dati trasformati in un sink di dati.

Le applicazioni che utilizzano l' DataStream API possono essere scritte in Java o Scala e possono essere lette da un flusso di dati Kinesis, un argomento di Amazon MSK o un'origine personalizzata.

L'applicazione elabora i dati utilizzando un connettore. Apache Flink utilizza i seguenti tipi di connettori:

- Origine: connettore utilizzato per leggere dati esterni.
- Sink: connettore utilizzato per scrivere in posizioni esterne.
- Operatore: connettore utilizzato per elaborare i dati all'interno dell'applicazione.

Un'applicazione tipica è costituita da almeno un flusso di dati con un'origine, un flusso di dati con uno o più operatori e almeno un sink di dati.

Per ulteriori informazioni sull'utilizzo dell' DataStream API, consulta. [Esamina i componenti DataStream dell'API](#)

API Table

Il modello di programmazione dell'API Table di Apache Flink si basa sui componenti seguenti:

- Ambiente tabellare: interfaccia per i dati sottostanti utilizzata per creare e ospitare una o più tabelle.
- Tabella: un oggetto che fornisce l'accesso a una tabella o una vista SQL.
- Origine della tabella: serve per leggere dati da un'origine esterna, ad esempio un argomento di Amazon MSK.
- Funzione della tabella: una query SQL o una chiamata API utilizzata per trasformare dati.
- Sink della tabella: serve per scrivere dati in un percorso esterno, ad esempio un bucket Amazon S3.

Le applicazioni create con l'API Table eseguono le seguenti operazioni:

- Creazione di un `TableEnvironment` collegandosi a un `Table Source`.
- Crea una tabella nel `TableEnvironment` utilizzando query SQL o funzioni API Table.
- Esecuzione di una query sulla tabella utilizzando API Table o SQL.
- Trasformazione dei risultati della query utilizzando funzioni Table o query SQL.
- Scrittura dei risultati della query o della funzione su un `Table Sink`.

Le applicazioni che utilizzano l'API Table possono essere scritte in Java o Scala e possono eseguire query sui dati utilizzando chiamate API o query SQL.

Per ulteriori informazioni sull'utilizzo dell'API Table, consulta [Componenti dell'API Review Table](#).

Crea il tuo servizio gestito per l'applicazione Apache Flink

Managed Service for Apache Flink è un AWS servizio che crea un ambiente per l'hosting dell'applicazione Apache Flink e fornisce le seguenti impostazioni:

- [Usa le proprietà di runtime](#): parametri che puoi fornire alla tua applicazione. È possibile modificare questi parametri senza ricompilare il codice dell'applicazione.
- [Implementa la tolleranza agli](#): in che modo l'applicazione viene ripristinata dopo interruzioni e riavvii.
- [Registrazione e monitoraggio in Amazon Managed Service per Apache Flink](#): In che modo l'applicazione registra gli eventi in Logs. CloudWatch
- [Implementa la scalabilità delle applicazioni](#): in che modo l'applicazione fornisce le risorse di elaborazione.

È possibile creare ed eseguire un'applicazione del servizio gestito per Apache Flink utilizzando la console o la AWS CLI. Per iniziare a creare un'applicazione del servizio gestito da Amazon per Apache Flink, consulta [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#).

Creare un servizio gestito per l'applicazione Apache Flink

Questo argomento contiene informazioni sulla creazione di un servizio gestito per l'applicazione Apache Flink.

Questo argomento contiene le sezioni seguenti:

- [Crea il codice dell'applicazione Managed Service for Apache Flink](#)
- [Crea la tua applicazione Managed Service for Apache Flink](#)
- [Avvia l'applicazione Managed Service for Apache Flink](#)
- [Verifica l'applicazione Managed Service for Apache Flink](#)
- [Abilita i rollback di sistema per la tua applicazione Managed Service for Apache Flink](#)

Crea il codice dell'applicazione Managed Service for Apache Flink

Questa sezione descrive i componenti utilizzati per creare il codice dell'applicazione Managed Service for Apache Flink.

Per il codice dell'applicazione, ti consigliamo di utilizzare la versione di Apache Flink supportata più recente. Per informazioni sull'aggiornamento delle applicazioni del servizio gestito per Apache Flink, consulta [Usa gli aggiornamenti di versione sul posto per Apache Flink](#).

Il codice dell'applicazione viene creato utilizzando [Apache Maven](#). Un progetto Apache Maven utilizza un file `pom.xml` per specificare le versioni dei componenti che utilizza.

Note

Il servizio gestito per Apache Flink supporta file JAR di dimensioni fino a 512 MB. Se si utilizza un file JAR di dimensioni superiori a queste, l'applicazione non viene avviata.

Le applicazioni possono ora utilizzare l'API Java da qualsiasi versione di Scala. È necessario includere la libreria standard Scala di propria scelta nelle applicazioni Scala.

Per informazioni sulla creazione di un'applicazione del servizio gestito per Apache Flink che utilizza Apache Beam, consulta [Utilizzo di Apache Beam con il servizio gestito per Apache Flink](#).

Specificate la versione Apache Flink dell'applicazione

Quando utilizzi il servizio gestito per il runtime di Apache Flink versione 1.1.0, devi specificare la versione di Apache Flink che l'applicazione utilizza durante la compilazione. Fornisci la versione di Apache Flink con il parametro `-Dflink.version`. Ad esempio, se utilizzi Apache Flink 1.19.1, fornisci quanto segue:

```
mvn package -Dflink.version=1.19.1
```

Per creare applicazioni con versioni precedenti di Apache Flink, vedi [Versioni precedenti](#)

Crea la tua applicazione Managed Service for Apache Flink

Dopo aver creato il codice dell'applicazione, procedi come segue per creare l'applicazione Managed Service for Apache Flink:

- Carica il codice dell'applicazione: carica il codice dell'applicazione in un bucket Amazon S3. Quando crei l'applicazione, specifica il nome del bucket S3 e il nome oggetto del codice dell'applicazione. Per un tutorial che mostra come caricare il codice dell'applicazione, consulta il [Tutorial: inizia a usare l' `DataStream` API in Managed Service for Apache Flink](#) tutorial.

- Creazione dell'applicazione del servizio gestito per Apache Flink: utilizza uno dei seguenti metodi per creare l'applicazione del servizio gestito per Apache Flink:
 - Crea la tua applicazione Managed Service for Apache Flink utilizzando la AWS console: puoi creare e configurare la tua applicazione utilizzando la AWS console.

Quando crei l'applicazione utilizzando la console, le risorse dipendenti dall'applicazione (come CloudWatch i flussi di log, i ruoli IAM e le politiche IAM) vengono create automaticamente.

Quando crei l'applicazione utilizzando la console, devi specificare la versione di Apache Flink utilizzata dall'applicazione selezionandola dal menu a discesa nella pagina Servizio gestito per Apache Flink: crea applicazione.

Per un tutorial su come utilizzare la console per creare un'applicazione, consulta il [Tutorial: inizia a usare l' `DataStream API` in Managed Service for Apache Flink](#) tutorial.

- Crea la tua applicazione Managed Service for Apache Flink utilizzando la AWS CLI: puoi creare e configurare la tua applicazione utilizzando la CLI. AWS

Quando crei l'applicazione utilizzando la CLI, devi anche creare manualmente le risorse dipendenti dall'applicazione (come flussi di CloudWatch log, ruoli IAM e politiche IAM).

Quando crei un'applicazione utilizzando la CLI, devi specificare la versione di Apache Flink utilizzata dall'applicazione utilizzando il parametro `RuntimeEnvironment` dell'operazione `CreateApplication`.

Note

È possibile modificare il nome `RuntimeEnvironment` di un'applicazione esistente. Per scoprire come, consulta [Usa gli aggiornamenti di versione sul posto per Apache Flink](#).

Avvia l'applicazione Managed Service for Apache Flink

Dopo aver creato il codice dell'applicazione, averlo caricato su S3 e aver creato l'applicazione del servizio gestito per Apache Flink, è il momento di avviare l'applicazione. L'avvio di un'applicazione del servizio gestito per Apache Flink richiede in genere alcuni minuti.

Per avviare l'applicazione, utilizza uno dei seguenti metodi:

- Avvia l'applicazione Managed Service for Apache Flink utilizzando la AWS console: puoi eseguire l'applicazione scegliendo Esegui nella pagina dell'applicazione nella console. AWS
- Avvia l'applicazione Managed Service for Apache Flink utilizzando l' AWS API: puoi eseguire l'applicazione utilizzando l'azione. [StartApplication](#)

Verifica l'applicazione Managed Service for Apache Flink

È possibile verificare il funzionamento dell'applicazione nei seguenti modi:

- Utilizzo CloudWatch dei registri: è possibile utilizzare CloudWatch Logs and CloudWatch Logs Insights per verificare che l'applicazione funzioni correttamente. Per informazioni sull'utilizzo di CloudWatch Logs con l'applicazione Managed Service for Apache Flink, consulta. [Registrazione e monitoraggio in Amazon Managed Service per Apache Flink](#)
- Utilizzo CloudWatch delle metriche: puoi utilizzare CloudWatch Metrics per monitorare l'attività dell'applicazione o l'attività delle risorse utilizzate dall'applicazione per l'input o l'output (come i flussi Kinesis, i flussi Firehose o i bucket Amazon S3). Per ulteriori informazioni sui CloudWatch parametri, consulta [Working with Metrics](#) nella Amazon CloudWatch User Guide.
- Monitoraggio delle posizioni di output: se l'applicazione scrive l'output in una posizione (come un bucket o un database Amazon S3), è possibile monitorare quella posizione per i dati scritti.

Abilita i rollback di sistema per la tua applicazione Managed Service for Apache Flink

Con la funzionalità di rollback del sistema, puoi ottenere una maggiore disponibilità dell'applicazione Apache Flink in esecuzione su Amazon Managed Service per Apache Flink. L'attivazione di questa configurazione consente al servizio di ripristinare automaticamente l'applicazione alla versione precedentemente in esecuzione quando un'azione come o si verifica un bug nel codice o nella configurazione. `UpdateApplication autoscaling`

Note

Per utilizzare la funzionalità di rollback del sistema, è necessario effettuare l'attivazione aggiornando l'applicazione. Per impostazione predefinita, le applicazioni esistenti non utilizzeranno automaticamente il rollback del sistema.

Come funziona

Quando avvii un'operazione applicativa, ad esempio un'azione di aggiornamento o di scalabilità, Amazon Managed Service for Apache Flink tenta innanzitutto di eseguire tale operazione. Se rileva problemi che impediscono il successo dell'operazione, come bug nel codice o autorizzazioni insufficienti, il servizio avvia automaticamente un'operazione. `RollbackApplication`

Il rollback tenta di ripristinare l'applicazione alla versione precedente eseguita correttamente, insieme allo stato dell'applicazione associata. Se il rollback ha esito positivo, l'applicazione continua a elaborare i dati con tempi di inattività minimi utilizzando la versione precedente. Se anche il rollback automatico fallisce, Amazon Managed Service for Apache Flink trasferisce l'applicazione allo `READY` stato, in modo che tu possa intraprendere ulteriori azioni, tra cui correggere l'errore e riprovare l'operazione.

È necessario attivare l'utilizzo dei rollback automatici del sistema. Da questo momento in poi, puoi abilitarlo utilizzando la console o l'API per tutte le operazioni sulla tua applicazione.

Il seguente esempio di richiesta di `UpdateApplication` azione abilita i rollback del sistema per un'applicazione:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationSystemRollbackConfigurationUpdate": {
      "RollbackEnabledUpdate": "true"
    }
  }
}
```

Esamina gli scenari comuni per il rollback automatico del sistema

I seguenti scenari illustrano i vantaggi dei rollback automatici del sistema:

- Aggiornamenti dell'applicazione: se aggiorni l'applicazione con un nuovo codice che presenta bug durante l'inizializzazione del job Flink tramite il metodo principale, il rollback automatico consente di ripristinare la versione funzionante precedente. Altri scenari di aggiornamento in cui i rollback del sistema sono utili includono:
 - [Se l'applicazione viene aggiornata per essere eseguita con un parallelismo superiore a `MaxParallelism`.](#)

- Se l'applicazione viene aggiornata per essere eseguita con sottoreti errate per un'applicazione VPC, si verifica un errore durante l'avvio del processo Flink.
- Aggiornamenti della versione di Flink: quando si esegue l'aggiornamento a una nuova versione di Apache Flink e l'applicazione aggiornata presenta un problema di compatibilità con le snapshot, il rollback del sistema consente di tornare automaticamente alla versione precedente di Flink.
- AutoScaling: Quando l'applicazione è scalabile ma riscontra problemi di ripristino da un punto di salvataggio, a causa della mancata corrispondenza dell'operatore tra l'istantanea e il grafico del lavoro Flink.

Usa APIs l'operazione per i rollback del sistema

Per offrire una migliore visibilità, Amazon Managed Service per Apache Flink ne offre due APIs relative alle operazioni delle applicazioni che possono aiutarti a tenere traccia degli errori e dei relativi rollback di sistema.

ListApplicationOperations

Questa API elenca tutte le operazioni eseguite sull'applicazione, incluse, UpdateApplication MaintenanceRollbackApplication, e altre in ordine cronologico inverso. L'esempio seguente di richiesta di ListApplicationOperations azione elenca le prime 10 operazioni dell'applicazione per l'applicazione:

```
{
  "ApplicationName": "MyApplication",
  "Limit": 10
}
```

Il seguente esempio di richiesta ListApplicationOperations aiuta a filtrare l'elenco in base agli aggiornamenti precedenti dell'applicazione:

```
{
  "ApplicationName": "MyApplication",
  "operation": "UpdateApplication"
}
```

DescribeApplicationOperation

Questa API fornisce informazioni dettagliate su un'operazione specifica elencata da ListApplicationOperations, incluso il motivo dell'errore, se applicabile. L'esempio seguente

di richiesta di `DescribeApplicationOperation` azione elenca i dettagli per un'operazione specifica dell'applicazione:

```
{
  "ApplicationName": "MyApplication",
  "OperationId": "xyzoperation"
}
```

Per informazioni sulla risoluzione dei problemi, consulta [Le migliori pratiche per il rollback del sistema](#).

Esegui un servizio gestito per l'applicazione Apache Flink

Questo argomento contiene informazioni sull'esecuzione di un servizio gestito per Apache Flink.

Quando si esegue l'applicazione di servizio gestito per Apache Flink, il servizio crea un processo Apache Flink. Un processo Apache Flink è il ciclo di vita dell'esecuzione dell'applicazione servizio gestito per Apache Flink. L'esecuzione del processo e le risorse che utilizza sono gestite dal Job Manager. Il Job Manager divide in attività l'esecuzione dell'applicazione. Ogni attività è gestita da un Task Manager. Quando si monitorano le prestazioni dell'applicazione, è possibile esaminare le prestazioni di ogni Task Manager o del Job Manager nel suo complesso.

Per informazioni sui job Apache Flink, consulta Jobs [and Scheduling nella documentazione](#) di Apache Flink.

Identifica lo stato della candidatura e del lavoro

Viene visualizzato lo status di esecuzione attuale dell'applicazione e del relativo processo:

- Status dell'applicazione: lo status attuale che ne descrive la fase di esecuzione. Gli status dell'applicazione includono le seguenti opzioni:
 - Stati costanti dell'applicazione: l'applicazione in genere rimane in uno di questi stati finché non viene apportata una modifica dello stato:
 - PRONTA: un'applicazione nuova o interrotta continua a comparire come PRONTA finché non viene eseguita.
 - IN ESECUZIONE: un'applicazione che è stata avviata correttamente si trova nello stato IN ESECUZIONE.

- Stati transitori dell'applicazione: un'applicazione che si trova in uno di questi stati è in genere in fase di transizione a un altro stato. Se una candidatura rimane in uno stato transitorio per un certo periodo di tempo, è possibile interromperla utilizzando l'[StopApplication](#) azione con il Force parametro impostato su `true`. Questi status includono le seguenti opzioni:
 - **STARTING**: Si verifica dopo l'azione [StartApplication](#). L'applicazione passa dallo status **READY** a **RUNNING**.
 - **STOPPING**: Si verifica dopo l'[StopApplication](#) azione. L'applicazione passa dallo status **RUNNING** a **READY**.
 - **DELETING**: Si verifica dopo l'[DeleteApplication](#) azione. L'applicazione è in fase di eliminazione.
 - **UPDATING**: Si verifica dopo l'[UpdateApplication](#) azione. L'applicazione si sta aggiornando e tornerà allo status **RUNNING** o **READY**.
 - **AUTOSCALING**: L'applicazione ha la `AutoScalingEnabled` proprietà [ParallelismConfiguration](#) impostata su `true` e il servizio aumenta il parallelismo dell'applicazione. Quando l'applicazione è in questo stato, l'unica azione API valida che è possibile utilizzare è l'[StopApplication](#) azione con il Force parametro impostato su `true`. Per ulteriori informazioni sul dimensionamento automatico, consulta [Utilizza il ridimensionamento automatico in Managed Service for Apache Flink](#).
 - **FORCE_STOPPING**: Si verifica dopo che l'[StopApplication](#) azione è stata richiamata con il Force parametro impostato su `true`. L'applicazione è in fase di arresto forzato. L'applicazione passa dallo stato **STARTING**, **UPDATING**, **STOPPING** o **AUTOSCALING** allo stato **READY**.
 - **ROLLING_BACK**: Si verifica dopo la chiamata [RollbackApplication](#) dell'azione. È in corso il ripristino dell'applicazione a una versione precedente. L'applicazione passa dallo status **UPDATING** o **AUTOSCALING** a **RUNNING**.
 - **MAINTENANCE**: si verifica quando il servizio gestito per Apache Flink applica le patch all'applicazione. Per ulteriori informazioni, consulta [Gestisci le attività di manutenzione per Managed Service for Apache Flink](#).

Puoi controllare lo stato dell'applicazione utilizzando la console o utilizzando l'[DescribeApplication](#) azione.

- Status del processo: quando l'applicazione passa allo status **RUNNING**, viene visualizzato uno status del processo che ne descrive l'attuale fase di esecuzione. Un processo parte dallo status **CREATED**, quindi passa a **RUNNING** una volta iniziato. Se si verificano condizioni di errore, l'applicazione passa allo status seguente:

- Per le applicazioni che utilizzano Apache Flink 1.11 e versioni successive, l'applicazione passa allo stato **RESTARTING**.
- Per le applicazioni che utilizzano Apache Flink 1.8 e versioni precedenti, l'applicazione passa allo stato **FAILING**.

L'applicazione passa quindi allo stato **RESTARTING** o **FAILED**, a seconda che il lavoro possa essere riavviato o meno.

Puoi controllare lo stato del lavoro esaminando il CloudWatch registro della tua candidatura per eventuali modifiche allo stato.

Esegui carichi di lavoro in batch

Il servizio gestito per Apache Flink supporta l'esecuzione di carichi di lavoro in batch di Apache Flink. In un processo batch, quando un processo Apache Flink passa allo stato **FINITO**, lo stato dell'applicazione del servizio gestito per Apache Flink è impostato su **PRONTO**. Per ulteriori informazioni sugli stati dei processi Flink, consulta [Processi e pianificazione](#).

Consulta le risorse dell'applicazione Managed Service for Apache Flink

Questa sezione descrive le risorse di sistema utilizzate dall'applicazione. Comprendere come il servizio gestito per Apache Flink alloca e utilizza le risorse è fondamentale per progettare, creare e mantenere un'applicazione di servizio gestito per Apache Flink efficiente e stabile.

Managed Service per le risorse delle applicazioni Apache Flink

Managed Service for Apache Flink è un AWS servizio che crea un ambiente per ospitare l'applicazione Apache Flink. Il servizio Managed Service for Apache Flink fornisce risorse utilizzando unità denominate Kinesis Processing Units (KPU). KPU

Una KPU rappresenta le seguenti risorse di sistema:

- un core CPU;
- 4 GB di memoria, di cui un GB di memoria nativa e tre GB di memoria heap;
- 50 GB di spazio su disco.

KPUs esegue le applicazioni in unità di esecuzione distinte denominate attività e sottoattività. Un'attività secondaria rappresenta l'equivalente di un thread.

Il numero di KPU elementi disponibili per un'applicazione è uguale all'`Parallelism` impostazione dell'applicazione, diviso per l'`ParallelismPerKPU` impostazione dell'applicazione.

Per ulteriori informazioni sul parallelismo delle applicazioni, consulta [Implementa la scalabilità delle applicazioni](#).

Risorse dell'applicazione Apache Flink

L'ambiente Apache Flink alloca le risorse per l'applicazione utilizzando unità chiamate slot di attività. Quando il servizio gestito per Apache Flink alloca risorse per l'applicazione, assegna uno o più slot di attività di Apache Flink a una singola KPU. Il numero di slot assegnati a una singola KPU è uguale all'impostazione `ParallelismPerKPU` dell'applicazione. Per ulteriori informazioni sugli slot di attività, vedere [Job Scheduling nella documentazione di](#) Apache Flink.

Parallelismo degli operatori

È possibile impostare il numero massimo di attività secondarie che un operatore può utilizzare. Questo valore è denominato parallelismo degli operatori. Per impostazione predefinita, il parallelismo di ogni operatore dell'applicazione è uguale al parallelismo dell'applicazione. Ciò significa che, per impostazione predefinita, ove necessario ogni operatore dell'applicazione può utilizzare tutte le attività secondarie in essa disponibili.

È possibile impostare il parallelismo degli operatori nell'applicazione utilizzando il metodo `setParallelism`. Questo metodo consente di controllare il numero di attività secondarie che ogni operatore può utilizzare contemporaneamente.

Per ulteriori informazioni sugli operatori, consulta Operatori nella [documentazione](#) di Apache Flink.

Concatenamento degli operatori

Solitamente ogni operatore utilizza un'attività secondaria separata; tuttavia se più operatori agiscono sempre in sequenza il runtime può assegnarli tutti alla stessa attività. Questo processo si chiama concatenazione degli operatori.

Se operano tutti sugli stessi dati, è possibile concatenare diversi operatori in sequenza in un'unica attività. Di seguito sono riportati alcuni criteri necessari affinché ciò accada:

- gli operatori eseguono un trasferimento semplice 1 a 1;
- tutti gli operatori dispongono dello stesso parallelismo degli operatori.

Quando l'applicazione concatena gli operatori in un'unica attività secondaria conserva le risorse di sistema, poiché il servizio non ha bisogno di eseguire operazioni di rete e allocare attività secondarie a ciascun operatore. Per determinare se l'applicazione utilizza la concatenazione degli operatori è sufficiente consultare il grafico dei processi nella console del servizio gestito per Apache Flink. Ogni vertice dell'applicazione rappresenta uno o più operatori. Il grafico mostra gli operatori che sono stati concatenati in un unico vertice.

Fatturazione al secondo in Managed Service for Apache Flink

Il servizio gestito per Apache Flink viene ora fatturato in incrementi di un secondo. È previsto un costo minimo di dieci minuti per applicazione. La fatturazione al secondo è applicabile alle applicazioni appena lanciate o già in esecuzione. Questa sezione descrive come Managed Service for Apache Flink contabilizza e fattura il tuo utilizzo. Per ulteriori informazioni sui prezzi di Managed Service for Apache Flink, consulta i prezzi di [Amazon Managed Service for Apache Flink](#).

Come funziona

Il servizio gestito per Apache Flink addebita all'utente la durata e il numero di unità di elaborazione Kinesis (KPU) fatturate in incrementi di un secondo nelle tariffe supportate. Regioni AWS Una singola KPU comprende 1 vCPU di elaborazione e 4 GB di memoria. Ti viene addebitata una tariffa oraria in base al numero di applicazioni KPU utilizzate per eseguire le tue applicazioni.

Ad esempio, un'applicazione in esecuzione per 20 minuti e 10 secondi verrà addebitata per 20 minuti e 10 secondi, moltiplicata per le risorse utilizzate. A un'applicazione in esecuzione per 5 minuti verrà addebitato il costo minimo di dieci minuti, moltiplicato per le risorse utilizzate.

Il servizio gestito per Apache Flink indica l'utilizzo in ore. Ad esempio, 15 minuti corrispondono a 0,25 ore.

Per le applicazioni Apache Flink, viene addebitata una singola KPU aggiuntiva per applicazione, utilizzata per l'orchestrazione. Alle applicazioni vengono inoltre addebitati i costi per l'esecuzione di storage e backup durevoli. Lo storage delle applicazioni in esecuzione viene utilizzato per le funzionalità di elaborazione dello stato in Managed Service for Apache Flink e viene addebitato in base al costo. GB/month. Durable backups are optional and provide point-in-time recovery for applications, charged per GB/month

In modalità streaming, Managed Service for Apache Flink ridimensiona automaticamente il numero di dati KPU richiesti dall'applicazione di elaborazione dei flussi in base alle variazioni delle esigenze di memoria e di calcolo. Puoi scegliere di fornire alla tua applicazione il numero richiesto di KPU.

Regione AWS disponibilità

Note

Al momento, la fatturazione al secondo non è disponibile nelle seguenti regioni: AWS GovCloud (Stati Uniti orientali), (Stati Uniti occidentali), Cina AWS GovCloud (Pechino) e Cina (Ningxia).

La fatturazione al secondo è disponibile nelle seguenti aree: Regioni AWS

- Stati Uniti orientali (Virginia settentrionale), us-east-1
- Stati Uniti orientali (Ohio): us-east-2
- Stati Uniti occidentali (California settentrionale) - us-west-1
- Stati Uniti occidentali (Oregon): us-west-2
- Africa (Città del Capo) - af-south-1
- Asia Pacifico (Hong Kong) - ap-east-1
- Asia Pacifico (Hyderabad) - ap-south-1
- Asia Pacifico (Giacarta) - ap-southeast-3
- Asia Pacifico (Melbourne) - ap-southeast-4
- Asia Pacifico (Mumbai) - ap-south-1
- Asia Pacifico (Osaka) - ap-northeast-3
- Asia Pacifico (Seoul) - ap-northeast-2
- Asia Pacifico (Singapore) - ap-southeast-1
- Asia Pacifico (Sydney): ap-southeast-2
- Asia Pacifico (Tokyo), ap-northeast-1
- Canada (Centrale) - ca-central-1
- Canada occidentale (Calgary) - ca-west-1

- Europa (Francoforte) - eu-central-1
- Europa (Irlanda) - eu-west-1
- Europa (Londra) - eu-west-2
- Europa (Milano) (eu-south-1)
- Europa (Parigi) - eu-west-3
- Europa (Spagna) (eu-south-2)
- Europa (Stoccolma) - eu-north-1
- Europa (Zurigo) (eu-central-2)
- Israele (Tel Aviv) - il-central-1
- Medio Oriente (Bahrein) - me-south-1
- Regione Medio Oriente (EAU) (me-central-1)
- Sud America (San Paolo) - sa-east-1

Esempi di prezzi

Puoi trovare esempi di prezzi nella pagina dei prezzi di Managed Service for Apache Flink. Per ulteriori informazioni, consulta la pagina dei prezzi di [Amazon Managed Service for Apache Flink](#). Di seguito sono riportati altri esempi con illustrazioni del rapporto sui costi di utilizzo per ciascuno di essi.

Un carico di lavoro intenso e di lunga durata

Sei un grande servizio di streaming video e vorresti creare una raccomandazione video in tempo reale basata sulle interazioni degli utenti. Si utilizza un'applicazione Apache Flink in Managed Service for Apache Flink per importare continuamente gli eventi di interazione degli utenti da più flussi di dati Kinesis e per elaborare gli eventi in tempo reale prima di inviarli a un sistema a valle. Gli eventi di interazione con l'utente vengono trasformati utilizzando diversi operatori. Ciò include il partizionamento dei dati per tipo di evento, l'arricchimento dei dati con metadati aggiuntivi, l'ordinamento dei dati per timestamp e il buffering dei dati per 5 minuti prima della consegna. L'applicazione prevede molte fasi di trasformazione che richiedono molta elaborazione e sono parallelizzabili. L'applicazione Flink è configurata per funzionare con 20 KPU per adattarsi al carico di lavoro. L'applicazione utilizza 1 GB di backup durevole delle applicazioni ogni giorno. Le tariffe mensili del servizio gestito per Apache Flink verranno calcolate come segue:

Costi mensili

Il prezzo nella regione Stati Uniti orientali (Virginia settentrionale) è di 0,11 USD per KPU/ora.

Managed Service for Apache Flink alloca 50 GB di storage delle applicazioni in esecuzione per KPU e addebita 0,10 USD per GB/mese.

- Costi mensili KPU: 24 ore* 30 giorni* (KPU 20+ 1 KPU aggiuntiva per l'applicazione di streaming) * 0,11 USD/ora = 1.584,00 USD
- Costi mensili di archiviazione delle applicazioni in esecuzione: 30 giorni* 20* 50 USD al mese = \$100,00 KPU GB/KPU * \$0.10/GB
- Costi mensili durevoli per lo storage delle applicazioni: 30 giorni x 1 GB * 0,023 GB/mese = 0,03 USD
- Spese totali: 1.584,00 USD + 100 USD + 0,03 USD = 1.684,03 USD

Report sull'utilizzo dei costi per Managed Service for Apache Flink sulla console Billing and Cost Management per il mese

Analisi Kinesis

- USD 1.684,03 - Stati Uniti orientali (Virginia settentrionale)
- Amazon Kinesis Analytics CreateSnapshot
 - 0,023 USD per GB al mese di backup durevoli delle applicazioni
 - 1 GB al mese: 0,03 USD
- Amazon Kinesis Analytics StartApplication
 - 0,10 USD per GB al mese di storage delle applicazioni in esecuzione
 - 1.000 GB al mese: 100 USD
 - 0,11 USD per unità di elaborazione Kinesis all'ora per le applicazioni Apache Flink
 - 15.120 KPU/ora: 1.584 USD

Un carico di lavoro in batch che dura circa 15 minuti ogni giorno

Utilizzi un'applicazione Apache Flink in Managed Service for Apache Flink per trasformare i dati di log in Amazon Simple Storage Service (Amazon S3) in modalità batch. I dati di registro vengono trasformati utilizzando diversi operatori. Ciò include l'applicazione di uno schema ai diversi eventi di registro, il partizionamento dei dati per tipo di evento e l'ordinamento dei dati per timestamp.

L'applicazione prevede molte fasi di trasformazione, ma nessuna richiede un uso intensivo dal punto di vista computazionale. Questa applicazione acquisisce dati a 2.000 record al secondo per 15 minuti

ogni giorno in un mese di 30 giorni. Non vengono creati backup durevoli delle applicazioni. Le tariffe mensili del servizio gestito per Apache Flink verranno calcolate come segue:

Costi mensili

Il prezzo nella regione Stati Uniti orientali (Virginia settentrionale) è di 0,11 USD per KPU/ora.

Managed Service for Apache Flink alloca 50 GB di storage delle applicazioni in esecuzione per KPU e addebita 0,10 USD per GB/mese.

- Carico di lavoro in batch: durante i 15 minuti al giorno, l'applicazione Managed Service for Apache Flink ne elabora 2.000 records/second, which takes 2KPU. $30 \text{ days/month} * 15 \text{ minutes/day} = 450 \text{ minutes/month}$
- Costi KPU mensili: $450 = 2,48 \text{ USD minutes/month} * (2\text{KPU} + 1 \text{ additional KPU for streaming application}) * \$0.11/\text{hour}$
- Costi mensili di archiviazione delle applicazioni in esecuzione: $450 \text{ minutes/month} * 2 \text{ KPU} * 50 \text{ GB/KPU} * \$0.10/\text{GB al mese} = 0,11 \text{ USD}$
- Costi totali: $2,48 \text{ USD} + 0,11 = 2,59 \text{ USD}$

Report sull'utilizzo dei costi per Managed Service for Apache Flink sulla console Billing and Cost Management per il mese

Analisi Kinesis

- 2,59 USD - Stati Uniti orientali (Virginia settentrionale)
- Amazon Kinesis Analytics StartApplication
 - 0,10 USD per GB al mese di backup di applicazioni in esecuzione
 - 1,042 GB al mese: 0,11 USD
 - 0,11 USD per unità di elaborazione Kinesis all'ora per le applicazioni Apache Flink
 - 22,5 KPU/ora: 2,48 USD

Un'applicazione di test che si interrompe e si avvia continuamente nella stessa ora, con costi minimi multipli

Sei una grande piattaforma di e-commerce che elabora milioni di transazioni ogni giorno. Vuoi sviluppare il rilevamento delle frodi in tempo reale. Utilizzi un'applicazione Apache Flink in Managed Service for Apache Flink per importare gli eventi di transazione da Kinesis Data Streams ed elaborare gli eventi in tempo reale con diverse fasi di trasformazione. Ciò include l'utilizzo di una finestra

scorrevole per aggregare gli eventi, il partizionamento degli eventi per tipi di eventi e l'applicazione di regole di rilevamento specifiche per diversi tipi di eventi. Durante lo sviluppo, si avvia e si arresta l'applicazione più volte per testare ed eseguire il debug del comportamento. In alcuni casi l'applicazione viene eseguita solo per pochi minuti. C'è un'ora in cui stai testando l'applicazione con 4 KPU e l'applicazione non utilizza alcun backup durevole dell'applicazione:

- Alle 10:05 si avvia l'applicazione, che viene eseguita per 30 minuti prima di essere interrotta alle 10:35.
- Alle 10:40, si riavvia l'applicazione, che viene eseguita per 5 minuti prima di essere interrotta alle 10:45.
- Alle 10:50, si riavvia l'applicazione, che viene eseguita per 2 minuti prima di essere interrotta alle 10:52.

Il servizio gestito per Apache Flink addebita un minimo di 10 minuti di utilizzo ogni volta che un'applicazione viene avviata. Il servizio gestito mensile per l'utilizzo di Apache Flink per l'applicazione verrà calcolato come segue:

- Prima volta che l'applicazione si avvia e si arresta: 30 minuti di utilizzo
- Seconda volta che l'applicazione si avvia e si arresta: 10 minuti di utilizzo (l'applicazione viene eseguita per 5 minuti arrotondati al costo minimo di 10 minuti)
- Terza volta che l'applicazione si avvia e si arresta: 10 minuti di utilizzo (l'applicazione viene eseguita per 2 minuti, arrotondati al costo minimo di 10 minuti)

In totale, all'applicazione verranno addebitati 50 minuti di utilizzo. Se non ci sono altri periodi del mese in cui l'applicazione è in esecuzione, gli addebiti mensili del servizio gestito per Apache Flink verranno calcolati come segue:

Costi mensili

Il prezzo nella regione Stati Uniti orientali (Virginia settentrionale) è di 0,11 USD per KPU/ora. Managed Service for Apache Flink alloca 50 GB di storage delle applicazioni in esecuzione per KPU e addebita 0,10 USD per GB/mese.

- Costi mensili KPU: 50 minuti* (KPU 4+1 KPU aggiuntiva per l'applicazione di streaming) * 0,11 USD/ora = 0,46 USD (arrotondato al centesimo più vicino)
- Costi mensili di archiviazione delle applicazioni in esecuzione: 50 minuti* KPU 4* 50 GB/KPU * \$0.10/GB mesi = 0,03 USD (arrotondati al centesimo più vicino)

- Spese totali: 0,46 USD + 0,03 = 0,49 USD

Report sull'utilizzo dei costi per Managed Service for Apache Flink sulla console Billing and Cost Management per il mese

Analisi Kinesis

- USD 0,49 - Stati Uniti orientali (Virginia settentrionale)
- Amazon Kinesis Analytics StartApplication
 - 0,10 USD per GB al mese di storage delle applicazioni in esecuzione
 - 0,232 GB al mese: 0,03 USD
 - 0,11 USD per unità di elaborazione Kinesis all'ora per le applicazioni Apache Flink
 - 4,167 KPU/ora - 0,46 USD

Esamina i componenti DataStream dell'API

La tua applicazione Apache Flink utilizza l' [DataStream API Apache Flink](#) per trasformare i dati in un flusso di dati.

Questa sezione descrive i diversi componenti che spostano, trasformano e tracciano i dati:

- [Usa i connettori per spostare i dati in Managed Service for Apache Flink con l'API DataStream](#) : questi componenti spostano i dati tra l'applicazione e le origini e le destinazioni dati esterne.
- [Trasforma i dati utilizzando gli operatori in Managed Service for Apache Flink con l'API DataStream](#) : questi componenti trasformano o raggruppano gli elementi di dati all'interno dell'applicazione.
- [Tieni traccia degli eventi in Managed Service for Apache Flink utilizzando l'API DataStream](#) : Questo argomento descrive come Managed Service for Apache Flink tiene traccia degli eventi quando si utilizza l' DataStream API.

Usa i connettori per spostare i dati in Managed Service for Apache Flink con l'API DataStream

Nell' DataStream API Amazon Managed Service for Apache Flink, i connettori sono componenti software che spostano i dati da e verso un'applicazione Managed Service for Apache Flink. I

connettori sono integrazioni flessibili che consentono di leggere file e directory. I connettori sono costituiti da moduli completi per l'interazione con i servizi Amazon e i sistemi di terze parti.

I tipi di connettori comprendono:

- [Aggiungi sorgenti di dati in streaming](#): invio di dati all'applicazione da un flusso di dati, un file o un'altra origine dati Kinesis.
- [Scrivi dati utilizzando i lavandini](#): invia dati dall'applicazione a un flusso di dati Kinesis, a un flusso Firehose o a un'altra destinazione di dati.
- [Usa I/O asincrono](#): fornisce l'accesso asincrono a un'origine dati (ad esempio, un database) per arricchire gli eventi di flusso.

Connettori disponibili

Il framework Apache Flink contiene connettori per l'accesso ai dati da vari tipi di origini. Per informazioni sui connettori disponibili nel framework Apache Flink, consulta [Connettori](#) nella [documentazione di Apache Flink](#).

Warning

Se hai applicazioni in esecuzione su Flink 1.6, 1.8, 1.11 o 1.13 e desideri eseguirle nelle regioni del Medio Oriente (Emirati Arabi Uniti), Asia Pacifico (Hyderabad), Israele (Tel Aviv), Europa (Zurigo), Medio Oriente (Emirati Arabi Uniti), Asia Pacifico (Melbourne) o Asia Pacifico (Giacarta), potresti dover ricostruire l'archivio delle applicazioni con un connettore aggiornato o eseguire l'aggiornamento a Flink 1.18.

I connettori Apache Flink sono archiviati nei propri archivi open source. Se stai eseguendo l'aggiornamento alla versione 1.18 o successiva, devi aggiornare le tue dipendenze. Per accedere al repository per i connettori Apache Flink, consulta [AWS flink-connector-aws](#)

Il precedente sorgente Kinesis non

`org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer` è più disponibile e potrebbe essere rimosso con le future release di Flink. Utilizzate [invece Kinesis Source](#).

Non esiste alcuna compatibilità a livello di stato tra `FlinkKinesisConsumer` e `KinesisStreamsSource`. Per i dettagli, consulta [Migrazione dei job esistenti al nuovo Kinesis Streams Source](#) nella documentazione di Apache Flink.

Di seguito sono riportate le linee guida consigliate:

Aggiornamenti dei connettori

Versione di Flink	Connettore usato	Risoluzione
1.19, 1.20	Sorgente Kinesis	Quando esegui l'aggiornamento a Managed Service for Apache Flink versione 1.19 e 1.20, assicurati di utilizzare il connettore e sorgente Kinesis Data Streams più recente. Deve essere una qualsiasi versione 5.0.0 o successiva. Per ulteriori informazioni, consulta Amazon Kinesis Data Streams Connector .
1.19, 1.20	Lavello Kinesis	Quando esegui l'aggiornamento a Managed Service for Apache Flink versione 1.19 e 1.20, assicurati di utilizzare il connettore sink Kinesis Data Streams più recente. Deve essere una qualsiasi versione 5.0.0 o successiva. Per ulteriori informazioni, vedere Kinesis Streams Sink .

Versione di Flink	Connettore usato	Risoluzione
1.19, 1.20	Sorgente DynamoDB Streams	Quando esegui l'aggiornamento a Managed Service for Apache Flink versione 1.19 e 1.20, assicurati di utilizzare il connettore e sorgente DynamoDB Streams più recente. Deve essere una qualsiasi versione 5.0.0 o successiva. Per ulteriori informazioni, consulta Amazon DynamoDB Connector .
1.19, 1.20	Lavandino DynamoDB	Quando esegui l'aggiornamento a Managed Service for Apache Flink versione 1.19 e 1.20, assicurati di utilizzare il connettore sink DynamoDB più recente. Deve essere una qualsiasi versione 5.0.0 o successiva. Per ulteriori informazioni, consulta Amazon DynamoDB Connector .

Versione di Flink	Connettore usato	Risoluzione
1.19, 1.20	Lavandino Amazon SQS	Quando esegui l'aggiornamento a Managed Service for Apache Flink versione 1.19 e 1.20, assicurati di utilizzare il connettore sink di Amazon SQS più recente. Deve essere una qualsiasi versione 5.0.0 o successiva. Per ulteriori informazioni, consulta Amazon SQS Sink .
1.19, 1.20	Servizio gestito Amazon per Prometheus Sink	Quando esegui l'aggiornamento a Managed Service for Apache Flink versione 1.19 e 1.20, assicurati di utilizzare il connettore sink di Amazon Managed Service for Prometheus più recente. Deve essere una qualsiasi versione 1.0.0 o successiva. Per ulteriori informazioni, vedere Prometheus Sink .

Aggiungi sorgenti di dati di streaming a Managed Service for Apache Flink

Apache Flink fornisce connettori per la lettura da file, socket, raccolte e origini personalizzate. Nel codice dell'applicazione, utilizzi un'[origine Apache Flink](#) per ricevere dati da un flusso. Questa sezione descrive le origini disponibili per i servizi Amazon.

Usa i flussi di dati Kinesis

`KinesisStreamsSource` Fornisce dati in streaming all'applicazione da un flusso di dati Amazon Kinesis.

Creazione di una **KinesisStreamsSource**

Il seguente esempio di codice illustra la creazione di una **KinesisStreamsSource**:

```
// Configure the KinesisStreamsSource
Configuration sourceConfig = new Configuration();
sourceConfig.set(KinesisSourceConfigOptions.STREAM_INITIAL_POSITION,
    KinesisSourceConfigOptions.InitialPosition.TRIM_HORIZON); // This is optional, by
    default connector will read from LATEST

// Create a new KinesisStreamsSource to read from specified Kinesis Stream.
KinesisStreamsSource<String> kdsSource =
    KinesisStreamsSource.<String>builder()
        .setStreamArn("arn:aws:kinesis:us-east-1:123456789012:stream/test-
stream")
        .setSourceConfig(sourceConfig)
        .setDeserializationSchema(new SimpleStringSchema())

        .setKinesisShardAssigner(ShardAssignerFactory.uniformShardAssigner()) // This is
        optional, by default uniformShardAssigner will be used.
        .build();
```

Per ulteriori informazioni sull'utilizzo di **aKinesisStreamsSource**, consulta [Amazon Kinesis Data Streams Connector](#) nella [documentazione di Apache Flink e KinesisConnectors](#) il nostro esempio pubblico su Github.

Crea un account che utilizza un consumatore EFO **KinesisStreamsSource**

KinesisStreamsSourceOra supporta [Enhanced Fan-Out \(EFO\)](#).

Se un consumatore Kinesis utilizza EFO, il servizio del flusso di dati Kinesis gli fornisce una larghezza di banda dedicata, anziché chiedere al consumatore di condividere la larghezza di banda fissa del flusso con gli altri consumatori che leggono dal flusso.

Per ulteriori informazioni sull'utilizzo di EFO con il consumatore Kinesis, [consulta FLIP-128: Enhanced Fan](#) Out for Kinesis Consumers. AWS

È possibile abilitare il consumatore EFO impostando i seguenti parametri sul consumatore Kinesis:

- **READER_TYPE**: imposta questo parametro su EFO per consentire all'applicazione di utilizzare un consumatore EFO per accedere ai dati di Kinesis Data Stream.

- **EFO_CONSUMER_NAME**: imposta questo parametro su un valore di stringa che sia unico tra i consumatori di questo flusso. Il riutilizzo di un nome consumatore nello stesso flusso di dati Kinesis causerà l'interruzione del precedente consumatore che utilizzava quel nome.

Per configurare un `KinesisStreamsSource` per l'utilizzo di EFO, aggiungi i seguenti parametri al consumatore:

```
sourceConfig.set(KinesisSourceConfigOptions.READER_TYPE,
    KinesisSourceConfigOptions.ReaderType.EFO);
sourceConfig.set(KinesisSourceConfigOptions.EFO_CONSUMER_NAME, "my-flink-efo-
consumer");
```

Per un esempio di un'applicazione Managed Service for Apache Flink che utilizza un consumatore EFO, vedi il nostro esempio [pubblico di Kinesis Connectors](#) su Github.

Usa Amazon MSK

L'origine `KafkaSource` fornisce dati di streaming all'applicazione da un argomento di Amazon MSK.

Creazione di una **KafkaSource**

Il seguente esempio di codice illustra la creazione di una `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");
```

Per ulteriori informazioni sull'utilizzo di una `KafkaSource`, consulta [Replica MSK](#).

Scrivi dati utilizzando i sinks in Managed Service for Apache Flink

Nel codice dell'applicazione, puoi utilizzare qualsiasi connettore [sink Apache Flink](#) per scrivere in sistemi esterni, inclusi AWS servizi come Kinesis Data Streams e DynamoDB.

Apache Flink fornisce anche sink per file e socket ed è possibile implementare sink personalizzati. Tra i diversi sink supportati, vengono utilizzati frequentemente i seguenti:

Usa i flussi di dati Kinesis

Apache Flink fornisce informazioni sul [connettore del flusso di dati Kinesis](#) nella documentazione di Apache Flink.

Per un esempio di applicazione che utilizza un flusso di dati Kinesis per l'input e l'output, consulta [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#).

Usa Apache Kafka e Amazon Managed Streaming per Apache Kafka (MSK)

Il [connettore Apache Flink Kafka](#) offre un supporto completo per la pubblicazione di dati su Apache Kafka e Amazon MSK, incluse le garanzie Exactly Once. [Per imparare a scrivere su Kafka, consulta gli esempi di connettori Kafka nella documentazione di Apache Flink](#).

Usa Amazon S3

Puoi utilizzare il `StreamingFileSink` di Apache Flink per scrivere oggetti in un bucket Amazon S3.

Per un esempio su come scrivere oggetti su S3, consulta [the section called "Sink S3"](#).

Usa Firehose

`FlinkKinesisFirehoseProducer` [È un sink Apache Flink affidabile e scalabile per l'archiviazione dell'output delle applicazioni utilizzando il servizio Firehose](#). In questa sezione viene descritto come impostare un progetto Maven per creare e utilizzare un `FlinkKinesisFirehoseProducer`.

Argomenti

- [Creazione di una `FlinkKinesisFirehoseProducer`](#)
- [Esempio di codice `FlinkKinesisFirehoseProducer`](#)

Creazione di una **`FlinkKinesisFirehoseProducer`**

Il seguente esempio di codice illustra la creazione di una `FlinkKinesisFirehoseProducer`:

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

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


Esempio di codice **FlinkKinesisFirehoseProducer**

Il seguente esempio di codice dimostra come creare e configurare un flusso di dati Apache Flink **FlinkKinesisFirehoseProducer** e inviarlo al servizio Firehose.

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

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

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

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

public class StreamingJob {

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

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

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

    private static DataStream<String>
    createSourceFromApplicationProperties(StreamExecutionEnvironment env)
```

```

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

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

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

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

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

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

public static void main(String[] args) throws Exception {

```

```
// set up the streaming execution environment
final StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();

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

DataStream<String> input = createSourceFromStaticConfig(env);

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

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

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

Per un tutorial completo su come utilizzare il lavello Firehose, vedere. [the section called “Firehose Sink”](#)

Utilizzare l'I/O asincrono nel servizio gestito per Apache Flink

Un operatore I/O asincrono arricchisce i dati del flusso utilizzando un'origine dati esterna, ad esempio un database. Il servizio gestito per Apache Flink arricchisce gli eventi di flusso in modo asincrono in modo che le richieste possano essere raggruppate in batch per una maggiore efficienza.

Per ulteriori informazioni, consulta I/O [asincrono](#) nella documentazione di Apache Flink.

Trasforma i dati utilizzando gli operatori in Managed Service for Apache Flink con l'API DataStream

Per trasformare i dati in entrata in un servizio gestito per Apache Flink viene utilizzato un operatore Apache Flink. Un operatore Apache Flink trasforma uno o più flussi di dati in un nuovo flusso di dati. Il nuovo flusso di dati contiene dati modificati dal flusso di dati originale. Apache Flink offre più di

25 operatori di elaborazione di flussi predefiniti. Per ulteriori informazioni, consulta [Operatori](#) nella documentazione di Apache Flink.

Questo argomento contiene le sezioni seguenti:

- [Usa gli operatori di trasformazione](#)
- [Usa gli operatori di aggregazione](#)

Usa gli operatori di trasformazione

Di seguito è riportato un esempio di semplice trasformazione del testo su uno dei campi di un flusso di dati JSON.

Questo codice crea un flusso di dati trasformato. Il nuovo flusso di dati contiene gli stessi dati del flusso originale, con la stringa " Company" aggiunta al contenuto del campo TICKER.

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

Usa gli operatori di aggregazione

Di seguito è riportato un esempio di operatore di aggregazione. Il codice crea un flusso di dati aggregato. L'operatore crea una finestra a cascata di 5 secondi e restituisce la somma dei valori PRICE per i record nella finestra con lo stesso valore TICKER.

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

Per altri esempi di codice, consulta [Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink](#).

Tieni traccia degli eventi in Managed Service for Apache Flink utilizzando l'API DataStream

Servizio gestito per Apache Flink tiene traccia degli eventi utilizzando i seguenti timestamp:

- **Tempo di elaborazione:** si riferisce all'ora di sistema della macchina che esegue la rispettiva operazione.
- **Ora evento:** si riferisce all'ora in cui ogni singolo evento si è verificato sul dispositivo di produzione.
- **Tempo di acquisizione:** si riferisce all'ora in cui gli eventi entrano nel servizio gestito per Apache Flink.

Si imposta il tempo utilizzato dall'ambiente di streaming utilizzando `setStreamTimeCharacteristic`.

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

Per ulteriori informazioni sui timestamp, consulta [Generazione di filigrane](#) nella documentazione di Apache Flink.

Componenti dell'API Review Table

La tua applicazione Apache Flink utilizza l'[API Apache Flink Table](#) per interagire con i dati in un flusso attraverso un modello relazionale. Utilizza l'API Table per accedere ai dati utilizzando origini Table, quindi utilizza le funzioni Table per trasformare e filtrare i dati della tabella. Puoi trasformare e filtrare i dati della tabella utilizzando funzioni API o comandi SQL.

Questa sezione contiene i seguenti argomenti:

- [Connettori API per tabelle](#): questi componenti spostano i dati tra l'applicazione e origini e destinazioni dati esterne.
- [Attributi temporali dell'API della tabella](#): questo argomento descrive il modo in cui il servizio gestito per Apache Flink tiene traccia degli eventi quando viene utilizzata l'API Table.

Connettori API per tabelle

Nel modello di programmazione Apache Flink, i connettori sono componenti utilizzati dall'applicazione per leggere o scrivere dati da fonti esterne, come altri AWS servizi.

Con l'API Apache Flink Table, puoi utilizzare i seguenti tipi di connettori:

- [Sorgenti API per tabelle](#): utilizzi i connettori di origine dell'API Table per creare tabelle all'interno dell'utente `TableEnvironment` utilizzando chiamate API o query SQL.
- [Table API sink](#): utilizzi i comandi SQL per scrivere dati di tabella su origini esterne come un argomento Amazon MSK o un bucket Amazon S3.

Sorgenti API per tabelle

Crei un'origine di tabella da un flusso di dati. Il codice seguente crea una tabella da un argomento di Amazon MSK:

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

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

Per ulteriori informazioni sulle sorgenti delle tabelle, consulta [Table & SQL Connectors](#) nella documentazione di Apache Flink.

Table API sink

Per scrivere i dati della tabella in un sink, crea il sink in SQL, quindi esegui il sink basato su SQL sull'oggetto `StreamTableEnvironment`.

Nell'esempio di codice seguente viene mostrato come scrivere dati di tabella su un sink Amazon S3:

```
final String s3Sink = "CREATE TABLE sink_table (" +
    "event_time TIMESTAMP," +
```

```
"ticker STRING," +
"price DOUBLE," +
"dt STRING," +
"hr STRING" +
")" +
" PARTITIONED BY (ticker,dt,hr)" +
" WITH" +
"(" +
" 'connector' = 'filesystem'," +
" 'path' = '" + s3Path + "'," +
" 'format' = 'json'" +
")";

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

Puoi utilizzare il parametro `format` per controllare il formato impiegato dal servizio gestito per Apache Flink per scrivere l'output nel sink. Per informazioni sui formati, consulta [Connettori supportati nella documentazione](#) di Apache Flink.

Sorgenti e sink definiti dall'utente

Puoi utilizzare i connettori Apache Kafka esistenti per inviare dati da e verso altri servizi AWS , come Amazon MSK e Amazon S3. Per interagire con altre origini e destinazioni dati, puoi definire fonti e sink personalizzati. Per ulteriori informazioni, consulta [Sources and Sinks definiti dall'utente](#) nella documentazione di Apache Flink.

Attributi temporali dell'API della tabella

Ogni record in un flusso di dati ha diversi timestamp che definiscono quando si sono verificati gli eventi correlati al record:

- Ora evento: un timestamp definito dall'utente che definisce quando si è verificato l'evento che ha creato il record.
- Tempo di acquisizione: l'ora in cui l'applicazione ha recuperato il record dal flusso di dati.
- Tempo di elaborazione: l'ora in cui l'applicazione ha elaborato il record.

Quando l'API Apache Flink Table crea finestre basate su tempi record, definisci quale di questi timestamp utilizza utilizzando il metodo `setStreamTimeCharacteristic`

Per ulteriori informazioni sull'utilizzo dei timestamp con l'API Table, consulta [Time Attributes](#) e [Timely Stream Processing](#) nella documentazione di Apache Flink.

Usa Python con Managed Service per Apache Flink

Note

Se stai sviluppando un'applicazione Python Flink su un nuovo Mac con chip Apple Silicon, potresti riscontrare alcuni [problemi noti con le dipendenze Python della versione 1.15](#). PyFlink In questo caso consigliamo di eseguire l'interprete Python in Docker. Per step-by-step istruzioni, consulta lo sviluppo della versione [PyFlink 1.15](#) su Apple Silicon Mac.

La versione 1.20 di Apache Flink include il supporto per la creazione di applicazioni utilizzando la versione 3.11 di Python. Per ulteriori informazioni, consulta [Flink Python Docs](#). È possibile creare un'applicazione del servizio gestito per Apache Flink con Python seguendo la procedura descritta di seguito:

- Crea il codice dell'applicazione Python come file di testo con un metodo `main`.
- Raggruppa il file di codice dell'applicazione e tutte le dipendenze Python o Java in un file zip e caricalo in un bucket Amazon S3.
- Crea la tua applicazione del servizio gestito per Apache Flink, specificando la posizione del codice Amazon S3, le proprietà dell'applicazione e le sue impostazioni.

Ad alto livello, l'API Python Table è un wrapper per l'API Java Table. Per informazioni sull'API Python Table, consulta il [Table API Tutorial](#) nella documentazione di Apache Flink.

Programma il tuo Managed Service per l'applicazione Apache Flink Python

Puoi codificare la tua applicazione del servizio gestito per Apache Flink per Python utilizzando l'API Apache Flink Python Table. Il motore Apache Flink traduce le istruzioni dell'API Python Table (in esecuzione nella macchina virtuale Python) in istruzioni dell'API Java Table (in esecuzione nella macchina virtuale Java).

Utilizza l'API Table Python seguendo la procedura descritta di seguito:

- Crea un riferimento a `StreamTableEnvironment`.

- Crea oggetti table dai tuoi dati di streaming di origine eseguendo query sul riferimento `StreamTableEnvironment`.
- Esegui interrogazioni sui tuoi oggetti table per creare tabelle di output.
- Scrivi le tue tabelle di output nelle tue destinazioni usando uno `StatementSet`.

Per iniziare a utilizzare l'API Python Table nel servizio gestito per Apache Flink, consulta. [Nozioni di base sull'utilizzo del servizio gestito da Amazon per Apache Flink per Python](#)

Leggi e scrivi dati in streaming

Per leggere e scrivere dati in streaming, esegui query SQL nell'ambiente di tabella.

Creare una tabella

Il seguente esempio di codice illustra una funzione definita dall'utente che crea una query SQL. La query SQL crea una tabella che interagisce con un flusso Kinesis:

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

Leggi i dati di streaming

Il seguente esempio di codice mostra come utilizzare la query `CreateTableSQL` precedente su un riferimento di ambiente di tabella per leggere i dati:

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

Scrivi dati di streaming

Il seguente esempio di codice mostra come utilizzare la query SQL dell'esempio `CreateTable` per creare un riferimento alla tabella di output e come utilizzare uno `StatementSet` per interagire con le tabelle per scrivere dati su un flusso Kinesis di destinazione:

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

Leggi le proprietà di runtime

È possibile utilizzare le proprietà di runtime per configurare l'applicazione senza cambiare il codice dell'applicazione.

È possibile specificare le proprietà dell'applicazione nello stesso modo in cui si specifica un'applicazione del servizio gestito per Apache Flink per Java. È possibile specificare le proprietà di runtime nei seguenti modi:

- Utilizzo dell'[CreateApplication](#)azione.
- Utilizzo dell'[UpdateApplication](#)azione.
- Configurando la tua applicazione tramite la console.

È possibile recuperare le proprietà dell'applicazione nel codice leggendo un file json chiamato `application_properties.json` creato dal runtime del servizio gestito di Apache Flink.

Il seguente esempio di codice mostra come effettuare la lettura delle proprietà dell'applicazione dal file `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)
```

Il seguente esempio di codice di funzione definito dall'utente mostra come effettuare la lettura di un gruppo di proprietà dall'oggetto delle proprietà dell'applicazione: `retrieves`:

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

Il seguente esempio di codice mostra come effettuare la lettura di una proprietà denominata `INPUT_STREAM_KEY` da un gruppo di proprietà restituito dall'esempio precedente:

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

Crea il pacchetto di codice della tua applicazione

Una volta creata l'applicazione Python, raggruppa il file di codice e le dipendenze in un file zip.

Il file zip deve contenere uno script python con un metodo `main` e può facoltivamente contenere quanto segue:

- File di codice Python
- Codice Java definito dall'utente nei file JAR
- Librerie Java nei file JAR

Note

Il file zip dell'applicazione deve contenere tutte le dipendenze dell'applicazione. Non puoi fare riferimento a librerie da altre origini per la tua applicazione.

Crea il tuo servizio gestito per l'applicazione Apache Flink Python

Specificate i vostri file di codice

Dopo che è stato creato, il pacchetto di codice dell'applicazione deve essere caricato in un bucket Amazon S3. Quindi crei l'applicazione utilizzando la console o l'[CreateApplication](#) azione.

Quando create l'applicazione utilizzando l'[CreateApplication](#) azione, specificate i file di codice e gli archivi nel file zip utilizzando uno speciale gruppo di proprietà dell'applicazione denominato `kinesis.analytics.flink.run.options`. Puoi definire i seguenti tipi di file:

- `python`: un file di testo contenente un metodo principale Python.

- **jarfile**: un file Java JAR contenente funzioni Java definite dall'utente.
- **pyFiles**: un file di risorse Python contenente risorse che devono essere utilizzate dall'applicazione.
- **pyArchives**: un file zip contenente i file di risorse per l'applicazione.

Per ulteriori informazioni sui tipi di file di codice Python di Apache Flink, consulta [Command-Line Interface](#) nella documentazione di Apache Flink.

Note

Il servizio gestito per Apache Flink non supporta i tipi di file `pyModule`, `pyExecutable` o `pyRequirements`. Tutto il codice, tutti i requisiti e tutte le dipendenze devono essere contenuti nel file zip. Non è possibile specificare le dipendenze da installare utilizzando `pip`.

Il seguente esempio di frammento json mostra come specificare le posizioni dei file all'interno del file zip dell'applicazione:

```
"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"
        }
      }
    ]
  },
}
```

Monitora l'applicazione Managed Service for Apache Flink Python

Utilizzate il CloudWatch registro dell'applicazione per monitorare l'applicazione Managed Service for Apache Flink Python.

Il servizio gestito per Apache Flink effettua il log dei seguenti messaggi per applicazioni Python:

- Messaggi scritti sulla console utilizzando `print()` nel metodo `main` dell'applicazione.

- Messaggi inviati in funzioni definite dall'utente utilizzando il pacchetto `logging`. Il seguente esempio di codice mostra come effettuare il log dell'applicazione da una funzione definita dall'utente:

```
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
```

- Messaggi di errore generati dall'applicazione.

Se l'applicazione genera un'eccezione nella funzione `main`, l'eccezione verrà visualizzata nei log dell'applicazione.

L'esempio seguente mostra una voce di log per un'eccezione generata dal codice Python:

```
2021-03-15 16:21:20.000 ----- Python Process Started
-----
2021-03-15 16:21:21.000 Traceback (most recent call last):
2021-03-15 16:21:21.000 " File "/tmp/flink-
web-6118109b-1cd2-439c-9dcd-218874197fa9/flink-web-upload/4390b233-75cb-4205-
a532-441a2de83db3_code/PythonKinesisSink/PythonUdfUndeclared.py", line 101, in
<module>"
2021-03-15 16:21:21.000     main()
2021-03-15 16:21:21.000 " File "/tmp/flink-
web-6118109b-1cd2-439c-9dcd-218874197fa9/flink-web-upload/4390b233-75cb-4205-
a532-441a2de83db3_code/PythonKinesisSink/PythonUdfUndeclared.py", line 54, in main"
2021-03-15 16:21:21.000 "     table_env.register_function("doNothingUdf",
doNothingUdf)"
2021-03-15 16:21:21.000 NameError: name 'doNothingUdf' is not defined
2021-03-15 16:21:21.000 ----- Python Process Exited
-----
2021-03-15 16:21:21.000 Run python process failed
2021-03-15 16:21:21.000 Error occurred when trying to start the job
```

 Note

Per evitare possibili problemi di prestazioni, si consiglia di utilizzare solo messaggi di log personalizzati durante lo sviluppo dell'applicazione.

Interroga i log con Insights CloudWatch

La seguente query CloudWatch Insights cerca i log creati dall'entrypoint Python durante l'esecuzione della funzione principale dell'applicazione:

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

Utilizzo delle proprietà di runtime in Managed Service for Apache Flink

È possibile utilizzare le proprietà di runtime per configurare l'applicazione senza ricompilare il codice dell'applicazione.

Questo argomento contiene le sezioni seguenti:

- [Gestisci le proprietà di runtime utilizzando la console](#)
- [Gestisci le proprietà di runtime utilizzando la CLI](#)
- [Accedi alle proprietà di runtime in un'applicazione Managed Service for Apache Flink](#)

Gestisci le proprietà di runtime utilizzando la console

È possibile aggiungere, aggiornare o rimuovere le proprietà di runtime dall'applicazione Managed Service for Apache Flink utilizzando AWS Management Console.

 Note

Se utilizzi una versione precedente supportata di Apache Flink e desideri aggiornare le tue applicazioni esistenti ad Apache Flink 1.19.1, puoi farlo utilizzando gli aggiornamenti di versione di Apache Flink in loco. Con gli aggiornamenti di versione in loco, mantieni la

tracciabilità delle applicazioni su un singolo ARN tra le versioni di Apache Flink, tra cui istantanee, log, metriche, tag, configurazioni Flink e altro ancora. È RUNNING READY possibile utilizzare questa funzionalità in qualsiasi stato. Per ulteriori informazioni, consulta [Usa gli aggiornamenti di versione sul posto per Apache Flink](#).

Aggiornamento delle proprietà di runtime per un'applicazione del servizio gestito per Apache Flink

1. Apri la console Managed Service for Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Scegli l'applicazione del servizio gestito per Apache Flink. Scegli Dettagli dell'applicazione.
3. Nella pagina della tua applicazione, scegli Configura.
4. Espandi la sezione Proprietà.
5. Utilizza i controlli nella sezione Proprietà per definire un gruppo di proprietà con coppie chiave-valore. Utilizza questi controlli per aggiungere, aggiornare o rimuovere gruppi di proprietà e proprietà di runtime.
6. Scegli Aggiorna.

Gestisci le proprietà di runtime utilizzando la CLI

Puoi aggiungere, aggiornare o rimuovere le proprietà di runtime utilizzando la [AWS CLI](#).

Questa sezione include esempi di richieste di operazioni API per la configurazione delle proprietà di runtime per un'applicazione. Per informazioni su come utilizzare un file JSON come input per un'operazione API, consulta [Codice di esempio dell'API Managed Service per Apache Flink](#).

Note

Sostituisci l'ID account di esempio (**012345678901**) nell'esempio seguente con il tuo ID account.

Aggiungi proprietà di runtime durante la creazione di un'applicazione

Il seguente esempio di richiesta per l'operazione [CreateApplication](#) aggiunge due gruppi di proprietà di runtime (ProducerConfigProperties e ConsumerConfigProperties) quando crei un'applicazione:

```
{
  "ApplicationName": "MyApplication",
  "ApplicationDescription": "my java test app",
  "RuntimeEnvironment": "FLINK-1_19",
  "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"
          }
        }
      ]
    }
  }
}
```

Aggiungere e aggiornare le proprietà di runtime in un'applicazione esistente

La seguente richiesta per l'operazione [UpdateApplication](#) aggiunge o aggiorna le proprietà di runtime per un'applicazione esistente:

```
{
  "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

Se utilizzi una chiave che non ha una proprietà di runtime corrispondente in un gruppo di proprietà, il servizio gestito per Apache Flink aggiunge la coppia chiave-valore come nuova proprietà. Se utilizzi una chiave per una proprietà di runtime esistente in un gruppo di proprietà, il servizio gestito per Apache Flink aggiorna il valore della proprietà.

Rimuovere le proprietà di runtime

La seguente richiesta per l'operazione [UpdateApplication](#) rimuove tutte le proprietà di runtime e i gruppi di proprietà da un'applicazione esistente:

```

{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 3,
  "ApplicationConfigurationUpdate": {
    "EnvironmentPropertyUpdates": {

```

```
    "PropertyGroups": []  
  }  
}  
}
```

Important

Se ometti un gruppo di proprietà esistente o una chiave di proprietà esistente in un gruppo di proprietà, tale gruppo di proprietà o proprietà vengono rimossi.

Accedi alle proprietà di runtime in un'applicazione Managed Service for Apache Flink

È possibile recuperare le proprietà di runtime nel codice dell'applicazione Java utilizzando il metodo statico `KinesisAnalyticsRuntime.getApplicationProperties()`, che restituisce un oggetto `Map<String, Properties>`.

Il seguente esempio di codice Java recupera le proprietà di runtime per l'applicazione:

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

Recupera un gruppo di proprietà (come oggetto `Java.Util.Properties`) nel modo seguente:

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

In genere si configura un'origine o un sink Apache Flink passando nell'oggetto `Properties` senza bisogno di recuperare le singole proprietà. Il seguente esempio di codice dimostra come creare un'origine Flink passando un oggetto `Properties` recuperato dalle proprietà di runtime:

```
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;  
}
```

Per alcuni esempi di codice, consulta [Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink](#).

Usa i connettori Apache Flink con Managed Service for Apache Flink

I connettori Apache Flink sono componenti software che spostano i dati da e verso un'applicazione Amazon Managed Service per Apache Flink. I connettori sono integrazioni flessibili che consentono di leggere file e directory. I connettori sono costituiti da moduli completi per l'interazione con i servizi Amazon e i sistemi di terze parti.

I tipi di connettori comprendono:

- **Fonti:** fornisci dati all'applicazione da un flusso di dati Kinesis, un file, un argomento di Apache Kafka, un file o altre fonti di dati.
- **Sinks:** invia i dati dall'applicazione a un flusso di dati Kinesis, a un flusso Firehose, a un argomento Apache Kafka o ad altre destinazioni di dati.
- **I/O asincrono:** fornisce l'accesso asincrono a una fonte di dati come un database per arricchire i flussi.

I connettori Apache Flink sono archiviati nei rispettivi archivi di origine. La versione e l'artefatto dei connettori Apache Flink cambiano a seconda della versione di Apache Flink in uso e se si utilizza l'API Table o SQL. `DataStream`

Amazon Managed Service per Apache Flink supporta oltre 40 connettori sorgente e sink Apache Flink predefiniti. La tabella seguente fornisce un riepilogo dei connettori più diffusi e delle versioni associate. È inoltre possibile creare sink personalizzati utilizzando il framework Async-Sink. Per ulteriori informazioni, consulta [The Generic Asynchronous Base Sink](#) nella documentazione di Apache Flink.

Per accedere all'archivio per i connettori Apache Flink, consulta. AWS [flink-connector-aws](#)

Connettori per le versioni Flink

Connector	Flink versione 1.15	Flink versione 1.18	Versioni Flink 1.19	Versioni Flink 1.20
Kinesis Data Stream - API di origine DataStream e tabella	flink-connector-kinesis, 1.15.4	flink-connector-kinesis, 4,3,0-1,18	flink-connector-kinesis, 5,0,0-1,19	flink-connector-kinesis, 5,0,0-1,20
API Kinesis Data Stream - Sink - DataStream e Table	flink-connector-aws-kinesis-stream, 1.15.4	flink-connector-aws-kinesis-stream, 4.3.0-1.18	flink-connector-aws-kinesis-stream, 5.0.0-1.19	flink-connector-aws-kinesis-stream, 5.0.0-1.20
Kinesis Data Streams - Sorgente/Sink - SQL	flink-sql-connector-kinesis, 1.15.4	flink-sql-connector-kinesis, 4,3,0-1,18	flink-sql-connector-kinesis, 5,0,0-1,19	flink-sql-connector-kinesis-stream, 5.0.0-1.20
Kafka e Table API DataStream	flink-connector-kafka, 1.15.4	flink-connector-kafka, 3,2,0-1,18	flink-connector-kafka, 3,3,0-1,19	flink-connector-kafka, 3,3,0-1,20
Kafka - SQL	flink-sql-connector-kafka, 1.15.4	flink-sql-connector-kafka, 3,2,0-1,18	flink-sql-connector-kafka, 3,3,0-1,19	flink-sql-connector-kafka, 3,3,0-1,20
API Firehose DataStream e Table	flink-connector-aws-kinesis-firehose, 1.15.4	flink-connector-aws-firehose, 4,3,0-1,18	flink-connector-aws-firehose, 5,0,0-1,19	flink-connector-aws-firehose, 5,0,0-1,20
Firehose - SQL	flink-sql-connector-aws-kinesis-firehose, 1.15.4	flink-sql-connector-aws-manichetta antincendio, 4.3.0-1.18	flink-sql-connector-aws-manichetta antincendio, 5.0.0-1.19	flink-sql-connector-aws-manichetta antincendio, 5.0.0-1.20

Connector	Flink versione 1.15	Flink versione 1.18	Versioni Flink 1.19	Versioni Flink 1.20
DynamoDB DataStream e API per tabelle	flink-connector-dynamodb, 3,0,0-1,15	flink-connector-dynamodb, 4,3,0-1,18	flink-connector-dynamodb, 5,0,0-1,19	flink-connector-dynamodb, 5,0,0-1,20
DynamoDB - SQL	flink-sql-connector-dynamodb, 3,0-1,15	flink-sql-connector-dynamodb, 4,3,0-1,18	flink-sql-connector-dynamodb, 5,0,0-1,19	flink-sql-connector-dynamodb, 5,0,0-1,20
OpenSearch - DataStream e Table API	-	flink-connector-opensearch, 1.2.0-1,18	flink-connector-opensearch, 1,2,0-1,19	flink-connector-opensearch, 1,2,0-1,19
OpenSearch - SQL	-	flink-sql-connector-opensearch, 1,2,0-1,18	flink-sql-connector-opensearch, 1,2,0-1,19	flink-sql-connector-opensearch, 1,2,0-1,19
Servizio gestito Amazon per Prometheus DataStream	-	flink-sql-connector-opensearch, 1,2,0-1,18	flink-connector-prometheus, 1,0,0-1,19	flink-connector-prometheus, 1,0,0-1,20
Amazon SQS DataStream e API per tabelle	-	flink-sql-connector-opensearch, 1,2,0-1,18	flink-connector-sqs, 5,0,0-1,19	flink-connector-sqs, 5,0,0-1,20

Per ulteriori informazioni sui connettori in Amazon Managed Service for Apache Flink, consulta:

- [DataStream Connettori API](#)
- [Tabella dei connettori API](#)

Problemi noti

Esiste un problema noto di Apache Flink open source con il connettore Apache Kafka in Apache Flink 1.15. Questo problema è stato risolto nelle versioni successive di Apache Flink.

Per ulteriori informazioni, consulta [the section called “Problemi noti”](#).

Implementazione della tolleranza agli errori in Managed Service for Apache Flink

La creazione di checkpoint è il metodo utilizzato per implementare la tolleranza agli errori nel servizio gestito da Amazon per Apache Flink. Un checkpoint è un up-to-date backup di un'applicazione in esecuzione che viene utilizzato per il ripristino immediato in caso di interruzione o failover imprevisti dell'applicazione.

[Per i dettagli sul checkpoint nelle applicazioni Apache Flink, consulta Checkpoints nella documentazione di Apache Flink.](#)

Uno snapshot è un backup dello stato dell'applicazione creato e gestito manualmente. Gli snapshot consentono di ripristinare l'applicazione a uno stato precedente chiamando [UpdateApplication](#). Per ulteriori informazioni, consulta [Gestione dei backup delle applicazioni tramite istantanee](#).

Se abilitato per l'applicazione, il servizio di creazione di checkpoint fornisce tolleranza agli errori creando e caricando backup dei dati dell'applicazione in caso di riavvii imprevisti della stessa. I riavvii imprevisti dell'applicazione potrebbero essere causati da riavvii imprevisti dei processi, errori delle istanze, ecc. Ciò conferisce all'applicazione la stessa semantica dell'esecuzione senza errori durante tali riavvii.

Se le istantanee sono abilitate per l'applicazione e configurate utilizzando l'applicazione, il servizio fornisce la semantica di elaborazione esatta [ApplicationRestoreConfiguration](#) durante gli aggiornamenti dell'applicazione o durante il ridimensionamento o la manutenzione relativi al servizio.

Configura il checkpoint in Managed Service for Apache Flink

È possibile configurare il comportamento di creazione di checkpoint dell'applicazione. Puoi definire se mantenere lo stato di creazione di checkpoint, con quale frequenza salvare lo stato dell'applicazione nei checkpoint e l'intervallo minimo tra la fine di un'operazione di creazione checkpoint e l'inizio di un'altra.

È possibile configurare le seguenti impostazioni utilizzando le operazioni API [CreateApplication](#) o [UpdateApplication](#):

- `CheckpointingEnabled`: indica se la creazione di checkpoint è abilitata nell'applicazione.
- `CheckpointInterval`: contiene il tempo in millisecondi tra le operazioni di checkpoint (persistenza).
- `ConfigurationType`: imposta questo valore su `DEFAULT` per utilizzare il comportamento di creazione di checkpoint predefinito. Imposta questo valore su `CUSTOM` per configurare altri valori.

Note

Il comportamento di checkpoint predefinito è il seguente:

- `CheckpointingEnabled`: vero
- `CheckpointInterval`: 60000
- `MinPauseBetweenCheckpoints`: 5000

Se `ConfigurationType` è impostato su `DEFAULT`, verranno utilizzati i valori precedenti, anche se sono impostati su altri valori utilizzando o impostando i valori nel codice dell'applicazione. AWS Command Line Interface

Note

A partire da Flink 1.15, il servizio gestito per Apache Flink utilizzerà `stop-with-savepoint` durante la creazione automatica di snapshot, ovvero per l'aggiornamento, il dimensionamento o l'arresto dell'applicazione.

- `MinPauseBetweenCheckpoints`: il tempo minimo in millisecondi tra la fine di un'operazione di checkpoint e l'inizio di un'altra. L'impostazione di questo valore impedisce all'applicazione di continuare a creare checkpoint quando tale operazione richiede più tempo di quanto specificato dall'`CheckpointInterval`.

Consulta gli esempi di API di checkpointing

Questa sezione include esempi di richieste di operazioni API per la configurazione della creazione di checkpoint per un'applicazione. Per informazioni su come utilizzare un file JSON come input per un'operazione API, consulta [Codice di esempio dell'API Managed Service per Apache Flink](#).

Configura il checkpoint per una nuova applicazione

La seguente richiesta di esempio per l'operazione [CreateApplication](#) configura la creazione di checkpoint durante la creazione di un'applicazione:

```
{
  "ApplicationName": "MyApplication",
  "RuntimeEnvironment": "FLINK-1_19",
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
          "FileKey": "myflink.jar",
          "ObjectVersion": "AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
    },
    "FlinkApplicationConfiguration": {
      "CheckpointConfiguration": {
        "CheckpointingEnabled": "true",
        "CheckpointInterval": 20000,
        "ConfigurationType": "CUSTOM",
        "MinPauseBetweenCheckpoints": 10000
      }
    }
  }
}
```

Disabilita il checkpoint per una nuova applicazione

La seguente richiesta di esempio per l'operazione [CreateApplication](#) disabilita la creazione di checkpoint durante la creazione di un'applicazione:

```
{
  "ApplicationName": "MyApplication",
  "RuntimeEnvironment": "FLINK-1_19",
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
          "FileKey": "myflink.jar",

```



```

    "ObjectVersion": "AbCdEfGhIjKlMnOpQrStUvWxYz12345"
  },
  "FlinkApplicationConfiguration": {
    "CheckpointConfiguration": {
      "CheckpointingEnabled": "false"
    }
  }
}

```

Configura il checkpoint per un'applicazione esistente

La seguente richiesta di esempio per l'operazione [UpdateApplication](#) configura la creazione di checkpoint per un'applicazione esistente:

```

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

```

Disabilita il checkpoint per un'applicazione esistente

La seguente richiesta di esempio per l'operazione [UpdateApplication](#) disabilita la creazione di checkpoint per un'applicazione esistente:

```

{
  "ApplicationName": "MyApplication",
  "ApplicationConfigurationUpdate": {
    "FlinkApplicationConfigurationUpdate": {
      "CheckpointConfigurationUpdate": {
        "CheckpointingEnabledUpdate": false,
        "CheckpointIntervalUpdate": 20000,
        "ConfigurationTypeUpdate": "CUSTOM",

```

```
        "MinPauseBetweenCheckpointsUpdate": 10000
    }
}
}
```

Gestione dei backup delle applicazioni tramite istantanee

Uno snapshot è l'implementazione nel servizio gestito per Apache Flink di un savepoint di Apache Flink. Uno snapshot è un backup dello stato dell'applicazione attivato, creato e gestito dall'utente o dal servizio. [Per informazioni su Apache Flink Savepoints, consulta Savepoints nella documentazione di Apache Flink.](#) Utilizzando le istantanee, è possibile riavviare un'applicazione da una particolare istantanea dello stato dell'applicazione.

Note

È consigliabile che l'applicazione crei uno snapshot più volte al giorno per riavviarsi correttamente con i dati di stato corretti. La frequenza corretta per l'acquisizione degli snapshot dipende dalla logica di business dell'applicazione. L'acquisizione di istantanee frequenti consente di ripristinare i dati più recenti, ma aumenta i costi e richiede più risorse di sistema.

Nel servizio gestito per Apache Flink vengono gestiti gli snapshot che utilizzano le seguenti operazioni API:

- [CreateApplicationSnapshot](#)
- [DeleteApplicationSnapshot](#)
- [DescribeApplicationSnapshot](#)
- [ListApplicationSnapshots](#)

Per il limite nel numero di snapshot per ogni applicazione, consulta [Servizio gestito per Apache Flink e quota di notebook Studio](#). Se l'applicazione raggiunge il limite di snapshot, non è possibile creare manualmente uno snapshot con una `LimitExceededException`.

Il servizio gestito per Apache Flink non elimina gli snapshot. Questi snapshot dovranno essere eliminati manualmente utilizzando l'operazione [DeleteApplicationSnapshot](#).

Per caricare uno snapshot salvato dello stato dell'applicazione all'avvio di un'applicazione, utilizza il parametro [ApplicationRestoreConfiguration](#) della [StartApplication](#) o l'operazione [UpdateApplication](#).

Questo argomento contiene le sezioni seguenti:

- [Gestisci la creazione automatica di istantanee](#)
- [Esegui il ripristino da un'istananea che contiene dati di stato incompatibili](#)
- [Consulta gli esempi di API snapshot](#)

Gestisci la creazione automatica di istantanee

Se `SnapshotsEnabled` è impostato su `true` in [ApplicationSnapshotConfiguration](#) for the application, Managed Service for Apache Flink crea e utilizza automaticamente le istantanee quando l'applicazione viene aggiornata, ridimensionata o interrotta per fornire una semantica di elaborazione esatta.

Note

L'impostazione di `ApplicationSnapshotConfiguration::SnapshotsEnabled` su `false` comporterà la perdita di dati durante gli aggiornamenti dell'applicazione.

Note

Il servizio gestito per Apache Flink attiva i savepoint intermedi durante la creazione degli snapshot. A partire dalla versione 1.15 di Flink, i savepoint intermedi non producono più effetti collaterali. [Vedi Attivazione dei punti di salvataggio.](#)

Gli snapshot creati in modo automatico hanno le seguenti qualità:

- L'istananea è gestita dal servizio, ma è possibile visualizzarla utilizzando l'azione. [ListApplicationSnapshots](#) Gli snapshot creati automaticamente vengono conteggiati in base al limite di snapshot.
- Se l'applicazione supera il limite di snapshot, gli snapshot creati manualmente avranno esito negativo, ma il servizio gestito per Apache Flink continuerà a creare snapshot con successo

quando l'applicazione verrà aggiornata, dimensionata o interrotta. È necessario eliminare manualmente le istantanee utilizzando l' [DeleteApplicationSnapshot](#) azione prima di crearne altre manualmente.

Esegui il ripristino da un'istananea che contiene dati di stato incompatibili

Poiché gli snapshot contengono informazioni sugli operatori, il ripristino dei dati di stato da uno snapshot per un operatore che è stato modificato rispetto alla versione precedente dell'applicazione potrebbe avere risultati imprevisti. Un'applicazione genera un errore se tenta di ripristinare i dati di stato da uno snapshot che non corrisponde all'operatore corrente. L'applicazione con errori rimarrà bloccata nello stato STOPPING o UPDATING.

Per consentire a un'applicazione di eseguire il ripristino da un'istananea che contiene dati di stato incompatibili, impostate il `AllowNonRestoredState` parametro [FlinkRunConfiguration](#) to `true` utilizzando l'azione. [UpdateApplication](#)

Quando un'applicazione viene ripristinata da uno snapshot obsoleto, si verifica il seguente comportamento:

- Operatore aggiunto: se viene aggiunto un nuovo operatore, il savepoint non ha dati di stato per il nuovo operatore. Non si verificherà alcun errore e non è necessario impostare `AllowNonRestoredState`.
- Operatore eliminato: se viene eliminato un operatore esistente, il savepoint contiene i dati di stato per l'operatore mancante. Si verificherà un errore a meno che `AllowNonRestoredState` non sia impostato su `true`.
- Operatore modificato: se vengono apportate modifiche compatibili, ad esempio la modifica del tipo di parametro in un tipo compatibile, l'applicazione può eseguire il ripristino dallo snapshot obsoleto. Per ulteriori informazioni sul ripristino da istantanee, consulta [Savepoints](#) nella documentazione di Apache Flink. Un'applicazione che utilizza Apache Flink versione 1.8 o successiva può essere ripristinata da uno snapshot con uno schema diverso. Un'applicazione che utilizza Apache Flink versione 1.6 non può essere ripristinata. Per two-phase-commit i sink, consigliamo di utilizzare lo snapshot di sistema (SW) anziché lo snapshot creato dall'utente (). `CreateApplicationSnapshot`

Per Flink, il servizio gestito per Apache Flink attiva savepoint intermedi durante la creazione degli snapshot. A partire da Flink 1.15, i savepoint intermedi non producono più effetti collaterali. Consulta [Attivazione dei savepoint](#).

Se è necessario riprendere un'applicazione incompatibile con i dati del savepoint esistenti, si consiglia di saltare il ripristino dall'istantanea impostando il parametro dell'azione su.

`ApplicationRestoreType` [StartApplication](#) `SKIP_RESTORE_FROM_SNAPSHOT`

Per ulteriori informazioni sul modo in cui Apache Flink gestisce i dati di stato incompatibili, consulta [Evoluzione dello schema di stato](#) nella documentazione di Apache Flink.

Consulta gli esempi di API snapshot

Questa sezione include esempi di richieste di operazioni API per l'utilizzo di snapshot con un'applicazione. Per informazioni su come utilizzare un file JSON come input per un'operazione API, consulta [Codice di esempio dell'API Managed Service per Apache Flink](#).

Abilita le istantanee per un'applicazione

La seguente richiesta di esempio per l'operazione [UpdateApplication](#) abilita gli snapshot per un'applicazione:

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

Creazione di una snapshot

La seguente richiesta di esempio per l'operazione [CreateApplicationSnapshot](#) crea uno snapshot dello stato corrente dell'applicazione:

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

Elenca le istantanee di un'applicazione

La seguente richiesta di esempio per l'operazione [ListApplicationSnapshots](#) elenca i primi 50 snapshot per lo stato corrente dell'applicazione:

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

Elenca i dettagli di un'istananea dell'applicazione

La seguente richiesta di esempio per l'operazione [DescribeApplicationSnapshot](#) elenca i dettagli relativi a uno snapshot specifico per l'applicazione:

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

Eliminazione di uno snapshot

La seguente richiesta di esempio per l'operazione [DeleteApplicationSnapshot](#) elimina uno snapshot salvato in precedenza. È possibile ottenere il valore `SnapshotCreationTimestamp` utilizzando [ListApplicationSnapshots](#) o [DeleteApplicationSnapshot](#):

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

Riavviare un'applicazione utilizzando un'istananea denominata

La seguente richiesta di esempio per l'operazione [StartApplication](#) avvia l'applicazione utilizzando lo stato salvato da uno snapshot specifico:

```
{
  "ApplicationName": "MyApplication",
  "RunConfiguration": {
```

```
    "ApplicationRestoreConfiguration": {
      "ApplicationRestoreType": "RESTORE_FROM_CUSTOM_SNAPSHOT",
      "SnapshotName": "MyCustomSnapshot"
    }
  }
}
```

Riavviare un'applicazione utilizzando l'istantanea più recente

La seguente richiesta di esempio per l'operazione [StartApplication](#) avvia l'applicazione utilizzando lo snapshot più recente:

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

Riavviare un'applicazione senza utilizzare alcuna istantanea

La seguente richiesta di esempio per l'operazione [StartApplication](#) avvia l'applicazione senza caricare lo stato dell'applicazione, anche se è presente uno snapshot:

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

Usa gli aggiornamenti di versione sul posto per Apache Flink

Con gli aggiornamenti di versione in loco per Apache Flink, è possibile mantenere la tracciabilità delle applicazioni su un singolo ARN tra le versioni di Apache Flink. Ciò include istantanee, log, metriche, tag, configurazioni Flink, aumenti dei limiti di risorse e altro ancora. VPCs

Puoi eseguire aggiornamenti di versione sul posto per Apache Flink per aggiornare le applicazioni esistenti a una nuova versione di Flink in Amazon Managed Service for Apache Flink. Per eseguire questa operazione, puoi utilizzare,, SDK o. AWS CLI AWS CloudFormation AWS AWS Management Console

Note

Non puoi utilizzare aggiornamenti di versione in loco per Apache Flink con Amazon Managed Service per Apache Flink Studio.

Questo argomento contiene le sezioni seguenti:

- [Aggiorna le applicazioni utilizzando aggiornamenti di versione in loco per Apache Flink](#)
- [Aggiorna la tua applicazione a una nuova versione di Apache Flink](#)
- [Ripristina gli aggiornamenti delle applicazioni](#)
- [Procedure consigliate e consigli generali per gli aggiornamenti delle applicazioni](#)
- [Precauzioni e problemi noti relativi agli aggiornamenti delle applicazioni](#)

Aggiorna le applicazioni utilizzando aggiornamenti di versione in loco per Apache Flink

Prima di iniziare, ti consigliamo di guardare questo video: [Aggiornamenti delle versioni in loco](#).

Per eseguire aggiornamenti di versione sul posto per Apache Flink, puoi utilizzare,, SDK o. AWS CLI AWS CloudFormation AWS AWS Management ConsoleÈ possibile utilizzare questa funzionalità con qualsiasi applicazione esistente utilizzata con Managed Service for Apache Flink in uno stato or. READY RUNNING Utilizza l' UpdateApplication API per aggiungere la possibilità di modificare il runtime di Flink.

Prima dell'aggiornamento: aggiorna la tua applicazione Apache Flink

Quando scrivi le tue applicazioni Flink, le raggruppi con le relative dipendenze in un JAR dell'applicazione e carichi il JAR nel tuo bucket Amazon S3. Da lì, Amazon Managed Service for Apache Flink esegue il job nel nuovo runtime Flink che hai selezionato. Potrebbe essere necessario aggiornare le applicazioni per ottenere la compatibilità con il runtime Flink a cui desideri eseguire

l'aggiornamento. Possono esserci delle incongruenze tra le versioni di Flink che impediscono l'aggiornamento della versione. Più comunemente, ciò avverrà con connettori per sorgenti (ingresso) o destinazioni (lavandini, uscite) e dipendenze di Scala. Flink 1.15 e le versioni successive di Managed Service for Apache Flink sono indipendenti dalla scala e il tuo JAR deve contenere la versione di Scala che intendi utilizzare.

Per aggiornare l'applicazione

1. Leggi i consigli della community di Flink sull'aggiornamento delle applicazioni con state. Vedi [Aggiornamento delle applicazioni e delle versioni di Flink](#).
2. Leggi l'elenco dei problemi e delle limitazioni più comuni. Per informazioni, consulta [Precauzioni e problemi noti relativi agli aggiornamenti delle applicazioni](#).
3. Aggiorna le tue dipendenze e testa le tue applicazioni localmente. Queste dipendenze sono in genere:
 1. Il runtime e l'API di Flink.
 2. Connettori consigliati per il nuovo runtime Flink. Puoi trovarli [nelle versioni Release](#) per il runtime specifico a cui desideri eseguire l'aggiornamento.
 3. Scala — Apache Flink è indipendente dalla scala a partire da Flink 1.15 incluso. È necessario includere le dipendenze di Scala che si desidera utilizzare nel JAR dell'applicazione.
4. Crea una nuova applicazione JAR su zipfile e caricala su Amazon S3. Ti consigliamo di utilizzare un nome diverso dal precedente JAR/zipFile. Se devi eseguire il rollback, utilizzerai queste informazioni.
5. Se esegui applicazioni con stato, ti consigliamo vivamente di scattare un'istantanea dell'applicazione corrente. In questo modo è possibile eseguire il rollback in modalità statefully in caso di problemi durante o dopo l'aggiornamento.

Aggiorna la tua applicazione a una nuova versione di Apache Flink

Puoi aggiornare la tua applicazione Flink utilizzando l'[UpdateApplication](#)azione.

Puoi chiamare l'UpdateApplicationAPI in diversi modi:

- Utilizza il flusso di lavoro di configurazione esistente su AWS Management Console.
 - Vai alla pagina della tua app su AWS Management Console.
 - Scegli Configura.

- Seleziona il nuovo runtime e l'istanza da cui vuoi iniziare, nota anche come configurazione di ripristino. Utilizza l'impostazione più recente come configurazione di ripristino per avviare l'app dall'ultima istanza. Seleziona la nuova applicazione aggiornata JAR/zip su Amazon S3.
- [Usa l'azione update-application. AWS CLI](#)
- Usa AWS CloudFormation (CFN).
 - Aggiorna il [RuntimeEnvironment](#) campo. In precedenza, AWS CloudFormation eliminava l'applicazione e ne creava una nuova, causando la perdita delle istanze e della cronologia dell'altra app. Ora AWS CloudFormation aggiorna la RuntimeEnvironment versione in uso e non elimina l'applicazione.
- Usa l' AWS SDK.
 - Consulta la documentazione SDK per il linguaggio di programmazione che preferisci. Per informazioni, consulta [UpdateApplication](#).

È possibile eseguire l'aggiornamento mentre l'applicazione è in RUNNING stato o mentre l'applicazione è arrestata in READY tale stato. Amazon Managed Service for Apache Flink esegue la convalida per verificare la compatibilità tra la versione di runtime originale e la versione di runtime di destinazione. Questo controllo di compatibilità viene eseguito quando lo esegui [UpdateApplication](#) mentre sei in RUNNING uno stato o successivamente [StartApplication](#) se esegui l'upgrade mentre sei in stato. READY

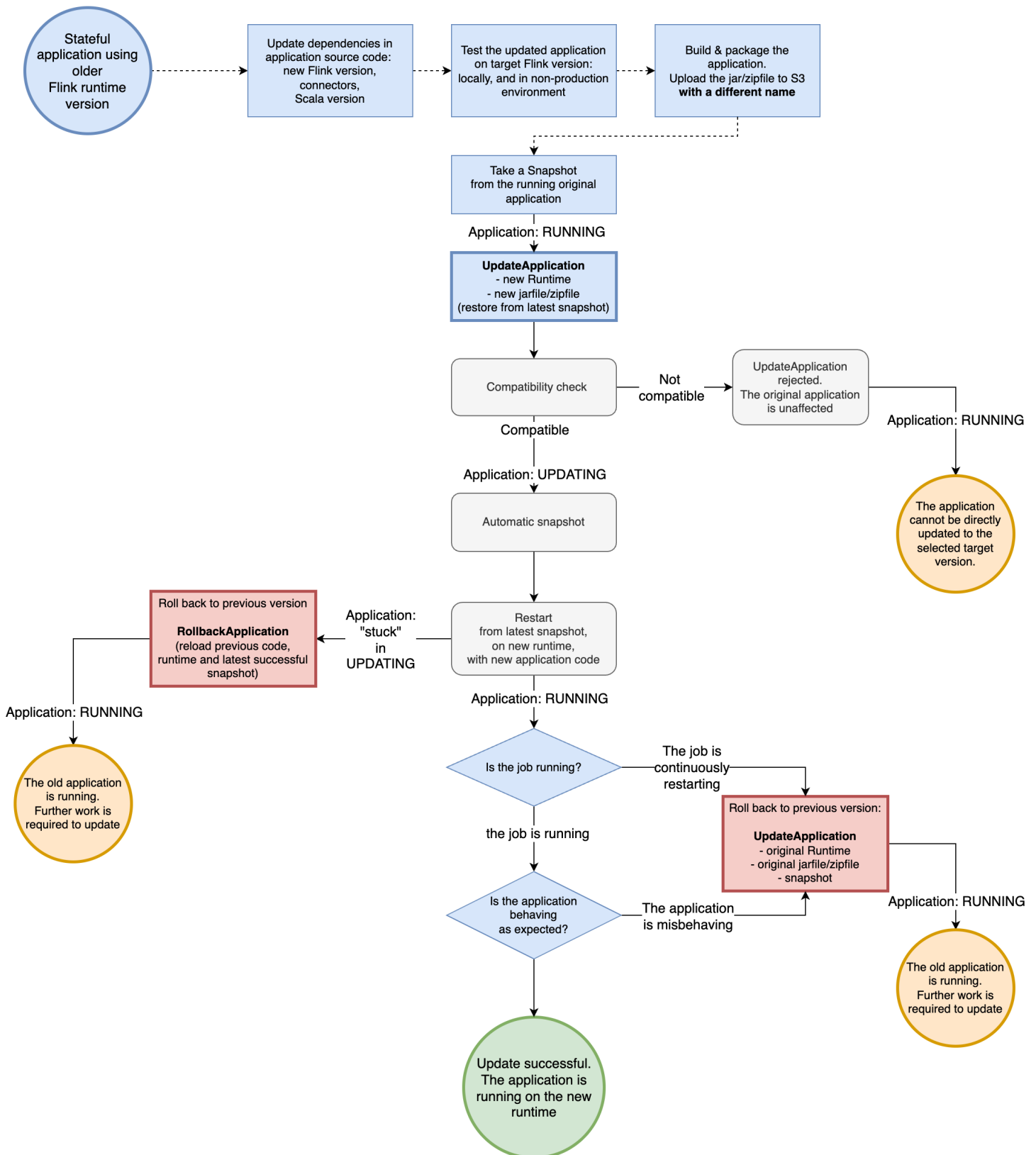
Aggiorna un'applicazione in **RUNNING** stato

L'esempio seguente mostra l'aggiornamento di un'app nello RUNNING stato denominato UpgradeTest Flink 1.18 negli Stati Uniti orientali (Virginia settentrionale) utilizzando AWS CLI e l'avvio dell'app aggiornata dall'ultima istanza.

```
aws --region us-east-1 kinesisanalyticstv2 update-application \  
--application-name UpgradeTest --runtime-environment-update "FLINK-1_18" \  
--application-configuration-update '{"ApplicationCodeConfigurationUpdate": \'\  
{ "CodeContentUpdate": { "S3ContentLocationUpdate": \'\  
{ "FileKeyUpdate": "flink_1_18_app.jar" } } }\' \  
--run-configuration-update '{"ApplicationRestoreConfiguration": \'\  
{ "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT" } }\' \  
--current-application-version-id ${current_application_version}
```

- Se hai abilitato gli snapshot del servizio e desideri continuare l'applicazione dallo snapshot più recente, Amazon Managed Service for Apache Flink verifica che il runtime dell'`RUNNING` applicazione corrente sia compatibile con il runtime di destinazione selezionato.
- Se hai specificato uno snapshot da cui continuare il runtime di destinazione, Amazon Managed Service for Apache Flink verifica che il runtime di destinazione sia compatibile con lo snapshot specificato. Se il controllo di compatibilità fallisce, la richiesta di aggiornamento viene rifiutata e l'applicazione rimane invariata nello stato `RUNNING`.
- Se scegli di avviare l'applicazione senza uno snapshot, Amazon Managed Service for Apache Flink non esegue alcun controllo di compatibilità.
- Se l'applicazione aggiornata fallisce o rimane bloccata in uno `UPDATING` stato transitorio, segui le istruzioni nella [Ripristina gli aggiornamenti delle applicazioni](#) sezione per tornare allo stato integro.

Flusso di processo per l'esecuzione di applicazioni statali



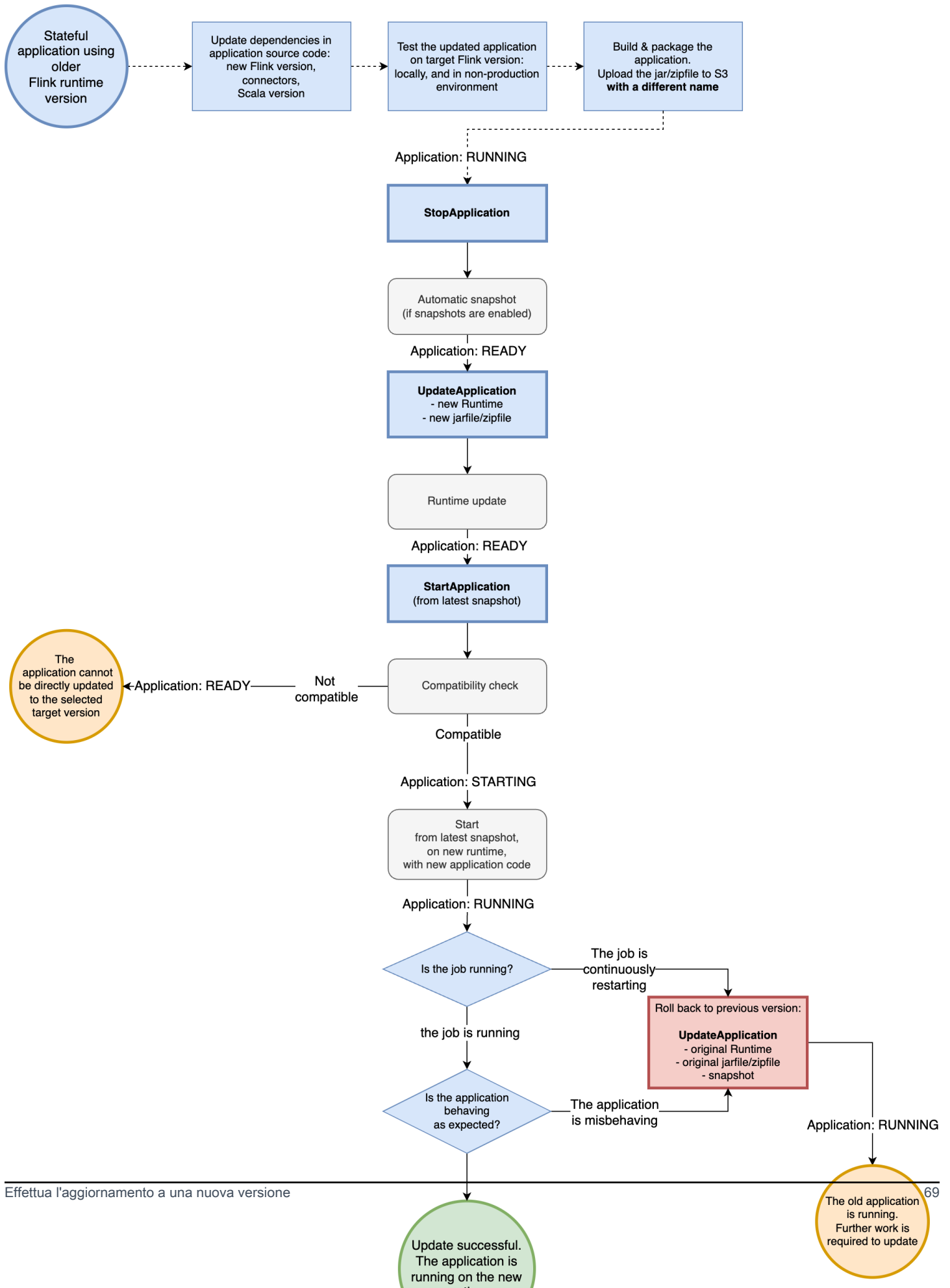
Aggiorna un'applicazione nello stato READY

L'esempio seguente mostra l'aggiornamento di un'app nello READY stato denominato UpgradeTest Flink 1.18 negli Stati Uniti orientali (Virginia settentrionale) utilizzando il. AWS CLI Non esiste un'istanza specificata per avviare l'app perché l'applicazione non è in esecuzione. È possibile specificare un'istanza quando si invia la richiesta di avvio dell'applicazione.

```
aws --region us-east-1 kinesisanalyticsv2 update-application \
--application-name UpgradeTest --runtime-environment-update "FLINK-1_18" \
--application-configuration-update '{"ApplicationCodeConfigurationUpdate": '\
'{"CodeContentUpdate": {"S3ContentLocationUpdate": '\
'{"FileKeyUpdate": "flink_1_18_app.jar"}}}' \
--current-application-version-id ${current_application_version}
```

- È possibile aggiornare il runtime delle applicazioni in READY stato a qualsiasi versione di Flink. Amazon Managed Service for Apache Flink non esegue alcun controllo finché non avvii l'applicazione.
- Amazon Managed Service for Apache Flink esegue solo controlli di compatibilità sullo snapshot selezionato per avviare l'app. Si tratta di controlli di compatibilità di base che seguono la tabella di compatibilità [Flink](#). Controllano solo la versione di Flink con cui è stata scattata l'istanza e la versione di Flink che hai scelto come target. Se il runtime Flink dell'istanza selezionata è incompatibile con il nuovo runtime dell'app, la richiesta di avvio potrebbe essere rifiutata.

Flusso di processo per applicazioni Ready State



Ripristina gli aggiornamenti delle applicazioni

Se riscontri problemi con l'applicazione o riscontri incongruenze nel codice dell'applicazione tra le versioni di Flink, puoi eseguire il rollback utilizzando AWS CLI, AWS CloudFormation, AWS SDK o AWS Management Console. Gli esempi seguenti mostrano come si presenta il rollback in diversi scenari di errore.

L'aggiornamento del runtime è riuscito, l'applicazione è in **RUNNING** stato, ma il processo non riesce e viene riavviato continuamente

Supponiamo che stiate cercando di aggiornare un'applicazione stateful denominata `TestApplication` da Flink 1.15 a Flink 1.18 negli Stati Uniti orientali (Virginia settentrionale). Tuttavia, l'applicazione Flink 1.18 aggiornata non si avvia o si riavvia costantemente, anche se l'applicazione è in stato `RUNNING`. Si tratta di uno scenario di errore comune. Per evitare ulteriori tempi di inattività, si consiglia di ripristinare immediatamente l'applicazione alla versione precedente in esecuzione (Flink 1.15) e di diagnosticare il problema in un secondo momento.

Per ripristinare l'applicazione alla versione precedente in esecuzione, utilizzate il comando [AWS CLI rollback-application](#) o l'azione API [RollbackApplication](#). Questa azione API ripristina le modifiche che hai apportato e che hanno portato alla versione più recente. Quindi riavvia l'applicazione utilizzando l'ultima istantanea riuscita.

Ti consigliamo vivamente di scattare un'istantanea con l'app esistente prima di tentare l'aggiornamento. Ciò contribuirà a evitare la perdita di dati o la necessità di rielaborarli.

In questo scenario di errore, l'applicazione non AWS CloudFormation verrà ripristinata automaticamente. È necessario aggiornare il CloudFormation modello in modo che punti al runtime precedente e al codice precedente per CloudFormation forzare l'aggiornamento dell'applicazione. Altrimenti, CloudFormation si presuppone che l'applicazione sia stata aggiornata quando passa allo `RUNNING` stato.

Ripristino di un'applicazione bloccata **UPDATING**

Se l'applicazione si blocca nello `AUTOSCALING` stato `UPDATING` o dopo un tentativo di aggiornamento, Amazon Managed Service for Apache Flink offre il AWS CLI comando [rollback-applications](#) o l'azione [RollbackApplications](#) API che può ripristinare l'applicazione alla versione precedente al blocco o allo stato `UPDATING` `AUTOSCALING`. Questa API ripristina le modifiche che hai apportato che hanno causato il blocco o lo stato transitorio dell'applicazione. `UPDATING` `AUTOSCALING`

Procedure consigliate e consigli generali per gli aggiornamenti delle applicazioni

- Testa il nuovo job/runtime senza stato in un ambiente non di produzione prima di tentare un aggiornamento di produzione.
- Valuta la possibilità di testare prima l'aggiornamento stateful con un'applicazione non di produzione.
- Assicurati che il tuo nuovo job graph abbia uno stato compatibile con l'istantanea che utilizzerai per avviare l'applicazione aggiornata.
 - Assicurati che i tipi memorizzati negli stati dell'operatore rimangano gli stessi. Se il tipo è cambiato, Apache Flink non può ripristinare lo stato dell'operatore.
 - Assicurati che l'operatore che IDs hai impostato utilizzando il `uid` metodo rimanga lo stesso. Apache Flink consiglia vivamente l'assegnazione di elementi unici IDs agli operatori. Per ulteriori informazioni, consulta [Assigning Operator IDs nella documentazione](#) di Apache Flink.

Se non lo assegnate IDs ai vostri operatori, Flink li genera automaticamente. In tal caso, potrebbero dipendere dalla struttura del programma e, se modificate, causare problemi di compatibilità. Flink utilizza Operator IDs per abbinare lo stato nell'istantanea all'operatore. La modifica dell'operatore IDs comporta il mancato avvio dell'applicazione, l'eliminazione dello stato memorizzato nell'istantanea e l'avvio del nuovo operatore senza stato.

- Non modificate la chiave utilizzata per memorizzare lo stato della chiave.
- Non modificate il tipo di input degli operatori statici come window o join. Ciò modifica implicitamente il tipo di stato interno dell'operatore, causando un'incompatibilità di stato.

Precauzioni e problemi noti relativi agli aggiornamenti delle applicazioni

Kafka Commit sul checkpoint fallisce ripetutamente dopo il riavvio del broker

Esiste un problema noto di Apache Flink open source con il connettore Apache Kafka nella versione 1.15 di Flink, causato da un bug critico di Kafka Client open source in Kafka Client 2.8.1. [Per ulteriori informazioni, vedi Kafka Commit on checkpointing fallisce ripetutamente dopo il riavvio del broker e non è in grado di ripristinare la connessione al coordinatore del gruppo dopo un'eccezione. KafkaConsumer commitOffsetAsync](#)

Per evitare questo problema, ti consigliamo di utilizzare Apache Flink 1.18 o versione successiva in Amazon Managed Service for Apache Flink.

Limitazioni note della compatibilità tra stati

- Se utilizzi l'API Table, Apache Flink non garantisce la compatibilità degli stati tra le versioni di Flink. Per ulteriori informazioni, consulta [Stateful Upgrades and Evolution](#) nella documentazione di Apache Flink.
- Gli stati di Flink 1.6 non sono compatibili con Flink 1.18. L'API rifiuta la tua richiesta se tenti di eseguire l'aggiornamento da 1.6 a 1.18 e versioni successive con state. Puoi eseguire l'aggiornamento alla versione 1.8, 1.11, 1.13 e 1.15 e scattare un'istantanea, quindi eseguire l'aggiornamento alla versione 1.18 e successive. Per ulteriori informazioni, consulta [Aggiornamento delle applicazioni e delle versioni Flink nella documentazione di Apache Flink](#).

Problemi noti con il connettore Flink Kinesis

- Se si utilizza Flink 1.11 o versioni precedenti e si utilizza il `amazon-kinesis-connector-flink` connettore per il supporto Enhanced-fan-out (EFO), è necessario eseguire ulteriori passaggi per un aggiornamento stateful a Flink 1.13 o versione successiva. Ciò è dovuto alla modifica del nome del pacchetto del connettore. Per ulteriori informazioni, consulta [amazon-kinesis-connector-flink](#).

Il `amazon-kinesis-connector-flink` connettore per Flink 1.11 e versioni precedenti utilizza la confezione `software.amazon.kinesis`, mentre il connettore Kinesis per Flink 1.13 e versioni successive utilizza `org.apache.flink.streaming.connectors.kinesis`. [Utilizzate questo strumento per supportare la migrazione: -state-migrator. amazon-kinesis-connector-flink](#)

- Se si utilizza Flink 1.13 o versioni precedenti `FlinkKinesisProducer` e si esegue l'aggiornamento a Flink 1.15 o successivo, per un aggiornamento statico è necessario continuare a utilizzare in Flink 1.15 o versione successiva, anziché la versione più recente. `FlinkKinesisProducer` `KinesisStreamsSink` Tuttavia, se hai già un uid set personalizzato sul tuo sink, dovresti essere in grado di passare a perché non mantiene lo stato. `KinesisStreamsSink` `FlinkKinesisProducer` Flink lo tratterà come lo stesso operatore perché uid è impostata una personalizzazione.

Applicazioni Flink scritte in Scala

- A partire da Flink 1.15, Apache Flink non include Scala nel runtime. È necessario includere la versione di Scala che si desidera utilizzare e altre dipendenze di Scala nel codice JAR/zip quando si esegue l'aggiornamento a Flink 1.15 o successivo. Per ulteriori informazioni, consulta [Amazon Managed Service for Apache Flink for Apache Flink release 1.15.2](#).

- Se la tua applicazione utilizza Scala e la stai aggiornando da Flink 1.11 o precedente (Scala 2.11) a Flink 1.13 (Scala 2.12), assicurati che il codice utilizzi Scala 2.12. Altrimenti, l'applicazione Flink 1.13 potrebbe non riuscire a trovare le classi Scala 2.11 nel runtime Flink 1.13.

Aspetti da considerare quando si esegue il downgrade dell'applicazione Flink

- Il downgrade delle applicazioni Flink è possibile, ma limitato ai casi in cui l'applicazione era precedentemente in esecuzione con la versione precedente di Flink. Per un aggiornamento statico, Managed Service for Apache Flink richiederà l'utilizzo di un'istanza scattata con una versione corrispondente o precedente per il downgrade
- Se state aggiornando il runtime da Flink 1.13 o versione successiva a Flink 1.11 o precedente e se l'app utilizza il backend di stato, l'applicazione fallirà continuamente. HashMap

Implementazione della scalabilità delle applicazioni in Managed Service for Apache Flink

È possibile configurare l'esecuzione parallela delle attività e l'allocazione delle risorse per il servizio gestito da Amazon per Apache Flink, al fine di implementare il dimensionamento. Per informazioni su come Apache Flink pianifica le istanze parallele di attività, consulta [Parallel Execution](#) nella documentazione di Apache Flink.

Argomenti

- [Configura il parallelismo dell'applicazione e la KPU ParallelismPer](#)
- [Allocazione delle unità di elaborazione Kinesis](#)
- [Aggiorna il parallelismo dell'applicazione](#)
- [Utilizza il ridimensionamento automatico in Managed Service for Apache Flink](#)
- [Considerazioni su maxParallelism](#)

Configura il parallelismo dell'applicazione e la KPU ParallelismPer

È possibile configurare l'esecuzione parallela per le attività dell'applicazione servizio gestito per Apache Flink (leggere da un'origine o eseguire un operatore, per esempio) utilizzando le proprietà [ParallelismConfiguration](#):

- **Parallelism**: utilizza questa proprietà per impostare il parallelismo predefinito dell'applicazione di Apache Flink. Tutti gli operatori, le origini e i sink vengono eseguiti con questo parallelismo, a meno che non siano sovrascritti nel codice dell'applicazione. Il valore predefinito è 1, il valore massimo predefinito è 256.
- **ParallelismPerKPU**: utilizza questa proprietà per impostare il numero di task paralleli che è possibile pianificare per ogni unità di elaborazione Kinesis (KPU, Kinesis Processing Unit) dell'applicazione. Il valore predefinito è 1, il valore massimo predefinito è 8. Per le applicazioni che prevedono operazioni di blocco (ad esempio I/O), un valore più elevato di **ParallelismPerKPU** implica di utilizzare le risorse KPU nella loro totalità.

Note

Il limite di **Parallelism** è uguale a **ParallelismPerKPU** volte il limite di KPUs (che ha un valore predefinito di 64). Il KPUs limite può essere aumentato richiedendo un aumento del limite. Per istruzioni su come richiedere un aumento del limite, consultare "Richiedere un aumento del limite" in [Service Quotas](#).

Per informazioni sull'impostazione del parallelismo delle attività per un operatore specifico, consulta [Impostazione del parallelismo: operatore](#) nella documentazione di Apache Flink.

Allocazione delle unità di elaborazione Kinesis

Managed Service for Apache Flink fornisce capacità come. KPUs Una singola KPU offre 1 vCPU e 4 GB di memoria. Per ogni KPU allocata, vengono forniti anche 50 GB di spazio di archiviazione delle applicazioni in esecuzione.

Managed Service for Apache Flink calcola il numero KPUs necessario per eseguire l'applicazione utilizzando le proprietà **Parallelism** and **ParallelismPerKPU**, come segue:

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

Il servizio gestito per Apache Flink fornisce rapidamente risorse alle applicazioni in risposta ai picchi della velocità di trasmissione effettiva o di attività di elaborazione. Rimuove gradualmente le risorse dall'applicazione dopo il superamento del picco di attività. Per disabilitare l'allocazione automatica delle risorse, è sufficiente impostare il valore **AutoScalingEnabled** su **false**, come descritto in seguito in [Aggiorna il parallelismo dell'applicazione](#).

Il limite predefinito KPU per l'applicazione è 64. Per istruzioni su come richiedere un aumento di tale limite, consulta "Richiedere un aumento del limite" in [Service Quotas](#).

Note

A scopo di orchestrazione viene addebitata una KPU aggiuntiva. Per ulteriori informazioni, consulta il [Piano tariffario del servizio gestito da Amazon per Apache Flink](#).

Aggiorna il parallelismo dell'applicazione

Questa sezione contiene esempi di richieste di azioni API che impostano il parallelismo di un'applicazione. Per ulteriori esempi e istruzioni sull'utilizzo dei blocchi di richiesta con le azioni API, consulta [Codice di esempio dell'API Managed Service per Apache Flink](#).

Il seguente esempio di richiesta per l'azione [CreateApplication](#) imposta il parallelismo durante la creazione di un'applicazione:

```
{
  "ApplicationName": "string",
  "RuntimeEnvironment": "FLINK-1_18",
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
          "FileKey": "myflink.jar",
          "ObjectVersion": "AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "CodeContentType": "ZIPFILE"
    },
    "FlinkApplicationConfiguration": {
      "ParallelismConfiguration": {
        "AutoScalingEnabled": "true",
        "ConfigurationType": "CUSTOM",
        "Parallelism": 4,
        "ParallelismPerKPU": 4
      }
    }
  }
}
```

```
}  
}
```

Il seguente esempio di richiesta per l'azione [UpdateApplication](#) imposta il parallelismo per un'applicazione pre-esistente:

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

Il seguente esempio di richiesta per l'azione [UpdateApplication](#) disabilita il parallelismo per un'applicazione pre-esistente:

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

Utilizza il ridimensionamento automatico in Managed Service for Apache Flink

Il servizio gestito per Apache Flink ridimensiona in modo elastico il parallelismo dell'applicazione, per adattarsi alla velocità di trasmissione effettiva dei dati della fonte e alla complessità dell'operatore

nella maggior parte delle situazioni. Il dimensionamento automatico è abilitato per impostazione predefinita. Il servizio gestito per Apache Flink monitora l'utilizzo delle risorse (CPU) da parte dell'applicazione e aumenta o diminuisce elasticamente il parallelismo dell'applicazione in base alle esigenze:

- L'applicazione si ridimensiona (aumenta il parallelismo) se il valore massimo della CloudWatch metrica `containerCPUUtilization` è superiore o superiore al 75% per 15 minuti. Ciò significa che l'`ScaleUp`azione viene avviata quando ci sono 15 punti dati consecutivi con un periodo di 1 minuto pari o superiore al 75%. Un'`ScaleUp`azione raddoppia la durata dell'applicazione. `CurrentParallelism` `ParallelismPerKPU`non viene modificato. Di conseguenza, KPU anche il numero di stanziati raddoppia.
- L'applicazione si riduce (diminuisce il parallelismo) quando l'utilizzo della CPU rimane inferiore al 10% per sei ore. Ciò significa che l'`ScaleDown`azione viene avviata quando ci sono 360 punti dati consecutivi con un periodo di 1 minuto inferiore al 10 per cento. Un'`ScaleDown`azione dimezza (arrotonda per eccesso) il parallelismo dell'applicazione. `ParallelismPerKPU`non viene modificato e KPU anche il numero di allocazioni viene dimezzato (arrotondato per eccesso).

Note

È possibile fare riferimento a un periodo massimo di `containerCPUUtilization` oltre 1 minuto per trovare la correlazione con un datapoint utilizzato per l'azione `Scaling`, ma non è necessario indicare il momento esatto in cui l'azione viene inizializzata.

Il servizio gestito per Apache Flink non ridurrà il valore di `CurrentParallelism` dell'applicazione a un valore inferiore rispetto all'impostazione `Parallelism`.

Quando il servizio gestito per Apache Flink dimensiona l'applicazione, questa comparirà nello status `AUTOSCALING`. È possibile verificare lo stato attuale dell'applicazione utilizzando le azioni o. [DescribeApplication](#) [ListApplications](#) Mentre il servizio sta scalando l'applicazione, l'unica azione API valida che puoi utilizzare è [StopApplication](#) con il `Force` parametro impostato `true` su.

È possibile utilizzare la proprietà `AutoScalingEnabled` (parte di [FlinkApplicationConfiguration](#)) per abilitare o disabilitare il dimensionamento automatico. Al tuo AWS account vengono addebitate le disposizioni relative al servizio gestito per KPU Apache Flink, che è una funzione dell'`parallelism`applicazione e delle impostazioni. `parallelismPerKPU` Un picco di attività aumenta i costi del servizio gestito per Apache Flink.

Per ulteriori informazioni sulle tariffe, consulta il [Piano tariffario del servizio gestito da Amazon per Apache Flink](#).

È fondamentale tenere presente quanto segue in merito al dimensionamento dell'applicazione:

- Il dimensionamento automatico è abilitato per impostazione predefinita.
- Il dimensionamento non si applica ai notebook Studio. Tuttavia, nel caso in cui si implementi un notebook Studio come applicazione con stato permanente, il dimensionamento verrà eseguito sull'applicazione implementata.
- L'applicazione ha un limite predefinito di 64. KPU Per ulteriori informazioni, consulta [Servizio gestito per Apache Flink e quota di notebook Studio](#).
- Quando il dimensionamento automatico aggiorna il parallelismo dell'applicazione, l'applicazione subisce un'interruzione. Segui i seguenti passaggi per evitare che l'applicazione si interrompa:
 - Disabilita il dimensionamento automatico
 - Configura `parallelism` e `parallelismPerKPU` con l'[UpdateApplication](#) azione della tua applicazione. Per ulteriori informazioni sull'impostazione delle impostazioni di parallelismo dell'applicazione, consulta [the section called "Aggiorna il parallelismo dell'applicazione"](#)
 - È fondamentale monitorare periodicamente l'utilizzo delle risorse dell'applicazione, per verificare che disponga di impostazioni di parallelismo adatte al carico di lavoro. Per informazioni sul monitoraggio dell'utilizzo delle risorse di allocazione, consulta [the section called "Metriche e dimensioni in Managed Service for Apache Flink"](#).

Implementa la scalabilità automatica personalizzata

Se desideri un controllo più preciso sulla scalabilità automatica o utilizzare metriche di attivazione diverse da quelle, puoi usare questo esempio: `containerCPUUtilization`

- [AutoScaling](#)

Questo esempio illustra come scalare l'applicazione Managed Service for Apache Flink utilizzando una CloudWatch metrica diversa dall'applicazione Apache Flink, incluse le metriche di Amazon MSK e Amazon Kinesis Data Streams, utilizzate come sorgenti o sink.

[Per ulteriori informazioni, consulta Monitoraggio avanzato e scalabilità automatica per Apache Flink.](#)

Implementa la scalabilità automatica pianificata

Se il carico di lavoro segue un profilo prevedibile nel tempo, potresti preferire scalare preventivamente la tua applicazione Apache Flink. Questo ridimensiona l'applicazione a un orario pianificato, invece di scalare in modo reattivo in base a una metrica. Per impostare la scalabilità verso l'alto e verso il basso a orari prestabiliti del giorno, puoi utilizzare questo esempio:

- [ScheduledScaling](#)

Considerazioni su maxParallelism

Il parallelismo massimo che un lavoro Flink può scalare è limitato dal minimo `maxParallelism` tra tutti gli operatori del lavoro. Ad esempio, se si dispone di un job semplice con solo una sorgente e un sink, e l'origine ha 16 e il sink ha 8, l'applicazione non può scalare oltre il parallelismo di 8. `maxParallelism`

Per sapere come viene calcolato il valore predefinito `maxParallelism` di un operatore e come sovrascriverlo, consulta [Impostazione del parallelismo massimo nella documentazione](#) di Apache Flink.

Come regola di base, tieni presente che se non definisci `maxParallelism` alcun operatore e avvii l'applicazione con un parallelismo inferiore o uguale a 128, tutti gli operatori avranno un parallelismo pari a 128. `maxParallelism`

Note

Il parallelismo massimo del processo è il limite massimo di parallelismo per scalare l'applicazione mantenendo lo stato.

Se modifichi un'applicazione esistente, l'applicazione non sarà in grado `maxParallelism` di riavviarsi da un'istantanea precedente scattata con la vecchia. `maxParallelism` È possibile riavviare l'applicazione solo senza un'istantanea.

Se si prevede di scalare l'applicazione fino a un parallelismo maggiore di 128, è necessario impostarlo in modo esplicito nell'`maxParallelism` applicazione.

- La logica di scalabilità automatica impedirà il ridimensionamento di un lavoro Flink a un parallelismo superiore al parallelismo massimo del lavoro.

- Se utilizzi una scalabilità automatica personalizzata o una scalabilità pianificata, configurale in modo che non superino il parallelismo massimo del lavoro.
- Se scalate manualmente l'applicazione oltre il parallelismo massimo, l'applicazione non si avvia.

Aggiungere tag al servizio gestito per le applicazioni Apache Flink

Questa sezione descrive come aggiungere tag di metadati chiave-valore alle applicazioni del servizio gestito per Apache Flink. Questi tag possono essere utilizzati per i seguenti scopi:

- Determinazione della fatturazione per singole applicazioni del servizio gestito per Apache Flink. Per ulteriori informazioni, consulta [Utilizzo dei tag per l'allocazione dei costi](#) nella Guida per l'utente di Billing and Cost Management.
- Controllo dell'accesso alle risorse dell'applicazione in base ai tag. Per ulteriori informazioni, consulta [Controllo degli accessi tramite tag](#) nella AWS Identity and Access Management Guida per l'utente.
- Ambiti definiti dall'utente. È possibile definire le funzionalità delle applicazioni in base alla presenza di tag utente.

Tieni in considerazione i seguenti concetti chiave durante il tagging:

- Il numero massimo di tag delle applicazioni include i tag di sistema. Il numero massimo di tag delle applicazioni definiti dall'utente è 50.
- Se un'azione include un elenco di tag con valori Key duplicati, il servizio genera `InvalidArgumentException`.

Questo argomento contiene le sezioni seguenti:

- [Aggiungi tag quando viene creata un'applicazione](#)
- [Aggiungi o aggiorna i tag per un'applicazione esistente](#)
- [Elenca i tag di un'applicazione](#)
- [Rimuovere i tag da un'applicazione](#)

Aggiungi tag quando viene creata un'applicazione

I tag vengono aggiunti durante la creazione di un'applicazione utilizzando il tags parametro dell'[CreateApplication](#)azione.

La richiesta di esempio seguente mostra il nodo Tags per una richiesta CreateApplication:

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

Aggiungi o aggiorna i tag per un'applicazione esistente

Si aggiungono tag a un'applicazione utilizzando l'[TagResource](#)azione. Non è possibile aggiungere tag a un'applicazione utilizzando l'[UpdateApplication](#)azione.

Per aggiornare un tag esistente, aggiungere un tag con la stessa chiave del tag esistente.

La seguente richiesta di esempio per l'operazione TagResource aggiunge nuovi tag o aggiorna i tag esistenti:

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

Elenca i tag di un'applicazione

Per elencare i tag esistenti, si utilizza l'[ListTagsForResource](#)azione.

L'esempio seguente di richiesta per l'[ListTagsForResource](#)azione elenca i tag per un'applicazione:

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

Rimuovere i tag da un'applicazione

Per rimuovere i tag da un'applicazione, si utilizza l'[UntagResource](#)azione.

La seguente richiesta di esempio per l'operazione `UntagResource` rimuove i tag per un'applicazione:

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

Utilizzo CloudFormation con Managed Service per Apache Flink

L'esercizio seguente mostra come avviare un'applicazione Flink creata AWS CloudFormation utilizzando una funzione Lambda nello stesso stack.

Prima di iniziare

Prima di iniziare questo esercizio, seguite i passaggi per creare un'applicazione Flink utilizzando at. AWS CloudFormation [AWS::KinesisAnalytics::Application](#)

Scrivere una funzione Lambda

Per avviare un'applicazione Flink dopo averla creata o aggiornata, utilizziamo l'API `kinesisanalyticsv2` [start-application](#). La chiamata verrà attivata da un AWS CloudFormation evento dopo la creazione

dell'applicazione Flink. Discuteremo come configurare lo stack per attivare la funzione Lambda più avanti in questo esercizio, ma prima concentriamoci sulla dichiarazione della funzione Lambda e sul relativo codice. In questo esempio utilizziamo il runtime di Python3.8.

```
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)})

```

Nel codice precedente, Lambda elabora gli eventi in AWS CloudFormation arrivo, filtra tutto Create e, ottiene lo stato dell'applicazione Update e, se lo stato lo è, la avvia. READY Per ottenere lo stato dell'applicazione, è necessario creare il ruolo Lambda, come illustrato di seguito.

Creare un ruolo Lambda

Crei un ruolo per Lambda per "parlare" con successo con l'applicazione e scrivere i log. Questo ruolo utilizza politiche gestite predefinite, ma potresti voler restringere il campo all'utilizzo di politiche personalizzate.

```
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: /
```

Tieni presente che le risorse Lambda verranno create dopo la creazione dell'applicazione Flink nello stesso stack, poiché dipendono da essa.

Richiama la funzione Lambda

Ora non resta che richiamare la funzione Lambda. A tale scopo, è possibile utilizzare una [risorsa personalizzata](#).

```
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
```

Questo è tutto ciò che serve per avviare l'applicazione Flink usando Lambda. Ora sei pronto per creare il tuo stack o utilizzare l'esempio completo riportato di seguito per vedere come funzionano nella pratica tutti questi passaggi.

Rivedi un esempio esteso

L'esempio seguente è una versione leggermente estesa dei passaggi precedenti con un'ulteriore `RunConfiguration` regolazione effettuata tramite [i parametri del modello](#). Questo è uno stack funzionante di prova. Assicurati di leggere le note di accompagnamento:

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_18'
    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 kinesisanalyticstv2 API to start an application.
        client_kda = boto3.client('kinesisanalyticstv2',
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

```

Potresti voler modificare i ruoli per Lambda e per l'applicazione stessa.

Prima di creare lo stack di cui sopra, non dimenticare di specificare i parametri.

parameters.json

[

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

Sostituisci YOUR_BUCKET_ARN e YOUR_JAR con i tuoi requisiti specifici. Puoi seguire questa [guida](#) per creare un bucket Amazon S3 e un jar di applicazioni.

Ora crea lo stack (sostituisci YOUR_REGION con una regione a tua scelta, ad esempio 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
```

Ora puoi accedere a <https://console.aws.amazon.com/cloudformation> e visualizzare i progressi. Una volta creata, dovresti vedere la tua applicazione Flink nello stato Starting. Potrebbero essere necessari alcuni minuti prima dell'Running.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- [Quattro modi per recuperare qualsiasi proprietà del AWS servizio utilizzando AWS CloudFormation \(Parte 1 di 3\)](#).
- [Procedura dettagliata: ricerca dell'immagine di Amazon Machine](#). IDs

Usa la dashboard di Apache Flink con Managed Service per Apache Flink

È possibile utilizzare il pannello di controllo di Apache Flink dell'applicazione per monitorare lo stato dell'applicazione del servizio gestito per Apache Flink. Il pannello di controllo dell'applicazione presenta le seguenti informazioni:

- Risorse in uso, inclusi task manager e task slot.
- Informazioni sui processi, inclusi quelli in esecuzione, completati, annullati e non riusciti.

Per informazioni sui task manager, i task slot e i processi di Apache Flink, consulta [Architettura di Apache Flink](#) sul sito Web di Apache Flink.

Tieni presente quanto segue riguardo all'utilizzo del pannello di controllo di Apache Flink con le applicazioni del servizio gestito per Apache Flink:

- Il pannello di controllo di Apache Flink per le applicazioni del servizio gestito per Apache Flink è di sola lettura. Non è possibile apportare modifiche all'applicazione del servizio gestito per Apache Flink utilizzando il pannello di controllo di Apache Flink.
- Il pannello di controllo di Apache Flink non è compatibile con Microsoft Internet Explorer.

Accedi alla dashboard Apache Flink della tua applicazione

Puoi accedere al pannello di controllo Apache Flink della tua applicazione tramite la console del servizio gestito per Apache Flink o richiedendo un endpoint URL sicuro tramite la CLI.

Accedi alla dashboard Apache Flink della tua applicazione utilizzando la console Managed Service for Apache Flink

Per accedere al pannello di controllo Apache Flink dell'applicazione dalla console, scegli Pannello di controllo Apache Flink nella pagina dell'applicazione.

Note

Quando apri il pannello di controllo dalla console del servizio gestito per Apache Flink, l'URL generato dalla console sarà valido per 12 ore.

Accedi alla dashboard Apache Flink della tua applicazione utilizzando il servizio gestito per Apache Flink CLI

Puoi utilizzare la CLI del servizio gestito per Apache Flink per generare un URL per accedere alla dashboard dell'applicazione. L'URL generato è valido per un intervallo di tempo specificato.

Note

Se non accedi entro tre minuti, non sarà più valido.

L'URL della dashboard viene generato utilizzando l'azione. [CreateApplicationPresignedUrl](#) È possibile specificare i valori seguenti per l'operazione:

- Nome dell'applicazione
- Tempo di validità dell'URL in secondi
- Specifica FLINK_DASHBOARD_URL come tipo di URL.

Versioni di rilascio

Questo argomento contiene informazioni sulle funzionalità supportate e sulle versioni dei componenti consigliate per ogni versione del servizio gestito per Apache Flink.

Note

Se utilizzi una versione di Apache Flink obsoleta, ti consigliamo di aggiornare l'applicazione alla versione Flink più recente supportata utilizzando la [Usa gli aggiornamenti di versione sul posto per Apache Flink](#) funzionalità di Managed Service for Apache Flink.

Versione Apache Flink	Stato - Amazon Managed Service per Apache Flink	Stato: comunità Apache Flink	Link
1.20.0	Supportato	Supportato	Servizio gestito Amazon per Apache Flink 1.20
1.19.1	Supportato	Supportato	Servizio gestito Amazon per Apache Flink 1.19
1.18.1	Supportato	Supportato	Servizio gestito Amazon per Apache Flink 1.18
1.15.2	Supportato	Non supportato.	Servizio gestito Amazon per Apache Flink 1.15
1.13.1	Supportato	Non supportato.	Nozioni di base: Flink 1.13.2
1.11.1	Obsoleto	Non supportato.	Informazioni sulle versioni precedenti

Versione Apache Flink	Stato - Amazon Managed Service per Apache Flink	Stato: comunità Apache Flink	Link
			del servizio gestito per Apache Flink (Questa versione non sarà supportata a partire da febbraio 2025)
1.8.2	Obsoleto	Non supportato.	Informazioni sulle versioni precedenti del servizio gestito per Apache Flink (Questa versione non sarà supportata a partire da febbraio 2025)
1.6.2	Obsoleto	Non supportato.	Informazioni sulle versioni precedenti del servizio gestito per Apache Flink (Questa versione non sarà supportata a partire da febbraio 2025)

Argomenti

- [Servizio gestito Amazon per Apache Flink 1.20](#)
- [Servizio gestito Amazon per Apache Flink 1.19](#)
- [Servizio gestito Amazon per Apache Flink 1.18](#)
- [Servizio gestito Amazon per Apache Flink 1.15](#)
- [Informazioni sulle versioni precedenti del servizio gestito per Apache Flink](#)

Servizio gestito Amazon per Apache Flink 1.20

Managed Service for Apache Flink ora supporta la versione 1.20.0 di Apache Flink. Questa sezione presenta le nuove funzionalità e le modifiche principali introdotte con il supporto di Managed Service for Apache Flink di Apache Flink 1.20.0. Apache Flink 1.20 dovrebbe essere l'ultima versione 1.x e una versione di Flink con supporto a lungo termine (LTS). Per ulteriori informazioni, consulta [FLIP-458: Long-Term Support for the Final Release of Apache Flink 1.x Line](#).

Note

Se utilizzi una versione precedente supportata di Apache Flink e desideri aggiornare le tue applicazioni esistenti ad Apache Flink 1.20.0, puoi farlo utilizzando gli aggiornamenti della versione di Apache Flink in loco. Per ulteriori informazioni, consulta [Usa gli aggiornamenti di versione sul posto per Apache Flink](#). Con gli aggiornamenti di versione in loco, mantieni la tracciabilità delle applicazioni su un singolo ARN tra le versioni di Apache Flink, tra cui istantanee, log, metriche, tag, configurazioni Flink e altro ancora.

Funzionalità supportate

Apache Flink 1.20.0 introduce miglioramenti nell'SQL, nella dashboard e nella dashboard di Flink. APIs DataStream APIs

Funzionalità supportate e documentazione correlata

Funzionalità supportate	Descrizione	Riferimento alla documentazione di Apache Flink
Aggiungi la clausola DISTRIBUTED BY	Molti motori SQL espongono i concetti di Partitioning Bucketing , o. Clustering Flink 1.20 introduce il concetto di to Flink. Bucketing	FLIP-376: aggiungi la clausola DISTRIBUTED BY
DataStream API: Supporta l'elaborazione completa delle partizioni	Flink 1.20 introduce il supporto integrato per le aggregazioni su flussi senza chiave	FLIP-380: Supporta l'elaborazione completa delle partizioni su Non-keyed DataStream

Funzionalità supportate	Descrizione	Riferimento alla documentazione di Apache Flink
Mostra il punteggio di distorsione dei dati su Flink Dashboard	tramite l'API. <code>FullPartitionWindow</code> La dashboard di Flink 1.20 ora mostra le informazioni sulla distorsione dei dati. Ogni operatore nell'interfaccia utente del grafico dei lavori di Flink mostra un punteggio aggiuntivo di distorsione dei dati.	FLIP-418: mostra il punteggio di distorsione dei dati sulla dashboard di Flink

[Per la documentazione sulla versione di Apache Flink 1.20.0, consultate Apache Flink Documentation v1.20.0.](#) [Per le note di rilascio di Flink 1.20, consultate Release notes - Flink 1.20](#)

Componenti

Componenti Flink 1.20

Componente	Versione
Java	11 (consigliata)
Python	3.11
Kinesis Data Analytics Flink Runtime () aws-kinesisanalytics-runtime	1.2.0
Connectors (Connettori)	Per informazioni sui connettori disponibili, consulta Connettori Apache Flink.
Apache Beam (solo applicazioni Beam)	Non esiste Apache Flink Runner compatibile per Flink 1.20. Per ulteriori informazioni, consulta Compatibilità delle versioni di Flink.

Problemi noti

Apache Beam

Al momento non esiste Apache Flink Runner compatibile per Flink 1.20 in Apache Beam. [Per ulteriori informazioni, consulta Compatibilità delle versioni di Flink.](#)

Servizio gestito Amazon per Apache Flink Studio

Amazon Managed Service for Apache Flink Studio utilizza i notebook Apache Zeppelin per fornire un'esperienza di sviluppo a interfaccia singola per lo sviluppo, il debug del codice e l'esecuzione di applicazioni di elaborazione di flussi Apache Flink. È necessario un aggiornamento a Flink Interpreter di Zeppelin per abilitare il supporto di Flink 1.20. Questo lavoro è programmato con la comunità Zeppelin. Aggiungeremo queste note quando il lavoro sarà completo. Puoi continuare a utilizzare Flink 1.15 con Amazon Managed Service per Apache Flink Studio. Per ulteriori informazioni, consulta [Creazione](#) di un notebook Studio.

Correzioni di bug con backport

Amazon Managed Service for Apache Flink supporta le correzioni della community Flink per problemi critici. Di seguito è riportato un elenco di correzioni di bug di cui abbiamo eseguito il backport:

Correzioni di bug ripristinate

Collegamento Apache Flink JIRA	Descrizione
FLINK-35886	Questa correzione risolve un problema che causa una contabilizzazione errata dei timeout di inattività delle filigrane quando un'attività secondaria viene sottoposta a contropressione/blocco.

Servizio gestito Amazon per Apache Flink 1.19

Managed Service for Apache Flink ora supporta la versione 1.19.1 di Apache Flink. Questa sezione presenta le nuove funzionalità e le modifiche principali introdotte con il supporto di Managed Service for Apache Flink di Apache Flink 1.19.1.

Note

Se utilizzi una versione precedente supportata di Apache Flink e desideri aggiornare le tue applicazioni esistenti ad Apache Flink 1.19.1, puoi farlo utilizzando gli aggiornamenti della versione di Apache Flink in loco. Per ulteriori informazioni, consulta [Usa gli aggiornamenti di versione sul posto per Apache Flink](#). Con gli aggiornamenti di versione in loco, mantieni la tracciabilità delle applicazioni su un singolo ARN tra le versioni di Apache Flink, tra cui istantanee, log, metriche, tag, configurazioni Flink e altro ancora.

Funzionalità supportate

Apache Flink 1.19.1 introduce miglioramenti nell'API SQL, come parametri denominati, parallelismo dei sorgenti personalizzato e stati diversi per vari operatori Flink. TTLs

Funzionalità supportate e documentazione correlata

Funzionalità supportate	Descrizione	Riferimento alla documentazione di Apache Flink
API SQL: Supporta la configurazione di stati diversi TTLs utilizzando SQL Hint	Gli utenti possono ora configurare lo stato TTL sui join regolari in streaming e sugli aggregati di gruppo.	FLIP-373: configurazione di stati diversi utilizzando SQL Hint TTLs
API SQL: Supporta parametri denominati per funzioni e procedure di chiamata	Gli utenti possono ora utilizzare e parametri denominati nelle funzioni, anziché fare affidamento sull'ordine dei parametri.	FLIP-378: Supporta parametri denominati per funzioni e procedure di chiamata
API SQL: impostazione del parallelismo per le sorgenti SQL	Gli utenti possono ora specificare il parallelismo per le sorgenti SQL.	FLIP-367: Supporta l'impostazione del parallelismo per sorgenti Table/SQL
API SQL: Finestra di sessione di supporto TVF	Gli utenti possono ora utilizzare e Table-Valued Functions della finestra di sessione.	FLINK-24024: sessione di supporto Window TVF

Funzionalità supportate	Descrizione	Riferimento alla documentazione di Apache Flink
API SQL: l'aggregazione Window TVF supporta gli input del registro delle modifiche	Gli utenti possono ora eseguire l'aggregazione delle finestre sugli input del changelog.	FLINK-20281: L'aggregazione delle finestre supporta l'input del flusso di log delle modifiche
Supporta Python 3.11	Flink ora supporta Python 3.11, che è il 10-60% più veloce rispetto a Python 3.10. Per ulteriori informazioni, consulta What's New in Python 3.11 .	FLINK-33030: aggiungi il supporto per Python 3.11
Fornisci metriche TwoPhaseCommitting per il sink	Gli utenti possono visualizzare le statistiche sullo stato dei committenti nei sink di committenza a due fasi.	FLIP-371: Fornisce un contesto di inizializzazione per la creazione di Committer in TwoPhaseCommittingSink
Trace Reporters per il riavvio del lavoro e il checkpoint	Gli utenti possono ora monitorare le tracce relative alla durata dei checkpoint e alle tendenze di recupero. In Amazon Managed Service for Apache Flink, abilitiamo i trace reporter SLF4j per impostazione predefinita, in modo che gli utenti possano monitorare checkpoint e tracce di lavoro tramite i log delle applicazioni. CloudWatch	FLIP-384: introducilo e usalo per creare tracce di checkpoint e ripristino TraceReporter

 Note

[Puoi attivare le seguenti funzionalità inviando una richiesta di assistenza:](#)

Funzionalità di attivazione e documentazione correlata

Funzionalità di attivazione	Descrizione	Riferimento alla documentazione di Apache Flink
Supporta l'utilizzo di intervalli di checkpoint più ampi quando la fonte sta elaborando il backlog	Si tratta di una funzionalità opzionale, in quanto gli utenti devono ottimizzare la configurazione in base alle proprie esigenze lavorative specifiche.	FLIP-309: Supporta l'utilizzo di intervalli di checkpoint più ampi quando l'origine sta elaborando il backlog
Reindirizza System.out e System.err nei log Java	Questa è una funzionalità opzionale. In Amazon Managed Service for Apache Flink, il comportamento predefinito consiste nell'ignorare l'output di System.out e System.err perché la migliore pratica in produzione consiste nell'utilizzare il logger Java nativo.	FLIP-390: Support System out ed err per essere reindirizzato a LOG o eliminato

[Per la documentazione sulla versione di Apache Flink 1.19.1, consultate Apache Flink Documentation v1.19.1.](#)

Modifiche ad Amazon Managed Service per Apache Flink 1.19.1

La registrazione di Trace Reporter è abilitata per impostazione predefinita

Apache Flink 1.19.1 ha introdotto le tracce di checkpoint e ripristino, permettendo agli utenti di eseguire meglio il debug dei checkpoint e dei problemi di ripristino dei job. In Amazon Managed Service for Apache Flink, queste tracce vengono registrate nel flusso di CloudWatch log, permettendo agli utenti di suddividere il tempo impiegato per l'inizializzazione dei processi e registrare la dimensione storica dei checkpoint.

La strategia di riavvio predefinita è ora exponential-delay

In Apache Flink 1.19.1, sono stati apportati miglioramenti significativi alla strategia di riavvio con ritardo esponenziale. In Amazon Managed Service for Apache Flink a partire da Flink 1.19.1 in poi, i job Flink utilizzano la strategia di riavvio con ritardo esponenziale per impostazione predefinita. Ciò significa che i lavori degli utenti verranno ripristinati più rapidamente in seguito a errori temporanei, ma non sovraccaricheranno i sistemi esterni se i riavvii dei processi persistono.

Correzioni di bug con backport

Amazon Managed Service for Apache Flink supporta le correzioni della community Flink per problemi critici. Ciò significa che il runtime è diverso dalla versione Apache Flink 1.19.1. Di seguito è riportato un elenco di correzioni di bug di cui abbiamo eseguito il backport:

Correzioni di bug ripristinate

Collegamento Apache Flink JIRA	Descrizione
FLINK-35531	Questa correzione risolve la regressione delle prestazioni introdotta nella versione 1.17.0 che causa scritture più lente su HDFS.
FLINK-35157	Questa correzione risolve il problema dei lavori Flink bloccati quando le sorgenti con allineamento della filigrana incontrano attività secondarie completate.
FLINK-34252	Questa correzione risolve il problema della generazione di filigrane che causa uno stato errato della filigrana IDLE.
FLINK-34252	Questa correzione risolve la regressione delle prestazioni durante la generazione di filigrane riducendo le chiamate di sistema.
FLINK-33936	Questa correzione risolve il problema dei record duplicati durante l'aggregazione in mini-batch su Table API.
FLINK-35498	Questa correzione risolve il problema dei conflitti tra i nomi degli argomenti durante la

Collegamento Apache Flink JIRA	Descrizione
	definizione di parametri denominati nell'API Table. UDFs
FLINK-33192	Questa correzione risolve il problema di una perdita di memoria di stato negli operatori di Windows a causa di una pulizia impropria del timer.
FLINK-35069	Questa correzione risolve il problema quando un job Flink si blocca attivando un timer alla fine di una finestra.
FLINK-35832	Questa correzione risolve il problema quando IFNULL restituisce risultati errati.
FLINK-35886	Questa correzione risolve il problema quando le attività in contropressione vengono considerate inattive.

Componenti

Componente	Versione
Java	11 (consigliata)
Python	3.11
Kinesis Data Analytics Flink Runtime () aws-kinesisanalytics-runtime	1.2.0
Connectors (Connettori)	Per informazioni sui connettori disponibili, consulta Connettori Apache Flink .
Apache Beam (solo applicazioni Beam)	A partire dalla versione 2.61.0. Per ulteriori informazioni, vedere Compatibilità della versione di Flink .

Problemi noti

Servizio gestito Amazon per Apache Flink Studio

Studio utilizza i notebook Apache Zeppelin per offrire un'esperienza di sviluppo a interfaccia singola per lo sviluppo, il debug del codice e l'esecuzione di applicazioni di elaborazione di flussi Apache Flink. È necessario un aggiornamento a Flink Interpreter di Zeppelin per abilitare il supporto di Flink 1.19. Questo lavoro è programmato con la community di Zeppelin e aggiorneremo queste note quando sarà completo. Puoi continuare a utilizzare Flink 1.15 con Amazon Managed Service per Apache Flink Studio. Per ulteriori informazioni, consulta [Creazione](#) di un notebook Studio.

Servizio gestito Amazon per Apache Flink 1.18

Managed Service for Apache Flink ora supporta la versione 1.18.1 di Apache Flink. Scopri le nuove funzionalità e le modifiche principali introdotte con il supporto di Managed Service for Apache Flink di Apache Flink 1.18.1.

Note

Se utilizzi una versione precedente supportata di Apache Flink e desideri aggiornare le tue applicazioni esistenti ad Apache Flink 1.18.1, puoi farlo utilizzando gli aggiornamenti di versione di Apache Flink in loco. Con gli aggiornamenti di versione in loco, mantieni la tracciabilità delle applicazioni su un singolo ARN tra le versioni di Apache Flink, tra cui istantanee, log, metriche, tag, configurazioni Flink e altro ancora. È RUNNING READY possibile utilizzare questa funzionalità in qualsiasi stato. Per ulteriori informazioni, consulta [Usa gli aggiornamenti di versione sul posto per Apache Flink](#).

Funzionalità supportate con riferimenti alla documentazione di Apache Flink

Caratteristiche supportate	Descrizione	Riferimento alla documentazione di Apache Flink
Connettore Opensearch	Questo connettore include un lavandino che fornisce at-least-once garanzie.	github: Connettore Opensearch h

Caratteristiche supportate	Descrizione	Riferimento alla documentazione di Apache Flink
Connettore Amazon DynamoDB	Questo connettore include un lavandino che fornisce at-least-once garanzie.	Amazon DynamoDB Sink
Connettore MongoDB	Questo connettore include una fonte e un sink che forniscono at-least-once garanzie.	Connettore MongoDB
Disaccoppia Hive con il pianificatore Flink	Puoi usare direttamente il dialetto Hive senza dover cambiare ulteriormente il JAR.	FLINK-26603: disaccoppia Hive con il pianificatore Flink
Disabilita WAL in Rocks DBWrite BatchWrapper per impostazione predefinita	Ciò fornisce tempi di ripristino più rapidi.	FLINK-32326: disabilita WAL in Rocks per impostazione predefinita DBWrite BatchWrapper
Migliora le prestazioni di aggregazione delle filigrane abilitando l'allineamento delle filigrane	Migliora le prestazioni di aggregazione delle filigrane abilitando l'allineamento delle filigrane e aggiunge il relativo benchmark.	FLINK-32524: prestazioni di aggregazione Watermark
Prepara l'allineamento delle filigrane per l'uso in produzione	Elimina il rischio di sovraccarico di lavori di grandi dimensioni JobManager	FLINK-32548: Prepara l'allineamento della filigrana
Configurabile per RateLimitingStrategy Async Sink	RateLimitingStrategy consente di configurare la decisione su cosa scalare, quando scalare e quanto scalare.	FLIP-242: introduce il configurabile RateLimitingStrategy per Async Sink

Caratteristiche supportate	Descrizione	Riferimento alla documentazione di Apache Flink
Recupera in blocco le statistiche di tabelle e colonne	Prestazioni di interrogazione migliorate.	FLIP-247: recupero in blocco delle statistiche di tabelle e colonne per determinate partizioni

[Per la documentazione sulla versione di Apache Flink 1.18.1, consultate l'annuncio di rilascio di Apache Flink 1.18.1.](#)

Modifiche ad Amazon Managed Service per Apache Flink con Apache Flink 1.18

Akka è stato sostituito con Pekko

Apache Flink ha sostituito Akka con Pekko in Apache Flink 1.18. Questa modifica è completamente supportata in Managed Service for Apache Flink di Apache Flink 1.18.1 e versioni successive. Non è necessario modificare le applicazioni a seguito di questa modifica. Per ulteriori informazioni, vedere [FLINK-32468: Replace Akka by Pekko](#).

Supporta l'esecuzione di Runtime in modalità Thread

Questa modifica di Apache Flink introduce una nuova modalità di esecuzione per il framework Pyflink Runtime, Process Mode. Process Mode ora può eseguire funzioni Python definite dall'utente nello stesso thread anziché in un processo separato.

Correzioni di bug con backport

Amazon Managed Service for Apache Flink supporta le correzioni della community Flink per problemi critici. Ciò significa che il runtime è diverso dalla versione Apache Flink 1.18.1. Di seguito è riportato un elenco di correzioni di bug di cui abbiamo eseguito il backport:

Correzioni di bug ripristinate

Collegamento Apache Flink JIRA	Descrizione
FLINK-33863	Questa correzione risolve il problema quando un ripristino dello stato non riesce per le istantanee compresse.
FLINK-34063	Questa correzione risolve il problema che si verifica quando gli operatori di origine perdono le suddivisioni quando è abilitata la compressione delle istantanee. Apache Flink offre una compressione opzionale (impostazione predefinita: disattivata) per tutti i checkpoint e i savepoint. Apache Flink ha identificato un bug in Flink 1.18.1 in cui lo stato dell'operatore non poteva essere ripristinato correttamente quando era abilitata la compressione delle istantanee. Ciò potrebbe comportare la perdita di dati o l'impossibilità di eseguire il ripristino dal checkpoint.
FLINK-35069	Questa correzione risolve il problema quando un job Flink si blocca attivando un timer alla fine di una finestra.
FLINK-35097	Questa correzione risolve il problema dei record duplicati in un connettore Table API Filesystem in formato raw.
FLINK-34379	Questa correzione risolve il problema dell'attivazione del filtraggio dinamico delle tabelle OutOfMemoryError .
FLINK-28693	Questa correzione risolve il problema dell'impossibilità dell'API Table di generare un grafico se la filigrana ha un'espressione ColumnBy.

Collegamento Apache Flink JIRA	Descrizione
FLINK-35217	Questa correzione risolve il problema di un checkpoint danneggiato durante una specifica modalità di errore del processo Flink.

Componenti

Componente	Versione
Java	11 (consigliata)
Scala	A partire dalla versione 1.15, Flink è indipendente da Scala. Per riferimento, MSF Flink 1.18 è stato verificato rispetto a Scala 3.3 (LTS).
Servizio gestito per Apache Flink Flink Runtime (<code>aws-kinesisanalytics-runtime</code>)	1.2.0
AWS Kinesis Connector (flink-connector-kinesis) [Fonte]	4.2.0-1,18
AWS Connettore Kinesis (flink-connector-kinesis) [Sink]	4.2,0-1,18
Apache Beam (solo applicazioni Beam)	A partire dalla versione 2.57.0. Per ulteriori informazioni, vedere Compatibilità delle versioni di Flink .

Problemi noti

Servizio gestito Amazon per Apache Flink Studio

Studio utilizza i notebook Apache Zeppelin per offrire un'esperienza di sviluppo a interfaccia singola per lo sviluppo, il debug del codice e l'esecuzione di applicazioni di elaborazione di flussi Apache Flink. È necessario un aggiornamento a Flink Interpreter di Zeppelin per abilitare il supporto di Flink 1.18. Questo lavoro è programmato con la community di Zeppelin e aggiorneremo queste note

quando sarà completo. Puoi continuare a utilizzare Flink 1.15 con Amazon Managed Service per Apache Flink Studio. Per ulteriori informazioni, consulta [Creazione](#) di un notebook Studio.

Inattività errata della filigrana quando la sottoattività viene sottoposta a contropressione

Esiste un problema noto nella generazione di filigrane quando una sottoattività viene sottoposta a contropressione, che è stato risolto a partire da Flink 1.19 e versioni successive. Ciò può manifestarsi come un picco nel numero di record in ritardo quando un grafico di lavoro Flink viene sottoposto a contropressione. Ti consigliamo di eseguire l'aggiornamento alla versione più recente di Flink per inserire questa correzione. Per ulteriori informazioni, consulta [Contabilità errata del timeout dell'inattività con filigrana quando la sottoattività viene contropressionata/bloccata](#).

Servizio gestito Amazon per Apache Flink 1.15

Managed Service per Apache Flink supporta le seguenti nuove funzionalità in Apache 1.15.2:

Funzionalità	Descrizione	Riferimento Apache FLIP
Async Sink	Un framework AWS contribuito per la creazione di destinazioni asincrone che consente agli sviluppatori di creare AWS connettori personalizzati con meno della metà dello sforzo precedente. Per ulteriori informazioni, consulta Generic Asynchronous Base Sink .	FLIP-171: Async Sink .
Kinesis Data Firehose Sink	AWS ha contribuito con un nuovo Amazon Kinesis Firehose Sink utilizzando il framework Async.	Amazon Kinesis Data Firehose Sink .
Stop with Savepoint	Stop with Savepoint garantisce un funzionamento pulito ininterrotto e, soprattutto, supporta la semantica exactly-	FLIP-34: Terminate/Suspend Job with Savepoint .

Funzionalità	Descrizione	Riferimento Apache FLIP
	once per i clienti che decidono di usarlo.	
Scala Decoupling	Gli utenti ora possono sfruttare l'API Java di qualsiasi versione di Scala, inclusa Scala 3. I clienti dovranno raggruppare la libreria standard Scala che hanno scelto nelle loro applicazioni Scala.	FLIP-28: Obiettivo a lungo termine: rendere flink-table privo di Scala.
Scala	Cfr. Scala Decoupling qui sopra	FLIP-28: Obiettivo a lungo termine: rendere flink-table privo di Scala.
Metriche unificate per i connettori.	Flink ha definito metriche standard per processi, attività e operatori. Il servizio gestito per Apache Flink continuerà a supportare le metriche sink e origine, e nella versione 1.15 verrà introdotto <code>numRestarts</code> in parallelo con <code>fullRestarts</code> per Availability Metrics.	FLIP-33: Standardize Connector Metrics e FLIP-179: Expose Standardized Operator Metrics.
Checkpoint delle attività completate	Questa funzionalità è abilitata di default in Flink 1.15 e consente di continuare a eseguire i checkpoint anche se alcune parti del grafico di processo hanno terminato l'elaborazione di tutti i dati, cosa che potrebbe accadere se contiene origini (batch) associate.	FLIP-147: Support Checkpoints After Tasks Finished.

Modifiche al servizio gestito da Amazon per Apache Flink con Apache Flink 1.15

Notebook Studio

Il servizio gestito per Apache Flink Studio ora supporta Apache Flink 1.15. Il servizio gestito per Apache Flink Studio utilizza i notebook Apache Zeppelin per offrire un'unica interfaccia per lo sviluppo, il debug del codice e l'esecuzione di applicazioni di elaborazione di flussi Apache Flink. Puoi saperne di più sul servizio gestito per Apache Flink Studio e su come iniziare qui: [Usa un notebook Studio con Managed Service per Apache Flink](#).

Connettore EFO

Quando esegui l'aggiornamento del servizio gestito per Apache Flink versione 1.15, assicurati di utilizzare il connettore EFO più recente, ossia qualsiasi versione 1.15.3 o successiva. Per ulteriori informazioni sul motivo, consulta [FLINK-29324](#).

Scala Decoupling

A partire da Flink 1.15.2, dovrai raggruppare la libreria standard Scala che hai scelto nelle tue applicazioni Scala.

Kinesis Data Firehose Sink

Quando esegui l'aggiornamento del servizio gestito per Apache Flink versione 1.15, assicurati di utilizzare il [Amazon Kinesis Data Firehose Sink](#) più recente.

Connettori Kafka

Quando esegui l'aggiornamento ad Amazon Managed Service for Apache Flink per Apache Flink versione 1.15, assicurati di utilizzare il connettore Kafka più recente. APIs Apache Flink è obsoleto e These for the Kafka sink non può eseguire il commit su Kafka for Flink 1.15 [FlinkKafkaConsumer](#). [FlinkKafkaProducer](#) APIs Assicurati di utilizzare [KafkaSource](#) e [KafkaSink](#).

Componenti

Componente	Versione
Java	11 (consigliata)
Scala	2.12

Componente	Versione
Servizio gestito per Apache Flink Flink Runtime (<code>aws-kinesisanalytics-runtime</code>)	1.2.0
AWS Connettore Kinesis (<code>flink-connector-kinesis</code>)	1.15.4
Apache Beam (solo applicazioni Beam)	2.33.0, con la versione Jackson 2.12.2

Problemi noti

Kafka Commit sul checkpoint fallisce ripetutamente dopo il riavvio del broker

Esiste un problema noto di Apache Flink open source con il connettore Apache Kafka nella versione 1.15 di Flink, causato da un bug critico di Kafka Client open source in Kafka Client 2.8.1. [Per ulteriori informazioni, vedi Kafka Commit on checkpointing fallisce ripetutamente dopo il riavvio del broker e non è in grado di ripristinare la connessione al coordinatore del gruppo dopo un'eccezione. KafkaConsumer commitOffsetAsync](#)

Per evitare questo problema, ti consigliamo di utilizzare Apache Flink 1.18 o versione successiva in Amazon Managed Service for Apache Flink.

Informazioni sulle versioni precedenti del servizio gestito per Apache Flink

Note

Le versioni 1.6, 1.8 e 1.11 di Apache Flink non sono supportate dalla community di Apache Flink da oltre tre anni. Ora prevediamo di interrompere il supporto per queste versioni in Amazon del servizio gestito per Apache Flink. A partire dal 5 novembre 2024, non potrai creare nuove applicazioni per queste versioni di Flink. Puoi continuare a eseguire le applicazioni esistenti in questo momento.

Per tutte le regioni ad eccezione delle regioni della Cina e AWS GovCloud (US) Regions, dal 24 febbraio 2025, non sarà più possibile creare, avviare o eseguire applicazioni utilizzando queste versioni di Apache Flink in Amazon Managed Service for Apache Flink.

Per le regioni della Cina e il AWS GovCloud (US) Regions19 marzo 2025, non sarà più possibile creare, avviare o eseguire applicazioni utilizzando queste versioni di Apache Flink in Amazon Managed Service for Apache Flink.

Puoi aggiornare le tue applicazioni in modo statico utilizzando la funzionalità di aggiornamento immediato delle versioni di Managed Service for Apache Flink. Per ulteriori informazioni, consulta [Usa gli aggiornamenti di versione sul posto per Apache Flink](#).

Note

La versione 1.13 di Apache Flink non è supportata dalla community Apache Flink da oltre tre anni. Ora prevediamo di terminare il supporto per questa versione in Amazon Managed Service for Apache Flink il 16 ottobre 2025. Dopo questa data, non sarà più possibile creare, avviare o eseguire applicazioni utilizzando Apache Flink versione 1.13 in Amazon Managed Service for Apache Flink.

Puoi aggiornare le tue applicazioni in modo statico utilizzando la funzionalità di aggiornamento delle versioni in loco in Managed Service for Apache Flink. Per ulteriori informazioni, consulta [Usa gli aggiornamenti di versione sul posto per Apache Flink](#).

La versione 1.15.2 è supportata da Managed Service for Apache Flink, ma non è più supportata dalla community Apache Flink.

Questo argomento contiene le sezioni seguenti:

- [Utilizzo del connettore di flussi Apache Flink Kinesis con versioni precedenti di Apache Flink](#)
- [Creazione di applicazioni con Apache Flink 1.8.2](#)
- [Creazione di applicazioni con Apache Flink 1.6.2](#)
- [Aggiornamento delle applicazioni](#)
- [Connettori disponibili in Apache Flink 1.6.2 e 1.8.2](#)
- [Nozioni di base: Flink 1.13.2](#)
- [Nozioni di base: Flink 1.11.1 - obsoleto](#)
- [Nozioni di base: Flink 1.8.2 - obsoleto](#)
- [Nozioni di base: Flink 1.6.2 - deprecazione](#)
- [Esempi di versioni precedenti \(precedenti\) del servizio gestito da Amazon per Apache Flink](#)

Utilizzo del connettore di flussi Apache Flink Kinesis con versioni precedenti di Apache Flink

Il connettore di flussi Apache Flink Kinesis non era incluso in Apache Flink prima della versione 1.11. Affinché l'applicazione possa utilizzare il connettore Apache Flink Kinesis con le versioni precedenti di Apache Flink, è necessario scaricare, compilare e installare la versione di Apache Flink utilizzata dall'applicazione. Questo connettore serve per utilizzare i dati di un flusso Kinesis utilizzato come origine dell'applicazione o per scrivere dati su un flusso Kinesis utilizzato per l'output dell'applicazione.

Note

Assicurati di creare il connettore con [KPL versione 0.14.0](#) o successiva.

Per scaricare e installare il codice di origine di Apache Flink 1.8.2, procedi come segue:

1. Assicurati di avere installato [Apache Maven](#) e che la variabile di ambiente JAVA_HOME punti a un JDK anziché a un JRE. Puoi testare la tua installazione di Apache Maven con il comando seguente:

```
mvn -version
```

2. Scarica il codice di origine di Apache Flink 1.8.2:

```
wget https://archive.apache.org/dist/flink/flink-1.8.2/flink-1.8.2-src.tgz
```

3. Decomprimi il codice di origine di Apache Flink:

```
tar -xvf flink-1.8.2-src.tgz
```

4. Passa alla directory del codice di origine di Apache Flink:

```
cd flink-1.8.2
```

5. Compila e installa Apache Flink:

```
mvn clean install -Pinclude-kinesis -DskipTests
```

 Note

Se stai compilando Flink su Microsoft Windows, devi aggiungere il parametro `-Drat.skip=true`.

Creazione di applicazioni con Apache Flink 1.8.2

Questa sezione contiene informazioni sui componenti utilizzati per creare applicazioni del servizio gestito per Apache Flink compatibili con Apache Flink 1.8.2.

Utilizza le seguenti versioni dei componenti per il servizio gestito per Apache Flink:

Componente	Versione
Java	1.8 (consigliata)
Apache Flink	1.8.2
Servizio gestito per Apache Flink per il runtime di Flink () <code>aws-kinesisanalytics-runtime</code>	1.0.1
Connettori Flink del servizio gestito per Apache Flink () <code>aws-kinesisanalytics-flink</code>	1.0.1
Apache Maven	3.1

Per compilare un'applicazione utilizzando Apache Flink 1.8.2, esegui Maven con il seguente parametro:

```
mvn package -Dflink.version=1.8.2
```

Per un esempio di file `pom.xml` per un'applicazione del servizio gestito per Apache Flink che utilizza Apache Flink 1.8.2, consulta l'[applicazione introduttiva del servizio gestito per Apache Flink 1.8.2](#).

Per informazioni su come creare e utilizzare il codice applicativo per un'applicazione del servizio gestito per Apache Flink, consulta [Creazione di un'applicazione](#).

Creazione di applicazioni con Apache Flink 1.6.2

Questa sezione contiene informazioni sui componenti utilizzati per creare applicazioni del servizio gestito per Apache Flink compatibili con Apache Flink 1.6.2.

Utilizza le seguenti versioni dei componenti per il servizio gestito per Apache Flink:

Componente	Versione
Java	1.8 (consigliata)
AWS SDK Java	1.11.379
Apache Flink	1.6.2
Servizio gestito per Apache Flink per il runtime di Flink () <code>aws-kinesisanalytics-runtime</code>	1.0.1
Connettori Flink del servizio gestito per Apache Flink () <code>aws-kinesisanalytics-flink</code>	1.0.1
Apache Maven	3.1
Apache Beam	Non supportato con Apache Flink 1.6.2.

Note

Quando utilizzi il servizio gestito per il runtime di Apache Flink versione 1.0.1, devi specificare la versione di Apache Flink nel file `pom.xml` anziché utilizzare il parametro `-Dflink.version` durante la compilazione del codice dell'applicazione.

Per un esempio di file `pom.xml` per un'applicazione del servizio gestito per Apache Flink che utilizza Apache Flink 1.6.2, consulta l'[applicazione introduttiva del servizio gestito per Apache Flink 1.6.2](#).

Per informazioni su come creare e utilizzare il codice applicativo per un'applicazione del servizio gestito per Apache Flink, consulta [Creazione di un'applicazione](#).

Aggiornamento delle applicazioni

Per aggiornare la versione Apache Flink di un'applicazione Amazon Managed Service for Apache Flink, utilizza la funzionalità di aggiornamento della versione Apache Flink sul posto utilizzando, SDK o. AWS CLI AWS AWS CloudFormation AWS Management Console Per ulteriori informazioni, consulta [Usa gli aggiornamenti di versione sul posto per Apache Flink](#).

Puoi utilizzare questa funzionalità con qualsiasi applicazione esistente che utilizza con Amazon del servizio gestito da Amazon per Apache Flink nel READY nostro stato. RUNNING

Connettori disponibili in Apache Flink 1.6.2 e 1.8.2

Il framework Apache Flink contiene connettori per l'accesso ai dati da una varietà di origini.

- Per informazioni sui connettori disponibili nel framework Apache Flink 1.6.2, consulta [Connettori \(1.6.2\)](#) nella [documentazione di Apache Flink \(1.6.2\)](#).
- Per informazioni sui connettori disponibili nel framework Apache Flink 1.8.2, consulta [Connettori \(1.8.2\)](#) nella [documentazione di Apache Flink \(1.8.2\)](#).

Nozioni di base: Flink 1.13.2

Questa sezione presenta i concetti fondamentali del servizio gestito per Apache Flink e dell'API. DataStream Descrive le opzioni disponibili per la creazione e il test delle applicazioni. Fornisce inoltre istruzioni per l'installazione degli strumenti necessari per completare i tutorial di questa guida e creare la tua prima applicazione.

Argomenti

- [Componenti dell'applicazione del servizio gestito per Apache Flink](#)
- [Prerequisiti per il completamento degli esercizi](#)
- [Fase 1: Impostazione di un AWS account e creazione di un utente amministratore](#)
- [Approfondimenti](#)
- [Fase 2: configurazione dell' AWS Command Line Interface \(AWS CLI\)](#)
- [Fase 3: creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink](#)
- [Passaggio 4: eliminare le AWS risorse](#)
- [Fase 5: fasi successive](#)

Componenti dell'applicazione del servizio gestito per Apache Flink

Per elaborare i dati, l'applicazione del servizio gestito per Apache Flink utilizza un'applicazione Java/ Apache Maven o Scala che elabora l'input e produce l'output utilizzando il runtime di Apache Flink.

L'applicazione del servizio gestito per Apache Flink include i seguenti componenti:

- **Proprietà di runtime:** è possibile utilizzare le proprietà di runtime per configurare l'applicazione senza ricompilare il codice dell'applicazione.
- **Origine:** l'applicazione consuma i dati utilizzando un'origine. Un connettore di origine legge i dati da un flusso di dati Kinesis, da un bucket Amazon S3, ecc. Per ulteriori informazioni, consulta [Aggiungi sorgenti di dati in streaming](#).
- **Operatori:** l'applicazione elabora i dati utilizzando uno o più operatori. Un operatore può trasformare, arricchire o aggregare i dati. Per ulteriori informazioni, consulta [Operatori](#).
- **Sink:** l'applicazione produce dati verso origini esterne utilizzando i sink. Un connettore sink scrive i dati su un flusso di dati Kinesis, un flusso Firehose, un bucket Amazon S3, ecc. Per ulteriori informazioni, consulta [Scrivi dati utilizzando i lavandini](#).

Dopo aver creato, compilato e compresso il codice dell'applicazione, caricherai il pacchetto di codice in un bucket Amazon Simple Storage Service (Amazon S3). Potrai quindi creare un'applicazione del servizio gestito per Apache Flink. Dovrai inserire la posizione del pacchetto di codice, un flusso di dati Kinesis come origine dati di streaming e in genere una posizione di streaming o di file che riceva i dati elaborati dall'applicazione.

Prerequisiti per il completamento degli esercizi

Per completare le fasi in questa guida, è richiesto quanto segue:

- [Java Development Kit \(JDK\) versione 11](#). Imposta la variabile di ambiente JAVA_HOME in modo che punti alla posizione di installazione di JDK.
- Ti consigliamo di utilizzare un ambiente di sviluppo (ad esempio [Eclipse Java Neon](#) o [IntelliJ IDEA](#)) per sviluppare e compilare l'applicazione.
- [Client Git](#). Installa il client Git se non lo hai già fatto.
- [Apache Maven Compiler Plugin](#). Maven deve trovarsi nel percorso di lavoro. Per testare l'installazione Apache Maven, immetti quanto segue:

```
$ mvn -version
```

Per iniziare, vai alla pagina [Configura un AWS account e crea un utente amministratore](#).

Fase 1: Impostazione di un AWS account e creazione di un utente amministratore

Registrazione per un account Account AWS

Se non si dispone di un Account AWS, completare la procedura seguente per crearne uno.

Per registrarti a un Account AWS

1. Apri la <https://portal.aws.amazon.com/billing/registrazione>.
2. Segui le istruzioni online.

Nel corso della procedura di registrazione riceverai una telefonata, durante la quale sarà necessario inserire un codice di verifica attraverso la tastiera del telefono.

Quando ti iscrivi a un Account AWS, Utente root dell'account AWS viene creato un. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente e utilizza solo l'utente root per eseguire [attività che richiedono l'accesso di un utente root](#).

AWS Al termine del processo di registrazione, riceverai un'e-mail di conferma da. Puoi visualizzare l'attività corrente del tuo account e gestire l'account in qualsiasi momento accedendo a <https://aws.amazon.com/e> scegliendo My Account (Il mio account).

Crea un utente con accesso amministrativo

Dopo aver effettuato la registrazione di un Account AWS, proteggi Utente root dell'account AWS AWS IAM Identity Center, abilita e crea un utente amministratore in modo da non utilizzare l'utente root per le attività quotidiane.

Proteggi i tuoi Utente root dell'account AWS

1. Accedi [AWS Management Console](#) come proprietario dell'account scegliendo Utente root e inserendo il tuo indirizzo Account AWS email. Nella pagina successiva, inserisci la password.

Per informazioni sull'accesso utilizzando un utente root, consulta la pagina [Signing in as the root user](#) della Guida per l'utente di Accedi ad AWS .

2. Abilita l'autenticazione a più fattori (MFA) per l'utente root.

Per ricevere istruzioni, consulta [Abilitare un dispositivo MFA virtuale per l'utente Account AWS root della \(console\)](#) nella Guida per l'utente IAM.

Crea un utente con accesso amministrativo

1. Abilita Centro identità IAM.

Per istruzioni, consulta [Abilitazione di AWS IAM Identity Center](#) nella Guida per l'utente di AWS IAM Identity Center .

2. In IAM Identity Center, assegna l'accesso amministrativo a un utente.

Per un tutorial sull'utilizzo di IAM Identity Center directory come fonte di identità, consulta [Configurare l'accesso utente con l'impostazione predefinita IAM Identity Center directory](#) nella Guida per l'AWS IAM Identity Center utente.

Accesso come utente amministratore

- Per accedere con l'utente IAM Identity Center, utilizza l'URL di accesso che è stato inviato al tuo indirizzo e-mail quando hai creato l'utente IAM Identity Center.

Per assistenza nell'accesso mediante un utente IAM Identity Center, consulta [AWS Accedere al portale di accesso](#) nella Guida per l'Accedi ad AWS utente.

Assegna l'accesso a ulteriori utenti

1. In IAM Identity Center, crea un set di autorizzazioni conforme alla best practice dell'applicazione di autorizzazioni con il privilegio minimo.

Segui le istruzioni riportate nella pagina [Creazione di un set di autorizzazioni](#) nella Guida per l'utente di AWS IAM Identity Center .

2. Assegna al gruppo prima gli utenti e poi l'accesso con autenticazione unica (Single Sign-On).

Per istruzioni, consulta [Aggiungere gruppi](#) nella Guida per l'utente di AWS IAM Identity Center .

Concessione dell'accesso programmatico

Gli utenti hanno bisogno di un accesso programmatico se desiderano interagire con AWS esternamente alla AWS Management Console. La modalità con cui concedere l'accesso programmatico dipende dal tipo di utente che accede ad AWS.

Per fornire agli utenti l'accesso programmatico, scegli una delle seguenti opzioni.

Quale utente necessita dell'accesso programmatico?	Per	Come
Identità della forza lavoro (Utenti gestiti nel centro identità IAM)	Utilizza credenziali temporane e per firmare le richieste programmatiche a AWS CLI,, AWS SDKs o. AWS APIs	Segui le istruzioni per l'interfaccia che desideri utilizzare. • Per la AWS CLI, vedere Configurazione dell'uso AWS IAM Identity Center nella AWS CLI Guida per l'utente. AWS Command Line Interface • Per AWS SDKs gli strumenti e AWS APIs, consulta l'autenticazione di IAM Identity Center nella Guida di riferimento AWS SDKs and Tools.
IAM	Utilizza credenziali temporane e per firmare le richieste programmatiche a AWS CLI, AWS SDKs, o. AWS APIs	Seguendo le istruzioni riportate in Utilizzo delle credenziali temporanee con le AWS risorse nella Guida per l'utente IAM .
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare richieste programmatiche a AWS CLI,, AWS SDKs o. AWS APIs	Segui le istruzioni per l'interfaccia che desideri utilizzare. • Per la AWS CLI, consulta Autenticazione tramite

Quale utente necessita dell'accesso programmatico?	Per	Come
		credenziali utente IAM nella Guida per l'utente.AWS Command Line Interface • Per gli strumenti AWS SDKs e gli strumenti, consulta Autenticazione tramite credenziali a lungo termine nella Guida di riferimento agli strumenti e agli AWS SDKs strumenti. • Per AWS APIs, consulta la sezione Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente IAM.

Approfondimenti

[Configura il \(\) AWS Command Line InterfaceAWS CLI](#)

Approfondimenti

[Fase 2: configurazione dell' AWS Command Line Interface \(AWS CLI\)](#)

Fase 2: configurazione dell' AWS Command Line Interface (AWS CLI)

In questa fase, viene scaricato e configurato il file AWS CLI da utilizzare con il servizio gestito per Apache Flink.

Note

Gli esercizi sulle Nozioni di base di questa guida presuppongono che tu disponga di credenziali di amministratore (`adminuser`) nell'account per eseguire le operazioni.

Note

Se la è già AWS CLI installata, potrebbe essere necessario eseguire l'aggiornamento per ottenere le funzionalità più recenti. Per ulteriori informazioni, consulta [Installazione dell' AWS Command Line Interface](#) nella Guida per l'utente dell'AWS Command Line Interface . Per verificare la versione di AWS CLI, esegui il comando seguente:

```
aws --version
```

Gli esercizi in questo tutorial richiedono la seguente AWS CLI versione della o successive:

```
aws-cli/1.16.63
```

Per configurare il AWS CLI

1. Scarica e configura la AWS CLI. Per le istruzioni, consulta i seguenti argomenti nella Guida per l'utente dell'AWS Command Line Interface :
 - [Installazione dell' AWS Command Line Interface](#)
 - [Configurazione della AWS CLI](#)
2. Aggiungi un profilo denominato per l'utente amministratore nel AWS CLI config file. Puoi usare questo profilo quando esegui i comandi della AWS CLI . Per ulteriori informazioni sui profili denominati, consulta [Profili denominati](#) in Guida per l'utente dell'AWS Command Line Interface .

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

Per un elenco delle AWS regioni disponibili, consulta [Regioni ed endpoint](#) in. Riferimenti generali di Amazon Web Services

 Note

Il codice e i comandi di esempio in questo tutorial utilizzano la regione Stati Uniti occidentali (Oregon). Per utilizzare una regione diversa, sostituisci la regione nel codice e nei comandi di questo tutorial con quella che desideri utilizzare.

3. Verifica la configurazione digitando il comando `help` riportato di seguito al prompt dei comandi:

```
aws help
```

Dopo aver configurato un AWS account e la AWS CLI, puoi provare l'esercizio successivo in cui viene configurata un'applicazione di esempio e testata la end-to-end configurazione.

Approfondimenti

[Fase 3: creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink](#)

Fase 3: creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink

In questo esercizio, viene creata un'applicazione del servizio gestito per Apache Flink con flussi di dati come origine e come sink.

Questa sezione contiene le fasi seguenti:

- [Crea due flussi di dati Amazon Kinesis](#)
- [Scrivi record di esempio nel flusso di input](#)
- [Scarica ed esamina il codice Java per lo streaming di Apache Flink](#)
- [Per creare il codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink](#)
- [Approfondimenti](#)

Crea due flussi di dati Amazon Kinesis

Prima di creare un'applicazione del servizio gestito per Apache Flink per questo esercizio, crea due flussi di dati Kinesis (`ExampleInputStream` e `ExampleOutputStream`). L'applicazione utilizza questi flussi per i flussi di origine e di destinazione dell'applicazione.

Puoi creare questi flussi utilizzando la console Amazon Kinesis o il comando AWS CLI seguente. Per istruzioni sulla console, consulta [Creazione e aggiornamento dei flussi di dati](#) nella Guida per gli sviluppatori del flusso di dati Amazon Kinesis.

Per creare i flussi di dati (AWS CLI)

1. Per creare il primo flusso (`ExampleInputStream`), utilizza il seguente comando Amazon Kinesis `create-stream` AWS CLI .

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

2. Per creare il secondo flusso utilizzato dall'applicazione per scrivere l'output, esegui lo stesso comando, modificando il nome del flusso in `ExampleOutputStream`.

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

Scrivi record di esempio nel flusso di input

In questa sezione, viene utilizzato uno script Python per scrivere record di esempio nel flusso per l'applicazione da elaborare.

Note

Questa sezione richiede [AWS SDK for Python \(Boto\)](#).

1. Crea un file denominato `stock.py` con i seguenti contenuti:

```
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. Successivamente nel tutorial, esegui lo script `stock.py` per inviare dati all'applicazione.

```
$ python stock.py
```

Scarica ed esamina il codice Java per lo streaming di Apache Flink

Il codice dell'applicazione Java per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito:

1. Clona il repository remoto con il comando seguente:

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

2. Passa alla directory `amazon-kinesis-data-analytics-java-examples/GettingStarted`.

Tieni presente quanto segue riguardo al codice dell'applicazione:

- Un file del [modello di oggetti del progetto \(pom.xml\)](#) contiene le informazioni sulla configurazione e le dipendenze dell'applicazione, incluse le librerie del servizio gestito per Apache Flink.
- Il file `BasicStreamingJob.java` contiene il metodo `main` che definisce la funzionalità dell'applicazione.
- L'applicazione utilizza un'origine Kinesis per leggere dal flusso di origine. Il seguente snippet crea l'origine Kinesis:

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

- L'applicazione crea connettori di origine e sink per accedere alle risorse esterne utilizzando un oggetto `StreamExecutionEnvironment`.
- L'applicazione crea connettori di origine e sink utilizzando proprietà statiche. Per usare proprietà dell'applicazione dinamiche, utilizza i metodi `createSourceFromApplicationProperties` e `createSinkFromApplicationProperties` per creare i connettori. Questi metodi leggono le proprietà dell'applicazione per configurare il connettori.

Per ulteriori informazioni sulle proprietà di runtime, consulta [Usa le proprietà di runtime](#).

Per creare il codice dell'applicazione

In questa sezione, viene utilizzato il compilatore Apache Maven per creare il codice Java per l'applicazione. Per ulteriori informazioni sull'installazione di Apache Maven e Java Development Kit (JDK), consulta [Soddisfa i prerequisiti per completare gli esercizi](#).

Per compilare il codice dell'applicazione

1. Per usare il codice dell'applicazione, compila il codice e comprimilo in un file JAR. È possibile compilare e creare un pacchetto del codice in uno di due modi:
 - Utilizzare lo strumento Maven a riga di comando. Crea il file JAR eseguendo il comando seguente nella directory che contiene il file `pom.xml`:

```
mvn package -Dflink.version=1.13.2
```

- Utilizza il tuo ambiente di sviluppo. Per informazioni dettagliate, consulta la documentazione relativa all'ambiente di sviluppo.

 Note

Il codice di origine fornito si basa sulle librerie di Java 11.

È possibile caricare il pacchetto come un file JAR, oppure comprimere il pacchetto e caricarlo come un file ZIP. Se crei l'applicazione utilizzando AWS CLI, specifica il tipo di contenuto del codice (JAR o ZIP).

2. Se si verificano errori durante la compilazione, verifica che la variabile di ambiente `JAVA_HOME` sia impostata correttamente.

Se l'applicazione viene compilata correttamente, viene creato il seguente file:

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

Caricamento del codice Java di streaming di Apache Flink

In questa sezione, viene creato un bucket Amazon Simple Storage Service (Amazon S3) e caricato il codice dell'applicazione.

Per caricare il codice dell'applicazione

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Seleziona Crea bucket.
3. Inserisci **ka-app-code-*<username>*** nel campo Nome bucket. Aggiungi un suffisso al nome del bucket, ad esempio il nome utente, per renderlo globalmente univoco. Scegli Next (Successivo).
4. Nella fase Configura opzioni, non modificare le impostazioni e scegli Successivo.
5. Nella fase Imposta autorizzazioni, non modificare le impostazioni e scegli Successivo.
6. Seleziona Crea bucket.
7. Nella console Amazon S3, scegli il ***<username>*** bucket ka-app-code- e scegli Carica.
8. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `aws-kinesis-analytics-java-apps-1.0.jar` creato nella fase precedente. Scegli Next (Successivo).
9. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink

È possibile creare ed eseguire un'applicazione del servizio gestito per Apache Flink utilizzando la console o la AWS CLI.

Note

Quando crei l'applicazione utilizzando la console, vengono create automaticamente le risorse AWS Identity and Access Management (IAM) e di File di CloudWatch log Amazon Logs. Quando crei l'applicazione utilizzando la AWS CLI, queste risorse vengono create separatamente.

Argomenti

- [Creazione ed esecuzione dell'applicazione \(console\)](#)
- [Crea ed esegui l'applicazione \(AWS CLI\)](#)

Creazione ed esecuzione dell'applicazione (console)

Segui questi passaggi per creare, configurare, aggiornare ed eseguire l'applicazione utilizzando la console.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.
3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:
 - Per Nome applicazione, immetti **MyApplication**.
 - Per Descrizione, inserisci **My java test app**.
 - Per Runtime, scegli Apache Flink.
 - Lascia il menu a discesa di Apache Flink versione 1.13.
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.

5. Scegli Crea applicazione.

Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`
- Ruolo: `kinesisanalytics-MyApplication-us-west-2`

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere ai flussi di dati Kinesis.

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
3. Nella pagina Riepilogo, scegli Modifica policy. Scegli la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (*012345678901*) con il tuo ID account.

```
{
  "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"
    }
]
}

```

Configurazione dell'applicazione

1. Nella MyApplication pagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:
 - Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **aws-kinesis-analytics-java-apps-1.0.jar**
3. In Accedi alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
4. Inserisci i seguenti dati:

ID gruppo	Chiave	Valore
ProducerConfigProperties	flink.inputstream.initpos	LATEST
ProducerConfigProperties	aws.region	us-west-2
ProducerConfigProperties	AggregationEnabled	false

5. In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
6. Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.
7. Scegli Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione Amazon, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: `/aws/kinesis-analytics/MyApplication`
- Flusso di log: `kinesis-analytics-log-stream`

Esecuzione dell'applicazione.

Il grafico del processo Flink può essere visualizzato eseguendo l'applicazione, aprendo il pannello di controllo di Apache Flink e scegliendo il processo Flink desiderato.

Interruzione dell'applicazione

Nella MyApplicationpagina, scegli Stop. Conferma l'operazione.

Aggiornamento dell'applicazione

Tramite la console, puoi aggiornare le impostazioni dell'applicazione, ad esempio le proprietà dell'applicazione, le impostazioni di monitoraggio e la posizione o il nome di file del JAR dell'applicazione. Puoi anche ricaricare il JAR dell'applicazione dal bucket Amazon S3 se è necessario aggiornare il codice dell'applicazione.

Nella MyApplicationpagina, scegli Configura. Aggiorna le impostazioni dell'applicazione e scegli Aggiorna.

Crea ed esegui l'applicazione (AWS CLI)

In questa sezione, verrà utilizzata la AWS CLI per creare ed eseguire il servizio gestito per Apache Flink. Il servizio gestito per Apache Flink utilizza il `kinesisanalyticsv2` AWS CLI comando per creare e interagire con le applicazioni del servizio gestito per Apache Flink.

Creazione di una policy di autorizzazione

Note

È necessario creare una policy di autorizzazione e un ruolo per l'applicazione. Se non si creano queste risorse IAM, l'applicazione non può accedere ai suoi dati e flussi di log.

Innanzitutto, crea una policy di autorizzazione con due istruzioni: una che concede le autorizzazioni per l'operazione `read` sul flusso di origine e un'altra che concede le autorizzazioni per operazioni `write` sul flusso di sink. Collega quindi la policy a un ruolo IAM (che verrà creato nella sezione successiva). Pertanto, quando il servizio gestito per Apache Flink assume il ruolo, il servizio disporrà delle autorizzazioni necessarie per leggere dal flusso di origine e scrivere nel flusso di sink.

Utilizza il codice seguente per creare la policy di autorizzazione

`AKReadSourceStreamWriteSinkStream`. Sostituisci `username` con il nome utente utilizzato per creare il bucket Amazon S3 per archiviare il codice dell'applicazione. Sostituisci l'ID account nei nomi della risorsa Amazon (ARNs) (`012345678901`) con il tuo ID account.

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

Per step-by-step istruzioni su come creare una politica di autorizzazioni, consulta il [Tutorial: Create and Attach Your First Customer Managed Policy](#) nella IAM User Guide.

Note

Per accedere ad altri servizi Amazon, puoi utilizzare AWS SDK per Java. Il servizio gestito per Apache Flink imposta automaticamente le credenziali richieste dall'SDK su quelle del ruolo IAM di esecuzione del servizio associato all'applicazione. Non sono richieste fasi aggiuntive.

Creazione di un ruolo IAM

In questa sezione, viene creato un ruolo IAM per l'applicazione del servizio gestito per Apache Flink che può essere assunto per leggere un flusso di origine e scrivere nel flusso di sink.

Il servizio gestito per Apache Flink non può accedere al tuo flusso senza autorizzazioni. Queste autorizzazioni possono essere assegnate con un ruolo IAM. Ad ogni ruolo IAM sono collegate due policy. La policy di attendibilità concede al servizio gestito per Apache Flink l'autorizzazione per assumere il ruolo e la policy di autorizzazione determina cosa può fare il servizio assumendo questo ruolo.

Collega la policy di autorizzazione creata nella sezione precedente a questo ruolo.

Per creare un ruolo IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nel riquadro di navigazione, seleziona Ruoli, quindi Crea nuovo ruolo.
3. In Seleziona tipo di identità attendibile, scegli Servizio AWS . In Scegli il servizio che utilizzerà questo ruolo, scegli Kinesis. In Seleziona il tuo caso d'uso, scegli Analisi dei dati Kinesis.

Scegli Successivo: Autorizzazioni.

4. Nella pagina Allega policy di autorizzazione, seleziona Successivo: esamina. Collega le policy di autorizzazione dopo aver creato il ruolo.
5. Nella pagina Crea ruolo, immetti **MF-stream-rw-role** per Nome ruolo. Scegli Crea ruolo.

È stato creato un nuovo ruolo IAM denominato MF-stream-rw-role. Successivamente, aggiorna le policy di attendibilità e di autorizzazione per il ruolo.

6. Collega la policy di autorizzazione al ruolo.

 Note

Per questo esercizio, il servizio gestito per Apache Flink assume questo ruolo per la lettura di dati da un flusso di dati Kinesis (origine) e la scrittura dell'output in un altro flusso di dati Kinesis. Pertanto, devi collegare la policy creata nella fase precedente, [the section called “Creazione di una policy di autorizzazione”](#).

- a. Nella pagina Riepilogo, scegli la scheda Autorizzazioni.
- b. Scegliere Collega policy.
- c. Nella casella di ricerca, immetti **AKReadSourceStreamWriteSinkStream** (la policy creata nella sezione precedente).
- d. Scegli la policy AKReadSourceStreamWriteSinkStream e seleziona Collega policy.

È stato creato il ruolo di esecuzione del servizio che l'applicazione utilizzerà per accedere alle risorse. Prendi nota dell'ARN del nuovo ruolo.

Per step-by-step istruzioni sulla creazione di un ruolo, consulta [Creating an IAM Role \(Console\)](#) nella IAM User Guide.

Creazione dell'applicazione del servizio gestito per Apache Flink

1. Salvare il seguente codice JSON in un file denominato `create_request.json`. Sostituisci l'ARN del ruolo di esempio con l'ARN per il ruolo creato in precedenza. Sostituisci il suffisso dell'ARN del bucket (*username*) con il suffisso scelto nella sezione precedente. Sostituisci l'ID account di esempio (*012345678901*) nel ruolo di esecuzione del servizio con il tuo ID account.

```
{
  "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. Esegui l'operazione [CreateApplication](#) con la richiesta precedente per creare l'applicazione:

```
aws kinesisanalyticstv2 create-application --cli-input-json file://
create_request.json
```

L'applicazione è ora creata. Avvia l'applicazione nella fase successiva.

Avvio dell'applicazione

In questa sezione, viene utilizzata l'operazione [StartApplication](#) per avviare l'applicazione.

Per avviare l'applicazione

1. Salvare il seguente codice JSON in un file denominato `start_request.json`.

```
{
```

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

2. Esegui l'operazione [StartApplication](#) con la richiesta precedente per avviare l'applicazione:

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

L'applicazione è ora in esecuzione. Puoi controllare i parametri del servizio gestito per Apache Flink sulla CloudWatch console Amazon per verificare che l'applicazione funzioni.

Interruzione dell'applicazione

In questa sezione, viene utilizzata l'operazione [StopApplication](#) per interrompere l'applicazione.

Per interrompere l'applicazione

1. Salvare il seguente codice JSON in un file denominato `stop_request.json`.

```
{
  "ApplicationName": "test"
}
```

2. Esegui l'operazione [StopApplication](#) con la seguente richiesta di interrompere l'applicazione:

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

L'applicazione è ora interrotta.

Aggiungi un'opzione CloudWatch di registrazione

Puoi utilizzare la AWS CLI per aggiungere un flusso di CloudWatch log Amazon all'applicazione. Per ulteriori informazioni sull'utilizzo CloudWatch dei log con la tua applicazione, consulta [the section called "Configurare la registrazione delle applicazioni in Managed Service per Apache Flink"](#).

Aggiornamento delle proprietà di ambiente

In questa sezione, viene utilizzata l'operazione [UpdateApplication](#) per modificare le proprietà di ambiente per l'applicazione senza ricompilare il codice dell'applicazione. In questo esempio, viene modificata la regione dei flussi di origine e destinazione.

Per aggiornare le proprietà di ambiente per l'applicazione

1. Salvare il seguente codice JSON in un file denominato `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. Esegui l'operazione [UpdateApplication](#) con la richiesta precedente per aggiornare le proprietà di ambiente:

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

Aggiornamento del codice dell'applicazione

Quando devi aggiornare il codice dell'applicazione con una nuova versione del pacchetto di codice, utilizza l'[UpdateApplication](#) AWS CLI operazione.

Note

Per caricare una nuova versione del codice dell'applicazione con lo stesso nome file, è necessario specificare la nuova versione dell'oggetto. Per ulteriori informazioni sull'uso delle versioni degli oggetti Amazon S3, consulta [Abilitazione o disattivazione del controllo delle versioni](#).

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione ed effettua la chiamata a Amazon S3 e lo stesso nome dell'oggetto `UpdateApplication`, nonché la nuova versione dell'oggetto. L'applicazione verrà riavviata con il nuovo pacchetto di codice.

Il seguente esempio di richiesta per l'operazione `UpdateApplication` ricarica il codice dell'applicazione e riavvia l'applicazione. Aggiorna l'`CurrentApplicationVersionId` alla versione corrente dell'applicazione. Puoi controllare la versione corrente dell'applicazione utilizzando le operazioni `ListApplications` o `DescribeApplication`. Aggiorna il suffisso del nome del bucket (`<username>`) con il suffisso che hai scelto nella sezione. [the section called “Crea due flussi di dati Amazon Kinesis”](#)

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

Approfondimenti

[Passaggio 4: eliminare le AWS risorse](#)

Passaggio 4: eliminare le AWS risorse

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Nozioni di base.

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione dei flussi di dati Kinesis](#)
- [Eliminazione dell'oggetto e del bucket Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)
- [Approfondimenti](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com /kines.](https://console.aws.amazon.com/kines.)
2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e quindi conferma l'eliminazione.

Eliminazione dei flussi di dati Kinesis

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink https://console.aws.amazon.com
2. Nel pannello del flusso di dati Kinesis, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.
4. Nella pagina Kinesis Streams, scegli, scegli Azioni ExampleOutputStream, scegli Elimina, quindi conferma l'eliminazione.

Eliminazione dell'oggetto e del bucket Amazon S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>

2. Scegli il **<username>**secchio ka-app-code -.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

Eliminazione delle risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-west-2.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytus-west-2. MyApplication
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

Approfondimenti

[Fase 5: fasi successive](#)

Fase 5: fasi successive

Ora che hai creato ed eseguito un'applicazione di base del servizio gestito per Apache Flink, consulta le seguenti risorse per esplorare soluzioni del servizio gestito per Apache Flink più avanzate.

- [La soluzione di AWS streaming di dati per Amazon Kinesis](#): la soluzione di AWS streaming di dati per Amazon Kinesis configura automaticamente AWS i servizi necessari per acquisire, archiviare, elaborare e distribuire dati in streaming con facilità. La soluzione offre diverse opzioni per risolvere i casi d'uso dei dati di streaming. L'opzione del servizio gestito per Apache Flink fornisce un esempio

di end-to-end streaming ETL in cui un'applicazione reale esegue operazioni analitiche su dati simulati dei taxi di New York. La soluzione configura tutte le AWS risorse necessarie come ruoli e policy IAM, un CloudWatch pannello di controllo e CloudWatch allarmi.

- [AWS Soluzione di streaming di dati per Amazon MSK](#): la soluzione di AWS streaming di dati per Amazon MSK fornisce AWS CloudFormation modelli in cui i dati fluiscono attraverso produttori, storage in streaming, consumatori e destinazioni.
- [Clickstream Lab con Apache Flink e Apache Kafka](#): un laboratorio end-to-end per casi d'uso clickstream che utilizza lo streaming gestito da Amazon per Apache Kafka per lo storage in streaming e il servizio gestito per Apache Flink per le applicazioni Apache Flink per l'elaborazione di flussi.
- [Workshop su Amazon del servizio gestito per Apache Flink](#): in questo workshop, viene creata un'architettura di end-to-end streaming per importare, analizzare e visualizzare i dati di streaming quasi in tempo reale. Hai deciso di migliorare le attività di una compagnia di taxi a New York City. Analizza i dati di telemetria di una flotta di taxi a New York City quasi in tempo reale per ottimizzare le operazioni della flotta.
- [Scopri Flink: esercitazioni pratiche](#): formazione introduttiva ufficiale su Apache Flink che ti consente di iniziare a scrivere applicazioni di streaming scalabili ETL, di analisi e basate su eventi.

Note

Tieni presente che il servizio gestito per Apache Flink non supporta la versione Apache Flink (1.12) utilizzata in questo corso di formazione. È possibile utilizzare Flink 1.15.2 nel servizio gestito per Apache Flink per Apache Flink.

Nozioni di base: Flink 1.11.1 - obsoleto

Note

Le versioni 1.6, 1.8 e 1.11 di Apache Flink non sono supportate dalla community di Apache Flink da oltre tre anni. Abbiamo intenzione di rendere obsolete queste versioni in Amazon Managed Service for Apache Flink il 5 novembre 2024. A partire da questa data, non sarà possibile creare nuove applicazioni per queste versioni di Flink. È possibile continuare a eseguire le applicazioni esistenti in questo momento. Puoi aggiornare le tue applicazioni in modo statico utilizzando la funzionalità di aggiornamento delle versioni in loco in Amazon

Managed Service for Apache Flink. Per ulteriori informazioni, consulta. [Usa gli aggiornamenti di versione sul posto per Apache Flink](#)

Questo argomento contiene una versione del [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#) tutorial che utilizza Apache Flink 1.11.1.

Questa sezione presenta i concetti fondamentali del servizio gestito per Apache Flink e dell'API. DataStream Descrive le opzioni disponibili per la creazione e il test delle applicazioni. Fornisce inoltre istruzioni per l'installazione degli strumenti necessari per completare i tutorial di questa guida e creare la tua prima applicazione.

Argomenti

- [Componenti dell'applicazione del servizio gestito per Apache Flink](#)
- [Prerequisiti per il completamento degli esercizi](#)
- [Fase 1: Impostazione di un AWS account e creazione di un utente amministratore](#)
- [Fase 2: configurazione di AWS Command Line Interface \(AWS CLI\)](#)
- [Fase 3: creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink](#)
- [Passaggio 4: eliminare le AWS risorse](#)
- [Fase 5: fasi successive](#)

Componenti dell'applicazione del servizio gestito per Apache Flink

Per elaborare i dati, l'applicazione del servizio gestito per Apache Flink utilizza un'applicazione Java/ Apache Maven o Scala che elabora l'input e produce l'output utilizzando il runtime di Apache Flink.

Un'applicazione del servizio gestito per Apache Flink include i componenti riportati di seguito:

- Proprietà di runtime: è possibile utilizzare le proprietà di runtime per configurare l'applicazione senza ricompilare il codice dell'applicazione.
- Origine: l'applicazione consuma i dati utilizzando un'origine. Un connettore di origine legge i dati da un flusso di dati Kinesis, da un bucket Amazon S3, ecc. Per ulteriori informazioni, consulta [Aggiungi sorgenti di dati in streaming](#).
- Operatori: l'applicazione elabora i dati utilizzando uno o più operatori. Un operatore può trasformare, arricchire o aggregare i dati. Per ulteriori informazioni, consulta [Operatori](#).

- Sink: l'applicazione produce dati verso origini esterne utilizzando i sink. Un connettore sink scrive i dati su un flusso di dati Kinesis, un flusso Firehose, un bucket Amazon S3, ecc. Per ulteriori informazioni, consulta [Scrivi dati utilizzando i lavandini](#).

Dopo aver creato, compilato e compresso il codice dell'applicazione, caricherai il pacchetto di codice in un bucket Amazon Simple Storage Service (Amazon S3). Potrai quindi creare un'applicazione del servizio gestito per Apache Flink. Dovrai inserire la posizione del pacchetto di codice, un flusso di dati Kinesis come origine dati di streaming e in genere una posizione di streaming o di file che riceva i dati elaborati dall'applicazione.

Prerequisiti per il completamento degli esercizi

Per completare le fasi in questa guida, è richiesto quanto segue:

- [Java Development Kit \(JDK\) versione 11](#). Imposta la variabile di ambiente JAVA_HOME in modo che punti alla posizione di installazione di JDK.
- Ti consigliamo di utilizzare un ambiente di sviluppo (ad esempio [Eclipse Java Neon](#) o [IntelliJ IDEA](#)) per sviluppare e compilare l'applicazione.
- [Client Git](#). Installa il client Git se non lo hai già fatto.
- [Apache Maven Compiler Plugin](#). Maven deve trovarsi nel percorso di lavoro. Per testare l'installazione Apache Maven, immetti quanto segue:

```
$ mvn -version
```

Per iniziare, vai alla pagina [Configura un AWS account e crea un utente amministratore](#).

Fase 1: Impostazione di un AWS account e creazione di un utente amministratore

Registrazione per un account Account AWS

Se non si dispone di un Account AWS, completare la procedura seguente per crearne uno.

Per registrarti a un Account AWS

1. Apri la <https://portal.aws.amazon.com/billing/registrazione>.
2. Segui le istruzioni online.

Nel corso della procedura di registrazione riceverai una telefonata, durante la quale sarà necessario inserire un codice di verifica attraverso la tastiera del telefono.

Quando ti iscrivi a un Account AWS, Utente root dell'account AWS viene creato un. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente e utilizza solo l'utente root per eseguire [attività che richiedono l'accesso di un utente root](#).

AWS Al termine del processo di registrazione, riceverai un'e-mail di conferma da. Puoi visualizzare l'attività corrente del tuo account e gestire l'account in qualsiasi momento accedendo a <https://aws.amazon.com/e> scegliendo My Account (Il mio account).

Crea un utente con accesso amministrativo

Dopo aver effettuato la registrazione di un Account AWS, proteggi Utente root dell'account AWS AWS IAM Identity Center, abilita e crea un utente amministratore in modo da non utilizzare l'utente root per le attività quotidiane.

Proteggi i tuoi Utente root dell'account AWS

1. Accedi [AWS Management Console](#) come proprietario dell'account scegliendo Utente root e inserendo il tuo indirizzo Account AWS email. Nella pagina successiva, inserisci la password.

Per informazioni sull'accesso utilizzando un utente root, consulta la pagina [Signing in as the root user](#) della Guida per l'utente di Accedi ad AWS .

2. Abilita l'autenticazione a più fattori (MFA) per l'utente root.

Per ricevere istruzioni, consulta [Abilitare un dispositivo MFA virtuale per l'utente Account AWS root della \(console\)](#) nella Guida per l'utente IAM.

Crea un utente con accesso amministrativo

1. Abilita Centro identità IAM.

Per istruzioni, consulta [Abilitazione di AWS IAM Identity Center](#) nella Guida per l'utente di AWS IAM Identity Center .

2. In IAM Identity Center, assegna l'accesso amministrativo a un utente.

Per un tutorial sull'utilizzo di IAM Identity Center directory come fonte di identità, consulta [Configurare l'accesso utente con l'impostazione predefinita IAM Identity Center directory](#) nella Guida per l'AWS IAM Identity Center utente.

Accesso come utente amministratore

- Per accedere con l'utente IAM Identity Center, utilizza l'URL di accesso che è stato inviato al tuo indirizzo e-mail quando hai creato l'utente IAM Identity Center.

Per assistenza nell'accesso mediante un utente IAM Identity Center, consulta [AWS Accedere al portale di accesso](#) nella Guida per l'Accedi ad AWS utente.

Assegna l'accesso a ulteriori utenti

1. In IAM Identity Center, crea un set di autorizzazioni conforme alla best practice dell'applicazione di autorizzazioni con il privilegio minimo.

Segui le istruzioni riportate nella pagina [Creazione di un set di autorizzazioni](#) nella Guida per l'utente di AWS IAM Identity Center .

2. Assegna al gruppo prima gli utenti e poi l'accesso con autenticazione unica (Single Sign-On).

Per istruzioni, consulta [Aggiungere gruppi](#) nella Guida per l'utente di AWS IAM Identity Center .

Concessione dell'accesso programmatico

Gli utenti hanno bisogno di un accesso programmatico se desiderano interagire con AWS esternamente alla AWS Management Console. La modalità con cui concedere l'accesso programmatico dipende dal tipo di utente che accede ad AWS.

Per fornire agli utenti l'accesso programmatico, scegli una delle seguenti opzioni.

Quale utente necessita dell'accesso programmatico?	Per	Come
Identità della forza lavoro (Utenti gestiti nel centro identità IAM)	Utilizza credenziali temporane e per firmare le richieste	Segui le istruzioni per l'interfaccia che desideri utilizzare.

Quale utente necessita dell'accesso programmatico?	Per	Come
	programmatiche a AWS CLI,, AWS SDKs o. AWS APIs	• Per la AWS CLI, vedere Configurazione dell'uso AWS IAM Identity Center nella AWS CLI Guida per l'utente.AWS Command Line Interface • Per AWS SDKs gli strumenti e AWS APIs, consulta l'autenticazione di IAM Identity Center nella Guida di riferimento AWS SDKs and Tools.
IAM	Utilizza credenziali temporanee e per firmare le richieste programmatiche a AWS CLI, AWS SDKs, o. AWS APIs	Seguendo le istruzioni riportate in Utilizzo delle credenziali temporanee con le AWS risorse nella Guida per l'utente IAM .

Quale utente necessita dell'accesso programmatico?	Per	Come
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare richieste programmatiche a AWS CLI,, AWS SDKs o. AWS APIs	Segui le istruzioni per l'interfaccia che desideri utilizzare. • Per la AWS CLI, consulta Autenticazione tramite credenziali utente IAM nella Guida per l'utente.AWS Command Line Interface • Per gli strumenti AWS SDKs e gli strumenti, consulta Autenticazione tramite credenziali a lungo termine nella Guida di riferimento agli strumenti e agli AWS SDKs strumenti. • Per AWS APIs, consulta la sezione Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente IAM.

Approfondimenti

[Configura il \(\) AWS Command Line InterfaceAWS CLI](#)

Fase 2: configurazione di AWS Command Line Interface (AWS CLI)

In questa fase, viene scaricato e configurato il file AWS CLI da utilizzare con il servizio gestito per Apache Flink.

Note

Gli esercizi sulle Nozioni di base di questa guida presuppongono che tu disponga di credenziali di amministratore (`adminuser`) nell'account per eseguire le operazioni.

Note

Se la è già AWS CLI installata, potrebbe essere necessario eseguire l'aggiornamento per ottenere le funzionalità più recenti. Per ulteriori informazioni, consulta [Installazione dell' AWS Command Line Interface](#) nella Guida per l'utente dell'AWS Command Line Interface . Per verificare la versione di AWS CLI, esegui il comando seguente:

```
aws --version
```

Gli esercizi in questo tutorial richiedono la seguente AWS CLI versione della o successive:

```
aws-cli/1.16.63
```

Per configurare il AWS CLI

1. Scarica e configura la AWS CLI. Per le istruzioni, consulta i seguenti argomenti nella Guida per l'utente dell'AWS Command Line Interface :
 - [Installazione dell' AWS Command Line Interface](#)
 - [Configurazione della AWS CLI](#)
2. Aggiungi un profilo denominato per l'utente amministratore nel AWS CLI config file. Puoi usare questo profilo quando esegui i comandi della AWS CLI . Per ulteriori informazioni sui profili denominati, consulta [Profili denominati](#) in Guida per l'utente dell'AWS Command Line Interface .

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

Per un elenco delle AWS regioni disponibili, consulta [Regioni ed endpoint](#) in. Riferimenti generali di Amazon Web Services

 Note

Il codice e i comandi di esempio in questo tutorial utilizzano la regione Stati Uniti occidentali (Oregon). Per utilizzare una regione diversa, sostituisci la regione nel codice e nei comandi di questo tutorial con quella che desideri utilizzare.

3. Verifica la configurazione digitando il comando `help` riportato di seguito al prompt dei comandi:

```
aws help
```

Dopo aver configurato un AWS account e la AWS CLI, puoi provare l'esercizio successivo in cui viene configurata un'applicazione di esempio e testata la end-to-end configurazione.

Approfondimenti

[Fase 3: creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink](#)

Fase 3: creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink

In questo esercizio, viene creata un'applicazione del servizio gestito per Apache Flink con flussi di dati come origine e come sink.

Questa sezione contiene le fasi seguenti:

- [Crea due flussi di dati Amazon Kinesis](#)
- [Scrivi record di esempio nel flusso di input](#)
- [Scarica ed esamina il codice Java per lo streaming di Apache Flink](#)
- [Per creare il codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink](#)
- [Approfondimenti](#)

Crea due flussi di dati Amazon Kinesis

Prima di creare un'applicazione del servizio gestito per Apache Flink per questo esercizio, crea due flussi di dati Kinesis (`ExampleInputStream` e `ExampleOutputStream`). L'applicazione utilizza questi flussi per i flussi di origine e di destinazione dell'applicazione.

Puoi creare questi flussi utilizzando la console Amazon Kinesis o il comando AWS CLI seguente. Per istruzioni sulla console, consulta [Creazione e aggiornamento dei flussi di dati](#) nella Guida per gli sviluppatori del flusso di dati Amazon Kinesis.

Per creare i flussi di dati (AWS CLI)

1. Per creare il primo flusso (`ExampleInputStream`), utilizza il seguente comando Amazon Kinesis `create-stream` AWS CLI .

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

2. Per creare il secondo flusso utilizzato dall'applicazione per scrivere l'output, esegui lo stesso comando, modificando il nome del flusso in `ExampleOutputStream`.

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

Scrivi record di esempio nel flusso di input

In questa sezione, viene utilizzato uno script Python per scrivere record di esempio nel flusso per l'applicazione da elaborare.

Note

Questa sezione richiede [AWS SDK for Python \(Boto\)](#).

1. Crea un file denominato `stock.py` con i seguenti contenuti:

```
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. Successivamente nel tutorial, esegui lo script `stock.py` per inviare dati all'applicazione.

```
$ python stock.py
```

Scarica ed esamina il codice Java per lo streaming di Apache Flink

Il codice dell'applicazione Java per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito:

1. Clona il repository remoto con il comando seguente:

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

2. Passa alla directory `amazon-kinesis-data-analytics-java-examples/GettingStarted`.

Tieni presente quanto segue riguardo al codice dell'applicazione:

- Un file del [modello di oggetti del progetto \(pom.xml\)](#) contiene le informazioni sulla configurazione e le dipendenze dell'applicazione, incluse le librerie del servizio gestito per Apache Flink.
- Il file `BasicStreamingJob.java` contiene il metodo `main` che definisce la funzionalità dell'applicazione.
- L'applicazione utilizza un'origine Kinesis per leggere dal flusso di origine. Il seguente snippet crea l'origine Kinesis:

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

- L'applicazione crea connettori di origine e sink per accedere alle risorse esterne utilizzando un oggetto `StreamExecutionEnvironment`.
- L'applicazione crea connettori di origine e sink utilizzando proprietà statiche. Per usare proprietà dell'applicazione dinamiche, utilizza i metodi `createSourceFromApplicationProperties` e `createSinkFromApplicationProperties` per creare i connettori. Questi metodi leggono le proprietà dell'applicazione per configurare il connettori.

Per ulteriori informazioni sulle proprietà di runtime, consulta [Usa le proprietà di runtime](#).

Per creare il codice dell'applicazione

In questa sezione, viene utilizzato il compilatore Apache Maven per creare il codice Java per l'applicazione. Per ulteriori informazioni sull'installazione di Apache Maven e Java Development Kit (JDK), consulta [Soddisfa i prerequisiti per completare gli esercizi](#).

Per compilare il codice dell'applicazione

1. Per usare il codice dell'applicazione, compila il codice e comprimilo in un file JAR. È possibile compilare e creare un pacchetto del codice in uno di due modi:

- Utilizzare lo strumento Maven a riga di comando. Crea il file JAR eseguendo il comando seguente nella directory che contiene il file `pom.xml`:

```
mvn package -Dflink.version=1.11.3
```

- Utilizza il tuo ambiente di sviluppo. Per informazioni dettagliate, consulta la documentazione relativa all'ambiente di sviluppo.

Note

Il codice di origine fornito si basa sulle librerie di Java 11. Assicurati che la versione Java del tuo progetto sia la 11.

È possibile caricare il pacchetto come un file JAR, oppure comprimere il pacchetto e caricarlo come un file ZIP. Se crei l'applicazione utilizzando AWS CLI, specifica il tipo di contenuto del codice (JAR o ZIP).

2. Se si verificano errori durante la compilazione, verifica che la variabile di ambiente `JAVA_HOME` sia impostata correttamente.

Se l'applicazione viene compilata correttamente, viene creato il seguente file:

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

Caricamento del codice Java di streaming di Apache Flink

In questa sezione, viene creato un bucket Amazon Simple Storage Service (Amazon S3) e caricato il codice dell'applicazione.

Per caricare il codice dell'applicazione

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Seleziona Crea bucket.
3. Inserisci **ka-app-code-*<username>*** nel campo Nome bucket. Aggiungi un suffisso al nome del bucket, ad esempio il nome utente, per renderlo globalmente univoco. Scegli Next (Successivo).
4. Nella fase Configura opzioni, non modificare le impostazioni e scegli Successivo.

5. Nella fase Imposta autorizzazioni, non modificare le impostazioni e scegli Successivo.
6. Seleziona Crea bucket.
7. Nella console Amazon S3, scegli il **<username>** bucket ka-app-code- e scegli Carica.
8. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `aws-kinesis-analytics-java-apps-1.0.jar` creato nella fase precedente. Scegli Next (Successivo).
9. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink

È possibile creare ed eseguire un'applicazione del servizio gestito per Apache Flink utilizzando la console o la AWS CLI.

Note

Quando crei l'applicazione utilizzando la console, vengono create automaticamente le risorse AWS Identity and Access Management (IAM) e di File di CloudWatch log Amazon Logs. Quando crei l'applicazione utilizzando la AWS CLI, queste risorse vengono create separatamente.

Argomenti

- [Creazione ed esecuzione dell'applicazione \(console\)](#)
- [Crea ed esegui l'applicazione \(AWS CLI\)](#)

Creazione ed esecuzione dell'applicazione (console)

Segui questi passaggi per creare, configurare, aggiornare ed eseguire l'applicazione utilizzando la console.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink `https://console.aws.amazon.com`
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.

3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:
 - Per Nome applicazione, immetti **MyApplication**.
 - Per Descrizione, inserisci **My java test app**.
 - Per Runtime, scegli Apache Flink.
 - Lascia il menu a discesa di Apache Flink 1.11 (versione consigliata).
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
5. Scegli Crea applicazione.

Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`
- Ruolo: `kinesisanalytics-MyApplication-us-west-2`

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere ai flussi di dati Kinesis.

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
3. Nella pagina Riepilogo, scegli Modifica policy. Scegli la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (`012345678901`) con il tuo ID account.

```
{
  "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"
    }
]
}

```

Configurazione dell'applicazione

1. Nella MyApplicationpagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:
 - Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **aws-kinesis-analytics-java-apps-1.0.jar**
3. In Accesso alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
4. In Proprietà, per ID gruppo, inserisci **ProducerConfigProperties**.
5. Immetti i valori e le proprietà dell'applicazione seguenti:

ID gruppo	Chiave	Valore
ProducerConfigProperties	flink.inputstream.initpos	LATEST
ProducerConfigProperties	aws.region	us-west-2

ID gruppo	Chiave	Valore
ProducerConfigProperties	AggregationEnabled	false

6. In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
7. Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.
8. Scegli Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione Amazon, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: /aws/kinesis-analytics/MyApplication
- Flusso di log: kinesis-analytics-log-stream

Esecuzione dell'applicazione.

Il grafico del processo Flink può essere visualizzato eseguendo l'applicazione, aprendo il pannello di controllo di Apache Flink e scegliendo il processo Flink desiderato.

Interruzione dell'applicazione

Nella MyApplicationpagina, scegli Stop. Conferma l'operazione.

Aggiornamento dell'applicazione

Tramite la console, puoi aggiornare le impostazioni dell'applicazione, ad esempio le proprietà dell'applicazione, le impostazioni di monitoraggio e la posizione o il nome di file del JAR dell'applicazione. Puoi anche ricaricare il JAR dell'applicazione dal bucket Amazon S3 se è necessario aggiornare il codice dell'applicazione.

Nella MyApplicationpagina, scegli Configura. Aggiorna le impostazioni dell'applicazione e scegli Aggiorna.

Crea ed esegui l'applicazione (AWS CLI)

In questa sezione, viene utilizzata la per creare ed AWS CLI eseguire l'applicazione del servizio gestito per Apache Flink. Un servizio gestito per Apache Flink utilizza il `kinesisanalyticsv2` AWS CLI comando per creare e interagire con le applicazioni del servizio gestito per Apache Flink.

Creazione di una policy di autorizzazione

Note

È necessario creare una policy di autorizzazione e un ruolo per l'applicazione. Se non si creano queste risorse IAM, l'applicazione non può accedere ai suoi dati e flussi di log.

Innanzitutto, crea una policy di autorizzazione con due istruzioni: una che concede le autorizzazioni per l'operazione `read` sul flusso di origine e un'altra che concede le autorizzazioni per operazioni `write` sul flusso di sink. Collega quindi la policy a un ruolo IAM (che verrà creato nella sezione successiva). Pertanto, quando il servizio gestito per Apache Flink assume il ruolo, il servizio disporrà delle autorizzazioni necessarie per leggere dal flusso di origine e scrivere nel flusso di sink.

Utilizza il codice seguente per creare la policy di autorizzazione

`AKReadSourceStreamWriteSinkStream`. Sostituisci `username` con il nome utente utilizzato per creare il bucket Amazon S3 per archiviare il codice dell'applicazione. Sostituisci l'ID account nei nomi della risorsa Amazon (ARNs) (`012345678901`) con il tuo ID account.

```
{
  "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"
    }
]
```

Per step-by-step istruzioni su come creare una politica di autorizzazioni, consulta il [Tutorial: Create and Attach Your First Customer Managed Policy](#) nella IAM User Guide.

Note

Per accedere ad altri servizi Amazon, puoi utilizzare AWS SDK per Java. Il servizio gestito per Apache Flink imposta automaticamente le credenziali richieste dall'SDK su quelle del ruolo IAM di esecuzione del servizio associato all'applicazione. Non sono richieste fasi aggiuntive.

Creazione di un ruolo IAM

In questa sezione, viene creato un ruolo IAM per l'applicazione del servizio gestito per Apache Flink che può essere assunto per leggere un flusso di origine e scrivere nel flusso di sink.

Il servizio gestito per Apache Flink non può accedere al tuo flusso senza autorizzazioni. Queste autorizzazioni possono essere assegnate con un ruolo IAM. Ad ogni ruolo IAM sono collegate due policy. La policy di attendibilità concede al servizio gestito per Apache Flink l'autorizzazione per assumere il ruolo e la policy di autorizzazione determina cosa può fare il servizio assumendo questo ruolo.

Collega la policy di autorizzazione creata nella sezione precedente a questo ruolo.

Per creare un ruolo IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nel riquadro di navigazione, seleziona Ruoli, quindi Crea nuovo ruolo.
3. In Seleziona tipo di identità attendibile, scegli Servizio AWS . In Scegli il servizio che utilizzerà questo ruolo, scegli Kinesis. In Seleziona il tuo caso d'uso, scegli Analisi dei dati Kinesis.

Scegli Successivo: Autorizzazioni.

4. Nella pagina Allega policy di autorizzazione, seleziona Successivo: esamina. Collega le policy di autorizzazione dopo aver creato il ruolo.
5. Nella pagina Crea ruolo, immetti **MF-stream-rw-role** per Nome ruolo. Scegli Crea ruolo.

È stato creato un nuovo ruolo IAM denominato MF-stream-rw-role. Successivamente, aggiorna le policy di attendibilità e di autorizzazione per il ruolo.

6. Collega la policy di autorizzazione al ruolo.

Note

Per questo esercizio, il servizio gestito per Apache Flink assume questo ruolo per la lettura di dati da un flusso di dati Kinesis (origine) e la scrittura dell'output in un altro flusso di dati Kinesis. Pertanto, devi collegare la policy creata nella fase precedente, [the section called “Creazione di una policy di autorizzazione”](#).

- a. Nella pagina Riepilogo, scegli la scheda Autorizzazioni.
- b. Scegliere Collega policy.
- c. Nella casella di ricerca, immetti **AKReadSourceStreamWriteSinkStream** (la policy creata nella sezione precedente).
- d. Scegli la policy AKReadSourceStreamWriteSinkStream e seleziona Collega policy.

È stato creato il ruolo di esecuzione del servizio che l'applicazione utilizzerà per accedere alle risorse. Prendi nota dell'ARN del nuovo ruolo.

Per step-by-step istruzioni sulla creazione di un ruolo, consulta [Creating an IAM Role \(Console\)](#) nella IAM User Guide.

Creazione dell'applicazione del servizio gestito per Apache Flink

1. Salvare il seguente codice JSON in un file denominato `create_request.json`. Sostituisci l'ARN del ruolo di esempio con l'ARN per il ruolo creato in precedenza. Sostituisci il suffisso dell'ARN del bucket (*username*) con il suffisso scelto nella sezione precedente. Sostituisci l>ID account di esempio (*012345678901*) nel ruolo di esecuzione del servizio con il tuo ID account.

```
{
  "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. Esegui l'operazione [CreateApplication](#) con la richiesta precedente per creare l'applicazione:

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

L'applicazione è ora creata. Avvia l'applicazione nella fase successiva.

Avvia l'applicazione

In questa sezione, viene utilizzata l'operazione [StartApplication](#) per avviare l'applicazione.

Per avviare l'applicazione

1. Salvare il seguente codice JSON in un file denominato `start_request.json`.

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

2. Esegui l'operazione [StartApplication](#) con la richiesta precedente per avviare l'applicazione:

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

L'applicazione è ora in esecuzione. Puoi controllare i parametri del servizio gestito per Apache Flink sulla CloudWatch console Amazon per verificare che l'applicazione funzioni.

Interruzione dell'applicazione

In questa sezione, viene utilizzata l'operazione [StopApplication](#) per interrompere l'applicazione.

Per interrompere l'applicazione

1. Salvare il seguente codice JSON in un file denominato `stop_request.json`.

```
{
  "ApplicationName": "test"
}
```

2. Esegui l'operazione [StopApplication](#) con la seguente richiesta di interrompere l'applicazione:

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

L'applicazione è ora interrotta.

Aggiungi un'opzione CloudWatch di registrazione

Puoi utilizzare la AWS CLI per aggiungere un flusso di CloudWatch log Amazon all'applicazione. Per ulteriori informazioni sull'utilizzo CloudWatch dei log con la tua applicazione, consulta [the section called "Configurare la registrazione delle applicazioni in Managed Service per Apache Flink"](#).

Aggiornamento delle proprietà dell'ambiente

In questa sezione, viene utilizzata l'operazione [UpdateApplication](#) per modificare le proprietà di ambiente per l'applicazione senza ricompilare il codice dell'applicazione. In questo esempio, viene modificata la regione dei flussi di origine e destinazione.

Per aggiornare le proprietà di ambiente per l'applicazione

1. Salvare il seguente codice JSON in un file denominato `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. Esegui l'operazione [UpdateApplication](#) con la richiesta precedente per aggiornare le proprietà di ambiente:

```
aws kinesisanalyticstv2 update-application --cli-input-json file://
update_properties_request.json
```

Aggiornamento del codice dell'applicazione

Quando devi aggiornare il codice dell'applicazione con una nuova versione del pacchetto di codice, utilizza l'[UpdateApplication](#) AWS CLI operazione.

Note

Per caricare una nuova versione del codice dell'applicazione con lo stesso nome file, è necessario specificare la nuova versione dell'oggetto. Per ulteriori informazioni sull'uso delle versioni degli oggetti Amazon S3, consulta [Abilitazione o disattivazione del controllo delle versioni](#).

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione ed effettua la chiamata a Amazon S3 e lo stesso nome dell'oggetto `UpdateApplication`, nonché la nuova versione dell'oggetto. L'applicazione verrà riavviata con il nuovo pacchetto di codice.

Il seguente esempio di richiesta per l'operazione `UpdateApplication` ricarica il codice dell'applicazione e riavvia l'applicazione. Aggiorna l'`CurrentApplicationVersionId` alla versione corrente dell'applicazione. Puoi controllare la versione corrente dell'applicazione utilizzando le operazioni `ListApplications` o `DescribeApplication`. Aggiorna il suffisso del nome del

bucket (*<username>*) con il suffisso che hai scelto nella sezione. [the section called “Crea due flussi di dati Amazon Kinesis”](#)

```
{
  "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": "SAMPLEUehYngP87ex1nzYIGYgfhyvpvDU"
        }
      }
    }
  }
}
```

Approfondimenti

[Passaggio 4: eliminare le AWS risorse](#)

Passaggio 4: eliminare le AWS risorse

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Nozioni di base.

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione dei flussi di dati Kinesis](#)
- [Eliminazione dell'oggetto e del bucket Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)
- [Approfondimenti](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com/kines.](https://console.aws.amazon.com/kines)

2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e quindi conferma l'eliminazione.

Eliminazione dei flussi di dati Kinesis

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nel pannello del flusso di dati Kinesis, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.
4. Nella pagina Kinesis Streams, scegli, scegli Azioni ExampleOutputStream, scegli Elimina, quindi conferma l'eliminazione.

Eliminazione dell'oggetto e del bucket Amazon S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il **<username>**secchio ka-app-code -.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

Eliminazione delle risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-west-2.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytus-west-2. MyApplication
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.

3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

Approfondimenti

[Fase 5: fasi successive](#)

Fase 5: fasi successive

Ora che hai creato ed eseguito un'applicazione di base del servizio gestito per Apache Flink, consulta le seguenti risorse per esplorare soluzioni del servizio gestito per Apache Flink più avanzate.

- [La soluzione di AWS streaming di dati per Amazon Kinesis](#): la soluzione di AWS streaming di dati per Amazon Kinesis configura automaticamente AWS i servizi necessari per acquisire, archiviare, elaborare e distribuire dati in streaming con facilità. La soluzione offre diverse opzioni per risolvere i casi d'uso dei dati di streaming. L'opzione del servizio gestito per Apache Flink fornisce un esempio di end-to-end streaming ETL in cui un'applicazione reale esegue operazioni analitiche su dati simulati dei taxi di New York. La soluzione configura tutte le AWS risorse necessarie come ruoli e policy IAM, un CloudWatch pannello di controllo e CloudWatch allarmi.
- [AWS Soluzione di streaming di dati per Amazon MSK](#): la soluzione di AWS streaming di dati per Amazon MSK fornisce AWS CloudFormation modelli in cui i dati fluiscono attraverso produttori, storage in streaming, consumatori e destinazioni.
- [Clickstream Lab con Apache Flink e Apache Kafka](#): un laboratorio end-to-end per casi d'uso clickstream che utilizza lo streaming gestito da Amazon per Apache Kafka per lo storage in streaming e il servizio gestito per Apache Flink per le applicazioni Apache Flink per l'elaborazione di flussi.
- [Workshop su Amazon del servizio gestito per Apache Flink](#): in questo workshop, viene creata un'architettura di end-to-end streaming per importare, analizzare e visualizzare i dati di streaming quasi in tempo reale. Hai deciso di migliorare le attività di una compagnia di taxi a New York City. Analizza i dati di telemetria di una flotta di taxi a New York City quasi in tempo reale per ottimizzare le operazioni della flotta.
- [Scopri Flink: esercitazioni pratiche](#): formazione introduttiva ufficiale su Apache Flink che ti consente di iniziare a scrivere applicazioni di streaming scalabili ETL, di analisi e basate su eventi.

 Note

Tieni presente che il servizio gestito per Apache Flink non supporta la versione Apache Flink (1.12) utilizzata in questo corso di formazione. È possibile utilizzare Flink 1.15.2 nel servizio gestito per Apache Flink per Apache Flink.

- [Esempi di codice di Apache Flink](#): un GitHub repository di un'ampia varietà di esempi di applicazioni Apache Flink.

Nozioni di base: Flink 1.8.2 - obsoleto

 Note

Le versioni 1.6, 1.8 e 1.11 di Apache Flink non sono supportate dalla community di Apache Flink da oltre tre anni. Abbiamo intenzione di rendere obsolete queste versioni in Amazon Managed Service for Apache Flink il 5 novembre 2024. A partire da questa data, non sarà possibile creare nuove applicazioni per queste versioni di Flink. È possibile continuare a eseguire le applicazioni esistenti in questo momento. Puoi aggiornare le tue applicazioni in modo statico utilizzando la funzionalità di aggiornamento delle versioni in loco in Amazon Managed Service for Apache Flink. Per ulteriori informazioni, consulta [Usa gli aggiornamenti di versione sul posto per Apache Flink](#)

Questo argomento contiene una versione del [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#) tutorial che utilizza Apache Flink 1.8.2.

Argomenti

- [Componenti dell'applicazione del servizio gestito per Apache Flink](#)
- [Prerequisiti per il completamento degli esercizi](#)
- [Fase 1: Impostazione di un AWS account e creazione di un utente amministratore](#)
- [Fase 2: configurazione dell' AWS Command Line Interface \(AWS CLI\)](#)
- [Fase 3: creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink](#)
- [Passaggio 4: eliminare le AWS risorse](#)

Componenti dell'applicazione del servizio gestito per Apache Flink

Per elaborare i dati, l'applicazione del servizio gestito per Apache Flink utilizza un'applicazione Java/ Apache Maven o Scala che elabora l'input e produce l'output utilizzando il runtime di Apache Flink.

Un'applicazione del servizio gestito per Apache Flink include i componenti riportati di seguito:

- **Proprietà di runtime:** è possibile utilizzare le proprietà di runtime per configurare l'applicazione senza ricompilare il codice dell'applicazione.
- **Origine:** l'applicazione consuma i dati utilizzando un'origine. Un connettore di origine legge i dati da un flusso di dati Kinesis, da un bucket Amazon S3, ecc. Per ulteriori informazioni, consulta [Aggiungi sorgenti di dati in streaming](#).
- **Operatori:** l'applicazione elabora i dati utilizzando uno o più operatori. Un operatore può trasformare, arricchire o aggregare i dati. Per ulteriori informazioni, consulta [Operatori](#).
- **Sink:** l'applicazione produce dati verso origini esterne utilizzando i sink. Un connettore sink scrive i dati su un flusso di dati Kinesis, un flusso Firehose, un bucket Amazon S3, ecc. Per ulteriori informazioni, consulta [Scrivi dati utilizzando i lavandini](#).

Dopo aver creato, compilato e compresso il codice dell'applicazione, caricherai il pacchetto di codice in un bucket Amazon Simple Storage Service (Amazon S3). Potrai quindi creare un'applicazione del servizio gestito per Apache Flink. Dovrai inserire la posizione del pacchetto di codice, un flusso di dati Kinesis come origine dati di streaming e in genere una posizione di streaming o di file che riceva i dati elaborati dall'applicazione.

Prerequisiti per il completamento degli esercizi

Per completare le fasi in questa guida, è richiesto quanto segue:

- [Java Development Kit \(JDK\) versione 8](#). Imposta la variabile di ambiente JAVA_HOME in modo che punti alla posizione di installazione di JDK.
- Per utilizzare il connettore Apache Flink Kinesis in questo tutorial, devi scaricare e installare Apache Flink. Per informazioni dettagliate, consultare [Utilizzo del connettore di flussi Apache Flink Kinesis con versioni precedenti di Apache Flink](#).
- Ti consigliamo di utilizzare un ambiente di sviluppo (ad esempio [Eclipse Java Neon](#) o [IntelliJ IDEA](#)) per sviluppare e compilare l'applicazione.
- [Client Git](#). Installa il client Git se non lo hai già fatto.

- [Apache Maven Compiler Plugin](#). Maven deve trovarsi nel percorso di lavoro. Per testare l'installazione Apache Maven, immetti quanto segue:

```
$ mvn -version
```

Per iniziare, vai alla pagina [Fase 1: Impostazione di un AWS account e creazione di un utente amministratore](#).

Fase 1: Impostazione di un AWS account e creazione di un utente amministratore

Registrazione per un account Account AWS

Se non si dispone di un Account AWS, completare la procedura seguente per crearne uno.

Per registrarti a un Account AWS

1. Apri la <https://portal.aws.amazon.com/billing/registrazione>.
2. Segui le istruzioni online.

Nel corso della procedura di registrazione riceverai una telefonata, durante la quale sarà necessario inserire un codice di verifica attraverso la tastiera del telefono.

Quando ti iscrivi a un Account AWS, Utente root dell'account AWS viene creato un. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente e utilizza solo l'utente root per eseguire [attività che richiedono l'accesso di un utente root](#).

AWS Al termine del processo di registrazione, riceverai un'e-mail di conferma da. Puoi visualizzare l'attività corrente del tuo account e gestire l'account in qualsiasi momento accedendo a <https://aws.amazon.com/e> scegliendo My Account (Il mio account).

Crea un utente con accesso amministrativo

Dopo aver effettuato la registrazione di un Account AWS, proteggi Utente root dell'account AWS AWS IAM Identity Center, abilita e crea un utente amministratore in modo da non utilizzare l'utente root per le attività quotidiane.

Proteggi i tuoi Utente root dell'account AWS

1. Accedi [AWS Management Console](#) come proprietario dell'account scegliendo Utente root e inserendo il tuo indirizzo Account AWS email. Nella pagina successiva, inserisci la password.

Per informazioni sull'accesso utilizzando un utente root, consulta la pagina [Signing in as the root user](#) della Guida per l'utente di Accedi ad AWS .

2. Abilita l'autenticazione a più fattori (MFA) per l'utente root.

Per ricevere istruzioni, consulta [Abilitare un dispositivo MFA virtuale per l'utente Account AWS root della \(console\)](#) nella Guida per l'utente IAM.

Crea un utente con accesso amministrativo

1. Abilita Centro identità IAM.

Per istruzioni, consulta [Abilitazione di AWS IAM Identity Center](#) nella Guida per l'utente di AWS IAM Identity Center .

2. In IAM Identity Center, assegna l'accesso amministrativo a un utente.

Per un tutorial sull'utilizzo di IAM Identity Center directory come fonte di identità, consulta [Configurare l'accesso utente con l'impostazione predefinita IAM Identity Center directory](#) nella Guida per l'AWS IAM Identity Center utente.

Accesso come utente amministratore

- Per accedere con l'utente IAM Identity Center, utilizza l'URL di accesso che è stato inviato al tuo indirizzo e-mail quando hai creato l'utente IAM Identity Center.

Per assistenza nell'accesso mediante un utente IAM Identity Center, consulta [AWS Accedere al portale di accesso](#) nella Guida per l'Accedi ad AWS utente.

Assegna l'accesso a ulteriori utenti

1. In IAM Identity Center, crea un set di autorizzazioni conforme alla best practice dell'applicazione di autorizzazioni con il privilegio minimo.

Segui le istruzioni riportate nella pagina [Creazione di un set di autorizzazioni](#) nella Guida per l'utente di AWS IAM Identity Center .

2. Assegna al gruppo prima gli utenti e poi l'accesso con autenticazione unica (Single Sign-On).

Per istruzioni, consulta [Aggiungere gruppi](#) nella Guida per l'utente di AWS IAM Identity Center .

Concessione dell'accesso programmatico

Gli utenti hanno bisogno di un accesso programmatico se desiderano interagire con AWS esternamente alla AWS Management Console. La modalità con cui concedere l'accesso programmatico dipende dal tipo di utente che accede ad AWS.

Per fornire agli utenti l'accesso programmatico, scegli una delle seguenti opzioni.

Quale utente necessita dell'accesso programmatico?	Per	Come
Identità della forza lavoro (Utenti gestiti nel centro identità IAM)	Utilizza credenziali temporane e per firmare le richieste programmatiche a AWS CLI,, AWS SDKs o. AWS APIs	Segui le istruzioni per l'interfaccia che desideri utilizzare. • Per la AWS CLI, vedere Configurazione dell'uso AWS IAM Identity Center nella AWS CLI Guida per l'utente.AWS Command Line Interface • Per AWS SDKs gli strumenti e AWS APIs, consulta l'autenticazione di IAM Identity Center nella Guida di riferimento AWS SDKs and Tools.
IAM	Utilizza credenziali temporane e per firmare le richieste programmatiche a AWS CLI, AWS SDKs, o. AWS APIs	Seguendo le istruzioni riportate in Utilizzo delle credenziali temporanee con le

Quale utente necessita dell'accesso programmatico?	Per	Come
		AWS risorse nella Guida per l'utente IAM.
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare richieste programmatiche a AWS CLI,, AWS SDKs o. AWS APIs	Segui le istruzioni per l'interfaccia che desideri utilizzare. • Per la AWS CLI, consulta Autenticazione tramite credenziali utente IAM nella Guida per l'utente.AWS Command Line Interface • Per gli strumenti AWS SDKs e gli strumenti, consulta Autenticazione tramite credenziali a lungo termine nella Guida di riferimento agli strumenti e agli AWS SDKs strumenti. • Per AWS APIs, consulta la sezione Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente IAM.

Fase 2: configurazione dell' AWS Command Line Interface (AWS CLI)

In questa fase, viene scaricato e configurato il file AWS CLI da utilizzare con il servizio gestito per Apache Flink.

Note

Gli esercizi sulle Nozioni di base di questa guida presuppongono che tu disponga di credenziali di amministratore (`adminuser`) nell'account per eseguire le operazioni.

Note

Se la è già AWS CLI installata, potrebbe essere necessario eseguire l'aggiornamento per ottenere le funzionalità più recenti. Per ulteriori informazioni, consulta [Installazione dell' AWS Command Line Interface](#) nella Guida per l'utente dell'AWS Command Line Interface . Per verificare la versione di AWS CLI, esegui il comando seguente:

```
aws --version
```

Gli esercizi in questo tutorial richiedono la seguente AWS CLI versione della o successive:

```
aws-cli/1.16.63
```

Per configurare il AWS CLI

1. Scarica e configura la AWS CLI. Per le istruzioni, consulta i seguenti argomenti nella Guida per l'utente dell'AWS Command Line Interface :
 - [Installazione dell' AWS Command Line Interface](#)
 - [Configurazione della AWS CLI](#)
2. Aggiungi un profilo denominato per l'utente amministratore nel AWS CLI config file. Puoi usare questo profilo quando esegui i comandi della AWS CLI . Per ulteriori informazioni sui profili denominati, consulta [Profili denominati](#) in Guida per l'utente dell'AWS Command Line Interface .

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

Per un elenco di regioni disponibili, consulta [Regioni ed endpoint](#) nella Riferimenti generali di Amazon Web Services.

 Note

Il codice e i comandi di esempio in questo tutorial utilizzano la regione Stati Uniti occidentali (Oregon). Per utilizzare una AWS regione diversa, sostituisci la regione nel codice e nei comandi di questo tutorial con quella che desideri utilizzare.

3. Verifica la configurazione digitando il comando help riportato di seguito al prompt dei comandi:

```
aws help
```

Dopo aver configurato un AWS account e la AWS CLI, puoi provare l'esercizio successivo in cui viene configurata un'applicazione di esempio e testata la end-to-end configurazione.

Approfondimenti

[Fase 3: creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink](#)

Fase 3: creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink

In questo esercizio, viene creata un'applicazione del servizio gestito per Apache Flink con flussi di dati come origine e come sink.

Questa sezione contiene le fasi seguenti:

- [Crea due flussi di dati Amazon Kinesis](#)
- [Scrivi record di esempio nel flusso di input](#)
- [Scarica ed esamina il codice Java per lo streaming di Apache Flink](#)
- [Per creare il codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink](#)
- [Approfondimenti](#)

Crea due flussi di dati Amazon Kinesis

Prima di creare un'applicazione del servizio gestito per Apache Flink per questo esercizio, crea due flussi di dati Kinesis (`ExampleInputStream` e `ExampleOutputStream`). L'applicazione utilizza questi flussi per i flussi di origine e di destinazione dell'applicazione.

Puoi creare questi flussi utilizzando la console Amazon Kinesis o il comando AWS CLI seguente. Per istruzioni sulla console, consulta [Creazione e aggiornamento dei flussi di dati](#) nella Guida per gli sviluppatori del flusso di dati Amazon Kinesis.

Per creare i flussi di dati (AWS CLI)

1. Per creare il primo flusso (`ExampleInputStream`), utilizza il seguente comando Amazon Kinesis `create-stream` AWS CLI .

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

2. Per creare il secondo flusso utilizzato dall'applicazione per scrivere l'output, esegui lo stesso comando, modificando il nome del flusso in `ExampleOutputStream`.

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

Scrivi record di esempio nel flusso di input

In questa sezione, viene utilizzato uno script Python per scrivere record di esempio nel flusso per l'applicazione da elaborare.

Note

Questa sezione richiede [AWS SDK for Python \(Boto\)](#).

1. Crea un file denominato `stock.py` con i seguenti contenuti:

```
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. Successivamente nel tutorial, esegui lo script `stock.py` per inviare dati all'applicazione.

```
$ python stock.py
```

Scarica ed esamina il codice Java per lo streaming di Apache Flink

Il codice dell'applicazione Java per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito:

1. Clona il repository remoto con il comando seguente:

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

2. Passa alla directory `amazon-kinesis-data-analytics-java-examples/GettingStarted_1_8`.

Tieni presente quanto segue riguardo al codice dell'applicazione:

- Un file del [modello di oggetti del progetto \(pom.xml\)](#) contiene le informazioni sulla configurazione e le dipendenze dell'applicazione, incluse le librerie del servizio gestito per Apache Flink.
- Il file `BasicStreamingJob.java` contiene il metodo `main` che definisce la funzionalità dell'applicazione.
- L'applicazione utilizza un'origine Kinesis per leggere dal flusso di origine. Il seguente snippet crea l'origine Kinesis:

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

- L'applicazione crea connettori di origine e sink per accedere alle risorse esterne utilizzando un oggetto `StreamExecutionEnvironment`.
- L'applicazione crea connettori di origine e sink utilizzando proprietà statiche. Per usare proprietà dell'applicazione dinamiche, utilizza i metodi `createSourceFromApplicationProperties` e `createSinkFromApplicationProperties` per creare i connettori. Questi metodi leggono le proprietà dell'applicazione per configurare il connettori.

Per ulteriori informazioni sulle proprietà di runtime, consulta [Usa le proprietà di runtime](#).

Per creare il codice dell'applicazione

In questa sezione, viene utilizzato il compilatore Apache Maven per creare il codice Java per l'applicazione. Per ulteriori informazioni sull'installazione di Apache Maven e Java Development Kit (JDK), consulta [Prerequisiti per il completamento degli esercizi](#).

Note

Per utilizzare il connettore Kinesis con versioni di Apache Flink precedenti alla 1.11, devi scaricare, creare e installare Apache Maven. Per ulteriori informazioni, consulta [the section](#)

called “Utilizzo del connettore di flussi Apache Flink Kinesis con versioni precedenti di Apache Flink”.

Per compilare il codice dell'applicazione

1. Per usare il codice dell'applicazione, compila il codice e comprimilo in un file JAR. È possibile compilare e creare un pacchetto del codice in uno di due modi:
 - Utilizzare lo strumento Maven a riga di comando. Crea il file JAR eseguendo il comando seguente nella directory che contiene il file `pom.xml`:

```
mvn package -Dflink.version=1.8.2
```

- Utilizza il tuo ambiente di sviluppo. Per informazioni dettagliate, consulta la documentazione relativa all'ambiente di sviluppo.

Note

Il codice di origine fornito si basa sulle librerie di Java 1.8. Assicurati che la versione Java del tuo progetto sia la 1.8.

È possibile caricare il pacchetto come un file JAR, oppure comprimere il pacchetto e caricarlo come un file ZIP. Se crei l'applicazione utilizzando AWS CLI, specifica il tipo di contenuto del codice (JAR o ZIP).

2. Se si verificano errori durante la compilazione, verifica che la variabile di ambiente `JAVA_HOME` sia impostata correttamente.

Se l'applicazione viene compilata correttamente, viene creato il seguente file:

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

Caricamento del codice Java di streaming di Apache Flink

In questa sezione, viene creato un bucket Amazon Simple Storage Service (Amazon S3) e caricato il codice dell'applicazione.

Per caricare il codice dell'applicazione

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Seleziona Crea bucket.
3. Inserisci **ka-app-code-*<username>*** nel campo Nome bucket. Aggiungi un suffisso al nome del bucket, ad esempio il nome utente, per renderlo globalmente univoco. Scegli Next (Successivo).
4. Nella fase Configura opzioni, non modificare le impostazioni e scegli Successivo.
5. Nella fase Imposta autorizzazioni, non modificare le impostazioni e scegli Successivo.
6. Seleziona Crea bucket.
7. Nella console Amazon S3, scegli il *<username>* bucket ka-app-code- e scegli Carica.
8. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `aws-kinesis-analytics-java-apps-1.0.jar` creato nella fase precedente. Scegli Next (Successivo).
9. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink

È possibile creare ed eseguire un'applicazione del servizio gestito per Apache Flink utilizzando la console o la AWS CLI.

Note

Quando crei l'applicazione utilizzando la console, vengono create automaticamente le risorse AWS Identity and Access Management (IAM) e di File di CloudWatch log Amazon Logs. Quando crei l'applicazione utilizzando la AWS CLI, queste risorse vengono create separatamente.

Argomenti

- [Creazione ed esecuzione dell'applicazione \(console\)](#)
- [Crea ed esegui l'applicazione \(AWS CLI\)](#)

Creazione ed esecuzione dell'applicazione (console)

Segui questi passaggi per creare, configurare, aggiornare ed eseguire l'applicazione utilizzando la console.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.
3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:
 - Per Nome applicazione, immetti **MyApplication**.
 - Per Descrizione, inserisci **My java test app**.
 - Per Runtime, scegli Apache Flink.
 - Lascia il menu a discesa di Apache Flink 1.8 (versione consigliata).
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
5. Scegli Crea applicazione.

Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: kinesis-analytics-service-*MyApplication-us-west-2*
- Ruolo: kinesisanalytics-*MyApplication-us-west-2*

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere ai flussi di dati Kinesis.

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.

2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
3. Nella pagina Riepilogo, scegli Modifica policy. Scegli la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (**012345678901**) con il tuo ID account.

```
{
  "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"
}
]
}

```

Configurazione dell'applicazione

1. Nella MyApplicationpagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:
 - Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **aws-kinesis-analytics-java-apps-1.0.jar**
3. In Accesso alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
4. Immetti i valori e le proprietà dell'applicazione seguenti:

ID gruppo	Chiave	Valore
ProducerConfigProperties	flink.inputstream.initpos	LATEST
ProducerConfigProperties	aws.region	us-west-2
ProducerConfigProperties	AggregationEnabled	false

5. In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
6. Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.
7. Scegli Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione Amazon, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: /aws/kinesis-analytics/MyApplication
- Flusso di log: kinesis-analytics-log-stream

Esecuzione dell'applicazione.

1. Nella MyApplicationpagina, scegli Esegui. Conferma l'operazione.
2. Quando l'applicazione è in esecuzione, aggiorna la pagina. La console mostra il Grafico dell'applicazione.

Interruzione dell'applicazione

Nella MyApplicationpagina, scegli Stop. Conferma l'operazione.

Aggiornamento dell'applicazione

Tramite la console, puoi aggiornare le impostazioni dell'applicazione, ad esempio le proprietà dell'applicazione, le impostazioni di monitoraggio e la posizione o il nome di file del JAR dell'applicazione. Puoi anche ricaricare il JAR dell'applicazione dal bucket Amazon S3 se è necessario aggiornare il codice dell'applicazione.

Nella MyApplicationpagina, scegli Configura. Aggiorna le impostazioni dell'applicazione e scegli Aggiorna.

Crea ed esegui l'applicazione (AWS CLI)

In questa sezione, verrà utilizzata la AWS CLI per creare ed eseguire il servizio gestito per Apache Flink. Il servizio gestito per Apache Flink utilizza il `kinesisanalyticsv2` AWS CLI comando per creare e interagire con le applicazioni del servizio gestito per Apache Flink.

Creazione di una policy di autorizzazione

Note

È necessario creare una policy di autorizzazione e un ruolo per l'applicazione. Se non si creano queste risorse IAM, l'applicazione non può accedere ai suoi dati e flussi di log.

Innanzitutto, crea una policy di autorizzazione con due istruzioni: una che concede le autorizzazioni per l'operazione `read` sul flusso di origine e un'altra che concede le autorizzazioni per operazioni `write` sul flusso di sink. Collega quindi la policy a un ruolo IAM (che verrà creato nella sezione successiva). Pertanto, quando il servizio gestito per Apache Flink assume il ruolo, il servizio disporrà delle autorizzazioni necessarie per leggere dal flusso di origine e scrivere nel flusso di sink.

Utilizza il codice seguente per creare la policy di autorizzazione `AKReadSourceStreamWriteSinkStream`. Sostituisci *username* con il nome utente utilizzato per creare il bucket Amazon S3 per archiviare il codice dell'applicazione. Sostituisci l'ID account nei nomi della risorsa Amazon (ARNs) (*012345678901*) con il tuo ID account.

```
{
  "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"
    }
]
}

```

Per step-by-step istruzioni su come creare una politica di autorizzazioni, consulta il [Tutorial: Create and Attach Your First Customer Managed Policy](#) nella IAM User Guide.

Note

Per accedere ad altri servizi Amazon, puoi utilizzare AWS SDK per Java. Il servizio gestito per Apache Flink imposta automaticamente le credenziali richieste dall'SDK su quelle del ruolo IAM di esecuzione del servizio associato all'applicazione. Non sono richieste fasi aggiuntive.

Creazione di un ruolo IAM

In questa sezione, viene creato un ruolo IAM per l'applicazione del servizio gestito per Apache Flink che può essere assunto per leggere un flusso di origine e scrivere nel flusso di sink.

Il servizio gestito per Apache Flink non può accedere al tuo flusso senza autorizzazioni. Queste autorizzazioni possono essere assegnate con un ruolo IAM. Ad ogni ruolo IAM sono collegate due policy. La policy di attendibilità concede al servizio gestito per Apache Flink l'autorizzazione per assumere il ruolo e la policy di autorizzazione determina cosa può fare il servizio assumendo questo ruolo.

Collega la policy di autorizzazione creata nella sezione precedente a questo ruolo.

Per creare un ruolo IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nel riquadro di navigazione, seleziona Ruoli, quindi Crea nuovo ruolo.
3. In Seleziona tipo di identità attendibile, scegli Servizio AWS . In Scegli il servizio che utilizzerà questo ruolo, scegli Kinesis. In Seleziona il tuo caso d'uso, scegli Analisi dei dati Kinesis.

Scegli Successivo: Autorizzazioni.

4. Nella pagina Allega policy di autorizzazione, seleziona Successivo: esamina. Collega le policy di autorizzazione dopo aver creato il ruolo.
5. Nella pagina Crea ruolo, immetti **MF-stream-rw-role** per Nome ruolo. Scegli Crea ruolo.

È stato creato un nuovo ruolo IAM denominato MF-stream-rw-role. Successivamente, aggiorna le policy di attendibilità e di autorizzazione per il ruolo.

6. Collega la policy di autorizzazione al ruolo.

Note

Per questo esercizio, il servizio gestito per Apache Flink assume questo ruolo per la lettura di dati da un flusso di dati Kinesis (origine) e la scrittura dell'output in un altro flusso di dati Kinesis. Pertanto, devi collegare la policy creata nella fase precedente, [the section called “Creazione di una policy di autorizzazione”](#).

- a. Nella pagina Riepilogo, scegli la scheda Autorizzazioni.
- b. Scegliere Collega policy.
- c. Nella casella di ricerca, immetti **AKReadSourceStreamWriteSinkStream** (la policy creata nella sezione precedente).
- d. Scegli la policy AKReadSourceStreamWriteSinkStream e seleziona Collega policy.

È stato creato il ruolo di esecuzione del servizio che l'applicazione utilizzerà per accedere alle risorse. Prendi nota dell'ARN del nuovo ruolo.

Per step-by-step istruzioni sulla creazione di un ruolo, consulta [Creating an IAM Role \(Console\)](#) nella IAM User Guide.

Creazione dell'applicazione del servizio gestito per Apache Flink

1. Salvare il seguente codice JSON in un file denominato `create_request.json`. Sostituisci l'ARN del ruolo di esempio con l'ARN per il ruolo creato in precedenza. Sostituisci il suffisso dell'ARN del bucket (*username*) con il suffisso scelto nella sezione precedente. Sostituisci l'ID account di esempio (*012345678901*) nel ruolo di esecuzione del servizio con il tuo ID account.

```
{
  "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. Esegui l'operazione [CreateApplication](#) con la richiesta precedente per creare l'applicazione:

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

L'applicazione è ora creata. Avvia l'applicazione nella fase successiva.

Avvia l'applicazione

In questa sezione, viene utilizzata l'operazione [StartApplication](#) per avviare l'applicazione.

Per avviare l'applicazione

1. Salvare il seguente codice JSON in un file denominato `start_request.json`.

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

2. Esegui l'operazione [StartApplication](#) con la richiesta precedente per avviare l'applicazione:

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

L'applicazione è ora in esecuzione. Puoi controllare i parametri del servizio gestito per Apache Flink sulla CloudWatch console Amazon per verificare che l'applicazione funzioni.

Interruzione dell'applicazione

In questa sezione, viene utilizzata l'operazione [StopApplication](#) per interrompere l'applicazione.

Per interrompere l'applicazione

1. Salvare il seguente codice JSON in un file denominato `stop_request.json`.

```
{
  "ApplicationName": "test"
}
```

2. Esegui l'operazione [StopApplication](#) con la seguente richiesta di interrompere l'applicazione:

```
aws kinesisanalyticstv2 stop-application --cli-input-json file://stop_request.json
```

L'applicazione è ora interrotta.

Aggiungi un'opzione CloudWatch di registrazione

Puoi utilizzare la AWS CLI per aggiungere un flusso di CloudWatch log Amazon all'applicazione.

Per ulteriori informazioni sull'utilizzo CloudWatch dei log con la tua applicazione, consulta [the section called "Configurare la registrazione delle applicazioni in Managed Service per Apache Flink"](#).

Aggiornamento delle proprietà dell'ambiente

In questa sezione, viene utilizzata l'operazione [UpdateApplication](#) per modificare le proprietà di ambiente per l'applicazione senza ricompilare il codice dell'applicazione. In questo esempio, viene modificata la regione dei flussi di origine e destinazione.

Per aggiornare le proprietà di ambiente per l'applicazione

1. Salvare il seguente codice JSON in un file denominato `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. Esegui l'operazione [UpdateApplication](#) con la richiesta precedente per aggiornare le proprietà di ambiente:

```
aws kinesisanalyticstv2 update-application --cli-input-json file://
update_properties_request.json
```

Aggiornamento del codice dell'applicazione

Quando devi aggiornare il codice dell'applicazione con una nuova versione del pacchetto di codice, utilizza l'[UpdateApplication](#) AWS CLI operazione.

Note

Per caricare una nuova versione del codice dell'applicazione con lo stesso nome file, è necessario specificare la nuova versione dell'oggetto. Per ulteriori informazioni sull'uso delle versioni degli oggetti Amazon S3, consulta [Abilitazione o disattivazione del controllo delle versioni](#).

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione ed effettua la chiamata a Amazon S3 e lo stesso nome dell'oggetto `UpdateApplication`, nonché la nuova versione dell'oggetto. L'applicazione verrà riavviata con il nuovo pacchetto di codice.

Il seguente esempio di richiesta per l'operazione `UpdateApplication` ricarica il codice dell'applicazione e riavvia l'applicazione. Aggiorna l'`CurrentApplicationVersionId` alla versione

corrente dell'applicazione. Puoi controllare la versione corrente dell'applicazione utilizzando le operazioni `ListApplications` o `DescribeApplication`. Aggiorna il suffisso del nome del bucket (`<username>`) con il suffisso che hai scelto nella sezione. [the section called “Crea due flussi di dati Amazon Kinesis”](#)

```
{
  "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": "SAMPLEUehYngP87ex1nzYIGYgfhyvpvDU"
        }
      }
    }
  }
}
```

Approfondimenti

[Passaggio 4: eliminare le AWS risorse](#)

Passaggio 4: eliminare le AWS risorse

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Nozioni di base.

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione dei flussi di dati Kinesis](#)
- [Eliminazione dell'oggetto e del bucket Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com /kines.](https://console.aws.amazon.com/kines)

2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Scegli Configura.
4. Nella sezione Snapshot, scegli Disabilita, quindi scegli Aggiorna.
5. Nella pagina dell'applicazione, scegli Elimina e conferma l'eliminazione.

Eliminazione dei flussi di dati Kinesis

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nel pannello del flusso di dati Kinesis, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.
4. Nella pagina Kinesis Streams, scegli, scegli Azioni ExampleOutputStream, scegli Elimina, quindi conferma l'eliminazione.

Eliminazione dell'oggetto e del bucket Amazon S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il **<username>**secchio ka-app-code -.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

Eliminazione delle risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-west-2.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytics--us-west-2. MyApplication
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Quindi scegli Elimina gruppo di log e conferma l'eliminazione.

Nozioni di base: Flink 1.6.2 - deprecazione

Note

Le versioni 1.6, 1.8 e 1.11 di Apache Flink non sono supportate dalla community di Apache Flink da oltre tre anni. Abbiamo intenzione di rendere obsolete queste versioni in Amazon Managed Service for Apache Flink il 5 novembre 2024. A partire da questa data, non sarà possibile creare nuove applicazioni per queste versioni di Flink. È possibile continuare a eseguire le applicazioni esistenti in questo momento. Puoi aggiornare le tue applicazioni in modo statico utilizzando la funzionalità di aggiornamento delle versioni in loco in Amazon Managed Service for Apache Flink. Per ulteriori informazioni, consulta [Usa gli aggiornamenti di versione sul posto per Apache Flink](#)

Questo argomento contiene una versione del [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#) tutorial che utilizza Apache Flink 1.6.2.

Argomenti

- [Componenti dell'applicazione del servizio gestito per Apache Flink](#)
- [Prerequisiti per il completamento degli esercizi](#)
- [Fase 1: Impostazione di un AWS account e creazione di un utente amministratore](#)
- [Fase 2: configurazione dell' AWS Command Line Interface \(AWS CLI\)](#)
- [Fase 3: creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink](#)
- [Passaggio 4: eliminare le AWS risorse](#)

Componenti dell'applicazione del servizio gestito per Apache Flink

Per elaborare i dati, l'applicazione del servizio gestito per Apache Flink utilizza un'applicazione Java/ Apache Maven o Scala che elabora l'input e produce l'output utilizzando il runtime di Apache Flink.

Un servizio gestito per Apache Flink include i seguenti componenti:

- **Proprietà di runtime:** è possibile utilizzare le proprietà di runtime per configurare l'applicazione senza ricompilare il codice dell'applicazione.
- **Origine:** l'applicazione consuma i dati utilizzando un'origine. Un connettore di origine legge i dati da un flusso di dati Kinesis, da un bucket Amazon S3, ecc. Per ulteriori informazioni, consulta [Aggiungi sorgenti di dati in streaming](#).
- **Operatori:** l'applicazione elabora i dati utilizzando uno o più operatori. Un operatore può trasformare, arricchire o aggregare i dati. Per ulteriori informazioni, consulta [Operatori](#).
- **Sink:** l'applicazione produce dati verso origini esterne utilizzando i sink. Un connettore sink scrive i dati su un flusso di dati Kinesis, un flusso Firehose, un bucket Amazon S3, ecc. Per ulteriori informazioni, consulta [Scrivi dati utilizzando i lavandini](#).

Dopo aver creato, compilato e compresso l'applicazione, caricherai il pacchetto di codice in un bucket Amazon Simple Storage Service (Amazon S3). Potrai quindi creare un'applicazione del servizio gestito per Apache Flink. Dovrai inserire la posizione del pacchetto di codice, un flusso di dati Kinesis come origine dati di streaming e in genere una posizione di streaming o di file che riceva i dati elaborati dall'applicazione.

Prerequisiti per il completamento degli esercizi

Per completare le fasi in questa guida, è richiesto quanto segue:

- [Java Development Kit \(JDK\) versione 8](#). Imposta la variabile di ambiente JAVA_HOME in modo che punti alla posizione di installazione di JDK.
- Ti consigliamo di utilizzare un ambiente di sviluppo (ad esempio [Eclipse Java Neon](#) o [IntelliJ IDEA](#)) per sviluppare e compilare l'applicazione.
- [Client Git](#). Installa il client Git se non lo hai già fatto.
- [Apache Maven Compiler Plugin](#). Maven deve trovarsi nel percorso di lavoro. Per testare l'installazione Apache Maven, immetti quanto segue:

```
$ mvn -version
```

Per iniziare, vai alla pagina [Fase 1: Impostazione di un AWS account e creazione di un utente amministratore](#).

Fase 1: Impostazione di un AWS account e creazione di un utente amministratore

Registrazione per un account Account AWS

Se non si dispone di un Account AWS, completare la procedura seguente per crearne uno.

Per registrarti a un Account AWS

1. Apri la <https://portal.aws.amazon.com/billing/registrazione>.
2. Segui le istruzioni online.

Nel corso della procedura di registrazione riceverai una telefonata, durante la quale sarà necessario inserire un codice di verifica attraverso la tastiera del telefono.

Quando ti iscrivi a un Account AWS, Utente root dell'account AWS viene creato un. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente e utilizza solo l'utente root per eseguire [attività che richiedono l'accesso di un utente root](#).

AWS Al termine del processo di registrazione, riceverai un'e-mail di conferma da. Puoi visualizzare l'attività corrente del tuo account e gestire l'account in qualsiasi momento accedendo a <https://aws.amazon.com/e> scegliendo My Account (Il mio account).

Crea un utente con accesso amministrativo

Dopo aver effettuato la registrazione di un Account AWS, proteggi Utente root dell'account AWS AWS IAM Identity Center, abilita e crea un utente amministratore in modo da non utilizzare l'utente root per le attività quotidiane.

Proteggi i tuoi Utente root dell'account AWS

1. Accedi [AWS Management Console](#) come proprietario dell'account scegliendo Utente root e inserendo il tuo indirizzo Account AWS email. Nella pagina successiva, inserisci la password.

Per informazioni sull'accesso utilizzando un utente root, consulta la pagina [Signing in as the root user](#) della Guida per l'utente di Accedi ad AWS .

2. Abilita l'autenticazione a più fattori (MFA) per l'utente root.

Per ricevere istruzioni, consulta [Abilitare un dispositivo MFA virtuale per l'utente Account AWS root della \(console\)](#) nella Guida per l'utente IAM.

Crea un utente con accesso amministrativo

1. Abilita Centro identità IAM.

Per istruzioni, consulta [Abilitazione di AWS IAM Identity Center](#) nella Guida per l'utente di AWS IAM Identity Center .

2. In IAM Identity Center, assegna l'accesso amministrativo a un utente.

Per un tutorial sull'utilizzo di IAM Identity Center directory come fonte di identità, consulta [Configurare l'accesso utente con l'impostazione predefinita IAM Identity Center directory](#) nella Guida per l'AWS IAM Identity Center utente.

Accesso come utente amministratore

- Per accedere con l'utente IAM Identity Center, utilizza l'URL di accesso che è stato inviato al tuo indirizzo e-mail quando hai creato l'utente IAM Identity Center.

Per assistenza nell'accesso mediante un utente IAM Identity Center, consulta [AWS Accedere al portale di accesso](#) nella Guida per l'Accedi ad AWS utente.

Assegna l'accesso a ulteriori utenti

1. In IAM Identity Center, crea un set di autorizzazioni conforme alla best practice dell'applicazione di autorizzazioni con il privilegio minimo.

Segui le istruzioni riportate nella pagina [Creazione di un set di autorizzazioni](#) nella Guida per l'utente di AWS IAM Identity Center .

2. Assegna al gruppo prima gli utenti e poi l'accesso con autenticazione unica (Single Sign-On).

Per istruzioni, consulta [Aggiungere gruppi](#) nella Guida per l'utente di AWS IAM Identity Center .

Concessione dell'accesso programmatico

Gli utenti hanno bisogno di un accesso programmatico se desiderano interagire con AWS esternamente alla AWS Management Console. La modalità con cui concedere l'accesso programmatico dipende dal tipo di utente che accede ad AWS.

Per fornire agli utenti l'accesso programmatico, scegli una delle seguenti opzioni.

Quale utente necessita dell'accesso programmatico?	Per	Come
Identità della forza lavoro (Utenti gestiti nel centro identità IAM)	Utilizza credenziali temporane e per firmare le richieste programmatiche a AWS CLI,, AWS SDKs o. AWS APIs	Segui le istruzioni per l'interfaccia che desideri utilizzare. - Per la AWS CLI, vedere Configurazione dell'uso AWS IAM Identity Center nella AWS CLI Guida per l'utente. AWS Command Line Interface - Per AWS SDKs gli strumenti e AWS APIs, consulta l'autenticazione di IAM Identity Center nella Guida di riferimento AWS SDKs and Tools.
IAM	Utilizza credenziali temporane e per firmare le richieste programmatiche a AWS CLI, AWS SDKs, o. AWS APIs	Seguendo le istruzioni riportate in Utilizzo delle credenziali temporanee con le AWS risorse nella Guida per l'utente IAM .
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare richieste programmatiche a AWS CLI,, AWS SDKs o. AWS APIs	Segui le istruzioni per l'interfaccia che desideri utilizzare. Per la AWS CLI, consulta Autenticazione tramite

Quale utente necessita dell'accesso programmatico?	Per	Come
		credenziali utente IAM nella Guida per l'utente.AWS Command Line Interface • Per gli strumenti AWS SDKs e gli strumenti, consulta Autenticazione tramite credenziali a lungo termine nella Guida di riferimento agli strumenti e agli AWS SDKs strumenti. • Per AWS APIs, consulta la sezione Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente IAM.

Fase 2: configurazione dell' AWS Command Line Interface (AWS CLI)

In questa fase, viene scaricato e configurato il file AWS CLI da utilizzare con il servizio gestito per Apache Flink.

Note

Gli esercizi sulle Nozioni di base di questa guida presuppongono che tu disponga di credenziali di amministratore (`adminuser`) nell'account per eseguire le operazioni.

Note

Se la è già AWS CLI installata, potrebbe essere necessario eseguire l'aggiornamento per ottenere le funzionalità più recenti. Per ulteriori informazioni, consulta [Installazione dell' AWS Command Line Interface](#) nella Guida per l'utente dell'AWS Command Line Interface . Per verificare la versione di AWS CLI, esegui il comando seguente:

```
aws --version
```

Gli esercizi in questo tutorial richiedono la seguente AWS CLI versione della o successive:

```
aws-cli/1.16.63
```

Per configurare il AWS CLI

1. Scarica e configura la AWS CLI. Per le istruzioni, consulta i seguenti argomenti nella Guida per l'utente dell'AWS Command Line Interface :
 - [Installazione dell' AWS Command Line Interface](#)
 - [Configurazione della AWS CLI](#)
2. Aggiungi un profilo denominato per l'utente amministratore nel AWS CLI config file. Puoi usare questo profilo quando esegui i comandi della AWS CLI . Per ulteriori informazioni sui profili denominati, consulta [Profili denominati](#) in Guida per l'utente dell'AWS Command Line Interface .

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

Per un elenco delle AWS regioni disponibili, consulta [Regioni ed endpoint](#) in. Riferimenti generali di Amazon Web Services

Note

Il codice e i comandi di esempio in questo tutorial utilizzano la regione Stati Uniti occidentali (Oregon). Per utilizzare una regione diversa, sostituisci la regione nel codice e nei comandi di questo tutorial con quella che desideri utilizzare.

3. Verifica la configurazione digitando il comando help riportato di seguito al prompt dei comandi:

```
aws help
```

Dopo aver configurato un AWS account e la AWS CLI, puoi provare l'esercizio successivo in cui viene configurata un'applicazione di esempio e testata la end-to-end configurazione.

Approfondimenti

[Fase 3: creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink](#)

Fase 3: creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink

In questo esercizio, viene creata un'applicazione del servizio gestito per Apache Flink con flussi di dati come origine e come sink.

Questa sezione contiene le fasi seguenti:

- [Crea due flussi di dati Amazon Kinesis](#)
- [Scrivi record di esempio nel flusso di input](#)
- [Scarica ed esamina il codice Java per lo streaming di Apache Flink](#)
- [Per creare il codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink](#)

Crea due flussi di dati Amazon Kinesis

Prima di creare un'applicazione del servizio gestito per Apache Flink per questo esercizio, crea due flussi di dati Kinesis (`ExampleInputStream` e `ExampleOutputStream`). L'applicazione utilizza questi flussi per i flussi di origine e di destinazione dell'applicazione.

Puoi creare questi flussi utilizzando la console Amazon Kinesis o il comando AWS CLI seguente. Per istruzioni sulla console, consulta [Creazione e aggiornamento dei flussi di dati](#) nella Guida per gli sviluppatori del flusso di dati Amazon Kinesis.

Per creare i flussi di dati (AWS CLI)

1. Per creare il primo flusso (`ExampleInputStream`), utilizza il seguente comando Amazon Kinesis `create-stream` AWS CLI .

```
$ aws kinesis create-stream \
--stream-name ExampleInputStream \
--shard-count 1 \
```

```
--region us-west-2 \  
--profile adminuser
```

2. Per creare il secondo flusso utilizzato dall'applicazione per scrivere l'output, esegui lo stesso comando, modificando il nome del flusso in `ExampleOutputStream`.

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

Scrivi record di esempio nel flusso di input

In questa sezione, viene utilizzato uno script Python per scrivere record di esempio nel flusso per l'applicazione da elaborare.

 Note

Questa sezione richiede [AWS SDK for Python \(Boto\)](#).

1. Crea un file denominato `stock.py` con i seguenti contenuti:

```
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. Successivamente nel tutorial, esegui lo script `stock.py` per inviare dati all'applicazione.

```
$ python stock.py
```

Scarica ed esamina il codice Java per lo streaming di Apache Flink

Il codice dell'applicazione Java per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito:

1. Clona il repository remoto con il comando seguente:

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

2. Passa alla directory `amazon-kinesis-data-analytics-java-examples/GettingStarted_1_6`.

Tieni presente quanto segue riguardo al codice dell'applicazione:

- Un file del [modello di oggetti del progetto \(pom.xml\)](#) contiene le informazioni sulla configurazione e le dipendenze dell'applicazione, incluse le librerie del servizio gestito per Apache Flink.
- Il file `BasicStreamingJob.java` contiene il metodo `main` che definisce la funzionalità dell'applicazione.
- L'applicazione utilizza un'origine Kinesis per leggere dal flusso di origine. Il seguente snippet crea l'origine Kinesis:

```
return env.addSource(new FlinkKinesisConsumer<>(inputStreamName,
```

```
new SimpleStringSchema(), inputProperties));
```

- L'applicazione crea connettori di origine e sink per accedere alle risorse esterne utilizzando un oggetto `StreamExecutionEnvironment`.
- L'applicazione crea connettori di origine e sink utilizzando proprietà statiche. Per usare proprietà dell'applicazione dinamiche, utilizza i metodi `createSourceFromApplicationProperties` e `createSinkFromApplicationProperties` per creare i connettori. Questi metodi leggono le proprietà dell'applicazione per configurare il connettore.

Per ulteriori informazioni sulle proprietà di runtime, consulta [Usa le proprietà di runtime](#).

Per creare il codice dell'applicazione

In questa sezione, viene utilizzato il compilatore Apache Maven per creare il codice Java per l'applicazione. Per ulteriori informazioni sull'installazione di Apache Maven e Java Development Kit (JDK), consulta [Prerequisiti per il completamento degli esercizi](#).

Note

Per utilizzare il connettore Kinesis con versioni di Apache Flink precedenti alla 1.11, è necessario scaricare il codice di origine per il connettore e compilarlo come descritto nella [documentazione di Apache Flink](#).

Per compilare il codice dell'applicazione

1. Per usare il codice dell'applicazione, compila il codice e comprimilo in un file JAR. È possibile compilare e creare un pacchetto del codice in uno di due modi:
 - Utilizzare lo strumento Maven a riga di comando. Crea il file JAR eseguendo il comando seguente nella directory che contiene il file `pom.xml`:

```
mvn package
```

Note

Il parametro `-DFlink.version` non è richiesto per il servizio gestito per il runtime di Apache Flink versione 1.0.1; è richiesto solo per la versione 1.1.0 e successive. Per

ulteriori informazioni, consulta [the section called “Specify the version of Apache Flink application”](#).

- Utilizza il tuo ambiente di sviluppo. Per informazioni dettagliate, consulta la documentazione relativa all'ambiente di sviluppo.

È possibile caricare il pacchetto come un file JAR, oppure comprimere il pacchetto e caricarlo come un file ZIP. Se crei l'applicazione utilizzando AWS CLI, specifica il tipo di contenuto del codice (JAR o ZIP).

2. Se si verificano errori durante la compilazione, verifica che la variabile di ambiente `JAVA_HOME` sia impostata correttamente.

Se l'applicazione viene compilata correttamente, viene creato il seguente file:

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

Caricamento del codice Java di streaming di Apache Flink

In questa sezione, viene creato un bucket Amazon Simple Storage Service (Amazon S3) e caricato il codice dell'applicazione.

Per caricare il codice dell'applicazione

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Seleziona Crea bucket.
3. Inserisci **ka-app-code-*<username>*** nel campo Nome bucket. Aggiungi un suffisso al nome del bucket, ad esempio il nome utente, per renderlo globalmente univoco. Scegli Next (Successivo).
4. Nella fase Configura opzioni, non modificare le impostazioni e scegli Successivo.
5. Nella fase Imposta autorizzazioni, non modificare le impostazioni e scegli Successivo.
6. Seleziona Crea bucket.
7. Nella console Amazon S3, scegli il ***<username>*** bucket ka-app-code- e scegli Carica.
8. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `aws-kinesis-analytics-java-apps-1.0.jar` creato nella fase precedente. Scegli Next (Successivo).
9. Nella fase Imposta autorizzazioni, non modificare le impostazioni. Scegli Next (Successivo).

10. Nella fase Imposta proprietà, non modificare le impostazioni. Scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink

È possibile creare ed eseguire un'applicazione del servizio gestito per Apache Flink utilizzando la console o la AWS CLI.

Note

Quando crei l'applicazione utilizzando la console, vengono create automaticamente le risorse AWS Identity and Access Management (IAM) e di File di CloudWatch log Amazon Logs. Quando crei l'applicazione utilizzando la AWS CLI, queste risorse vengono create separatamente.

Argomenti

- [Creazione ed esecuzione dell'applicazione \(console\)](#)
- [Crea ed esegui l'applicazione \(AWS CLI\)](#)

Creazione ed esecuzione dell'applicazione (console)

Segui questi passaggi per creare, configurare, aggiornare ed eseguire l'applicazione utilizzando la console.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.
3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:
 - Per Nome applicazione, immetti **MyApplication**.
 - Per Descrizione, inserisci **My java test app**.
 - Per Runtime, scegli Apache Flink.

 Note

Il servizio gestito per Apache Flink utilizza Apache Flink versione 1.8.2 o 1.6.2.

- Cambia il menu a discesa della versione in quello di Apache Flink 1.6.
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
 5. Scegli Crea applicazione.

 Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`
- Ruolo: `kinesisanalytics-MyApplication-us-west-2`

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere ai flussi di dati Kinesis.

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
3. Nella pagina Riepilogo, scegli Modifica policy. Scegli la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (*012345678901*) con il tuo ID account.

```
{
  "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"
    }
]
}

```

Configurazione dell'applicazione

1. Nella MyApplication pagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:
 - Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **java-getting-started-1.0.jar**
3. In Accesso alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
4. Immetti i valori e le proprietà dell'applicazione seguenti:

ID gruppo	Chiave	Valore
ProducerConfigProperties	flink.inputstream.initpos	LATEST
ProducerConfigProperties	aws.region	us-west-2
ProducerConfigProperties	AggregationEnabled	false

5. In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
6. Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.

7. Scegli Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione Amazon, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: `/aws/kinesis-analytics/MyApplication`
- Flusso di log: `kinesis-analytics-log-stream`

Esecuzione dell'applicazione.

1. Nella MyApplicationpagina, scegli Esegui. Conferma l'operazione.
2. Quando l'applicazione è in esecuzione, aggiorna la pagina. La console mostra il Grafico dell'applicazione.

Interruzione dell'applicazione

Nella MyApplicationpagina, scegli Stop. Conferma l'operazione.

Aggiornamento dell'applicazione

Tramite la console, puoi aggiornare le impostazioni dell'applicazione, ad esempio le proprietà dell'applicazione, le impostazioni di monitoraggio e la posizione o il nome di file del JAR dell'applicazione. Puoi anche ricaricare il JAR dell'applicazione dal bucket Amazon S3 se è necessario aggiornare il codice dell'applicazione.

Nella MyApplicationpagina, scegli Configura. Aggiorna le impostazioni dell'applicazione e scegli Aggiorna.

Crea ed esegui l'applicazione (AWS CLI)

In questa sezione, verrà utilizzata la AWS CLI per creare ed eseguire il servizio gestito per Apache Flink. Il servizio gestito per Apache Flink utilizza il `kinesisanalyticsv2` AWS CLI comando per creare e interagire con le applicazioni del servizio gestito per Apache Flink.

Creazione di una policy di autorizzazione

Innanzitutto, crea una policy di autorizzazione con due istruzioni: una che concede le autorizzazioni per l'operazione `read` sul flusso di origine e un'altra che concede le autorizzazioni per operazioni `write` sul flusso di sink. Collega quindi la policy a un ruolo IAM (che verrà creato nella sezione successiva). Pertanto, quando il servizio gestito per Apache Flink assume il ruolo, il servizio disporrà delle autorizzazioni necessarie per leggere dal flusso di origine e scrivere nel flusso di sink.

Utilizza il codice seguente per creare la policy di autorizzazione

`AKReadSourceStreamWriteSinkStream`. Sostituisci *username* con il nome utente utilizzato per creare il bucket Amazon S3 per archiviare il codice dell'applicazione. Sostituisci l'ID account nei nomi della risorsa Amazon (ARNs) (*012345678901*) con il tuo ID account.

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

```
}
```

Per step-by-step istruzioni su come creare una politica di autorizzazioni, consulta il [Tutorial: Create and Attach Your First Customer Managed Policy](#) nella IAM User Guide.

Note

Per accedere ad altri servizi Amazon, puoi utilizzare AWS SDK per Java. Il servizio gestito per Apache Flink imposta automaticamente le credenziali richieste dall'SDK su quelle del ruolo IAM di esecuzione del servizio associato all'applicazione. Non sono richieste fasi aggiuntive.

Creazione di un ruolo IAM

In questa sezione, viene creato un ruolo IAM per l'applicazione del servizio gestito per Apache Flink che può essere assunto per leggere un flusso di origine e scrivere nel flusso di sink.

Il servizio gestito per Apache Flink non può accedere al tuo flusso senza autorizzazioni. Queste autorizzazioni possono essere assegnate con un ruolo IAM. Ad ogni ruolo IAM sono collegate due policy. La policy di attendibilità concede al servizio gestito per Apache Flink l'autorizzazione per assumere il ruolo e la policy di autorizzazione determina cosa può fare il servizio assumendo questo ruolo.

Collega la policy di autorizzazione creata nella sezione precedente a questo ruolo.

Per creare un ruolo IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nel riquadro di navigazione, seleziona Ruoli, quindi Crea nuovo ruolo.
3. In Seleziona tipo di identità attendibile, scegli Servizio AWS . In Scegli il servizio che utilizzerà questo ruolo, scegli Kinesis. In Seleziona il tuo caso d'uso, scegli Analisi dei dati Kinesis.

Scegli Successivo: Autorizzazioni.

4. Nella pagina Allega policy di autorizzazione, seleziona Successivo: esamina. Collega le policy di autorizzazione dopo aver creato il ruolo.
5. Nella pagina Crea ruolo, immetti **MF-stream-rw-role** per Nome ruolo. Scegli Crea ruolo.

È stato creato un nuovo ruolo IAM denominato `MF-stream-rw-role`. Successivamente, aggiorna le policy di attendibilità e di autorizzazione per il ruolo.

6. Collega la policy di autorizzazione al ruolo.

 Note

Per questo esercizio, il servizio gestito per Apache Flink assume questo ruolo per la lettura di dati da un flusso di dati Kinesis (origine) e la scrittura dell'output in un altro flusso di dati Kinesis. Pertanto, devi collegare la policy creata nella fase precedente, [the section called “Creazione di una policy di autorizzazione”](#).

- a. Nella pagina Riepilogo, scegli la scheda Autorizzazioni.
- b. Scegliere Collega policy.
- c. Nella casella di ricerca, immetti **AKReadSourceStreamWriteSinkStream** (la policy creata nella sezione precedente).
- d. Scegli la policy **AKReadSourceStreamWriteSinkStream** e seleziona Collega policy.

È stato creato il ruolo di esecuzione del servizio che l'applicazione utilizzerà per accedere alle risorse. Prendi nota dell'ARN del nuovo ruolo.

Per step-by-step istruzioni sulla creazione di un ruolo, consulta [Creating an IAM Role \(Console\)](#) nella IAM User Guide.

Creazione dell'applicazione del servizio gestito per Apache Flink

1. Salvare il seguente codice JSON in un file denominato `create_request.json`. Sostituisci l'ARN del ruolo di esempio con l'ARN per il ruolo creato in precedenza. Sostituisci il suffisso dell'ARN del bucket (*username*) con il suffisso scelto nella sezione precedente. Sostituisci l'ID account di esempio (*012345678901*) nel ruolo di esecuzione del servizio con il tuo ID account.

```
{
  "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. Esegui l'operazione [CreateApplication](#) con la richiesta precedente per creare l'applicazione:

```

aws kinesisanalyticstv2 create-application --cli-input-json file://
create_request.json

```

L'applicazione è ora creata. Avvia l'applicazione nella fase successiva.

Avvia l'applicazione

In questa sezione, viene utilizzata l'operazione [StartApplication](#) per avviare l'applicazione.

Per avviare l'applicazione

1. Salvare il seguente codice JSON in un file denominato `start_request.json`.

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

2. Esegui l'operazione [StartApplication](#) con la richiesta precedente per avviare l'applicazione:

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

L'applicazione è ora in esecuzione. Puoi controllare i parametri del servizio gestito per Apache Flink sulla CloudWatch console Amazon per verificare che l'applicazione funzioni.

Interruzione dell'applicazione

In questa sezione, viene utilizzata l'operazione [StopApplication](#) per interrompere l'applicazione.

Per interrompere l'applicazione

1. Salvare il seguente codice JSON in un file denominato `stop_request.json`.

```
{
  "ApplicationName": "test"
}
```

2. Esegui l'operazione [StopApplication](#) con la seguente richiesta di interrompere l'applicazione:

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

L'applicazione è ora interrotta.

Aggiungi un'opzione CloudWatch di registrazione

Puoi utilizzare la AWS CLI per aggiungere un flusso di CloudWatch log Amazon all'applicazione. Per ulteriori informazioni sull'utilizzo CloudWatch dei log con la tua applicazione, consulta [the section called "Configurare la registrazione delle applicazioni in Managed Service per Apache Flink"](#).

Aggiornamento delle proprietà dell'ambiente

In questa sezione, viene utilizzata l'operazione [UpdateApplication](#) per modificare le proprietà di ambiente per l'applicazione senza ricompilare il codice dell'applicazione. In questo esempio, viene modificata la regione dei flussi di origine e destinazione.

Per aggiornare le proprietà di ambiente per l'applicazione

1. Salvare il seguente codice JSON in un file denominato `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. Esegui l'operazione [UpdateApplication](#) con la richiesta precedente per aggiornare le proprietà di ambiente:

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

Aggiornamento del codice dell'applicazione

Quando devi aggiornare il codice dell'applicazione con una nuova versione del pacchetto di codice, utilizza l'[UpdateApplication](#) AWS CLI operazione.

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione ed effettua la chiamata UpdateApplication a Amazon S3 e lo stesso nome dell'oggetto. L'applicazione verrà riavviata con il nuovo pacchetto di codice.

Il seguente esempio di richiesta per l'operazione UpdateApplication ricarica il codice dell'applicazione e riavvia l'applicazione. Aggiorna l'CurrentApplicationVersionId alla versione corrente dell'applicazione. Puoi controllare la versione corrente dell'applicazione utilizzando le operazioni ListApplications o DescribeApplication. Aggiorna il suffisso del nome del bucket (*<username>*) con il suffisso che hai scelto nella sezione. [the section called “Crea due flussi di dati Amazon Kinesis”](#)

```
{  
  "ApplicationName": "test",  
  "CurrentApplicationVersionId": 1,  
  "ApplicationConfigurationUpdate": {  
    "ApplicationCodeConfigurationUpdate": {  
      "CodeContentUpdate": {  
        "S3ContentLocationUpdate": {  
          "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",  
          "FileKeyUpdate": "java-getting-started-1.0.jar"  
        }  
      }  
    }  
  }  
}
```

Passaggio 4: eliminare le AWS risorse

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Nozioni di base.

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione dei flussi di dati Kinesis](#)
- [Eliminazione dell'oggetto e del bucket Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com/kines.](https://console.aws.amazon.com/kines)
2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Scegli Configura.
4. Nella sezione Snapshot, scegli Disabilita, quindi scegli Aggiorna.
5. Nella pagina dell'applicazione, scegli Elimina e conferma l'eliminazione.

Eliminazione dei flussi di dati Kinesis

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink https://console.aws.amazon.com
2. Nel pannello del flusso di dati Kinesis, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.
4. Nella pagina Kinesis Streams, scegli, scegli Azioni ExampleOutputStream, scegli Elimina, quindi conferma l'eliminazione.

Eliminazione dell'oggetto e del bucket Amazon S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il **<username>**secchio ka-app-code -.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

Eliminazione delle risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.

2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-west-2.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytics--us-west-2. MyApplication
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Quindi scegli Elimina gruppo di log e conferma l'eliminazione.

Esempi di versioni precedenti (precedenti) del servizio gestito da Amazon per Apache Flink

Note

Per gli esempi attuali, vedere. [Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink](#)

Questa sezione fornisce esempi di creazione e utilizzo di applicazioni nel servizio gestito per Apache Flink. Sono inclusi codici e step-by-step istruzioni di esempio per aiutarti a creare applicazioni del servizio gestito per Apache Flink e testare i risultati.

Prima di esplorare questi esempi, ti consigliamo di esaminare quanto segue:

- [Come funziona](#)
- [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#)

Note

Gli esempi presuppongono che tu stia utilizzando la regione Stati Uniti occidentali (Oregon) (us-west-2). Se utilizzi una regione diversa, aggiorna il codice dell'applicazione, i comandi e i ruoli IAM in modo appropriato.

Argomenti

- [DataStream Esempi di API](#)
- [Esempi di Python](#)
- [Esempi di Scala](#)

DataStream Esempi di API

Negli esempi seguenti viene illustrato come creare applicazioni utilizzando l'API Flink DataStream .

Argomenti

- [Esempio: finestra a cascata](#)
- [Esempio: finestra scorrevole](#)
- [Esempio: scrittura in un bucket Amazon S3](#)
- [Tutorial: utilizzo di un'applicazione del servizio gestito per Apache Flink per replicare i dati da un argomento in un cluster MSK a un altro in un VPC](#)
- [Esempio: utilizzo di un consumatore EFO con un flusso di dati Kinesis](#)
- [Esempio: scrittura su Firehose](#)
- [Esempio: lettura da uno stream Kinesis in un altro account](#)
- [Tutorial: utilizzo di un truststore personalizzato con Amazon MSK](#)

Esempio: finestra a cascata**Note**

Per gli esempi attuali, vedi [Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink](#).

In questo esercizio, viene creata un'applicazione del servizio gestito per Apache Flink che aggrega i dati utilizzando una finestra a cascata. L'aggregazione è abilitata in Flink per impostazione predefinita. Per disabilitarla, esegui le seguenti operazioni:

```
sink.producer.aggregation-enabled' = 'false'
```

Note

Per impostare i prerequisiti richiesti per questo esercizio, completa prima l'esercizio [Tutorial: inizia a usare l' `DataStream API in Managed Service for Apache Flink`](#).

Questo argomento contiene le sezioni seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrivi record di esempio nel flusso di input](#)
- [Scarica ed esamina il codice dell'applicazione](#)
- [Per creare il codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink](#)
- [Pulizia delle risorse AWS](#)

Creazione delle risorse dipendenti

Prima di creare un'applicazione del servizio gestito per Apache Flink per questo esercizio, è necessario creare le seguenti risorse dipendenti:

- Due flussi di dati Kinesis (`ExampleInputStream` e `ExampleOutputStream`)
- Un bucket Amazon S3 per archiviare il codice dell'applicazione (`ka-app-code-<username>`)

Puoi creare i flussi Kinesis e un bucket S3 utilizzando la console. Per istruzioni sulla creazione di queste risorse, consulta i seguenti argomenti:

- [Creazione e aggiornamento dei flussi di dati](#) nella Guida per gli sviluppatori del flusso di dati Amazon Kinesis. Assegna un nome al flusso di dati **`ExampleInputStream`** e **`ExampleOutputStream`**.

- [Come si crea un bucket S3?](#) nella Guida per l'utente di Amazon Simple Storage Service. Assegna al bucket Amazon S3 un nome globalmente univoco aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

Scrivi record di esempio nel flusso di input

In questa sezione, viene utilizzato uno script Python per scrivere record di esempio nel flusso per l'applicazione da elaborare.

Note

Questa sezione richiede [AWS SDK for Python \(Boto\)](#).

1. Crea un file denominato `stock.py` con i seguenti contenuti:

```
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. Esegui lo script `stock.py`:

```
$ python stock.py
```

Mantieni lo script in esecuzione mentre completi il resto del tutorial.

Scarica ed esamina il codice dell'applicazione

Il codice dell'applicazione Java per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito.

1. Installa il client Git se non lo hai già fatto. Per ulteriori informazioni, consulta [Installazione di Git](#).
2. Clona il repository remoto con il comando seguente:

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

3. Passa alla directory `amazon-kinesis-data-analytics-java-examples/TumblingWindow`.

Il codice dell'applicazione si trova nei file `TumblingWindowStreamingJob.java`. Tieni presente quanto segue riguardo al codice dell'applicazione:

- L'applicazione utilizza un'origine Kinesis per leggere dal flusso di origine. Il seguente snippet crea l'origine Kinesis:

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

- Aggiungi l'istruzione d'importazione seguente:

```
import  
    org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows; //  
flink 1.13 onward
```

- L'applicazione utilizza l'operatore `timeWindow` per trovare il conteggio dei valori per ogni simbolo azionario in una finestra a cascata di 5 secondi. Il codice seguente crea l'operatore e invia i dati aggregati a un nuovo sink del flusso di dati Kinesis:

```
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());
```

Per creare il codice dell'applicazione

Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito:

1. Installa Java e Maven se non lo hai già fatto. Per ulteriori informazioni, consulta [Completa i prerequisiti richiesti](#) nel tutorial [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#).
2. Compila l'applicazione con il seguente comando:

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

 Note

Il codice di origine fornito si basa sulle librerie di Java 11.

La compilazione dell'applicazione crea il file JAR dell'applicazione (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

Caricamento del codice Java di streaming di Apache Flink

In questa sezione, il codice dell'applicazione viene caricato nel bucket Amazon S3 creato nella sezione [Creazione delle risorse dipendenti](#).

1. Nella console Amazon S3, scegli il **<username>** bucket `ka-app-code-` e scegli Carica.

2. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `aws-kinesis-analytics-java-apps-1.0.jar` creato nella fase precedente.
3. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink

Segui questi passaggi per creare, configurare, aggiornare ed eseguire l'applicazione utilizzando la console.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo `/flink https://console.aws.amazon.com`
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.
3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:
 - Per Nome applicazione, inserisci **MyApplication**.
 - Per Runtime, scegli Apache Flink.

Note

Il servizio gestito per Apache Flink utilizza Apache Flink versione 1.15.2.

- Lascia il menu a discesa di Apache Flink 1.15.2 (versione consigliata).
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
 5. Scegli Crea applicazione.

Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`
- Ruolo: `kinesisanalytics-MyApplication-us-west-2`

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere ai flussi di dati Kinesis.

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
3. Nella pagina Riepilogo, scegli Modifica policy. Scegli la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (`012345678901`) con il tuo ID account.

```
{
  "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"
    }
]
}

```

Configurazione dell'applicazione

1. Nella MyApplication pagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:
 - Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **aws-kinesis-analytics-java-apps-1.0.jar**

3. In Accesso alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
4. In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
5. Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.
6. Scegli Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: /aws/kinesis-analytics/MyApplication
- Flusso di log: kinesis-analytics-log-stream

Questo flusso di log viene utilizzato per monitorare l'applicazione. Non si tratta dello stesso flusso di log utilizzato dall'applicazione per inviare i risultati.

Esecuzione dell'applicazione.

1. Nella MyApplicationpagina, scegli Esegui. Lascia selezionata l'opzione Esegui senza snapshot e conferma l'operazione.
2. Quando l'applicazione è in esecuzione, aggiorna la pagina. La console mostra il Grafico dell'applicazione.

Puoi controllare i parametri del servizio gestito per Apache Flink sulla CloudWatch console per verificare che l'applicazione funzioni.

Pulizia delle risorse AWS

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Finestra a cascata.

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione dei flussi di dati Kinesis](#)
- [Eliminazione dell'oggetto e del bucket Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Apri la console del servizio gestito per Apache Flink all'indirizzo [/flink https://console.aws.amazon.com](https://console.aws.amazon.com/flink)
2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e quindi conferma l'eliminazione.

Eliminazione dei flussi di dati Kinesis

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com /kines.](https://console.aws.amazon.com/kinesis)
2. Nel pannello del flusso di dati Kinesis, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.
4. Nella pagina Kinesis Streams, scegli, scegli Azioni ExampleOutputStream, scegli Elimina, quindi conferma l'eliminazione.

Eliminazione dell'oggetto e del bucket Amazon S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il **<username>**secchio ka-app-code -.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

Eliminazione delle risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-west-2.

5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytics--us-west-2. MyApplication
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

Esempio: finestra scorrevole

Note

Per gli esempi attuali, vedi [Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink](#).

Note

Per impostare i prerequisiti richiesti per questo esercizio, completa innanzitutto l'esercizio [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#).

Questo argomento contiene le sezioni seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrivi record di esempio nel flusso di input](#)
- [Scarica ed esamina il codice dell'applicazione](#)
- [Per creare il codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink](#)
- [Pulizia AWS delle risorse](#)

Creazione delle risorse dipendenti

Prima di creare un'applicazione del servizio gestito per Apache Flink per questo esercizio, è necessario creare le seguenti risorse dipendenti:

- Due flussi di dati Kinesis (`ExampleInputStream` e `ExampleOutputStream`).
- Un bucket Amazon S3 per archiviare il codice dell'applicazione (`ka-app-code-<username>`)

Puoi creare i flussi Kinesis e un bucket S3 utilizzando la console. Per istruzioni sulla creazione di queste risorse, consulta i seguenti argomenti:

- [Creazione e aggiornamento dei flussi di dati](#) nella Guida per gli sviluppatori del flusso di dati Amazon Kinesis. Assegna un nome ai flussi di dati **ExampleInputStream** e **ExampleOutputStream**.
- [Come si crea un bucket S3?](#) nella Guida per l'utente di Amazon Simple Storage Service. Assegna al bucket Amazon S3 un nome globalmente univoco aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

Scrivi record di esempio nel flusso di input

In questa sezione, viene utilizzato uno script Python per scrivere record di esempio nel flusso per l'applicazione da elaborare.

Note

Questa sezione richiede [AWS SDK for Python \(Boto\)](#).

1. Crea un file denominato `stock.py` con i seguenti contenuti:

```
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. Esegui lo script `stock.py`:

```
$ python stock.py
```

Mantieni lo script in esecuzione mentre completi il resto del tutorial.

Scarica ed esamina il codice dell'applicazione

Il codice dell'applicazione Java per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito.

1. Installa il client Git se non lo hai già fatto. Per ulteriori informazioni, consulta [Installazione di Git](#).
2. Clona il repository remoto con il comando seguente:

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

3. Passa alla directory `amazon-kinesis-data-analytics-java-examples/SlidingWindow`.

Il codice dell'applicazione si trova nei file

`SlidingWindowStreamingJobWithParallelism.java`. Tieni presente quanto segue riguardo al codice dell'applicazione:

- L'applicazione utilizza un'origine Kinesis per leggere dal flusso di origine. Il seguente snippet crea l'origine Kinesis:

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

- L'applicazione utilizza l'operatore `timeWindow` per trovare il valore minimo per ogni simbolo azionario in una finestra di 10 secondi che scorre di 5 secondi. Il codice seguente crea l'operatore e invia i dati aggregati a un nuovo sink del flusso di dati Kinesis:
- Aggiungi l'istruzione d'importazione seguente:

```
import
    org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows; //
flink 1.13 onward
```

- L'applicazione utilizza l'operatore `timeWindow` per trovare il conteggio dei valori per ogni simbolo azionario in una finestra a cascata di 5 secondi. Il codice seguente crea l'operatore e invia i dati aggregati a un nuovo sink del flusso di dati Kinesis:

```
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());
```

Per creare il codice dell'applicazione

Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito:

1. Installa Java e Maven se non lo hai già fatto. Per ulteriori informazioni, consulta [Completa i prerequisiti richiesti](#) nel tutorial [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#).
2. Compila l'applicazione con il seguente comando:

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

 Note

Il codice di origine fornito si basa sulle librerie di Java 11.

La compilazione dell'applicazione crea il file JAR dell'applicazione (target/aws-kinesis-analytics-java-apps-1.0.jar).

Caricamento del codice Java di streaming di Apache Flink

In questa sezione, il codice dell'applicazione viene caricato nel bucket Amazon S3 creato nella sezione [Creazione delle risorse dipendenti](#).

1. Nella console Amazon S3, scegli il **<username>** bucket ka-app-code-, quindi scegli Carica.
2. Nella fase Seleziona file, scegli Aggiungi file. Individua il file aws-kinesis-analytics-java-apps-1.0.jar creato nella fase precedente.
3. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink

Segui questi passaggi per creare, configurare, aggiornare ed eseguire l'applicazione utilizzando la console.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink https://console.aws.amazon.com
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.
3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:
 - Per Nome applicazione, inserisci **MyApplication**.
 - Per Runtime, scegli Apache Flink.

- Lascia il menu a discesa di Apache Flink 1.15.2 (versione consigliata).
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
 5. Scegli Crea applicazione.

Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`
- Ruolo: `kinesisanalytics-MyApplication-us-west-2`

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere ai flussi di dati Kinesis.

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
3. Nella pagina Riepilogo, scegli Modifica policy. Scegli la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (`012345678901`) con il tuo ID account.

```
{
  "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"
    }

```



```
]
}
```

Configurazione dell'applicazione

1. Nella MyApplication pagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:
 - Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **aws-kinesis-analytics-java-apps-1.0.jar**
3. In Accesso alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
4. In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
5. Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.
6. Scegli Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione Amazon, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: `/aws/kinesis-analytics/MyApplication`
- Flusso di log: `kinesis-analytics-log-stream`

Questo flusso di log viene utilizzato per monitorare l'applicazione. Non si tratta dello stesso flusso di log utilizzato dall'applicazione per inviare i risultati.

Configurazione del parallelismo dell'applicazione

Questo esempio di applicazione utilizza l'esecuzione parallela di attività. Il seguente codice dell'applicazione imposta il parallelismo dell'operatore `min`:

```
.setParallelism(3) // Set parallelism for the min operator
```

Il parallelismo dell'applicazione non può essere maggiore del parallelismo fornito, che ha un valore predefinito di 1. Per aumentare il parallelismo dell'applicazione, utilizza la seguente azione: AWS CLI

```
aws kinesisanalyticsv2 update-application
  --application-name MyApplication
  --current-application-version-id <VersionId>
  --application-configuration-update "{\"FlinkApplicationConfigurationUpdate\
\": { \"ParallelismConfigurationUpdate\": {\"ParallelismUpdate\": 5,
  \"ConfigurationTypeUpdate\": \"CUSTOM\" }}}"
```

È possibile recuperare l'ID della versione corrente dell'applicazione utilizzando le [DescribeApplication](#)azioni o. [ListApplications](#)

Esecuzione dell'applicazione.

Il grafico del processo Flink può essere visualizzato eseguendo l'applicazione, aprendo il pannello di controllo di Apache Flink e scegliendo il processo Flink desiderato.

Puoi controllare i parametri del servizio gestito per Apache Flink sulla CloudWatch console per verificare che l'applicazione funzioni.

Pulizia AWS delle risorse

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Finestra scorrevole.

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione dei flussi di dati Kinesis](#)
- [Eliminazione dell'oggetto e del bucket Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>

2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e quindi conferma l'eliminazione.

Eliminazione dei flussi di dati Kinesis

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com /kines.](https://console.aws.amazon.com/kines.)
2. Nel pannello del flusso di dati Kinesis, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.
4. Nella pagina Kinesis Streams, scegli, scegli Azioni ExampleOutputStream, scegli Elimina, quindi conferma l'eliminazione.

Eliminazione dell'oggetto e del bucket Amazon S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il **<username>**secchio ka-app-code -.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

Eliminazione delle risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-west-2.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytics--us-west-2. MyApplication
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.

4. Scegli Elimina gruppo di log e conferma l'eliminazione.

Esempio: scrittura in un bucket Amazon S3

In questo esercizio, viene creato un servizio gestito per Apache Flink con un flusso di dati Kinesis come origine e un bucket Amazon S3 come sink. Utilizzando il sink, puoi verificare l'output dell'applicazione nella console Amazon S3.

Note

Per impostare i prerequisiti richiesti per questo esercizio, completa prima l'esercizio [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#).

Questo argomento contiene le sezioni seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrivi record di esempio nel flusso di input](#)
- [Scarica ed esamina il codice dell'applicazione](#)
- [Modifica del codice dell'applicazione](#)
- [Per creare il codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink](#)
- [Verifica dell'output dell'applicazione](#)
- [Facoltativo: personalizzazione dell'origine e del sink](#)
- [Pulizia AWS delle risorse](#)

Creazione delle risorse dipendenti

Prima di creare un servizio gestito per Apache Flink per questo esercizio, è necessario creare le seguenti risorse dipendenti:

- Un flusso di dati Kinesis (`ExampleInputStream`).
- Un bucket Amazon S3 per archiviare il codice e l'output dell'applicazione (`ka-app-code-<username>`)

Note

Il servizio gestito per Apache Flink non può scrivere dati su Amazon S3 con la crittografia lato server abilitata sul servizio gestito per Apache Flink.

Puoi creare il flusso Kinesis e il bucket Amazon S3 utilizzando la console. Per istruzioni sulla creazione di queste risorse, consulta i seguenti argomenti:

- [Creazione e aggiornamento dei flussi di dati](#) nella Guida per gli sviluppatori del flusso di dati Amazon Kinesis. Assegna un nome al flusso di dati **ExampleInputStream**.
- [Come si crea un bucket S3?](#) nella Guida per l'utente di Amazon Simple Storage Service. Assegna al bucket Amazon S3 un nome univoco globale aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***. Crea due cartelle (**code** e **data**) nel bucket Amazon S3.

L'applicazione crea le seguenti CloudWatch risorse se non sono già presenti:

- Un gruppo di log chiamato `/AWS/KinesisAnalytics-java/MyApplication`.
- Un flusso di log denominato `kinesis-analytics-log-stream`.

Scrivi record di esempio nel flusso di input

In questa sezione, viene utilizzato uno script Python per scrivere record di esempio nel flusso per l'applicazione da elaborare.

Note

Questa sezione richiede [AWS SDK for Python \(Boto\)](#).

1. Crea un file denominato `stock.py` con i seguenti contenuti:

```
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. Esegui lo script `stock.py`:

```
$ python stock.py
```

Mantieni lo script in esecuzione mentre completi il resto del tutorial.

Scarica ed esamina il codice dell'applicazione

Il codice dell'applicazione Java per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito.

1. Installa il client Git se non lo hai già fatto. Per ulteriori informazioni, consulta [Installazione di Git](#).
2. Clona il repository remoto con il comando seguente:

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

3. Passa alla directory `amazon-kinesis-data-analytics-java-examples/S3Sink`.

Il codice dell'applicazione si trova nei file `S3StreamingSinkJob.java`. Tieni presente quanto segue riguardo al codice dell'applicazione:

- L'applicazione utilizza un'origine Kinesis per leggere dal flusso di origine. Il seguente snippet crea l'origine Kinesis:

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

- È necessario aggiungere l'istruzione d'importazione seguente:

```
import
    org.apache.flink.streaming.api.windowing.assigners.TumblingProcessingTimeWindows;
```

- L'applicazione utilizza un sink Apache Flink S3 per scrivere su Amazon S3.

Il sink legge i messaggi in una finestra a cascata, li codifica in oggetti bucket S3 e invia gli oggetti codificati al sink S3. Il codice seguente codifica gli oggetti per l'invio ad Amazon S3:

```
input.map(value -> { // Parse the JSON
    JsonNode jsonNode = jsonParser.readValue(value, JsonNode.class);
    return new Tuple2<>(jsonNode.get("ticker").toString(), 1);
}).returns(Types.TUPLE(Types.STRING, Types.INT))
    .keyBy(v -> v.f0) // Logically partition the stream for each word
    .window(TumblingProcessingTimeWindows.of(Time.minutes(1)))
    .sum(1) // Count the appearances by ticker per partition
    .map(value -> value.f0 + " count: " + value.f1.toString() + "\n")
    .addSink(createS3SinkFromStaticConfig());
```

Note

L'applicazione utilizza un oggetto Flink `StreamingFileSink` per scrivere su Amazon S3. Per ulteriori informazioni su `StreamingFileSink`, consultate la documentazione [StreamingFileSink](#) di [Apache Flink](#).

Modifica del codice dell'applicazione

In questa sezione, viene modificato il codice dell'applicazione per scrivere l'output nel bucket Amazon S3.

Aggiorna la riga seguente con il nome utente per specificare la posizione di output dell'applicazione:

```
private static final String s3SinkPath = "s3a://ka-app-code-<username>/data";
```

Per creare il codice dell'applicazione

Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito:

1. Installa Java e Maven se non lo hai già fatto. Per ulteriori informazioni, consulta [Completa i prerequisiti richiesti](#) nel tutorial [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#).
2. Compila l'applicazione con il seguente comando:

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

La compilazione dell'applicazione crea il file JAR dell'applicazione (target/aws-kinesis-analytics-java-apps-1.0.jar).

Note

Il codice di origine fornito si basa sulle librerie di Java 11.

Caricamento del codice Java di streaming di Apache Flink

In questa sezione, il codice dell'applicazione viene caricato nel bucket Amazon S3 creato nella sezione [Creazione delle risorse dipendenti](#).

1. Nella console Amazon S3, scegli il **<username>** bucket ka-app-code-, accedi alla cartella del codice e scegli Carica.
2. Nella fase Seleziona file, scegli Aggiungi file. Individua il file aws-kinesis-analytics-java-apps-1.0.jar creato nella fase precedente.
3. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink

Segui questi passaggi per creare, configurare, aggiornare ed eseguire l'applicazione utilizzando la console.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.
3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:
 - Per Nome applicazione, inserisci **MyApplication**.
 - Per Runtime, scegli Apache Flink.
 - Lascia il menu a discesa di Apache Flink 1.15.2 (versione consigliata).
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
5. Scegli Crea applicazione.

Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Per Nome applicazione, inserisci **MyApplication**.
- Per Runtime, scegli Apache Flink.
- Lascia la versione Apache Flink 1.15.2 (versione consigliata).

6. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
7. Scegli Crea applicazione.

 Note

Quando crei un servizio gestito per Apache Flink tramite la console, hai la possibilità di richiedere la creazione di un ruolo e una policy IAM per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`
- Ruolo: `kinesisanalytics-MyApplication-us-west-2`

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere ai flussi di dati Kinesis.

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
3. Nella pagina Riepilogo, scegli Modifica policy. Scegli la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (*012345678901*) con il tuo ID account. Sostituisci <username> con il tuo nome utente.

```
{
    "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"
  },
]
}

```

Configurazione dell'applicazione

1. Nella MyApplicationpagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:
 - Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **code/aws-kinesis-analytics-java-apps-1.0.jar**
3. In Accesso alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
4. In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
5. Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.
6. Scegli Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: /aws/kinesis-analytics/MyApplication
- Flusso di log: kinesis-analytics-log-stream

Questo flusso di log viene utilizzato per monitorare l'applicazione. Non si tratta dello stesso flusso di log utilizzato dall'applicazione per inviare i risultati.

Esecuzione dell'applicazione.

1. Nella MyApplicationpagina, scegli Esegui. Lascia selezionata l'opzione Esegui senza snapshot e conferma l'operazione.
2. Quando l'applicazione è in esecuzione, aggiorna la pagina. La console mostra il Grafico dell'applicazione.

Verifica dell'output dell'applicazione

Nella console Amazon S3, apri la cartella dati nel bucket S3.

Dopo alcuni minuti, verranno visualizzati gli oggetti contenenti dati aggregati provenienti dall'applicazione.

Note

L'aggregazione è abilitata in Flink per impostazione predefinita. Per disabilitarla, esegui le seguenti operazioni:

```
sink.producer.aggregation-enabled' = 'false'
```

Facoltativo: personalizzazione dell'origine e del sink

In questa sezione viene effettuata la personalizzazione delle impostazioni relative agli oggetti origine e sink.

Note

Dopo aver modificato le sezioni di codice descritte nelle sezioni seguenti, effettua le seguenti operazioni per ricaricare il codice dell'applicazione:

- Ripeti i passaggi indicati nella sezione [the section called “Per creare il codice dell'applicazione”](#) per compilare il codice dell'applicazione aggiornato.
- Ripeti i passaggi indicati nella sezione [the section called “Caricamento del codice Java di streaming di Apache Flink”](#) per caricare il codice dell'applicazione aggiornato.
- Nella pagina dell'applicazione nella console, scegli Configura, quindi scegli Aggiorna per ricaricare il codice dell'applicazione aggiornato nell'applicazione.

Questa sezione contiene quanto segue:

- [Configurazione del partizionamento dei dati](#)
- [Configurazione della frequenza di lettura](#)
- [Configurazione del buffering di scrittura](#)

Configurazione del partizionamento dei dati

In questa sezione, vengono configurati i nomi delle cartelle che il file sink di streaming crea nel bucket S3. Per farlo, aggiungerai un assegnatore di bucket al file sink di streaming.

Per personalizzare i nomi delle cartelle creati nel bucket S3, procedi come segue:

1. Aggiungi le seguenti istruzioni d'importazione all'inizio del file `S3StreamingSinkJob.java`:

```
import
    org.apache.flink.streaming.api.functions.sink.filesystem.rollingpolicies.DefaultRollingPolicy;
import
    org.apache.flink.streaming.api.functions.sink.filesystem.bucketassigners.DateTimeBucketAssigner;
```

2. Aggiorna il metodo `createS3SinkFromStaticConfig()` nel codice in modo che sia simile a quanto segue:

```
private static StreamingFileSink<String> createS3SinkFromStaticConfig() {

    final StreamingFileSink<String> sink = StreamingFileSink
        .forRowFormat(new Path(s3SinkPath), new
        SimpleStringEncoder<String>("UTF-8"))
        .withBucketAssigner(new DateTimeBucketAssigner("yyyy-MM-dd--HH"))
        .withRollingPolicy(DefaultRollingPolicy.create().build())
        .build();
    return sink;
}
```

L'esempio di codice precedente utilizza il `DateTimeBucketAssigner` con un formato di data personalizzato per creare cartelle nel bucket S3. Il `DateTimeBucketAssigner` utilizza l'ora corrente del sistema per creare i nomi dei bucket. Se desideri creare un assegnatore di bucket personalizzato per personalizzare ulteriormente i nomi delle cartelle create, puoi creare una classe che implementi [BucketAssigner](#). Implementa la logica personalizzata utilizzando il metodo `getBucketId`.

Un'implementazione personalizzata del `BucketAssigner` può utilizzare il parametro [Context](#) per ottenere ulteriori informazioni su un record al fine di determinarne la cartella di destinazione.

Configurazione della frequenza di lettura

In questa sezione, viene configurata la frequenza delle letture sul flusso di origine.

Per impostazione predefinita, il consumatore del flusso di dati Kinesis legge dal flusso di origine cinque volte al secondo. Questa frequenza causerà problemi se c'è più di un client che legge dal flusso o se l'applicazione deve ritentare la lettura di un record. È possibile evitare questi problemi impostando la frequenza di lettura del consumatore.

Per stabilire la frequenza di lettura del consumatore Kinesis, viene impostato l'intervallo `SHARD_GETRECORDS_INTERVAL_MILLIS`.

Il seguente esempio di codice imposta l'intervallo `SHARD_GETRECORDS_INTERVAL_MILLIS` su un secondo:

```
kinesisConsumerConfig.setProperty(ConsumerConfigConstants.SHARD_GETRECORDS_INTERVAL_MILLIS, "1000");
```

Configurazione del buffering di scrittura

In questa sezione, vengono configurate la frequenza di scrittura e altre impostazioni del sink.

Per impostazione predefinita, l'applicazione scrive nel bucket di destinazione ogni minuto. È possibile modificare questo intervallo e altre impostazioni configurando l'oggetto `DefaultRollingPolicy`.

Note

Il file sink di streaming Apache Flink scrive nel suo bucket di output ogni volta che l'applicazione crea un checkpoint. Per impostazione predefinita, l'applicazione crea un checkpoint ogni minuto. Per aumentare l'intervallo di scrittura del sink S3, è necessario aumentare anche l'intervallo di checkpoint.

Per configurare l'oggetto `DefaultRollingPolicy`, procedi come segue:

1. Aumenta l'impostazione `CheckpointInterval` dell'applicazione. Il seguente input per l'[UpdateApplication](#) operazione imposta l'intervallo del checkpoint su 10 minuti:

```
{
  "ApplicationConfigurationUpdate": {
    "FlinkApplicationConfigurationUpdate": {
```

```

        "CheckpointConfigurationUpdate": {
            "ConfigurationTypeUpdate" : "CUSTOM",
            "CheckpointIntervalUpdate": 600000
        }
    },
    "ApplicationName": "MyApplication",
    "CurrentApplicationVersionId": 5
}

```

Per utilizzare il codice precedente, specifica la versione dell'applicazione corrente. È possibile recuperare la versione dell'applicazione utilizzando l'azione. [ListApplications](#)

2. Aggiungi la seguente istruzione d'importazione all'inizio del file `S3StreamingSinkJob.java`:

```
import java.util.concurrent.TimeUnit;
```

3. Aggiorna il metodo `createS3SinkFromStaticConfig` nel file `S3StreamingSinkJob.java` in modo che abbia il seguente aspetto:

```

private static StreamingFileSink<String> createS3SinkFromStaticConfig() {

    final StreamingFileSink<String> sink = StreamingFileSink
        .forRowFormat(new Path(s3SinkPath), new
SimpleStringEncoder<String>("UTF-8"))
        .withBucketAssigner(new DateTimeBucketAssigner("yyyy-MM-dd--HH"))
        .withRollingPolicy(
            DefaultRollingPolicy.create()
                .withRolloverInterval(TimeUnit.MINUTES.toMillis(8))
                .withInactivityInterval(TimeUnit.MINUTES.toMillis(5))
                .withMaxPartSize(1024 * 1024 * 1024)
                .build())
        .build();
    return sink;
}

```

L'esempio di codice precedente imposta la frequenza di scrittura nel bucket Amazon S3 su 8 minuti.

Per ulteriori informazioni sulla configurazione del file sink di streaming Apache Flink, consulta [Formati a righe di codice](#) nella [documentazione di Apache Flink](#).

Pulizia AWS delle risorse

Questa sezione include le procedure per l'AWS eliminazione delle risorse create nel tutorial di Amazon S3.

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione del flusso di dati Kinesis](#)
- [Eliminazione di oggetti e bucket Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Apri la console del servizio gestito per Apache Flink all'indirizzo `/flink https://console.aws.amazon.com`
2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e conferma l'eliminazione.

Eliminazione del flusso di dati Kinesis

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com /kines.](https://console.aws.amazon.com/kines.)
2. Nel pannello del flusso di dati Kinesis, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.

Eliminazione di oggetti e bucket Amazon S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il **<username>**secchio ka-app-code -.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

Eliminazione delle risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.

3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-west-2.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytics--us-west-2. MyApplication
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

Tutorial: utilizzo di un'applicazione del servizio gestito per Apache Flink per replicare i dati da un argomento in un cluster MSK a un altro in un VPC

Note

Per gli esempi attuali, vedi. [Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink](#)

Il seguente tutorial illustra come creare un Amazon VPC con un cluster Amazon MSK e due argomenti e come creare un'applicazione del servizio gestito per Apache Flink che legge da un argomento Amazon MSK e scrive su un altro.

Note

Per impostare i prerequisiti richiesti per questo esercizio, completa prima l'esercizio [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#).

Questo tutorial contiene le sezioni seguenti:

- [Creazione di un Amazon VPC con un cluster Amazon MSK](#)

- [Creazione del codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione dell'applicazione](#)
- [Configurazione dell'applicazione](#)
- [Esecuzione dell'applicazione.](#)
- [Eseguire il test dell'applicazione](#)

Creazione di un Amazon VPC con un cluster Amazon MSK

Per creare un esempio di VPC e di cluster di Amazon MSK a cui accedere da un'applicazione del servizio gestito per Apache Flink, segui il tutorial [Nozioni di base sull'utilizzo di Amazon MSK](#).

Quando completi il tutorial, tieni presente quanto segue:

- Nella [Fase 3: creazione di un argomento](#), ripeti il comando `kafka-topics.sh --create` per creare un argomento di destinazione denominato `AWSKafkaTutorialTopicDestination`:

```
bin/kafka-topics.sh --create --zookeeper ZooKeeperConnectionString --replication-  
factor 3 --partitions 1 --topic AWS KafkaTutorialTopicDestination
```

- Registra l'elenco dei server di bootstrap per il cluster. Puoi ottenere l'elenco dei server di bootstrap con il seguente comando (sostituisci *ClusterArn* con l'ARN del cluster MSK):

```
aws kafka get-bootstrap-brokers --region us-west-2 --cluster-arn ClusterArn  
{...  
  "BootstrapBrokerStringTls": "b-2.awskafkatutorialcluste.t79r6y.c4.kafka.us-  
west-2.amazonaws.com:9094,b-1.awskafkatutorialcluste.t79r6y.c4.kafka.us-  
west-2.amazonaws.com:9094,b-3.awskafkatutorialcluste.t79r6y.c4.kafka.us-  
west-2.amazonaws.com:9094"  
}
```

- Quando segui i passaggi dei tutorial, assicurati di utilizzare la AWS regione selezionata nel codice, nei comandi e nelle voci della console.

Creazione del codice dell'applicazione

In questa sezione, viene scaricato e compilato il file JAR dell'applicazione. Consigliamo l'uso di Java 11.

Il codice dell'applicazione Java per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito.

1. Installa il client Git se non lo hai già fatto. Per ulteriori informazioni, consulta [Installazione di Git](#).
2. Clona il repository remoto con il comando seguente:

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

3. Il codice dell'applicazione si trova nel file `amazon-kinesis-data-analytics-java-examples/KafkaConnectors/KafkaGettingStartedJob.java`. È possibile esaminare il codice per acquisire familiarità con la struttura del codice dell'applicazione del servizio gestito per Apache Flink.
4. Utilizza lo strumento Maven a riga di comando o il tuo ambiente di sviluppo preferito per creare il file JAR. Per compilare il file JAR utilizzando lo strumento Maven a riga di comando, inserisci quanto segue:

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

Se la compilazione ha esito positivo, viene creato il file seguente:

```
target/KafkaGettingStartedJob-1.0.jar
```

Note

Il codice di origine fornito si basa sulle librerie di Java 11. Se utilizzi un ambiente di sviluppo,

Caricamento del codice Java di streaming di Apache Flink

In questa sezione, il codice dell'applicazione viene caricato nel bucket Amazon S3 creato nel tutorial [Tutorial: inizia a usare l' `DataStream API` in Managed Service for Apache Flink](#).

Note

Se il bucket Amazon S3 è stato eliminato dal tutorial Nozioni di base, segui nuovamente il passaggio [the section called “Carica il file JAR del codice dell'applicazione”](#).

1. Nella console Amazon S3, scegli il **<username>** bucket ka-app-code- e scegli Carica.
2. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `KafkaGettingStartedJob-1.0.jar` creato nella fase precedente.
3. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink da /flink. <https://console.aws.amazon.com>
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.
3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:
 - Per Nome applicazione, immetti **MyApplication**.
 - Per Runtime, scegli Apache Flink versione 1.15.2.
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
5. Scegli Crea applicazione.

Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`
- Ruolo: `kinesisanalytics-MyApplication-us-west-2`

Configurazione dell'applicazione

1. Nella MyApplicationpagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:

- Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **KafkaGettingStartedJob-1.0.jar**
3. In Accesso alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.

 Note

Una volta specificate le risorse dell'applicazione utilizzando la console (come CloudWatch i log o un Amazon VPC), quest'ultima modifica il ruolo di esecuzione dell'applicazione e concede l'autorizzazione ad accedere a tali risorse.

4. In Proprietà, scegli Aggiungi gruppo. Inserisci le proprietà seguenti:

ID gruppo	Chiave	Valore
KafkaSource	topic	AWS KafkaTutorialTopic
KafkaSource	bootstrap.servers	<i>The bootstrap server list you saved previously</i>
KafkaSource	security.protocol	SSL
KafkaSource	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSource	ssl.truststore.password	changeit

 Note

Il file ssl.truststore.password per il certificato predefinito è "changeit"; non è necessario modificare questo valore se si utilizza il certificato predefinito.

Scegli nuovamente Aggiungi gruppo. Inserisci le proprietà seguenti:

ID gruppo	Chiave	Valore
KafkaSink	topic	AWS KafkaTutorialTopic Destination
KafkaSink	bootstrap.servers	<i>The bootstrap server list you saved previously</i>
KafkaSink	security.protocol	SSL
KafkaSink	ssl.truststore.location	/usr/lib/jvm/java-11-amazon- corretto/lib/security/cacerts
KafkaSink	ssl.truststore.password	changeit
KafkaSink	transaction.timeout.ms	1000

Il codice dell'applicazione legge le proprietà dell'applicazione di cui sopra per configurare l'origine e il sink utilizzati per interagire con il VPC e il cluster Amazon MSK. Per ulteriori informazioni sull'utilizzo delle proprietà, consulta [Usa le proprietà di runtime](#).

5. In Snapshot, scegli Disabilita. In questo modo sarà più semplice aggiornare l'applicazione senza caricare dati non validi sullo stato dell'applicazione.
6. In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
7. Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.
8. Nella sezione Cloud privato virtuale (VPC), seleziona il VPC da associare all'applicazione. Scegli le sottoreti e il gruppo di sicurezza associati al VPC che l'applicazione dovrà utilizzare per accedere alle risorse VPC.
9. clicca su Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: /aws/kinesis-analytics/MyApplication
- Flusso di log: kinesis-analytics-log-stream

Questo flusso di log viene utilizzato per monitorare l'applicazione.

Esecuzione dell'applicazione.

Il grafico del processo Flink può essere visualizzato eseguendo l'applicazione, aprendo la dashboard di Apache Flink e scegliendo il processo Flink desiderato.

Eseguire il test dell'applicazione

In questa sezione, viene effettuata la scrittura di record sull'argomento di origine. L'applicazione legge i record dall'argomento di origine e li scrive nell'argomento di destinazione. Verifica che l'applicazione funzioni scrivendo record sull'argomento di origine e leggendo record dall'argomento di destinazione.

Per scrivere e leggere i record degli argomenti, segui le istruzioni in [Fase 6: produzione e consumo di dati](#) nel tutorial [Nozioni di base sull'utilizzo di Amazon MSK](#).

Per leggere dall'argomento di destinazione, usa il nome dell'argomento di destinazione anziché l'argomento di origine nella tua seconda connessione al cluster:

```
bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString --  
consumer.config client.properties --topic AWS KafkaTutorialTopicDestination --from-  
beginning
```

Se non viene visualizzato alcun record nell'argomento di destinazione, consulta la sezione [Impossibile accedere alle risorse in un VPC](#) dell'argomento [Risolvi i problemi relativi al servizio gestito per Apache Flink](#).

Esempio: utilizzo di un consumatore EFO con un flusso di dati Kinesis

Note

Per gli esempi attuali, vedi. [Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink](#)

In questo esercizio, viene creata un'applicazione del servizio gestito per Apache Flink in grado di leggere da un flusso di dati Kinesis utilizzando un consumatore con [fan-out avanzato](#) (EFO). Se un consumatore Kinesis utilizza EFO, il servizio del flusso di dati Kinesis gli fornisce una larghezza di banda dedicata, anziché chiedere al consumatore di condividere la larghezza di banda fissa del flusso con gli altri utenti che leggono dal flusso.

Per ulteriori informazioni sull'utilizzo di EFO con il consumatore Kinesis, consulta [FLIP-128: fan-out avanzato per consumatori Kinesis](#).

L'applicazione creata in questo esempio utilizza il connettore AWS Kinesis (flink-connector-kinesis) 1.15.3.

Note

Per impostare i prerequisiti richiesti per questo esercizio, completa innanzitutto l'esercizio [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#).

Questo argomento contiene le sezioni seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrivi record di esempio nel flusso di input](#)
- [Scarica ed esamina il codice dell'applicazione](#)
- [Per creare il codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink](#)
- [Pulizia AWS delle risorse](#)

Creazione delle risorse dipendenti

Prima di creare un'applicazione del servizio gestito per Apache Flink per questo esercizio, è necessario creare le seguenti risorse dipendenti:

- Due flussi di dati Kinesis (`ExampleInputStream` e `ExampleOutputStream`)
- Un bucket Amazon S3 per archiviare il codice dell'applicazione (`ka-app-code-<username>`)

Puoi creare i flussi Kinesis e un bucket S3 utilizzando la console. Per istruzioni sulla creazione di queste risorse, consulta i seguenti argomenti:

- [Creazione e aggiornamento dei flussi di dati](#) nella Guida per gli sviluppatori del flusso di dati Amazon Kinesis. Assegna un nome al flusso di dati **ExampleInputStream** e **ExampleOutputStream**.
- [Come si crea un bucket S3?](#) nella Guida per l'utente di Amazon Simple Storage Service. Assegna al bucket Amazon S3 un nome globalmente univoco aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

Scrivi record di esempio nel flusso di input

In questa sezione, viene utilizzato uno script Python per scrivere record di esempio nel flusso per l'applicazione da elaborare.

Note

Questa sezione richiede [AWS SDK for Python \(Boto\)](#).

1. Crea un file denominato `stock.py` con i seguenti contenuti:

```
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. Esegui lo script `stock.py`:

```
$ python stock.py
```

Mantieni lo script in esecuzione mentre completi il resto del tutorial.

Scarica ed esamina il codice dell'applicazione

Il codice dell'applicazione Java per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito.

1. Installa il client Git se non lo hai già fatto. Per ulteriori informazioni, consulta [Installazione di Git](#).
2. Clona il repository remoto con il comando seguente:

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

3. Passa alla directory `amazon-kinesis-data-analytics-java-examples/EfoConsumer`.

Il codice dell'applicazione si trova nei file `EfoApplication.java`. Tieni presente quanto segue riguardo al codice dell'applicazione:

- Puoi abilitare il consumatore EFO impostando i seguenti parametri sul consumatore Kinesis:
 - `RECORD_PUBLISHER_TYPE`: imposta questo parametro su EFO per consentire all'applicazione di utilizzare un consumatore EFO per accedere ai dati del flusso di dati Kinesis.
 - `EFO_CONSUMER_NAME`: imposta questo parametro su un valore di stringa che sia unico tra i consumatori di questo flusso. Il riutilizzo di un nome utente nello stesso flusso di dati Kinesis causerà l'interruzione del precedente consumatore che utilizzava quel nome.
- Il seguente esempio di codice illustra come assegnare valori alle proprietà di configurazione del consumatore al fine di utilizzare un consumatore EFO per leggere dal flusso di origine:

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

Per creare il codice dell'applicazione

Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito:

1. Installa Java e Maven se non lo hai già fatto. Per ulteriori informazioni, consulta [Completa i prerequisiti richiesti](#) nel tutorial [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#).
2. Compila l'applicazione con il seguente comando:

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

Note

Il codice di origine fornito si basa sulle librerie di Java 11.

La compilazione dell'applicazione crea il file JAR dell'applicazione (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

Caricamento del codice Java di streaming di Apache Flink

In questa sezione, il codice dell'applicazione viene caricato nel bucket Amazon S3 creato nella sezione [Creazione delle risorse dipendenti](#).

1. Nella console Amazon S3, scegli il **<username>** bucket `ka-app-code-` e scegli Carica.

2. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `aws-kinesis-analytics-java-apps-1.0.jar` creato nella fase precedente.
3. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink

Segui questi passaggi per creare, configurare, aggiornare ed eseguire l'applicazione utilizzando la console.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo `/flink https://console.aws.amazon.com`
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.
3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:
 - Per Nome applicazione, inserisci **MyApplication**.
 - Per Runtime, scegli Apache Flink.

Note

Il servizio gestito per Apache Flink utilizza Apache Flink versione 1.15.2.

- Lascia il menu a discesa di Apache Flink 1.15.2 (versione consigliata).
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
 5. Scegli Crea applicazione.

Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`
- Ruolo: `kinesisanalytics-MyApplication-us-west-2`

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere ai flussi di dati Kinesis.

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
3. Nella pagina Riepilogo, scegli Modifica policy. Scegli la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (`012345678901`) con il tuo ID account.

Note

Queste autorizzazioni garantiscono all'applicazione la possibilità di accedere al consumatore EFO.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "logs:DescribeLogGroups",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*",
        "arn:aws:s3:::ka-app-code-<username>/aws-kinesis-analytics-java-apps-1.0.jar"
      ]
    },
    {

```

```

        "Sid": "DescribeLogStreams",
        "Effect": "Allow",
        "Action": "logs:DescribeLogStreams",
        "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:*"
    },
    {
        "Sid": "PutLogEvents",
        "Effect": "Allow",
        "Action": "logs:PutLogEvents",
        "Resource": "arn:aws:logs:us-west-2:012345678901:log-group:/aws/
kinesis-analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    },
    {
        "Sid": "ListCloudwatchLogGroups",
        "Effect": "Allow",
        "Action": [
            "logs:DescribeLogGroups"
        ],
        "Resource": [
            "arn:aws:logs:us-west-2:012345678901:log-group:*"
        ]
    },
    {
        "Sid": "AllStreams",
        "Effect": "Allow",
        "Action": [
            "kinesis:ListShards",
            "kinesis:ListStreamConsumers",
            "kinesis:DescribeStreamSummary"
        ],
        "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/*"
    },
    {
        "Sid": "Stream",
        "Effect": "Allow",
        "Action": [
            "kinesis:DescribeStream",
            "kinesis:RegisterStreamConsumer",
            "kinesis:DeregisterStreamConsumer"
        ],
        "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
    },

```

```

    {
      "Sid": "WriteOutputStream",
      "Effect": "Allow",
      "Action": "kinesis:*",
      "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
    },
    {
      "Sid": "Consumer",
      "Effect": "Allow",
      "Action": [
        "kinesis:DescribeStreamConsumer",
        "kinesis:SubscribeToShard"
      ],
      "Resource": [
        "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream/
consumer/my-efo-flink-app",
        "arn:aws:kinesis:us-west-2:012345678901:stream/ExampleInputStream/
consumer/my-efo-flink-app:*"
      ]
    }
  ]
}

```

Configurazione dell'applicazione

1. Nella MyApplication pagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:
 - Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **aws-kinesis-analytics-java-apps-1.0.jar**
3. In Accesso alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
4. In Proprietà, scegli Crea gruppo.
5. Immetti i valori e le proprietà dell'applicazione seguenti:

ID gruppo	Chiave	Valore
ConsumerConfigProperties	flink.stream.recoverdpublisher	EF0
ConsumerConfigProperties	flink.stream.efo.consumername	basic-efo-flink-app
ConsumerConfigProperties	INPUT_STREAM	ExampleInputStream
ConsumerConfigProperties	flink.inputstream.initpos	LATEST
ConsumerConfigProperties	AWS_REGION	us-west-2

6. In Proprietà, scegli Crea gruppo.
7. Immetti i valori e le proprietà dell'applicazione seguenti:

ID gruppo	Chiave	Valore
ProducerConfigProperties	OUTPUT_STREAM	ExampleOutputStream
ProducerConfigProperties	AWS_REGION	us-west-2
ProducerConfigProperties	AggregationEnabled	false

8. In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
9. Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.
10. Scegli Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: `/aws/kinesis-analytics/MyApplication`
- Flusso di log: `kinesis-analytics-log-stream`

Questo flusso di log viene utilizzato per monitorare l'applicazione. Non si tratta dello stesso flusso di log utilizzato dall'applicazione per inviare i risultati.

Esecuzione dell'applicazione.

Il grafico del processo Flink può essere visualizzato eseguendo l'applicazione, aprendo il pannello di controllo di Apache Flink e scegliendo il processo Flink desiderato.

Puoi controllare i parametri del servizio gestito per Apache Flink sulla CloudWatch console per verificare che l'applicazione funzioni.

Puoi anche controllare nella console del flusso di dati Kinesis, all'interno della scheda Fan-out avanzato del flusso di dati, il nome del tuo consumatore (). `basic-efo-flink-app`

Pulizia AWS delle risorse

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Finestra per l'eliminazione delle risorse create nel tutorial Finestra

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione dei flussi di dati Kinesis](#)
- [Eliminazione del bucket e degli oggetti Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e quindi conferma l'eliminazione.

Eliminazione dei flussi di dati Kinesis

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com /kines.](https://console.aws.amazon.com/kinesis)
2. Nel pannello del flusso di dati Kinesis, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.
4. Nella pagina Kinesis Streams, scegli, scegli Azioni ExampleOutputStream, scegli Elimina, quindi conferma l'eliminazione.

Eliminazione del bucket e degli oggetti Amazon S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il **<username>**secchio ka-app-code -.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

Eliminazione delle risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-west-2.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytics--us-west-2. MyApplication
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Quindi scegli Elimina gruppo di log e conferma l'eliminazione.

Esempio: scrittura su Firehose

Note

Per gli esempi attuali, vedi. [Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink](#)

In questo esercizio, viene creata un'applicazione del servizio gestito per Apache Flink con un flusso di dati Kinesis come origine e un flusso Firehose come sink. Utilizzando il sink, puoi verificare l'output dell'applicazione in un bucket Amazon S3.

Note

Per impostare i prerequisiti richiesti per questo esercizio, completa prima l'esercizio [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#).

Questa sezione contiene le fasi seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrivi record di esempio nel flusso di input](#)
- [Scarica ed esamina il codice Java per lo streaming di Apache Flink](#)
- [Per creare il codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink](#)
- [Pulizia delle risorse AWS](#)

Creazione delle risorse dipendenti

Prima di creare un servizio gestito per Apache Flink per questo esercizio, è necessario creare le seguenti risorse dipendenti:

- Un flusso di dati Kinesis (`ExampleInputStream`)
- Un flusso Firehose su cui l'applicazione scrive l'output (`ExampleDeliveryStream`).
- Un bucket Amazon S3 per archiviare il codice dell'applicazione (`ka-app-code-<username>`)

Puoi creare il flusso Kinesis, i bucket Amazon S3 e il flusso Firehose utilizzando la console. Per istruzioni sulla creazione di queste risorse, consulta i seguenti argomenti:

- [Creazione e aggiornamento dei flussi di dati](#) nella Guida per gli sviluppatori del flusso di dati Amazon Kinesis. Assegna un nome al flusso di dati **ExampleInputStream**.
- [Creazione di un flusso di distribuzione Amazon Kinesis Data Firehose](#) nella Guida per sviluppatori Amazon Data Firehose. Assegna un nome al tuo stream Firehose. **ExampleDeliveryStream** Quando crei il flusso Firehose, crei anche la destinazione S3 del flusso Firehose e il ruolo IAM.
- [Come si crea un bucket S3?](#) nella Guida per l'utente di Amazon Simple Storage Service. Assegna al bucket Amazon S3 un nome globalmente univoco aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

Scrivi record di esempio nel flusso di input

In questa sezione, viene utilizzato uno script Python per scrivere record di esempio nel flusso per l'applicazione da elaborare.

Note

Questa sezione richiede [AWS SDK for Python \(Boto\)](#).

1. Crea un file denominato `stock.py` con i seguenti contenuti:

```
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. Esegui lo script `stock.py`:

```
$ python stock.py
```

Mantieni lo script in esecuzione mentre completi il resto del tutorial.

Scarica ed esamina il codice Java per lo streaming di Apache Flink

Il codice dell'applicazione Java per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito:

1. Clona il repository remoto con il comando seguente:

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

2. Passa alla directory `amazon-kinesis-data-analytics-java-examples/FirehoseSink`.

Il codice dell'applicazione si trova nei file `FirehoseSinkStreamingJob.java`. Tieni presente quanto segue riguardo al codice dell'applicazione:

- L'applicazione utilizza un'origine Kinesis per leggere dal flusso di origine. Il seguente snippet crea l'origine Kinesis:

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

- L'applicazione utilizza un sink Firehose per scrivere dati in un flusso Firehose. Il seguente snippet crea il sink Firehose:

```
private static KinesisFirehoseSink<String> createFirehoseSinkFromStaticConfig() {
    Properties sinkProperties = new Properties();
    sinkProperties.setProperty(AWS_REGION, region);

    return KinesisFirehoseSink.<String>builder()
        .setFirehoseClientProperties(sinkProperties)
        .setSerializationSchema(new SimpleStringSchema())
        .setDeliveryStreamName(outputDeliveryStreamName)
        .build();
}
```

Per creare il codice dell'applicazione

Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito:

1. Installa Java e Maven se non lo hai già fatto. Per ulteriori informazioni, consulta [Completa i prerequisiti richiesti](#) nel tutorial [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#).
2. Per utilizzare il connettore Kinesis per la seguente applicazione, è necessario scaricare, creare e installare Apache Maven. Per ulteriori informazioni, consulta [the section called “Utilizzo del connettore di flussi Apache Flink Kinesis con versioni precedenti di Apache Flink”](#).
3. Compila l'applicazione con il seguente comando:

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

 Note

Il codice di origine fornito si basa sulle librerie di Java 11.

La compilazione dell'applicazione crea il file JAR dell'applicazione (`target/aws-kinesis-analytics-java-apps-1.0.jar`).

Caricamento del codice Java di streaming di Apache Flink

In questa sezione, il codice dell'applicazione viene caricato nel bucket Amazon S3 creato nella sezione [Creazione delle risorse dipendenti](#).

Per caricare il codice dell'applicazione

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Nella console, scegli il **<username>** bucket `ka-app-code-`, quindi scegli Carica.
3. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `java-getting-started-1.0.jar` creato nella fase precedente.
4. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink

È possibile creare ed eseguire un'applicazione del servizio gestito per Apache Flink utilizzando la console o la AWS CLI.

 Note

Quando crei l'applicazione utilizzando la console, vengono create automaticamente le risorse AWS Identity and Access Management (IAM) e di File di CloudWatch log Amazon Logs. Quando crei l'applicazione utilizzando la AWS CLI, queste risorse vengono create separatamente.

Argomenti

- [Creazione ed esecuzione dell'applicazione \(console\)](#)
- [Crea ed esegui l'applicazione \(AWS CLI\)](#)

Creazione ed esecuzione dell'applicazione (console)

Segui questi passaggi per creare, configurare, aggiornare ed eseguire l'applicazione utilizzando la console.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.
3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:
 - Per Nome applicazione, immetti **MyApplication**.
 - Per Descrizione, inserisci **My java test app**.
 - Per Runtime, scegli Apache Flink.

Note

Il servizio gestito per Apache Flink utilizza Apache Flink versione 1.15.2.

- Lascia il menu a discesa di Apache Flink 1.15.2 (versione consigliata).
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
 5. Scegli Crea applicazione.

Note

Quando crei l'applicazione tramite la console, hai la possibilità di richiedere la creazione di un ruolo e una policy IAM per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`

- Ruolo: `kinesisanalytics-MyApplication-us-west-2`

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere ai flussi di dati Kinesis e Firehose.

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
3. Nella pagina Riepilogo, scegli Modifica policy. Scegli la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (`012345678901`) con il tuo ID account.

```
{
  "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"
    }
]
}

```

Configurazione dell'applicazione

1. Nella MyApplication pagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:

- Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **java-getting-started-1.0.jar**
3. In Accesso alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
 4. In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
 5. Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.
 6. Scegli Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: /aws/kinesis-analytics/MyApplication
- Flusso di log: kinesis-analytics-log-stream

Esecuzione dell'applicazione.

Il grafico del processo Flink può essere visualizzato eseguendo l'applicazione, aprendo il pannello di controllo di Apache Flink e scegliendo il processo Flink desiderato.

Interruzione dell'applicazione

Nella MyApplicationpagina, scegli Stop. Conferma l'operazione.

Aggiornamento dell'applicazione

Tramite la console, puoi aggiornare le impostazioni dell'applicazione, ad esempio le proprietà dell'applicazione, le impostazioni di monitoraggio e la posizione o il nome di file del JAR dell'applicazione.

Nella MyApplicationpagina, scegli Configura. Aggiorna le impostazioni dell'applicazione e scegli Aggiorna.

Note

Per aggiornare il codice dell'applicazione sulla console, è necessario modificare il nome oggetto del JAR, utilizzare un bucket S3 diverso o utilizzare la AWS CLI come descritto nella sezione [the section called “Aggiornamento del codice dell'applicazione”](#). Se il nome del file o il bucket non cambiano, il codice dell'applicazione non viene ricaricato quando scegli **Aggiorna** nella pagina **Configura**.

Crea ed esegui l'applicazione (AWS CLI)

In questa sezione, verrà utilizzata la AWS CLI per creare ed eseguire il servizio gestito per Apache Flink.

Creazione di una policy di autorizzazione

Innanzitutto, crea una policy di autorizzazione con due istruzioni: una che concede le autorizzazioni per l'operazione `read` sul flusso di origine e un'altra che concede le autorizzazioni per operazioni `write` sul flusso di sink. Collega quindi la policy a un ruolo IAM (che verrà creato nella sezione successiva). Pertanto, quando il servizio gestito per Apache Flink assume il ruolo, il servizio disporrà delle autorizzazioni necessarie per leggere dal flusso di origine e scrivere nel flusso di sink.

Utilizza il codice seguente per creare la policy di autorizzazione

`AKReadSourceStreamWriteSinkStream`. Sostituisci *username* con il nome utente che userai per creare il bucket Amazon S3 in cui archiviare il codice dell'applicazione. Sostituisci l'ID account nei nomi della risorsa Amazon (ARNs) (*012345678901*) con il tuo ID account.

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

Per step-by-step istruzioni su come creare una politica di autorizzazioni, consulta il [Tutorial: Create and Attach Your First Customer Managed Policy](#) nella IAM User Guide.

Note

Per accedere ad altri servizi Amazon, puoi utilizzare AWS SDK per Java. Il servizio gestito per Apache Flink imposta automaticamente le credenziali richieste dall'SDK su quelle del ruolo IAM di esecuzione del servizio associato all'applicazione. Non sono richieste fasi aggiuntive.

Creazione di un ruolo IAM

In questa sezione, viene creato un ruolo IAM per l'applicazione del servizio gestito per Apache Flink che può essere assunto per leggere un flusso di origine e scrivere nel flusso di sink.

Il servizio gestito per Apache Flink non può accedere al tuo flusso senza autorizzazioni. Queste autorizzazioni possono essere assegnate con un ruolo IAM. Ad ogni ruolo IAM sono collegate due policy. La policy di attendibilità concede al servizio gestito per Apache Flink l'autorizzazione ad assumere questo ruolo. La policy di autorizzazione determina cosa può fare il servizio gestito per Apache Flink dopo aver assunto il ruolo.

Collega la policy di autorizzazione creata nella sezione precedente a questo ruolo.

Per creare un ruolo IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nel riquadro di navigazione, seleziona Ruoli, quindi Crea nuovo ruolo.
3. In Seleziona tipo di identità attendibile, scegli Servizio AWS . In Scegli il servizio che utilizzerà questo ruolo, scegli Kinesis. In Seleziona il tuo caso d'uso, scegli Analisi dei dati Kinesis.

Scegli Successivo: Autorizzazioni.

4. Nella pagina Allega policy di autorizzazione, seleziona Successivo: esamina. Collega le policy di autorizzazione dopo aver creato il ruolo.
5. Nella pagina Crea ruolo, immetti **MF-stream-rw-role** per Nome ruolo. Scegli Crea ruolo.

È stato creato un nuovo ruolo IAM denominato MF-stream-rw-role. Successivamente, aggiorna le policy di attendibilità e di autorizzazione per il ruolo.

6. Collega la policy di autorizzazione al ruolo.

Note

Per questo esercizio, il servizio gestito per Apache Flink assume questo ruolo per la lettura di dati da un flusso di dati Kinesis (origine) e la scrittura dell'output in un altro flusso di dati Kinesis. Pertanto, devi collegare la policy creata nella fase precedente, [the section called “Creazione di una policy di autorizzazione”](#).

- a. Nella pagina Riepilogo, scegli la scheda Autorizzazioni.
- b. Scegliere Collega policy.
- c. Nella casella di ricerca, immetti **AKReadSourceStreamWriteSinkStream** (la policy creata nella sezione precedente).
- d. Scegli la policy AKReadSourceStreamWriteSinkStream e seleziona Collega policy.

È stato creato il ruolo di esecuzione del servizio che l'applicazione utilizzerà per accedere alle risorse. Prendi nota dell'ARN del nuovo ruolo.

Per step-by-step istruzioni sulla creazione di un ruolo, consulta [Creating an IAM Role \(Console\)](#) nella IAM User Guide.

Creazione dell'applicazione del servizio gestito per Apache Flink

1. Salvare il seguente codice JSON in un file denominato `create_request.json`. Sostituisci l'ARN del ruolo di esempio con l'ARN per il ruolo creato in precedenza. Sostituisci il suffisso dell'ARN del bucket con il suffisso scelto nella sezione [the section called “Creazione delle risorse dipendenti”](#) (ka-app-code-*<username>*). Sostituisci l'ID account di esempio (*012345678901*) nel ruolo di esecuzione del servizio con il tuo ID account.

```
{
  "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. Esegui l'operazione [CreateApplication](#) con la richiesta precedente per creare l'applicazione:

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

L'applicazione è ora creata. Avvia l'applicazione nella fase successiva.

Avvia l'applicazione

In questa sezione, viene utilizzata l'operazione [StartApplication](#) per avviare l'applicazione.

Per avviare l'applicazione

1. Salvare il seguente codice JSON in un file denominato `start_request.json`.

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

2. Esegui l'operazione [StartApplication](#) con la richiesta precedente per avviare l'applicazione:

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

L'applicazione è ora in esecuzione. Puoi controllare i parametri del servizio gestito per Apache Flink sulla CloudWatch console Amazon per verificare che l'applicazione funzioni.

Interruzione dell'applicazione

In questa sezione, viene utilizzata l'operazione [StopApplication](#) per interrompere l'applicazione.

Per interrompere l'applicazione

1. Salvare il seguente codice JSON in un file denominato `stop_request.json`.

```
{
  "ApplicationName": "test"
}
```

2. Esegui l'operazione [StopApplication](#) con la seguente richiesta di interrompere l'applicazione:

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

L'applicazione è ora interrotta.

Aggiungi un'opzione CloudWatch di registrazione

Puoi utilizzare la AWS CLI per aggiungere un flusso di CloudWatch log Amazon all'applicazione. Per ulteriori informazioni sull'utilizzo CloudWatch dei log con la tua applicazione, consulta [the section called “Configurare la registrazione delle applicazioni in Managed Service per Apache Flink”](#).

Aggiornamento del codice dell'applicazione

Quando devi aggiornare il codice dell'applicazione con una nuova versione del pacchetto di codice, utilizza l'[UpdateApplication](#) AWS CLI operazione.

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione ed effettua la chiamata UpdateApplication a Amazon S3 e lo stesso nome dell'oggetto.

Il seguente esempio di richiesta per l'operazione UpdateApplication ricarica il codice dell'applicazione e riavvia l'applicazione. Aggiorna l'CurrentApplicationVersionId alla versione corrente dell'applicazione. Puoi controllare la versione corrente dell'applicazione utilizzando le operazioni ListApplications o DescribeApplication. Aggiorna il suffisso del nome del bucket (< *username* >) con il suffisso che hai scelto nella sezione. [the section called “Creazione delle risorse dipendenti”](#)

```
{
  "ApplicationName": "test",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationCodeConfigurationUpdate": {
      "CodeContentUpdate": {
        "S3ContentLocationUpdate": {
          "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
          "FileKeyUpdate": "java-getting-started-1.0.jar"
        }
      }
    }
  }
}
```

Pulizia delle risorse AWS

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Nozioni di base.

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione del flusso di dati Kinesis](#)
- [Eliminazione del flusso Firehose](#)
- [Eliminazione dell'oggetto e del bucket Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [Elimina le tue risorse CloudWatch](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Scegli Configura.
4. Nella sezione Snapshot, scegli Disabilita, quindi scegli Aggiorna.
5. Nella pagina dell'applicazione, scegli Elimina e conferma l'eliminazione.

Eliminazione del flusso di dati Kinesis

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com /kines.](https://console.aws.amazon.com/kines.)
2. Nel pannello del flusso di dati Kinesis, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.

Eliminazione del flusso Firehose

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com /kines.](https://console.aws.amazon.com/kines.)
2. Nel pannello Firehose, scegliete. ExampleDeliveryStream
3. Nella ExampleDeliveryStreampagina, scegliete Delete Firehose stream e confermate l'eliminazione.

Eliminazione dell'oggetto e del bucket Amazon S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>

2. Scegli il **<username>**secchio ka-app-code -.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.
4. Se hai creato un bucket Amazon S3 per la destinazione del flusso Firehose, eliminalo.

Eliminazione delle risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-west-2.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Se hai creato una nuova policy per il flusso Firehose, eliminala.
7. Nella barra di navigazione, scegli Ruoli.
8. Scegli il ruolo kinesis-analytics--us-west-2. MyApplication
9. Quindi scegli Elimina ruolo e conferma l'eliminazione.
10. Se hai creato un nuovo ruolo per il flusso Firehose, eliminalo.

Elimina le tue risorse CloudWatch

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

Esempio: lettura da uno stream Kinesis in un altro account

Note

Per gli esempi attuali, vedi [Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink](#).

Questo esempio dimostra come creare un'applicazione del servizio gestito per Apache Flink che legge i dati da un flusso Kinesis in un account diverso. In questo esempio, vengono utilizzati un

account per il flusso Kinesis di origine e un secondo account per l'applicazione del servizio gestito per Apache Flink e il sink del flusso Kinesis.

Questo argomento contiene le sezioni seguenti:

- [Prerequisiti](#)
- [Configurazione](#)
- [Creazione flusso Kinesis di origine](#)
- [Crea e aggiorna i ruoli e le politiche IAM](#)
- [Aggiornamento dello script Python](#)
- [Aggiornamento dell'applicazione Java](#)
- [Crea, carica ed esegui l'applicazione](#)

Prerequisiti

- In questo tutorial, viene modificato l'esempio delle Nozioni di base per leggere i dati da un flusso Kinesis in un altro account. Completa il tutorial [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#) prima di procedere.
- Sono necessari due AWS account per completare questo tutorial: uno per il flusso di origine e uno per il flusso dell'applicazione e di sink. Per il flusso dell'applicazione e di sink, utilizza l' AWS account che hai usato per il tutorial Nozioni di base. Utilizza un account AWS diverso per il flusso di origine.

Configurazione

Accedi ai tuoi due AWS account utilizzando profili denominati. Modifica le AWS credenziali e i file di configurazione per includere due profili che contengano la regione e le informazioni sulla connessione per i due account.

Il seguente file di credenziali di esempio contiene due profili denominati, `ka-source-stream-account-profile` e `ka-sink-stream-account-profile`. Per l'account del flusso di sink, utilizza l'account che hai usato per il tutorial Nozioni di base.

```
[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
```

Il seguente file di configurazione di esempio contiene gli stessi profili denominati con le informazioni sulla regione e sul formato di output.

```
[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

Questo tutorial non utilizza il profilo `ka-sink-stream-account-profile`. È incluso come esempio di accesso a due AWS account diversi utilizzando i profili.

Per ulteriori informazioni sull'utilizzo di profili denominati con AWS CLI, consulta [Named Profiles](#) nella AWS Command Line Interfacedocumentazione.

Creazione flusso Kinesis di origine

In questa sezione, viene creato il flusso Kinesis nell'account di origine.

Inserisci il comando seguente per creare il flusso Kinesis che l'applicazione utilizzerà come input. Tieni presente che il parametro `--profile` specifica il profilo dell'account da utilizzare.

```
$ aws kinesis create-stream \
--stream-name SourceAccountExampleInputStream \
--shard-count 1 \
--profile ka-source-stream-account-profile
```

Crea e aggiorna i ruoli e le politiche IAM

Per consentire l'accesso agli oggetti su più AWS account, crea un ruolo e una policy IAM nell'account di origine. Quindi, modifica la policy IAM nell'account di sink. Per ulteriori informazioni sulla creazione

di ruoli e policy IAM, consulta i seguenti argomenti nella Guida per l'utente AWS Identity and Access Management :

- [Creazione di ruoli IAM](#)
- [Creazione di policy IAM](#)

Ruoli e politiche degli account Sink

1. Modifica la policy `kinesis-analytics-service-MyApplication-us-west-2` dal tutorial Nozioni di base. Questa policy consente di assumere il ruolo nell'account di origine per leggere il flusso di origine.

Note

Quando si utilizza la console per creare l'applicazione, la console crea una policy denominata `kinesis-analytics-service-<application name>-<application region>` e un ruolo denominato `kinesisanalytics-<application name>-<application region>`.

Aggiungi alla policy la sezione evidenziata di seguito. Sostituisci l'ID dell'account di esempio (`SOURCE01234567`) con l'ID dell'account che utilizzerai per il flusso di origine.

```
{
  "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. Apri il ruolo `kinesis-analytics-MyApplication-us-west-2` e prendi nota del nome della risorsa Amazon (ARN). Lo utilizzerai nella sezione successiva. Il ruolo ARN è simile al seguente.


```
arn:aws:iam::SINK012345678:role/service-role/kinesis-analytics-MyApplication-us-west-2
```

Ruoli e politiche dell'account di origine

1. Crea una policy nell'account di origine denominato KA-Source-Stream-Policy. Utilizza il JSON seguente per la policy. Sostituisci il numero di account di esempio con il numero dell'account di origine.

```
{
  "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. Crea un ruolo nell'account di origine denominato MF-Source-Stream-Role. Esegui le seguenti operazioni per creare il ruolo utilizzando il caso d'uso Flink gestito:
 1. Nella console di gestione IAM, scegli Crea ruolo.
 2. Nella pagina Crea ruolo, scegli Servizio AWS . Nell'elenco dei servizi, scegli Kinesis.
 3. Nella sezione Seleziona il tuo caso d'uso, scegli Servizio gestito per Apache Flink.
 4. Scegli Successivo: autorizzazioni.
 5. Aggiungi la policy di autorizzazione KA-Source-Stream-Policy che hai creato nel passaggio precedente effettuando le seguenti operazioni: Scegli Successivo: Tag.
 6. Seleziona Successivo: Rivedi.

7. Denomina il ruolo KA-Source-Stream-Role. L'applicazione utilizzerà questo ruolo per accedere al flusso di origine.
3. Aggiungi l'ARN `kinesis-analytics-MyApplication-us-west-2` dall'account sink alla relazione di attendibilità del ruolo KA-Source-Stream-Role nell'account di origine:
 1. Nella console IAM, apri KA-Source-Stream-Role.
 2. Scegli la scheda Relazioni di attendibilità.
 3. Seleziona Modifica relazione di attendibilità.
 4. Utilizza il codice seguente per la relazione di attendibilità. Sostituisci l'ID account di esempio (*SINK012345678*) con il tuo ID account sink.

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

Aggiornamento dello script Python

In questa sezione, viene aggiornato lo script Python che genera dati di esempio ai fini dell'utilizzo del profilo dell'account di origine.

Aggiorna lo script `stock.py` con le seguenti modifiche evidenziate.

```
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")

```

Aggiornamento dell'applicazione Java

In questa sezione, viene aggiornato il codice dell'applicazione Java in modo che assuma il ruolo dell'account di origine durante la lettura dal flusso di origine.

Apporta le modifiche seguenti al file `BasicStreamingJob.java`. Sostituisci il numero dell'account di origine di esempio (*SOURCE01234567*) con il tuo numero dell'account di origine.

```

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");
}
}
```

Crea, carica ed esegui l'applicazione

Effettua le seguenti operazioni per aggiornare ed eseguire l'applicazione:

1. Crea nuovamente l'applicazione eseguendo il seguente comando nella directory con il file `pom.xml`.

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

2. Elimina il file JAR precedente dal bucket Amazon Simple Storage Service (Amazon S3), quindi carica il nuovo file `aws-kinesis-analytics-java-apps-1.0.jar` nel bucket S3.
3. Nella pagina dell'applicazione nella console del servizio gestito per Apache Flink, scegli Configura, Aggiorna per ricaricare il file JAR dell'applicazione.
4. Esegui lo script `stock.py` per inviare i dati al flusso di origine.

```
python stock.py
```

L'applicazione ora legge i dati dal flusso Kinesis nell'altro account.

Per verificare che l'applicazione funzioni, puoi controllare il parametro `PutRecords.Bytes` del flusso `ExampleOutputStream`. Se c'è attività nel flusso di output, l'applicazione funziona correttamente.

Tutorial: utilizzo di un truststore personalizzato con Amazon MSK

Note

Per gli esempi attuali, consulta. [Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink](#)

origine dati corrente APIs

[Se utilizzi l'origine dati corrente APIs, l'applicazione può sfruttare l'utilità Amazon MSK Config Providers descritta qui.](#) Ciò consente alla KafkaSource funzione di accedere al keystore e al truststore per il TLS reciproco in Amazon S3.

```
...
// define names of config providers:
builder.setProperty("config.providers", "secretsmanager,s3import");

// provide implementation classes for each provider:
builder.setProperty("config.providers.secretsmanager.class",
    "com.amazonaws.kafka.config.providers.SecretsManagerConfigProvider");
builder.setProperty("config.providers.s3import.class",
    "com.amazonaws.kafka.config.providers.S3ImportConfigProvider");

String region = appProperties.get(Helpers.S3_BUCKET_REGION_KEY).toString();
String keystoreS3Bucket = appProperties.get(Helpers.KEYSTORE_S3_BUCKET_KEY).toString();
String keystoreS3Path = appProperties.get(Helpers.KEYSTORE_S3_PATH_KEY).toString();
String truststoreS3Bucket =
    appProperties.get(Helpers.TRUSTSTORE_S3_BUCKET_KEY).toString();
String truststoreS3Path = appProperties.get(Helpers.TRUSTSTORE_S3_PATH_KEY).toString();
String keystorePassSecret =
    appProperties.get(Helpers.KEYSTORE_PASS_SECRET_KEY).toString();
String keystorePassSecretField =
    appProperties.get(Helpers.KEYSTORE_PASS_SECRET_FIELD_KEY).toString();

// region, etc..
builder.setProperty("config.providers.s3import.param.region", region);

// properties
builder.setProperty("ssl.truststore.location", "${s3import:" + region + ":" +
    truststoreS3Bucket + "/" + truststoreS3Path + "}");
builder.setProperty("ssl.keystore.type", "PKCS12");
```

```
builder.setProperty("ssl.keystore.location", "${s3import:" + region + ":" +  
    keystoreS3Bucket + "/" + keystoreS3Path + "}");  
builder.setProperty("ssl.keystore.password", "${secretsmanager:" + keystorePassSecret +  
    ":" + keystorePassSecretField + "}");  
builder.setProperty("ssl.key.password", "${secretsmanager:" + keystorePassSecret + ":"  
    + keystorePassSecretField + "}");  
...
```

Maggiori informazioni e una procedura dettagliata sono disponibili [qui](#).

Eredità SourceFunction APIs

Se utilizzi la versione legacy SourceFunction APIs, l'applicazione utilizzerà schemi di serializzazione e deserializzazione personalizzati che sostituiscono il open metodo per caricare il truststore personalizzato. Ciò rende il truststore disponibile per l'applicazione dopo il riavvio dell'applicazione o la sostituzione dei thread.

Il truststore personalizzato viene recuperato e archiviato utilizzando il seguente codice:

```
public static void initializeKafkaTruststore() {  
    ClassLoader classLoader = Thread.currentThread().getContextClassLoader();  
    URL inputUrl = classLoader.getResource("kafka.client.truststore.jks");  
    File dest = new File("/tmp/kafka.client.truststore.jks");  
  
    try {  
        FileUtils.copyURLToFile(inputUrl, dest);  
    } catch (Exception ex) {  
        throw new FlinkRuntimeException("Failed to initialize Kakfa truststore", ex);  
    }  
}
```

Note

Apache Flink richiede che il truststore sia in [formato JKS](#).

Note

Per impostare i prerequisiti richiesti per questo esercizio, completa innanzitutto l'esercizio [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#).

Il seguente tutorial illustra come connettersi in modo sicuro (crittografia in transito) a un cluster Kafka che utilizza certificati server emessi da un'autorità di certificazione (CA) personalizzata, privata o addirittura ospitata autonomamente.

Per connettere in modo sicuro qualsiasi client Kafka tramite TLS a un cluster Kafka, il client Kafka (come l'applicazione Flink di esempio) deve considerare attendibile l'intera catena di attendibilità presentata dai certificati del server del cluster Kafka (dalla CA emittente fino alla CA di livello root). Come esempio di truststore personalizzato, verrà utilizzato un cluster Amazon MSK con l'autenticazione TLS reciproca (MTLS) abilitata. Ciò implica che i nodi del cluster MSK utilizzano certificati server emessi da un'autorità privata per la gestione del AWS certificato (ACM CA privata) che è privata per l'account e la regione dell'utente e pertanto non considerata attendibile dal truststore predefinito della macchina virtuale Java (JVM) che esegue l'applicazione Flink.

Note

- Un keystore serve per archiviare chiavi private e certificati di identità che un'applicazione deve presentare al server o al client per la verifica.
- Un truststore serve per archiviare i certificati delle autorità di certificazione (CA) che verificano il certificato presentato dal server in una connessione SSL.

È possibile utilizzare la tecnica illustrata in questo tutorial anche per le interazioni tra un'applicazione del servizio gestito per Apache Flink e altre origini Apache Kafka, tra cui:

- [Un cluster Apache Kafka personalizzato ospitato in \(AWS Amazon o EC2 Amazon EKS\)](#)
- Un cluster [Confluent](#) Kafka ospitato in AWS
- Un cluster Kafka on-premises accessibile tramite [AWS Direct Connect](#) o VPN

Questo tutorial contiene le sezioni seguenti:

- [Creazione di un VPC con un cluster Amazon MSK](#)
- [Crea un truststore personalizzato e applicalo al tuo cluster](#)
- [Creazione del codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione dell'applicazione](#)
- [Configurazione dell'applicazione](#)

- [Esecuzione dell'applicazione.](#)
- [Eseguire il test dell'applicazione](#)

Creazione di un VPC con un cluster Amazon MSK

Per creare un esempio di VPC e di cluster di Amazon MSK a cui accedere da un'applicazione del servizio gestito per Apache Flink, segui il tutorial [Nozioni di base sull'utilizzo di Amazon MSK](#).

Quando completi il tutorial, esegui anche le operazioni seguenti:

- Nella [Fase 3: creazione di un argomento](#), ripeti il comando `kafka-topics.sh --create` per creare un argomento di destinazione denominato `AWS KafkaTutorialTopicDestination`:

```
bin/kafka-topics.sh --create --bootstrap-server ZooKeeperConnectionString --
replication-factor 3 --partitions 1 --topic AWSKafkaTutorialTopicDestination
```

Note

Se il comando `kafka-topics.sh` restituisce una `ZooKeeperClientTimeoutException`, verifica che il gruppo di sicurezza del cluster Kafka disponga di una regola in entrata per consentire tutto il traffico proveniente dall'indirizzo IP privato dell'istanza client.

- Registra l'elenco dei server di bootstrap per il cluster. Puoi ottenere l'elenco dei server di bootstrap con il seguente comando (sostituisci *ClusterArn* con l'ARN del cluster MSK):

```
aws kafka get-bootstrap-brokers --region us-west-2 --cluster-arn ClusterArn
{...
  "BootstrapBrokerStringTls": "b-2.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-1.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-3.awskafkatutorialcluste.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094"
}
```

- Quando segui i passaggi di questo tutorial e i tutorial relativi ai prerequisiti, assicurati di utilizzare la AWS regione selezionata nel codice, nei comandi e nelle voci della console.

Crea un truststore personalizzato e applicalo al tuo cluster

In questa sezione, viene creata un'autorità di certificazione (CA) personalizzata da utilizzare per generare un truststore personalizzato e applicarlo al cluster MSK.

Per creare e applicare un truststore personalizzato, segui il tutorial [Autenticazione client](#) nella Guida per gli sviluppatori di Streaming gestito da Amazon per Apache Kafka.

Creazione del codice dell'applicazione

In questa sezione, viene scaricato e compilato il file JAR dell'applicazione.

Il codice dell'applicazione Java per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito.

1. Installa il client Git se non lo hai già fatto. Per ulteriori informazioni, consulta [Installazione di Git](#).
2. Clona il repository remoto con il comando seguente:

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

3. Il codice dell'applicazione si trova in `amazon-kinesis-data-analytics-java-examples/CustomKeystore`. È possibile esaminare il codice per acquisire familiarità con la struttura del codice del servizio gestito per Apache Flink.
4. Utilizza lo strumento Maven a riga di comando o il tuo ambiente di sviluppo preferito per creare il file JAR. Per compilare il file JAR utilizzando lo strumento Maven a riga di comando, inserisci quanto segue:

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

Se la compilazione ha esito positivo, viene creato il file seguente:

```
target/flink-app-1.0-SNAPSHOT.jar
```

Note

Il codice di origine fornito si basa sulle librerie di Java 11.

Caricamento del codice Java di streaming di Apache Flink

In questa sezione, il codice dell'applicazione viene caricato nel bucket Amazon S3 creato nel tutorial [Tutorial: inizia a usare l' `DataStream API` in Managed Service for Apache Flink](#).

Note

Se il bucket Amazon S3 è stato eliminato dal tutorial Nozioni di base, segui nuovamente il passaggio [the section called “Carica il file JAR del codice dell'applicazione”](#).

1. Nella console Amazon S3, scegli il **<username>** bucket `ka-app-code-` e scegli Carica.
2. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `flink-app-1.0-SNAPSHOT.jar` creato nella fase precedente.
3. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo `/flink https://console.aws.amazon.com`
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.
3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:
 - Per Nome applicazione, immetti **MyApplication**.
 - Per Runtime, scegli Apache Flink versione 1.15.2.
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
5. Scegli Crea applicazione.

Note

Quando crei un servizio gestito per Apache Flink tramite la console, hai la possibilità di richiedere la creazione di un ruolo e una policy IAM per l'applicazione. L'applicazione utilizza

questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`
- Ruolo: `kinesisanalytics-MyApplication-us-west-2`

Configurazione dell'applicazione

1. Nella MyApplication pagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:
 - Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **flink-app-1.0-SNAPSHOT.jar**
3. In Accesso alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.

Note

Quando vengono specificate le risorse dell'applicazione utilizzando la console (come i file di log o VPC), la console modifica il ruolo di esecuzione dell'applicazione per concedere l'autorizzazione all'accesso a tali risorse.

4. In Proprietà, scegli Aggiungi gruppo. Inserisci le proprietà seguenti:

ID gruppo	Chiave	Valore
KafkaSource	topic	AWS KafkaTutorialTopic
KafkaSource	bootstrap.servers	<i>The bootstrap server list you saved previously</i>
KafkaSource	security.protocol	SSL
KafkaSource	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts

ID gruppo	Chiave	Valore
KafkaSource	ssl.truststore.password	changeit

 Note

Il file `ssl.truststore.password` per il certificato predefinito è "changeit"; non è necessario modificare questo valore se si utilizza il certificato predefinito.

Scegli nuovamente Aggiungi gruppo. Inserisci le proprietà seguenti:

ID gruppo	Chiave	Valore
KafkaSink	topic	AWS KafkaTutorialTopic Destination
KafkaSink	bootstrap.servers	<i>The bootstrap server list you saved previously</i>
KafkaSink	security.protocol	SSL
KafkaSink	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
KafkaSink	ssl.truststore.password	changeit
KafkaSink	transaction.timeout.ms	1000

Il codice dell'applicazione legge le proprietà dell'applicazione di cui sopra per configurare l'origine e il sink utilizzati per interagire con il VPC e il cluster Amazon MSK. Per ulteriori informazioni sull'utilizzo delle proprietà, consulta [Usa le proprietà di runtime](#).

5. In Snapshot, scegli Disabilita. In questo modo sarà più semplice aggiornare l'applicazione senza caricare dati non validi sullo stato dell'applicazione.

6. In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
7. Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.
8. Nella sezione Cloud privato virtuale (VPC), seleziona il VPC da associare all'applicazione. Scegli le sottoreti e il gruppo di sicurezza associati al VPC che l'applicazione dovrà utilizzare per accedere alle risorse VPC.
9. clicca su Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: /aws/kinesis-analytics/MyApplication
- Flusso di log: kinesis-analytics-log-stream

Questo flusso di log viene utilizzato per monitorare l'applicazione.

Esecuzione dell'applicazione.

Il grafico del processo Flink può essere visualizzato eseguendo l'applicazione, aprendo la dashboard di Apache Flink e scegliendo il processo Flink desiderato.

Eseguire il test dell'applicazione

In questa sezione, viene effettuata la scrittura di record sull'argomento di origine. L'applicazione legge i record dall'argomento di origine e li scrive nell'argomento di destinazione. Verifica che l'applicazione funzioni scrivendo record sull'argomento di origine e leggendo record dall'argomento di destinazione.

Per scrivere e leggere i record degli argomenti, segui le istruzioni in [Fase 6: produzione e consumo di dati](#) nel tutorial [Nozioni di base sull'utilizzo di Amazon MSK](#).

Per leggere dall'argomento di destinazione, usa il nome dell'argomento di destinazione anziché l'argomento di origine nella tua seconda connessione al cluster:

```
bin/kafka-console-consumer.sh --bootstrap-server BootstrapBrokerString --  
consumer.config client.properties --topic AWS KafkaTutorialTopicDestination --from-  
beginning
```

Se non viene visualizzato alcun record nell'argomento di destinazione, consulta la sezione [Impossibile accedere alle risorse in un VPC](#) dell'argomento [Risolvi i problemi relativi al servizio gestito per Apache Flink](#).

Esempi di Python

Negli esempi seguenti viene illustrato come creare applicazioni utilizzando Python con l'API Table di Apache Flink.

Argomenti

- [Esempio: creazione di una finestra a cascata in Python](#)
- [Esempio: creazione di una finestra scorrevole in Python](#)
- [Esempio: invio di dati di streaming ad Amazon S3 in Python](#)

Esempio: creazione di una finestra a cascata in Python

Note

Per gli esempi attuali, vedi. [Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink](#)

In questo esercizio, viene creata un'applicazione Python del servizio gestito per Apache Flink che aggrega dati utilizzando una finestra a cascata.

Note

Per impostare i prerequisiti richiesti per questo esercizio, completa prima l'esercizio [Tutorial: Nozioni di base sull'utilizzo del servizio gestito per Apache Flink](#).

Questo argomento contiene le sezioni seguenti:

- [Creazione delle risorse dipendenti](#)

- [Scrivi record di esempio nel flusso di input](#)
- [Scarica ed esamina il codice dell'applicazione](#)
- [Comprimi e carica il codice Python in streaming di Apache Flink](#)
- [Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink](#)
- [Pulizia AWS delle risorse](#)

Creazione delle risorse dipendenti

Prima di creare un'applicazione del servizio gestito per Apache Flink per questo esercizio, è necessario creare le seguenti risorse dipendenti:

- Due flussi di dati Kinesis (`ExampleInputStream` e `ExampleOutputStream`)
- Un bucket Amazon S3 per archiviare il codice dell'applicazione (`ka-app-code-<username>`)

Puoi creare i flussi Kinesis e un bucket S3 utilizzando la console. Per istruzioni sulla creazione di queste risorse, consulta i seguenti argomenti:

- [Creazione e aggiornamento dei flussi di dati](#) nella Guida per gli sviluppatori del flusso di dati Amazon Kinesis. Assegna un nome ai flussi di dati **ExampleInputStream** e **ExampleOutputStream**.
- [Come si crea un bucket S3?](#) nella Guida per l'utente di Amazon Simple Storage Service. Assegna al bucket Amazon S3 un nome globalmente univoco aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

Scrivi record di esempio nel flusso di input

In questa sezione, viene utilizzato uno script Python per scrivere record di esempio nel flusso per l'applicazione da elaborare.

Note

Questa sezione richiede [AWS SDK for Python \(Boto\)](#).

 Note

Lo script Python in questa sezione utilizza la AWS CLI. È necessario configurare AWS CLI per utilizzare le credenziali dell'account e la regione predefinita. Per configurare la AWS CLI, immetti quanto segue:

```
aws configure
```

1. Crea un file denominato `stock.py` con i seguenti contenuti:

```
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. Esegui lo script `stock.py`:

```
$ python stock.py
```

Mantieni lo script in esecuzione mentre completi il resto del tutorial.

Scarica ed esamina il codice dell'applicazione

Il codice dell'applicazione Python per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito.

1. Installa il client Git se non lo hai già fatto. Per ulteriori informazioni, consulta [Installazione di Git](#).
2. Clona il repository remoto con il comando seguente:

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

3. Passa alla directory `amazon-kinesis-data-analytics-java-examples/python/TumblingWindow`.

Il codice dell'applicazione si trova nei file `tumbling-windows.py`. Tieni presente quanto segue riguardo al codice dell'applicazione:

- L'applicazione utilizza un'origine della tabella Kinesis per leggere dal flusso di origine. Il seguente snippet richiama la funzione `create_table` per creare l'origine della tabella Kinesis:

```
table_env.execute_sql(  
    create_input_table(input_table_name, input_stream, input_region,  
        stream_initpos)  
)
```

La funzione `create_table` utilizza un comando SQL per creare una tabella supportata dall'origine di streaming:

```
def create_input_table(table_name, stream_name, region, stream_initpos):  
    return """ CREATE TABLE {0} (  
        ticker VARCHAR(6),  
        price DOUBLE,  
        event_time TIMESTAMP(3),  
        WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND  
    )
```

```

PARTITIONED BY (ticker)
WITH (
  'connector' = 'kinesis',
  'stream' = '{1}',
  'aws.region' = '{2}',
  'scan.stream.initpos' = '{3}',
  'format' = 'json',
  'json.timestamp-format.standard' = 'ISO-8601'
) """.format(table_name, stream_name, region, stream_initpos)

```

- L'applicazione utilizza l'operatore Tumble per aggregare i record all'interno di una finestra a cascata specificata e per restituire i record aggregati come oggetto tabella:

```

tumbling_window_table = (
    input_table.window(
        Tumble.over("10.seconds").on("event_time").alias("ten_second_window")
    )
    .group_by("ticker, ten_second_window")
    .select("ticker, price.min as price, to_string(ten_second_window.end) as
event_time")

```

- L'applicazione utilizza il connettore Kinesis Flink, dal [flink-sql-connector-kinesis-1.15.2.jar](https://www.apache.org/flink-connector-kinesis).

Comprimi e carica il codice Python in streaming di Apache Flink

In questa sezione, il codice dell'applicazione viene caricato nel bucket Amazon S3 creato nella sezione [Creazione delle risorse dipendenti](#).

1. Utilizza l'applicazione di compressione che preferisci per comprimere i file `tumbling-windows.py` e `flink-sql-connector-kinesis-1.15.2.jar`. Assegna un nome all'archivio `myapp.zip`.
2. Nella console Amazon S3, scegli il **<username>** bucket `ka-app-code-` e scegli Carica.
3. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `myapp.zip` creato nella fase precedente.
4. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink

Segui questi passaggi per creare, configurare, aggiornare ed eseguire l'applicazione utilizzando la console.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.
3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:
 - Per Nome applicazione, inserisci **MyApplication**.
 - Per Runtime, scegli Apache Flink.

Note

Il servizio gestito per Apache Flink utilizza Apache Flink versione 1.15.2.

- Lascia il menu a discesa di Apache Flink 1.15.2 (versione consigliata).
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
 5. Scegli Crea applicazione.

Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`
- Ruolo: `kinesisanalytics-MyApplication-us-west-2`

Configurazione dell'applicazione

1. Nella MyApplicationpagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:
 - Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **myapp.zip**
3. In Accesso alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
4. In Proprietà, scegli Aggiungi gruppo.
5. Immetti i seguenti dati:

ID gruppo	Chiave	Valore
consumer.config.0	input.stream.name	ExampleInputStream
consumer.config.0	aws.region	us-west-2
consumer.config.0	scan.stream.initpos	LATEST

Scegli Save (Salva).

6. In Proprietà, scegli di nuovo Aggiungi gruppo.
7. Immetti i seguenti dati:

ID gruppo	Chiave	Valore
producer.config.0	output.stream.name	ExampleOutputStream
producer.config.0	aws.region	us-west-2
producer.config.0	shard.count	1

8. In Proprietà, scegli nuovamente Aggiungi gruppo. In Nome del gruppo, inserisci **kinesis.analytics.flink.run.options**. Questo speciale gruppo di proprietà indica all'applicazione dove trovare le relative risorse di codice. Per ulteriori informazioni, consulta [Specificate i vostri file di codice](#).

9. Immetti i seguenti dati:

ID gruppo	Chiave	Valore
kinesis.analytics.flink.run.options	python	tumbling-windows.py
kinesis.analytics.flink.run.options	jarfile	flink-sql-connector-kinesis-1.15.2.jar

10. In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.

11. Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.

12. Scegli Aggiorna.

 Note

Quando scegli di abilitare la CloudWatch registrazione, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: /aws/kinesis-analytics/MyApplication
- Flusso di log: kinesis-analytics-log-stream

Questo flusso di log viene utilizzato per monitorare l'applicazione. Non si tratta dello stesso flusso di log utilizzato dall'applicazione per inviare i risultati.

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere ai flussi di dati Kinesis.

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
3. Nella pagina Riepilogo, scegli Modifica policy. Scegli la scheda JSON.

4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (**012345678901**) con il tuo ID account.

```
{
  "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"
}
]
```

Esecuzione dell'applicazione.

Il grafico del processo Flink può essere visualizzato eseguendo l'applicazione, aprendo il pannello di controllo di Apache Flink e scegliendo il processo Flink desiderato.

Puoi controllare i parametri del servizio gestito per Apache Flink sulla CloudWatch console per verificare che l'applicazione funzioni.

Pulizia AWS delle risorse

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Finestra a cascata.

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione dei flussi di dati Kinesis](#)
- [Eliminazione dell'oggetto e del bucket Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e quindi conferma l'eliminazione.

Eliminazione dei flussi di dati Kinesis

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com /kines.](https://console.aws.amazon.com/kinesis)
2. Nel pannello del flusso di dati Kinesis, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.
4. Nella pagina Kinesis Streams, scegli, scegli Azioni ExampleOutputStream, scegli Elimina, quindi conferma l'eliminazione.

Eliminazione dell'oggetto e del bucket Amazon S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il **<username>**secchio ka-app-code -.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

Eliminazione delle risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-west-2.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytics--us-west-2. MyApplication
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

Esempio: creazione di una finestra scorrevole in Python

Note

Per gli esempi attuali, vedi. [Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink](#)

Note

Per impostare i prerequisiti richiesti per questo esercizio, completa innanzitutto l'esercizio [Tutorial: Nozioni di base sull'utilizzo del servizio gestito per Apache Flink](#).

Questo argomento contiene le sezioni seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrivi record di esempio nel flusso di input](#)
- [Scarica ed esamina il codice dell'applicazione](#)
- [Comprimi e carica il codice Python in streaming di Apache Flink](#)
- [Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink](#)
- [Pulizia AWS delle risorse](#)

Creazione delle risorse dipendenti

Prima di creare un'applicazione del servizio gestito per Apache Flink per questo esercizio, è necessario creare le seguenti risorse dipendenti:

- Due flussi di dati Kinesis (ExampleInputStream e ExampleOutputStream)

- Un bucket Amazon S3 per archiviare il codice dell'applicazione (ka-app-code-*<username>*)

Puoi creare i flussi Kinesis e un bucket S3 utilizzando la console. Per istruzioni sulla creazione di queste risorse, consulta i seguenti argomenti:

- [Creazione e aggiornamento dei flussi di dati](#) nella Guida per gli sviluppatori del flusso di dati Amazon Kinesis. Assegna un nome ai flussi di dati **ExampleInputStream** e **ExampleOutputStream**.
- [Come si crea un bucket S3?](#) nella Guida per l'utente di Amazon Simple Storage Service. Assegna al bucket Amazon S3 un nome globalmente univoco aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

Scrivi record di esempio nel flusso di input

In questa sezione, viene utilizzato uno script Python per scrivere record di esempio nel flusso per l'applicazione da elaborare.

 Note

Questa sezione richiede [AWS SDK for Python \(Boto\)](#).

 Note

Lo script Python in questa sezione utilizza la AWS CLI. È necessario configurare AWS CLI per utilizzare le credenziali dell'account e la regione predefinita. Per configurare la AWS CLI, immetti quanto segue:

```
aws configure
```

1. Crea un file denominato `stock.py` con i seguenti contenuti:

```
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. Esegui lo script `stock.py`:

```
$ python stock.py
```

Mantieni lo script in esecuzione mentre completi il resto del tutorial.

Scarica ed esamina il codice dell'applicazione

Il codice dell'applicazione Python per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito.

1. Installa il client Git se non lo hai già fatto. Per ulteriori informazioni, consulta [Installazione di Git](#).
2. Clona il repository remoto con il comando seguente:

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

3. Passa alla directory `amazon-kinesis-data-analytics-java-examples/python/SlidingWindow`.

Il codice dell'applicazione si trova nei file `sliding-windows.py`. Tieni presente quanto segue riguardo al codice dell'applicazione:

- L'applicazione utilizza un'origine della tabella Kinesis per leggere dal flusso di origine. Il seguente snippet richiama la funzione `create_input_table` per creare l'origine della tabella Kinesis:

```
table_env.execute_sql(
    create_input_table(input_table_name, input_stream, input_region,
        stream_initpos)
)
```

La funzione `create_input_table` utilizza un comando SQL per creare una tabella supportata dall'origine di streaming:

```
def create_input_table(table_name, stream_name, region, stream_initpos):
    return """ CREATE TABLE {0} (
        ticker VARCHAR(6),
        price DOUBLE,
        event_time TIMESTAMP(3),
        WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND
    )
    PARTITIONED BY (ticker)
    WITH (
        'connector' = 'kinesis',
        'stream' = '{1}',
        'aws.region' = '{2}',
        'scan.stream.initpos' = '{3}',
        'format' = 'json',
        'json.timestamp-format.standard' = 'ISO-8601'
    ) """ .format(table_name, stream_name, region, stream_initpos)
}
```

- L'applicazione utilizza l'operatore `Slide` per aggregare i record all'interno di una finestra scorrevole specificata e per restituire i record aggregati come oggetto tabella:

```
sliding_window_table = (
    input_table
    .window(
```

```
        Slide.over("10.seconds")
        .every("5.seconds")
        .on("event_time")
        .alias("ten_second_window")
    )
    .group_by("ticker, ten_second_window")
    .select("ticker, price.min as price, to_string(ten_second_window.end) as
event_time")
    )
```

- [L'applicazione utilizza il connettore Kinesis Flink, dal file -1.15.2.jar. flink-sql-connector-kinesis](#)

Comprimi e carica il codice Python in streaming di Apache Flink

In questa sezione, il codice dell'applicazione viene caricato nel bucket Amazon S3 creato nella sezione [Creazione delle risorse dipendenti](#).

Questa sezione descrive come creare il pacchetto di un'applicazione Python.

1. Utilizza l'applicazione di compressione che preferisci per comprimere i file `sliding-windows.py` e `flink-sql-connector-kinesis-1.15.2.jar`. Assegna un nome all'archivio `myapp.zip`.
2. Nella console Amazon S3, scegli il **<username>** bucket `ka-app-code-` e scegli Carica.
3. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `myapp.zip` creato nella fase precedente.
4. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink

Segui questi passaggi per creare, configurare, aggiornare ed eseguire l'applicazione utilizzando la console.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo `/flink https://console.aws.amazon.com`
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.

3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:

- Per Nome applicazione, inserisci **MyApplication**.
- Per Runtime, scegli Apache Flink.

 Note

Il servizio gestito per Apache Flink utilizza Apache Flink versione 1.15.2.

- Lascia il menu a discesa di Apache Flink 1.15.2 (versione consigliata).
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
5. Scegli Crea applicazione.

 Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`
- Ruolo: `kinesisanalytics-MyApplication-us-west-2`

Configurazione dell'applicazione

1. Nella MyApplicationpagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:
 - Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **myapp.zip**
3. In Accesso alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
4. In Proprietà, scegli Aggiungi gruppo.

5. Immetti i valori e le proprietà dell'applicazione seguenti:

ID gruppo	Chiave	Valore
consumer.config.0	input.stream.name	ExampleInputStream
consumer.config.0	aws.region	us-west-2
consumer.config.0	scan.stream.initpos	LATEST

Scegli Save (Salva).

6. In Proprietà, scegli nuovamente Aggiungi gruppo.
7. Immetti i valori e le proprietà dell'applicazione seguenti:

ID gruppo	Chiave	Valore
producer.config.0	output.stream.name	ExampleOutputStream
producer.config.0	aws.region	us-west-2
producer.config.0	shard.count	1

8. In Proprietà, scegli nuovamente Aggiungi gruppo. In Nome del gruppo, inserisci **kinesis.analytics.flink.run.options**. Questo speciale gruppo di proprietà indica all'applicazione dove trovare le relative risorse di codice. Per ulteriori informazioni, consulta [Specificate i vostri file di codice](#).
9. Immetti i valori e le proprietà dell'applicazione seguenti:

ID gruppo	Chiave	Valore
kinesis.analytics.flink.run.options	python	sliding-windows.py
kinesis.analytics.flink.run.options	jarfile	flink-sql-connector-kinesis_1.15.2.jar

10. In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
11. Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.
12. Scegli Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: /aws/kinesis-analytics/MyApplication
- Flusso di log: kinesis-analytics-log-stream

Questo flusso di log viene utilizzato per monitorare l'applicazione. Non si tratta dello stesso flusso di log utilizzato dall'applicazione per inviare i risultati.

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere ai flussi di dati Kinesis.

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
3. Nella pagina Riepilogo, scegli Modifica policy. Scegli la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (**012345678901**) con il tuo ID account.

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

Esecuzione dell'applicazione.

Il grafico del processo Flink può essere visualizzato eseguendo l'applicazione, aprendo il pannello di controllo di Apache Flink e scegliendo il processo Flink desiderato.

Puoi controllare i parametri del servizio gestito per Apache Flink sulla CloudWatch console per verificare che l'applicazione funzioni.

Pulizia AWS delle risorse

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Finestra scorrevole.

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione dei flussi di dati Kinesis](#)
- [Eliminazione dell'oggetto e del bucket Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e quindi conferma l'eliminazione.

Eliminazione dei flussi di dati Kinesis

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com /kines.](https://console.aws.amazon.com/kines)
2. Nel pannello del flusso di dati Kinesis, scegli. ExampleInputStream

3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.
4. Nella pagina Kinesis Streams, scegli, scegli Azioni ExampleOutputStream, scegli Elimina, quindi conferma l'eliminazione.

Eliminazione dell'oggetto e del bucket Amazon S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il **<username>**secchio ka-app-code -.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

Eliminazione delle risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-west-2.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytics--us-west-2. MyApplication
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

Esempio: invio di dati di streaming ad Amazon S3 in Python

Note

Per gli esempi attuali, vedi. [Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink](#)

In questo esercizio, viene creata un'applicazione Python del servizio gestito per Apache Flink che trasmette i dati a un sink Amazon Simple Storage Service.

Note

Per impostare i prerequisiti richiesti per questo esercizio, completa prima l'esercizio [Tutorial: Nozioni di base sull'utilizzo del servizio gestito per Apache Flink](#).

Questo argomento contiene le sezioni seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrivi record di esempio nel flusso di input](#)
- [Scarica ed esamina il codice dell'applicazione](#)
- [Comprimi e carica il codice Python in streaming di Apache Flink](#)
- [Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink](#)
- [Pulizia AWS delle risorse](#)

Creazione delle risorse dipendenti

Prima di creare un'applicazione del servizio gestito per Apache Flink per questo esercizio, è necessario creare le seguenti risorse dipendenti:

- Un flusso di dati Kinesis (`ExampleInputStream`)
- Un bucket Amazon S3 per archiviare il codice e l'output dell'applicazione (`ka-app-code-<username>`)

 Note

Il servizio gestito per Apache Flink non può scrivere dati su Amazon S3 con la crittografia lato server abilitata sul servizio gestito per Apache Flink.

Puoi creare il flusso Kinesis e il bucket Amazon S3 utilizzando la console. Per istruzioni sulla creazione di queste risorse, consulta i seguenti argomenti:

- [Creazione e aggiornamento dei flussi di dati](#) nella Guida per gli sviluppatori del flusso di dati Amazon Kinesis. Assegna un nome al flusso di dati **ExampleInputStream**.
- [Come si crea un bucket S3?](#) nella Guida per l'utente di Amazon Simple Storage Service. Assegna al bucket Amazon S3 un nome globalmente univoco aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

Scrivi record di esempio nel flusso di input

In questa sezione, viene utilizzato uno script Python per scrivere record di esempio nel flusso per l'applicazione da elaborare.

 Note

Questa sezione richiede [AWS SDK for Python \(Boto\)](#).

 Note

Lo script Python in questa sezione utilizza la AWS CLI. È necessario configurare AWS CLI per utilizzare le credenziali dell'account e la regione predefinita. Per configurare la AWS CLI, immetti quanto segue:

```
aws configure
```

1. Crea un file denominato `stock.py` con i seguenti contenuti:

```
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. Esegui lo script `stock.py`:

```
$ python stock.py
```

Mantieni lo script in esecuzione mentre completi il resto del tutorial.

Scarica ed esamina il codice dell'applicazione

Il codice dell'applicazione Python per questo esempio è disponibile da [GitHub](#). Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito.

1. Installa il client Git se non lo hai già fatto. Per ulteriori informazioni, consulta [Installazione di Git](#).
2. Clona il repository remoto con il comando seguente:

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

3. Passa alla directory `amazon-kinesis-data-analytics-java-examples/python/S3Sink`.

Il codice dell'applicazione si trova nei file `streaming-file-sink.py`. Tieni presente quanto segue riguardo al codice dell'applicazione:

- L'applicazione utilizza un'origine della tabella Kinesis per leggere dal flusso di origine. Il seguente snippet richiama la funzione `create_source_table` per creare l'origine della tabella Kinesis:

```
table_env.execute_sql(  
    create_source_table(input_table_name, input_stream, input_region,  
    stream_initpos)  
)
```

La funzione `create_source_table` utilizza un comando SQL per creare una tabella supportata dall'origine di streaming

```
import datetime  
import json  
import random  
import boto3  
  
STREAM_NAME = "ExampleInputStream"  
  
def get_data():  
    return {  
        'event_time': datetime.datetime.now().isoformat(),  
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),  
        'price': round(random.random() * 100, 2)}  
  
def generate(stream_name, kinesis_client):  
    while True:  
        data = get_data()  
        print(data)  
        kinesis_client.put_record(  
            StreamName=stream_name,
```



```

        Data=json.dumps(data),
        PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))

```

- L'applicazione utilizza il connettore filesystem per inviare i record a un bucket Amazon S3:

```

def create_sink_table(table_name, bucket_name):
    return """ CREATE TABLE {0} (
        ticker VARCHAR(6),
        price DOUBLE,
        event_time VARCHAR(64)
    )
    PARTITIONED BY (ticker)
    WITH (
        'connector'='filesystem',
        'path'='s3a://{1}/',
        'format'='json',
        'sink.partition-commit.policy.kind'='success-file',
        'sink.partition-commit.delay' = '1 min'
    ) """ .format(table_name, bucket_name)

```

- [L'applicazione utilizza il connettore Kinesis Flink, dal file -1.15.2.jar. flink-sql-connector-kinesis](#)

Comprimi e carica il codice Python in streaming di Apache Flink

In questa sezione, il codice dell'applicazione viene caricato nel bucket Amazon S3 creato nella sezione [Creazione delle risorse dipendenti](#).

1. Utilizza l'applicazione di compressione che preferisci per comprimere i file `streaming-file-sink.py` e [flink-sql-connector-kinesis-1.15.2.jar](#). Assegna un nome all'archivio `myapp.zip`.
2. Nella console Amazon S3, scegli il **<username>** bucket `ka-app-code-` e scegli Carica.
3. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `myapp.zip` creato nella fase precedente.
4. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Flink

Segui questi passaggi per creare, configurare, aggiornare ed eseguire l'applicazione utilizzando la console.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.
3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:
 - Per Nome applicazione, inserisci **MyApplication**.
 - Per Runtime, scegli Apache Flink.

Note

Il servizio gestito per Apache Flink utilizza Apache Flink versione 1.15.2.

- Lascia il menu a discesa di Apache Flink 1.15.2 (versione consigliata).
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
 5. Scegli Crea applicazione.

Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`
- Ruolo: `kinesisanalytics-MyApplication-us-west-2`

Configurazione dell'applicazione

1. Nella MyApplicationpagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:
 - Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **myapp.zip**
3. In Accesso alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
4. In Proprietà, scegli Aggiungi gruppo.
5. Immetti i valori e le proprietà dell'applicazione seguenti:

ID gruppo	Chiave	Valore
consumer.config.0	input.stream.name	ExampleInputStream
consumer.config.0	aws.region	us-west-2
consumer.config.0	scan.stream.initpos	LATEST

Scegli Save (Salva).

6. In Proprietà, scegli nuovamente Aggiungi gruppo. In Nome del gruppo, inserisci **kinesis.analytics.flink.run.options**. Questo speciale gruppo di proprietà indica all'applicazione dove trovare le relative risorse di codice. Per ulteriori informazioni, consulta [Specificate i vostri file di codice](#).
7. Immetti i valori e le proprietà dell'applicazione seguenti:

ID gruppo	Chiave	Valore
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. In Proprietà, scegli nuovamente Aggiungi gruppo. In Nome del gruppo, inserisci **sink.config.0**. Questo speciale gruppo di proprietà indica all'applicazione dove trovare le relative risorse di codice. Per ulteriori informazioni, consulta [Specificate i vostri file di codice](#).
9. Immetti i valori e le proprietà dell'applicazione seguenti (sostituisci *bucket-name* con il nome effettivo del bucket Amazon S3):

ID gruppo	Chiave	Valore
sink.config.0	output.bucket.name	<i>bucket-name</i>

10. In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
11. Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.
12. Scegli Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: /aws/kinesis-analytics/MyApplication
- Flusso di log: kinesis-analytics-log-stream

Questo flusso di log viene utilizzato per monitorare l'applicazione. Non si tratta dello stesso flusso di log utilizzato dall'applicazione per inviare i risultati.

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere ai flussi di dati Kinesis.

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
3. Nella pagina Riepilogo, scegli Modifica policy. Scegli la scheda JSON.

4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (**012345678901**) con il tuo ID account.

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

```

Esecuzione dell'applicazione.

Il grafico del processo Flink può essere visualizzato eseguendo l'applicazione, aprendo il pannello di controllo di Apache Flink e scegliendo il processo Flink desiderato.

Puoi controllare i parametri del servizio gestito per Apache Flink sulla CloudWatch console per verificare che l'applicazione funzioni.

Pulizia AWS delle risorse

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Finestra scorrevole.

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)

- [Eliminazione del flusso di dati Kinesis](#)
- [Eliminazione di oggetti e bucket Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e conferma l'eliminazione.

Eliminazione del flusso di dati Kinesis

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com /kines.](https://console.aws.amazon.com/kines.)
2. Nel pannello del flusso di dati Kinesis, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.

Eliminazione di oggetti e bucket Amazon S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il **<username>**secchio ka-app-code -.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

Eliminazione delle risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-west-2.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytics--us-west-2. MyApplication

8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

Esempi di Scala

Negli esempi seguenti viene illustrato come creare applicazioni utilizzando Scala con Apache Flink.

Argomenti

- [Esempio: creazione di una finestra a cascata in Scala](#)
- [Esempio: creazione di una finestra scorrevole in Scala](#)
- [Esempio: invio di dati di streaming ad Amazon S3 in Scala](#)

Esempio: creazione di una finestra a cascata in Scala

Note

Per gli esempi attuali, vedi [Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink](#).

Note

A partire dalla versione 1.15, Flink non supporta più Scala. Le applicazioni possono ora utilizzare l'API Java da qualsiasi versione di Scala. Flink utilizza ancora Scala internamente in alcuni componenti chiave, ma non espone Scala nel classloader del codice utente. Per questo motivo, gli utenti devono aggiungere le dipendenze di Scala nei propri archivi jar. Per ulteriori informazioni sulle modifiche a Scala in Flink 1.15, consulta [Scala non più disponibile nella versione 1.15](#).

In questo esercizio, viene creata una semplice applicazione di streaming che utilizza Scala 3.2.0 e l'API Java DataStream di Flink. L'applicazione legge i dati dal flusso Kinesis, li aggrega utilizzando finestre scorrevoli e scrive i risultati per generare il flusso Kinesis.

Note

Per impostare i prerequisiti richiesti per questo esercizio, completa prima l'esercizio [Nozioni di base \(Scala\)](#).

Questo argomento contiene le sezioni seguenti:

- [Scaricate ed esaminate il codice dell'applicazione](#)
- [Compilazione e caricamento del codice dell'applicazione](#)
- [Creazione ed esecuzione dell'applicazione \(console\)](#)
- [Creazione ed esecuzione dell'applicazione \(CLI\)](#)
- [Aggiornamento del codice dell'applicazione](#)
- [Pulizia AWS delle risorse](#)

Scaricate ed esaminate il codice dell'applicazione

Il codice dell'applicazione Python per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito.

1. Installa il client Git se non lo hai già fatto. Per ulteriori informazioni, consulta [Installazione di Git](#).
2. Clona il repository remoto con il comando seguente:

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

3. Passa alla directory `amazon-kinesis-data-analytics-java-examples/scala/TumblingWindow`.

Tieni presente quanto segue riguardo al codice dell'applicazione:

- Un file `build.sbt` contiene le informazioni sulla configurazione e le dipendenze dell'applicazione, incluse le librerie del servizio gestito per Apache Flink.

- Il file `BasicStreamingJob.scala` contiene il metodo principale che definisce la funzionalità dell'applicazione.
- L'applicazione utilizza un'origine Kinesis per leggere dal flusso di origine. Il seguente snippet crea l'origine Kinesis:

```
private def createSource: FlinkKinesisConsumer[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")

  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,
    defaultInputStreamName),
    new SimpleStringSchema, inputProperties)
}
```

L'applicazione utilizza anche un sink Kinesis per scrivere nel flusso dei risultati. Il seguente snippet crea il sink Kinesis:

```
private def createSink: KinesisStreamsSink[String] = {
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties
  val outputProperties = applicationProperties.get("ProducerConfigProperties")

  KinesisStreamsSink.builder[String]
    .setKinesisClientProperties(outputProperties)
    .setSerializationSchema(new SimpleStringSchema)
    .setStreamName(outputProperties.getProperty(streamNameKey,
      defaultOutputStreamName))
    .setPartitionKeyGenerator((element: String) => String.valueOf(element.hashCode))
    .build
}
```

- L'applicazione utilizza l'operatore finestra per trovare il conteggio dei valori per ogni simbolo azionario in una finestra a cascata di 5 secondi. Il codice seguente crea l'operatore e invia i dati aggregati a un nuovo sink del flusso di dati Kinesis:

```
environment.addSource(createSource)
  .map { value =>
    val jsonNode = jsonParser.readValue(value, classOf[JsonNode])
    new Tuple2[String, Int](jsonNode.get("ticker").toString, 1)
  }
  .returns(Types.TUPLE(Types.STRING, Types.INT))
  .keyBy(v => v.f0) // Logically partition the stream for each ticker
```

```
.window(TumblingProcessingTimeWindows.of(Time.seconds(10)))  
.sum(1) // Sum the number of tickers per partition  
.map { value => value.f0 + "," + value.f1.toString + "\n" }  
.sinkTo(createSink)
```

- L'applicazione crea connettori di origine e sink per accedere alle risorse esterne utilizzando un `StreamExecutionEnvironment` oggetto.
- L'applicazione crea connettori di origine e sink utilizzando proprietà dinamiche. Questi metodi leggono le proprietà dell'applicazione di runtime per configurare il connettori. Per ulteriori informazioni sulle proprietà di runtime, consulta [Proprietà di runtime](#).

Compilazione e caricamento del codice dell'applicazione

In questa sezione, il codice dell'applicazione viene compilato e caricato in un bucket Amazon S3.

Compilazione del codice dell'applicazione

Userai lo strumento di compilazione [SBT](#) per creare il codice Scala per l'applicazione. Per installare SBT, consulta [Installazione di sbt con setup cs](#). Dovrai inoltre installare il Java Development Kit (JDK). Consulta [Prerequisiti per il completamento degli esercizi](#).

1. Per usare il codice dell'applicazione, compila il codice e comprimilo in un file JAR. Puoi compilare e comprimere il codice con SBT:

```
sbt assembly
```

2. Se l'applicazione viene compilata correttamente, viene creato il seguente file:

```
target/scala-3.2.0/tumbling-window-scala-1.0.jar
```

Caricamento del codice Scala di streaming di Apache Flink

In questa sezione, viene creato un bucket Amazon S3 e caricato il codice dell'applicazione.

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Seleziona Crea bucket.
3. Immetti `ka-app-code-<username>` nel campo Nome bucket. Aggiungi un suffisso al nome del bucket, ad esempio il nome utente, per renderlo globalmente univoco. Scegli Next (Successivo).

4. Nella fase Configura opzioni, non modificare le impostazioni e scegli Successivo.
5. Nella fase Imposta autorizzazioni, non modificare le impostazioni e scegli Successivo.
6. Seleziona Crea bucket.
7. Scegli il bucket `ka-app-code-<username>`, quindi scegli Carica.
8. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `tumbling-window-scala-1.0.jar` creato nella fase precedente.
9. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione ed esecuzione dell'applicazione (console)

Segui questi passaggi per creare, configurare, aggiornare ed eseguire l'applicazione utilizzando la console.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo `/flink https://console.aws.amazon.com`
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.
3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:
 - Per Nome applicazione, immetti **MyApplication**.
 - Per Descrizione, inserisci **My Scala test app**.
 - Per Runtime, scegli Apache Flink.
 - Lascia la versione Apache Flink 1.15.2 (versione consigliata).
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
5. Scegli Crea applicazione.

Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza

questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`
- Ruolo: `kinesisanalytics-MyApplication-us-west-2`

Configurazione dell'applicazione

Per configurare l'applicazione, utilizza la procedura seguente.

Per configurare l'applicazione

1. Nella MyApplicationpagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:
 - Per Bucket Amazon S3, inserisci `ka-app-code-<username>`.
 - Per Percorso dell'oggetto Amazon S3, inserisci `tumbling-window-scala-1.0.jar`
3. In Accesso alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna `kinesis-analytics-MyApplication-us-west-2` per il ruolo IAM.
4. In Proprietà, scegli Aggiungi gruppo.
5. Immetti i seguenti dati:

ID gruppo	Chiave	Valore
ConsumerConfigProperties	input.stream.name	ExampleInputStream
ConsumerConfigProperties	aws.region	us-west-2
ConsumerConfigProperties	flink.stream.initpos	LATEST

Scegli Save (Salva).

6. In Proprietà, scegli di nuovo Aggiungi gruppo.

7. Immetti i seguenti dati:

ID gruppo	Chiave	Valore
ProducerConfigProperties	output.stream.name	ExampleOutputStream
ProducerConfigProperties	aws.region	us-west-2

8. In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
9. Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.
10. Scegli Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione Amazon, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: /aws/kinesis-analytics/MyApplication
- Flusso di log: kinesis-analytics-log-stream

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere al bucket Amazon S3.

Modifica della policy IAM per aggiungere le autorizzazioni per i bucket S3

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
3. Nella pagina Riepilogo, scegli Modifica policy. Scegli la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (**012345678901**) con il tuo ID account.

```

{
  "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"
    }
]
}

```

Esecuzione dell'applicazione.

Il grafico del processo Flink può essere visualizzato eseguendo l'applicazione, aprendo il pannello di controllo di Apache Flink e scegliendo il processo Flink desiderato.

Interruzione dell'applicazione

Per interrompere l'applicazione, nella MyApplication pagina, scegli Stop. Conferma l'operazione.

Creazione ed esecuzione dell'applicazione (CLI)

In questa sezione, verrà utilizzata la AWS Command Line Interface per creare ed eseguire il servizio gestito per Apache Flink. Utilizza il AWS CLI comando della `kinesisanalyticsv2` per creare e interagire con il servizio gestito per Apache Flink.

Creazione di una policy di autorizzazione

Note

È necessario creare una policy di autorizzazione e un ruolo per l'applicazione. Se non crei queste risorse IAM, l'applicazione non può accedere ai suoi dati e flussi di log.

Innanzitutto, crea una policy di autorizzazione con due istruzioni: una che concede le autorizzazioni per l'operazione di lettura sul flusso di origine e un'altra che concede le autorizzazioni per operazioni di scrittura sul flusso di sink. Collega quindi la policy a un ruolo IAM (che verrà creato nella sezione successiva). Pertanto, quando il servizio gestito per Apache Flink assume il ruolo, il servizio disporrà delle autorizzazioni necessarie per leggere dal flusso di origine e scrivere nel flusso di sink.

Utilizza il codice seguente per creare la policy di autorizzazione

AKReadSourceStreamWriteSinkStream. Sostituisci **username** con il nome utente utilizzato per creare il bucket Amazon S3 per archiviare il codice dell'applicazione. Sostituisci l'ID account nei nomi della risorsa Amazon (ARNs) (**012345678901**) con il tuo ID account. Il ruolo di esecuzione del servizio **MF-stream-rw-role** deve essere adattato al ruolo specifico del cliente.

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

Per step-by-step istruzioni su come creare una politica di autorizzazioni, consulta il [Tutorial: Create and Attach Your First Customer Managed Policy](#) nella IAM User Guide.

Creazione di un ruolo IAM

In questa sezione, viene creato un ruolo IAM per l'applicazione del servizio gestito per Apache Flink che può essere assunto per leggere un flusso di origine e scrivere nel flusso di sink.

Il servizio gestito per Apache Flink non può accedere al tuo flusso senza autorizzazioni. Queste autorizzazioni possono essere assegnate con un ruolo IAM. Ad ogni ruolo IAM sono collegate due policy. La policy di attendibilità concede al servizio gestito per Apache Flink l'autorizzazione per assumere il ruolo e la policy di autorizzazione determina cosa può fare il servizio assumendo questo ruolo.

Collega la policy di autorizzazione creata nella sezione precedente a questo ruolo.

Per creare un ruolo IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nel riquadro di navigazione, scegli Ruoli e quindi Crea ruolo.
3. In Seleziona tipo di entità attendibile, scegli Servizio AWS .
4. In Scegli il servizio che utilizzerà questo ruolo, scegli Kinesis.
5. In Seleziona il tuo caso d'uso, scegli Servizio gestito per Apache Flink.
6. Scegli Successivo: autorizzazioni.
7. Nella pagina Allega policy di autorizzazione, seleziona Successivo: esamina. Collega le policy di autorizzazione dopo aver creato il ruolo.

8. Nella pagina Crea ruolo, immetti **MF-stream-rw-role** per Nome ruolo. Scegli Crea ruolo.

È stato creato un nuovo ruolo IAM denominato **MF-stream-rw-role**. Successivamente, aggiorna le policy di trust e di autorizzazione per il ruolo

9. Collega la policy di autorizzazione al ruolo.

Note

Per questo esercizio, il servizio gestito per Apache Flink assume questo ruolo per la lettura di dati da un flusso di dati Kinesis (origine) e la scrittura dell'output in un altro flusso di dati Kinesis. Pertanto, devi collegare la policy creata nella fase precedente, [Creazione di una policy di autorizzazione](#).

- a. Nella pagina Riepilogo, scegli la scheda Autorizzazioni.
- b. Scegliere Collega policy.
- c. Nella casella di ricerca, immetti **AKReadSourceStreamWriteSinkStream** (la policy creata nella sezione precedente).
- d. Scegli la policy **AKReadSourceStreamWriteSinkStream** e seleziona Collega policy.

È stato creato il ruolo di esecuzione del servizio che l'applicazione utilizzerà per accedere alle risorse. Prendi nota dell'ARN del nuovo ruolo.

Per step-by-step istruzioni sulla creazione di un ruolo, consulta [Creating an IAM Role \(Console\)](#) nella IAM User Guide.

Creazione dell'applicazione

Salvare il seguente codice JSON in un file denominato `create_request.json`. Sostituisci l'ARN del ruolo di esempio con l'ARN per il ruolo creato in precedenza. Sostituisci il suffisso dell'ARN del bucket (username) con il suffisso scelto nella sezione precedente. Sostituisci l'ID account di esempio (012345678901) nel ruolo di esecuzione del servizio con il tuo ID account. `ServiceExecutionRole` deve includere il ruolo utente IAM creato nella sezione precedente.

```
"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"
    }
  ]
}

```

Esegui [CreateApplication](#) con la seguente richiesta per creare l'applicazione:

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

L'applicazione è ora creata. Avvia l'applicazione nella fase successiva.

Avvia l'applicazione

In questa sezione, viene utilizzata l'operazione [StartApplication](#) per avviare l'applicazione.

Per avviare l'applicazione

1. Salvare il seguente codice JSON in un file denominato `start_request.json`.

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

2. Esegui l'operazione `StartApplication` con la richiesta precedente per avviare l'applicazione:

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

L'applicazione è ora in esecuzione. Puoi controllare i parametri del servizio gestito per Apache Flink sulla CloudWatch console Amazon per verificare che l'applicazione funzioni.

Interruzione dell'applicazione

In questa sezione, viene utilizzata l'operazione [StopApplication](#) per interrompere l'applicazione.

Per interrompere l'applicazione

1. Salvare il seguente codice JSON in un file denominato `stop_request.json`.

```
{
  "ApplicationName": "tumbling_window"
}
```

2. Esegui l'operazione `StopApplication` con la richiesta precedente per interrompere l'applicazione:

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

L'applicazione è ora interrotta.

Aggiungi un'opzione CloudWatch di registrazione

Puoi utilizzare la AWS CLI per aggiungere un flusso di CloudWatch log Amazon all'applicazione. Per ulteriori informazioni sull'utilizzo CloudWatch dei log con l'applicazione, consulta [Configurazione della registrazione dell'applicazione](#).

Aggiornamento delle proprietà dell'ambiente

In questa sezione, viene utilizzata l'operazione [UpdateApplication](#) per modificare le proprietà di ambiente per l'applicazione senza ricompilare il codice dell'applicazione. In questo esempio, viene modificata la regione dei flussi di origine e destinazione.

Per aggiornare le proprietà di ambiente per l'applicazione

1. Salvare il seguente codice JSON in un file denominato `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. Esegui l'operazione `UpdateApplication` con la richiesta precedente per aggiornare le proprietà di ambiente:

```
aws kinesisanalyticstv2 update-application --cli-input-json file://  
update_properties_request.json
```

Aggiornamento del codice dell'applicazione

Quando devi aggiornare il codice dell'applicazione con una nuova versione del pacchetto di codice, utilizza l'operazione della [UpdateApplication](#) CLI.

Note

Per caricare una nuova versione del codice dell'applicazione con lo stesso nome file, è necessario specificare la nuova versione dell'oggetto. Per ulteriori informazioni sull'uso delle versioni degli oggetti Amazon S3, consulta [Abilitazione o disattivazione del controllo delle versioni](#).

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione ed effettua la chiamata a Amazon S3 e lo stesso nome dell'oggetto `UpdateApplication`, nonché la nuova versione dell'oggetto. L'applicazione verrà riavviata con il nuovo pacchetto di codice.

Il seguente esempio di richiesta per l'operazione `UpdateApplication` ricarica il codice dell'applicazione e riavvia l'applicazione. Aggiorna `CurrentApplicationVersionId` alla versione corrente dell'applicazione. Puoi controllare la versione corrente dell'applicazione utilizzando le operazioni `ListApplications` o `DescribeApplication`. Aggiorna il suffisso del nome del bucket (`<username>`) con il suffisso che hai scelto nella sezione [Creazione delle risorse dipendenti](#).

```
{  
  "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": "SAMPLEUehYngP87ex1nzYIGYgfhyvpvDU"  
  }  
}  
}  
}  
}
```

Pulizia AWS delle risorse

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Finestra a cascata.

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione dei flussi di dati Kinesis](#)
- [Eliminazione dell'oggetto e del bucket Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e quindi conferma l'eliminazione.

Eliminazione dei flussi di dati Kinesis

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com/kines.](https://console.aws.amazon.com/kines)
2. Nel pannello del flusso di dati Kinesis, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.
4. Nella pagina Kinesis Streams, scegli, scegli Azioni ExampleOutputStream, scegli Elimina, quindi conferma l'eliminazione.

Eliminazione dell'oggetto e del bucket Amazon S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il **<username>**secchio ka-app-code -.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

Eliminazione delle risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-west-2.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytics--us-west-2. MyApplication
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

Esempio: creazione di una finestra scorrevole in Scala

Note

Per gli esempi attuali, vedi [Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink](#).

 Note

A partire dalla versione 1.15, Flink non supporta più Scala. Le applicazioni possono ora utilizzare l'API Java da qualsiasi versione di Scala. Flink utilizza ancora Scala internamente in alcuni componenti chiave, ma non espone Scala nel classloader del codice utente. Per questo motivo, gli utenti devono aggiungere le dipendenze di Scala nei propri archivi jar. Per ulteriori informazioni sulle modifiche a Scala in Flink 1.15, consulta [Scala non più disponibile nella versione 1.15](#).

In questo esercizio, viene creata una semplice applicazione di streaming che utilizza Scala 3.2.0 e l'API Java DataStream di Flink. L'applicazione legge i dati dal flusso Kinesis, li aggrega utilizzando finestre scorrevoli e scrive i risultati per generare il flusso Kinesis.

 Note

Per impostare i prerequisiti richiesti per questo esercizio, completa prima l'esercizio [Nozioni di base \(Scala\)](#).

Questo argomento contiene le sezioni seguenti:

- [Scaricate ed esaminate il codice dell'applicazione](#)
- [Compilazione e caricamento del codice dell'applicazione](#)
- [Creazione ed esecuzione dell'applicazione \(console\)](#)
- [Creazione ed esecuzione dell'applicazione \(CLI\)](#)
- [Aggiornamento del codice dell'applicazione](#)
- [Pulizia AWS delle risorse](#)

Scaricate ed esaminate il codice dell'applicazione

Il codice dell'applicazione Python per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito.

1. Installa il client Git se non lo hai già fatto. Per ulteriori informazioni, consulta [Installazione di Git](#).
2. Clona il repository remoto con il comando seguente:

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

3. Passa alla directory `amazon-kinesis-data-analytics-java-examples/scala/SlidingWindow`.

Tieni presente quanto segue riguardo al codice dell'applicazione:

- Un file `build.sbt` contiene le informazioni sulla configurazione e le dipendenze dell'applicazione, incluse le librerie del servizio gestito per Apache Flink.
- Il file `BasicStreamingJob.scala` contiene il metodo principale che definisce la funzionalità dell'applicazione.
- L'applicazione utilizza un'origine Kinesis per leggere dal flusso di origine. Il seguente snippet crea l'origine Kinesis:

```
private def createSource: FlinkKinesisConsumer[String] = {  
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties  
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")  
  
  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,  
    defaultInputStreamName),  
    new SimpleStringSchema, inputProperties)  
}
```

L'applicazione utilizza anche un sink Kinesis per scrivere nel flusso dei risultati. Il seguente snippet crea il sink Kinesis:

```
private def createSink: KinesisStreamsSink[String] = {  
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties  
  val outputProperties = applicationProperties.get("ProducerConfigProperties")  
  
  KinesisStreamsSink.builder[String]  
    .setKinesisClientProperties(outputProperties)  
    .setSerializationSchema(new SimpleStringSchema)  
    .setStreamName(outputProperties.getProperty(streamNameKey,  
    defaultOutputStreamName))  
    .setPartitionKeyGenerator((element: String) => String.valueOf(element.hashCode))  
    .build  
}
```

- L'applicazione utilizza l'operatore finestra per trovare il conteggio dei valori per ogni simbolo azionario in una finestra a cascata di 10 secondi che scorre di 5 secondi. Il codice seguente crea l'operatore e invia i dati aggregati a un nuovo sink del flusso di dati Kinesis:

```
environment.addSource(createSource)
  .map { value =>
    val jsonNode = jsonParser.readValue(value, classOf[JsonNode])
    new Tuple2[String, Double](jsonNode.get("ticker").toString,
    jsonNode.get("price").asDouble)
  }
  .returns(Types.TUPLE(Types.STRING, Types.DOUBLE))
  .keyBy(v => v.f0) // Logically partition the stream for each word
  .window(SlidingProcessingTimeWindows.of(Time.seconds(10), Time.seconds(5)))
  .min(1) // Calculate minimum price per ticker over the window
  .map { value => value.f0 + String.format(":%.2f", value.f1) + "\n" }
  .sinkTo(createSink)
```

- L'applicazione crea connettori di origine e sink per accedere alle risorse esterne utilizzando un `StreamExecutionEnvironment` oggetto.
- L'applicazione crea connettori di origine e sink utilizzando proprietà dinamiche. Questi metodi leggono le proprietà dell'applicazione di runtime per configurare il connettori. Per ulteriori informazioni sulle proprietà di runtime, consulta [Proprietà di runtime](#).

Compilazione e caricamento del codice dell'applicazione

In questa sezione, il codice dell'applicazione viene compilato e caricato in un bucket Amazon S3.

Compilazione del codice dell'applicazione

Userai lo strumento di compilazione [SBT](#) per creare il codice Scala per l'applicazione. Per installare SBT, consulta [Installazione di sbt con setup cs](#). Dovrai inoltre installare il Java Development Kit (JDK). Consulta [Prerequisiti per il completamento degli esercizi](#).

1. Per usare il codice dell'applicazione, compila il codice e comprimilo in un file JAR. Puoi compilare e comprimere il codice con SBT:

```
sbt assembly
```

2. Se l'applicazione viene compilata correttamente, viene creato il seguente file:

```
target/scala-3.2.0/sliding-window-scala-1.0.jar
```

Caricamento del codice Scala di streaming di Apache Flink

In questa sezione, viene creato un bucket Amazon S3 e caricato il codice dell'applicazione.

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Seleziona Crea bucket.
3. Immetti `ka-app-code-<username>` nel campo Nome bucket. Aggiungi un suffisso al nome del bucket, ad esempio il nome utente, per renderlo globalmente univoco. Scegli Next (Successivo).
4. Nella fase Configura opzioni, non modificare le impostazioni e scegli Successivo.
5. Nella fase Imposta autorizzazioni, non modificare le impostazioni e scegli Successivo.
6. Seleziona Crea bucket.
7. Scegli il bucket `ka-app-code-<username>`, quindi scegli Carica.
8. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `sliding-window-scala-1.0.jar` creato nella fase precedente.
9. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione ed esecuzione dell'applicazione (console)

Segui questi passaggi per creare, configurare, aggiornare ed eseguire l'applicazione utilizzando la console.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo `/flink https://console.aws.amazon.com`
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.
3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:
 - Per Nome applicazione, immetti **MyApplication**.
 - Per Descrizione, inserisci **My Scala test app**.

- Per Runtime, scegli Apache Flink.
 - Lascia la versione Apache Flink 1.15.2 (versione consigliata).
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
 5. Scegli Crea applicazione.

Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`
- Ruolo: `kinesisanalytics-MyApplication-us-west-2`

Configurazione dell'applicazione

Per configurare l'applicazione, utilizza la procedura seguente.

Per configurare l'applicazione

1. Nella MyApplicationpagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:
 - Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **sliding-window-scala-1.0.jar**.
3. In Accesso alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
4. In Proprietà, scegli Aggiungi gruppo.
5. Immetti i seguenti dati:

ID gruppo	Chiave	Valore
ConsumerConfigProperties	input.stream.name	ExampleInputStream
ConsumerConfigProperties	aws.region	us-west-2
ConsumerConfigProperties	flink.stream.initpos	LATEST

Scegli Save (Salva).

- In Proprietà, scegli di nuovo Aggiungi gruppo.
- Immetti i seguenti dati:

ID gruppo	Chiave	Valore
ProducerConfigProperties	output.stream.name	ExampleOutputStream
ProducerConfigProperties	aws.region	us-west-2

- In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
- Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.
- Scegli Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione Amazon, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: `/aws/kinesis-analytics/MyApplication`

- Flusso di log: `kinesis-analytics-log-stream`

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere al bucket Amazon S3.

Modifica della policy IAM per aggiungere le autorizzazioni per i bucket S3

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
3. Nella pagina Riepilogo, scegli Modifica policy. Scegli la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (`012345678901`) con il tuo ID account.

```
{
  "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"
    }
]
}

```

Esecuzione dell'applicazione.

Il grafico del processo Flink può essere visualizzato eseguendo l'applicazione, aprendo il pannello di controllo di Apache Flink e scegliendo il processo Flink desiderato.

Interruzione dell'applicazione

Per interrompere l'applicazione, nella MyApplication pagina, scegli Stop. Conferma l'operazione.

Creazione ed esecuzione dell'applicazione (CLI)

In questa sezione, verrà utilizzata la AWS Command Line Interface per creare ed eseguire il servizio gestito per Apache Flink. Utilizza il AWS CLI comando della kinesisanalyticsv2 per creare e interagire con il servizio gestito per Apache Flink.

Creazione di una policy di autorizzazione

Note

È necessario creare una policy di autorizzazione e un ruolo per l'applicazione. Se non crei queste risorse IAM, l'applicazione non può accedere ai suoi dati e flussi di log.

Innanzitutto, crea una policy di autorizzazione con due istruzioni: una che concede le autorizzazioni per l'operazione di lettura sul flusso di origine e un'altra che concede le autorizzazioni per operazioni di scrittura sul flusso di sink. Collega quindi la policy a un ruolo IAM (che verrà creato nella sezione successiva). Pertanto, quando il servizio gestito per Apache Flink assume il ruolo, il servizio disporrà delle autorizzazioni necessarie per leggere dal flusso di origine e scrivere nel flusso di sink.

Utilizza il codice seguente per creare la policy di autorizzazione

AKReadSourceStreamWriteSinkStream. Sostituisci **username** con il nome utente utilizzato per creare il bucket Amazon S3 per archiviare il codice dell'applicazione. Sostituisci l'ID account nei nomi della risorsa Amazon (ARNs) (**012345678901**) con il tuo ID account.

```
{
  "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"
        }
    ]
}

```

Per step-by-step istruzioni su come creare una politica di autorizzazioni, consulta il [Tutorial: Create and Attach Your First Customer Managed Policy](#) nella IAM User Guide.

Creazione di un ruolo IAM

In questa sezione, viene creato un ruolo IAM per l'applicazione del servizio gestito per Apache Flink che può essere assunto per leggere un flusso di origine e scrivere nel flusso di sink.

Il servizio gestito per Apache Flink non può accedere al tuo flusso senza autorizzazioni. Queste autorizzazioni possono essere assegnate con un ruolo IAM. Ad ogni ruolo IAM sono collegate due policy. La policy di attendibilità concede al servizio gestito per Apache Flink l'autorizzazione per assumere il ruolo e la policy di autorizzazione determina cosa può fare il servizio assumendo questo ruolo.

Collega la policy di autorizzazione creata nella sezione precedente a questo ruolo.

Per creare un ruolo IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nel riquadro di navigazione, scegli Ruoli e quindi Crea ruolo.
3. In Seleziona tipo di entità attendibile, scegli Servizio AWS .
4. In Scegli il servizio che utilizzerà questo ruolo, scegli Kinesis.
5. In Seleziona il tuo caso d'uso, scegli Servizio gestito per Apache Flink.
6. Scegli Successivo: autorizzazioni.
7. Nella pagina Allega policy di autorizzazione, seleziona Successivo: esamina. Collega le policy di autorizzazione dopo aver creato il ruolo.
8. Nella pagina Crea ruolo, immetti **MF-stream-rw-role** per Nome ruolo. Scegli Crea ruolo.

È stato creato un nuovo ruolo IAM denominato `MF-stream-rw-role`. Successivamente, aggiorna le policy di trust e di autorizzazione per il ruolo

9. Collega la policy di autorizzazione al ruolo.

 Note

Per questo esercizio, il servizio gestito per Apache Flink assume questo ruolo per la lettura di dati da un flusso di dati Kinesis (origine) e la scrittura dell'output in un altro flusso di dati Kinesis. Pertanto, devi collegare la policy creata nella fase precedente, [Creazione di una policy di autorizzazione](#).

- a. Nella pagina Riepilogo, scegli la scheda Autorizzazioni.
- b. Scegliere Collega policy.
- c. Nella casella di ricerca, immetti **AKReadSourceStreamWriteSinkStream** (la policy creata nella sezione precedente).
- d. Scegli la policy `AKReadSourceStreamWriteSinkStream` e seleziona Collega policy.

È stato creato il ruolo di esecuzione del servizio che l'applicazione utilizzerà per accedere alle risorse. Prendi nota dell'ARN del nuovo ruolo.

Per step-by-step istruzioni sulla creazione di un ruolo, consulta [Creating an IAM Role \(Console\)](#) nella IAM User Guide.

Creazione dell'applicazione

Salvare il seguente codice JSON in un file denominato `create_request.json`. Sostituisci l'ARN del ruolo di esempio con l'ARN per il ruolo creato in precedenza. Sostituisci il suffisso dell'ARN del bucket (username) con il suffisso scelto nella sezione precedente. Sostituisci l'ID account di esempio (012345678901) nel ruolo di esecuzione del servizio con l'ID account.

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

Esegui [CreateApplication](#) con la seguente richiesta per creare l'applicazione:

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

L'applicazione è ora creata. Avvia l'applicazione nella fase successiva.

Avvia l'applicazione

In questa sezione, viene utilizzata l'operazione [StartApplication](#) per avviare l'applicazione.

Per avviare l'applicazione

1. Salvare il seguente codice JSON in un file denominato `start_request.json`.

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

2. Esegui l'operazione `StartApplication` con la richiesta precedente per avviare l'applicazione:

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

L'applicazione è ora in esecuzione. Puoi controllare i parametri del servizio gestito per Apache Flink sulla CloudWatch console Amazon per verificare che l'applicazione funzioni.

Interruzione dell'applicazione

In questa sezione, viene utilizzata l'operazione [StopApplication](#) per interrompere l'applicazione.

Per interrompere l'applicazione

1. Salvare il seguente codice JSON in un file denominato `stop_request.json`.

```
{
  "ApplicationName": "sliding_window"
}
```

2. Esegui l'operazione `StopApplication` con la richiesta precedente per interrompere l'applicazione:

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

L'applicazione è ora interrotta.

Aggiungi un'opzione CloudWatch di registrazione

Puoi utilizzare la AWS CLI per aggiungere un flusso di CloudWatch log Amazon all'applicazione. Per ulteriori informazioni sull'utilizzo CloudWatch dei log con l'applicazione, consulta [Configurazione della registrazione dell'applicazione](#).

Aggiornamento delle proprietà dell'ambiente

In questa sezione, viene utilizzata l'operazione [UpdateApplication](#) per modificare le proprietà di ambiente per l'applicazione senza ricompilare il codice dell'applicazione. In questo esempio, viene modificata la regione dei flussi di origine e destinazione.

Per aggiornare le proprietà di ambiente per l'applicazione

1. Salvare il seguente codice JSON in un file denominato `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. Esegui l'operazione `UpdateApplication` con la richiesta precedente per aggiornare le proprietà di ambiente:

```
aws kinesisanalyticstv2 update-application --cli-input-json file://
update_properties_request.json
```

Aggiornamento del codice dell'applicazione

Quando devi aggiornare il codice dell'applicazione con una nuova versione del pacchetto di codice, utilizza l'operazione della [UpdateApplication](#) CLI.

Note

Per caricare una nuova versione del codice dell'applicazione con lo stesso nome file, è necessario specificare la nuova versione dell'oggetto. Per ulteriori informazioni sull'uso delle versioni degli oggetti Amazon S3, consulta [Abilitazione o disattivazione del controllo delle versioni](#).

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione ed effettua la chiamata a Amazon S3 e lo stesso nome

dell'oggettoUpdateApplication, nonché la nuova versione dell'oggetto. L'applicazione verrà riavviata con il nuovo pacchetto di codice.

Il seguente esempio di richiesta per l'operazione UpdateApplication ricarica il codice dell'applicazione e riavvia l'applicazione. Aggiorna l'CurrentApplicationVersionId alla versione corrente dell'applicazione. Puoi controllare la versione corrente dell'applicazione utilizzando le operazioni ListApplications o DescribeApplication. Aggiorna il suffisso del nome del bucket (<username>) con il suffisso che hai scelto nella sezione [Creazione delle risorse dipendenti](#).

```
{
  "ApplicationName": "sliding_window",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationCodeConfigurationUpdate": {
      "CodeContentUpdate": {
        "S3ContentLocationUpdate": {
          "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
          "FileKeyUpdate": "-1.0.jar",
          "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"
        }
      }
    }
  }
}
```

Pulizia AWS delle risorse

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Finestra scorrevole.

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione dei flussi di dati Kinesis](#)
- [Eliminazione dell'oggetto e del bucket Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e quindi conferma l'eliminazione.

Eliminazione dei flussi di dati Kinesis

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com /kines.](https://console.aws.amazon.com/kinesis)
2. Nel pannello del flusso di dati Kinesis, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.
4. Nella pagina Kinesis Streams, scegli, scegli Azioni ExampleOutputStream, scegli Elimina, quindi conferma l'eliminazione.

Eliminazione dell'oggetto e del bucket Amazon S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il **<username>**secchio ka-app-code -.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

Eliminazione delle risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-west-2.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytics--us-west-2. MyApplication
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

Esempio: invio di dati di streaming ad Amazon S3 in Scala

Note

Per gli esempi attuali, vedi [Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink](#).

Note

A partire dalla versione 1.15, Flink non supporta più Scala. Le applicazioni possono ora utilizzare l'API Java da qualsiasi versione di Scala. Flink utilizza ancora Scala internamente in alcuni componenti chiave, ma non espone Scala nel classloader del codice utente. Per questo motivo, gli utenti devono aggiungere le dipendenze di Scala nei propri archivi jar. Per ulteriori informazioni sulle modifiche a Scala in Flink 1.15, consulta [Scala non più disponibile nella versione 1.15](#).

In questo esercizio, viene creata una semplice applicazione di streaming che utilizza Scala 3.2.0 e l'API Java DataStream di Flink. L'applicazione legge i dati dal flusso Kinesis, li aggrega utilizzando finestre scorrevoli e scrive i risultati su S3.

Note

Per impostare i prerequisiti richiesti per questo esercizio, completa prima l'esercizio [Nozioni di base \(Scala\)](#). È sufficiente creare una cartella aggiuntiva **data/** nel ka-app-code bucket Amazon S3 -. <username>

Questo argomento contiene le sezioni seguenti:

- [Scarica ed esamina il codice dell'applicazione](#)
- [Compilazione e caricamento del codice dell'applicazione](#)
- [Creazione ed esecuzione dell'applicazione \(console\)](#)
- [Creazione ed esecuzione dell'applicazione \(CLI\)](#)
- [Aggiornamento del codice dell'applicazione](#)
- [Pulizia AWS delle risorse](#)

Scarica ed esamina il codice dell'applicazione

Il codice dell'applicazione Python per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito.

1. Installa il client Git se non lo hai già fatto. Per ulteriori informazioni, consulta [Installazione di Git](#).
2. Clona il repository remoto con il comando seguente:

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

3. Passa alla directory `amazon-kinesis-data-analytics-java-examples/scala/S3Sink`.

Tieni presente quanto segue riguardo al codice dell'applicazione:

- Un file `build.sbt` contiene le informazioni sulla configurazione e le dipendenze dell'applicazione, incluse le librerie del servizio gestito per Apache Flink.
- Il file `BasicStreamingJob.scala` contiene il metodo principale che definisce la funzionalità dell'applicazione.
- L'applicazione utilizza un'origine Kinesis per leggere dal flusso di origine. Il seguente snippet crea l'origine Kinesis:

```
private def createSource: FlinkKinesisConsumer[String] = {  
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties  
  val inputProperties = applicationProperties.get("ConsumerConfigProperties")  
  
  new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,  
    defaultInputStreamName),  
    new SimpleStringSchema, inputProperties)  
}
```

L'applicazione utilizza inoltre un `StreamingFileSink` per scrivere in un bucket Amazon S3:

```
def createSink: StreamingFileSink[String] = {  
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties  
  val s3SinkPath =  
    applicationProperties.get("ProducerConfigProperties").getProperty("s3.sink.path")  
  
  StreamingFileSink  
    .forRowFormat(new Path(s3SinkPath), new SimpleStringEncoder[String]("UTF-8"))  
    .build()  
}
```

- L'applicazione crea connettori di origine e sink per accedere alle risorse esterne utilizzando un `StreamExecutionEnvironment` oggetto.
- L'applicazione crea connettori di origine e sink utilizzando proprietà dinamiche. Questi metodi leggono le proprietà dell'applicazione di runtime per configurare il connettori. Per ulteriori informazioni sulle proprietà di runtime, consulta [Proprietà di runtime](#).

Compilazione e caricamento del codice dell'applicazione

In questa sezione, il codice dell'applicazione viene compilato e caricato in un bucket Amazon S3.

Compilazione del codice dell'applicazione

Userai lo strumento di compilazione [SBT](#) per creare il codice Scala per l'applicazione. Per installare SBT, consulta [Installazione di sbt con setup cs](#). Dovrai inoltre installare il Java Development Kit (JDK). Consulta [Prerequisiti per il completamento degli esercizi](#).

1. Per usare il codice dell'applicazione, compila il codice e comprimilo in un file JAR. Puoi compilare e comprimere il codice con SBT:

```
sbt assembly
```

2. Se l'applicazione viene compilata correttamente, viene creato il seguente file:

```
target/scala-3.2.0/s3-sink-scala-1.0.jar
```

Caricamento del codice Scala di streaming di Apache Flink

In questa sezione, viene creato un bucket Amazon S3 e caricato il codice dell'applicazione.

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Seleziona Crea bucket.
3. Immetti `ka-app-code-<username>` nel campo Nome bucket. Aggiungi un suffisso al nome del bucket, ad esempio il nome utente, per renderlo globalmente univoco. Scegli Next (Successivo).
4. Nella fase Configura opzioni, non modificare le impostazioni e scegli Successivo.
5. Nella fase Imposta autorizzazioni, non modificare le impostazioni e scegli Successivo.
6. Seleziona Crea bucket.
7. Scegli il bucket `ka-app-code-<username>`, quindi scegli Carica.
8. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `s3-sink-scala-1.0.jar` creato nella fase precedente.
9. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione ed esecuzione dell'applicazione (console)

Segui questi passaggi per creare, configurare, aggiornare ed eseguire l'applicazione utilizzando la console.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo `/flink https://console.aws.amazon.com`
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.
3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:
 - Per Nome applicazione, immetti **MyApplication**.
 - Per Descrizione, inserisci **My java test app**.
 - Per Runtime, scegli Apache Flink.
 - Lascia la versione Apache Flink 1.15.2 (versione consigliata).

4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
5. Scegli Crea applicazione.

Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`
- Ruolo: `kinesisanalytics-MyApplication-us-west-2`

Configurazione dell'applicazione

Per configurare l'applicazione, utilizza la procedura seguente.

Per configurare l'applicazione

1. Nella MyApplicationpagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:
 - Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **s3-sink-scala-1.0.jar**
3. In Accesso alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
4. In Proprietà, scegli Aggiungi gruppo.
5. Immetti i seguenti dati:

ID gruppo	Chiave	Valore
ConsumerConfigProperties	input.stream.name	ExampleInputStream

ID gruppo	Chiave	Valore
ConsumerConfigProperties	aws.region	us-west-2
ConsumerConfigProperties	flink.stream.initpos	LATEST

Scegli Save (Salva).

- In Proprietà, scegli Aggiungi gruppo.
- Immetti i seguenti dati:

ID gruppo	Chiave	Valore
ProducerConfigProperties	s3.sink.path	s3a://ka-app-code- <user-name> /data

- In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
- Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.
- Scegli Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione Amazon, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: /aws/kinesis-analytics/MyApplication
- Flusso di log: kinesis-analytics-log-stream

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere al bucket Amazon S3.

Modifica della policy IAM per aggiungere le autorizzazioni per i bucket S3

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
3. Nella pagina Riepilogo, scegli Modifica policy. Scegli la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (**012345678901**) con il tuo ID account.

```
{
  "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"
    }
]
}

```

Esecuzione dell'applicazione.

Il grafico del processo Flink può essere visualizzato eseguendo l'applicazione, aprendo il pannello di controllo di Apache Flink e scegliendo il processo Flink desiderato.

Interruzione dell'applicazione

Per interrompere l'applicazione, nella MyApplication pagina, scegli Stop. Conferma l'operazione.

Creazione ed esecuzione dell'applicazione (CLI)

In questa sezione, verrà utilizzata la AWS Command Line Interface per creare ed eseguire il servizio gestito per Apache Flink. Utilizza il AWS CLI comando della `kinesisanalyticsv2` per creare e interagire con il servizio gestito per Apache Flink.

Creazione di una policy di autorizzazione

Note

È necessario creare una policy di autorizzazione e un ruolo per l'applicazione. Se non crei queste risorse IAM, l'applicazione non può accedere ai suoi dati e flussi di log.

Innanzitutto, crea una policy di autorizzazione con due istruzioni: una che concede le autorizzazioni per l'operazione di lettura sul flusso di origine e un'altra che concede le autorizzazioni per operazioni di scrittura sul flusso di sink. Collega quindi la policy a un ruolo IAM (che verrà creato nella sezione successiva). Pertanto, quando il servizio gestito per Apache Flink assume il ruolo, il servizio disporrà delle autorizzazioni necessarie per leggere dal flusso di origine e scrivere nel flusso di sink.

Utilizza il codice seguente per creare la policy di autorizzazione

AKReadSourceStreamWriteSinkStream. Sostituisci **username** con il nome utente utilizzato per creare il bucket Amazon S3 per archiviare il codice dell'applicazione. Sostituisci l'ID account nei nomi della risorsa Amazon (ARNs) (**012345678901**) con il tuo ID account.

```
{
  "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"
  }
]
}

```

Per step-by-step istruzioni su come creare una politica di autorizzazioni, consulta il [Tutorial: Create and Attach Your First Customer Managed Policy](#) nella IAM User Guide.

Creazione di un ruolo IAM

In questa sezione, viene creato un ruolo IAM per l'applicazione del servizio gestito per Apache Flink che può essere assunto per leggere un flusso di origine e scrivere nel flusso di sink.

Il servizio gestito per Apache Flink non può accedere al tuo flusso senza autorizzazioni. Queste autorizzazioni possono essere assegnate con un ruolo IAM. Ad ogni ruolo IAM sono collegate due policy. La policy di attendibilità concede al servizio gestito per Apache Flink l'autorizzazione per assumere il ruolo e la policy di autorizzazione determina cosa può fare il servizio assumendo questo ruolo.

Collega la policy di autorizzazione creata nella sezione precedente a questo ruolo.

Per creare un ruolo IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nel riquadro di navigazione, scegli Ruoli e quindi Crea ruolo.
3. In Seleziona tipo di entità attendibile, scegli Servizio AWS .
4. In Scegli il servizio che utilizzerà questo ruolo, scegli Kinesis.
5. In Seleziona il tuo caso d'uso, scegli Servizio gestito per Apache Flink.
6. Scegli Successivo: autorizzazioni.
7. Nella pagina Allega policy di autorizzazione, seleziona Successivo: esamina. Collega le policy di autorizzazione dopo aver creato il ruolo.
8. Nella pagina Crea ruolo, immetti **MF-stream-rw-role** per Nome ruolo. Scegli Crea ruolo.

È stato creato un nuovo ruolo IAM denominato MF-stream-rw-role. Successivamente, aggiorna le policy di trust e di autorizzazione per il ruolo

9. Collega la policy di autorizzazione al ruolo.

Note

Per questo esercizio, il servizio gestito per Apache Flink assume questo ruolo per la lettura di dati da un flusso di dati Kinesis (origine) e la scrittura dell'output in un altro flusso di dati Kinesis. Pertanto, devi collegare la policy creata nella fase precedente, [Creazione di una policy di autorizzazione](#).

- a. Nella pagina Riepilogo, scegli la scheda Autorizzazioni.

- b. Scegliere Collega policy.
- c. Nella casella di ricerca, immetti **AKReadSourceStreamWriteSinkStream** (la policy creata nella sezione precedente).
- d. Scegli la policy AKReadSourceStreamWriteSinkStream e seleziona Collega policy.

È stato creato il ruolo di esecuzione del servizio che l'applicazione utilizzerà per accedere alle risorse. Prendi nota dell'ARN del nuovo ruolo.

Per step-by-step istruzioni sulla creazione di un ruolo, consulta [Creating an IAM Role \(Console\)](#) nella IAM User Guide.

Creazione dell'applicazione

Salvare il seguente codice JSON in un file denominato `create_request.json`. Sostituisci l'ARN del ruolo di esempio con l'ARN per il ruolo creato in precedenza. Sostituisci il suffisso dell'ARN del bucket (username) con il suffisso scelto nella sezione precedente. Sostituisci l'ID account di esempio (012345678901) nel ruolo di esecuzione del servizio con l'ID account.

```
{
  "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"
  }
]
}

```

Esegui [CreateApplication](#) con la seguente richiesta per creare l'applicazione:

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

L'applicazione è ora creata. Avvia l'applicazione nella fase successiva.

Avvia l'applicazione

In questa sezione, viene utilizzata l'operazione [StartApplication](#) per avviare l'applicazione.

Per avviare l'applicazione

1. Salvare il seguente codice JSON in un file denominato `start_request.json`.

```

{
  "ApplicationName": "s3_sink",
  "RunConfiguration": {
    "ApplicationRestoreConfiguration": {
      "ApplicationRestoreType": "RESTORE_FROM_LATEST_SNAPSHOT"
    }
  }
}

```

2. Esegui l'operazione `StartApplication` con la richiesta precedente per avviare l'applicazione:

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

L'applicazione è ora in esecuzione. Puoi controllare i parametri del servizio gestito per Apache Flink sulla CloudWatch console Amazon per verificare che l'applicazione funzioni.

Interruzione dell'applicazione

In questa sezione, viene utilizzata l'operazione [StopApplication](#) per interrompere l'applicazione.

Per interrompere l'applicazione

1. Salvare il seguente codice JSON in un file denominato `stop_request.json`.

```
{
  "ApplicationName": "s3_sink"
}
```

2. Esegui l'operazione `StopApplication` con la richiesta precedente per interrompere l'applicazione:

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

L'applicazione è ora interrotta.

Aggiungi un'opzione CloudWatch di registrazione

Puoi utilizzare la AWS CLI per aggiungere un flusso di CloudWatch log Amazon all'applicazione. Per ulteriori informazioni sull'utilizzo CloudWatch dei log con l'applicazione, consulta [Configurazione della registrazione dell'applicazione](#).

Aggiornamento delle proprietà dell'ambiente

In questa sezione, viene utilizzata l'operazione [UpdateApplication](#) per modificare le proprietà di ambiente per l'applicazione senza ricompilare il codice dell'applicazione. In questo esempio, viene modificata la regione dei flussi di origine e destinazione.

Per aggiornare le proprietà di ambiente per l'applicazione

1. Salvare il seguente codice JSON in un file denominato `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. Esegui l'operazione `UpdateApplication` con la richiesta precedente per aggiornare le proprietà di ambiente:

```
aws kinesisanalyticstv2 update-application --cli-input-json file://
update_properties_request.json
```

Aggiornamento del codice dell'applicazione

Quando devi aggiornare il codice dell'applicazione con una nuova versione del pacchetto di codice, utilizza l'operazione della [UpdateApplication](#) CLI.

Note

Per caricare una nuova versione del codice dell'applicazione con lo stesso nome file, è necessario specificare la nuova versione dell'oggetto. Per ulteriori informazioni sull'uso delle

versioni degli oggetti Amazon S3, consulta [Abilitazione o disattivazione del controllo delle versioni](#).

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione ed effettua la chiamata a Amazon S3 e lo stesso nome dell'oggetto `UpdateApplication`, nonché la nuova versione dell'oggetto. L'applicazione verrà riavviata con il nuovo pacchetto di codice.

Il seguente esempio di richiesta per l'operazione `UpdateApplication` ricarica il codice dell'applicazione e riavvia l'applicazione. Aggiorna `CurrentApplicationVersionId` alla versione corrente dell'applicazione. Puoi controllare la versione corrente dell'applicazione utilizzando le operazioni `ListApplications` o `DescribeApplication`. Aggiorna il suffisso del nome del bucket (`<username>`) con il suffisso che hai scelto nella sezione [Creazione delle risorse dipendenti](#).

```
{
  "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": "SAMPLEUehYngP87ex1nzYIGYgfhyvDU"
        }
      }
    }
  }
}
```

Pulizia AWS delle risorse

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Finestra a cascata.

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione dei flussi di dati Kinesis](#)
- [Eliminazione dell'oggetto e del bucket Amazon S3](#)

- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e quindi conferma l'eliminazione.

Eliminazione dei flussi di dati Kinesis

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com /kines.](https://console.aws.amazon.com/kines.)
2. Nel pannello del flusso di dati Kinesis, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.
4. Nella pagina Kinesis Streams, scegli, scegli Azioni ExampleOutputStream, scegli Elimina, quindi conferma l'eliminazione.

Eliminazione dell'oggetto e del bucket Amazon S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il **<username>**secchio ka-app-code -.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

Eliminazione delle risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-west-2.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytics--us-west-2. MyApplication

8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Quindi scegli Elimina gruppo di log e conferma l'eliminazione.

Usa un notebook Studio con Managed Service per Apache Flink

I notebook Studio per il servizio gestito per Apache Flink consentono di interrogare in modo interattivo i flussi di dati in tempo reale e di creare ed eseguire facilmente applicazioni di elaborazione dei flussi utilizzando SQL, Python e Scala standard. Con pochi clic nella console di AWS gestione, puoi avviare un notebook serverless per interrogare i flussi di dati e ottenere risultati in pochi secondi.

Un notebook è un ambiente di sviluppo basato sul Web. Con i notebook, si ottiene un'esperienza di sviluppo semplice e interattiva combinata con le funzionalità avanzate fornite da Apache Flink. I notebook Studio utilizzano notebook basati su [Apache Zeppelin](#) e utilizzano [Apache Flink](#) come motore di elaborazione del flusso. I notebook Studio combinano perfettamente queste tecnologie per rendere le analisi avanzate sui flussi di dati accessibili a sviluppatori con tutte le competenze.

Apache Zeppelin fornisce ai notebook Studio una suite completa di strumenti di analisi, tra cui:

- Visualizzazione dei dati
- Esportazione di dati nei file
- Controllo del formato di output per un'analisi più semplice

Per una guida all'utilizzo del servizio gestito per Apache Flink e Apache Zeppelin, consulta [Tutorial: crea un notebook Studio in Managed Service per Apache Flink](#). Per ulteriori informazioni su Apache Zeppelin, consulta la [documentazione su Apache Zeppelin](#).

[Con un notebook, puoi modellare le query utilizzando l'API Apache Flink Table e SQL in SQL, Python o Scala o l'API in Scala. DataStream](#) Con pochi clic, puoi quindi promuovere il notebook Studio a un'applicazione di elaborazione di flussi del servizio gestito per Apache Flink in esecuzione continua e non interattiva per i carichi di lavoro di produzione.

Questo argomento contiene le sezioni seguenti:

- [Usa la versione corretta di Studio Notebook Runtime](#)
- [Crea un taccuino da Studio](#)
- [Esegui un'analisi interattiva dei dati in streaming](#)
- [Implementa come applicazione con uno stato durevole](#)
- [Verifica le autorizzazioni IAM per i notebook Studio](#)

- [Usa connettori e dipendenze](#)
- [Implementa funzioni definite dall'utente](#)
- [Abilita il checkpointing](#)
- [Aggiorna Studio Runtime](#)
- [Lavora con AWS Glue](#)
- [Esempi e tutorial per notebook Studio in Managed Service for Apache Flink](#)
- [Risolvi i problemi relativi ai notebook Studio per Managed Service for Apache Flink](#)
- [Crea policy IAM personalizzate per i notebook Managed Service for Apache Flink Studio](#)

Usa la versione corretta di Studio Notebook Runtime

Con Amazon Managed Service per Apache Flink Studio, puoi interrogare i flussi di dati in tempo reale e creare ed eseguire applicazioni di elaborazione dei flussi utilizzando SQL, Python e Scala standard in un notebook interattivo. [I notebook Studio sono basati su Apache Zeppelin e utilizzano Apache Flink come motore di elaborazione dello stream.](#)

Note

Deprecheremo Studio Runtime con la versione 1.11 di Apache Flink il 5 novembre 2024. A partire da questa data, non sarà possibile eseguire nuovi notebook o creare nuove applicazioni utilizzando questa versione. Si consiglia di eseguire l'aggiornamento al runtime più recente (Apache Flink 1.15 e Apache Zeppelin 0.10) prima di tale data. Per indicazioni su come aggiornare il notebook, consulta [Aggiorna Studio Runtime](#)

Studio Runtime

Versione Apache Flink	Versione Apache Zeppelin	Versione di Python	
1.15	0.1	3.8	Consigliato
1.13	0.9	3.8	Supportata fino al 16 ottobre 2024

Versione Apache Flink	Versione Apache Zeppelin	Versione di Python	
1.11	0.9	3.7	Data di scadenza il 24 febbraio 2025

Crea un taccuino da Studio

Un notebook Studio contiene query o programmi scritti in SQL, Python o Scala che vengono eseguiti su dati di streaming e restituiscono risultati analitici. Puoi creare la tua applicazione utilizzando la console o la CLI e fornire query per l'analisi dei dati dalla tua origine dati.

L'applicazione ha i seguenti componenti:

- Un'origine dati, ad esempio un cluster Amazon MSK, un flusso di dati Kinesis o un bucket Amazon S3.
- Un AWS Glue database. Questo database contiene tabelle in cui sono archiviati gli schemi e gli endpoint di origine e destinazione dei dati. Per ulteriori informazioni, consulta [Lavora con AWS Glue](#).
- Il tuo codice dell'applicazione. Il codice implementa la tua query o il tuo programma di analisi.
- Le impostazioni dell'applicazione e le proprietà di runtime. Per informazioni sulle impostazioni dell'applicazione e le proprietà di runtime, consulta i seguenti argomenti nella [Guida per gli sviluppatori di applicazioni Apache Flink](#):
 - Parallelismo e dimensionamento delle applicazioni: l'impostazione Parallelismo dell'applicazione serve per controllare il numero di query che l'applicazione può eseguire contemporaneamente. Le query possono inoltre trarre vantaggio da un aumento del parallelismo se hanno più percorsi di esecuzione, ad esempio nelle seguenti circostanze:
 - Durante l'elaborazione di più partizioni di un flusso di dati Kinesis
 - Durante il partizionamento dei dati utilizzando l'operatore KeyBy.
 - Quando si utilizzano più operatori finestra

Per ulteriori informazioni sul dimensionamento dell'applicazione, consulta [Dimensionamento delle applicazioni nel servizio gestito per Apache Flink](#).

- Registrazione e monitoraggio: per informazioni sulla registrazione e il monitoraggio delle applicazioni, consulta [Registrazione e monitoraggio nel servizio gestito da Amazon per Apache Flink](#).
- La tua applicazione utilizza checkpoint e savepoint per la tolleranza agli errori. I checkpoint e i savepoint non sono abilitati per impostazione predefinita per i notebook Studio.

Puoi creare il tuo taccuino Studio utilizzando AWS Management Console o AWS CLI.

Quando crei l'applicazione dalla console, hai a disposizione le seguenti opzioni:

- Nella console Amazon MSK, scegli un cluster, quindi scegli **Elabora dati in tempo reale**.
- Nella console del flusso di dati Kinesis, scegli un flusso di dati, quindi nella scheda **Applicazioni** scegli **Elabora dati in tempo reale**.
- Nella console del servizio gestito per Apache Flink, scegli la scheda **Studio**, quindi scegli **Crea notebook Studio**.

Per un tutorial, consulta [Rilevamento degli eventi con il servizio gestito per Apache Flink](#).

Per un esempio di una soluzione notebook Studio più avanzata, consulta [Apache Flink sul servizio gestito da Amazon per Apache Flink Studio](#).

Esegui un'analisi interattiva dei dati in streaming

Utilizza un notebook serverless basato su Apache Zeppelin per interagire con i tuoi dati di streaming. Il notebook può contenere più note e ogni nota può contenere uno o più paragrafi in cui scrivere il codice.

L'esempio seguente di query SQL mostra come recuperare dati da un'origine dati:

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

Per altri esempi di query SQL di Flink Streaming, consulta [Esempi e tutorial per notebook Studio in Managed Service for Apache Flink](#) quanto segue e [Query](#) nella documentazione di Apache Flink.

È possibile utilizzare le query SQL di Flink nel notebook Studio per interrogare i dati di streaming. Puoi anche usare Python (Table API) e Scala (Table and Datastream APIs) per scrivere programmi

per interrogare i tuoi dati di streaming in modo interattivo. Puoi visualizzare i risultati delle query o dei programmi, aggiornarli in pochi secondi ed eseguirli nuovamente per visualizzare i risultati aggiornati.

Interpreti Flink

Puoi specificare la lingua utilizzata dal servizio gestito per Apache Flink per eseguire l'applicazione utilizzando un interprete. Con il servizio gestito da Amazon per Apache Flink puoi utilizzare i seguenti interpreti:

Nome	Classe	Descrizione
<code>%flink</code>	<code>FlinkInterpreter</code>	Crea <code>ExecutionEnvironment/StreamExecutionEnvironment/BatchTableEnvironment/StreamTableEnvironment</code> fornisce un ambiente Scala
<code>%flink.pyflink</code>	<code>PyFlinkInterpreter</code>	Fornisce un ambiente python
<code>%flink.ipynlink</code>	<code>IPyFlinkInterpreter</code>	Fornisce un ambiente ipython
<code>%flink.ssql</code>	<code>FlinkStreamSqlInterpreter</code>	Fornisce un ambiente SQL di flusso
<code>%flink.bsqli</code>	<code>FlinkBatchSqlInterpreter</code>	Fornisce un ambiente sql in batch

Per ulteriori informazioni sugli interpreti Flink, consulta [Interprete Flink per Apache Zeppelin](#).

Se si utilizzano `%flink.pyflink` o `%flink.ipynlink` come interpreti, è necessario utilizzare il `ZeppelinContext` per visualizzare i risultati all'interno del notebook.

Per esempi più PyFlink specifici, consulta [Interroga i flussi di dati in modo interattivo utilizzando Managed Service per Apache Flink Studio](#) e Python.

Variabili dell'ambiente tabellare Apache Flink

Apache Zeppelin fornisce l'accesso alle risorse dell'ambiente tabellare utilizzando variabili di ambiente.

Puoi accedere alle risorse dell'ambiente tabellare Scala con le seguenti variabili:

Variabile	Risorsa
<code>senv</code>	<code>StreamExecutionEnvironment</code>
<code>stenv</code>	<code>StreamTableEnvironment for blink planner</code>

Puoi accedere alle risorse dell'ambiente tabellare Python con le seguenti variabili:

Variabile	Risorsa
<code>s_env</code>	<code>StreamExecutionEnvironment</code>
<code>st_env</code>	<code>StreamTableEnvironment for blink planner</code>

Per ulteriori informazioni sull'utilizzo degli ambienti tabellari, consulta [Concetti e API comuni](#) nella documentazione di Apache Flink.

Implementa come applicazione con uno stato durevole

Puoi creare il codice ed esportarlo in Amazon S3. Puoi promuovere il codice che hai scritto nella nota in un'applicazione di elaborazione di flussi in esecuzione continua. Esistono due modalità per eseguire un'applicazione Apache Flink nel servizio gestito per Apache Flink: con un notebook Studio, hai la possibilità di sviluppare il codice in modo interattivo, visualizzarne i risultati in tempo reale e visualizzarlo all'interno della nota. Dopo aver implementato una nota per l'esecuzione in modalità streaming, il servizio gestito per Apache Flink crea un'applicazione che viene eseguita continuamente, legge i dati dalle origini, scrive nelle destinazioni, mantiene lo stato dell'applicazione a lungo termine e viene automaticamente dimensionato in base al throughput dei flussi di origine.

Note

Il bucket S3 in cui si esporta il codice dell'applicazione deve trovarsi nella stessa regione del notebook Studio.

È possibile implementare una nota dal notebook Studio solo se soddisfa i seguenti criteri:

- I paragrafi devono essere ordinati in sequenza. Quando distribuite l'applicazione, tutti i paragrafi all'interno di una nota verranno eseguiti in sequenza (left-to-right, top-to-bottom) così come appaiono nella nota. È possibile controllare questo ordine selezionando Esegui tutti i paragrafi nella nota.
- Il tuo codice è una combinazione di Python ed SQL o di Scala ed SQL. Al momento non supportiamo Python e Scala insieme per. `deploy-as-application`
- La nota dovrebbe avere solo i seguenti interpreti: `%flink`, `%flink.sql`, `%flink.pyflink`, `%flink.ipynb`, `%md`.
- L'uso dell'oggetto `z` [del contesto Zeppelin](#) non è supportato. I metodi che non restituiscono nulla si limiteranno a registrare un avviso. Altri metodi genereranno eccezioni in Python o non riusciranno a compilare in Scala.
- Una nota deve generare un singolo processo Apache Flink.
- Le note con i [moduli dinamici](#) non sono supportate per l'implementazione come applicazione.
- I paragrafi `%md` ([Markdown](#)) verranno ignorati durante l'implementazione come applicazione, poiché si prevede che contengano documentazione leggibile dall'uomo che non è adatta all'esecuzione come parte dell'applicazione risultante.
- I paragrafi disabilitati per l'esecuzione in Zeppelin verranno ignorati durante l'implementazione come applicazione. Anche se un paragrafo disabilitato utilizza un interprete incompatibile, ad esempio `%flink.ipynb` in una nota con `%flink` and `%flink.sql` interpreti, verrà ignorato durante l'utilizzo della nota come applicazione e non genererà alcun errore.
- Affinché la distribuzione dell'applicazione abbia successo, deve essere presente almeno un paragrafo con il codice sorgente (Flink SQL PyFlink o Flink Scala) abilitato per l'esecuzione.
- L'impostazione del parallelismo nella direttiva dell'interprete all'interno di un paragrafo (ad esempio `%flink.sql(parallelism=32)`) verrà ignorata nelle applicazioni implementate da una nota. È invece possibile aggiornare l'applicazione distribuita tramite l' AWS API AWS Command Line Interface o per modificare le AWS Management Console impostazioni di parallelismo e/o ParallelismPer KPU in base al livello di parallelismo richiesto dall'applicazione oppure è possibile abilitare la scalabilità automatica per l'applicazione distribuita.
- Se stai implementando come applicazione con stato durevole, il VPC deve avere accesso a Internet. Se il VPC non dispone di accesso a Internet, consulta [Implementa come applicazione con uno stato durevole in un VPC senza accesso a Internet](#).

Criteri Scala/Python

- Nel tuo codice Scala o Python, usa il [pianificatore Blink](#) (`senv`, `stenv` per Scala; `s_env`, `st_env` Python) e non il vecchio pianificatore "Flink" (`stenv_2` per Scala, `st_env_2` per Python). Il progetto Apache Flink consiglia l'utilizzo del pianificatore Blink per i casi d'uso in produzione, e questo è il pianificatore predefinito in Zeppelin e Flink.
- I tuoi paragrafi in Python non devono utilizzare [invocazioni/assegnazioni di shell](#) utilizzando `!` o [comandi IPython magici](#) come `%timeit` o `%conda` nelle note pensate per essere distribuite come applicazioni.
- Non è possibile utilizzare le classi di casi Scala come parametri di funzioni passate a operatori di flusso di dati di ordine superiore come `map` e `filter`. Per informazioni sulle classi di casi di Scala, consulta [CLASSI DI CASI](#) nella documentazione di Scala.

Criteri SQL

- Le istruzioni `SELECT` semplici non sono consentite, in quanto non esiste nulla di equivalente alla sezione di output di un paragrafo in cui è possibile fornire i dati.
- In ogni paragrafo, le istruzioni DDL (`USE`, `CREATE`, `ALTER`, `DROP`, `SET`, `RESET`) devono precedere le istruzioni DML (`INSERT`). Questo perché le istruzioni DML contenute in un paragrafo devono essere inviate insieme come un unico processo Flink.
- Dovrebbe esserci al massimo un paragrafo contenente istruzioni DML. Questo perché, per questa `deploy-as-application` funzionalità, supportiamo solo l'invio di un singolo lavoro a Flink.

Per ulteriori informazioni e un esempio, consulta [Traduzione, redazione e analisi dei dati di streaming utilizzando le funzioni SQL con il servizio gestito da Amazon per Apache Flink, Amazon Translate e Amazon Comprehend](#).

Verifica le autorizzazioni IAM per i notebook Studio

Il servizio gestito per Apache Flink crea automaticamente un ruolo IAM quando crei un notebook Studio tramite la AWS Management Console. Inoltre associa a quel ruolo una policy che consente i seguenti accessi:

Servizio	Accesso
CloudWatch Registri	Elenco
Amazon EC2	Elenco
AWS Glue	Lettura/scrittura
Servizio gestito per Apache Flink	Lettura
Servizio gestito per Apache Flink V2	Lettura
Amazon S3	Lettura/scrittura

Usa connettori e dipendenze

I connettori consentono di leggere e scrivere dati utilizzando tecnologie diverse. Il servizio gestito per Apache Flink include tre connettori predefiniti nel notebook Studio. Puoi inoltre utilizzare connettori personalizzati. Per ulteriori informazioni sui connettori, consulta [Connettori tabella ed SQL](#) nella documentazione di Apache Flink.

Connettori predefiniti

Se utilizzi il AWS Management Console per creare il tuo notebook Studio, Managed Service for Apache Flink include i seguenti connettori personalizzati per impostazione predefinita: `flink-sql-connector-kinesis`, e `flink-connector-kafka_2.12 aws-msk-iam-auth`. Per creare un notebook Studio tramite la console senza questi connettori personalizzati, scegli l'opzione Crea con impostazioni personalizzate. Quindi, quando arrivi alla pagina Configurazioni, deselecta le caselle di controllo accanto ai due connettori.

Se utilizzi l'[CreateApplication](#) API per creare il tuo notebook Studio, i `flink-connector-kafka` connettori `flink-sql-connector-flink` e non sono inclusi per impostazione predefinita. Per aggiungerli, specificali come MavenReference nel tipo di dati CustomArtifactsConfiguration, come mostrato negli esempi seguenti.

Il connettore `aws-msk-iam-auth` è il connettore da utilizzare con Amazon MSK che include la funzionalità di autenticazione automatica con IAM.

 Note

Le versioni dei connettori mostrate nell'esempio seguente sono le uniche supportate.

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"  
  }  
}]
```

Per aggiungere questi connettori a un notebook esistente, utilizza l'operazione [UpdateApplication](#) API e specificali come MavenReference tipo di CustomArtifactsConfigurationUpdate dati.

Note

È possibile impostare `failOnError` su `true` per il connettore `flink-sql-connector-kinesis` nell'API della tabella.

Aggiungi dipendenze e connettori personalizzati

Per utilizzare il AWS Management Console per aggiungere una dipendenza o un connettore personalizzato al tuo notebook Studio, procedi nel seguente modo:

1. Carica il file del connettore personalizzato in Amazon S3.
2. In AWS Management Console, scegli l'opzione di creazione personalizzata per creare il tuo taccuino Studio.
3. Segui il flusso di lavoro per la creazione del notebook Studio fino alla fase Configurazioni.
4. Nella sezione Connettori personalizzati, scegli Aggiungi connettore personalizzato.
5. Specifica la posizione Amazon S3 della dipendenza o del connettore personalizzato.
6. Scegli Save changes (Salva modifiche).

Per aggiungere un JAR di dipendenza o un connettore personalizzato quando crei un nuovo notebook Studio utilizzando l'[CreateApplication](#) API, specifica la posizione Amazon S3 del JAR di dipendenza o del connettore personalizzato nel CustomArtifactsConfiguration tipo di dati. Per aggiungere una dipendenza o un connettore personalizzato a un notebook Studio esistente, richiama l'operazione [UpdateApplication](#) API e specifica la posizione Amazon S3 del JAR della dipendenza o del connettore personalizzato nel tipo di dati. CustomArtifactsConfigurationUpdate

Note

Quando includi una dipendenza o un connettore personalizzato, devi inserire anche tutte le relative dipendenze transitive non incluse al suo interno.

Implementa funzioni definite dall'utente

Le funzioni definite dall'utente (UDFs) sono punti di estensione che consentono di richiamare la logica utilizzata di frequente o la logica personalizzata che non può essere espressa diversamente nelle query. Puoi usare Python o un linguaggio JVM come Java o Scala per implementare i paragrafi all'interno del tuo UDFs notebook Studio. Puoi anche aggiungere al tuo notebook Studio file JAR esterni che contengono file JAR UDFs implementati in un linguaggio JVM.

Quando implementate le classi astratte di JARs quel registro `UserDefinedFunction` (o le vostre classi astratte), utilizzate l'ambito fornito in Apache Maven, le dichiarazioni di `compileOnly` dipendenza in Gradle, l'ambito fornito in SBT o una direttiva equivalente nella configurazione di build del progetto UDF. Ciò consente la compilazione del codice sorgente UDF con Flink, ma le classi API Flink APIs non sono a loro volta incluse negli artefatti di compilazione. Fai riferimento a questo [pom](#) tratto dal jar UDF di esempio, che rispetta tale prerequisito su un progetto Maven.

Note

Per un esempio di configurazione, consulta [Traduzione, redazione e analisi dei dati di streaming utilizzando le funzioni SQL con il servizio gestito da Amazon per Apache Flink, Amazon Translate e Amazon Comprehend](#) sul blog Machine Learning AWS .

Per utilizzare la console per aggiungere file JAR UDF al notebook Studio, segui questi passaggi:

1. Carica il file JAR UDF su Amazon S3.
2. In AWS Management Console, scegli l'opzione di creazione personalizzata per creare il tuo taccuino Studio.
3. Segui il flusso di lavoro per la creazione del notebook Studio fino alla fase Configurazioni.
4. Nella sezione Funzioni definite dall'utente, scegli Aggiungi una funzione definita dall'utente.
5. Specifica la posizione Amazon S3 del file JAR o del file ZIP che contiene l'implementazione dell'UDF.
6. Scegli Save changes (Salva modifiche).

Per aggiungere un JAR UDF quando crei un nuovo notebook Studio utilizzando l'[CreateApplication](#) API, specifica la posizione JAR nel tipo di `CustomArtifactConfiguration` dati. Per aggiungere un file JAR UDF a un notebook Studio esistente,

richiamate l'operazione [UpdateApplication](#) API e specificate la posizione JAR nel `CustomArtifactsConfigurationUpdate` tipo di dati. In alternativa, è possibile utilizzare il AWS Management Console per aggiungere file JAR UDF al notebook Studio.

Considerazioni sulle funzioni definite dall'utente

- Il servizio gestito per Apache Flink Studio utilizza la [terminologia di Apache Zeppelin](#), in cui un notebook è un'istanza Zeppelin che può contenere più note. Ogni nota può quindi contenere più paragrafi. Con il servizio gestito per Apache Flink Studio, il processo di interpretazione è condiviso tra tutte le note del taccuino. Quindi, se si esegue una registrazione esplicita della [createTemporarySystemfunzione utilizzando Function](#) in una nota, è possibile fare riferimento alla stessa così com'è in un'altra nota dello stesso taccuino.

L'operazione Implementa come applicazione funziona tuttavia su una singola nota e non su tutte le note del notebook. Quando si esegue l'implementazione come applicazione, per generare l'applicazione vengono utilizzati solo i contenuti della nota attiva. Qualsiasi registrazione esplicita di funzioni eseguita in altri notebook non fa parte delle dipendenze dell'applicazione generate. Inoltre, mentre l'opzione Implementa come applicazione è in esecuzione, si verifica una registrazione implicita della funzione convertendo il nome della classe principale del JAR in una stringa minuscola.

Ad esempio, se `TextAnalyticsUDF` è la classe principale per il JAR UDF, una registrazione implicita darà come risultato il nome della funzione `textanalyticsudf`. Quindi, se la registrazione esplicita di una funzione nella nota 1 di Studio si verifica come segue, tutte le altre note in quel notebook (ad esempio la nota 2) possono fare riferimento alla funzione in base al nome `myNewFuncNameForClass` grazie all'interprete condiviso:

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

Tuttavia, durante l'operazione di implementazione come applicazione sulla nota 2, questa registrazione esplicita non verrà inclusa nelle dipendenze e quindi l'applicazione implementata non funzionerà come previsto. A causa della registrazione implicita, per impostazione predefinita tutti i riferimenti a questa funzione dovrebbero essere con `textanalyticsudf` e non con `myNewFuncNameForClass`.

Se è necessaria una registrazione personalizzata del nome della funzione, si prevede che la nota 2 stessa contenga un altro paragrafo per eseguire un'altra registrazione esplicita come segue:

```
%flink(parallelism=1)
import com.amazonaws.kinesis.udf.textanalytics.TextAnalyticsUDF
# re-register the JAR for UDF with custom name
stenv.createTemporarySystemFunction("myNewFuncNameForClass", new TextAnalyticsUDF())
```

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

- Se il codice JAR UDF include Flink SDKs, configurate il progetto Java in modo che il codice sorgente UDF possa essere compilato con Flink SDKs, ma le classi Flink SDK non siano esse stesse incluse nell'elemento di build, ad esempio il JAR.

È possibile utilizzare l'ambito `provided` in Apache Maven, dichiarazioni di dipendenza `compileOnly` in Gradle, l'ambito `provided` in SBT o una direttiva equivalente nella configurazione di build del progetto UDF. Puoi fare riferimento a questo [pom](#) tratto dal jar UDF di esempio, che rispetta tale prerequisito su un progetto Maven. Per un step-by-step tutorial completo, consulta questo articolo [Traduci, correggi e analizza i dati di streaming utilizzando le funzioni SQL con Amazon Managed Service for Apache Flink, Amazon Translate e Amazon Comprehend](#).

Abilita il checkpointing

È possibile abilitare la creazione di checkpoint utilizzando le impostazioni dell'ambiente. Per informazioni sulla creazione di checkpoint, consulta [Tolleranza agli errori](#) nella [Guida per gli sviluppatori del servizio gestito per Apache Flink](#).

Imposta l'intervallo di checkpoint

Il seguente esempio di codice Scala imposta l'intervallo di checkpoint dell'applicazione su un minuto:

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

Il seguente esempio di codice Python imposta l'intervallo di checkpoint dell'applicazione su un minuto:

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

Imposta il tipo di checkpoint

Il seguente esempio di codice Scala imposta la modalità di checkpoint dell'applicazione su EXACTLY_ONCE (impostazione predefinita):

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

Il seguente esempio di codice Python imposta la modalità di checkpoint dell'applicazione su EXACTLY_ONCE (impostazione predefinita):

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

Aggiorna Studio Runtime

Questa sezione contiene informazioni su come aggiornare il notebook Studio Runtime. Ti consigliamo di eseguire sempre l'aggiornamento all'ultima versione supportata di Studio Runtime.

Aggiorna il tuo notebook a un nuovo Studio Runtime

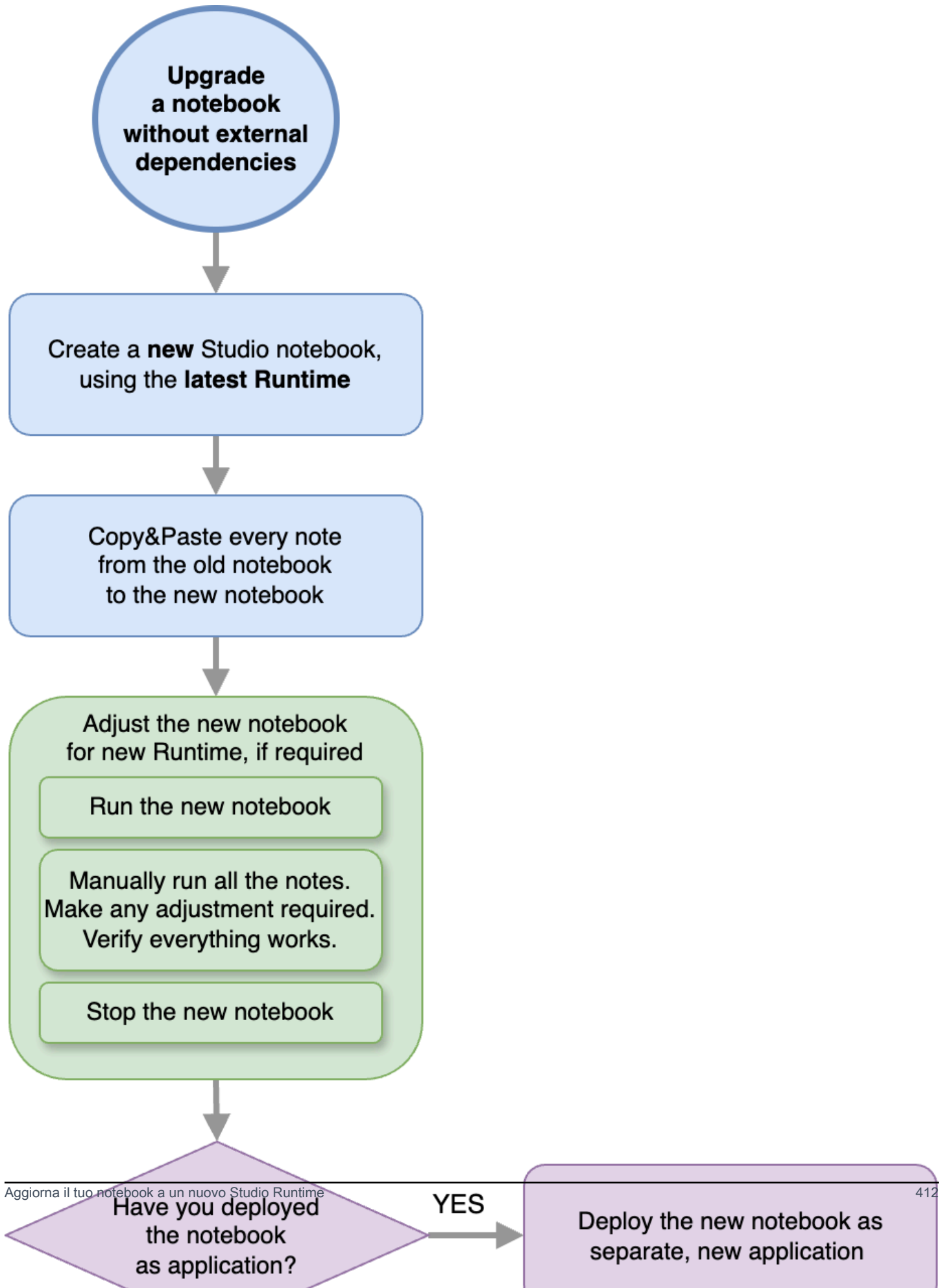
A seconda di come utilizzi Studio, i passaggi per aggiornare il tuo Runtime sono diversi. Seleziona l'opzione più adatta al tuo caso d'uso.

Query SQL o codice Python senza dipendenze esterne

Se si utilizza SQL o Python senza dipendenze esterne, utilizzare il seguente processo di aggiornamento del Runtime. Ti consigliamo di eseguire l'aggiornamento alla versione più recente di Runtime. Il processo di aggiornamento è lo stesso, indipendentemente dalla versione di Runtime da cui si esegue l'aggiornamento.

1. Crea un nuovo notebook Studio utilizzando la versione più recente del Runtime.
2. Copia e incolla il codice di ogni nota dal vecchio taccuino al nuovo taccuino.
3. Nel nuovo taccuino, modifica il codice per renderlo compatibile con qualsiasi funzionalità di Apache Flink modificata rispetto alla versione precedente.
 - Esegui il nuovo notebook. Apri il taccuino ed esegui la nota per nota, in sequenza, e verifica se funziona.
 - Apporta le modifiche necessarie al codice.
 - Arresta il nuovo notebook.
4. Se hai distribuito il vecchio notebook come applicazione:
 - Implementa il nuovo notebook come nuova applicazione separata.
 - Arresta la vecchia applicazione.
 - Esegui la nuova applicazione senza istantanea.
5. Arresta il vecchio notebook se è in esecuzione. Avvia il nuovo notebook, se necessario, per un uso interattivo.

Flusso di processo per l'aggiornamento senza dipendenze esterne

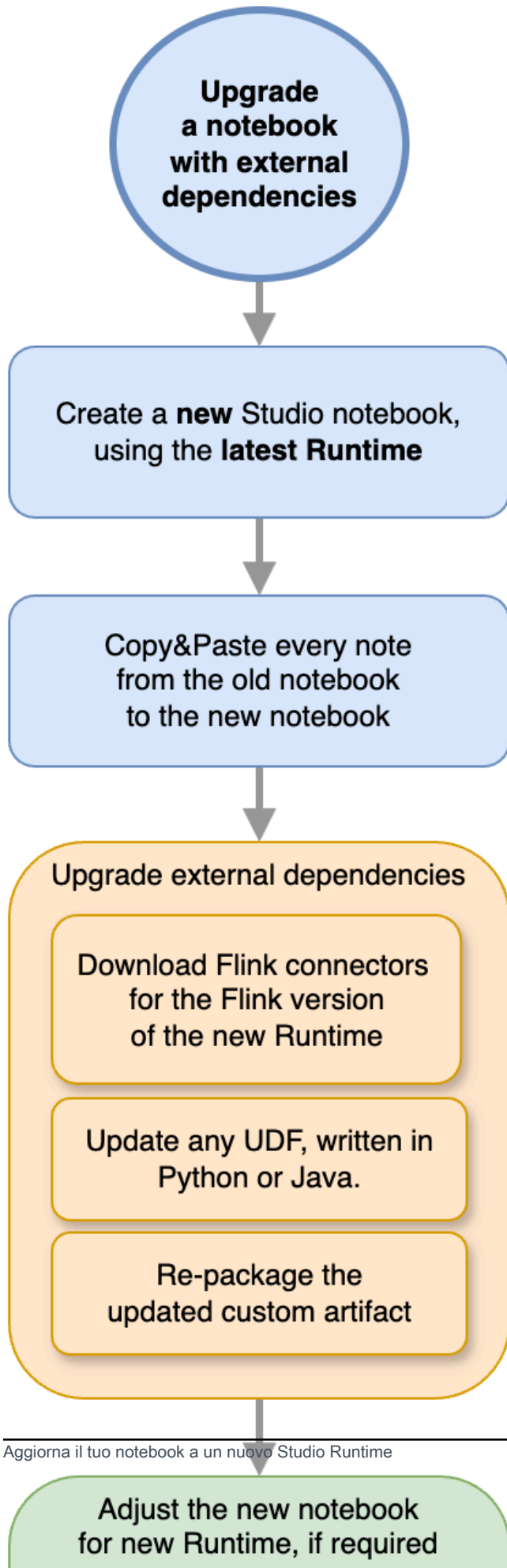


Query SQL o codice Python con dipendenze esterne

Segui questo processo se utilizzi SQL o Python e utilizzi dipendenze esterne come connettori o artefatti personalizzati, come funzioni definite dall'utente implementate in Python o Java. Ti consigliamo di eseguire l'aggiornamento alla versione più recente del Runtime. Il processo è lo stesso, indipendentemente dalla versione di Runtime da cui si esegue l'aggiornamento.

1. Crea un nuovo notebook Studio utilizzando la versione più recente del Runtime.
2. Copia e incolla il codice di ogni nota dal vecchio taccuino al nuovo taccuino.
3. Aggiorna le dipendenze esterne e gli artefatti personalizzati.
 - Cerca nuovi connettori compatibili con la versione Apache Flink del nuovo Runtime. Fate riferimento a [Table & SQL Connectors](#) nella documentazione di Apache Flink per trovare i connettori corretti per la versione Flink.
 - Aggiorna il codice delle funzioni definite dall'utente in modo che corrisponda alle modifiche nell'API Apache Flink e a qualsiasi dipendenza Python o JAR utilizzata dalle funzioni definite dall'utente. Reimpacchetta il tuo artefatto personalizzato aggiornato.
 - Aggiungi questi nuovi connettori e artefatti al nuovo notebook.
4. Nel nuovo notebook, modifica il codice per renderlo compatibile con qualsiasi funzionalità di Apache Flink modificata rispetto alla versione precedente.
 - Esegui il nuovo notebook. Apri il taccuino ed esegui nota per nota, in sequenza, e verifica se funziona.
 - Apporta le modifiche necessarie al codice.
 - Arresta il nuovo notebook.
5. Se hai distribuito il vecchio notebook come applicazione:
 - Implementa il nuovo notebook come nuova applicazione separata.
 - Arresta la vecchia applicazione.
 - Esegui la nuova applicazione senza istantanea.
6. Arresta il vecchio notebook se è in esecuzione. Avvia il nuovo notebook, se necessario, per un uso interattivo.

Flusso di processo per l'aggiornamento con dipendenze esterne



Lavora con AWS Glue

Il notebook Studio archivia e riceve informazioni sulle fonti di dati e sui relativi sink. AWS Glue Quando si crea il notebook Studio, si specifica il AWS Glue database che contiene le informazioni di connessione. Quando si accede alle fonti di dati e ai sink, si specificano le AWS Glue tabelle contenute nel database. AWS Glue Le tabelle forniscono l'accesso alle AWS Glue connessioni che definiscono le posizioni, gli schemi e i parametri delle fonti e delle destinazioni dei dati.

I notebook Studio utilizzano le proprietà delle tabelle per archiviare dati specifici dell'applicazione. Per ulteriori informazioni, consulta [Proprietà tabella](#).

Per un esempio di come configurare una AWS Glue connessione, un database e una tabella da utilizzare con i notebook Studio, consulta [Creare un database AWS Glue](#) il tutorial. [Tutorial: crea un notebook Studio in Managed Service per Apache Flink](#)

Proprietà tabella

Oltre ai campi dati, le AWS Glue tabelle forniscono altre informazioni al notebook di Studio utilizzando le proprietà delle tabelle. Managed Service for Apache Flink utilizza le seguenti proprietà della AWS Glue tabella:

- [Definisci i valori temporali di Apache Flink](#): queste proprietà definiscono il modo in cui il servizio gestito per Apache Flink genera i valori del tempo di elaborazione dei dati interni di Apache Flink.
- [Usa il connettore Flink e le proprietà di formato](#): queste proprietà forniscono informazioni sui flussi di dati.

Per aggiungere una proprietà a una AWS Glue tabella, procedi come segue:

1. Accedi a AWS Management Console e apri la AWS Glue console all'indirizzo <https://console.aws.amazon.com/glue/>.
2. Dall'elenco di tabelle, seleziona la tabella che l'applicazione utilizza per memorizzare le informazioni sulla connessione dati. Scegli Operazione > Modifica dettagli tabella.
3. In Proprietà tabella, inserisci **managed-flink.proctime** per chiave e **user_action_time** per valore.

Definisci i valori temporali di Apache Flink

Apache Flink fornisce valori temporali che descrivono quando si sono verificati eventi di elaborazione del flusso, come [Tempo di elaborazione](#) e [Ora evento](#). Per includere questi valori nell'output dell'applicazione, definisci le proprietà nella AWS Glue tabella che indicano al runtime di Managed Service for Apache Flink di emettere questi valori nei campi specificati.

Le chiavi e i valori utilizzati nelle proprietà della tabella sono i seguenti:

Tipo timestamp	Chiave	Valore
Tempo di elaborazione	managed-flink.proctime	Il nome della colonna che AWS Glue verrà utilizzato per esporre il valore. Questo nome di colonna non corrisponde a una colonna di tabella esistente.
Ora dell'evento	managed-flink.rowtime	Il nome della colonna che AWS Glue verrà utilizzato per esporre il valore. Questo nome di colonna corrisponde a una colonna di tabella esistente.
	managed-flink.watermark. <i>column_name</i> .millisecondi	L'intervallo della filigrana in millisecondi

Usa il connettore Flink e le proprietà di formato

Fornisci informazioni sulle origini dati ai connettori Flink dell'applicazione utilizzando le proprietà della tabella AWS Glue . Di seguito sono riportati alcuni esempi delle proprietà utilizzate per i connettori dal servizio gestito per Apache Flink:

Tipo di connettore	Chiave	Valore
Kafka	format	Il formato utilizzato per deserializzare e serializz

Tipo di connettore	Chiave	Valore
Kinesis		are i messaggi di Kafka, ad esempio o. json csv
	scan.startup.mode	La modalità di avvio per il consumatore Kafka, ad esempio o. earliest-offset timestamp
	format	Il formato utilizzato per deserializzare e serializzare i record del flusso di dati Kinesis, ad esempio o. json csv
	aws.region	La AWS regione in cui è definito lo stream.
S3 (Filesystem)	format	Il formato utilizzato per deserializzare e serializzare i file, ad esempio o. json csv
	path	Il percorso Amazon S3, ad es. s3://mybucket/

Per ulteriori informazioni su altri connettori oltre a Kinesis e Apache Kafka, consulta la documentazione relativa al tuo connettore.

Esempi e tutorial per notebook Studio in Managed Service for Apache Flink

Argomenti

- [Tutorial: creare un notebook Studio in Managed Service per Apache Flink](#)
- [Tutorial: distribuzione di un notebook Studio come servizio gestito per l'applicazione Apache Flink con stato durevole](#)
- [Visualizza esempi di query per analizzare i dati in un notebook Studio](#)

Tutorial: creare un notebook Studio in Managed Service per Apache Flink

Il seguente tutorial mostra come creare un notebook Studio che legge i dati da un flusso di dati Kinesis o da un cluster Amazon MSK.

Questo tutorial contiene le sezioni seguenti:

- [Completa i prerequisiti](#)
- [Creare un database AWS Glue](#)
- [Passaggi successivi: creare un notebook Studio con Kinesis Data Streams o Amazon MSK](#)
- [Creazione di un notebook Studio con il flusso di dati Kinesis](#)
- [Crea un notebook Studio con Amazon MSK](#)
- [Pulisci l'applicazione e le risorse dipendenti](#)

Completa i prerequisiti

Assicurati che la tua AWS CLI sia la versione 2 o successiva. Per installare la versione più recente AWS CLI, consulta [Installazione, aggiornamento e disinstallazione della AWS CLI versione 2](#).

Creare un database AWS Glue

Il notebook Studio utilizza un database [AWS Glue](#) per i metadati sull'origine dati Amazon MSK.

Creare un AWS Glue database

1. Apri la AWS Glue console all'indirizzo <https://console.aws.amazon.com/glue/>.
2. Scegli Aggiungi database. Nella finestra Aggiungi database, inserisci **default** per Nome database. Scegli Create (Crea) .

Passaggi successivi: creare un notebook Studio con Kinesis Data Streams o Amazon MSK

Con questo tutorial, puoi creare un notebook Studio che utilizza il flusso di dati Kinesis o Amazon MSK:

- [Creazione di un notebook Studio con il flusso di dati Kinesis](#): con il flusso di dati Kinesis, viene creata rapidamente un'applicazione che utilizza un flusso di dati Kinesis come origine. È sufficiente creare un flusso di dati Kinesis come risorsa dipendente.

- [Crea un notebook Studio con Amazon MSK](#): con Amazon MSK, viene creata un'applicazione che utilizza un cluster Amazon MSK come origine. È necessario creare un Amazon VPC, un'istanza EC2 client Amazon e un cluster Amazon MSK come risorse dipendenti.

Creazione di un notebook Studio con il flusso di dati Kinesis

Questo tutorial descrive come creare un notebook Studio che utilizza un flusso di dati Kinesis come origine.

Questo tutorial contiene le sezioni seguenti:

- [Completa i prerequisiti](#)
- [Crea una tabella AWS Glue](#)
- [Creazione di un notebook Studio con il flusso di dati Kinesis](#)
- [Invio di dati al flusso di dati Kinesis](#)
- [Test del notebook Studio](#)

Completa i prerequisiti

Prima di creare un notebook Studio, è necessario creare un flusso di dati Kinesis (ExampleInputStream). L'applicazione utilizza questo flusso come origine dell'applicazione.

Puoi creare questo flusso utilizzando la console Amazon Kinesis o il comando AWS CLI seguente. Per istruzioni sulla console, consulta [Creazione e aggiornamento dei flussi di dati](#) nella Guida per gli sviluppatori del flusso di dati Amazon Kinesis. Assegna un nome al flusso **ExampleInputStream** e imposta il Numero di partizioni aperte su **1**.

Per creare lo stream (ExampleInputStream) utilizzando il AWS CLI, usa il seguente comando Amazon Kinesis create-stream AWS CLI .

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

Crea una tabella AWS Glue

Il notebook Studio utilizza un database [AWS Glue](#) per i metadati sull'origine dati del flusso di dati Kinesis.

Note

Puoi creare prima il database manualmente oppure lasciare che il servizio gestito per Apache Flink lo crei automaticamente quando viene creato il notebook. Allo stesso modo, è possibile creare manualmente la tabella come descritto in questa sezione oppure utilizzare il codice del connettore di creazione tabella per il servizio gestito per Apache Flink nel notebook all'interno di Apache Zeppelin per creare la tabella tramite un'istruzione DDL. Puoi quindi effettuare il check-in AWS Glue per assicurarti che la tabella sia stata creata correttamente.

Creazione di una tabella

1. Accedi a AWS Management Console e apri la AWS Glue console all'indirizzo <https://console.aws.amazon.com/glue/>.
2. Se non disponi già di un AWS Glue database, scegli Database dalla barra di navigazione a sinistra. Scegli Aggiungi database. Nella finestra Aggiungi database, inserisci **default** per Nome database. Scegli Create (Crea) .
3. Nella barra di navigazione a sinistra, seleziona Tabelle. Nella pagina Tabelle, scegli Aggiungi tabelle > Aggiungi tabella manualmente.
4. Nella pagina Imposta le proprietà della tabella, inserisci **stock** per Nome tabella. Assicurati di selezionare il database creato in precedenza. Scegli Next (Successivo).
5. Nella pagina Aggiungi un datastore, scegli Kinesis. Per Nome del flusso, inserisci **ExampleInputStream**. Per URL di origine di Kinesis, inserisci **https://kinesis.us-east-1.amazonaws.com**. Se copi e incolli l'URL di origine di Kinesis, assicurati di eliminare gli spazi iniziali o finali. Scegli Next (Successivo).
6. Nella pagina Classificazione, scegli JSON. Scegli Next (Successivo).
7. Nella pagina Definisci uno schema, scegli Aggiungi colonna per aggiungere una colonna. Aggiungi colonne con le seguenti proprietà:

Nome colonna	Tipo di dati
ticker	string
price	double

Scegli Next (Successivo).

8. Nella pagina successiva, verifica le impostazioni e scegli Fine.
9. Scegli la tabella appena creata dall'elenco delle tabelle.
10. Scegli Modifica tabella e aggiungi una proprietà con la chiave `managed-flink.proctime` e il valore `proctime`.
11. Scegli Applica.

Creazione di un notebook Studio con il flusso di dati Kinesis

Ora che hai creato le risorse utilizzate dall'applicazione, puoi creare il notebook Studio.

Per creare la tua applicazione, puoi usare il AWS Management Console o il AWS CLI.

- [Crea un taccuino Studio utilizzando il AWS Management Console](#)
- [Crea un taccuino Studio utilizzando il AWS CLI](#)

Crea un taccuino Studio utilizzando il AWS Management Console

1. [Aprire la console Managed Service for Apache Flink a casa https://console.aws.amazon.com/managed-flink/?region=us-east-1#/applicazioni/dashboard](https://console.aws.amazon.com/managed-flink/?region=us-east-1#/applicazioni/dashboard).
2. Nella pagina Applicazioni del servizio gestito per Apache Flink, scegli la scheda Studio. Scegli Crea notebook Studio.

Note

Puoi anche creare un notebook Studio dalle console Amazon MSK o del flusso di dati Kinesis selezionando il cluster Amazon MSK o il flusso di dati Kinesis di input e scegliendo *Elabora dati in tempo reale*.

3. Nella pagina Crea notebook Studio, immetti le seguenti informazioni:

- Inserisci **MyNotebook** per il nome del notebook.
- Scegli l'impostazione predefinita per il database AWS Glue.

Scegli Crea notebook Studio.

4. Nella pagina, scegli Esegui. MyNotebook Attendi che lo stato mostri In esecuzione. Si applicano costi quando il notebook è in funzione.

Crea un taccuino Studio utilizzando il AWS CLI

Per creare il tuo taccuino Studio utilizzando AWS CLI, procedi come segue:

1. Verifica l'ID del tuo account. Questo valore è necessario per creare l'applicazione.
2. Crea il ruolo `arn:aws:iam::AccountID:role/ZeppeleinRole` e aggiungi le seguenti autorizzazioni al ruolo creato automaticamente dalla console.

`"kinesis:GetShardIterator",`

`"kinesis:GetRecords",`

`"kinesis:ListShards"`

3. Crea un file denominato `create.json` con i seguenti contenuti. Sostituisci i valori segnaposto con le tue informazioni.

```
{
  "ApplicationName": "MyNotebook",
  "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
  "ApplicationMode": "INTERACTIVE",
  "ServiceExecutionRole": "arn:aws:iam::AccountID:role/ZeppeleinRole",
  "ApplicationConfiguration": {
    "ApplicationSnapshotConfiguration": {
      "SnapshotsEnabled": false
    },
    "ZeppelinApplicationConfiguration": {
      "CatalogConfiguration": {
        "GlueDataCatalogConfiguration": {
          "DatabaseARN": "arn:aws:glue:us-east-1:AccountID:database/default"
        }
      }
    }
  }
}
```

```
}  
}  
}
```

4. Per creare l'applicazione, esegui il comando riportato di seguito:

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

5. Una volta completata l'esecuzione del comando, visualizzerai un output che mostra i dettagli per il nuovo notebook Studio. Di seguito è riportato un esempio di output.

```
{  
  "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. Per avviare l'applicazione, esegui il comando riportato di seguito. Sostituisci il valore di esempio con il tuo ID account.

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

Invio di dati al flusso di dati Kinesis

Per inviare i dati di test al flusso di dati Kinesis, procedi come segue:

1. Apri [Kinesis Data Generator](#).
2. Scegli Crea un utente Cognito con. CloudFormation
3. La AWS CloudFormation console si apre con il modello Kinesis Data Generator. Scegli Next (Successivo).
4. Nella pagina Specifica i dettagli dello stack, inserisci il nome utente e la password per l'utente Cognito. Scegli Next (Successivo).
5. Nella pagina Configura opzioni dello stack, scegli Successivo.

6. Nella pagina Review Kinesis-Data-Generator-Cognito -User, scegli l'opzione Riconosco che AWS CloudFormation potrebbe creare le risorse IAM. casella di controllo. Scegli Crea stack.
7. Attendi che lo AWS CloudFormation stack finisca di essere creato. Una volta completato lo stack, apri lo stack Kinesis-Data-Generator-Cognito-User nella console e scegli la scheda Output. AWS CloudFormation KinesisDataGeneratorUrlApri l'URL elencato per il valore di output.
8. Nella pagina Amazon Kinesis Data Generator, accedi con le credenziali create nel passaggio 4.
9. Nella pagina successiva, specifica i seguenti valori:

Region	us-east-1
Stream/Flusso Firehose	ExampleInputStream
Record al secondo	1

Per Modello di record, incolla il seguente codice:

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

10. Scegli Invia dati.
11. Il generatore invierà dati al flusso di dati Kinesis.

Lascia il generatore in esecuzione mentre completi la sezione successiva.

Test del notebook Studio

In questa sezione, il notebook Studio viene utilizzato per eseguire query sui dati del flusso di dati Kinesis.

1. [Aprire la console Managed Service for Apache Flink a casa https://console.aws.amazon.com/managed-flink/?region=us-east-1#/applicazioni/dashboard](https://console.aws.amazon.com/managed-flink/?region=us-east-1#/applicazioni/dashboard).
2. Nella pagina Applicazioni del servizio gestito per Apache Flink, scegli la scheda Notebook Studio. Scegli MyNotebook.
3. Nella pagina, scegli Apri in Apache Zeppelin. MyNotebook

L'interfaccia Apache Zeppelin viene aperta in una nuova scheda.

4. Nella sezione Ti diamo il benvenuto su Zeppelin!, scegli Nota Zeppelin.
5. Nella pagina Nota Zeppelin, inserisci la seguente query in una nuova nota:

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

Seleziona l'icona dell'esecuzione.

Dopo un breve periodo, la nota visualizza i dati del flusso di dati Kinesis.

Per aprire il pannello di controllo di Apache Flink per la tua applicazione e visualizzare gli aspetti operativi, scegli PROCESSO FLINK. Per ulteriori informazioni sul pannello di controllo di Flink, consulta [Pannello di controllo di Apache Flink](#) nella [Guida per gli sviluppatori del servizio gestito per Apache Flink](#).

Per altri esempi di query Streaming SQL in Flink, consulta [Query](#) nella [documentazione di Apache Flink](#).

Crea un notebook Studio con Amazon MSK

Questo tutorial descrive come creare un notebook Studio che utilizza un cluster Amazon MSK come origine.

Questo tutorial contiene le sezioni seguenti:

- [Configurare un cluster Amazon MSK](#)
- [Aggiungi un gateway NAT al tuo VPC](#)
- [Crea una AWS Glue connessione e una tabella](#)
- [Crea un notebook Studio con Amazon MSK](#)
- [Invio di dati al cluster Amazon MSK](#)

- [Test del notebook Studio](#)

Configurare un cluster Amazon MSK

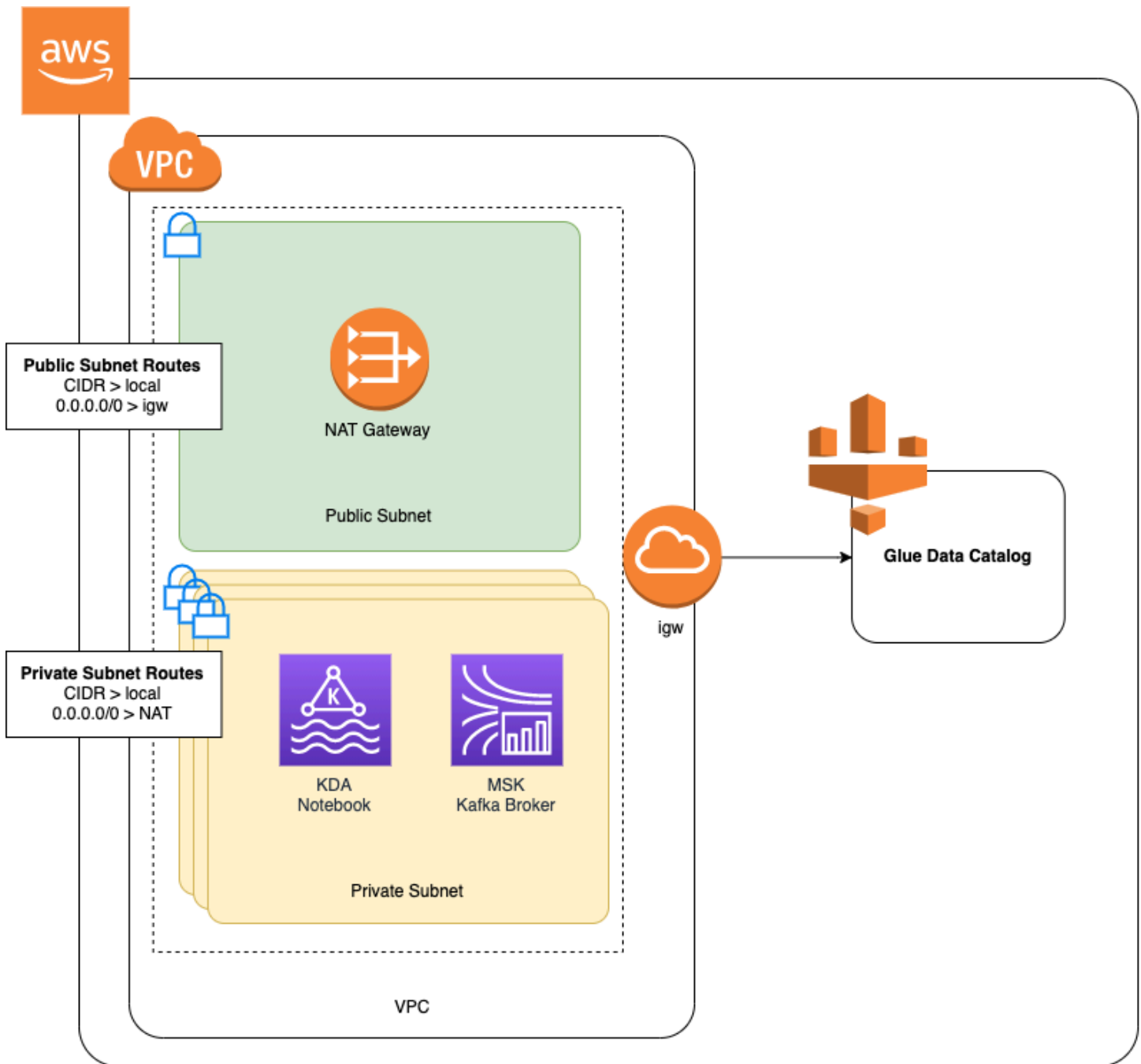
Per questo tutorial, è necessario un cluster Amazon MSK che consenta l'accesso al testo in chiaro. Se non disponi già di un cluster Amazon MSK, segui il tutorial [Getting Started Using Amazon MSK](#) per creare un Amazon VPC, un cluster Amazon MSK, un argomento e un'istanza client Amazon. EC2

Seguendo il tutorial, completa le seguenti operazioni:

- Nella [Fase 3: creazione di un cluster Amazon MSK](#), passaggio 4, modifica il valore `ClientBroker` da TLS a **PLAINTEXT**.

Aggiungi un gateway NAT al tuo VPC

Se hai creato un cluster Amazon MSK seguendo il tutorial [Nozioni di base per l'uso di Amazon MSK](#) o se il tuo Amazon VPC esistente non dispone già di un gateway NAT per le sue sottoreti private, devi aggiungere un gateway NAT al tuo Amazon VPC. Il diagramma seguente illustra l'architettura generale.



Per creare un gateway NAT per il tuo Amazon VPC, procedi come segue:

1. Apri la console Amazon VPC all'indirizzo <https://console.aws.amazon.com/vpc/>.
2. Scegli Gateway NAT dalla barra di navigazione a sinistra.
3. Nella pagina Gateway NAT, scegli Crea gateway NAT.
4. Nella pagina Crea gateway NAT, specifica i seguenti valori:

Nome: opzionale

ZeppelinGateway

Sottorete

AWS KafkaTutorialSubnet1

ID di allocazione IP elastico

Scegli un IP elastico disponibile. Se non è IP disponibile alcun Elastic IP, scegli Allocate Elastic IP, quindi scegli l'IP Elastic creato dalla console.

Scegli Crea un gateway NAT.

5. Nella barra di navigazione a sinistra, seleziona Tabelle di routing.
6. Seleziona Crea tabella di routing.
7. Nella pagina Crea tabella di routing, fornisci le seguenti informazioni:
 - Tag nome: **ZeppelinRouteTable**
 - VPC: scegli il tuo VPC (ad esempio VPC).AWS KafkaTutorial

Scegli Create (Crea) .

8. Nell'elenco delle tabelle dei percorsi, scegli. ZeppelinRouteTable Seleziona la scheda Route, seleziona Modifica route.
9. Nella scheda Modifica route scegli Aggiungi route.
10. Per Destinazione, inserisci **0.0.0.0/0**. Per Target, scegli NAT Gateway, ZeppelinGateway. Seleziona Salva route. Scegli Chiudi.
11. Nella pagina Tabelle delle rotte, con l'ZeppelinRouteTableopzione selezionata, scegli la scheda Associazioni di sottoreti. Scegli Modifica associazioni sottorete.
12. Nella pagina Modifica associazioni di sottoreti, scegli AWS KafkaTutorialSubnet2 e AWS KafkaTutorialSubnet 3. Seleziona Salva.

Crea una AWS Glue connessione e una tabella

Il notebook Studio utilizza un database [AWS Glue](#) per i metadati sull'origine dati Amazon MSK. In questa sezione, crei una AWS Glue connessione che descrive come accedere al tuo cluster Amazon

MSK è una AWS Glue tabella che descrive come presentare i dati della tua origine dati a client come il tuo notebook Studio.

Creazione di una connessione

1. Accedi AWS Management Console e apri la AWS Glue console all'indirizzo <https://console.aws.amazon.com/glue/>.
2. Se non disponi già di un AWS Glue database, scegli Database dalla barra di navigazione a sinistra. Scegli Aggiungi database. Nella finestra Aggiungi database, inserisci **default** per Nome database. Scegli Create (Crea) .
3. Scegli Connessioni dalla barra di navigazione a sinistra. Scegli Aggiungi connessione.
4. Nella finestra Aggiungi connessione, fornisci i seguenti valori:
 - Per Nome connessione, inserisci **ZeppelinConnection**.
 - Per Tipo di connessione, scegli Kafka.
 - Per il server bootstrap Kafka URLs, fornisci la stringa del broker bootstrap per il tuo cluster. È possibile ottenere i broker di bootstrap dalla console MSK o immettendo il seguente comando CLI:

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

- Deseleziona la casella di controllo Richiedi connessione SSL.

Scegli Next (Successivo).

5. Nella pagina VPC, fornisci i seguenti valori:
 - Per VPC, scegli il nome del tuo VPC (ad esempio VPC). AWS KafkaTutorial
 - Per Subnet, scegli 2.AWS KafkaTutorialSubnet
 - Per Gruppi di sicurezza, scegli tutti i gruppi disponibili.

Scegli Next (Successivo).

6. Nella pagina Proprietà di connessione/Accesso alla connessione, scegli Finisci.

Creazione di una tabella

Note

È possibile creare manualmente la tabella come descritto nei passaggi seguenti oppure utilizzare il codice del connettore di creazione tabella per il servizio gestito per Apache Flink nel notebook all'interno di Apache Zeppelin per creare la tabella tramite un'istruzione DDL. È quindi possibile effettuare il check-in AWS Glue per assicurarsi che la tabella sia stata creata correttamente.

1. Nella barra di navigazione a sinistra, seleziona Tabelle. Nella pagina Tabelle, scegli Aggiungi tabelle > Aggiungi tabella manualmente.
2. Nella pagina Imposta le proprietà della tabella, inserisci **stock** per Nome tabella. Assicurati di selezionare il database creato in precedenza. Scegli Next (Successivo).
3. Nella pagina Aggiungi un datastore, scegli Kafka. Per il nome dell'argomento, inserisci il nome dell'argomento (ad es. AWS KafkaTutorialTopic). Per Connessione, scegli ZeppelinConnection.
4. Nella pagina Classificazione, scegli JSON. Scegli Next (Successivo).
5. Nella pagina Definisci uno schema, scegli Aggiungi colonna per aggiungere una colonna. Aggiungi colonne con le seguenti proprietà:

Nome colonna	Tipo di dati
ticker	string
price	double

Scegli Next (Successivo).

6. Nella pagina successiva, verifica le impostazioni e scegli Fine.
7. Scegli la tabella appena creata dall'elenco delle tabelle.
8. Scegliete Modifica tabella e aggiungete le seguenti proprietà:
 - chiave:managed-flink.proctime, valore: proctime
 - chiave:flink.properties.group.id, valore: test-consumer-group
 - chiave:flink.properties.auto.offset.reset, valore: latest

- chiave:classification, valore: json

Senza queste coppie chiave/valore, il notebook Flink genera un errore.

9. Scegli Applica.

Crea un notebook Studio con Amazon MSK

Ora che hai creato le risorse utilizzate dall'applicazione, puoi creare il notebook Studio.

È possibile creare l'applicazione utilizzando il o il AWS Management Console . AWS CLI

- [Crea un taccuino Studio utilizzando il AWS Management Console](#)
- [Crea un taccuino Studio utilizzando il AWS CLI](#)

Note

Un notebook Studio può essere creato anche dalla console Amazon MSK scegliendo un cluster esistente, quindi selezionando **Elabora dati in tempo reale**.

Crea un taccuino Studio utilizzando il AWS Management Console

1. [Aprire la console Managed Service for Apache Flink a casa https://console.aws.amazon.com/managed-flink/?region=us-east-1#/applicazioni/dashboard](https://console.aws.amazon.com/managed-flink/?region=us-east-1#/applicazioni/dashboard).
2. Nella pagina Applicazioni del servizio gestito per Apache Flink, scegli la scheda Studio. Scegli **Crea notebook Studio**.

Note

Per creare un notebook Studio dalle console Amazon MSK o del flusso di dati Kinesis, seleziona il cluster Amazon MSK o il flusso di dati Kinesis di input, quindi scegli **Elabora dati in tempo reale**.

3. Nella pagina **Crea notebook Studio**, immetti le seguenti informazioni:
 - Inserisci **MyNotebook** per Nome notebook Studio.
 - Scegli l'impostazione predefinita per il database AWS Glue.

Scegli Crea notebook Studio.

4. Nella pagina, scegli la scheda Configurazione. MyNotebook Nella sezione Reti, scegli Modifica.
5. Nella MyNotebook pagina Modifica rete per, scegli la configurazione VPC basata sul cluster Amazon MSK. Scegli il cluster Amazon MSK per Cluster Amazon MSK. Scegli Save changes (Salva modifiche).
6. Nella MyNotebook pagina, scegli Esegui. Attendi che lo stato mostri In esecuzione.

Crea un taccuino Studio utilizzando il AWS CLI

Per creare il tuo taccuino Studio utilizzando il AWS CLI, procedi come segue:

1. Assicurati di disporre delle informazioni riportate di seguito. Questi valori sono necessari per creare l'applicazione.
 - ID dell'account.
 - L'ID della sottorete IDs e del gruppo di sicurezza per l'Amazon VPC che contiene il cluster Amazon MSK.
2. Crea un file denominato `create.json` con i seguenti contenuti. Sostituisci i valori segnaposto con le tue informazioni.

```
{
  "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. Per creare l'applicazione, esegui il comando riportato di seguito:

```
aws kinesisanalyticstv2 create-application --cli-input-json file://create.json
```

4. Una volta completata l'esecuzione del comando, dovresti visualizzare un output simile al seguente, che mostra i dettagli per il nuovo notebook Studio:

```

{
  "ApplicationDetail": {
    "ApplicationARN": "arn:aws:kinesisanalyticstv2:us-east-1:012345678901:application/MyNotebook",
    "ApplicationName": "MyNotebook",
    "RuntimeEnvironment": "ZEPPELIN-FLINK-3_0",
    "ApplicationMode": "INTERACTIVE",
    "ServiceExecutionRole": "arn:aws:iam::012345678901:role/ZepplinRole",
    ...
  }
}

```

5. Per avviare l'applicazione, esegui il comando riportato di seguito. Sostituisci il valore segnaposto con il tuo ID account.

```
aws kinesisanalyticstv2 start-application --application-arn
arn:aws:kinesisanalyticstv2:us-east-1:012345678901:application/MyNotebook\
```

Invio di dati al cluster Amazon MSK

In questa sezione, esegui uno script Python nel tuo EC2 client Amazon per inviare dati alla tua origine dati Amazon MSK.

1. Connect al tuo EC2 client Amazon.
2. Esegui i seguenti comandi per installare Python versione 3, Pip e il pacchetto Kafka per Python e conferma le operazioni:

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

3. Configuralo AWS CLI sul tuo computer client inserendo il seguente comando:

```
aws configure
```

Fornisci le credenziali del tuo account e **us-east-1** per region.

4. Crea un file denominato `stock.py` con i seguenti contenuti. Sostituisci il valore di esempio con la stringa Bootstrap Brokers del tuo cluster Amazon MSK e aggiorna il nome dell'argomento se l'argomento non è: `AWS KafkaTutorialTopic`

```
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. Esegui lo script con il comando seguente:

```
$ python3 stock.py
```

6. Lascia lo script in esecuzione mentre completi la sezione seguente.

Test del notebook Studio

In questa sezione, il notebook Studio viene utilizzato per eseguire query sui dati del cluster Amazon MSK.

1. [Aprire la console Managed Service for Apache Flink a casa? https://console.aws.amazon.com/managed-flink/?region=us-east-1#/applicazioni/dashboard](https://console.aws.amazon.com/managed-flink/?region=us-east-1#/applicazioni/dashboard).
2. Nella pagina Applicazioni del servizio gestito per Apache Flink, scegli la scheda Notebook Studio. Scegli MyNotebook.
3. Nella pagina, scegli Apri in Apache Zeppelin. MyNotebook

L'interfaccia di Apache Zeppelin viene aperta in una nuova scheda.

4. Nella pagina Ti diamo il benvenuto su Zeppelin!, scegli Nuova nota Zeppelin.
5. Nella pagina Nota Zeppelin, inserisci la seguente query in una nuova nota:

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

Seleziona l'icona dell'esecuzione.

L'applicazione mostra i dati del cluster Amazon MSK.

Per aprire il pannello di controllo di Apache Flink per la tua applicazione e visualizzare gli aspetti operativi, scegli PROCESSO FLINK. Per ulteriori informazioni sul pannello di controllo di Flink, consulta [Pannello di controllo di Apache Flink](#) nella [Guida per gli sviluppatori del servizio gestito per Apache Flink](#).

Per altri esempi di query Streaming SQL in Flink, consulta [Query](#) nella [documentazione di Apache Flink](#).

Pulisci l'applicazione e le risorse dipendenti

Eliminazione del notebook Studio

1. Apri la console del servizio gestito per Apache Flink.
2. Scegli MyNotebook.
3. Scegli Operazioni, quindi Elimina.

Eliminare il AWS Glue database e la connessione

1. Apri la AWS Glue console all'indirizzo <https://console.aws.amazon.com/glue/>.
2. Scegli Database dalla barra di navigazione a sinistra. Seleziona la casella di controllo accanto a Predefinito per selezionarla. Scegli Operazione, Elimina database. Conferma la selezione.
3. Scegli Connessioni dalla barra di navigazione a sinistra. Seleziona la casella di controllo accanto a ZeppelinConnection per selezionarla. Scegli Operazione, Elimina connessione. Conferma la selezione.

Eliminazione del ruolo e della policy IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Dalla barra di navigazione sinistra, scegli Ruoli.
3. Usa la barra di ricerca per cercare il ZeppelinRoleruolo.
4. Scegli il ZeppelinRoleruolo. Scegli Elimina ruolo. Conferma l'eliminazione.

Elimina il tuo gruppo di CloudWatch log

La console crea automaticamente un CloudWatch gruppo e un flusso di log quando crei l'applicazione utilizzando la console. Non ottieni un gruppo di log e un flusso di log se l'applicazione viene creata utilizzando la AWS CLI.

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Scegli Gruppi di log dalla barra di navigazione a sinistra.
3. Scegli il gruppo/AWS/KinesisAnalytics/MyNotebooklog.
4. Scegli Operazioni > Elimina gruppo/i di log. Conferma l'eliminazione.

Pulisci le risorse di Kinesis Data Streams

Per eliminare il flusso di dati Kinesis, apri la console del flusso di dati Kinesis, seleziona il flusso Kinesis e scegli Operazioni > Elimina.

Eliminazione delle risorse MSK

Segui i passaggi in questa sezione se hai creato un cluster Amazon MSK per questo tutorial. Questa sezione contiene istruzioni per ripulire l'istanza EC2 del client Amazon, Amazon VPC e il cluster Amazon MSK.

Elimina il tuo cluster Amazon MSK

Segui questi passaggi se hai creato un cluster Amazon MSK per questo tutorial.

1. Aprire la console Amazon MSK a <https://console.aws.amazon.com/msk/casa?region=us-east-1#/home/>.
2. Scegli AWS KafkaTutorialCluster. Scegli Elimina. Inserisci **delete** nella finestra che appare e conferma la selezione.

Interruzione di un'istanza client

Segui questi passaggi se hai creato un'istanza EC2 client Amazon per questo tutorial.

1. Apri la EC2 console Amazon all'indirizzo <https://console.aws.amazon.com/ec2/>.
2. Scegli Istanze nella barra di navigazione a sinistra.
3. Scegli la casella di controllo accanto a ZeppelinClientper selezionarla.
4. Scegli Stato istanza > Termina istanza.

Eliminazione di Amazon VPC

Segui questi passaggi se hai creato un Amazon VPC per questo tutorial.

1. Apri la EC2 console Amazon all'indirizzo <https://console.aws.amazon.com/ec2/>.
2. Scegli Interfacce di rete dalla barra di navigazione a sinistra.
3. Inserisci l'ID VPC nella barra di ricerca e premi Invio per effettuare la ricerca.
4. Seleziona la casella di controllo nell'intestazione della tabella per selezionare tutte le interfacce di rete visualizzate.
5. Seleziona Operazioni > Scollega. Nella finestra che appare, scegli Abilita in Forza distacco. Scegli Scollega e attendi che tutte le interfacce di rete raggiungano lo stato Disponibile.
6. Seleziona la casella di controllo nell'intestazione della tabella per selezionare nuovamente tutte le interfacce di rete visualizzate.
7. Scegli Operazioni > Elimina. Conferma l'operazione.
8. Apri la console Amazon VPC all'indirizzo <https://console.aws.amazon.com/vpc/>.
9. Seleziona AWS KafkaTutorialVPC. Scegli Operazioni > Elimina VPC. Inserisci **delete** e conferma l'eliminazione.

Tutorial: distribuzione di un notebook Studio come servizio gestito per l'applicazione Apache Flink con stato durevole

Il seguente tutorial mostra come implementare un notebook Studio come un'applicazione del servizio gestito per Apache Flink con stato durevole.

Questo tutorial contiene le sezioni seguenti:

- [Completamento dei prerequisiti](#)
- [Implementazione di un'applicazione con stato durevole utilizzando la AWS Management Console](#)
- [Implementazione di un'applicazione con stato durevole utilizzando la AWS CLI](#)

Completamento dei prerequisiti

Crea un nuovo notebook Studio seguendo le istruzioni in [Tutorial: crea un notebook Studio in Managed Service per Apache Flink](#) e utilizzando il flusso di dati Kinesis o Amazon MSK. Assegna un nome al notebook Studio ExampleTestDeploy.

Implementazione di un'applicazione con stato durevole utilizzando la AWS Management Console

1. Aggiungi una posizione del bucket S3 in cui desideri che il codice del pacchetto venga archiviato in Posizione del codice dell'applicazione (opzionale) nella console. Ciò consente di eseguire i passaggi necessari per implementare ed eseguire l'applicazione direttamente dal notebook.
2. Aggiungi le autorizzazioni necessarie al ruolo dell'applicazione per abilitare il ruolo che stai utilizzando per leggere e scrivere su un bucket Amazon S3 e per avviare un'applicazione del servizio gestito per Apache Flink:

- Amazon S3 FullAccess
- Amazon ha gestito- flinkFullAccess
- Accesso alle fonti, alle destinazioni e VPCs , se applicabile. Per ulteriori informazioni, consulta [Verifica le autorizzazioni IAM per i notebook Studio](#).

3. Utilizza il seguente codice di esempio:

```
%flink.sql(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. Con il lancio di questa funzionalità, vedrai un nuovo menu a discesa nell'angolo in alto a destra di ogni nota del notebook con il nome del notebook. Puoi eseguire le operazioni indicate di seguito:
- Visualizza le impostazioni del notebook Studio nella AWS Management Console.
 - Crea una nota Zeppelin ed esportala in Amazon S3. A questo punto, fornisci un nome per l'applicazione e scegli Crea ed esporta. Riceverai una notifica al termine dell'esportazione.
 - Se necessario, puoi visualizzare ed eseguire eventuali test aggiuntivi sull'eseguibile in Amazon S3.

- Una volta completata la compilazione, potrai implementare il codice come un'applicazione di streaming Kinesis con stato durevole e scalabilità automatica.
- Dal menu a discesa, scegli Implementa nota Zeppelin come applicazione di streaming Kinesis. Controlla il nome dell'applicazione e scegli Distribuisci tramite AWS console.
- Questo ti porterà alla AWS Management Console pagina per la creazione di un servizio gestito per l'applicazione Apache Flink. Tieni presente che il nome dell'applicazione, il parallelismo, la posizione del codice, i ruoli predefiniti di Glue DB, VPC (se applicabile) e IAM sono stati precompilati. Verifica che i ruoli IAM dispongano delle autorizzazioni necessarie per le origini e le destinazioni. Gli snapshot sono abilitati per impostazione predefinita per la gestione dello stato durevole delle applicazioni.
- Scegli Crea applicazione.
- Puoi scegliere Configura e modificare qualsiasi impostazione, quindi scegli Esegui per avviare l'applicazione di streaming.

Implementazione di un'applicazione con stato durevole utilizzando la AWS CLI

Per distribuire un'applicazione utilizzando il AWS CLI, è necessario aggiornare il modello di servizio fornito con le informazioni relative AWS CLI alla Beta 2. Per informazioni sull'utilizzo del modello di servizio aggiornato, consulta [Completa i prerequisiti](#).

Il codice di esempio seguente crea un nuovo notebook Studio:

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

```

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

```

Il seguente esempio di codice avvia un notebook Studio:

```

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

```

Il codice seguente restituisce l'URL della pagina del notebook Apache Zeppelin di un'applicazione:

```

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

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

```

Visualizza esempi di query per analizzare i dati in un notebook Studio

Le seguenti query di esempio mostrano come analizzare i dati utilizzando le query a finestra in un notebook di Studio.

- [Crea tabelle con Amazon MSK/Apache Kafka](#)
- [Crea tabelle con Kinesis](#)
- [Interroga una finestra che si apre](#)
- [Interroga una finestra scorrevole](#)
- [Usa SQL interattivo](#)
- [Usa il connettore BlackHole SQL](#)
- [Usa Scala per generare dati di esempio](#)
- [Usa Scala interattiva](#)
- [Usa Python interattivo](#)
- [Usa una combinazione di Python, SQL e Scala interattivi](#)
- [Usa un flusso di dati Kinesis tra account](#)

Per informazioni sulle impostazioni delle query SQL di Apache Flink, consulta [Flink sui notebook Zeppelin per l'analisi dei dati interattiva](#).

Per visualizzare la tua applicazione nel pannello di controllo di Apache Flink, scegli PROCESSO FLINK nella pagina Nota Zeppelin della tua applicazione.

Per ulteriori informazioni sulle query a finestra, consulta [Finestre](#) nella [documentazione di Apache Flink](#).

Per altri esempi di query Streaming SQL di Apache Flink, consulta [Query](#) nella [documentazione di Apache Flink](#).

Crea tabelle con Amazon MSK/Apache Kafka

Puoi utilizzare il connettore Amazon MSK Flink con il servizio gestito per Apache Flink Studio per autenticare la connessione con l'autenticazione Plaintext, SSL o IAM. Crea tabelle utilizzando proprietà specifiche in base alle tue esigenze.

```
-- 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'  
);
```

Puoi combinarle con altre proprietà in [Connettore SQL Apache Kafka](#).

Crea tabelle con Kinesis

Nell'esempio seguente, viene creata una tabella usando 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'  
);
```

Per ulteriori informazioni sulle altre proprietà utilizzabili, consulta [Connettore SQL per il flusso di dati Amazon Kinesis](#).

Interroga una finestra che si apre

La seguente query Streaming SQL di Flink seleziona dalla tabella `ZeppelinTopic` il prezzo più alto in ogni finestra a cascata di cinque secondi:

```
%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)
```

Interroga una finestra scorrevole

La seguente query Streaming SQL di Apache Flink seleziona il prezzo più alto in ogni finestra scorrevole di cinque secondi dalla tabella ZeppelinTopic:

```
%flink.sql(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)
```

Usa SQL interattivo

Questo esempio converte il massimo dell'ora evento e del tempo di elaborazione e la somma dei valori della tabella chiave-valore. Assicurati di avere lo script di generazione dei dati di esempio dal [the section called “Usa Scala per generare dati di esempio”](#) in esecuzione. Per provare altre query SQL, ad esempio il filtraggio e i join, nel notebook Studio, consulta [Queries](#) nella documentazione di Apache Flink.

```
%flink.sql(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.sql(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`;
```

Usa il connettore BlackHole SQL

Il connettore BlackHole SQL non richiede la creazione di un flusso di dati Kinesis o di un cluster Amazon MSK per testare le query. Per informazioni sul connettore BlackHole SQL, consulta [BlackHole SQL Connector nella documentazione di Apache Flink](#). In questo esempio, il catalogo predefinito è un catalogo in memoria.

```
%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

```

Usa Scala per generare dati di esempio

Questo esempio utilizza Scala per generare dati di esempio. È possibile utilizzare questi dati di esempio per testare varie query. Utilizza l'istruzione di creazione tabella per creare la tabella chiave-valore.

```

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`
```

Usa Scala interattiva

Questa è la traduzione in Scala di [the section called “Usa SQL interattivo”](#). Per altri esempi di Scala, consulta [API Table](#) nella documentazione di Apache Flink.

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

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

```
%flink(parallelism=4)

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

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

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

```
%flink(parallelism=4)
```

```
// An tumbling window view that displays the number of records observed per (event  
time) second.
```

```
val query02 = stenv  
  .from("`key-values`")  
  .window(Tumble over 1.seconds on $"et" as $"w")  
  .groupBy($"w", $"key")  
  .select(  
    $"w".start.as("window"),  
    $"key",  
    $"value".sum().as("sum")  
  ).asView("query02")
```

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

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

Usa Python interattivo

Questa è la traduzione in Python di [the section called “Usa SQL interattivo”](#). Per altri esempi di Python, consulta [API Table](#) nella documentazione di Apache Flink.

```
%flink.pyflink  
from pyflink.table.table import Table  
  
def as_view(table, name):  
    if (name in st_env.list_temporary_views()):  
        st_env.drop_temporary_view(name)  
    st_env.create_temporary_view(name, table)  
    return table  
  
Table.as_view = as_view
```

```
%flink.pyflink(parallelism=16)
```

```
# A view that computes many records from the `key-values` we have seen so far, along
with the current processing and event time
st_env \
  .from_path("`keyvalues`") \
  .select(", ".join([
    "max(et) as et",
    "max(pt) as pt",
    "sum(value) as sum"
  ])) \
  .as_view("query01")
```

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

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

```
%flink.pyflink(parallelism=16)

# A view that computes many records from the `key-values` we have seen so far, along
with the current processing and event time
st_env \
  .from_path("`key-values`") \
  .window(Tumble.over("1.seconds").on("et").alias("w")) \
  .group_by("w, key") \
  .select(", ".join([
    "w.start as window",
    "key",
    "sum(value) as sum"
  ])) \
  .as_view("query02")
```

```
%flink.ssql(type=update, parallelism=16, refreshInterval=1000)

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

Usa una combinazione di Python, SQL e Scala interattivi

Puoi usare qualsiasi combinazione di SQL, Python e Scala nel tuo notebook per l'analisi interattiva. In un notebook Studio che intendi implementare come applicazione con stato durevole, puoi utilizzare una combinazione di SQL e Scala. Questo esempio mostra le sezioni che vengono ignorate e quelle che vengono implementate nell'applicazione con stato durevole.

```
%flink.sql
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.sql
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!")
```

Usa un flusso di dati Kinesis tra account

Per utilizzare un flusso di dati Kinesis che si trova in un account diverso da quello su cui è installato il notebook Studio, crea un ruolo di esecuzione del servizio nell'account su cui è in esecuzione il notebook Studio e una politica di attendibilità dei ruoli nell'account su cui è in esecuzione

il flusso di dati. Utilizza `aws.credentials.provider`, `aws.credentials.role.arn` e `aws.credentials.role.sessionName` nel connettore Kinesis nell'istruzione DDL di creazione tabella per creare una tabella in base al flusso di dati.

Utilizza il seguente ruolo di esecuzione del servizio per l'account notebook Studio.

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

Utilizza la policy `AmazonKinesisFullAccess` e la seguente policy di attendibilità dei ruoli per l'account del flusso di dati.

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

Usa il paragrafo seguente per l'istruzione di creazione tabella.

```
%flink.sql
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'  
)
```

Risolvi i problemi relativi ai notebook Studio per Managed Service for Apache Flink

Questa sezione contiene informazioni sulla risoluzione dei problemi per i notebook Studio.

Arresta un'applicazione bloccata

Per arrestare un'applicazione bloccata in uno stato transitorio, richiama l'[StopApplication](#) azione con il Force parametro impostato su. true Per ulteriori informazioni, consulta [Applicazioni in esecuzione](#) nella [Guida per gli sviluppatori del servizio gestito per Apache Flink](#).

Implementa come applicazione con uno stato durevole in un VPC senza accesso a Internet

La deploy-as-application funzione Managed Service for Apache Flink Studio non supporta applicazioni VPC senza accesso a Internet. Si consiglia di creare l'applicazione in Studio e quindi di utilizzare il servizio gestito per Apache Flink per creare manualmente un'applicazione Flink e selezionare il file zip che hai creato nel tuo notebook.

I passaggi seguenti descrivono come farlo:

1. Compila ed esporta l'applicazione Studio in Amazon S3. Dovrebbe trattarsi di un file zip.
2. Crea manualmente un'applicazione del servizio gestito per Apache Flink con il percorso del codice che fa riferimento alla posizione del file zip in Amazon S3. Inoltre, dovrai configurare l'applicazione con le seguenti variabili env (2 groupID, 3 var in totale):
 - a. python: source/note.py
 - b. file jar: lib/ .jar PythonApplicationDependencies
3. kinesis.analytics.flink.run.options
4. managed.deploy_as_app.options

- DatabasEarn: *<glue database ARN (Amazon Resource Name)>*
5. Potrebbe essere necessario concedere ai ruoli IAM del servizio gestito per Apache Flink Studio e del servizio gestito per Apache Flink le autorizzazioni per i servizi utilizzati dall'applicazione. È possibile utilizzare lo stesso ruolo IAM per entrambe le applicazioni.

Deploy-as-app riduzione delle dimensioni e dei tempi di costruzione

Le applicazioni Studio deploy-as-app for Python racchiudono tutto ciò che è disponibile nell'ambiente Python perché non possiamo determinare di quali librerie hai bisogno. Ciò può comportare una dimensione maggiore del necessario. deploy-as-app La procedura seguente mostra come ridurre le dimensioni dell'applicazione deploy-as-app Python disinstallando le dipendenze.

Se stai creando un'applicazione Python con deploy-as-app funzionalità di Studio, potresti prendere in considerazione la possibilità di rimuovere i pacchetti Python preinstallati dal sistema se le tue applicazioni non dipendono da. Ciò non solo contribuirà a ridurre le dimensioni finali degli artefatti per evitare di violare il limite di servizio relativo alle dimensioni delle applicazioni, ma migliorerà anche i tempi di creazione delle applicazioni dotate di questa funzionalità. deploy-as-app

È possibile eseguire il seguente comando per elencare tutti i pacchetti Python installati con le rispettive dimensioni installate e rimuovere selettivamente i pacchetti con dimensioni significative.

```
%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 è richiesto per il funzionamento di Flink Python. Questo pacchetto e le sue dipendenze non devono mai essere rimossi.

Di seguito è riportato l'elenco dei pacchetti Python preinstallati in Studio V2 che possono essere presi in considerazione per la rimozione:

```
scipy  
statsmodels
```



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

Per rimuovere un pacchetto Python dal notebook Zeppelin:

1. Controlla se la tua applicazione dipende dal pacchetto, o da uno dei pacchetti che la utilizzano, prima di rimuoverla. È possibile identificare i dipendenti di un pacchetto utilizzando [pipdeptree](#).
2. Esegui il seguente comando per rimuovere un pacchetto:

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

3. Se hai bisogno di recuperare un pacchetto rimosso per errore, esegui il seguente comando:

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

Example Esempio: rimuovi il **scipy** pacchetto prima di distribuire l'applicazione deploy-as-app Python con funzionalità.

1. Utilizza pipdeptree per scoprire tutti i consumatori scipy e verificare se puoi rimuovere scipy in sicurezza.

- Installa lo strumento tramite notebook:

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

- Ottieni l'albero delle dipendenze invertito di scipy eseguendo:

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

Verrà visualizzato un output simile al seguente (condensato per brevità):

```
...
-----
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. Ispeziona attentamente l'utilizzo di seaborn, statsmodels e plotnine nelle tue applicazioni. Se le tue applicazioni non dipendono da scipy, seaborn, statemodels o da plotnine, puoi rimuovere tutti questi pacchetti o solo quelli che non sono necessari alle tue applicazioni.
3. Rimuovi il pacchetto eseguendo:

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

Annullare i lavori

Questa sezione mostra come annullare i processi di Apache Flink a cui non puoi accedere da Apache Zeppelin. Se desideri annullare un processo di questo tipo, vai al pannello di controllo di Apache Flink, copia l'ID del processo, quindi utilizzalo in uno dei seguenti esempi.

Per annullare un singolo processo:

```
%flink.pyflink
import requests

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

Per annullare tutti i processi in esecuzione:

```
%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))
```

Per annullare tutti i processi:

```
%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)
```

Riavvia l'interprete Apache Flink

Per riavviare l'interprete Apache Flink sul notebook Studio

1. Scegli Configurazione accanto all'angolo in alto a destra della schermata.
2. Scegli Interprete.
3. Scegli riavvia e poi OK.

Crea policy IAM personalizzate per i notebook Managed Service for Apache Flink Studio

Normalmente si utilizzano policy IAM gestite per consentire all'applicazione di accedere a risorse dipendenti. Se hai bisogno di un controllo più preciso sulle autorizzazioni dell'applicazione, puoi utilizzare una policy IAM personalizzata. Questa sezione contiene esempi di politiche IAM personalizzate.

Note

Nei seguenti esempi di policy, sostituisci il testo segnaposto con i valori dell'applicazione.

Questo argomento contiene le sezioni seguenti:

- [AWS Glue](#)
- [CloudWatch Registri](#)
- [Flussi Kinesis](#)
- [Cluster Amazon MSK](#)

AWS Glue

La seguente politica di esempio concede le autorizzazioni per accedere a un database. AWS Glue

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "GlueTable",
      "Effect": "Allow",
      "Action": [
        "glue:GetConnection",
        "glue:GetTable",
        "glue:GetTables",
        "glue:GetDatabase",
        "glue:CreateTable",
        "glue:UpdateTable"
      ],
      "Resource": [
        "arn:aws:glue:<region>:<accountId>:connection/*",
        "arn:aws:glue:<region>:<accountId>:table/<database-name>/*",
        "arn:aws:glue:<region>:<accountId>:database/<database-name>",
        "arn:aws:glue:<region>:<accountId>:database/hive",
        "arn:aws:glue:<region>:<accountId>:catalog"
      ]
    },
    {
      "Sid": "GlueDatabase",
      "Effect": "Allow",
      "Action": "glue:GetDatabases",
      "Resource": "*"
    }
  ]
}
```

CloudWatch Registri

La seguente politica concede le autorizzazioni per accedere ai registri: CloudWatch

```
{
  "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

Se si crea l'applicazione utilizzando la console, la console aggiunge le politiche necessarie per accedere a CloudWatch Logs al ruolo dell'applicazione.

Flussi Kinesis

L'applicazione può utilizzare un flusso Kinesis come origine o destinazione. L'applicazione necessita delle autorizzazioni di lettura per leggere da un flusso di origine e delle autorizzazioni di scrittura per scrivere su un flusso di destinazione.

La seguente policy concede le autorizzazioni per la lettura da un flusso Kinesis utilizzato come origine:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "KinesisShardDiscovery",
      "Effect": "Allow",
      "Action": "kinesis:ListShards",
      "Resource": "*"
    },
    {
      "Sid": "KinesisShardConsumption",
      "Effect": "Allow",
      "Action": [
        "kinesis:GetShardIterator",
        "kinesis:GetRecords",
        "kinesis:DescribeStream",
        "kinesis:DescribeStreamSummary",
        "kinesis:RegisterStreamConsumer",
        "kinesis:DeregisterStreamConsumer"
      ],
      "Resource": "arn:aws:kinesis:<region>:<accountId>:stream/<stream-name>"
    },
    {
      "Sid": "KinesisEfoConsumer",
      "Effect": "Allow",
      "Action": [
        "kinesis:DescribeStreamConsumer",
        "kinesis:SubscribeToShard"
      ],
      "Resource": "arn:aws:kinesis:<region>:<account>:stream/<stream-name>/consumer/*"
    }
  ]
}
```

La seguente politica concede le autorizzazioni di scrittura su un flusso Kinesis utilizzato come destinazione:

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

Se l'applicazione accede a un flusso Kinesis crittografato, è necessario concedere autorizzazioni aggiuntive per accedere al flusso e alla chiave di crittografia del flusso.

La seguente policy concede le autorizzazioni per accedere a un flusso di origine crittografato e alla chiave di crittografia del flusso:

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

La seguente policy concede le autorizzazioni per accedere a un flusso di destinazione crittografato e alla chiave di crittografia del flusso:

```
{
```

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

Cluster Amazon MSK

Per concedere l'accesso a un cluster Amazon MSK, concedi l'accesso al VPC del cluster. Per esempi di policy per l'accesso a un Amazon VPC, consulta [Autorizzazioni delle applicazioni VPC](#).

Inizia a usare Amazon Managed Service per Apache Flink (DataStream API)

Questa sezione presenta i concetti fondamentali di Managed Service for Apache Flink e l'implementazione di un'applicazione in Java utilizzando l'API. DataStream Descrive le opzioni disponibili per la creazione e il test delle applicazioni. Fornisce inoltre istruzioni per l'installazione degli strumenti necessari per completare i tutorial di questa guida e creare la tua prima applicazione.

Argomenti

- [Esamina i componenti dell'applicazione Managed Service for Apache Flink](#)
- [Soddisfa i prerequisiti per completare gli esercizi](#)
- [Configura un AWS account e crea un utente amministratore](#)
- [Configura il \(\) AWS Command Line InterfaceAWS CLI](#)
- [Crea ed esegui un servizio gestito per l'applicazione Apache Flink](#)
- [Pulisci le risorse AWS](#)
- [Esplora risorse aggiuntive](#)

Esamina i componenti dell'applicazione Managed Service for Apache Flink

Note

Amazon Managed Service per Apache Flink supporta tutti i linguaggi Apache Flink APIs e potenzialmente tutti i linguaggi JVM. [Per ulteriori informazioni, consulta Flink's. APIs](#)
A seconda dell'API scelta, la struttura dell'applicazione e l'implementazione sono leggermente diverse. Questo tutorial introduttivo illustra l'implementazione delle applicazioni che utilizzano l' DataStream API in Java.

Per elaborare i dati, l'applicazione Managed Service for Apache Flink utilizza un'applicazione Java che elabora l'input e produce l'output utilizzando il runtime Apache Flink.

Una tipica applicazione Managed Service for Apache Flink ha i seguenti componenti:

- **Proprietà di runtime:** è possibile utilizzare le proprietà di runtime per passare i parametri di configurazione all'applicazione e modificarli senza modificare e ripubblicare il codice.
- **Fonti:** l'applicazione utilizza dati da una o più fonti. Una fonte utilizza un [connettore](#) per leggere i dati da un sistema esterno, come un flusso di dati Kinesis o un bucket Kafka. Per ulteriori informazioni, consulta [Aggiungi sorgenti di dati in streaming](#).
- **Operatori:** l'applicazione elabora i dati utilizzando uno o più operatori. Un operatore può trasformare, arricchire o aggregare i dati. Per ulteriori informazioni, consulta [Operatori](#).
- **Sinks:** l'applicazione invia i dati a fonti esterne tramite i sink. Un sink utilizza un [connettore](#) v per inviare dati a un flusso di dati Kinesis, un argomento di Kafka, Amazon S3 o un database relazionale. Puoi anche usare un connettore speciale per stampare l'output solo a scopo di sviluppo. Per ulteriori informazioni, consulta [Scrivi dati utilizzando i lavandini](#).

L'applicazione richiede alcune dipendenze esterne, come i connettori Flink utilizzati dall'applicazione o potenzialmente una libreria Java. Per essere eseguita in Amazon Managed Service for Apache Flink, l'applicazione deve essere impacchettata insieme alle dipendenze in un fat-jar e caricata in un bucket Amazon S3. Potrai quindi creare un'applicazione del servizio gestito per Apache Flink. Devi passare la posizione del pacchetto di codice, insieme a qualsiasi altro parametro di configurazione di runtime.

Questo tutorial dimostra come utilizzare Apache Maven per creare un pacchetto dell'applicazione e come eseguire l'applicazione localmente nell'IDE di tua scelta.

Soddisfa i prerequisiti per completare gli esercizi

Per completare le fasi in questa guida, è richiesto quanto segue:

- [Client Git](#). Installa il client Git, se non l'hai già fatto.
- [Java Development Kit \(JDK\) versione 11](#). Installa un Java JDK 11 e imposta la variabile di ambiente JAVA_HOME in modo che punti alla posizione di installazione di JDK. Se non disponi di un JDK 11, puoi utilizzare [Amazon Corretto 11](#) o qualsiasi altro JDK standard a tua scelta.
- Per verificare che il JDK sia installato correttamente, esegui il comando seguente. L'output sarà diverso se utilizzi un JDK diverso da Amazon Corretto. Assicurati che la versione sia 11.x.

```
$ java --version
```

```
openjdk 11.0.23 2024-04-16 LTS
```

```
OpenJDK Runtime Environment Corretto-11.0.23.9.1 (build 11.0.23+9-LTS)
OpenJDK 64-Bit Server VM Corretto-11.0.23.9.1 (build 11.0.23+9-LTS, mixed mode)
```

- [Apache Maven](#). Installa Apache Maven se non l'hai già fatto. Per informazioni su come installarlo, consulta [Installazione](#) di Apache Maven.
- Per testare l'installazione Apache Maven, immetti quanto segue:

```
$ mvn -version
```

- IDE per lo sviluppo locale. Ti consigliamo di utilizzare un ambiente di sviluppo come [Eclipse Java Neon](#) o [IntelliJ IDEA](#) per sviluppare e compilare l'applicazione.
- Per testare l'installazione Apache Maven, immetti quanto segue:

```
$ mvn -version
```

Per iniziare, vai alla pagina [Configura un AWS account e crea un utente amministratore](#).

Configura un AWS account e crea un utente amministratore

Prima di utilizzare il servizio gestito per Apache Flink per la prima volta, completa le seguenti attività:

Registrati per un Account AWS

Se non ne hai uno Account AWS, completa i seguenti passaggi per crearne uno.

Per iscriverti a un Account AWS

1. Apri la <https://portal.aws.amazon.com/billing/registrazione>.
2. Segui le istruzioni online.

Nel corso della procedura di registrazione riceverai una telefonata, durante la quale sarà necessario inserire un codice di verifica attraverso la tastiera del telefono.

Quando ti iscrivi a un Account AWS, Utente root dell'account AWS viene creato un. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best practice di sicurezza, assegna l'accesso amministrativo a un utente e utilizza solo l'utente root per eseguire [attività che richiedono l'accesso di un utente root](#).

AWS ti invia un'email di conferma dopo il completamento della procedura di registrazione. In qualsiasi momento, puoi visualizzare l'attività corrente del tuo account e gestirlo accedendo a <https://aws.amazon.com/> e scegliendo Il mio account.

Crea un utente con accesso amministrativo

Dopo esserti registrato Account AWS, proteggi Utente root dell'account AWS AWS IAM Identity Center, abilita e crea un utente amministrativo in modo da non utilizzare l'utente root per le attività quotidiane.

Proteggi i tuoi Utente root dell'account AWS

1. Accedi [AWS Management Console](#) come proprietario dell'account scegliendo Utente root e inserendo il tuo indirizzo Account AWS email. Nella pagina successiva, inserisci la password.

Per informazioni sull'accesso utilizzando un utente root, consulta la pagina [Signing in as the root user](#) della Guida per l'utente di Accedi ad AWS .

2. Abilita l'autenticazione a più fattori (MFA) per l'utente root.

Per istruzioni, consulta [Abilitare un dispositivo MFA virtuale per l'utente Account AWS root \(console\)](#) nella Guida per l'utente IAM.

Crea un utente con accesso amministrativo

1. Abilita Centro identità IAM.

Per istruzioni, consulta [Abilitazione di AWS IAM Identity Center](#) nella Guida per l'utente di AWS IAM Identity Center .

2. In IAM Identity Center, assegna l'accesso amministrativo a un utente.

Per un tutorial sull'utilizzo di IAM Identity Center directory come fonte di identità, consulta [Configurare l'accesso utente con l'impostazione predefinita IAM Identity Center directory](#) nella Guida per l'AWS IAM Identity Center utente.

Accesso come utente amministratore

- Per accedere con l'utente IAM Identity Center, utilizza l'URL di accesso che è stato inviato al tuo indirizzo e-mail quando hai creato l'utente IAM Identity Center.

Per informazioni sull'accesso utilizzando un utente IAM Identity Center, consulta [AWS Accedere al portale di accesso](#) nella Guida per l'Accedi ad AWS utente.

Assegna l'accesso a ulteriori utenti

1. In IAM Identity Center, crea un set di autorizzazioni conforme alla best practice dell'applicazione di autorizzazioni con il privilegio minimo.

Segui le istruzioni riportate nella pagina [Creazione di un set di autorizzazioni](#) nella Guida per l'utente di AWS IAM Identity Center .

2. Assegna al gruppo prima gli utenti e poi l'accesso con autenticazione unica (Single Sign-On).

Per istruzioni, consulta [Aggiungere gruppi](#) nella Guida per l'utente di AWS IAM Identity Center .

Concessione dell'accesso programmatico

Gli utenti hanno bisogno di un accesso programmatico se vogliono interagire con l' AWS AWS Management Console esterno di. Il modo per concedere l'accesso programmatico dipende dal tipo di utente che accede. AWS

Per fornire agli utenti l'accesso programmatico, scegli una delle seguenti opzioni.

Quale utente necessita dell'accesso programmatico?	Per	Come
Identità della forza lavoro (Utenti gestiti nel centro identità IAM)	Utilizza credenziali temporane e per firmare le richieste programmatiche a AWS CLI,, AWS SDKs o. AWS APIs	Segui le istruzioni per l'interfaccia che desideri utilizzare. • Per la AWS CLI, vedere Configurazione dell'uso AWS IAM Identity Center nella AWS CLI Guida per l'utente.AWS Command Line Interface • Per AWS SDKs gli strumenti e AWS APIs, consulta l'autenticazione di IAM

Quale utente necessita dell'accesso programmatico?	Per	Come
		Identity Center nella Guida di riferimento AWS SDKs and Tools.
IAM	Utilizza credenziali temporanee e per firmare le richieste programmatiche a AWS CLI, AWS SDKs, o. AWS APIs	Seguendo le istruzioni riportate in Utilizzo delle credenziali temporanee con le AWS risorse nella Guida per l'utente IAM .
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare richieste programmatiche a AWS CLI,, AWS SDKs o. AWS APIs	Segui le istruzioni per l'interfaccia che desideri utilizzare. • Per la AWS CLI, consulta Autenticazione tramite credenziali utente IAM nella Guida per l'utente.AWS Command Line Interface • Per gli strumenti AWS SDKs e gli strumenti, consulta Autenticazione tramite credenziali a lungo termine nella Guida di riferimento agli strumenti e agli AWS SDKs strumenti. • Per AWS APIs, consulta la sezione Gestione delle chiavi di accesso per gli utenti IAM nella Guida per l'utente IAM.

Fase successiva

[Configura il \(\) AWS Command Line InterfaceAWS CLI](#)

Configura il () AWS Command Line InterfaceAWS CLI

In questo passaggio, si scarica e si configura AWS CLI per l'utilizzo con Managed Service for Apache Flink.

Note

Gli esercizi sulle Nozioni di base di questa guida presuppongono che tu disponga di credenziali di amministratore (`adminuser`) nell'account per eseguire le operazioni.

Note

Se è già AWS CLI installato, potrebbe essere necessario eseguire l'aggiornamento per ottenere le funzionalità più recenti. Per ulteriori informazioni, consulta [Installazione dell' AWS Command Line Interface](#) nella Guida per l'utente dell'AWS Command Line Interface . Per verificare la versione di AWS CLI, esegui il seguente comando:

```
aws --version
```

Gli esercizi di questo tutorial richiedono la seguente AWS CLI versione o successiva:

```
aws-cli/1.16.63
```

Per configurare il AWS CLI

1. Scarica e configura la AWS CLI. Per le istruzioni, consulta i seguenti argomenti nella Guida per l'utente dell'AWS Command Line Interface :
 - [Installazione dell' AWS Command Line Interface](#)
 - [Configurazione della AWS CLI](#)
2. Aggiungere un profilo denominato per l'utente amministratore nel AWS CLI config file. Puoi usare questo profilo quando esegui i comandi della AWS CLI . Per ulteriori informazioni sui profili denominati, consulta [Profili denominati](#) in Guida per l'utente dell'AWS Command Line Interface .

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

```
aws_secret_access_key = adminuser secret access key  
region = aws-region
```

Per un elenco delle AWS regioni disponibili, consulta [Regioni ed endpoint](#) in. Riferimenti generali di Amazon Web Services

Note

Il codice e i comandi di esempio in questo tutorial utilizzano la regione us-east-1 Stati Uniti orientali (Virginia settentrionale). Per utilizzare una regione diversa, sostituisci la regione nel codice e nei comandi di questo tutorial con quella che desideri utilizzare.

3. Verifica la configurazione digitando il comando help riportato di seguito al prompt dei comandi:

```
aws help
```

Dopo aver configurato un AWS account AWS CLI, potete provare l'esercizio successivo, in cui configurate un'applicazione di esempio e verificate la configurazione. end-to-end

Approfondimenti

[Crea ed esegui un servizio gestito per l'applicazione Apache Flink](#)

Crea ed esegui un servizio gestito per l'applicazione Apache Flink

In questo passaggio, crei un servizio gestito per l'applicazione Apache Flink con flussi di dati Kinesis come origine e sink.

Questa sezione contiene le fasi seguenti:

- [Crea risorse dipendenti](#)
- [Configurazione dell'ambiente di sviluppo locale](#)
- [Scarica ed esamina il codice Java per lo streaming di Apache Flink](#)
- [Scrivi record di esempio nel flusso di input](#)
- [Esegui l'applicazione localmente](#)
- [Osserva i dati di input e output nei flussi Kinesis](#)
- [Interrompi l'esecuzione locale dell'applicazione](#)

- [Compila e impacchetta il codice dell'applicazione](#)
- [Carica il file JAR del codice dell'applicazione](#)
- [Crea e configura l'applicazione Managed Service for Apache Flink](#)
- [Approfondimenti](#)

Crea risorse dipendenti

Prima di creare un'applicazione del servizio gestito per Apache Flink per questo esercizio, è necessario creare le seguenti risorse dipendenti:

- Due flussi di dati Kinesis per input e output
- Un bucket Amazon S3 per archiviare il codice dell'applicazione

Note

Questo tutorial presuppone che l'applicazione venga distribuita nella regione us-east-1 Stati Uniti orientali (Virginia settentrionale). Se utilizzi un'altra regione, adatta tutti i passaggi di conseguenza.

Crea due flussi di dati Amazon Kinesis

Prima di creare un'applicazione del servizio gestito per Apache Flink per questo esercizio, crea due flussi di dati Kinesis (`ExampleInputStream` e `ExampleOutputStream`). L'applicazione utilizza questi flussi per i flussi di origine e di destinazione dell'applicazione.

Puoi creare questi stream utilizzando la console Amazon Kinesis o il AWS CLI seguente comando. Per istruzioni sulla console, consulta [Creazione e aggiornamento dei flussi di dati](#) nella Guida per gli sviluppatori del flusso di dati Amazon Kinesis. Per creare gli stream utilizzando AWS CLI, usa i seguenti comandi, adattandoli alla regione utilizzata per la tua applicazione.

Per creare i flussi di dati (AWS CLI)

1. Per creare il primo stream (`ExampleInputStream`), usa il seguente comando Amazon Kinesis `create-stream` AWS CLI :

```
$ aws kinesis create-stream \
```

```
--stream-name ExampleInputStream \  
--shard-count 1 \  
--region us-east-1 \  

```

2. Per creare il secondo stream utilizzato dall'applicazione per scrivere l'output, esegui lo stesso comando, modificando il nome dello stream in `ExampleOutputStream`:

```
$ aws kinesis create-stream \  
--stream-name ExampleOutputStream \  
--shard-count 1 \  
--region us-east-1 \  

```

Crea un bucket Amazon S3 per il codice dell'applicazione

Puoi creare un bucket Amazon S3 utilizzando la relativa console. [Per informazioni su come creare un bucket Amazon S3 utilizzando la console, consulta *Creating a bucket nella Amazon S3 User Guide*](#). Assegna un nome al bucket Amazon S3 utilizzando un nome univoco globale, ad esempio aggiungendo il tuo nome di accesso.

Note

Assicurati di creare il bucket nella regione che usi per questo tutorial (us-east-1).

Altre risorse

Quando crei la tua applicazione, Managed Service for Apache Flink crea automaticamente CloudWatch le seguenti risorse Amazon se non esistono già:

- Un gruppo di log denominato `/AWS/KinesisAnalytics-java/<my-application>`
- Un flusso di log denominato `kinesis-analytics-log-stream`

Configurazione dell'ambiente di sviluppo locale

Per lo sviluppo e il debug, puoi eseguire l'applicazione Apache Flink sulla tua macchina direttamente dal tuo IDE preferito. Tutte le dipendenze di Apache Flink vengono gestite come normali dipendenze Java utilizzando Apache Maven.

Note

Sulla tua macchina di sviluppo, devi avere Java JDK 11, Maven e Git installati. [Ti consigliamo di utilizzare un ambiente di sviluppo come Eclipse Java Neon o IntelliJ IDEA](#). Per verificare di soddisfare tutti i prerequisiti, vedere. [Soddisfa i prerequisiti per completare gli esercizi](#) Non è necessario installare un cluster Apache Flink sul computer.

Autentica la tua sessione AWS

L'applicazione utilizza i flussi di dati Kinesis per pubblicare i dati. Quando si esegue localmente, è necessario disporre di una sessione AWS autenticata valida con autorizzazioni di scrittura nel flusso di dati Kinesis. Usa i seguenti passaggi per autenticare la tua sessione:

1. Se non hai configurato il profilo AWS CLI e un profilo denominato con credenziali valide, consulta. [Configura il \(\) AWS Command Line InterfaceAWS CLI](#)
2. Verifica che AWS CLI sia configurato correttamente e che gli utenti dispongano delle autorizzazioni per scrivere nel flusso di dati Kinesis pubblicando il seguente record di test:

```
$ aws kinesis put-record --stream-name ExampleOutputStream --data TEST --partition-key TEST
```

3. Se il tuo IDE ha un plugin con cui integrarti AWS, puoi utilizzarlo per passare le credenziali all'applicazione in esecuzione nell'IDE. Per ulteriori informazioni, vedere [AWS Toolkit for IntelliJ IDEA](#) [AWS e Toolkit for Eclipse](#).

Scarica ed esamina il codice Java per lo streaming di Apache Flink

Il codice dell'applicazione Java per questo esempio è disponibile da. GitHub Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito:

1. Clona il repository remoto con il comando seguente:

```
git clone https://github.com/aws-samples/amazon-managed-service-for-apache-flink-examples.git
```

2. Passa alla directory `amazon-managed-service-for-apache-flink-examples/tree/main/java/GettingStarted`.

Esamina i componenti dell'applicazione

L'applicazione è interamente implementata nella `com.amazonaws.services.msf.BasicStreamingJob` classe. Il `main()` metodo definisce il flusso di dati per elaborare i dati di streaming ed eseguirli.

Note

Per un'esperienza di sviluppo ottimizzata, l'applicazione è progettata per funzionare senza modifiche al codice sia su Amazon Managed Service per Apache Flink che localmente, per lo sviluppo nel tuo IDE.

- Per leggere la configurazione di runtime in modo che funzioni durante l'esecuzione in Amazon Managed Service for Apache Flink e nel tuo IDE, l'applicazione rileva automaticamente se è in esecuzione autonoma localmente nell'IDE. In tal caso, l'applicazione carica la configurazione di runtime in modo diverso:
 1. Quando l'applicazione rileva che è in esecuzione in modalità autonoma nell'IDE, forma il `application_properties.json` file incluso nella cartella delle risorse del progetto. Segue il contenuto del file.
 2. Quando l'applicazione viene eseguita in Amazon Managed Service for Apache Flink, il comportamento predefinito carica la configurazione dell'applicazione dalle proprietà di runtime che definirai nell'applicazione Amazon Managed Service for Apache Flink. Per informazioni, consulta [Crea e configura l'applicazione Managed Service for Apache Flink](#).

```
private static Map<String, Properties>
loadApplicationProperties(StreamExecutionEnvironment env) throws IOException {
    if (env instanceof LocalStreamEnvironment) {
        LOGGER.info("Loading application properties from '{}'",
LOCAL_APPLICATION_PROPERTIES_RESOURCE);
        return KinesisAnalyticsRuntime.getApplicationProperties(
            BasicStreamingJob.class.getClassLoader()

.getResource(LOCAL_APPLICATION_PROPERTIES_RESOURCE).getPath());
    } else {
        LOGGER.info("Loading application properties from Amazon Managed Service for
Apache Flink");
        return KinesisAnalyticsRuntime.getApplicationProperties();
    }
}
```

```
}
```

- Il `main()` metodo definisce il flusso di dati dell'applicazione e lo esegue.
- Inizializza gli ambienti di streaming predefiniti. In questo esempio, mostriamo come creare sia l'API `StreamExecutionEnvironment` da utilizzare con l' `DataStream` API sia `StreamTableEnvironment` quello da utilizzare con SQL e l'API `Table`. I due oggetti di ambiente sono due riferimenti separati allo stesso ambiente di runtime, da utilizzare in modo diverso APIs.

```
StreamExecutionEnvironment env =  
    StreamExecutionEnvironment.getExecutionEnvironment();
```

- Caricate i parametri di configurazione dell'applicazione. Questo li caricherà automaticamente dalla posizione corretta, a seconda di dove è in esecuzione l'applicazione:

```
Map<String, Properties> applicationParameters = loadApplicationProperties(env);
```

- L'applicazione definisce una fonte utilizzando il connettore [Kinesis Consumer](#) per leggere i dati dal flusso di input. La configurazione del flusso di input è definita in `PropertyGroupId = InputStream0`. Il nome e la regione del flusso si trovano nelle proprietà denominate `stream.name` `aws.region` rispettivamente. Per semplicità, questa fonte legge i record come una stringa.

```
private static FlinkKinesisConsumer<String> createSource(Properties  
    inputProperties) {  
    String inputStreamName = inputProperties.getProperty("stream.name");  
    return new FlinkKinesisConsumer<>(inputStreamName, new SimpleStringSchema(),  
        inputProperties);  
}  
...  
  
public static void main(String[] args) throws Exception {  
    ...  
    SourceFunction<String> source =  
        createSource(applicationParameters.get("InputStream0"));  
    DataStream<String> input = env.addSource(source, "Kinesis Source");  
    ...  
}
```

- L'applicazione definisce quindi un sink utilizzando il connettore [Kinesis Streams Sink](#) per inviare i dati al flusso di output. Il nome e la regione del flusso di output sono definiti in `PropertyGroupId =OutputStream0`, in modo simile al flusso di input. Il sink è collegato direttamente all'unità interna `DataStream` che riceve i dati dalla fonte. In un'applicazione reale, si verifica una trasformazione tra sorgente e sink.

```
private static KinesisStreamsSink<String> createSink(Properties outputProperties) {
    String outputStreamName = outputProperties.getProperty("stream.name");
    return KinesisStreamsSink.<String>builder()
        .setKinesisClientProperties(outputProperties)
        .setSerializationSchema(new SimpleStringSchema())
        .setStreamName(outputStreamName)
        .setPartitionKeyGenerator(element ->
            String.valueOf(element.hashCode()))
        .build();
}
...
public static void main(String[] args) throws Exception {
    ...
    Sink<String> sink = createSink(applicationParameters.get("OutputStream0"));
    input.sinkTo(sink);
    ...
}
```

- Infine, esegui il flusso di dati che hai appena definito. Questa deve essere l'ultima istruzione del `main()` metodo, dopo aver definito tutti gli operatori richiesti dal flusso di dati:

```
env.execute("Flink streaming Java API skeleton");
```

Utilizzate il file pom.xml

Il file `pom.xml` definisce tutte le dipendenze richieste dall'applicazione e configura il plugin Maven Shade per creare il fat-jar che contiene tutte le dipendenze richieste da Flink.

- `provided` Alcune dipendenze hanno un ambito. Queste dipendenze sono disponibili automaticamente quando l'applicazione viene eseguita in Amazon Managed Service for Apache Flink. Sono necessarie per compilare l'applicazione o per eseguirla localmente nel tuo IDE. Per ulteriori informazioni, consulta [Esegui l'applicazione localmente](#). Assicurati di utilizzare la stessa versione di Flink del runtime che utilizzerai in Amazon Managed Service for Apache Flink.

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-clients</artifactId>
  <version>${flink.version}</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-streaming-java</artifactId>
  <version>${flink.version}</version>
  <scope>provided</scope>
</dependency>
```

- È necessario aggiungere ulteriori dipendenze Apache Flink al pom con l'ambito predefinito, come il connettore [Kinesis](#) utilizzato da questa applicazione. Per ulteriori informazioni, consulta [Usa i connettori Apache Flink](#). È inoltre possibile aggiungere eventuali dipendenze Java aggiuntive richieste dall'applicazione.

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-kinesis</artifactId>
  <version>${aws.connector.version}</version>
</dependency>
```

- Il plugin Maven Java Compiler assicura che il codice sia compilato con Java 11, la versione JDK attualmente supportata da Apache Flink.
- Il plugin Maven Shade impacchetta il fat-jar, escludendo alcune librerie fornite dal runtime. Inoltre specifica due trasformatori: `e.ServicesResourceTransformer` e `ManifestResourceTransformer`. Quest'ultimo configura la classe contenente il main metodo per avviare l'applicazione. Se rinomini la classe principale, non dimenticare di aggiornare questo trasformatore.

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-shade-plugin</artifactId>
  ...
  <transformer
    implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
```

```
<mainClass>com.amazonaws.services.msf.BasicStreamingJob</mainClass>
</transformer>

...
</plugin>
```

Scrivi record di esempio nel flusso di input

In questa sezione, invierai record di esempio allo stream per l'elaborazione dell'applicazione. Sono disponibili due opzioni per generare dati di esempio, utilizzando uno script Python o il [Kinesis Data Generator](#).

Genera dati di esempio usando uno script Python

Puoi usare uno script Python per inviare record di esempio allo stream.

Note

Per eseguire questo script Python, devi usare Python 3.x e avere installata la libreria SDK [AWS for Python](#) (Boto).

Per iniziare a inviare i dati di test al flusso di input Kinesis:

1. Scarica lo script `stock.py` Python del generatore di [dati dal repository Data generator GitHub](#).
2. Esegui lo script `stock.py`:

```
$ python stock.py
```

Mantieni lo script in esecuzione mentre completi il resto del tutorial. Ora puoi eseguire la tua applicazione Apache Flink.

Generazione di dati di esempio utilizzando Kinesis Data Generator

In alternativa all'utilizzo dello script Python, puoi usare [Kinesis Data Generator](#), disponibile anche in una [versione ospitata](#), per inviare dati di esempio casuali allo stream. Kinesis Data Generator viene eseguito nel browser e non è necessario installare nulla sul computer.

Per configurare ed eseguire Kinesis Data Generator:

1. Segui le istruzioni nella [documentazione di Kinesis Data Generator](#) per configurare l'accesso allo strumento. Eseguirai un AWS CloudFormation modello che imposta un utente e una password.
2. Accedi a Kinesis Data Generator tramite l'URL generato dal CloudFormation modello. Puoi trovare l'URL nella scheda Output dopo aver completato il CloudFormation modello.
3. Configura il generatore di dati:
 - Regione: Seleziona la regione che stai utilizzando per questo tutorial: us-east-1
 - Stream/flusso di consegna: seleziona il flusso di input che l'applicazione utilizzerà: `ExampleInputStream`
 - Record al secondo: 100
 - Modello di registrazione: copia e incolla il seguente modello:

```
{
  "event_time" : "{{date.now("YYYY-MM-DDTkk:mm:ss.SSSS")}}",
  "ticker" : "{{random.arrayElement(
    ["AAPL", "AMZN", "MSFT", "INTC", "TBV"]
  )}}",
  "price" : {{random.number(100)}}
}
```

4. Prova il modello: scegli Modello di test e verifica che il record generato sia simile al seguente:

```
{ "event_time" : "2024-06-12T15:08:32.04800", "ticker" : "INTC", "price" : 7 }
```

5. Avvia il generatore di dati: scegli Seleziona Invia dati.

Kinesis Data Generator sta ora inviando dati a. `ExampleInputStream`

Esegui l'applicazione localmente

Puoi eseguire ed eseguire il debug dell'applicazione Flink localmente nel tuo IDE.

Note

Prima di continuare, verificate che i flussi di input e output siano disponibili. Per informazioni, consulta [Crea due flussi di dati Amazon Kinesis](#). Inoltre, verifica di disporre dell'autorizzazione per leggere e scrivere da entrambi gli stream. Per informazioni, consulta [Autentica la tua sessione AWS](#).

La configurazione dell'ambiente di sviluppo locale richiede Java 11 JDK, Apache Maven e un IDE per lo sviluppo in Java. Verifica di soddisfare i prerequisiti richiesti. Per informazioni, consulta [Soddisfa i prerequisiti per completare gli esercizi](#).

Importa il progetto Java nel tuo IDE

Per iniziare a lavorare sull'applicazione nel tuo IDE, devi importarla come progetto Java.

Il repository che avete clonato contiene diversi esempi. Ogni esempio è un progetto separato. Per questo tutorial, importa il contenuto della `./java/GettingStarted` sottodirectory nel tuo IDE.

Inserisci il codice come progetto Java esistente usando Maven.

Note

Il processo esatto per importare un nuovo progetto Java varia a seconda dell'IDE in uso.

Controllate la configurazione locale dell'applicazione

Quando viene eseguita localmente, l'applicazione utilizza la configurazione contenuta nel `application_properties.json` file nella cartella delle risorse del progetto sotto `./src/main/resources`. È possibile modificare questo file per utilizzare diversi nomi o regioni di stream Kinesis.

```
[
  {
    "PropertyGroupId": "InputStream0",
    "PropertyMap": {
      "stream.name": "ExampleInputStream",
      "flink.stream.initpos": "LATEST",
      "aws.region": "us-east-1"
    }
  },
  {
    "PropertyGroupId": "OutputStream0",
    "PropertyMap": {
      "stream.name": "ExampleOutputStream",
      "aws.region": "us-east-1"
    }
  }
]
```

```
}  
]
```

Configura la configurazione di esecuzione dell'IDE

Puoi eseguire ed eseguire il debug dell'applicazione Flink dal tuo IDE direttamente eseguendo la classe principale `com.amazonaws.services.msf.BasicStreamingJob`, come faresti con qualsiasi applicazione Java. Prima di eseguire l'applicazione, è necessario configurare la configurazione Run. La configurazione dipende dall'IDE in uso. Ad esempio, vedete le [configurazioni di esecuzione/debug nella](#) documentazione di IntelliJ IDEA. In particolare, è necessario configurare quanto segue:

1. Aggiungi le **provided** dipendenze al classpath. Ciò è necessario per assicurarsi che le dipendenze con provided scope vengano passate all'applicazione durante l'esecuzione locale. Senza questa configurazione, l'applicazione visualizza immediatamente un `class not found` errore.
2. Passa AWS le credenziali per accedere agli stream Kinesis all'applicazione. Il modo più veloce è usare [AWS Toolkit for IntelliJ IDEA](#). Utilizzando questo plugin IDE nella configurazione Run, è possibile selezionare un profilo specifico AWS. AWS l'autenticazione avviene utilizzando questo profilo. Non è necessario passare direttamente AWS le credenziali.
3. Verifica che l'IDE esegua l'applicazione utilizzando JDK 11.

Esegui l'applicazione nel tuo IDE

Dopo aver impostato la configurazione Run per `BasicStreamingJob`, potete eseguirla o eseguire il debug come una normale applicazione Java.

Note

Non è possibile eseguire il fat-jar generato da Maven direttamente dalla riga di `java -jar ...` comando. Questo jar non contiene le dipendenze principali di Flink necessarie per eseguire l'applicazione in modo autonomo.

Quando l'applicazione viene avviata correttamente, registra alcune informazioni sul minicluster autonomo e sull'inizializzazione dei connettori. Seguono una serie di registri INFO e alcuni registri WARN che Flink normalmente emette all'avvio dell'applicazione.

```

13:43:31,405 INFO  com.amazonaws.services.msf.BasicStreamingJob      [] -
Loading application properties from 'flink-application-properties-dev.json'
13:43:31,549 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Flink Kinesis Consumer is going to read the following streams:
ExampleInputStream,
13:43:31,676 INFO  org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils []
- The configuration option taskmanager.cpu.cores required for local execution is not
set, setting it to the maximal possible value.
13:43:31,676 INFO  org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils
[] - The configuration option taskmanager.memory.task.heap.size required for local
execution is not set, setting it to the maximal possible value.
13:43:31,676 INFO  org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils []
- The configuration option taskmanager.memory.task.off-heap.size required for local
execution is not set, setting it to the maximal possible value.
13:43:31,676 INFO  org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils []
- The configuration option taskmanager.memory.network.min required for local execution
is not set, setting it to its default value 64 mb.
13:43:31,676 INFO  org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils []
- The configuration option taskmanager.memory.network.max required for local execution
is not set, setting it to its default value 64 mb.
13:43:31,676 INFO  org.apache.flink.runtime.taskexecutor.TaskExecutorResourceUtils [] -
The configuration option taskmanager.memory.managed.size required for local execution
is not set, setting it to its default value 128 mb.
13:43:31,677 INFO  org.apache.flink.runtime.minicluster.Minicluster      [] -
Starting Flink Mini Cluster
....

```

Una volta completata l'inizializzazione, l'applicazione non emette ulteriori voci di registro. Durante il flusso di dati, non viene emesso alcun registro.

Per verificare se l'applicazione sta elaborando correttamente i dati, puoi controllare i flussi Kinesis di input e output, come descritto nella sezione seguente.

Note

Il comportamento normale di un'applicazione Flink è quello di non emettere registri relativi al flusso di dati. L'emissione di registri su ogni record può essere utile per il debug, ma può comportare un notevole sovraccarico durante l'esecuzione in produzione.

Osserva i dati di input e output nei flussi Kinesis

Puoi osservare i record inviati al flusso di input da (Python di esempio che genera) o Kinesis Data Generator (link) utilizzando Data Viewer nella console Amazon Kinesis.

Per osservare i record

1. [Apri la console Kinesis in /kinesis. https://console.aws.amazon.com](https://console.aws.amazon.com/kinesis)
2. Verifica che la regione sia la stessa in cui stai eseguendo questo tutorial, che per impostazione predefinita è us-east-1 US East (Virginia settentrionale). Cambia la regione se non corrisponde.
3. Scegli Data Streams.
4. Seleziona lo stream che desideri osservare, oppure ExampleInputStream o ExampleOutputStream.
5. Scegli la scheda Visualizzatore dati.
6. Scegli uno Shard, mantieni Ultimo come posizione iniziale, quindi scegli Ottieni record. Potresti visualizzare l'errore «Nessun record trovato per questa richiesta». In tal caso, scegli Riprova a recuperare i record. Vengono visualizzati i record più recenti pubblicati nello stream.
7. Scegli il valore nella colonna Dati per esaminare il contenuto del record in formato JSON.

Interrompi l'esecuzione locale dell'applicazione

Arresta l'esecuzione dell'applicazione nel tuo IDE. L'IDE di solito fornisce un'opzione di «stop». La posizione e il metodo esatti dipendono dall'IDE che stai utilizzando.

Compila e impacchetta il codice dell'applicazione

In questa sezione, si utilizza Apache Maven per compilare il codice Java e impacchettarlo in un file JAR. Puoi compilare e impacchettare il codice usando lo strumento da riga di comando Maven o il tuo IDE.

Per compilare e impacchettare usando la riga di comando Maven:

Passa alla directory contenente il GettingStarted progetto Java ed esegui il seguente comando:

```
$ mvn package
```

Per compilare e impacchettare usando il tuo IDE:

Esegui `mvn package` dalla tua integrazione IDE Maven.

In entrambi i casi, viene creato il seguente file JAR: `target/amazon-msf-java-stream-app-1.0.jar`

Note

L'esecuzione di un «progetto di compilazione» dal tuo IDE potrebbe non creare il file JAR.

Carica il file JAR del codice dell'applicazione

In questa sezione, carichi il file JAR creato nella sezione precedente nel bucket Amazon Simple Storage Service (Amazon S3) creato all'inizio di questo tutorial. Se non hai completato questo passaggio, consulta (link).

Per caricare il file JAR del codice dell'applicazione

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il bucket che hai creato in precedenza per il codice dell'applicazione.
3. Scegli Carica.
4. Scegliere Add files (Aggiungi file).
5. Passa al file JAR generato nel passaggio precedente: `target/amazon-msf-java-stream-app-1.0.jar`.
6. Scegli Carica senza modificare altre impostazioni.

Warning

Assicurati di selezionare il file JAR corretto in `<repo-dir>/java/GettingStarted/target/amazon-msf-java-stream-app-1.0.jar`.

La `target` directory contiene anche altri file JAR che non è necessario caricare.

Crea e configura l'applicazione Managed Service for Apache Flink

È possibile creare ed eseguire un'applicazione del servizio gestito per Apache Flink utilizzando la console o la AWS CLI. Per questo tutorial, utilizzerai la console.

 Note

Quando crei l'applicazione utilizzando la console, le tue risorse AWS Identity and Access Management (IAM) e Amazon CloudWatch Logs vengono create automaticamente. Quando crei l'applicazione utilizzando AWS CLI, crei queste risorse separatamente.

Argomenti

- [Creazione dell'applicazione](#)
- [Modifica la policy IAM](#)
- [Configura l'applicazione](#)
- [Esecuzione dell'applicazione.](#)
- [Osserva le metriche dell'applicazione in esecuzione](#)
- [Osserva i dati di output nei flussi Kinesis](#)
- [Arresta l'applicazione](#)

Creazione dell'applicazione

Per creare l'applicazione

1. Aprire la console Managed Service for Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Verifica che sia selezionata la regione corretta: us-east-1 Stati Uniti orientali (Virginia settentrionale)
3. Apri il menu a destra e scegli Applicazioni Apache Flink, quindi Crea applicazione di streaming. In alternativa, scegli Crea applicazione di streaming nel contenitore Get started della pagina iniziale.
4. Nella pagina Crea applicazione di streaming:
 - Scegliete un metodo per configurare l'applicazione di elaborazione dello stream: scegliete Crea da zero.
 - Configurazione Apache Flink, versione Application Flink: scegli Apache Flink 1.20.
5. Configura la tua applicazione
 - Nome dell'applicazione: invio **MyApplication**.

- Descrizione: immettere **My java test app**.
 - Accesso alle risorse dell'applicazione: scegli Crea/aggiorna il ruolo IAM **kinesis-analytics-MyApplication-us-east-1** con le policy richieste.
6. Configura il tuo modello per le impostazioni dell'applicazione
 - Modelli: scegli Sviluppo.
 7. Scegli Crea applicazione di streaming nella parte inferiore della pagina.

Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: **kinesis-analytics-service-MyApplication-us-east-1**
- Ruolo: **kinesisanalytics-MyApplication-us-east-1**

Amazon Managed Service for Apache Flink era precedentemente noto come Kinesis Data Analytics. Il nome delle risorse che vengono create automaticamente ha il prefisso per garantire la compatibilità con le versioni precedenti. **kinesis-analytics-**

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere ai flussi di dati Kinesis.

Per modificare la politica

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-east-1** creata dalla console nella sezione precedente.
3. Scegli Modifica, quindi scegli la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (**012345678901**) con l'ID del tuo account.


```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3:::my-bucket/kinesis-analytics-placeholder-s3-object"
      ]
    },
    {
      "Sid": "ListCloudwatchLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:012345678901:log-group:*"
      ]
    },
    {
      "Sid": "ListCloudwatchLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
      ]
    },
    {
      "Sid": "PutCloudwatchLogs",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [

```

```

        "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ],
    {
        "Sid": "ReadInputStream",
        "Effect": "Allow",
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleInputStream"
    },
    {
        "Sid": "WriteOutputStream",
        "Effect": "Allow",
        "Action": "kinesis:*",
        "Resource": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleOutputStream"
    }
]
}

```

5. Scegli Avanti nella parte inferiore della pagina, quindi scegli Salva modifiche.

Configura l'applicazione

Modifica la configurazione dell'applicazione per impostare l'elemento del codice dell'applicazione.

Per modificare la configurazione

1. Nella MyApplicationpagina, scegli Configura.
2. Nella sezione Posizione del codice dell'applicazione:
 - Per il bucket Amazon S3, seleziona il bucket creato in precedenza per il codice dell'applicazione. Scegli Sfoglia e seleziona il bucket corretto, quindi seleziona Scegli. Non fare clic sul nome del bucket.
 - Per Percorso dell'oggetto Amazon S3, inserisci **amazon-msf-java-stream-app-1.0.jar**
3. Per le autorizzazioni di accesso, scegli Crea/aggiorna il ruolo IAM **kinesis-analytics-MyApplication-us-east-1** con le politiche richieste.
4. Nella sezione Proprietà di runtime, aggiungi le seguenti proprietà.
5. Scegliete Aggiungi nuovo elemento e aggiungete ciascuno dei seguenti parametri:

ID gruppo	Chiave	Valore
InputStream0	stream.name	ExampleInputStream
InputStream0	aws.region	us-east-1
OutputStream0	stream.name	ExampleOutputStream
OutputStream0	aws.region	us-east-1

6. Non modificate nessuna delle altre sezioni.
7. Scegli Save changes (Salva modifiche).

Note

Quando scegli di abilitare la CloudWatch registrazione di Amazon, Managed Service for Apache Flink crea un gruppo di log e un flusso di log per te. I nomi di tali risorse sono i seguenti:

- Gruppo di log: /aws/kinesis-analytics/MyApplication
- Flusso di log: kinesis-analytics-log-stream

Esecuzione dell'applicazione.

L'applicazione è ora configurata e pronta per l'esecuzione.

Per eseguire l'applicazione

1. Sulla console per Amazon Managed Service for Apache Flink, scegli La mia applicazione e scegli Esegui.
2. Nella pagina successiva, nella pagina di configurazione del ripristino dell'applicazione, scegli Esegui con l'ultima istantanea, quindi scegli Esegui.

Lo stato nell'applicazione descrive in dettaglio le transizioni da Ready Starting e poi a Running quando l'applicazione è stata avviata.

Quando l'applicazione è nello `Running` stato, ora puoi aprire la dashboard di Flink.

Per aprire il pannello di controllo

1. Scegli Apri la dashboard di Apache Flink. La dashboard si apre in una nuova pagina.
2. Nell'elenco dei lavori in esecuzione, scegli il singolo lavoro che puoi vedere.

Note

Se imposti le proprietà di Runtime o modifichi le policy IAM in modo errato, lo stato dell'applicazione potrebbe trasformarsi in `Running`, ma la dashboard di Flink mostra che il job viene riavviato continuamente. Si tratta di uno scenario di errore comune se l'applicazione non è configurata correttamente o non dispone delle autorizzazioni per accedere alle risorse esterne.

Quando ciò accade, controlla la scheda Eccezioni nella dashboard di Flink per vedere la causa del problema.

Osserva le metriche dell'applicazione in esecuzione

Nella MyApplication pagina, nella sezione Amazon CloudWatch metrics, puoi vedere alcune delle metriche fondamentali dell'applicazione in esecuzione.

Per visualizzare le metriche

1. Accanto al pulsante Aggiorna, seleziona 10 secondi dall'elenco a discesa.
2. Quando l'applicazione è in esecuzione ed è integra, puoi vedere la metrica di uptime aumentare continuamente.
3. La metrica fullrestarts deve essere zero. Se è in aumento, la configurazione potrebbe presentare problemi. Per esaminare il problema, consulta la scheda Eccezioni nella dashboard di Flink.
4. La metrica del numero di checkpoint non riusciti deve essere pari a zero in un'applicazione integra.

 Note

Questa dashboard mostra un set fisso di metriche con una granularità di 5 minuti. Puoi creare una dashboard applicativa personalizzata con qualsiasi metrica nella dashboard. CloudWatch

Osserva i dati di output nei flussi Kinesis

Assicurati di continuare a pubblicare i dati sull'input, usando lo script Python o il Kinesis Data Generator.

È ora possibile osservare l'output dell'applicazione in esecuzione su Managed Service for Apache Flink utilizzando il Data Viewer in <https://console.aws.amazon.com/kinesis/>, analogamente a quanto già fatto in precedenza.

Per visualizzare l'output

1. [Apri la console Kinesis in /kinesis. https://console.aws.amazon.com](https://console.aws.amazon.com/kinesis/)
2. Verifica che la regione sia la stessa che stai usando per eseguire questo tutorial. Per impostazione predefinita, è US-East-1US East (Virginia settentrionale). Se necessario, modificare la regione.
3. Scegli Data Streams.
4. Seleziona lo stream che desideri osservare. Ai fini di questo tutorial, utilizza `ExampleOutputStream`.
5. Scegli la scheda Data viewer.
6. Seleziona uno Shard, mantieni Ultimo come posizione iniziale, quindi scegli Ottieni record. Potresti visualizzare l'errore «nessun record trovato per questa richiesta». In tal caso, scegli Riprova a recuperare i record. Vengono visualizzati i record più recenti pubblicati nello stream.
7. Seleziona il valore nella colonna Dati per esaminare il contenuto del record in formato JSON.

Arresta l'applicazione

Per interrompere l'applicazione, vai alla pagina della console dell'applicazione Managed Service for Apache Flink denominata. `MyApplication`

Per interrompere l'applicazione

1. Dall'elenco a discesa Azione, scegli Stop.
2. Lo stato nell'applicazione descrive in dettaglio le transizioni da Running e quindi a Ready quando l'applicazione viene completamente interrotta. Stopping

Note

Non dimenticare di interrompere anche l'invio di dati al flusso di input dallo script Python o dal Kinesis Data Generator.

Approfondimenti

[Pulisci le risorse AWS](#)

Pulisci le risorse AWS

Questa sezione include le procedure per la pulizia AWS delle risorse create in questo tutorial Getting Started (DataStream API).

Questo argomento contiene le sezioni seguenti:

- [Eliminare l'applicazione Managed Service for Apache Flink](#)
- [Eliminare i flussi di dati Kinesis](#)
- [Elimina oggetti e bucket Amazon S3](#)
- [Elimina le tue risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)
- [Esplora risorse aggiuntive per Apache Flink](#)

Eliminare l'applicazione Managed Service for Apache Flink

Per eliminare l'applicazione, utilizza la procedura seguente.

1. [Apri la console Kinesis in /kinesis. https://console.aws.amazon.com](https://console.aws.amazon.com/kinesis)
2. Nel pannello Managed Service for Apache Flink, scegli. MyApplication
3. Dall'elenco a discesa Azioni, scegliete Elimina e quindi confermate l'eliminazione.

Eliminare i flussi di dati Kinesis

1. Apri la console Managed Service for Apache Flink all'indirizzo /flink. <https://console.aws.amazon.com>
2. Scegli Flussi di dati.
3. Seleziona i due stream che hai creato e. `ExampleInputStream` `ExampleOutputStream`
4. Dall'elenco a discesa Azioni, scegli Elimina, quindi conferma l'eliminazione.

Elimina oggetti e bucket Amazon S3

Utilizza le seguenti procedure per eliminare oggetti e bucket Amazon S3.

Per eliminare l'oggetto dal bucket S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Seleziona il bucket S3 che hai creato per l'elemento dell'applicazione.
3. Seleziona l'elemento dell'applicazione che hai caricato, denominato. `amazon-msf-java-stream-app-1.0.jar`
4. Scegliete Elimina e confermate l'eliminazione.

Per eliminare il bucket S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Seleziona il bucket che hai creato per gli artefatti.
3. Scegliete Elimina e confermate l'eliminazione.

Note

Il bucket S3 deve essere vuoto per eliminarlo.

Elimina le tue risorse IAM

Per eliminare le risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.

2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-east-1.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytics- MyApplication -us-east-1.
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Quindi scegli Elimina gruppo di log e conferma l'eliminazione.

Esplora risorse aggiuntive per Apache Flink

[Esplora risorse aggiuntive](#)

Esplora risorse aggiuntive

Ora che hai creato ed eseguito un'applicazione di base del servizio gestito per Apache Flink, consulta le seguenti risorse per esplorare soluzioni del servizio gestito per Apache Flink più avanzate.

- [Workshop Amazon Managed Service per Apache Flink](#): in questo workshop, crei un'architettura di end-to-end streaming per importare, analizzare e visualizzare i dati di streaming quasi in tempo reale. Hai deciso di migliorare le attività di una compagnia di taxi a New York City. Analizza i dati di telemetria di una flotta di taxi a New York City quasi in tempo reale per ottimizzare le operazioni della flotta.
- [Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink](#): questa sezione della Guida per gli sviluppatori fornisce esempi di creazione e utilizzo di applicazioni nel servizio gestito per Apache Flink. Includono codice di esempio e step-by-step istruzioni per aiutarti a creare un servizio gestito per le applicazioni Apache Flink e testare i risultati.

- [Scopri Flink: esercitazioni pratiche](#): formazione introduttiva ufficiale su Apache Flink che ti consente di iniziare a scrivere applicazioni di streaming scalabili ETL, di analisi e basate su eventi.

Nozioni di base sull'utilizzo del servizio gestito da Amazon per Apache Flink (API Table)

Questa sezione presenta i concetti fondamentali del servizio gestito per Apache Flink e dell'implementazione di un'applicazione in Java utilizzando l'API Table e SQL. Descrive come passare da un'applicazione APIs all'altra all'interno della stessa applicazione e descrive le opzioni disponibili per la creazione e il test delle applicazioni. Fornisce inoltre istruzioni per l'installazione degli strumenti necessari per completare i tutorial di questa guida e creare la tua prima applicazione.

Argomenti

- [Esamina i componenti dell'applicazione del servizio gestito per Apache Flink](#)
- [Completa i prerequisiti obbligatori](#)
- [Creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink](#)
- [Approfondimenti](#)
- [Pulizia AWS delle risorse](#)
- [Esplora altre risorse](#)

Esamina i componenti dell'applicazione del servizio gestito per Apache Flink

Note

Il servizio gestito per Apache Flink supporta tutti i linguaggi [Apache Flink APIs](#) e potenzialmente tutti i linguaggi JVM. A seconda dell'API scelta, la struttura dell'applicazione e l'implementazione sono leggermente diverse. Questo tutorial illustra l'implementazione di applicazioni che utilizzano l'API Table e SQL e l'integrazione con l'DataStream API, implementata in Java.

Per elaborare i dati, l'applicazione del servizio gestito per Apache Flink utilizza un'applicazione Java che elabora l'input e produce l'output utilizzando il runtime di Apache Flink.

Una tipica applicazione Apache Flink include i seguenti componenti:

- **Proprietà di runtime:** è possibile utilizzare le proprietà di runtime per passare i parametri di configurazione all'applicazione senza modificare e ripubblicare il codice.
- **Fonti:** l'applicazione utilizza dati da una o più fonti. Una fonte utilizza un [connettore](#) per leggere i dati da un sistema esterno, ad esempio un flusso di dati Kinesis o un argomento Amazon MSK. Per lo sviluppo o il test, puoi anche fare in modo che le fonti generino dati di test in modo casuale. Per ulteriori informazioni, consulta [Aggiungi sorgenti di dati di streaming a Managed Service for Apache Flink](#). Con SQL o Table API, le fonti sono definite come tabelle di origine.
- **Trasformazioni:** l'applicazione elabora i dati attraverso una o più trasformazioni in grado di filtrare, arricchire o aggregare i dati. Quando si utilizza SQL o Table API, le trasformazioni sono definite come interrogazioni su tabelle o viste.
- **Sinks:** l'applicazione invia dati a sistemi esterni tramite sink. Un sink utilizza un [connettore](#) per inviare i dati a un sistema esterno, ad esempio un flusso di dati Kinesis, un argomento Amazon MSK, un bucket Amazon S3 o un database relazionale. Puoi anche usare un connettore speciale per stampare l'output solo a scopo di sviluppo. Quando si utilizza SQL o Table API, i sink sono definiti come tabelle sink in cui inserire i risultati. Per ulteriori informazioni, consulta [Scrivi dati utilizzando i sinks in Managed Service for Apache Flink](#).

L'applicazione richiede alcune dipendenze esterne, come i connettori Flink utilizzati dall'applicazione o potenzialmente una libreria Java. Per eseguire Amazon Managed Service for Apache Flink, devi impacchettare l'applicazione insieme alle dipendenze in un FAT-jar e caricarla in un bucket Amazon S3. Potrai quindi creare un'applicazione del servizio gestito per Apache Flink. Devi passare la posizione del pacchetto di codice, insieme ad altri parametri di configurazione di runtime. Questo tutorial dimostra come utilizzare Apache Maven per impacchettare l'applicazione e come eseguire l'applicazione localmente nell'IDE di tua scelta.

Completa i prerequisiti obbligatori

Prima di iniziare questo tutorial, completa le prime due fasi di [Inizia a usare Amazon Managed Service per Apache Flink \(DataStream API\)](#):

- [Soddisfa i prerequisiti per completare gli esercizi](#)
- [Configura il \(\) AWS Command Line InterfaceAWS CLI](#)

Per iniziare, consulta [Creazione di un'applicazione](#).

Creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink

In questo esercizio, viene creata un'applicazione del servizio gestito per Apache Flink con flussi di dati Kinesis come origine e come origine e come origine.

Questa sezione contiene le fasi seguenti.

- [Creazione delle risorse dipendenti](#)
- [Configurazione dell'ambiente di sviluppo locale](#)
- [Scarica ed esamina il codice Java per lo streaming di Apache Flink](#)
- [Esegui l'applicazione localmente](#)
- [Osservare l'applicazione che scrive i dati in un bucket S3](#)
- [Interruzione dell'applicazione in esecuzione locale](#)
- [Compila e impacchetta il codice dell'applicazione](#)
- [Caricare il file JAR del codice dell'applicazione](#)
- [Creazione e configurazione dell'applicazione del servizio gestito per Apache Flink](#)

Creazione delle risorse dipendenti

Prima di creare un servizio gestito per Apache Flink per questo esercizio, devi creare le risorse dipendenti seguenti:

- Un bucket Amazon S3 per memorizzare il codice dell'applicazione e per scrivere l'output dell'applicazione.

Note

Questo tutorial presuppone che l'applicazione stia distribuendo l'applicazione nella regione us-east-1. Se si utilizza un'altra regione, è necessario adattare tutti i passaggi di conseguenza.

Creazione di un bucket Amazon S3

Puoi creare un bucket Amazon S3 utilizzando la relativa console. Per istruzioni per la creazione di questa risorsa, consulta gli argomenti riportati di seguito:

- [Come si crea un bucket S3?](#) nella Guida per l'utente di Amazon Simple Storage Service. Assegna al bucket Amazon S3 un nome univoco globale aggiungendo il tuo nome di accesso.

Note

Assicurati di creare il bucket nella regione che utilizzi per questo tutorial. L'impostazione di default per il tutorial è us-east-1.

Altre risorse

Quando crei la tua applicazione, il servizio gestito per Apache Flink crea le seguenti CloudWatch risorse Amazon, se non esistono già:

- Un gruppo di log chiamato `/AWS/KinesisAnalytics-java/<my-application>`.
- Un flusso di log denominato `kinesis-analytics-log-stream`.

Configurazione dell'ambiente di sviluppo locale

Per lo sviluppo e il debug, puoi eseguire l'applicazione Apache Flink sulla tua macchina, direttamente dal tuo IDE preferito. Tutte le dipendenze di Apache Flink vengono gestite come normali dipendenze Java utilizzando Maven.

Note

Sulla tua macchina di sviluppo, devi avere Java JDK 11, Maven e Git installati. [Ti consigliamo di utilizzare un ambiente di sviluppo come Eclipse Java Neon o IntelliJ IDEA](#). Per verificare di soddisfare tutti i prerequisiti, vedere. [Soddisfa i prerequisiti per completare gli esercizi](#) Non è necessario installare un cluster Apache Flink sul computer.

Autentica la tua sessione AWS

L'applicazione utilizza i flussi di dati Kinesis per pubblicare i dati. Quando si esegue localmente, è necessario disporre di una sessione AWS autenticata valida con autorizzazioni di scrittura nel flusso di dati Kinesis. Per autenticare la sessione, segui i seguenti passaggi:

1. Se non hai configurato il profilo AWS CLI e un profilo denominato con credenziali valide, consulta. [Configura il \(\) AWS Command Line InterfaceAWS CLI](#)
2. Se l'IDE dispone di un plug-in con cui integrarsi AWS, è possibile utilizzarlo per passare le credenziali all'applicazione in esecuzione nell'IDE. Per ulteriori informazioni, vedere [AWS Toolkit for IntelliJ IDEA](#) [AWS e Toolkit per compilare l'applicazione](#) o eseguire Eclipse.

Scarica ed esamina il codice Java per lo streaming di Apache Flink

Il codice dell'applicazione per questo esempio è disponibile da GitHub.

Per scaricare il codice dell'applicazione Java

1. Clona il repository remoto con il comando seguente:

```
git clone https://github.com/aws-samples/amazon-managed-service-for-apache-flink-examples.git
```

2. Passa alla directory `./java/GettingStartedTable`.

Esamina i componenti dell'applicazione

L'applicazione è interamente implementata nella `com.amazonaws.services.msf.BasicTableJob` classe. Il `main()` metodo definisce sorgenti, trasformazioni e pozzi. L'esecuzione viene avviata da un'istruzione di esecuzione alla fine di questo metodo.

Note

Per un'esperienza di sviluppo ottimale, l'applicazione è progettata per funzionare senza modifiche al codice sia su Amazon Managed Service per Apache Flink che localmente, per lo sviluppo nel tuo IDE.

- Per leggere la configurazione di runtime in modo che funzioni durante l'esecuzione in Amazon Managed Service for Apache Flink e nel tuo IDE, l'applicazione rileva automaticamente se è in esecuzione autonoma localmente nell'IDE. In tal caso, l'applicazione carica la configurazione di runtime in modo diverso:
 1. Quando l'applicazione rileva che è in esecuzione in modalità autonoma nell'IDE, forma il `application_properties.json` file incluso nella cartella delle risorse del progetto. Segue il contenuto del file.
 2. Quando l'applicazione viene eseguita in Amazon del servizio gestito per Apache Flink, il comportamento predefinito carica la configurazione dell'applicazione dalle proprietà di runtime definite nell'applicazione Amazon del servizio gestito per Apache Flink. Consultare [Crea e configura l'applicazione Managed Service for Apache Flink](#).

```
private static Map<String, Properties>
loadApplicationProperties(StreamExecutionEnvironment env) throws IOException {
    if (env instanceof LocalStreamEnvironment) {
        LOGGER.info("Loading application properties from '{}'",
LOCAL_APPLICATION_PROPERTIES_RESOURCE);
        return KinesisAnalyticsRuntime.getApplicationProperties(
            BasicStreamingJob.class.getClassLoader()

.getResource(LOCAL_APPLICATION_PROPERTIES_RESOURCE).getPath());
    } else {
        LOGGER.info("Loading application properties from Amazon Managed Service for
Apache Flink");
        return KinesisAnalyticsRuntime.getApplicationProperties();
    }
}
```

- Il `main()` metodo definisce il flusso di dati dell'applicazione e lo esegue.
- Inizializza gli ambienti di streaming predefiniti. In questo esempio, mostriamo come creare sia l'API `StreamExecutionEnvironment` da utilizzare con l' `DataStream` API, sia `StreamTableEnvironment` quello da utilizzare con SQL e l'API `Table`. I due oggetti di ambiente sono due riferimenti separati allo stesso ambiente di runtime, da utilizzare in modo diverso APIs.

```
StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();
```

```
StreamTableEnvironment tableEnv = StreamTableEnvironment.create(env,
    EnvironmentSettings.newInstance().build());
```

- Caricate i parametri di configurazione dell'applicazione. Questo li caricherà automaticamente dalla posizione corretta, a seconda di dove è in esecuzione l'applicazione:

```
Map<String, Properties> applicationParameters = loadApplicationProperties(env);
```

- [Il connettore FileSystem sink utilizzato dall'applicazione per scrivere risultati nei file di output di Amazon S3 quando Flink completa un checkpoint.](#) È necessario abilitare i checkpoint per scrivere file nella destinazione. Quando l'applicazione è in esecuzione in Amazon Managed Service for Apache Flink, la configurazione dell'applicazione controlla il checkpoint e lo abilita per impostazione predefinita. Al contrario, quando viene eseguita localmente, i checkpoint sono disabilitati per impostazione predefinita. L'applicazione rileva che viene eseguita localmente e configura il checkpoint ogni 5.000 ms.

```
if (env instanceof LocalStreamEnvironment) {
    env.enableCheckpointing(5000);
}
```

- Questa applicazione non riceve dati da una fonte esterna effettiva. Genera dati casuali da elaborare tramite il [DataGen connettore](#). Questo connettore è disponibile per DataStream API, SQL e Table API. Per dimostrare l'integrazione tra APIs, l'applicazione utilizza la versione DataStream API perché offre maggiore flessibilità. Ogni record viene generato da una funzione generatrice chiamata `StockPriceGeneratorFunction` in questo caso, in cui è possibile inserire una logica personalizzata.

```
DataGeneratorSource<StockPrice> source = new DataGeneratorSource<>(
    new StockPriceGeneratorFunction(),
    Long.MAX_VALUE,
    RateLimiterStrategy.perSecond(recordPerSecond),
    TypeInformation.of(StockPrice.class));
```

- Nell' `DataStream API`, i record possono avere classi personalizzate. Le classi devono seguire regole specifiche in modo che Flink possa usarle come record. Per ulteriori informazioni, consulta [Tipi di dati supportati](#). In questo esempio, la `StockPrice` classe è un [POJO](#).
- La sorgente viene quindi collegata all'ambiente di esecuzione, generando un file `DataStream diStockPrice`. Questa applicazione non utilizza la [semantica degli eventi](#) e non genera

una filigrana. Esegui il DataGenerator codice sorgente con un parallelismo pari a 1, indipendentemente dal parallelismo del resto dell'applicazione.

```
DataStream<StockPrice> stockPrices = env.fromSource(  
    source,  
    WatermarkStrategy.noWatermarks(),  
    "data-generator"  
).setParallelism(1);
```

- Quanto segue nel flusso di elaborazione dei dati viene definito utilizzando l'API Table e SQL. Per fare ciò, convertiamo l' `DataStream` of `StockPrices` in una tabella. Lo schema della tabella viene automaticamente dedotto dalla `StockPrice` classe.

```
Table stockPricesTable = tableEnv.fromDataStream(stockPrices);
```

- Il seguente frammento di codice mostra come definire una vista e una query utilizzando l'API programmatica Table:

```
Table filteredStockPricesTable = stockPricesTable.  
    select(  
        $("eventTime").as("event_time"),  
        $("ticker"),  
        $("price"),  
        dateFormat($("eventTime"), "yyyy-MM-dd").as("dt"),  
        dateFormat($("eventTime"), "HH").as("hr")  
    ).where($("price").isGreater(50));  
  
tableEnv.createTemporaryView("filtered_stock_prices", filteredStockPricesTable);
```

- Viene definita una tabella sink per scrivere i risultati in un bucket Amazon S3 come file JSON. Per illustrare la differenza rispetto alla definizione di una vista a livello di codice, con l'API Table la tabella sink viene definita utilizzando SQL.

```
tableEnv.executeSql("CREATE TABLE s3_sink (" +  
    "eventTime TIMESTAMP(3)," +  
    "ticker STRING," +  
    "price DOUBLE," +  
    "dt STRING," +  
    "hr STRING" +  
    ") PARTITIONED BY ( dt, hr ) WITH (" +  
    "'connector' = 'filesystem'," +  
    "'format' = 'json'," +
```

```
    "'path' = 's3a:/' + s3Path + "' +  
    "');" );
```

- L'ultimo passaggio consiste nell'inserire la visualizzazione filtrata dei prezzi delle azioni nella tabella sink. `executeInsert()` Questo metodo avvia l'esecuzione del flusso di dati che abbiamo definito finora.

```
filteredStockPricesTable.executeInsert("s3_sink");
```

Usa il file pom.xml

Il file `pom.xml` definisce tutte le dipendenze richieste dall'applicazione e configura il plugin Maven Shade per creare il fat-jar che contiene tutte le dipendenze richieste da Flink.

- `provided` Alcune dipendenze hanno un ambito. Queste dipendenze sono disponibili automaticamente quando l'applicazione viene eseguita in Amazon del servizio gestito da Amazon per Apache Flink. Sono necessarie per l'applicazione o per l'applicazione localmente nel tuo IDE. Per ulteriori informazioni, consulta (aggiornamento a TableAPI). [Esegui l'applicazione localmente](#) Assicurati di utilizzare la stessa versione di Flink del runtime che utilizzerai in Amazon per Apache Flink. Per utilizzare TableAPI e SQL, devi includere `flink-table-planner-loader` e `flink-table-runtime-dependencies`, entrambi con ambito. `provided`

```
<dependency>  
  <groupId>org.apache.flink</groupId>  
  <artifactId>flink-streaming-java</artifactId>  
  <version>${flink.version}</version>  
  <scope>provided</scope>  
</dependency>  
<dependency>  
  <groupId>org.apache.flink</groupId>  
  <artifactId>flink-clients</artifactId>  
  <version>${flink.version}</version>  
  <scope>provided</scope>  
</dependency>  
<dependency>  
  <groupId>org.apache.flink</groupId>  
  <artifactId>flink-table-planner-loader</artifactId>  
  <version>${flink.version}</version>  
  <scope>provided</scope>  
</dependency>
```

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-table-runtime</artifactId>
  <version>${flink.version}</version>
  <scope>provided</scope>
</dependency>
```

- È necessario aggiungere ulteriori dipendenze Apache Flink al pom con l'ambito predefinito. [Ad esempio, il DataGen connettore, il connettore FileSystem SQL e il formato JSON.](#)

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-datagen</artifactId>
  <version>${flink.version}</version>
</dependency>
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-connector-files</artifactId>
  <version>${flink.version}</version>
</dependency>
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-json</artifactId>
  <version>${flink.version}</version>
</dependency>
```

- Per scrivere su Amazon S3 durante l'esecuzione locale, nell'ambito è incluso anche il file system S3 Hadoop. provided

```
<dependency>
  <groupId>org.apache.flink</groupId>
  <artifactId>flink-s3-fs-hadoop</artifactId>
  <version>${flink.version}</version>
  <scope>provided</scope>
</dependency>
```

- Il plugin Maven Java Compiler assicura che il codice sia compilato con Java 11, la versione JDK attualmente supportata da Apache Flink.

- Il plugin Maven Shade impacchetta il fat-jar, escludendo alcune librerie fornite dal runtime. Inoltre specifica due trasformatori: `e. ServicesResourceTransformer` e `ManifestResourceTransformer`. Quest'ultimo configura la classe contenente il main metodo per avviare l'applicazione. Se rinomini la classe principale, non dimenticare di aggiornare questo trasformatore.

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-shade-plugin</artifactId>
  ...
  <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">
    <mainClass>com.amazonaws.services.msf.BasicStreamingJob</mainClass>
  </transformer>
  ...
</plugin>
```

Esegui l'applicazione localmente

Puoi eseguire ed eseguire il debug dell'applicazione Flink localmente nel tuo IDE.

Note

Prima di continuare, verifica che i flussi di input e di output siano disponibili. Consultare [Crea due flussi di dati Amazon Kinesis](#). Inoltre, verifica di disporre dell'autorizzazione per leggere e scrivere da entrambi gli stream. Consultare [Autentica la tua sessione AWS](#).

La configurazione dell'ambiente di sviluppo locale richiede Java 11 JDK, Apache Maven e un IDE per lo sviluppo in Java. Verifica di soddisfare i prerequisiti richiesti. Consultare [Soddisfa i prerequisiti per completare gli esercizi](#).

Importa il progetto Java nel tuo IDE

Per iniziare a lavorare sull'applicazione nel tuo IDE, devi importarla come progetto Java.

Il repository che avete clonato contiene diversi esempi. Ogni esempio è un progetto separato. Per questo tutorial, importa il contenuto della `./jave/GettingStartedTable` sottodirectory nel tuo IDE.

Inserisci il codice come progetto Java esistente usando Maven.

 Note

Il processo esatto per importare un nuovo progetto Java varia a seconda dell'IDE in uso.

Modificate la configurazione locale dell'applicazione

Quando viene eseguita localmente, l'applicazione utilizza la configurazione contenuta nel `application_properties.json` file nella cartella delle risorse del progetto sotto `./src/main/resources`. Per questa applicazione tutorial, i parametri di configurazione sono il nome del bucket e il percorso in cui verranno scritti i dati.

Modifica la configurazione e il nome del bucket Amazon S3 in modo che corrisponda al bucket creato all'inizio di questo tutorial.

```
[
  {
    "PropertyGroupId": "bucket",
    "PropertyMap": {
      "name": "<bucket-name>",
      "path": "output"
    }
  }
]
```

 Note

La proprietà di configurazione name deve contenere solo il nome del bucket, ad esempio. `my-bucket-name` Non includere alcun prefisso, ad esempio una `s3://` barra finale. Se modificate il percorso, omettete le barre iniziali o finali.

Configurare la configurazione di esecuzione dell'IDE

Puoi eseguire ed eseguire il debug dell'applicazione Flink dal tuo IDE direttamente eseguendo la classe principale `com.amazonaws.services.msf.BasicTableJob`, come faresti con qualsiasi applicazione Java. Prima di eseguire l'applicazione, è necessario configurare la configurazione Esegui. La configurazione dipende dall'IDE che stai utilizzando. Ad esempio, vedete le [configurazioni](#)

[di esecuzione/debug nella](#) documentazione di IntelliJ IDEA. In particolare, è necessario configurare quanto segue:

1. Aggiungi le **provided** dipendenze al classpath. Ciò è necessario per assicurarsi che le dipendenze con provided scope vengano passate all'applicazione durante l'esecuzione locale. Senza questa configurazione, l'applicazione visualizza immediatamente un `class not found` errore.
2. Passa AWS le credenziali per accedere agli stream Kinesis all'applicazione. Il modo più veloce è usare [AWS Toolkit for IntelliJ IDEA](#). Utilizzando questo plugin IDE nella configurazione Run, è possibile selezionare un profilo specifico AWS. AWS l'autenticazione avviene utilizzando questo profilo. Non è necessario passare AWS le credenziali direttamente.
3. Verifica che l'IDE esegua l'applicazione utilizzando JDK 11.

Esegui l'applicazione nel tuo IDE

Dopo aver impostato la configurazione Run per `BasicTableJob`, potete eseguirla o eseguire il debug come una normale applicazione Java.

Note

Non è possibile eseguire il fat-jar generato da Maven direttamente dalla riga di `java -jar ...` comando. Questo jar non contiene le dipendenze principali di Flink necessarie per eseguire l'applicazione in modo autonomo.

Quando l'applicazione viene avviata correttamente, registra alcune informazioni sul minicluster autonomo e sull'inizializzazione dei connettori. Seguono una serie di registri INFO e alcuni registri WARN che Flink normalmente emette all'avvio dell'applicazione.

```
21:28:34,982 INFO    com.amazonaws.services.msf.BasicTableJob
                    [] - Loading application properties from 'flink-application-properties-
dev.json'
21:28:35,149 INFO    com.amazonaws.services.msf.BasicTableJob
                    [] - s3Path is ExampleBucket/my-output-bucket
...
```

Una volta completata l'inizializzazione, l'applicazione non emette ulteriori voci di registro. Durante il flusso di dati, non viene emesso alcun registro.

Per verificare se l'applicazione elabora correttamente i dati, potete controllare il contenuto del bucket di output, come descritto nella sezione seguente.

Note

Il comportamento normale di un'applicazione Flink è quello di non emettere registri relativi al flusso di dati. L'emissione di registri su ogni record può essere utile per il debug, ma può comportare un notevole sovraccarico durante l'esecuzione in produzione.

Osservare l'applicazione che scrive i dati in un bucket S3

Questa applicazione di esempio genera dati casuali internamente e li scrive nel bucket S3 di destinazione che hai configurato. A meno che non sia stato modificato il percorso di configurazione predefinito, i dati verranno scritti nel output percorso seguito dal partizionamento di dati e ore, nel formato. `./output/<yyyy-MM-dd>/<HH>`

Il [connettore FileSystem sink](#) crea nuovi file sul checkpoint Flink. Quando viene eseguita localmente, l'applicazione esegue un checkpoint ogni 5 secondi (5.000 millisecondi), come specificato nel codice.

```
if (env instanceof LocalStreamEnvironment) {  
    env.enableCheckpointing(5000);  
}
```

Per sfogliare il bucket S3 e osservare il file scritto dall'applicazione

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il bucket creato in precedenza.
3. Passa al output percorso, quindi alle cartelle di data e ora che corrispondono all'ora corrente nel fuso orario UTC.
4. Aggiorna periodicamente per osservare la comparsa di nuovi file ogni 5 secondi.
5. Seleziona e scarica un file per osservarne il contenuto.

Note

Per impostazione predefinita, i file non hanno estensioni. Il contenuto è formattato come JSON. Puoi aprire i file con qualsiasi editor di testo per ispezionarne il contenuto.

Interruzione dell'applicazione in esecuzione locale

Arresta l'esecuzione dell'applicazione nel tuo IDE. L'IDE di solito fornisce un'opzione di «stop». La posizione e il metodo esatti dipendono dall'IDE.

Compila e impacchetta il codice dell'applicazione

In questa sezione utilizzerai Apache Maven per compilare il codice Java e comprimerlo in un file JAR. Puoi compilare e comprimere il codice utilizzando lo strumento da riga di comando Maven o il tuo IDE.

Per compilare e impacchettare utilizzando la riga di comando Maven

Passare alla directory che contiene il GettingStarted progetto Java ed esegui il comando seguente:

```
$ mvn package
```

Per compilare e impacchettare usando il tuo IDE

Esegui `mvn package` dalla tua integrazione IDE Maven.

In entrambi i casi, `target/amazon-msf-java-table-app-1.0.jar` viene creato il file JAR.

Note

L'esecuzione di un progetto di compilazione dal tuo IDE potrebbe non creare il file JAR.

Caricare il file JAR del codice dell'applicazione

In questa sezione, il file JAR creato nella sezione precedente viene caricato nel bucket Amazon S3 creato all'inizio di questo tutorial. Se l'hai già fatto, completa. [Creazione di un bucket Amazon S3](#)

Per caricare il codice dell'applicazione

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il bucket che hai creato in precedenza per il codice dell'applicazione.
3. Scegli il campo Carica.

4. Scegliere Add files (Aggiungi file).
5. Passare al file JAR generato nella sezione precedente: `target/amazon-msf-java-table-app-1.0.jar`.
6. Scegli Carica senza modificare altre impostazioni.

 Warning

Assicurati di selezionare il file JAR corretto in `<repo-dir>/java/GettingStarted/target/amazon/msf-java-table-app-1.0.jar`.

La directory di destinazione contiene anche altri file JAR che non è necessario caricare.

Creazione e configurazione dell'applicazione del servizio gestito per Apache Flink

È possibile creare e creare un'applicazione del servizio gestito per Apache Flink utilizzando la console o la AWS CLI. Per questo tutorial utilizzerai la console.

 Note

Quando crei l'applicazione utilizzando la console, vengono create automaticamente le risorse AWS Identity and Access Management (IAM) e di File di CloudWatch log Amazon Logs.

Quando crei l'applicazione utilizzando la AWS CLI, è necessario creare queste risorse separatamente.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo `/flink https://console.aws.amazon.com`
2. Verifica che sia selezionata la regione corretta: Stati Uniti orientali (Virginia settentrionale) `us-east-1`.
3. Nel menu a destra, scegli Applicazioni Apache Flink, quindi scegli Crea applicazione di streaming. In alternativa, scegli Crea applicazione di streaming nella sezione Guida introduttiva della pagina iniziale.
4. Nella pagina Crea applicazione di streaming, completa quanto segue:

- Per Scegli un metodo per configurare l'applicazione di elaborazione dello stream, scegli Crea da zero.
 - Per la configurazione di Apache Flink, versione Application Flink, scegli Apache Flink 1.19.
 - Nella sezione Configurazione dell'applicazione, completa quanto segue:
 - Per Nome applicazione, immetti **MyApplication**.
 - Per Descrizione, inserisci **My Java Table API test app**.
 - Per accedere alle risorse dell'applicazione, scegli Create/update IAM role kinesis-analytics-MyApplication-us-east-1 with required policies.
 - In Modello per le impostazioni dell'applicazione, completa quanto segue:
 - Per Modelli, scegliete Sviluppo.
5. Scegli Crea applicazione di streaming.

Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: kinesis-analytics-service-*MyApplication-us-east-1*
- Ruolo: kinesisanalytics-*MyApplication-us-east-1*

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere al bucket Amazon S3.

Modifica della policy IAM per aggiungere le autorizzazioni per i bucket S3

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-east-1** creata dalla console nella sezione precedente.
3. Scegli Modifica, quindi scegli la scheda JSON.

4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'ID account di esempio (**012345678901**) con l'ID del tuo account e **<bucket-name>** con il nome del bucket S3 che hai creato.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3::my-bucket/kinesis-analytics-placeholder-s3-object"
      ]
    },
    {
      "Sid": "ListCloudwatchLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:012345678901:log-group:*"
      ]
    },
    {
      "Sid": "ListCloudwatchLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
      ]
    },
    {
      "Sid": "PutCloudwatchLogs",
      "Effect": "Allow",
      "Action": [
```

```

        "logs:PutLogEvents"
    ],
    "Resource": [
        "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
},
{
    "Sid": "WriteOutputBucket",
    "Effect": "Allow",
    "Action": "s3:*",
    "Resource": [
        "arn:aws:s3:::my-bucket"
    ]
}
]
}

```

5. Scegli Avanti e quindi seleziona Salva modifiche.

Configurazione dell'applicazione

Modificate l'applicazione per impostare l'elemento del codice dell'applicazione.

Per configurare l'applicazione

1. Nella MyApplicationpagina, scegli Configura.
2. Nella sezione Posizione del codice dell'applicazione, scegli Configura.
 - Per il bucket Amazon S3, seleziona il bucket creato in precedenza per il codice dell'applicazione. Scegli Sfoglia e seleziona il bucket corretto, quindi scegli Scegli. Non fare clic sul nome del bucket.
 - Per Percorso dell'oggetto Amazon S3, inserisci **amazon-msf-java-table-app-1.0.jar**
3. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-east-1** per il ruolo IAM.
4. Nella sezione Proprietà di runtime, aggiungi le seguenti proprietà.
5. Scegliete Aggiungi nuovo elemento e aggiungete ciascuno dei seguenti parametri:

ID gruppo	Chiave	Valore
bucket	name	your-bucket-name
bucket	path	output

6. Non modificare altre impostazioni.
7. Scegli Save changes (Salva modifiche).

Note

Quando scegli di abilitare la CloudWatch registrazione Amazon, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: /aws/kinesis-analytics/MyApplication
- Flusso di log: kinesis-analytics-log-stream

Esecuzione dell'applicazione.

L'applicazione è ora configurata e pronta per l'esecuzione.

Per eseguire l'applicazione

1. Torna alla pagina della console in Amazon del servizio gestito da Amazon per Apache Flink e scegli. MyApplication
2. Scegli Esegui per avviare l'applicazione.
3. Nella configurazione di ripristino dell'applicazione, scegli Esegui con l'ultima istantanea.
4. Seleziona Esegui.
5. Lo stato nell'applicazione descrive in dettaglio le transizioni da Starting e Ready a Running dopo l'avvio dell'applicazione.

Quando l'applicazione è in Running stato, puoi aprire la dashboard di Flink.

Per aprire la dashboard e visualizzare il lavoro

1. Scegli Open Apache Flink dashboard. La dashboard viene aperta in una nuova pagina.
2. Nell'elenco Running Jobs, scegli il singolo lavoro che puoi vedere.

Note

Se hai impostato le proprietà di runtime o hai modificato le policy IAM in modo errato, lo stato dell'applicazione potrebbe cambiare inRunning, ma la dashboard di Flink mostra che il job viene riavviato continuamente. Si tratta di uno scenario di errore comune quando l'applicazione non è configurata correttamente o non dispone delle autorizzazioni per accedere alle risorse esterne.

Quando ciò accade, controlla la scheda Eccezioni nella dashboard di Flink per indagare sulla causa del problema.

Osserva le metriche dell'applicazione in esecuzione

Nella MyApplicationpagina, nella sezione Amazon CloudWatch metrics, puoi vedere alcune delle metriche fondamentali dell'applicazione in esecuzione.

Per visualizzare i parametri

1. Accanto al pulsante Aggiorna, seleziona 10 secondi dall'elenco a discesa.
2. Quando l'applicazione è in esecuzione ed è integra, puoi vedere la metrica di uptime aumentare continuamente.
3. La metrica fullrestarts deve essere zero. Se è in aumento, la configurazione potrebbe presentare dei problemi. Consulta la scheda Eccezioni nella dashboard di Flink per esaminare il problema.
4. La metrica del numero di checkpoint non riusciti deve essere pari a zero in un'applicazione integra.

Note

Questa dashboard mostra un set fisso di metriche con una granularità di 5 minuti. Puoi creare una dashboard applicativa personalizzata con qualsiasi metrica nella dashboard. CloudWatch

Osservare l'applicazione che scrive i dati nel bucket di destinazione

Ora puoi osservare l'applicazione in esecuzione in Amazon del servizio gestito da Amazon per Apache Flink mentre scrive i file in Amazon S3.

Per osservare i file, segui la stessa procedura utilizzata per verificare che i file fossero scritti quando l'applicazione era in esecuzione localmente. Consultare [Osservare l'applicazione che scrive i dati in un bucket S3](#).

Ricordate che l'applicazione scrive nuovi file sul checkpoint Flink. Quando è in esecuzione su Amazon Managed Service for Apache Flink, i checkpoint sono abilitati per impostazione predefinita e vengono eseguiti ogni 60 secondi. L'applicazione crea nuovi file all'incirca ogni minuto.

Interruzione dell'applicazione

Per interrompere l'applicazione, vai alla pagina della console dell'applicazione Managed Service for Apache Flink denominata `MyApplication`

Per interrompere l'applicazione

1. Dall'elenco a discesa Azione, scegli Stop.
2. Lo stato nell'applicazione descrive in dettaglio le transizioni da Running e quindi a Ready quando l'applicazione viene completamente interrotta. Stopping

Note

Non dimenticare di interrompere anche l'invio di dati al flusso di input dallo script Python o dal Kinesis Data Generator.

Approfondimenti

[Pulizia AWS delle risorse](#)

Pulizia AWS delle risorse

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Nozioni di base (API Table).

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione di oggetti e di bucket Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)
- [Approfondimenti](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

Per eliminare l'applicazione, utilizza la procedura seguente.

Per eliminare l'applicazione

1. [Apri la console Kinesis all'indirizzo https://console.aws.amazon.com/kines.](https://console.aws.amazon.com/kines.)
2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Dall'elenco a discesa Azioni, scegliete Elimina e confermate l'eliminazione.

Eliminazione di oggetti e di bucket Amazon S3

Per eliminare gli oggetti e il bucket S3, utilizza la procedura seguente.

Per eliminare l'oggetto dell'applicazione dal bucket S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Seleziona il bucket S3 che hai creato.
3. Seleziona il nome dell'elemento dell'applicazione che hai caricato `amazon-msf-java-table-app-1.0.jar`, scegli Elimina, quindi conferma l'eliminazione.

Per eliminare tutti i file di output scritti dall'applicazione

1. Scegliere la cartella output.
2. Scegliere Delete (Elimina).
3. Confermare l'eliminazione definitiva del contenuto.

Per eliminare il bucket S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Seleziona il bucket S3 che hai creato.
3. Scegli Delete (Elimina) e conferma l'eliminazione.

Eliminazione delle risorse IAM

Per eliminare le risorse IAM, segui la procedura riportata di seguito.

Per eliminare le risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-east-1.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytics- MyApplication -us-east-1.
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

Per eliminare le CloudWatch risorse, utilizza la procedura seguente.

Per eliminare le tue CloudWatch risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

Approfondimenti

[Esplora altre risorse](#)

Esplora altre risorse

Ora che hai creato ed eseguito un'applicazione del servizio gestito per Apache Flink che utilizza l'API Table, consulta [Esplora risorse aggiuntive](#) in [Inizia a usare Amazon Managed Service per Apache Flink \(DataStream API\)](#).

Nozioni di base sull'utilizzo del servizio gestito da Amazon per Apache Flink per Python

Questa sezione presenta i concetti fondamentali di un servizio gestito per Apache Flink utilizzando Python e l'API Table. Descrive le opzioni disponibili per la creazione e il test delle applicazioni. Fornisce inoltre istruzioni per l'installazione degli strumenti necessari per completare i tutorial di questa guida e creare la tua prima applicazione.

Argomenti

- [Analisi dei componenti di un'applicazione del servizio gestito per Apache Flink](#)
- [Soddisfare i prerequisiti](#)
- [Creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink per Python](#)
- [Pulizia AWS delle risorse](#)

Analisi dei componenti di un'applicazione del servizio gestito per Apache Flink

Note

Servizio gestito da Amazon per Apache Flink supporta tutti gli [Apache Flink](#). APIs A seconda dell'API scelta, la struttura dell'applicazione è leggermente diversa. Un approccio popolare nello sviluppo di un'applicazione Apache Flink in Python consiste nel definire il flusso dell'applicazione utilizzando SQL incorporato nel codice Python. Questo è l'approccio che seguiamo nel seguente tutorial Gettgin Started.

Per elaborare i dati, l'applicazione del servizio gestito per Apache Flink utilizza uno script Python per definire il flusso di dati che elabora l'input e produce l'output utilizzando il runtime di Apache Flink.

Un'applicazione del servizio gestito per Apache Flink include i componenti riportati di seguito:

- Proprietà di runtime: è possibile utilizzare le proprietà di runtime per configurare l'applicazione senza ricompilare il codice dell'applicazione.
- Fonti: l'applicazione utilizza dati da una o più fonti. Un'origine utilizza un [connettore](#) per leggere i dati da un sistema esterno, ad esempio un flusso di dati Kinesis o un argomento Amazon MSK.

Puoi anche utilizzare connettori speciali per generare dati dall'interno dell'applicazione. Quando si utilizza SQL, l'applicazione definisce le fonti come tabelle di origine.

- **Trasformazioni:** l'applicazione elabora i dati utilizzando una o più trasformazioni in grado di filtrare, arricchire o aggregare i dati. Quando si utilizza SQL, l'applicazione definisce le trasformazioni come query SQL.
- **Sinks:** l'applicazione invia dati a fonti esterne tramite sink. Un sink utilizza un [connettore](#) per inviare dati a un sistema esterno, ad esempio un flusso di dati Kinesis, un argomento Amazon MSK, un bucket Amazon S3 o un database relazionale. Puoi anche usare un connettore speciale per stampare l'output per scopi di sviluppo. Quando si utilizza SQL, l'applicazione definisce i sink come tabelle sink in cui inserire i risultati. Per ulteriori informazioni, consulta [Scrivi dati utilizzando i sinks in Managed Service for Apache Flink](#).

La tua applicazione Python potrebbe richiedere anche dipendenze esterne, come librerie Python aggiuntive o qualsiasi connettore Flink utilizzato dall'applicazione. Quando impacchetti l'applicazione, dovete includere tutte le dipendenze richieste dall'applicazione. Questo tutorial dimostra come includere le dipendenze dei connettori e come impacchettare l'applicazione per la distribuzione su Amazon Managed Service for Apache Flink.

Soddisfare i prerequisiti

Per completare questo tutorial, è necessario disporre di quanto segue:

- Python 3.11 , [preferibilmente utilizzando un ambiente autonomo come VirtualEnv \(venv\), Conda o Miniconda](#).
- [Client Git](#): installa il client Git se non l'hai già fatto.
- [Java Development Kit \(JDK\) versione 11](#): installa un Java JDK 11 e imposta la variabile di JAVA_HOME ambiente in modo che punti alla posizione di installazione. Se non disponi di un JDK 11, puoi utilizzare uno qualsiasi JDK standard di [Amazon Corretto](#) nostra scelta.
- Per verificare che il JDK sia installato correttamente, esegui il seguente comando. L'output sarà diverso se utilizzi un JDK diverso da Amazon Corretto 11. Assicurati che la versione sia 11.x.

```
$ java --version
```

```
openjdk 11.0.23 2024-04-16 LTS
OpenJDK Runtime Environment Corretto-11.0.23.9.1 (build 11.0.23+9-LTS)
OpenJDK 64-Bit Server VM Corretto-11.0.23.9.1 (build 11.0.23+9-LTS, mixed mode)
```

- [Apache Maven](#): installa Apache Maven se non è già stato fatto. [Per ulteriori informazioni, consulta Installazione di Apache Maven.](#)
- Per testare l'installazione di Apache Maven, utilizzare il comando seguente:

```
$ mvn -version
```

Note

Sebbene l'applicazione sia scritta in Python, Apache Flink viene eseguito nella Java Virtual Machine (JVM). Distribuisce la maggior parte delle dipendenze, come il connettore Kinesis, come file JAR. [Per gestire queste dipendenze e impacchettare l'applicazione in un file ZIP, utilizzate Apache Maven.](#) Questo tutorial spiega come farlo.

Warning

Ti consigliamo di utilizzare Python 3.11 per lo sviluppo locale. Questa è la stessa versione di Python utilizzata da Amazon Managed Service per Apache Flink con il runtime Flink 1.19. L'installazione della libreria Python Flink 1.19 su Python 3.12 potrebbe non riuscire. Se hai un'altra versione di Python installata di default sul tuo computer, ti consigliamo di creare un ambiente autonomo come usare VirtualEnv Python 3.11.

IDE per lo sviluppo locale

Ti consigliamo di utilizzare un ambiente di sviluppo come [PyCharmVisual Studio Code](#) per sviluppare e compilare l'applicazione.

Quindi, completa i primi due passaggi di [Inizia a usare Amazon Managed Service per Apache Flink \(DataStream API\)](#):

- [Configura un AWS account e crea un utente amministratore](#)
- [Configura il \(\) AWS Command Line InterfaceAWS CLI](#)

Per iniziare, consulta [Creazione di un'applicazione.](#)

Creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink per Python

In questa sezione, viene creata un'applicazione del servizio gestito per Apache Flink per applicazioni Python con un flusso Kinesis come origine e come sink.

Questa sezione contiene le fasi seguenti.

- [Creazione delle risorse dipendenti](#)
- [Configurazione dell'ambiente di sviluppo locale](#)
- [Scarica ed esamina il codice Python per lo streaming di Apache Flink](#)
- [Gestione delle dipendenze JAR](#)
- [Scrivi record di esempio nel flusso di input](#)
- [Esegui l'applicazione localmente](#)
- [Osserva i dati di input e output nei flussi Kinesis](#)
- [Arresta l'esecuzione dell'applicazione in locale](#)
- [Package del codice dell'applicazione](#)
- [Caricare il pacchetto dell'applicazione in un bucket Amazon S3](#)
- [Creazione e configurazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Approfondimenti](#)

Creazione delle risorse dipendenti

Prima di creare un servizio gestito per Apache Flink per questo esercizio, devi creare le risorse dipendenti seguenti:

- Due flussi Kinesis per l'input e l'output.
- Un bucket Amazon S3 per memorizzare il codice dell'applicazione.

Note

Questo tutorial presuppone che l'applicazione venga distribuita nella regione Stati Uniti orientali 1. Se si utilizza un'altra regione, è necessario adattare tutti i passaggi di conseguenza.

Creazione di due flussi Kinesis

Prima di creare un'applicazione del servizio gestito per Apache Flink per questo esercizio, crea due flussi di dati Kinesis (`ExampleInputStream` e `ExampleOutputStream`) nella stessa regione che utilizzerai per distribuire l'applicazione (us-east-1 in questo esempio). L'applicazione utilizza questi flussi per i flussi di origine e di destinazione dell'applicazione.

Puoi creare questi flussi utilizzando la console Amazon Kinesis o il comando AWS CLI seguente. Per istruzioni sulla console, consulta [Creazione e aggiornamento dei flussi di dati](#) nella Guida per gli sviluppatori del flusso di dati Amazon Kinesis.

Per creare i flussi di dati (AWS CLI)

1. Per creare il primo flusso (`ExampleInputStream`), utilizza il seguente comando Amazon Kinesis `create-stream` AWS CLI .

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

2. Per creare il secondo flusso utilizzato dall'applicazione per scrivere l'output, esegui lo stesso comando, modificando il nome del flusso in `ExampleOutputStream`.

```
$ aws kinesis create-stream \
--stream-name ExampleOutputStream \
--shard-count 1 \
--region us-east-1
```

Creazione di un bucket Amazon S3

Puoi creare un bucket Amazon S3 utilizzando la relativa console. Per istruzioni per la creazione di questa risorsa, consulta gli argomenti riportati di seguito:

- [Come si crea un bucket S3?](#) nella Guida per l'utente di Amazon Simple Storage Service. Assegna al bucket Amazon S3 un nome univoco globale, ad esempio aggiungendo il tuo nome di accesso.

 Note

Assicurati di creare il bucket S3 nella regione che utilizzi per questo tutorial (us-east-1).

Altre risorse

Quando crei la tua applicazione, il servizio gestito per Apache Flink crea le seguenti CloudWatch risorse Amazon, se non esistono già:

- Un gruppo di log chiamato `/AWS/KinesisAnalytics-java/<my-application>`.
- Un flusso di log denominato `kinesis-analytics-log-stream`.

Configurazione dell'ambiente di sviluppo locale

Per lo sviluppo e il debug, puoi eseguire l'applicazione Python Flink sul tuo computer. Puoi avviare l'applicazione dalla riga di comando con `python main.py` o in un IDE Python a tua scelta.

 Note

Sulla macchina di sviluppo, devi avere Python 3.10 o 3.11, Java 11, Apache Maven e Git installati. [Ti consigliamo di usare un IDE come Visual Studio Code. PyCharm](#) Per verificare che tutti i prerequisiti siano soddisfatti, consulta [Soddisfa i prerequisiti per completare gli esercizi](#) prima di procedere.

Installare la PyFlink libreria

Per sviluppare l'applicazione ed eseguirla localmente, è necessario installare la libreria Flink Python.

1. Crea un ambiente Python autonomo VirtualEnv usando Conda o qualsiasi strumento Python simile.
2. Installa la PyFlink libreria in quell'ambiente. Usa la stessa versione di runtime di Apache Flink che utilizzerai in Servizio gestito da Amazon per Apache Flink. Attualmente, il runtime consigliato è 1.19.1.

```
$ pip install apache-flink==1.19.1
```


3. Assicurati che l'ambiente sia attivo quando esegui l'applicazione. Se esegui l'applicazione nell'IDE, assicurati che l'IDE utilizzi l'ambiente come runtime. Il processo dipende dall'IDE che stai utilizzando.

Note

Devi solo installare la PyFlink libreria. Non è necessario installare un cluster Apache Flink sul computer.

Autentica la tua sessione AWS

L'applicazione utilizza i flussi di dati Kinesis per pubblicare i dati. Quando si esegue localmente, è necessario disporre di una sessione AWS autenticata valida con autorizzazioni di scrittura nel flusso di dati Kinesis. Per autenticare la sessione, segui questi passaggi:

1. Se non hai configurato il profilo AWS CLI e un profilo denominato con credenziali valide, consulta. [Configura il \(\) AWS Command Line InterfaceAWS CLI](#)
2. Verifica che AWS CLI sia configurato correttamente e che gli utenti dispongano delle autorizzazioni per scrivere nel flusso di dati Kinesis pubblicando il seguente record di test:

```
$ aws kinesis put-record --stream-name ExampleOutputStream --data TEST --partition-key TEST
```

3. Se il tuo IDE ha un plugin con cui integrarti AWS, puoi utilizzarlo per passare le credenziali all'applicazione in esecuzione nell'IDE. Per ulteriori informazioni, vedere [AWS Toolkit for PyCharm](#), [AWS Toolkit for Visual Studio Code](#) e [AWS Toolkit for IntelliJ IDEA](#).

Scarica ed esamina il codice Python per lo streaming di Apache Flink

Il codice dell'applicazione Python per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito:

1. Clona il repository remoto con il comando seguente:

```
git clone https://github.com/aws-samples/amazon-managed-service-for-apache-flink-examples.git
```

2. Passa alla directory `./python/GettingStarted`.

Esamina i componenti dell'applicazione

Il codice dell'applicazione si trova in `main.py`. Utilizziamo SQL incorporato in Python per definire il flusso dell'applicazione.

Note

Per un'esperienza di sviluppo ottimizzata, l'applicazione è progettata per funzionare senza modifiche al codice sia su Amazon Managed Service per Apache Flink che localmente, per lo sviluppo sulla tua macchina. L'applicazione utilizza la variabile di ambiente `IS_LOCAL = true` per rilevare quando è in esecuzione localmente. È necessario impostare la variabile `IS_LOCAL = true` di ambiente sulla shell o nella configurazione di esecuzione dell'IDE.

- L'applicazione configura l'ambiente di esecuzione e legge la configurazione di runtime. Per funzionare sia su Amazon Managed Service for Apache Flink che localmente, l'applicazione controlla la `IS_LOCAL` variabile.
- Di seguito è riportato il comportamento predefinito quando l'applicazione viene eseguita in Servizio gestito da Amazon per Apache Flink:
 1. Carica le dipendenze incluse nel pacchetto dell'applicazione. Per ulteriori informazioni, consulta (collegamento)
 2. Carica la configurazione dalle proprietà di runtime che definisci nell'applicazione Amazon per Apache Flink. Per ulteriori informazioni, consulta (collegamento)
- Quando l'applicazione rileva `IS_LOCAL = true` quando l'applicazione viene eseguita localmente:
 1. Carica le dipendenze esterne dal progetto.
 2. Carica la configurazione dal `application_properties.json` file incluso nel progetto.

```
...
APPLICATION_PROPERTIES_FILE_PATH = "/etc/flink/application_properties.json"
...
is_local = (
    True if os.environ.get("IS_LOCAL") else False
)
...
if is_local:
    APPLICATION_PROPERTIES_FILE_PATH = "application_properties.json"
    CURRENT_DIR = os.path.dirname(os.path.realpath(__file__))
```

```
table_env.get_config().get_configuration().set_string(
    "pipeline.jars",
    "file:/// " + CURRENT_DIR + "/target/pyflink-dependencies.jar",
)
```

- L'applicazione definisce una tabella di origine con un'CREATE TABLE istruzione, utilizzando il [Kinesis Connector](#). Questa tabella legge i dati dal flusso Kinesis di input. L'applicazione prende il nome dello stream, la regione e la posizione iniziale dalla configurazione di runtime.

```
table_env.execute_sql(f"""
    CREATE TABLE prices (
        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' = '{input_stream_name}',
        'aws.region' = '{input_stream_region}',
        'format' = 'json',
        'json.timestamp-format.standard' = 'ISO-8601'
    ) """)
```

- L'applicazione definisce anche una tabella sink utilizzando il [Kinesis Connector](#) in questo esempio. Questo racconto invia i dati allo stream Kinesis in uscita.

```
table_env.execute_sql(f"""
    CREATE TABLE output (
        ticker VARCHAR(6),
        price DOUBLE,
        event_time TIMESTAMP(3)
    )
    PARTITIONED BY (ticker)
    WITH (
        'connector' = 'kinesis',
        'stream' = '{output_stream_name}',
        'aws.region' = '{output_stream_region}',
        'sink.partitioner-field-delimiter' = ';',
        'sink.batch.max-size' = '100',
        'format' = 'json',
        'json.timestamp-format.standard' = 'ISO-8601'
    ) """)
```

```
)""")
```

- Infine, l'applicazione esegue un codice SQL `INSERT INTO . . .` derivante dalla tabella sorgente. In un'applicazione più complessa, è probabile che siano necessari passaggi aggiuntivi per trasformare i dati prima di scriverli nel sink.

```
table_result = table_env.execute_sql("""INSERT INTO output
    SELECT ticker, price, event_time FROM prices""")
```

- È necessario aggiungere un altro passaggio alla fine della `main()` funzione per eseguire l'applicazione localmente:

```
if is_local:
    table_result.wait()
```

Senza questa istruzione, l'applicazione viene terminata immediatamente quando viene eseguita localmente. Non devi eseguire questa istruzione quando esegui la tua applicazione in Servizio gestito da Amazon per Apache Flink.

Gestione delle dipendenze JAR

Un' PyFlink applicazione richiede in genere uno o più connettori. L'applicazione in questo tutorial utilizza il [Kinesis](#) Connector. Poiché Apache Flink viene eseguito in Java JVM, i connettori vengono distribuiti come file JAR, indipendentemente dal fatto che l'applicazione venga implementata in Python. È necessario impacchettare queste dipendenze con l'applicazione quando la si distribuisce su Amazon Managed Service for Apache Flink.

In questo esempio, mostriamo come usare Apache Maven per recuperare le dipendenze e impacchettare l'applicazione da eseguire su Managed Service for Apache Flink.

Note

Esistono modi alternativi per recuperare e impacchettare le dipendenze. Questo esempio dimostra un metodo che funziona correttamente con uno o più connettori. Consente inoltre di eseguire l'applicazione localmente, per lo sviluppo e su Managed Service for Apache Flink senza modifiche al codice.

Usa il file pom.xml

Apache Maven utilizza il `pom.xml` file per controllare le dipendenze e il pacchetto delle applicazioni.

Tutte le dipendenze JAR sono specificate nel file del `pom.xml` blocco. `<dependencies>...</dependencies>`

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/
xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  ...
  <dependencies>
    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-connector-kinesis</artifactId>
      <version>4.3.0-1.19</version>
    </dependency>
  </dependencies>
  ...
```

Per trovare l'elemento e la versione del connettore corretti da utilizzare, vedere. [Usa i connettori Apache Flink con Managed Service for Apache Flink](#) Assicurati di fare riferimento alla versione di Apache Flink che stai utilizzando. Ad esempio, noi utilizziamo il connettore Kinesis. Per Apache Flink 1.19, la versione del connettore è. `4.3.0-1.19`

Note

Se si utilizza Apache Flink 1.19, non è stata rilasciata alcuna versione del connettore specifica per questa versione. Utilizzate i connettori rilasciati per la versione 1.18.

Dipendenze da scaricare e impacchettare

Usa Maven per scaricare le dipendenze definite nel `pom.xml` file e impacchettarle per l'applicazione Python Flink.

1. Vai alla directory che contiene il progetto Python Getting Started chiamato. `python/GettingStarted`
2. Esegui il comando seguente:

```
$ mvn package
```

Maven crea un nuovo file chiamato. `./target/pyflink-dependencies.jar` Quando sviluppate localmente sulla vostra macchina, l'applicazione Python cerca questo file.

Note

Se dimentichi di eseguire questo comando, quando tenti di eseguire l'applicazione, l'applicazione fallirà e restituirà l'errore: Could not find any factory for identifier «kinesis».

Scrivi record di esempio nel flusso di input

In questa sezione, invierai record di esempio al flusso per l'applicazione da elaborare. Sono disponibili due opzioni per generare dati di esempio, utilizzando uno script Python o il [Kinesis Data Generator](#).

Genera dati di esempio usando uno script Python

Puoi usare uno script Python per inviare record di esempio allo stream.

Note

Per eseguire questo script Python, devi usare Python 3.x e avere installata la libreria SDK [AWS for Python](#) (Boto).

Per iniziare a inviare i dati di test al flusso di input Kinesis:

1. Scarica lo script `stock.py` Python del generatore di [dati dal repository Data generator GitHub](#).
2. Esegui lo script `stock.py`:

```
$ python stock.py
```

Mantieni lo script in esecuzione mentre completi il resto del tutorial. Ora puoi eseguire l'applicazione Apache Flink.

Generazione di dati di esempio utilizzando Kinesis Data Generator

In alternativa all'utilizzo dello script Python, puoi usare [Kinesis Data Generator](#), disponibile anche in una [versione ospitata](#), per inviare dati di esempio casuali allo stream. Kinesis Data Generator viene eseguito nel browser e non è necessario installare nulla sul computer.

Per configurare ed eseguire Kinesis Data Generator:

1. Segui le istruzioni nella [documentazione di Kinesis Data Generator](#) per configurare l'accesso allo strumento. Eseguirai un AWS CloudFormation modello che imposta un utente e una password.
2. Accedi a Kinesis Data Generator tramite l'URL generato dal CloudFormation modello. Puoi trovare l'URL nella scheda Output dopo aver completato il CloudFormation modello.
3. Configura il generatore di dati:
 - Regione: Seleziona la regione che stai utilizzando per questo tutorial: us-east-1
 - Stream/flusso di consegna: seleziona il flusso di input che l'applicazione utilizzerà: `ExampleInputStream`
 - Record al secondo: 100
 - Modello di registrazione: copia e incolla il seguente modello:

```
{
  "event_time" : "{{date.now("YYYY-MM-DDTkk:mm:ss.SSSSS")}}",
  "ticker" : "{{random.arrayElement(
    ["AAPL", "AMZN", "MSFT", "INTC", "TBV"]
  )}}",
  "price" : {{random.number(100)}}
}
```

4. Prova il modello: scegli Modello di test e verifica che il record generato sia simile al seguente:

```
{ "event_time" : "2024-06-12T15:08:32.04800", "ticker" : "INTC", "price" : 7 }
```

5. Avvia il generatore di dati: scegli Seleziona Invia dati.

Kinesis Data Generator sta ora inviando dati a `ExampleInputStream`

Esegui l'applicazione localmente

Puoi testare l'applicazione localmente, eseguendola dalla riga di comando con `python main.py` o dal tuo IDE.

Per eseguire l'applicazione localmente, è necessario che sia installata la versione corretta della PyFlink libreria, come descritto nella sezione precedente. Per ulteriori informazioni, consulta ([collegamento](#))

Note

Prima di continuare, verifica che i flussi di input e output siano disponibili. Consultare [Crea due flussi di dati Amazon Kinesis](#). Inoltre, verifica di disporre dell'autorizzazione per leggere e scrivere da entrambi gli stream. Consultare [Autentica la tua sessione AWS](#).

Importa il progetto Python nel tuo IDE

Per iniziare a lavorare sull'applicazione nel tuo IDE, devi importarla come progetto Python.

Il repository che hai clonato contiene diversi esempi. Ogni esempio è un progetto separato. Per questo tutorial, importa il contenuto della `./python/GettingStarted` sottodirectory nel tuo IDE.

Importa il codice come progetto Python esistente.

Note

Il processo esatto per importare un nuovo progetto Python varia a seconda dell'IDE utilizzato.

Controllate la configurazione locale dell'applicazione

Quando viene eseguita localmente, l'applicazione utilizza la configurazione contenuta nel `application_properties.json` file nella cartella delle risorse del progetto sotto `./src/main/resources`. È possibile modificare questo file per utilizzare diversi nomi o regioni di stream Kinesis.

```
[
  {
    "PropertyGroupId": "InputStream0",
    "PropertyMap": {
      "stream.name": "ExampleInputStream",
      "flink.stream.initpos": "LATEST",
      "aws.region": "us-east-1"
    }
  },
]
```



```
{
  "PropertyGroupId": "OutputStream0",
  "PropertyMap": {
    "stream.name": "ExampleOutputStream",
    "aws.region": "us-east-1"
  }
}
```

Esegui la tua applicazione Python localmente

Puoi eseguire l'applicazione localmente, dalla riga di comando come un normale script Python o dall'IDE.

Per eseguire l'applicazione dalla riga di comando

1. Assicurati che l'ambiente Python standalone come Conda VirtualEnv o dove hai installato la libreria Python Flink sia attualmente attivo.
2. Assicurati di aver corso `mvn package` almeno una volta.
3. Imposta la variabile di ambiente `IS_LOCAL = true`:

```
$ export IS_LOCAL=true
```

4. Esegui l'applicazione come un normale script Python.

```
$python main.py
```

Per eseguire l'applicazione dall'interno dell'IDE

1. Configura il tuo IDE per eseguire lo `main.py` script con la seguente configurazione:
 1. Usa l'ambiente Python autonomo come Conda VirtualEnv o dove hai installato la libreria PyFlink
 2. Usa le AWS credenziali per accedere ai flussi di dati Kinesis di input e output.
 3. Imposta `IS_LOCAL = true`.
2. Il processo esatto per impostare la configurazione di esecuzione dipende dall'IDE e varia.
3. Dopo aver configurato l'IDE, esegui lo script Python e usa gli strumenti forniti dall'IDE mentre l'applicazione è in esecuzione.

Ispezionate i log delle applicazioni localmente

Quando viene eseguita localmente, l'applicazione non mostra alcun registro nella console, a parte alcune righe stampate e visualizzate all'avvio dell'applicazione. PyFlink scrive i log in un file nella directory in cui è installata la libreria Python Flink. L'applicazione stampa la posizione dei log all'avvio. Per trovare i log, è anche possibile eseguire il comando seguente:

```
$ python -c "import pyflink;import
os;print(os.path.dirname(os.path.abspath(pyflink.__file__))+'/log')"
```

1. Elenca i file nella directory di registrazione. Di solito si trova un solo `.log` file.
2. Coda il file mentre l'applicazione è in esecuzione: `tail -f <log-path>/<log-file>.log`.

Osserva i dati di input e output nei flussi Kinesis

Puoi osservare i record inviati al flusso di input da (Python di esempio che genera) o Kinesis Data Generator ([link](#)) utilizzando Data Viewer nella console Amazon Kinesis.

Per osservare i record:

Arresta l'esecuzione dell'applicazione in locale

Arresta l'esecuzione dell'applicazione nel tuo IDE. L'IDE di solito fornisce un'opzione di «stop». La posizione e il metodo esatti dipendono dall'IDE.

Package del codice dell'applicazione

In questa sezione, si utilizza Apache Maven per impacchettare il codice dell'applicazione e tutte le dipendenze richieste in un file.zip.

Esegui nuovamente il comando del pacchetto Maven:

```
$ mvn package
```

Questo comando genera il file. `target/managed-flink-pyflink-getting-started-1.0.0.zip`

Caricare il pacchetto dell'applicazione in un bucket Amazon S3

In questa sezione, carichi il file .zip creato nella sezione precedente nel bucket Amazon Simple Storage Service (Amazon S3) creato all'inizio di questo tutorial. Se non hai completato questo passaggio, consulta (link).

Per caricare il file JAR del codice dell'applicazione

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il bucket che hai creato in precedenza per il codice dell'applicazione.
3. Scegli Carica.
4. Scegliere Add files (Aggiungi file).
5. Passa al file.zip generato nel passaggio precedente:target/managed-flink-pyflink-getting-started-1.0.0.zip.
6. Scegli Carica senza modificare altre impostazioni.

Creazione e configurazione dell'applicazione del servizio gestito per Apache Flink

È possibile creare e configurare un'applicazione del servizio gestito per Apache Flink utilizzando la console o la AWS CLI Per questo tutorial, utilizzeremo la console.

Creazione dell'applicazione

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Verifica che sia selezionata la regione corretta: Stati Uniti orientali (Virginia settentrionale) us-east-1.
3. Apri il menu a destra e scegli Applicazioni Apache Flink, quindi Crea applicazione di streaming. In alternativa, scegli Crea applicazione di streaming dalla sezione Guida introduttiva della pagina iniziale.
4. Nella pagina Crea applicazioni di streaming:
 - Per Scegli un metodo per configurare l'applicazione di elaborazione dello stream, scegli Crea da zero.
 - Per la configurazione di Apache Flink, versione Application Flink, scegli Apache Flink 1.19.

- Per la configurazione dell'applicazione:
 - Per Nome applicazione, immetti **MyApplication**.
 - Per Descrizione, inserisci **My Python test app**.
 - In Accesso alle risorse dell'applicazione, scegli Crea/aggiorna il ruolo IAM kinesis-analytics-MyApplication-us-east-1 con le politiche richieste.
- Per le impostazioni di Template for applications:
 - Per Modelli, scegli Sviluppo.
- Scegli Crea applicazione di streaming.

Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: kinesis-analytics-service-*MyApplication-us-west-2*
- Ruolo: kinesisanalytics-*MyApplication-us-west-2*

Il servizio gestito da Amazon per Apache Flink era precedentemente noto come Analisi dei dati Amazon per Apache Flink era precedentemente noto come Analisi dei dati Amazon. Il nome delle risorse generate automaticamente ha il prefisso per garantire la compatibilità con le versioni precedenti. kinesis-analytics

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere al bucket Amazon S3.

Modifica della policy IAM per aggiungere le autorizzazioni per i bucket S3

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-east-1** creata dalla console nella sezione precedente.
3. Scegli Modifica, quindi scegli la scheda JSON.

4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (**012345678901**) con il tuo ID account.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadCode",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion"
      ],
      "Resource": [
        "arn:aws:s3::my-bucket/kinesis-analytics-placeholder-s3-object"
      ]
    },
    {
      "Sid": "ListCloudwatchLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:012345678901:log-group:*"
      ]
    },
    {
      "Sid": "ListCloudwatchLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
      ]
    },
    {
      "Sid": "PutCloudwatchLogs",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ]
    }
  ]
}
```

```

    ],
    "Resource": [
        "arn:aws:logs:us-east-1:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleInputStream"
  },
  {
    "Sid": "WriteOutputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleOutputStream"
  }
]
}

```

5. Scegli Avanti e quindi seleziona Salva modifiche.

Configurazione dell'applicazione

Modifica la configurazione dell'applicazione per impostare l'elemento del codice dell'applicazione.

Per configurare l'applicazione

1. Nella MyApplicationpagina, scegli Configura.
2. Nella sezione Posizione del codice dell'applicazione:
 - Per il bucket Amazon S3, seleziona il bucket creato in precedenza per il codice dell'applicazione. Scegli Sfoglia e seleziona il bucket corretto, quindi scegli Scegli. Non selezionare il nome del bucket.
 - Per Percorso dell'oggetto Amazon S3, inserisci **managed-flink-pyflink-getting-started-1.0.0.zip**
3. Per le autorizzazioni di accesso, scegli Crea/aggiorna il ruolo IAM **kinesis-analytics-MyApplication-us-east-1** con le politiche richieste.

4. Passa alle proprietà di Runtime e mantieni i valori predefiniti per tutte le altre impostazioni.
5. Scegliete Aggiungi nuovo elemento e aggiungete ciascuno dei seguenti parametri:

ID gruppo	Chiave	Valore
InputStream0	stream.name	ExampleInputStream
InputStream0	flink.stream.initpos	LATEST
InputStream0	aws.region	us-east-1
OutputStream0	stream.name	ExampleOutputStream
OutputStream0	aws.region	us-east-1
kinesis.analytics.flink.run.options	python	main.py
kinesis.analytics.flink.run.options	jarfile	lib/pyflink-dependencies.jar

6. Non modificare nessuna delle altre sezioni e scegli Salva modifiche.

Note

Quando scegli di abilitare la CloudWatch registrazione Amazon, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: /aws/kinesis-analytics/MyApplication
- Flusso di log: kinesis-analytics-log-stream

Esecuzione dell'applicazione.

L'applicazione è ora configurata e pronta all'esecuzione.

Per eseguire l'applicazione

1. Sulla console per Amazon Managed Service for Apache Flink, scegli La mia applicazione e scegli Esegui.
2. Nella pagina successiva, nella pagina di configurazione del ripristino dell'applicazione, scegli Esegui con l'ultima istantanea, quindi scegli Esegui.

Lo stato nell'applicazione descrive in dettaglio le transizioni da Ready Starting e poi a Running quando l'applicazione è stata avviata.

Quando l'applicazione è nello Running stato, ora puoi aprire la dashboard di Flink.

Per aprire il pannello di controllo

1. Scegli Apri la dashboard di Apache Flink. La dashboard si apre in una nuova pagina.
2. Nell'elenco dei lavori in esecuzione, scegli il singolo lavoro che puoi vedere.

Note

Se imposti le proprietà di Runtime o modifichi le policy IAM in modo errato, lo stato dell'applicazione potrebbe trasformarsi inRunning, ma la dashboard di Flink mostra che il job viene riavviato continuamente. Si tratta di uno scenario di errore comune se l'applicazione non è configurata correttamente o non dispone delle autorizzazioni per accedere alle risorse esterne.

Quando ciò accade, controlla la scheda Eccezioni nella dashboard di Flink per vedere la causa del problema.

Osserva le metriche dell'applicazione in esecuzione

Nella MyApplicationpagina, nella sezione Amazon CloudWatch metrics, puoi vedere alcune delle metriche fondamentali dell'applicazione in esecuzione.

Per visualizzare i parametri

1. Accanto al pulsante Aggiorna, seleziona 10 secondi dall'elenco a discesa.
2. Quando l'applicazione è in esecuzione ed è integra, puoi vedere la metrica di uptime aumentare continuamente.

3. La metrica `fullrestarts` deve essere zero. Se è in aumento, la configurazione potrebbe presentare dei problemi. Per esaminare il problema, consulta la scheda Eccezioni nella dashboard di Flink.
4. La metrica del numero di checkpoint non riusciti deve essere pari a zero in un'applicazione integra.

Note

Questa dashboard mostra un set fisso di metriche con una granularità di 5 minuti. Puoi creare una dashboard applicativa personalizzata con qualsiasi metrica nella dashboard. CloudWatch

Osserva i dati di output nei flussi Kinesis

Assicurati di continuare a pubblicare i dati sull'input, usando lo script Python o il Kinesis Data Generator.

È ora possibile osservare l'output dell'applicazione in esecuzione su Managed Service for Apache Flink utilizzando il Data Viewer in <https://console.aws.amazon.com/kinesis/>, analogamente a quanto già fatto in precedenza.

Per visualizzare l'output

1. [Apri la console Kinesis all'indirizzo https://console.aws.amazon.com/kinesis/](https://console.aws.amazon.com/kinesis/).
2. Verifica che la regione sia la stessa che stai utilizzando per eseguire questo tutorial. Per impostazione predefinita, è USEast-1US East (N. Virginia). Modificare la regione se necessario.
3. Scegli Data Streams.
4. Seleziona lo stream che desideri osservare. Ai fini di questo tutorial, utilizza `ExampleOutputStream`.
5. Scegli la scheda Data viewer.
6. Seleziona uno Shard, mantieni Ultimo come posizione iniziale, quindi scegli Ottieni record. Potresti visualizzare l'errore «nessun record trovato per questa richiesta». In tal caso, scegli Riprova a recuperare i record. Vengono visualizzati i record più recenti pubblicati nello stream.
7. Seleziona il valore nella colonna Dati per esaminare il contenuto del record in formato JSON.

Arresta l'applicazione

Per interrompere l'applicazione, vai alla pagina della console dell'applicazione Managed Service for Apache Flink denominata. `MyApplication`

Per interrompere l'applicazione

1. Dall'elenco a discesa Azione, scegli Stop.
2. Lo stato nell'applicazione descrive in dettaglio le transizioni da Running e quindi a Ready quando l'applicazione viene completamente interrotta. Stopping

Note

Non dimenticare di interrompere anche l'invio di dati al flusso di input dallo script Python o dal Kinesis Data Generator.

Approfondimenti

[Pulizia AWS delle risorse](#)

Pulizia AWS delle risorse

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Nozioni di base (Python).

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione dei flussi di dati Kinesis](#)
- [Eliminazione di oggetti e bucket Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

Per eliminare l'applicazione, utilizza la procedura seguente.

Per eliminare l'applicazione

1. [Apri la console Kinesis all'indirizzo https://console.aws.amazon.com /kines.](https://console.aws.amazon.com/kines.)
2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Dall'elenco a discesa Azioni, scegliete Elimina e confermate l'eliminazione.

Eliminazione dei flussi di dati Kinesis

1. Apri la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Scegli Flussi di dati.
3. Seleziona i due stream che hai creato e. ExampleInputStream ExampleOutputStream
4. Dall'elenco a discesa Azioni, scegli Elimina, quindi conferma l'eliminazione.

Eliminazione di oggetti e bucket Amazon S3

Per eliminare gli oggetti e il bucket S3, utilizza la procedura seguente.

Per eliminare l'oggetto dal bucket S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Seleziona il bucket S3 che hai creato per l'artefatto dell'applicazione.
3. Seleziona l'elemento dell'applicazione che hai caricato, denominato. amazon-msf-java-stream-app-1.0.jar
4. Scegli Delete (Elimina) e conferma l'eliminazione.

Per eliminare il bucket S3

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Seleziona il bucket che hai creato per gli artefatti.
3. Scegli Delete (Elimina) e conferma l'eliminazione.

Note

Il bucket S3 deve essere vuoto per eliminarlo.

Eliminazione delle risorse IAM

Per eliminare le risorse IAM, segui la procedura riportata di seguito.

Per eliminare le risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-east-1.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytics- MyApplication -us-east-1.
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

Per eliminare le CloudWatch risorse, segui la procedura seguente.

Per eliminare le tue CloudWatch risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Quindi scegli Elimina gruppo di log e conferma l'eliminazione.

Nozioni di base (Scala)

Note

A partire dalla versione 1.15, Flink non supporta più Scala. Le applicazioni possono ora utilizzare l'API Java da qualsiasi versione di Scala. Flink utilizza ancora Scala internamente in alcuni componenti chiave, ma non espone Scala nel classloader del codice utente. Per questo motivo, devi aggiungere le dipendenze di Scala nei tuoi archivi JAR.

Per ulteriori informazioni sulle modifiche a Scala in Flink 1.15, consulta [Scala non più disponibile nella versione 1.15](#).

In questo esercizio, viene creata un'applicazione del servizio gestito per Apache Flink per Scala con un flusso Kinesis come origine e come sink.

Questo argomento contiene le sezioni seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrivi record di esempio nel flusso di input](#)
- [Scarica ed esamina il codice dell'applicazione](#)
- [Per creare e compilare il codice dell'applicazione](#)
- [Creazione ed esecuzione dell'applicazione \(console\)](#)
- [Creazione ed esecuzione dell'applicazione \(CLI\)](#)
- [Eliminazione AWS delle risorse](#)

Creazione delle risorse dipendenti

Prima di creare un'applicazione del servizio gestito per Apache Flink per questo esercizio, è necessario creare le seguenti risorse dipendenti:

- Due flussi Kinesis per l'input e l'output.
- Un bucket Amazon S3 per archiviare il codice dell'applicazione (ka-app-code-*<username>*)

Puoi creare i flussi Kinesis e un bucket S3 utilizzando la console. Per istruzioni sulla creazione di queste risorse, consulta i seguenti argomenti:

- [Creazione e aggiornamento dei flussi di dati](#) nella Guida per gli sviluppatori del flusso di dati Amazon Kinesis. Assegna un nome ai flussi di dati **ExampleInputStream** e **ExampleOutputStream**.

Per creare i flussi di dati (AWS CLI)

- Per creare il primo flusso (ExampleInputStream), utilizza il seguente comando della di Amazon Kinesis per creare AWS CLI il primo flusso.

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

- Per creare il secondo flusso utilizzato dall'applicazione per scrivere l'output, esegui lo stesso comando, modificando il nome del flusso in ExampleOutputStream.

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

- Consulta [Come si crea un bucket S3?](#) nella Guida per l'utente di Amazon Simple Storage Service. Assegna al bucket Amazon S3 un nome univoco globale aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

Altre risorse

Quando crei la tua applicazione, il servizio gestito per Apache Flink crea le seguenti CloudWatch risorse Amazon, se non esistono già:

- Un gruppo di log denominato /AWS/KinesisAnalytics-java/MyApplication
- Un flusso di log denominato kinesis-analytics-log-stream

Scrivi record di esempio nel flusso di input

In questa sezione, viene utilizzato uno script Python per scrivere record di esempio nel flusso per l'applicazione da elaborare.

Note

Questa sezione richiede [AWS SDK for Python \(Boto\)](#).

Note

Lo script Python in questa sezione utilizza la AWS CLI. È necessario configurare AWS CLI per utilizzare le credenziali dell'account e la regione predefinita. Per configurare la AWS CLI, immetti quanto segue:

```
aws configure
```

1. Crea un file denominato `stock.py` con i seguenti contenuti:

```
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. Esegui lo script `stock.py`:

```
$ python stock.py
```

Mantieni lo script in esecuzione mentre completi il resto del tutorial.

Scarica ed esamina il codice dell'applicazione

Il codice dell'applicazione Python per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito.

1. Installa il client Git se non lo hai già fatto. Per ulteriori informazioni, consulta [Installazione di Git](#).
2. Clona il repository remoto con il comando seguente:

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

3. Passa alla directory `amazon-kinesis-data-analytics-java-examples/scala/GettingStarted`.

Tieni presente quanto segue riguardo al codice dell'applicazione:

- Un file `build.sbt` contiene le informazioni sulla configurazione e le dipendenze dell'applicazione, incluse le librerie del servizio gestito per Apache Flink.
- Il file `BasicStreamingJob.scala` contiene il metodo principale che definisce la funzionalità dell'applicazione.
- L'applicazione utilizza un'origine Kinesis per leggere dal flusso di origine. Il seguente snippet crea l'origine Kinesis:

```
private def createSource: FlinkKinesisConsumer[String] = {  
    val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties  
    val inputProperties = applicationProperties.get("ConsumerConfigProperties")  
  
    new FlinkKinesisConsumer[String](inputProperties.getProperty(streamNameKey,  
        defaultInputStreamName),  
        new SimpleStringSchema, inputProperties)
```



```
}
```

L'applicazione utilizza anche un sink Kinesis per scrivere nel flusso dei risultati. Il seguente snippet crea il sink Kinesis:

```
private def createSink: KinesisStreamsSink[String] = {  
  val applicationProperties = KinesisAnalyticsRuntime.getApplicationProperties  
  val outputProperties = applicationProperties.get("ProducerConfigProperties")  
  
  KinesisStreamsSink.builder[String]  
    .setKinesisClientProperties(outputProperties)  
    .setSerializationSchema(new SimpleStringSchema)  
    .setStreamName(outputProperties.getProperty(streamNameKey,  
defaultOutputStreamName))  
    .setPartitionKeyGenerator((element: String) => String.valueOf(element.hashCode))  
    .build  
}
```

- L'applicazione crea connettori di origine e sink per accedere alle risorse esterne utilizzando un `StreamExecutionEnvironment` oggetto.
- L'applicazione crea connettori di origine e sink utilizzando proprietà dinamiche. Questi metodi leggono le proprietà dell'applicazione di runtime per configurare il connettori. Per ulteriori informazioni sulle proprietà di runtime, consulta [Proprietà di runtime](#).

Per creare e compilare il codice dell'applicazione

In questa sezione, viene compilato e caricato il codice dell'applicazione nel bucket Amazon S3 creato nella sezione [Creazione delle risorse dipendenti](#).

Compilazione del codice dell'applicazione

In questa sezione, userai lo strumento di compilazione [SBT](#) per creare il codice Scala per l'applicazione. Per installare SBT, consulta [Installazione di sbt con setup cs](#). Dovrai inoltre installare il Java Development Kit (JDK). Consulta [Prerequisiti per il completamento degli esercizi](#).

1. Per usare il codice dell'applicazione, compila il codice e comprimilo in un file JAR. Puoi compilare e comprimere il codice con SBT:

```
sbt assembly
```

2. Se l'applicazione viene compilata correttamente, viene creato il seguente file:

```
target/scala-3.2.0/getting-started-scala-1.0.jar
```

Caricamento del codice Scala di streaming di Apache Flink

In questa sezione, viene creato un bucket Amazon S3 e caricato il codice dell'applicazione.

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Seleziona Crea bucket.
3. Immetti ka-app-code-<username> nel campo Nome bucket. Aggiungi un suffisso al nome del bucket, ad esempio il nome utente, per renderlo globalmente univoco. Scegli Next (Successivo).
4. Nella fase Configura opzioni, non modificare le impostazioni e scegli Successivo.
5. Nella fase Imposta autorizzazioni, non modificare le impostazioni e scegli Successivo.
6. Seleziona Crea bucket.
7. Scegli il bucket ka-app-code-<username>, quindi scegli Carica.
8. Nella fase Seleziona file, scegli Aggiungi file. Individua il file getting-started-scala-1.0.jar creato nella fase precedente.
9. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione ed esecuzione dell'applicazione (console)

Segui questi passaggi per creare, configurare, aggiornare ed eseguire l'applicazione utilizzando la console.

Creazione dell'applicazione

1. Aprire la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.
3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:

- Per Nome applicazione, immetti **MyApplication**.
 - Per Descrizione, inserisci **My scala test app**.
 - Per Runtime, scegli Apache Flink.
 - Mantieni la versione di Apache Flink versione 1.19.1.
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
 5. Scegli Crea applicazione.

Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`
- Ruolo: `kinesisanalytics-MyApplication-us-west-2`

Configurazione dell'applicazione

Per configurare l'applicazione, utilizza la procedura seguente.

Per configurare l'applicazione

1. Nella MyApplicationpagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:
 - Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **getting-started-scala-1.0.jar**.
3. In Accesso alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
4. In Proprietà, scegli Aggiungi gruppo.
5. Immetti i seguenti dati:

ID gruppo	Chiave	Valore
ConsumerConfigProperties	input.stream.name	ExampleInputStream
ConsumerConfigProperties	aws.region	us-west-2
ConsumerConfigProperties	flink.stream.initpos	LATEST

Scegli Save (Salva).

- In Proprietà, scegli di nuovo Aggiungi gruppo.
- Immetti i seguenti dati:

ID gruppo	Chiave	Valore
ProducerConfigProperties	output.stream.name	ExampleOutputStream
ProducerConfigProperties	aws.region	us-west-2

- In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
- Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.
- Scegli Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione Amazon, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: `/aws/kinesis-analytics/MyApplication`

- Flusso di log: `kinesis-analytics-log-stream`

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere al bucket Amazon S3.

Modifica della policy IAM per aggiungere le autorizzazioni per i bucket S3

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
3. Nella pagina Riepilogo, scegli Modifica policy. Scegli la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account di esempio IDs (`012345678901`) con il tuo ID account.

```
{
  "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"
    }
]
}

```

Esecuzione dell'applicazione.

Il grafico del processo Flink può essere visualizzato eseguendo l'applicazione, aprendo il pannello di controllo di Apache Flink e scegliendo il processo Flink desiderato.

Interruzione dell'applicazione

Per interrompere l'applicazione, nella MyApplication pagina, scegli Stop. Conferma l'operazione.

Creazione ed esecuzione dell'applicazione (CLI)

In questa sezione, verrà utilizzata la AWS Command Line Interface per creare ed eseguire il servizio gestito per Apache Flink. Utilizza il AWS CLI comando della kinesisanalyticsv2 per creare e interagire con il servizio gestito per Apache Flink.

Creazione di una policy di autorizzazione

Note

È necessario creare una policy di autorizzazione e un ruolo per l'applicazione. Se non crei queste risorse IAM, l'applicazione non può accedere ai suoi dati e flussi di log.

Innanzitutto, crea una policy di autorizzazione con due istruzioni: una che concede le autorizzazioni per l'operazione di lettura sul flusso di origine e un'altra che concede le autorizzazioni per operazioni di scrittura sul flusso di sink. Collega quindi la policy a un ruolo IAM (che verrà creato nella sezione successiva). Pertanto, quando il servizio gestito per Apache Flink assume il ruolo, il servizio disporrà delle autorizzazioni necessarie per leggere dal flusso di origine e scrivere nel flusso di sink.

Utilizza il codice seguente per creare la policy di autorizzazione `AKReadSourceStreamWriteSinkStream`. Sostituisci **username** con il nome utente utilizzato per creare il bucket Amazon S3 per archiviare il codice dell'applicazione. Sostituisci l'ID account nei nomi della risorsa Amazon (ARNs) (**012345678901**) con il tuo ID account.

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

Per step-by-step istruzioni su come creare una politica di autorizzazioni, consulta il [Tutorial: Create and Attach Your First Customer Managed Policy](#) nella IAM User Guide.

Creazione di una policy IAM

In questa sezione, viene creato un ruolo IAM per l'applicazione del servizio gestito per Apache Flink che può essere assunto per leggere un flusso di origine e scrivere nel flusso di sink.

Il servizio gestito per Apache Flink non può accedere al tuo flusso senza autorizzazioni. Queste autorizzazioni possono essere assegnate con un ruolo IAM. Ad ogni ruolo IAM sono collegate due policy. La policy di attendibilità concede al servizio gestito per Apache Flink l'autorizzazione per assumere il ruolo e la policy di autorizzazione determina cosa può fare il servizio assumendo questo ruolo.

Collega la policy di autorizzazione creata nella sezione precedente a questo ruolo.

Per creare un ruolo IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nel riquadro di navigazione, scegli Ruoli e quindi Crea ruolo.
3. In Seleziona tipo di entità attendibile, scegli Servizio AWS .
4. In Scegli il servizio che utilizzerà questo ruolo, scegli Kinesis.
5. In Seleziona il tuo caso d'uso, scegli Servizio gestito per Apache Flink.
6. Scegli Successivo: autorizzazioni.
7. Nella pagina Allega policy di autorizzazione, seleziona Successivo: esamina. Collega le policy di autorizzazione dopo aver creato il ruolo.
8. Nella pagina Crea ruolo, immetti **MF-stream-rw-role** per Nome ruolo. Scegli Crea ruolo.

È stato creato un nuovo ruolo IAM denominato MF-stream-rw-role. Successivamente, aggiorna le policy di trust e di autorizzazione per il ruolo

9. Collega la policy di autorizzazione al ruolo.

Note

Per questo esercizio, il servizio gestito per Apache Flink assume questo ruolo per la lettura di dati da un flusso di dati Kinesis (origine) e la scrittura dell'output in un altro flusso di dati Kinesis. Pertanto, devi collegare la policy creata nella fase precedente, [Creazione di una policy di autorizzazione](#).

- Nella pagina Riepilogo, scegli la scheda Autorizzazioni.
- Scegliere Collega policy.
- Nella casella di ricerca, immetti **AKReadSourceStreamWriteSinkStream** (la policy creata nella sezione precedente).
- Scegli la policy AKReadSourceStreamWriteSinkStream e seleziona Collega policy.

È stato creato il ruolo di esecuzione del servizio che l'applicazione utilizzerà per accedere alle risorse. Prendi nota dell'ARN del nuovo ruolo.

Per step-by-step istruzioni sulla creazione di un ruolo, consulta [Creating an IAM Role \(Console\)](#) nella IAM User Guide.

Creazione dell'applicazione

Salvare il seguente codice JSON in un file denominato `create_request.json`. Sostituisci l'ARN del ruolo di esempio con l'ARN per il ruolo creato in precedenza. Sostituisci il suffisso dell'ARN del bucket (username) con il suffisso scelto nella sezione precedente. Sostituisci l>ID account di esempio (012345678901) nel ruolo di esecuzione del servizio con l>ID account.

```
{
  "ApplicationName": "getting_started",
  "ApplicationDescription": "Scala getting started application",
  "RuntimeEnvironment": "FLINK-1_19",
  "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"
  }
]
}

```

Esegui [CreateApplication](#) con la seguente richiesta per creare l'applicazione:

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

L'applicazione è ora creata. Avvia l'applicazione nella fase successiva.

Avvia l'applicazione

In questa sezione, viene utilizzata l'operazione [StartApplication](#) per avviare l'applicazione.

Per avviare l'applicazione

1. Salvare il seguente codice JSON in un file denominato `start_request.json`.

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

2. Esegui l'operazione `StartApplication` con la richiesta precedente per avviare l'applicazione:

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

L'applicazione è ora in esecuzione. Puoi controllare i parametri del servizio gestito per Apache Flink sulla CloudWatch console Amazon per verificare che l'applicazione funzioni.

Interruzione dell'applicazione

In questa sezione, viene utilizzata l'operazione [StopApplication](#) per interrompere l'applicazione.

Per interrompere l'applicazione

1. Salvare il seguente codice JSON in un file denominato `stop_request.json`.

```
{
  "ApplicationName": "s3_sink"
}
```

2. Esegui l'operazione `StopApplication` con la richiesta precedente per interrompere l'applicazione:

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

L'applicazione è ora interrotta.

Aggiungi un'opzione CloudWatch di registrazione

Puoi utilizzare la AWS CLI per aggiungere un flusso di CloudWatch log Amazon all'applicazione. Per ulteriori informazioni sull'utilizzo CloudWatch dei log con l'applicazione, consulta [Configurazione della registrazione dell'applicazione](#).

Aggiornamento delle proprietà dell'ambiente

In questa sezione, viene utilizzata l'operazione [UpdateApplication](#) per modificare le proprietà di ambiente per l'applicazione senza ricompilare il codice dell'applicazione. In questo esempio, viene modificata la regione dei flussi di origine e destinazione.

Per aggiornare le proprietà di ambiente per l'applicazione

1. Salvare il seguente codice JSON in un file denominato `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. Esegui l'operazione `UpdateApplication` con la richiesta precedente per aggiornare le proprietà di ambiente:

```
aws kinesisanalyticstv2 update-application --cli-input-json file://  
update_properties_request.json
```

Aggiornamento del codice dell'applicazione

Quando devi aggiornare il codice dell'applicazione con una nuova versione del pacchetto di codice, utilizza l'operazione [UpdateApplication](#) CLI.

Note

Per caricare una nuova versione del codice dell'applicazione con lo stesso nome file, è necessario specificare la nuova versione dell'oggetto. Per ulteriori informazioni sull'uso delle versioni degli oggetti Amazon S3, consulta [Abilitazione o disattivazione del controllo delle versioni](#).

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione ed effettua la chiamata, specificando lo stesso bucket Amazon S3 e lo stesso nome dell'oggetto `UpdateApplication`, nonché la nuova versione dell'oggetto, nonché la nuova versione dell'oggetto. L'applicazione verrà riavviata con il nuovo pacchetto di codice.

Il seguente esempio di richiesta per l'operazione `UpdateApplication` ricarica il codice dell'applicazione e riavvia l'applicazione. Aggiorna `CurrentApplicationVersionId` alla versione corrente dell'applicazione. Puoi controllare la versione corrente dell'applicazione utilizzando le operazioni `ListApplications` o `DescribeApplication`. Aggiorna il suffisso del nome del bucket (`<username>`) con il suffisso che hai scelto nella sezione [Creazione delle risorse dipendenti](#).

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

Eliminazione AWS delle risorse

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Finestra a cascata.

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione dei flussi di dati Kinesis](#)
- [Eliminazione dell'oggetto Amazon S3 e del bucket](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Aprire la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e quindi conferma l'eliminazione.

Eliminazione dei flussi di dati Kinesis

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com /kines.](https://console.aws.amazon.com/kines)
2. Nel pannello del flusso di dati Kinesis, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.
4. Nella pagina Kinesis Streams, scegli, scegli Azioni ExampleOutputStream, scegli Elimina, quindi conferma l'eliminazione.

Eliminazione dell'oggetto Amazon S3 e del bucket

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il **<username>**secchio ka-app-code -.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

Eliminazione delle risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-west-2.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo della kinesis-analytics-us-west-2. MyApplication
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Quindi scegli Elimina gruppo di log e conferma l'eliminazione.

Utilizzo di Apache Beam con il servizio gestito per Apache Flink

Note

Non esiste Apache Flink Runner compatibile per Flink 1.20. Per ulteriori informazioni, consulta la [compatibilità delle versioni di Flink](#) nella documentazione di Apache Beam. >

È possibile utilizzare il framework [Apache Beam](#) con l'applicazione del servizio gestito per Apache Flink per elaborare i dati di streaming. Le applicazioni del servizio gestito per Apache Flink che utilizzano Apache Beam utilizzano il [runner Apache Flink](#) per eseguire le pipeline Beam.

Per un tutorial sull'utilizzo di Apache Beam in un'applicazione del servizio gestito per Apache Flink, consulta [Usa CloudFormation](#).

Questo argomento contiene le sezioni seguenti:

- [Limitazioni del runner Apache Flink con il servizio gestito per Apache Flink](#)
- [Funzionalità di Apache Beam con il servizio gestito per Apache Flink](#)
- [Creazione di un'applicazione utilizzando Apache Beam](#)

Limitazioni del runner Apache Flink con il servizio gestito per Apache Flink

Tieni presente quanto segue sull'utilizzo del runner Apache Flink con il servizio gestito per Apache Flink:

- I parametri di Apache Beam non sono visualizzabili nella console del servizio gestito per Apache Flink.
- Apache Beam è supportato solo nelle applicazioni del servizio gestito per Apache Flink che utilizzano Apache Flink versione 1.8 e successive. Apache Beam non è supportato nelle applicazioni del servizio gestito per Apache Flink che utilizzano Apache Flink versione 1.6.

Funzionalità di Apache Beam con il servizio gestito per Apache Flink

Il servizio gestito per Apache Flink supporta le stesse funzionalità di Apache Beam supportate dal runner Apache Flink. Per informazioni sulle funzionalità supportate dal runner Apache Flink, consulta la [Matrice di compatibilità Beam](#).

È consigliabile testare l'applicazione Apache Flink nel servizio gestito per Apache Flink per verificare che tutte le funzionalità di cui l'applicazione ha bisogno siano supportate.

Creazione di un'applicazione utilizzando Apache Beam

In questo esercizio, viene creata un'applicazione del servizio gestito per Apache Flink che trasforma i dati utilizzando [Apache Beam](#). Apache Beam è un modello di programmazione per l'elaborazione di dati di streaming. Per informazioni sull'utilizzo di Apache Beam con il servizio gestito per Apache Flink, consulta [Utilizzo di Apache Beam con il servizio gestito per Apache Flink](#).

Note

Per impostare i prerequisiti richiesti per questo esercizio, completa innanzitutto l'esercizio [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#).

Questo argomento contiene le sezioni seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrivi record di esempio nel flusso di input](#)
- [Scarica ed esamina il codice dell'applicazione](#)
- [Per creare il codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink](#)
- [Pulizia AWS delle risorse](#)
- [Passaggi successivi](#)

Creazione delle risorse dipendenti

Prima di creare un'applicazione del servizio gestito per Apache Flink per questo esercizio, è necessario creare le seguenti risorse dipendenti:

- Due flussi di dati Kinesis (`ExampleInputStream` e `ExampleOutputStream`)
- Un bucket Amazon S3 per archiviare il codice dell'applicazione (`ka-app-code-<username>`)

Puoi creare i flussi Kinesis e un bucket S3 utilizzando la console. Per istruzioni sulla creazione di queste risorse, consulta i seguenti argomenti:

- [Creazione e aggiornamento dei flussi di dati](#) nella Guida per gli sviluppatori del flusso di dati Amazon Kinesis. Assegna un nome ai flussi di dati **ExampleInputStream** e **ExampleOutputStream**.
- [Come si crea un bucket S3?](#) nella Guida per l'utente di Amazon Simple Storage Service. Assegna al bucket Amazon S3 un nome globalmente univoco aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

Scrivi record di esempio nel flusso di input

In questa sezione, viene utilizzato uno script Python per scrivere stringhe casuali nel flusso per l'applicazione da elaborare.

Note

Questa sezione richiede [AWS SDK for Python \(Boto\)](#).

1. Crea un file denominato `ping.py` con i seguenti contenuti:

```
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. Esegui lo script `ping.py`:

```
$ python ping.py
```

Mantieni lo script in esecuzione mentre completi il resto del tutorial.

Scarica ed esamina il codice dell'applicazione

Il codice dell'applicazione Java per questo esempio è disponibile da GitHub. Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito.

1. Installa il client Git se non lo hai già fatto. Per ulteriori informazioni, consulta [Installazione di Git](#).
2. Clona il repository remoto con il comando seguente:

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

3. Passa alla directory `amazon-kinesis-data-analytics-java-examples/Beam`.

Il codice dell'applicazione si trova nei file `BasicBeamStreamingJob.java`. Tieni presente quanto segue riguardo al codice dell'applicazione:

- L'applicazione utilizza Beam [ParDo](#) per elaborare i record in entrata richiamando una funzione di trasformazione personalizzata denominata `PingPongFn`

Il codice per richiamare la funzione `PingPongFn` è il seguente:

```
.apply("Pong transform",
    ParDo.of(new PingPongFn()))
```

- Le applicazioni del servizio gestito per Apache Flink che utilizzano Apache Beam richiedono i seguenti componenti. Se questi componenti e versioni non vengono inclusi in `pom.xml`, l'applicazione carica le versioni errate dalle dipendenze dell'ambiente e, poiché le versioni non corrispondono, si blocca in fase di runtime.

```
<jackson.version>2.10.2</jackson.version>
...
<dependency>
  <groupId>com.fasterxml.jackson.module</groupId>
  <artifactId>jackson-module-jaxb-annotations</artifactId>
  <version>2.10.2</version>
</dependency>
```

- La funzione di trasformazione PingPongFn passa i dati di input nel flusso di output, a meno che i dati di input non siano ping, nel qual caso nel flusso di output viene emessa la stringa pong\n.

Il codice della funzione di trasformazione è il seguente:

```
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());
    }
  }
}
```

Per creare il codice dell'applicazione

Per scaricare il codice dell'applicazione, esegui le operazioni descritte di seguito:

1. Installa Java e Maven se non lo hai già fatto. Per ulteriori informazioni, consulta [Completa i prerequisiti richiesti](#) nel tutorial [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#).
2. Compila l'applicazione con il seguente comando:

```
mvn package -Dflink.version=1.15.2 -Dflink.version.minor=1.8
```

Note

Il codice di origine fornito si basa sulle librerie di Java 11.

La compilazione dell'applicazione crea il file JAR dell'applicazione (`target/basic-beam-app-1.0.jar`).

Caricamento del codice Java di streaming di Apache Flink

In questa sezione, il codice dell'applicazione viene caricato nel bucket Amazon S3 creato nella sezione [Creazione delle risorse dipendenti](#).

1. Nella console Amazon S3, scegli il **<username>** bucket `ka-app-code-` e scegli Carica.
2. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `basic-beam-app-1.0.jar` creato nella fase precedente.
3. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Il codice dell'applicazione è ora archiviato in un bucket Amazon S3 accessibile dall'applicazione.

Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink

Segui questi passaggi per creare, configurare, aggiornare ed eseguire l'applicazione utilizzando la console.

Creazione dell'applicazione

1. Aprire la console del servizio gestito per Apache Flink all'indirizzo `/flink https://console.aws.amazon.com`
2. Nella dashboard del servizio gestito per Apache Flink, scegli Crea un'applicazione di analisi.
3. Nella pagina Servizio gestito per Apache Flink: crea applicazione, fornisci i dettagli dell'applicazione nel modo seguente:

- Per Nome applicazione, inserisci **MyApplication**.
- Per Runtime, scegli Apache Flink.

 Note

Apache Beam non è attualmente compatibile con Apache Flink versione 1.19 o successiva.

- Seleziona Apache Flink versione 1.15 dal menu a discesa della versione.
4. Per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
 5. Scegli Crea applicazione.

 Note

Quando crei un'applicazione del servizio gestito per Apache Flink tramite la console, hai la possibilità di avere un ruolo e una policy IAM creati per l'applicazione. L'applicazione utilizza questo ruolo e questa policy per accedere alle sue risorse dipendenti. Queste risorse IAM sono denominate utilizzando il nome dell'applicazione e la Regione come segue:

- Policy: `kinesis-analytics-service-MyApplication-us-west-2`
- Ruolo: `kinesis-analytics-MyApplication-us-west-2`

Modifica la policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere ai flussi di dati Kinesis.

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
3. Nella pagina Riepilogo, scegli Modifica policy. Scegli la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci l'account **012345678901** segnaposto IDs () con il tuo ID account.

```
{
```

```

"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"
    }
]
}

```

Configurazione dell'applicazione

1. Nella MyApplicationpagina, scegli Configura.
2. Nella pagina Configura applicazione, fornisci la Posizione del codice:
 - Per Bucket Amazon S3, inserisci **ka-app-code-*<username>***.
 - Per Percorso dell'oggetto Amazon S3, inserisci **basic-beam-app-1.0.jar**
3. In Accedi alle risorse dell'applicazione, per Autorizzazioni di accesso, scegli Crea/aggiorna **kinesis-analytics-MyApplication-us-west-2** per il ruolo IAM.
4. Inserisci i seguenti dati:

ID gruppo	Chiave	Valore
BeamApplicationProperties	InputStreamName	ExampleInputStream
BeamApplicationProperties	OutputStreamName	ExampleOutputStream
BeamApplicationProperties	AwsRegion	us-west-2

5. In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
6. Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.

7. Scegli Aggiorna.

Note

Quando scegli di abilitare la CloudWatch registrazione, il servizio gestito per Apache Flink crea automaticamente un gruppo di log e un flusso di log. I nomi di tali risorse sono i seguenti:

- Gruppo di log: /aws/kinesis-analytics/MyApplication
- Flusso di log: kinesis-analytics-log-stream

Questo flusso di log viene utilizzato per monitorare l'applicazione. Non si tratta dello stesso flusso di log utilizzato dall'applicazione per inviare i risultati.

Esecuzione dell'applicazione.

Il grafico del processo Flink può essere visualizzato eseguendo l'applicazione, aprendo il pannello di controllo di Apache Flink e scegliendo il processo Flink desiderato.

Puoi controllare i parametri del servizio gestito per Apache Flink sulla CloudWatch console per verificare che l'applicazione funzioni.

Pulizia AWS delle risorse

Questa sezione include le procedure per l'eliminazione AWS delle risorse create nel tutorial Finestra a cascata.

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione dei flussi di dati Kinesis](#)
- [Eliminazione di un oggetto Amazon S3 e un bucket](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)

Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Aprire la console del servizio gestito per Apache Flink all'indirizzo /flink <https://console.aws.amazon.com>
2. Nel pannello del servizio gestito per Apache Flink, scegli. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e quindi conferma l'eliminazione.

Eliminazione dei flussi di dati Kinesis

1. [Aprire la console Kinesis all'indirizzo https://console.aws.amazon.com/kinesis.](https://console.aws.amazon.com/kinesis)
2. Nel pannello del flusso di dati Kinesis Data Streams, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.
4. Nella pagina Kinesis Streams, scegli, scegli Azioni ExampleOutputStream, scegli Elimina, quindi conferma l'eliminazione.

Eliminazione di un oggetto Amazon S3 e un bucket

1. Apri la console Amazon S3 all'indirizzo. <https://console.aws.amazon.com/s3/>
2. Scegli il **<username>**secchio ka-app-code -.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

Eliminazione delle risorse IAM

1. Aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Nella barra di navigazione, scegli Policy.
3. Nel controllo filtro, inserisci kinesis.
4. Scegli la politica kinesis-analytics-service- MyApplication -us-west-2.
5. Seleziona Operazioni di policy e quindi Elimina.
6. Nella barra di navigazione, scegli Ruoli.
7. Scegli il ruolo kinesis-analytics-us-west-2. MyApplication
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo/aws/kinesis-analytics/MyApplicationlog.
4. Quindi scegli Elimina gruppo di log e conferma l'eliminazione.

Passaggi successivi

Ora che hai creato ed eseguito un'applicazione di base del servizio gestito per Apache Flink che trasforma i dati utilizzando Apache Beam, consulta la seguente applicazione come esempio di una soluzione del servizio gestito per Apache Flink più avanzata.

- [Workshop in streaming su Beam nel servizio gestito per Apache Flink](#): in questo workshop, viene esplorato un esempio completo che combina aspetti di batch e streaming in un'unica pipeline Apache Beam uniforme.

Workshop di formazione, laboratori e implementazioni di soluzioni

end-to-end Gli esempi seguenti illustrano soluzioni avanzate di Managed Service per Apache Flink.

Argomenti

- [Distribuisci, gestisci e ridimensiona le applicazioni con Amazon Managed Service per Apache Flink](#)
- [Sviluppa le applicazioni Apache Flink localmente prima di distribuirle su Managed Service for Apache Flink](#)
- [Usa il rilevamento degli eventi con Managed Service per Apache Flink Studio](#)
- [Usa la soluzione di AWS streaming di dati per Amazon Kinesis](#)
- [Esercitati a usare un laboratorio Clickstream con Apache Flink e Apache Kafka](#)
- [Configurare il ridimensionamento personalizzato utilizzando Application Auto Scaling](#)
- [Visualizza un esempio di CloudWatch dashboard Amazon](#)
- [Usa modelli per la soluzione di AWS streaming di dati per Amazon MSK](#)
- [Esplora altre soluzioni Managed Service for Apache Flink su GitHub](#)

Distribuisci, gestisci e ridimensiona le applicazioni con Amazon Managed Service per Apache Flink

Questo workshop illustra lo sviluppo di un'applicazione Apache Flink in Java, come eseguire ed eseguire il debug nel tuo IDE e come impacchettare, distribuire ed eseguire su Amazon Managed Service per Apache Flink. Imparerai anche come ridimensionare, monitorare e risolvere i problemi della tua applicazione.

[Workshop Amazon Managed Service per Apache Flink.](#)

Sviluppa le applicazioni Apache Flink localmente prima di distribuirle su Managed Service for Apache Flink

Questo workshop illustra le basi per iniziare a sviluppare applicazioni Apache Flink a livello locale con l'obiettivo a lungo termine di implementarle su Managed Service for Apache Flink for Apache Flink.

[Guida introduttiva allo sviluppo locale con Apache Flink](#)

Usa il rilevamento degli eventi con Managed Service per Apache Flink Studio

Questo workshop descrive il rilevamento di eventi con il servizio gestito per Apache Flink Studio e la sua implementazione come servizio gestito per le applicazioni Apache Flink.

[Rilevamento degli eventi con servizio gestito per Apache Flink per Apache Flink](#)

Usa la soluzione di AWS streaming di dati per Amazon Kinesis

La soluzione AWS Streaming Data per Amazon Kinesis configura automaticamente i AWS servizi necessari per acquisire, archiviare, elaborare e distribuire dati in streaming. La soluzione offre diverse opzioni per risolvere i casi d'uso dei dati di streaming. L'opzione Managed Service for Apache Flink fornisce un esempio di end-to-end streaming ETL che dimostra un'applicazione reale che esegue operazioni analitiche su dati simulati dei taxi di New York.

Ogni soluzione include i seguenti componenti:

- Un AWS CloudFormation pacchetto per implementare l'esempio completo.
- Una CloudWatch dashboard per la visualizzazione delle metriche delle applicazioni.
- CloudWatch allarmi sulle metriche applicative più rilevanti.
- tutti i ruoli e le politiche IAM necessari.

[Soluzione di streaming di dati per Amazon Kinesis](#)

Esercitati a usare un laboratorio Clickstream con Apache Flink e Apache Kafka

Un laboratorio end-to-end per i casi d'uso clickstream che si avvale dello streaming gestito da Amazon per Apache Kafka per l'archiviazione in streaming e servizio gestito per Apache Flink per le applicazioni Apache Flink per l'elaborazione di flussi.

[Clickstream Lab](#)

Configurare il ridimensionamento personalizzato utilizzando Application Auto Scaling

Due esempi che mostrano come scalare automaticamente le applicazioni Managed Service for Apache Flink utilizzando Application Auto Scaling. Ciò consente di configurare politiche di ridimensionamento personalizzate e attributi di ridimensionamento personalizzati.

- [Servizio gestito per Apache Flink App Autoscaling](#)
- [Dimensionamento pianificato](#)

Per ulteriori informazioni su come eseguire il ridimensionamento personalizzato, consulta [Abilitare il ridimensionamento pianificato e basato su metriche per Amazon Managed Service for Apache Flink](#).

Visualizza un esempio di CloudWatch dashboard Amazon

Un CloudWatch pannello di controllo di esempio per il monitoraggio del servizio gestito per le applicazioni Apache Flink. La dashboard di esempio include anche un'[applicazione dimostrativa](#), per meglio visualizzare le funzionalità della dashboard.

[Servizio gestito per Apache Flink Metrics Dashboard](#)

Usa modelli per la soluzione di AWS streaming di dati per Amazon MSK

La soluzione AWS Streaming Data per Amazon MSK fornisce AWS CloudFormation modelli in cui i dati fluiscono attraverso produttori, storage in streaming, consumatori e destinazioni.

[AWS Soluzione di streaming di dati per Amazon MSK](#)

Esplora altre soluzioni Managed Service for Apache Flink su GitHub

end-to-endGli esempi seguenti illustrano soluzioni avanzate di Managed Service per Apache Flink e sono disponibili su: GitHub

- [Servizio gestito da Amazon per Apache Flink: utilità dell'analisi comparativa](#)

- [Snapshot Manager: servizio gestito da Amazon per Apache Flink](#)
- [Streaming ETL con Apache Flink e servizio gestito da Amazon per Apache Flink](#)
- [Analisi in tempo reale del sentiment relativo al feedback dei clienti](#)

Usa utilità pratiche per Managed Service for Apache Flink

Le seguenti utilità possono semplificare l'utilizzo del servizio gestito per Apache Flink:

Argomenti

- [Snapshot manager](#)
- [Analisi comparativa](#)

Snapshot manager

È buona prassi per le applicazioni Flink avviare regolarmente savepoint/snapshot per consentire un ripristino degli errori più semplice. Lo Snapshot manager automatizza questa attività e offre i seguenti vantaggi:

- scatta un nuovo snapshot del servizio gestito per Apache Flink in esecuzione per un'applicazione Apache Flink;
- conta il numero di snapshot dell'applicazione;
- controlla se il conteggio è superiore al numero richiesto di snapshot;
- elimina gli snapshot più vecchi e che superano il numero richiesto.

[Per un esempio, vedete Snapshot manager.](#)

Analisi comparativa

Il servizio gestito per Apache Flink di utilità per l'analisi comparativa Flink contribuisce alla pianificazione della capacità, ai test per l'integrazione e all'analisi comparativa del servizio gestito per Apache Flink per le applicazioni Apache Flink.

Puoi trovare un esempio consultando [Analisi comparativa](#)

Esempi di creazione e utilizzo di Managed Service per applicazioni Apache Flink

Questa sezione fornisce esempi di creazione e utilizzo di applicazioni nel servizio gestito per Apache Flink. Includono codice di esempio e step-by-step istruzioni per aiutarti a creare servizi gestiti per applicazioni Apache Flink e testare i risultati.

Prima di esplorare questi esempi, ti consigliamo di esaminare quanto segue:

- [Come funziona](#)
- [Tutorial: inizia a usare l' DataStream API in Managed Service for Apache Flink](#)

Note

Questi esempi presuppongono che si stia utilizzando la regione Stati Uniti orientali (Virginia settentrionale) (us-east-1). Se utilizzi una regione diversa, aggiorna il codice dell'applicazione, i comandi e i ruoli IAM in modo appropriato.

Argomenti

- [Esempi di Java per Managed Service for Apache Flink](#)
- [Esempi in Python per Managed Service for Apache Flink](#)
- [Esempi di Scala per Managed Service for Apache Flink](#)

Esempi di Java per Managed Service for Apache Flink

Gli esempi seguenti mostrano come creare applicazioni scritte in Java.

Note

La maggior parte degli esempi è progettata per essere eseguita sia localmente, sulla tua macchina di sviluppo e sul tuo IDE preferito, sia su Amazon Managed Service for Apache Flink. Dimostrano i meccanismi che è possibile utilizzare per passare i parametri

dell'applicazione e come impostare correttamente la dipendenza per eseguire l'applicazione in entrambi gli ambienti senza modifiche.

Migliora le prestazioni di serializzazione definendone la personalizzazione TypeInfo

Questo esempio illustra come definire qualcosa di personalizzato TypeInfo sul record o sull'oggetto di stato per evitare che la serializzazione ritorni alla serializzazione Kryo, meno efficiente. Ciò è necessario, ad esempio, quando gli oggetti contengono un `o. List Map` Per ulteriori informazioni, consulta [Tipi di dati e serializzazione nella documentazione](#) di Apache Flink. L'esempio mostra anche come verificare se la serializzazione dell'oggetto ricade sulla meno efficiente serializzazione di Kryo.

Esempio di codice: [CustomTypeInfo](#)

Inizia con l' DataStream API

Questo esempio mostra una semplice applicazione, che legge da un flusso di dati Kinesis e scrive su un altro flusso di dati Kinesis, utilizzando l'API. `DataStream` L'esempio dimostra come configurare il file con le dipendenze corrette, creare Uber-JAR e quindi analizzare i parametri di configurazione, in modo da poter eseguire l'applicazione sia localmente, nell'IDE e su Amazon Managed Service for Apache Flink.

Esempio di codice: [GettingStarted](#)

Inizia con l'API Table e SQL

Questo esempio mostra una semplice applicazione che utilizza l'`TableAPI` e SQL. Dimostra come integrare l'`DataStreamAPI` con l'`TableAPI` o SQL nella stessa applicazione Java. Dimostra inoltre come utilizzare il `DataGen` connettore per generare dati di test casuali dall'interno dell'applicazione Flink stessa, senza richiedere un generatore di dati esterno.

Esempio completo: [GettingStartedTable](#)

Usa S3Sink (API) DataStream

Questo esempio dimostra come utilizzare le `DataStream API` per scrivere file JSON `FileSink` in un bucket S3.

Esempio di codice: [S3Sink](#)

Usa una sorgente Kinesis, standard o consumatori EFO e sink (API) DataStream

Questo esempio dimostra come configurare una sorgente che utilizza un flusso di dati Kinesis, utilizzando lo standard consumer o EFO, e come configurare un sink per il flusso di dati Kinesis.

Esempio di codice: [KinesisConnectors](#)

Usa un sink Amazon Data Firehose (API) DataStream

Questo esempio mostra come inviare dati ad Amazon Data Firehose (precedentemente noto come Kinesis Data Firehose).

Esempio di codice: [KinesisFirehoseSink](#)

Usa il connettore per lavabo Prometheus

Questo esempio dimostra l'uso del connettore [sink Prometheus per scrivere dati di serie temporali su Prometheus](#).

Esempio di codice: [PrometheusSink](#)

Usa le aggregazioni di finestre (API) DataStream

Questo esempio illustra quattro tipi di aggregazione di finestre nell'API. DataStream

1. Finestra scorrevole basata sul tempo di elaborazione
2. Finestra scorrevole basata sull'ora dell'evento
3. Tumbling Window basata sul tempo di elaborazione
4. Tumbling Window basata sull'ora dell'evento

[Esempio di codice: Windowing](#)

Utilizzare parametri personalizzato

Questo esempio mostra come aggiungere metriche personalizzate all'applicazione Flink e inviarle alle metriche. CloudWatch

Esempio di codice: [CustomMetrics](#)

Usa Kafka Configuration Provider per recuperare keystore e truststore personalizzati per MTL in fase di esecuzione

Questo esempio illustra come è possibile utilizzare i provider di configurazione Kafka per configurare un keystore e un truststore personalizzati con certificati per l'autenticazione MTLS per il connettore Kafka. Questa tecnica consente di caricare i certificati personalizzati richiesti da Amazon S3 e i segreti dell' AWS Secrets Manager avvio dell'applicazione.

Esempio di codice: [Kafka-MTLS-KeyStore- ConfigProviders](#)

Usa Kafka Configuration Provider per recuperare i segreti per l'autenticazione SASL/SCRAM in fase di esecuzione

Questo esempio illustra come utilizzare i provider di configurazione Kafka per recuperare le credenziali AWS Secrets Manager e scaricare il truststore da Amazon S3 per configurare l'autenticazione SASL/SCRAM su un connettore Kafka. Questa tecnica consente di caricare i certificati personalizzati richiesti da Amazon S3 e i segreti dell' AWS Secrets Manager avvio dell'applicazione.

Esempio di codice: [Kafka- - SASL_SSL ConfigProviders](#)

Usa i provider di configurazione Kafka per recuperare keystore e truststore personalizzati per MTL in fase di esecuzione con Table API/SQL

Questo esempio illustra come è possibile utilizzare i provider di configurazione Kafka nella tabella API /SQL per configurare un keystore e un truststore personalizzati con certificati per l'autenticazione MTLS per il connettore Kafka. Questa tecnica consente di caricare i certificati personalizzati richiesti da Amazon S3 e i segreti dell' AWS Secrets Manager avvio dell'applicazione.

Esempio di codice: [Kafka-MTLS-KeyStore-SQL](#) - ConfigProviders

Usa le uscite laterali per dividere uno stream

Questo esempio illustra come sfruttare le [uscite laterali](#) in Apache Flink per suddividere uno stream su attributi specifici. Questo modello è particolarmente utile quando si cerca di implementare il concetto di Dead Letter Queues (DLQ) nelle applicazioni di streaming.

Esempio di codice: [SideOutputs](#)

Usa l'I/O asincrono per chiamare un endpoint esterno

Questo esempio illustra come utilizzare [Apache Flink Async I/O](#) per chiamare un endpoint esterno in modo non bloccante, con nuovi tentativi sugli errori recuperabili.

[Esempio](#) di codice: AsyncIO

Esempi in Python per Managed Service for Apache Flink

I seguenti esempi mostrano come creare applicazioni scritte in Python.

Note

La maggior parte degli esempi è progettata per essere eseguita sia localmente, sulla tua macchina di sviluppo e sul tuo IDE preferito, sia su Amazon Managed Service for Apache Flink. Dimostrano il semplice meccanismo che è possibile utilizzare per passare i parametri dell'applicazione e come impostare correttamente la dipendenza per eseguire l'applicazione in entrambi gli ambienti senza modifiche.

Dipendenze del progetto

La maggior parte PyFlink degli esempi richiede una o più dipendenze come file JAR, ad esempio per i connettori Flink. Queste dipendenze devono quindi essere impacchettate con l'applicazione quando vengono distribuite su Amazon Managed Service for Apache Flink.

Gli esempi seguenti includono già gli strumenti che consentono di eseguire l'applicazione localmente per lo sviluppo e il test e di impacchettare correttamente le dipendenze richieste. Questo strumento richiede l'utilizzo di Java JDK11 e Apache Maven. Fate riferimento al README contenuto in ogni esempio per le istruzioni specifiche.

Examples (Esempi)

Inizia con PyFlink

Questo esempio dimostra la struttura di base di un' PyFlink applicazione che utilizza SQL incorporato nel codice Python. Questo progetto fornisce anche uno scheletro per qualsiasi PyFlink applicazione che includa dipendenze JAR come i connettori. La sezione README fornisce una guida dettagliata

su come eseguire l'applicazione Python localmente per lo sviluppo. L'esempio mostra anche come includere una singola dipendenza JAR, il connettore Kinesis SQL in questo esempio, nell'PyFlinkapplicazione.

Esempio di codice: [GettingStarted](#)

Aggiungi dipendenze Python

Questo esempio mostra come aggiungere dipendenze Python all' PyFlink applicazione nel modo più generale. Questo metodo funziona per dipendenze semplici, come Boto3, o dipendenze complesse contenenti librerie C come. PyArrow

Esempio di codice: [PythonDependencies](#)

Usa le aggregazioni di finestre (API) DataStream

Questo esempio dimostra quattro tipi di aggregazione di finestre in SQL incorporati in un'applicazione Python.

1. Sliding Window in base al tempo di elaborazione
2. Finestra scorrevole basata sull'ora dell'evento
3. Tumbling Window basata sul tempo di elaborazione
4. Tumbling Window basata sull'ora dell'evento

[Esempio di codice: Windowing](#)

Usa un lavandino S3

Questo esempio mostra come scrivere l'output su Amazon S3 come file JSON, utilizzando SQL incorporato in un'applicazione Python. È necessario abilitare il checkpoint per il sink S3 per scrivere e ruotare i file su Amazon S3.

[Esempio di codice: S3Sink](#)

Usa una funzione definita dall'utente (UDF)

Questo esempio dimostra come definire una funzione definita dall'utente, implementarla in Python e utilizzarla nel codice SQL eseguito in un'applicazione Python.

[Esempio di codice: UDF](#)

Usa un sink Amazon Data Firehose

Questo esempio dimostra come inviare dati ad Amazon Data Firehose utilizzando SQL.

Esempio di codice: [FirehoseSink](#)

Esempi di Scala per Managed Service for Apache Flink

Negli esempi seguenti viene illustrato come creare applicazioni utilizzando Scala con Apache Flink.

Configura un'applicazione in più fasi

Questo esempio mostra come configurare un'applicazione Flink in Scala. Dimostra come configurare il progetto SBT per includere dipendenze e creare Uber-JAR.

Esempio di codice: [GettingStarted](#)

Utilizzo del servizio gestito da Amazon per Apache Flink

LaPer, la sicurezza del cloud ha la massima priorità. AWS In quanto AWS cliente, puoi trarre vantaggio da un'architettura di data center e di rete progettata per soddisfare i requisiti delle aziende più esigenti a livello di sicurezza.

La sicurezza è una responsabilità condivisa tra AWS e l'utente. Il [modello di responsabilità condivisa](#) descrive questo come sicurezza del cloud e sicurezza nel cloud:

- Sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che esegue AWS i servizi in AWS Cloud. AWS fornisce, inoltre, servizi che puoi utilizzare in modo sicuro. L'efficacia della nostra sicurezza è regolarmente testata e verificata da revisori di terze parti come parte dei [programmi di conformitàAWS](#). Per ulteriori informazioni sui programmi di conformità che si applicano al servizio gestito per Apache Flink, consulta [Servizi AWS coperti dal programma di conformità](#).
- Sicurezza nel cloud: la tua responsabilità è determinata dal AWS servizio che utilizzi. L'utente è anche responsabile per altri fattori, tra cui la riservatezza dei dati, i requisiti dell'azienda e leggi e normative applicabili.

Questa documentazione consente di comprendere come applicare il modello di responsabilità condivisa quando si usa il servizio gestito per Apache Flink. I seguenti argomenti illustrano come configurare il servizio gestito per Apache Flink per soddisfare gli obiettivi di sicurezza e conformità. Scoprirai anche come utilizzare altri servizi Amazon che semplificano il monitoraggio e la protezione delle risorse del servizio gestito per Apache Flink.

Argomenti

- [Protezione dei dati nel servizio gestito da Amazon per Apache Flink](#)
- [Identity and Access Management per il servizio gestito da Amazon per Apache Flink](#)
- [Convalida della conformità per il servizio gestito da Amazon per Apache Flink](#)
- [Resilienza nel servizio gestito da Amazon per Apache Flink](#)
- [Sicurezza dell'infrastruttura nel servizio gestito per Apache Flink](#)
- [Best practice di sicurezza per il servizio gestito per Apache Flink](#)

Protezione dei dati nel servizio gestito da Amazon per Apache Flink

Puoi proteggere i tuoi dati utilizzando gli strumenti forniti da AWS. Il servizio gestito per Apache Flink può funzionare con servizi che supportano la crittografia dei dati, tra cui Firehose e Amazon S3.

Crittografia dei dati nel servizio gestito per Apache Flink

Crittografia a riposo

Tieni presente quanto segue sulla crittografia dei dati a riposo con il servizio gestito per Apache Flink:

- Puoi crittografare i dati sul flusso di dati Kinesis in entrata utilizzando [StartStreamEncryption](#). Per ulteriori informazioni, consulta [Cos'è la crittografia lato server per il flusso di dati Kinesis?](#).
- I dati di output possono essere crittografati quando sono inattivi utilizzando Firehose per archiviare i dati in un bucket Amazon S3 crittografato. È possibile specificare la chiave di crittografia utilizzata dal bucket Amazon S3. Per ulteriori informazioni, consulta [Protezione dei dati mediante la crittografia lato server con chiavi gestite da KMS \(SSE-KMS\)](#).
- Il servizio gestito per Apache Flink può leggere da qualsiasi origine di streaming e scrivere su qualsiasi destinazione di streaming o database. Assicurati che le tue origini e destinazioni crittografino tutti i dati in transito e i dati a riposo.
- Il codice dell'applicazione viene crittografato mentre è inattivo.
- Lo storage durevole delle applicazioni viene crittografato quando è inattivo.
- Lo storage delle applicazioni in esecuzione viene crittografato quando è inattivo.

Crittografia in transito

Il servizio gestito per Apache Flink crittografa tutti i dati in transito. La crittografia in transito è abilitata per tutte le applicazioni del servizio gestito per Apache Flink e non può essere disabilitata.

Il servizio gestito per Apache Flink crittografa i dati in transito nei seguenti scenari:

- Dati in transito dal flusso di dati Kinesis al servizio gestito per Apache Flink.
- Dati in transito tra componenti interni all'interno del servizio gestito per Apache Flink.
- Dati in transito tra il servizio gestito per Apache Flink e Firehose.

Gestione delle chiavi

La crittografia dei dati nel servizio gestito per Apache Flink utilizza chiavi gestite dal servizio. Le chiavi gestite dal cliente non sono supportate.

Identity and Access Management per il servizio gestito da Amazon per Apache Flink

AWS Identity and Access Management (IAM) è un Servizio AWS che aiuta gli amministratori a controllare in modo sicuro l'accesso alle AWS risorse. Gli amministratori IAM controllano chi può essere autenticato (connesso) e autorizzato (dispone di autorizzazioni) a utilizzare le risorse del servizio gestito per Apache Flink. IAM è un Servizio AWS che è possibile utilizzare senza alcun costo aggiuntivo.

Argomenti

- [Destinatari](#)
- [Autenticazione con identità](#)
- [Gestione dell'accesso con policy](#)
- [Funzionamento del servizio gestito da Amazon per Apache Flink con IAM](#)
- [Esempi di policy basate su identità per il servizio gestito da Amazon per Apache Flink](#)
- [Risoluzione dei problemi di identità e accesso al servizio gestito da Amazon per Apache Flink](#)
- [Prevenzione del problema "confused deputy" tra servizi](#)

Destinatari

Le modalità di utilizzo di AWS Identity and Access Management (IAM) cambiano in base alle operazioni eseguite nel servizio gestito per Apache Flink.

Utente del servizio: se utilizzi il servizio gestito per Apache Flink per eseguire il processo, l'amministratore ti fornisce le credenziali e le autorizzazioni necessarie. All'aumentare del numero di funzionalità del servizio gestito per Apache Flink utilizzate per il processo, potrebbero essere necessarie ulteriori autorizzazioni. La comprensione della gestione dell'accesso ti consente di richiedere le autorizzazioni corrette all'amministratore. Se non riesci ad accedere a una funzionalità del servizio gestito per Apache Flink, consulta [Risoluzione dei problemi di identità e accesso al servizio gestito da Amazon per Apache Flink](#).

Amministratore del servizio: se sei il responsabile delle risorse del servizio gestito per Apache Flink presso la tua azienda, probabilmente disponi dell'accesso completo al servizio gestito per Apache Flink. Il tuo compito è determinare a quali funzionalità e risorse del servizio gestito per Apache Flink gli utenti del servizio possono accedere. Devi inviare le richieste all'amministratore IAM per cambiare le autorizzazioni degli utenti del servizio. Esamina le informazioni contenute in questa pagina per comprendere i concetti di base relativi a IAM. Per ulteriori informazioni su come la tua azienda può utilizzare l'IAM con il servizio gestito per Apache Flink, consulta [Funzionamento del servizio gestito da Amazon per Apache Flink con IAM](#).

Amministratore IAM: un amministratore IAM potrebbe essere interessato a ottenere dei dettagli su come scrivere policy per gestire l'accesso al servizio gestito per Apache Flink. Per visualizzare esempi di policy basate su identità del servizio gestito per Apache Flink utilizzabili in IAM, consulta [Esempi di policy basate su identità per il servizio gestito da Amazon per Apache Flink](#).

Autenticazione con identità

L'autenticazione è la procedura di accesso ad AWS utilizzando le credenziali dell'identità. Devi essere autenticato (connesso a AWS) come utente autenticato (connesso a) come utente IAM o assumere un ruolo IAM. Utente root dell'account AWS

Puoi accedere ad AWS come identità federata utilizzando le credenziali fornite tramite un'origine di identità. AWS IAM Identity Center Gli esempi di identità federate comprendono gli utenti del centro identità IAM, l'autenticazione Single Sign-On (SSO) dell'azienda e le credenziali di Google o Facebook. Se accedi come identità federata, l'amministratore ha configurato in precedenza la federazione delle identità utilizzando i ruoli IAM. Se accedi ad AWS tramite la federazione, assumi indirettamente un ruolo.

A seconda del tipo di utente, è possibile accedere alla AWS Management Console o al portale di AWS accesso. Per ulteriori informazioni sull'accesso a AWS, vedi [Come accedere al tuo Account AWS nella Guida per l'Accedi ad AWS utente](#).

Se accedi ad in modo AWS programmatico, AWS fornisce un Software Development Kit (SDK) e un'interfaccia della linea di comando (CLI) per firmare crittograficamente le richieste utilizzando le tue credenziali. Se non utilizzi AWS gli strumenti, devi firmare le richieste personalmente. Per ulteriori informazioni sul metodo consigliato per la firma delle richieste, consulta [Signature Version 4 AWS per le richieste API](#) nella Guida per l'utente IAM.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. Ad esempio, AWS consiglia ad esempio di utilizzare l'autenticazione

a più fattori (MFA) per aumentare la sicurezza del tuo account. Per ulteriori informazioni, consulta [Autenticazione a più fattori](#) nella Guida per l'utente di AWS IAM Identity Center e [Utilizzo dell'autenticazione a più fattori \(MFA\)AWS in IAM](#) nella Guida per l'utente IAM.

Account AWS Utente root

Quando crei un Account AWS, inizi con una singola identità di accesso che ha accesso completo a tutti i Servizi AWS e le risorse nell'account. Tale identità è detta utente Account AWS root ed è possibile accedervi con l'indirizzo e-mail e la password utilizzati per creare l'account. Si consiglia vivamente di non utilizzare l'utente root per le attività quotidiane. Conserva le credenziali dell'utente root e utilizzale per eseguire le operazioni che solo l'utente root può eseguire. Per un elenco completo delle attività che richiedono l'accesso come utente root, consulta la sezione [Attività che richiedono le credenziali dell'utente root](#) nella Guida per l'utente IAM.

Identità federata

Come best practice, richiedere agli utenti umani, compresi quelli che richiedono l'accesso di amministratore, di utilizzare la federazione con un provider di identità per accedere a Servizi AWS utilizzando credenziali temporanee.

Un'identità federata è un utente della directory degli utenti aziendali, un provider di identità Web, AWS Directory Service, la directory Identity Center o qualsiasi utente che accede Servizi AWS ad utilizzando credenziali fornite tramite un'origine di identità. Quando le identità federate accedono a Account AWS, assumono ruoli e i ruoli forniscono credenziali temporanee.

Per la gestione centralizzata degli accessi, consigliamo di utilizzare AWS IAM Identity Center. È possibile creare utenti e gruppi in IAM Identity Center oppure connettersi e sincronizzarsi con un gruppo di utenti e gruppi nell'origine di identità per utilizzarli in tutte le applicazioni Account AWS e gli. Per ulteriori informazioni su IAM Identity Center, consulta [Cos'è IAM Identity Center?](#) nella Guida per l'utente di AWS IAM Identity Center .

Utenti e gruppi IAM

Un [utente IAM](#) è una identità all'interno di Account AWS che dispone di autorizzazioni specifiche per un singolo utente o applicazione. Ove possibile, consigliamo di fare affidamento a credenziali temporanee invece di creare utenti IAM con credenziali a lungo termine come le password e le chiavi di accesso. Tuttavia, se si hanno casi d'uso specifici che richiedono credenziali a lungo termine con utenti IAM, si consiglia di ruotare le chiavi di accesso. Per ulteriori informazioni, consulta la pagina [Rotazione periodica delle chiavi di accesso per casi d'uso che richiedono credenziali a lungo termine](#) nella Guida per l'utente IAM.

Un [gruppo IAM](#) è un'identità che specifica un insieme di utenti IAM. Non è possibile eseguire l'accesso come gruppo. È possibile utilizzare gruppi per specificare le autorizzazioni per più utenti alla volta. I gruppi semplificano la gestione delle autorizzazioni per set di utenti di grandi dimensioni. Ad esempio, è possibile avere un gruppo denominato IAMAdminse concedere a tale gruppo le autorizzazioni per amministrare le risorse IAM.

Gli utenti sono diversi dai ruoli. Un utente è associato in modo univoco a una persona o un'applicazione, mentre un ruolo è destinato a essere assunto da chiunque ne abbia bisogno. Gli utenti dispongono di credenziali a lungo termine permanenti, mentre i ruoli forniscono credenziali temporanee. Per ulteriori informazioni, consulta [Casi d'uso per utenti IAM](#) nella Guida per l'utente IAM.

Ruoli IAM

Un [ruolo IAM](#) è un'identità interna all'utente Account AWS che dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. Per assumere temporaneamente un ruolo IAM nella AWS Management Console, è possibile [passare da un ruolo utente a un ruolo IAM \(console\)](#). Puoi assumere un ruolo chiamando un'operazione AWS CLI o AWS API oppure utilizzando un URL personalizzato. Per ulteriori informazioni sui metodi per l'utilizzo dei ruoli, consulta [Utilizzo di ruoli IAM](#) nella Guida per l'utente IAM.

I ruoli IAM con credenziali temporanee sono utili nelle seguenti situazioni:

- **Accesso utente federato:** per assegnare le autorizzazioni a una identità federata, è possibile creare un ruolo e definire le autorizzazioni per il ruolo. Quando un'identità federata viene autenticata, l'identità viene associata al ruolo e ottiene le autorizzazioni da esso definite. Per ulteriori informazioni sulla federazione dei ruoli, consulta [Create a role for a third-party identity provider \(federation\)](#) nella Guida per l'utente IAM. Se utilizzi IAM Identity Center, configura un set di autorizzazioni. IAM Identity Center mette in correlazione il set di autorizzazioni con un ruolo in IAM per controllare a cosa possono accedere le identità dopo l'autenticazione. Per informazioni sui set di autorizzazioni, consulta [Set di autorizzazioni](#) nella Guida per l'utente di AWS IAM Identity Center.
- **Autorizzazioni utente IAM temporanee:** un utente IAM o un ruolo può assumere un ruolo IAM per ottenere temporaneamente autorizzazioni diverse per un'attività specifica.
- **Accesso multi-account:** è possibile utilizzare un ruolo IAM per permettere a un utente (un principale affidabile) con un account diverso di accedere alle risorse nell'account. I ruoli sono lo strumento principale per concedere l'accesso multi-account. Tuttavia, per alcuni dei Servizi AWS, è possibile collegare una policy direttamente a una risorsa (anziché utilizzare un ruolo come un proxy).

Per informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta [Accesso a risorse multi-account in IAM](#) nella Guida per l'utente IAM.

- **Accesso cross-service** - Alcuni Servizi AWS utilizzano funzionalità in altri Servizi AWS. Ad esempio, quando effettui una chiamata in un servizio, è comune che tale servizio esegua applicazioni in Amazon EC2 o memorizzi oggetti in Amazon S3. Un servizio può eseguire questa operazione utilizzando le autorizzazioni dell'entità chiamante, utilizzando un ruolo di servizio o utilizzando un ruolo collegato al servizio.
- **Forward access sessions (FAS)**: quando si utilizza un utente o un ruolo IAM per eseguire operazioni in AWS, si viene considerati un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra operazione in un servizio diverso. La tecnologia FAS utilizza le autorizzazioni del principale che effettua la chiamata a un Servizio AWS, combinate con la richiesta di un Servizio AWS per effettuare richieste a servizi downstream. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che comporta interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli delle policy relative alle richieste FAS, consulta [Forward access sessions](#).
- **Ruolo di servizio**: un ruolo di servizio è un [ruolo IAM](#) che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione [Create a role to delegate permissions to an Servizio AWS](#) nella Guida per l'utente IAM.
- **Ruolo collegato al servizio**: un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un Servizio AWS. Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati ai servizi sono visualizzati nell'account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.
- **Applicazioni in esecuzione su Amazon EC2**: è possibile utilizzare un ruolo IAM per gestire credenziali temporanee per le applicazioni in esecuzione su un' EC2 istanza che eseguono richieste AWS API AWS CLI o. Ciò è preferibile all'archiviazione delle chiavi di accesso nell' EC2 istanza. Per assegnare un AWS ruolo a un' EC2 istanza e renderlo disponibile per tutte le relative applicazioni, occorre creare un profilo di istanza collegato all'istanza. Un profilo di istanza contiene il ruolo e consente ai programmi in esecuzione sull' EC2 istanza di ottenere le credenziali temporanee. Per ulteriori informazioni, consulta [Utilizzare un ruolo IAM per concedere le autorizzazioni alle applicazioni in esecuzione su EC2 istanze Amazon](#) nella IAM User Guide.

Gestione dell'accesso con policy

Per controllare l'accesso ad AWS è possibile creare policy e collegarle a AWS identità o risorse. Una policy è un oggetto in AWS che, se associato a un'identità o risorsa, ne definisce le relative autorizzazioni. AWS valuta queste policy quando un principale (utente, utente root o sessione ruolo) effettua una richiesta. Le autorizzazioni nelle policy determinano l'approvazione o il rifiuto della richiesta. La maggior parte delle policy viene memorizzata in AWS sotto forma di documenti JSON. Per ulteriori informazioni sulla struttura e sui contenuti dei documenti delle policy JSON, consulta [Panoramica delle policy JSON](#) nella Guida per l'utente IAM.

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale principale può eseguire operazioni su quali risorse e in quali condizioni.

Per impostazione predefinita, utenti e ruoli non dispongono di autorizzazioni. Per concedere agli utenti l'autorizzazione a eseguire operazioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. L'amministratore può quindi aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Le policy IAM definiscono le autorizzazioni relative a un'operazione, a prescindere dal metodo utilizzato per eseguirla. Ad esempio, supponiamo di disporre di una policy che consente l'operazione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dall' AWS Management Console AWS CLI, dall'o dall' AWS API.

Policy basate sull'identità

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta [Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente](#) nella Guida per l'utente IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono integrate direttamente in un singolo utente, gruppo o ruolo. Le policy gestite sono policy standalone che possono essere collegate a più utenti, gruppi e ruoli in. Account AWS Le policy gestite includono le policy AWS gestite da e le policy gestite dal cliente. Per informazioni su come scegliere tra una policy gestita o una policy inline, consulta [Scelta fra policy gestite e policy inline](#) nella Guida per l'utente IAM.

Policy basate sulle risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile collegare a una risorsa. Esempi di policy basate sulle risorse sono le policy di attendibilità dei ruoli IAM e le policy dei bucket Amazon S3. Nei servizi che supportano policy basate sulle risorse, gli amministratori dei servizi possono utilizzarli per controllare l'accesso a una risorsa specifica. Quando è collegata a una risorsa, una policy definisce le operazioni che un principale può eseguire su tale risorsa e a quali condizioni. È necessario [specificare un principale](#) in una policy basata sulle risorse. I principali possono includere account, utenti federati, utenti federati o. Servizi AWS

Le policy basate sulle risorse sono policy inline che si trovano in tale servizio. Non è possibile utilizzare le policy AWS gestite da IAM in una policy basata su risorse.

Liste di controllo accessi di rete (ACLs)

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni ad accedere a una risorsa. ACLs sono simili alle policy basate su risorse, anche se non utilizzano il formato del documento di policy JSON.

Amazon S3 AWS WAF, e Amazon VPC sono esempi di servizi che supportano ACLs. Per maggiori informazioni ACLs, consulta [Panoramica dell'elenco di controllo degli accessi](#) nella Guida per gli sviluppatori di Amazon Simple Storage Service.

Altri tipi di policy

AWS supporta tipi di policy meno comuni aggiuntivi. Questi tipi di policy possono impostare il numero massimo di autorizzazioni concesse dai tipi di policy più comuni.

- **Limiti delle autorizzazioni:** un limite delle autorizzazioni è una funzionalità avanzata nella quale si imposta il numero massimo di autorizzazioni che una policy basata su identità può concedere a un'entità IAM (utente o ruolo IAM). È possibile impostare un limite delle autorizzazioni per un'entità. Le autorizzazioni risultanti sono l'intersezione delle policy basate su identità dell'entità e i relativi limiti delle autorizzazioni. Le policy basate su risorse che specificano l'utente o il ruolo nel campo `Principal` sono condizionate dal limite delle autorizzazioni. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni sui limiti delle autorizzazioni, consulta [Limiti delle autorizzazioni per le entità IAM](#) nella Guida per l'utente IAM.
- **Policy di controllo dei servizi (SCPs):** SCPs sono policy JSON che specificano il numero massimo di autorizzazioni per un'organizzazione o unità organizzativa (OU) in AWS Organizations AWS

Organizations è un servizio per il raggruppamento e la gestione centralizzata di più Account AWS di proprietà dell'azienda. Se si abilitano tutte le caratteristiche in un'organizzazione, è possibile applicare le policy di controllo del servizio (SCPs) a uno o tutti i propri account. Una SCP limita le autorizzazioni per le entità negli account membri, compreso ogni. Utente root dell'account AWS Per ulteriori informazioni su Organizations and SCPs, consulta [le politiche di controllo dei servizi](#) nella Guida AWS Organizations per l'utente.

- Policy di controllo delle risorse (RCPs): RCPs sono policy JSON che consentono di impostare il numero massimo di autorizzazioni disponibili per le risorse nei tuoi account senza aggiornare le policy IAM collegate a ciascuna risorsa di tua proprietà. L'RCP limita le autorizzazioni per le risorse negli account membri e può influire sulle autorizzazioni valide per le identità, tra cui Utente root dell'account AWS, indipendentemente dal fatto che appartengano o meno alla tua organizzazione. Per ulteriori informazioni su Organizations e RCPs, incluso un elenco di Servizi AWS tale supporto RCPs, vedere [Resource control policies \(RCPs\)](#) nella Guida per l'AWS Organizations utente.
- Policy di sessione: le policy di sessione sono policy avanzate che vengono trasmesse come parametro quando si crea in modo programmatico una sessione temporanea per un ruolo o un utente federato. Le autorizzazioni della sessione risultante sono l'intersezione delle policy basate su identità del ruolo o dell'utente e le policy di sessione. Le autorizzazioni possono anche provenire da una policy basata su risorse. Un rifiuto esplicito in una qualsiasi di queste policy sostituisce l'autorizzazione. Per ulteriori informazioni, consulta [Policy di sessione](#) nella Guida per l'utente IAM.

Più tipi di policy

Quando più tipi di policy si applicano a una richiesta, le autorizzazioni risultanti sono più complicate da comprendere. Per informazioni su come AWS determina se consentire una richiesta quando sono coinvolti più tipi di policy, consultare [Logica di valutazione delle policy](#) nella Guida per l'utente di IAM.

Funzionamento del servizio gestito da Amazon per Apache Flink con IAM

Prima di utilizzare IAM per gestire l'accesso al servizio gestito per Apache Flink, scopri quali funzionalità di IAM sono disponibili per l'uso con il servizio gestito per Apache Flink.

Funzionalità IAM utilizzabili con il servizio gestito da Amazon per Apache Flink

Funzionalità IAM	Supporto del servizio gestito per Apache Flink
Policy basate su identità	Sì

Funzionalità IAM	Supporto del servizio gestito per Apache Flink
Policy basate su risorse	No
Azioni di policy	Sì
Risorse relative alle policy	Sì
Chiavi di condizione delle policy	No
ACLs	No
ABAC (tag nelle policy)	Sì
Credenziali temporanee	Sì
Autorizzazioni del principale	Sì
Ruoli di servizio	No
Ruoli collegati al servizio	No

Per ottenere un quadro generale del funzionamento del servizio gestito per Apache Flink e di altri AWS servizi con la maggior parte delle funzionalità di IAM, consulta [AWS Servizi supportati da IAM nella Guida](#) per l'utente IAM.

Policy basate su identità per servizio gestito da Amazon per Apache Flink

Supporta le policy basate su identità: sì

Le policy basate su identità sono documenti di policy di autorizzazione JSON che è possibile allegare a un'identità (utente, gruppo di utenti o ruolo IAM). Tali policy definiscono le operazioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta [Definizione di autorizzazioni personalizzate IAM con policy gestite dal cliente](#) nella Guida per l'utente IAM.

Con le policy basate su identità di IAM, è possibile specificare quali operazioni e risorse sono consentite o respinte, nonché le condizioni in base alle quali le operazioni sono consentite o respinte. Non è possibile specificare l'entità principale in una policy basata sull'identità perché si applica all'utente o al ruolo a cui è associato. Per informazioni su tutti gli elementi utilizzabili in una policy

JSON, consulta [Guida di riferimento agli elementi delle policy JSON IAM](#) nella Guida per l'utente di IAM.

Esempi di policy basate su identità per il servizio gestito da Amazon per Apache Flink

Per visualizzare esempi di policy basate su identità del servizio gestito per Apache Flink, consulta [Esempi di policy basate su identità per il servizio gestito da Amazon per Apache Flink](#).

Policy basate su risorse all'interno del servizio gestito per Apache Flink

Il servizio gestito da Amazon per Apache Flink al momento non supporta il controllo degli accessi basato su risorse.

Accesso a risorse multi-account dall'applicazione del servizio gestito per Apache Flink

Per consentire a un'applicazione Managed Service for Apache Flink di accedere a una risorsa come uno stream Amazon Kinesis o un bucket Amazon S3, devi creare un ruolo IAM nell'account della risorsa. Il ruolo deve disporre di autorizzazioni sufficienti per accedere alla risorsa. È inoltre necessario aggiungere una politica di fiducia che autorizzi l'intero account dell'applicazione Managed Service for Apache Flink ad assumere il ruolo.

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

Inoltre, il ruolo IAM assegnato all'applicazione del servizio gestito per Apache Flink deve consentire l'assunzione del ruolo nell'account della risorsa.

```
{
  "Version": "2012-10-17",
```

```
"Statement": [  
  {  
    "Sid": "AllowAssumingRoleInStreamAccount",  
    "Effect": "Allow",  
    "Action": "sts:AssumeRole",  
    "Resource": "arn:aws:iam::Stream-account-ID:role/Role-to-assume"  
  }  
]  
}
```

Per ulteriori informazioni, consulta [Accesso a risorse multi-account in IAM](#) nella Guida per l'utente IAM.

Operazioni di policy per il servizio gestito per Apache Flink

Supporta le operazioni di policy: si

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale principale può eseguire operazioni su quali risorse, e in quali condizioni.

L'elemento `Action` di una policy JSON descrive le operazioni che è possibile utilizzare per consentire o negare l'accesso a un criterio. Le operazioni della policy hanno spesso lo stesso nome dell'operazione AWS API. Ci sono alcune eccezioni, ad esempio le operazioni di sola autorizzazione che non hanno un'operazione API corrispondente. Esistono anche alcune operazioni che richiedono più operazioni in una policy. Queste operazioni aggiuntive sono denominate operazioni dipendenti.

Includi le operazioni in una policy per concedere le autorizzazioni a eseguire l'operazione associata.

Per visualizzare un elenco di operazioni del servizio gestito per Apache Flink, consulta [Operazioni definite dal servizio gestito da Amazon per Apache Flink](#) nella Guida di riferimento per l'autorizzazione del servizio.

Le operazioni di policy nel servizio gestito per Apache Flink utilizzano il seguente prefisso prima dell'operazione:

```
Kinesis Analytics
```

Per specificare più operazioni in una sola istruzione, occorre separarle con la virgola.

```
"Action": [  

```

```
"Kinesis Analytics:action1",  
"Kinesis Analytics:action2"  
]
```

È possibile specificare più azioni tramite caratteri jolly (*). Ad esempio, per specificare tutte le azioni che iniziano con la parola Describe, includi la seguente azione:

```
"Action": "Kinesis Analytics:Describe*"
```

Per visualizzare esempi di policy basate su identità del servizio gestito per Apache Flink, consulta [Esempi di policy basate su identità per il servizio gestito da Amazon per Apache Flink](#).

Risorse di policy per il servizio gestito per Apache Flink

Supporta le risorse di policy: sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale principale può eseguire operazioni su quali risorse, e in quali condizioni.

L'elemento JSON `Resource` della policy specifica l'oggetto o gli oggetti ai quali si applica l'operazione. Le istruzioni devono includere un elemento `Resource` o un elemento `NotResource`. Come best practice, specifica una risorsa utilizzando il suo [nome della risorsa Amazon \(ARN\)](#). È possibile eseguire questa operazione per operazioni che supportano un tipo di risorsa specifico, note come autorizzazioni a livello di risorsa.

Per le azioni che non supportano le autorizzazioni a livello di risorsa, ad esempio le operazioni di elenco, utilizza un carattere jolly (*) per indicare che l'istruzione si applica a tutte le risorse.

```
"Resource": "*"
```

Per visualizzare un elenco di tipi di risorse del servizio gestito per Apache Flink e i relativi tipi di risorse ARNs, consulta [Resources Defined by Amazon Managed Service for Apache Flink](#) nella Guida di riferimento per l'autorizzazione del servizio. Per informazioni sulle operazioni con cui è possibile specificare l'ARN di ogni risorsa, consulta [Operazioni definite dal servizio gestito da Amazon per Apache Flink](#).

Per visualizzare esempi di policy basate su identità del servizio gestito per Apache Flink, consulta [Esempi di policy basate su identità per il servizio gestito da Amazon per Apache Flink](#).

Chiavi di condizione delle policy per il servizio gestito per Apache Flink

Supporta le chiavi di condizione delle policy specifiche del servizio: sì

Gli amministratori possono utilizzare le policy AWS JSON per specificare chi ha accesso a cosa. In altre parole, quale principale può eseguire operazioni su quali risorse, e in quali condizioni.

L'elemento `Condition`(o blocco `Condition`) consente di specificare le condizioni in cui un'istruzione è in vigore. L'elemento `Condition` è facoltativo. È possibile compilare espressioni condizionali che utilizzano [operatori di condizione](#), ad esempio uguale a o minore di, per soddisfare la condizione nella policy con i valori nella richiesta.

Se specifichi più elementi `Condition` in un'istruzione o più chiavi in un singolo elemento `Condition`, questi vengono valutati da AWS utilizzando un'operazione AND logica. Se specifichi più valori per una singola chiave di condizione, AWS valuta la condizione utilizzando un'OR operazione logica. Tutte le condizioni devono essere soddisfatte prima che le autorizzazioni dell'istruzione vengano concesse.

È possibile anche utilizzare variabili segnaposto quando specifichi le condizioni. Ad esempio, è possibile autorizzare un utente IAM ad accedere a una risorsa solo se è stata taggata con il relativo nome utente IAM. Per ulteriori informazioni, consulta [Elementi delle policy IAM: variabili e tag](#) nella Guida per l'utente di IAM.

AWS supporta chiavi di condizione globali e chiavi di condizione specifiche per il servizio. Per visualizzare tutte le chiavi di [condizione AWS globali](#) di nella Guida per l'utente di IAM.

Per visualizzare un elenco completo delle chiavi di condizione del servizio gestito per Apache Flink, consulta [Chiavi di condizione per il servizio gestito da Amazon per Apache Flink](#) nella Guida di riferimento per l'autorizzazione del servizio. Per informazioni su operazioni e risorse con cui è possibile utilizzare una chiave di condizione, consulta [Operazioni definite dal servizio gestito da Amazon per Apache Flink](#).

Per visualizzare esempi di policy basate su identità del servizio gestito per Apache Flink, consulta [Esempi di policy basate su identità per il servizio gestito da Amazon per Apache Flink](#).

Liste di controllo degli accessi (ACLs) nel servizio gestito per Apache Flink

Supporti ACLs: no

Le liste di controllo degli accessi (ACLs) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni ad accedere a una risorsa. ACLs sono simili alle policy basate su risorse, anche se non utilizzano il formato del documento di policy JSON.

Controllo degli accessi basato su attributi (ABAC) con il servizio gestito per Apache Flink

Supporta ABAC (tag nelle policy): sì

Il controllo dell'accesso basato su attributi (ABAC) è una strategia di autorizzazione che definisce le autorizzazioni in base agli attributi. In AWS, questi attributi sono chiamati tag. È possibile collegare dei tag alle entità IAM (utenti o ruoli) e a numerose AWS risorse. L'assegnazione di tag alle entità e alle risorse è il primo passaggio di ABAC. In seguito, vengono progettate policy ABAC per consentire operazioni quando il tag dell'entità principale corrisponde al tag sulla risorsa a cui si sta provando ad accedere.

La strategia ABAC è utile in ambienti soggetti a una rapida crescita e aiuta in situazioni in cui la gestione delle policy diventa impegnativa.

Per controllare l'accesso basato su tag, fornisci informazioni sui tag nell'[elemento condizione](#) di una policy utilizzando le chiavi di condizione `aws:ResourceTag/key-name`, `aws:RequestTag/key-name` o `aws:TagKeys`.

Se un servizio supporta tutte e tre le chiavi di condizione per ogni tipo di risorsa, il valore per il servizio è Yes (Sì). Se un servizio supporta tutte e tre le chiavi di condizione solo per alcuni tipi di risorsa, allora il valore sarà Parziale.

Per ulteriori informazioni su ABAC, consulta [Definizione delle autorizzazioni con autorizzazione ABAC](#) nella Guida per l'utente IAM. Per visualizzare un tutorial con i passaggi per l'impostazione di ABAC, consulta [Utilizzo del controllo degli accessi basato su attributi \(ABAC\)](#) nella Guida per l'utente di IAM.

Utilizzo delle credenziali temporanee con il servizio gestito da Amazon per Apache Flink

Supporta le credenziali temporanee: sì

Alcuni Servizi AWS non funzionano quando si accede utilizzando credenziali temporanee. Per ulteriori informazioni, inclusi i che Servizi AWS funzionano con credenziali temporanee, consulta [Servizi AWS supportati da IAM](#) nella Guida per l'utente di IAM.

Le credenziali temporanee sono utilizzate se si accede alla AWS Management Console utilizzando qualsiasi metodo che non sia la combinazione di nome utente e password. Ad esempio, quando accedi alla AWS utilizzando il collegamento Single Sign-On (SSO) della tua azienda, tale processo crea in automatico credenziali temporanee. Le credenziali temporanee vengono create in automatico anche quando accedi alla console come utente e poi cambi ruolo. Per ulteriori informazioni sullo scambio dei ruoli, consulta [Passaggio da un ruolo utente a un ruolo IAM \(console\)](#) nella Guida per l'utente IAM.

È possibile creare manualmente credenziali temporanee utilizzando la AWS CLI o l'AWS API. È quindi possibile utilizzare tali credenziali temporanee per accedere. AWS consiglia di generare le credenziali temporanee dinamicamente anziché utilizzare chiavi di accesso a lungo termine. Per ulteriori informazioni, consulta [Credenziali di sicurezza provvisorie in IAM](#).

Autorizzazioni del principale tra servizi per il servizio gestito per Apache Flink

Supporta l'inoltro delle sessioni di accesso (FAS): sì

Quando si utilizza un utente o un ruolo IAM per eseguire operazioni in AWS, si viene considerati entità. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra operazione in un servizio diverso. La tecnologia FAS utilizza le autorizzazioni del principale che effettua la chiamata a un Servizio AWS, combinate con la richiesta di un Servizio AWS per effettuare richieste a servizi downstream. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che comporta interazioni con altri Servizi AWS o risorse per essere completata. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le azioni. Per i dettagli delle policy relative alle richieste FAS, consulta [Forward access sessions](#).

Ruoli di servizio per il servizio gestito da Amazon per Apache Flink

Supporta i ruoli di servizio: sì

Un ruolo di servizio è un [ruolo IAM](#) che un servizio assume per eseguire operazioni per tuo conto. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione [Create a role to delegate permissions to an Servizio AWS](#) nella Guida per l'utente IAM.

 Warning

La modifica delle autorizzazioni per un ruolo di servizio potrebbe compromettere la funzionalità del servizio gestito per Apache Flink. Modifica i ruoli di servizio solo quando il servizio gestito per Apache Flink fornisce le indicazioni per farlo.

Ruoli collegati ai servizi per il servizio gestito da Amazon per Apache Flink

Supporta ruoli collegati ai servizi: Sì

Un ruolo collegato al servizio è un tipo di ruolo di servizio collegato a un. Servizio AWS Il servizio può assumere il ruolo per eseguire un'azione per tuo conto. I ruoli collegati ai servizi sono visualizzati nell'account Account AWS e sono di proprietà del servizio. Un amministratore IAM può visualizzare le autorizzazioni per i ruoli collegati ai servizi, ma non modificarle.

Per ulteriori informazioni su come creare e gestire i ruoli collegati ai servizi, consulta [Servizi AWS supportati da IAM](#). Trova un servizio nella tabella che include un Yes nella colonna Service-linked role (Ruolo collegato ai servizi). Scegli il collegamento Sì per visualizzare la documentazione relativa al ruolo collegato ai servizi per tale servizio.

Esempi di policy basate su identità per il servizio gestito da Amazon per Apache Flink

Per impostazione predefinita, gli utenti e i ruoli non dispongono dell'autorizzazione per creare o modificare risorse del servizio gestito per Apache Flink. Inoltre, non possono eseguire attività utilizzando AWS Management Console, AWS Command Line Interface (AWS CLI) o AWS l'API. Per concedere agli utenti l'autorizzazione a eseguire operazioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. L'amministratore può quindi aggiungere le policy IAM ai ruoli e gli utenti possono assumere i ruoli.

Per informazioni su come creare una policy basata su identità IAM utilizzando questi documenti di policy JSON di esempio, consulta [Creazione di policy IAM \(console\)](#) nella Guida per l'utente IAM.

Per informazioni dettagliate sulle operazioni e sui tipi di risorse definiti dal servizio gestito per Apache Flink incluso il formato di ARNs per ogni tipo di risorsa, consulta [Operazioni, risorse e chiavi di condizione per il servizio gestito da Amazon per Apache Flink nella Guida di riferimento per l'autorizzazione del servizio](#).

Argomenti

- [Best practice per le policy](#)
- [Utilizzo della console del servizio gestito per Apache Flink](#)
- [Consentire agli utenti di visualizzare le loro autorizzazioni](#)

Best practice per le policy

Le policy basate su identità determinano se qualcuno può creare, accedere o eliminare risorse del servizio gestito per Apache Flink nel tuo account. Queste azioni possono comportare costi aggiuntivi per l'Account AWS. Quando crei o modifichi policy basate su identità, segui queste linee guida e raccomandazioni:

- Nozioni di base sulle policy AWS gestite da e passaggio alle autorizzazioni con privilegio minimo: per le informazioni di base su come concedere autorizzazioni a utenti e carichi di lavoro, utilizza le policy AWS gestite da che concedono le autorizzazioni per molti casi d'uso comuni. Sono disponibili nel tuo Account AWS. Ti consigliamo pertanto di ridurre ulteriormente le autorizzazioni definendo policy gestite dal AWS cliente specifiche per i tuoi casi d'uso. Per ulteriori informazioni, consulta [Policy gestite da AWS](#) o [Policy gestite da AWS per le funzioni dei processi](#) nella Guida per l'utente IAM.
- Applica le autorizzazioni con privilegio minimo: quando imposti le autorizzazioni con le policy IAM, concedi solo le autorizzazioni richieste per eseguire un'attività. È possibile farlo definendo le azioni che possono essere intraprese su risorse specifiche in condizioni specifiche, note anche come autorizzazioni con privilegi minimi. Per ulteriori informazioni sull'utilizzo di IAM per applicare le autorizzazioni, consulta [Policy e autorizzazioni in IAM](#) nella Guida per l'utente IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso: per limitare l'accesso a operazioni e risorse è possibile aggiungere una condizione alle tue policy. Ad esempio, è possibile scrivere una condizione di policy per specificare che tutte le richieste devono essere inviate utilizzando SSL. È inoltre possibile utilizzare le condizioni per concedere l'accesso alle azioni di servizio se vengono utilizzate tramite uno specifico Servizio AWS, ad esempio AWS CloudFormation. Per ulteriori informazioni, consulta la sezione [Elementi delle policy JSON di IAM: condizione](#) nella Guida per l'utente IAM.
- Utilizzo di IAM Access Analyzer per convalidare le policy IAM e garantire autorizzazioni sicure e funzionali: IAM Access Analyzer convalida le policy nuove ed esistenti in modo che aderiscano alla sintassi della policy IAM (JSON) e alle best practice di IAM. IAM Access Analyzer offre oltre 100 controlli delle policy e consigli utili per creare policy sicure e funzionali. Per ulteriori informazioni,

consulta [Convalida delle policy per il Sistema di analisi degli accessi IAM](#) nella Guida per l'utente IAM.

- Richiesta dell'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o utenti root nel Account AWS tuo, attiva MFA per una maggiore sicurezza. Per richiedere la MFA quando vengono chiamate le operazioni API, aggiungi le condizioni MFA alle policy. Per ulteriori informazioni, consulta [Protezione dell'accesso API con MFA](#) nella Guida per l'utente IAM.

Per maggiori informazioni sulle best practice in IAM, consulta [Best practice di sicurezza in IAM](#) nella Guida per l'utente di IAM.

Utilizzo della console del servizio gestito per Apache Flink

Per accedere alla console del servizio gestito da Amazon per Apache Flink, è necessario disporre di un set di autorizzazioni minimo. Queste autorizzazioni devono consentire di elencare e visualizzare i dettagli relativi alle risorse del servizio gestito per Apache Flink nel tuo Account AWS. Se crei una policy basata sull'identità più restrittiva rispetto alle autorizzazioni minime richieste, la console non funzionerà nel modo previsto per le entità (utenti o ruoli) associate a tale policy.

Non sono necessarie le autorizzazioni minime della console per gli utenti che effettuano chiamate solo all' AWS API di AWS CLI o di. Al contrario, concedi l'accesso solo alle operazioni che corrispondono all'operazione API che stanno cercando di eseguire.

Per garantire che gli utenti e i ruoli possano continuare a utilizzare la console del servizio gestito per Apache Flink, collega alle entità anche il servizio gestito per Apache Flink ConsoleAccess o la policy ReadOnly AWS gestita. Per ulteriori informazioni, consulta [Aggiunta di autorizzazioni a un utente](#) nella Guida per l'utente IAM.

Consentire agli utenti di visualizzare le loro autorizzazioni

Questo esempio mostra in che modo è possibile creare una policy che consente agli utenti IAM di visualizzare le policy inline e gestite che sono collegate alla relativa identità utente. Questa policy include le autorizzazioni per completare questa operazione sulla console o a livello di codice utilizzando o l' AWS CLI API. AWS

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
```

```

    "Effect": "Allow",
    "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsForUser",
        "iam:ListAttachedUserPolicies",
        "iam:ListUserPolicies",
        "iam:GetUser"
    ],
    "Resource": ["arn:aws:iam::*:user/${aws:username}"]
},
{
    "Sid": "NavigateInConsole",
    "Effect": "Allow",
    "Action": [
        "iam:GetGroupPolicy",
        "iam:GetPolicyVersion",
        "iam:GetPolicy",
        "iam:ListAttachedGroupPolicies",
        "iam:ListGroupPolicies",
        "iam:ListPolicyVersions",
        "iam:ListPolicies",
        "iam:ListUsers"
    ],
    "Resource": "*"
}
]
}

```

Risoluzione dei problemi di identità e accesso al servizio gestito da Amazon per Apache Flink

Utilizza le informazioni seguenti per diagnosticare e risolvere i problemi comuni che possono verificarsi durante l'utilizzo del servizio gestito per Apache Flink e di IAM.

Argomenti

- [Non ho l'autorizzazione a eseguire un'operazione nel servizio gestito per Apache Flink](#)
- [Non sono autorizzato a eseguire iam:M: PassRole](#)
- [Voglio consentire alle persone esterne al mio AWS account di accedere alle mie risorse del servizio gestito per Apache Flink.](#)

Non ho l'autorizzazione a eseguire un'operazione nel servizio gestito per Apache Flink

Se la AWS Management Console indica che non sei autorizzato a eseguire un'operazione, devi contattare l'amministratore per assistenza. L'amministratore è la persona da cui si sono ricevuti il nome utente e la password.

Il seguente esempio di errore si verifica quando l'utente `mateojackson` prova a utilizzare la console per visualizzare i dettagli relativi a una risorsa `my-example-widget` fittizia, ma non dispone di autorizzazioni Kinesis Analytics:`GetWidget` fittizie.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform: Kinesis
Analytics:GetWidget on resource: my-example-widget
```

In questo caso, Mateo richiede al suo amministratore di aggiornare le policy per poter accedere alla risorsa `my-example-widget` utilizzando l'operazione Kinesis Analytics:`GetWidget`.

Non sono autorizzato a eseguire iam:M: PassRole

Se ricevi un errore che indica che non hai l'autorizzazione a eseguire l'operazione `iam:PassRole`, le tue policy devono essere aggiornate per poter passare un ruolo al servizio gestito per Apache Flink.

Alcuni Servizi AWS consentono di passare un ruolo esistente a tale servizio, invece di creare un nuovo ruolo di servizio o un ruolo collegato ai servizi. Per eseguire questa operazione, è necessario disporre delle autorizzazioni per trasmettere il ruolo al servizio.

L'errore di esempio seguente si verifica quando un utente IAM denominato `marymajor` cerca di utilizzare la console per eseguire un'operazione nel servizio gestito per Apache Flink. Tuttavia, l'azione richiede che il servizio disponga delle autorizzazioni concesse da un ruolo di servizio. Mary non dispone delle autorizzazioni per passare il ruolo al servizio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

In questo caso, le policy di Mary devono essere aggiornate per poter eseguire l'operazione `iam:PassRole`.

Per ulteriore assistenza, contatta AWS l'amministratore. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire alle persone esterne al mio AWS account di accedere alle mie risorse del servizio gestito per Apache Flink.

È possibile creare un ruolo con il quale utenti in altri account o persone esterne all'organizzazione possono accedere alle tue risorse. È possibile specificare chi è attendibile per l'assunzione del ruolo. Per servizi che supportano policy basate su risorse o liste di controllo accessi (ACLs), puoi utilizzare tali policy per concedere alle persone l'accesso.

Per ulteriori informazioni, consulta gli argomenti seguenti:

- Per scoprire se il servizio gestito per Apache Flink supporta queste funzionalità, consulta [Funzionamento del servizio gestito da Amazon per Apache Flink con IAM](#).
- Per informazioni su come garantire l'accesso alle risorse negli Account AWS che possiedi, consultare [Fornire l'accesso a un utente IAM in un altro Account AWS che si possiede](#) nella Guida per l'utente di IAM.
- Per capire come fornire l'accesso alle risorse ad di terze parti Account AWS, consulta Concessione dell'[accesso agli Account AWS di proprietà di terze parti](#) nella Guida per l'utente di IAM.
- Per informazioni su come fornire l'accesso tramite la federazione delle identità, consulta [Fornire l'accesso a utenti autenticati esternamente \(Federazione delle identità\)](#) nella Guida per l'utente IAM.
- Per informazioni sulle differenze di utilizzo tra ruoli e policy basate su risorse per l'accesso multi-account, consulta [Accesso a risorse multi-account in IAM](#) nella Guida per l'utente IAM.

Prevenzione del problema "confused deputy" tra servizi

In AWS, la rappresentazione tra servizi può verificarsi quando un servizio (il servizio di chiamata) effettua una chiamata a un altro servizio (il servizio chiamato). Il servizio chiamante può essere manipolato per agire sulle risorse di un altro cliente, anche se non dovrebbe avere le autorizzazioni corrette, e ciò porta al problema del "confused deputy".

Per evitare i «confused vice», AWS fornisce strumenti che ti aiutano a proteggere i tuoi dati per tutti i servizi con principali del servizio a cui è stato concesso l'accesso alle risorse del tuo account. Questa sezione si concentra sulla prevenzione dell'interferenza tra servizi specifici del servizio gestito per Apache Flink. Tuttavia, per saperne di più su questo argomento puoi consultare la sezione [Problema del "confused deputy"](#) della Guida per l'utente IAM.

Nel contesto del servizio gestito per Apache Flink, consigliamo di utilizzare le chiavi [SourceArn](#):[e SourceAccount](#) [aw:global](#) condition context nella policy di attendibilità del ruolo per limitare l'accesso al ruolo solo alle richieste generate dalle risorse previste.

Utilizza `aws:SourceArn` se desideri consentire l'associazione di una sola risorsa all'accesso tra servizi. Utilizza `aws:SourceAccount` se desideri consentire l'associazione di qualsiasi risorsa in tale account all'uso tra servizi.

Il valore di `aws:SourceArn` deve essere l'ARN della risorsa utilizzata dal servizio gestito per Apache Flink, specificato con il seguente formato:
`arn:aws:kinesisanalytics:region:account:resource`.

L'approccio consigliato per proteggersi dal problema del "confused deputy" è quello di usare la chiave di contesto della condizione globale `aws:SourceArn` con l'ARN completo della risorsa.

Se non conosci l'ARN completo della risorsa o se scegli più risorse, utilizza la chiave `aws:SourceArn` con caratteri jolly (*) per le parti sconosciute dell'ARN. Ad esempio:
`arn:aws:kinesisanalytics::111122223333:*`.

Le policy dei ruoli fornite al servizio gestito per Apache Flink e le policy di attendibilità dei ruoli generati per te possono utilizzare queste chiavi.

Per proteggerti dal problema del "confused deputy", completa le seguenti operazioni:

Per proteggersi dal problema del "confused deputy"

1. Accedere alla Console di AWS gestione di e aprire la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Scegli Ruoli, quindi scegli il ruolo che desideri modificare.
3. Seleziona Modifica policy di attendibilità.
4. Nella pagina Modifica policy di attendibilità, sostituisci la policy JSON predefinita con una policy che utilizza una o entrambe le chiavi di contesto della condizione globale `aws:SourceArn` e `aws:SourceAccount`. Vedi la policy di esempio riportata di seguito:
5. Scegli Aggiorna policy.

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

Convalida della conformità per il servizio gestito da Amazon per Apache Flink

Revisori di terze parti valutano la sicurezza e la conformità del servizio gestito da Amazon per Apache Flink nell'ambito di più AWS programmi per la conformità di. Sono inclusi SOC, PCI e HIPAA.

Per un elenco di AWS servizi che rientrano nell'ambito di programmi di conformità specifici, consulta: Per informazioni generali, consulta [Programmi di conformità di AWS](#).

Puoi scaricare i report di audit di terze parti utilizzando AWS Artifact. Per ulteriori informazioni, consulta [Download dei rapporti in AWS Artifact](#).

La tua responsabilità di conformità durante l'utilizzo del servizio gestito per Apache Flink è determinata dalla riservatezza dei dati, dagli obiettivi di conformità dell'azienda e dalle leggi e normative applicabili. Se l'utilizzo del servizio gestito per Apache Flink è soggetto a conformità con standard quali HIPAA o PCI, AWS fornisce risorse utili:

- [Guide Quick Start per la sicurezza e conformità](#): queste guide alla distribuzione illustrano considerazioni relative all'architettura e forniscono procedure per la distribuzione di ambienti di base incentrati sulla sicurezza e sulla conformità su AWS

- [Architetture per la sicurezza e la conformità HIPAA su Amazon Web Services](#). Questo whitepaper descrive in che modo le aziende possono utilizzare AWS per creare applicazioni conformi ai requisiti HIPAA.
- [AWS Risorse per la conformità](#) di: questa raccolta di cartelle di lavoro e guide potrebbe essere utile al tuo settore e alla tua posizione.
- [AWS Config](#): questo AWS servizio valuta il livello di conformità delle configurazioni delle risorse con pratiche interne, linee guida e regolamenti di settore.
- [AWS Security Hub](#): questo AWS servizio fornisce una visione completa dello stato di sicurezza all'interno di AWS che consente di verificare la conformità con standard industriali di sicurezza e best practice.

FedRAMP

Il programma di conformità agli standard AWS FedRAMP include il servizio gestito per Apache Flink come servizio autorizzato da FedRAMP. Se sei un cliente federale o commerciale, puoi utilizzare il servizio per elaborare e archiviare carichi di lavoro sensibili nel limite di autorizzazione della regione AWS GovCloud (Stati Uniti) con dati fino al livello di impatto elevato, nonché nelle regioni Stati Uniti orientali (Virginia settentrionale), Stati Uniti orientali (Ohio), Stati Uniti occidentali (California settentrionale) e Stati Uniti occidentali (Oregon) con dati fino a un livello moderato.

[Puoi richiedere l'accesso ai pacchetti di sicurezza AWS FedRAMP di tramite il FedRAMP PMO, il AWS Sales Account Manager, oppure puoi scaricarli tramite Artifact at Artifact. AWSAWS](#)

Per ulteriori informazioni, consulta [FedRAMP](#).

Resilienza nel servizio gestito da Amazon per Apache Flink

L'infrastruttura AWS globale di è basata su AWS regioni e zone di disponibilità. AWS forniscono più zone di disponibilità fisicamente separate e isolate che sono connesse tramite reti altamente ridondanti, a bassa latenza e con throughput elevato. Con le zone di disponibilità, è possibile progettare e gestire applicazioni e database che eseguono il failover automatico tra zone di disponibilità senza interruzioni. Le zone di disponibilità sono più disponibili, tolleranti ai guasti e scalabili rispetto alle infrastrutture tradizionali a data center singolo o multiplo.

Per ulteriori informazioni sulle AWS regioni e sulle zone di disponibilità, consulta [Infrastruttura AWS globale](#) di.

Oltre all'infrastruttura AWS globale di, un servizio gestito per Apache Flink offre numerose funzionalità per supportare la resilienza dei dati e le esigenze di backup.

Ripristino di emergenza

Il servizio gestito per Apache Flink viene eseguito in una modalità serverless e si occupa di degradazioni host, disponibilità della zona di disponibilità e altri problemi correlati all'infrastruttura eseguendo la migrazione automatica. Il servizio gestito per Apache Flink raggiunge questo obiettivo attraverso numerosi meccanismi ridondanti. Ogni applicazione del servizio gestito per Apache Flink viene eseguita in un cluster Apache Flink a tenant singolo. Il cluster Apache Flink viene eseguito JobManager in modalità ad alta disponibilità utilizzando Zookeeper su più zone di disponibilità. Il servizio gestito per Apache Flink utilizza Apache Flink tramite Amazon EKS. In Amazon EKS vengono utilizzati più pod Kubernetes per ogni AWS regione nelle zone di disponibilità. In caso di errore, il servizio gestito per Apache Flink tenta innanzitutto di ripristinare l'applicazione all'interno del cluster Apache Flink in esecuzione utilizzando i checkpoint dell'applicazione, se disponibili.

Il servizio gestito per Apache Flink esegue il backup dello stato dell'applicazione utilizzando checkpoint e snapshot:

- I checkpoint sono backup dello stato dell'applicazione che il servizio gestito per Apache Flink crea periodicamente in modo automatico e utilizza per il ripristino dai guasti.
- Gli snapshot sono backup dello stato dell'applicazione creati e ripristinati manualmente.

Per ulteriori informazioni su checkpoint e snapshot, consulta [Implementa la tolleranza agli](#).

Controllo delle versioni

Il controllo delle versioni dello stato dell'applicazione archiviate viene eseguito nel modo seguente:

- Il controllo delle versioni dei checkpoint viene eseguito in modo automatico dal servizio. Se il servizio utilizza un checkpoint per riavviare l'applicazione, verrà utilizzato il checkpoint più recente.
- I savepoint vengono versionati utilizzando il parametro dell'azione.
SnapshotName [CreateApplicationSnapshot](#)

Il servizio gestito per Apache Flink crittografa i dati archiviati nei checkpoint e nei savepoint.

Sicurezza dell'infrastruttura nel servizio gestito per Apache Flink

In qualità di servizio gestito, il servizio gestito per Apache Flink è protetto dalle procedure di sicurezza di rete AWS globale descritte nel whitepaper [Amazon Web Services: Panoramica dei processi di sicurezza](#).

Utilizza le chiamate API AWS pubblicate di per accedere al servizio gestito per Apache Flink tramite la rete. Tutte le chiamate API per il servizio gestito per Apache Flink sono protette tramite Transport Layer Security (TLS) e autenticate tramite IAM. I client devono supportare TLS 1.2 o versioni successive. I client devono, inoltre, supportare le suite di crittografia con PFS (Perfect Forward Secrecy), ad esempio Ephemeral Diffie-Hellman (DHE) o Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). La maggior parte dei sistemi moderni, come Java 7 e versioni successive, supporta tali modalità.

Inoltre, le richieste devono essere firmate utilizzando un ID chiave di accesso e una chiave di accesso segreta associata a un principale IAM. In alternativa, è possibile utilizzare [AWS Security Token Service](#) (AWS STS) per generare le credenziali di sicurezza temporanee per sottoscrivere le richieste.

Best practice di sicurezza per il servizio gestito per Apache Flink

Il servizio gestito da Amazon per Apache Flink fornisce una serie di funzionalità di sicurezza che occorre valutare durante lo sviluppo e l'implementazione delle policy di sicurezza. Le seguenti best practice sono linee guida generali e non rappresentano una soluzione di sicurezza completa. Poiché queste best practice potrebbero non essere appropriate o sufficienti per l'ambiente, gestiscile come considerazioni utili anziché prescrizioni.

Implementazione dell'accesso con privilegi minimi

Quando concedi le autorizzazioni, puoi decidere chi le ottiene e verso quali risorse del servizio gestito per Apache Flink. È possibile abilitare operazioni specifiche che si desidera consentire su tali risorse. Pertanto è necessario concedere solo le autorizzazioni necessarie per eseguire un'attività. L'implementazione dell'accesso con privilegi minimi è fondamentale per ridurre i rischi di sicurezza e l'impatto risultante da errori o intenzioni dannose.

Uso di ruoli IAM per accedere ad altri servizi Amazon

L'applicazione del servizio gestito per Apache Flink deve disporre di credenziali valide per accedere alle risorse di altri servizi, come i flussi di dati Kinesis, i flussi Firehose o i bucket Amazon S3. Non è

consigliabile archiviare AWS credenziali direttamente nell'applicazione o in un bucket Amazon S3. Si tratta di credenziali a lungo termine che non vengono automaticamente ruotate e potrebbero avere un impatto aziendale significativo se vengono compromesse.

Al contrario, è consigliabile utilizzare un ruolo IAM per gestire le credenziali temporanee per l'applicazione per accedere ad altre risorse. Quando utilizzi un ruolo, non devi necessariamente usare credenziali a lungo termine per accedere ad altre risorse.

Per ulteriori informazioni, consulta gli argomenti seguenti nella Guida per l'utente IAM:

- [Ruoli IAM](#)
- [Scenari comuni per ruoli: utenti, applicazioni e servizi](#)

Implementazione della crittografia lato server

I dati a riposo e i dati in transito sono crittografati nel servizio gestito per Apache Flink e questa crittografia non può essere disabilitata. È necessario implementare la crittografia lato server nelle risorse dipendenti, come i flussi di dati Kinesis, i flussi Firehose e i bucket Amazon S3. Per ulteriori informazioni su come implementare la crittografia lato server nelle risorse dipendenti, consulta [Protezione dei dati](#).

Utilizzalo per monitorare le chiamate API CloudTrail

Il servizio gestito per Apache Flink è integrato con l' AWS CloudTrail, un servizio che fornisce una registrazione delle operazioni eseguite da un utente, un ruolo o un servizio Amazon nel servizio gestito per Apache Flink.

Le informazioni raccolte da consentono CloudTrail di determinare la richiesta effettuata al servizio gestito per Apache Flink, l'indirizzo IP da cui è partita la richiesta, l'autore della richiesta, il momento in cui è stata eseguita e altri dettagli.

Per ulteriori informazioni, consulta [the section called “Registra il servizio gestito per le chiamate API Apache Flink con AWS CloudTrail”](#).

Registrazione e monitoraggio in Amazon Managed Service per Apache Flink

Il monitoraggio è importante per garantire l'affidabilità, la disponibilità e le prestazioni delle applicazioni del servizio gestito per Apache Flink. È necessario raccogliere i dati di monitoraggio da tutte le parti della AWS soluzione in modo da poter eseguire più facilmente il debug di un errore multipunto, se si verifica.

Prima di iniziare il monitoraggio del servizio gestito per Apache Flink, è opportuno creare un piano di monitoraggio che includa le risposte alle seguenti domande:

- Quali sono gli obiettivi del monitoraggio?
- Di quali risorse si intende eseguire il monitoraggio?
- Con quale frequenza sarà eseguito il monitoraggio di queste risorse?
- Quali strumenti di monitoraggio verranno utilizzati?
- Chi eseguirà i processi di monitoraggio?
- Chi deve ricevere una notifica quando si verifica un problema?

Il passaggio successivo consiste nello stabilire una baseline per le prestazioni normali del servizio gestito per Apache Flink nel tuo ambiente. Per farlo, devi misurare le prestazioni in diversi orari e in diverse condizioni di carico. Quando monitori il servizio gestito per Apache Flink, vengono archiviati i dati di monitoraggio storici. In questo modo, puoi confrontare i dati con i dati sulle prestazioni correnti, identificare i normali modelli di prestazioni e le anomalie e ideare metodi per risolvere i problemi.

Argomenti

- [Accesso al servizio gestito per Apache Flink](#)
- [Monitoraggio in Managed Service per Apache Flink](#)
- [Configurare la registrazione delle applicazioni in Managed Service per Apache Flink](#)
- [Analizza i log con CloudWatch Logs Insights](#)
- [Metriche e dimensioni in Managed Service for Apache Flink](#)
- [Scrivi messaggi personalizzati in Logs CloudWatch](#)
- [Registra il servizio gestito per le chiamate API Apache Flink con AWS CloudTrail](#)

Accesso al servizio gestito per Apache Flink

La registrazione è importante per consentire alle applicazioni di produzione di comprendere errori e guasti. Tuttavia, il sottosistema di registrazione deve raccogliere e inoltrare le voci di registro ai CloudWatch registri. Sebbene alcune operazioni di registrazione siano buone e auspicabili, una registrazione estesa può sovraccaricare il servizio e causare ritardi nell'applicazione Flink. Registrare eccezioni e avvisi è sicuramente un'idea valida, ma non è possibile generare un messaggio di log per ogni messaggio elaborato dall'applicazione Flink. Flink è ottimizzato per una latenza elevata e bassa, mentre il sottosistema di registrazione non lo è. Nel caso in cui sia davvero necessario generare un output di registro per ogni messaggio elaborato, utilizzane un altro DataStream interno all'applicazione Flink e un sink appropriato per inviare i dati ad Amazon CloudWatch S3 o. Non utilizzare il sistema di registrazione Java per questo scopo. Inoltre, l'impostazione del Debug Monitoring Log Level del servizio gestito per Apache Flink genera una grande quantità di traffico, che può creare contropressione. Utilizzala solo mentre indaghi attivamente i problemi relativi all'applicazione.

Interroga i log con Logs Insights CloudWatch

CloudWatch Logs Insights è un potente servizio per interrogare i log su larga scala. Le sue funzionalità sono particolarmente indicate per i clienti che desiderano effettuare ricerche rapide nei log per individuare e mitigare gli errori durante gli eventi operativi.

La query seguente cerca le eccezioni in tutti i log del task manager e le ordina in base al momento in cui si sono verificate.

```
fields @timestamp, @message
| filter isPresent(throwableInformation.0) or isPresent(throwableInformation) or
  @message like /(Error|Exception)/
| sort @timestamp desc
```

Per altre query utili, consulta [Query di esempio](#).

Monitoraggio in Managed Service per Apache Flink

Quando esegui applicazioni di streaming in produzione, decidi di eseguire l'applicazione in modo continuo e indefinito. È fondamentale implementare il monitoraggio e i sistemi di avviso corretti di tutti i componenti, non solo dell'applicazione Flink, altrimenti rischi di non affrontare tempestivamente

i problemi emergenti e di accorgerti di un evento operativo solo quando ormai è troppo difficile da mitigare. Gli aspetti generali da monitorare includono:

- L'origine sta importando dati?
- I dati vengono letti dall'origine (dal punto di vista dell'origine)?
- L'applicazione Flink riceve dati?
- L'applicazione Flink è in grado di restare al passo o è in ritardo?
- L'applicazione Flink mantiene i dati nel sink (dal punto di vista dell'applicazione)?
- Il sink riceve dati?

Successivamente, è necessario prendere in considerazione parametri più specifici per l'applicazione Flink. Questa [CloudWatch dashboard](#) offre un buon punto di partenza. Per ulteriori informazioni sui parametri da monitorare per le applicazioni di produzione, consulta [Usa gli CloudWatch allarmi con Amazon Managed Service per Apache Flink](#). Tali parametri includono:

- `records_lag_max` e `millisbehindLatest`: se l'applicazione consuma da Kinesis o Kafka, questi parametri indicano se è in ritardo e deve essere dimensionata per restare al passo con il carico corrente. Si tratta di un parametro generico valido e facile da tracciare per tutti i tipi di applicazioni; tuttavia, può essere utilizzato solo per il dimensionamento reattivo, ovvero quando l'applicazione è già in ritardo.
- `CPUUtilization` `heapMemoryUtilization`: queste metriche forniscono una buona indicazione dell'utilizzo complessivo delle risorse dell'applicazione e possono essere utilizzate per la scalabilità proattiva, a meno che l'applicazione non sia vincolata all'I/O.
- `downtime`: un tempo di inattività maggiore di zero indica un errore dell'applicazione. Se il valore è maggiore di 0, l'applicazione non sta elaborando alcun dato.
- `lastCheckpointSize` e `lastCheckpointDuration`— Queste metriche monitorano la quantità di dati archiviati nello stato e il tempo necessario per superare un checkpoint. Se i checkpoint aumentano di dimensioni o richiedono molto tempo, l'applicazione dedica continuamente tempo a effettuare checkpoint e ha meno cicli per l'elaborazione effettiva dei dati. In alcuni punti, i checkpoint possono diventare troppo grandi o impiegare così tanto tempo da non funzionare. Oltre ai valori assoluti, i clienti dovrebbero considerare la possibilità di monitorare la frequenza di modifica con `RATE(lastCheckpointSize)` e `RATE(lastCheckpointDuration)`.
- `numberOfFailedCheckpoint`: questa metrica conta il numero di checkpoint non riusciti dall'avvio dell'applicazione. A seconda dell'applicazione, un malfunzionamento occasionale dei checkpoint può essere accettabile. Tuttavia, se i checkpoint non riescono regolarmente, è probabile che

l'applicazione non sia integra e che necessiti di ulteriore attenzione. Si consiglia di controllare che il parametro `RATE(numberOfFailedCheckpoints)` evidenzi problemi sul gradiente e non su valori assoluti.

Configurare la registrazione delle applicazioni in Managed Service per Apache Flink

Aggiungendo un'opzione di CloudWatch registrazione Amazon alla tua applicazione Managed Service for Apache Flink, puoi monitorare gli eventi dell'applicazione o i problemi di configurazione.

Questo argomento descrive come configurare l'applicazione per scrivere gli eventi dell'applicazione in un CloudWatch flusso di Logs. Un'opzione CloudWatch di registrazione è una raccolta di impostazioni e autorizzazioni dell'applicazione che l'applicazione utilizza per configurare il modo in cui scrive gli eventi dell'applicazione nei registri. CloudWatch È possibile aggiungere e configurare un'opzione CloudWatch di registrazione utilizzando AWS Management Console o (). AWS Command Line Interface AWS CLI

Tieni presente quanto segue sull'aggiunta di un'opzione CloudWatch di registrazione all'applicazione:

- Quando aggiungi un'opzione di CloudWatch registrazione utilizzando la console, Managed Service for Apache Flink crea automaticamente il gruppo di CloudWatch log e il flusso di log e aggiunge le autorizzazioni necessarie all'applicazione per scrivere nel flusso di log.
- Quando aggiungete un'opzione di CloudWatch registrazione utilizzando l'API, dovete anche creare il gruppo e il flusso di log dell'applicazione e aggiungere le autorizzazioni necessarie all'applicazione per scrivere nel flusso di log.

Configura la CloudWatch registrazione utilizzando la console

Quando abiliti CloudWatch la registrazione per la tua applicazione nella console, vengono creati automaticamente un CloudWatch gruppo di log e un flusso di log. Inoltre, la policy di autorizzazione dell'applicazione viene aggiornata con le autorizzazioni di scrittura sul flusso.

Managed Service for Apache Flink crea un gruppo di log denominato utilizzando la seguente convenzione, *ApplicationName* dov'è il nome dell'applicazione.

```
/aws/kinesis-analytics/ApplicationName
```

Il servizio gestito per Apache Flink crea un flusso di log nel nuovo gruppo di log con il nome seguente.

```
kinesis-analytics-log-stream
```

È possibile impostare il monitoraggio del livello dei parametri e il monitoraggio del livello di log utilizzando la sezione Monitoraggio del livello di log della pagina Configura applicazione. Per ulteriori informazioni sui livelli di log dell'applicazione, consulta [the section called “Controlla i livelli di monitoraggio delle applicazioni”](#).

Configurare la CloudWatch registrazione utilizzando la CLI

Per aggiungere un'opzione CloudWatch di registrazione utilizzando il AWS CLI, completa quanto segue:

- Crea un gruppo di CloudWatch log e un flusso di log.
- Aggiungi un'opzione di registrazione quando crei un'applicazione utilizzando l'[CreateApplication](#)azione o aggiungi un'opzione di registrazione a un'applicazione esistente utilizzando l'[AddApplicationCloudWatchLoggingOption](#)azione.
- Aggiungi alla policy dell'applicazione le autorizzazioni per scrivere sui log.

Crea un gruppo di CloudWatch log e un flusso di log

Puoi creare un gruppo di CloudWatch log e uno stream utilizzando la console CloudWatch Logs o l'API. Per informazioni sulla creazione di un gruppo di CloudWatch log e di un flusso di log, consulta [Lavorare con gruppi di log e flussi di log](#).

Utilizzate le opzioni di CloudWatch registrazione delle applicazioni

Utilizza le seguenti azioni API per aggiungere un'opzione di CloudWatch registro a un'applicazione nuova o esistente o modificare un'opzione di registro per un'applicazione esistente. Per ulteriori informazioni su come utilizzare un file JSON come input per un'operazione API, consulta [Codice di esempio dell'API Managed Service per Apache Flink](#).

Aggiungi un'opzione di CloudWatch registro durante la creazione di un'applicazione

L'esempio seguente mostra come utilizzare l'`CreateApplication`azione per aggiungere un'opzione di CloudWatch registro quando si crea un'applicazione. Nell'esempio, sostituire *Amazon Resource Name (ARN) of the CloudWatch Log stream to add to the*

new application con le tue informazioni. Per ulteriori informazioni sull'operazione, consulta [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:::amzn-s3-demo-bucket",
          "FileKey": "myflink.jar"
        }
      },
      "CodeContentType": "ZIPFILE"
    }
  },
  "CloudWatchLoggingOptions": [{
    "LogStreamARN": "<Amazon Resource Name (ARN) of the CloudWatch log stream to add to the new application>"
  }]
}
```

Aggiungere un'opzione di CloudWatch registro a un'applicazione esistente

L'esempio seguente mostra come utilizzare

l'AddApplicationCloudWatchLoggingOptionazione per aggiungere un'opzione di CloudWatch registro a un'applicazione esistente. Nell'esempio, sostituisci ciascuno *user input placeholder* con le tue informazioni. Per ulteriori informazioni sull'operazione, consulta [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>
}
```

Aggiorna un'opzione di CloudWatch registro esistente

L'esempio seguente mostra come utilizzare l'UpdateApplicationazione per modificare un'opzione di CloudWatch registro esistente. Nell'esempio, sostituisci ognuna *user input placeholder* con le tue informazioni. Per ulteriori informazioni sull'operazione, consulta [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>
}
```

Eliminare un'opzione di CloudWatch registro da un'applicazione

L'esempio seguente mostra come utilizzare l>DeleteApplicationCloudWatchLoggingOptionazione per eliminare un'opzione di CloudWatch registro esistente. Nell'esempio, sostituisci ognuna *user input placeholder* con le tue informazioni. Per ulteriori informazioni sull'operazione, consulta [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>
}
```

Imposta il livello di registrazione dell'applicazione

Per impostare il livello di registrazione dell'applicazione, utilizza il parametro [MonitoringConfiguration](#) dell'operazione [CreateApplication](#) o il parametro [MonitoringConfigurationUpdate](#) dell'operazione [UpdateApplication](#).

Per ulteriori informazioni sui livelli di log dell'applicazione, consulta [the section called “Controlla i livelli di monitoraggio delle applicazioni”](#).

Imposta il livello di registrazione dell'applicazione durante la creazione di un'applicazione

La seguente richiesta di esempio per l'operazione [CreateApplication](#) imposta il livello di log dell'applicazione su INFO.

```
{
  "ApplicationName": "MyApplication",
  "ApplicationDescription": "My Application Description",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
          "FileKey": "myflink.jar",
          "ObjectVersion": "AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "CodeContentType": "ZIPFILE"
    },
    "FlinkApplicationConfiguration": {
      "MonitoringConfiguration": {
        "ConfigurationType": "CUSTOM",
        "LogLevel": "INFO"
      }
    },
    "RuntimeEnvironment": "FLINK-1_15",
    "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole"
  }
}
```

Aggiornare il livello di registrazione dell'applicazione

La seguente richiesta di esempio per l'operazione [UpdateApplication](#) imposta il livello di log dell'applicazione su INFO.

```
{
  "ApplicationConfigurationUpdate": {
    "FlinkApplicationConfigurationUpdate": {
      "MonitoringConfigurationUpdate": {
        "ConfigurationTypeUpdate": "CUSTOM",
        "LogLevelUpdate": "INFO"
      }
    }
  }
}
```

```

    }
  }
}

```

Aggiungi le autorizzazioni per la scrittura nel flusso di log CloudWatch

Il servizio gestito per Apache Flink necessita delle autorizzazioni per scrivere errori di configurazione errata. CloudWatch È possibile aggiungere queste autorizzazioni al ruolo AWS Identity and Access Management (IAM) assunto da Managed Service for Apache Flink.

Per ulteriori informazioni sull'utilizzo di un ruolo IAM per il servizio gestito per Apache Flink, consulta [Identity and Access Management per il servizio gestito da Amazon per Apache Flink](#).

Policy di attendibilità

Per concedere al servizio gestito per Apache Flink le autorizzazioni per assumere un ruolo IAM, puoi collegare la seguente policy di attendibilità al ruolo di esecuzione del servizio.

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "kinesisanalytics.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

Policy delle autorizzazioni

Per concedere a un'applicazione le autorizzazioni per scrivere eventi di registro CloudWatch da una risorsa Managed Service for Apache Flink, puoi utilizzare la seguente politica di autorizzazioni IAM. Fornisci gli Amazon Resource Names (ARNs) corretti per il gruppo di log e lo stream.

```

{
  "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:*"
  ]
}

```

Controlla i livelli di monitoraggio delle applicazioni

Puoi controllare la generazione dei messaggi di log dell'applicazione utilizzando il Monitoraggio del livello dei parametri e il Monitoraggio del livello di log dell'applicazione.

Il monitoraggio del livello dei parametri dell'applicazione controlla la granularità dei messaggi di log. I livelli dei parametri di monitoraggio sono definiti nel modo seguente:

- Applicazione: i parametri si riferiscono all'intera applicazione.
- Attività: i parametri si riferiscono a ciascuna attività. Per ulteriori informazioni sulle attività, consulta [the section called “Implementa la scalabilità delle applicazioni”](#).
- Operatore: i parametri si riferiscono a ciascun operatore. Per ulteriori informazioni sugli operatori, consulta [the section called “Operatori”](#).
- Parallelismo: i parametri si riferiscono al parallelismo delle applicazioni. Puoi impostare questo livello di metrica solo utilizzando il [MonitoringConfigurationUpdate](#) parametro dell' [UpdateApplication](#) API. Non è possibile impostare questo livello dei parametri utilizzando la console. Per ulteriori informazioni sul parallelismo, consulta [the section called “Implementa la scalabilità delle applicazioni”](#).

Il monitoraggio del livello di log dell'applicazione controlla la verbosità del log dell'applicazione. I livelli di log di monitoraggio sono definiti nel modo seguente:

- **Errore:** eventi catastrofici potenziali dell'applicazione.
- **Avviso:** situazioni potenzialmente dannose dell'applicazione.
- **Informazioni:** eventi di errore informativi e transitori dell'applicazione. Ti consigliamo di utilizzare questo livello di registrazione.
- **Debug:** eventi informativi dettagliati che sono particolarmente utili per il debug di un'applicazione.
Nota: utilizza questo livello solo per scopi di debug temporanei.

Applica le migliori pratiche di registrazione

È consigliabile che l'applicazione utilizzi il livello di registrazione delle informazioni. Consigliamo questo livello per garantire la visualizzazione degli errori di Apache Flink, che vengono registrati a livello di informazioni anziché a livello di errore.

Ti consigliamo di utilizzare il livello di debug solo temporaneamente durante l'analisi dei problemi delle applicazioni. Torna al livello Informazioni quando il problema è stato risolto. L'utilizzo del livello di registrazione di debug influirà in modo significativo sulle prestazioni dell'applicazione.

Una registrazione eccessiva può inoltre influire in modo significativo sulle prestazioni dell'applicazione. Ti consigliamo, ad esempio, di non scrivere una voce di registro per ogni record elaborato. Una registrazione eccessiva può causare gravi rallentamenti nell'elaborazione dei dati e può portare a una contropressione nella lettura dei dati dalle origini.

Eseguire la risoluzione dei problemi di registrazione

Se i log delle applicazioni non vengono scritti nel flusso di log, verifica quanto segue:

- Verifica che il ruolo e le policy IAM dell'applicazione siano corretti. La policy dell'applicazione richiede le seguenti autorizzazioni per accedere al flusso di log:
 - `logs:PutLogEvents`
 - `logs:DescribeLogGroups`
 - `logs:DescribeLogStreams`

Per ulteriori informazioni, consulta [the section called “Aggiungi le autorizzazioni per la scrittura nel flusso di log CloudWatch”](#).

- Verifica che l'applicazione sia in esecuzione. Per verificare lo stato dell'applicazione, visualizza la pagina dell'applicazione nella console o utilizza le [ListApplications](#)azioni [DescribeApplication](#)o.

- Monitora CloudWatch le metriche, ad esempio downtime per diagnosticare altri problemi relativi alle applicazioni. Per informazioni sulla lettura delle CloudWatch metriche, consulta. [???](#)

Usa CloudWatch Logs Insights

Dopo aver abilitato CloudWatch la registrazione nell'applicazione, è possibile utilizzare CloudWatch Logs Insights per analizzare i log dell'applicazione. Per ulteriori informazioni, consulta [the section called “Analizza i log con CloudWatch Logs Insights”](#).

Analizza i log con CloudWatch Logs Insights

Dopo aver aggiunto un'opzione di CloudWatch registrazione all'applicazione come descritto nella sezione precedente, puoi utilizzare CloudWatch Logs Insights per interrogare i flussi di log alla ricerca di eventi o errori specifici.

CloudWatch Logs Insights consente di cercare e analizzare in modo interattivo i dati di registro in Logs. CloudWatch

Per informazioni su come iniziare a usare CloudWatch Logs Insights, consulta [Analizzare i dati di log con Logs Insights](#). CloudWatch

Esecuzione di una query di esempio

Questa sezione descrive come eseguire una query di esempio di CloudWatch Logs Insights.

Prerequisiti

- Gruppi di log e flussi di log esistenti configurati in CloudWatch Logs.
- Registri esistenti archiviati in Logs. CloudWatch

Se utilizzi servizi come AWS CloudTrail Amazon Route 53 o Amazon VPC, probabilmente hai già configurato i log di tali servizi in modo che accedano a Logs. CloudWatch [Per ulteriori informazioni sull'invio di log a Logs, consulta Getting CloudWatch Started with Logs](#). CloudWatch

Le query in CloudWatch Logs Insights restituiscono un insieme di campi degli eventi di registro o il risultato di un'aggregazione matematica o di altre operazioni eseguite sugli eventi di registro. Questo tutorial illustra una query che restituisce un elenco di eventi di log.

Per eseguire una query di esempio di Logs CloudWatch Insights

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nel riquadro di navigazione, seleziona Informazioni dettagliate.
3. L'editor di query della pagina Logs Insights contiene una query predefinita che restituisce gli ultimi 20 eventi di log. Sopra l'editor di query, seleziona un gruppo di log su cui eseguire la query.

Quando si seleziona un gruppo di log, CloudWatch Logs Insights rileva automaticamente i campi nei dati del gruppo di log e li visualizza nei Campi rilevati nel riquadro destro. Inoltre, visualizza un grafico a barre di eventi di log in questo gruppo di log nel tempo. Questo grafico a barre mostra la distribuzione di eventi nel gruppo di log che corrisponde alla query e all'intervallo di tempo, non solo gli eventi visualizzati nella tabella.

4. Scegli Esegui query.

Vengono visualizzati i risultati della query. In questo esempio, i risultati sono gli ultimi 20 eventi di log di qualsiasi tipo.

5. Per visualizzare tutti i campi di uno degli eventi di log restituiti, scegli la freccia a sinistra dell'evento di log.

Per ulteriori informazioni su come eseguire e modificare le query di CloudWatch Logs Insights, consulta [Eseguire e modificare](#) una query di esempio.

Esamina le query di esempio

Questa sezione contiene query di esempio di CloudWatch Logs Insights per l'analisi dei log delle applicazioni Managed Service for Apache Flink. Queste query cercano diversi esempi di condizioni di errore e fungono da modelli per scrivere query che trovano altre condizioni di errore.

Note

Sostituisci la regione (*us-west-2*), l'ID account (*012345678901*) e il nome dell'applicazione (*YourApplication*) nei seguenti esempi di query con la regione e l'ID account dell'applicazione.

Questo argomento contiene le sezioni seguenti:

- [Analizza le operazioni: distribuzione delle attività](#)
- [Analisi delle operazioni: modifica del parallelismo](#)
- [Analizza gli errori: accesso negato](#)
- [Analizza gli errori: fonte o sink non trovati](#)
- [Analizza gli errori: errori relativi alle attività dell'applicazione](#)

Analizza le operazioni: distribuzione delle attività

La seguente query di CloudWatch Logs Insights restituisce il numero di attività che Apache Flink Job Manager distribuisce tra i Task Manager. È necessario impostare l'intervallo di tempo della query in modo che corrisponda a un solo processo eseguito, in modo che la query non restituisca attività dai processi precedenti. Per ulteriori informazioni sui parametri, consulta [Implementa la scalabilità delle applicazioni](#).

```
fields @timestamp, message
| filter message like /Deploying/
| parse message " to flink-taskmanager-*" as @tmid
| stats count(*) by @tmid
| sort @timestamp desc
| limit 2000
```

La seguente query di CloudWatch Logs Insights restituisce le sottoattività assegnate a ciascun Task Manager. Il numero totale di sottoattività è la somma del parallelismo di ogni attività. Il parallelismo delle attività deriva dal parallelismo degli operatori ed è lo stesso del parallelismo dell'applicazione per impostazione predefinita, a meno che non ne modifichi il codice specificando `setParallelism`. Per ulteriori informazioni sull'impostazione del parallelismo degli operatori, consulta [Impostazione del parallelismo: livello dell'operatore](#) nella [documentazione di Apache Flink](#).

```
fields @timestamp, @tmid, @subtask
| filter message like /Deploying/
| parse message "Deploying * to flink-taskmanager-*" as @subtask, @tmid
| sort @timestamp desc
| limit 2000
```

Per ulteriori informazioni sulla pianificazione delle attività, consulta [Processi e pianificazione](#) nella [documentazione di Apache Flink](#).

Analisi delle operazioni: modifica del parallelismo

La seguente query di CloudWatch Logs Insights restituisce le modifiche al parallelismo di un'applicazione (ad esempio, dovute al ridimensionamento automatico). Questa query restituisce anche le modifiche manuali al parallelismo dell'applicazione. Per ulteriori informazioni sul dimensionamento automatico, consulta [the section called “Usa il ridimensionamento automatico”](#).

```
fields @timestamp, @parallelism
| filter message like /property: parallelism.default, /
| parse message "default, *" as @parallelism
| sort @timestamp asc
```

Analizza gli errori: accesso negato

La seguente query di CloudWatch Logs Insights restituisce i Access Denied log.

```
fields @timestamp, @message, @messageType
| filter applicationARN like /arn:aws:kinesisanalyticsus-
west-2:012345678901:application\YourApplication/
| filter @message like /AccessDenied/
| sort @timestamp desc
```

Analizza gli errori: fonte o sink non trovati

La seguente query di CloudWatch Logs Insights restituisce i ResourceNotFound log. ResourceNotFound log risultano se non viene trovato un codice sorgente o un sink Kinesis.

```
fields @timestamp,@message
| filter applicationARN like /arn:aws:kinesisanalyticsus-
west-2:012345678901:application\YourApplication/
| filter @message like /ResourceNotFoundException/
| sort @timestamp desc
```

Analizza gli errori: errori relativi alle attività dell'applicazione

La seguente query di CloudWatch Logs Insights restituisce i log degli errori relativi alle attività di un'applicazione. Questi log vengono generati se lo stato di un'applicazione passa da RUNNING a RESTARTING.

```
fields @timestamp,@message
```

```
| filter applicationARN like /arn:aws:kinesisanalyticsus-  
west-2:012345678901:application\YourApplication/  
| filter @message like /switched from RUNNING to RESTARTING/  
| sort @timestamp desc
```

Per le applicazioni che utilizzano Apache Flink versione 1.8.2 e precedenti, gli errori relativi alle attività comporteranno invece il passaggio dello stato dell'applicazione da RUNNING a FAILED. Quando utilizzi Apache Flink versione 1.8.2 e precedenti, la query seguente ti consente di cercare gli errori relativi alle attività dell'applicazione:

```
fields @timestamp,@message  
| filter applicationARN like /arn:aws:kinesisanalyticsus-  
west-2:012345678901:application\YourApplication/  
| filter @message like /switched from RUNNING to FAILED/  
| sort @timestamp desc
```

Metriche e dimensioni in Managed Service for Apache Flink

Quando il tuo Managed Service per Apache Flink elabora un'origine dati, Managed Service for Apache Flink riporta le seguenti metriche e dimensioni ad Amazon. CloudWatch

Parametri di applicazione

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
backPressuredTimeMsPerSecond*	Millisecondi	Il tempo (in millisecondi) in cui questa attività o questo operatore vengono sottoposti a contropressione al secondo.	Attività, operatore, parallelismo	*Disponibile solo per le applicazioni del servizio gestito per Apache Flink che eseguono la versione 1.13 di Flink. Questi parametri	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
				possono essere utili per identificare i rallentamenti in un'applicazione.	
busyTimeMsPerSecond*	Millisecondi	Il tempo (in millisecondi) occupato dall'attività o dall'operatore (né inattivo né in contropressione) al secondo. Può essere NaN, se il valore non può essere calcolato.	Attività, operatore, parallelismo	*Disponibile solo per le applicazioni del servizio gestito per Apache Flink che eseguono la versione 1.13 di Flink. Questi parametri possono essere utili per identificare i rallentamenti in un'applicazione.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
cpuUtilization	Percentuale	La percentuale complessiva di utilizzo della CPU tra i task manager. Ad esempio, se sono presenti cinque task manager, il servizio gestito per Apache Flink pubblica cinque esempi di questa metrica per intervallo di report.	Applicazione	È possibile utilizzare questa metrica per monitorare e l'utilizzo minimo, medio e massimo della CPU nell'applicazione. La CPUUtilization metrica tiene conto solo dell'utilizzo della CPU del processo TaskManager JVM in esecuzione all'interno del contenitore.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
containerCPUUtilization	Percentuale	Percentuale complessiva di utilizzo della CPU tra i container del task manager nel cluster di applicazioni Flink. Ad esempio, se sono presenti cinque task manager, corrispondenti sono cinque TaskManager contenuti in e Managed Service for Apache Flink pubblica 2* cinque esempi di questa metrica per intervallo di report di 1 minuto.	Applicazione	Il calcolo per container viene svolto come segue: Tempo totale della CPU (in secondi) consumato dal container * 100/Limit e CPU del contenitore (in / secondi) CPUs La CPUUtilization metrica tiene conto solo dell'utilizzo della CPU del processo TaskManager JVM in esecuzione all'interno del contenitore. Esistono altri componenti in esecuzione e all'esterno della JVM nello stesso container. Il	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
				parametro container CPUUtilization fornisce un quadro più completo, includendo tutti i processi in termini di esaurimento della CPU nel container e gli errori che ne derivano.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
containerMemoryUtilization	Percentuale	Percentuale complessiva di utilizzo della memoria tra i container del task manager nel cluster di applicazioni Flink. Ad esempio, se sono presenti cinque task manager, corrispondenti sono cinque TaskManager contenuti e Managed Service for Apache Flink pubblica 2* cinque esempi di questa metrica per intervallo di report di 1 minuto.	Applicazione	Il calcolo per container viene svolto come segue: Utilizzo della memoria del container (byte) * 100/ limite di memoria del container in base alle specifiche di implementazione del pod (in byte) Le ManagedMemoryUtilizations metriche HeapMemoryUtilization and tengono conto solo di parametri di memoria specifici come Heap Memory Usage of TaskManager JVM o	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
				Managed Memory (utilizzo della memoria al di fuori di JVM per processi nativi come RockSDB State Backend) . Il parametro container MemoryUtilization fornisce un quadro più completo, includendo la memoria del set di lavoro, che monitora meglio l'esaurimento della memoria totale. Una volta esaurito, si riaccenderà al pod. Out of Memory Error TaskManager	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
<code>containerDiskUtilization</code>	Percentuale	Percentuale complessiva di utilizzo del disco tra i container del task manager nel cluster di applicazioni Flink. Ad esempio, se ci sono cinque task manager, corrispondenti sono cinque TaskManager contenitori e Managed Service for Apache Flink pubblica 2* cinque esempi di questa metrica per intervallo di report di 1 minuto.	Applicazione	Il calcolo per container viene svolto come segue: Utilizzo del disco in byte * 100/limite del disco per il container in byte Per i container, rappresenta l'utilizzo del filesystem su cui è impostato il volume root del container.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
<code>currentInputWatermark</code>	Millisecondi	L'ultima filigrana che ha ricevuto application/operator/task/thread	Applicazione, operatore, attività, parallelo	Questo record viene generato solo per dimensioni con due input. Questo è il valore minimo degli ultimi watermark ricevuti.	
<code>currentOutputWatermark</code>	Millisecondi	L'ultima filigrana che ha emesso application/operator/task/thread	Applicazione, operatore, attività, parallelo		

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
downtime	Millisecondi	Per i processi attualmente in una situazione di guasto/ripristino, il tempo trascorso durante questa interruzione.	Applicazione	Questo parametro misura il tempo trascorso durante la mancata riuscita o il ripristino di un processo. Questo parametro restituisce 0 per i processi in esecuzione e -1 per i processi completati. Se questo parametro non è 0 o -1, significa che il processo di Apache Flink per l'applicazione non è stato eseguito.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
fullRestarts	Conteggio	Il numero totale di volte in cui questo processo è stato riavviato completamente da quando è stato inviato. Questo parametro non misura i riavvii granulari.	Applicazione	È possibile utilizzare questa metrica per valutare lo stato generale delle applicazioni. I riavvii possono verificarsi durante la manutenzione interna del servizio gestito per Apache Flink. Un numero di riavvii superiore al normale può indicare un problema con l'applicazione.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
heapMemoryUtilization	Percentuale	Utilizzo complessivo della memoria heap tra i task manager. Ad esempio, se sono presenti cinque task manager, il servizio gestito per Apache Flink pubblica cinque esempi di questa metrica per intervallo di report.	Applicazione	È possibile utilizzare questa metrica per monitorare e l'utilizzo minimo, medio e massimo della memoria heap nell'applicazione. Tiene conto HeapMemoryUtilization solo di metriche di memoria specifiche come Heap Memory Usage di JVM. TaskManager	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
<code>idleTimeMsPerSecond*</code>	Millisecondi	Il tempo (in millisecondi) di inattività (nessun dato da elaborare) di questa attività o di questo operatore al secondo. Il tempo di inattività esclude il tempo di contropressione, quindi se l'attività è in contropressione non è inattiva.	Attività, operatore, parallelismo	*Disponibile solo per le applicazioni del servizio gestito per Apache Flink che eseguono la versione 1.13 di Flink. Questi parametri possono essere utili per identificare i rallentamenti in un'applicazione.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
lastCheckpointSize	Byte	La dimensione totale dell'ultimo checkpoint	Applicazione	È possibile utilizzare questo parametro per determinare l'utilizzo dello storage delle applicazioni in esecuzione. Se il valore di questo parametro aumenta, ciò potrebbe indicare la presenza di un problema con l'applicazione, ad esempio una perdita di memoria o un collo di bottiglia.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
lastCheckpointDuration	Millisecondi	Il tempo impiegato per completare e l'ultimo checkpoint	Applicazione	Questo parametro misura il tempo impiegato per completare il checkpoint più recente. Se il valore di questo parametro aumenta, ciò potrebbe indicare la presenza di un problema con l'applicazione, ad esempio una perdita di memoria o un collo di bottiglia. In alcuni casi, è possibile risolvere questo problema disabilitando il checkpoint.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
managedMemoryUsed*	Byte	La quantità di memoria attualmente in uso.	Applicazione, operatore, attività, paralleli smo	*Disponibile solo per le applicazioni del servizio gestito per Apache Flink che eseguono la versione 1.13 di Flink. Si riferisce alla memoria gestita da Flink all'esterno dell'heap Java. Viene utilizzato per il backend di stato RocksDB ed è disponibile anche per le applicazioni.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
<code>managedMemoryTotal</code> *	Byte	La quantità totale di memoria gestita.	Applicazione, operatore, attività, paralleli smo	*Disponibile solo per le applicazioni del servizio gestito per Apache Flink che eseguono la versione 1.13 di Flink. Si riferisce alla memoria gestita da Flink all'esterno dell'heap Java. Viene utilizzato per il backend di stato RocksDB ed è disponibile anche per le applicazioni. Il parametro <code>ManagedMemoryUtilizations</code> tiene conto solo di parametri di memoria specifici come la memoria gestita (utilizzo della memoria al di fuori	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
				di JVM per processi nativi come RocksDB State Backend)	
managedMemoryUtilization*	Percentuale	managedMemoryUsedDerivato da/managedMemoryTotal	Applicazione, operatore, attività, parallelo	*Disponibile solo per le applicazioni del servizio gestito per Apache Flink che eseguono la versione 1.13 di Flink. Si riferisce alla memoria gestita da Flink all'esterno dell'heap Java. Viene utilizzato per il backend di stato RocksDB ed è disponibile anche per le applicazioni.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
numberOffFailedCheckpoints	Conteggio	Il numero di volte in cui il checkpoint non è andato a buon fine.	Applicazione	È possibile utilizzare questo parametro per monitorare lo stato e l'avanzamento delle applicazioni. I checkpoint potrebbero non riuscire a causa di problemi dell'applicazione, come problemi di throughput o di autorizzazioni.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
numRecordsIn*	Conteggio	Il numero totale di record ricevuti da questa applicazione, operatore o attività.	Applicazione, operatore, attività, parallelo	*Per applicare la statistica Somma su un periodo di tempo (secondi/minuto): • Seleziona il parametro al livello corretto. Se stai monitorando il parametro per un operatore, devi selezionare i parametri dell'operatore corrispondenti. • Poiché il servizio gestito per Apache Flink esegue 4 snapshot metrici al minuto, è necessario utilizzare	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
				il seguente parametro matematico: $m1/4$ dove $m1$ è la statistica Somma su un periodo (secondi/minuto) Il livello del parametro specifica se questo parametro misura il numero totale di record ricevuti dall'intera applicazione, da un operatore specifico o da un'attività specifica.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
numRecordsInPeriod*	Numero/secondo	Il numero totale di record ricevuti da questa applicazione, operatore o attività al secondo.	Applicazione, operatore, attività, parallelo	*Per applicare la statistica Somma su un periodo di tempo (secondi/minuto): • Seleziona il parametro al livello corretto. Se stai monitorando il parametro per un operatore, devi selezionare i parametri dell'operatore corrispondenti. • Poiché il servizio gestito per Apache Flink esegue 4 snapshot metrici al minuto, è necessario utilizzare	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
				il seguente parametro matematico: $m1/4$ dove $m1$ è la statistica Somma su un periodo (secondi/minuto) Il livello del parametro specifica se questo parametro misura il numero totale di record ricevuti dall'intera applicazione, da un operatore specifico o da un'attività specifica.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
numRecordsOut*	Conteggio	Il numero totale di record generati da questa applicazione, operatore o attività.	Applicazione, operatore, attività, parallelo	*Per applicare la statistica Somma su un periodo di tempo (secondi/minuto): • Seleziona il parametro al livello corretto. Se stai monitorando il parametro per un operatore, devi selezionare i parametri dell'operatore corrispondenti. • Poiché il servizio gestito per Apache Flink esegue 4 snapshot metrici al minuto, è necessari o utilizzare	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
				il seguente parametro matematico: $m1/4$ dove $m1$ è la statistica Somma su un periodo (secondi/minuto) Il livello del parametro specifica se questo parametro misura il numero totale di record ricevuti dall'intera applicazione, da un operatore specifico o da un'attività specifica.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
numLateRecordsDropped*	Conteggio	Applicazione, operatore, attività, paralleli smo		*Per applicare la statistica Somma su un periodo di tempo (secondi/ minuto): • Seleziona il parametro al livello corretto. Se stai monitorando il parametro per un operatore , devi selezionare i parametri dell'operatore corrispondenti. • Poiché il servizio gestito per Apache Flink esegue 4 snapshot metrici al minuto, è necessari o utilizzare	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
				il seguente parametro matematico: $m1/4$ dove $m1$ è la statistica Somma su un periodo (secondi/minuto) Il numero di record di questo operatore o attività è diminuito a causa dell'arrivo in ritardo.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
numRecordsOutPerSecond*	Numero/secondo	Il numero totale di record generati da questa applicazione, operatore o attività al secondo.	Applicazione, operatore, attività, parallelo	*Per applicare la statistica Somma su un periodo di tempo (secondi/minuto): • Seleziona il parametro al livello corretto. Se stai monitorando il parametro per un operatore, devi selezionare i parametri dell'operatore corrispondenti. • Poiché il servizio gestito per Apache Flink esegue 4 snapshot metrici al minuto, è necessario utilizzare	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
				il seguente parametro matematico: $m1/4$ dove $m1$ è la statistica Somma su un periodo (secondi/minuto) Il livello del parametro specifica se questo parametro misura il numero totale di record ricevuti dall'intera applicazione, da un operatore specifico o da un'attività specifica.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
<code>oldGenerationGCCount</code>	Conteggio	Il numero totale di vecchie operazioni di rimozione di oggetti inutili (garbage collection) che si sono verificate in tutti i task manager.	Applicazione		
<code>oldGenerationGCTime</code>	Millisecondi	Il tempo totale impiegato per eseguire le vecchie operazioni di rimozione di oggetti inutili (garbage collection).	Applicazione	È possibile utilizzare questo parametro per monitorare la somma, la media e il tempo massimo di rimozione di oggetti inutili (garbage collection).	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
threadCount	Conteggio	Il numero totale di thread live utilizzati dall'applicazione.	Applicazione	Questo parametro misura il numero di thread utilizzati dal codice dell'applicazione. È diverso dal parallelismo dell'applicazione.	
uptime	Millisecondi	Il tempo in cui il processo è stato eseguito senza interruzioni.	Applicazione	È possibile utilizzare questo parametro per determinare se un processo viene eseguito correttamente. Questo parametro restituisce -1 per i processi completati.	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
KPUs *	Conteggio	Il numero totale di KPUs dati utilizzati dall'applicazione.	Applicazione	*Questa metrica riceve un campione per periodo di fatturazione (un'ora). Per visualizzare il numero di interruzioni KPUs nel tempo, usa MAX o AVG per un periodo di almeno un'ora (1). Il conteggio delle KPU include le KPU. orchestration Per ulteriori informazioni, consulta Managed Service for Apache Flink Pricing.	

Metriche del connettore Kinesis Data Streams

AWS emette tutti i record per Kinesis Data Streams oltre ai seguenti:

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
<code>millisbehindLatest</code>	Millisecondi	Il numero di millisecondi in cui il consumatore si trova rispetto all'estremità del flusso, a indicare il ritardo rispetto all'ora corrente del consumatore.	Applicazione (per Stream), Parallelismo (per) <code>ShardId</code>	- Un valore di 0 indica che l'elaborazione dei record è aggiornata e che non sono presenti nuovi record da elaborare in questo momento. Il parametro di una particolare partizione può essere specificato in base al nome del flusso e all'ID della partizione. - Il valore -1 indica che il servizio non ha ancora riportato un valore per il parametro.
<code>bytesRequestedPerFetch</code>	Byte	I byte richiesti in una singola chiamata a <code>getRecords</code> .	Applicazione (per Stream), Parallelismo (per) <code>ShardId</code>	

Metriche del connettore Amazon MSK

AWS emette tutti i record per Amazon MSK oltre ai seguenti:

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
currentoffsets	N/D	L'offset di lettura corrente del consumer, per ogni partizione. Il parametro di una particolare partizione può essere specificato in base al nome dell'argomento e all'ID della partizione.	Applicazione (per argomento), parallelismo (per) PartitionId	
commitsFailed	N/D	Il numero totale di errori di commit di offset su Kafka, se il commit di offset e la creazione di checkpoint sono abilitati.	Applicazione, operatore, attività, parallelismo	Restituire le compensazioni a Kafka è solo un modo per esporre i progressi dei consumer, quindi un errore di commit non pregiudica l'integrità degli offset delle partizioni bloccati di Flink.
commitsSuccessful	N/D	Il numero totale di commit di offset riusciti	Applicazione, operatore,	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
		verso Kafka, se il commit di offset e la creazione di checkpoint sono abilitati.	attività, parallelo	
committed offsets	N/D	Gli ultimi offset eseguiti con successo su Kafka, per ogni partizione. Il parametro di una particolare partizione può essere specificato in base al nome dell'argomento e all'ID della partizione.	Applicazione (per argomento), parallelismo (per) PartitionId	
records_lag_max	Conteggio	Il ritardo massimo in termini di numero di record per ogni partizione e in questa finestra	Applicazione, operatore, attività, parallelo	
bytes_consumed_rate	Byte	Il numero medio di byte consumati al secondo per un argomento	Applicazione, operatore, attività, parallelo	

Metriche di Apache Zeppelin

Per i notebook Studio, AWS emette le seguenti metriche a livello di applicazione:,,,
e. KPIs `cpuUtilization` `heapMemoryUtilization` `oldGenerationGCTime` `oldGenerationGCCount` `threadCount` Inoltre, genera i parametri mostrati nella tabella seguente, anche a livello di applicazione.

Parametro	Unità	Descrizione	Nome Prometheus
<code>zeppelinCpuUtilization</code>	Percentuale	Percentuale complessiva di utilizzo della CPU nel server Apache Zeppelin.	<code>process_cpu_usage</code>
<code>zeppelinHeapMemoryUtilization</code>	Percentuale	Percentuale complessiva di utilizzo della memoria heap per il server Apache Zeppelin.	<code>jvm_memory_used_bytes</code>
<code>zeppelinThreadCount</code>	Conteggio	Il numero totale di thread live utilizzati dal server Apache Zeppelin.	<code>jvm_threads_live_threads</code>
<code>zeppelinWaitingJobs</code>	Conteggio	Il numero di processi di Apache Zeppelin in coda in attesa di un thread.	<code>jetty_threads_jobs</code>
<code>zeppelinServerUptime</code>	Secondi	Il tempo totale in cui il server è stato attivo e in funzione.	<code>process_uptime_seconds</code>

CloudWatch Visualizza le metriche

Puoi visualizzare i CloudWatch parametri per la tua applicazione utilizzando la CloudWatch console Amazon o il AWS CLI.

Per visualizzare le metriche utilizzando la console CloudWatch

1. Apri la CloudWatch console all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nel pannello di navigazione, seleziona Parametri.
3. Nel riquadro CloudWatch Metriche per categoria per Managed Service for Apache Flink, scegli una categoria di metriche.
4. Nel riquadro superiore, scorri verso il basso per visualizzare l'elenco completo dei parametri.

Per visualizzare le metriche utilizzando il AWS CLI

- Al prompt dei comandi utilizza il comando seguente.

```
aws cloudwatch list-metrics --namespace "AWS/KinesisAnalytics" --region region
```

Imposta i livelli di CloudWatch report sulle metriche

Puoi controllare il livello dei parametri dell'applicazione creati dall'applicazione. Il servizio gestito per Apache Flink supporta i seguenti livelli di parametri:

- Applicazione: l'applicazione riporta solo il livello più elevato dei parametri per ogni applicazione. I parametri del servizio gestito per Apache Flink vengono pubblicati al livello di Applicazione per impostazione predefinita.
- Attività: l'applicazione riporta le dimensioni dei parametri specifiche dell'attività per i parametri definiti con il livello di report del parametro Attività, ad esempio il numero di record in entrata e in uscita dall'applicazione al secondo.
- Operatore: l'applicazione riporta le dimensioni dei parametri specifiche dell'operatore per i parametri definiti con il livello di report del parametro Operatore, ad esempio i parametri per ogni operazione di filtro o mappa.

- **Parallelismo:** l'applicazione riporta i livelli parametri Task e Operator per ciascun thread di esecuzione. Tale livello di report non è consigliato per applicazioni con un parallelismo superiore a 64 a causa di costi eccessivi.

Note

È necessario utilizzare questo livello di parametri solo per la risoluzione dei problemi a causa della quantità di dati relativi ai parametri generati dal servizio. È possibile impostare questo livello di parametri solo utilizzando la CLI. Questo livello di parametri non è disponibile nella console.

Il livello predefinito è Applicazione. L'applicazione riporta i parametri al livello corrente e a tutti i livelli superiori. Ad esempio, se il livello di reporting è impostato su Operatore l'applicazione riporta i parametri Applicazione, Attività e Operatore.

Puoi impostare il livello di segnalazione delle CloudWatch metriche utilizzando il `MonitoringConfiguration` parametro dell'[CreateApplication](#) azione o il `MonitoringConfigurationUpdate` parametro dell'[UpdateApplication](#) azione. L'esempio seguente di richiesta per l'[UpdateApplication](#) azione imposta il livello di segnalazione delle CloudWatch metriche su Task:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 4,
  "ApplicationConfigurationUpdate": {
    "FlinkApplicationConfigurationUpdate": {
      "MonitoringConfigurationUpdate": {
        "ConfigurationTypeUpdate": "CUSTOM",
        "MetricsLevelUpdate": "TASK"
      }
    }
  }
}
```

Puoi anche configurare il livello di registrazione utilizzando il parametro `LogLevel` dell'operazione [CreateApplication](#) o il parametro `LogLevelUpdate` dell'operazione [UpdateApplication](#). Puoi utilizzare i seguenti livelli di log:

- **ERROR:** registra gli eventi di errore potenzialmente recuperabili.

- **WARN:** registra gli eventi di avviso che potrebbero causare un errore.
- **INFO:** registra gli eventi informativi.
- **DEBUG:** registra gli eventi di debug generali.

Per ulteriori informazioni sui livelli di registrazione di Log4j, consulta [Livelli di registro personalizzati](#) nella documentazione di [Apache Log4j](#).

Usa metriche personalizzate con Amazon Managed Service per Apache Flink

Managed Service for Apache Flink espone 19 metriche a, incluse le metriche relative all'utilizzo delle risorse e al throughput CloudWatch. Inoltre, puoi creare parametri personalizzati per tenere traccia dei dati specifici dell'applicazione, come l'elaborazione di eventi o l'accesso a risorse esterne.

Questo argomento contiene le sezioni seguenti:

- [Come funziona](#)
- [Visualizza esempi per la creazione di una classe di mappatura](#)
- [Visualizza metriche personalizzate](#)

Come funziona

I parametri personalizzati nel servizio gestito per Apache Flink utilizzano il sistema delle metriche di Apache Flink. I parametri di Apache Flink hanno i seguenti attributi:

- **Tipo:** il tipo di un parametro descrive il modo in cui vengono misurati e riportati i dati. I tipi di parametri Apache Flink disponibili includono contatore, misuratore e istogramma. Per ulteriori informazioni sui tipi di parametri Apache Flink, consulta [Tipi di parametri](#).

Note

AWS CloudWatch Metrics non supporta il tipo di metrica Histogram Apache Flink. CloudWatch può visualizzare solo le metriche Apache Flink dei tipi Count, Gauge e Meter.

- **Ambito:** l'ambito di una metrica è costituito dal relativo identificatore e da un insieme di coppie chiave-valore che indicano come verrà riportata la metrica. CloudWatch L'identificatore di un parametro è costituito dai seguenti elementi:

- Un ambito di sistema, che indica il livello al quale viene riportato il parametro (ad esempio Operatore).
- Un ambito utente, che definisce attributi come le variabili utente o i nomi dei gruppi di parametri. Questi attributi sono definiti utilizzando [MetricGroup.addGroup\(key, value\)](#) o [MetricGroup.addGroup\(name\)](#).

Per ulteriori informazioni sugli ambiti, consulta [Ambito](#).

Per ulteriori informazioni sulle metriche di Apache Flink, consulta [Parametri](#) nella [documentazione di Apache Flink](#).

Per creare una metrica personalizzata nel servizio gestito per Apache Flink, puoi accedere al sistema metrico Apache Flink da qualsiasi funzione utente che estende `RichFunction` tramite chiamata [GetMetricGroup](#). Questo metodo restituisce un [MetricGroup](#) oggetto che puoi utilizzare per creare e registrare metriche personalizzate. Managed Service for Apache Flink riporta tutte le metriche create con la chiave di gruppo `to.KinesisAnalytics CloudWatch` I parametri personalizzati che definisci hanno le seguenti caratteristiche:

- Il parametro personalizzato ha un nome parametro e un nome gruppo. [Questi nomi devono essere composti da caratteri alfanumerici secondo le regole di denominazione di Prometheus](#).
- Gli attributi definiti nell'ambito dell'utente (ad eccezione del gruppo di `KinesisAnalytics` metriche) vengono pubblicati come dimensioni. `CloudWatch`
- Per impostazione predefinita, i parametri personalizzati vengono pubblicati a livello di `Application`.
- Le dimensioni (attività/operatore/parallelismo) vengono aggiunte al parametro in base al livello di monitoraggio dell'applicazione. Il livello di monitoraggio dell'applicazione viene impostato utilizzando il [MonitoringConfiguration](#) parametro dell'[CreateApplication](#) azione o il [MonitoringConfigurationUpdate](#) parametro o dell'[UpdateApplication](#) azione.

Visualizza esempi per la creazione di una classe di mappatura

I seguenti esempi di codice mostrano come creare una classe di mappatura che crea e incrementa una metrica personalizzata e come implementare la classe di mappatura nell'applicazione aggiungendola a un oggetto. `DataStream`

Record count (metrica personalizzata)

Il seguente esempio di codice mostra come creare una classe di mappatura che crea un parametro che conta i record in un flusso di dati (la stessa funzionalità del parametro `numRecordsIn`):

```
private static class NoOpMapperFunction extends RichMapFunction<String, String> {
    private transient int valueToExpose = 0;
    private final String customMetricName;

    public NoOpMapperFunction(final String customMetricName) {
        this.customMetricName = customMetricName;
    }

    @Override
    public void open(Configuration config) {
        getRuntimeContext().getMetricGroup()
            .addGroup("KinesisAnalytics")
            .addGroup("Program", "RecordCountApplication")
            .addGroup("NoOpMapperFunction")
            .gauge(customMetricName, (Gauge<Integer>) () -> valueToExpose);
    }

    @Override
    public String map(String value) throws Exception {
        valueToExpose++;
        return value;
    }
}
```

Nell'esempio precedente, la variabile `valueToExpose` viene incrementata per ogni record elaborato dall'applicazione.

Dopo aver definito la classe di mappatura, crea un flusso all'interno dell'applicazione che implementa la mappa:

```
DataStream<String> noopMapperFunctionAfterFilter =
    kinesisProcessed.map(new NoOpMapperFunction("FilteredRecords"));
```

Per il codice completo di questa applicazione, consulta [Applicazione del parametro personalizzato del numero di record](#).

Mettrica personalizzata per il conteggio delle parole

Il seguente esempio di codice mostra come creare una classe di mappatura che crea una metrica che conta le parole in un flusso di dati:

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

Nell'esempio precedente, la variabile `counter` viene incrementata per ogni parola elaborata dall'applicazione.

Dopo aver definito la classe di mappatura, crea un flusso all'interno dell'applicazione che implementa la mappa:

```
// 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());
```

Per il codice completo di questa applicazione, consulta [Applicazione del parametro personalizzato del conteggio di parole](#).

Visualizza metriche personalizzate

Le metriche personalizzate per la tua applicazione vengono visualizzate nella console CloudWatch Metrics nel AWS/KinesisAnalyticsdashboard, nel gruppo di metriche dell'applicazione.

Usa gli CloudWatch allarmi con Amazon Managed Service per Apache Flink

Utilizzando gli allarmi Amazon CloudWatch Metric, controlla una CloudWatch metrica per un periodo di tempo specificato. L'allarme esegue una o più operazioni basate sul valore del parametro o espressione relativa a una soglia su un certo numero di periodi. Un esempio di operazione sta inviando una notifica a un argomento Amazon Simple Notification Service (Amazon SNS).

Per ulteriori informazioni sugli CloudWatch allarmi, consulta [Using Amazon CloudWatch Alarms](#).

Controlla gli allarmi consigliati

Questa sezione contiene gli allarmi consigliati per il monitoraggio delle applicazioni del servizio gestito per Apache Flink.

La tabella descrive gli allarmi consigliati e contiene le seguenti colonne:

- **Espressione di parametro:** il parametro o l'espressione di parametro da testare rispetto alla soglia.
- **Statistica:** la statistica utilizzata per controllare il parametro, ad esempio Media.
- **Soglia:** l'utilizzo di questo allarme richiede la determinazione di una soglia che definisca il limite delle prestazioni previste dell'applicazione. È necessario determinare questa soglia monitorando l'applicazione in condizioni normali.
- **Descrizione:** cause che potrebbero attivare questo allarme e possibili soluzioni per la condizione.

Espressioni di parametro	Statistic	Threshold	Descrizione
<code>downtime > 0</code>	Media	0	Un tempo di inattività maggiore di zero indica che l'applicazione non è riuscita. Se il valore è maggiore di 0, l'applicazione non sta elaborando alcun dato. Consigliato per tutte le applicazioni. La Downtime metrica misura la durata di un'interruzione. Un tempo di inattività maggiore di zero indica che l'applicazione non è riuscita. Per la risoluzione dei problemi, vedere L'applicazione si sta riavviando .
<code>RATE (numberOfFailedCheckpoints) > 0</code>	Media	0	Questa metrica conta il numero di checkpoint non riusciti dall'avvio dell'applicazione. A seconda dell'applicazione, un malfunzionamento occasionale dei checkpoint può essere accettabile. Tuttavia, se i checkpoint non riescono regolarmente

Espressioni di parametro	Statistic	Threshold	Descrizione
			nte, è probabile che l'applicazione non sia integra e che necessiti di ulteriore attenzione. Consiglia mo di monitorare RATE (numberOfFailedCheckpoints) per generare allarmi sul gradiente e non sui valori assoluti. Consigliato per tutte le applicazioni. Utilizza questa metrica per monitorare lo stato delle applicazioni e l'avanzamento del checkpoint. L'applicazione salva i dati sullo stato nei checkpoint quando è integra. Il checkpoint può fallire a causa di timeout se l'applicazione non sta procedendo nell'elaborazione dei dati di input. Per la risoluzione dei problemi, vedere. Il checkpoint è in fase di interruzione

Espressioni di parametro	Statistic	Threshold	Descrizione
<code>Operator.numRecord</code> <code>sOutPerSecond</code> < soglia	Media	Il numero minimo di record emessi dall'applicazione in condizioni normali.	Consigliato per tutte le applicazioni. Un calo al di sotto di questa soglia può indicare che l'applicazione non sta facendo i progressi previsti sui dati di input. Per la risoluzione dei problemi, vedere La velocità effettiva è troppo lenta.

Espressioni di parametro	Statistic	Threshold	Descrizione
<code>records_lag_max millisbehindLatest > soglia</code>	Massimo	La latenza massima prevista in condizioni normali.	Se l'applicazione utilizza Kinesis o Kafka, queste metriche indicano se l'applicazione è in ritardo e deve essere ridimensionata per stare al passo con il carico corrente. Si tratta di un parametro generico valido e facile da tracciare per tutti i tipi di applicazioni; tuttavia, può essere utilizzato solo per il dimensionamento reattivo, ovvero quando l'applicazione è già in ritardo. Consigliato per tutte le applicazioni. Usa la <code>records_lag_max</code> metrica per una sorgente Kafka o la <code>millisbehindLatest</code> una sorgente di flusso Kinesis. Il superamento di questa soglia può indicare che l'applicazione non sta facendo i progressi previsti

Espressioni di parametro	Statistic	Threshold	Descrizione
			sui dati di input. Per la risoluzione dei problemi, vedere La velocità effettiva è troppo lenta.

Espressioni di parametro	Statistic	Threshold	Descrizione
<code>lastCheckpointDuration</code> > soglia	Massimo	La durata massima prevista del checkpoint in condizioni normali.	Monitora la quantità di dati archiviati nello stato e il tempo necessario per completarlo e un checkpoint. Se i checkpoint aumentano di dimensioni o richiedono molto tempo, l'applicazione dedica continuamente tempo a effettuare checkpoint e ha meno cicli per l'elaborazione effettiva dei dati. In alcuni punti, i checkpoint possono diventare troppo grandi o impiegare così tanto tempo da non funzionare. Oltre ai valori assoluti, i clienti dovrebbero considerare la possibilità di monitorare la frequenza di modifica con <code>RATE(lastCheckpointSize)</code> e <code>RATE(lastCheckpointDuration)</code> .

Espressioni di parametro	Statistic	Threshold	Descrizione
			) . Se il valore aumenta <code>lastCheckpointDuration</code> continuamente, il superamento di questa soglia può indicare che l'applicazione non sta facendo i progressi previsti sui dati di input o che vi sono problemi di integrità dell'applicazione, come la contropressione. Per la risoluzione dei problemi, vedere Crescita statale illimitata .

Espressioni di parametro	Statistic	Threshold	Descrizione
<code>lastCheckpointSize > soglia</code>	Massimo	La dimensione massima prevista del checkpoint in condizioni normali.	Monitora la quantità di dati archiviati nello stato e il tempo necessari o per completarlo e un checkpoint. Se i checkpoint aumentano di dimensioni o richiedono molto tempo, l'applicazione dedica continuamente tempo a effettuare checkpoint e ha meno cicli per l'elaborazione effettiva dei dati. In alcuni punti, i checkpoint possono diventare troppo grandi o impiegare così tanto tempo da non funzionare. Oltre ai valori assoluti, i clienti dovrebbero considerare la possibilità di monitorare la frequenza di modifica con <code>RATE(lastCheckpointSize)</code> e <code>RATE(lastCheckpointDuration)</code> .

Espressioni di parametro	Statistic	Threshold	Descrizione
			) . Se il valore aumenta <code>lastCheckpointSize</code> continuamente, il superamento di questa soglia può indicare che l'applicazione sta accumulando dati sullo stato. Se i dati sullo stato diventano troppo grandi, l'applicazione può esaurire la memoria durante il ripristino da un checkpoint, oppure il ripristino da un checkpoint potrebbe richiedere troppo tempo. Per la risoluzione dei problemi, vedere. Crescita statale illimitata

Espressioni di parametro	Statistic	Threshold	Descrizione
heapMemoryUtilization > soglia	Massimo	Ciò fornisce una buona indicazione dell'utilizzo complessivo delle risorse dell'applicazione e può essere utilizzato per la scalabilità proattiva, a meno che l'applicazione non sia vincolata all'I/O. La heapMemoryUtilization dimensione massima prevista in condizioni normali, con un valore consigliato del 90 per cento.	È possibile utilizzare questa metrica per monitorare l'utilizzo o massimo della memoria da parte dei task manager nell'applicazione. Se l'applicazione raggiunge questa soglia, è necessario fornire più risorse. A tale scopo, è necessario abilitare il ridimensionamento automatico o aumentare il parallelismo dell'applicazione. Per ulteriori informazioni sull'aumento delle risorse, vedere. Implementa la scalabilità delle applicazioni

Espressioni di parametro	Statistic	Threshold	Descrizione
<code>cpuUtilization</code> > soglia	Massimo	Ciò fornisce una buona indicazione dell'utilizzo complessivo delle risorse dell'applicazione e può essere utilizzato per la scalabilità proattiva, a meno che l'applicazione non sia vincolata all'I/O. La <code>cpuUtilization</code> dimensione massima prevista in condizioni normali, con un valore consigliato dell'80%.	È possibile utilizzare questa metrica per monitorare l'utilizzo massimo della CPU dei task manager nell'applicazione. Se l'applicazione raggiunge questa soglia, è necessario fornire più risorse. A tale scopo, è necessario abilitare il ridimensionamento automatico o aumentare il parallelismo dell'applicazione. Per ulteriori informazioni sull'aumento delle risorse, vedere. Implementa la scalabilità delle applicazioni

Espressioni di parametro	Statistic	Threshold	Descrizione
<code>threadsCount > soglia</code>	Massimo	La <code>threadsCount</code> dimensione massima prevista in condizioni normali.	È possibile utilizzare questa metrica per verificare eventuali perdite di thread nei task manager dell'applicazione. Se questa metrica raggiunge questa soglia, controllate il codice dell'applicazione per verificare e se i thread vengono creati senza essere chiusi.
<code>(oldGarbageCollectTime * 100)/60_000 over 1 min period') > soglia</code>	Massimo	La <code>oldGarbageCollectTime</code> durata massima prevista. Si consiglia di impostare una soglia in modo che il tempo di raccolta dei rifiuti tipico sia pari al 60 per cento della soglia specificata, ma la soglia corretta per l'applicazione può variare.	Se questa metrica aumenta continuamente, ciò può indicare la presenza di una perdita di memoria nei task manager dell'applicazione.

Espressioni di parametro	Statistic	Threshold	Descrizione
<code>RATE(oldGarbageCollectionCount) > soglia</code>	Massimo	Il massimo previsto di <code>oldGarbageCollectionCount</code> in condizioni normali. La soglia corretta per la tua candidatura può variare.	Se questa metrica aumenta continuamente, ciò può indicare la presenza di una perdita di memoria nei task manager dell'applicazione.
<code>Operator.currentOutputWatermark - Operator.currentInputWatermark > soglia</code>	Minimo	L'incremento minimo previsto della filigrana in condizioni normali. La soglia corretta per la tua candidatura può variare.	Se questa metrica aumenta continuamente, ciò può indicare che l'applicazione sta elaborando eventi sempre più vecchi o che un'attività secondaria a monte non invia una filigrana da un periodo di tempo sempre più lungo.

Scrivi messaggi personalizzati in Logs CloudWatch

Puoi scrivere messaggi personalizzati nel registro dell'applicazione Managed Service for Apache Flink. CloudWatch Puoi farlo utilizzando la libreria [log4j](#) di Apache o la libreria [Simple Logging Facade for Java \(SLF4J\)](#).

Argomenti

- [Scrivi nei CloudWatch log usando Log4J](#)
- [Scrivi nei log usando CloudWatch J SLF4](#)

Scrivi nei CloudWatch log usando Log4J

1. Aggiungi le dipendenze seguenti al file `pom.xml` dell'applicazione:

```
<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. Includi l'oggetto dalla libreria:

```
import org.apache.logging.log4j.Logger;
```

3. Crea un'istanza dell'oggetto `Logger`, inserendo la classe dell'applicazione:

```
private static final Logger log =
  LogManager.getLogger.getLogger(YourApplicationClass.class);
```

4. Scrivi nel log utilizzando `log.info`. Un gran numero di messaggi viene scritto nel log dell'applicazione. Per rendere i messaggi personalizzati più facili da filtrare, utilizza il livello di log `INFO` dell'applicazione.

```
log.info("This message will be written to the application's CloudWatch log");
```

L'applicazione scrive un record nel log con un messaggio simile al seguente:

```
{
  "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"
```

```
}
```

Scrivi nei log usando CloudWatch J SLF4

1. Aggiungi le dipendenze seguenti al file `pom.xml` dell'applicazione:

```
<dependency>
  <groupId>org.slf4j</groupId>
  <artifactId>slf4j-log4j12</artifactId>
  <version>1.7.7</version>
  <scope>runtime</scope>
</dependency>
```

2. Includi gli oggetti della libreria:

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

3. Crea un'istanza dell'oggetto `Logger`, inserendo la classe dell'applicazione:

```
private static final Logger log =
  LoggerFactory.getLogger(YourApplicationClass.class);
```

4. Scrivi nel log utilizzando `log.info`. Un gran numero di messaggi viene scritto nel log dell'applicazione. Per rendere i messaggi personalizzati più facili da filtrare, utilizza il livello di log `INFO` dell'applicazione.

```
log.info("This message will be written to the application's CloudWatch log");
```

L'applicazione scrive un record nel log con un messaggio simile al seguente:

```
{
  "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"
```

```
}
```

Registra il servizio gestito per le chiamate API Apache Flink con AWS CloudTrail

Il servizio gestito per Apache Flink è integrato con AWS CloudTrail, un servizio che fornisce un registro delle azioni intraprese da un utente, ruolo o un AWS servizio in Managed Service for Apache Flink. CloudTrail acquisisce tutte le chiamate API per Managed Service for Apache Flink come eventi. Le chiamate acquisite includono le chiamate dalla console del servizio gestito per Apache Flink e le chiamate di codice al servizio gestito per le operazioni dell'API del servizio gestito per Apache Flink. Se crei un trail, puoi abilitare la distribuzione continua di CloudTrail eventi a un bucket Amazon S3, inclusi gli eventi per Managed Service for Apache Flink. Se non configuri un percorso, puoi comunque visualizzare gli eventi più recenti nella CloudTrail console nella cronologia degli eventi. Utilizzando le informazioni raccolte da CloudTrail, è possibile determinare la richiesta effettuata a Managed Service for Apache Flink, l'indirizzo IP da cui è stata effettuata la richiesta, chi ha effettuato la richiesta, quando è stata effettuata e dettagli aggiuntivi.

Per ulteriori informazioni CloudTrail, consulta la Guida per l'[AWS CloudTrail utente](#).

Informazioni su Managed Service for Apache Flink in CloudTrail

CloudTrail è abilitato sul tuo AWS account al momento della creazione dell'account. Quando si verifica un'attività in Managed Service for Apache Flink, tale attività viene registrata in un CloudTrail evento insieme ad altri eventi di AWS servizio nella cronologia degli eventi. Puoi visualizzare, cercare e scaricare gli eventi recenti nel tuo AWS account. Per ulteriori informazioni, consulta [Visualizzazione degli eventi con la cronologia degli CloudTrail eventi](#).

Per una registrazione continua degli eventi nel tuo AWS account, inclusi gli eventi per Managed Service for Apache Flink, crea un percorso. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Per impostazione predefinita, quando crei un percorso nella console, il percorso si applica a tutte le AWS regioni. Il trail registra gli eventi di tutte le regioni della AWS partizione e consegna i file di log al bucket Amazon S3 specificato. Inoltre, puoi configurare altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei log. CloudTrail Per ulteriori informazioni, consulta gli argomenti seguenti:

- [Panoramica della creazione di un trail](#)
- [CloudTrail Servizi e integrazioni supportati](#)

- [Configurazione delle notifiche Amazon SNS per CloudTrail](#)
- [Ricezione di file di CloudTrail registro da più regioni](#) e [ricezione di file di CloudTrail registro da più account](#)

Tutte le azioni di Managed Service for Apache Flink vengono registrate CloudTrail e documentate nel riferimento all'API [Managed Service for Apache Flink](#). Ad esempio, le chiamate alle [UpdateApplication](#) azioni [CreateApplication](#) e generano voci nei file di registro. CloudTrail

Ogni evento o voce di log contiene informazioni sull'utente che ha generato la richiesta. Le informazioni di identità consentono di determinare quanto segue:

- Se la richiesta è stata effettuata con credenziali utente root o AWS Identity and Access Management (IAM).
- Se la richiesta è stata effettuata con le credenziali di sicurezza temporanee per un ruolo o un utente federato.
- Se la richiesta è stata effettuata da un altro AWS servizio.

Per ulteriori informazioni, consulta [Elemento CloudTrail userIdentity](#).

Comprendi le voci dei file di registro di Managed Service for Apache Flink

Un trail è una configurazione che consente la distribuzione di eventi come file di log in un bucket Amazon S3 specificato dall'utente. CloudTrail i file di registro contengono una o più voci di registro. Un evento rappresenta una singola richiesta proveniente da qualsiasi fonte e include informazioni sull'azione richiesta, la data e l'ora dell'azione, i parametri della richiesta e così via. CloudTrail i file di registro non sono una traccia ordinata dello stack delle chiamate API pubbliche, quindi non vengono visualizzati in un ordine specifico.

L'esempio seguente mostra una voce di CloudTrail registro che illustra le azioni [AddApplicationCloudWatchLoggingOption](#) e [DescribeApplication](#).

```
{
  "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": "kinesisanalytics.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": "kinesisanalytics.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"
    }
}
```

Ottimizza le prestazioni in Amazon Managed Service per Apache Flink

Questo argomento descrive le tecniche per monitorare e migliorare le prestazioni dell'applicazione del servizio gestito per Apache Flink.

Argomenti

- [Risolvi i problemi relativi alle prestazioni](#)
- [Utilizza le migliori pratiche in materia di prestazioni](#)
- [Monitora le prestazioni](#)

Risolvi i problemi relativi alle prestazioni

Questa sezione contiene un elenco di caratteristiche la cui verifica permette di diagnosticare e risolvere i problemi relativi alle prestazioni.

Se l'origine dati è un flusso Kinesis, i problemi di prestazioni in genere si presentano nella forma di una metrica `millisBehindLatest` elevata o in crescita. Per altre fonti, è consigliabile controllare una metrica simile che rappresenta il ritardo nella lettura dalla fonte.

Comprendi il percorso dei dati

Quando analizzi un problema relativo alle prestazioni dell'applicazione, considera l'intero percorso compiuto dai dati. I seguenti componenti dell'applicazione possono ridurre l'efficacia delle prestazioni e creare una congestione, se non sono progettati o predisposti correttamente:

- Fonti e destinazioni dei dati: assicurati che le risorse esterne con cui interagisce l'applicazione siano adeguatamente fornite per la velocità effettiva dell'applicazione.
- Dati sullo stato: è fondamentale assicurarsi che l'applicazione non interagisca in maniera troppo frequente con l'archivio degli stati precedenti.

È possibile ottimizzare il serializzatore utilizzato dall'applicazione. Il serializzatore Kryo predefinito può gestire qualsiasi tipo serializzabile, tuttavia è possibile utilizzare un serializzatore più performante se l'applicazione memorizza solo dati in tipi POJO. Per informazioni sui serializzatori Apache Flink, consulta [Tipi di dati e serializzazione](#) nella documentazione di Apache Flink.

- Operatori: la logica di business implementata dagli operatori non deve essere eccessivamente complicata; è meglio evitare, inoltre, che ogni record elaborato crei o utilizzi risorse. Assicurati inoltre che l'applicazione non crei finestre scorrevoli o a cascata con troppa frequenza.

Soluzioni per la risoluzione dei problemi

Questa sezione contiene potenziali soluzioni ai problemi relativi alle prestazioni.

Argomenti

- [CloudWatch livelli di monitoraggio](#)
- [Metrica della CPU dell'applicazione](#)
- [Parallelismo delle applicazioni](#)
- [Registrazione dell'applicazione](#)
- [Parallelismo dell'operatore](#)
- [Logica dell'applicazione](#)
- [Memoria dell'applicazione](#)

CloudWatch livelli di monitoraggio

Verificare che i livelli di CloudWatch monitoraggio non siano impostati su un'impostazione troppo dettagliata.

L'impostazione Debug Monitoraggio del livello dei log genera una grande quantità di traffico, che può creare una congestione. È consigliabile utilizzarla unicamente quando si esaminano in maniera attiva i problemi relativi all'applicazione.

Se l'applicazione ha un'impostazione Parallelism elevata, l'utilizzo del Parallelism livello delle metriche di monitoraggio genererà analogamente una grande quantità di traffico, che può portare a una congestione. È consigliabile utilizzare questo livello di metriche solo quando Parallelism per l'applicazione è a un livello ridotto o durante l'analisi di problemi relativi all'applicazione.

Per ulteriori informazioni, consulta [Controlla i livelli di monitoraggio delle applicazioni](#).

Metrica della CPU dell'applicazione

Controlla la metrica CPU dell'applicazione. Se è superiore al 75%, è possibile consentire all'applicazione di allocare una quantità superiore di risorse a sé stessa, abilitando il dimensionamento automatico.

Quando il dimensionamento automatico è abilitato e l'utilizzo della CPU è superiore al 75% per 15 minuti, l'applicazione alloca una quantità superiore di risorse. Per ulteriori informazioni sul dimensionamento è possibile consultare le sezioni [Gestire correttamente il dimensionamento](#) e [Implementa la scalabilità delle applicazioni](#).

Note

Le applicazioni vengono dimensionate automaticamente solo in risposta all'utilizzo della CPU. L'applicazione non si ridimensiona automaticamente in risposta ad altre metriche di sistema, come `heapMemoryUtilization`. Se il livello di utilizzo dell'applicazione per altre metriche è elevato, è consigliabile aumentarne manualmente il parallelismo.

Parallelismo delle applicazioni

Consente di aumentare il parallelismo dell'applicazione. Il parallelismo dell'applicazione viene aggiornato utilizzando il `ParallelismConfigurationUpdate` parametro dell'azione.

[UpdateApplication](#)

Il massimo KPU per un'applicazione è 64 per impostazione predefinita e può essere aumentato richiedendo un aumento del limite.

È importante inoltre assegnare il parallelismo a ciascun operatore in base al relativo carico di lavoro, anziché limitarsi ad aumentare il parallelismo dell'applicazione. Leggi [Parallelismo dell'operatore](#) quanto segue.

Registrazione dell'applicazione

Controlla se l'applicazione registra un log per ogni record elaborato. La creazione di un log per ogni record provocherà gravi rallentamenti nell'elaborazione dei dati durante i periodi in cui l'applicazione presenta una velocità di trasmissione effettiva elevata. Per verificare questa condizione, interroga i log alla ricerca delle voci di registro create dall'applicazione per ogni nuovo record che elabora. Per ulteriori informazioni relative all'analisi dei log delle applicazioni, consulta [the section called “Analizza i log con CloudWatch Logs Insights”](#).

Parallelismo dell'operatore

Verifica che il carico di lavoro dell'applicazione sia distribuito in modo uniforme tra i processi di lavoro.

Per informazioni sull'ottimizzazione del carico di lavoro degli operatori dell'applicazione, consulta.

[Dimensionamento degli operatori](#)

Logica dell'applicazione

Esamina la logica dell'applicazione per individuare eventuali azioni poco efficienti o dalle prestazioni ridotte, come l'accesso a una dipendenza esterna (un database o un servizio web, per esempio), l'accesso allo stato dell'applicazione, ecc. Una dipendenza esterna può inoltre limitare le prestazioni, se non è efficiente o non è accessibile in modo affidabile, il che può causare errori HTTP 500 ricorrenti nella restituzione della dipendenza esterna.

L'impiego dell'IO asincrono può rivelarsi utile se l'applicazione si avvale di una dipendenza esterna per l'arricchimento o l'elaborazione dei dati in entrata. Per ulteriori informazioni, consulta [Async I/O](#) nella [documentazione di Apache Flink](#).

Memoria dell'applicazione

Verifica l'assenza di dispersione delle risorse all'interno dell'applicazione. Se l'applicazione non elimina correttamente i thread o la memoria, è possibile che si verifichi un picco o un aumento graduale dei parametri `millisBehindLatest`, `CheckpointSize` e `CheckpointDuration`. Tale situazione può causare, inoltre, errori nel task manager o nel job manager.

Utilizza le migliori pratiche in materia di prestazioni

Questa sezione descrive considerazioni particolari relative alla progettazione di un'applicazione che garantisca prestazioni elevate.

Gestire correttamente il dimensionamento

Questa sezione contiene informazioni sulla gestione del dimensionamento a livello di applicazione e di operatore.

Questa sezione contiene i seguenti argomenti:

- [Gestire correttamente il dimensionamento dell'applicazione](#)

- [Gestire correttamente il dimensionamento degli operatori](#)

Gestire correttamente il dimensionamento dell'applicazione

È possibile utilizzare il dimensionamento automatico per gestire picchi imprevisti nell'attività dell'applicazione. L'applicazione KPU aumenterà automaticamente se vengono soddisfatti i seguenti criteri:

- il dimensionamento automatico è abilitato per l'applicazione;
- l'utilizzo della CPU rimane superiore al 75% per 15 minuti.

Se la scalabilità automatica è abilitata, ma l'utilizzo della CPU non rimane al di sotto di questa soglia, l'applicazione non verrà scalata. KPU Se si verifica un picco nell'utilizzo della CPU che non soddisfa questa soglia o un picco in una metrica di utilizzo diversa (ad esempio `heapMemoryUtilization`) aumenta manualmente il ridimensionamento, per consentire all'applicazione di gestire al meglio i picchi di attività.

Note

Se l'applicazione ha aggiunto automaticamente più risorse tramite il dimensionamento automatico, l'applicazione rilascerà le nuove risorse dopo un periodo di inattività. Il ridimensionamento delle risorse influirà temporaneamente sulle prestazioni.

Per ulteriori informazioni sul dimensionamento, consulta [Implementa la scalabilità delle applicazioni](#).

Gestire correttamente il dimensionamento degli operatori

È possibile migliorare le prestazioni dell'applicazione verificando che il carico di lavoro dell'applicazione sia distribuito in modo uniforme tra i processi di lavoro e che gli operatori dell'applicazione dispongano delle risorse di sistema necessarie per garantire stabilità e ottime prestazioni.

È possibile impostare il parallelismo per ogni operatore nel codice dell'applicazione utilizzando l'impostazione `parallelism`. Nel caso in cui il parallelismo per un operatore non sia impostato, verrà utilizzata l'impostazione del parallelismo a livello di applicazione. Gli operatori che utilizzano l'impostazione del parallelismo a livello di applicazione possono potenzialmente utilizzare tutte le risorse di sistema disponibili per l'applicazione, rendendola poco stabile.

Per determinare al meglio il parallelismo per ogni operatore, è fondamentale considerare i requisiti di risorse relativi dell'operatore rispetto agli altri operatori dell'applicazione. È consigliabile impostare gli operatori che richiedono più risorse su un'impostazione di parallelismo degli operatori più elevata rispetto agli operatori che ne richiedono di meno.

Il parallelismo totale dell'operatore per l'applicazione è la somma del parallelismo per tutti gli operatori dell'applicazione. È possibile ottimizzare il parallelismo totale degli operatori per l'applicazione determinando il migliore rapporto tra tale parallelismo e il totale degli slot di attività disponibili per l'applicazione. Generalmente, un rapporto stabile tra il parallelismo totale dell'operatore e gli slot di attività è 4:1, ovvero l'applicazione ha uno slot di attività disponibile per ogni quattro attività secondarie dell'operatore disponibili. Un'applicazione con operatori che richiedono più risorse può richiedere un rapporto di 3:1 o 2:1, mentre un'applicazione con operatori che richiedono meno risorse può essere stabile anche con un rapporto di 10:1.

È possibile impostare il rapporto per l'operatore utilizzando [Usa le proprietà di runtime](#), in modo da ottimizzare il parallelismo dell'operatore senza compilare e caricare il codice dell'applicazione.

L'esempio di codice seguente mostra il procedimento per impostare il parallelismo dell'operatore come rapporto regolabile del parallelismo dell'applicazione attuale:

```
Map<String, Properties> applicationProperties =
    KinesisAnalyticsRuntime.getApplicationProperties();
operatorParallelism =
    StreamExecutionEnvironment.getParallelism() /
    Integer.getInteger(

    applicationProperties.get("OperatorProperties").getProperty("MyOperatorParallelismRatio")
    );
```

Per informazioni sulle attività secondarie, gli slot di attività e altre risorse dell'applicazione, consulta [Consulta le risorse dell'applicazione Managed Service for Apache Flink](#).

Per controllare la distribuzione del carico di lavoro tra i processi di lavoro dell'applicazione, utilizza l'impostazione `Parallelism` e il metodo di partizione `keyBy`. Per ulteriori informazioni, consulta gli argomenti seguenti nella [documentazione Apache Flink](#):

- [Esecuzione parallela](#)
- [DataStream Trasformazioni](#)

Monitoraggio dell'utilizzo delle risorse di dipendenza esterne

Se si verifica un rallentamento delle prestazioni in una destinazione (come Kinesis Streams, Firehose, DynamoDB o Service), l'applicazione subirà una contropressione. OpenSearch Verifica che le dipendenze esterne siano state predisposte correttamente per la velocità di trasmissione effettiva dell'applicazione.

Note

I malfunzionamenti in altri servizi possono causare errori all'interno dell'applicazione. Se riscontri errori nella tua applicazione, controlla i log dei servizi di destinazione per individuare eventuali errori. CloudWatch

Avviamento dell'applicazione Apache Flink in locale

Per risolvere i problemi di memoria, è consigliabile avviare l'applicazione in un'installazione locale di Flink. In tal modo sarà possibile accedere a strumenti di debug, come lo stack trace e gli heap dump, che non possono essere impiegati quando si esegue l'applicazione nel servizio gestito per Apache Flink.

Per informazioni sulla creazione di un'installazione locale di Flink, consulta [Standalone](#) nella documentazione di Apache Flink.

Monitora le prestazioni

Questa sezione descrive gli strumenti per il monitoraggio delle prestazioni di un'applicazione.

Monitora le prestazioni utilizzando le CloudWatch metriche

Monitora l'utilizzo delle risorse, la velocità effettiva, il checkpoint e i tempi di inattività dell'applicazione utilizzando le metriche. CloudWatch Per informazioni sull'utilizzo delle CloudWatch metriche con l'applicazione Managed Service for Apache Flink, consulta. [???](#)

Monitora le prestazioni utilizzando CloudWatch registri e allarmi

È possibile monitorare le condizioni di errore che potrebbero causare problemi di prestazioni utilizzando CloudWatch Logs.

Le condizioni di errore vengono visualizzate nei log quando lo status del processo di Apache Flink cambia da RUNNING a FAILED.

Gli CloudWatch allarmi vengono utilizzati per creare notifiche relative a problemi di prestazioni, come l'utilizzo delle risorse o le metriche dei checkpoint che superano una soglia di sicurezza o modifiche impreviste dello stato dell'applicazione.

Per informazioni sulla creazione di CloudWatch allarmi per un'applicazione Managed Service for Apache Flink, consulta. [???](#)

Servizio gestito per Apache Flink e quota di notebook Studio

Note

Le versioni 1.6, 1.8 e 1.11 di Apache Flink non sono supportate dalla community di Apache Flink da oltre tre anni. Ora prevediamo di terminare il supporto per queste versioni in Amazon Managed Service for Apache Flink. A partire dal 5 novembre 2024, non potrai creare nuove applicazioni per queste versioni di Flink. Puoi continuare a eseguire le applicazioni esistenti in questo momento.

Per tutte le regioni ad eccezione delle regioni della Cina e AWS GovCloud (US) Regions, dal 5 febbraio 2025, non sarà più possibile creare, avviare o eseguire applicazioni utilizzando queste versioni di Apache Flink in Amazon Managed Service for Apache Flink.

Per le regioni della Cina e il AWS GovCloud (US) Regions, non sarà più possibile creare, avviare o eseguire applicazioni utilizzando queste versioni di Apache Flink in Amazon Managed Service for Apache Flink.

Puoi aggiornare le tue applicazioni in modo statico utilizzando la funzionalità di aggiornamento immediato delle versioni di Managed Service for Apache Flink. Per ulteriori informazioni, consulta [Usa gli aggiornamenti di versione sul posto per Apache Flink](#).

Quando lavori con un servizio gestito da Amazon per Apache Flink, tieni presente la quota seguente:

- Puoi creare fino a 100 applicazioni Managed Service for Apache Flink per regione nel tuo account. Puoi creare un caso per richiedere applicazioni aggiuntive tramite il modulo di aumento del limite di quote di servizio. Per ulteriori informazioni, consulta [Centro di Supporto AWS](#).

Per un elenco delle regioni che supportano il servizio gestito per Apache Flink, consulta [Regioni ed endpoint del servizio gestito per Apache Flink](#).

- Per impostazione predefinita, il numero di unità di elaborazione Kinesis (KPU) è limitato a 64. Per istruzioni su come richiedere un aumento di questa quota, consulta [Per richiedere un aumento della quota in Service Quotas](#). Assicurati di specificare il prefisso dell'applicazione a cui deve essere applicato il nuovo limite di KPU.

Con Managed Service for Apache Flink, all' AWS account vengono addebitate le risorse allocate, anziché le risorse utilizzate dall'applicazione. Ti viene addebitata una tariffa oraria basata sul numero massimo di quelle utilizzate per eseguire l' KPU applicazione di elaborazione dello stream. Una singola KPU offre 1 vCPU e 4 GB di memoria. Per ogni KPU, il servizio fornisce anche 50 GB di spazio di archiviazione delle applicazioni in esecuzione.

- È possibile creare fino a 1.000 istantanee di Managed Service for Apache Flink per applicazione. Per ulteriori informazioni, consulta [Gestisci i backup delle applicazioni utilizzando le istantanee](#).
- Puoi assegnare fino a 50 tag per applicazione.
- La dimensione massima per un file JAR dell'applicazione è di 512 MB. Se questo limite viene superato, l'applicazione non verrà avviata.

Per i notebook Studio, si applicano le quote seguenti. Per richiedere una quota superiore, [crea un caso di supporto](#).

- `websocketMessageSize` = 5 MiB
- `noteSize` = 5 MiB
- `noteCount` = 1.000
- Max cumulative UDF size = 100 MiB
- Max cumulative dependency jar size = 300 MiB

Gestisci le attività di manutenzione per Managed Service for Apache Flink

Managed Service for Apache Flink corregge periodicamente le applicazioni con aggiornamenti di sicurezza del sistema operativo e dell'immagine del contenitore per mantenere la conformità e raggiungere gli obiettivi di sicurezza. AWS Una finestra di manutenzione per un'applicazione Managed Service for Apache Flink è una finestra temporale di 8 ore durante la quale Managed Service for Apache Flink esegue attività di manutenzione delle applicazioni su un'applicazione. La manutenzione potrebbe iniziare in giorni diversi a seconda della pianificazione del Region AWS team di assistenza. Consulta la tabella nella sezione seguente per le finestre temporali di manutenzione.

Come parte della procedura di manutenzione, l'applicazione Managed Service for Apache Flink verrà riavviata. Ciò causa un periodo di inattività da 10 a 30 secondi durante la finestra di manutenzione dell'applicazione. La durata effettiva del periodo di inattività dipende dallo stato, dalle dimensioni e dalla frequenza dello snapshot/checkpoint dell'applicazione. Per informazioni su come ridurre al minimo l'impatto di questo periodo di inattività, consulta [the section called “Tolleranza agli errori: checkpoint e savepoint”](#). Puoi scoprire se Managed Service for Apache Flink ha eseguito un'azione di manutenzione sulla tua applicazione utilizzando l'API. `ListApplicationOperations` Per ulteriori informazioni, consulta [Identificare quando è stata effettuata la manutenzione dell'applicazione](#).

Finestre temporali di manutenzione in Regioni AWS

Regione AWS	Finestra temporale di manutenzione
AWS GovCloud (Stati Uniti occidentali)	06:00 - 14:00 UTC
AWS GovCloud (Stati Uniti orientali)	03:00 - 11:00 UTC
Stati Uniti orientali (Virginia settentrionale)	03:00 - 11:00 UTC
Stati Uniti orientali (Ohio)	03:00 - 11:00 UTC
Stati Uniti occidentali (California settentrionale)	06:00 - 14:00 UTC
Stati Uniti occidentali (Oregon)	06:00 - 14:00 UTC
Asia Pacifico (Hong Kong)	13:00 - 21:00 UTC
Asia Pacifico (Mumbai)	16:30 - 00:30 UTC

Regione AWS	Finestra temporale di manutenzione
Asia Pacifico (Hyderabad)	16:30 - 00:30 UTC
Asia Pacifico (Seoul)	13:00 - 21:00 UTC
Asia Pacifico (Singapore)	14:00 - 22:00 UTC
Asia Pacifico (Sydney)	12:00 - 20:00 UTC
Asia Pacifico (Giacarta)	15:00 - 23:00 UTC
Asia Pacifico (Tokyo)	13:00 - 21:00 UTC
Canada (Centrale)	03:00 - 11:00 UTC
Cina (Pechino)	13:00 - 21:00 UTC
Cina (Ningxia)	13:00 - 21:00 UTC
Europa (Francoforte)	06:00 - 14:00 UTC
Europa (Zurigo)	20:00 - 04:00 UTC
Europa (Irlanda)	22:00 - 06:00 UTC
Europa (Londra)	22:00 - 06:00 UTC
Europa (Stoccolma)	23:00 - 07:00 UTC
Europa (Milano)	21:00 - 05:00 UTC
Europa (Spagna)	21:00 - 05:00 UTC
Africa (Città del Capo)	20:00 - 04:00 UTC
Europa (Irlanda)	22:00 - 06:00 UTC
Europa (Londra)	23:00 - 07:00 UTC
Europa (Parigi)	23:00 - 07:00 UTC

Regione AWS	Finestra temporale di manutenzione
Europa (Stoccolma)	23:00 - 07:00 UTC
Medio Oriente (Bahrein)	13:00 - 21:00 UTC
Medio Oriente (Emirati Arabi Uniti)	18:00 - 02:00 UTC
Sud America (San Paolo)	19:00 - 03:00 UTC
Israele (Tel Aviv)	20:00 - 04:00 UTC

Scegli una finestra di manutenzione

Managed Service for Apache Flink ti avvisa dei prossimi eventi di manutenzione pianificata tramite e-mail e notifiche. AWS Health In Managed Service for Apache Flink, puoi modificare l'ora del giorno in cui inizia la manutenzione utilizzando l'`UpdateApplicationMaintenanceConfigurationAPI` e aggiornando la configurazione della finestra di manutenzione. Per ulteriori informazioni, consulta [UpdateApplicationMaintenanceConfiguration](#). Managed Service for Apache Flink utilizza la configurazione di manutenzione aggiornata la prossima volta che pianifica la manutenzione dell'applicazione. Se si richiama questa operazione dopo che il servizio ha già pianificato la manutenzione, il servizio applica l'aggiornamento della configurazione la prossima volta che pianifica la manutenzione dell'applicazione.

Note

Per fornire il massimo livello di sicurezza possibile, Managed Service for Apache Flink non supporta alcuna eccezione per rinunciare alla manutenzione, sospendere la manutenzione o eseguire la manutenzione in giorni specifici.

Identifica quando è avvenuta la manutenzione della tua applicazione

Puoi scoprire se Managed Service for Apache Flink ha eseguito un'azione di manutenzione sulla tua applicazione utilizzando l'`ListApplicationOperationsAPI`.

Di seguito è riportato un esempio di richiesta `ListApplicationOperations` che può aiutarti a filtrare l'elenco per la manutenzione dell'applicazione:

```
{  
  "ApplicationName": "MyApplication",  
  "operation": "ApplicationMaintenance"  
}
```


Raggiungi la prontezza di produzione per il tuo servizio gestito per le applicazioni Apache Flink

Questa è una raccolta di aspetti importanti sull'esecuzione di applicazioni di produzione sul servizio gestito da Amazon per Apache Flink. Non si tratta di un elenco esaustivo, ma piuttosto del minimo indispensabile a cui prestare attenzione prima di inserire un'applicazione in produzione.

Effettua il test di caricamento delle tue applicazioni

Alcuni problemi con le applicazioni si manifestano solo in condizioni di carico elevato. Abbiamo assistito a casi in cui le applicazioni sembravano funzionanti, eppure un evento operativo ha notevolmente amplificato il carico sull'applicazione. Ciò può avvenire in modo completamente indipendente dall'applicazione stessa. Se l'origine dati o il data sink non sono disponibili per un paio d'ore, l'applicazione Flink non può fare progressi. Una volta risolto il problema, si accumula un arretrato di dati non elaborati, che può esaurire completamente le risorse disponibili. Il carico può quindi amplificare bug o problemi di prestazioni che non erano mai emersi prima.

È quindi essenziale eseguire test di carico adeguati per le applicazioni di produzione. Le domande a cui è necessario rispondere durante questi test di carico includono:

- L'applicazione è stabile in presenza di carichi elevati e sostenuti?
- L'applicazione può ancora registrare un savepoint in caso di picco di carico?
- Quanto tempo è necessario per elaborare un backlog di 1 ora? E quanto tempo è necessario per 24 ore (a seconda della conservazione massima dei dati nello stream)?
- La velocità di trasmissione effettiva dell'applicazione aumenta quando l'applicazione viene dimensionata?

Quando si utilizzano dati da un flusso di dati, questi scenari possono essere simulati immettendoli nel flusso per un certo periodo di tempo. Quindi avviate l'applicazione e fate in modo che consumi i dati dall'inizio del tempo. Ad esempio, utilizzate una posizione iniziale di `TRIM_HORIZON` nel caso di un flusso di dati Kinesis.

Definisci il parallelismo massimo

Il parallelismo massimo definisce il parallelismo massimo a cui un'applicazione che ha uno stato può dimensionarsi. Viene definito quando lo stato viene creato per la prima volta e non è possibile dimensionare l'operatore oltre questo valore massimo senza eliminare lo stato.

Il parallelismo massimo viene impostato quando lo stato viene creato per la prima volta.

Per impostazione predefinita, il parallelismo massimo è impostato su:

- 128, se il parallelismo è ≤ 128
- $\text{MIN}(\text{nextPowerOfTwo}(\text{parallelism} + (\text{parallelism} / 2)), 2^{15})$: se il parallelismo è > 128

Se intendete scalare l'applicazione con un parallelismo superiore a 128, dovrete definire esplicitamente il parallelismo massimo.

È possibile definire il parallelismo massimo a livello di applicazione, con o con un solo operatore. `env.setMaxParallelism(x)` Salvo diversa indicazione, tutti gli operatori ereditano il parallelismo Max dell'applicazione.

Per ulteriori informazioni, vedere [Impostazione del parallelismo massimo nella documentazione](#) di Apache Flink.

Impostare un UUID per tutti gli operatori

Un UUID viene utilizzato nell'operazione in cui Flink mappa un savepoint a un singolo operatore. L'impostazione di un UUID specifico per ciascun operatore fornisce una mappatura stabile per il processo di ripristino del savepoint.

```
.map(...).uid("my-map-function")
```

Per maggiori informazioni, consulta [Checklist di preparazione per la produzione](#).

Mantenimento delle best practice per il servizio gestito per le applicazioni Apache Flink

Questa sezione contiene informazioni e consigli per lo sviluppo di un servizio gestito stabile e performante per applicazioni Apache Flink.

Argomenti

- [Riduci al minimo le dimensioni di uber JAR](#)
- [Tolleranza agli errori: checkpoint e savepoint](#)
- [Versioni di connettori non supportate](#)
- [Prestazioni e parallelismo](#)
- [Impostazione del parallelismo per operatore](#)
- [Registrazione](#)
- [Codifica](#)
- [Credenziali root.](#)
- [Lettura da fonti con pochi shard/partizioni](#)
- [Intervallo di aggiornamento del notebook Studio](#)
- [Prestazioni ottimali del notebook Studio](#)
- [Come le strategie di watermark e le partizioni inattive influiscono sulle finestre temporali](#)
- [Impostare un UUID per tutti gli operatori](#)
- [Aggiungi ServiceResourceTransformer al plugin Maven shade](#)

Riduci al minimo le dimensioni di uber JAR

Java/Scala application must be packaged in an uber (super/fat) JAR e include tutte le dipendenze aggiuntive richieste che non sono già fornite dal runtime. Tuttavia, la dimensione di uber JAR influisce sui tempi di avvio e riavvio dell'applicazione e può far sì che il JAR superi il limite di 512 MB.

Per ottimizzare i tempi di implementazione, il tuo uber JAR non dovrebbe includere quanto segue:

- Eventuali dipendenze fornite dal runtime, come illustrato nell'esempio seguente. Dovrebbero avere un `provided` ambito nel file POM o `compileOnly` nella configurazione di Gradle.

- Qualsiasi dipendenza utilizzata solo per i test, ad esempio JUnit Mockito. Dovrebbero avere un `test` ambito nel file POM o `testImplementation` nella configurazione di Gradle.
- Eventuali dipendenze non effettivamente utilizzate dall'applicazione.
- Qualsiasi dato o metadato statico richiesto dall'applicazione. I dati statici devono essere caricati dall'applicazione in fase di esecuzione, ad esempio da un datastore o da Amazon S3.
- Consulta questo [file di esempio POM](#) per i dettagli sulle impostazioni di configurazione precedenti.

Dipendenze fornite

Il runtime del servizio gestito per Apache Flink fornisce una serie di dipendenze. Queste dipendenze non devono essere incluse nel fat JAR e devono avere un `provided` ambito nel file POM o essere escluse esplicitamente dalla configurazione. `maven-shade-plugin` Ognuna di queste dipendenze incluse nel fat JAR viene ignorata in fase di esecuzione, ma aumenta le dimensioni del JAR aggiungendo un sovraccarico durante la distribuzione.

Dipendenze fornite dal runtime, nelle versioni di runtime 1.18, 1.19 e 1.20:

- `org.apache.flink:flink-core`
- `org.apache.flink:flink-java`
- `org.apache.flink:flink-streaming-java`
- `org.apache.flink:flink-scala_2.12`
- `org.apache.flink:flink-table-runtime`
- `org.apache.flink:flink-table-planner-loader`
- `org.apache.flink:flink-json`
- `org.apache.flink:flink-connector-base`
- `org.apache.flink:flink-connector-files`
- `org.apache.flink:flink-clients`
- `org.apache.flink:flink-runtime-web`
- `org.apache.flink:flink-metrics-code`
- `org.apache.flink:flink-table-api-java`
- `org.apache.flink:flink-table-api-bridge-base`
- `org.apache.flink:flink-table-api-java-bridge`

- `org.apache.logging.log4j:log4j-slf4j-impl`
- `org.apache.logging.log4j:log4j-api`
- `org.apache.logging.log4j:log4j-core`
- `org.apache.logging.log4j:log4j-1.2-api`

Inoltre, il runtime fornisce la libreria utilizzata per recuperare le proprietà di runtime dell'applicazione in Managed Service for Apache Flink, `com.amazonaws:aws-kinesisanalytics-runtime:1.2.0`

Tutte le dipendenze fornite dal runtime devono utilizzare i seguenti consigli per non includerle in uber JAR:

- In Maven (`pom.xml`) e SBT (`build.sbt`), usa `scope.provided`
- In Gradle (`build.gradle`), usa la configurazione `compileOnly`

Qualsiasi dipendenza fornita accidentalmente inclusa in uber JAR verrà ignorata in fase di esecuzione a causa del caricamento della prima classe principale di Apache Flink. Per ulteriori informazioni, consulta la [parent-first-patterns](#) documentazione di Apache Flink.

Connectors (Connettori)

La maggior parte dei connettori, ad eccezione del FileSystem connettore, che non sono inclusi nel runtime devono essere inclusi nel file POM con l'ambito predefinito `()`. `compile`

Altri consigli

Di norma, il file JAR di Apache Flink uber fornito a Managed Service for Apache Flink deve contenere il codice minimo richiesto per eseguire l'applicazione. Le dipendenze che includono le classi di origine, i set di dati di test o lo stato di avvio non devono essere incluse in questo jar. Se è necessario inserire risorse statiche in fase di esecuzione, separa questo problema in una risorsa come Amazon S3. Ne sono un esempio i bootstrap di stato o un modello di inferenza.

Prenditi del tempo per considerare il tuo albero delle dipendenze approfondito e rimuovere le dipendenze non di runtime.

Sebbene Managed Service for Apache Flink supporti file di dimensioni jar da 512 MB, questa dovrebbe essere vista come un'eccezione alla regola. Apache Flink attualmente supporta file di

dimensioni jar di ~104 MB tramite la sua configurazione predefinita, e questa dovrebbe essere la dimensione massima di destinazione di un jar necessaria.

Tolleranza agli errori: checkpoint e savepoint

Utilizza i checkpoint e i savepoint per implementare la tolleranza agli errori nella tua applicazione del servizio gestito per Apache Flink. Quando sviluppi e gestisci un'applicazione, tieni presente quanto indicato di seguito:

- Ti consigliamo di mantenere abilitato il checkpoint per la tua applicazione. La creazione di checkpoint offre tolleranza agli errori per l'applicazione durante la manutenzione programmata e anche per guasti imprevisti dovuti a problemi di servizio, errori di dipendenza dall'applicazione e altri problemi. Per ulteriori informazioni sulla manutenzione, consulta [Gestisci le attività di manutenzione per Managed Service for Apache Flink](#).
- Imposta `ApplicationSnapshotConfiguration::` su `false` durante lo sviluppo o `SnapshotsEnabled` la risoluzione dei problemi dell'applicazione. A ogni arresto dell'applicazione viene creato uno snapshot, il che può causare problemi se l'applicazione non è integra o non è performante. Imposta `SnapshotsEnabled` su `true` una volta che l'applicazione è in produzione ed è stabile.

Note

È consigliabile impostare l'applicazione in modo da creare un'istantanea più volte al giorno per riavviarsi con i dati di stato corretti. La frequenza corretta per l'acquisizione degli snapshot dipende dalla logica di business dell'applicazione. L'acquisizione di snapshot frequenti consente di ripristinare i dati più recenti, ma aumenta i costi e richiede più risorse di sistema.

Per ulteriori informazioni sul monitoraggio dei tempi di inattività delle applicazioni, consulta [???](#).

Per ulteriori informazioni sull'implementazione della tolleranza di errore, consulta [Implementa la tolleranza agli](#).

Versioni di connettori non supportate

A partire dalla versione 1.15 o successiva, il servizio gestito per Apache Flink impedisce automaticamente l'avvio o l'aggiornamento delle applicazioni se utilizzano versioni dei connettori

Kinesis non supportate incluse nell'applicazione. JARs Quando esegui l'aggiornamento del servizio gestito per Apache Flink versione 1.15 o successiva, assicurati di utilizzare il connettore Kinesis più recente. Si tratta di qualsiasi versione uguale o successiva alla versione 1.15.2. Tutte le altre versioni non sono supportate dal servizio gestito per Apache Flink in quanto potrebbero causare problemi di coerenza o errori con la funzione Arresto con savepoint, impedendo le operazioni di interruzione/aggiornamento. Per ulteriori informazioni sulla compatibilità dei connettori nelle versioni di Amazon Managed Service per Apache Flink, consulta [Connettori Apache Flink](#).

Prestazioni e parallelismo

L'applicazione può essere dimensionata per soddisfare qualsiasi throughput ottimizzando il parallelismo delle applicazioni ed evitando problemi di prestazioni. Quando sviluppi e gestisci un'applicazione, tieni presente quanto indicato di seguito:

- Verifica che tutte le origini e i sink dell'applicazione siano sufficientemente forniti e che non subiscano limitazioni della larghezza della banda della rete. Se le origini e i sink sono altri AWS servizi, monitora l'utilizzo [CloudWatch](#) di tali servizi.
- Per le applicazioni con un parallelismo molto elevato, controlla se gli alti livelli di parallelismo vengono applicati a tutti gli operatori dell'applicazione. Per impostazione predefinita, Apache Flink applica lo stesso parallelismo dell'applicazione per tutti gli operatori nel grafico dell'applicazione. Ciò può comportare problemi di approvvigionamento su origini o sink o rallentamenti nell'elaborazione dei dati degli operatori. Puoi modificare il parallelismo di ogni operatore nel codice con [setParallelism](#).
- Comprendi il significato delle impostazioni di parallelismo per gli operatori della tua applicazione. Se modifichi il parallelismo di un operatore, potresti non essere in grado di ripristinare l'applicazione da uno snapshot creato quando l'operatore aveva un parallelismo incompatibile con le impostazioni correnti. Per ulteriori informazioni sull'impostazione del parallelismo degli operatori, consulta [Impostazione esplicita del parallelismo massimo per gli operatori](#).

Per ulteriori informazioni sul dimensionamento semplice, consulta [Implementa la scalabilità delle applicazioni](#).

Impostazione del parallelismo per operatore

Per impostazione predefinita, tutti gli operatori hanno il parallelismo impostato a livello di applicazione. È possibile sovrascrivere il parallelismo di un singolo operatore utilizzando l'

DataStream API. `.setParallelism(x)` È possibile impostare il parallelismo dell'operatore su qualsiasi parallelismo uguale o inferiore al parallelismo dell'applicazione.

Se possibile, definisci il parallelismo dell'operatore in funzione del parallelismo dell'applicazione. In questo modo, il parallelismo dell'operatore varierà con il parallelismo dell'applicazione. Se utilizzi il dimensionamento automatico, ad esempio, tutti gli operatori varieranno il loro parallelismo nella stessa proporzione:

```
int appParallelism = env.getParallelism();  
...  
...ops.setParallelism(appParallelism/2);
```

In alcuni casi, potrebbe essere preferibile impostare il parallelismo degli operatori su una costante. Ad esempio, impostando il parallelismo di un'origine di flusso Kinesis sul numero di partizioni. In questi casi, puoi considerare di passare il parallelismo dell'operatore come parametro di configurazione dell'applicazione per modificarlo senza modificare il codice, ad esempio per eseguire il resharding del flusso di origine.

Registrazione

È possibile monitorare le prestazioni e le condizioni di errore dell'applicazione utilizzando File di CloudWatch log. Quando configuri la registrazione per applicazioni specifiche, tieni presente quanto indicato di seguito:

- Abilita CloudWatch la registrazione per l'applicazione in modo da poter eseguire il debug di eventuali problemi di runtime.
- Non creare una voce di log per ogni record elaborato nell'applicazione. Ciò causa gravi rallentamenti durante l'elaborazione e potrebbe portare a una contropressione nell'elaborazione dei dati.
- Crea CloudWatch allarmi per avvisarti quando l'applicazione non funziona correttamente. Per ulteriori informazioni, consulta [???](#)

Per ulteriori informazioni sull'implementazione della registrazione, consulta [???](#).

Codifica

È possibile rendere l'applicazione performante e stabile utilizzando le pratiche di programmazione consigliate. Quando scrivi il codice di applicazione, tieni presente quanto segue:

- Non utilizzare `system.exit()` nel codice dell'applicazione, né nel metodo `main` dell'applicazione né nelle funzioni definite dall'utente. Se desideri chiudere l'applicazione dall'interno del codice, lancia un'eccezione derivata da `Exception` o `RuntimeException` contenente un messaggio su cosa è andato storto con l'applicazione.

Tieni presente quanto segue su come il servizio gestisce questa eccezione:

- Se l'eccezione viene generata dal metodo `main` della tua applicazione, il servizio la inserirà in una `ProgramInvocationException` quando l'applicazione passerà allo stato `RUNNING` e il job manager non riuscirà a inviare il processo.
- Se l'eccezione viene generata da una funzione definita dall'utente, il job manager interromperà il processo e lo riavvierà, e i dettagli dell'eccezione verranno scritti nel log delle eccezioni.
- Prendi in considerazione l'idea di ombreggiare il file JAR dell'applicazione e le sue dipendenze incluse. L'ombreggiatura è consigliata in caso di potenziali conflitti nei nomi dei pacchetti tra l'applicazione e il runtime di Apache Flink. Se si verifica un conflitto, i log dell'applicazione possono contenere un'eccezione di tipo `java.util.concurrent.ExecutionException`. Per ulteriori informazioni sull'ombreggiatura del file JAR dell'applicazione, consulta [Apache Maven Shade Plugin](#).

Credenziali root.

Non dovresti inserire credenziali a lungo termine nelle applicazioni di produzione (o in qualsiasi altra). Le credenziali a lungo termine vengono probabilmente archiviate in un sistema di controllo delle versioni e possono facilmente perdersi. È invece possibile associare un ruolo all'applicazione del servizio gestito per Apache Flink e concedere le autorizzazioni per tale ruolo. L'applicazione Flink in esecuzione può quindi selezionare credenziali temporanee con le rispettive autorizzazioni dall'ambiente. Nel caso in cui sia necessaria l'autenticazione per un servizio che non è integrato nativamente con IAM, ad esempio un database che richiede un nome utente e una password per l'autenticazione, dovresti prendere in considerazione la possibilità di archiviare i [AWS segreti in Secrets Manager](#).

Molti servizi AWS nativi supportano l'autenticazione:

- [Kinesis Data ProcessTaxiStream Streams — .java](#)
- Amazon MSK — <https://github.com/aws/aws-msk-iam-authusing-the-amazon-msk/#> - library-for-iam-authentication
- [Amazon Elasticsearch Service — .java AmazonElasticsearchSink](#)
- Amazon S3: pronto all'uso nel servizio gestito per Apache Flink

Lettura da fonti con pochi shard/partizioni

Durante la lettura da Apache Kafka o da un flusso di dati Kinesis, potrebbe esserci una discrepanza tra il parallelismo del flusso (il numero di partizioni per Kafka e il numero di partizioni per Kinesis) e il parallelismo dell'applicazione. Con un design ingenuo, il parallelismo di un'applicazione non può superare il parallelismo di un flusso: ogni sottoattività di un operatore di origine può leggere solo da 1 o più shard/partizioni. Ciò significa che per un flusso con sole 2 partizioni e un'applicazione con un parallelismo di 8, solo due sottoattività consumano effettivamente il flusso e 6 sottoattività rimangono inattive. Ciò può limitare notevolmente il throughput dell'applicazione, in particolare se la deserializzazione è costosa e viene eseguita dall'origine (impostazione predefinita).

Per mitigare questo effetto, puoi dimensionare il flusso. Tuttavia, ciò potrebbe non essere sempre auspicabile o possibile. In alternativa, è possibile ristrutturare il codice sorgente in modo che non esegua alcuna serializzazione e trasmetta semplicemente il `byte[]`. È quindi possibile [ribilanciare](#) i dati per distribuirli uniformemente tra tutte le attività e quindi deserializzare i dati in quell'area. In questo modo, è possibile sfruttare tutte le sottoattività per la deserializzazione e questa operazione potenzialmente costosa non è più vincolata dal numero di shard/partizioni del flusso.

Intervallo di aggiornamento del notebook Studio

Se modifichi l'intervallo di aggiornamento dei risultati del paragrafo, impostalo su un valore pari almeno a 1000 millisecondi.

Prestazioni ottimali del notebook Studio

Abbiamo eseguito il test utilizzando la seguente dichiarazione e abbiamo ottenuto le prestazioni ottimali se `events-per-second` moltiplicate per `number-of-keys` erano inferiori a 25.000.000. Questo era per `events-per-second` inferiore a 150.000.

```
SELECT key, sum(value) FROM key-values GROUP BY key
```

Come le strategie di watermark e le partizioni inattive influiscono sulle finestre temporali

Durante la lettura di eventi da Apache Kafka e dal flusso di dati Kinesis, l'origine può impostare l'ora dell'evento in base agli attributi del flusso. Nel caso di Kinesis, l'ora dell'evento è uguale all'ora approssimativa di arrivo degli eventi. Tuttavia, impostare l'ora dell'evento all'origine degli eventi non è sufficiente per consentire a un'applicazione Flink di utilizzare tale orario. L'origine deve inoltre generare watermark che diffondano le informazioni sull'ora dell'evento dall'origine a tutti gli altri operatori. La [documentazione di Flink](#) offre una buona panoramica di come funziona questo processo.

Per impostazione predefinita, il timestamp di un evento letto da Kinesis è impostato sull'ora di arrivo approssimativa determinata da Kinesis. Un ulteriore prerequisito affinché l'ora dell'evento funzioni nell'applicazione è una strategia di watermark.

```
WatermarkStrategy<String> s = WatermarkStrategy
    .<String>forMonotonousTimestamps()
    .withIdleness(Duration.ofSeconds(...));
```

La strategia di watermark viene quindi applicata a un `DataStream` con il metodo `assignTimestampsAndWatermarks`. Esistono alcune utili strategie integrate:

- `forMonotonousTimestamps()` utilizzerà semplicemente l'ora dell'evento (ora di arrivo approssimativa) e genererà periodicamente il valore massimo come watermark (per ogni sottoattività specifica)
- `forBoundedOutOfOrderness(Duration.ofSeconds(...))` simile alla strategia precedente, ma utilizzerà l'ora e la durata dell'evento per la generazione del watermark.

Dalla [documentazione di Flink](#):

Ogni sottoattività parallela di una funzione di origine di solito genera le proprie filigrane in modo indipendente. Questi watermark definiscono l'ora dell'evento in quella particolare origine parallela.

Man mano che i watermark scorrono attraverso il programma di streaming, anticipano l'ora dell'evento presso gli operatori a cui arrivano. Ogni volta che un operatore anticipa l'orario dell'evento, genera un nuovo watermark a valle per gli operatori successivi.

Alcuni operatori utilizzano più flussi di input, ad esempio un'unione o operatori che seguono una funzione `keyBy(...)` o `partition(...)`. L'ora corrente degli eventi di un operatore di questo tipo è la durata minima degli eventi dei suoi flussi di input. Man mano che i flussi di input aggiornano gli orari degli eventi, così fa anche l'operatore.

Ciò significa che se un'attività secondaria di origine utilizza una partizione inattiva, gli operatori a valle non ricevono nuovi watermark da quella sottoattività e quindi l'elaborazione si blocca per tutti gli operatori a valle che utilizzano finestre temporali. Per evitare ciò, i clienti possono aggiungere l'opzione `withIdleness` alla strategia watermark. Con questa opzione, un operatore esclude i watermark dalle sottoattività inattive durante il calcolo dell'orario dell'evento da parte dell'operatore. La sottoattività inattiva quindi non blocca più l'avanzamento dell'orario degli eventi negli operatori a valle.

A seconda dell'assegnatore di shard utilizzato, ad alcuni worker potrebbe non essere assegnato alcun shard Kinesis. In tal caso, questi worker manifesteranno il comportamento `idle source` anche se tutti gli shard Kinesis forniscono continuamente dati sugli eventi. È possibile mitigare questo rischio utilizzando l'operatore `uniformShardAssigner` di origine. In questo modo si garantisce che tutte le sottoattività di origine abbiano degli shard da elaborare purché il numero di lavoratori sia inferiore o uguale al numero di shard attivi.

Tuttavia, l'opzione di inattività con le strategie di watermark integrate non farà avanzare la durata dell'evento se nessuna sottoattività legge alcun evento, ovvero se non ci sono eventi nel flusso. Ciò diventa particolarmente visibile nei casi di test in cui un insieme finito di eventi viene letto dal flusso. Poiché l'ora dell'evento non avanza dopo la lettura dell'ultimo evento, l'ultima finestra (contenente l'ultimo evento) non si chiuderà.

Riepilogo

- L'`withIdleness` impostazione non genererà nuove filigrane nel caso in cui uno shard sia inattivo. Escluderà l'ultimo watermark inviato dalle sottoattività inattive dal calcolo del watermark minimo negli operatori a valle.
- Con le strategie integrate di watermark, l'ultima finestra aperta non si chiuderà (a meno che non vengano inviati nuovi eventi che fanno avanzare il watermark, ma ciò crea una nuova finestra che poi rimane aperta).
- Anche quando l'ora è impostata dal flusso Kinesis, possono comunque verificarsi eventi in ritardo se una partizione viene consumata più velocemente di altre (ad esempio durante l'inizializzazione dell'app o quando si utilizza il momento in `TRIM_HORIZON` cui tutte le partizioni esistenti vengono consumate in parallelo, ignorando la relazione padre/figlio).

- Le `withIdleness` impostazioni della strategia watermark sembrano interrompere le impostazioni specifiche del codice sorgente Kinesis per gli shard inattivi. (`ConsumerConfigConstants.SHARD_IDLE_INTERVAL_MILLIS`)

Esempio

La seguente applicazione legge da un flusso e crea finestre di sessione in base all'ora dell'evento.

```
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(1 -> 01)
    .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("XXXXXXXXXXXXXXXXX Window with " + count + " events");
            System.out.println("; Watermark: " + timestamp + ", " +
                Instant.ofEpochMilli(timestamp));
        }
    });
```

```

        for (Long l : iterable) {
            System.out.println(l);
        }
    }
});

```

Nell'esempio seguente, 8 eventi vengono scritti in un flusso di 16 partizioni (i primi 2 e l'ultimo evento finiscono nella stessa partizione).

```

$ 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

```

Questo input dovrebbe generare 5 finestre di sessione: evento 1,2,3; evento 4,5; evento 6; evento 7; evento 8. Tuttavia, il programma produce solo le prime 4 finestre.

```

11:59:21,529 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 5 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000006,HashKeyRange: {StartingHashKey:
127605887595351923798765477786913079296,EndingHashKey:
148873535527910577765226390751398592511},SequenceNumberRange: {StartingSequenceNumber:
49627894338480851089309627289524549239292625588395704418,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,530 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 5 will start consuming seeded shard StreamShardHandle{streamName='hp-16',

```

```

shard='{ShardId: shardId-000000000006,HashKeyRange: {StartingHashKey:
127605887595351923798765477786913079296,EndingHashKey:
148873535527910577765226390751398592511},SequenceNumberRange: {StartingSequenceNumber:
49627894338480851089309627289524549239292625588395704418,}}}' from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,530 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 6 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000007,HashKeyRange: {StartingHashKey:
148873535527910577765226390751398592512,EndingHashKey:
170141183460469231731687303715884105727},SequenceNumberRange: {StartingSequenceNumber:
49627894338503151834508157912666084957565273949901684850,}}}', starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,530 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 6 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000010,HashKeyRange: {StartingHashKey:
212676479325586539664609129644855132160,EndingHashKey:
233944127258145193631070042609340645375},SequenceNumberRange: {StartingSequenceNumber:
49627894338570054070103749782090692112383219034419626146,}}}', starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,530 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 6 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000007,HashKeyRange: {StartingHashKey:
148873535527910577765226390751398592512,EndingHashKey:
170141183460469231731687303715884105727},SequenceNumberRange: {StartingSequenceNumber:
49627894338503151834508157912666084957565273949901684850,}}}' from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,531 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 4 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000005,HashKeyRange: {StartingHashKey:
106338239662793269832304564822427566080,EndingHashKey:
127605887595351923798765477786913079295},SequenceNumberRange: {StartingSequenceNumber:
49627894338458550344111096666383013521019977226889723986,}}}', starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 4 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000005,HashKeyRange: {StartingHashKey:
106338239662793269832304564822427566080,EndingHashKey:
127605887595351923798765477786913079295},SequenceNumberRange: {StartingSequenceNumber:
49627894338458550344111096666383013521019977226889723986,}}}' from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 3 will be seeded with initial shard StreamShardHandle{streamName='hp-16',

```



```

shard='{ShardId: shardId-000000000004,HashKeyRange: {StartingHashKey:
85070591730234615865843651857942052864,EndingHashKey:
106338239662793269832304564822427566079},SequenceNumberRange: {StartingSequenceNumber:
49627894338436249598912566043241477802747328865383743554,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 2 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000003,HashKeyRange: {StartingHashKey:
63802943797675961899382738893456539648,EndingHashKey:
85070591730234615865843651857942052863},SequenceNumberRange: {StartingSequenceNumber:
49627894338413948853714035420099942084474680503877763122,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 3 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-0000000000015,HashKeyRange: {StartingHashKey:
319014718988379809496913694467282698240,EndingHashKey:
340282366920938463463374607431768211455},SequenceNumberRange: {StartingSequenceNumber:
49627894338681557796096402897798370703746460841949528306,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 2 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-0000000000014,HashKeyRange: {StartingHashKey:
297747071055821155530452781502797185024,EndingHashKey:
319014718988379809496913694467282698239},SequenceNumberRange: {StartingSequenceNumber:
49627894338659257050897872274656834985473812480443547874,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 3 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-0000000000004,HashKeyRange: {StartingHashKey:
85070591730234615865843651857942052864,EndingHashKey:
106338239662793269832304564822427566079},SequenceNumberRange: {StartingSequenceNumber:
49627894338436249598912566043241477802747328865383743554,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,532 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 2 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-0000000000003,HashKeyRange: {StartingHashKey:
63802943797675961899382738893456539648,EndingHashKey:
85070591730234615865843651857942052863},SequenceNumberRange: {StartingSequenceNumber:
49627894338413948853714035420099942084474680503877763122,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',

```

```

shard='{ShardId: shardId-000000000001,HashKeyRange: {StartingHashKey:
21267647932558653966460912964485513216,EndingHashKey:
42535295865117307932921825928971026431},SequenceNumberRange: {StartingSequenceNumber:
49627894338369347363316974173816870647929383780865802258,}}'}', starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000009,HashKeyRange: {StartingHashKey:
191408831393027885698148216680369618944,EndingHashKey:
212676479325586539664609129644855132159},SequenceNumberRange: {StartingSequenceNumber:
49627894338547753324905219158949156394110570672913645714,}}'}', starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey:
21267647932558653966460912964485513215},SequenceNumberRange: {StartingSequenceNumber:
49627894338347046618118443550675334929656735419359821826,}}'}', starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000012,HashKeyRange: {StartingHashKey:
255211775190703847597530955573826158592,EndingHashKey:
276479423123262501563991868538311671807},SequenceNumberRange: {StartingSequenceNumber:
49627894338614655560500811028373763548928515757431587010,}}'}', starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000008,HashKeyRange: {StartingHashKey:
170141183460469231731687303715884105728,EndingHashKey:
191408831393027885698148216680369618943},SequenceNumberRange: {StartingSequenceNumber:
49627894338525452579706688535807620675837922311407665282,}}'}', starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000001,HashKeyRange: {StartingHashKey:
21267647932558653966460912964485513216,EndingHashKey:
42535295865117307932921825928971026431},SequenceNumberRange: {StartingSequenceNumber:
49627894338369347363316974173816870647929383780865802258,}}'}' from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,533 INFO  org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 7 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000011,HashKeyRange: {StartingHashKey:
233944127258145193631070042609340645376,EndingHashKey:

```

```

255211775190703847597530955573826158591}],SequenceNumberRange: {StartingSequenceNumber:
49627894338592354815302280405232227830655867395925606578,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,533 INFO
org.apache.flink.streaming.connectors.kinesis.internal.KinesisDataFetcher [] -
Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey:
21267647932558653966460912964485513215},SequenceNumberRange: {StartingSequenceNumber:
49627894338347046618118443550675334929656735419359821826,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,568 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 1 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000002,HashKeyRange: {StartingHashKey:
42535295865117307932921825928971026432,EndingHashKey:
63802943797675961899382738893456539647},SequenceNumberRange: {StartingSequenceNumber:
49627894338391648108515504796958406366202032142371782690,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,568 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 1 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000013,HashKeyRange: {StartingHashKey:
276479423123262501563991868538311671808,EndingHashKey:
297747071055821155530452781502797185023},SequenceNumberRange: {StartingSequenceNumber:
49627894338636956305699341651515299267201164118937567442,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,568 INFO
org.apache.flink.streaming.connectors.kinesis.internal.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.internal.KinesisDataFetcher [] -
Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000009,HashKeyRange: {StartingHashKey:
191408831393027885698148216680369618944,EndingHashKey:
212676479325586539664609129644855132159},SequenceNumberRange: {StartingSequenceNumber:
49627894338547753324905219158949156394110570672913645714,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 1
11:59:23,244 INFO
org.apache.flink.streaming.connectors.kinesis.internal.KinesisDataFetcher [] -
Subtask 6 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000010,HashKeyRange: {StartingHashKey:

```

```

212676479325586539664609129644855132160,EndingHashKey:
233944127258145193631070042609340645375},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:
319014718988379809496913694467282698240,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:
49627894338592354815302280405232227830655867395925606578,}}'} 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
XXXXXXXXXXXXXXXX Window with 3 events; Watermark: 1648030809281, 2022-03-23T10:20:09.281Z
3
1
2
XXXXXXXXXXXXXXXX Window with 2 events; Watermark: 1648030834104, 2022-03-23T10:20:34.104Z
4
5
XXXXXXXXXXXXXXXX Window with 1 events; Watermark: 1648030834104, 2022-03-23T10:20:34.104Z
6
XXXXXXXXXXXXXXXX Window with 1 events; Watermark: 1648030887170, 2022-03-23T10:21:27.170Z
7

```

L'output mostra solo 4 finestre (manca l'ultima finestra contenente l'evento 8). Ciò è dovuto all'ora dell'evento e alla strategia di watermark. L'ultima finestra non può chiudersi perché, secondo le strategie di watermark predefinite, il tempo non avanza mai oltre l'ora dell'ultimo evento letto dal flusso. Ma perché la finestra si chiuda, il tempo deve avanzare di oltre 10 secondi dopo l'ultimo evento. In questo caso, l'ultimo watermark è 2022-03-23T 10:21:27.170 Z, ma per chiudere la finestra della sessione è necessario un watermark di 10 secondi e 1 millisecondo dopo.

Se l'`withIdleness` opzione viene rimossa dalla strategia di watermark, nessuna finestra di sessione si chiuderà mai, perché la «watermark globale» dell'operatore finestra non può avanzare.

All'avvio dell'applicazione Flink (o in caso di distorsione dei dati), alcune partizioni potrebbero essere consumate più velocemente di altre. Ciò può far sì che alcuni watermark vengano generati troppo presto da una sottoattività (la sottoattività può

generare il watermark in base al contenuto di una partizione senza aver consumato le altre partizioni a cui è abbonata). I modi per mitigare il problema sono diverse strategie di watermarking, che aggiungono un buffer di sicurezza o consentono esplicitamente l'arrivo di eventi tardivi. (`forBoundedOutOfOrderness(Duration.ofSeconds(30))` (`allowedLateness(Time.minutes(5))`)

Impostare un UUID per tutti gli operatori

Quando Managed Service for Apache Flink avvia un processo Flink per un'applicazione con uno snapshot, il processo Flink può non avviarsi a causa di determinati problemi. Uno di questi è la mancata corrispondenza degli ID dell'operatore. Flink si aspetta un operatore esplicito e coerente IDs per gli operatori grafici del processo Flink. Se non è impostato in modo esplicito, Flink genera un ID per gli operatori. Questo perché Flink utilizza questi operatori IDs per identificare in modo univoco gli operatori in un grafico del processo e li utilizza per memorizzare lo stato di ciascun operatore in un savepoint.

Il problema della mancata corrispondenza degli ID degli operatori si verifica quando Flink non trova una mappatura 1:1 tra l'operatore IDs di un grafico del processo e l'operatore IDs definito in un savepoint. Ciò accade quando non IDs sono impostati operatori coerenti espliciti e Flink genera operatori IDs che potrebbero non essere coerenti con ogni creazione di grafici del processo. La probabilità che le applicazioni riscontrino questo problema è elevata durante gli interventi di manutenzione. Per evitare questo problema, consigliamo ai clienti di impostare l'UUID per tutti gli operatori nel codice Flink. Per ulteriori informazioni, consulta l'argomento [Impostare un UUID per tutti gli operatori](#) in [Preparazione per la produzione](#).

Aggiungi ServiceResourceTransformer al plugin Maven shade

Flink utilizza le interfacce SPI ([Service Provider Interfaces](#)) di Java per caricare componenti come connettori e formati. Le dipendenze multiple di Flink che utilizzano SPI [possono causare conflitti nell'uber-jar e comportamenti imprevisti delle applicazioni](#). Ti consigliamo [ServiceResourceTransformer](#) di aggiungere il plugin Maven Shade, definito nel pom.xml.

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-shade-plugin</artifactId>
      <executions>
```

```
        <execution>
          <id>shade</id>
          <phase>package</phase>
          <goals>
            <goal>shade</goal>
          </goals>
          <configuration>
            <transformers combine.children="append">
              <!-- The service transformer is needed to merge META-
INF/services files -->
              <transformer
implementation="org.apache.maven.plugins.shade.resource.ServicesResourceTransformer"/>
              <!-- ... -->
            </transformers>
          </configuration>
        </execution>
      </executions>
    </plugin>
```

Funzioni statiche di Apache Flink

[Stateful Functions](#) è un'API che semplifica la creazione di applicazioni stateful distribuite. Si basa su funzioni con stato persistente in grado di interagire dinamicamente e con solide garanzie di coerenza.

Un'applicazione Stateful Functions è di base semplicemente un'applicazione Apache Flink e quindi può essere distribuita sul servizio gestito per Apache Flink. Tuttavia, ci sono un paio di differenze tra la creazione di pacchetti Stateful Functions per un cluster Kubernetes e per il servizio gestito per Apache Flink. L'aspetto più importante di un'applicazione Stateful Functions è che la [configurazione del modulo](#) contiene tutte le informazioni di runtime necessarie per configurare il runtime di Stateful Functions. Questa configurazione viene in genere assemblata in un container specifico di Stateful Functions e distribuita su Kubernetes. Non è tuttavia possibile farlo con il servizio gestito per Apache Flink.

Di seguito è riportato un adattamento dell'esempio di StateFun Python per Managed Service for Apache Flink:

Modello di applicazione Apache Flink

Invece di utilizzare un container per il runtime di Stateful Functions, i clienti possono compilare un container di applicazioni Flink che richiami semplicemente il runtime Stateful Functions e contenga le dipendenze richieste. Per Flink 1.13, le dipendenze richieste sono simili a queste:

```
<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>
```


Il metodo principale dell'applicazione Flink per richiamare il runtime Stateful Function è il seguente:

```
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);
}
```

Tieni presente che questi componenti sono generici e indipendenti dalla logica implementata nella Stateful Function.

Ubicazione della configurazione del modulo

La configurazione del modulo Stateful Functions deve essere inclusa nel percorso della classe per essere individuabile per il runtime Stateful Functions. È meglio includerla nella cartella delle risorse dell'applicazione Flink e inserirla nel file jar.

Analogamente a quanto avviene per una comune applicazione Apache Flink, puoi quindi utilizzare Maven per creare un file jar uber e distribuirlo sul servizio gestito per Apache Flink.

Impostazioni Apache Flink

Il servizio gestito per Apache Flink è un'implementazione del framework Apache Flink. Il servizio gestito per Apache Flink utilizza i valori predefiniti descritti in questa sezione. Alcuni di questi valori possono essere impostati dal servizio gestito per le applicazioni Apache Flink nel codice, mentre altri non possono essere modificati.

Usa i link in questa sezione per saperne di più sulle impostazioni di Apache flink e su quali sono modificabili.

Questo argomento contiene le sezioni seguenti:

- [Configurazione di Apache Flink](#)
- [Backend statale](#)
- [Checkpoint](#)
- [Savepoint](#)
- [Dimensioni del mucchio](#)
- [Debloating del buffer](#)
- [Proprietà di configurazione modificabili di Flink](#)
- [Visualizza le proprietà Flink configurate](#)

Configurazione di Apache Flink

Il servizio gestito per Apache Flink fornisce una configurazione Flink predefinita composta da valori consigliati da Apache Flink per la maggior parte delle proprietà e alcune basati su profili comuni per le applicazioni. Per ulteriori informazioni sulla configurazione di Flink, consulta [Configurazione](#). La configurazione predefinita fornita dal servizio funziona per la maggior parte delle applicazioni. Tuttavia, per modificare le proprietà di configurazione di Flink per migliorare le prestazioni di determinate applicazioni con elevato parallelismo, elevato utilizzo di memoria e stato o abilitare nuove funzionalità di debug in Apache Flink, puoi modificare determinate proprietà richiedendo un caso di supporto. Per ulteriori informazioni, consulta [Centro di supporto di AWS](#). È possibile controllare la configurazione corrente dell'applicazione utilizzando la [dashboard di Apache Flink](#).

Backend statale

Il servizio gestito per Apache Flink archivia i dati transitori in un backend di stato. Il servizio gestito per Apache Flink utilizza il backend Rocks. DBState Chiamare setStateBackend per impostare un backend diverso non ha alcun effetto.

Abilitiamo le seguenti funzionalità sul backend di stato:

- Snapshot incrementali del backend di stato
- Snapshot del backend di stato asincroni
- Ripristino locale dei checkpoint

Per ulteriori informazioni sui backend di stato, consulta [State Backends](#) nella documentazione di Apache Flink.

Checkpoint

Il servizio gestito per Apache Flink utilizza una configurazione di checkpoint predefinita con i seguenti valori. Alcuni di questi valori possono essere modificati utilizzando. [CheckpointConfiguration](#) È necessario CheckpointConfiguration.ConfigurationType impostare su Managed Service CUSTOM for Apache Flink per utilizzare i valori di checkpoint modificati.

Impostazione	Può essere modificato?	In che modo	Valore predefinito
CheckpointingEnabled	Modificabile	Crea applicazione Aggiorna applicazione AWS CloudFormation	True
CheckpointInterval	Modificabile	Crea applicazione Aggiorna applicazione AWS CloudFormation	60000
MinPauseBetweenCheckpoints	Modificabile	Crea applicazione	5000

Impostazione	Può essere modificato?	In che modo	Valore predefinito
		Aggiorna applicazione	
		AWS CloudFormation	
Checkpoint non allineati	Modificabile	Caso di supporto	False
Numero di checkpoint simultanei	Non modificabile	N/D	1
Modalità di checkpoint	Non modificabile	N/D	Esattamente una volta
Policy di conservazione dei checkpoint	Non modificabile	N/D	In caso di errore
Timeout checkpoint	Non modificabile	N/D	60 minuti
Max. di checkpoint mantenuti	Non modificabile	N/D	1
Ubicazione di checkpoint e savepoint	Non modificabile	N/D	Archiviamo dati durevoli su checkpoint e savepoint in un bucket S3 di proprietà del servizio.

Savepoint

Per impostazione predefinita, quando si esegue il ripristino da un savepoint, l'operazione di ripristino proverà a mappare tutto lo stato del savepoint nel programma con cui si sta eseguendo il ripristino. Se hai eliminato un operatore, per impostazione predefinita, il ripristino da un savepoint con dati corrispondenti all'operatore mancante avrà esito negativo. È possibile consentire il successo dell'operazione impostando il `AllowNonRestoredState` parametro dell'applicazione [FlinkRunConfiguration](#) su `true`. Ciò consentirà all'operazione di ripristino di ignorare uno stato che non può essere mappato nel nuovo programma.

Per ulteriori informazioni, consulta la sezione [Consentire lo stato non ripristinato](#) nella [documentazione di Apache Flink](#).

Dimensioni del mucchio

Il servizio gestito per Apache Flink alloca 3 GB di heap JVM per ogni KPU e riserva 1 GB per le allocazioni di codice nativo. Per informazioni sull'aumento della capacità delle applicazioni, consulta [the section called “Implementa la scalabilità delle applicazioni”](#).

Per ulteriori informazioni sulle dimensioni heap di JVM, consultare [Configurazione](#) nella [documentazione di Apache Flink](#).

Debloating del buffer

Il debloating del buffer può aiutare le applicazioni che hanno una congestione elevata. Se l'applicazione presenta checkpoint/savepoint non riusciti, potrebbe essere utile abilitare questa funzionalità. A tal scopo, richiedi un [caso di supporto](#).

Per ulteriori informazioni, consulta [Il meccanismo di debloating del buffer](#) nella documentazione di [Apache Flink](#).

Proprietà di configurazione modificabili di Flink

Di seguito sono riportate le impostazioni di configurazione di Flink che è possibile modificare utilizzando un case di [supporto](#). È possibile modificare più di una proprietà alla volta e per più applicazioni contemporaneamente specificando il prefisso dell'applicazione. Se ci sono altre proprietà di configurazione Flink al di fuori di questo elenco che desideri modificare, specifica la proprietà esatta nel tuo caso.

strategia di riavvio

A partire da Flink 1.19 e versioni successive, utilizziamo la strategia di `exponential-delay` riavvio per impostazione predefinita. Tutte le versioni precedenti utilizzano la strategia di `fixed-delay` riavvio per impostazione predefinita.

`restart-strategy:`

`restart-strategy.fixed-delay.delay:`

`restart-strategy.exponential-delay.backoff-multiplier:`

`restart-strategy.exponential-delay.initial-backoff:`

`restart-strategy.exponential-delay.jitter-factor:`

`restart-strategy.exponential-delay.reset-backoff-threshold:`

Checkpoint e backend di stato

`state.backend:`

`state.backend.fs.memory-threshold:`

`state.backend.incremental:`

Checkpoint

`execution.checkpointing.unaligned:`

`execution.checkpointing.interval-during-backlog:`

Metriche native di RocksDB

Le metriche native di RocksDB non vengono spedite a CloudWatch. Una volta abilitate, è possibile accedere a queste metriche dalla dashboard di Flink o dalla REST API di Flink con strumenti personalizzati.

Managed Service for Apache Flink consente ai clienti di accedere all'ultima [API REST](#) di Flink (o alla versione supportata in uso) in modalità di sola lettura utilizzando l'API. [CreateApplicationPresignedUrl](#). Questa API viene utilizzata dalla dashboard di Flink, ma può essere utilizzata anche da strumenti di monitoraggio personalizzati.

`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:`

Opzioni RockSDB

`state.backend.rocksdb.compaction.style:`
`state.backend.rocksdb.memory.partitioned-index-filters:`

`state.backend.rocksdbs.thread.num:`

Opzioni avanzate di backend di stato

`state.storage.fs.memory-threshold:`

Opzioni complete TaskManager

`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:`

Configurazione della memoria

`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:`

Opzioni di cluster avanzate

`cluster.intercept-user-system-exit:`

`cluster.processes.halt-on-fatal-error:`

Configurazioni del file system

`fs.s3.connection.maximum:`

`fs.s3a.connection.maximum:`

`fs.s3a.threads.max:`

`s3.upload.max.concurrent.uploads:`

Opzioni avanzate di tolleranza ai guasti

`heartbeat.timeout:`

`jobmanager.execution.failover-strategy:`

Configurazione della memoria

`jobmanager.memory.heap.size:`

Metriche

`metrics.latency.interval:`

Opzioni avanzate per l'endpoint e il client REST

`rest.flamegraph.enabled:`

`rest.server.numThreads:`

Opzioni di sicurezza SSL avanzate

`security.ssl.internal.handshake-timeout:`

Opzioni di pianificazione avanzate

`slot.request.timeout:`

Opzioni avanzate per l'interfaccia utente web di Flink

`web.timeout:`

Visualizza le proprietà Flink configurate

Puoi visualizzare le proprietà di Apache Flink che hai configurato personalmente o che hai richiesto di modificare attraverso un [caso di supporto](#) tramite la dashboard di Apache Flink e seguendo questi passaggi:

1. Vai alla dashboard di Flink
2. Scegli Job Manager nel riquadro di navigazione a sinistra.
3. Scegli Configurazione per visualizzare l'elenco delle proprietà di Flink.

Configura Managed Service per Apache Flink per accedere alle risorse in un Amazon VPC

È possibile configurare un'applicazione di servizio gestito per Apache Flink affinché si connetta a sottoreti private in un cloud privato virtuale (VPC) nell'account. L'Amazon Virtual Private Cloud (Amazon VPC) consente di creare una rete privata per le risorse come database, istanze di cache o servizi interni. Per accedere alle risorse private nel corso dell'esecuzione è sufficiente connettere l'applicazione al VPC.

Questo argomento contiene le sezioni seguenti:

- [Concetti di Amazon VPC](#)
- [Autorizzazioni per applicazioni VPC](#)
- [Accesso a Internet e ai servizi per un'applicazione Managed Service per Apache Flink connessa a VPC](#)
- [Usa l'API VPC Managed Service for Apache Flink](#)
- [Esempio: utilizzare un VPC per accedere ai dati in un cluster Amazon MSK](#)

Concetti di Amazon VPC

Amazon VPC è il livello di rete per Amazon. EC2 Se non conosci Amazon EC2, consulta [Che cos'è Amazon EC2?](#) nella Amazon EC2 User Guide for Linux Instances per una breve panoramica.

Di seguito sono riportati i concetti chiave per VPCs:

- Un cloud privato virtuale (VPC) è una rete virtuale dedicata al tuo AWS account.
- una sottorete è un intervallo di indirizzi IP nel VPC;
- una tabella di routing contiene un insieme di regole denominate route, che consentono di determinare la direzione del traffico di rete;
- un gateway Internet è un componente VPC dimensionato orizzontalmente, ridondante e ad alta disponibilità che consente la comunicazione tra le istanze nel VPC e Internet. Non comporta, pertanto, alcun rischio a livello di disponibilità o vincoli di larghezza di banda per il traffico di rete;
- Un endpoint VPC ti consente di connettere privatamente il tuo VPC ai servizi supportati AWS e ai servizi endpoint VPC alimentati da PrivateLink senza richiedere un gateway Internet, un dispositivo NAT, una connessione VPN o una connessione. AWS Direct Connect Le istanze nel VPC non

richiedono indirizzi IP pubblici per comunicare con le risorse nel servizio. Il traffico tra il VPC e gli altri servizi rimane all'interno della rete Amazon.

Per ulteriori informazioni sul cloud privato virtuale di Amazon consulta la [Guida per l'utente di Amazon VPC](#).

Il servizio gestito per Apache Flink crea [interfacce di rete elastiche](#) in una delle sottoreti fornite nella configurazione del cloud privato virtuale per l'applicazione. Il numero di interfacce di rete elastiche create nelle sottoreti VPC può variare a seconda del parallelismo e del parallelismo per KPU dell'applicazione. Per ulteriori informazioni relative al dimensionamento delle applicazioni, consulta [Implementa la scalabilità delle applicazioni](#).

Note

Le configurazioni VPC non sono supportate per le applicazioni SQL.

Note

Il servizio gestito Apache Flink gestisce lo stato dei checkpoint e degli snapshot per le applicazioni dotate di una configurazione VPC.

Autorizzazioni per applicazioni VPC

Questa sezione descrive le politiche di autorizzazione necessarie all'applicazione per funzionare con il VPC. Per ulteriori informazioni sull'utilizzo delle politiche di autorizzazione, consulta [Identity and Access Management per il servizio gestito da Amazon per Apache Flink](#).

La seguente politica di autorizzazione concede all'applicazione le autorizzazioni necessarie per interagire con un VPC. Per utilizzare questa politica di autorizzazione è necessario aggiungerla al ruolo di esecuzione dell'applicazione.

Aggiungi una politica di autorizzazioni per l'accesso a un 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

Quando specifichi le risorse dell'applicazione utilizzando la console (come CloudWatch Logs o Amazon VPC), la console modifica il ruolo di esecuzione dell'applicazione per concedere l'autorizzazione all'accesso a tali risorse. Se l'applicazione è stata creata senza l'utilizzo della console, sarà necessario modificarne manualmente il ruolo di esecuzione.

Accesso a Internet e ai servizi per un'applicazione Managed Service per Apache Flink connessa a VPC

Per impostazione automatica, un'applicazione di servizio gestito per Apache Flink connessa a un VPC nell'account non ha accesso a Internet, a meno che non sia il VPC a fornirglielo. Se l'applicazione richiede l'accesso a Internet, devono verificarsi le seguenti condizioni:

- l'applicazione di servizio gestito per Apache Flink deve essere configurata unicamente con sottoreti private;
- il VPC deve contenere un gateway o un'istanza NAT in una sottorete pubblica;
- deve esistere una route per il traffico in uscita dalle sottoreti private verso il gateway NAT in una sottorete pubblica.

Note

Diversi servizi offrono gli [endpoint VPC](#). È possibile sfruttare gli endpoint VPC per connettersi ai servizi Amazon all'interno di un cloud privato virtuale senza accesso a Internet.

La sottorete sarà pubblica o privata in base alla tabella di routing corrispondente. Ogni tabella di routing ha una route predefinita, che determina l'hop successivo per i pacchetti con una destinazione pubblica.

- Per una sottorete privata: la route predefinita punta a un gateway NAT (nat-...) o a un'istanza NAT (eni-...).
- Per una sottorete pubblica: la route predefinita punta a un gateway Internet (igw-...).

Dopo aver configurato il VPC con una sottorete pubblica (con un NAT) e una o più sottoreti private, è necessario identificare le sottoreti private e pubbliche seguendo questi passaggi:

- nel pannello di navigazione della console del VPC, seleziona Sottoreti;
- selezionane una, poi passa alla scheda Tabella di routing; verifica il percorso predefinito:
 - Sottorete pubblica: destinazione: 0.0.0.0/0, obiettivo: igw-...
 - Sottorete privata: destinazione: 0.0.0.0/0, obiettivo: nat-... o eni-...

Per associare l'applicazione di servizio gestito per Apache Flink a sottoreti private:

- Apri la console Managed Service for Apache Flink su /flink <https://console.aws.amazon.com>
- nella pagina Applicazioni di servizio gestito per Apache Flink seleziona prima l'applicazione interessata, poi Dettagli dell'applicazione.
- nella pagina dell'applicazione, seleziona Configura;

- nella sezione Connettività VPC scegli il VPC da associare all'applicazione; seleziona le sottoreti e il gruppo di sicurezza associati al VPC che l'applicazione deve utilizzare per accedere alle risorse del cloud privato virtuale;
- clicca su Aggiorna.

Informazioni correlate

[Creazione di un VPC con sottoreti pubbliche e private](#)

[Nozioni di base sul gateway NAT](#)

Usa l'API VPC Managed Service for Apache Flink

Utilizza il seguente servizio gestito per le operazioni dell'API Apache Flink per gestire la tua applicazione. VPCs Per informazioni sull'utilizzo delle API per il servizio gestito per Apache Flink consulta [Codice di esempio di API](#).

Crea applicazione

Usa l'[CreateApplication](#) azione per aggiungere una configurazione VPC all'applicazione durante la creazione.

Il seguente codice di richiesta di esempio per l'azione CreateApplication include una configurazione VPC al momento della creazione dell'applicazione:

```
{
  "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:::amzn-s3-demo-bucket",
          "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

Usa l'[AddApplicationVpcConfiguration](#) azione per aggiungere una configurazione VPC all'applicazione dopo che è stata creata.

Il seguente codice di richiesta di esempio per l'azione AddApplicationVpcConfiguration aggiunge una configurazione VPC a un'applicazione esistente:

```

{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 9,
  "VpcConfiguration": {
    "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
    "SubnetIds": [ "subnet-0123456789abcdef0" ]
  }
}

```

DeleteApplicationVpcConfiguration

Usa l'[DeleteApplicationVpcConfiguration](#) azione per rimuovere una configurazione VPC dalla tua applicazione.

Il seguente codice di richiesta di esempio per l'azione `AddApplicationVpcConfiguration` rimuove una configurazione VPC esistente da un'applicazione:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 9,
  "VpcConfigurationId": "1.1"
}
```

Aggiorna l'applicazione

Utilizza l'[UpdateApplication](#) azione per aggiornare tutte le configurazioni VPC di un'applicazione contemporaneamente.

Il seguente codice di richiesta di esempio per l'azione `UpdateApplication` aggiorna tutte le configurazioni VPC per un'applicazione:

```
{
  "ApplicationConfigurationUpdate": {
    "VpcConfigurationUpdates": [
      {
        "SecurityGroupIdUpdates": [ "sg-0123456789abcdef0" ],
        "SubnetIdUpdates": [ "subnet-0123456789abcdef0" ],
        "VpcConfigurationId": "2.1"
      }
    ]
  },
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 9
}
```

Esempio: utilizzare un VPC per accedere ai dati in un cluster Amazon MSK

Per un tutorial completo sull'accesso ai dati in un VPC da un cluster Amazon MSK, consulta [Replica MSK](#).

Risoluzione dei problemi relativi al servizio gestito per Apache Flink

I seguenti argomenti possono aiutarti a risolvere i problemi che potresti riscontrare con Amazon Managed Service for Apache Flink.

Scegli l'argomento appropriato per esaminare le soluzioni.

Argomenti

- [Risoluzione dei problemi di sviluppo](#)
- [Risoluzione dei problemi di runtime](#)

Risoluzione dei problemi di sviluppo

Questa sezione contiene informazioni sulla diagnosi e la risoluzione dei problemi di sviluppo con l'applicazione Managed Service for Apache Flink.

Argomenti

- [Le migliori pratiche per il rollback del sistema](#)
- [Le migliori pratiche di configurazione di Hudi](#)
- [Grafici a candela di Apache Flink](#)
- [Problema con il provider di credenziali con il connettore EFO 1.15.2](#)
- [Applicazioni con connettori Kinesis non supportati](#)
- [Errore di compilazione: «Impossibile risolvere le dipendenze per il progetto»](#)
- [Scelta non valida: «kinesisanalyticsv2»](#)
- [UpdateApplication l'azione non consiste nel ricaricare il codice dell'applicazione](#)
- [S3 StreamingFileSink FileNotFoundExceptions](#)
- [FlinkKafkaConsumer problema con stop with savepoint](#)
- [Blocco di Async sink per Flink 1.15](#)
- [L'elaborazione del codice sorgente dei flussi di dati di Amazon Kinesis non funziona correttamente durante il re-sharding](#)

- [Domande frequenti e risoluzione dei problemi relativi ai progetti di incorporamento vettoriale in tempo reale](#)

Le migliori pratiche per il rollback del sistema

Con il rollback automatico del sistema e le funzionalità di visibilità delle operazioni in Amazon Managed Service for Apache Flink, puoi identificare e risolvere i problemi con le tue applicazioni.

Rollback del sistema

Se l'operazione di aggiornamento o ridimensionamento dell'applicazione non riesce a causa di un errore del cliente, ad esempio un bug nel codice o un problema di autorizzazione, Amazon Managed Service for Apache Flink tenta automaticamente di ripristinare la versione precedente in esecuzione se hai attivato questa funzionalità. Per ulteriori informazioni, consulta [Abilita i rollback di sistema per la tua applicazione Managed Service for Apache Flink](#). Se l'autorollback fallisce o se non hai effettuato l'attivazione o l'annullamento del consenso, l'applicazione verrà inserita nello stato. READY Per aggiornare la tua applicazione, completa i seguenti passaggi:

Rollback manuale

Se l'applicazione non procede ed è in uno stato transitorio da molto tempo, o se l'applicazione è passata correttamenteRunning, ma si riscontrano problemi a valle come errori di elaborazione in un'applicazione Flink aggiornata correttamente, è possibile ripristinarla manualmente utilizzando l'API. RollbackApplication

1. ChiamaRollbackApplication: questa operazione ripristinerà la versione precedente in esecuzione e ripristinerà lo stato precedente.
2. Monitora l'operazione di rollback utilizzando l'DescribeApplicationOperationAPI.
3. Se il rollback fallisce, utilizza i passaggi precedenti di rollback del sistema.

Visibilità delle operazioni

L'ListApplicationOperationsAPI mostra la cronologia di tutte le operazioni dei clienti e del sistema sull'applicazione.

1. Ottieni l'OperationID dell'operazione non riuscita dall'elenco.
2. Chiama DescribeApplicationOperation e controlla lo stato e lo StatusDescription.
3. Se un'operazione non è riuscita, la descrizione indica un potenziale errore da esaminare.

Bug comuni relativi ai codici di errore: utilizzate le funzionalità di rollback per ripristinare l'ultima versione funzionante. Risolvi i bug e riprova l'aggiornamento.

Problemi di autorizzazione: utilizza il `DescribeApplicationOperation` per visualizzare le autorizzazioni richieste. Aggiorna le autorizzazioni dell'applicazione e riprova.

Problemi relativi al servizio Amazon Managed Service for Apache Flink: verifica AWS Health Dashboard o apri una richiesta di supporto.

Le migliori pratiche di configurazione di Hudi

Per eseguire i connettori Hudi su Managed Service for Apache Flink, consigliamo le seguenti modifiche alla configurazione.

Disabilitazione di `hoodie.embed.timeline.server`

Il connettore Hudi su Flink configura un server timeline (TM) incorporato sul Flink jobmanager (JM) per memorizzare nella cache i metadati per migliorare le prestazioni quando il parallelismo dei lavori è elevato. Ti consigliamo di disabilitare questo server incorporato su Managed Service for Apache Flink perché disabilitiamo la comunicazione non Flink tra JM e TM.

Se questo server è abilitato, Hudi write tenterà prima di connettersi al server integrato su JM, quindi tornerà alla lettura dei metadati da Amazon S3. Ciò significa che Hudi subisce un timeout di connessione che ritarda le scritture di Hudi e causa un impatto sulle prestazioni del servizio gestito per Apache Flink.

Grafici a candela di Apache Flink

I grafici a candela sono abilitati per impostazione predefinita nelle applicazioni nelle versioni di servizio gestito per Apache Flink che li supportano. I grafici a candela possono influire sulle prestazioni dell'applicazione se rimangono aperti, come indicato nella [documentazione di Flink](#).

Per disabilitare i grafici a candela per l'applicazione è sufficiente creare un caso e richiederne la disabilitazione per l'ARN dell'applicazione. Per ulteriori informazioni, consulta il [Centro di supporto AWS](#).

Problema con il provider di credenziali con il connettore EFO 1.15.2

Esiste un [problema noto](#) con le versioni del connettore EFO del flusso di dati Kinesis fino alla 1.15.2, in cui `FlinkKinesisConsumer` non rispetta la configurazione `Credential Provider`. Le

configurazioni valide vengono ignorate a causa del problema, che comporta l'utilizzo del provider di credenziali AUTO. Ciò può causare problemi nell'utilizzo dell'accesso multi-account a Kinesis tramite il connettore EFO.

Per risolvere questo errore, è sufficiente utilizzare la versione 1.15.3 o successiva del connettore EFO.

Applicazioni con connettori Kinesis non supportati

Managed Service for Apache Flink per Apache Flink versione 1.15 o successiva [rifiuterà automaticamente l'avvio o l'aggiornamento delle applicazioni se utilizzano versioni di Kinesis Connector](#) non supportate (precedenti alla versione 1.15.2) raggruppate in applicazioni o archivi (ZIP). JARs

Errore di rifiuto

Verrà visualizzato il seguente errore quando si inviano chiamate di creazione/aggiornamento all'applicazione tramite:

```
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
```

Come risolvere il problema

- Aggiorna la dipendenza dell'applicazione da `flink-connector-kinesis`. Se utilizzi Maven come strumento di compilazione del progetto, segui [Aggiorna una dipendenza da Maven](#). Se utilizzi Gradle, segui [Aggiorna una dipendenza da Gradle](#).
- Crea di nuovo un pacchetto con l'applicazione.
- Carica oggetti in un bucket Amazon S3.
- Invia nuovamente la richiesta di creazione/aggiornamento con l'applicazione modificata appena caricata nel bucket Amazon S3.
- Se il messaggio di errore continua a ripresentarsi, controlla nuovamente le dipendenze dell'applicazione. Crea un ticket per richiedere assistenza nel caso in cui il problema persista.

Aggiorna una dipendenza da Maven

1. Apri il file `pom.xml` del progetto.
2. Individua le dipendenze del progetto. Hanno questo aspetto:

```
<project>

  ...

  <dependencies>

    ...

    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-connector-kinesis</artifactId>
    </dependency>

    ...

  </dependencies>

  ...

</project>
```

3. Aggiorna `flink-connector-kinesis` a una versione uguale o successiva alla 1.15.2. Per esempio:

```
<project>

  ...

  <dependencies>

    ...

    <dependency>
      <groupId>org.apache.flink</groupId>
      <artifactId>flink-connector-kinesis</artifactId>
      <version>1.15.2</version>
    </dependency>

  ...

</project>
```

```
...

</dependencies>

...

</project>
```

Aggiorna una dipendenza da Gradle

1. Apri il `build.gradle` del progetto (o `build.gradle.kts` per le applicazioni Kotlin).
2. Individua le dipendenze del progetto. Hanno questo aspetto:

```
...

dependencies {

    ...

    implementation("org.apache.flink:flink-connector-kinesis")

    ...

}

...
```

3. Aggiorna `flink-connector-kinesis` a una versione uguale o successiva alla 1.15.2. Per esempio:

```
...

dependencies {

    ...

    implementation("org.apache.flink:flink-connector-kinesis:1.15.2")

    ...

}
```

...

Errore di compilazione: «Impossibile risolvere le dipendenze per il progetto»

Per compilare il servizio gestito per le applicazioni di esempio di Apache Flink, è necessario scaricare e compilare il connettore Kinesis di Apache Flink e aggiungerlo al repository locale di Maven. Se il connettore non è stato aggiunto al repository verrà visualizzato un errore di compilazione simile al seguente:

```
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
```

Per risolvere questo errore, è necessario scaricare il codice sorgente di Apache Flink (dalla versione 1.8.2) per il connettore. <https://flink.apache.org/downloads.html> Per istruzioni su come scaricare, compilare e installare il codice sorgente di Apache Flink, consulta [the section called “Utilizzo del connettore di flussi Apache Flink Kinesis con versioni precedenti di Apache Flink”](#).

Scelta non valida: «kinesisanalyticsv2»

Per utilizzare l'API v2 del servizio gestito per Apache Flink è necessaria la versione più recente di AWS Command Line Interface (AWS CLI).

[Per informazioni sull'aggiornamento di, vedere Installazione di nella Guida per l'utente. AWS CLI](#)
[AWS Command Line Interface](#)

UpdateApplication l'azione non consiste nel ricaricare il codice dell'applicazione

L'[UpdateApplication](#) azione non ricaricherà il codice dell'applicazione con lo stesso nome di file se non viene specificata la versione dell'oggetto S3. Per ricaricare il codice dell'applicazione con lo stesso nome di file, è necessario abilitare il controllo delle versioni sul bucket S3 e specificare la nuova versione dell'oggetto utilizzando il parametro `ObjectVersionUpdate`. Per ulteriori informazioni sul controllo delle versioni degli oggetti in un bucket S3, consulta [Abilitazione e sospensione del controllo delle versioni](#).

S3 StreamingFileSink FileNotFoundExceptions

Le applicazioni del servizio gestito per Apache Flink possono essere eseguite in un file delle parti in corso `FileNotFoundException` quando si parte da snapshot se manca un file delle parti in corso a cui fa riferimento il relativo savepoint. Quando si verifica questa modalità di errore, lo stato dell'operatore dell'applicazione di servizio gestito per Apache Flink non è in genere ripristinabile e deve essere riavviato senza utilizzare uno snapshot `SKIP_RESTORE_FROM_SNAPSHOT`. Vedi il seguente esempio di stack trace:

```
java.io.FileNotFoundException: No such file or directory: s3://amzn-s3-demo-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 scrive i record su file system supportati dai File System.](#) Dato che i flussi in entrata possono essere illimitati, i dati vengono organizzati in file delle parti di dimensioni limitate; man mano che i dati vengono scritti, i nuovi file sono aggiunti. Il ciclo di vita delle parti e la politica di rollover determinano le tempistiche, la dimensione e la denominazione dei file delle parti.

Durante il checkpoint e il savepoint (cattura degli snapshot), tutti i file in sospeso vengono rinominati e salvati. Tuttavia, i file delle parti in corso non sono salvati, ma rinominati; il riferimento viene mantenuto nei metadati del checkpoint o del savepoint, in modo da poter essere utilizzato durante il ripristino dei processi. I file delle parti in corso verranno infine contrassegnati come in sospeso, rinominati e salvati da un checkpoint o savepoint successivi.

Di seguito sono riportate le cause principali e le misure di mitigazione nel caso in cui manchino i file delle parti in corso:

- Istantanea obsoleta utilizzata per avviare l'applicazione Managed Service for Apache Flink: solo l'ultima istantanea del sistema scattata quando un'applicazione viene arrestata o aggiornata può essere utilizzata per avviare un'applicazione Managed Service for Apache Flink con Amazon S3. `StreamingFileSink` Per evitare questa classe di errori è consigliabile utilizzare lo snapshot di sistema più recente.
- Ciò accade ad esempio quando si sceglie uno snapshot creato utilizzando `CreateSnapshot` invece di uno innescato dal sistema durante l'arresto o l'aggiornamento. Il punto di salvataggio della vecchia istantanea conserva un out-of-date riferimento al file di parte in corso che è stato rinominato e salvato dal checkpoint o dal punto di salvataggio successivo.
- Ciò può verificarsi anche quando viene selezionato uno snapshot innescato dal sistema da un evento Stop/Aggiorna che non sia il più recente. Un esempio è un'applicazione con lo snapshot del sistema disabilitato ma `RESTORE_FROM_LATEST_SNAPSHOT` configurato. In genere, le applicazioni Managed Service for Apache Flink con Amazon `StreamingFileSink` S3 devono sempre avere lo snapshot di sistema abilitato e configurato. `RESTORE_FROM_LATEST_SNAPSHOT`
- File delle parti In corso rimosso: poiché il file delle parti In corso si trova in un bucket S3, può essere rimosso da altri componenti o attori che hanno accesso al bucket.
 - [Ciò può accadere quando l'app è stata interrotta per troppo tempo e il file di parte in corso a cui fa riferimento il punto di salvataggio dell'app è stato rimosso dalla politica del ciclo di vita del bucket S3. `MultiPartUpload`](#) Per evitare questa classe di errori, assicurarsi che la politica sul ciclo di vita del bucket MPU S3 si estenda su un periodo sufficientemente ampio per il caso d'utilizzo.
- Ciò può accadere anche quando il file delle parti In corso è stato rimosso manualmente o da un altro componente del sistema. Per evitare questa classe di errori, assicurarsi che i file delle parti In corso non vengano rimossi da altri attori o componenti.
- Condizione di competizione in cui viene innescato un checkpoint automatico dopo il checkpoint: ciò influisce sulle versioni di servizio gestito per Apache Flink fino alla 1.13 inclusa. Questo problema è stato risolto nella versione 1.15 di Managed Service for Apache Flink. Esegui la migrazione dell'applicazione alla versione più recente di Managed Service for Apache Flink per evitare che si verifichino recidive. Ti suggeriamo anche di migrare da a. `StreamingFileSink` [FileSink](#)
- Quando le applicazioni vengono arrestate o aggiornate, il servizio gestito per Apache Flink innesca un savepoint e arresta l'applicazione in due passaggi. Se tra i due passaggi si innesca un checkpoint automatico il savepoint sarà inutilizzabile, in quanto il relativo file delle parti in corso verrebbe rinominato e potenzialmente salvato.

FlinkKafkaConsumer problema con stop with savepoint

Quando si utilizza la versione precedente FlinkKafkaConsumer , è possibile che l'applicazione rimanga bloccata in UPDATING, STOP o SCALING, se sono abilitate le istantanee di sistema. Non è disponibile alcuna correzione pubblicata per questo [problema](#), pertanto si consiglia di eseguire l'aggiornamento alla nuova versione per [KafkaSource](#) mitigare il problema.

Se si utilizza FlinkKafkaConsumer con gli snapshot abilitati, esiste la possibilità che, quando il processo Flink elabora un arresto con una richiesta di savepoint API, FlinkKafkaConsumer possa non riuscire e mostrare un errore di runtime che segnala un `ClosedException`. In queste condizioni l'applicazione Flink si blocca e dà luogo a checkpoint non riusciti.

Blocco di Async sink per Flink 1.15

Esiste un [problema noto](#) con i AWS connettori per l'interfaccia di implementazione di Apache Flink. AsyncSink Ciò riguarda le applicazioni che utilizzano Flink 1.15 con i seguenti connettori:

- Per applicazioni Java:
 - KinesisStreamsSink – `org.apache.flink:flink-connector-kinesis`
 - KinesisStreamsSink – `org.apache.flink:flink-connector-aws-kinesis-streams`
 - KinesisFirehoseSink – `org.apache.flink:flink-connector-aws-kinesis-firehose`
 - DynamoDbSink – `org.apache.flink:flink-connector-dynamodb`
- Applicazioni Flink: SQL/TableAPI/Python
 - kinesis – `org.apache.flink:flink-sql-connector-kinesis`
 - kinesis – `org.apache.flink:flink-sql-connector-aws-kinesis-streams`
 - firehose – `org.apache.flink:flink-sql-connector-aws-kinesis-firehose`
 - dynamodb – `org.apache.flink:flink-sql-connector-dynamodb`

Le applicazioni interessate presenteranno le seguenti caratteristiche:

- il processo di Flink è nello stato `RUNNING`, ma non elabora i dati;
- non si verificano riavvii del processo;
- I checkpoint sono in fase di interruzione.

Il problema è causato da un [bug](#) nell' AWS SDK che impedisce la visualizzazione di alcuni errori al chiamante quando utilizza il client HTTP asincrono. Il risultato è che il sink continuerà ad attendere il completamento di una "richiesta in corso" durante un'operazione di pulizia del checkpoint.

Questo problema è stato risolto in AWS SDK a partire dalla versione 2.20.144.

Di seguito sono riportate le istruzioni su come aggiornare i connettori interessati per utilizzare la nuova versione di AWS SDK nelle applicazioni:

Argomenti

- [Aggiornamento dell'applicazione Java.](#)
- [Aggiornamento di applicazioni Python](#)

Aggiornamento dell'applicazione Java.

Per aggiornare le applicazioni Java è sufficiente seguire il seguente procedimento:

`flink-connector-kinesis`

Se l'applicazione utilizza `flink-connector-kinesis`:

Il connettore Kinesis utilizza lo shading per impacchettare alcune dipendenze, incluso l' AWS SDK, nella barra dei connettori. Per aggiornare la versione AWS SDK, utilizza la seguente procedura per sostituire queste classi ombreggiate:

Maven

1. Aggiungi il connettore Kinesis e i moduli AWS SDK richiesti come dipendenze del progetto.
2. Configurare `maven-shade-plugin`:
 - a. Aggiungi un filtro per escludere le classi AWS SDK ombreggiate durante la copia del contenuto del jar del connettore Kinesis.
 - b. Aggiungi la regola di riposizionamento per spostare le classi AWS SDK aggiornate nel pacchetto, prevista dal connettore Kinesis.

`pom.xml`

```
<project>
  ...
```

```

<dependencies>
  ...
  <dependency>
    <groupId>org.apache.flink</groupId>
    <artifactId>flink-connector-kinesis</artifactId>
    <version>1.15.4</version>
  </dependency>

  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>kinesis</artifactId>
    <version>2.20.144</version>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>netty-nio-client</artifactId>
    <version>2.20.144</version>
  </dependency>
  <dependency>
    <groupId>software.amazon.awssdk</groupId>
    <artifactId>sts</artifactId>
    <version>2.20.144</version>
  </dependency>
  ...
</dependencies>
...
<build>
  ...
  <plugins>
    ...
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-shade-plugin</artifactId>
      <version>3.1.1</version>
      <executions>
        <execution>
          <phase>package</phase>
          <goals>
            <goal>shade</goal>
          </goals>
          <configuration>
            ...
            <filters>
              ...
            </filters>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>

```

```

        <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. Aggiungi il connettore Kinesis e i moduli AWS SDK richiesti come dipendenze del progetto.
2. Modificare la configurazione di shadowJar:
 - a. Escludi le classi AWS SDK ombreggiate quando copi il contenuto del jar del connettore Kinesis.
 - b. Trasferisci le classi AWS SDK aggiornate in un pacchetto previsto dal connettore Kinesis.

build.gradle

```

...
dependencies {
    ...
    flinkShadowJar("org.apache.flink:flink-connector-kinesis:1.15.4")

    flinkShadowJar("software.amazon.awssdk:kinesis:2.20.144")
    flinkShadowJar("software.amazon.awssdk:sts:2.20.144")
    flinkShadowJar("software.amazon.awssdk:netty-nio-client:2.20.144")
    ...
}
...
shadowJar {
    configurations = [project.configurations.flinkShadowJar]

    exclude("software/amazon/kinesis/shaded/software/amazon/awssdk/**/*")
    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")  
}  
...
```

Altri connettori interessati

Se l'applicazione utilizza un altro connettore interessato:

Per aggiornare la versione SDK, la versione AWS SDK deve essere applicata nella configurazione di build del progetto.

Maven

Aggiungi la distinta base AWS SDK (BOM) alla sezione di gestione delle dipendenze del `pom.xml` file per applicare la versione SDK per il progetto.

`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

Aggiungi la dipendenza della piattaforma dalla distinta base (BOM) dell' AWS SDK per applicare la versione SDK per il progetto. Sarà necessaria la versione Gradle 5.0 o versioni successive:

build.gradle

```
...
dependencies {
    ...
    flinkShadowJar(platform("software.amazon.awssdk:bom:2.20.144"))
    ...
}
...
```

Aggiornamento di applicazioni Python

Le applicazioni Python possono utilizzare i connettori in due maniere diverse: per impacchettare connettori e altre dipendenze Java come parte di un singolo uber jar o utilizzare direttamente il connettore jar. Per correggere le applicazioni interessate dal blocco di Async sink:

- Se l'applicazione utilizza un uber jar, seguire le istruzioni per [Aggiornamento dell'applicazione Java](#).
- Per ricostruire i connettori jar dalla sorgente, seguire i passaggi seguenti:

Creazione di connettori dalla fonte:

Prerequisiti, simili ai [requisiti per la costruzione](#) di Flink:

- Java 11
- Maven 3.2.5

flink-sql-connector-kinesis

1. Scarica il codice sorgente per Flink 1.15.4:

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Decomprimi il codice sorgente:

```
tar -xvf flink-1.15.4-src.tgz
```

3. Vai alla directory del connettore Kinesis

```
cd flink-1.15.4/flink-connectors/flink-connector-kinesis/
```

4. Compila e installa il connettore jar, specificando la versione SDK richiesta. AWS Per velocizzare la compilazione, usa `-DskipTests` per saltare l'esecuzione dei test e `-Dfast` per saltare i controlli aggiuntivi del codice sorgente:

```
mvn clean install -DskipTests -Dfast -Daws.sdkv2.version=2.20.144
```

5. Vai alla directory del connettore Kinesis

```
cd ../flink-sql-connector-kinesis
```

6. Compilare e installare il connettore jar sql:

```
mvn clean install -DskipTests -Dfast
```

7. Il jar risultante sarà disponibile all'indirizzo:

```
target/flink-sql-connector-kinesis-1.15.4.jar
```

flink-sql-connector-aws-kinesis-stream

1. Scarica il codice sorgente per Flink 1.15.4:

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Decomprimi il codice sorgente:

```
tar -xvf flink-1.15.4-src.tgz
```

3. Vai alla directory del connettore Kinesis

```
cd flink-1.15.4/flink-connectors/flink-connector-aws-kinesis-streams/
```

4. Compila e installa il connector jar, specificando la versione SDK richiesta. AWS Per velocizzare la compilazione, usa `-DskipTests` per saltare l'esecuzione dei test e `-Dfast` per saltare i controlli aggiuntivi del codice sorgente:

```
mvn clean install -DskipTests -Dfast -Daws.sdk.version=2.20.144
```

5. Vai alla directory del connettore Kinesis

```
cd ../flink-sql-connector-aws-kinesis-streams
```

6. Compilare e installare il connettore jar sql:

```
mvn clean install -DskipTests -Dfast
```

7. Il jar risultante sarà disponibile all'indirizzo:

```
target/flink-sql-connector-aws-kinesis-streams-1.15.4.jar
```

flink-sql-connector-aws-kinesis-firehose

1. Scarica il codice sorgente per Flink 1.15.4:

```
wget https://archive.apache.org/dist/flink/flink-1.15.4/flink-1.15.4-src.tgz
```

2. Decomprimi il codice sorgente:

```
tar -xvf flink-1.15.4-src.tgz
```

3. Accedi alla directory del connettore

```
cd flink-1.15.4/flink-connectors/flink-connector-aws-kinesis-firehose/
```

4. Compila e installa il connettore jar, specificando la versione SDK richiesta. AWS Per velocizzare la compilazione, usa `-DskipTests` per saltare l'esecuzione dei test e `-Dfast` per saltare i controlli aggiuntivi del codice sorgente:

```
mvn clean install -DskipTests -Dfast -Daws.sdk.version=2.20.144
```

5. Accedi alla directory del connettore sql

```
cd ../flink-sql-connector-aws-kinesis-firehose
```

6. Compilare e installare il connettore jar sql:

```
mvn clean install -DskipTests -Dfast
```

7. Il jar risultante sarà disponibile all'indirizzo:

```
target/flink-sql-connector-aws-kinesis-firehose-1.15.4.jar
```

flink-sql-connector-dynamodb

1. Scarica il codice sorgente per 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. Decomprimi il codice sorgente:

```
tar -xvf flink-connector-aws-3.0.0-src.tgz
```

3. Accedi alla directory del connettore

```
cd flink-connector-aws-3.0.0
```

4. Compila e installa il connettore jar, specificando la versione SDK richiesta. AWS Per velocizzare la compilazione, usa `-DskipTests` per saltare l'esecuzione dei test e `-Dfast` per saltare i controlli aggiuntivi del codice sorgente:

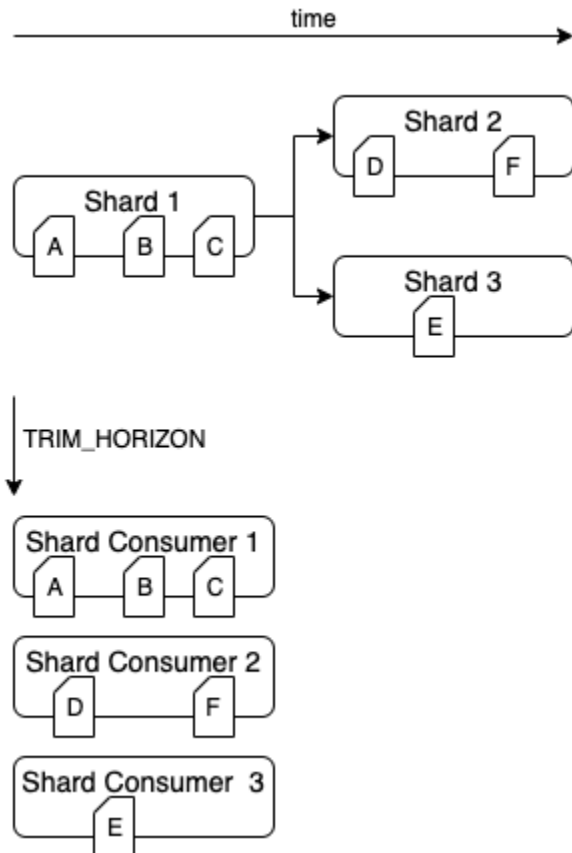
```
mvn clean install -DskipTests -Dfast -Dflink.version=1.15.4 -  
Daws.sdk.version=2.20.144
```

5. Il jar risultante sarà disponibile all'indirizzo:

```
flink-sql-connector-dynamodb/target/flink-sql-connector-dynamodb-3.0.0.jar
```

L'elaborazione del codice sorgente dei flussi di dati di Amazon Kinesis non funziona correttamente durante il re-sharding

L'implementazione attuale di `FlinkKinesisConsumer` non fornisce forti garanzie di ordinamento tra gli shard Kinesis. Ciò può comportare l'out-of-order elaborazione durante la ripartizione di Kinesis Stream, in particolare per le applicazioni Flink che presentano ritardi di elaborazione. In alcune circostanze, ad esempio gli operatori delle finestre basati sugli orari degli eventi, gli eventi potrebbero essere eliminati a causa del ritardo che ne deriva.



Si tratta di un [problema noto](#) in Open source di Flink. Fino a quando non sarà disponibile la correzione del connettore, è fondamentale assicurarsi che le applicazioni Flink non accumulino ritardi rispetto al flusso di dati Kinesis nel corso del partizionamento. Assicurandoti che il ritardo di elaborazione sia tollerato dalle tue app Flink, puoi ridurre al minimo l'impatto dell'elaborazione e il rischio di perdita dei out-of-order dati.

Domande frequenti e risoluzione dei problemi relativi ai progetti di incorporamento vettoriale in tempo reale

Consulta le seguenti domande frequenti e le sezioni sulla risoluzione dei problemi relativi ai blueprint di incorporamento vettoriale in tempo reale. [Per ulteriori informazioni sui blueprint di incorporamento vettoriale in tempo reale, consulta Blueprint di incorporamento vettoriale in tempo reale.](#)

[Per la risoluzione generale dei problemi relativi all'applicazione Managed Service for Apache Flink, consulta -runtime.html. <https://docs.aws.amazon.com/managed-flink/latest/java/troubleshooting>](#)

Argomenti

- [Progetti di incorporamento vettoriale in tempo reale - Domande frequenti](#)
- [Blueprint di incorporamento vettoriale in tempo reale: risoluzione dei problemi](#)

Progetti di incorporamento vettoriale in tempo reale - Domande frequenti

Consulta le seguenti domande frequenti sui progetti di incorporamento vettoriale in tempo reale. [Per ulteriori informazioni sui blueprint di incorporamento vettoriale in tempo reale, consulta Blueprint di incorporamento vettoriale in tempo reale.](#)

Domande frequenti

- [Quali risorse crea AWS questo modello?](#)
- [Quali sono le mie azioni una volta completata l'implementazione dello AWS CloudFormation stack?](#)
- [Quale dovrebbe essere la struttura dei dati negli argomenti di origine di Amazon MSK?](#)
- [Posso specificare parti di un messaggio da incorporare?](#)
- [Posso leggere dati da più argomenti di Amazon MSK?](#)
- [Posso usare regex per configurare i nomi degli argomenti di Amazon MSK?](#)
- [Qual è la dimensione massima di un messaggio che può essere letto da un argomento di Amazon MSK?](#)
- [Che tipo di è supportato? OpenSearch](#)
- [Perché devo utilizzare una raccolta di ricerca vettoriale, un indice vettoriale e aggiungere un campo vettoriale nella mia raccolta Serverless? OpenSearch](#)
- [Cosa devo impostare come dimensione per il mio campo vettoriale?](#)
- [Che aspetto ha l'output nell'indice OpenSearch configurato?](#)

- [Posso specificare campi di metadati da aggiungere al documento memorizzato nell'indice? OpenSearch](#)
- [Devo aspettarmi voci duplicate nell' OpenSearchindice?](#)
- [Posso inviare dati a più OpenSearch indici?](#)
- [Posso implementare più applicazioni di incorporamento vettoriale in tempo reale in un'unica applicazione? Account AWS](#)
- [Più applicazioni di incorporamento vettoriale in tempo reale possono utilizzare la stessa fonte di dati o lo stesso sink?](#)
- [L'applicazione supporta la connettività tra account?](#)
- [L'applicazione supporta la connettività tra regioni?](#)
- [Il mio cluster e la mia OpenSearch raccolta Amazon MSK possono trovarsi in VPCs sottoreti diverse?](#)
- [Quali modelli di incorporamento sono supportati dall'applicazione?](#)
- [Posso ottimizzare le prestazioni della mia applicazione in base al mio carico di lavoro?](#)
- [Quali tipi di autenticazione Amazon MSK sono supportati?](#)
- [Cos'è sink.os.bulkFlushIntervalMillis e come posso impostarlo?](#)
- [Quando distribuisco la mia applicazione Managed Service for Apache Flink, da quale punto dell'argomento Amazon MSK inizierà a leggere i messaggi?](#)
- [Come posso usare sink.os.starting.offset?](#)
- [Quali strategie di suddivisione in blocchi sono supportate?](#)
- [Come posso leggere i record nel mio datastore vettoriale?](#)
- [Dove posso trovare nuovi aggiornamenti al codice sorgente?](#)
- [Posso apportare una modifica al AWS CloudFormation modello e aggiornare l'applicazione Managed Service for Apache Flink?](#)
- [AWS Monitorerà e gestirà l'applicazione per mio conto?](#)
- [Questa applicazione sposta i miei dati al di fuori del mio Account AWS?](#)

Quali risorse crea AWS questo modello?

Per trovare le risorse distribuite nel tuo account, accedi alla AWS CloudFormation console e identifica il nome dello stack che inizia con il nome che hai fornito per l'applicazione Managed Service for

Apache Flink. Scegli la scheda Risorse per controllare le risorse che sono state create come parte dello stack. Le seguenti sono le risorse chiave create dallo stack:

- Servizio gestito di incorporamento vettoriale in tempo reale per l'applicazione Apache Flink
- Bucket Amazon S3 per contenere il codice sorgente per l'applicazione di incorporamento vettoriale in tempo reale
- CloudWatch gruppo di log e flusso di log per l'archiviazione dei log
- Funzioni Lambda per il recupero e la creazione di risorse
- Ruoli e policy IAM per Lambdas, Managed Service for Apache Flink e accesso ad Amazon Bedrock e Amazon Service OpenSearch
- Politica di accesso ai dati per Amazon OpenSearch Service
- Endpoint VPC per accedere ad Amazon Bedrock e Amazon Service OpenSearch

Quali sono le mie azioni una volta completata l'implementazione dello AWS CloudFormation stack?

Una volta completata la distribuzione AWS CloudFormation dello stack, accedi alla console Managed Service for Apache Flink e trova l'applicazione blueprint Managed Service for Apache Flink. Scegli la scheda Configura e conferma che tutte le proprietà di runtime siano configurate correttamente. Potrebbero passare alla pagina successiva. Quando sei sicuro delle impostazioni, scegli Esegui. L'applicazione inizierà a importare i messaggi dal tuo argomento.

[Per verificare la presenza di nuove versioni, <https://github.com/aws-labs/real-time-vectorization-of-streaming-data/releases>.](https://github.com/aws-labs/real-time-vectorization-of-streaming-data/releases)

Quale dovrebbe essere la struttura dei dati negli argomenti di origine di Amazon MSK?

Al momento supportiamo dati di origine strutturati e non strutturati.

- I dati non strutturati sono indicati con `in.STRING source.msk.data.type` I dati vengono letti così come sono dal messaggio in arrivo.
- Attualmente supportiamo i dati JSON strutturati, indicati con `in.JSON source.msk.data.type` I dati devono essere sempre in formato JSON. Se l'applicazione riceve un codice JSON non valido, l'applicazione fallirà.
- Quando utilizzi JSON come tipo di dati di origine, assicurati che ogni messaggio in tutti gli argomenti di origine sia un JSON valido. Se sottoscrivi uno o più argomenti che non contengono oggetti JSON con questa impostazione, l'applicazione avrà esito negativo. Se uno o più argomenti

contengono una combinazione di dati strutturati e non strutturati, si consiglia di configurare i dati di origine come non strutturati nell'applicazione Managed Service for Apache Flink.

Posso specificare parti di un messaggio da incorporare?

- Per i `source.msk.data.type` dati di input non strutturati `STRING`, l'applicazione incorporerà sempre l'intero messaggio e memorizzerà l'intero messaggio nell'indice configurato. `OpenSearch`
- Per i dati di input strutturati `where source.msk.data.type is JSON`, è possibile configurare in modo `embed.input.config.json.fieldsToEmbed` da specificare quale campo dell'oggetto JSON deve essere selezionato per l'incorporamento. Funziona solo per i campi JSON di primo livello e non funziona con i messaggi annidati JSONs e contenenti un array JSON. Usa `.*` per incorporare l'intero JSON.

Posso leggere dati da più argomenti di Amazon MSK?

Sì, con questa applicazione puoi leggere dati da più argomenti di Amazon MSK. I dati di tutti gli argomenti devono essere dello stesso tipo (`STRING` o `JSON`), altrimenti l'applicazione potrebbe fallire. I dati di tutti gli argomenti vengono sempre archiviati in un unico `OpenSearch` indice.

Posso usare regex per configurare i nomi degli argomenti di Amazon MSK?

`source.msk.topic.names` non supporta un elenco di espressioni regolari. Supportiamo un elenco separato da virgole di nomi di argomenti o `.*` espressioni regolari per includere tutti gli argomenti.

Qual è la dimensione massima di un messaggio che può essere letto da un argomento di Amazon MSK?

La dimensione massima di un messaggio che può essere elaborato è limitata dal limite di `InvokeModel` corpi di Amazon Bedrock, attualmente impostato su 25.000.000. Per ulteriori informazioni, consulta [InvokeModel](#).

Che tipo di è supportato? `OpenSearch`

Supportiamo sia i `OpenSearch` domini che le raccolte. Se utilizzi una `OpenSearch` raccolta, assicurati di utilizzare una raccolta vettoriale e di creare un indice vettoriale da utilizzare per questa applicazione. Questo vi permetterà di utilizzare le funzionalità del database `OpenSearch` vettoriale per interrogare i vostri dati. Per ulteriori informazioni, consulta la [spiegazione delle funzionalità dei database vettoriali di Amazon OpenSearch Service](#).

Perché devo utilizzare una raccolta di ricerca vettoriale, un indice vettoriale e aggiungere un campo vettoriale nella mia raccolta Serverless? OpenSearch

Il tipo di raccolta di ricerca vettoriale in OpenSearch Serverless offre una funzionalità di ricerca per similarità scalabile e ad alte prestazioni. Semplifica la creazione di moderne esperienze di ricerca aumentata di machine learning (ML) e applicazioni generative di intelligenza artificiale (AI). Per ulteriori informazioni, consulta [Lavorare con le raccolte di ricerca vettoriale](#).

Cosa devo impostare come dimensione per il mio campo vettoriale?

Imposta la dimensione del campo vettoriale in base al modello di incorporamento che desideri utilizzare. Fate riferimento alla tabella seguente e confermate questi valori dalla rispettiva documentazione.

Dimensioni dei campi vettoriali

Nome del modello di incorporamento vettoriale Amazon Bedrock	Supporto per le dimensioni di output offerto dal modello
Incorporamenti di testo Amazon Titan V1	1.536
Incorporamenti di testo Amazon Titan V2	1.024 (impostazione predefinita), 384, 256
Amazon Titan Multimodal Embeddings G1	1.024 (impostazione predefinita), 384, 256
Cohere Embed English	1,024
Cohere Embed Multilingual	1,024

Che aspetto ha l'output nell'indice OpenSearch configurato?

Ogni documento dell' OpenSearch indice contiene i seguenti campi:

- `original_data`: i dati utilizzati per generare gli incorporamenti. Per il tipo `STRING`, è l'intero messaggio. Per l'oggetto `JSON`, è l'oggetto `JSON` utilizzato per gli incorporamenti. Potrebbe essere l'intero `JSON` nel messaggio o i campi specificati nel `JSON`. Ad esempio, se il nome fosse selezionato per essere incorporato nei messaggi in arrivo, l'output sarebbe il seguente:

```
"original_data": "{ \"name\": \"John Doe\" }"
```

- `embedded_data`: una matrice vettoriale float di incorporamenti generata da Amazon Bedrock

- data: timestamp UTC in cui il documento è stato archiviato OpenSearch

Posso specificare campi di metadati da aggiungere al documento memorizzato nell'indice?
OpenSearch

No, al momento non supportiamo l'aggiunta di campi aggiuntivi al documento finale memorizzato nell'OpenSearch indice.

Devo aspettarmi voci duplicate nell' OpenSearchindice?

A seconda di come è stata configurata l'applicazione, è possibile che nell'indice vengano visualizzati messaggi duplicati. Un motivo comune è il riavvio dell'applicazione. L'applicazione è configurata per impostazione predefinita per iniziare a leggere dal primo messaggio nell'argomento di origine. Quando si modifica la configurazione, l'applicazione si riavvia ed elabora nuovamente tutti i messaggi dell'argomento. Per evitare la rielaborazione, vedete [Come si usa source.msk.starting.offset?](#) e imposta correttamente l'offset iniziale per la tua applicazione.

Posso inviare dati a più OpenSearch indici?

No, l'applicazione supporta l'archiviazione dei dati in un unico OpenSearch indice. Per configurare l'output di vettorizzazione su più indici, è necessario distribuire un servizio gestito separato per le applicazioni Apache Flink.

Posso implementare più applicazioni di incorporamento vettoriale in tempo reale in un'unica applicazione? Account AWS

Sì, puoi distribuire più servizi gestiti di incorporamento vettoriale in tempo reale per applicazioni Apache Flink in un'unica Account AWS applicazione se ogni applicazione ha un nome univoco.

Più applicazioni di incorporamento vettoriale in tempo reale possono utilizzare la stessa fonte di dati o lo stesso sink?

Sì, puoi creare più servizi gestiti di incorporamento vettoriale in tempo reale per applicazioni Apache Flink che leggono i dati dallo stesso argomento o archiviano i dati nello stesso indice.

L'applicazione supporta la connettività tra account?

No, affinché l'applicazione funzioni correttamente, il cluster Amazon MSK e la OpenSearch raccolta devono trovarsi nello stesso Account AWS punto in cui stai tentando di configurare l'applicazione Managed Service for Apache Flink.

L'applicazione supporta la connettività tra regioni?

No, l'applicazione consente solo di distribuire un'applicazione Managed Service for Apache Flink con un cluster Amazon MSK e una OpenSearch raccolta nella stessa regione dell'applicazione Managed Service for Apache Flink.

Il mio cluster e la mia OpenSearch raccolta Amazon MSK possono trovarsi in VPCs sottoreti diverse?

Sì, supportiamo il cluster e la OpenSearch raccolta Amazon MSK in diverse VPCs sottoreti purché si trovino nella stessa. Account AWS Vedi (Risoluzione dei problemi generali di MSF) per verificare che la configurazione sia corretta.

Quali modelli di incorporamento sono supportati dall'applicazione?

Attualmente, l'applicazione supporta tutti i modelli supportati da Bedrock. Ciò include:

- Titan Embeddings G1 - Text
- Incorporamenti di testo Amazon Titan V2
- Amazon Titan Multimodal Embeddings G1
- Cohere Embed English
- Cohere Embed Multilingual

Posso ottimizzare le prestazioni della mia applicazione in base al mio carico di lavoro?

Sì. La velocità effettiva dell'applicazione dipende da una serie di fattori, tutti controllabili dai clienti:

1. AWS MSF KPU: L'applicazione viene distribuita con fattore di parallelismo predefinito 2 e parallelismo per KPU 1, con ridimensionamento automatico attivato. Tuttavia, ti consigliamo di configurare la scalabilità per l'applicazione Managed Service for Apache Flink in base ai tuoi carichi di lavoro. Per ulteriori informazioni, consulta [Review Managed Service for Apache Flink application Resources](#).
2. Amazon Bedrock: in base al modello on-demand di Amazon Bedrock selezionato, potrebbero essere applicate quote diverse. Controlla le quote di servizio in Bedrock per vedere il carico di lavoro che il servizio sarà in grado di gestire. Per ulteriori informazioni, consulta [Quotas for Amazon Bedrock](#).

3. Amazon OpenSearch Service: inoltre, in alcune situazioni, potresti notare che questo OpenSearch è il collo di bottiglia nella tua pipeline. Per informazioni sulla scalabilità, consulta [OpenSearch Ridimensionamento dei domini OpenSearch Amazon Service](#).

Quali tipi di autenticazione Amazon MSK sono supportati?

Supportiamo solo il tipo di autenticazione IAM MSK.

Cos'è **`sink.os.bulkFlushIntervalMillis`** e come posso impostarlo?

Quando si inviano dati ad Amazon OpenSearch Service, l'intervallo di trasferimento in blocco è l'intervallo in cui viene eseguita la richiesta in blocco, indipendentemente dal numero di azioni o dalla dimensione della richiesta. Il valore predefinito è impostato su 1 millisecondo.

L'impostazione di un intervallo di lavaggio può aiutare a garantire che i dati vengano indicizzati tempestivamente, ma può anche comportare un aumento del sovraccarico se impostato su un valore troppo basso. Considerate il vostro caso d'uso e l'importanza di un'indicizzazione tempestiva quando scegliete un intervallo di flush.

Quando distribuisco la mia applicazione Managed Service for Apache Flink, da quale punto dell'argomento Amazon MSK inizierà a leggere i messaggi?

L'applicazione inizierà a leggere i messaggi dall'argomento Amazon MSK all'offset specificato dalla `source.msk.starting.offset` configurazione impostata nella configurazione di runtime dell'applicazione. Se non `source.msk.starting.offset` è impostato in modo esplicito, il comportamento predefinito dell'applicazione consiste nell'iniziare a leggere dal primo messaggio disponibile nell'argomento.

Come posso usare **`source.msk.starting.offset`**?

Imposta esplicitamente `source.msk.starting.offset` su uno dei seguenti valori, in base al comportamento desiderato:

- EARLEST: L'impostazione predefinita, che legge dall'offset più vecchio nella partizione. Questa è una buona scelta soprattutto se:
 - Hai appena creato argomenti e applicazioni consumer di Amazon MSK.
 - È necessario riprodurre i dati, in modo da poter creare o ricostruire lo stato. Ciò è importante quando si implementa il pattern di approvvigionamento degli eventi o quando si inizializza un nuovo servizio che richiede una visualizzazione completa della cronologia dei dati.

- **ULTIMO**: L'applicazione Managed Service for Apache Flink leggerà i messaggi dalla fine della partizione. Ti consigliamo questa opzione se ti interessa solo la produzione di nuovi messaggi e non hai bisogno di elaborare dati storici. In questa impostazione, il consumatore ignorerà i messaggi esistenti e leggerà solo i nuovi messaggi pubblicati dal produttore originale.
- **COMMITTED**: L'applicazione Managed Service for Apache Flink inizierà a consumare i messaggi provenienti dall'offset impegnato del gruppo di consumatori. Se l'offset confermato non esiste, verrà utilizzata la strategia di ripristino EARLIEST.

Quali strategie di suddivisione in blocchi sono supportate?

Stiamo usando la libreria [langchain](#) per suddividere gli input. Il chunking viene applicato solo se la lunghezza dell'input è maggiore di quella scelta. `maxSegmentSizeInChars` Supportiamo i seguenti cinque tipi di chunking:

- **SPLIT_BY_CHARACTER**: Inserirà il maggior numero di caratteri possibile in ogni blocco in cui la lunghezza di ogni blocco non è maggiore di. `maxSegmentSizeInChars` Non gli interessa lo spazio bianco, quindi può tagliare le parole.
- **SPLIT_BY_WORD**: troverà i caratteri di spaziatura in base ai quali suddividere. Nessuna parola viene interrotta.
- **SPLIT_BY_SENTENCE**: I limiti delle frasi vengono rilevati utilizzando la libreria Apache OpenNLP con il modello di frase in inglese.
- **SPLIT_BY_LINE**: troverà nuovi caratteri di riga in base ai quali suddividere.
- **SPLIT_BY_PARAGRAPH**: troverà nuovi caratteri di riga consecutivi in base ai quali suddividere in blocchi.

Le strategie di suddivisione rientrano nell'ordine precedente, a cui ricorrono le strategie di suddivisione in blocchi più grandi. **SPLIT_BY_PARAGRAPH** **SPLIT_BY_CHARACTER** Ad esempio, quando si utilizza **SPLIT_BY_LINE**, se una riga è troppo lunga, la riga verrà suddivisa in blocchi per frase, dove ogni blocco può contenere quante più frasi possibile. Se ci sono frasi troppo lunghe, verranno suddivise in blocchi a livello di parola. Se una parola è troppo lunga, verrà divisa per carattere.

Come posso leggere i record nel mio datastore vettoriale?

1. Quando è `source.msk.data.type STRING`

- `original_data`: l'intera stringa originale del messaggio Amazon MSK.

- `embedded_data`: vettore di incorporamento creato se non è vuoto (suddivisione in blocchi applicata) o creato da `chunk_data` se non è stato applicato alcun chunking. `original_data`
- `chunk_data`: presente solo quando i dati originali sono stati suddivisi in blocchi. Contiene la parte del messaggio originale utilizzata per creare l'incorporamento. `embedded_data`

2. Quando è `source.msk.data.type` JSON

- `original_data`: l'intero codice JSON originale del messaggio Amazon MSK dopo l'applicazione del filtro delle chiavi JSON.
- `embedded_data`: vettore di incorporamento creato se non è vuoto (suddivisione in blocchi applicata) o creato da `chunk_data` se non è stata applicata alcuna suddivisione in blocchi. `original_data`
- `chunk_key`: presente solo quando i dati originali erano suddivisi in blocchi. Contiene la chiave JSON da cui proviene il blocco. `original_data` Ad esempio, può assomigliare alle chiavi o `jsonKey1.nestedJsonKeyA` ai metadati annidati nell'esempio di `original_data`
- `chunk_data`: presente solo quando i dati originali erano suddivisi in blocchi. Contiene la parte del messaggio originale utilizzata per creare l'incorporamento. `embedded_data`

Sì, con questa applicazione puoi leggere dati da più argomenti di Amazon MSK. I dati di tutti gli argomenti devono essere dello stesso tipo (STRING o JSON), altrimenti l'applicazione potrebbe fallire. I dati di tutti gli argomenti vengono sempre archiviati in un unico OpenSearch indice.

Dove posso trovare nuovi aggiornamenti al codice sorgente?

Vai a <https://github.com/aws-labs/real-time-vectorization-of-streaming-data/releases> per verificare la presenza di nuove versioni.

Posso apportare una modifica al AWS CloudFormation modello e aggiornare l'applicazione Managed Service for Apache Flink?

No, apportare una modifica al AWS CloudFormation modello non aggiorna l'applicazione Managed Service for Apache Flink. Qualsiasi nuova modifica AWS CloudFormation implica la necessità di implementare un nuovo stack.

AWS Monitorerà e gestirà l'applicazione per mio conto?

No, non AWS monitorerà, ridimensionerà, aggiornerà o applicherà patch a questa applicazione per tuo conto.

Questa applicazione sposta i miei dati al di fuori del mio Account AWS?

Tutti i dati letti e archiviati dall'applicazione Managed Service for Apache Flink rimangono all'interno dell'utente Account AWS e non escono mai dal suo account.

Blueprint di incorporamento vettoriale in tempo reale: risoluzione dei problemi

Consulta i seguenti argomenti di risoluzione dei problemi relativi ai blueprint di incorporamento vettoriale in tempo reale. [Per ulteriori informazioni sui blueprint di incorporamento vettoriale in tempo reale, consulta Blueprint di incorporamento vettoriale in tempo reale.](#)

Argomenti sulla risoluzione dei problemi

- [La mia distribuzione CloudFormation dello stack non è riuscita o è stata ripristinata. Cosa posso fare per risolvere il problema?](#)
- [Non voglio che la mia applicazione inizi a leggere i messaggi dall'inizio degli argomenti di Amazon MSK. Cosa devo fare?](#)
- [Come faccio a sapere se c'è un problema con la mia applicazione Managed Service for Apache Flink e come posso eseguirne il debug?](#)
- [Quali sono le metriche chiave che devo monitorare per la mia applicazione Managed Service for Apache Flink?](#)

La mia distribuzione CloudFormation dello stack non è riuscita o è stata ripristinata. Cosa posso fare per risolvere il problema?

- Vai al tuo stack CFN e trova il motivo dell'errore dello stack. Potrebbe essere correlato alla mancanza di autorizzazioni, alle collisioni tra i nomi AWS delle risorse, tra le altre cause. Risolvi la causa principale dell'errore di distribuzione. Per ulteriori informazioni, consulta la [guida alla CloudWatch risoluzione dei problemi](#).
- [Facoltativo] Può esserci un solo endpoint VPC per servizio per VPC. Se hai distribuito più blueprint di incorporamento vettoriale in tempo reale per scrivere nelle raccolte di Amazon OpenSearch Service nello stesso VPC, è possibile che condividano gli endpoint VPC. Questi potrebbero essere già presenti nel tuo account per il VPC oppure il primo stack di blueprint di incorporamento vettoriale in tempo reale creerà endpoint VPC per Amazon Bedrock e Amazon OpenSearch Service che verranno utilizzati da tutti gli altri stack distribuiti nel tuo account. Se uno stack fallisce, controlla se quello stack ha creato endpoint VPC per Amazon Bedrock e Amazon OpenSearch Service ed eliminali se non vengono utilizzati in nessun'altra parte del tuo account. Per i passaggi

per eliminare gli endpoint VPC, [vedi Come posso eliminare in modo sicuro la mia applicazione? \(elimina\)](#).

- Potrebbero esserci altri servizi o applicazioni nel tuo account che utilizzano l'endpoint VPC. La sua eliminazione potrebbe causare interruzioni della rete per altri servizi. Fai attenzione nell'eliminare questi endpoint.

Non voglio che la mia applicazione inizi a leggere i messaggi dall'inizio degli argomenti di Amazon MSK. Cosa devo fare?

È necessario impostare `source.msk.starting.offset` esplicitamente uno dei seguenti valori, a seconda del comportamento desiderato:

- Offset più antico: l'offset più vecchio nella partizione.
- Ultimo offset: i consumatori leggeranno i messaggi dalla fine della partizione.
- Offset confermato: legge l'ultimo messaggio elaborato dal consumatore all'interno di una partizione.

Come faccio a sapere se c'è un problema con la mia applicazione Managed Service for Apache Flink e come posso eseguirne il debug?

Utilizza la [guida alla risoluzione dei problemi relativi a Managed Service for Apache Flink per eseguire il debug dei problemi relativi](#) a Managed Service for Apache Flink con la tua applicazione.

Quali sono le metriche chiave che devo monitorare per la mia applicazione Managed Service for Apache Flink?

- Tutte le metriche disponibili per una normale applicazione Managed Service for Apache Flink possono aiutarti a monitorare la tua applicazione. Per ulteriori informazioni, consulta [Metriche e dimensioni in Managed Service](#) for Apache Flink.
- Per monitorare i parametri di Amazon Bedrock, consulta i parametri Amazon [CloudWatch per Amazon](#) Bedrock.
- Abbiamo aggiunto due nuove metriche per il monitoraggio delle prestazioni di generazione di incorporamenti. Li trovi sotto il nome dell'`EmbeddingGeneration` operazione in CloudWatch Le due metriche sono:
 - `BedrockTitanEmbeddingTokenCount`: numero di token presenti in una singola richiesta ad Amazon Bedrock.

- `BedrockEmbeddingGenerationLatencyMs`: riporta il tempo impiegato per inviare e ricevere una risposta da Amazon Bedrock per la generazione di incorporamenti, in millisecondi.
- Per le raccolte serverless di Amazon OpenSearch Service, puoi utilizzare metriche come `IngestionDocumentErrors` e `IngestionDataRate` altre. Per ulteriori informazioni, consulta [Monitoring OpenSearch Serverless with Amazon CloudWatch](#).
- Per i OpenSearch parametri assegnati, consulta [Monitoraggio dei parametri OpenSearch del cluster con Amazon CloudWatch](#).

Risoluzione dei problemi di runtime

Questa sezione contiene informazioni sulla diagnosi e la risoluzione dei problemi di runtime relativi al servizio gestito per l'applicazione Apache Flink.

Argomenti

- [Strumenti per la risoluzione dei problemi](#)
- [Problemi relativi all'applicazione](#)
- [L'applicazione si sta riavviando](#)
- [La velocità effettiva è troppo lenta](#)
- [Crescita statale illimitata](#)
- [Operatori vincolati all'I/O](#)
- [Limitazione della larghezza di banda della rete in upstream o all'origine da un flusso di dati Kinesis](#)
- [Checkpoint](#)
- [Il checkpoint è in fase di interruzione](#)
- [Errori del checkpoint per l'applicazione Apache Beam](#)
- [Contropressione](#)
- [Disallineamento dei dati](#)
- [Disallineamento dello stato](#)
- [Integrazione con risorse in diverse regioni](#)

Strumenti per la risoluzione dei problemi

Lo strumento principale per rilevare i problemi delle applicazioni sono gli CloudWatch allarmi. Utilizzando gli CloudWatch allarmi, è possibile impostare soglie per le CloudWatch metriche che

indicano condizioni di errore o di collo di bottiglia nell'applicazione. Per informazioni sugli allarmi consigliati, consulta. CloudWatch [Usa gli CloudWatch allarmi con Amazon Managed Service per Apache Flink](#)

Problemi relativi all'applicazione

Questa sezione contiene soluzioni per le situazioni di errore che possono verificarsi con il servizio gestito per l'applicazione Apache Flink.

Argomenti

- [L'applicazione è bloccata in uno stato transitorio](#)
- [La creazione di istantanee non riesce](#)
- [Impossibile accedere alle risorse in un VPC](#)
- [I dati vengono persi durante la scrittura su un bucket Amazon S3](#)
- [L'applicazione è nello stato RUNNING ma non sta elaborando dati](#)
- [Istantanea, aggiornamento dell'applicazione o errore di interruzione dell'applicazione: InvalidApplicationConfigurationException](#)
- [java.nio.file. NoSuchFileException:/usr/local/openjdk-8/lib/security/cacerts](#)

L'applicazione è bloccata in uno stato transitorio

Se l'applicazione rimane in uno stato transitorio (STARTING, UPDATING, o AUTOSCALING) STOPPING, è possibile interromperla utilizzando l'[StopApplication](#) azione con il Force parametro impostato su true. Non è possibile forzare l'interruzione di un'applicazione che si trovi nello stato DELETING. In alternativa, se l'applicazione si trova nello stato UPDATING o AUTOSCALING, è possibile ripristinarla alla versione precedente. Quando si esegue il rollback di un'applicazione, vengono caricati i dati sullo stato dell'ultimo snapshot scattato con successo. Se l'applicazione non dispone di snapshot, il servizio gestito per Apache Flink rifiuta la richiesta di ripristino. Per ulteriori informazioni sul rollback di un'applicazione, vedi [RollbackApplication](#) azione.

Note

L'arresto forzato dell'applicazione può causare la perdita o la duplicazione dei dati. Consigliamo di scattare snapshot con frequenza, per evitare di perdere o duplicare i dati mentre l'applicazione viene riavviata.

Le applicazioni possono bloccarsi a causa delle seguenti ragioni:

- Dimensioni eccessive dello stato dell'applicazione: se lo stato dell'applicazione è troppo grande o troppo ingombrante può causare il blocco dell'applicazione durante un'operazione di checkpoint o di snapshot. Controlla le metriche `lastCheckpointDuration` e `lastCheckpointSize` dell'applicazione, alla ricerca di valori in costante aumento o eccessivamente alti.
- Il codice dell'applicazione è troppo pesante: verifica che il file JAR dell'applicazione non superi 512 MB. I file JAR di dimensioni superiori a 512 MB non sono supportati.
- Generazione di snapshot nell'applicazione non riuscita: il servizio gestito per Apache Flink scatta uno snapshot dell'applicazione nel corso di una richiesta [UpdateApplication](#) o [StopApplication](#). In seguito, il servizio utilizza lo stato di snapshot e ripristina l'applicazione utilizzandone la configurazione aggiornata per fornire semantiche di elaborazione exactly-once. Se la creazione automatica di snapshot non riesce, consulta quanto [La creazione di istantanee non riesce](#) segue:
- Ripristino da uno snapshot non riuscito: rimuovendo o modificando un operatore in un aggiornamento dell'applicazione e tentando il ripristino da uno snapshot, quest'ultimo avrà esito negativo per impostazione predefinita se l'istananea contiene dati sullo stato dell'operatore mancante. Inoltre, l'applicazione rimarrà bloccata nello status STOPPED o UPDATING. Per modificare questo comportamento e consentire il corretto ripristino, modifica il `AllowNonRestoredState` parametro dell'applicazione [FlinkRunConfiguration](#) in `true`. Ciò consentirà all'operazione di ripristino di ignorare i dati sullo stato che non possono essere mappati al nuovo programma.
- L'inizializzazione dell'applicazione richiede più tempo: servizio gestito per Apache Flink utilizza un timeout interno di 5 minuti (impostazione graduale) mentre attende l'avvio di un processo Flink. Se il processo non viene avviato entro questo timeout, verrà visualizzato un CloudWatch registro come segue:

```
Flink job did not start within a total timeout of 5 minutes for application: %s under account: %s
```

l'errore precedentemente descritto si verifica quando le operazioni definite nel metodo `main` del processo Flink impiegano più di 5 minuti, causando il timeout della creazione del processo Flink sul servizio gestito per Apache Flink. Ti consigliamo di controllare JobManager log di Flink e il codice dell'applicazione per vedere se questo ritardo nel `main` metodo è previsto. In caso contrario, è necessario adottare misure per risolvere il problema, in modo che il procedimento richieda meno di 5 minuti.

Per controllare lo status dell'applicazione è sufficiente utilizzare le azioni [ListApplications](#) o [DescribeApplication](#).

La creazione di istantanee non riesce

Il servizio gestito per Apache Flink non può scattare uno snapshot nelle seguenti circostanze:

- l'applicazione ha superato il limite di snapshot; il limite è di mille unità. Per ulteriori informazioni, consulta [Gestisci i backup delle applicazioni utilizzando le istantanee](#).
- L'applicazione non dispone delle autorizzazioni per accedere alla fonte o al sink.
- Il codice dell'applicazione non funziona correttamente.
- L'applicazione presenta altri problemi di configurazione.

Se si verifica un'eccezione durante l'acquisizione di uno snapshot durante l'aggiornamento o l'arresto di un'applicazione, imposta la proprietà `SnapshotsEnabled` del [ApplicationSnapshotConfiguration](#) dell'applicazione su `false` e prova a eseguire nuovamente la richiesta.

Gli snapshot possono non andare a buon fine se gli operatori dell'applicazione non sono allocati correttamente. Per informazioni sull'ottimizzazione delle prestazioni degli operatori, consulta [Dimensionamento degli operatori](#).

Dopo che l'applicazione è tornata a uno stato integro, si consiglia di impostare la proprietà `SnapshotsEnabled` dell'applicazione su `true`.

Impossibile accedere alle risorse in un VPC

Se l'applicazione utilizza un VPC in esecuzione su Amazon VPC, è possibile verificare che l'applicazione riesca ad accedere alle proprie risorse eseguendo i passaggi seguenti:

- Controlla i CloudWatch log per verificare la presenza del seguente errore. Questo errore indica che l'applicazione non riesce ad accedere alle risorse nel VPC:

```
org.apache.kafka.common.errors.TimeoutException: Failed to update metadata after 60000 ms.
```

Se visualizzi questo errore, verifica che le tabelle di routing siano configurate correttamente e le impostazioni di connessione dei connettori siano corrette.

Per informazioni sulla configurazione e l'analisi dei CloudWatch log, consulta. [Registrazione e monitoraggio in Amazon Managed Service per Apache Flink](#)

I dati vengono persi durante la scrittura su un bucket Amazon S3

Potrebbe verificarsi una perdita di dati nel corso della scrittura dell'output su un bucket Amazon S3 utilizzando la versione 1.6.2 di Apache Flink. Quando si usa Amazon S3 per l'output diretto è consigliabile utilizzare l'ultima versione supportata di Apache Flink. Per scrivere su un bucket Amazon S3 utilizzando Apache Flink 1.6.2, consigliamo di utilizzare Firehose. Per ulteriori informazioni sull'utilizzo di Firehose with Managed Service for Apache Flink, vedere. [Firehose Sink](#)

L'applicazione è nello stato RUNNING ma non sta elaborando dati

È possibile controllare lo stato dell'applicazione utilizzando le azioni [ListApplications](#) o [DescribeApplication](#). Se la tua applicazione entra nello RUNNING stato ma non sta scrivendo dati sul tuo sink, puoi risolvere il problema aggiungendo un flusso di CloudWatch log Amazon alla tua applicazione. Per ulteriori informazioni, consulta [Utilizzate le opzioni di CloudWatch registrazione delle applicazioni](#). Il flusso di log contiene messaggi che consentono di risolvere eventuali problemi relativi alle applicazioni.

Istantanea, aggiornamento dell'applicazione o errore di interruzione dell'applicazione: InvalidApplicationConfigurationException

Un errore di questo tipo potrebbe verificarsi nel corso di uno snapshot o un'operazione che ne crea uno, come l'aggiornamento o l'arresto di un'applicazione:

```
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
```

tale errore ha luogo, quindi, quando l'applicazione non è in grado di creare uno snapshot.

Ecco come procedere se l'errore si verifica mentre scatti uno snapshot o svolgi un'operazione che ne crea uno:

- disattiva gli snapshot per l'applicazione; È possibile eseguire questa operazione nella console Managed Service for Apache Flink o utilizzando il `SnapshotsEnabledUpdate` parametro dell'azione. [UpdateApplication](#)
- prova a comprendere perché è impossibile creare gli snapshot; Per ulteriori informazioni, consulta [L'applicazione è bloccata in uno stato transitorio](#).
- Quando l'applicazione torna a funzionare sarà possibile riattivare gli snapshot.

`java.nio.file. NoSuchFileException:/usr/local/openjdk-8/lib/security/cacerts`

La posizione del truststore SSL è stata aggiornata in una precedente implementazione. Per il parametro `ssl.truststore.location` sono invece da prediligere i seguenti parametri:

```
/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
```

L'applicazione si sta riavviando

Se l'applicazione non è integra, il processo Apache Flink non riesce e si riavvia continuamente. Questa sezione descrive i sintomi e le procedure di risoluzione dei problemi relativi a questa condizione.

Caratteristiche

Questa condizione può avere i seguenti sintomi:

- La metrica `FullRestarts` non è zero. Questa metrica rappresenta il numero di volte in cui il lavoro dell'applicazione è stato riavviato dall'avvio dell'applicazione.
- La metrica `Downtime` non è zero. Questa metrica rappresenta il numero di millisecondi in cui l'applicazione si trova nello stato `FAILING` o `RESTARTING`.
- Il log dell'applicazione contiene le modifiche allo stato `RESTARTING` o `FAILED`. È possibile interrogare il registro dell'applicazione per queste modifiche di stato utilizzando la seguente query CloudWatch Logs Insights: [Analizza gli errori: errori relativi alle attività dell'applicazione](#)

Cause e soluzioni

Le seguenti condizioni possono causare l'instabilità e il riavvio ripetuto dell'applicazione:

- L'operatore genera un'eccezione: se un'eccezione in un operatore dell'applicazione non viene gestita, l'applicazione fallisce (interpretando che l'errore non può essere gestito dall'operatore). L'applicazione si riavvia dal checkpoint più recente per mantenere la semantica di elaborazione singola. Di conseguenza, Downtime non è zero durante questi periodi di riavvio. Per evitare che ciò accada, si consiglia di gestire eventuali eccezioni riutilizzabili nel codice dell'applicazione.

È possibile esaminare le cause di questa condizione interrogando i log dell'applicazione per verificare eventuali modifiche allo stato dell'applicazione da RUNNING a FAILED. Per ulteriori informazioni, consulta [the section called “Analizza gli errori: errori relativi alle attività dell'applicazione”](#).

- Il provisioning dei flussi di dati Kinesis non è corretto: se una fonte o un sink per l'applicazione è un flusso di dati Kinesis, controlla le [metriche relative](#) allo stream per verificare la presenza di errori. `ReadProvisionedThroughputExceeded` `WriteProvisionedThroughputExceeded`

Se visualizzi questi errori, puoi aumentare il throughput disponibile per il flusso Kinesis aumentando il numero di partizioni del flusso. Per ulteriori informazioni, consulta [Come posso modificare il numero di partizioni aperte nei flussi di dati Kinesis?](#)

- Altre origini o sink non vengono forniti correttamente o non sono disponibili: verifica che l'applicazione stia fornendo correttamente origini e sink. Verifica che tutte le fonti o i sink utilizzati nell'applicazione (come altri AWS servizi o fonti o destinazioni esterne) siano ben forniti, che non siano soggetti a limitazioni di lettura o scrittura o che non siano periodicamente non disponibili.

Se riscontri problemi relativi al throughput con i servizi dipendenti, aumenta le risorse disponibili per tali servizi o investiga la causa di eventuali errori o indisponibilità.

- Gli operatori non vengono forniti correttamente: se il carico di lavoro sui thread di uno degli operatori dell'applicazione non è distribuito correttamente, l'operatore può sovraccaricarsi e l'applicazione può arrestarsi in modo anomalo. Per informazioni sulla regolazione del parallelismo degli operatori, consulta [Gestire correttamente il dimensionamento degli operatori](#).
- L'applicazione fallisce con `DaemonException`: questo errore viene visualizzato nel registro dell'applicazione se si utilizza una versione di Apache Flink precedente alla 1.11. Potrebbe essere necessario eseguire l'aggiornamento a una versione successiva di Apache Flink in modo da utilizzare una versione KPL 0.14 o successiva.

- L'applicazione fallisce con `TimeoutException`, `FlinkException`, o `RemoteTransportException`: Questi errori possono comparire nel registro dell'applicazione se i task manager si bloccano. Se l'applicazione è sovraccarica, i task manager possono subire un carico eccessivo sulla CPU o sulle risorse di memoria, con conseguenti errori.

Questi errori possono essere simili ai seguenti:

- `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`

Per risolvere questa condizione, verifica quanto segue:

- Controlla le tue CloudWatch metriche per rilevare picchi insoliti nell'utilizzo della CPU o della memoria.
- Controlla se l'applicazione presenta problemi di throughput. Per ulteriori informazioni, consulta [Risolvi i problemi relativi alle prestazioni](#).
- Esamina il log dell'applicazione per individuare eventuali eccezioni non gestite che il codice della tua applicazione sta generando.
- L'applicazione fallisce con l'errore `JaxbAnnotationModule Not Found`: questo errore si verifica se l'applicazione utilizza Apache Beam, ma non ha le dipendenze o le versioni di dipendenza corrette. Le applicazioni del servizio gestito Apache Flink che utilizzano Apache Beam devono utilizzare le seguenti versioni delle dipendenze:

```
<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>
```

Se non fornisci la versione corretta di `jackson-module-jaxb-annotations` come dipendenza esplicita, l'applicazione la carica dalle dipendenze dell'ambiente e, poiché le versioni non corrispondono, l'applicazione si blocca in fase di runtime.

Per maggiori informazioni sull'utilizzo di Apache Beam con il servizio gestito per Apache Flink, consulta [Usa CloudFormation](#).

- L'applicazione fallisce con java.io. IOException: numero insufficiente di buffer di rete

Ciò accade quando un'applicazione non dispone di memoria sufficiente per i buffer di rete. I buffer di rete facilitano la comunicazione tra le sottoattività. Vengono utilizzati per archiviare i record prima della trasmissione su una rete e per archiviare i dati in entrata prima di suddividerli in record e consegnarli a sottoattività. Il numero di buffer di rete richiesti varia direttamente in base al parallelismo e alla complessità del grafico di processo. Esistono diversi approcci per mitigare questo problema:

- È possibile configurare un `parallelismPerKpu` più basso in modo da aumentare le memoria allocata per sottoattività e buffer di rete. Tieni presente che una riduzione `parallelismPerKpu` aumenterà la KPU e quindi i costi. Per evitare ciò, puoi mantenere la stessa quantità di KPU riducendo il parallelismo per uno stesso fattore.
- È possibile semplificare il grafico di processo riducendo il numero di operatori o concatenandoli in modo da ridurre il numero di buffer necessari.
- Altrimenti, puoi contattare <https://aws.amazon.com/premiumsupport/> per una configurazione personalizzata del buffer di rete.

La velocità effettiva è troppo lenta

Se l'applicazione non elabora i dati di streaming in entrata abbastanza velocemente, le prestazioni non saranno ottimali e l'applicazione poco stabile. Questa sezione descrive le caratteristiche e le procedure di risoluzione dei problemi relativi a questa problematica.

Caratteristiche

Questa situazione può presentare i seguenti sintomi:

- Se l'origine dati per l'applicazione è un flusso Kinesis, la metrica `millisbehindLatest` del flusso aumenta in maniera continua.
- Se l'origine dati per l'applicazione è un cluster Amazon MSK, le metriche del cluster relative al ritardo per l'utente aumentano in modo continuo. Per ulteriori informazioni, è possibile consultare [Monitoraggio del ritardo per l'utente](#) nella [Guida per gli sviluppatori Amazon MSK](#).
- Se l'origine dati per l'applicazione è un servizio o una fonte diversa, è consigliabile controllare le metriche o i dati disponibili in materia di ritardi per l'utente.

Cause e soluzioni

Le cause del rallentamento della velocità di trasmissione effettiva delle applicazioni possono essere molteplici. Se l'applicazione non riesce ad adeguarsi agli input, verifica gli aspetti seguenti:

- Se il ritardo nella velocità di trasmissione effettiva aumenta e poi si riduce, controlla se l'applicazione è in fase di riavvio. L'applicazione interromperà l'elaborazione dell'input durante il riavvio, causando un aumento dei ritardi. Per ulteriori informazioni sugli errori all'interno dell'applicazione, consulta [L'applicazione si sta riavviando](#).
- Se il ritardo nella velocità di trasmissione effettiva è costante, verifica che le prestazioni dell'applicazione siano ottimizzate. Per informazioni sull'ottimizzazione delle prestazioni dell'applicazione, consulta [Risolvi i problemi relativi alle prestazioni](#).
- Se il ritardo nella velocità di trasmissione effettiva aumenta in maniera costante ma non improvvisa e le prestazioni dell'applicazione sono ottimizzate, è necessario potenziare le risorse dell'applicazione. Per informazioni sull'aumento delle risorse dell'applicazione, consulta [Implementa la scalabilità delle applicazioni](#).
- Se l'applicazione legge da un cluster Kafka in un'altra regione e `FlinkKafkaConsumer` e `KafkaSource` sono per lo più inattivi (livelli alti di `idleTimeMsPerSecond` o bassi di `CPUUtilization`) nonostante il forte ritardo per l'utente, è possibile aumentare il valore di `receive.buffer.byte`, per esempio 2097152. Per ulteriori informazioni consigliamo di consultare la sezione Ambiente ad alta latenza nelle [configurazioni MSK personalizzate](#).

Per le procedure di risoluzione dei problemi relativi a rallentamenti nella velocità di trasmissione effettiva o all'aumento del ritardo per l'utente nell'origine dell'applicazione, consulta [Risolvi i problemi relativi alle prestazioni](#).

Crescita statale illimitata

Se l'applicazione non elimina correttamente le informazioni obsolete sullo stato, tali informazioni accumuleranno continuamente e causeranno problemi di prestazioni o stabilità delle applicazioni. Questa sezione descrive i sintomi e le procedure di risoluzione dei problemi relativi a questa condizione.

Caratteristiche

Questa condizione può avere i seguenti sintomi:

- La metrica `lastCheckpointDuration` sta aumentando gradualmente o mostra un picco.

- La metrica `lastCheckpointSize` sta aumentando gradualmente o mostra un picco.

Cause e soluzioni

Le seguenti condizioni possono causare l'accumulo di dati sullo stato dell'applicazione:

- L'applicazione mantiene i dati sullo stato più a lungo del necessario.
- L'applicazione utilizza query a finestra con una durata troppo lunga.
- Non hai impostato il TTL per i tuoi dati di stato. Per ulteriori informazioni, consulta [State Time-To-Live \(TTL\) nella documentazione](#) di Apache Flink.
- Stai eseguendo un'applicazione che dipende dalla versione 2.25.0 o successiva di Apache Beam. Puoi disattivare la nuova versione della trasformazione di lettura [estendendo i tuoi](#) esperimenti e il `BeamApplicationProperties` valore chiave. `use_deprecated_read` Per ulteriori informazioni, consulta la [documentazione di Apache Beam](#).

A volte le applicazioni devono far fronte a una crescita continua delle dimensioni degli stati, il che non è sostenibile a lungo termine (dopotutto un'applicazione Flink viene eseguita indefinitamente). A volte, ciò può essere ricondotto al fatto che le applicazioni archiviano i dati in uno stato indeterminato e non eseguono correttamente il processo di invecchiamento delle vecchie informazioni. Ma altre volte ci sono solo aspettative irragionevoli su ciò che Flink può offrire. Le applicazioni possono utilizzare aggregazioni su ampie finestre temporali, che coprono giorni o addirittura settimane. A meno che non [AggregateFunctions](#) vengano utilizzati, che consentono aggregazioni incrementali, Flink deve mantenere invariati gli eventi dell'intera finestra.

Inoltre, quando si utilizzano funzioni di processo per implementare operatori personalizzati, l'applicazione deve rimuovere dallo stato i dati che non sono più necessari per la logica di business. In tal caso, [lo stato time-to-live](#) può essere utilizzato per invecchiare automaticamente i dati in base al tempo di elaborazione. Il servizio gestito per Apache Flink utilizza checkpoint incrementali e quindi il TTL dello stato si basa sulla [compattazione RocksDB](#). È possibile osservare una riduzione effettiva delle dimensioni dello stato (indicata dalla dimensione del checkpoint) solo in seguito a un'operazione di compattazione. In particolare per i checkpoint di dimensioni inferiori a 200 MB, è improbabile che si verifichi una riduzione delle dimensioni dei checkpoint a causa della scadenza dello stato. Tuttavia, i savepoint si basano su una copia pulita dello stato che non contiene dati obsoleti, quindi è possibile attivare uno snapshot nel servizio gestito per Apache Flink per forzare la rimozione dello stato obsoleto.

A scopo di debug, può essere utile disabilitare i checkpoint incrementali per verificare più rapidamente che la dimensione del checkpoint diminuisca o si stabilizzi (ed evitare l'effetto della compattazione in RocksBS). Tuttavia, per farlo occorre inoltrare un ticket al team di assistenza.

Operatori vincolati all'I/O

È meglio evitare dipendenze da sistemi esterni sul percorso dei dati. Spesso è molto più efficace mantenere invariato un set di dati di riferimento piuttosto che effettuare query verso un sistema esterno per arricchire singoli eventi. Tuttavia, a volte ci sono dipendenze che non possono essere facilmente trasferite allo stato, ad esempio se si desidera arricchire gli eventi con un modello di machine learning hostato su Amazon Sagemaker.

Gli operatori che si interfacciano con sistemi esterni tramite la rete possono diventare un collo di bottiglia e generare una congestione. Si consiglia vivamente di utilizzare [AsyncIO](#) per implementare la funzionalità, ridurre i tempi di attesa delle singole chiamate ed evitare il rallentamento dell'intera applicazione.

Inoltre, per le applicazioni con operatori legati all'I/O può essere utile aumentare l'impostazione [ParallelismPerKPU](#) dell'applicazione Managed Service for Apache Flink. Questa configurazione descrive il numero di sottoattività parallele che un'applicazione è in grado di eseguire per ogni KPU (Kinesis Processing Unit). Aumentando il valore dal valore predefinito di 1 a, ad esempio, 4, l'applicazione sfrutta le stesse risorse (e ha lo stesso costo) ma può dimensionare a fino a 4 volte il parallelismo. Funziona bene per le applicazioni vincolate all'I/O, ma causa un sovraccarico aggiuntivo per le applicazioni non vincolate.

Limitazione della larghezza di banda della rete in upstream o all'origine da un flusso di dati Kinesis

Sintomo: l'applicazione riscontra `LimitExceededExceptions` dal suo flusso di dati Kinesis di origine in upstream.

Causa potenziale: l'impostazione predefinita per il connettore Kinesis della libreria di Apache Flink per la lettura dall'origine del flusso di dati Kinesis è configurata su un'impostazione predefinita molto aggressiva per il numero massimo di record recuperati per chiamata `GetRecords`. Apache Flink è configurato per impostazione predefinita per recuperare 10.000 record per `GetRecords` chiamata (questa chiamata viene effettuata di default ogni 200 ms), sebbene il limite per shard sia di soli 1.000 record.

Questo comportamento predefinito può portare a una limitazione della larghezza di banda della rete quando si tenta di utilizzare dati dal flusso di dati Kinesis, il che influisce sulle prestazioni e sulla stabilità dell'applicazione.

Puoi confermarlo controllando la CloudWatch `ReadProvisionedThroughputExceeded` metrica e visualizzando i periodi prolungati o sostenuti in cui questa metrica è maggiore di zero.

Puoi anche visualizzarlo nei CloudWatch log della tua applicazione Amazon Managed Service for Apache Flink osservando gli errori continui. `LimitExceededException`

Risoluzione: puoi fare una delle due cose per risolvere questo scenario:

- Ridurre il limite predefinito per il numero di record recuperati per chiamata `GetRecords`
- Abilita Adaptive Reads nella tua applicazione Amazon Managed Service for Apache Flink. Per ulteriori informazioni sulla funzionalità Adaptive Reads, consultare [SHARD_USE_ADAPTIVE_READS](#)

Checkpoint

I checkpoint rappresentano il meccanismo di Flink per garantire che lo stato di un'applicazione sia in grado di tollerare eventuali errori. Questo meccanismo consente a Flink di ripristinare lo stato degli operatori in caso di errori del processo, nonché di fornire all'applicazione le semantiche di un'esecuzione senza errori. Con il servizio gestito per Apache Flink, lo stato di un'applicazione viene archiviato in RocksDB, un archivio chiave-valore integrato il cui stato operativo opera su disco. Quando viene eseguito un checkpoint, lo stato viene caricato anche su Amazon S3: anche se il disco viene perso, pertanto, il checkpoint può essere utilizzato per ripristinare lo stato delle applicazioni.

Per ulteriori informazioni consulta la sezione [Come funzionano gli snapshot dello stato?](#).

Fasi di checkpoint

Le fasi principali di un'attività secondaria per effettuare il checkpoint di un operatore in Flink sono 5:

- In attesa [Ritardo di avvio]: Flink utilizza le barriere di checkpoint inserite nel flusso: in questa fase, pertanto, il tempo è il periodo di attesa dell'operatore prima che la barriera del checkpoint lo raggiunga.
- Allineamento [Durata dell'allineamento]: in questa fase l'attività secondaria ha raggiunto una barriera ma è in attesa delle barriere provenienti da altri flussi di input.

- Sincronizzazione del checkpoint [Durata della sincronizzazione]: in questa fase l'attività secondaria effettua uno snapshot dello stato dell'operatore e blocca tutte le altre attività presenti sulla attività secondaria.
- Checkpoint asincrono [Durata asincrona]: questa fase consiste in gran parte nel caricamento dello stato su Amazon S3 da parte dell'attività secondaria. Durante questa fase, l'attività secondaria non è più bloccata e può elaborare i record.
- Riconoscimento: di solito è una fase breve ed è semplicemente la sottoattività che invia una conferma a JobManager e esegue anche eventuali messaggi di commit (ad esempio con i sinks di Kafka).

Ciascuna di queste fasi (esclusa Riconoscimento) corrisponde a un parametro di durata per i checkpoint disponibile nell'interfaccia utente Web di Flink, che può aiutare a individuare la causa del prolungamento delle tempistiche del checkpoint.

Per vedere la definizione esatta di ciascuna delle metriche disponibili sui checkpoint, visita la scheda [Cronologia](#).

Analisi

Quando si esamina la durata prolungata di un checkpoint, l'elemento più importante da determinare è il collo di bottiglia del checkpoint, vale a dire quale operatore e attività secondaria impiega la quantità maggiore di tempo ad arrivare al checkpoint e quale fase di tale attività secondaria richiede un periodo di tempo prolungato. Può essere determinato utilizzando l'interfaccia utente Web di Flink nell'attività di checkpoint dei processi. L'interfaccia web di Flink fornisce dati e informazioni che aiutano a risolvere le problematiche relative ai checkpoint. Per un'analisi completa, consulta [Monitoraggio dei checkpoint](#).

Per determinare quale operatore richiede una quantità di tempo prolungata per arrivare al checkpoint e merita approfondimenti, il primo elemento da esaminare è la durata end-to-end di ogni operatore nel grafico del processo. Conformemente alla documentazione di Flink, la definizione della durata è:

La durata a partire dal timestamp del trigger fino all'ultimo riconoscimento (o n/a se non è stato ancora ricevuto nessun riconoscimento). La durata end-to-end di un checkpoint completo è determinata dall'ultima attività secondaria che riconosce il checkpoint. Tale periodo è generalmente superiore a quello di cui le attività secondarie necessitano per effettuare la verifica dello stato.

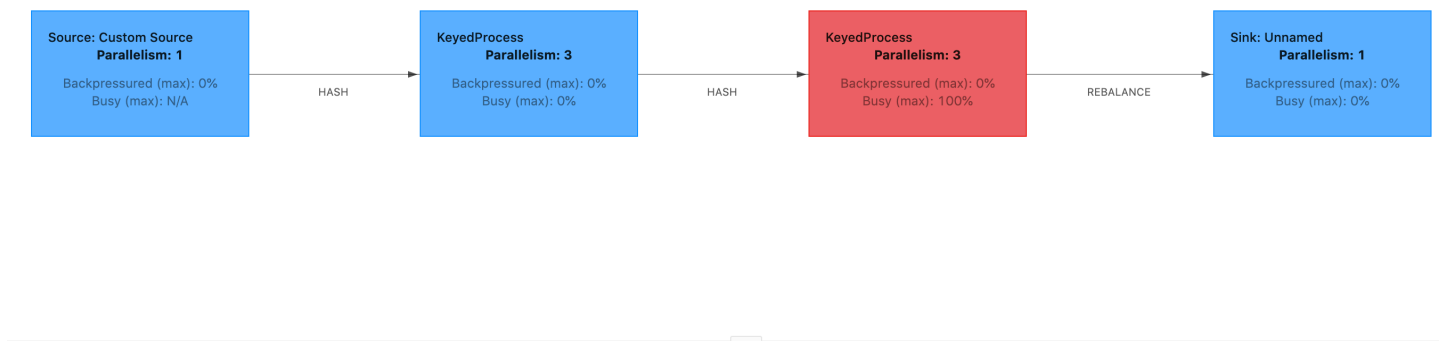
Le diverse durate della verifica forniscono inoltre informazioni più dettagliate relativamente agli aspetti che richiedono più tempo.

Una durata della sincronizzazione elevata indica problematiche nel corso dello scatto dello snapshot. In questa fase `snapshotState()` è attivato per le classi che implementano l'interfaccia `snapshotState`; può trattarsi di codice utente, quindi i thread dump possono essere utili per indagare su questo aspetto.

Una durata asincrona prolungata suggerirebbe che il caricamento dello stato su Amazon S3 richiede una quantità di tempo prolungata. Ciò può verificarsi se le dimensioni dello stato sono importanti o se vengono caricati diversi file di stato. In tal caso, è consigliabile analizzare l'utilizzo dello stato da parte dall'applicazione, nonché assicurarsi che, dove possibile, vengano utilizzate le strutture di dati native di Flink ([Utilizzare Keyed State](#)). Il servizio gestito per Apache Flink configura Flink in modo tale da ridurre al minimo il numero di chiamate Amazon S3, garantendone l'ottimizzazione. Di seguito è riportato un esempio di statistiche sulle verifiche di un operatore, che rivela quanto la durata asincrona sia relativamente lunga rispetto alle precedenti statistiche di verifica dell'operatore.

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	
I D	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:														

Un ritardo di avvio elevato indicherebbe che la maggior parte del tempo viene impiegata ad aspettare che la barriera del checkpoint raggiunga l'operatore. Ciò indica che l'applicazione impiega un periodo di tempo superiore per elaborare i record, il che significa che la barriera scorre lentamente attraverso il grafico del processo. Tale situazione si verifica se il processo è soggetto a contropressione o se uno o più operatori sono costantemente occupati. Di seguito è riportato un esempio di un caso in cui il secondo operatore è occupato `JobGraph`. `KeyedProcess`



Puoi indagare su cosa sta impiegando così tanto tempo utilizzando Flink Flame Graphs o TaskManager i thread dump. Una volta identificata la problematica, è possibile analizzarla ulteriormente utilizzando i grafici a fiamma o i thread dump.

Thread dump

I thread dump sono uno strumento di debug che si trova a un livello leggermente inferiore rispetto ai grafici a fiamma. Un thread dump restituisce lo stato di esecuzione di tutti i thread in un determinato momento. Flink esegue un thread dump JVM, ossia uno stato di esecuzione di tutti i thread all'interno del processo Flink. Lo stato di un thread è rappresentato da una traccia dello stack del thread, insieme ad alcune informazioni aggiuntive. I grafici a fiamma vengono creati utilizzando diverse stack trace in rapida successione. Il grafico è una visualizzazione composta dalle stack trace, che semplifica l'identificazione dei percorsi di codice comuni.

```

"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(KeyedProcessOperator
  at app//org.apache.flink.streaming.runtime.tasks.OneInputStreamTask
$StreamTaskNetworkOutput.emitRecord(OneInputStreamTask.java:205)
  at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.processElement(AbstractStream
  at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.emitNext(AbstractStreamTask
  
```

```
at app//  
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.processInput(StreamOneInputProces  
...
```

Qui sopra è riportato un frammento di un thread dump tratto dall'interfaccia utente di Flink per un singolo thread. La prima riga contiene alcune informazioni generali su questo thread, tra cui:

- Il nome KeyedProcess del thread (1/3) #0
- Priorità del thread prio=5
- Un thread unico Id = 1423
- Stato del thread ESEGUIBILE

Il nome di un thread solitamente fornisce informazioni sul suo scopo generale. I thread degli operatori possono essere identificati dal loro nome poiché i thread dell'operatore hanno lo stesso nome dell'operatore, oltre all'indicazione della sottoattività a cui sono correlati, ad esempio, il thread KeyedProcess (1/3) #0 proviene dall'KeyedProcessoperatore e proviene dalla prima sottoattività (su 3).

I thread possono trovarsi in uno dei seguenti stati:

- NUOVO: il thread è stato creato ma non è ancora stato elaborato
- ESEGUIBILE: il thread è in esecuzione sulla CPU
- BLOCCATO: il thread è in attesa che un altro thread rilasci il blocco
- IN ATTESA: il thread è in attesa e utilizza un metodo `wait()`, `join()` o `park()`
- ATTESA_A TEMPO: il thread è in attesa con un metodo di sospensione, attesa, collegamento o prenotazione, ma con un tempo di attesa massimo.

Note

La profondità massima di un singolo stacktrace nel thread dump su Flink 1.13 è limitata a 8.

Note

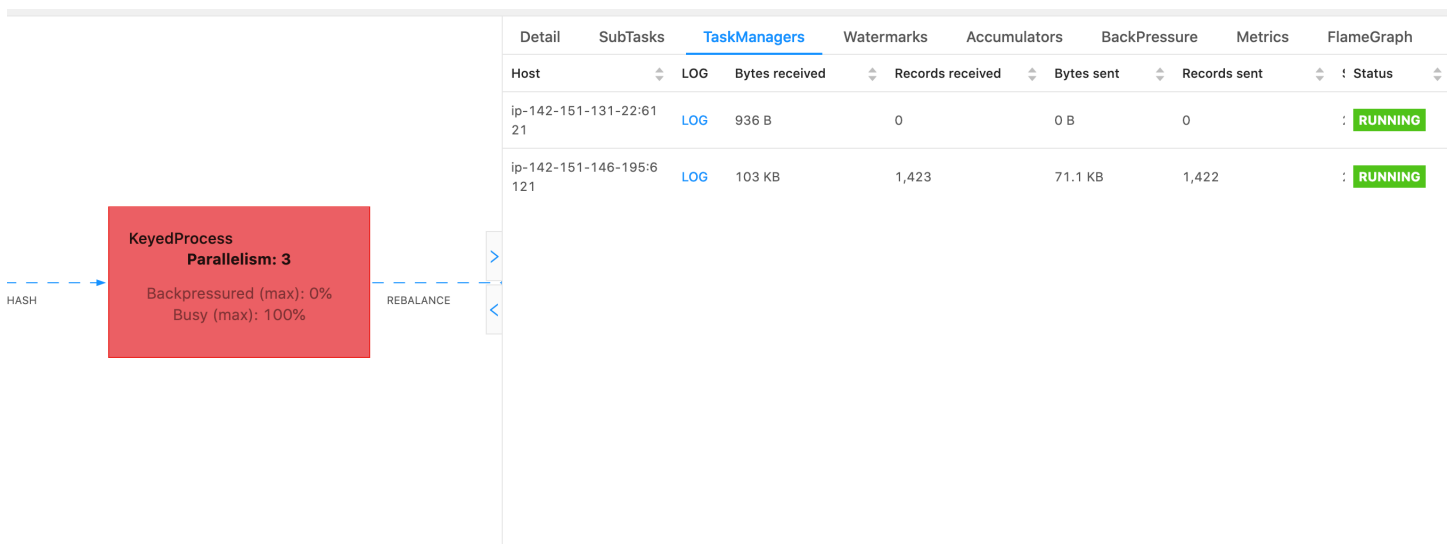
I thread dump dovrebbero rappresentare l'ultima soluzione per il debug dei problemi relativi alle prestazioni in un'applicazione Flink, poiché possono essere complessi da leggere,

richiedono l'acquisizione di più campioni e l'analisi manuale. Ove possibile, è preferibile utilizzare i grafici a fiamma.

Thread dump su Flink

Su Flink è possibile eseguire un thread dump scegliendo l'opzione Task Manager nella barra di navigazione a sinistra dell'interfaccia utente di Flink, selezionando un determinato task manager e accedendo alla scheda Thread dump. Il thread dump può essere scaricato, copiato in un editor di testo (o in un analizzatore di thread dump) o analizzato direttamente all'interno della visualizzazione di testo nell'interfaccia utente Web di Flink; quest'ultima opzione, però, potrebbe risultare macchinosa.

Per determinare quale Task Manager utilizzare, è possibile utilizzare un thread dump della TaskManagersscheda quando si sceglie un particolare operatore. Ciò dimostra che l'operatore è in esecuzione su diverse attività secondarie di un operatore e può essere eseguito su diversi Task Manager.



Detail	SubTasks	TaskManagers	Watermarks	Accumulators	BackPressure	Metrics	FlameGraph
Host	LOG	Bytes received	Records received	Bytes sent	Records sent	Status	
ip-142-151-131-22:61	LOG	936 B	0	0 B	0	RUNNING	
ip-142-151-146-195:6121	LOG	103 KB	1,423	71.1 KB	1,422	RUNNING	

Il dump sarà composto da diverse stack trace. Quando si esamina il dump, tuttavia, i più importanti sono quelli relativi a un operatore. Possono essere individuati facilmente, poiché i thread degli operatori hanno lo stesso nome dell'operatore, oltre a un'indicazione dell'attività secondaria a cui sono legati. Ad esempio, la seguente traccia dello stack proviene dall'KeyedProcessoperatore ed è la prima sottoattività.

```
"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(KeyedProcessOperator
    at app//org.apache.flink.streaming.runtime.tasks.OneInputStreamTask
$StreamTaskNetworkOutput.emitRecord(OneInputStreamTask.java:205)
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.processElement(AbstractStreamTask
    at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.emitNext(AbstractStreamTask
    at app//
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.processInput(StreamOneInputProcessor
    ...

```

Questo può creare confusione se ci sono più operatori con lo stesso nome, tuttavia per aggirare il problema è possibile rinominare gli operatori. Per esempio:

```

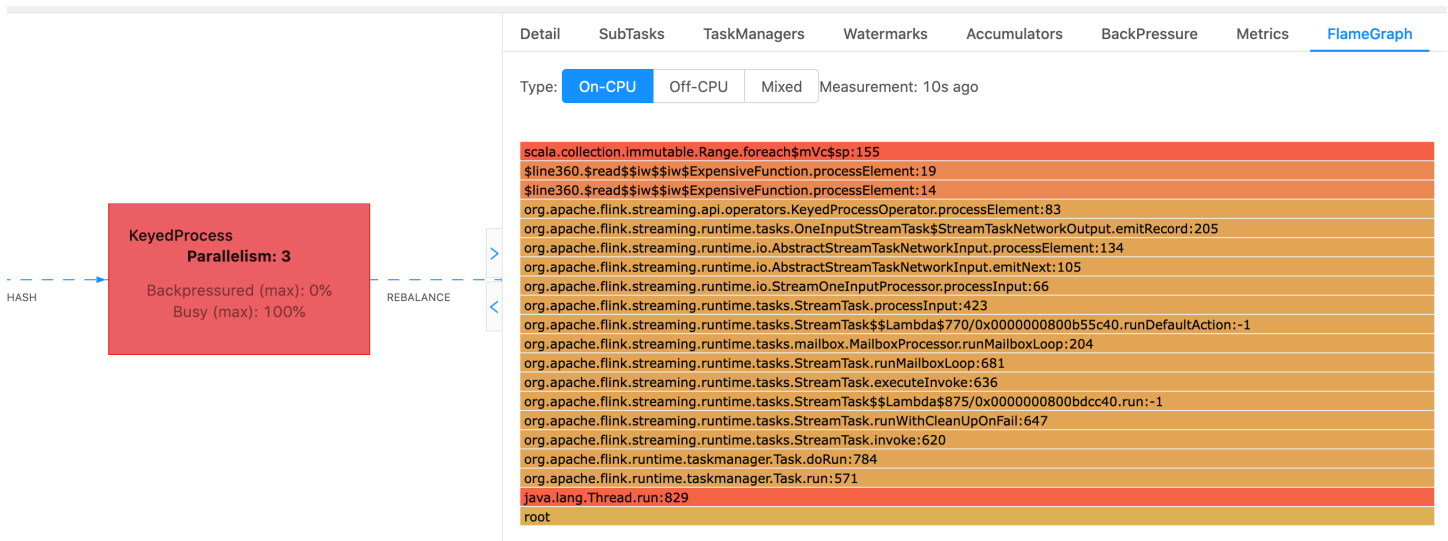
....
.process(new ExpensiveFunction).name("Expensive function")

```

Grafici a fiamma

I grafici a fiamma sono uno strumento per il debug che consente di visualizzare le stack trace del codice di destinazione, il che consente di identificare i percorsi di codice più frequenti. Vengono creati campionando più volte le stack trace. L'asse x di un grafico a fiamma mostra i diversi profili dello stack, mentre l'asse y mostra la profondità dello stack e ne richiama la stack trace. Un rettangolo in un grafico a fiamma rappresenta uno stack frame, mentre la larghezza di un riquadro mostra la frequenza con cui compare negli stack. Per ulteriori dettagli sui grafici a fiamma e su come usarli, consulta [Grafici a fiamma](#).

In Flink, è possibile accedere al grafico a fiamma di un operatore tramite l'interfaccia utente Web selezionando un operatore e quindi scegliendo la scheda. FlameGraph Una volta raccolto un numero sufficiente di campioni, verrà visualizzato il grafico a fiamma. Di seguito è riportato il messaggio ProcessFunction che ha impiegato molto tempo FlameGraph per arrivare al checkpoint.



Questo è un grafico a fiamma molto semplice e mostra che tutto il tempo impiegato dalla CPU viene speso in un'unica occhiata all'interno `processElement` dell'operatore. `ExpensiveFunction` È inoltre possibile ottenere il numero di riga, per determinare la posizione esatta in cui avviene l'esecuzione del codice.

Il checkpoint è in fase di interruzione

Se l'applicazione non è correttamente ottimizzata o predisposta, i checkpoint possono non riuscire. Questa sezione descrive le caratteristiche e le procedure di risoluzione dei problemi relativi alla problematica precedentemente analizzata.

Caratteristiche

Se l'applicazione non riesce ad effettuare i checkpoint, `numberOfFailedCheckpoints` sarà maggiore di zero.

I checkpoint possono non riuscire a causa di errori diretti (per esempio errori delle applicazioni) o di errori temporanei (per esempio l'esaurirsi delle risorse dell'applicazione). Verifica i log e i parametri dell'applicazione, alla ricerca delle seguenti caratteristiche:

- errori nel codice;
- errori di accesso ai servizi dipendenti dell'applicazione;
- errori nella serializzazione dei dati. Se il serializzatore predefinito non è in grado di serializzare i dati dell'applicazione, l'applicazione cesserà di funzionare. Per informazioni sull'utilizzo di un serializzatore personalizzato nell'applicazione, consulta [Tipi di dati e serializzazione](#) nella documentazione di Apache Flink.

- errori relativi all'esaurimento della memoria;
- picchi o aumenti costanti nelle seguenti metriche:
 - `heapMemoryUtilization`
 - `oldGenerationGCTime`
 - `oldGenerationGCCount`
 - `lastCheckpointSize`
 - `lastCheckpointDuration`

Per ulteriori informazioni sul monitoraggio dei checkpoint, consulta [Monitoring Checkpointing](#) nella documentazione di Apache [Flink](#).

Cause e soluzioni

I messaggi di errore del log delle applicazioni mostrano la causa degli errori diretti. Gli errori temporanei possono essere provocati dalle seguenti cause:

- Le KPU fornite dall'applicazione non sono sufficienti. Per ulteriori informazioni su come aumentarne la quantità, consulta [Implementa la scalabilità delle applicazioni](#).
- La dimensione dello stato dell'applicazione è troppo grande. È possibile monitorare le dimensioni dello stato dell'applicazione utilizzando la metrica `lastCheckpointSize`.
- I dati sullo stato dell'applicazione sono distribuiti in modo non uniforme tra le chiavi. Se l'applicazione utilizza l'operatore `KeyBy`, assicurati che i dati in entrata siano divisi equamente tra le chiavi. Assegnare la maggior parte dei dati a un'unica chiave implica la creazione di un collo di bottiglia, che provocherà errori.
- L'applicazione subisce una contropressione della memoria o della rimozione di oggetti inutili (garbage collection). Consigliamo di monitorare `heapMemoryUtilization`, `oldGenerationGCTime` e `oldGenerationGCCount` dell'applicazione, per rilevare eventuali picchi o valori in aumento costante.

Errori del checkpoint per l'applicazione Apache Beam

Se l'applicazione Beam è configurata con l'[shutdownSourcesAfterIdleMs](#) impostazione impostata su 0ms, i checkpoint potrebbero non essere attivati perché le attività sono nello stato «FINITO». La presente sezione descrive le caratteristiche e le soluzioni relative a questa situazione.

Caratteristiche

Vai ai CloudWatch log dell'applicazione Managed Service for Apache Flink e controlla se il seguente messaggio di registro è stato registrato. Il seguente messaggio di log indica che il checkpoint non è andato a buon fine, poiché alcune attività sono state completate.

```
{
  "locationInformation":
    "org.apache.flink.runtime.checkpoint.CheckpointCoordinator.onTriggerFailure(CheckpointCoordinator",
    "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"
}
```

Ciò può accadere anche nella dashboard di Flink, dove alcune attività sono passate allo stato "FINITO" e il checkpoint non è più possibile.

Detail	SubTasks	TaskManagers	Watermarks	Accumulators	BackPressure	Metrics	FlameGraph			
ID	Bytes Received	Records Received	Bytes Sent	Records Sent	Attempt	Host	Start Time	Duration	Status	More
0	0 B	0	0 B	0	1	sea3-ws-aggr-r3-pc-1	2022-06-06 11:16:03	13m 57s	RUNNING	...
1	0 B	0	0 B	0	1	sea3-ws-aggr-r3-pc-1	2022-06-06 11:16:03	3s	FINISHED	...
2	0 B	0	0 B	0	1	sea3-ws-aggr-r3-pc-1	2022-06-06 11:16:03	3s	FINISHED	...
3	0 B	0	0 B	0	1	sea3-ws-aggr-r3-pc-1	2022-06-06 11:16:03	3s	FINISHED	...
4	0 B	0	0 B	0	1	sea3-ws-aggr-r3-pc-1	2022-06-06 11:16:03	3s	FINISHED	...

Causa

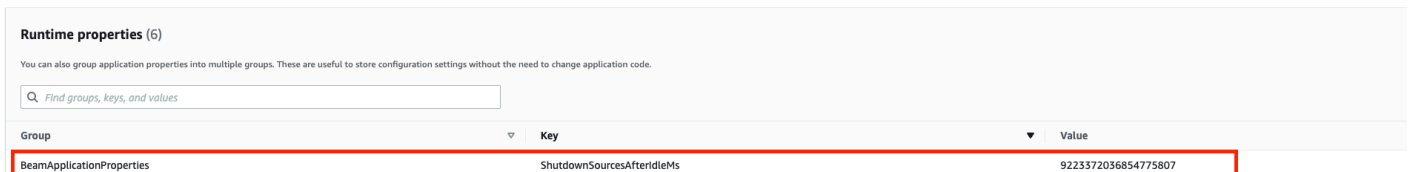
`shutdownSourcesAfterIdleMs` è una variabile di configurazione di Beam che spegne le sorgenti che sono rimaste inattive per il tempo configurato di millisecondi. Il checkpoint non è più possibile quando una fonte viene chiusa. La conseguenza potrebbe essere un [checkpoint non riuscito](#).

Uno dei motivi per cui le attività passano allo stato «FINITO» è quando `shutdownSourcesAfterIdleMs` è impostato su 0 ms, il che significa che le attività inattive verranno chiuse immediatamente.

Soluzione

Per evitare che le attività entrino immediatamente nello stato «FINITO», imposta su `Long.MAX_VALUE`. `shutdownSourcesAfter IdleMs` Esistono due maniere per assicurarsene:

- Opzione 1: se la configurazione di beam è impostata nella pagina di configurazione dell'applicazione Managed Service for Apache Flink, puoi aggiungere una nuova coppia di valori chiave per impostare Ms come segue: `shutdpwnSourcesAfteridle`



Group	Key	Value
BeamApplicationProperties	ShutdownSourcesAfterIdleMs	9223372036854775807

- Opzione 2: se la configurazione di beam è impostata nel file JAR, puoi impostare come segue: `shutdownSourcesAfter IdleMs`

```
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
```

Contropressione

Flink utilizza la contropressione per adattare la velocità di elaborazione dei singoli operatori.

L'operatore può avere difficoltà a continuare a elaborare il volume di messaggi che riceve per diverse ragioni. L'operazione potrebbe richiedere più risorse della CPU di quante ne abbia a disposizione l'operatore. Quest'ultimo può attendere il completamento delle operazioni di I/O. Se un operatore non è in grado di elaborare rapidamente gli eventi, crea una contropressione negli operatori a monte che alimentano l'operatore lento. Ciò causa un rallentamento degli operatori a monte, il che può propagare ulteriormente la contropressione alla fonte e far sì che la sorgente si adatti alla velocità

di trasmissione effettiva complessiva dell'applicazione, rallentando anche quest'ultima. Per una descrizione più approfondita della contropressione e del suo funzionamento, consulta [Come Apache Flink™ gestisce la contropressione](#).

Individuare gli operatori più lenti all'interno di un'applicazione consente di ottenere informazioni fondamentali per comprendere la causa principale dei problemi di prestazioni dell'applicazione. Le informazioni relative alla contropressione sono [rese visibili tramite la dashboard di Flink](#). Per identificare l'operatore lento è sufficiente individuare quello con un valore di contropressione elevato più vicino a un sink (l'operatore B, nell'esempio seguente). L'operatore responsabile dei rallentamenti è uno degli operatori a valle (l'operatore C nell'esempio). A causa della contropressione l'operatore B non è in grado di inoltrare l'output a C, l'operatore lento a causa del quale non riesce a elaborare gli eventi in maniera rapida.

```
A (backpressured 93%) -> B (backpressured 85%) -> C (backpressured 11%) -> D  
(backpressured 0%)
```

Una volta identificato l'operatore lento, prova a individuare i motivi di tale fenomeno. Talvolta le ragioni potrebbero essere molteplici e l'individuazione del problema rivelarsi complessa, tanto da richiedere diversi giorni dedicati ad attività di debugging e profilazione. Di seguito sono riportati alcune delle cause più ovvie e comuni, alcune delle quali sono ulteriormente descritte di seguito:

- l'operatore esegue un I/O lento, ad esempio chiamate di rete (valuta di ricorrere all'utilizzo di `AsyncIO`);
- i dati hanno subito un disallineamento e un operatore riceve più eventi rispetto agli altri (esamina il numero di messaggi in entrata/in uscita dalle singole attività secondarie (ad esempio istanze dello stesso operatore) nella dashboard di Flink).
- Questa operazione richiede l'impiego di molte risorse; se non vi è alcun disallineamento dei dati, è consigliabile aumentare orizzontalmente le operazioni legate alla CPU o alla memoria e aumentare `ParallelismPerKPU` per le operazioni legate all'I/O
- Log dettagliato nell'operatore: è consigliabile ridurlo al minimo per le applicazioni di produzione o considerare, invece, di inviare l'output di debug a un flusso di dati.

Test della produttività con il Discarding Sink

Il [discarding sink](#) ignora tutti gli eventi che riceve mentre l'applicazione è ancora in esecuzione (un'applicazione senza sink non può essere eseguita). Si rivela estremamente utile per effettuare test e profilazioni della velocità di trasmissione effettiva, nonché per verificare il corretto dimensionamento

dell'applicazione. Si tratta inoltre di un controllo di integrità estremamente pragmatico, atto a verificare se a causare la contropressione siano i sink o l'applicazione. Limitarsi a effettuare un controllo dei parametri di contropressione, tuttavia, rappresenta spesso la soluzione più semplice e diretta.

Sostituire tutti i sink di un'applicazione con un discarding sink e creare una fonte fittizia che genera dati simili a quelli di produzione consente di misurare la massima velocità di trasmissione effettiva dell'applicazione relativamente a una determinata impostazione di parallelismo. È inoltre possibile aumentare il parallelismo per verificare che l'applicazione si dimensiona in maniera corretta e non presenti ostacoli che emergono solo a velocità di trasmissione effettiva più elevate (ad esempio, a causa del disallineamento dei dati).

Disallineamento dei dati

Un'applicazione Flink viene eseguita su un cluster in modo distribuito. Per impiegare la scalabilità orizzontale su più nodi Flink utilizza il concetto di flussi con chiave, il che significa che gli eventi di un flusso vengono partizionati in base a una chiave specifica, ad esempio l'ID del cliente, per cui Flink può elaborare diverse partizioni su nodi diversi. Molti operatori Flink vengono quindi valutati sulla base di queste partizioni, ad esempio [Keyed Windows](#), [Process Functions](#) e [Async I/O](#).

La scelta di una chiave di partizione dipende spesso dalla logica commerciale. Al contempo, molte delle best practice per [DynamoDB](#) e Spark, per esempio, possono essere applicate anche a Flink. Tra di esse:

- garantire un'elevata cardinalità delle chiavi di partizione;
- evitare il disallineamento tra le partizioni del volume degli eventi.

È possibile identificare i disallineamenti nelle partizioni confrontando i record delle attività secondarie che siano stati ricevuti/inviati (ad esempio, istanze dello stesso operatore) nella dashboard di Flink. Inoltre, il monitoraggio del servizio gestito per Apache Flink può essere configurato per mostrare le metriche relative a numRecordsIn/Out e numRecordsInPerSecond/OutPerSecond anche per quanto riguarda le attività secondarie.

Disallineamento dello stato

Per gli operatori stateful, ovvero gli operatori che conservano lo stato per la propria logica commerciale, come Windows, il disallineamento dei dati provoca sempre il disallineamento dello stato. Alcune attività secondarie ricevono una quantità più elevata di eventi rispetto ad altre a causa del disallineamento dei dati, conservando così una maggiore quantità di dati nello stato. Tuttavia,

anche per un'applicazione con partizioni bilanciate in modo uniforme, è possibile che si verifichi un disallineamento nella quantità di dati conservati nello stato. Ad esempio, per le finestre di sessione alcuni utenti e sessioni possono essere molto più lunghi di altri. Se le sessioni più lunghe fanno parte della stessa partizione, ciò può portare a uno squilibrio della dimensione dello stato gestita da diverse attività secondarie dello stesso operatore.

Il disallineamento dello stato non solo aumenta la quantità di memoria e di risorse del disco impiegate per le singole attività secondarie, ma può anche ridurre le prestazioni complessive dell'applicazione. Quando un'applicazione esegue un checkpoint o un savepoint, lo stato dell'operatore viene conservato in Amazon S3, per proteggere lo stato da errori del nodo o del cluster. Durante questo processo (specialmente con la semantica Exactly Once abilitata per impostazione predefinita su Managed Service for Apache Flink), l'elaborazione si blocca da un punto di vista esterno fino a quando non riesce a fallire perché una singola sottoattività non è in checkpoint/savepoint has completed. If there is data skew, the time to complete the operation can be bound by a single subtask that has accumulated a particularly high amount of state. In extreme cases, taking checkpoints/savepoints grado di persistere nello stato.

Analogamente al disallineamento dei dati, il disallineamento dello stato può rallentare notevolmente un'applicazione.

La dashboard di Flink può essere sfruttata per identificare il disallineamento dello stato. Individua un checkpoint o un savepoint recente e confronta la quantità di dati archiviati per le singole attività secondarie nei dettagli.

Integrazione con risorse in diverse regioni

È possibile abilitare l'utilizzo di `StreamingFileSink` per la scrittura su un bucket Amazon S3 in una regione diversa rispetto all'applicazione di servizio gestito per Apache Flink tramite un'impostazione richiesta per la replica interregionale nella configurazione di Flink. A tale scopo, invia un ticket e richiedi assistenza al [Supporto AWS Centro](#).

Cronologia dei documenti per Amazon Managed Service for Apache Flink

La tabella che segue descrive le modifiche importanti apportate alla documentazione rispetto all'ultima versione del servizio gestito per Apache Flink.

- Versione API: 23-05-2018
- Ultimo aggiornamento della documentazione: 30 agosto 2023

Modifica	Descrizione	Data
Analisi dei dati Kinesis è ora noto come servizio gestito per Apache Flink	Non sono state apportate modifiche agli endpoint del servizio, all'interfaccia a riga di comando APIs, alle politiche di accesso IAM, alle CloudWatch metriche o ai dashboard di fatturazione. AWS Le applicazioni esistenti continueranno a funzionare come prima. Per ulteriori informazioni, consulta Cos'è il servizio gestito per Apache Flink?	30 agosto 2023
Supporto per Apache Flink versione 1.15.2	Il servizio gestito per Apache Flink ora supporta le applicazioni che utilizzano Apache Flink versione 1.15.2. Crea applicazioni di analisi dei dati Kinesis utilizzando l'API delle tabelle di Apache Flink. Per ulteriori informazioni, consulta Creazione di un'applicazione.	22 novembre 2022

Modifica	Descrizione	Data
Supporto per Apache Flink versione 1.13.2	Il servizio gestito per Apache Flink ora supporta le applicazioni che utilizzano Apache Flink versione 1.13.2. Crea applicazioni di analisi dei dati Kinesis utilizzando l'API delle tabelle di Apache Flink. Per ulteriori informazioni, consulta Nozioni di base: Flink 1.13.2 .	13 ottobre 2021
Supporto per Python	Il servizio gestito per Apache Flink ora supporta le applicazioni che utilizzano Python con l'API delle tabelle di Apache Flink e SQL. Per ulteriori informazioni, consulta Usa Python .	25 marzo 2021
Supporto per Apache Flink 1.11.1	Il servizio gestito per Apache Flink ora supporta le applicazioni che utilizzano Apache Flink 1.11.1. Crea applicazioni di analisi dei dati Kinesis utilizzando l'API delle tabelle di Apache Flink. Per ulteriori informazioni, consulta Creazione di un'applicazione .	19 novembre 2020
Pannello di controllo di Apache Flink	Usa il pannello di controllo di Apache Flink per monitorare lo stato e le prestazioni delle applicazioni. Per ulteriori informazioni, consulta Usa la dashboard Apache Flink .	19 novembre 2020

Modifica	Descrizione	Data
Consumatore EFO	Crea applicazioni che utilizzano o un consumatore con fan-out avanzato (EFO) per leggere da un flusso di dati Kinesis. Per ulteriori informazioni, consulta Consumatore EFO .	6 ottobre 2020
Apache Beam	Crea applicazioni che utilizzano Apache Beam per elaborare i dati di streaming. Per ulteriori informazioni, consulta Usa CloudFormation .	15 settembre 2020
Prestazioni	Come risolvere i problemi di prestazioni delle applicazioni e come creare un'applicazione performante. Per ulteriori informazioni, consulta ??? .	21 luglio 2020
Keystore personalizzato	Come accedere a un cluster Amazon MSK che utilizza un keystore personalizzato per la crittografia in transito. Per ulteriori informazioni, consulta Truststore personalizzato .	10 giugno 2020
CloudWatch Allarmi	Consigli per la creazione di CloudWatch allarmi con Managed Service for Apache Flink. Per ulteriori informazioni, consulta ??? .	5 giugno 2020

Modifica	Descrizione	Data
Nuove metriche CloudWatch	Managed Service for Apache Flink ora invia 22 parametri ad Amazon Metrics. CloudWatch Per ulteriori informazioni, consulta ??? .	12 maggio 2020
Metriche personalizzate CloudWatch	Definisci parametri specifici dell'applicazione e inviali ad Amazon Metrics. CloudWatch Per ulteriori informazioni, consulta ??? .	12 maggio 2020
Esempio: lettura da un flusso Kinesis in un account diverso	Scopri come accedere a uno stream Kinesis in un altro AWS account nella tua applicazione Managed Service for Apache Flink. Per ulteriori informazioni, consulta Multi-account .	30 marzo 2020
Supporto per Apache Flink 1.8.2	Il servizio gestito per Apache Flink ora supporta le applicazioni che utilizzano Apache Flink 1.8.2. Usa il Streaming FileSink connettore Flink per scrivere l'output direttamente su S3. Per ulteriori informazioni, consulta Creazione di un'applicazione .	17 dicembre 2019

Modifica	Descrizione	Data
VPC del servizio gestito per Apache Flink	Configura un'applicazione del servizio gestito per Apache Flink per la connessione a un cloud privato virtuale. Per ulteriori informazioni, consulta Configura MSF per accedere alle risorse in un Amazon VPC .	25 novembre 2019
Best practice del servizio gestito per Apache Flink	Procedure consigliate per la creazione e l'amministrazione delle applicazioni del servizio gestito per Apache Flink. Per ulteriori informazioni, consulta ??? .	14 ottobre 2019
Analisi dei file di log dell'applicazione del servizio gestito per Apache Flink	Usa CloudWatch Logs Insights per monitorare l'applicazione Managed Service for Apache Flink. Per ulteriori informazioni, consulta ??? .	26 giugno 2019
Proprietà di runtime di un'applicazione del servizio gestito per Apache Flink	Lavora con le proprietà di runtime nel servizio gestito per Apache Flink. Per ulteriori informazioni, consulta Usa le proprietà di runtime .	24 giugno 2019

Modifica	Descrizione	Data
Servizio gestito di tagging per applicazioni Flink	Utilizza il tagging delle applicazioni per determinare i costi per applicazione, controllare l'accesso o per scopi definiti dall'utente. Per ulteriori informazioni, consulta Aggiungere tag al servizio gestito per le applicazioni Apache Flink .	8 maggio 2019
Servizio di registrazione gestito per le chiamate API Apache Flink con AWS CloudTrail	Il servizio gestito per Apache Flink è integrato con AWS CloudTrail, un servizio che fornisce una registrazione delle azioni intraprese da un utente, ruolo o AWS servizio in Managed Service for Apache Flink. Per ulteriori informazioni, consulta ??? .	22 marzo 2019
Creare un'applicazione (Firehose Sink)	Esercitati a creare un servizio gestito per Apache Flink con un flusso di dati Amazon Kinesis come origine e un flusso Amazon Data Firehose come sink. Per ulteriori informazioni, consulta Firehose Sink .	13 dicembre 2018
Versione pubblica	Questa è la versione iniziale della Guida per gli sviluppatori del servizio gestito per Apache Flink per applicazioni Java.	27 novembre 2018

Codice di esempio dell'API Managed Service per Apache Flink

Questo argomento contiene esempi di blocchi di richiesta per le operazioni del servizio gestito per Apache Flink.

Per utilizzare JSON come input per un'azione con AWS Command Line Interface (AWS CLI), salvate la richiesta in un file JSON. Passa quindi il nome del file nell'operazione utilizzando il parametro `--cli-input-json`.

L'esempio seguente mostra come utilizzare un file JSON con un'operazione.

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

Per ulteriori informazioni sull'utilizzo di JSON con AWS CLI, consulta [Generate CLI Skeleton e CLI Input](#) JSON Parameters nella Guida per l'utente.AWS Command Line Interface

Argomenti

- [AddApplicationCloudWatchLoggingOption](#)
- [AddApplicationInput](#)
- [AddApplicationInputProcessingConfiguration](#)
- [AddApplicationOutput](#)
- [AddApplicationReferenceDataSource](#)
- [AddApplicationVpcConfiguration](#)
- [CreateApplication](#)
- [CreateApplicationSnapshot](#)
- [DeleteApplication](#)
- [DeleteApplicationCloudWatchLoggingOption](#)
- [DeleteApplicationInputProcessingConfiguration](#)
- [DeleteApplicationOutput](#)
- [DeleteApplicationReferenceDataSource](#)
- [DeleteApplicationSnapshot](#)

- [DeleteApplicationVpcConfiguration](#)
- [DescribeApplication](#)
- [DescribeApplicationSnapshot](#)
- [DiscoverInputSchema](#)
- [ListApplications](#)
- [ListApplicationSnapshots](#)
- [StartApplication](#)
- [StopApplication](#)
- [UpdateApplication](#)

AddApplicationCloudWatchLoggingOption

Il seguente codice di richiesta di esempio per l'[AddApplicationCloudWatchLoggingOption](#)azione aggiunge un'opzione di CloudWatch registrazione Amazon a un'applicazione Managed Service for Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "CloudWatchLoggingOption": {
    "LogStreamARN": "arn:aws:logs:us-east-1:123456789123:log-group:my-log-group:log-stream:My-LogStream"
  },
  "CurrentApplicationVersionId": 2
}
```

AddApplicationInput

Il seguente codice di richiesta di esempio per l'[AddApplicationInput](#)azione aggiunge un input dell'applicazione a un'applicazione Managed Service for Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 2,
  "Input": {
    "InputParallelism": {
      "Count": 2
    }
  }
}
```

```

    },
    "InputSchema": {
      "RecordColumns": [
        {
          "Mapping": "$.TICKER",
          "Name": "TICKER_SYMBOL",
          "SqlType": "VARCHAR(50)"
        },
        {
          "SqlType": "REAL",
          "Name": "PRICE",
          "Mapping": "$.PRICE"
        }
      ],
      "RecordEncoding": "UTF-8",
      "RecordFormat": {
        "MappingParameters": {
          "JSONMappingParameters": {
            "RecordRowPath": "$"
          }
        },
        "RecordFormatType": "JSON"
      }
    },
    "KinesisStreamsInput": {
      "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleInputStream"
    }
  }
}

```

AddApplicationInputProcessingConfiguration

Il seguente codice di richiesta di esempio per l'[AddApplicationInputProcessingConfiguration](#)azione aggiunge una configurazione di elaborazione dell'input dell'applicazione a un'applicazione Managed Service for Apache Flink:

```

{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 2,
  "InputId": "2.1",
  "InputProcessingConfiguration": {

```

```
"InputLambdaProcessor": {
  "ResourceARN": "arn:aws:lambda:us-
east-1:012345678901:function:MyLambdaFunction"
}
}
```

AddApplicationOutput

Il seguente codice di richiesta di esempio per l'[AddApplicationOutput](#)azione aggiunge un flusso di dati Kinesis come output dell'applicazione a un'applicazione Managed Service for Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 2,
  "Output": {
    "DestinationSchema": {
      "RecordFormatType": "JSON"
    },
    "KinesisStreamsOutput": {
      "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/
ExampleOutputStream"
    },
    "Name": "DESTINATION_SQL_STREAM"
  }
}
```

AddApplicationReferenceDataSource

Il seguente codice di richiesta di esempio per l'[AddApplicationReferenceDataSource](#)azione aggiunge un'origine di dati di riferimento dell'applicazione CSV a un'applicazione Managed Service for Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 5,
  "ReferenceDataSource": {
    "ReferenceSchema": {
      "RecordColumns": [
        {
```

```

        "Mapping": "$.TICKER",
        "Name": "TICKER",
        "SqlType": "VARCHAR(4)"
    },
    {
        "Mapping": "$.COMPANYNAME",
        "Name": "COMPANY_NAME",
        "SqlType": "VARCHAR(40)"
    },
],
"RecordEncoding": "UTF-8",
"RecordFormat": {
    "MappingParameters": {
        "CSVMappingParameters": {
            "RecordColumnDelimiter": " ",
            "RecordRowDelimiter": "\r\n"
        }
    },
    "RecordFormatType": "CSV"
},
},
"S3ReferenceDataSource": {
    "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
    "FileKey": "TickerReference.csv"
},
"TableName": "string"
}
}

```

AddApplicationVpcConfiguration

Il seguente codice di richiesta di esempio per l'azione [AddApplicationVpcConfiguration](#) aggiunge una configurazione VPC a un'applicazione esistente:

```

{
    "ApplicationName": "MyApplication",
    "CurrentApplicationVersionId": 9,
    "VpcConfiguration": {
        "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
        "SubnetIds": [ "subnet-0123456789abcdef0" ]
    }
}

```

CreateApplication

Il seguente codice di richiesta di esempio per l'[CreateApplication](#) azione crea un servizio gestito per l'applicazione Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "ApplicationDescription": "My-Application-Description",
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
  "CloudWatchLoggingOptions": [
    {
      "LogStreamARN": "arn:aws:logs:us-east-1:123456789123:log-group:my-log-group:log-stream:My-LogStream"
    }
  ],
  "ApplicationConfiguration": {
    "EnvironmentProperties": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap": {
            "aws.region": "us-east-1",
            "flink.stream.initpos": "LATEST"
          }
        },
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap": {
            "aws.region": "us-east-1"
          }
        }
      ]
    },
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
          "FileKey": "myflink.jar",
          "ObjectVersion": "AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "CodeContentType": "ZIPFILE"
    }
  }
}
```

```
    },
    "FlinkApplicationConfiguration": {
      "ParallelismConfiguration": {
        "ConfigurationType": "CUSTOM",
        "Parallelism": 2,
        "ParallelismPerKPU": 1,
        "AutoScalingEnabled": true
      }
    }
  }
}
```

CreateApplicationSnapshot

Il seguente codice di richiesta di esempio per l'[CreateApplicationSnapshot](#)azione crea un'istantanea dello stato dell'applicazione:

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

DeleteApplication

Il seguente codice di richiesta di esempio per l'[DeleteApplication](#)azione elimina un'applicazione Managed Service for Apache Flink:

```
{"ApplicationName": "MyApplication",
 "CreateTimestamp": 12345678912}
```

DeleteApplicationCloudWatchLoggingOption

Il seguente codice di richiesta di esempio per l'[DeleteApplicationCloudWatchLoggingOption](#)azione elimina un'opzione di CloudWatch registrazione Amazon da un'applicazione Managed Service for Apache Flink:

```
{
  "ApplicationName": "MyApplication",
```



```
"CloudWatchLoggingOptionId": "3.1"
"CurrentApplicationVersionId": 3
}
```

DeleteApplicationInputProcessingConfiguration

Il seguente codice di richiesta di esempio per l'[DeleteApplicationInputProcessingConfiguration](#)azione rimuove una configurazione di elaborazione dell'input da un'applicazione Managed Service for Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 4,
  "InputId": "2.1"
}
```

DeleteApplicationOutput

Il seguente codice di richiesta di esempio per l'[DeleteApplicationOutput](#)azione rimuove l'output di un'applicazione da un'applicazione Managed Service for Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 4,
  "OutputId": "4.1"
}
```

DeleteApplicationReferenceDataSource

Il seguente codice di richiesta di esempio per l'[DeleteApplicationReferenceDataSource](#)azione rimuove un'origine dati di riferimento dell'applicazione da un'applicazione Managed Service for Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 5,
  "ReferenceId": "5.1"
}
```

DeleteApplicationSnapshot

Il seguente codice di richiesta di esempio per l'[DeleteApplicationSnapshot](#)azione elimina un'istantanea dello stato dell'applicazione:

```
{
  "ApplicationName": "MyApplication",
  "SnapshotCreationTimestamp": 12345678912,
  "SnapshotName": "MySnapshot"
}
```

DeleteApplicationVpcConfiguration

Il seguente codice di richiesta di esempio per l'azione [DeleteApplicationVpcConfiguration](#) rimuove una configurazione VPC esistente da un'applicazione:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 9,
  "VpcConfigurationId": "1.1"
}
```

DescribeApplication

Il seguente codice di richiesta di esempio per l'[DescribeApplication](#)azione restituisce i dettagli su un'applicazione Managed Service for Apache Flink:

```
{"ApplicationName": "MyApplication"}
```

DescribeApplicationSnapshot

Il seguente codice di richiesta di esempio per l'[DescribeApplicationSnapshot](#)azione restituisce dettagli su un'istantanea dello stato dell'applicazione:

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

```
}
```

DiscoverInputSchema

Il seguente esempio di codice di richiesta per l'[DiscoverInputSchema](#)azione genera uno schema da una sorgente di streaming:

```
{
  "InputProcessingConfiguration": {
    "InputLambdaProcessor": {
      "ResourceARN": "arn:aws:lambda:us-east-1:012345678901:function:MyLambdaFunction"
    }
  },
  "InputStartingPositionConfiguration": {
    "InputStartingPosition": "NOW"
  },
  "ResourceARN": "arn:aws:kinesis:us-east-1:012345678901:stream/ExampleInputStream",
  "S3Configuration": {
    "BucketARN": "string",
    "FileKey": "string"
  },
  "ServiceExecutionRole": "string"
}
```

Il seguente codice di richiesta di esempio per l'[DiscoverInputSchema](#)azione genera uno schema da una fonte di riferimento:

```
{
  "S3Configuration": {
    "BucketARN": "arn:aws:s3:::amzn-s3-demo-bucket",
    "FileKey": "TickerReference.csv"
  },
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole"
}
```

ListApplications

Il seguente codice di richiesta di esempio per l'[ListApplications](#)azione restituisce un elenco di applicazioni Managed Service for Apache Flink presenti nell'account:

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

ListApplicationSnapshots

Il seguente codice di richiesta di esempio per l'[ListApplicationSnapshots](#) azione restituisce un elenco di istantanee dello stato dell'applicazione:

```
{"ApplicationName": "MyApplication",
  "Limit": 50,
  "NextToken": "aBcDeFgHiJkLmNoPqRsTuVwXyZ0123"
}
```

StartApplication

Il seguente codice di richiesta di esempio per l'[StartApplication](#) azione avvia un'applicazione Managed Service for Apache Flink e carica lo stato dell'applicazione dall'ultima istantanea (se presente):

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

StopApplication

Il seguente codice di richiesta di esempio per l'azione [API_StopApplication interrompe un'applicazione](#) Managed Service for Apache Flink:

```
{"ApplicationName": "MyApplication"}
```

UpdateApplication

Il seguente codice di richiesta di esempio per l'[UpdateApplication](#) azione aggiorna un'applicazione Managed Service for Apache Flink per modificare la posizione del codice dell'applicazione:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationCodeConfigurationUpdate": {
      "CodeContentTypeUpdate": "ZIPFILE",
      "CodeContentUpdate": {
        "S3ContentLocationUpdate": {
          "BucketARNUpdate": "arn:aws:s3:::amzn-s3-demo-bucket",
          "FileKeyUpdate": "my_new_code.zip",
          "ObjectVersionUpdate": "2"
        }
      }
    }
  }
}
```

Documentazione di riferimento delle API del servizio gestito per Apache Flink

Per informazioni sui servizi gestiti per APIs Apache Flink, consulta Managed Service for Apache Flink [API Reference](#).

Questo contenuto è stato spostato nelle versioni Release. Per informazioni, consulta [Versioni di rilascio](#).